

# **ROBOTIKA PRO STŘEDNÍ ŠKOLY: PROGRAMUJEME ARDUINO**

Milan Novák  
Jiří Pech



EVROPSKÁ UNIE  
Evropské strukturální a investiční fondy  
Operační program Výzkum, vývoj a vzdělávání





Jihočeská univerzita  
v Českých Budějovicích  
University of South Bohemia  
in České Budějovice

# ROBOTIKA PRO STŘEDNÍ ŠKOLY: PROGRAMUJEME ARDUINO

PhDr. Milan Novák, Ph.D.; Mgr. Jiří Pech, Ph.D.

Recenzent:  
Ing. Michal Šerý, Ph.D.

Vydavatel:  
Jihočeská univerzita v Českých Budějovicích, Pedagogická fakulta

Obálka:  
Mgr. Pavel Pfauser

Rok vydání: 2020

ISBN 978-80-7394-786-6

9 788073 947866 >



Podléhá licenci Creative Commons  
Uveďte původ - Zachovajte licenci 4.0



# JAK PRACOVAT S UČEBNICÍ

KRÁTKÉ SEZNÁMENÍ, JAK PRACOVAT S UČEBNÍCÍ.

Učebnice se skládá z jednotlivých kapitol, které obsahují konkrétní probírané téma. Materiály pro každé téma jsou rozděleny do tří základních částí a doplněny o další podpůrné materiály.

Všechny materiály jsou k dispozici na serveru GitHub:

[HTTPS://GITHUB.COM/NOWIS75/PRIM](https://github.com/nowis75/prim)

Na tomto serveru je každá **složka lekce** rozdělena na dílčí **pod složky**, které mají následující strukturu:

- 01\_Pro\_ucitele
- 02\_Pro\_zaky
- 03\_Samostudium
- 04\_Zdrojove\_kody\_prikladu
- 05\_Zaverecny\_projekt
- 06\_Dalsi\_materialy

## 1. PRO UČITELE

Tato složka obsahuje soubory určené učitelům, které představují doporučený průchod hodinou. Dále jsou zde soubory ukázkově vyřešených samostatných úloh, které mají žáci řešit v rámci pracovních listů.

Dále jsou zde prezentace k hodině v MS PowerPoint (složka *PowerPoint\_prezentace*), které kopírují průvodce hodinou a celá lekce v jednom souboru se vsemi částmi (složka *Vse\_v\_jednom*).

## **2. PRO ŽÁKY**

Složka obsahuje pracovní listy, které kopírují strukturu PODROBNÉHO PRŮVODCE HODINOU vyučujícího. Soubory s pracovními listy jsou samostatně umístěny v příslušné složce, aby bylo snadné je vytisknout. Pracovní listy lze rozeslat i elektronicky.

Správná řešení samostatných úkolů jsou ve složce **PRO UČITELE**.

## **3. SAMOSTUDIUM**

Složka obsahuje materiál, který může být studijním doplňkem. Jsou zde podrobné informace k probíranému tématu. Lze jej využít jako studijní materiál jak pro studenty, tak i pro vyučující, kteří se s probíranou látkou seznamují.

Jsou zde také podrobně okomentovány schémata zapojení a programové kódy všech příkladů použitych v pracovních listech.

Součástí je i námět na projekt, který využívá naučených znalostí z hodiny.

## **4. ZDROJOVÉ KÓDY**

Složka obsahuje soubory zdrojových kódů, které lze otevřít v prostředí Arduino. Jsou zde uvedeny kódy všech vzorových příkladů i samostatných úkolů.

Pokud je to nutné, tak lze ve složce nalézt potřebné knihovny pro prostředí Arduino.

## **5. ZÁVĚREČNÝ PROJEKT**

Složka obsahuje soubor, ve kterém je uveden postup konstrukce samostatného projektu dále podsložku s tiskovými šablonami pro případnou konstrukci. Šablony jsou uvedeny ve zdrojových souborech, nebo v PDF, které lze jednoduše vytisknout

## **6. DALŠÍ MATERIÁLY**

Ve složce se nachází zejména soubory schémat zapojení všech obvodů uvedených v učebnici, vztahující se k dané lekci.

# VÍTEJTE VE SVĚTE ARDUINO

ARDUINO USNADŇUJE PRÁCI S MALÝMI POČÍTAČI, NAZÝVAJÍCÍMI SE MIKROKOTROLERY, POMOCÍ KTERÝCH LZE VYTVÁŘET ZAJÍMAVÉ PROJEKTY, KTERÉ MOHOU BÝT ZÁKLADEM ROBOTICKÝCH PLATFOREM.

Každý z nás, každý den, používá různé technologie. Většina z nás nechává programování inženýrům, protože si myslíme, že kódování a elektronika jsou komplikované a obtížné. Ve skutečnosti to může být zábavné a vzrušující. Díky tomu se designéři, umělci, fandové a studenti všech věkových kategorií učí vytvářet věci, které se rozsvítí, pohybují a reagují na lidi, zvířata, rostliny a zbytek světa.

V průběhu let bylo Arduino používáno jako „mozek“ v tisících kreativních projektech. Kolem této platformy se shromáždila celosvětová komunita tvůrců, která vytváří a využívá otevřený zdrojový kód. Platforma Arduino je snadno použitelná pro začátečníky, ale je dostatečně flexibilní pro pokročilé uživatele. Software lze spustit na počítačích Mac, Windows a Linux. Učitelé a studenti jej používají k vytváření levných vědeckých nástrojů, k prokázání principů chemie a fyziky nebo k začátku programování a robotiky.

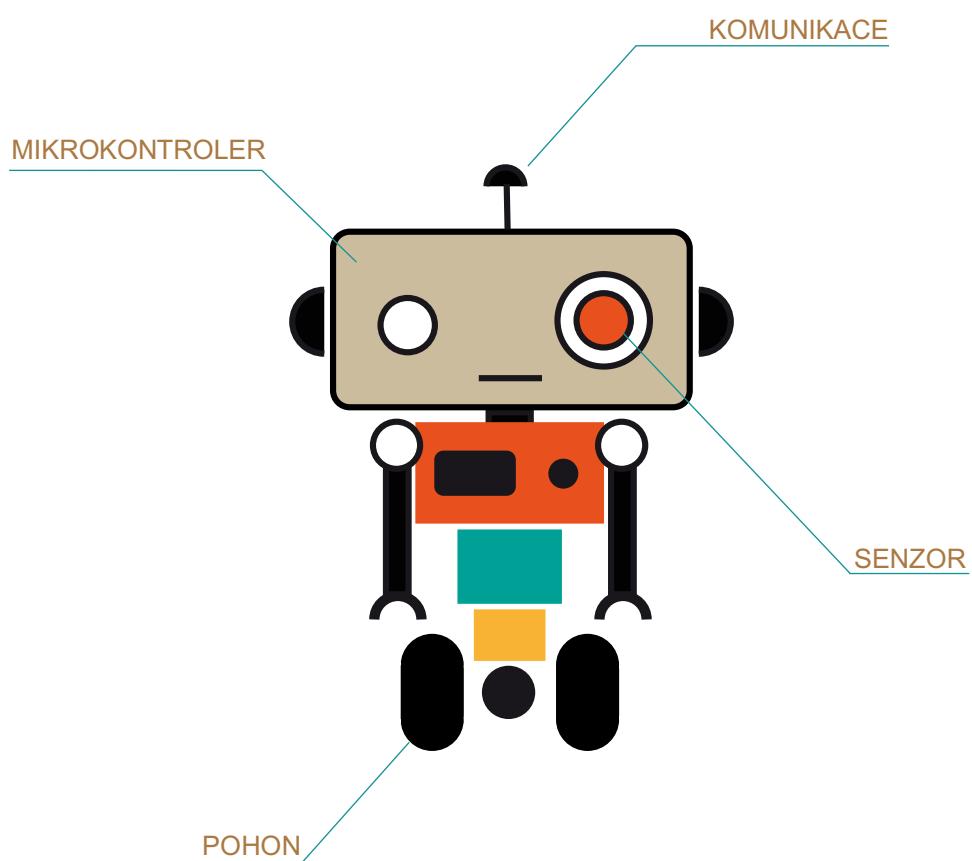
Aniž bychom to možná tušíme, dennodenně se setkáváte s desítkami systémů, které jsou zabudovány v různých časovačích, hračkách, dálkových ovladačích, mikrovlnných troubách, a dokonce i zubních kartáčcích. Provádí pouze jeden úkol, ale ten dělají spolehlivě. Jsou naprogramovány tak, aby snímaly nejrůznější veličiny a ovlivňovali různé akční členy.

Kombinace těchto malých a jednodušších systémů jsou schopny vytvořit jeden celek, kterým může být robot. O něm lze hovořit jako o zařízení, které je schopné reagovat na podněty z okolí a zpětně na ně působit. Může být zcela autonomní nebo počítačem řízený. Roboty se skládají ze subsystémů, které jsou akční (aktuátory tj. pohony), řídící (mikrokontrolery), senzorické, komunikační a napájecí.

Senzory naslouchají svému okolí. Snímají teplotu, vlhkost, nebo hluk a přeměňují tuto energii na elektrický signál. Ale mohou to být i tlačítka, která snímají dotyk a stisk pomocí prstů. Senzorů je mnoho druhů a jsou všude kolem nás.

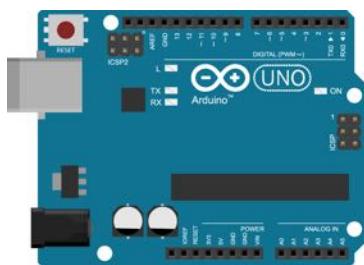
Aktuátory provádí různé akce ve fyzickém světě. Přeměňují elektrickou energii zpět na světelnou, tepelnou a pohybovou.

Mikrokontrolery naslouchají senzorům a mluví s aktuátory. Rozhodují se, co mají dělat na základě programu, který pro ně napišete. Mikrokontrolery a elektronika, kterou k nim připojíte jsou kostrou projektů.

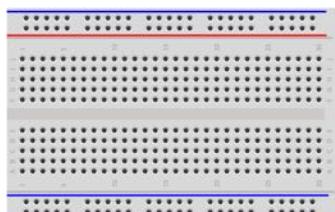


# OBSAH SADY ARDUINO

DOPORUČENÁ SADA - ADEEPT ULTIMATE STARTER KIT FOR ARDUINO UNO R3  
OBSAHUJE SOUČÁSTKY, KTERÉ JSOU POUŽITY V PROJEKTECH TÉTO UČEBNICE



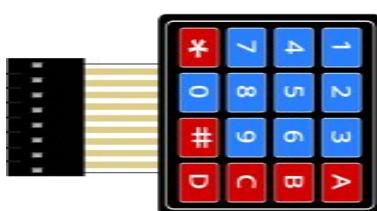
**Arduino Uno** – základní vývojová deska mikrokontroleru která tvoří srdce všech projektů. Jedná se o jednoduchý počítač, který nemá prozatím rozhraní pro interakci s ním. Budete vytvářet elektronická zapojení, rozhraní pro interakci a volání mikrokontroleru pro komunikaci s ostatními komponentami.



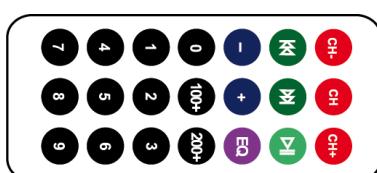
**Kontaktní pole** – deska, která umožňuje sestavovat elektronické obvody bez nutnosti používat pájku. Součástky se zapojují do řádků a sloupců desky.



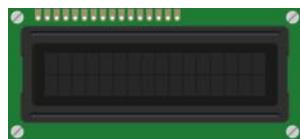
**Držák baterií AA** – tímto držákem klasických tužkových baterií můžete napájet desku Arduino nebo zařízení připojené ke kontaktnímu poli.



**Klávesnice 4x4** – zařízení pro zasílání uživatelských vstupů do mikrokontroleru. Jedná se o maticovou klávesnici, která využívá schéma kódování, což umožňuje menší množství výstupních pinů. To je výhodnější pro menší kabeláž.



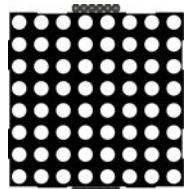
**IR dálkový ovladač** – bezdrátové vstupní zařízení, které využívá infračerveného záření. Pro příjem odesílaného signálu se používá infračervený přijímač.



**LCD display** – jeden z typů displeje pro zobrazení alfanumerických znaků. Existuje celá řada LCD displejů . V sadě je displej se 2 řádky po 16 znacích.



**LED Bar Graph** – sloupcový LED displej, který se skládá ze samostatných diod, umístěných v jednom pouzdře. Displej se používá pro vizualizaci analogových hodnot.



**Maticový LED displej** – je zobrazovací zařízení, které slouží k zobrazování statických i dynamických informací. Jedná se o maticové zařízení 8x8.



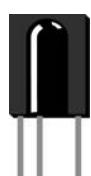
**1 bitový sedmissegmentový displej** – je display pro zobrazení číslic a speciálních znaků. Lze jej využít pro zobrazení teploty



**4 bitový sedmissegmentový displej** – má stejnou funkčnost jako 1 bitový, ale lze jej využít pro zobrazení více hodnot, takže se hodí na zobrazení teploty, rychlosti, hodnot senzorů apod.



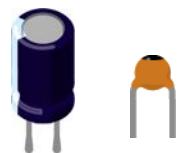
**Fotorezistor** – je součástka, která mění svůj odpor podle toho, jak intenzivní záření na ní dopadá.



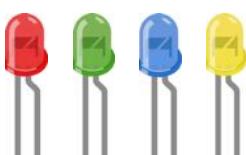
**Infračervený přijímač** – umožňuje detektovat signály z IR dálkových ovladačů a při jeho připojení k Arduinu můžeme tyto signály převést na proměnné a dále využít.



**Křížový ovladač, joystick** – je herní modul, který umožňuje snímat pohyb ve dvou směrech. Samotný joystick je tvořen dvěma potenciometry, jejichž hodnota určuje směr.



**Kondenzátory** - jsou schopny podle kapacity udržet elektrický náboj. Lze je použít k zachycení napěťových špiček v napájecím napětí. Kondenzátory jsou keramické nebo elektrolytické.



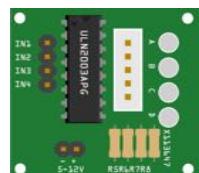
**LED diody** – jsou polovodičové součástky, schopné vyzařovat světlo. Používají se převážně k signalizaci. V současné době se využívají i k osvětlení.



**Usměrňovací diody** - je elektrotechnická součástka, jejímž úkolem je v elektrickém obvodu propouštět elektrický proud jen jedním směrem.



**Ovladač stejnosměrného motoru** – je určen k jednoduchému ovládání motorů mikrokontrolérem nebo logickými obvody.



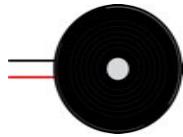
**Řídící deska pro krokový motor** – se používá k řízení krokového motoru. Poskytuje vyšší proudový výstup.



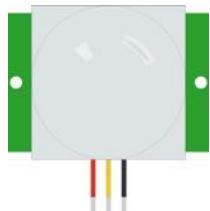
**Krokový motor** – elektromechanické zařízení, přeměňující elektrické impulzy na mechanický pohyb. Mezi výhody tohoto motoru patří snadné otočení o konkrétní úhel, velká síla, okamžitá odezva při zastavení nebo spouštění motoru.



**Stejnosměrný motor** – je určený k pohybu robotických částí. Nejčastěji se používá v mobilních robotech nebo v zařízeních, kde je vyžadován plynulý rotační pohyb.



**Pasivní bzučák** – součástka, která vytváří zvuky na základě elektrických pulzů.



**Pohybové čidlo** – snímá své okolí pod úhlem přibližně 120 stupňů. Pokud v tomto snímaném prostoru proběhne větší změna teploty, tzn. například projde člověk či zvíře, je tato změna zaznamenána díky změně výstupního napětí.



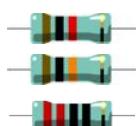
**Posuvný registr** – pomůže rozšířit nedostatek pinů desky Arduino.



**Potenciometr** – součástka, která při otáčení osou mění poměr odporu na její dráze. Lze je použít například při změně jasu displeje nebo LED diody.



**Relé** – je elektricky ovládaný spínač. Umožňuje ovládat zařízení s velkými proudovými nároky, a to bez obav o Arduino, protože vytváří nový napájecí okruh.



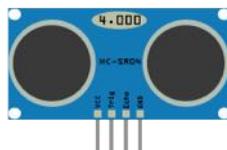
**Rezistor** – je pasivní elektrotechnická součástka, která je zařazována do obvodu z důvodu snížení elektrického proudu nebo získání úbytku napětí.



**Tranzistor** – je polovodičová součástka, kterou lze jednoduše používat jako zesilovač, spínač, stabilizátor aj.



**Servomotor** – slouží pro nastavení polohy ovládaného mechanizmu a následné držení v této poloze. Stejnosměrné servo motory se mohou využít pro ovládání robotické paže nebo pro nastavení kormidla u leteckých modelů.



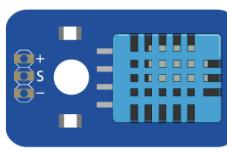
**Sonar** – je senzor k měření vzdálenosti. Měření je založeno na principu měření doby mezi vysláním akustického signálu a přjetím odraženého signálu.



**Tlačítko** – je jednoduchá mechanická součástka, která slouží ke spojení a rozpojení obvodu.



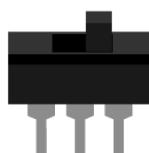
**Termistor** – se používá pro měření teploty. Jedná se o rezistor, který v závislosti na teplotě svůj odpor mění podle přesně definované charakteristiky.



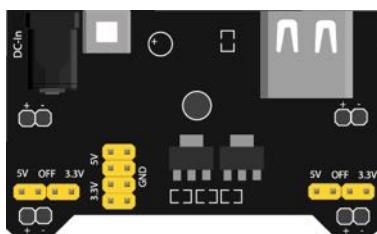
**Teplotní a vlhkostní čidlo** – obsahuje vše potřebné. Stačí ho připojit rovnou k desce Arduino a načítat hodnoty z okolí, které se obnovují přibližně každé 2 sekundy.



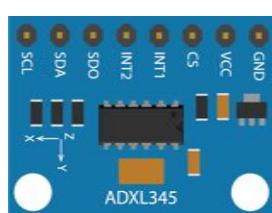
**Vibrační senzor** – umožňuje detektovat vibrace ve svém okolí. Vlastní pohyb v rámci snímače vede ke změně elektrického napětí, která se dá změřit mikrokontrolérem.



**Třípolohový přepínač** – lze využít při přepínání různých módů a stavů.

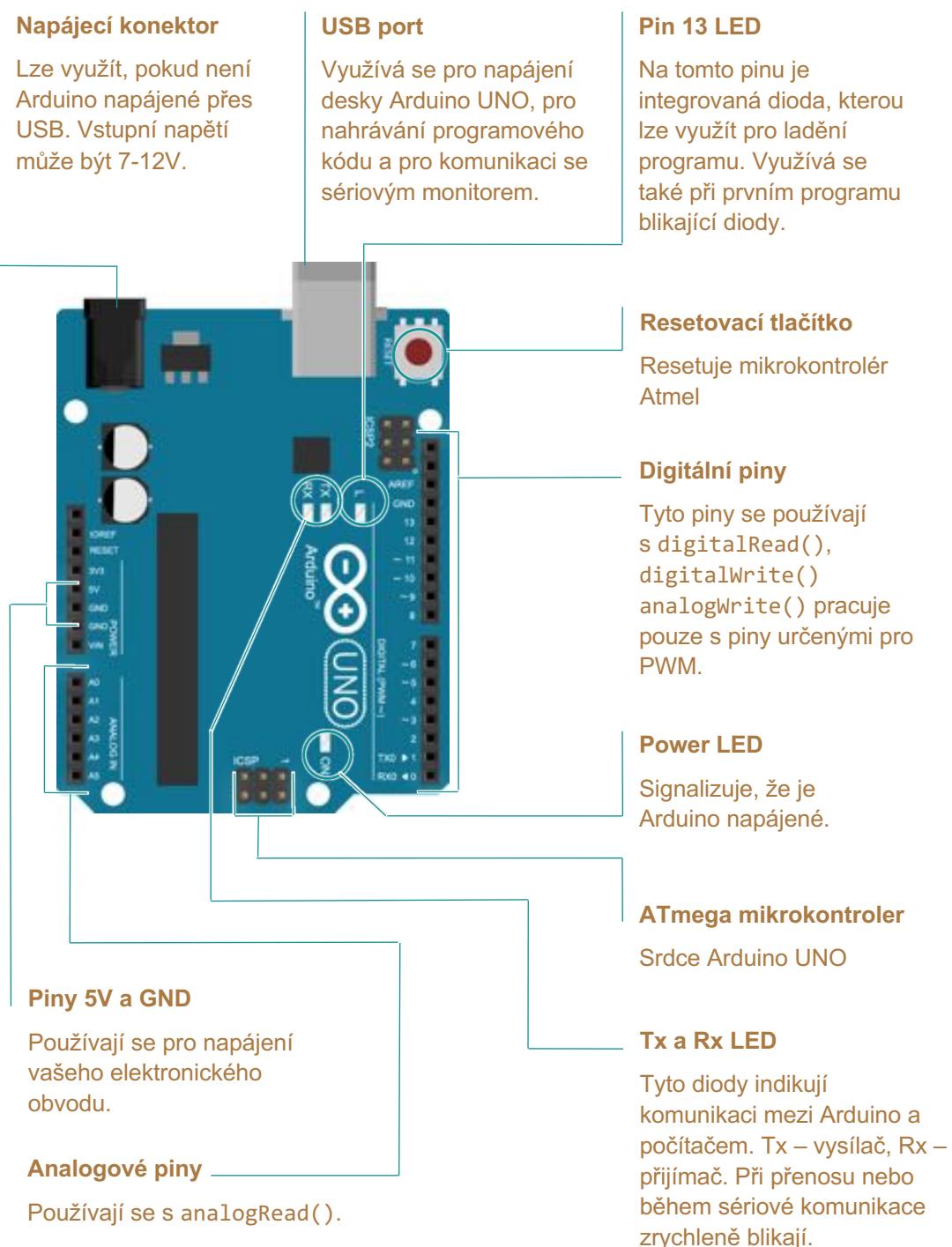


**Napájecí modul** – se připevňuje do kontaktního pole a je určený pro napájení dalších komponent na tomto poli. Vstupní napájení může být mezi 6.5-12V. Výstupní napětí je 3.3V nebo 5V nezávisle na velikosti vstupního napětí.



**Akcelerometr** – modul, který umožňuje snímat pohyb v osách X, Y a Z. Obvod ADXL345 zajišťuje veškeré měření a nám stačí na analogovém výstupu pro každou osu měřit měnící se napětí a přepočítat jej na natočení ve stupních.

# DESKA ARDUINO



# NASTAVENÍ ARDUINO

PŘED SAMOTNÝM POUŽÍVÁNÍ DESKY ARDUINO UNO JE ZAPOTŘEBÍ STÁHNOUT PROGRAM PRO KOMUNIKACI S DESKOU.

Arduino **IDE** umožňuje psát programy a nahrávat je do desky. Existují dvě možnosti, jak program pro komunikaci s vývojovou deskou získat.

- ① Pokud máte spolehlivé připojení k internetu, tak lze využít online **IDE editor**. Umožní vám vkládat programový kód do cloudového uložiště a tento kód bude přístupný z jakéhokoliv počítače. Vždy budete mít dostupné nejnovější vývojové rozhraní, bez nutnosti instalace aktualizací knihoven.

**[HTTPS://CREATE.ARDUINO.CC/EDITOR](https://create.arduino.cc/editor)**

- ② Pokud radši pracujete v offline režimu, stáhněte si nejnovější **IDE editor** do počítače.

**[HTTPS://WWW.ARDUINO.CC/DOWNLOAD](https://www.arduino.cc/download)**

Dále jsou uvedeny postupy instalace pro operační systémy: Windows, Mac OS X.

## INSTALACE PRO WINDOWS

Stáhněte si nejnovější verzi IDE editoru. Můžete si vybrat mezi verzí pro instalaci (.exe), nebo balíček Zip. Doporučuje se použít první verzi, která přímo nainstaluje vše, co je potřeba k používání softwaru Arduino (IDE), včetně ovladačů. Pomocí balíčku Zip se musí nainstalovat ovladače ručně.



Soubor Zip je užitečný v případě, že chcete mít přenosnou instalaci. To je vhodné například ve školách, kde studenti nemají administrační práva pro instalaci softwaru.

- ① Po stažení exe souboru pro instalaci na tento soubor dvakrát klikněte, aby se spustil instalační průvodce. Pokud se zobrazí výstražné okno operačního systému, s dotazem, zda se má softwaru důvěřovat, klikněte na tlačítko, že ano. Potvrďte, že souhlasíte s licenčními podmínkami a klikněte na tlačítko Next pro výběr instalačního adresáře. Následně klikněte na Install.
- ② Po proběhnutí instalace připojte USB kabel k desce Arduino a k vašemu počítači. Na desce se automaticky rozsvítí dioda signalizující napájení z USB.
- ③ Windows po připojení desky provede instalaci ovladačů. Jestliže se vás systém zeptá, zda má nainstalovat ovladače k desce Arduino, zvolte možnost Automaticky nebo nalézt v tomto počítači.
- ④ Jestliže instalace neproběhne automaticky, klikněte na Menu Start a otevřete Ovládací panely. Potom přejděte do správce zařízení.
- ⑤ Najděte zařízení Arduino v kategorii „Ostatní zařízení“ nebo „Neznámé zařízení“ a kliknutím pravého tlačítka myši vyberte „Aktualizovat ovladač“ nebo „Aktualizujte ovladače programu“.
- ⑥ Klikněte na „Procházet“ a vyberte adresář „Ovladače“ v adresáři Arduino. Klikněte na tlačítko „OK“ a „Další“. Jestliže se otevře dialogové okno, klikněte na „Pokračovat“, systém začne instalovat ovladače.
- ⑦ Ve „Správci zařízení“, v kategorii „Porty (COM & LPT)“ by se měl objevit port podobný názvu „Arduino UNO (COM4)“.

**Výborně, máte nainstalované rozhraní pro programování v Arduino.**

## INSTALACE MAC OS X

- ① Jestliž používáte 10.8 nebo novější verzi systému MAC OS X, přejděte do Předvoleb systému a otevřete panel „Zabezpečení a soukromí“. V nabídce „Obecné“ v části „Povolit aplikace stažené“ vyberte položku „Z App Storu a od známých vývojářů“.
- ② Pokud je Arduino IDE stažené, dvakrát klikněte na soubor .zip pro rozbalení aplikace.
- ③ Zkopírujte aplikaci Arduino do adresáře „Aplikace“ nebo do jiného, který je k tomu určený.
- ④ Připojte desku Arduino pomocí USB kabelu k počítači. Deska bude automaticky napájena z USB spojení a rozsvítí se zelená LED dioda.
- ⑤ V systému MAC OS X není nutné instalovat ovladače desky.
- ⑥ V závislosti na verzi systému OS X se při spuštění programu Arduino IDE může objevit dialogové okno upozorňující na to, že spouštěná aplikace není od ověřeného vývojáře. Klikněte na „Otevřít“. Aplikace se spustí.

**Výborně, máte nainstalované rozhraní pro programování v Arduino.**

## KOMUNIKACE S ARDUINO

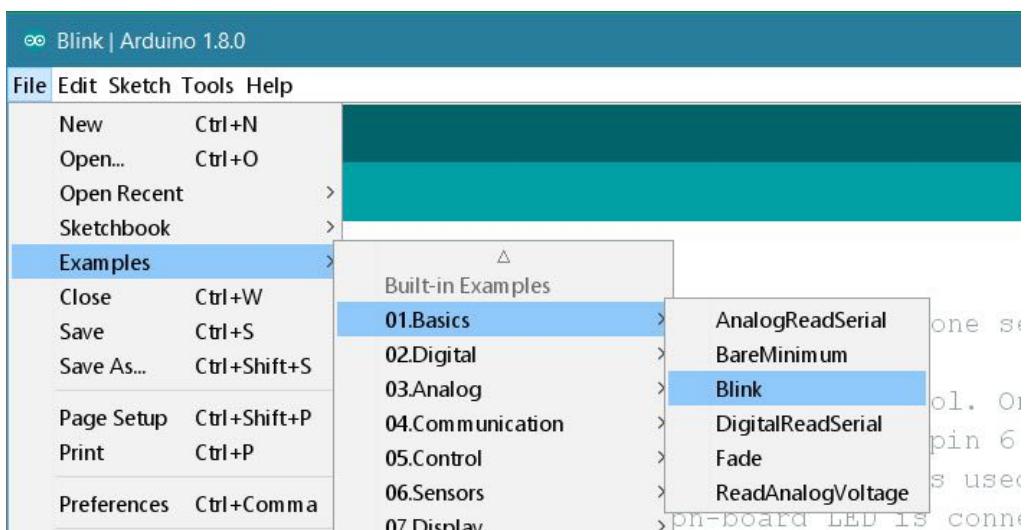
Nyní, když máte nainstalované rozhraní Arduino IDE a připojenou desku Arduino UNO k vašemu počítači, můžete nahrát program.

① Poklepáním na ikonu aplikace Arduino dojde k otevření. Jestliže se IDE nahraje s jiným jazykem, můžete si jej změnit ve vlastnostech aplikace. Po změně jazyka se musí aplikace opětovně spustit.

② Nyní provedte zkoušku komunikace desky s počítačem a testovací nahrání programu.

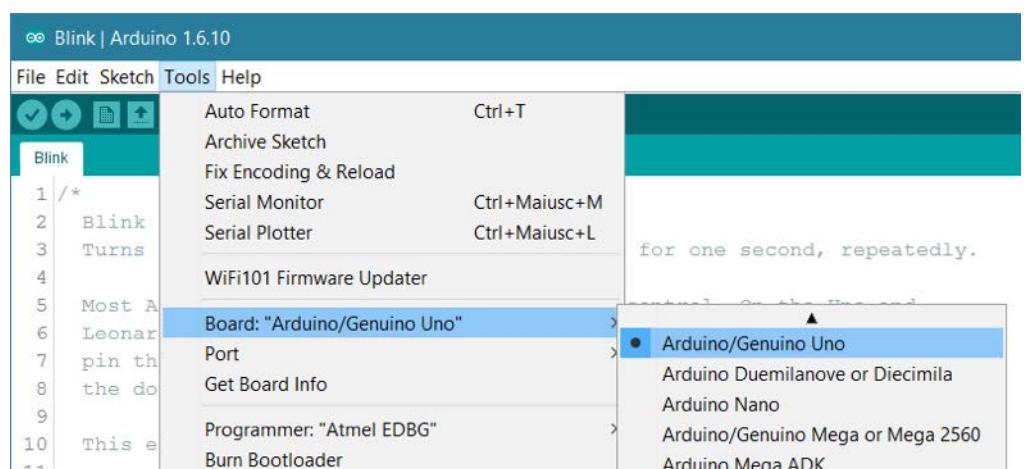
Využijete k tomu připraveného programu. Umístění příkladu je zde:

**FILE>EXAMPLES>01.BASICS>BLINK**



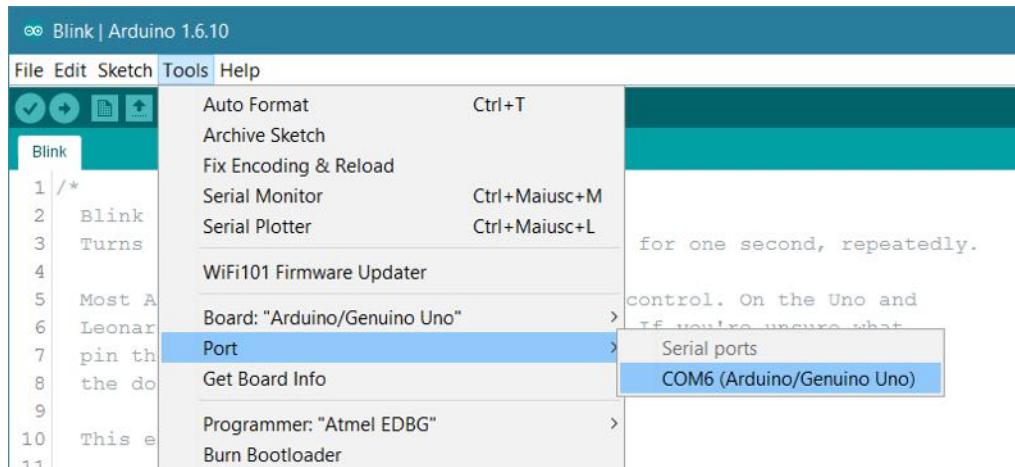
③ Z menu aplikace vyberte odpovídají desku, která je připojena k počítači:

**TOOLS>BOARD**



- ④ Vyberte správný sériový port, ke kterému je deska Arduino připojena:

**TOOLS>SERIAL PORT**



V operačním systému Windows jsou názvy portů označeny písmeny COM a číslem. Z uvedeného menu stačí vybrat odpovídající port. V operačním systému MAC je označení portu v podobě: /dev/tty.usbmodem. Ze seznamu vyberte ten

- ⑤ Nyní nahrajte program Blink do své desky Arduino. To provedete kliknutím na tlačítko **UPLOAD**, v levém horním rohu programu.
- ⑥ Nyní byste měli vidět v dolní části IDE indikaci postupu nahrávání a na desce by měly blikat diody TX a RX. Jestliže je nahrávání v pořádku ukončeno, zobrazí se zpráva **DONE UPLOADING**.
- ⑦ Během několika sekund by se měla rozblíkat LED dioda na desce Arduino. Tato dioda je označena písmenem **L**. Pokud tomu tak opravdu je a dioda bliká, vaše deska a připojení je plně funkční a můžete se plně věnovat projektům.



# PROBLÉMY A JEJICH ŘEŠENÍ

## PROČ NEJDE NAHRÁT PROGRAM DO DESKY ARDUINO?

Tento problém může být způsoben celou řadou okolností. Může se jednat o nenainstalování ovladačů, špatný výběr sériového portu v Arduino IDE, přístup k sériovému portu, nezapojený USB kabel apod. Zde je uvedeno několik postupů, jak problémy vyřešit.

### ARDUINO SOFTWARE

- Ujistěte se, že máte vybranou správnou desku. Výběr desky najeznete v **Nástroje>Deska**. Název desky, kterou používáte, najeznete přímo na desce.
- Zkontrolujte zda je vybrán správný sériový port: **Nástroje>Sériový port**. Pokud není vidět správný sériový port, zkuste restartovat počítač a pak znova spustit Arduino IDE. V operačním systému Windows se jedná o COM port, který zkontrolujte ve Správci zařízení v kategorii Porty. V MAC počítači by měl port vypadat podobně jako /dev/tty.usbmodem621.

### OVLADAČE

Ovladače poskytují cestu, jak má váš počítač komunikovat s hardwarem (deskou Arduino). V případě Arduino ovladače poskytují virtuální sériový port.

Nejjednodušší způsob, jak zkontrolovat, zda jsou ovladače správně nainstalovány, je otevření nabídky **Nástroje> Sériový port** v softwaru Arduino s připojenou deskou připojenou k počítači. Po připojení desky k počítači by se měly objevit nové položky oproti tomu, když je deska Arduino odpojena od počítače.

- V systému Windows (zejména v 64bitové verzi) možná budete muset jít do Správce zařízení a aktualizovat ovladače pro Uno nebo Mega 2560. Stačí kliknout pravým tlačítkem na zařízení (deska by měla být připojena k počítači) a nasměrovat průvodce aktualizací ovladače k příslušnému souboru INF. Ten se nachází v adresáři ovladače v adresáři softwaru Arduino (nikoliv v podadresáři FTDI USB Drivers).

### PŘÍSTUP K SÉRIOVÉMU PORTU

- V systému Windows, může dojít k tomu, že se software bude spouštět velmi pomalu. V tomto případě zkuste vypnout ve Správci zařízení sériové porty Bluetooth, nebo jiné

síťové porty COM. Software Arduino při svém spuštění prohledává všechny sériové (COM) porty v počítači a tyto síťové porty mohou někdy způsobit velké zpoždění nebo selhání.

- Zkontrolujte, zda nejsou spuštěny žádné programy, které využívají aplikace pro synchronizaci mobilních zařízení, Bluetooth ovladače USB, virtuální démonové nástroje atd.
- Ujistěte se, že software brány firewall, ne blokuje přístup k sériovému portu.

### FYZICKÉ PŘIPOJENÍ

- Nejprve se ujistěte, že je vaše deska připojena k počítači a svítí na ni zelená kontrolka.
- Arduino Uno a Mega 2560 mohou mít problémy s připojením k počítači Mac přes rozbočovač USB. Pokud se ve vašem menu **Nástroje> Sériový port** nic neobjeví, zkuste připojit desku přímo k počítači a restartovat Arduino IDE.
- Při nahrávání kódu odpojte digitální piny 0 a 1, protože jsou sdíleny při sériové komunikaci s počítačem (mohou být připojeny a použity po nahrání kódu).
- Zkuste nahrát kód do desky, ke které není nic připojené (samozřejmě kromě kabelu USB).
- Ujistěte se, že se deska nedotýká ničeho kovového nebo vodivého.
- Zkuste jiný kabel USB, někdy může být nefunkční.

### ZAVADĚČ

- Ujistěte se, že na vaší desce Arduino nedošlo ke zničení bootloaderu - zavaděče. Chcete-li to zkontrolovat, resetujte desku. Vestavěná LED dioda L (která je připojena k pinu 13) by měla blikat. Pokud tomu tak není, na vaší desce nemusí být bootloader.

### PROČ SOFTWARE ARDUINO NEFUNGUJE PO AKTUALIZACI JAVA NA POČÍTAČI MAC?

- Nejnovější aktualizace Java od společnosti Apple se pokouší použít 64bitovou verzi nativních knihoven, ale aplikace Arduino je dodávána s 32bitovou verzí knihovny RXTX. Pokud spustíte Arduino, bude vrácena chyba, jako tato:

```
Uncaught           exception       in      main      method:  
java.lang.UnsatisfiedLinkError:      /Applications/arduino-  
0016/Arduino
```

```
16.app/Contents/Resources/Java/librxtxSerial.jnilib:      no
suitable image found.  Did find: /Applications/arduino-
0016/Arduino
16.app/Contents/Resources/Java/librxtxSerial.jnilib:      no
matching architecture in universal wrapper
```

Chcete-li tento problém odstranit, klikněte ve Finderu na aplikaci Arduino (např. Arduino.app) a z otevřeného menu vyberte Informace. Na informačním panelu zaškrtněte políčko **Otevřít v režimu 32 bitů**. Potom byste měli být schopni spustit Arduino normálně.

### JAKÝ TYP NAPÁJECÍHO ZDROJE MÁM POUŽÍVAT U MÉ DESKY ARDUINO?

- Deska Arduino může běžně pracovat s napájením, které je k dispozici na USB portu počítače, ke kterému je připojen, v závislosti na počtu a druhu doplňkových modulů použitých s deskou Arduino a na jmenovitém proudu USB počítače (liší se podle výrobce a modelu počítače). Pokud zjistíte, že je zapotřebí k vaší desce Arduino nutné dodatečné napájení, nebo pokud potřebujete ovládat desku Arduino odpojenou od USB portu, musí být použitý zdroj, který poskytuje 7 až 12 V. Lze využít napájecí adaptéry běžně dostupné v maloobchodních prodejnách, ale ujistěte se, že mají správný konektor pro zapojení do elektrické zásuvky na desce Arduino.

### PROČ SOFTWARE ARDUINO PŘI POKUSU O NAHRÁNÍ PROGRAMU ZAMRZNE? (V SYSTÉMU WINDOWS)

- To může být způsobeno konfliktem s procesem Logitech "LVPrcSrv.exe". Otevřete **Správce úloh** a zjistěte, zda je tento program spuštěn. Pokud ano, před dalším pokusem o nahrání programu jej zrušte.

### CO UDĚLAT, POKUD SE PŘI SPUŠTĚNÍ ARDUINO.EXE V SYSTÉMU WINDOWS VYSKYTNÉ CHYBA?

- Pokud se při poklepání na spustitelný soubor arduino.exe v systému Windows zobrazí chybová zpráva, například:

**Arduino has encountered a problem and needs to close.**

budete muset spustit Arduino pomocí souboru run.bat. Buďte trpěliví, může chvíli trvat, než se prostředí Arduino otevře.

### ZOBRAZENÍ CHYBY "NELZE NAJÍT HLAVNÍ TŘÍDU."

- Pokud se při spuštění programu Arduino vyskytne chyba:

```
Java Virtual Machine Launcher: Could not find the main class. Program will exit.
```

ujistěte se, že jste správně extrahovali obsah souboru .zip Arduino, zejména pak, že adresář **lib** je přímo v adresáři Arduino a obsahuje soubor **pde.jar**.

### PROČ SE SOFTWARE ARDUINO A NABÍDKA NÁSTROJŮ DLOUHO OTVÍRAJÍ (V SYSTÉMU WINDOWS)?

- Pokud spuštění softwaru Arduino trvá dlouhou dobu, nebo se zdá, že zamrzl při pokusu o otevření nabídky Nástroje, dochází ke konfliktu s jiným zařízením ve vašem systému. Software Arduino se při spuštění a při otevření nabídky Nástroje pokusí získat seznam všech COM portů v počítači. Je možné, že port COM vytvořený jedním ze zařízení v počítači zpomaluje tento proces. Podívejte se do Správce zařízení. Zkuste zakázat zařízení, která využívají porty COM (například zařízení Bluetooth).

### PROČ SE POUŽÍVANÁ DESKA NEZOBRAZUJE V ARDUINO IDE A NENÍ VIDĚT ODPOVÍDAJÍCÍ SÉRIOVÝ PORT?

- Pokud používáte USB pro připojení desky Arduino, ujistěte se, že jste nainstalovali ovladače FTDI. Pokud používáte adaptér USB na virtuální sériový port, ujistěte se, že jste nainstalovali ovladače.
- Ujistěte se, že je deska zapojena: nabídka sériového portu se obnovuje při každém otevření nabídky Nástroje, takže pokud jste právě odpojili desku, nebude v nabídce.
- Zkontrolujte, zda nejsou spuštěny žádné programy, které kontrolují sériové porty, jako jsou aplikace pro synchronizaci mobilních zařízení, ovladače Bluetooth, virtuální démonové nástroje atd.
- V systému Windows může být port COM přiřazený k desce příliš vysoký.

## PROČ SE ZDÁ, ŽE SE PROGRAM ÚSPĚŠNĚ NAHRÁVÁ, ALE NIC NESTANE?

- Z nabídky **Nástroje>Mikrokontroléry** jste vybrali nesprávnou položku. Ujistěte se, že vybraný mikrokontroler odpovídá tomu, který používáte (buď ATmega8 nebo ATmega168) - jméno je napsáno na největším čipu na desce.
- Alternativně může být program pro desku příliš velký. Pokud je větší, bude nahrán pouze část programu, ale software to nebude vědět a vaše deska se bude neustále resetovat, zastavovat a resetovat.

## JAK LZE ZMENŠIT VELIKOST PROGRAMU?

- Čip ATmega168 na desce Arduino je sice levný, ale má k dispozici pouze 16kB pro programový kód, což není moc a 2 kB je používán bootloaderem.
- Pokud používáte plovoucí bod, zkuste přepsat kód s celočíselnou matematikou, což by mělo ušetřit zhruba 2 kB. Odstraňte všechny příkazy **#include** v horní části kódu pro knihovny, které nepoužíváte.
- V opačném případě zjistěte, zda můžete program zkrátit.

## PROČ NEFUNGUJE PWM (ANALOGOVÝ VÝSTUP), KDYŽ VOLÁM ANALOGWRITE () NA PINECH JINÝCH NEŽ 3, 5, 6, 9, 10 NEBO 11?

- Mikrokontrolér na desce Arduino (ATmega168) podporuje pouze PWM / analogWrite() na určitých pinech. Volání analogWrite() na jakémkoliv jiném pinu poskytne 5 voltů pro hodnoty větší než 128 a 0 voltů pro hodnoty menší než 128.

# POZNEJTE SVÉ NÁSTROJE

V RÁMCI PŘÍPRAVY NA SESTAVOVÁNÍ OBVODŮ A JEJICH PROGRAMOVÁNÍ JE VHODNÉ SE SEZNÁMIT S NĚKOLIKA ZÁKLADNÍMI POJMY, SOUČÁSTKAMI A ZAPOJENÍMI.

Elektřina je typ energie, podobně jako teplo, síla nebo světlo. Elektrická energie proudí vodiči. Elektrickou energii můžete přeměnit na jiné formy energie. Můžete například, rozsvítit světla nebo poslouchat hudbu z reproduktoru.

Komponenty, jako jsou uvedené reproduktory nebo žárovky, jsou **elektrickými měniči**. Měniče mění ostatní typy energie na elektrickou a naopak. Součástky, které mění ostatní formy energie na elektrickou energii jsou často nazývány **senzory** a součástky, které mění elektrickou energii na ostatní formy energie jsou nazývány **pohony**. Budete vytvářet obvody s různých komponent. Obvody jsou uzavřené smyčky vodičů se zdrojem energie (baterie). Snažíme se, aby tato energie byla účelně využita.

V elektrickém obvodu elektřina „teče“ z bodu o vyšší potenciální energii (tentot bod se obvykle značí znaménkem **+**) do místa s nižší potenciální energií. Zemnící bod (označován znaménkem **-** nebo zkratkou GND) je obecně bodem s nejmenší potenciální energií v elektrických obvodech. V obvodech, které budete stavět, je tok elektrické energie pouze v jednosměrně. Říkáme, že obvodem protéká stejnosměrný elektrický proud, zkráceně označovaný **DC**, který protéká obvodem stále stejným směrem, na rozdíl od proudu střídavého, který je označován **AC** a mění svůj směr 50 nebo 60krát za sekundu. To je elektrická energie, která „přichází ze zásuvky“.

Existuje několik termínů, které byste měli při práci s elektrickými obvody znát. **Proud** (měřený v ampérech je označován symbolem **I**) vyjadřuje množství elektrického náboje, který projde vodičem o určitém průřezu za jednotku času. **Napětí** (měřeno ve voltech a značeno symbolem **U**) je rozdíl elektrických potenciálů mezi dvěma body. A nakonec **odpor** měřený v ohmech a značený symbolem **R**), který brání pohybu volných nosičů elektrického náboje, tedy proudu.

## JEŠTĚ NĚKOLIK VĚCÍ O OBVODECH

- Elektrický obvod musí být uzavřený, tzn. že musí existovat úplná cesta od zdroje (napájení) k bodu s nejmenší energií, jinak obvod nebude fungovat.

- Veškerá elektrická energie se vyčerpá v obvodu a jeho komponentech. Komponenty přemění elektrickou energii na jiné druhy energie, tzn. napětí a proud je přeměněno na jinou formu energie (teplo, světlo, hudba atd.).
- Elektrický proud při průchodu obvodem bude vždy stejný ve vstupním i výstupním bodě.
- Elektrický proud bude vždy hledat cestu s nejmenším odporem. Pokud budou existovat dvě cesty, tak proud si vybere tu s menším odporem. Pokud vytvoříte přímé spojení zdroje se zemí bez odporu, tak dojde ke zkratu. Při zkratu může zdroj energie a vodiče přeměnit elektrickou energii na světlo a teplo, což může vést ke zničení zdroje nebo k požáru.

## OHMŮV ZÁKON

Ohmův zákon vyjadřuje vztah mezi elektrickým odporem, napětím a proudem. Jedná se zároveň o definici elektrického odporu.



Pro zapamatování vzorce k výpočtu veličin Ohmova zákona můžete využít schéma, ve kterém jsou znázorněny vztahy mezi proudem, napětím a odporem. Zakryjte výšeč veličiny, kterou chcete vypočítat. Zbylé veličiny určují vztah.

$$U = I \cdot R$$



$$I = U / R$$



$$R = U / I$$



## CO JE KONTAKTNÍ DESKA?

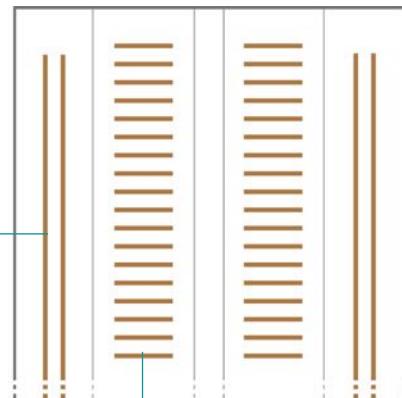
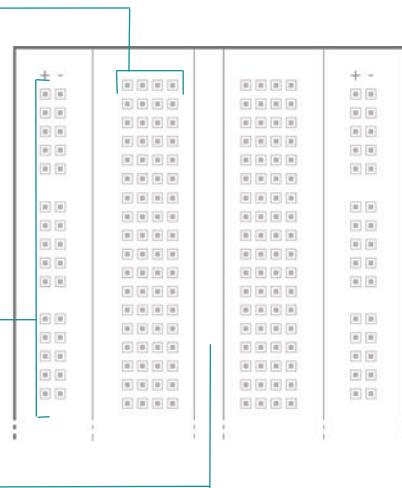
Kontaktní deska je základní místo pro vytváření elektronických obvodů. Měla by být součástí vaší Arduino sady. Výhodou kontaktní desky je, že nemusíte pro vytváření spojů elektronických součástí používat pájku. Jednoduše využijete otvorů v desce pro přidání částí obvodu viz Obr. 1 - Princip kontaktní desky.

Každá řada horizontálních zdírek kontaktního pole, jsou spojeny vodičem, který je uvnitř kontaktní desky.

Vertikální řada zdírek je propojena vodičem po celé délce kontaktní desky. Tyto zdírky jsou obvykle používány pro přivedení napájení a země.

Oddělující sloupek dvou polovin kontaktní desky.

Uvnitř kontaktní desky jsou vodiče propojující zdírky, jak je vidět na obrázku.



Obr. 1 - Princip kontaktní desky

## STAVÍME OBVOD

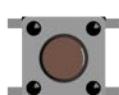
Pro seznámení s kontaktní deskou si vytvoříme první, jednoduchý a elektronický obvod. Tento obvod bude obsahovat spínač, rezistor a LED diodu. Deska Arduino nám poslouží pouze pro napájení desky. V dalších projektech využijeme pro komunikaci vstupních a výstupních pinů pro komplexnější řízení obvodu.



**LED** je anglická zkratka Light-Emitting Diode – dioda emitující světlo, přesně přeloženo: světlo vyzařující dioda. Je to polovodičová elektronická součástka, jejíž vlastností je schopnost vyzařovat světlo. Přeměňuje elektrickou energii na světelnou. LED diody jsou polarizované součástky. Dlouhá nožička diody je **anoda** (+) a připojuje se vždy ke zdroji napájení, kratší nožička je **katoda** (-) a připojuje se na zem.



**Rezistor** je elektronická součástka, která implementuje elektrický odpor jako obvodový prvek. V elektronických obvodech se používají rezistory ke snížení velikosti elektrického proudu nebo k získání určitého úbytku napětí. Když dáme rezistor do série s LED diodou, tak napětí za rezistorem bude menší. Tato funkce rezistoru o správné hodnotě zajistí, že dioda může svítit méně, ale nedojde k jejímu zničení.



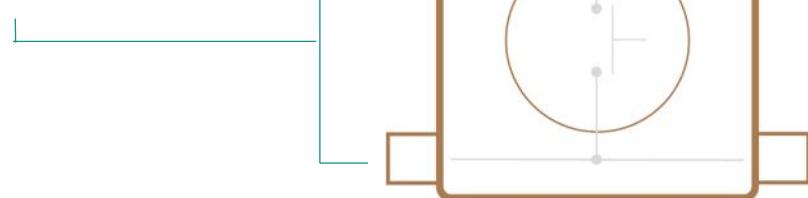
**Spínač** přerušuje elektrický obvod a při jeho stisknutí se naopak obvod spojí. Existuje celá řada spínačů. Mohou být páčkové, tlačítkové, rozepínací, spínací apod.

## ZAPOJENÍ SPÍNAČE

Tyto dva konektory jsou vzájemně propojeny



Tyto dva konektory nejsou vzájemně propojeny. Tvoří samotný spínač



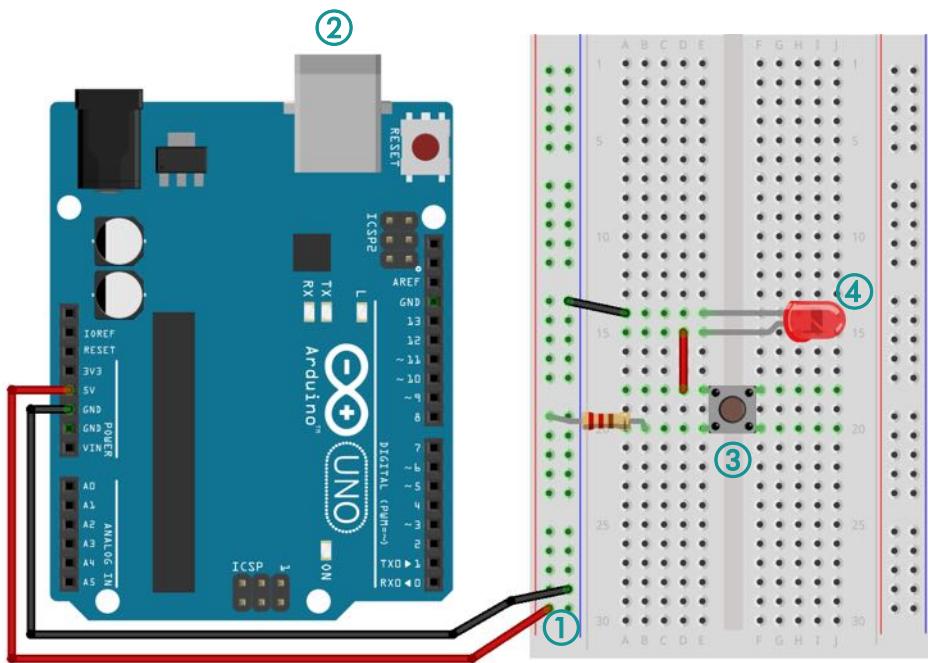
## SCHÉMA SPÍNAČE A VYPÍNAČE



A – symbol spínače



B – symbol vypínače



Jak bylo uvedeno výše, v tomto obvodu bude deska Arduino sloužit pouze k napájení kontaktní desky. Když připojíme kabel USB k desce a druhý konec k počítači, nebo 9V zdroji, Arduino bude poskytovat mezi pinem 5V a zemí GND napětí 5V. Symbol na desce 5V = 5 voltů.

- ① Před sestavováním obvodu odpojte USB kabel nebo napájení formou externího zdroje.
- ② Červený vodič připojte na desku Arduino do pinu 5V a druhý konec vodiče do kontaktního pole, do části určené pro napájení (červený pruh). Černý vodič připojte na desku Arduino do pinu GND. Druhý konec vodiče připojte do kontaktního pole, do části pro zem (modrý pruh). Pravidlem při zapojování elektronických obvodů je, že červený vodič je určený vždy pro napájený a černý pro zem.
- ③ Nyní, když je přivedeno do desky napětí, tlačítko umístěte na střed kontaktního pole. Nožičky spínače směřují od středu desky ke kontaktům.
- ④ Pro přivedení napětí k diodě se použije rezistor  $220\Omega$ . Ten je připojen před tlačítko. Určení hodnoty rezistoru je uvedeno v následující kapitole. Důležité je, aby delší kontakt diody byl připojen k druhé části spínače. Kratší kontakt diody je připojen na zem. Když jsou všechny části obvodu zapojené, připojte k desce Arduino USB kabel.

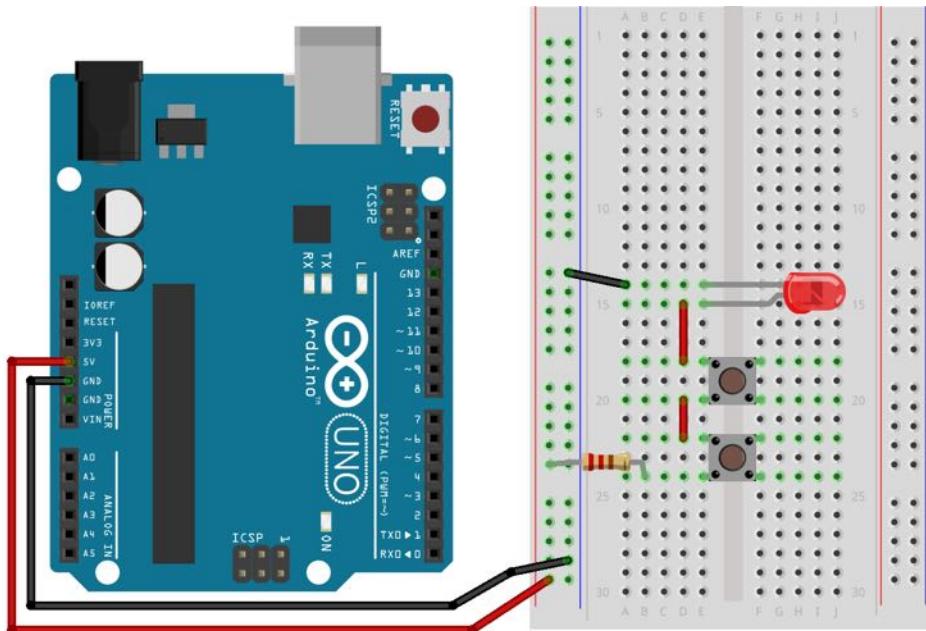
**Pokud vše funguje, jak má, při stisknutí tlačítka se rozsvítí dioda.**

## SÉRIOVÝ OBVOD

ZAPOJENÍ ELEKTRONICKÝCH SOUČÁSTEK DO SÉRIE ZNAMENÁ, ŽE JSOU ZAPOJENY ZA SEBOU.



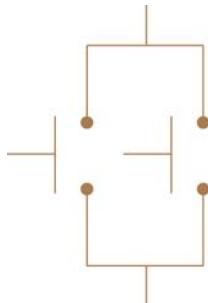
Nejdříve odpojte napájení a přidejte do obvodu na kontaktní pole jeden spínač. Tento spínač bude zapojen do série s předchozím, tak je uvedeno na obrázku Obr. 2 - Zapojení do série. Anodu diody připojte k druhému spínači. Katoda diody je připojena na zem. Nyní připojte napájení. Dioda bude svítit pouze tehdy, pokud budou stisknuta obě tlačítka, protože jsou spínače zapojeny do série a okruh musí být uzavřen.



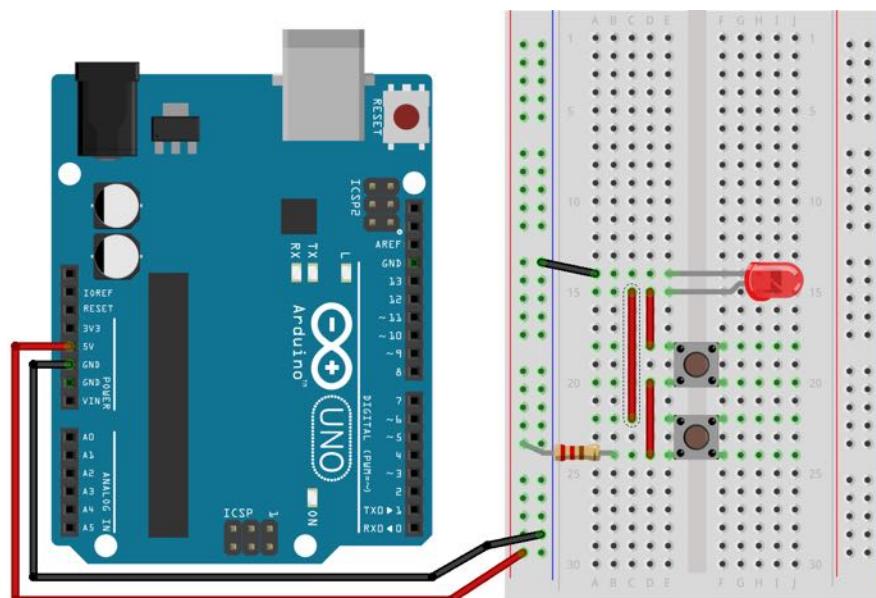
Obr. 2 - Zapojení do série

## PARALELNÍ OBVOD

ZAPOJENÍ ELEKTRONICKÝCH SOUČÁSTEK PARALELNĚ ZNAMENÁ, ŽE JSOU ZAPOJENY VEDLE SEBE.



Nyní, když jste zvládli zapojení spínačů do série, zkusíme je zapojit paralelně. Ujistěte se, že spínače jsou na kontaktním poli na stejném místě. Odstraňte pouze červenou propojku mezi oběma spínači a přidejte propojky tak, jak je uvedeno na obrázku Obr. 3 - Paralelní zapojení. Opět připojte napájení. Nyní, když stisknete některé z tlačítek, dioda se rozsvítí.

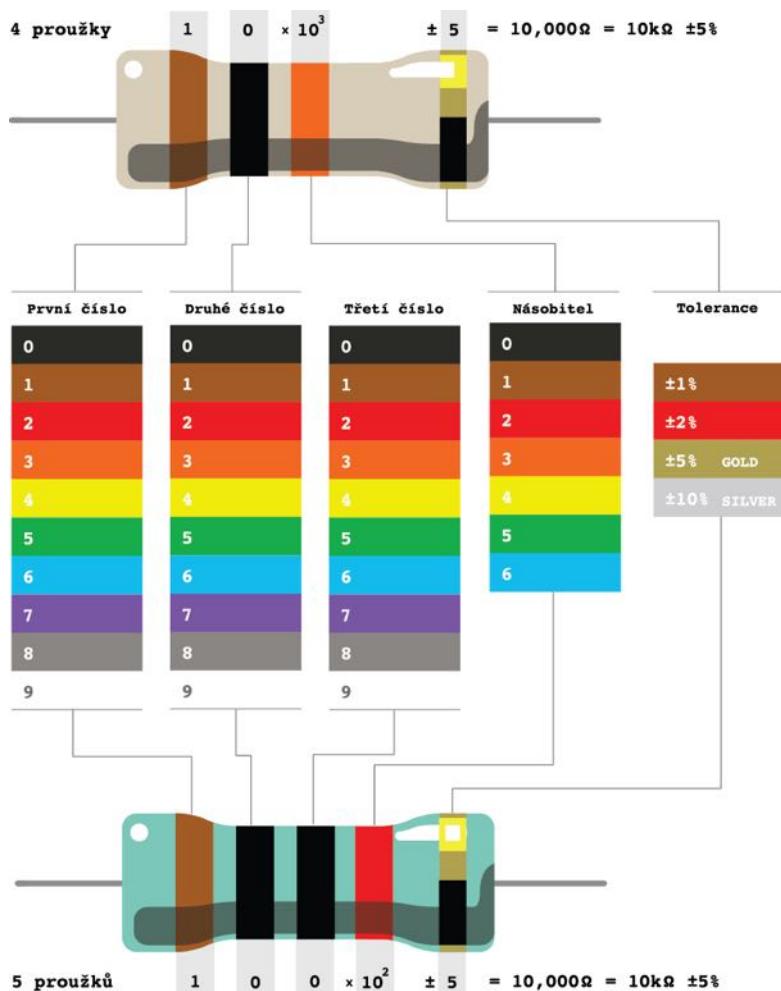


Obr. 3 - Paralelní zapojení

## JAK SE ČTOU HODNOTY REZISTORŮ

Hodnoty rezistorů jsou vyznačeny pomocí barevných proužků. Toto značení bylo zavedeno již v roce 1920.

Nejdříve se musí zjistit, na které straně rezistoru jsou proužky blíže ke kraji. Tam je začátek barevného kódu a odtud se stanoví pořadí proužků k opačnému konci. První dva proužky zleva (u pěti proužkového kódu tři) udávají dvojcíslí (trojčíslí), které patří do číselné řady jmenovitých hodnot. Třetí proužek (čtvrtý u 5-ti proužkového) určuje mocninu daného čísla. Čtvrtý proužek (pátý u 5-ti proužkového) značí dovolenou toleranci v procentech. Stříbrný nebo zlatý proužek na kraji rezistoru poslouží jako dobrý orientační bod, protože tento proužek je vždy vpravo.



# 1. PRVNÍ PROGRAM

LEKCE JE ZAMĚŘENA NA PRVNÍ SEZNÁMENÍ S PROGRAMOVATELNOU DESKOU ARDUINO, KTERÁ JE SOUČÁSTÍ DOPORUČENÉ SADY. VYUŽÍVÁ SE K TOMU JEDNODUCHÝCH OBVODŮ S LED DIODOU A BZUČÁKEM.

## CÍLE

- ① Co je digitální vstup a výstup.
- ② Ovládání LED diody pomocí desky Arduino.
- ③ První seznámení s programovacím jazykem Wiring a jeho základní strukturou.
- ④ Seznámení s programem Arduino IDE pro nahrávání kódu do desky.
- ⑤ Princip a zapojení piezzo bzučáku.
- ⑥ Zvládnutí programování ovládání LED diody a bzučáku.

Čas: **2x45 min**

Úroveň: ■■■■■

Vychází z: **1**



LED



Rezistor 220Ω



Piezzo bzučák

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví základní obvod, ve kterém použijí LED diodu s rezistorem. Tento obvod naprogramují podle vzorového příkladu a samostatných úkolů. Měli by se především seznámit s deskou Arduino a kontaktním polem. Naučí se postup při psaní programu a jeho nahrávání do desky.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 1x LED dioda, 1x rezistor  $220\Omega$ , 2x vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 1.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⏰ 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude naučit se základům programování ovládání elektronických obvodů, které řídí mikrokontrolér. Ty pak mohou tvorit části robotických systémů, inteligentních domů, ovládání automobilů apod.

### **ZEPTEJTE SE STUDENTŮ**

#### **→ Víte, co to jsou embedded systémy?**

Embedded systémy (ES) – někdy se uvádí vestavěné systémy, jsou kombinací hardwarového a softwarového vybavení. Jsou to systémy jednoúčelové, určené pro definované činnosti. ES bývají součástí nějakého systému většího. Obsahují řídící počítač, který je zcela zabudován do zařízení, které ovládá a plní konkrétní účel.

#### **→ Kde všude se s embedded systémy můžete potkat?**

Dnes již prakticky všude: televize, mobilní telefony, automobily atd.



Studenti si sady mohou prohlédnout. Vysvětlete, že se s jednotlivými částmi budou postupně seznamovat na praktických příkladech. Není nutné všechny součásti sady Arduino vysvětlovat.

Ukažte jim pouze desku Arduino a řekněte, že tato deska obsahuje vstupy a výstupy, kterým se říká **piny** a srdcem desky je procesor (mikrokontrolér).

## **2. KROK** 10 minut

Nyní přistupte k sestavení prvního obvodu. Schéma obvodu můžete promítat pomocí dataprojektoru, nebo jej studenti sestaví podle pracovního listu. Vysvětlete jim princip kontaktního pole. Upozorněte studenty na to, že dioda má kratší a delší vývod. Na delší vývod musí být přivedeno napětí a na kratší zem z desky Arduino. Napětí se přivádí z výstupního pinu, který budeme ovládat programem.

Řekněte studentům, že LED dioda se může poškodit, proto se k napájení přidává rezistor.

## **3. KROK** 10 minut

Ať si studenti spustí Arduino IDE a napíší základní program.



Uvedený příklad studenti ani nemusí psát, stačí využít hotového příkladu. V Arduino IDE stačí otevřít **File > Examples > Basic > Blink**. Otevře se základní programový kód.

Řekněte studentům, aby připojili USB kabel k desce a do počítače. Pokud mají program připravený, ukažte jim, jak mají nastavit v rozhraní Arduino IDE desku, do které budou kód nahrávat, a port.

Kliknutím na ikonu pro upload kódu ať nahrají program do desky Arduino.



#### ZEPTEJTE SE STUDENTŮ

→ Co se děje s diodou, pokud je program v pořádku nahrán?

Dioda bliká v intervalu jedné sekundy.

Studentům vysvětlete programový kód, zejména pak základní strukturu programu a použité funkce pro zápis hodnot na pinu desky.

#### 4. KROK 15 minut

Pokud studenti vše zvládli, mohou řešit **samostatně úkoly**. Pokud je nestihnou v této hodině, mohou v nich pokračovat v hodině další.

#### ÚKOLY PRO STUDENTY

- Připojte druhou LED diodu, třeba na pin 12.
- Sestavte program pro střídavé blikání obou LED diod, vždy po 1 sekundě.
- Sestavte program pro střídavé blikání LED diod, ale tentokrát každá blikne dvakrát, vždy po 1 sekundě. Mezi oběma diodami bude pauza dvě sekundy.
- Upravte program tak, aby diody blikaly postupně, tj. rozsvítí se první, po 0,25s se rozsvítí druhá, v čase 0,5s zhasne první a v čase 0,75 shasne druhá. V čase 1 sekundy se rozsvítí první a pořád dokola.



# PRACOVNÍ LIST – LED DIODA

PRVNÍ SEZNÁMENÍ SE SESTAVOVÁNÍM ELEKTRONICKÝCH OBVODŮ A PROGRAMOVÁNÍM DESKY ARDUINO. V TÉTO ČÁSTI VYTVOŘÍTE JEDNODUCHÝ ELEKTRONICKÝ OBVOD S LED DIODOU.

## CO SE NAUČÍTE

- ① Základní použití desky Arduino.
- ② Princip a používání kontaktního pole.
- ③ Zapojení jednoduchého obvodu.
- ④ Vytvoření prvního programu pro ovládání LED diody.



## CO BUDETE POTŘEBOVAT

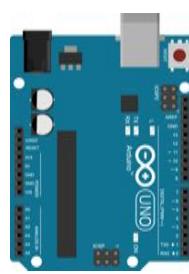
- ① LED diodu.
- ② Rezistor 220Ω.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



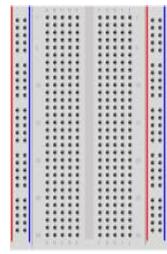
Led



Rezistor 220Ω



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

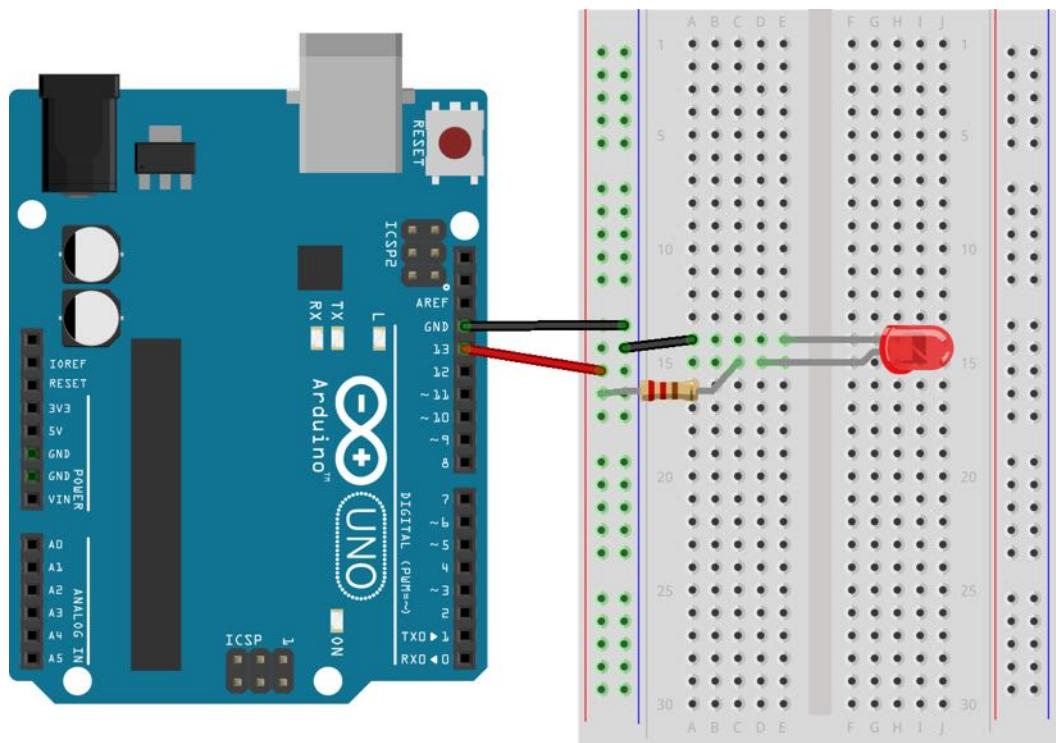
## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.



### DEJTE SI POZOR

- Pozor si dejte na to, jak zapojujete LED diodu. Delší vývod musí být připojen přes rezistor k pinu. Kratší vývod je připojen na zem (pin GND).
- Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá, černá, zlatá.



② Spusťte program Arduino IDE a napište následující programový kód.

```
void setup() {
    pinMode(13, OUTPUT); // nastavení pinu 13 jako výstup
}

void loop() {
    digitalWrite(13, HIGH); // nastavení hodnoty HIGH, pro
                           // rozsvícení LED diody
    delay(1000);          // přerušení programu na 1 sekundu
    digitalWrite(13, LOW); // nastavení hodnoty LOW, pro zhasnutí
                           // LED diody
    delay(1000);
}
```

③ Po napsání programu připojte USB kabel k desce a k počítači.

④ V programu Arduino IDE nastavte odpovídající desku. V menu **Tools > Board > Arduino UNO**.

⑤ Dále nastavte port v Menu **Tools > Seriál Ports > vyberte odpovídající port**.

⑥ Pro nahrání programu do desky Arduino, klikněte na ikonu ➔



### OTÁZKA

➔ Co se děje s diodou, pokud je program v pořádku nahrán?

Pokud vše funguje, tak výborně. Můžete se vrhnout na samostatné úkoly.

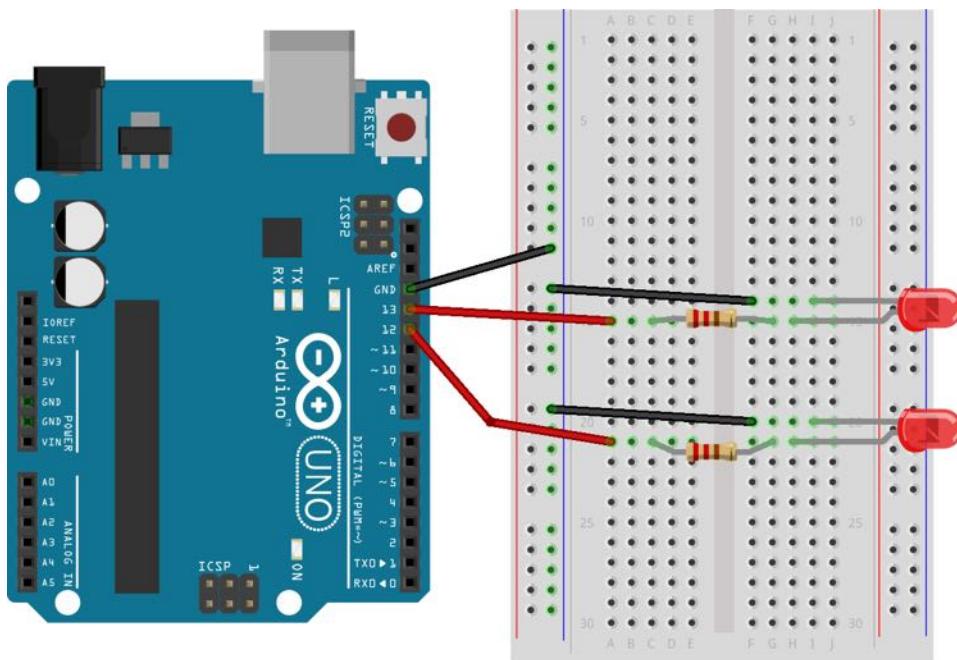
### ÚKOLY PRO VÁS

- ➔ A) Připojte druhou LED diodu, třeba na pin 12.
- ➔ B) Sestavte program pro střídavé blikání obou LED diod, vždy po 1 sekundě.
- ➔ C) Sestavte program pro střídavé blikání LED diod, ale tentokrát každá blikne dvakrát, vždy po 1 sekundě. Mezi oběma diodami bude pauza dvě sekundy.
- ➔ D) Upravte program tak, aby diody blikaly postupně tj. rozsvítí se první, po 0,25 s se rozsvítí druhá, v čase 0,5 s zhasne první a v čase 0,75 s zhasne druhá. V čase 1 s rozsvítí první a pořád dokola.



# ŘEŠENÍ ÚLOH

Úkol A)



Úkol B)

```
1 void setup() {
2     pinMode(13, OUTPUT);
3     pinMode(12, OUTPUT);
4 }
5
6 void loop() {
7     digitalWrite(13, HIGH);
8     digitalWrite(12, LOW);
9     delay(1000);
10    digitalWrite(13, LOW);
11    digitalWrite(12, HIGH);
12    delay(1000);
13 }
```

Úkol C)

```
1 void setup() {
2     pinMode(13, OUTPUT);
3     pinMode(12, OUTPUT);
4 }
5
6 void loop() {
7     digitalWrite(12, LOW);
8     digitalWrite(13, HIGH);
9     delay(1000);
10    digitalWrite(13, LOW);
11    delay(1000);
12    digitalWrite(13, HIGH);
13    delay(1000);
14    digitalWrite(13, LOW);
15    delay(2000);
16    digitalWrite(12, HIGH);
17    delay(1000);
18    digitalWrite(12, LOW);
19    delay(1000);
20    digitalWrite(12, HIGH);
21    delay(1000);
22    digitalWrite(12, LOW);
23    delay(2000);
24 }
```

Úkol D)

```
1 void setup() {
2     pinMode(13, OUTPUT);
3     pinMode(12, OUTPUT);
4 }
5
6 void loop() {
7     digitalWrite(13, HIGH);
8     delay(250);
9     digitalWrite(12, HIGH);
10    delay(250);
11    digitalWrite(13, LOW);
12    delay(250);
13    digitalWrite(12, LOW);
14    delay(250);
15 }
```

# PRŮVODCE HODINOU II



Studenti budou pokračovat v programování elektronických obvodů. Tentokrát v zapojení sestaví základní obvod s využitím piezzo bzučák. Jedná se o volné pokračování předchozího zapojení s LED diodou. Studenti využijí nabyté znalosti získané při zapojování a programování elektronického obvodu s LED diodou. V programování využijí nové funkce určené pro ovládání bzučáku.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 1x piezzo bzučák, 1x rezistor  $220\Omega$ , 1x LED, 2x vodiče typu zástrčka- zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 1, která je ke stažení na serveru GitHub
- ⑤ Pracovní listy pro studenty (ke stažení na GitHub).

## 1. KROK ⏱ 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že v této hodině navází na předchozí příklad, který se týkal zapojení a ovládání obvodu s LED diodou. Naučí se programovat ovládání piezzo bzučáku zapojeného do obvodu, který je velmi podobný obvodu s LED diodou.



### RYCHLÉ OPAKOVÁNÍ

- ➔ Zapojte obvod z předchozí hodiny, kde jste použili LED diodu.
  - ↘ Naprogramujte tento obvod tak, že LED dioda bude blikat v intervalu 1 sekundy.



Opakování můžete urychlit tím, že studenti využijí vzorový příklad, dostupný v Arduino IDE. Stačí otevřít **File > Examples > Basic > Blink**. Otevře se základní programový kód.

## 2. KROK ⏱ 5 minut

Sestavení příkladu s LED diodou vede k tomu, že jej lze velmi snadno inovovat na obvod s piezzo bzučákem. Ať studenti vymění diodu za bzučák podle přiloženého schématu v pracovním listu nebo podle promítaného obrazu pomocí dataprojektoru z prezentace.

## 3. KROK ⏱ 10 minut

Ať si studenti spustí Arduino IDE a napíší základní program. Řekněte studentům, aby připojili USB kabel k desce a do počítače. Kliknutím na ikonu pro upload kódu ➔ ať nahrají program do desky Arduino.



### JENOM PRO PŘIPOMENUTÍ

➔ Pokud nepůjde nahrát program do desky, studenti si musí zkontrolovat, zda mají v Arduino IDE vybranou odpovídající desku a komunikační port.

Studentům vysvětlete programový kód, zejména použité funkce týkající se bzučáku – **Tone**, **noTone**. Určitě nezapomeňte na deklaraci proměnné **bzucak**.



Při vysvětlování programového kódu zkuste formou otázek, zda studenti na funkcionality funkcí **Tone**, **noTone** přijdou sami.

### **ZEPTEJTE SE STUDENTŮ**

→ **Co se stane, když změníte číslo ve funkci Tone za proměnnou bzucak?**

Zvuk bzúčáku změní svou výšku.

→ **Přidejte před funkci noTone dvě lomítka – symbol komentáře.**

→ **Co se po přidání komentáře stane?**

Vyřadí se funkce noTone a bzúčák bude neustále pískat.



## **4. KROK** 20 minut

Pokud studenti vše zvládli, mohou řešit **samostatně úkoly**. Tentokrát postupně použijí jak bzúčák, tak LED diodu.

### **ÚKOLY PRO STUDENTY**

→ **A) Napište program, který přehraje základní stupnici. Interval tónů je na vás.**

(Frekvence: 261, 294, 329, 349, 392, 440, 493, 523)

→ **C) Změňte intervaly a pořadí tónů tak, aby vznikla melodie.**

→ **B) Přidejte do obvodu na libovolný pin LED diodu. Dioda bude blikat v rytmu tónů.**



### **MOŽNÝ NÁPAD**

→ Studenti mohou realizovat závěrečný projekt, který je uveden jako „Mluvící robot“. Předpokladem je, že si z přiložené šablony doma vyrobí model robota, do kterého zabudují LED diodu a bzúčák.

→ Robota naprogramují tak, aby mluvil vlastní, robotickou řečí. Inspirací může být robot R2D2 z Hvězdných válek.

# PRACOVNÍ LIST – PIEZZO BZUČÁK

POKRAČOVÁNÍ V SEZNAMOVÁNÍ SE S PLATFORMOU ARDUINO. TENTOKRÁT BUDETE PROGRAMOVAT ELEKTRONICKÝ OBVOD, VE KTERÉM JE ZAPOJEN PIEZZO BZUČÁK A NÁSLEDNĚ PŘIDÁTE LED DIODU.

## CO SE NAUČÍTE

- ① Zopakujete si, jak vytvořit elektronický obvod.
- ② Dokážete zapojit bzučák.
- ③ Vytvoření programu pro ovládání bzučáku.
- ④ Naprogramujete obvod s bzučákem a LED diodou.



## CO BUDETE POTŘEBOVAT

- ① Bzučák.
- ② Rezistor 220Ω.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

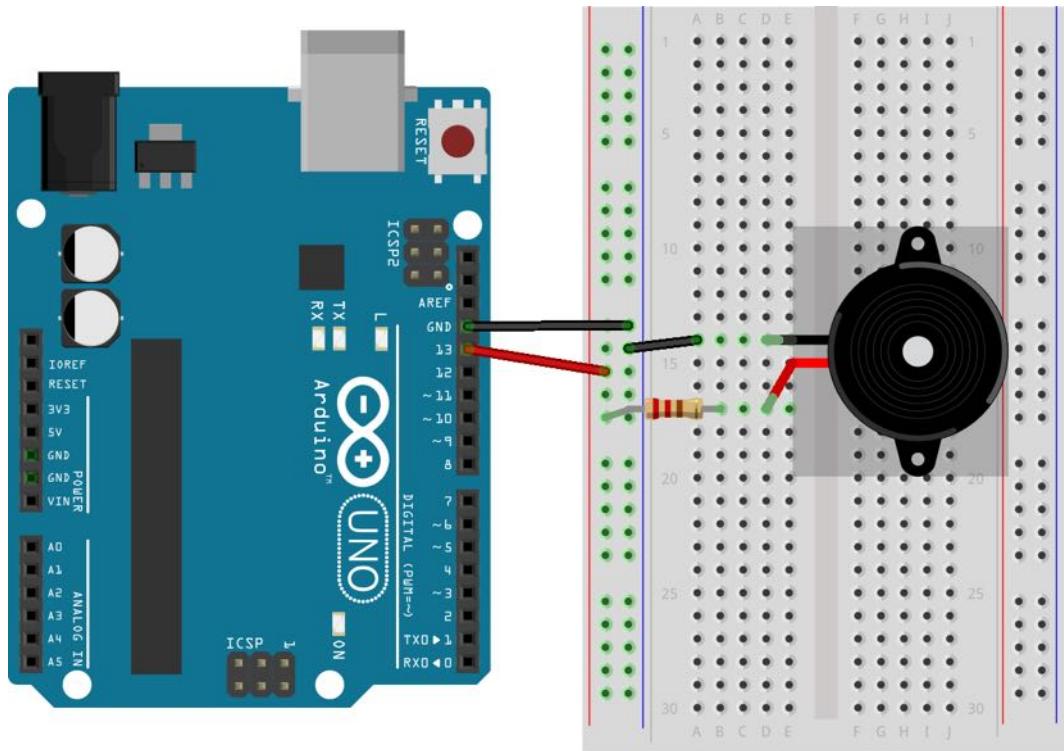
A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.



## DEJTE SI POZOR

- ➔ Pozor si dejte na to, jak zapojujete bzučák. Bzučák má na sobě nálepku s vyznačením kontaktu pro napájení, tj. symbol plus (+).
  - ➔ Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá, černá, zlatá.



② Spusťte program Arduino IDE a napište následující programový kód.

```
const int bzucak=13; // definice proměnné bzucak

void setup() {
    pinMode(bzucak, OUTPUT); // nastavení pinu jako výstup
}

void loop() {
    tone(bzucak, 440); // funkce pro přehrání tónu
    delay(1000);
    noTone(bzucak); // funkce pro přerušení tónu
    delay(1000);
}
```

③ Po napsání programu připojte USB kabel k desce a k počítači.

- ④ V programu Arduino IDE nastavte odpovídající desku. V menu **Tools > Board > Arduino UNO**.
- ⑤ Dále nastavte port v Menu **Tools > Seriál Ports > vyberte odpovídající port**.
- ⑥ Pro nahrání programu do desky Arduino klikněte na ikonu ➔



### OTÁZKA

- ➔ K čemu slouží v tomto příkladu funkce `delay(1000)`?  
Pokud nevíte, tak vyzkoušejte změnit hodnotu parametru a spusťte program s touto novou hodnotou.

Pokud vše funguje, tak výborně. Můžete se vrhnout na samostatné úkoly.

### ÚKOLY PRO VÁS



- ➔ A) Napište program, který přehraje základní stupnici. Interval tónů je na vás.  
(Frekvence: 261, 294, 329, 349, 392, 440, 493, 523)
- ➔ B) Změňte intervaly a pořadí tónů tak, aby vznikla melodie.
- ➔ C) Přidejte do obvodu na libovolný pin LED diodu. Dioda bude blikat v rytmu tónů.

# ŘEŠENÍ ÚLOH

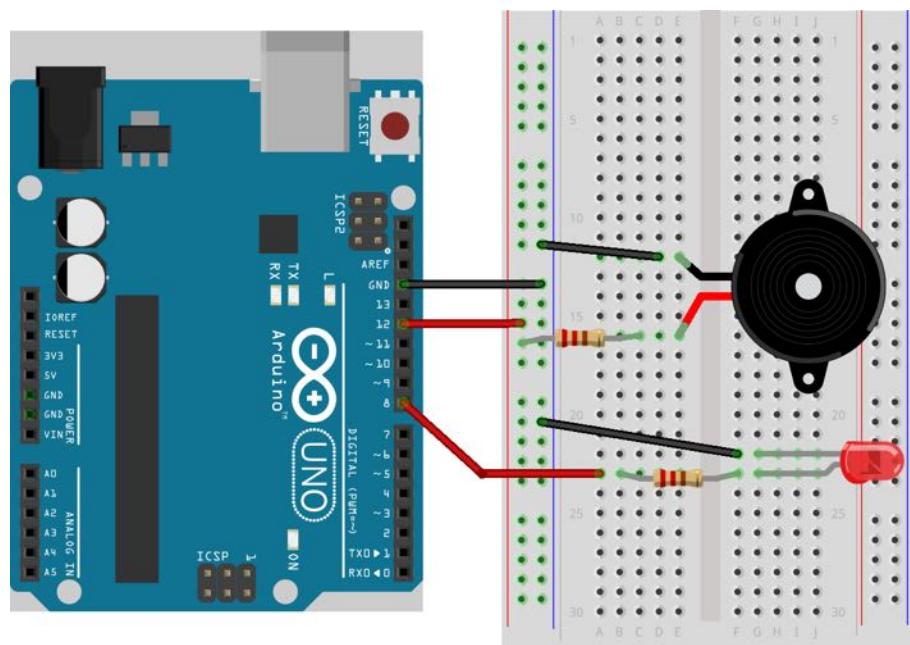
Úkol A)

```
1 const int pinBzucak=13;
2
3 void setup() {
4     pinMode(pinBzucak, OUTPUT);
5 }
6
7 void loop() {
8     // Ton C
9     tone(pinBzucak, 261);
10    delay(1000);
11    noTone(pinBzucak);
12    delay(1000);
13    // Ton D
14    tone(pinBzucak, 294);
15    delay(1000);
16    noTone(pinBzucak);
17    delay(1000);
18    // Ton E
19    tone(pinBzucak, 329);
20    delay(1000);
21    noTone(pinBzucak);
22    delay(1000);
23    // Ton F
24    tone(pinBzucak, 349);
25    delay(1000);
26    noTone(pinBzucak);
27    delay(1000);
28
29    // Další část kódu se neustále opakuje. Mění se pouze
30    // frekvence, které jsou parametrem funkce tone().
31
32 }
```

### Úkol B)

```
1 // Uvedeny kód ukazuje část melodie Jingle Bells.  
2 // Poradi tonu je: E E E P, E E E P, E G C D E P  
3 // Symbol P značí pauzu a bude mít hodnotu 0  
4  
5 const int pinBzucak=13;  
6  
7 void setup() {  
8     pinMode(pinBzucak, OUTPUT);  
9 }  
10  
11 void loop() {  
12     tone(pinBzucak, 329); delay(1000);  
13     noTone(pinBzucak); delay(100);  
14     tone(pinBzucak, 329); delay(1000);  
15     noTone(pinBzucak); delay(100);  
16     tone(pinBzucak, 329); delay(1000);  
17     noTone(pinBzucak); delay(100);  
18     tone(pinBzucak, 0); delay(1000);  
19  
20     noTone(pinBzucak); delay(100);  
21     tone(pinBzucak, 329); delay(1000);  
22     noTone(pinBzucak); delay(100);  
23     tone(pinBzucak, 329); delay(1000);  
24     noTone(pinBzucak); delay(100);  
25     tone(pinBzucak, 329); delay(1000);  
26     noTone(pinBzucak); delay(100);  
27     tone(pinBzucak, 0); delay(1000);  
28  
29     noTone(pinBzucak); delay(100);  
30     tone(pinBzucak, 329); delay(1000);  
31     noTone(pinBzucak); delay(100);  
32     tone(pinBzucak, 392); delay(1000);  
33     noTone(pinBzucak); delay(100);  
34     tone(pinBzucak, 261); delay(1000);  
35     noTone(pinBzucak); delay(100);  
36     tone(pinBzucak, 294); delay(1000);  
37     noTone(pinBzucak); delay(100);  
38     tone(pinBzucak, 329); delay(1000);  
39     tone(pinBzucak, 0); delay(1000);  
40 }  
41 }
```

### Úkol C)



```
1 // Ukázka kódu pro mluvícího robota
2
3 const int pinLed=8;           // pin pro LED
4 const int pinBzucak=12;       // pin pro bzucak
5
6 void setup() {
7     pinMode(pinLed, OUTPUT);
8     pinMode(pinBzucak, OUTPUT);
9 }
10
11 void loop() {
12     tone(pinBzucak, 433);      // neustale se opakujici
13     digitalWrite(pinLed, HIGH); // blok kódu, kde se dale
14     delay(100);                // meni frekvence tonu
15     noTone(pinBzucak);        // a pauza mezi tony
16     digitalWrite(pinLed, LOW);
17     delay(100);
18
19     tone(pinBzucak, 1033);
20     digitalWrite(pinLed, HIGH);
21     delay(300);
22     noTone(pinBzucak);
23     digitalWrite(pinLed, LOW);
```

```
24     delay(300);
25
26     tone(pinBzucak, 600);
27     digitalWrite(pinLed, HIGH);
28     delay(200);
29     noTone(pinBzucak);
30     digitalWrite(pinLed, LOW);
31     delay(200);
32
33     tone(pinBzucak, 800);
34     digitalWrite(pinLed, HIGH);
35     delay(500);
36     noTone(pinBzucak);
37     digitalWrite(pinLed, LOW);
38     delay(500);
39
40 }
```

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU A ŘEŠENÍ ÚKOLŮ A MOŽNÝCH PROBLÉMŮ PŘI NEFUNKČNOSTI OBVODŮ.

## OBSAH PRŮVODCE

- ① Digitální vstup a výstup s popisem funkcí pro jejich ovládání.
- ② Podrobný popis zapojení obvodu s LED diodou.
- ③ Základní seznámení s jazykem Wiring.
- ④ Popis rozhraní Arduino IDE pro nahrání kódu do desky.
- ⑤ Řešené problémy při zapojení LED diody.
- ⑥ Seznámení s programem Arduino IDE pro nahrávání kódu do desky.
- ⑦ Princip a zapojení piezzo bzučáku s podrobným vysvětlením.
- ⑧ Programový kód pro ovládání obvodu s bzučákem.
- ⑨ Technická část pro závěrečný projekt – Mluvící robot.
- ⑩ Vysvětlení řešení samostatných úkolů.

## DIGITÁLNÍ VSTUP A VÝSTUP

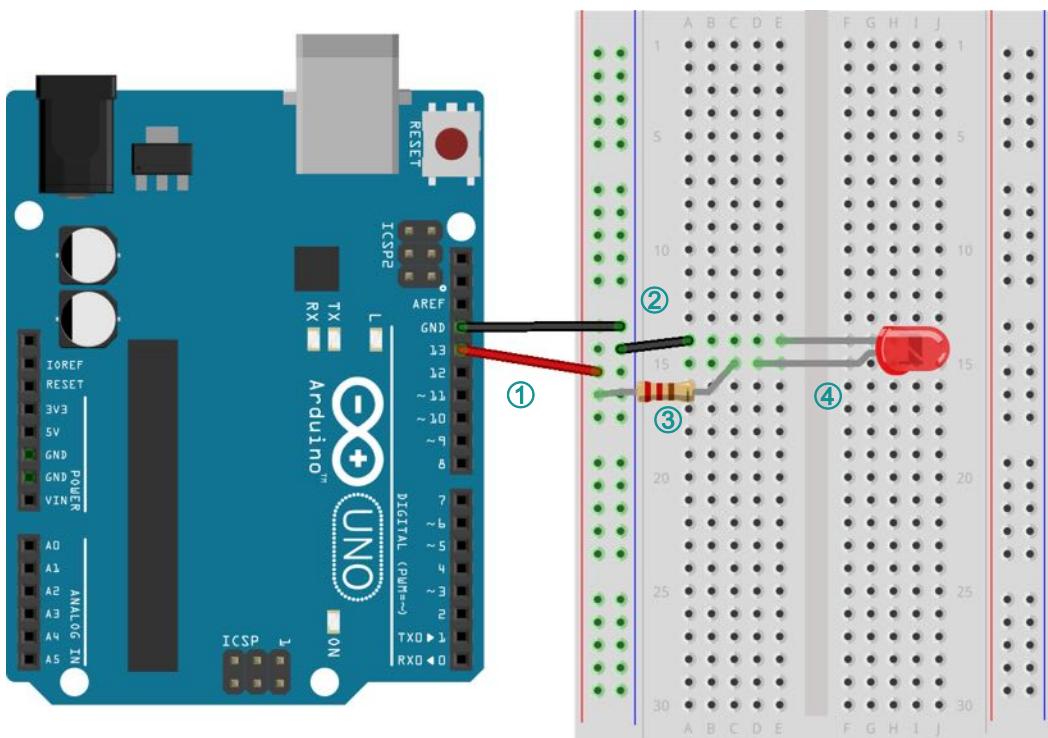
Téměř všechny piny Arduina lze použít jako digitální vstup nebo výstup. Jejich směr lze přepnout zavolením funkce **pinMode** a to v libovolném okamžiku. To znamená, že jeden pin může jednu chvíli sloužit jako vstup a poté i jako výstup. Jelikož se jedná o digitální piny, napětí na nich je na desce Arduino interpretováno pomocí dvou logických hodnot – **HIGH** a **LOW**. Často se tyto hodnoty interpretují také jako logická jedna a nula.

Logická hodnota **HIGH** je reprezentována napětím blízkým napájecímu, tzn. v případě desky Arduino UNO to je 5 V a logická hodnota **LOW** poskytuje napětí blízké 0 V.



Než si ukážeme konkrétní aplikaci programového kódu, vytvoříme elektronický obvod, na kterém bude vysvětlení digitálního vstupu a výstupu názornější.

## ZAPOJENÍ OBVODU S LED DIODOU



Obr. 1 - Zapojení LED na digitální pin

- ① Zapojení obvodu je velmi jednoduché. Na digitální pin **13** je připojen červený vodič. Jeho druhý konec je připojen do kontaktního pole, do zdířky v části pro napájení.
- ② Černý vodič je veden z pinu země **GND** a připojen na kontaktní pole. Dále je připojen k diodě.
- ③ Aby nedošlo k rychlému zničení LED diody, je jí předřazen rezistor **220Ω**. Jedna část rezistoru je zapojena do zdířky kontaktního pole, kde je připojen pin **13** a druhá k anodě LED diody.
- ④ Samotná dioda je připojena do kontaktního pole tak, že anoda (delší kontakt) je spojena s rezistorem a katoda se zemí (černý vodič).



Všimněte si, že obvod je uzavřený. Nikde není mechanický prvek, který by ho rozpojoval tak, jak bylo uvedeno v zapojení znázorňující sériový a paralelní obvod. Ovládání diody bude v tomto obvodu zajišťovat programový kód.



### KRÁTCHE O JAZYKU WIRING

Arduino je možné programovat v jazyce C nebo C++. Nejjednodušší je však používat knihovnu Wiring. Ta je v současné době pro programování Arduina velmi rozšířená. Kvůli její komplexnosti se o ní mluví jako o samostatném programovacím jazyku.

Program pro Arduino v jazyce Wiring se nazývá sketch [skeč] a skládá se z několika základních částí:

- ① deklarace tzv. globálních proměnných, definice konstant a načtení externích knihoven,
- ② definice funkce **setup()**,
- ③ definice funkce **loop()**,
- ④ definice zbývajících funkcí programu.

Podrobné informace a referenční příručku jazyka Wiring naleznete v kapitole **Programovací jazyk Wiring**.

## PROGRAMOVÝ KÓD

K vytváření skečů v programování v jazyce Wiring můžeme použít jakýkoliv textový editor. My ale využijeme software Arduino IDE, který poskytuje kromě možnosti psaní programového kódu i nahrání do desky Arduino. Popis hlavních funkcí softwaru Arduino IDE je uveden v kapitole, [Upload programu do desky](#).

### ZÁKLADNÍ PŘÍKLAD

Zde je uveden základní programový kód, který rozbliká LED diodu.

```
1 void setup() {  
2     pinMode(13, OUTPUT);  
3 }  
4  
5 void loop() {  
6     digitalWrite(13, HIGH);  
7     delay(1000);  
8     digitalWrite(13, LOW);  
9     delay(1000);  
10 }  
11 }
```



- ① Funkce **setup()** je inicializační a vykoná se pouze jednou při způštění programu.
- ② Funkce **pinMode()** nastaví pin 13 jako výstup **OUTPUT**.
- ③ Funkce **loop()** je výkonná a je volána následně po funkci **setup()**. Vykonává se stále dokola.
- ④ Funkce **digitalWrite()** rozsvítí LED diodu připojenou na pin **13** tak, že přepne výstup na hodnotu **HIGH**. Tím se pošle na uvedený pin 5V, což zajistí rozsvícení diody.
- ⑤ Funkce **delay()** přeruší běh programu na dobu uvedenou v závorce. Hodnota se zadává v milisekundách, tj. 1000ms = 1s.
- ⑥ Opět zavoláme funkci **digitalWrite()**, tentokrát však s hodnotou **LOW**, což znamená, že na pinu **13** bude napětí změněno na 0V. Tím dojde k zhasnutí LED diody.
- ⑦ Následuje opět funkce pro pozastavení běhu programu na 1 sekundu.

A to je celý program. Mimo základního kódu ve funkcích **setup()** a **loop()** tento program neobsahuje žádné deklarace, knihovny či jiné funkce. Protože veškerý výkonný kód je uveden ve funkci **loop()**, která ho vykonává pořád dokola, dioda bude neustále blikat.

## UPLOAD PROGRAMU DO DESKY

Program již sice máme napsaný a víme, co má dělat, ale ještě není nahrán v desce Arduino. K tomu využijeme stejný software, jaký jsme použili pro psaní kódu.

- ① Klikněte na ikonu pro ověření kódu. Mělo by se objevit něco jako **Binary sketch size: 934 bytes (of a 30720 byte maximum)**. Tedy že skeč přeložený do binární podoby zabírá 934 bajtů.
- ② Pokud máte přeloženo, klikněte na ikonu pro nahrání do desky. IDE přeloží skeč a nahraje do Arduina. Počká pár sekund a pošle signál RESET. Dioda by se měla rozblíkat.



### Ověřit

Ověří, zda napsaný kód je v pořádku a zkompiluje ho.



### Nahrát

Zkompiluje kód, nahraje jej do nakonfigurované desky.



### Nový

Vytvoří nový skeč.



### Uložit

Uloží skeč.



### Sériový monitor

Otevře sériový monitor.



### NELZE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Chyba v kódu** – zkонтrolujte, jestli je programový kód opravdu správně napsán.

Pokud bude existovat syntaktická chyba, kód se do desky nenahraje.

**Správná deska** – přesvědčte se, že máte správně zvolenou desku v nabídkách

**Tools>Board**.

**Správný port** – před nahráním náčrtu je třeba vybrat správný port v nabídkách

**Tools>Ports**.

### DIODA NEBLIKÁ

**Zapojení diody** – zkонтrolujte, zda je LED dioda správně zapojená, tzn. delší vývod z diody (anoda) je připojen přes rezistor na pin 13.

**Rezistor** – zkонтrolujte hodnotu rezistoru, zda není příliš veliká.



(Př. 1) Připojte ještě jednu LED diodu. Použijte k tomu libovolný pin. Upravte skeč tak, aby diody střídavě blikaly.

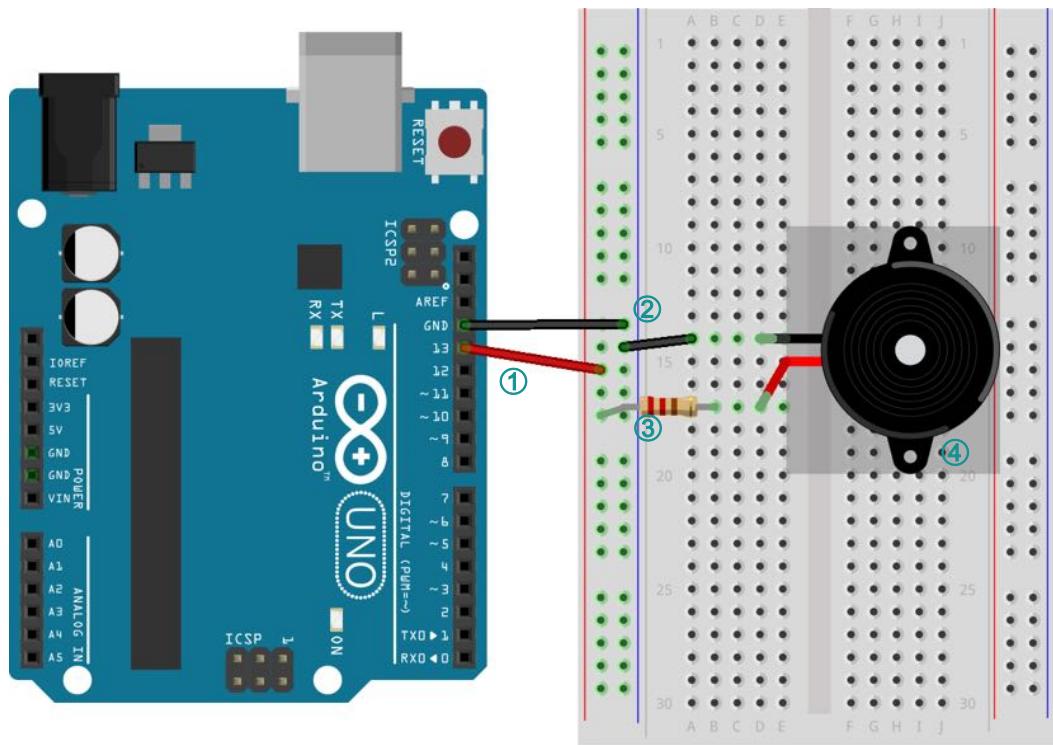
## ZAPOJENÍ OBVODU S BZUČÁKEM

Tento příklad je lehkou inovací předchozího zapojení LED diody. Využívá se zde všech předchozích informací, pouze se použije zařízení, které přeměňuje elektrickou energii na zvuk. Tímto zařízením je bzučák.



### Co je piezzo bzučák?

Piezzo bzučák je malé zařízení, které se rozvibruje, pokud se do něj přivede elektrické napětí. Tím se také rozvibují částice v okolí a vzniká tak zvuková vlna.



Obr. 2 - Zapojení bzučáku

- ① Na digitální pin **13** je připojen červený vodič. Jeho druhý konec je připojen do kontaktního pole, do zdírky v části pro napájení.
- ② Černý vodič je veden z pinu země **GND** a připojen na kontaktní pole. Dále je vodič připojen k bzučáku.

- ③ Aby nedošlo ke zničení bzučáku, je předřazen rezistor **220Ω**. Jedna část rezistoru je zapojena do zdírky kontaktního pole, kde je připojen pin **13** a druhá ke kontaktu bzučáku, který je označen znaménkem **+**.
- ④ Bzučák je připojen do kontaktního pole tak, že kontakt označený znaménkem **+** je spojen s rezistorem a druhý kontakt se zemí (černý vodič).

## PROGRAMOVÝ KÓD

### ZÁKLADNÍ PŘÍKLAD

Příklad je ukázkou využití základních funkcí pro práci s bzučákem.

```

1  const int bzucak=13;           | ①
2
3
4  void setup() {
5      pinMode(bzucak, OUTPUT);   | ②
6  }
7
8  void loop() {
9      tone(bzucak, 440);        | ③
10     delay(1000);
11     noTone(bzucak);          | ④
12     delay(1000);
13 }
```

- ① Deklarace proměnné **bzucak**, která obsahuje číslo pinu, na který je připojen kladný vodič bzučáku.
- ② Definování zvoleného pinu jako výstupního.
- ③ Funkce **tone()** hraje tón na zadaném číslu pinu o dané frekvenci.
- ④ Tón hraje tak dlouho, dokud trvá průběh funkce **delay()**. Tón bude přerušen funkcí **noTone()**, která má jediný parametr a tím je číslo pinu.

Předchozí příklad lze ještě zjednodušit tak, že přidáme třetí parametr do funkce **tone()**. Tímto parametrem je délka trvání tónu.

```
1 const int bzucak=13;                                ①
2
3 void setup() {
4     pinMode(bzucak, OUTPUT);
5     tone(bzucak, 440, 2000);                         ②
6 }
7
8 void loop() {                                       ③
9 }
10 }
```

- ① Deklarace proměnné **bzucak**, která obsahuje číslo pinu, na který je připojen kladný vodič bzučáku.
- ② Definování zvoleného pinu jako výstupního.
- ③ Funkce **tone()** hraje tón a v uvedeném příkladu se využívá třech vstupních parametrů: číslo pinu, frekvence tónu, doba přehrávání. Tato funkce je zámerně napsána v části **setup()**. Pokud by byla v **loop()**, pak by se tón přehrál neustále dokola.



(Př. 2) Zapojte obvod s diodou a piezzo bzučákem společně. Napište program, který bude představovat mluvu robota. Robot bude vydávat zvuky podle vlastního jazyka a zároveň mu bude blikat nos.



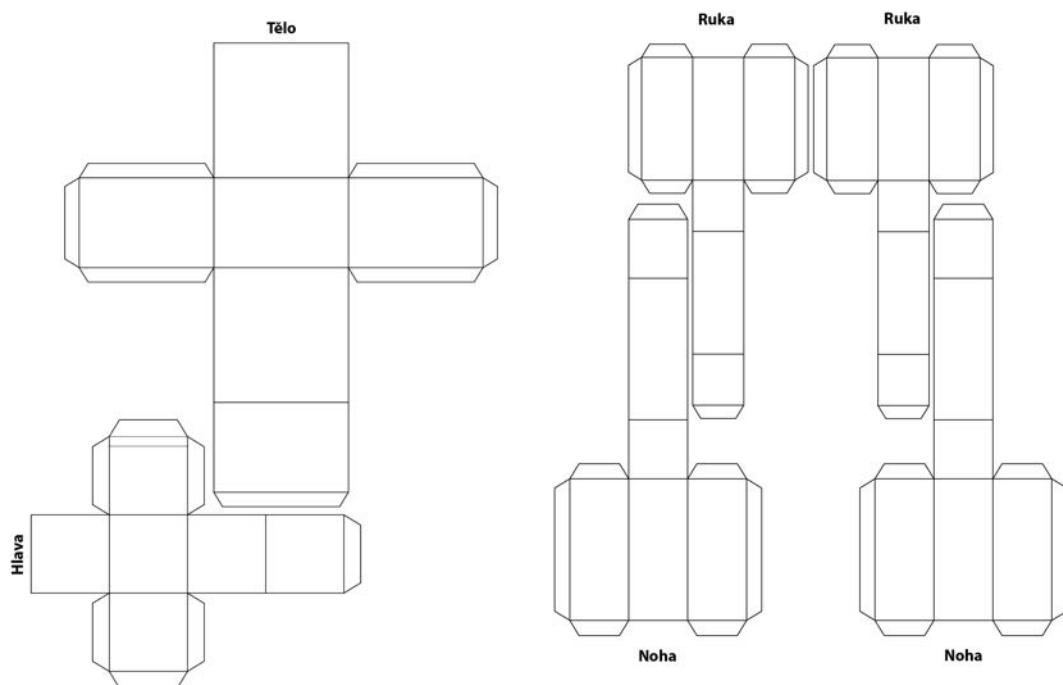
Po zapojení a naprogramování zadанého úkolu sestrojte robota podle přiloženého návodu. Do něj zabudujte diodu a bzučák.

## MLUVÍCÍ ROBOT

Pro vytvoření robota budeme potřebovat: karton (tvrdší papír), lepidlo, nůžky.

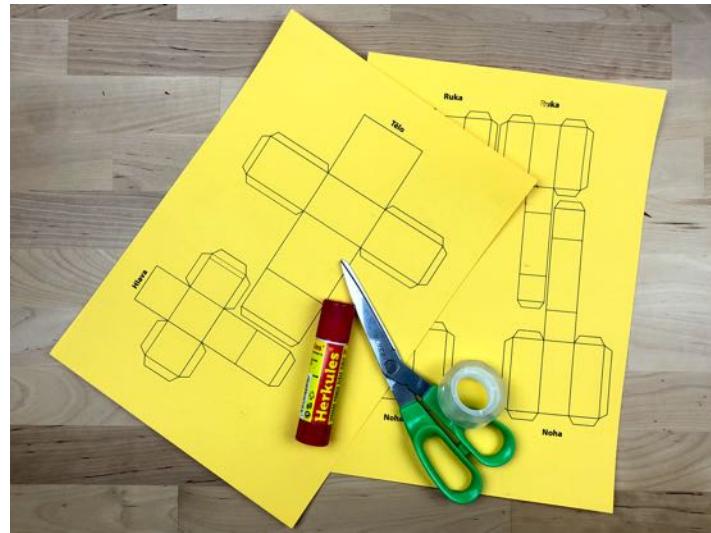
## PAPÍROVÁ KONSTRUKCE

Robota si stáhněte a vytiskněte na karton. Můžete si vybarvit nebo vytvořit zcela vlastního robota podle své fantazie.



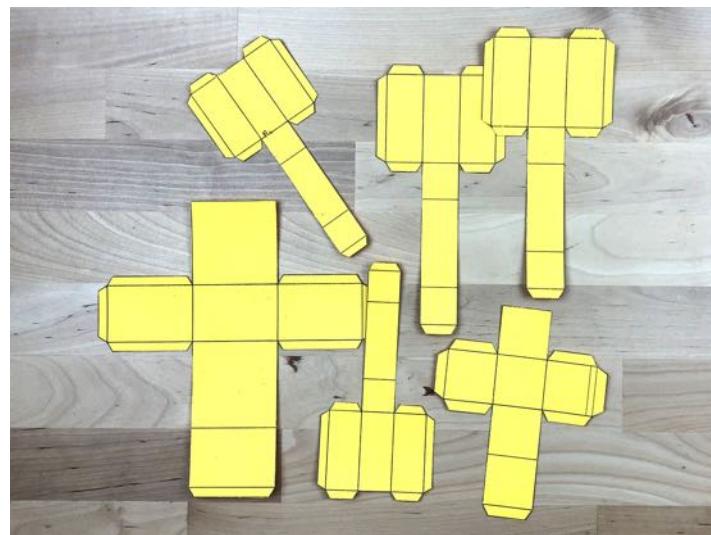
Obr. 3 - Šablona robota

Podle obrázku Obr. 3 - Šablona robota je vidět, že se robot skládá z několika kvádrů. Šablonu si stáhněte z adresy <https://github.com/Nowis75/PRIM/raw/master/sablona-robot.pdf>.



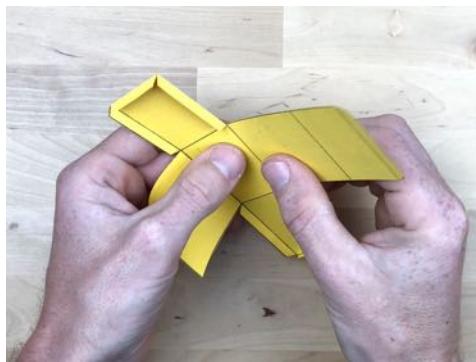
Obr. 4 - Vytiskněná šablona

Podle plných čar vystříhněte vytiskněnou šablonu.



Obr. 5 - Vystřížené části robota

Jednotlivé kvádry složte tak, že naohýbáte jednotlivé hrany podle přerušovaných čar.  
Lepidlo nanesete na spojnicové spoje.



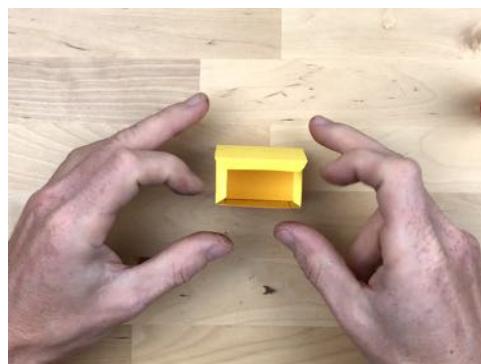
Obr. 6 – Ohýbání lepících hran



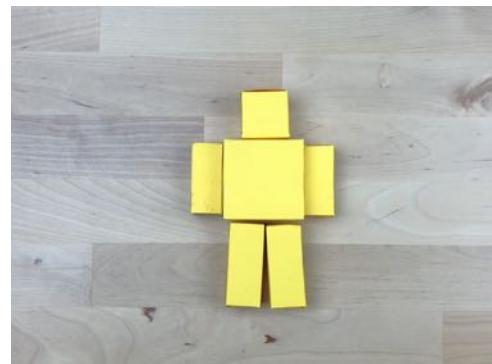
Obr. 7 – Lepení kvádru

Všechny vyštízené tvary složte a slepte. U hlavy a těla nechte horní část kvádru (víko) nezalepné.

Do hlavy robota nainstalujte LED diodu. Udělejte otvor pro diodu a pro vodiče. Pro připojení LED diody použijte vodiče zástrčka > samice.



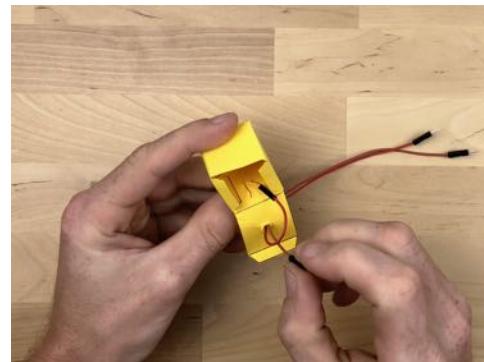
Obr. 8 – Slepěný kvádr



Obr. 9 – Robot z kvádrů

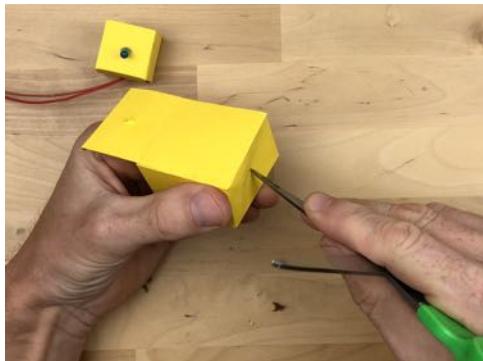


Obr. 10 – Instalace vodičů

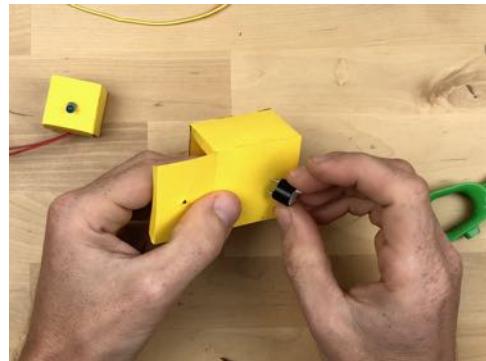


Obr. 11 – Instalace vodičů

Do těla robota nainstalujte piezzo bzučák. Bzučák stačí připevnit na zadní část těla a dovnitř přivést dva vodiče typu zástrčka > samice.



Obr. 12 – Tělo robota



Obr. 13 – Instalace piezzo bzučáku

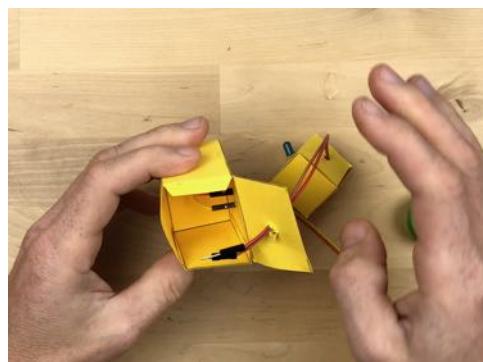


Obr. 14 – Instalace vodičů

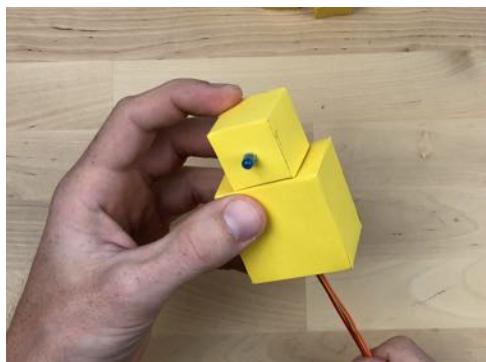


Obr. 15 – Připevnění bzučáku vodiči

Hlavu spojíme s tělem tak, že protáhneme vodiče víkem těla a jeho spodní částí. Z těla budou vycházet čtyři dráty.



Obr. 16 – Instalace vodičů

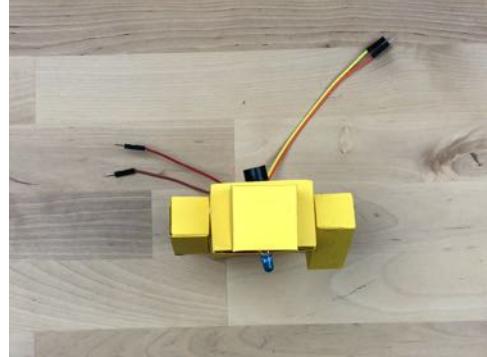


Obr. 17 – Připevnění bzučáku vodiči

Nyní se přilepí hlava k tělu. Ostatní části budou přilepeny také k tělu, Obr. 20 – Sestavený robot.



Obr. 18 – Lepení hlavy k tělu



Obr. 19 – Pohled na robota ze shora

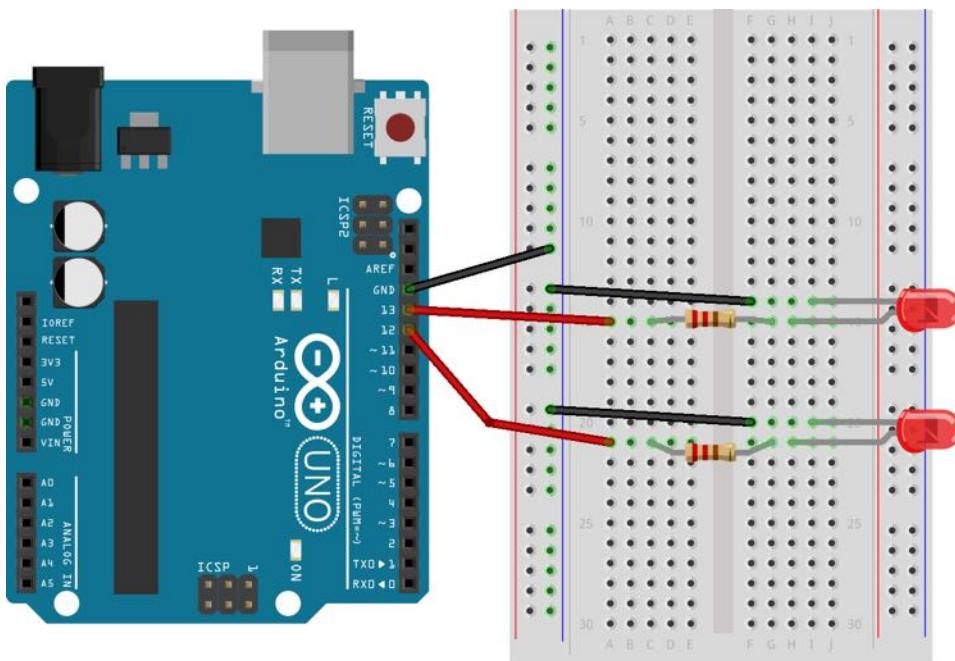


Obr. 20 – Sestavený robot

## ŘEŠENÍ ÚLOH

## (PŘ. 1

Řešení je velmi jednoduché. Je důležité, aby obě diody měly své napájení. Pro jednu z pinu 13 a druhou z pinu 12. Rezistory jsou zapojeny přes oddělovací polovinu kontaktní desky. Vodič země je přiveden do společné části (modrá čára) kontaktní desky a dále rozveden pro každou diodu zvlášť.

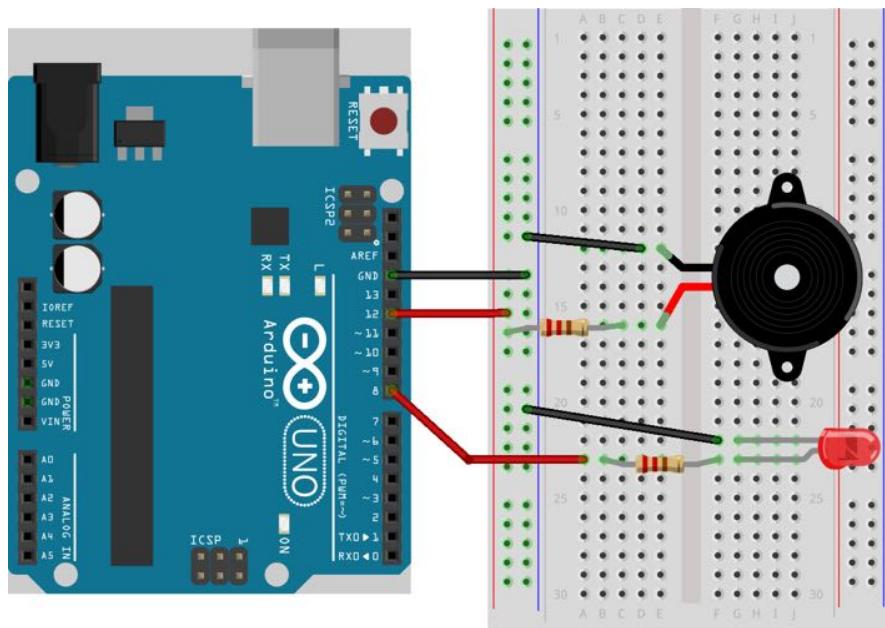


Programový kód je rozšířen o definici druhého výstupního pinu a následné střídání hodnot **HIGH** a **LOW** pro každou diodu pomocí funkce **digitalWrite()**.

```
1 void setup() {
2     pinMode(13, OUTPUT);
3     pinMode(12, OUTPUT);
4 }
5
6 void loop() {
7     digitalWrite(13, HIGH);
8     digitalWrite(12, LOW);
9     delay(1000);
10    digitalWrite(13, LOW);
11    digitalWrite(12, HIGH);
12    delay(1000);
13 }
```

## (PŘ. 2

Příklad je spojením úloh, které se zabývaly LED diodou a bzučákem odděleně. Stačí správně nadefinovat výstupní piny.



V programovém kódu se opakovaně využívá volání funkce **digitalWrite()** pro rozsvěcování a zhasínání diody. Pro zvuk se využívá funkce **tone()**. Důležité je pořadí funkcí a přerušení programu pomocí funkce **delay()**.

```
1 // Ukázka kódu pro mluvícího robota
2
3 const int pinLed=12;           // pin pro LED
4 const int pinBzucak=8;        // pin pro bzucak
5
6 void setup() {
7     pinMode(pinLed, OUTPUT);
8     pinMode(pinBzucak, OUTPUT);
9 }
10
11 void loop() {
12     tone(pinBzucak, 433);      // neustale se opakujici
13     digitalWrite(pinLed, HIGH); // blok kódu, kde se dale
14     delay(100);               // meni frekvence tonu
15     noTone(pinBzucak);        // a pauza mezi tony
```

```
digitalWrite(pinLed, LOW);    //  
delay(100);                  //  
  
tone(pinBzucak, 1033);  
digitalWrite(pinLed, HIGH);  
delay(300);  
noTone(pinBzucak);  
digitalWrite(pinLed, LOW);  
delay(300);  
  
tone(pinBzucak, 600);  
digitalWrite(pinLed, HIGH);  
delay(200);  
noTone(pinBzucak);  
digitalWrite(pinLed, LOW);  
delay(200);  
  
tone(pinBzucak, 800);  
digitalWrite(pinLed, HIGH);  
delay(500);  
noTone(pinBzucak);  
digitalWrite(pinLed, LOW);  
delay(500);  
  
}
```

## 2. SVĚTELNÁ ANIMACE

V TÉTO LEKCI SE ZAMĚŘÍME NA DŮLEŽITÉ PROGRAMOVÉ STRUKTURY, JAKÝMI JSOU VLASTNÍ FUNKCE, POLE A CYKLUS FOR. TO VŠE BUDE APLIKOVÁNO NA JEDINÝ OBVOD, KTERÝ OBSAHUJE NĚKOLIK LED DIOD, ZE KTERÝCH LZE VYTVOŘIT EFEKTNÍ SVĚTELNÉ ANIMACE.

### CÍLE

- ① Rozšířená práce s LED diodami.
- ② Zafixování si dovedností při sestavování obvodů.
- ③ Zavedení vlastní funkce v programovém kódu.
- ④ Vysvětlení a použití příkazu cyklu **for**.
- ⑤ Projekt osmipixelové animace.

Čas: **2x45 min**

Úroveň: ■■■■■

Vychází z: **1**



LED dioda – 8 kusů



Rezistor  $220\Omega$  – 8 kusů

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví obvod, ve kterém použijí několik LED diod s rezistory. Tento obvod naprogramují podle vzorového příkladu, který obsahuje definici **vlastní funkce**. Součástí jsou jednoduché samostatné úkoly, ve kterých studenti využijí znalosti z první lekce, a které vedou k upevnění základů práce s deskou Arduino.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 8x LED dioda, 8x rezistor  $220\Omega$ , vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 2.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⏱ 15 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude se naučit vytvářet vlastní funkce, které zjednoduší a zpřehlední programový kód.



Pro začátek, v rámci opakování, ať studenti sestaví obvod se dvěma diodami a použijí programový kód, který zajistí jejich blikání v intervalu 1 sekunda.



Tento obvod studenti v následujícím úkolu rozšíří o další diody.



### ÚKOL PRO STUDENTY

- Připojte dalších 6 LED diod s předřadnými rezistory.
- Sestavte program, který postupně rozsvítí všechny LED diody a následně je bude opět postupně zhasinat.

## 2. KROK 10 minut

Nyní studentům ukažte prostřednictvím dataprojektoru nebo pracovního listu základní kód, který obsahuje deklaraci a použití vlastní funkce. Vysvětlete studentům základní strukturu deklarace funkce. Zmiňte se o návratovém typu **void**.



### RYCHLÝ TIP

- Pro urychlení práce připomeňte studentům, ať využijí klávesových zkratek pro kopírování (Ctrl+C) a vkládání obsahu (Ctrl+V). Tím se psaní kódu značně zrychlí.

## 3. KROK 20 minut

Na základě vysvětlení principu vlastních funkcí, budou studenti řešit úkoly.



### ÚKOLY PRO STUDENTY

- Upravte programový kód z předchozího příkladu tak, aby byl maximálně zjednodušen. Využijte k tomu vlastní funkci.
- Vytvořte program, který bude simulovat tzv. běžící světlo. Tzn. vždy bude postupně rozsvěcována jediná dioda z celé řady. Po dosažení jedné strany světlo poběží zpět.
- Sestavte program pro běh dvou světel proti sobě (jedno běží zleva doprava). Po dobahnutí se celý cyklus opakuje.



Pokud nyní nebudete pokračovat v hodině a je to možné, bylo by dobré, aby studenti nechali obvod složený do další hodiny. Využijí jej při programování světelných animací s **poli** a cyklem **for**.

# PRACOVNÍ LIST – VLASTNÍ FUNKCE

KAŽDÝ DEN POUŽÍVÁME MONitory NA POČÍTAČÍCH, TELEFONECH NEBO TABLETECH. DISPLEJE NA VĚTŠINĚ ZAŘÍZENÍCH JSOU TVOŘENY Z MILIONŮ PIXELŮ, ZE KTERÝCH SE SKLÁDAJÍ KONKRÉTNÍ OBRAZY. PIXELY MŮŽEME PŘIROVNAT K DROBNÝM LED DIODÁM, KTERÉ POČÍTAČ ROZSVĚCUJE V RŮZNÝCH BARVÁCH A KOMBINACÍCH. TY PAK DOHROMADY NA OBRAZOVCE TVOŘÍ TEXT, OBRÁZKY A VIDEA.

## CO SE NAUČÍTE

- ① Upevníme dovednosti při sestavování obvodů.
- ② Deklarovat vlastní funkce.
- ③ Zapojovat a ovládat další piny desky Arduino.



## CO BUDETE POTŘEBOVAT

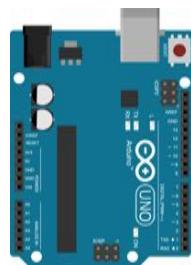
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8x.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



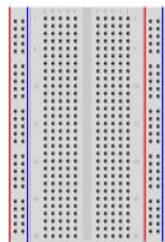
RGB led – 8 kusů



Rezistor  $220\Omega$  – 8 kusů



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

- ① V rámci opakování sestrojte obvod se dvěma diodami.
- ② Napište program, který diody rozblíká v intervalu 1 sekundy.



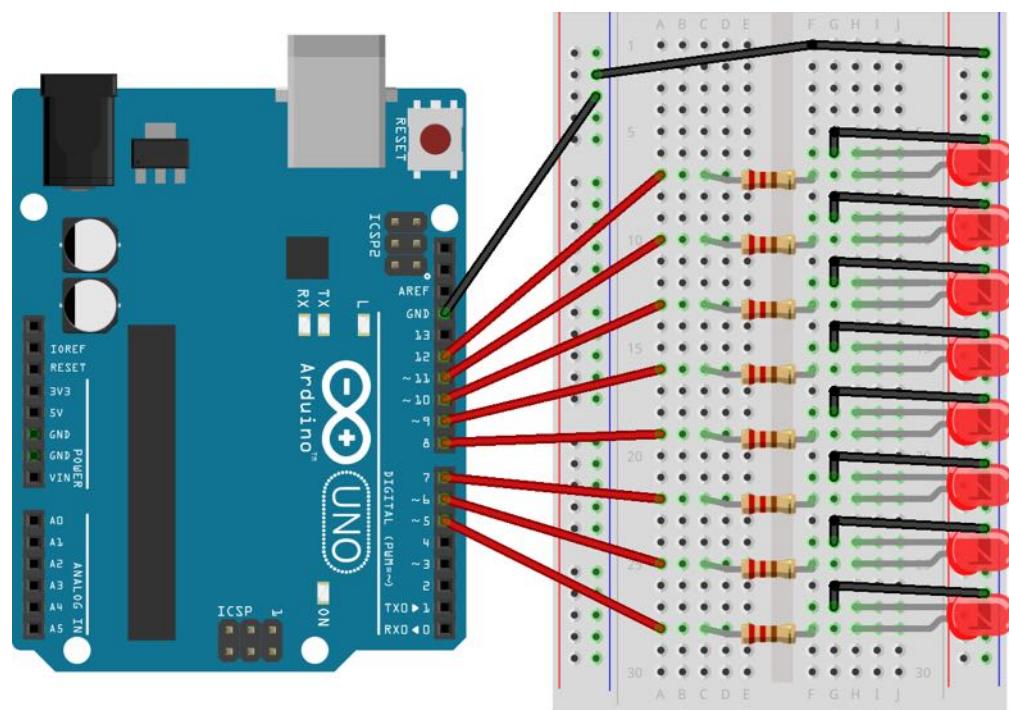
### DEJTE SI POZOR

- Pozor si dejte na to, jak zapojujete LED diody. Delší vývod musí být připojen přes rezistor k pinu. Kratší vývod je připojen na zem (pin GND).
- Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá, černá, zlatá.
- Všimněte si, jak je zapojen vodič zemnění. Pro přehlednost je veden na druhou stranu kontaktního pole. Následně je zemnění vedeno ke každé LED diodě zvlášť (černý vodič).



### ÚKOL PRO VÁS

- Připojte dalších 6 LED diod s předřadnými rezistory podle přiloženého schématu.



- ③ Napište program tak, že využijete deklarovanou vlastní funkci podle následujícího programu.

```
void setup() {
    pinMode(5, OUTPUT);      // dioda 1
    pinMode(6, OUTPUT);      // dioda 2
    pinMode(7, OUTPUT);      // dioda 3
    pinMode(8, OUTPUT);      // dioda 4
    pinMode(9, OUTPUT);      // dioda 5
    pinMode(10, OUTPUT);     // dioda 6
    pinMode(11, OUTPUT);     // dioda 7
    pinMode(12, OUTPUT);     // dioda 8
}

void loop() {
    changeLED (5);
    changeLED (6);
    changeLED (7);
    changeLED (8);
    changeLED (9);
    changeLED (10);
    changeLED (11);
    changeLED (12);
}

void changeLED(int pin) {
    digitalWrite(pin, HIGH);   // rozsvícení diody
    delay(50);
    digitalWrite(pin, LOW);    // zhasnutí diod
    delay(50);
}
```

- ④ Nahrajte program do desky Arduino, kliknutím na ikonu ➔

### ÚKOLY PRO VÁS

- ➔ A) Vytvořte program, který bude simulovat tzv. běžící světlo. Tzn. vždy bude postupně rozsvěcována jediná dioda z celé řady. Po dosažení jedné strany světlo poběží zpět.
- ➔ B) Sestavte program pro běh dvou světel proti sobě (jedno běží zleva a druhé zprava). Po doběhnutí se celý cyklus opakuje.



# ŘEŠENÍ ÚLOH

Úkol A)

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     changeLED (5);
14     changeLED (6);
15     changeLED (7);
16     changeLED (8);
17     changeLED (9);
18     changeLED (10);
19     changeLED (11);
20     changeLED (12);
21     changeLED (11);
22     changeLED (10);
23     changeLED (9);
24     changeLED (8);
25     changeLED (7);
26     changeLED (6);
27 }
28
29
30 void changeLED(int pin) {
31     digitalWrite(pin, HIGH);    // rozsvícení diody
32     delay(50);
33     digitalWrite(pin, LOW);    // zhasnutí diody
34     delay(50);
35 }
```

### Úkol B)

Řešení je několik. Jedním z nich je upravení vlastní funkce. Funkci upravíme tak, že přidáme druhý parametr, kterým bude opět číslo pinu. Tzn., že lze rozsvítit dvě diody najednou.

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     changeLED (5, 12);
14     changeLED (6, 11);
15     changeLED (7, 10);
16     changeLED (8, 9);
17     changeLED (9, 8);
18     changeLED (10, 7);
19     changeLED (11, 6);
20     changeLED (12, 5);
21 }
22
23 void changeLED(int pinA, int pinB) {
24     digitalWrite(pinA, HIGH);   // rozsvícení diody A
25     digitalWrite(pinB, HIGH);   // rozsvícení diody B
26     delay(50);
27     digitalWrite(pinA, LOW);    // zhasnutí diod A
28     digitalWrite(pinB, LOW);    // zhasnutí diod B
29     delay(50);
30 }
```

# PRŮVODCE HODINOU II



Studenti využijí obvod z přechozí hodiny s několika LED diodami a rezistory. Pro naprogramování tohoto obvodu ovšem využijí nové programovací struktury – **pole**. Využití této nové programovací struktury má studenty připravit na další krok, kterým budou cykly. Díky tomu se programový kód velmi zjednoduší.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 8x LED dioda, 8x rezistor 220Ω, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 2, která je ke stažení na GitHub
- ⑤ Pracovní listy pro studenty (ke stažení na GitHub).

## 1. KROK 🕒 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní této hodiny bude naučit se vytvářet efektivnější programovací struktury. Určitě využijí poznatků týkajících se vlastních funkcí, které se naučili programovat předešlou hodinu.



Studenti mohou využít již složený obvod z minulé hodiny. Pokud jej složený nemají, tak jej v rámci opakování sestrojí. Pro další pokračování studentům připomeňte, že diody by měly být zapojeny do pinů 5-12 za sebou. Usnadní to vysvětlování principu polí.

## 2. KROK 10 minut

Nyní studentům vysvětlete princip polí.

- ① Atď si studenti otevřou programový kód z předchozího příkladu.
- ② Ukažte studentům prostřednictvím dataprojektoru nebo pracovního listu základní deklarace polí.
- ③ Vysvětlete, jak se přistupuje k jednotlivým prvkům pole. Nezapomeňte zdůraznit, že k prvkům pole se přistupuje pomocí čísla indexu, který začíná hodnotou **0**.



### ZEPTEJTE SE STUDENTŮ

- **Když se podíváte na otevřený programový kód, jak byste v něm využili pole?**

Pole by se dalo využít při definici pinů. Jednotlivá čísla pinů mohou být prvky pole.

- **V čem byste spatřovali výhodu ve využití pole?**

Výhodou je zejména to, že pokud budeme chtít změnit např. pořadí přístupu k jednotlivým pinům, stačí změnit pořadí čísel pinů v deklarovaném poli a nemusí se upravovat celý program.

## 3. KROK 10 minut

Jako návod pro zodpovězení otázek může být kód uvedený v pracovním listu pro tuto hodinu. Kód si žáci mohou napsat a vyzkoušet.

## 4. KROK 15 minut

### ÚKOL PRO STUDENTY

- A) Upravte otevřený program tak, aby čísla pinů byla nahrazena odkazem na prvky pole.
- B) Změňte směr běžícího světla z opačné strany.
- C) Upravte pořadí prvků v poli tak, aby se diody rozsvěcovali od středu ke krajům.



# PRACOVNÍ LIST – POLE

V TÉTO ČÁSTI BUDETE POKRAČOVAT V PROGRAMOVÁNÍ SOUSTAVY LED DIOD. NAUČÍTE SE OPTIMALIZOVAT PROGRAMOVÝ KÓD POMOCÍ PROGRAMOVÉ STRUKTURY – POLE.

## CO SE NAUČÍTE

- ① Definovat pole pro Arduino.
- ② Využívat pole při definici pinů.
- ③ Procvičíte si přístupy k prvkům pole.



## CO BUDETE POTŘEBOVAT

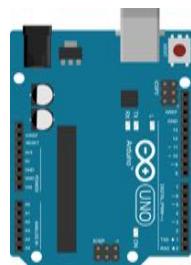
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8x.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



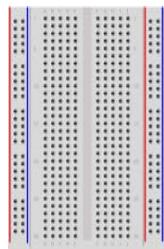
RGB led – 8 kusů



Rezistor  $220\Omega$  – 8 kusů



Deska Arduino

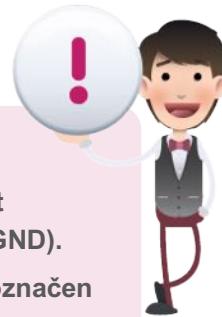


Kontaktní pole

POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

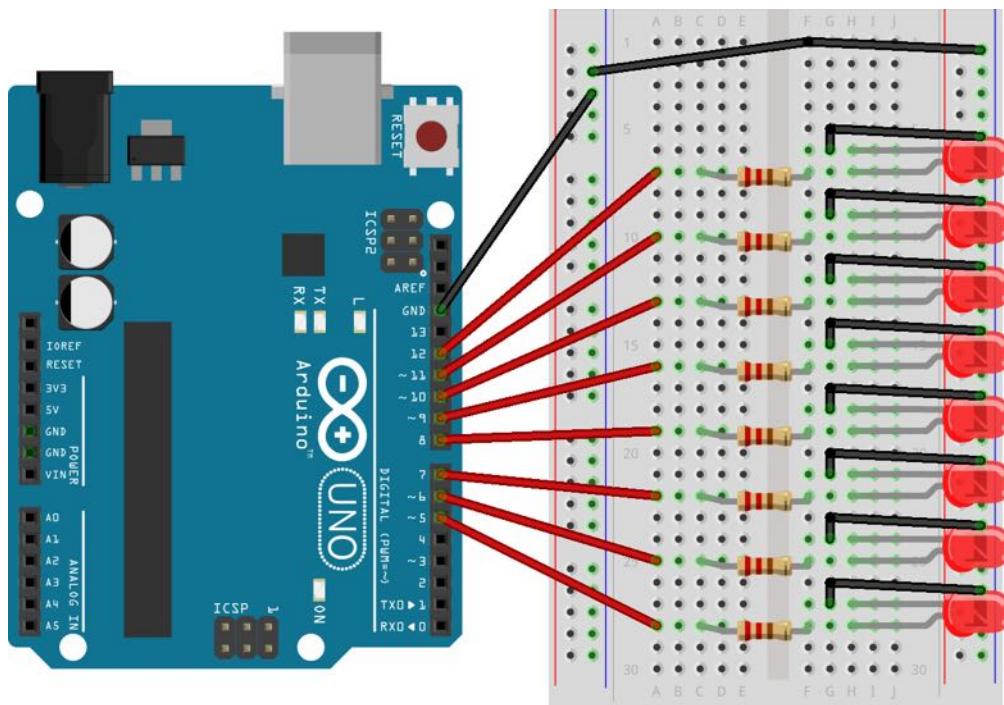
- ① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do programování. V opačném případě obvod musíte opět složit podle přiloženého schématu.



### DEJTE SI POZOR

- Pozor si dejte na to, jak zapojujete LED diody. Delší vývod musí být připojen přes rezistor k pinu. Kratší vývod je připojen na zem (pin GND).
- Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá černá, zlatá.
- Všimněte si, jak je zapojen vodič zemnění. Pro přehlednost je veden na druhou stranu kontaktního pole. Následně je zemnění vedeno ke každé LED diodě zvlášť (černý vodič).

- ② Otevřete programový kód z minulé hodiny, kde je definována vlastní funkce pro jezdící světlo.



## DEKLARACE POLÍ

Deklarace polí lze provádět několika způsoby:

```
1 int myInts[6];
2 int myPins[] = {2, 4, 8, 3, 6};
3 int mySendVals[5] = {2, 4, -8, 3, 2};
4 char message[5] = "hello";
5 char message[5] = {"h", "e", "l", "l", "o"};
```

- ① **myInt[6]** – deklarace pole bez jeho inicializace s šesti prvky.
- ② **myPins[]** – deklarace pole bez uvedení jeho konkrétní velikosti, kompilátor nejdříve spočítá prvky a pak se vytvoří pole o odpovídající velikosti.
- ③ **mySendVals[5]** – deklarace pole s jeho přesným počtem prvků.
- ④ **message[5]** – zvláštním typem pole je pole znaků, který umožňuje specifický způsob přiřazení hodnoty.

## PŘÍSTUP K POLI

Prvky v poli jsou indexovány vždy od nuly. Tzn. první prvek v poli je na indexu 0. To znamená, že pole o deseti prvcích má poslední prvek s indexem 9.

```
1 // přístup k prvkům od nultého indexu
2 mySendVals[0] == 2; // index 0 tj. první prvek
3 mySendVals[1] == 4; // index 1 tj. druhý prvek
4 mySendVals[2] == -8; // atd.
5
6 // přidělení hodnoty pole
7 mySendVals[0] = 12; // na index 0 bude přidělena hodnota 12
8
9 // získání hodnoty z pole
10 x = mySendVals[4]; // do x se uloží hodnota z indexu 4 tj. 2
```

## OTÁZKY PRO VÁS

- ➔ Když se podíváte na otevřený programový kód, jak byste v něm využili pole?
- ➔ V čem byste spatřovali výhodu ve využití pole?



Pokud nedokážete odpovědět na uvedené otázky, vyzkoušejte funkčnost následujícího programu.

```
1 int pinArray[] = {5, 6, 7};  
2  
3 void setup() {  
4     pinMode(pinArray[0], OUTPUT);      // dioda 1  
5     pinMode(pinArray[1], OUTPUT);      // dioda 2  
6     pinMode(pinArray[2], OUTPUT);      // dioda 3  
7 }  
8  
9 void loop() {  
10    changeLED (pinArray[0]);  
11    changeLED (pinArray[1]);  
12    changeLED (pinArray[2]);  
13 }  
14  
15 void changeLED(int pin) {  
16     digitalWrite(pin, HIGH);        // rozsvícení diody  
17     delay(50);  
18     digitalWrite(pin, LOW);        // zhasnutí diod  
19     delay(50);  
20 }
```

## ÚKOLY PRO VÁS

- ➔ A) Upravte otevřený program s vlastní funkcí tak, aby čísla pinů byla nahrazena odkazem na prvky pole.
- ➔ B) Změňte směr běžícího světla z opačné strany.
- ➔ C) Upravte pořadí prvků v poli tak, aby se diody rozsvěcovali od středu ke krajům.



# ŘEŠENÍ ÚLOH

Úkol A)

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};
2
3 void setup() {
4     pinMode(pinArray[0], OUTPUT);      // dioda 1
5     pinMode(pinArray[1], OUTPUT);      // dioda 2
6     pinMode(pinArray[2], OUTPUT);      // dioda 3
7     pinMode(pinArray[3], OUTPUT);      // dioda 4
8     pinMode(pinArray[4], OUTPUT);      // dioda 5
9     pinMode(pinArray[5], OUTPUT);      // dioda 6
10    pinMode(pinArray[6], OUTPUT);      // dioda 7
11    pinMode(pinArray[7], OUTPUT);      // dioda 8
12 }
13
14 void loop() {
15     changeLED (pinArray[0]);
16     changeLED (pinArray[1]);
17     changeLED (pinArray[2]);
18     changeLED (pinArray[3]);
19     changeLED (pinArray[4]);
20     changeLED (pinArray[5]);
21     changeLED (pinArray[6]);
22     changeLED (pinArray[7]);
23 }
24
25 void changeLED(int pin) {
26     digitalWrite(pin, HIGH);        // rozsvícení diody
27     delay(50);
28     digitalWrite(pin, LOW);        // zhasnutí diody
29     delay(50);
30 }
```

### Úkol B)

Řešení tohoto úkolu je velmi jednoduché. Stačí změnit pořadí čísel pinů v poli **pinArray[]**.

```
1 // Puvodni poradi cisel pinu
2 // int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};
3
4 int pinArray[] = {12, 11, 10, 9, 8, 7 , 6, 5};
5
6 void setup() {
7     pinMode(pinArray[0], OUTPUT);      // dioda 1
8     pinMode(pinArray[1], OUTPUT);      // dioda 2
9     pinMode(pinArray[2], OUTPUT);      // dioda 3
10    pinMode(pinArray[3], OUTPUT);      // dioda 4
11    pinMode(pinArray[4], OUTPUT);      // dioda 5
12    pinMode(pinArray[5], OUTPUT);      // dioda 6
13    pinMode(pinArray[6], OUTPUT);      // dioda 7
14    pinMode(pinArray[7], OUTPUT);      // dioda 8
15 }
16
17 void loop() {
18     changeLED (pinArray[0]);
19     changeLED (pinArray[1]);
20     changeLED (pinArray[2]);
21     changeLED (pinArray[3]);
22     changeLED (pinArray[4]);
23     changeLED (pinArray[5]);
24     changeLED (pinArray[6]);
25     changeLED (pinArray[7]);
26 }
27
28 void changeLED(int pin) {
29     digitalWrite(pin, HIGH);        // rozsviceni diody
30     delay(50);
31     digitalWrite(pin, LOW);        // zhasnuti diody
32     delay(50);
33 }
```

### Úkol C)

Řešení může spočívat v úpravě funkce **changeLED** tak, aby měla dva vstupní parametry.

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};
2
3 void setup() {
4     pinMode(pinArray[0], OUTPUT);      // dioda 1
5     pinMode(pinArray[1], OUTPUT);      // dioda 2
6     pinMode(pinArray[2], OUTPUT);      // dioda 3
7     pinMode(pinArray[3], OUTPUT);      // dioda 4
8     pinMode(pinArray[4], OUTPUT);      // dioda 5
9     pinMode(pinArray[5], OUTPUT);      // dioda 6
10    pinMode(pinArray[6], OUTPUT);      // dioda 7
11    pinMode(pinArray[7], OUTPUT);      // dioda 8
12 }
13
14 void loop() {
15     changeLED (pinArray[3], pinArray[4]);
16     changeLED (pinArray[2], pinArray[5]);
17     changeLED (pinArray[1], pinArray[6]);
18     changeLED (pinArray[0], pinArray[7]);
19 }
20
21 void changeLED(int pin, int pin1) {
22     digitalWrite(pin, HIGH);        // rozsvícení diody
23     digitalWrite(pin1, HIGH);
24     delay(50);
25     digitalWrite(pin, LOW);        // zhasnutí diod
26     digitalWrite(pin1, LOW);
27     delay(50);
28 }
```

# PRŮVODCE HODINOU III



Žáci využijí obvod z přechozí hodiny s několika LED diodami a rezistory. Pro naprogramování tohoto obvodu ovšem využijí novou programovací strukturu – cyklus **for**. Využití této nové programovací struktury má studentům ukázat, jak se celý kód zjednoduší.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 8x LED dioda, 8x rezistor 220Ω, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 2, která je ke stažení na GitHub
- ⑤ Pracovní listy pro studenty (ke stažení na GitHub).

## 1. KROK ⏱ 10 minut

V tomto kroku žákům ukažte prostřednictvím dataprojektoru a prezentace základní strukturu příkazu cyklu **for**.

Důležité je poukázat především na důležitost přírůstku. Přírůstek v cyklu **for** může hodnotu inkrementovat (++) ale také dekrementovat (--).



Pokud žáci mají obvod složený z minulé hodiny, mohou začít hned programovat.

V opačném případě, v rámci opakování se jej musí složit, podle přiloženého schématu zapojení.



## 2. KROK 10 minut

V krátkosti žáky seznamte se sériovým monitorem. Vysvětlení a ukázka použití mají žáci v pracovních listech a lze jej také ukázat prostřednictvím dataprojektoru a přiložené prezentace.



### ÚKOL PRO STUDENTY

- A) Uvedený příklad v popisu syntaxe (praktická ukázka), upravte tak, aby se v sériovém monitoru vypisovala hodnota proměnné i.



### ZEPTEJTE SE STUDENTŮ

- Podařilo se vám vypisovat hodnotu proměnné i?

Studenti si zopakují/naučí práci se sériovým monitorem, který otevřou kliknutím na ikonu

- V jakém rozmezí se zobrazovaly proměnné i?

Zobrazí se hodnoty 0-254.

- Které části programového kódu by se hodilo využít v cyklu for, aby se kód zjednodušil?

Určitě by to mohla být část, kde se definují piny pomocí funkce pinMode() a dále při volání vlastní funkce changeLed().

## 3. KROK 10 minut

Studenti nyní mohou řešit samostatné úkoly, které spočívají pouze v inovaci existujícího kódu.



### ÚKOLY PRO STUDENTY

- B) Předchozí úkol, ve kterém jste čísla pinů nahradili prvky pole, upravte tak, abyste použili příkaz cyklu for a světlo diod probíhalo z jedné strany na druhou, neustále dokola.
- C) Upravte programový kód tak, aby se běžící světlo pohybovalo z jedné strany na druhou a zpět.



#### MOŽNÝ NÁPAD

- Studenti si mohou podle přiložené šablony doma připravit jednoduchý stojánek na diody.
- Pak již stačí diody umístit do tohoto stojánku a pouhou změnou pořadí pinů definovaných v poli vytvářet světelné animace.

# PRACOVNÍ LIST – CYKLUS FOR

V TÉTO ČÁSTI BUDETE POKRAČOVAT V PROGRAMOVÁNÍ SOUSTAVY LED DIOD.  
NAUČÍTE SE OPTIMALIZOVAT PROGRAMOVÝ KÓD POMOCÍ CYKLU FOR.

## CO SE NAUČÍTE

- ① Programovat cyklus **for**.
- ② Procházení pole pomocí cyklu **for**.
- ③ Pracovat se sériovým monitorem.



## CO BUDETE POTŘEBOVAT

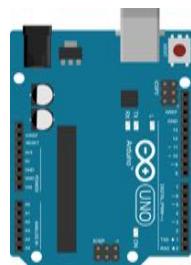
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8x.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



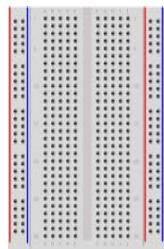
RGB led – 8 kusů



Rezistor  $220\Omega$  – 8 kusů



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

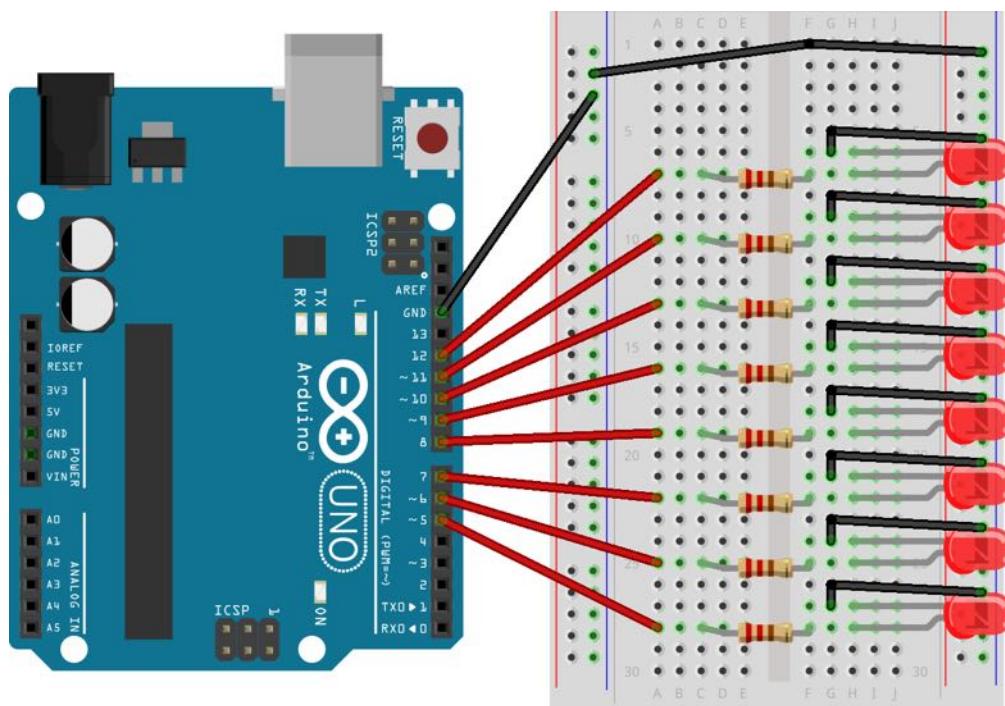
- ① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do programování. V opačném případě obvod musíte opět složit podle přiloženého schématu.



### DEJTE SI POZOR

- Pozor si dejte na to, jak zapojujete LED diody. Delší vývod musí být připojen přes rezistor k pinu. Kratší vývod je připojen na zem (pin GND).
- Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá černá, zlatá.
- Všimněte si, jak je zapojen vodič zemnění. Pro přehlednost je veden na druhou stranu kontaktního pole. Následně je zemnění vedeno ke každé LED diodě zvlášť (černý vodič).

- ② Otevřete programový kód z minulé hodiny, kde jste pracovali s polem.



## CYKLUS FOR

Cyklus **for** se používá k opakování bloku příkazů, uzavřených do složených závorek. Využívá čítače inkrementů (přírůstků), který se používá pro ukončení průchodu cyklu. Cyklus **for** je vhodný pro jakékoli opakující se operace a je často používán v kombinaci s **poli**, pro jejich průchod.

## SYNTAXE

```
1  for (inicIALIZACE; podmÍNKA; pÍRÚSTEK) {  
2      // blok příkazů  
3  }  
4  
5  // praktická ukázka  
6  void setup()  
7  {  
8      for (int i=0; i < 255; i++){  
9          analogWrite(PWMpin, i);  
10         delay(10);  
11     }  
12 }
```

V první řadě nastane **inicIALIZACE**, a to minimálně jednou. V každém průchodu je testována **podmÍNKA**. Pokud podmínka nabude hodnoty **True**, provede se blok příkazů a zvětší se **pÍRÚSTEK**. Podmínka je opět testována. Když podmínka nabude hodnoty **False**, smyčka se ukončí.



Uvedená položka **pÍRÚSTEK** se také nazývá složeným operátorem. Zkracuje nám zápis operací a v případě cyklu **for** hodnotu proměnné zvyšují **i++** a nebo snižují **i--**.

## SÉRIOVÝ MONITOR

Sériový monitor se využívá při čtení informací v textové podobě. Souvisí se sériovou komunikací, kterou můžeme využít například pokud chceme získávat nějaké hodnoty z Arduina nebo naopak je do něj posílat.



### **Serial.begin(9600)**

Funkce pro zahájení sériové komunikace. Zpravidla ji stačí zavolat v části **setup()**. Parametrem je rychlosť komunikace, ktorá odpovedá počtu prenosov za sekundu.

### **Serial.println(hodnota)**

Funkce slouží k odeslání hodnoty z Arduina do počítače nebo jiného zařízení. Nám poslouží také k výpisu hodnot v sériovém monitoru.

Sériový monitor si spusťte kliknutím v IDE Arduino na ikonu



### **ÚKOL PRO VÁS**

A) Výše uvedený příklad v popisu **syntaxe** (praktická ukázka), upravte tak, aby se v sériovém monitoru vypisovala hodnota proměnné *i*.



### **OTÁZKA PRO VÁS**

- ➔ Podařilo se vám vypisovat hodnotu proměnné *i*?
- ➔ V jakém rozmezí se zobrazovaly proměnné *i*?
- ➔ Ve které části programového kódu s polí by se hodilo využít cyklu for, aby se kód zjednodušil?



### **ÚKOLY PRO VÁS**

- ➔ B) Předchozí úkol, ve kterém jste čísla pinů nahradili prvky pole, upravte tak, abyste použili příkaz cyklu for a světlo diod probíhalo z jedné strany na druhou, neustále dokola.
- ➔ C) Upravte programový kód tak, aby se běžící světlo pohybovalo z jedné strany na druhou a zpět.

## ŘEŠENÍ ÚLOH

### Úkol A)

Řešení je velmi jednoduchý a slouží pouze k opakování.

```
1 void setup() {
2     Serial.begin(9600);
3     for (int i=0; i<255; i++) {
4         Serial.println(i);
5         delay(10);
6     }
7 }
8
9 void loop() {
10 }
```

### Úkol B)

Všimněte si, že byla také deklarována proměnná **timer**, pro definici pauzy programu.

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};
2 int count = 0;
3 int timer = 50;
4
5 void setup() {
6     for (count=0;count<8;count++) {
7         pinMode(pinArray[count], OUTPUT);
8     }
9 }
10
11 void loop() {
12     for (count=0; count<8; count++) {
13         changeLED(pinArray[count]);
14     }
15 }
16
17 void changeLED(int pin) {
18     digitalWrite(pin, HIGH);
19     delay(timer);
20     digitalWrite(pin, LOW);
21     delay(timer);
22 }
```

Úkol C)

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};
2 int count = 0;
3 int timer = 50;
4
5 void setup() {
6     for (count=0;count<8;count++) {
7         pinMode(pinArray[count], OUTPUT);
8     }
9 }
10
11 void loop() {
12     for (count=0; count<8; count++){ // jeden směr
13         changeLED(pinArray[count]);
14     }
15     for (count=7; count>=0; count--){ // druhý směr
16         changeLED(pinArray[count]);
17     }
18 }
19
20 void changeLED(int pin) {
21     digitalWrite(pin, HIGH);
22     delay(timer);
23     digitalWrite(pin, LOW);
24     delay(timer);
25 }
```

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU, KTERÝ JE ZAMĚŘEN NA POUŽÍVÁNÍ POLÍ A CYKLU FOR. ŘEŠENÉ ÚKOLY ZOHLEDŇUJÍ NOVĚ PROBRANÉ UČIVO A STAVÍ NA PŘEDCHOZÍCH ZNALOSTECH.

## OBSAH PRŮVODCE

- ① Deklarace a používání vlastních funkcí.
- ② Definice polí v Arduinu.
- ③ Použití příkazu cyklu **for**.
- ④ Programové kódy pro vysvětlení používání polí a cyklu.
- ⑤ Řešené problémy při zapojení sestavy LED diod.
- ⑥ Technická část pro závěrečný projekt – stojan na diody.
- ⑦ Vysvětlení řešení samostatných úkolů.

## VLASTNÍ FUNKCE V ARDUINO

Všimněme si, že při rozsvěcování a zhasínání LED diody neustále vykonáváme stejný postup příkazů. Pro blikání jediné diody musíme napsat minimálně čtyři řádky kódu. Nyní si představme, že diod máme devět a každou chceme ovládat zvlášť. To se nám program o dost řádků rozroste. A v tomto případě nám mohou pomoci **funkce**.



V počítačové vědě je **funkce** podprogram (nazývaný také jako postup, metoda nebo rutina), tj. část kódu v rámci většího programu, který provádí konkrétní úkol a je relativně nezávislý na zbývajícím kódu.

### PROČ MÁME FUNKCE?

Rozdelení programového kódu do funkcí umožňuje programátorům vytvářet modulární části kódu, které provádějí definovaný úkol. Následně se vrací do místa kódu, kde byla funkce volána. Typicky funkce použijeme, pokud chceme například provést stejnou akci v programu několikrát.

- Funkce programátorovi pomáhají s organizací programového kódu.
- Funkce kodifikují jednu akci na jednom místě, takže stačí funkci jednou odladit a libovolně mnohokrát používat.
- Tím se snižuje chybovost při případné modifikaci funkce.
- Funkce dělají kód jednodušší a kompaktnější, protože opakující části kódu jsou používány pouze ve funkci.
- Funkce usnadňují opětovné používání kódu v ostatních programech tím, že se stávají více modulární. Pěkným vedlejším efektem je větší čitelnost kódu.

### JAK PSÁT FUNKCE?

V programovém kódu pro Arduino jsou dvě základní a povinné funkce **setup()** a **loop()**. Další definované funkce musí být mimo těla těchto funkcí. Struktura zápisu funkce by mohla vypadat následovně:

```
1 navratovytyp nazevFunkce(argumenty) {  
2     // tělo funkce  
3     return hodnota; // v návratovém typu  
4 }
```

**navratovytyp** – specifikuje v jakém datovém typu bude vrácen výsledek dané funkce. Pokud budeme chtít vykonat pouze nějakou úlohu, která nebude vracet výsledek v podobě např. výpočtu nebo textu, tak si vystačíme s definicí pomocí slova **void**.

**nazevFunkce** – název funkce by měl uvádět, co bude funkce dělat. Název by měl být jednoduchý a přitom popisný, aby bylo na první pohled jasné, co funkce opravdu dělá.

// tělo funkce – obsahuje konkrétní programový kód, který se má vykonat. Syntaxe kódu se řídí stejnými pravidly jako části kódu mimo funkci.

**argumenty** – se skládají z datového typu a jména. Např. bychom mohli ve funkci pracovat se dvěma čísly, takže argumenty by vypadaly: **int x, int y**.

**return navratovytyp** – vrací například výsledek výpočtu, se kterým může program dále pracovat mimo definovanou funkci.

Konkrétní podoba funkce by mohla vypadat následovně:

```
1 int soucet(int x, int y) {  
2     vysledek = x + y; // tělo funkce  
3     return vysledek;  
4 }
```

## POLE V ARDUINU

Proměnou si můžeme představit jako konkrétní knihu o konkrétním názvu. Pole si lze představit jako knihovnu, v které máme knihy narovnány do poliček a každá kniha je očíslovaná. Konkrétní knihu vybereme zvolením daného čísla.



Pole je sada proměnných, které jsou přístupné s indexovým číslem. Pole v programovacím jazyce C, na němž je založeno Arduino, mohou být komplikovaná, ale použití jednoduchých polí je poměrně snadné.

### VYTVÁŘENÍ POLE

Deklarace polí lze provádět několika způsoby:

```
1 int myInts[6];
2 int myPins[] = {2, 4, 8, 3, 6};
3 int mySendVals[5] = {2, 4, -8, 3, 2};
4 char message[5] = "hello";
```

- ① **myInt[6]** – deklarace pole bez jeho inicializace.
- ② **myPins[]** – deklarace pole bez uvedení jeho konkrétní velikosti, kompilátor nejdříve spočítá prvky a pak pole vytvoří o odpovídající velikosti.
- ③ **mySendVals[6]** – deklarace pole s jeho přesným počtem prvků.
- ④ **message[6]** – zvláštním typem pole je pole znaků, který umožňuje specifický způsob přiřazení hodnoty.

### PŘÍSTUP K POLI

Prvky v poli jsou indexovány vždy od nuly. Tzn. první prvek v poli je na indexu 0. To znamená, že pole o deseti prvcích má poslední prvek s indexem 9.

```
1 // přístup k prvkům od nultého indexu
2 mySendVals[0] == 2; // index 0 tj. první prvek
3 mySendVals[1] == 4; // index 1 tj. druhý prvek
4 mySendVals[2] == -8; // atd.
5
6 // přidělení hodnoty pole
7 mySendVals[0] = 12; // na index 0 bude přidělena hodnota 12
8
9 // získání hodnoty z pole
10 x = mySendVals[4]; // do x se uloží hodnota z indexu 4 tj. 2
```

## CYKLUS FOR

Cyklus **for** se používá k opakování bloku příkazů, uzavřených do složených závorek. Využívá čítače inkrementů (přírůstků), který se používá pro ukončení průchodu cyklu. Cyklus **for** je vhodný pro jakékoliv opakující se operace a je často používán v kombinaci s **poli** pro jejich průchod.

### SYNTAXE

```
1  for (inicIALIZACE; PODMÍNKA; PŘÍRŮSTEK) {  
2      // blok příkazů  
3  }  
4  
5 // praktická ukázka  
6 void loop()  
{  
7     for (int i=0; i <= 255; i++){  
8         analogWrite(PWMpin, i);  
9         delay(10);  
10    }  
11}  
12 }
```

Nejprve nastane **inicIALIZACE**, a to minimálně jednou. V každém průchodu je testována **PODMÍNKA**. Pokud podmínka nabude hodnoty **True**, provede se blok příkazů a zvětší se **PŘÍRŮSTEK**. Podmínka je opět testována. Když podmínka nabude hodnoty **False**, smyčka se ukončí.

## SÉRIOVÁ KOMUNIKACE

Pokud chceme od Arduina získávat nějaké hodnoty, nebo mu je posílat, přichází na řadu právě sériová komunikace. Ta je také důležitým nástrojem při ladění programů. Když něco nefunguje a nevíme proč, může být užitečné si nechat vypisovat hodnotu proměnné, nebo určitý text na libovolném místě programu a tím zjistit co přesně kód provádí. Dá se také využít při komunikaci Arduina a dalších zařízení (jiné Arduino, moduly atd.).

Pro čtení informací v textové podobě na počítači se používá tzv. **Serial monitor**. Ten se spustí kliknutím na ikonu  v IDE Arduina. Po spuštění Serial monitoru se musí ještě nastavit rychlosť komunikace (300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 a 115200).



### Základní funkce sériové komunikace

#### **Serial.begin(9600)**

Funkce pro zahájení sériové komunikace. Zpravidla ji stačí zavolat v části **setup()**. Parametrem je rychlosť komunikace, která odpovídá počtu přenosů za sekundu.

#### **Serial.println(hodnota)**

Funkce slouží k odeslání hodnoty z Arduina do počítače nebo jiného zařízení. Nám poslouží také k výpisu hodnot v sériovém monitoru.

## SYNTAXE

```
1 void setup() {
2     Serial.begin(9600);
3     for (count=0; count<255; count++) {
4         Serial.println(count);
5         delay(10);
6     }
7 }
8
9 void loop() {
10}
11}
```

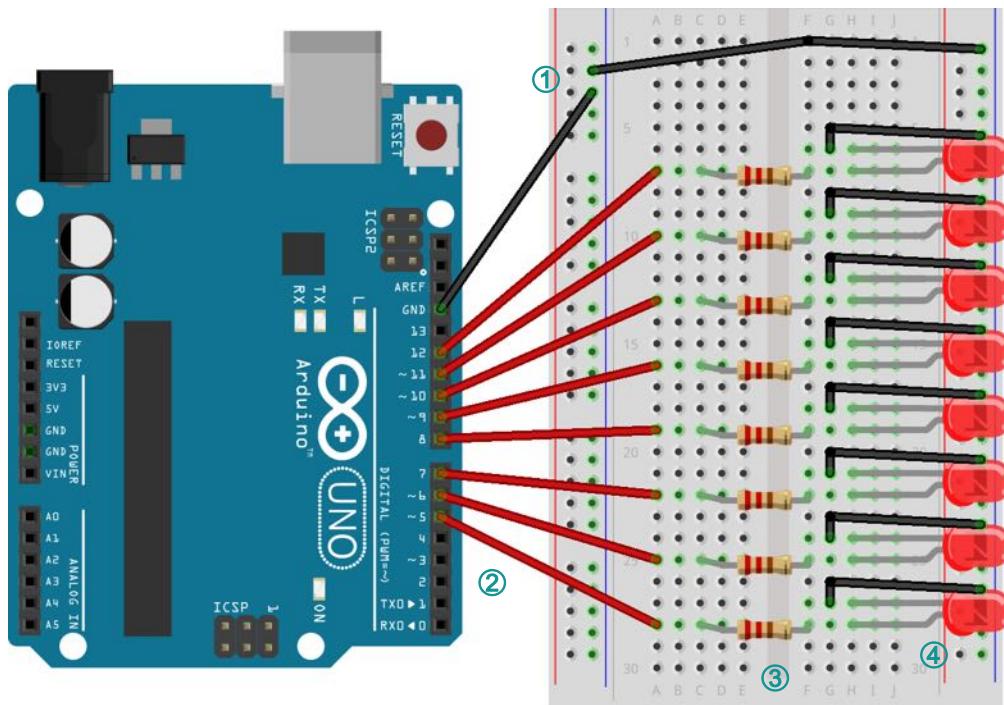
- ① Pokud se bude využívat sériové komunikace, je nutné nastavit rychlosť prenosu.
- ② Výpis hodnôt v sériovom monitoru. V tomto prípade se bude vypisovať hodnota premennej **i**. Každá hodnota bude na novom riadku, ak bychom chceli hodnoty všetkých vedľa seba v jednom riadku, použijeme funkciu **Serial.print(count)**.

## SVĚTELNÁ ANIMACE

Vytvoříme nástroj na světelnou animaci, který vás naučí řídit větší počet pinů desky Arduino. Nejprve využijeme kontaktní pole, kde si otestujeme zapojení všech diod. Následně si vytvoříte kartonovou konstrukci, do které LED diody umístíte a připojíte pomocí vodičů.

## ZAPOJENÍ OBVODU

Zapojení obvodu vychází z přechozího příkladu zapojení LED diody s tím rozdílem, že zde bude zapojeno osm diod. Každá z osmi diod bude ovládána nezávisle.



Obr. 1 - Zapojení osmi LED diod

- ① Vodič zemnění z desky Arduino zapojte do kontaktní desky k **modré** čáře. Tím se zpřehlední celé zapojení.
- ② Najděte 8 rezistorů o hodnotě  $220\Omega$  (dva oranžové proužky, hnědý a zlatý). Je důležité, aby tyto rezistory byly zapojeny do kontaktní desky tak, aby propojovaly její obě

poloviny. Pokud by tomu tak nebylo, došlo by ke zkratování. Rezistory zajistí snížení proudu, aby nedošlo ke zničení LED diod.

- ③ Zapojte 8 LED diod do kontaktní desky tak, jak je uvedeno na obrázku Obr. 1 - Zapojení osmi LED diod. Delší vývody každé LED diody jsou připojeny k rezistorům. Kratší vývod je propojen s **uzemněním**.
- ④ Druhá strana rezistorů je zapojena do digitálních pinů 5, 6, 7, 8, 9, 10, 11 a 12.

## PROGRAMOVÝ KÓD

Při psaní kódu pro všechny příklady bude použité jediné zapojení uvedené na obrázku Obr. 1 - Zapojení osmi LED diod.



Příklady na sebe navazují, takže pro jejich inovace stačí provádět pouze dílčí úpravy programového kódu. Není nutné vždy psát celý program od začátku.

## ZÁKLADNÍ PŘÍKLAD – POSTUPNÉ ROZSVĚCOVÁNÍ LED DIOD

Ukázka „hrubého“ programového kódu pro postupné rozsvěcování a zhasnání LED diod.

Tím se vytvoří efekt „běžícího světla“.

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     digitalWrite(5, HIGH);    // rozsvícení diody 1
14     delay(50);
15     digitalWrite(5, LOW);     // zhasnutí diod
16     delay(50);
17     digitalWrite(6, HIGH);    // rozsvícení diody 2
18     delay(50);
19     digitalWrite(6, LOW);     // zhasnutí diod
20     delay(50);
21     digitalWrite(7, HIGH);    // rozsvícení diody 3
22     delay(50);
23     digitalWrite(7, LOW);     // zhasnutí diod
24     delay(50);
25     digitalWrite(8, HIGH);    // rozsvícení diody 4
26     delay(50);
27     digitalWrite(8, LOW);     // zhasnutí diod
28     delay(50);
29     digitalWrite(9, HIGH);    // rozsvícení diody 5
30     delay(50);
31     digitalWrite(9, LOW);     // zhasnutí diod
32     delay(50);
33     digitalWrite(10, HIGH);   // rozsvícení diody 6
34     delay(50);
35     digitalWrite(10, LOW);    // zhasnutí diod
36     delay(50);
37     digitalWrite(11, HIGH);   // rozsvícení diody 7
38     delay(50);
39     digitalWrite(11, LOW);    // zhasnutí diod
40     delay(50);
41 }
```

①

②

```
42   digitalWrite(12, HIGH);      // rozsvícení diody 8
43   delay(50);
44   digitalWrite(12, LOW);       // zhasnutí diod
45   delay(50);
46 }
```

⑤ Pro každou LED diodu je definován výstupní pin pomocí funkce **pinMode()**.

V příkladu jsou definovány piny **5, 6, 7, 8, 9, 10, 11, 12**.

⑥ Pro rozsvícení diody se využívá funkce **digitalWrite()** s parametrem **číslo pinu** a hodnotu **HIGH**.

Pro zhasnutí diody se volá také funkce **digitalWrite()** s parametrem **číslo pinu**, ale s hodnotou **LOW**. Aby bylo zřetelné postupné rozsvěcování a zhasínání, je použita funkce pro pozastavení běhu programu **delay()**.



(Př. 1) Změňte předchozí příklad tak, aby se nejdříve všechny LED diody postupně rozsvítily a následně se od konce opět postupně zhasínaly.



### NESVÍTÍ DIODY

**Zapojení LED diody** – zkontrolujte zapojení LED diod v kontaktním poli. Ujistěte se, že jednotlivé diody mají anodu a katodu správně zapojenou. **Anoda** musí být zapojena k rezistoru a následně do digitálního pinu na desce Arduino. **Katoda** je zapojena na zemnění desky.

**Zapojení v desce** – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů desky Arduino.

**Rezistory** – zkontrolujte, zda jste použili správné rezistory.

### NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

## ZÁKLADNÍ PŘÍKLAD – VLASTNÍ FUNKCE

Příklad pro rozsvícení a zhasnání LED diod je jednoduchý, ale příliš dlouhý a tím i málo přehledný. Provedeme tedy jeho inovaci s využitím vlastní funkce.

Nadefinujeme funkci **changeLED(int pin)**, která má jediný parametr. Tím je číslo pinu, na který je dioda připojena.

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     changeLED (5);
14     changeLED (6);
15     changeLED (7);
16     changeLED (8);
17     changeLED (9);
18     changeLED (10);
19     changeLED (11);
20     changeLED (12);
21 }
22
23 void changeLED(int pin) {
24     digitalWrite(pin, HIGH);    // rozsvícení diody 1
25     delay(50);
26     digitalWrite(pin, LOW);    // zhasnutí diod
27     delay(50);
28 }
```

②

①

- ① Deklarace funkce **changeLED()**, která rozsvěcí a zhasíná konkrétní diodu, definovanou číslem digitálního pinu jako jediného parametru funkce.
- ② Funkce **changeLED()** se volá ve smyčce funkce **loop()**.



(Př. 2) Změňte předchozí příklad s využitím funkce tak, aby se diody nejdříve postupně všechny rozsvítily a následně postupně zhasínaly. Návod vám může být, že se bude muset změnit funkce **changeLED()**.

## ZÁKLADNÍ PŘÍKLAD – POLE A CYKLUS FOR

Následující příklad využívá kombinaci programových struktur jako jsou **funkce**, **pole** a cyklus **for**. Program tak získá na přehlednosti a větší flexibilitě. Příklad ukazuje jezdící světlo z jedné strany na druhou a zpět.

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12};          | ①
2 int count = 0;                                         | ②
3 int timer = 50;                                        | ③
4
5 void setup() {
6     for (count=0; count<8; count++) {
7         pinMode(pinArray[count], OUTPUT);                | ④
8     }
9 }
10
11 void loop() {
12     for (count=0; count<8; count++) {
13         changeLED(pinArray[count]);                   | ⑤
14     }
15     for (count=7; count>=0; count--) {
16         changeLED(pinArray[count]);                   | ⑥
17     }
18 }
19
20 void changeLED(int pin) {
21     digitalWrite(pin, HIGH);
22     delay(timer);
23     digitalWrite(pin, LOW);
24     delay(timer);
25 }
```

- ① Definice pole **pinArray[]**, které obsahuje čísla pinů, na které jsou připojeny LED diody. Výhodou této konstrukce je možnost jednoduchého rozšíření o další LED diody.
- ② Definice proměnné **count**, která je iterátorem v cyklech **for**.
- ③ Definice proměnné **timer**, která je parametrem funkce **delay**, která přerušuje běh programu proto, aby bylo vidět probliknutí LED diody.
- ④ Pomocí cyklu **for** jsou definovány piny jako výstupy desky Arduino. Všimněte si, že proměnná **count** začíná hodnotou **0** a cyklus bude probíhat, dokud její hodnota bude

menší než **8**. Proč? Připomeňme si, že prvky pole jsou indexovány od nuly a zapojených diod máme osm. Takže od 0 do 7 je to právě hodnota 8.

- ⑤ První cyklus **for** ve funkci **loop()** rozsvěcí LED diody ve vzestupném pořadí. V každém kroku se volá funkce **changeLED()** s jediným parametrem, kterým je číslo pinu, na který je konkrétní dioda připojena.
- ⑥ Druhý cyklus ve funkci **loop()** určuje sestupné pořadí rozsvěcení. Všimněte si proměnné **count**, která se v každém kroku zmenšuje díky dekrementačnímu operátoru **--**.
- ⑦ Funkce **changeLED()** rozsvěcí a zhasíná konkrétní diodu připojenou na pin desky Arduino, který je funkci předáván jako parametr.



(Př. 3) Vytvořte libovolnou světelnou animaci. Můžete využít předchozí příklad a změnit pořadí pinů v poli, přidat další kombinaci pořadí pinů.



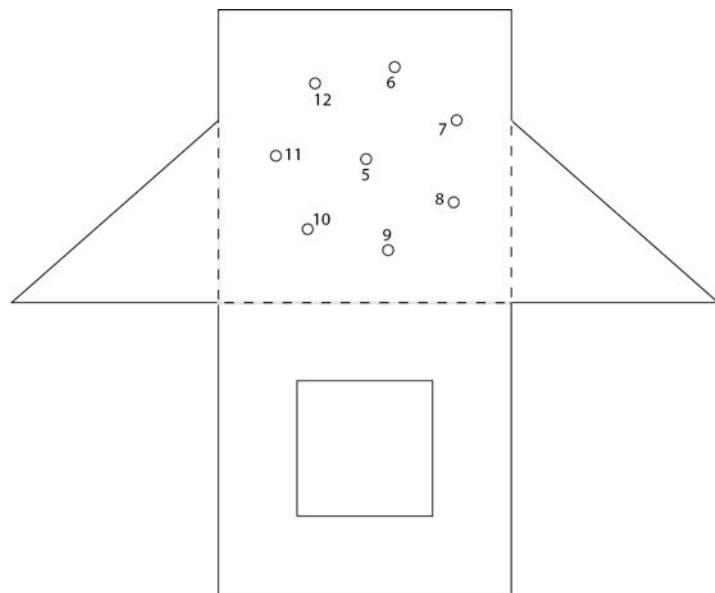
Pokud budete mít připravenou a funkční programovou část, můžete vytvořit stojánek pro LED diody. Tím animace bude efektnější. Návod, je uveden v následující kapitole.

## STOJAN PRO LED DIODY

Pro vytvoření stojanu budeme potřebovat: karton (tvrdší papír), nůžky a tavnou pistoli (lepící pásku).

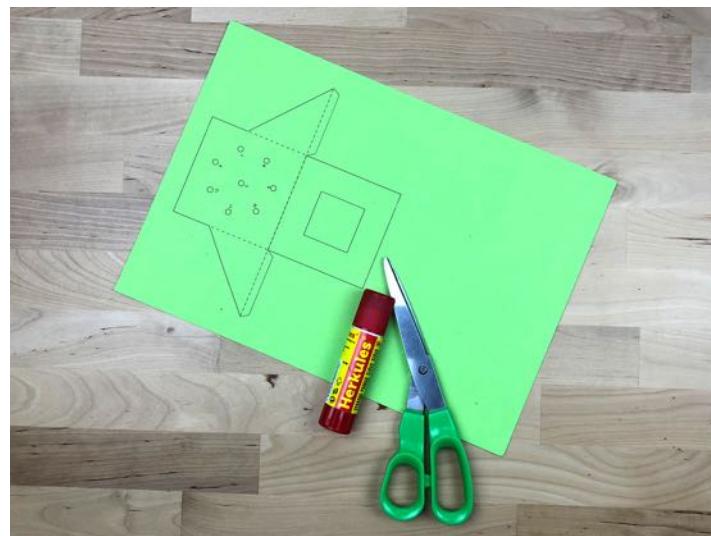
## PAPÍROVÁ KONSTRUKCE

Konstrukce stojánku je velice jednoduchá. Stačí si vytisknout na pevný papír přiloženou šablonu, která je na obrázku Obr. 2 - Šablona stojanu.

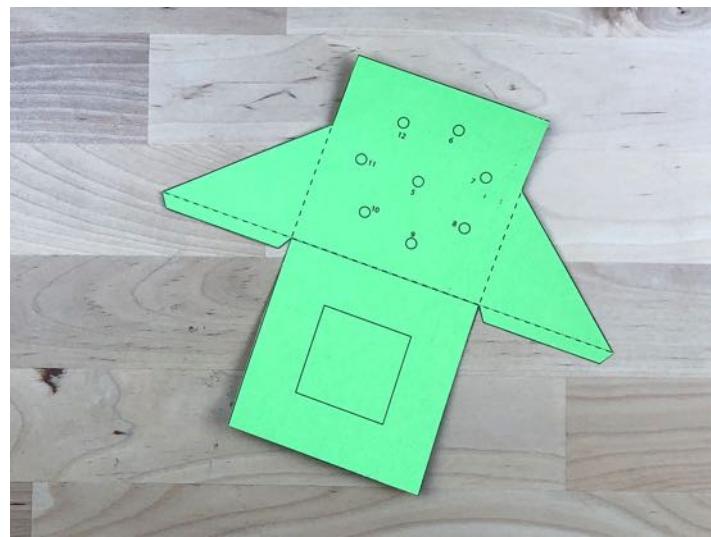


Obr. 2 - Šablona stojanu

Vytištěnou šablonu vystříhněte podle plných čar.



Obr. 3 - Vytištěná šablona



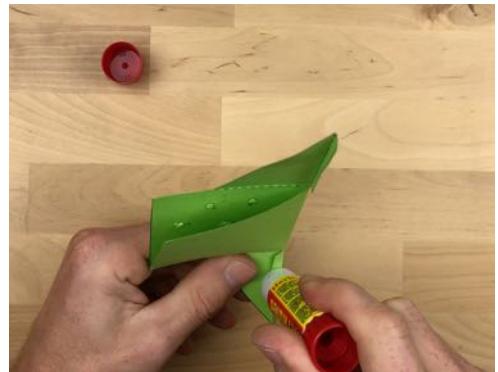
Obr. 4 - Vystřížená šablona

Do naznačených otvorů, které jsou očíslovány podle čísel pinů, do kterých jsou připojeny jednotlivé LED diody, udělejte otvory. Velikost otvorů odpovídá průměru LED diod.

Následně naohýbejte šablonu podle přerušovaných čar. Na lepící hrany naneste lepidlo.

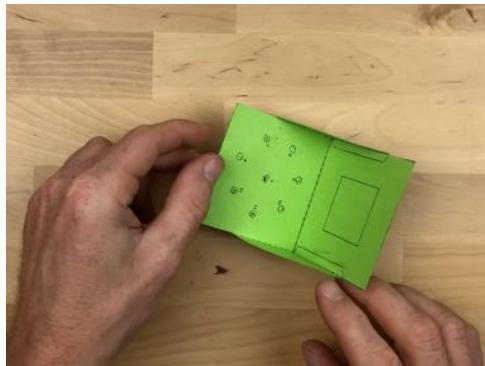


Obr. 5 – Ohýbání lepicích hran

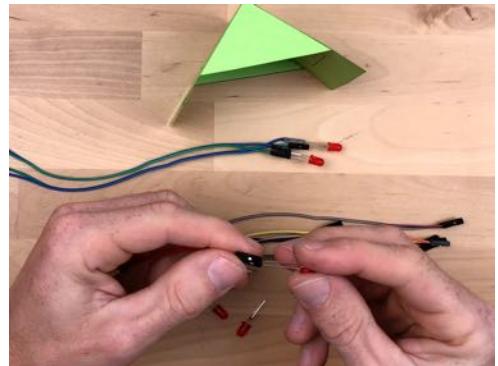


Obr. 6 – Lepení hran

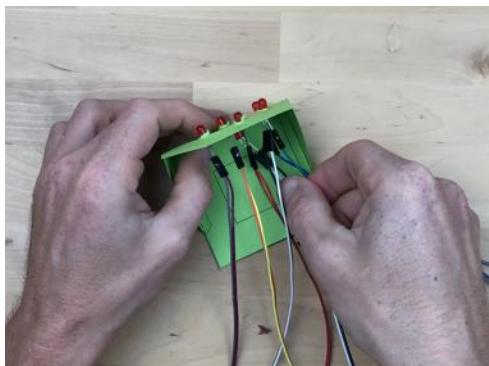
Lepící hrany přilepte k základně stojanu Obr. 7 – Přilepení hran k základně. Než zasunete LED diody do stojanu, připojte k nim vodiče Obr. 8 – Připojení LED diod.



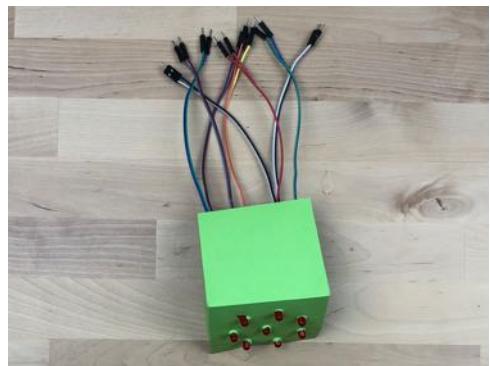
Obr. 7 – Přilepení hran k základně



Obr. 8 – Připojení LED diod



Obr. 9 – Vložení LED diod do stojanu



Obr. 10 – Složený stojan

## ŘEŠENÍ PŘÍKLADŮ

## (PŘ. 1

V tomto příkladu se jedná o přeskládání pořadí volání funkce **digitalWrite()** s hodnotami **HIGH** a **LOW**.

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     digitalWrite(5, HIGH);    // rozsvícení diody 1
14     delay(50);
15     digitalWrite(6, HIGH);    // rozsvícení diody 2
16     delay(50);
17     digitalWrite(7, HIGH);    // rozsvícení diody 3
18     delay(50);
19     digitalWrite(8, HIGH);    // rozsvícení diody 4
20     delay(50);
21     digitalWrite(9, HIGH);    // rozsvícení diody 5
22     delay(50);
23     digitalWrite(10, HIGH);   // rozsvícení diody 6
24     delay(50);
25     digitalWrite(11, HIGH);   // rozsvícení diody 7
26     delay(50);
27     digitalWrite(12, HIGH);   // rozsvícení diody 8
28     delay(50);
29     digitalWrite(5, LOW);     // zhasnutí diody
30     delay(50);
31     digitalWrite(6, LOW);     // zhasnutí diody
32     delay(50);
33     digitalWrite(7, LOW);     // zhasnutí diody
34     delay(50);
35     digitalWrite(8, LOW);     // zhasnutí diody
36     delay(50);
37     digitalWrite(9, LOW);     // zhasnutí diody
38     delay(50);
39     digitalWrite(10, LOW);    // zhasnutí diody
40     delay(50);
41     digitalWrite(11, LOW);    // zhasnutí diody
42     delay(50);
43     digitalWrite(12, LOW);    // zhasnutí diody
44 }
```

```
44 |     digitalWrite(12, LOW);      // zhasnutí diody
45 |     delay(50);
46 |
47 |
48 }
```

## (PŘ. 2

Funkce **changeLED()** musí být rozšířena o další parametr, který bude definovat stav LED diody. Tento stav bude definován **0** nebo **1** (LOW, HIGH).

```
1 void setup() {
2     pinMode(5, OUTPUT);      // dioda 1
3     pinMode(6, OUTPUT);      // dioda 2
4     pinMode(7, OUTPUT);      // dioda 3
5     pinMode(8, OUTPUT);      // dioda 4
6     pinMode(9, OUTPUT);      // dioda 5
7     pinMode(10, OUTPUT);     // dioda 6
8     pinMode(11, OUTPUT);     // dioda 7
9     pinMode(12, OUTPUT);     // dioda 8
10 }
11
12 void loop() {
13     changeLED (5, 1);
14     changeLED (6, 1);
15     changeLED (7, 1);
16     changeLED (8, 1);
17     changeLED (9, 1);
18     changeLED (10, 1);
19     changeLED (11, 1);
20     changeLED (12, 1);
21     changeLED (5, 0);
22     changeLED (6, 0);
23     changeLED (7, 0);
24     changeLED (8, 0);
25     changeLED (9, 0);
26     changeLED (10, 0);
27     changeLED (11, 0);
28     changeLED (12, 0);
29 }
30
31 void changeLED(int pin, int state) {
32     digitalWrite(pin, state); // změna stavu diody
33     delay(50);
34 }
```

### (PŘ. 3

První varianta příkladu ukazuje pouze rozšíření základního pole o další kombinace pořadí pinů. V kódu je ukázka, jak automaticky zjistit délku pole.

```
1 int pinArray[] = {5, 6, 7, 8, 9, 10, 11, 12, 11, 10, 9, 8,
2 7, 6, 5, 6, 5, 7, 5, 8, 5, 9, 5, 10, 5, 11, 5, 12};
3
4 int count = 0;
5 int timer = 50;
6
7 // Zjisteni velikosti pole
8 int sizeArray=sizeof(pinArray)/sizeof(int);
9
10 void setup() {
11     for (count=0; count<8; count++) {
12         pinMode(pinArray[count], OUTPUT);
13     }
14 }
15
16 void loop() {
17     for (count=0; count<sizeArray; count++) {
18         changeLED(pinArray[count]);
19     }
20 }
21
22 void changeLED(int pin) {
23     digitalWrite(pin, HIGH);
24     delay(timer);
25     digitalWrite(pin, LOW);
26     delay(timer);
27 }
```

Druhá ukázka pracuje s vícerozměrným polem, kde jednotlivé kombinace pořadí pinů jsou oddělená pole.

```
1 int pinArray[3][8] = {  
2     {5, 6, 7, 8, 9, 10, 11, 12},  
3     {12, 11, 10, 9, 8, 7, 6, 5},  
4     {8, 7, 6, 5, 9, 10, 11, 12}  
5 };  
6 int count = 0;  
7 int timer = 50;  
8 int i;  
9  
10  
11 void setup() {  
12     for (count=0; count<8; count++) {  
13         pinMode(pinArray[0][count], OUTPUT);  
14     }  
15 }  
16  
17 void loop() {  
18     for (i=0; count<3; i++) {  
19         for (count=0; count<8; count++) {  
20             changeLED(pinArray[i][count]);  
21         }  
22     }  
23 }  
24  
25 void changeLED(int pin) {  
26     digitalWrite(pin, HIGH);  
27     delay(timer);  
28     digitalWrite(pin, LOW);  
29     delay(timer);  
30 }
```

Průchod vícerozměrným polem se realizuje dvěma cykly **for**. První cyklus (**řádek 18**) prochází jednotlivá pole a druhý cyklus prochází samotné prvky pole (**řádek 19**).

## 3. ŘÍZENÍ SERVOMOTORU

DŮLEŽITOU OBLASTÍ ROBOTIKY JSOU POHONY. TATO LEKCE SE BUDE ZABÝVAT ZÁKLADNÍM POUŽITÍM SERVOMOTORŮ. ABY BYLO MOŽNÉ SERVOMOTOR SPRÁVNĚ OVLÁDAT, MUSÍME SI VYSVĚTLIT POUŽITÍ PODMÍNKOVÉHO PŘÍKAZU IF.

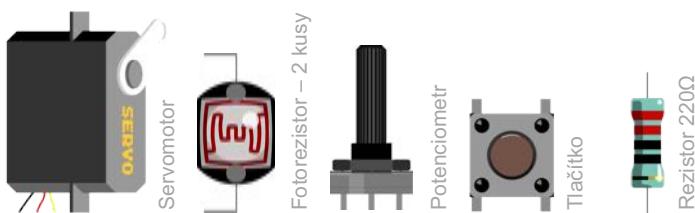
### CÍLE

- ① Základní zapojení servomotoru.
- ② Zapojení potenciometru.
- ③ Zapojení fotorezistorů.
- ④ Použití podmínkového příkazu **if**.
- ⑤ Projekt natáčející se květinou za světlem.

Čas: **2x45 min**

Úroveň: ■■■■■

Vychází z: **2**



POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví jednoduchý obvod, ve kterém bude zapojen servomotor. Na tomto obvodu jim bude vysvětlen princip servomotoru a jeho programování. Tento obvod dále rozšíří o potenciometr a využijí analogových vstupů desky Arduino pro čtení hodnot potenciometru. Součástí jsou jednoduché samostatné úkoly, které pomohou objasnit princip servomotorů.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, servomotor, potenciometr, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 3.
- ⑤ Pracovní listy pro studenty .

## 1. KROK 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude naučit se ovládat servomotor.

### ZEPTEJTE SE STUDENTŮ

#### ➔ K čemu si myslíte, že se servomotory využívají?

Servomotory jsou motory pro pohony, u kterých lze na rozdíl od běžného motoru nastavit přesnou polohu natočení osy.

Používají se tam, kde je potřeba větší síla pro pohyb mechanických prvků: roboti a pohyb jejich ramen, automatické ostření fotoaparátu, posuvy obráběcích strojů, natáčení slunečních panelů atd.



Studenti ať zapojí servomotor podle zobrazeného schématu, které je součástí pracovních listů nebo přiložené prezentace, kterou lze promítat pomocí dataprojektoru.

## KROK 🕒 10 minut

Nyní studentům ukažte prostřednictvím dataprojektoru nebo pracovního listu základní kód, který bude natáčet osu servomotoru na zadaný úhel.



### RYCHLÝ TIP

- Vysvětlete, že pro ovládání servomotoru využíváme knihovnu připojenou pomocí příkazu #include. Tato knihovna nám ulehčí značně práci při ovládání.

```
1 #include <Servo.h> // Připojení knihovny
2
3 Servo myservo; // Vytvoření instance třídy pro každý servomotor
4 int pos = 90; // Pozice servomotoru 90°
5
6 void setup()
7 {
8     // Definice pinu na který je připojen signální vodič
9     // servomotoru
10    myservo.attach(9);
11 }
12
13 void loop()
14 {
15     myservo.write(pos); // Nastavení pozice servomotoru
16     delay(15);
17 }
18
```

Studenti ať program nahrají do desky a odzkouší, zda se servomotor natočí do definované polohy.



### ZEPTEJTE SE STUDENTŮ

→ Co se stane když změníte hodnotu ve funkci delay( )?

Změnou hodnoty se bude měnit prodleva natočení servomotoru.



### RYCHLÝ TIP

→ Ať studenti změní hodnotu v proměnné pos v rozmezí 0 - 180. Ať si zároveň vyzkouší, jak servomotor reaguje na změnu hodnoty ve funkci delay( ).

→ Nezapomeňte, že při každé změně se musí program opět nahrát do desky.



### ÚKOL PRO STUDENTY

→ A) Změňte program tak, aby se postupně osa servomotoru nastavila do pozic 10°, 40°, 80°, 120°, 160°.

Řešení je velmi jednoduché. Stačí zkopirovat nastavení pozice servomotoru s příslušnými hodnotami úhlů a mezi ně vložit funkci delay( ).

## 2. KROK 15 minut

Na základě zvládnutí principů ovládání servomotoru budou studenti řešit následující úkol.



### ÚKOL PRO STUDENTY

→ B) Napište program, který bude otáčet osou servomotoru od 0° do 180°. Po dosažení krajní pozice se bude otáčet zpět. To se bude provádět neustále dokola.

Studenti k tomu využijí poznatku z Lekce 2., kde programovali pohyb světla pomocí cyklu for. Příklad z Lekce 2. je jeho obdobou a může jim být návodem.

### 3. KROK 15 minut

Pokud studenti zvládli předchozí úkol, tak můžete ještě ukázat, jak obvod se servomotorem rozšířit o potenciometr, který bude servomotor ovládat.

Ať studenti zapojí do obvodu potenciometr podle přiloženého schématu v pracovním listu nebo podle promítaného obrazu pomocí dataprojektoru.



#### RYCHLÝ TIP

- ➔ Během zapojování můžete vysvětlit princip potenciometru. Zejména pak to, že hodnoty se čtou pomocí funkce `analogRead()`, jejíž hodnoty vrací číslo 0-1023, které je úměrné velikosti napětí.

Po zapojení obvodu studentům ukažte programový kód, ve kterém upozorněte na funkci `map()`. Studenti by mohli přijít na její princip změnou číselných parametrů.

# PRACOVNÍ LIST – SERVOMOTOR

SERVOMOTOR JE SPECIÁLNÍ MOTOR, KTERÝ SE POUŽÍVÁ VŠUDE TAM, KDE SE VYŽADUJE PŘESNÉ NASTAVENÍ POLOHY. SERVOMOTORY SE VYUŽÍVAJÍ NAPŘÍKLAD PRO OVLÁDÁNÍ ROBOTICKÉ RUKY.

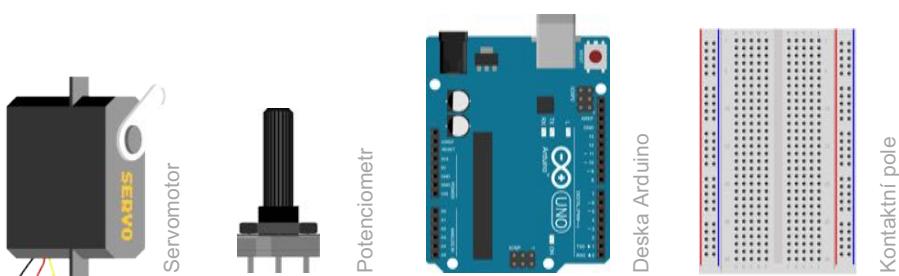
## CO SE NAUČÍTE

- ① Zapojit a ovládat servomotor.
- ② Zopakujete si programování cyklu **for**.
- ③ Zapojovat potenciometr a ovládat s ním servomotor.



## CO BUDETE POTŘEBOVAT

- ① Servomotor.
- ② Potenciometr.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

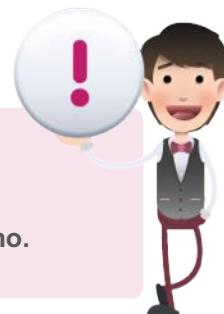
## OTÁZKA PRO VÁS

→ Kde všude se můžete setkat se servomotory a k čemu se ještě dají využít?



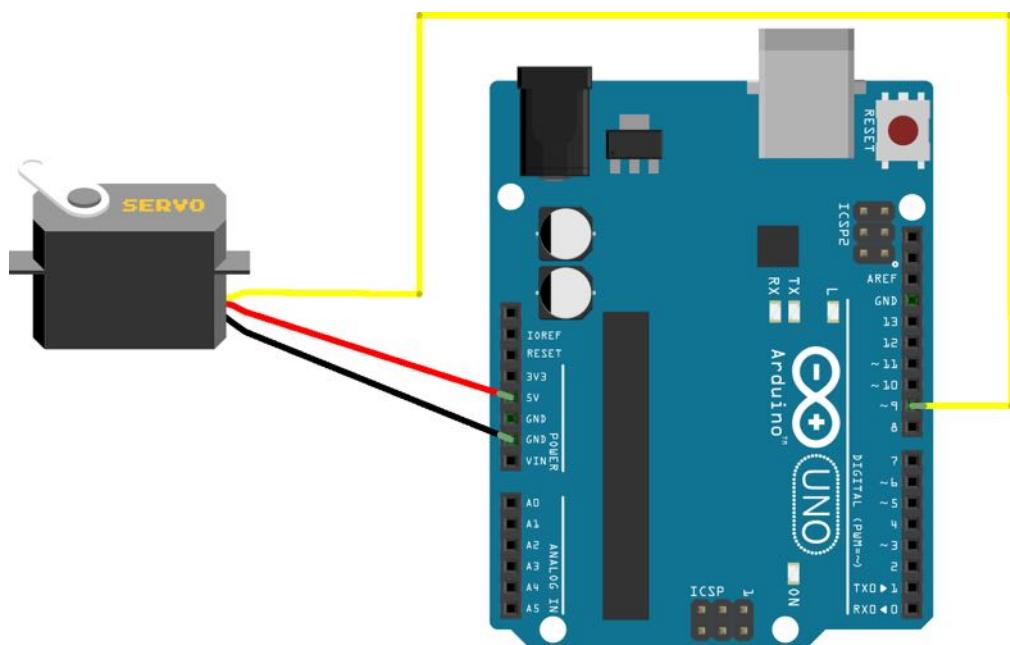
## A JDĚTE NA TO ...

- ① Podle přiloženého schématu zapojte obvod se servomotorem.



### DEJTE SI POZOR

→ Pozor si dejte na zapojení vodičů servomotoru. Červený je připojen k napájení, hnědý na zemnění a oranžový na pin PWM desky Arduino.



- ② Napište program, který servomotor nastavuje na pozici 90°.

```

1 #include <Servo.h> // Připojení knihovny
2
3 Servo myservo; // Vytvoření instance třídy pro každý servomotor
4 int pos = 90; // Pozice servomotoru 90°
5
6 void setup()
7 {
8     // Definice pinu na který je připojen signální vodič
9     // servomotoru
10    myservo.attach(9);
11 }
12
13 void loop()
14 {
15     myservo.write(pos); // Nastavení pozice servomotoru
16     delay(15);
17 }
18

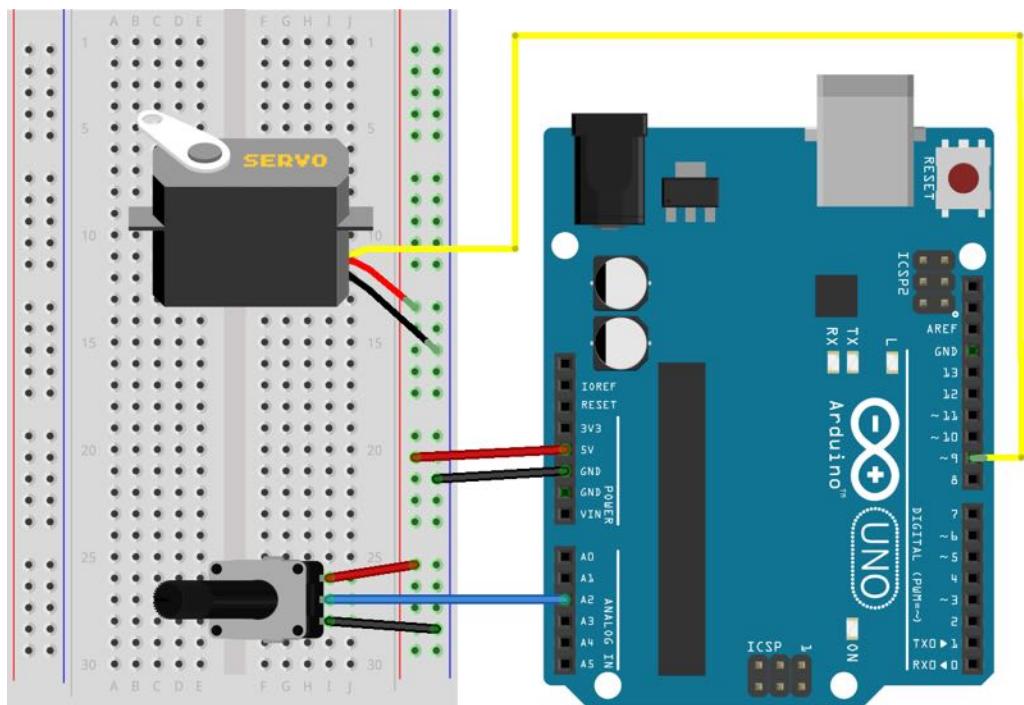
```



### ÚKOLY PRO VÁS

- ➔ Změňte program tak, aby se postupně osa servomotoru nastavila do pozic 10°, 40°, 80°, 120°, 160°.
- ➔ Napište program, který bude otáčet osou servomotoru od 0° do 180°. Po dosažení krajní pozice se bude otáčet zpět. To se bude provádět neustále dokola.

- ③ Pokud jste úkoly splnili a vše funguje jak má, zkuste si zapojit a naprogramovat ještě jeden obvod. Do stávajícího obvodu stačí pouze přidat potenciometr podle přiloženého schématu.



- ④ Napište program, který zajistí ovládání servomotoru potenciometrem.

```

1 #include <Servo.h>
2
3 Servo myservo;
4 int pos = 0;
5
6 void setup()
7 {
8     myservo.attach(9);
9 }
10
11 void loop()
12 {
13     pos = analogRead(A2);
14     pos = map(pos, 0, 1023, 0, 179);
15     myservo.write(pos);
16     delay(5);
17 }
```

### NEZAPOMEŇTE

→ Nahrajte program do desky Arduino, kliknutím na ikonu 



### VYZKOUŠEJTE

- Co se bude dít, pokud změníte hodnoty první dvojice parametrů 0 a 1023 ve funkci `map()`.
- Co se změní, když upravíte hodnoty druhé dvojice parametrů 0 a 179 ve funkci `map()`.



## ŘEŠENÍ ÚLOH

Úkol A)

```
1 #include <Servo.h>
2
3 Servo myservo;
4
5 void setup()
6 {
7 myservo.attach(9);
8 }
9
10 void loop()
11 {
12     myservo.write(10);
13     delay(1000);
14     myservo.write(40);
15     delay(1000);
16     myservo.write(80);
17     delay(1000);
18     myservo.write(120);
19     delay(1000);
20     myservo.write(160);
21     delay(1000);
22 }
```

### Úkol B)

Pro řešení této úlohy stačí využít znalostí z předchozí lekce, kde se pomocí cyklu **for** ovládalo běžící světlo.

```
1 #include <Servo.h>
2
3 Servo myservo;
4 int pos = 0;
5
6 void setup()
7 {
8     myservo.attach(9);
9 }
10
11 void loop()
12 {
13     // Úprava mezní hodnoty natočení servomotoru na 180
14     for(pos = 0; pos <= 180; pos += 1)
15     {
16         myservo.write(pos);
17         // Úprava prodlevy, aby se servo ještě více zpomalilo
18         delay(20);
19     }
20     // Úprava mezní hodnoty natočení servomotoru ze 180 na 0
21     for(pos = 180; pos >= 0; pos -= 1)
22     {
23         myservo.write(pos);
24         // Úprava prodlevy, aby se servo zrychlilo
25         delay(5);
26     }
27 }
```

# PRŮVODCE HODINOU II



Studenti mohou využít základního zapojení obvodu z předchozí hodiny. Toto zapojení bude rozšířeno o nové komponenty, kterými jsou tlačítka a fotorezistory. Studenti si prohloubí znalosti o čtení analogových hodnot a jejich zpracování. Přestože se pravděpodobně s tlačítky ještě nesetkali, jejich použití nebude složité.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, servomotor, 2x tlačítko, 2x fotorezistor, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 3, která je ke stažení na GitHub.
- ⑤ Pracovní listy pro studenty (ke stažení na GitHub).



## 1. KROK 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní této hodiny bude využití znalostí z hodiny předešlé, kdy se seznámili se servomotorem a jeho ovládáním. Servomotor se dá ovládat různými prvky, proto si studenti vyzkouší ovládání pomocí tlačítek a fotorezistorů.



Studenti mohou využít již složený obvod z minulé hodiny. Pokud jej složený nemají, tak jej v rámci opakování sestrojí. Stačí pouze zapojení servomotoru.

## 2. KROK 5 minut

Vysvětlete studentům strukturu podmínkového příkazu **if** a možnost jeho použití, například při testování hodnot vrácených z analogových zařízení, jako je potenciometr, tlačítko nebo fotorezistor.

## 3. KROK 10 minut

- ① Nyní ať si studenti zapojí obvod s jedním tlačítkem. Při stisku tlačítka se bude servomotor otáčet v jednom směru. Např. od 0° do 180°. Schéma zapojení je součástí pracovního listu nebo jej lze promítnout prostřednictvím dataprojektoru.
- ② Ať si studenti po zapojení obvodu napíší vzorový programový kód, který ovládá servomotor pouze jedním tlačítkem.
- ③ Kód ať nahrají do desky Arduino a otestují.



### NA CO SI DÁT POZOR

- Při zapojování tlačítka je nutné dbát na jeho orientaci a správné zapojení rezistoru.



### ZEPTEJTE SE STUDENTŮ

- Jak vysvětlíte funkčnost programového kódu, zejména pak podmínkové příkazy **if** od řádku 19.

## 4. KROK 10 minut



### ÚKOL PRO STUDENTY

- A) Do obvodu s jedním tlačítkem přidejte druhé tlačítko. Naprogramujete kód tak, aby jedno stisknuté tlačítko otáčelo servomotorem od 0° do 180° a druhé tlačítko od 180° do 0°.

## 5. KROK 15 minut

- ① Studentům řekněte, že poslední projekt týkající se servomotoru bude využívat nový prvek, kterým je **fotorezistor**. Budou použity fotorezistory dva. Servomotor se bude natáčet směrem k fotorezistoru, na který bude dopadat intenzivnější světlo.
- ② Studenti ať sestrojí obvod podle uvedeného schématu zapojení.

### ZEPTEJTE SE STUDENTŮ

→ Jakou podobnost vidíte v zapojení fotorezistorů se zapojením potenciometru?

Kombinace fotorezistorů vytváří tzv. proudový dělič a jeho datový vodič je připojen opět na analogový pin A0.



### ÚKOL PRO STUDENTY

→ B) Zkuste naprogramovat ovládání servomotoru tak, že pokud posvítíte na první fotorezistor, servomotor se bude otáčet v jednom směru a pokud posvítíte na druhý fotorezistor, bude se otáčet ve směru druhém.

Jako zdroj světla můžete použít mobilní telefon.



# PRACOVNÍ LIST – SERVOMOTOR, TLAČÍTKA A FOTOREZISTORY

V TÉTO ČÁSTI BUDETE POKRAČOVAT V PROGRAMOVÁNÍ OVLÁDÁNÍ SERVOMOTORU. NAUČÍTE SE PRACOVAT S TLAČÍTKY, FOTOREZISTORY A VYUŽIJETE K TOMU NOVÝ PODMÍNKOVÝ PŘÍKAZ.

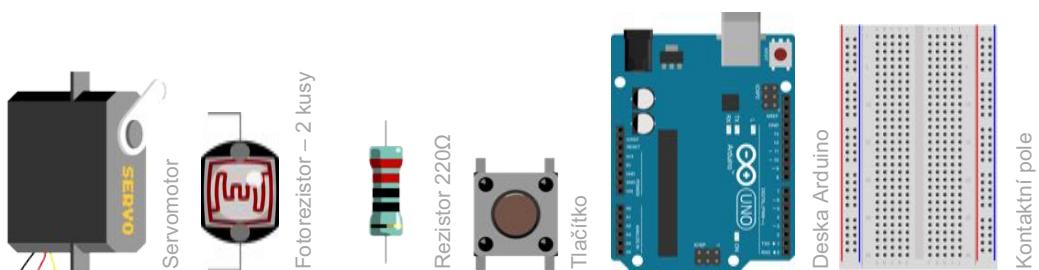
## CO SE NAUČÍTE

- ① Zapojení tlačítka do obvodu pro ovládání servomotoru.
- ② Použití podmínkového příkazu **if**.
- ③ Rozšíření znalostí o čtení analogových hodnot.
- ④ Zapojení fotorezistorů pro ovládání servomotoru.



## CO BUDETE POTŘEBOVAT

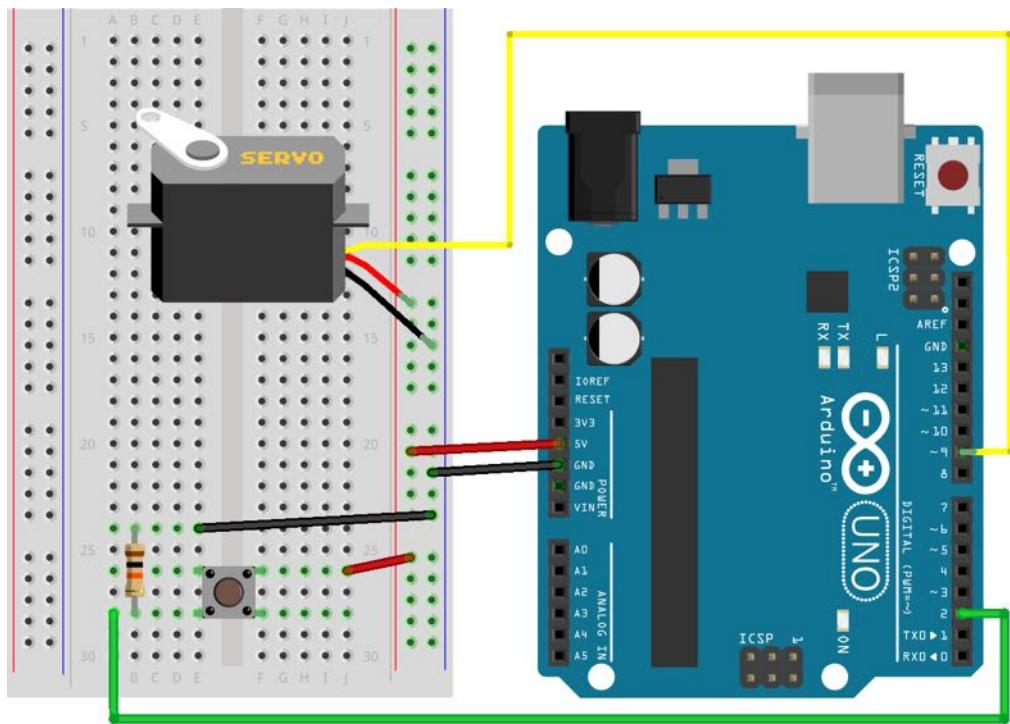
- ① Servomotor.
- ② 2x tlačítko.
- ③ 2x fotorezistor
- ④ Desku Arduino.
- ⑤ Kontaktní pole.
- ⑥ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

- ① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do jeho vylepšení o zapojení tlačítka. V opačném případě obvod musíte opět složit podle přiloženého schématu.

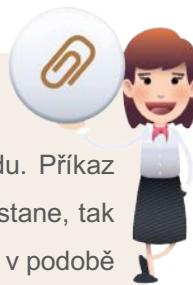


**NA CO SI DÁT POZOR**

➔ Při zapojování tlačítka je nutné dbát na jeho orientaci a správné zapojení rezistoru.



- ② Napište uvedený programový kód pro ovládání servomotoru tlačítkem.



## VÍTE JAK FUNGUJE PODMÍNKOVÝ PŘÍKAZ IF?

Podmínkový příkaz **if...else** umožňuje větší kontrolu nad tokem kódu. Příkaz začíná klíčovým slovem **if** a první podmínkou. Tato podmínka, když nastane, tak se vykoná blok příkazů. Při nesplnění první podmínky lze využít alternativu v podobě příkazu **else**, za kterým následuje opět blok příkazů. Příkaz **else** může být doplněn o další podmínsku **if**. Takže program se prochází postupně přes jednotlivé podmínky a pokud není žádná splněna, tak skončí v samostatné části **else** bez podmínky.

### SYNTAXE

```
1 | if (podmínka A) {  
2 |     // blok příkazů A  
3 | }else if (podmínka B) {  
4 |     // blok příkazů B  
5 | }else{  
6 |     // blok příkazů C  
7 | }
```

```

1 #include <Servo.h>
2
3 int servoPin = 9; // Definice pinu pro řízení servomotoru
4 int Button = 2; // Definice pinu pro čtení z tlačítka
5 int servoPos = 0; // Výchozí pozice servomotoru
6 int delayPeriod = 2; // Prodleva při natáčení servomotoru
7
8 Servo myservo;
9
10 void setup()
11 {
12     myservo.attach(servoPin);
13     myservo.write(servoPos);
14     pinMode(Button, INPUT); // Vyhrazení pinu pro tlačítko
15 }
16
17 void loop()
18 {
19     if(digitalRead(Button) == LOW)
20     {
21         if(servoPos < 180)
22         {
23             servoPos++;
24         }
25         myservo.write(servoPos);
26         delay(delayPeriod);
27     }
28 }
```

### OTÁZKA PRO VÁS

→ Jak vysvětlíte funkčnost programového kódu, zejména pak podmínkové příkazy `if` od řádku 19.



- ③ Nahrajte uvedený programový kód do desky a otestujte.

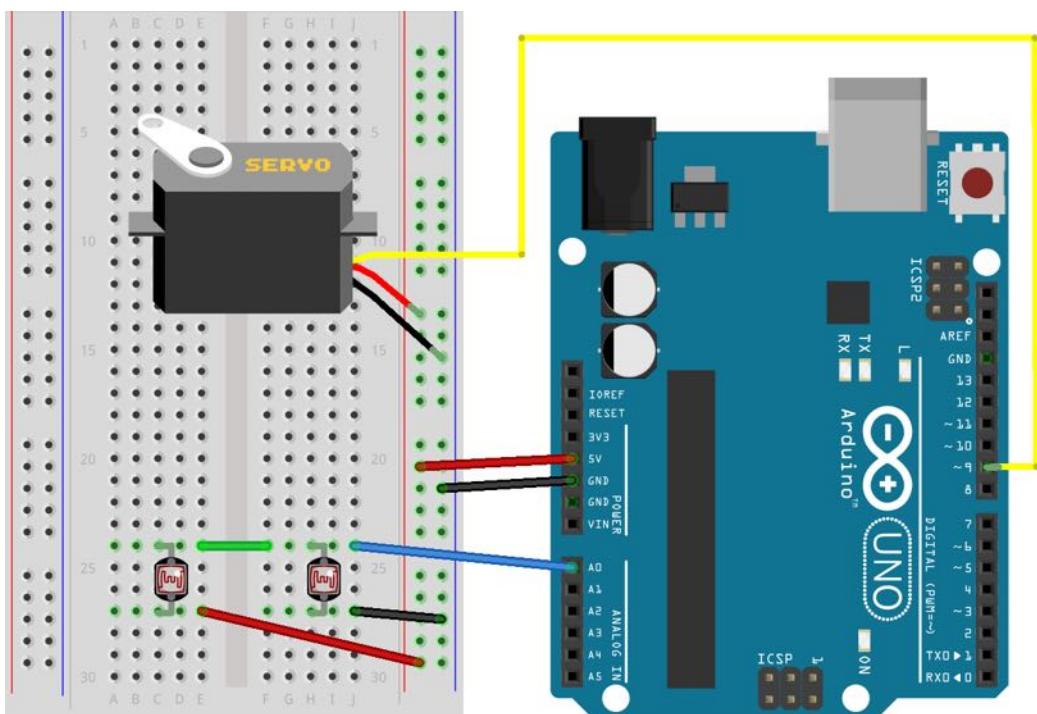


### ÚKOL PRO VÁS

→ A) Do obvodu s jedním tlačítkem přidejte druhé tlačítko. Naprogramujete kód tak, aby jedno stisknuté tlačítko otáčelo servomotorem od  $0^\circ$  do  $180^\circ$  a druhé tlačítko od  $180^\circ$  do  $0^\circ$ .

Povedlo se? Tak výborně.

- ④ Nyní náš obvod změníme. Odpojte tlačítka a zapojte fotorezistory podle následujícího schématu.



### OTÁZKA PRO VÁS

→ Jakou podobnost vidíte v zapojení fotorezistorů se zapojením potenciometru?





### ÚKOL PRO VÁS

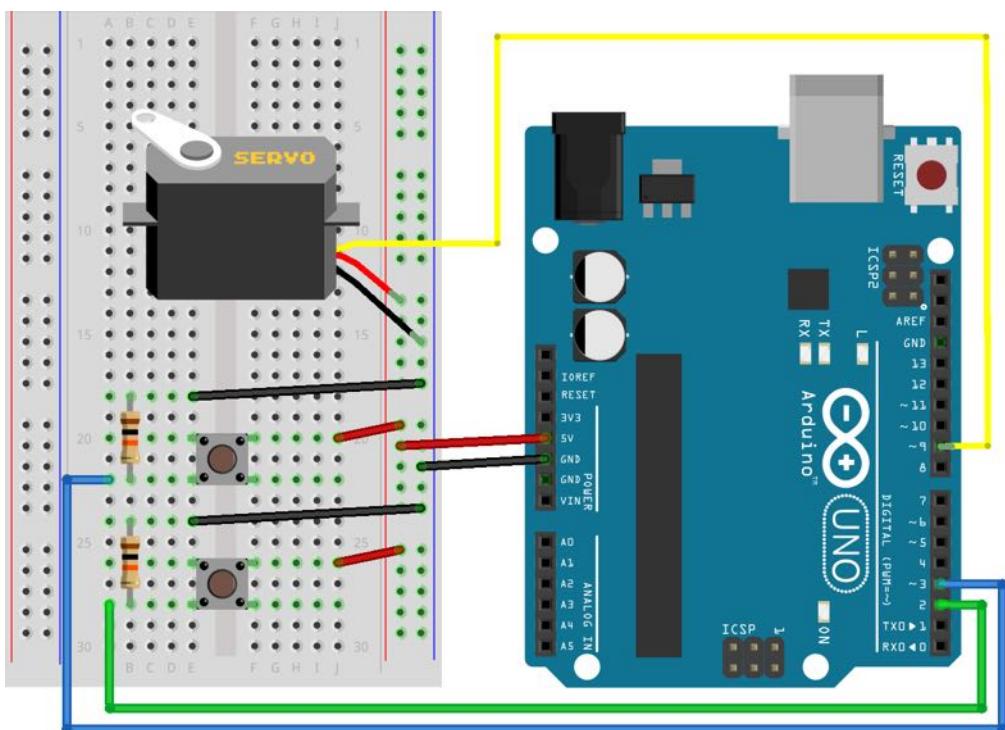
→ B) Zkuste naprogramovat ovládání servomotoru tak, že pokud posvítíte na první fotorezistor, servomotor se bude otáčet v jednom směru a pokud posvítíte na druhý fotorezistor, bude se otáčet ve směru druhém.

Jako zdroj světla můžete použít mobilní telefon.

# ŘEŠENÍ ÚLOH

## Úkol A)

Zapojení druhého tlačítka a naprogramování jeho chování.



```
1 #include <Servo.h>
2
3 Servo myservo;
4 int servoPin = 9;
5 int leftButton = 2;
6 int rightButton = 3;
7 int servoPos = 90;
8 int delayPeriod = 2;
9
10 void setup()
11 {
12     myservo.attach(servoPin);
13     myservo.write(servoPos);
14     pinMode(leftButton, INPUT);
15     pinMode(rightButton, INPUT);
16 }
17
18 void loop()
19 {
20     if(digitalRead(leftButton) == LOW)
21     {
22         if(servoPos > 0)
23         {
24             servoPos--;
25         }
26         myservo.write(servoPos);
27         delay(delayPeriod);
28     }
29
30     if(digitalRead(rightButton) == LOW)
31     {
32         if(servoPos < 180)
33         {
34             servoPos++;
35         }
36         myservo.write(servoPos);
37         delay(delayPeriod);
38     }
39 }
```

### Úkol B)

Pro zapojení fotorezistorů je využit tzv. proudový dělič. Díky tomu využijeme jediný analogový vstup a celý příklad lze přirovnat k zapojení servomotoru s potenciometrem.

```
1 #include <Servo.h>
2
3 int sensorPin = A0;
4 int servoPin = 9;
5
6 int sensorValue = 0;
7 int servoPos = 90;
8
9 Servo myservo;
10
11 void setup() {
12     pinMode(sensorPin, INPUT);
13     myservo.attach( servoPin );
14 }
15
16 void loop() {
17     sensorValue = analogRead(sensorPin);
18
19     if (sensorValue < 512 )
20     {
21         if (servoPos < 180)
22         {
23             servoPos++;
24         }
25     }
26
27     if (sensorValue > 512 )
28     {
29         if (servoPos > 0)
30         {
31             servoPos--;
32         }
33     }
34
35     myservo.write(servoPos);
36     delay(100);
37 }
```

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALITY OBVODŮ A PROGRAMOVÉHO KÓDU, KTERÝ JE ZAMĚŘEN NA POUŽÍVÁNÍ SERVOMOTORŮ S VYUŽITÍM PODMÍNKOVÉHO PŘÍKAZU IF. ŘEŠENÉ ÚKOLY ZOHLEDŇUJÍ NOVĚ PROBRANÉ VĚCI A STAVÍ NA PŘEDCHOZÍCH ZNALOSTECH.

## OBSAH PRŮVODCE

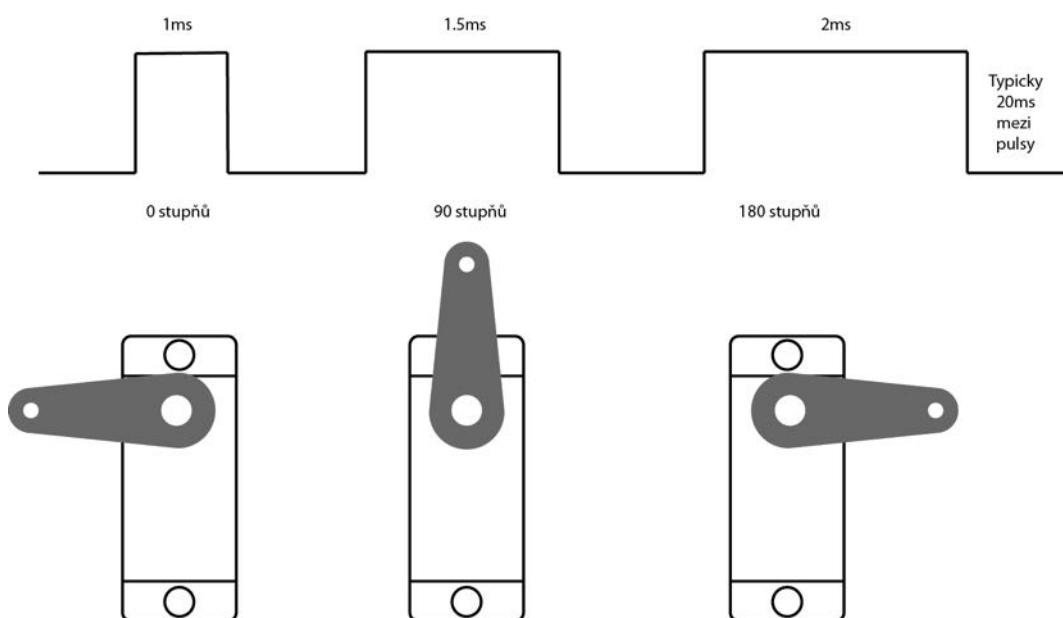
- ① Princip servomotorů.
- ② Vysvětlení podmínkového příkazu **if..else**.
- ③ Příklad zapojení servomotoru.
- ④ Princip a zapojení potenciometru v Arduino pro ovládání servomotoru.
- ⑤ Programové kódy pro vysvětlení používání podmínkového příkazu **if..else..**
- ⑥ Řešené problémy při zapojení servomotoru.
- ⑦ Technická část pro závěrečný projekt – slunečnice.
- ⑧ Vysvětlení řešení samostatných úkolů.

## SERVOMOTORY

Servomotory jsou speciální typy motorů, které obvykle neumožňují otáčení neustále dokola, ale udržují nastavený úhel natočení. Využívají se například pro ovládání robotické ruky, změny pozice klapek, křídélek nebo kormidla u leteckých modelů. Jejich hlavní výhodou je malý rozměr a hmotnost s relativně velkou silou.

Servomotory jsou řízeny posíláním elektrického impulsu s proměnnou šířkou nebo **modulací šířky impulzu** (PWM). Impuls je předáván přes řídící kabel. Servomotor se obvykle může otáčet o  $90^\circ$  v obou směrech o celkovém rozsahu  $180^\circ$ . Neutrální poloha motoru je definována jako poloha, ve které má servomotor stejné množství potencionálního otáčení v obou směrech. PWM posílaná k motoru určuje polohu hřídele a na základě trvání impulsu odesланého řídícím vodičem. Rotor se vrátí do požadované polohy.

Servomotor očekává puls každých 20 milisekund (ms) a délka impulsu určuje, jak daleko se motor otáčí. Například 1,5ms puls přivede motor do polohy  $90^\circ$ . Kratší než 1,5 ms se pohybuje proti směru hodinových ručiček k poloze  $0^\circ$  a delší než 1,5 ms otáčí servomotorem ve směru hodinových ručiček směrem k  $180^\circ$  pozici.



Obr. 1 - Princip servomotoru

## PODMÍNKOVÝ PŘÍKAZ IF..ELSE

Podmínkový příkaz **if...else** umožňuje větší kontrolu nad tokem kódu. Příkaz začíná klíčovým slovem **if** a první podmínkou. Když tato podmínka nastane, tak se vykoná blok příkazů. Při nesplnění první podmínky lze využít alternativu v podobě příkazu **else**, za kterým následuje opět blok příkazů. Příkaz **else** může být doplněn o další podmínku **if**. Program se prochází postupně přes jednotlivé podmínky a pokud není žádná splněna, tak skončí v samostatné části **else** bez podmínky.

### SYNTAXE

```
1  if (podmínka A) {  
2      // blok příkazů A  
3  }else if (podmínka B) {  
4      // blok příkazů B  
5  }else{  
6      // blok příkazů C  
7  }
```

Všimněte si, že blok **else if** může být použit s nebo bez ukončovacího bloku **else** a naopak. Je povolen neomezený počet takových **else if** větví.

Pro podmínky příkazu **if..else** se využívá porovnávacích operátorů.

```
1  x == y (x se rovná y)  
2  x != y (x není rovno y)  
3  x < y (x je menší než y)  
4  x > y (x je větší než y)  
5  x <= y (x je menší nebo rovno y)  
6  x >= y (x je větší nebo roven y)
```



### DEJTE SI POZOR

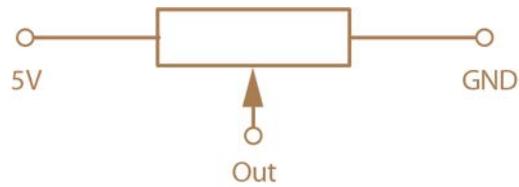
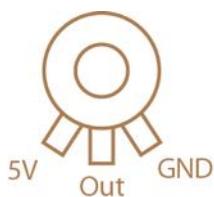
→ Musíte si dát pozor na to, abyste náhodou nepoužili přiřazovací operátor. Tj. znaménko jednoduchého rovná se. Např. `if(x=10)`. Takto uvedený operátor by proměnné `x` přiřadil hodnotu 10. Pro porovnání, je nutné použít dvojité rovná se. Např. `if(x==10)`.

## POTENCIOMETR V ARDUINU

Potenciometr je součástka, poskytující proměnlivý odpor, který lze číst deskou Arduino jako analogovou hodnotu. Potenciometry mají zpravidla tři vývody.

- ① Jeden vývod je připojení k napětí.
- ② Druhý vývod (prostřední) je připojen do analogového vstupu desky Arduino.
- ③ Třetí vývod je připojen k zemnění.

Otáčením kolíku potenciometru měníme velikost odporu na obou stranách snímací plochy, která je připojena k prostřednímu vývodu potenciometru. Pokud bude hřídelí otočeno do krajní polohy (minimální), bude hodnota výstupního napětí 0V a přečteme hodnotu 0. Otočením hřídele na druhou stranu do krajní polohy (maximální) bude hodnota napětí 5V, ale na analogovém vstupu přečteme hodnotu 1023. Funkce `analogRead()` vrací číslo mezi 0 a 1023, které je úměrné hodnotě velikosti napětí.

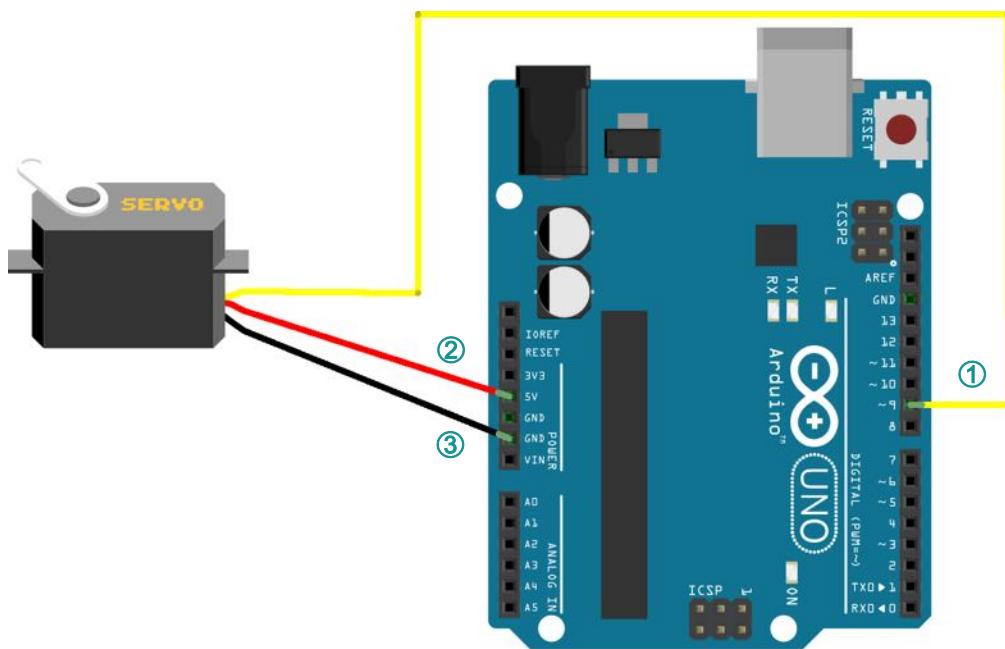


Obr. 2 - Vývody potenciometru

## ZÁKLADNÍ PŘÍKLAD

### ZAPOJENÍ OBVODU

Základní příklad se omezí pouze na jednoduché zapojení servomotoru bez přídavného potenciometru.



Obr. 3 – Základní zapojení servomotoru

- ① Řídící vodič servomotoru je připojen do desky Arduino na jeden z digitálních pinů, který umožňuje PWM. V příkladu to je pin číslo 9.
- ② Napájení servomotoru je zajištěno přes červený vodič, který je připojen na pin 5V. Pro proudově náročnější servomotory lze využít externí zdroj napájení.
- ③ Vodič zemnění je připojen na pin GND.

## PROGRAMOVÝ KÓD

Základní programový kód využívá již známých konstrukcí, které byly použity v předchozích lekcích. Proto je vhodné si zopakovat především příkaz cyklu **for**.



Pro práci se servomotory se využívá již v Arduinu připravená knihovna **Servo**.

Pokud by nebyla nainstalovaná, musí se přidat podle návodu v úvodní kapitole.

### ZÁKLADNÍ PŘÍKLAD – ZMĚNA ÚHLU NATOČENÍ

Tento program postupně mění úhel natočení z 0° až 180° a zpět.

```
1 #include <Servo.h>                                ①
2
3 Servo myservo;                                     ②
4 int pos = 0;                                         ③
5
6 void setup()                                         ④
7 {
8     myservo.attach(9);
9 }
10
11 void loop()                                         ⑤
12 {
13     for(pos = 0; pos <= 180; pos++)                  ⑥
14     {
15         myservo.write(pos);                         ⑦
16         delay(15);
17     }
18     for(pos = 180; pos >= 0; pos--)                  ⑧
19     {
20         myservo.write(pos);                         ⑨
21         delay(15);
22     }
23 }
```

- ① Připojení knihovny pro ovládání servomotoru.
- ② Vytvoření instance **myservo** třídy pro každý motor.
- ③ Deklarace proměnné **pos**, obsahující pozici motoru – úhel natočení.
- ④ Definice pinu, na který je motor připojen. Zde je na pinu **9**.
- ⑤ Cyklus **for** zajišťující otáčení od  $0^\circ$  do  $180^\circ$ .
- ⑥ Nastavení motoru na aktuální úhel.
- ⑦ Přerušení běhu programu, aby měl motor čas se natočit.
- ⑧ Cyklus **for** zajišťující otáčení od  $180^\circ$  zpět na  $0^\circ$ .
- ⑨ Nastavení motoru na aktuální pozici definovanou v proměnné **pos**.
- ⑩ Přerušení běhu programu, aby měl motor čas se natočit do nové pozice.



(Př. 1) Naprogramujte servomotor tak, aby se otáčel od  $0^\circ$  do  $120^\circ$  pomaleji a při zpětném chodu aby se otáčel rychleji.



#### SERVOMOTOR NEFUNGUJE

**Zapojení servomotoru** – zkontrolujte zapojení signálního vodiče servomotoru. Tento vodič musí být na některém z pinů poskytujících PWM. Dále zkontrolujte, zda nemáte zapojený vodič napájené (červený) a zemnění (hnědý) naopak.

#### NEJDE NAHRÁT KÓD DO DESKY

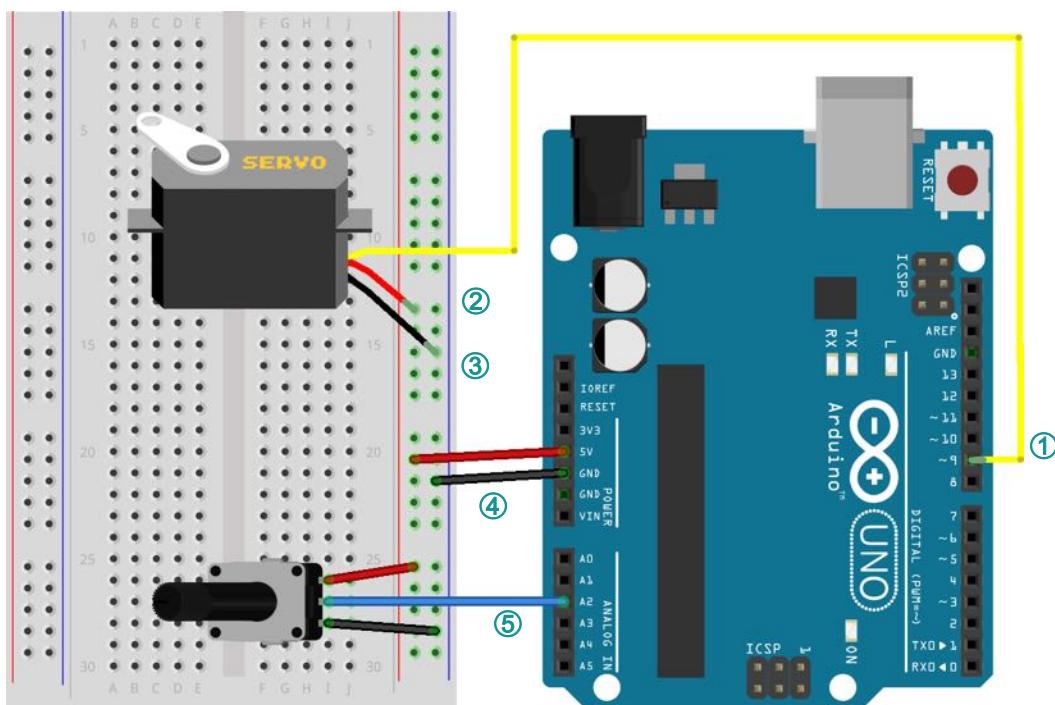
**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

# SERVOMOTOR OVLÁDÁNÝ POTENCIOMETREM

## ZAPOJENÍ OBVODU

Základní příklad je rozšířen o zapojení potenciometru, pomocí kterého lze ovládat natočení servomotoru.



Obr. 4 - Zapojení potenciometru pro ovládání servomotoru

- ① Řídící vodič servomotoru je připojen do desky Arduino na jeden z digitálních pinů, který umožňuje PWM. V příkladu to je pin číslo **9**.
- ② Napájení servomotoru je zajištěno přes červený vodič, který je připojen do kontaktního pole. Pro proudově náročnější servomotory lze využít externí zdroj napájení.
- ③ Vodič zemnění je připojen také do kontaktního pole, do svislého bloku pro zem.
- ④ Do kontaktního pole je přivedeno napájení (červený vodič) a zemnění (černý vodič) přímo z desky Arduino.
- ⑤ Prostřední kontakt potenciometru je přiveden na analogový pin desky Arduino **A2**. Krajní vývody potenciometru jsou přivedeny do kontaktního pole, červený vodič do části pro napájení a černý vodič do části pro zemnění.

## PROGRAMOVÝ KÓD

Programový kód pro tento příklad je jednodušší, protože nevyužívá cykly **for** pro nastavování pozice, ale přímo potenciometru.



Víme, že při čtení hodnoty potenciometru pomocí funkce `analogRead()`, bude hodnota na analogovém vstupu nabývat rozmezí 0 - 1023. Servomotor ale může být otáčen od 0° do 180°. Jazyk wiring nabízí funkci, která velmi jednoduše namapuje rozmezí hodnot potenciometru do rozmezí hodnot servomotoru. Jedná se o funkci `map()`.

`map(hodnota, zMin, zMax, doMin, doMax)`.

## OVLÁDÁNÍ SERVOMOTORU POTENCIOMETREM

Otáčením potenciometru se bude měnit poloha servomotoru.

```
1 #include <Servo.h>
2
3 Servo myservo;
4 int pos = 0;
5
6 void setup()
7 {
8     myservo.attach(9);
9 }
10
11 void loop()
12 {
13     pos = analogRead(A0);
14     pos = map(pos, 0, 1023, 0, 179);
15     myservo.write(pos);
16     delay(5);
17 }
```

- ① Připojení knihovny pro ovládání servomotoru.
- ② Vytvoření instance `myservo` třídy pro každý motor.
- ③ Deklarace proměnné `pos`, obsahující pozici motoru – úhel natočení.

- ④ Definice pinu, na který je motor připojen. Zde je na pinu **9**.
- ⑤ Čtení hodnot potenciometru, které jsou v rozmezí 0 – 1023.
- ⑥ Namapování hodnot potenciometru (0-1023) a rozmezí servomotoru (0-179).
- ⑦ Natočení servomotoru do příslušné pozice uložené v proměnné **pos**.
- ⑧ Krátké přerušení běhu programu, aby měl servomotor čas provést změnu natočení.

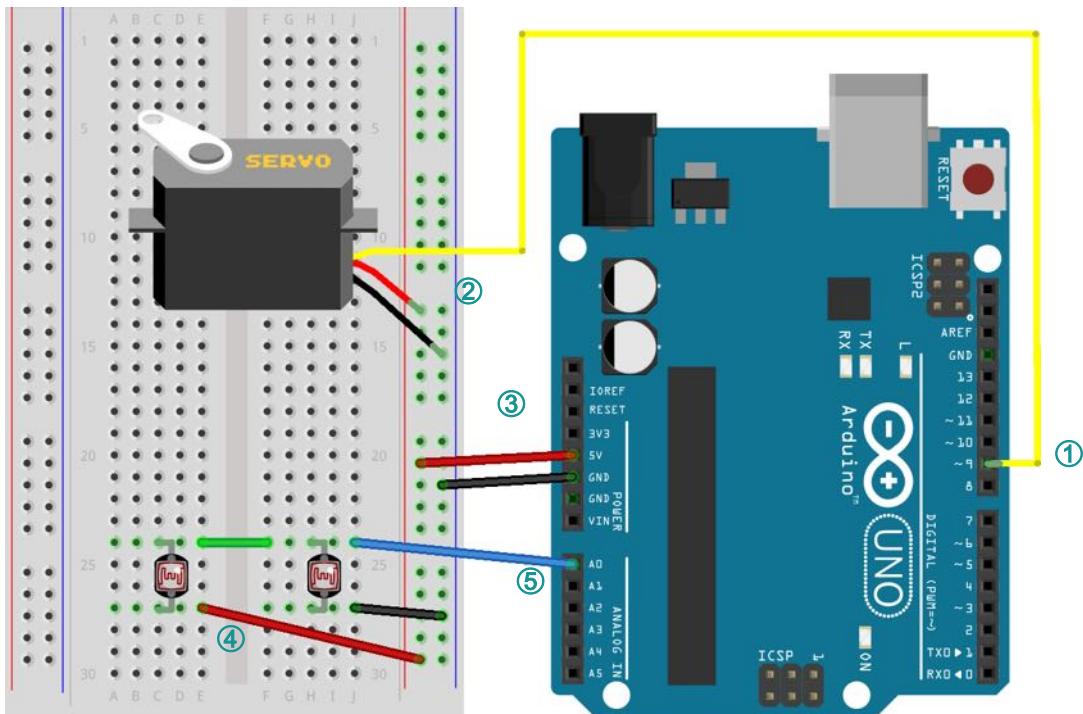


(Př. 2) Zkuste ovládat servomotor pomocí dvou tlačítek. Každé tlačítko bude ovládat servomotor v jednom směru. Dokud bude tlačítka stisknuta, tak se servomotor bude otáčet.

## SERVOMOTOR OVLÁDANÝ FOTOREZISTORY

### ZAPOJENÍ OBVODU

Tento příklad demonstруje ovládání servomotoru v závislosti na osvětlení fotorezistorů. Lze jej spojit s výrobou konstrukce, která představuje slunečníci, která se bude natáčet podle osvitu vždy jednoho z připojených fotorezistorů.



Obr. 5 - Zapojení fotorezistorů pro ovládání servomotoru

- ① Řídící vodič servomotoru je připojen do desky Arduino na jeden z digitálních pinů, který umožňuje PWM. V příkladu to je pin číslo **9**.
- ② Napájení servomotoru je zajištěno přes červený vodič, který je připojen do kontaktního pole. Pro proudově náročnější servomotory lze využít externí zdroj napájení. Do kontaktního pole je zapojen i vodič zemnění.
- ③ Do kontaktního pole je přivedeno napájení (červený vodič) a zemnění (černý vodič) přímo z desky Arduino.
- ④ Fotorezistory jsou zapojeny v kontaktním poli odděleně a propojeny vodičem podle schématu. (Takto zapojené fotorezistory představují tzv. proudový dělič.)
- ⑤ Hodnoty z fotorezistorů jsou předávány do desky Arduino prostřednictvím analogového pinu **A0**.

## PROGRAMOVÝ KÓD

```

1 #include <Servo.h>                                ①
2
3 int sensorPin = A0;                                ②
4 int servoPin = 9;                                  ③
5
6 int sensorValue = 0;                               ④
7 int servoPos = 90;                                ⑤
8
9 Servo myservo;                                    ⑥
10
11 void setup() {                                     ⑦
12     pinMode(sensorPin, INPUT);                    ⑧
13     myservo.attach(servoPin);                   ⑨
14     myservo.write(servoPos);                    ⑩
15 }
16
17 void loop() {                                     ⑪
18     sensorValue = analogRead(sensorPin);        ⑫
19
20     if (sensorValue < (512) )                  ⑬
21     {
22         if (servoPos < 180)                   ⑭
23         {
24             servoPos++;                      ⑮
25         }
26     }
27 }
```

```

28
29     if (sensorValue > (512) )
30     {
31         if (servoPos > 0)
32         {
33             servoPos--;
34         }
35     }
36
37     myservo.write(servoPos);
38
39     delay(100);
40 }
```

- ① Připojení knihovny pro ovládání servomotoru.
- ② Deklarace proměnné pro číslo pinu, na který jsou připojeny fotorezistory.
- ③ Deklarace proměnné pro číslo pinu, na který je připojen datový vodič servomotoru.
- ④ Deklarace proměnné pro hodnoty z fotorezistorů.
- ⑤ Deklarace proměnné s hodnotou pro nastavení servomotoru.
- ⑥ Vytvoření instance **myservo** třídy pro každý motor.
- ⑦ Definice pinu, na který jsou připojeny fotorezistory jako vstup.
- ⑧ Definice pinu, na který je motor připojen. Zde je na pinu **9**.
- ⑨ Nastavení servomotoru do výchozí pozice – 90°.
- ⑩ Čtení hodnot fotorezistoru, které jsou v rozmezí 0 – 1023.
- ⑪ Testování hodnot fotorezistoru. Pokud je hodnota větší než polovina maxima (512), bude se servomotor otáčet od poloviny do 180°.
- ⑫ Průběžné otáčení servomotoru do příslušné pozice uložené ve zvětšující se proměnné **servoPos**.
- ⑬ Test na druhou část pohybu od 512 do 0.
- ⑭ Dekrementace proměnné **servoPos**, která představuje úhel natočení servomotoru.
- ⑮ Zajištění natočení servomotoru.



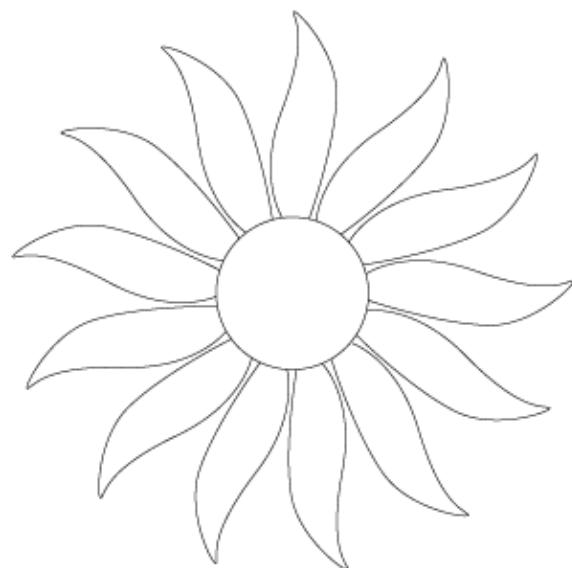
Pokud budete mít připravenou a funkční programovou část, můžete vytvořit květinu, která se otáčí za světlem. Návod je v následující kapitole.

## OTÁČEJÍCÍ SE KVĚTINA

Pro vytvoření otáčející se květiny za světlem budeme potřebovat: karton (tvrdší papír), nůžky a tavnou pistoli (lepící pásku), brčko.

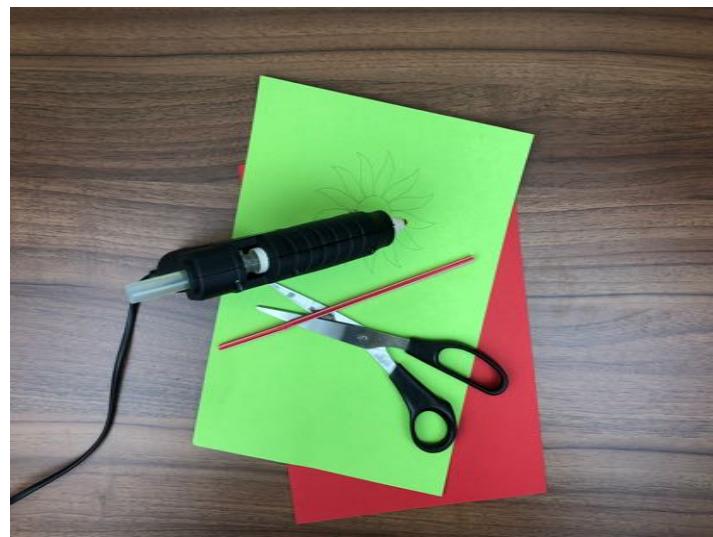
## PAPÍROVÁ KONSTRUKCE

Konstrukce květiny je jednoduchá. Stačí si vytisknout na pevný papír přiloženou šablonu, která je na obrázku Obr. 6 - Šablona květiny.



Obr. 6 - Šablona květiny

Vytištěnou šablonu vystříhněte podle plných čar.



Obr. 7 - Vytištěná šablona



Obr. 8 - Vystřížená šablona

Na vystříženou šablonu naneste tavnou pistoli lepidlo a následně vystřížené červené kolečko.



Obr. 9 – Přilepení středu

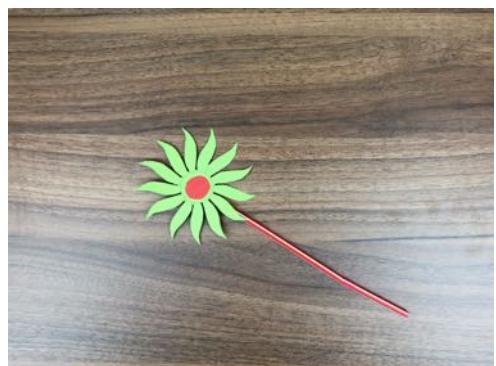


Obr. 10 – Lepení brčka

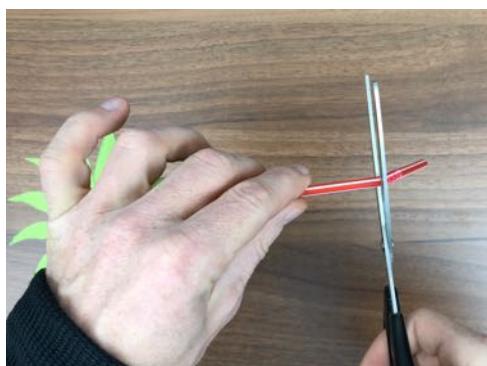
Naneste lepidlo z tavné pistole na brčko, které přilepte na květinu.



Obr. 11 – Přilepení brčka



Obr. 12 – Celý pohled na květinu



Obr. 13 – Zastřížení brčka



Obr. 14 – Připevnění květiny na servo

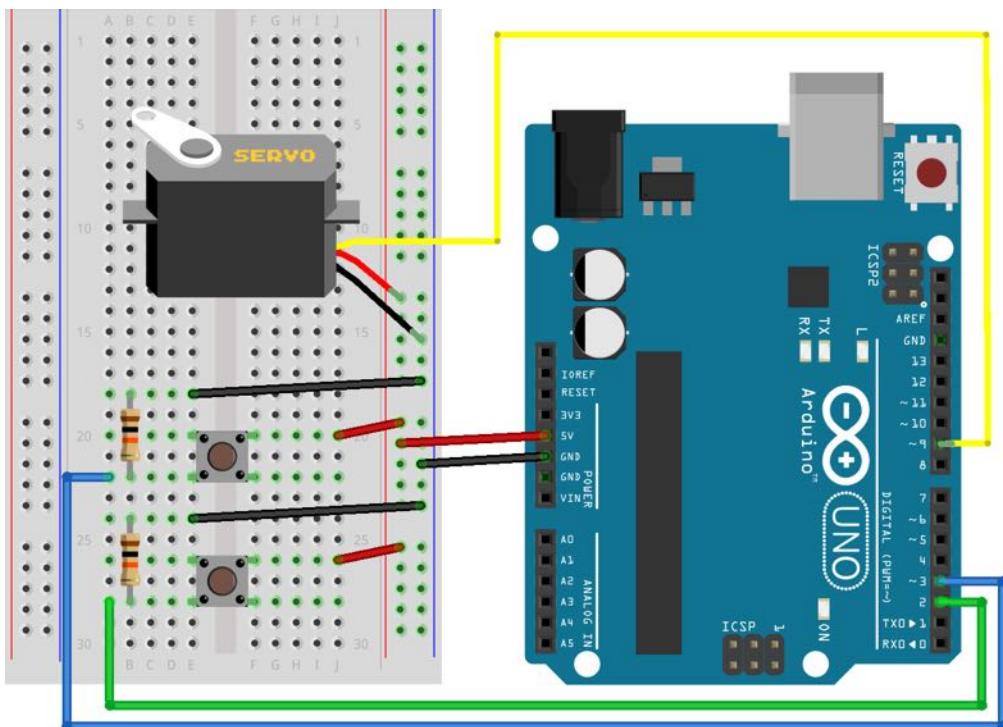
## ŘEŠENÍ PŘÍKLADŮ

(Př. 1) Naprogramujte servomotor tak, aby se otáčel od 0° do 120° pomaleji a při zpětném chodu aby se otáčel rychleji. Řešení příkladu spočívá ve velmi jednoduché úpravě.

```
1 #include <Servo.h>
2
3 Servo myservo;
4 int pos = 0;
5
6 void setup()
7 {
8     myservo.attach(9);
9 }
10
11 void loop()
12 {
13     // Úprava mezní hodnoty natočení servomotoru na 120
14     for(pos = 0; pos <= 120; pos++)
15     {
16         myservo.write(pos);
17         // Úprava prodlevy, aby se servo ještě více zpomalilo
18         delay(20);
19     }
20     // Úprava mezní hodnoty natočení servomotoru ze 120 na 0
21     for(pos = 120; pos >= 0; pos--)
22     {
23         myservo.write(pos);
24         // Úprava prodlevy, aby se servo zrychlilo
25         delay(5);
26     }
27 }
```

## (PŘ. 2) ZKUSTE OVLÁDAT SERVOMOTOR POMOCÍ DVOU TLAČÍTEK.

Tlačítka určují směr otáčení servomotoru.



Obr. 7 - Zapojení tlačítek pro ovládání servomotoru

```
1 #include <Servo.h>
2
3 Servo myservo;
4 int servoPin = 9;
5 int leftButton = 2;
6 int rightButton = 3;
7 int servoPos = 90;
8 int delayPeriod = 2;
9
10 void setup()
11 {
12     myservo.attach(servoPin);
13     myservo.write(servoPos);
14     pinMode(leftButton, INPUT);
15     pinMode(rightButton, INPUT);
16 }
17
18 void loop()
19 {
20     if(digitalRead(leftButton) == LOW)
21     {
22         if(servoPos > 0)
23         {
24             servoPos--;
25         }
26         myservo.write(servoPos);
27         delay(delayPeriod);
28     }
29
30     if(digitalRead(rightButton) == LOW)
31     {
32         if(servoPos < 180)
33         {
34             servoPos++;
35         }
36         myservo.write(servoPos);
37         delay(delayPeriod);
38     }
39 }
```

## 4. RGB DIODA

LEKCE JE ZAMĚŘENA NA ZAJÍMAVOU KOMPONENTU, KTEROU JE RGB DIODA. POMOCÍ RGB DIODY LZE VYTВÁŘET ZAJÍMAVÉ EFEKTY S VYUŽITÍM VELMI JEDNODUCHÉHO ZAPOJENÍ A PROGRAMOVÁNÍ.

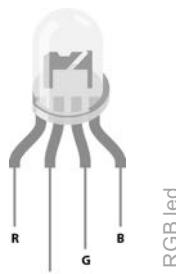
### CÍLE

- ① Jak funguje RGB dioda.
- ② Vysvětlení PWM.
- ③ Zapojení RGB diody.
- ④ Upevnění znalostí o aplikaci podmírkových příkazů a cyklů.
- ⑤ Zopakování práce s vlastními funkcemi.
- ⑥ Závěrečný projekt – Magická lampa.

Čas: **2x45 min**

Úroveň: ■■■■■

Vychází z: **2,3**



POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví základní obvod, ve kterém použijí RGB LED diodou s rezistorem. Tento obvod budou programovat převážně v samostatných úkolech. Tato lekce je zaměřena na základy používání RGB diody a ukázku PWM.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, RGB LED dioda, 3x rezistor  $220\Omega$ , 4x vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 4, která je ke stažení na GitHub.
- ⑤ Pracovní listy pro studenty (ke stažení na GitHub).



## 1. KROK

5 minut

- ① Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude naučit se základům programování **RGB diody**.



### ZKUSTE SE ZEPTAT STUDENTŮ

➔ Víte nebo dokážete uhodnout, na jakém principu pracuje RGB dioda?

- ② V rámci úvodu RGB diodu studentům ukažte, vysvětlete její princip a upozorněte je, že se dá ovládat prostřednictvím PWM.

- ③ V krátkosti vysvětlete, co znamená zkratka PWM.
- ④ Upozorněte na to, jak se skládají barvy. Ukažte studentům prostřednictvím dataprojektoru nebo v pracovním listu obrázek kombinace barev.



#### CO JE RGB DIODA?

- RGB dioda pracuje podobně jako klasická jednobarevná LED dioda. Její princip lze vyčíst ze zkratky RGB, což znamená označení anglického názvu barev – červená, zelená, modrá.
- Takže RGB dioda ve skutečnosti představuje tři LED diody v jednom balení.



#### ZEPTEJTE SE STUDENTŮ

- Jakou roli hraje PWM ve spojení s RGB diodou?  
PWM využijeme, pokud chceme plynulé přechody mezi barvami.

## 2. KROK 10 minut

Řekněte studentům, aby sestavili obvod podle zobrazeného schématu buď promítaného dataprojektorem nebo z pracovního listu.

## 3. KROK 10 minut

- ① Ať si studenti spustí Arduino IDE a napíší základní program pro zobrazení konkrétní barvy.
- ② Klinutím na ikonu pro upload kódu ať nahrají program do desky Arduino.



#### ZEPTEJTE SE STUDENTŮ

- Jakou barvou dioda svítí, pokud je program v pořádku spuštěn?  
Dioda svítí modrou barvou.



### DEJTE SI POZOR

- Pozor na zapojení RGB diody. Musíte si uvědomit, jestli máte RGB diodu se společnou katodou nebo anodou.
- V doporučené sadě je dioda se společnou anodou, tzn. k nejdelšímu vývodu je připojen pin +5V z desky Arduino.

## 4. KROK ⌚ 20 minut

V této lekci si vystačíme s jediným zapojeným obvodem a v rámci samostatných úkolů se soustředíme na využití již naučených programových struktur.



### ÚKOL PRO STUDENTY

- A) Napište program, který bude měnit barvu RGB diody na zelenou, červenou a modrou, vždy po 1 sekundě.



### AŤ STUDENTI VYZKOUŠÍ

- Ať studenti v kódu vyměnít logické hodnoty `LOW` a `HIGH` za odpovídající hodnoty PWM v rozsahu 0 – 255.  
Studenti by si měli uvědomit, jak funguje PWM.



### ÚKOL PRO STUDENTY

- B) Napište program, který bude měnit barvu RGB diody na tyrkysovou, žlutou a fialovou, vždy po 1 sekundě.

# PRACOVNÍ LIST – RGB DIODA I

V TÉTO LEKCI SE SEZNÁMÍTE SE ZAJÍMAVOU KOMPONENTOU, KTEROU JE RGB DIODA. JEJÍ ZAPOJENÍ A PROGRAMOVÁNÍ JE VELICE JEDNODUCHÉ.

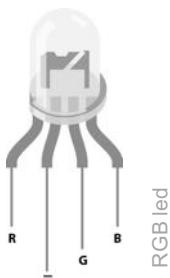
## CO SE NAUČÍTE

- ① Jak pracuje RGB dioda a její zapojení.
- ② Ovládání RGB diody pomocí PWM.
- ③ Využití a opakování vlastních funkcí.



## CO BUDETE POTŘEBOVAT

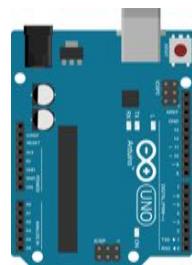
- ① RGB diodu.
- ② Rezistor  $220\Omega$ .
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka (M-M).



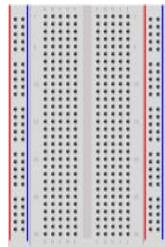
POUŽITÉ SOUČÁSTKY



3x Rezistor  $220\Omega$



Deska Arduino



Kontaktní pole



### OTÁZKA PRO VÁS

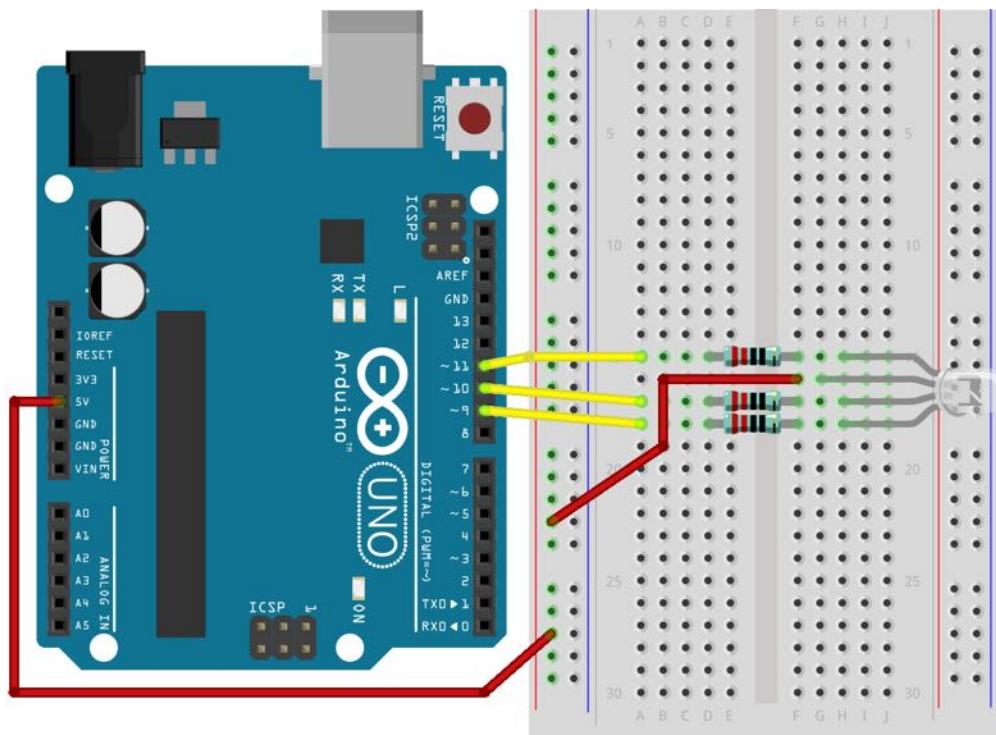
→ Víte nebo dokážete uhodnout, na jakém principu pracuje RGB dioda?

## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.

### DEJTE SI POZOR

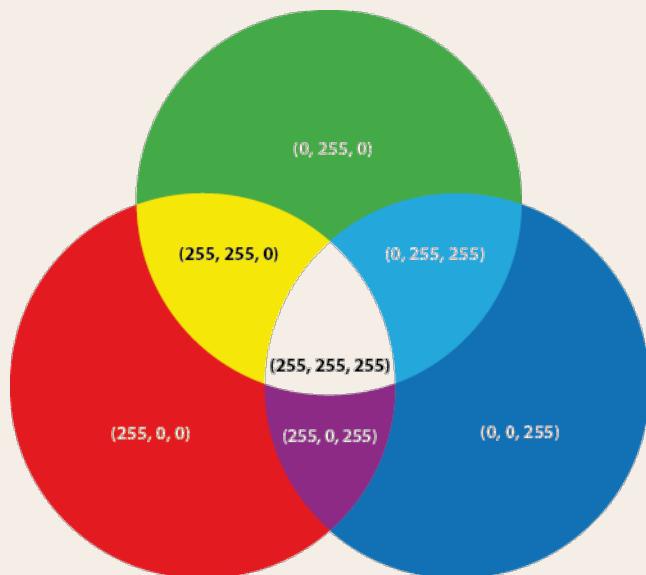
- Pozor si dejte na to, jak zapojujete RGB diodu. Na nejdelší vývod musí být připojen pin +5V, který vede na napájení z desky Arduino. Kratší vývody jsou připojeny na piny PWM (~) desky Arduino.
- Dejte si pozor na hodnotu rezistorů. Zkontrolujte si, že jsou barevně označeny v pořadí červená, červená, modrá, černá, zlatá.





### PRINCIP SKLÁDÁNÍ BAREV

- RGB diody jsou skvělé, protože je lze využít k zobrazení pestrých barev. Červená, zelená a modrá jsou základní barvy a v přídavné barevné paletě je lze kombinovat a vytvářet další odstíny.



② Spusťte program Arduino IDE a napište následující programový kód.

```
1 void setup() {
2     pinMode(11, OUTPUT);          //červená
3     pinMode(10, OUTPUT);          //zelená
4     pinMode(9, OUTPUT);          //modrá
5 }
6
7 void loop() {
8     digitalWrite(11, HIGH);       //červená
9     digitalWrite(10, HIGH);       //zelená
10    digitalWrite(9, LOW);        //modrá
11 }
```

- ③ Po napsání programu připojte USB kabel k desce a k počítači a kliknutím na ikonu pro upload ➔ nahrajte kód do desky Arduino.



### OTÁZKA

➔ Jakou barvou dioda svítí, pokud je program v pořádku nahrán?



### ÚKOL PRO VÁS

➔ A) Napište program, který bude měnit barvu RGB diody na zelenou, červenou a modrou, vždy po 1 sekundě.



### OTÁZKA

➔ Jakou roli hraje PWM ve spojení s RGB diodou?



### VYZKOUŠEJTE SI

➔ Změňte logické hodnoty LOW a HIGH na odpovídající hodnoty PWM v rozsahu 0 – 255.

Pro správnou funkci RGB diodu s využitím PWM je nutné pracovat s funkcí analogwrite(pin, hodnota).



### FUNKCE ANALOGWRITE()

→ Funkce analogWrite(pin, hodnota) posílá analogový signál na uvedený pin ve formě PWM. Při použití této funkce již nepracujeme pouze s krajními hodnotami, ale v plném rozsahu 0-255.



### ÚKOLY PRO VÁS

→ B) Napište program, který bude měnit barvu RGB diody na tyrkysovou, žlutou a fialovou, vždy po 1 sekundě.

# PRŮVODCE HODINOU II



Tato část je pokračování předchozí hodiny. Žáci si prostřednictvím samostatných úkolů zopakují práci s RGB diodou a zafixují již získané znalosti, týkajících se používání cyklu **for**, podmínkového příkazu **if** a vytváření vlastních funkcí. Praktickým výstupem hodiny může být závěrečný projekt tzv. magická lampa.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, RGB LED dioda, 3x rezistor  $220\Omega$ , 4x vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Žáci by měli mít z domova připravenou konstrukci magické lampy.
- ⑤ Prezentace k lekci 4, která je ke stažení na GitHub.
- ⑥ Pracovní listy pro studenty (ke stažení na GitHub).



## 1. KROK 10 minut

- ① Na úvod rozdejte žákům sady Arduino. Řekněte, že budou pokračovat v procvičování programování RGB diody.
- ② Žáci mohou využít zapojení z minulé hodiny nebo v rámci opakování zapojit RGB diodu znovu, podle schématu v pracovním listu.
- ③ Žáci atď si otevřou poslední program z předchozí hodiny, který vytvořili v rámci samostatného úkolu.

### **ZEPTEJTE SE STUDENTŮ**

- Jak byste zjednodušili programový kód, aby RGB dioda pro každou barvu třikrát blikla a vy jste nemuseli neustále opakovat stejnou část kódu?

Vytvoříme vlastní funkci, která bude obsahovat kombinaci kódu pro konkrétní barvu. Takovou funkcí může být

```
setColor(int redC, int greenC, int blueC).
```



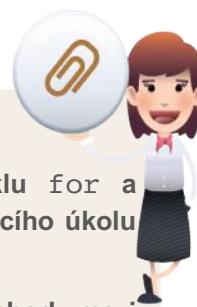
### **ÚKOL PRO STUDENTY**

- C) Napište program, který bude měnit barvu RGB diody na tyrkysovou, žlutou a fialovou. Každá barva blikne třikrát, vždy po 1 sekundě. Prodleva přechodu mezi barvami bude 3 sekundy. Úkol vám ulehčí naprogramování vlastní funkce.



## **2. KROK** 25 minut

Poslední částí lekce je závěrečný projekt, který se nazývá „Magická lampa“. Jedná se o plynulé přechody jednotlivých barev.



### **JAK NA TO?**

- Připomeňte studentům, že se již seznámili s příkazem cyklu `for` a podmínkovým příkazem `if`. Oba dva určitě při řešení následujícího úkolu využijí.
- Ať si studenti uvědomí, jaké složky barev se musí měnit, aby přechody mezi barvami byly patrné a plynulé.



### **ÚKOL PRO STUDENTY**

- D) Napište program, který bude plynule měnit barvy. Vymyslete jej tak, aby byly „namixovány“ postupně všechny možné odstíny.



### FUNKCE ANALOGWRITE()

➔ Funkce `analogWrite(pin, hodnota)` posílá analogový signál na uvedený pin ve formě PWM. Při použití této funkce již nepracujeme pouze s krajními hodnotami, ale v plném rozsahu 0-255.

### 3. KROK 🕒 10 minut

V poslední části hodiny by žáci měli spojit konstrukci magické lampy, elektronické zapojení a otestovat její funkčnost.



Pokud se budou požívat pouze krajní hodnoty pro definici barev, tj. 0 a 255, potom si vystačíme s funkcí `digitalWrite()`. Ovšem pokud budeme chtít řešit plynulý přechod mezi barvami s využitím PWM, musíme použít funkci `analogWrite()`.

# PRACOVNÍ LIST – RGB DIODA II

TATO LEKCE JE POKRAČOVÁNÍM PŘEDCHOZÍ. OPĚT BUDETE PRACOVAT S RGB DIODOU. TENTOKRÁT SI V RÁMCI PRAKTICKÉHO PROJEKTU ZOPAKUJETE NAUČENÉ PROGRAMOVACÍ PŘÍKAZY JAKO JE CYKLUS FOR NEBO PODMÍNKA IF.

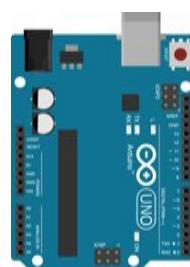
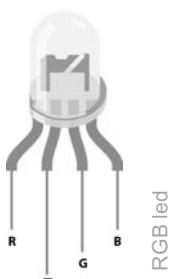
## CO SE NAUČÍTE

- ① Prohloubíte si znalosti v programování RGB diody.
- ② Opakování použití příkazů **for** a **if**.
- ③ Praktické využití ovládání RGB diody pomocí PWM.

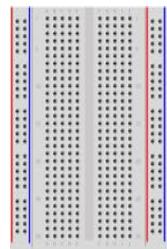


## CO BUDETE POTŘEBOVAT

- ① RGB diodu.
- ② 3x rezistor 220Ω.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka (M-M).



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

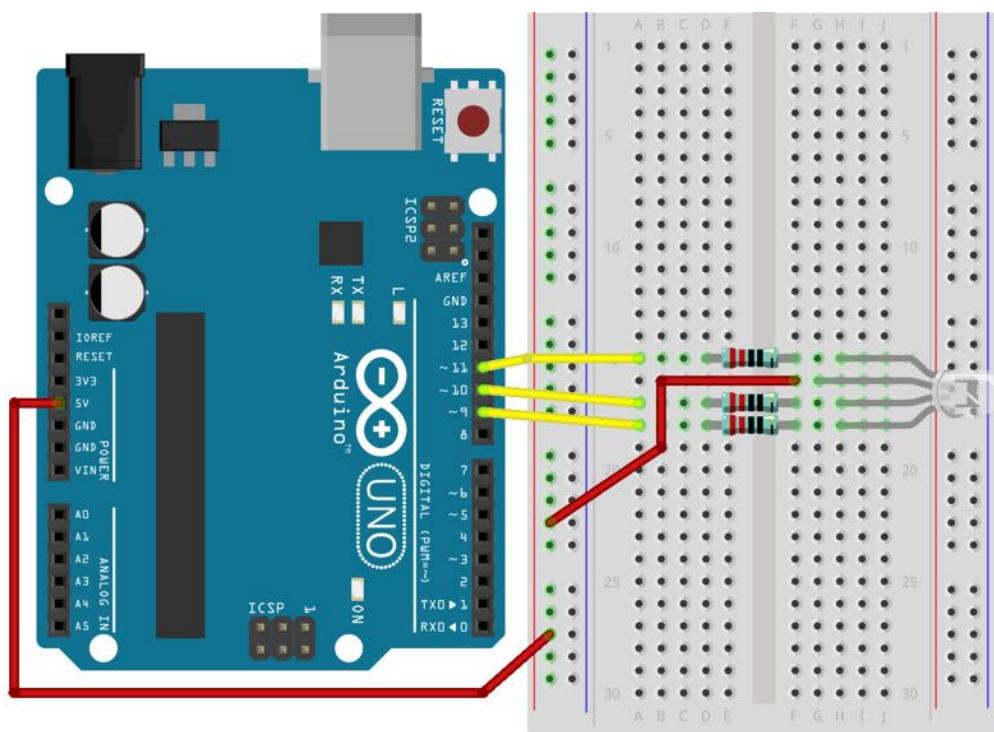
## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.
- ② Pokud máte složený obvod z předchozí hodiny, můžete rovnou začít programovat.



### DEJTE SI POZOR

- Pozor si dejte na to, jak zapojujete RGB diodu. Na nejdelší vývod musí být připojen pin napájení +5V z desky Arduino. Kratší vývody jsou připojeny na piny PWM (~) desky Arduino.
- Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá, černá, zlatá.



- ③ Otevřete poslední program z předchozí hodiny, který jste vytvořili v rámci samostatného úkolu. Program měnil barvu RGB diody na tyrkysovou a fialovou po 1 sekundě.



### OTÁZKA PRO VÁS

- Jak byste zjednodušili programový kód, aby RGB dioda pro každou barvu třikrát blikla a vy jste nemuseli neustále opakovat stejnou část kódu?



### ÚKOL PRO VÁS

- C) Napište program, který bude měnit barvu RGB diody na tyrkysovou, žlutou a fialovou. Každá barva blikne třikrát vždy po 1 sekundě. Prodleva přechodu mezi barvami bude 3 sekundy. V úkolu použijte vlastní funkci.

- ④ Pokud vše funguje a příklady vám přišli jednoduché, pak je tu pro vás jeden opravdu zajímavý. Nazvali jsme ho „Magická lampa“.



### ÚKOL PRO VÁS

- D) Napište program, který bude plynule měnit barvy. Vymyslete jej tak, aby byly „namixovány“ postupně všechny možné odstíny.



### JAK NA TO?

- Určitě jste se již seznámili s příkazem cyklu `for` a podmínkovým příkazem `if`. Oba dva určitě při řešení úkolu využijete.  
→ Uvědomte si, jaké složky barev se musí měnit, aby přechody mezi barvami byly patrné a plynulé.



### JIŽ ZNÁTE, ALE PRO JISTOTU

Do teď se používali pro definici barev pouze krajní hodnoty, tj. 0 a 255, takže jsme si vystačili s funkcí **digitalWrite()**. Ovšem pokud budeme chtít řešit plynulý přechod mezi barvami s využitím PWM, musíme použít funkci



### FUNKCE ANALOGWRITE()

➔ Funkce `analogWrite(pin, hodnota)` posílá analogový signál na uvedený pin ve formě PWM. Při použití této funkce již nepracujeme pouze s krajními hodnotami, ale v plném rozsahu 0-255.

## ŘEŠENÍ ÚLOH

Úkol A)

```
1 void setup() {
2     pinMode(11, OUTPUT);          //červená
3     pinMode(10, OUTPUT);          //zelená
4     pinMode(9, OUTPUT);           //modrá
5 }
6
7 void loop() {
8     // Kód pro zelenou barvu
9     digitalWrite(11, 255);
10    digitalWrite(10, 0);
11    digitalWrite(9, 255);
12    delay(3000);
13    // Kód pro modrou barvu
14    digitalWrite(11, 255);
15    digitalWrite(10, 255);
16    digitalWrite(9, 0);
17    delay(3000);
18    // Kód pro červenou barvu
19    digitalWrite(11, 0);
20    digitalWrite(10, 255);
21    digitalWrite(9, 255);
22    delay(3000);
23 }
```

Úkol B)

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     // Kód pro tyrkysovou barvu
9     digitalWrite(11, 255);
10    digitalWrite(10, 0);
11    digitalWrite(9, 0);
12
13    // Kód pro žlutou barvu
14    digitalWrite(11, 0);
15    digitalWrite(10, 0);
16    digitalWrite(9, 255);
17
18    // Kód pro fialovou barvu
19    digitalWrite(11, 0);
20    digitalWrite(10, 255);
21    digitalWrite(9, 0);
22}
23
```

Úkol C)

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     //tyrkysová barva
9     setColor(255,0,0);
10    delay(1000);
11    setColor(255,0,0);
12    delay(1000);
13    setColor(255,0,0);
14    delay(3000);
15    //žlutá barva
16    setColor(0,0,255);
17    delay(1000);
18    setColor(0,0,255);
19    delay(1000);
20    setColor(0,0,255);
21    delay(3000);
22    //fialová barva
23    setColor(0,255,0);
24    delay(1000);
25    setColor(0,255,0);
26    delay(1000);
27    setColor(0,255,0);
28    delay(3000);
29 }
30
31 void setColor(int redC, int greenC, int blueC ) {
32     digitalWrite(11, redC);
33     digitalWrite(10, greenC);
34     digitalWrite(9, blueC);
35 }
```

Úkol D)

```
1 const int redPin = 11;
2 const int greenPin = 10;
3 const int bluePin = 9;
4
5 int redIntens;
6 int greenIntens;
7 int blueIntens;
8
9 int x;
10
11 int display_time = 10;
12 int common_anode=1;
13
14 void setup(){
15     pinMode(redPin, OUTPUT);
16     pinMode(greenPin, OUTPUT);
17     pinMode(bluePin, OUTPUT);
18 }
19
20 void loop(){
21     for (x = 0; x < 767; x++){
22
23         if(x <= 255){
24             redIntens = 255 - x;
25             greenIntens = x;
26             blueIntens = 0;
27         }else if (x <= 511){
28             redIntens = 0;
29             greenIntens = 255 - (x - 256);
30             blueIntens = (x - 256);
31         }else{
32             redIntens = (x - 512);
33             greenIntens = 0;
34             blueIntens = 255 - (x - 512);
35         }
36
37         setColor(redIntens, blueIntens, greenIntens);
38         delay(display_time);
39     }
40 }
41 void setColor(int redC, int greenC, int blueC){
42     if(common_anode==1){
43         redC=255-redC;
44         greenC=255-greenC;
45         blueC=255-blueC;
46     }
47 }
```

```
49 |     analogWrite (redPin, redC);
50 |     analogWrite (greenPin, greenC);
51 |     analogWrite (bluePin, blueC);
52 | }
```

# PODROBNÝ PRŮVODCE TEORIÍ

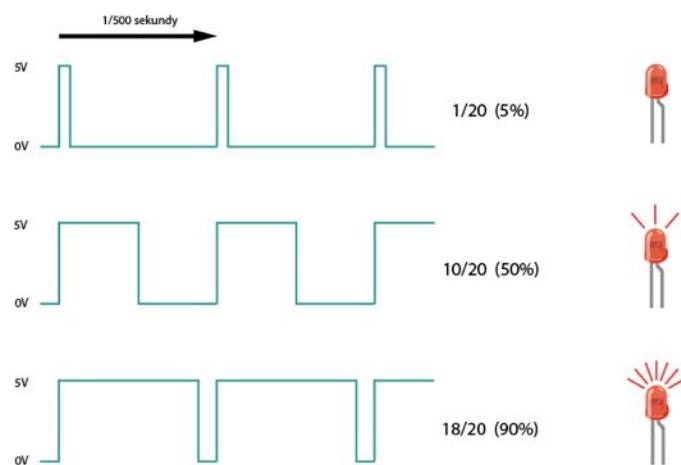
POUŽITÍ TŘÍBAREVNÉ DIODY LZE VYUŽÍT NAPŘÍKLAD K VYTVOŘENÍ LAMPY, KTERÁ BUDE PLYNULE MĚNIT BARVU.

## OBSAH PRŮVODCE

- ① Seznámení se s RGB diodou a její funkčností.
- ② Princip skládání barev a PWM.
- ③ Správné zapojení RGB diody s využitím zkušeností z předchozích příkladů. Zejména se jedná o volbu rezistorů a jejich zapojení.
- ④ Zavedení pojmu a znalostí týkající se analogových výstupů a mapování hodnot.
- ⑤ Využití již zavedeného podmínkového příkazu **if**.
- ⑥ Zavedení vlastní funkce v programovém kódu.
- ⑦ Použití příkazu cyklu **for**.

## PWM

Arduino má na svých výstupech napětí 5V. Toto napětí nelze souvisle měnit. Jak tedy ovládat například jas diody? K tomu se využívá technika, která se nazývá **Pulsně Šířková Modulace**. Z anglického Pulse Width Modulation se používá všeobecně známá zkratka **PWM**. Při ovládání jasu diody PWM velmi rychle přepíná na výstupu pinů hodnoty s logickou nulou (0V) a logickou jedničkou (+5V). To se děje v určitém čase. Tyto změny jsou tak rychlé, že je lidské oko nedokáže díky své setrvačnosti zachytit. Z poměru času, ve kterém je na výstupu +5V ku stavu 0V se pak odvíjí intenzita svícení LED diody nebo rychlosť otáčení motoru – Obr. 1.



Obr. 1 – Pulsně šířková modulace

Střídavé nastavování jedniček a nul pomocí programu, by poměrně hodně zaměstnalo procesor, a navíc by se nedosáhlo dostatečně vysokého frekvence spínání. Arduino ale poskytuje možnost využít modulu čítačů, které umožňují generování PWM hardwarově. Stačí vhodně nastavit čítač do režimu PWM, určit periodu a plnění. Na výstupu je pak trvale



generován signál s pulzně šířkovou modulací automaticky.

Jediným omezením je to, že nelze použít libovolného piny, ale pouze těch, na které je výstup čítačů přiveden. Arduino UNO má šest pinů určených pro PWM - **3, 5, 6, 9, 10 a 11**.

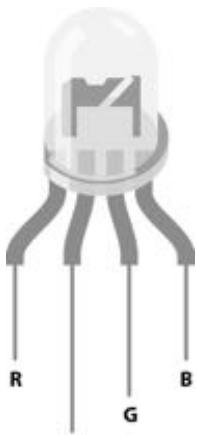
Lze je rozpoznat symbolem ~ před číslem.

V samotném programu se nastavuje pouze plnění, což je poměr doby zapnutí a celé periody. Takže rozsah 0-255, odpovídá poměru 0-100%.

## LED DIODA RGB

V předchozích příkladech jste získali zkušenosti se zapojením a chováním obyčejných LED diod. RGB dioda pracuje obdobně. Její označení RGB je odvozeno od anglického pojmenování barev – červená, zelená, modrá. Tato LED dioda ve skutečnosti představuje tři LED diody v jednom balení: jedna je červená, jedna zelená a jedna modrá.

Existují dva druhy RGB diod. Dioda se **společnou anodou** a se **společnou katodou**. Anoda je pozitivní kontakt a katoda je negativní kontakt. V závislosti na druhu RGB diody se liší i její zapojení, i když pro Arduino to nemá prakticky žádný význam, protože obě využívají stejné napětí.

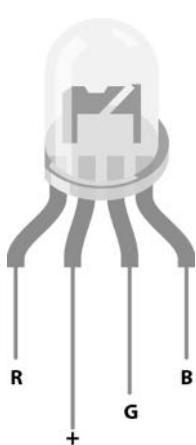


Obr. 2 – RGB dioda se společnou katodou

RGB dioda se **společnou katodou** má vlastní kladný (anodový) vývod, ale všechny sdílejí jeden záporný (katodový) vývod. RGB diody, mají vývody o různé délce. Při běžných LED diodách je krátký vývod negativní, ale u RGB diody se společnou katodou je negativní vývod ten nejdelší.

RGB dioda se **společnou anodou** má vlastní záporný (katodový) vývod, ale všechny sdílejí jeden kladný (anodový) vývod. U této diody je nejdelší vývod kladný.

Ostatní kratší vývody mají označení podle barev a u obou typů RGB diod jsou stejné. Liší se pouze jejich zapojení do obvodu. Vzhledem k tomu, že každý kratší vývod odpovídá konkrétní barvě, lze RGB diodu zapojit do obvodu, stejně jako by to byly tři samostatné LED diody.

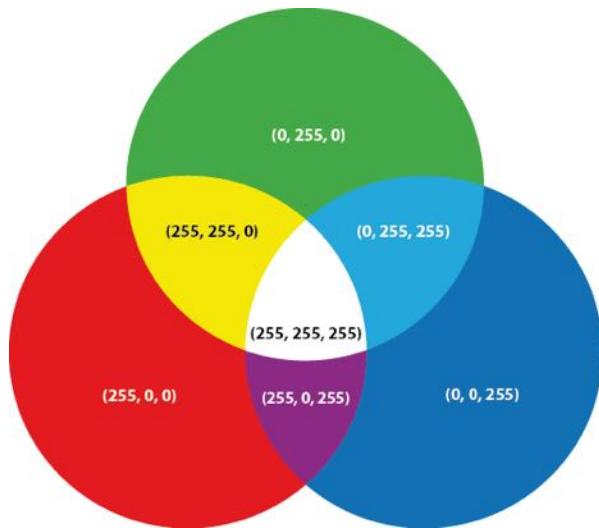


Obr. 3 - RGB dioda se společnou anodou

Pro RGB diody se **společnou katodou** se vývody R, G, B připojí k napájení nebo na výstupní pin Arduina. Před kladné vývody se ještě zapojují rezistory, které zajistí, aby nedošlo k poškození RGB diody.

Pro RGB diody se **společnou anodou** se vývody R, G, B připojí přímo k výstupním pinům Arduina a anoda se připojí na napájení s předřadným rezistorem.

RGB diody jsou skvělé, protože je lze využít k zobrazení pestrých barev. Červená, zelená a modrá jsou základní barvy a v přídavné barevné paletě je lze kombinovat a vytvářet další odstíny. Možnosti kombinace barev jsou uvedeny na obrázku Obr. 4 - Kombinace barev.



Obr. 4 - Kombinace barev

Na obrázku je také vidět, že kombinace barev je dána trojicí hodnot 0, 255. S ohledem na Arduino a PWM lze tyto limitní hodnoty přiřadit logickým úrovním 0 – **LOW** a 255 – **HIGH**. Z obrázku Obr. 4 - Kombinace barev je patrné, že pro zelenou barvu by kombinace hodnot vypadala: **[LOW, HIGH, LOW]**. To platí pro diody se společnou katodou. Pro diodu se společnou anodou by hodnoty byly opačně: **[HIGH, LOW, HIGH]**. Kombinace pro jednotlivé barvy jsou uvedeny v tabulce Tab 1 - Kombinace barev.

	<b>Společná katoda</b>	<b>Společná anoda</b>
<b>Červená</b>	<b>HIGH, LOW, LOW</b>	LOW, <b>HIGH, HIGH</b>
<b>Modrá</b>	LOW, LOW, <b>HIGH</b>	<b>HIGH, HIGH</b> , LOW
<b>Zelená</b>	LOW, <b>HIGH</b> , LOW	<b>HIGH</b> , LOW, <b>HIGH</b>

Tab 1 - Kombinace barev



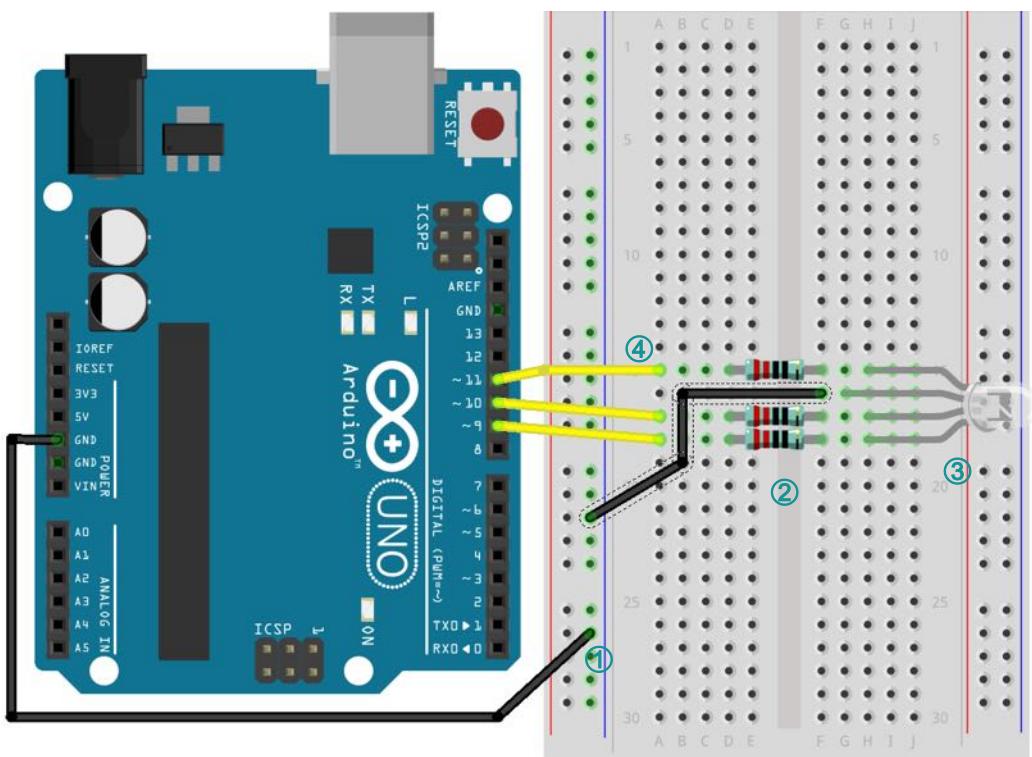
Pokud se budou požívat pouze krajní hodnoty pro definici barev, tj. 0 a 255, potom si vystačíme s funkcí **digitalWrite()**. Ovšem pokud budeme chtít řešit plynulý přechod mezi barvami s využitím PWM, musíme použít funkci **analogWrite()**.



#### FUNKCE ANALOGWRITE()

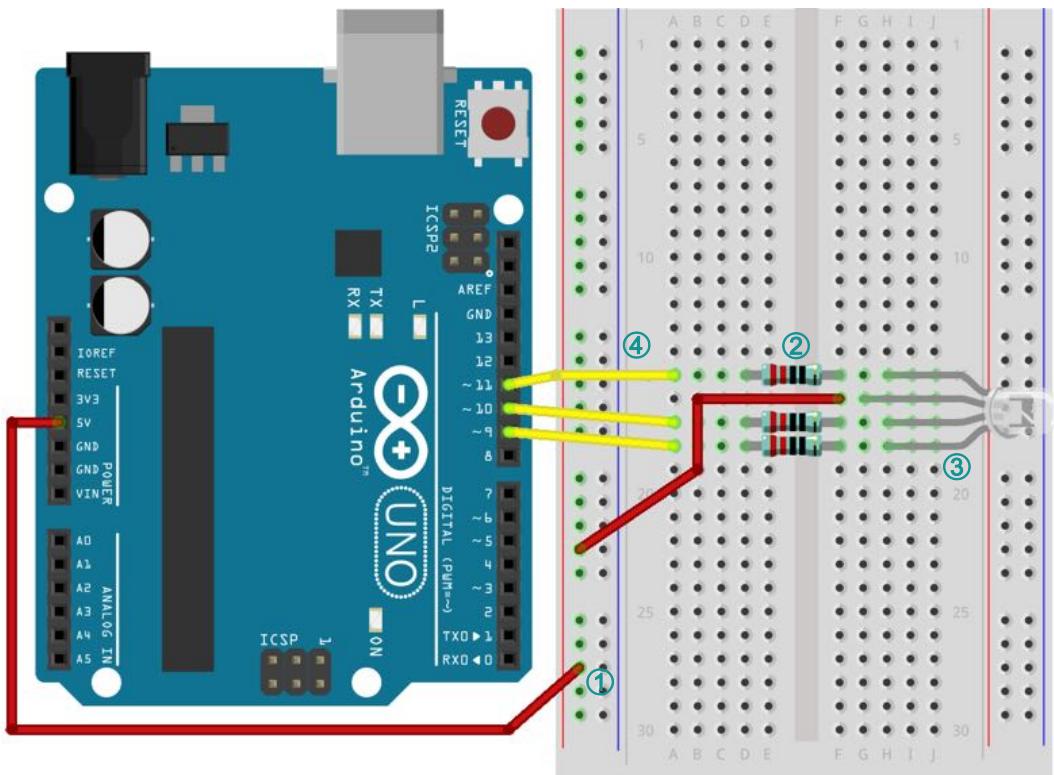
- ➔ Funkce `analogWrite(pin, hodnota)` posílá analogový signál na uvedený pin ve formě PWM. Při použití této funkce již nepracujeme pouze s krajními hodnotami, ale v plném rozsahu 0-255.

## ZAPOJENÍ OBVODU



Obr. 5 - Zapojení RGB diody se společnou katodou

- ① Vodič napájení z Arduino zapojte do kontaktní desky k **modré** čáře. Tím se zpřehlední celé zapojení.
- ② Najděte tři rezistory o hodnotě  $220\Omega$  (dva oranžové proužky, modrý, černý a zlatý). Je důležité, aby tyto rezistory byly zapojeny do kontaktní desky tak, aby propojovaly její obě poloviny. Pokud by tomu tak nebylo, došlo by ke zkratování. Rezistory zajistí snížení proudu, aby nedošlo ke zničení RGB diody.
- ③ RGB diodu zapojte do kontaktní desky orientovanou tak, jak je uvedeno na obrázku Obr. 6 - Zapojení RGB diody se společnou katodou. Nejdélší vývod je propojen s **uzemněním**. Zbylé vodiče jsou propojeny s rezistory.
- ④ Druhá strana rezistorů je zapojena do digitálních pinů 9, 10 a 11.



Obr. 6 - Zapojení RGB diody se společnou anodou

- ① Vodič napájení 5V z Arduino zapojte do kontaktní desky k **červené** čáře.
- ② Najděte rezistory o hodnotě  $220\Omega$  (dva oranžové proužky, modrý, černý a zlatý). Je důležité, aby tyto rezistor byly zapojeny do kontaktní desky tak, aby propojovaly její obě poloviny. Pokud by tomu tak nebylo, došlo by ke zkratování. Rezistory opět zajistí snížení proudu, aby nedošlo ke zničení RGB diody.
- ③ RGB diodu zapojte do kontaktní desky orientovanou tak, jak je uvedeno na obrázku Obr. 6 - Zapojení RGB diody se společnou anodou. Nejdelší vývod je propojen s **napájením** z desky Arduino.
- ④ Zbylé vodiče jsou připojeny do digitálních pinů 9, 10 a 11.

Lze říci, že na vývody RGB diody – anody i katody, bude zasílán signál PWM. Podle hodnoty na jednotlivých vývodech tohoto signálu se mění barva RGB diody.

## PROGRAMOVÝ KÓD

Při psaní kódu pro všechny příklady si vystačíme se základním zapojením podle schématu na Obr. 2 – RGB dioda se společnou katodou nebo Obr. 3 - RGB dioda se společnou anodou.



Příklady na sebe navazují, takže pro jejich inovace stačí provádět pouze dílčí úpravy programového kódu. Není nutné vždy psát celý program od začátku.

## ZÁKLADNÍ PŘÍKLAD

Zde je uveden základní programový kód, který zobrazí na RGB diodě se společnou anodou modrou barvu.

```
1 void setup() {  
2     pinMode(11, OUTPUT);          //červená  
3     pinMode(10, OUTPUT);          //zelená  
4     pinMode(9, OUTPUT);           //modrá  
5 }  
6  
7 void loop() {  
8     digitalWrite(11, HIGH);        //červená  
9     digitalWrite(10, HIGH);        //zelená  
10    digitalWrite(9, LOW);         //modrá  
11 }
```



- ① Každý pin, který ovládá RGB diodu, je definován funkcí **pinMode()** pro nastavení výstupu. V příkladu jsou definovány piny **11, 10, 9**.
- ② Podobně jako u jednobarevné LED diody i v případě RGB se využívá funkce **digitalWrite()**. Tentokrát se používá pro každou barvu (vývod) RGB diody. Pro zobrazení modré barvy jsou zasílány signály do vývodu pro červenou a zelenou barvu. Vzhledem k tomu, že se jedná o diodu se společnou anodou jsou hodnoty **HIGH** a **LOW** zadány inverzně, ve srovnání s Obr. 4 - Kombinace barev.



(Př. 1) Změňte logické hodnoty ve funkci `digitalWrite()` tak, aby RGB dioda svítila zeleně, modře a červeně.



### NESVÍTÍ DIODA

**Zapojení RGB diody** – zkонтrolujte její zapojení v kontaktním poli. Ujistěte se, zda se jedná o diodu se společnou anodou nebo katodou. Zapojení se tím významně liší.

**Zapojení v desce** – zkонтrolujte, zda jsou vývody zapojeny do odpovídajících pinů desky Arduino. Musí být využity piny, které poskytují PWM, tedy piny, které mají u svého čísla symbol `~`.

**Rezistory** – zkонтrolujte, zda jste použili správné rezistory.

### NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

## ZÁKLADNÍ PŘÍKLAD – INOVACE S PWM

Nahrazení logických hodnot **HIGH** a **LOW** konkrétními číselnými hodnotami z rozsahu PWM.

```
1 void setup() {  
2     pinMode(11, OUTPUT);          //červená  
3     pinMode(10, OUTPUT);          //zelená  
4     pinMode(9, OUTPUT);           //modrá  
5 }  
6  
7 void loop() {  
8     digitalWrite(11, 255);        //červená  
9     digitalWrite(10, 255);        //zelená  
10    digitalWrite(9, 0);           //modrá  
11 }
```



Příklad je totožný se základní variantou. Rozdíl je v nahrazení logických hodnot **HIGH** a **LOW** číselnými hodnotami **255** a **0**. Výhodou této varianty je možnost definovat celou škálu barev v závislosti na rozsahu hodnot **0-255**.



(Př. 2) Změňte hodnoty ve funkci **digitalWrite()** tak, aby RGB dioda svítila tyrkysově, žlutě a fialově. Vyzkoušejte, jak se bude měnit barva se změnou hodnot v uvedené funkci.



Pro definici barvy v hodnotách RGB lze využít některý z grafických programů pro úpravu fotografií nebo jednoduchý nástroj, který je na adrese: <http://www.colorpicker.com>. Musí být ale dodržen princip skládání barev pro RGB diodu. K pochopení lze využít animaci:

<https://milannovak.gitbooks.io/arduino/content/rgb-led-dioda/teorie.html>

## ZÁKLADNÍ PŘÍKLAD – PRŮBĚŽNÁ ZMĚNA BAREV

Příklad ukazuje, jak průběžně měnit barvy RGB diody v nekonečné smyčce. K tomu se využívá funkce `delay()`. Jediným parametrem je doba, na kterou se běh programu pozastaví. Doba se uvádí v milisekundách, tzn. 1000ms = 1s.

```
1 void setup() {
2     pinMode(11, OUTPUT);          //červená
3     pinMode(10, OUTPUT);          //zelená
4     pinMode(9, OUTPUT);          //modrá
5 }
6
7 void loop() {
8     // Kód pro zelenou barvu
9     digitalWrite(11, 0);
10    digitalWrite(10, 255);
11    digitalWrite(9, 255);
12    delay(3000);                └─①
13    // Kód pro modrou barvu
14    digitalWrite(11, 255);
15    digitalWrite(10, 0);
16    digitalWrite(9, 255);
17    delay(3000);                └─①
18    // Kód pro červenou barvu
19    digitalWrite(11, 255);
20    digitalWrite(10, 0);
21    digitalWrite(9, 255);
22    delay(3000);                └─①
23 }
```

- ① Každý blok příkazů, které zajišťují rozsvícení diody v konkrétní barvě, je oddělen funkcí `delay(3000)`. Tím je zajištěno, že každá barva RGB bude svítit 3 sekundy.



(Př. 3) Přidejte další tři barvy tak, aby každá třikrát blikla a pak se zobrazila barva další. Interval blikání nastavte na 1 sekundu. Prodleva přechodu mezi barvami bude 3 sekundy.



Kombinace číselných hodnot PWM pro definici barvy, je dána tím, zda se používá dioda se společnou anodou nebo katodou. V uvedeném příkladu se barvy definují pro RGB diody se **společnou anodou**. Pro diodu se **společnou katodou** budou hodnoty opačné.

## ZÁKLADNÍ PŘÍKLAD – FUNKCE PRO ZOBRAZENÍ BARVY

V předchozích příkladech jsme si vystačili se základní strukturou programového kódu. Vše fungovalo, jak mělo, ale při složitějších úlohách začíná být programový kód nepřehledný. K zpřehlednění nám opět pomůže využití vlastní **funkce**.

Zavedeme funkci **void setColor(int redC, int greenC, int blueC)**. Tato funkce má tři parametry, kterými budou čísla, která charakterizuje slovo **int**, což je opět datový typ (integer).

```
1 void setup() {  
2     pinMode(11, OUTPUT);          //červená  
3     pinMode(10, OUTPUT);          //zelená  
4     pinMode(9, OUTPUT);           //modrá  
5 }  
6  
7 void loop() {  
8     setColor(255,0,255);         //zelená barva  
9     delay(2000);  
10    setColor(255,255,0);          //modrá barva  
11    delay(2000);  
12 }  
13  
14 void setColor(int redC, int greenC, int blueC ) {  
15     digitalWrite(11, redC);  
16     digitalWrite(10, greenC);  
17     digitalWrite(9, blueC);  
18 }
```



- ① Pro složení hodnot konkrétní barvy je deklarovaná funkce **setColor** se třemi parametry.
- ② Deklarovaná funkce **setColor** se volá ve smyčce (mj. také ve funkci **loop**) s číselnými hodnotami konkrétní barvy jako jejími parametry. Funkce se volá s pauzami 2 sekundy.



(Př. 4) Přidejte další tři barvy tak, aby každá třikrát blikla a pak se zobrazila barva další. Interval blikání nastavte na 1 sekundu. Prodleva přechodu mezi barvami bude 3 sekundy.

Na rozdíl od předchozího příkladu, tentokrát použijte funkci pro zobrazení barvy.



(Př. 5) Neustále řešíme, jak bude kombinace barev vypadat pro RGB diodu se společnou anodou a katodou. Upravte programový kód tak, aby se zadávaly hodnoty stejně pro jakýkoliv typ diody. Využijte k tomu podmínkový příkaz **if**.

## PROJEKT - MAGICKÁ LAMPA

Tento projekt využije všechny poznatky z předchozích příkladů. Využije se zde i cyklu **for**, díky kterému bude magická lampa plynule měnit barvy v jednotlivých odstínech.

```
1 const int redPin = 11;          | ①
2 const int greenPin = 10;        | ②
3 const int bluePin = 9;         |
4
5 int redIntens;                | ③
6 int greenIntens;              | ④
7 int blueIntens;              |
8
9 int x;                         | ⑤
10
11 int display_time = 10;         | ⑥
12 int common_anode=1;
13
14 void setup(){                  | ⑦
15     pinMode(redPin, OUTPUT);   |
16     pinMode(greenPin, OUTPUT); |
17     pinMode(bluePin, OUTPUT);  |
18 }
19
20 void loop(){                   | ⑧
21     for (x = 0; x < 767; x++){|
22
23         if(x <= 255){          | ⑨
24             redIntens = 255 - x;|
25             greenIntens = x;    |
26             blueIntens = 0;     |
27         }else if (x <= 511){    | ⑩
28             redIntens = 0;      |
29             greenIntens = 255 - (x - 256);|
30             blueIntens = (x - 256);|
31         }else{                  |
32             redIntens = (x - 512);|
33             greenIntens = 0;    |
34             blueIntens = 255 - (x - 512);|
35         }
36
37     setColor(redIntens, blueIntens, greenIntens);|
38     delay(display_time);|
39 }
40 }
```

```

43 void setColor(int redC, int greenC, int blueC){
44     if(common_anode==1){
45         redC=255-redC;
46         greenC=255-greenC;
47         blueC=255-blueC;
48     }
49     analogWrite (redPin, redC);
50     analogWrite (greenPin, greenC);
51     analogWrite (bluePin, blueC);
}

```

⑪

⑫

- ① Definice pinů pro připojení RGB diody.
- ② Deklarace proměnných **redIntens**, **greenIntens**, **blueIntens**, které budou obsahovat PWM hodnoty pro zobrazení konkrétní barvy.
- ③ Deklarace proměnné **x** pro kroky cyklu **for**.
- ④ Deklarace proměnné **display\_time**, kde je uložen údaj o prodlevě při plynulé změně barev diody. Proměnná **common\_anode** zajišťuje informaci, zda se používá dioda se společnou anodou (hodnota 1) nebo katodou (hodnota jiná než 1).
- ⑤ Vyhrazení použitých pinů na desce Arduino. Hodnoty jsou definovány v ①.
- ⑥ Začátek cyklu **for**. Cyklus **for** bude probíhat od 0 do 767. Hodnota 767 vychází z limitní hodnoty pro každou barvu, která je 255. Objasnění vyplýne z níže uvedeného popisu podmínky **if**.
- ⑦ První část podmínky zajistí, že výchozí barvou bude **červená**. Když si za proměnou x v prvním kroku cyklu dosadíme konkrétní hodnotu 0, získáme kombinaci PWM hodnot **redIntens=255-0; greenIntens=0; blueIntens=0;** což je červená barva.
- ⑧ Druhá část podmínky sleduje proměnnou od x=256. Opět, když dosadíme konkrétní hodnoty v prvním kroku cyklu, vyjde nám následující: **redIntens=0; greenIntens=255-(256-256); blueIntens=(256-256);** což je kombinace 0, 255, 0 a tedy zelená barva.
- ⑨ Třetí část podmínky pracuje s hodnotami vyššími jak 511. Po dosazení konkrétních hodnot: **redIntens=(512-512); greenIntens=0; blueIntens=255-(512-512);** vyjde kombinace 0, 0, 255 tj. modrá barva. Vzhledem k průběžné změně hodnot proměnné x dochází k plynulému přechodu mezi barvami.

- ⑩ Volání vlastní funkce pro zobrazení konkrétní barvy, se třemi parametry, jejichž hodnoty jsou průběžně počítány v bloku podmínkového příkazu **if**. Aby se barva stihla zobrazit, je provedena prodleva pomocí **delay**.
- ⑪ V závislosti na tom, jestli se používá dioda se společnou anodou nebo katodou, se upraví výsledné hodnoty barevných kombinací.
- ⑫ Samotné zobrazení konkrétní barvy.



Nyní, když už je připravena programová část a je plně funkční, je čas si projekt ukázat v praktickém „balení“. Podpořte svou kreativitu a vytvořte jednoduché stínidlo pro magickou lampu. Návod je uveden v následující kapitole.

## STÍNÍTKO PRO MAGICKOU LAMPU

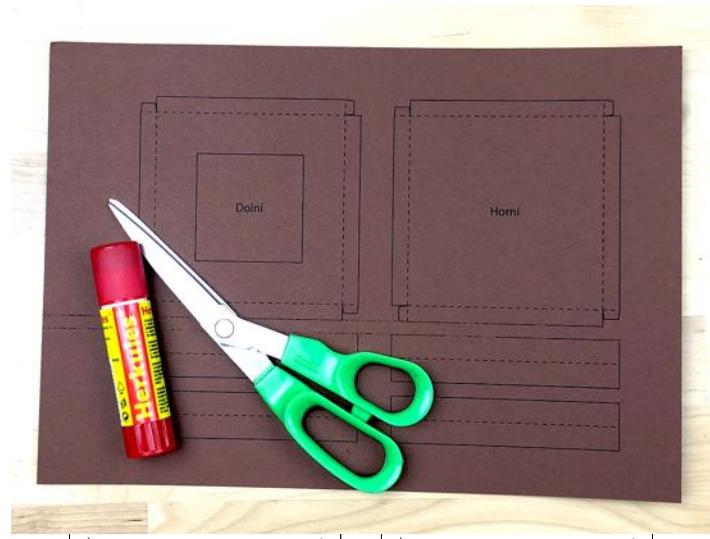
Pro vytvoření stínítka budeme potřebovat: tmavý karton (tvrdší papír), pauzák (průsvitný papír), lepidlo, nůžky.

### PAPÍROVÁ KONSTRUKCE

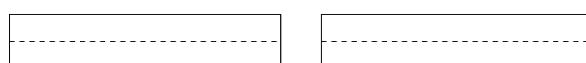
V této kapitole je ukázka základní papírové konstrukce, kterou můžete začít. Doporučujeme, abyste si později vzhled přizpůsobili podle vlastních potřeb a nápadů. Pokud máte již svoji jasnou představu jak by lampa mohla vypadat, nemusíte využít přiložených šablon.

Na tiskárně si vytiskněte přiloženou šablonu na pevný papír, podle obrázku Obr. 5 - Šablona stínítka. Stínítko bude mít tvar krychle.

V první řadě vystříhněte jednotlivé díly z vytisknuté šablony Obr. 6 - Vystřížení dílů ze šablony. Připravte si průsvitný papír (pauzák), který později upravíte pro velikost, která bude odpovídat složené konstrukci.



Obr. 6 - Vystřížení dílů ze šablony



Obr. 5 - Šablona stínítka

## SLOŽENÍ DÍLŮ

Připravte si vyštířené části konstrukce stínítka. Měli byste mít horní a dolní díl, sloupky a samozřejmě průsvitný papír Obr. 7 - Díly stínítka.

Nejprve připravte dolní díl, tj. ten s otvorem. Podle přerušovaných čar ohněte okraje do pravého úhlu. Do vnitřní části základny ohněte i spojovací západky, na které naneste trochu lepidla, jak je znázorněno na obrázku Obr. 8 - Složení základny. Stejný postup aplikujte i pro složení horního dílu.



Obr. 7 - Díly stínítka



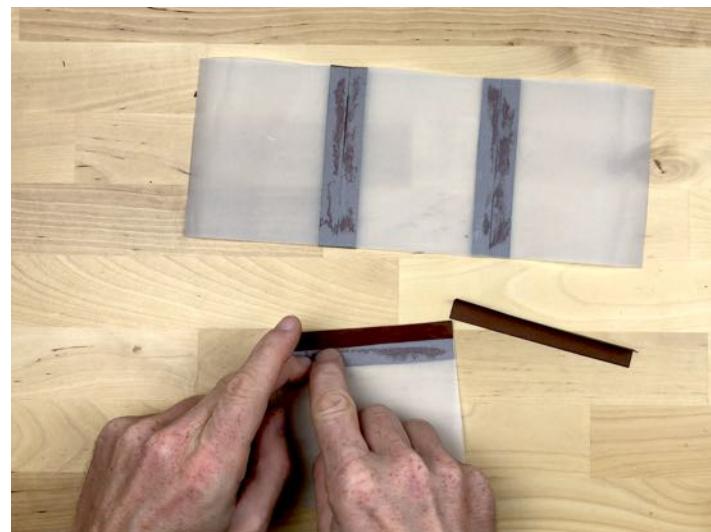
Obr. 8 - Složení základny



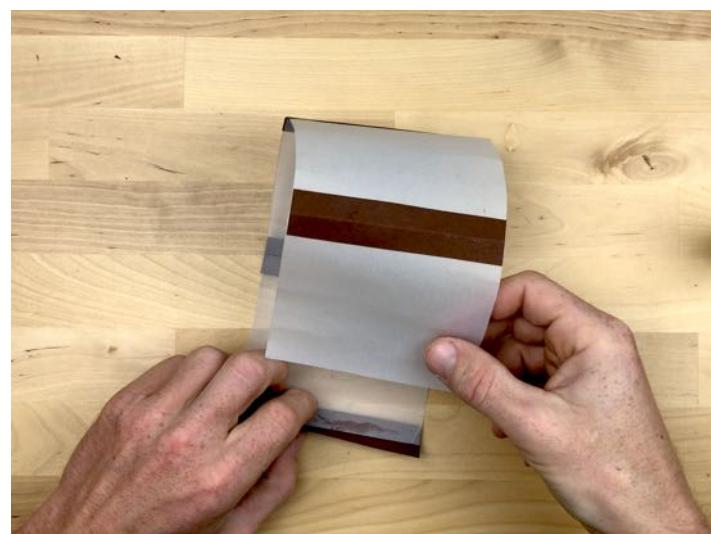
Obr. 9 - Složené díly

Dále naohýbejte všechny postranní díly. Ohnutí je vždy provedeno podle přerušované čáry vedené středem dílů. Jakmile máte složené všechny díly, mělo by všech šest částí vypadat tak jako na obrázku Obr. 9 - Složené díly.

K naohýbaným rohům přilepte průhledný papír (pauzák) tak, jak je vidět na obrázku Obr. 10 - Lepení bočních rohů



Obr. 10 - Lepení bočních rohů



Obr. 11 - Lepení bočních stěn

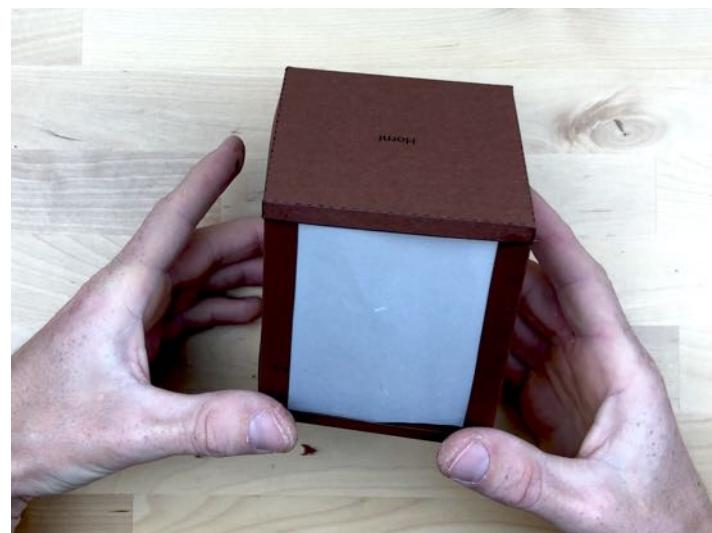
Když jsou nalepeny všechny boční stěny, spojte poslední díl s posledním rohem, jak je vidět na obrázku Obr. 11 - Lepení bočních stěn.

Všechny stěny vložte do podstavy. Podstavu můžete mírně namazat lepidlem a stěny k podstavě přilepte.



Obr. 12 - Přiložení bočních stěn

Následně přidejte horní díl. Naneste trochu lepidla do rohů horního dílu a jednoduše jej přiložte na konstrukci stínítka viz. Obr. 13 - Přiložení horního dílu.



Obr. 13 - Přiložení horního dílu

Kompletní stínítko je zobrazeno na obrázku Obr. 14 - Hotové stínítko. Nyní stačí stínítko přiložit na nepájivé pole s RGB diodou a spustit program.



Obr. 14 - Hotové stínítko

## ŘEŠENÍ ÚLOH

## (PŘ. 1

Kombinace hodnot pro jednotlivé barvy pro RGB diodu se **společnou anodou**. Pro RGB diodu se **společnou katodou** budou hodnoty **HIGH** a **LOW** opačně. Jednotlivé bloky kódu pro zelenou, modrou a červenou barvu se vkládají zvlášť. Po každou barvu se kód nahraje do desky a dioda by měla svítit definovanou barvou. Následně se změní kombinace hodnot **HIGH** a **LOW** a opět se kód nahraje do desky.

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     // Kód pro zelenou barvu
9     digitalWrite(11, HIGH);
10    digitalWrite(10, LOW);
11    digitalWrite(9, HIGH);
12
13    // Kód pro modrou barvu
14    digitalWrite(11, HIGH);
15    digitalWrite(10, HIGH);
16    digitalWrite(9, LOW);
17
18    // Kód pro červenou barvu
19    digitalWrite(11, LOW);
20    digitalWrite(10, HIGH);
21    digitalWrite(9, HIGH);
22 }
```

## (PŘ. 2

Kombinace hodnot PWM pro jednotlivé barvy pro RGB diodu se společnou anodou. Pro každou barvu se opět nahraje kód do desky zvlášť.

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     // Kód pro tyrkysovou barvu
9     digitalWrite(11, 255);
10    digitalWrite(10, 0);
11    digitalWrite(9, 0);
12
13    // Kód pro žlutou barvu
14    digitalWrite(11, 0);
15    digitalWrite(10, 0);
16    digitalWrite(9, 255);
17
18    // Kód pro fialovou barvu
19    digitalWrite(11, 0);
20    digitalWrite(10, 255);
21    digitalWrite(9, 0);
22 }
```

### (PŘ. 3

Řešení je velice jednoduché a vede spíše k upevnění již naučeného, ale je ukázkou dlouhého a neefektivního programového kódu, pokud se nevyužije programových struktur, jakými jsou např. funkce.

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     // Kód pro zelenou barvu
9     digitalWrite(11, 255);
10    digitalWrite(10, 0);
11    digitalWrite(9, 255);
12    delay(1000);
13    digitalWrite(11, 255);
14    digitalWrite(10, 0);
15    digitalWrite(9, 255);
16    delay(1000);
17    digitalWrite(11, 255);
18    digitalWrite(10, 0);
19    digitalWrite(9, 255);
20    delay(3000);
21
22     // Kód pro modrou barvu
23     digitalWrite(11, 255);
24     digitalWrite(10, 255);
25     digitalWrite(9, 0);
26     delay(3000);
27     digitalWrite(11, 255);
28     digitalWrite(10, 255);
29     digitalWrite(9, 0);
30     delay(3000);
31     digitalWrite(11, 255);
32     digitalWrite(10, 255);
33     digitalWrite(9, 0);
34     delay(3000);
35     // Výše uvedené řádky kódu opakujeme pro další
36     // barvy. Dioda třikrát blikne a změní
37     // se barva.
38 }
39
```

#### (PŘ. 4

Řešení s využitím vlastní funkce poskytuje značné zjednodušení a zpřehlednění programového kódu.

```
1 void setup() {
2     pinMode(11, OUTPUT);
3     pinMode(10, OUTPUT);
4     pinMode(9, OUTPUT);
5 }
6
7 void loop() {
8     //zelená barva
9     setColor(255,0,255);
10    delay(1000);
11    setColor(255,0,255);
12    delay(1000);
13    setColor(255,0,255);
14    delay(3000);
15    //modrá barva
16    setColor(255,255,0);
17    delay(1000);
18    setColor(255,255,0);
19    delay(1000);
20    setColor(255,255,0);
21    delay(3000);
22    //červená barva
23    setColor(0,255,255);
24    delay(1000);
25    setColor(0,255,255);
26    delay(1000);
27    setColor(0,255,255);
28    delay(3000);
29
30    // Výše uvedené řádky kódu opakujeme pro další
31    // barvy. Dioda třikrát blikne a změní
32    // se barva.
33 }
34
35 void setColor(int redC, int greenC, int blueC ) {
36     digitalWrite(11, redC);
37     digitalWrite(10, greenC);
38     digitalWrite(9, blueC);
39 }
```

## (PŘ. 5

Příklad využívá podmínkového příkazu **if**, který testuje hodnotu proměnné **common\_anode**. Pokud bude hodnota proměnné **1**, bude se jednat o zapojení RGB diody se společnou anodou a tím se zadávané hodnoty pro definici barvy odečtou od čísla **255**. V opačném případě, pokud bude hodnota proměnné rovna jiné hodnotě než **1**, budou zadávané hodnoty ponechány.

```
1 int common_anode=1;           //pokud se bude jednat o
2                                         //diodu se společnou
3                                         //anodou, hodnota proměnné
4                                         //bude rovna 1, v opačném
5                                         //případě změňte na jinou
6
7 void setup() {
8     pinMode(11, OUTPUT);
9     pinMode(10, OUTPUT);
10    pinMode(9, OUTPUT);
11 }
12
13 void loop() {
14     setColor(255,0,0);          //červená barva
15     delay(2000);
16     setColor(0,0,255);         //modrá barva
17     delay(2000);
18 }
19
20 void setColor(int redC, int greenC, int blueC ){
21     if(common_anode==1){
22         redC=255-redC;
23         greenC=255-redC;
24         blueC=255-redC;
25     }
26
27     digitalWrite(11, redC);
28     digitalWrite(10, greenC);
29     digitalWrite(9, blueC);
}
```

## 5. STEJNOSMĚRNÝ MOTOR

SERVOMOTORY JIŽ ZNÁME, ALE V ROBOTICE NEJSOU JEDINÝMI POHONNÝMI PROSTŘEDKY. JEDNODUŠŠÍ JSOU MOTORY STEJNOSMĚRNÉ, KTERÉ V TÉTO ČÁSTI VYUŽIJEME PRO VENTILÁTOR S PLYNULOU REGULACÍ OTÁČEK.

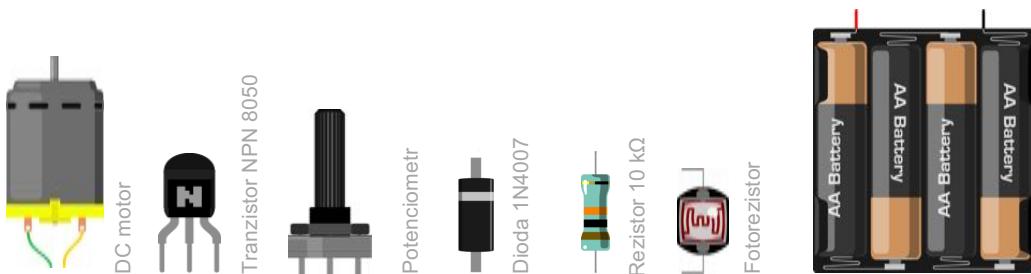
### CÍLE

- ① Pochopení principu stejnosměrného motoru.
- ② Seznámení s principy tranzistoru.
- ③ Zapojení tranzistoru pro regulaci otáček motoru.
- ④ Zapojení externího napájení motoru.
- ⑤ Využití potenciometru pro regulaci motoru.
- ⑥ Projekt větráku.

Čas: **45 min**

Úroveň: ■ ■ ■ ■ ■

Vychází z: **1, 2**



POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví obvod, ve kterém bude zapojen stejnosměrný motor. Na tomto obvodu jim bude vysvětlen princip regulace otáček motoru a jeho programování. Tento obvod dále rozšíří o regulaci pomocí potenciometru a v závěrečném projektu použijí fototranzistor. Seznámí se a využijí analogových vstupů desky Arduino pro čtení hodnot potenciometru a fotorezistoru.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, DC motor, potenciometr, usměrňovací dioda, 2x fotorezistor, vodiče typu zástrčka-zástrčka, externí zdroj napájení.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 5.
- ⑤ Pracovní listy pro studenty.

## 1. KROK 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude se naučit ovládat stejnosměrný motor.

### ZEPTEJTE SE STUDENTŮ

➔ **Věděli byste na jakém principu pracuje stejnosměrný motor?**

Základním principem, který se uplatňuje pro změnu stejnosměrného proudu na mechanickou energii je indukce.



Studenti ať zapojí stejnosměrný motor podle zobrazeného schématu, který je součástí pracovních listů, nebo přiložené prezentace, kterou lze promítat pomocí dataprojektoru.



#### UPOZORNĚNÍ

- Upozorněte studenty na externí zdroj napájení, který je v tomto příkladu zajištěn zásobníkem na baterie. Ze sady lze využít i napájecí modul, ke kterému je připojen adaptér.
- Rekněte studentům, proč je externí zdroj použitý.



#### ZEPTEJTE SE STUDENTŮ

- Jakou funkci zastává usměrňovací dioda v obvodu?  
Chrání obvod proti zpětnému proudu.



#### RYCHLÝ TIP

- Vysvětlete, proč je v obvodu použitý tranzistor. Pro vysvětlení použijte schéma v prezentaci nebo pracovním listu.

## 2. KROK 10 minut

Nyní studentům ukažte prostřednictvím dataprojektoru nebo pracovního listu základní kód, který roztočí motor o konstantní rychlosťi.

```
1 const int transistorPin = 9;
2 const int speedMotor = 200;
3
4 void setup() {
5     pinMode(transistorPin, OUTPUT);
6 }
7
8 void loop() {
9     analogWrite(transistorPin, speedMotor);
10}
```

Studenti ať program nahrají do desky a odzkouší, zda se motor začne otáčet.



#### ZEPTEJTE SE STUDENTŮ

→ V jakém rozsahu hodnot můžete měnit rychlosť motoru?

V rozsahu PWM 0-255. Rychlosť motoru je definováno v promenné speedMotor.

### 3. KROK 15 minut

Na základě zvládnutí principů ovládání motoru, budou studenti řešit následující úkol.

#### ÚKOL PRO STUDENTY

→ A) Do základního obvodu připojte potenciometr.

Studenti k tomu využijí poznatku z Lekce 3., kde programovali ovládání servomotoru.

→ B) Naprogramujte ovládání otáček motoru pomocí potenciometru.

Studenti využijí funkci map( ) pro nastavení rozsahu PWM vůči rozsahu hodnot potenciometru.



## 4. KROK 10 minut

Pokud studenti zvládli předchozí úkol, tak se mohou zamyslet na řešením následujícího úkolu.

### ÚKOL PRO STUDENTY

→ C) Vyměňte potenciometr za fotorezistor. Co je jiného v zapojení obvodu oproti předchozímu úkolu?

Studenti opět mohou využít zapojení s fotorezistory v Lekci 3. Zkuste studenty navést k tomu, aby místo dvou fotorezistorů využili jen jeden a druhý nahradili obyčejným rezistorem. Upozorněte studenty, že se opět jedná o zapojení napěťového děliče.



# PRACOVNÍ LIST – STEJNOSMĚRNÝ MOTOR

STEJNOSMĚRNÝ MOTOR TVOŘÍ ZÁKLAD POHONŮ V ROBOTICE. POUŽÍVÁJÍ SE ZEJMÉNA PRO POHYB ROBOTICKÝCH PROSTŘEDKŮ, ALE TAKÉ PRO VENTILÁTORY APOD.

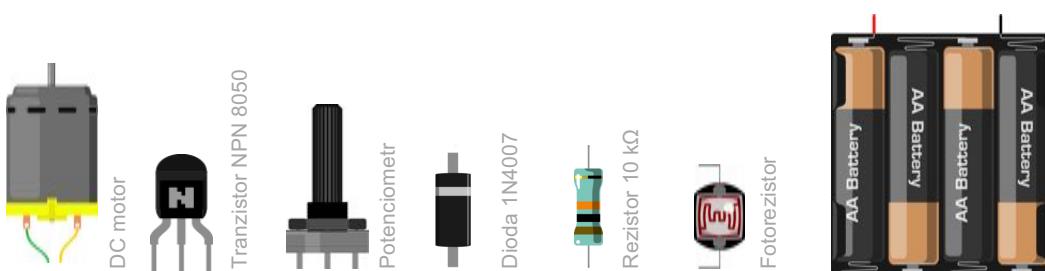
## CO SE NAUČÍTE

- ① Princip stejnosměrného motoru.
- ② Princip tranzistoru a jeho zapojení pro řízení motoru.
- ③ Zapojení externího napájení motoru.
- ④ Využití potenciometru pro regulaci otáček motoru.



## CO BUDETE POTŘEBOVAT

- ① Servomotor.
- ② Potenciometr.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Zásobník na baterie pro externí napájení.
- ⑥ Vodič typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

## OTÁZKA PRO VÁS

- Na jakém principu pracuje a jakého fyzikálního jevu využívá stejnosměrný motor?



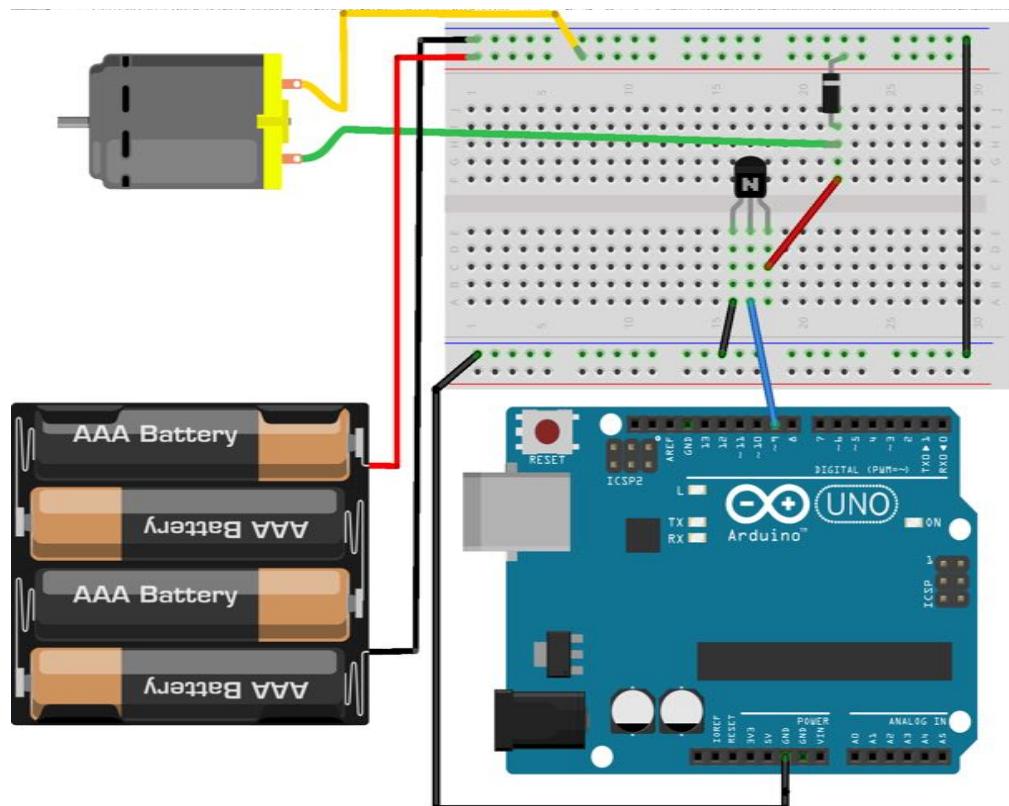
## A JDĚTE NA TO ...

- ① Podle přiloženého schématu zapojte obvod se stejnosměrným motorem.



### DEJTE SI POZOR

- Pozor si dejte zejména na zapojení tranzistoru a usměrňovací diody.





### OTÁZKA PRO VÁS

→ Věděli byste jakou funkci zastává usměrňovací dioda v obvodu?

- ② Napište program, který roztočí stejnosměrný motor.

```
1 const int transistorPin = 9;
2 const int speedMotor = 200;
3
4 void setup() {
5     pinMode(transistorPin, OUTPUT);
6 }
7
8 void loop() {
9     analogWrite(transistorPin, speedMotor);
10 }
```



### OTÁZKA PRO VÁS

→ V jakém rozsahu hodnot můžete měnit rychlosť motoru?

- ③ Pokud jste úkoly splnili a vše funguje, jak má, zkuste si zapojit a naprogramovat ještě jeden obvod. Do stávajícího obvodu stačí pouze přidat potenciometr podle přiloženého schématu.



### ÚKOLY PRO VÁS

- Do základního obvodu připojte potenciometr.
- Naprogramujte ovládání otáček motoru pomocí potenciometru.

- ④ Pokud jste zvládli předchozí úkol, tak se zkuste zamyslet nad řešením úkolu následujícího.

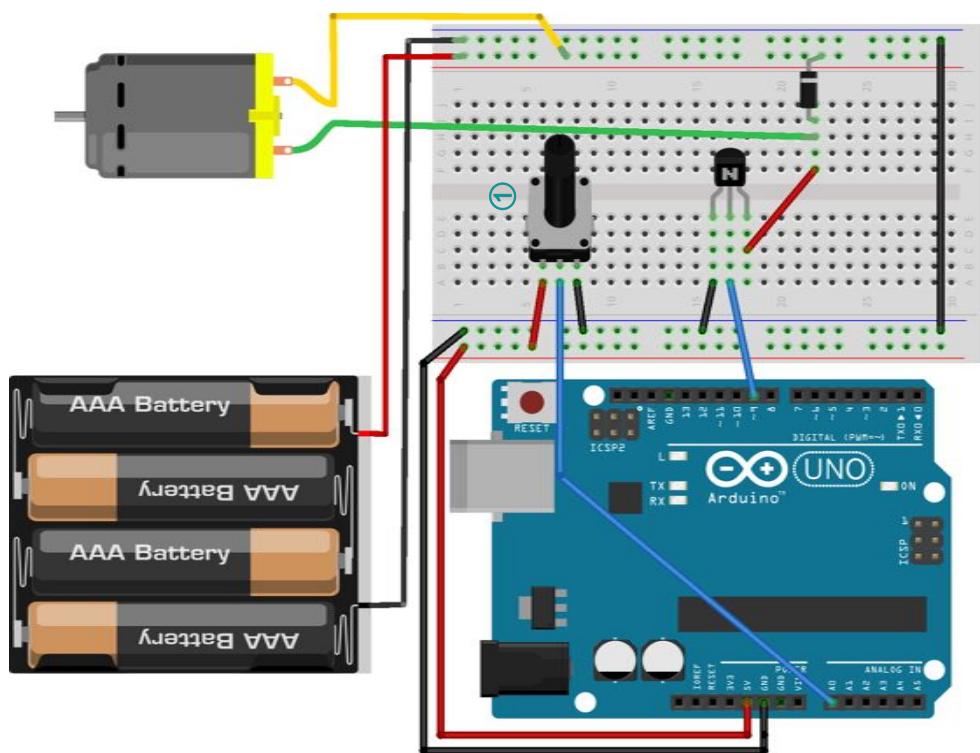


#### VYZKOUŠEJTE

→ Vyměňte potenciometr za fotorezistor. Co je jiného v zapojení obvodu oproti předchozímu úkolu, kdy byl připojen potenciometr?

# ŘEŠENÍ ÚLOH

Úkol A)

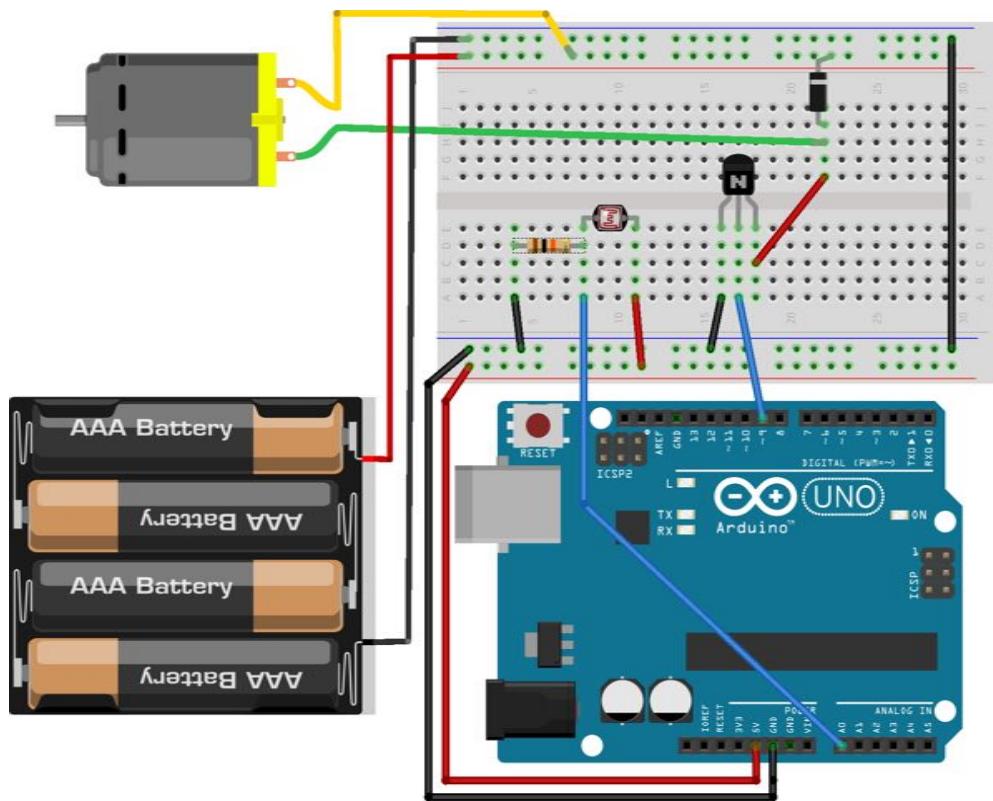


Úkol B)

```
1 const int transistorPin = 9;
2
3 void setup() {
4     pinMode(transistorPin, OUTPUT);
5 }
6
7 void loop() {
8     int sensorValue = analogRead(A0);
9     int outputValue = map(sensorValue, 0, 1023, 0, 255);
10    analogWrite(transistorPin, outputValue);
11 }
```

### **Úkol B)**

Programový kód zůstane stejný jako v příkladu, kde je zapojen potenciometr.



# PODROBNÝ PRŮVODCE TEORIÍ

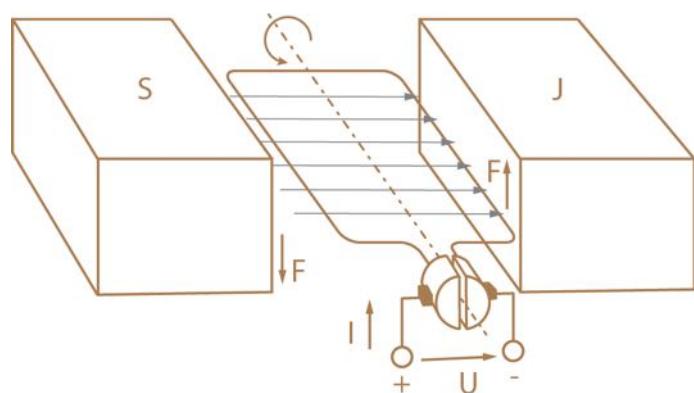
PODROBNÝ POPIS FUNKCIONALIT STEJNOSMĚRNÝCH MOTORŮ A MOŽNOSTÍ REGULACE PROSTŘEDNICTVÍM TRANZISTORU. JSOU ZDE TAKÉ PODROBNĚ VYSVĚTELNÉ PROGRAMOVÉ KÓDY JEDNOTLIVÝCH PŘÍKLADŮ.

## OBSAH PRŮVODCE

- ① Pochopení principu stejnosměrného motoru.
- ② Seznámení s principy tranzistoru.
- ③ Zapojení tranzistoru pro regulaci otáček motoru.
- ④ Zapojení externího napájení motoru.
- ⑤ Využití potenciometru pro regulaci motoru.
- ⑥ Projekt větráku.

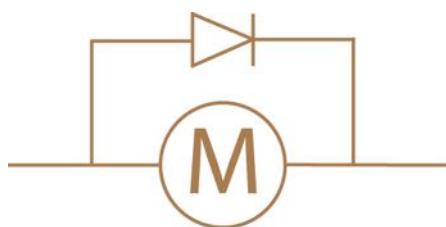
## O STEJNOSMĚRNÝCH MOTORECH V ARDUINU

Stejnosměrný motor je elektrické zařízení, které mění stejnosměrný proud na mechanickou energii. Základním jevem při této změně je indukce. To je jev, při kterém ve vodiči vzniká indukované elektromotorické napětí a indukovaný proud, v důsledku změny magnetického pole. Ke svorkám motoru přivádíme stejnosměrný proud, který prochází vodiči kotvy. Protože se tyto vodiče nacházejí v magnetickém poli, působí na ně jistá síla a motor se otáčí Obr. 1 - Princip stejnosměrného motoru.



Obr. 1 - Princip stejnosměrného motoru

Ovládání motorů pomocí Arduina je více komplikované než práce s LED diodou. V první řadě motor pro svou činnost potřebuje větší proud, než deska Arduino poskytuje prostřednictvím pinů, a to zejména při svém spouštění. Ve druhé řadě může motor díky indukci generovat tzv. zpětný proud, který může zničit součástky v obvodu. Proti tomuto jevu se využívá usměrňovací dioda. Usměrňovací diodu zapojujeme paralelně s motorem Obr. 2 - Usměrňovací dioda s motorem.



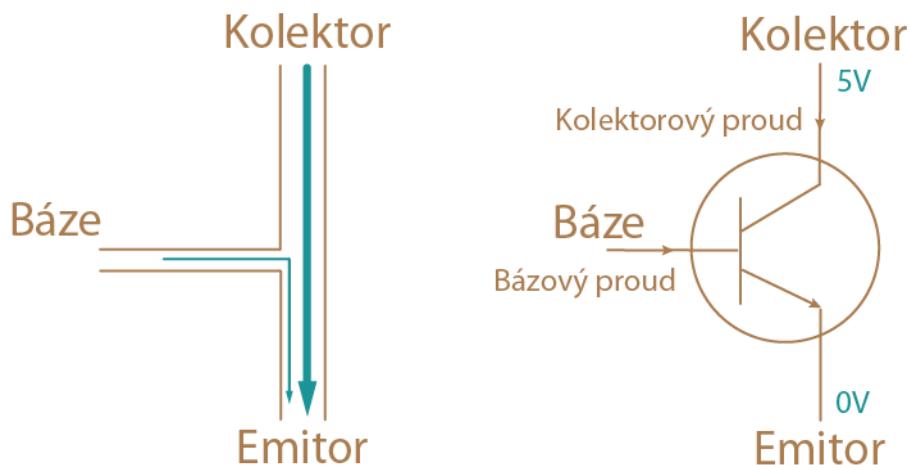
Obr. 2 - Usměrňovací dioda s motorem

## TRANZISTOR PRO ŘÍZENÍ OTÁČEK MOTORU

Vzhledem k tomu, že i malý stejnosměrný motor, bude pravděpodobně potřebovat pro svou činnost větší elektrický proud, než poskytuje deska Arduino, nelze jej připojit přímo, protože by došlo k poškození desky. Motor musí být napájen ze samostatného zdroje. Jak regulovat jeho otáčky prostřednictvím programového kódu? Tady se přímo nabízí využít možností tranzistoru.

Tranzistor je polovodičová součástka, která může fungovat jako „digitální spínač“, který využívá pouze malého proudu, který poskytuje digitální pin desky Arduino pro řízení mnohem většího proudu motoru. Existuje celá řada tranzistorů, ale jejich princip je stejný

Obr. 3 - Princip tranzistoru.



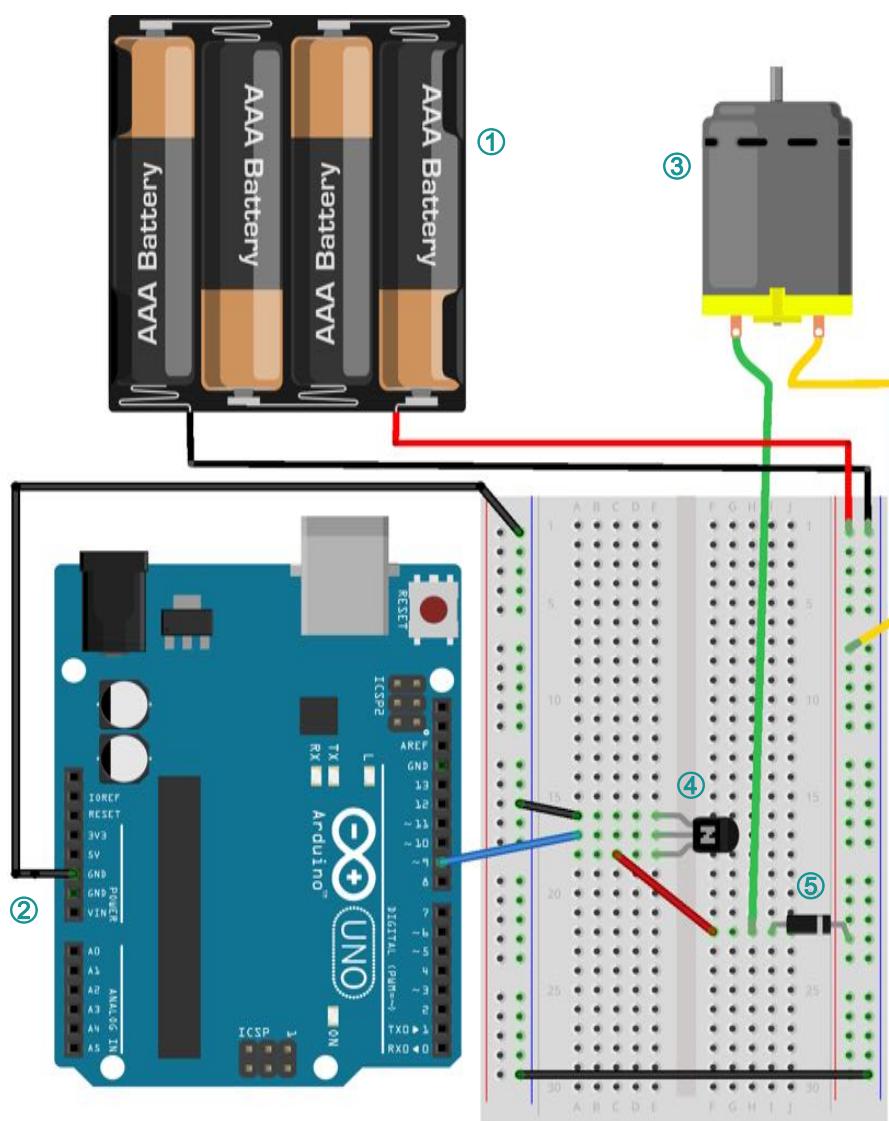
Obr. 3 - Princip tranzistoru

Tranzistor má tři vývody. Velký elektrický proud protéká z kolektoru do emitoru, ale to se bude dít pouze tehdy, jestliže bude také protékat proud z báze. Tento proud je malý a postačuje takový, který je dodáván právě digitálním výstupem desky Arduino.

## ZAPOJENÍ OBVODU

### ZÁKLADNÍ PŘÍKLAD

Základní příklad představuje zapojení stejnosměrného motoru s tranzistorem pro jeho regulaci. Regulaci otáček budeme nastavovat přímo v programovém kódu.



Obr. 4 - Zapojení stejnosměrného motoru

- ① V první řadě přivedeme externí napájení do kontaktního pole. Tímto napájením může být bateriové pouzdro s tužkovými bateriemi, nebo lze využít napájecí modul s adaptérem.

- ② Do kontaktního pole, do druhé části, přivedeme zem z desky Arduino. Zem z externího napájecího zdroje a z desky Arduino jsou propojeny.
- ③ Stejnosměrný motor připojíme do kontaktního pole. Využijeme k tomu vodiče typu „zástrčka“. Vodiče k motoru připevníme buď připájením, nebo vytvořením oček a přelepením hmotou z tavné pistole nebo lepící páskou. Jeden vodič z motoru připojíme do kontaktního pole, k napájení. Druhý vodič připojíme do kontaktního pole, do střední části, ze které dále povedeme vodič k tranzistoru na **Kolektor**.
- ④ Tranzistor je vložen přímo do kontaktního pole. **Emitor** je připojený k zemnění. **Báze** je připojena k desce Arduino, do digitálního pinu **9**. Kolektor je připojen k usměrňovací diodě a k motoru.
- ⑤ Usměrňovací dioda je paralelně připojena k motoru a chrání obvod proti zpětnému proudu.

## PROGRAMOVÝ KÓD

```

1 const int transistorPin = 9;
2 const int speedMotor = 200;
3
4 void setup() {
5   pinMode(transistorPin, OUTPUT);
6 }
7
8 void loop() {
9   analogWrite(transistorPin, speedMotor);
10}

```

The diagram shows a circuit connection. On the left, there is a vertical line representing a power source. A horizontal line connects to a vertical line labeled '1'. Another horizontal line connects to a vertical line labeled '2'. From '2', a line goes down to a vertical line labeled '3'. To the right of '3', there is a symbol for a motor with two terminals. A line from '3' connects to one terminal of the motor. The other terminal of the motor is connected to a diode symbol, which is also connected to ground. This represents a half-bridge driver circuit for a DC motor using an NPN transistor and a Zener diode for反向 protection.

- ① Nadefinujeme si konstantu **transistorPin**, která obsahuje číslo digitálního pinu desky Arduino, na který je připojena báze tranzistoru, tj. číslo **9**.
- ② Definice konstanty **speedMotor**, která určuje rychlosť motoru. Rychlosť motoru se nastavuje v rozmezí hodnot 0-255.
- ③ Definice pinu, na který je připojena báze tranzistoru, jako výstup.
- ④ Funkce **analogWrite()** nám posílá na výstup definovaný konstantou **transistorPin** nastavenou hodnotu rychlosťi **speedMotor**.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, motor by se měl roztočit. Pokud chcete změnit rychlosť, změňte hodnotu konstanty **speedMotor**.



### MOTOR SE NETOČÍ

**Zapojení tranzistoru** – zkontrolujte zapojení tranzistoru, aby byly opravdu jednotlivé vývody připojeny, jak je uvedeno na schématu zapojení.

**Zapojení diody** – zkontrolujte, aby usměrňovací dioda byla zapojena paralelně k motoru, a to v propustném směru od kolektoru tranzistoru.

Baterie, externí zdroj – ujistěte se, že baterie v externím zdroji jsou nabité, popř. externí zdroj zapojen do kontaktního pole.

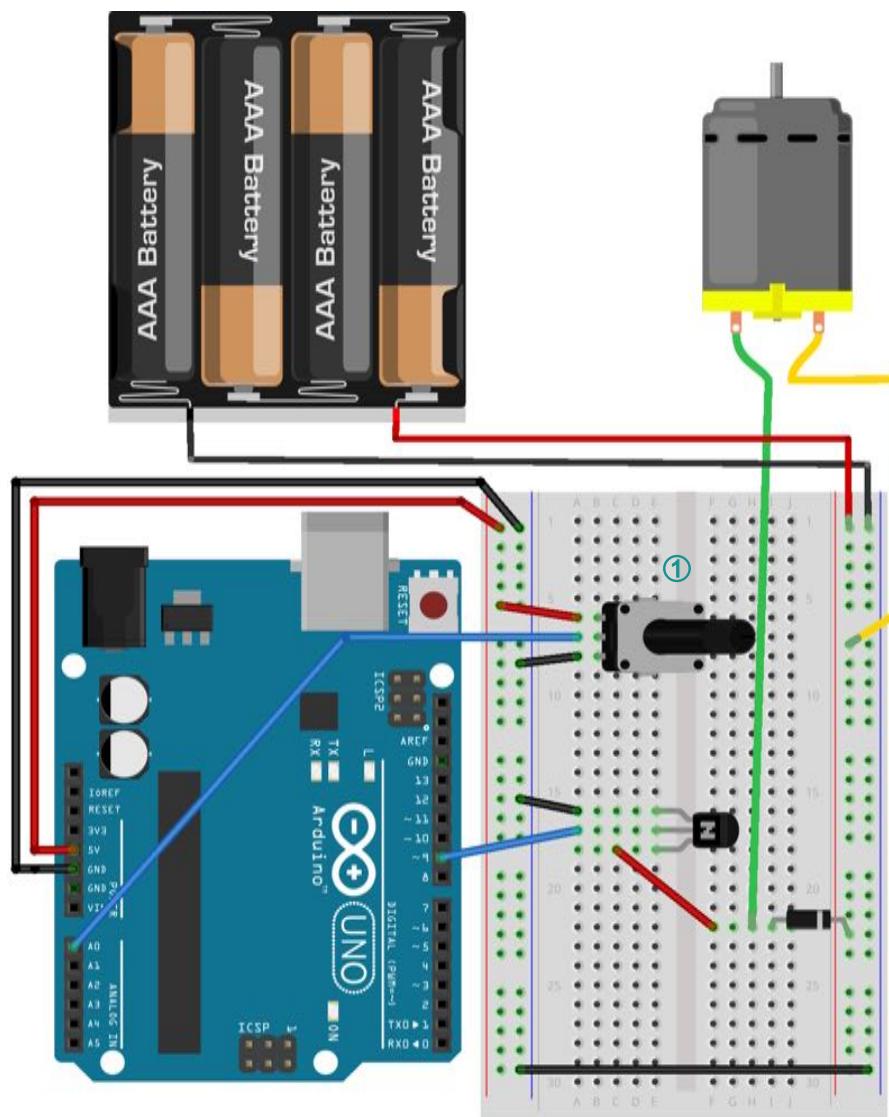
### NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

## ZÁKLADNÍ PŘÍKLAD – POTENCIOMETR

V zapojení obvodu využijme základní zapojení a přidáme potenciometr, který nám umožní plynulou regulaci otáček. Stejnosměrného motoru.



Obr. 5 - Zapojení stejnosměrného motoru a potenciometru

- ① Potenciometr je zasazen do kontaktního pole a je regulátorem otáček motoru. Jeho prostřední vývod je připojen do analogového pinu **A0** desky Arduino. Krajní vývody potenciometru jsou připojeny k napájení z desky Arduino a k zemi.

## PROGRAMOVÝ KÓD

```
1 const int transistorPin = 9;           └─①
2
3 void setup() {
4     pinMode(transistorPin, OUTPUT);    └─②
5 }
6
7 void loop() {
8     int sensorValue = analogRead(A0);   └─③
9     int outputValue = map(sensorValue, 0, 1023, 0, 255); └─④
10    analogWrite(transistorPin, outputValue);      └─⑤
11 }
```

- ① Nadefinujeme si konstantu **transistorPin**, která obsahuje číslo digitálního pinu desky Arduino, na který je připojena báze tranzistoru, tj. číslo **9**.
- ② Definice pinu, na který je připojena báze tranzistoru, jako výstup.
- ③ Do proměnné **sensorValue**, uložíme aktuální hodnotu z potenciometru.
- ④ Protože víme, že krajní hodnoty potenciometru jsou v rozmezí 0-1023, je nutné toto rozmezí namapovat do hodnot 0-255. V Arduino je k tomu učena přímo funkce **map()**. Tato funkce má pět parametrů. Aktuální hodnota na analogovém vstupu, minimální a maximální hodnota, kterých může proměnná **sensorValue** nabývat. Poslední dvojicí parametrů jsou odpovídající krajní hodnoty, které se budou posílat dále.
- ⑤ Funkce **analogWrite()** nám posílá na výstup definovaný konstantou **transistorPin** hodnotu rychlosti, uloženou v proměnné **outputValue**.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, motor by se měl roztočit, pokud je potenciometr mimo krajní minimální hodnot. Tentokrát rychlosť regulujete pomocí potenciometru.



## MOTOR

**Nemění se otáčky** – zkontrolujte zapojení potenciometru. Prostřední vývod je zapojen do analogového vstupu desky Arduino. Krajní vývody jsou připojeny k napájení z desky Arduino a k zemi.

### NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.



(Př. 1) Vyměňte v obvodu potenciometr za fotorezistor. Jak se změní zapojení a programový kód? Radou vám může být, využití sériového monitoru pro zmapování krajních hodnot fotorezistoru.



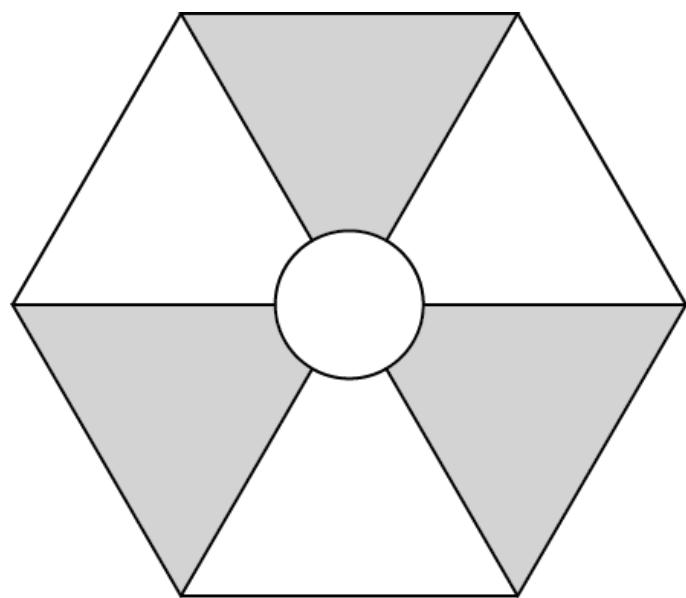
Pokud motor opravdu mění své otáčky. Můžeme si jej opatřit jednoduchým větráčkem. V následující kapitole je návod, jak na to.

## VĚTRÁČEK NA MOTOR

Pro vytvoření větráku budeme potřebovat: pevný papír, nůžky, tavnou pistoli, obyčejnou gumu na gumování.

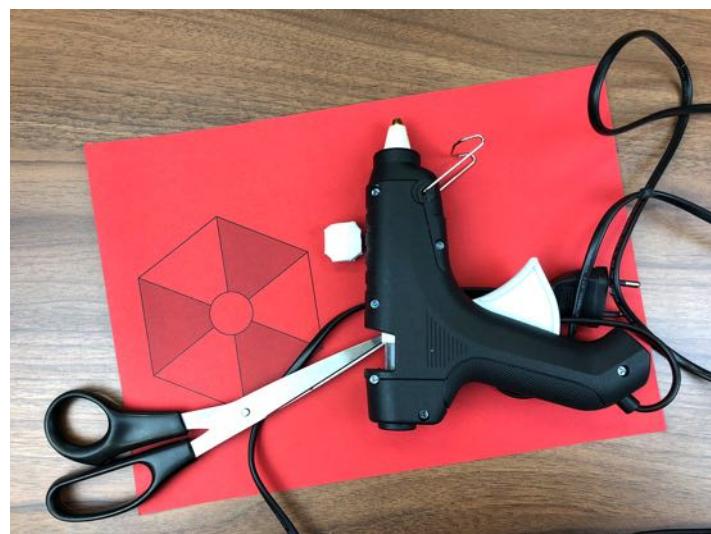
## PAPÍROVÁ KONSTRUKCE

Konstrukce větráčku je velice jednoduchá. Stačí si vytisknout na pevný papír přiloženou šablonu, která je na obrázku Obr. 6 – Šablona větráku.

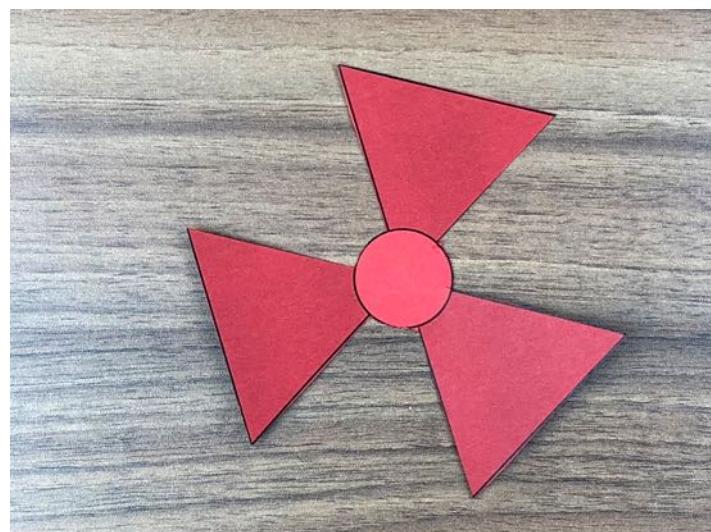


Obr. 6 – Šablona větráku

Vytiskněnou šablonu vystříhněte podle plných čar. Záleží na vás, jestli ponecháte šedé části nebo bílé, jako lopatky větráku.



Obr. 7 - Vytiskněná šablona

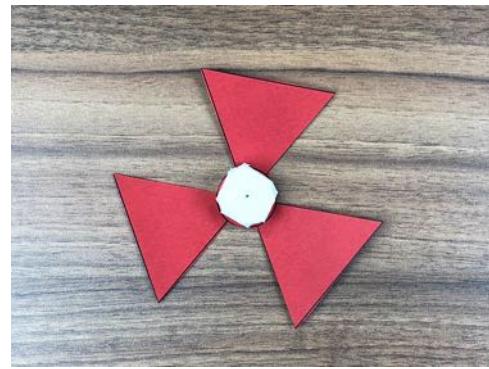


Obr. 8 - Vystřížená šablona

Na vystříženou šablonu, nalepíme tavnou pistoli gumu, která bude tvořit tzv. unášeč motoru.



Obr. 9 – Lepení gumy na vrtuli



Obr. 10 – Vrtule s gumovým unášečem

Gumu propíchněte, aby šla lépe nasadit na hřídel motoru a následně ji nasaděte.



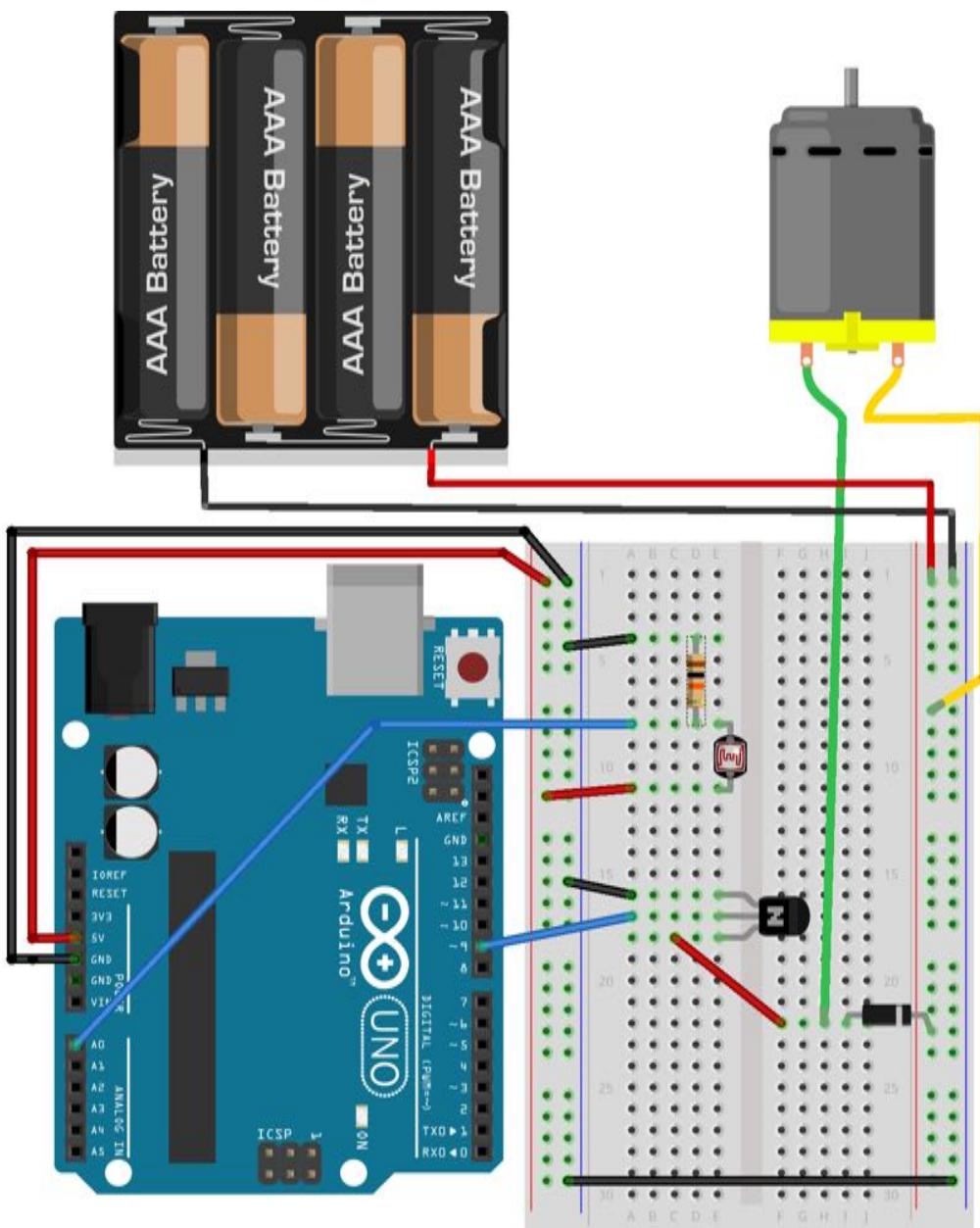
Obr. 11 – Nasazení vrtule na motor

## ŘEŠENÍ PŘÍKLADŮ

### (PŘ. 1)

Obvod byl upraven tak, že místo potenciometru byl do kontaktního pole přidán fotorezistor. Fotorezistor má, ale pouze dva vývody, takže se muselo využít rozdělení obvodu pomocí rezistoru, který má hodnotu  $10\text{k}\Omega$ .

Programový kód můžeme ponechat stejný, jako při zapojení s potenciometrem.



Obr. 12 - Zapojení motoru s fotorezistorem

## 6. SNÍMÁME A ZOBRAZUJEME TEPLITU

V TÉTO LEKCI SE ZAMĚŘÍME NA PRAKTICKÉ VYUŽITÍ JIŽZNÁMÝCH PROGRAMOVÝCH STRUKTUR, JAKÝMI JSOU VLASTNÍ FUNKCE. SEZNÁMÍME SE S MOŽNOSTMI SENZORŮ PRO MĚŘENÍ TEPLOTY A VLHKOSTI. TYTO JEDNODUCHÉ SEZNORY JSOU VSTUPNÍ BRANOU DO ROZSÁHLÉ OBLASTI ZJIŠŤOVÁNÍ NEJRŮZNĚJŠÍCH VELIČIN, KTERÉ LZE VYUŽÍT PRO OVLIVŇOVÁNÍ CHOVÁNÍ TECHNICKÝCH PROSTŘEDKŮ.

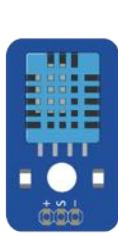
### CÍLE

- ① Správa a instalace externích knihoven.
- ② Senzory pro měření teploty a vlhkosti.
- ③ Zapojení a programování termistoru.
- ④ Základy matematických operací v Arduino.
- ⑤ Zapojení a programování čidla DHT11 pro měření teploty a vlhkosti.
- ⑥ Zobrazování hodnot prostřednictvím sériového monitoru.
- ⑦ Zobrazování hodnot veličin pomocí LCD displeje.

Čas: **2x45 min**

Úroveň: ■■■■■

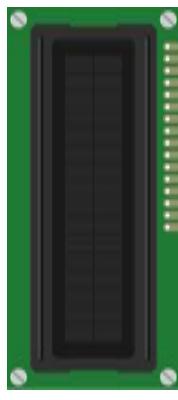
Vychází z: **3, 4, 5**



Teploměr



Termistor



LCD displej



Potenciometr



Rezistor 220Ω

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví dva obvody, ve kterých použijí dva senzory – termistor a čidlo teploty a vlhkosti DHT11. Tyto obvody naprogramují podle vzorového programového kódu. V programu využijí již získané vědomosti. Dále si zopakují jak pracovat se sériovým monitorem pro zobrazení výstupních hodnot. Součástí jsou jednoduché samostatné úkoly.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, termistor, rezistor  $220\Omega$ , čidlo teploty a vlhkosti DHT11, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 6.
- ⑤ Pracovní listy pro studenty.

## 1. KROK 🕒 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude si ukázat praktické možnosti využití čidel pro snímání teploty a vlhkosti.

### ZEPTEJTE SE STUDENTŮ

- ➔ **Kde se můžete v praktickém životě setkat se senzory teploty nebo vlhkosti?**  
Při regulaci a řízení v elektrárnách, domácnostech (inteligentní domy) atd.
- ➔ **K čemu byste využili senzory teploty a vlhkosti vy?**



## KROK 🕒 5 minut

Zobrazte studentům prostřednictvím dataprojektoru nebo pracovního listu zapojení termistoru.



### RYCHLÝ TIP

- Během zapojování, studentům vysvětlete, že se v obvodu využívá děliče napětí. Důvodem je to, že termistor poskytuje změnu odporu a ten přímo Arduino nepřečte. Co ale přečte? Změnu napětí Vout.
- Pro převod odporu termistoru na teplotu se využívá Steinhart-Hartova rovnice.

## 2. KROK 🕒 10 minut

Nyní studentům ukažte prostřednictvím dataprojektoru nebo pracovního listu základní kód, který obsahuje výpočet pomocí Steinhart-Hartovy rovnice.



### NA CO SE SOUSTŘEDIT?

- V programové části se soustředěte na matematické operátory. Jejich posloupnost vykonávání je dána matematickými pravidly.
- Upozorněte studenty na funkce pro výpis hodnot v sériovém monitoru. Sériový monitor si studenti otevřou kliknutím na ikonu v IDE rozhraní Arduino.

### ZEPEJTE SE STUDENTŮ

- Když program nahrajete a spustíte, v jakých jednotkách si myslíte, že jsou hodnoty v sériovém monitoru zobrazovány?  
Hodnoty jsou zobrazovány v kelvinech.



### 3. KROK 10 minut

Na základě opakování při používání vlastních funkcí, studenti budou řešit následující úkol.



#### ÚKOL PRO STUDENTY

- V programovém kódu vytvořte dvě funkce, které budou převádět teplotu ze stupňů Kelvina na stupně Celsia a Fahrinheita.

### 4. KROK 5 minut

Po úspěšném splnění přechozího úkolu ukažte studentům zapojení dalšího senzoru, který měří teplotu a vlhkost. Toto zapojení je velmi jednoduché.

### 5. KROK 10 minut

Řekněte studentům, aby napsali programový kód pro práci s čidlem teploty a vlhkosti.



#### KNIHOVNA DHT11

- Pro správnou funkcionality čidla musí být k dispozici nainstalovaná podpůrná knihovna. Ukažte studentům, jak tuto knihovnu nainstalovat.



Pro tuto chvíli je to vše. Ale pokud to jde, ponechte zapojený obvod s čidlem teploty a vlhkosti. Příští hodinu budeme pokračovat, tentokrát v zobrazení hodnot pomocí LCD displeje.

# PRACOVNÍ LIST – SNÍMÁME TEPLITU

TEPLOTA A VLHKOST, TO JSOU DVĚ VELIČINY, KTERÉ LZE POMOCÍ JEDNODUCHÝCH SENZORŮ VELMI DOBŘE MĚŘIT. NAMĚŘENÉ HODNOTY PAK MOHOU OVLIVŇOVAT AKTUÁTORY V ROBOTICKÉM NEBO AUTOMATIZOVANÉM PROSTŘEDKU.

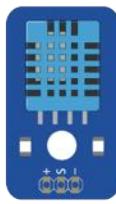
## CO SE NAUČÍTE

- ① Zapojovat termistor pro měření teploty.
- ② Programovat kód pro zobrazení teploty v sériovém monitoru.
- ③ Procvičíte si práci s matematickými operátory v programovém kódu.
- ④ Zapojovat a programovat čidlo teploty a vlhkosti DHT11.



## CO BUDETE POTŘEBOVAT

- ① Termistor - 1x.
- ② Rezistor  $220\Omega$  – 1x.
- ③ Čidlo teploty a vlhkosti DHT11.
- ④ Desku Arduino.
- ⑤ Kontaktní pole.
- ⑥ Vodiče typu zástrčka-zástrčka.



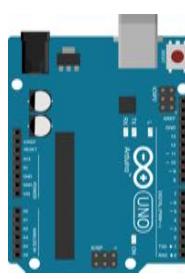
Tepelné čidlo



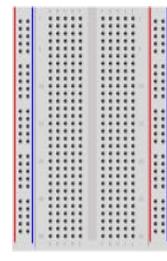
Termistor



Rezistor  $220\Omega$



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

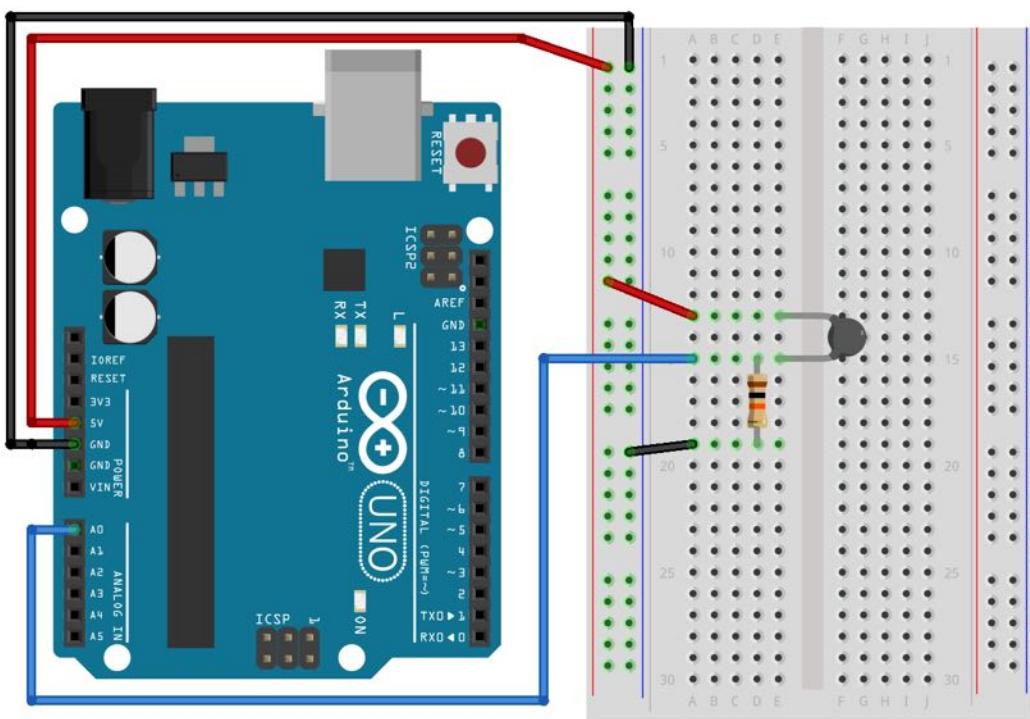
## A JDĚTE NA TO ...



## OTÁZKY PRO VÁS

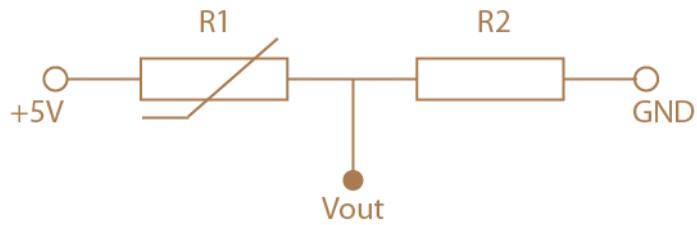
- Kde se můžete v praktickém životě setkat se senzory teploty nebo vlhkosti?
  - K čemu byste využili senzory teploty a vlhkosti vy?

① Podle schématu sestavte obvod s termistorem.



RYCHLÝ TIP

- K zapojení termistoru se využívá děliče napětí. Důvodem je, že termistor poskytuje změnu odporu a ten přímo Arduino nepřečte. Co ale přečte? Změnu napětí  $V_{out}$ .
  - Pro převod odporu termistoru na teplotu se využívá Steinhart-Hartova rovnice.



*Dělič napětí.*

- ② Napište program, který využívá Steinhart-Hartovu rovnici pro převod odporu termistoru na teplotu.

#### NA CO SE SOUSTŘEDIT?

- V programovém kódu se soustředěte na matematické operátory.
- Výpis výsledku měření si zobrazte pomocí sériového monitoru, který otevřete v rozhraní IDE Arduino kliknutím na ikonu .



```

int termistorPin = 0;
int Vout;
float R2 = 10000;
float logR1, R1, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;

void setup() {
    Serial.begin(9600);
}

void loop() {
    Vout = analogRead(termistorPin);
    R1 = R2 * (1023.0 / (float)Vout - 1.0);
    logR1 = log(R1);
    T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1));

    Serial.print("Teplota: ");
    Serial.print(T);

    delay(500);
}

```

- ③ Nahrajte program do desky Arduino, kliknutím na ikonu ➔

### OTÁZKA PRO VÁS

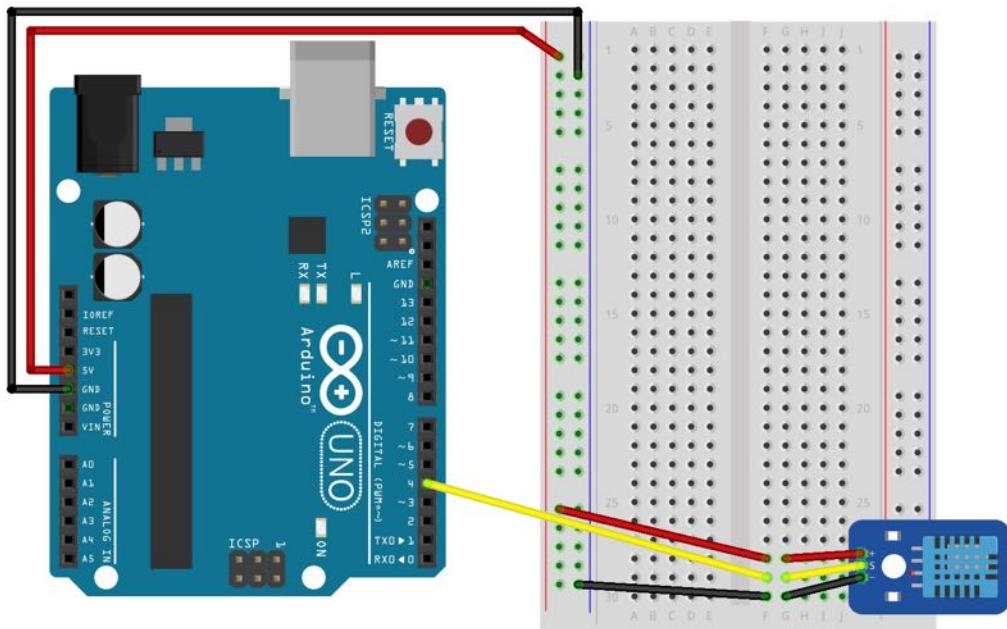
→ Když program nahrajete a spustíte, v jakých jednotkách si myslíte, že jsou hodnoty v sériovém monitoru zobrazovány?



### ÚKOL PRO STUDENTY

→ A) V programovém kódu vytvořte dvě funkce, které budou převádět teplotu ze stupňů Kelvina na stupně Celsia a Fahreinheita.

- ④ Pokud máte předchozí úkol vyřešen, zkuste ještě jednoduché zapojení čidla pro měření teploty a vlhkosti.



- ⑤ Napište následující programový kód.



### KNIHOVNA DHT11

➔ Pro správnou funkcionality čidla musí být k dispozici nainstalovaná podpůrná knihovna.

```
#include <dht11.h>

dht11 cidlo;

int dhtpin=7;

void setup(){
    Serial.begin(9600);
}

void loop()
{
    cidlo.read(dhtpin);
    Serial.print("Teplota = ");
    Serial.println(cidlo.temperature);
    Serial.print("Vlhkost = ");
    Serial.println(cidlo.humidity);
    delay(1000);
}
```

- ⑥ Pokud jste zvládli i zapojení čidla DHT11, tak je to pro tuto chvíli vše. Jestliže je to možné, nechte si zapojení této čidle do příští hodiny. Určitě jej využijete.

# ŘEŠENÍ ÚLOH

Úkol A)

```
1 int termistorPin = 0;
2 int Vout;
3 float R2 = 10000;
4 float logR1, R1, T, Tc, Tf;
5 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 =
6 2.019202697e-07;
7
8 void setup() {
9     Serial.begin(9600);
10 }
11
12 void getFahrein(Tc) {
13     return (Tc * 9.0)/ 5.0 + 32.0;
14 }
15
16 void getCelsius(T) {
17     return T - 273.15;
18 }
19
20 void loop() {
21     Vout = analogRead(termistorPin);
22     R1 = R2 * (1023.0 / (float)Vout - 1.0);
23     logR1 = log(R1);
24     T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1));
25
26     Tc = getCelsius(T);
27     Tf = getFahrein(Tc);
28
29     Serial.print("Teplota: ");
30     Serial.print(Tf);
31     Serial.print(" F; ");
32     Serial.print(Tc);
33     Serial.println(" C");
34
35     delay(500);
36 }
```

# PRŮVODCE HODINOU II



Studenti využijí zkušenosti práce s obvodem z přechozí hodiny, kde pracovali s čidlem teploty a vlhkosti. Tentokrát, ale pro zobrazení na snímaných hodnot nebudou využívat sériového monitoru, ale přímo zobrazovacího zařízení v podobě LCD displeje. Jedná se o komponentu, se kterou se lze v dnešní době setkat na každém kroku.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino, kontaktní pole, čidlo teploty a vlhkosti DHT11, LCD displej, potenciometr, vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 6.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⌚ 5 minut

Řekněte studentům, že náplní této hodiny bude se naučit využívat LCD displej pro zobrazení naměřených hodnot z čidla teploty a vlhkosti, které se naučili zapojovat a programovat minulou hodinu.

### OTÁZKA PRO STUDENTY

➔ **Kde všude se každý den setkáváte s displeji.**

Existuje celá řada displejů. Dnes se můžeme setkat s barevnými 3D na mobilních telefonech, ale i nadále z důvodu nižší spotřeby se využívají tzv. segmentové, např. jako informační tabule.

➔ **Na jakém principu LCD displej pracuje?**

V našem příkladu využíváme displej, který umožňuje zobrazit 16 znaků ve dvou řádcích. Každý znak lze zobrazit pomocí matice 5x10 pixelů.



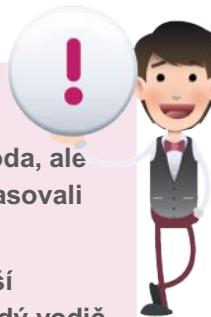
## 2. KROK 15 minut

Nyní ať si studenti sestaví obvod s čidlem DHT11. Pokud jej mají sestavený z minulé hodiny, mají to jednodušší a mohou rovnou pokračovat v zapojení LCD displeje.

Schéma zapojení můžete studentům promítat nebo je součástí pracovních listů.

### NA CO SI DÁT POZOR?

- ➔ LCD displej lze umístit přímo do kontaktního pole, což je velká výhoda, ale studenti si musí dát pozor na to, aby jednotlivé konektory přesně pasovali na zdírky kontaktního pole. Jinak je zohýbají.
- ➔ Zapojení LCD displeje není složité, ale přeci jenom obsahuje již větší množství vodičů. Studenti ať postupují systematicky a opravdu každý vodič kontrolují, zda je ve správném pinu.



## 3. KROK 5 minut

Vysvětlete princip vypisování textu na LCD displeji. Řekněte studentům, že vše prakticky zajišťuje jediná funkce `setCursor()`. Tedy za předpokladu, že se v programovém kódu využije knihovna `LiquidCrystal.h`.

## 4. KROK 10 minut

V tomto kroku ať studenti napíší program pro zobrazení teploty a vlhkosti na LCD displeji, který jim je ukázán pomocí dataprojektoru nebo pracovního listu.

## 5. KROK 10 minut

Studenti nyní mohou řešit samostatný úkol, který spočívá pouze v inovaci základního kódu.



### ÚKOL PRO STUDENTY

- A) Změňte programový kód základního příkladu tak, aby se kromě teploty ve stupních Celsia, na LCD displeji, střídavě zobrazovala teplota v Kelvinech a Fahrenheitech.

# PRACOVNÍ LIST – LCD displej

V TÉTO ČÁSTI VYUŽIJETE ZNALOSTÍ Z MINULÉ HODINY. JIŽ VÍTE JAK SNÍMAT TEPLITU A VLHKOST, ALE JEJICH ZOBRAZENÍ BYLO MOŽNÉ POUZE V SÉRIOVÉM MONITORU. NYNÍ K TOMU VYUŽIJETE LCD displej.

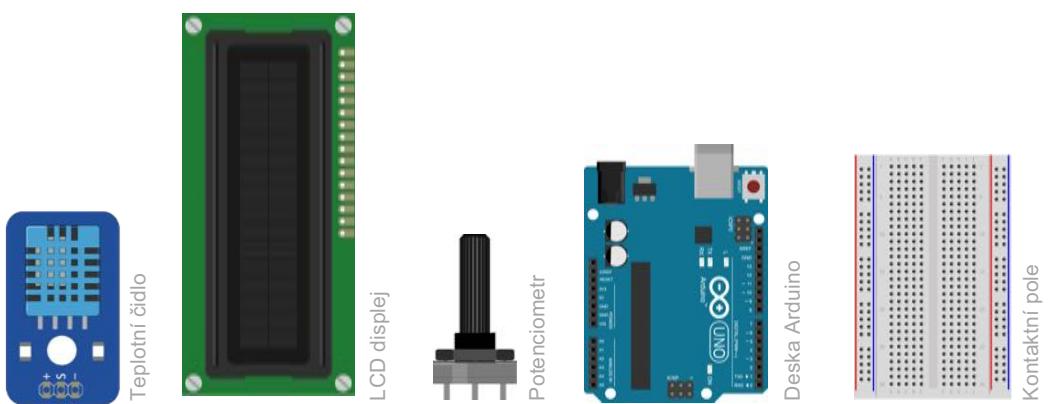
## CO SE NAUČÍTE

- ① Zapojit LCD displej.
- ② Jak naprogramovat zobrazování dat na LCD displeji.



## CO BUDETE POTŘEBOVAT

- ① Čidlo teploty a vlhkosti DHT11.
- ② LCD displej.
- ③ Potenciometr
- ④ Desku Arduino.
- ⑤ Kontaktní pole.
- ⑥ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

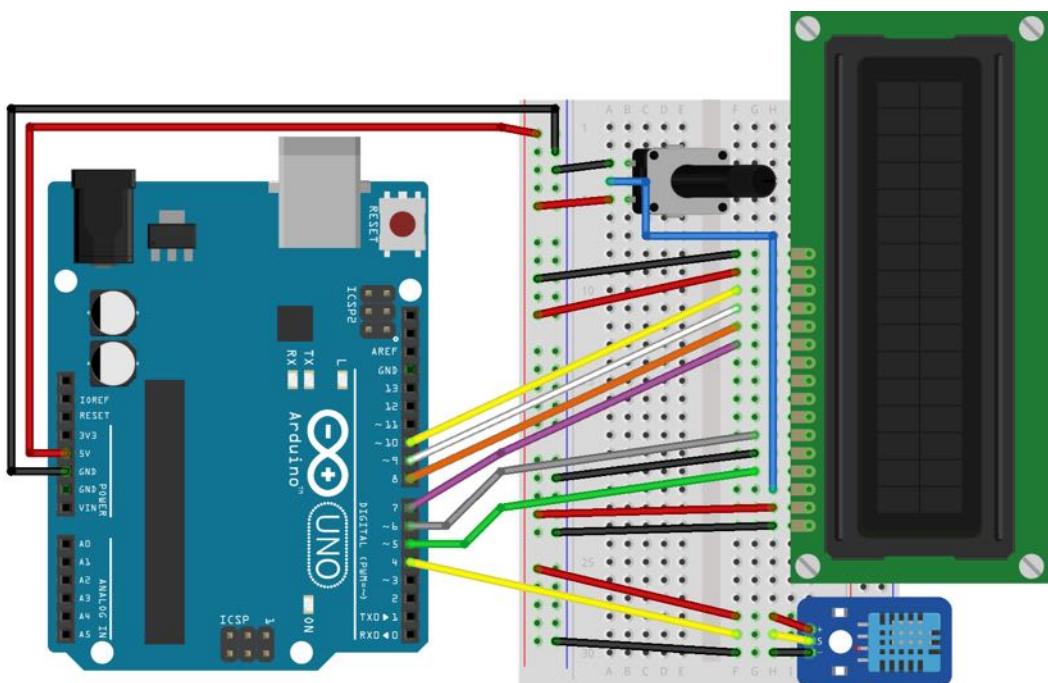
## A JDĚTE NA TO ...



### OTÁZKA PRO STUDENTY

- Kde všude se každý den setkáváte s displeji.
- Na jakém principu LCD displej pracuje?

- ① Pokud máte složený elektronický obvod z minulé hodiny, tj. zapojené čidlo teploty a vlhkosti, můžete se pustit rovnou do jeho rozšíření o LCD displej a potenciometr. V opačném případě obvod musíte obvod složit celý.



### NA CO SI DÁT POZOR?

- LCD displej lze umístit přímo do kontaktního pole, cože velká výhoda, ale musíte dát pozor na to, aby jednotlivé konektory přesně pasovali na zdířky kontaktního pole. Jinak je zohýbají.
- Zapojení LCD displeje není složité, ale přeci jenom obsahuje již větší množství vodičů. Postupujte systematicky a opravdu každý vodič kontrolujte, zda je ve správném pinu.





### PRINCIP LCD DISPLEJE

LCD displeje typu 16×1 nebo 16×2 znaků jsou velmi jednoduché, ale i efektivní při zobrazování krátkých informací. Jak z názvu vyplývá, 16×1 displej obsahuje jeden řádek o šestnácti znacích. U displeje 16×2 už to jsou dva řádky, takže celkem třicet dva znaků. Každý znak se skládá z 5×10 pixelů, takže na zobrazení jednoho znaku je nutné nastavit 50 pixelů. Naštěstí starost o řízení jednotlivých pixelů obstarává řídící obvod.

V Arduino nám práci s LCD displejem zjednoduší využití knihovny **LiquidCrystal.h**. Potom již stačí využívat funkci **setCursor()**.

- ② Nyní napište základní programový kód pro zobrazení hodnot naměřené teploty a vlhkosti na LCD displeji.

```
#include <dht11.h>
#include <LiquidCrystal.h>

int rsPin = 5;
int ePin = 6;
int d4Pin = 7;
int d5Pin = 8;
int d6Pin = 9;
int d7Pin = 10;
LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);

dht11 cidlo;
int dhtpin = 4;

void setup()
{
    LCD.begin(16,2);
    LCD.clear();
    LCD.setCursor(0,0);
    LCD.print("Teplota: ");

    LCD.setCursor(0,1);
    LCD.print("Vlhkost: ");
}
```

```
void loop()
{
    cidlo.read(dhtpin);

    LCD.setCursor(9,0);
    LCD.print(cidlo.temperature);
    LCD.setCursor(13,0);
    LCD.print((char)223);
    LCD.print("C");

    LCD.setCursor(9,1);
    LCD.print(cidlo.humidity);
    delay(500);
}
```

- ③ Nahrajte program do desky Arduino, kliknutím na ikonu ➔



#### ÚKOL PRO VÁS

- ➔ A) Změňte programový kód základního příkladu tak, aby se kromě teploty ve stupních Celsia, na LCD displeji, střídavě zobrazovala teplota v Kelvinech a Fahrenheitech.

# ŘEŠENÍ ÚLOH

Úkol A)

```
1 #include <dht11.h> // pripojení knihovny
2 #include <LiquidCrystal.h>
3
4 int rsPin = 5;
5 int ePin = 6;
6 int d4Pin = 7;
7 int d5Pin = 8;
8 int d6Pin = 9;
9 int d7Pin = 10;
10 LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);
11
12 dht11 cidlo;
13 int dhtpin = 4;
14
15 // funkce pro vypocet ve stupnich Fahrenheita
16 double Fahrenheit(double celsius){
17     return 1.8 * celsius + 32;
18 }
19
20 // funkce pro vypocet ve stupnich Kelvina
21 double Kelvin(double celsius){
22     return celsius + 273.15;
23 }
24
25 void setup(){
26     LCD.begin(16,2);
27     LCD.clear();
28     LCD.setCursor(0,0);
29     LCD.print("Teplota: ");
30
31     LCD.setCursor(0,1);
32     LCD.print("Vlhkost: ");
33 }
34
35 void loop(){
36     cidlo.read(dhtpin);
37
38     double celsia=cidlo.temperature; // zjistění stupnů Celsia
39     double kelviny=Kelvin(celsia); // zjistění stupnů Kelvina
40     double fahrenheit=Fahrenheit(celsia); // zjistění stupnů
41                                         // Fahrenheita
42
43 }
```

```
44 // v první rade se vypise vlhkost
45 LCD.setCursor(9,1);
46 LCD.print(cidlo.humidity);
47
48 // vypis teploty ve stupnich celsia
49 LCD.setCursor(9,0);
50 LCD.print(celsia);
51 LCD.setCursor(15,0); // musí se nastavit kurzor kde se
52 LCD.print(" "); // zobrazuji staré hodnoty v K a F
53 LCD.setCursor(13,0);
54 LCD.print((char)223);
55 LCD.print("C");
56
57 delay(2000);
58
59 // vypis teploty ve stupnich kelvina
60 LCD.setCursor(9,0);
61 LCD.print(kelviny);
62 LCD.setCursor(15,0);
63 LCD.print("K");
64
65 delay(2000);
66
67 // vypis teploty ve stupnich fahrenheinta
68 LCD.setCursor(9,0);
69 LCD.print(fahrenheity);
70 LCD.setCursor(15,0);
71 LCD.print("F");
72
73 delay(2000);
74 }
75
76
```

# PRŮVODCE HODINOU III



V této části budou studenti řešit zejména úkoly, které jsou spojeny s projektem automatického skleníku. Tento projekt ukazuje, jak provázat několik elektronických komponent dohromady a jak je také naprogramovat. Studenti by měli využít všechny doposud nabyté vědomosti z předchozích lekcí. Úkoly by měli řešit převážně samostatně, bez nutné větší účasti vyučujícího.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino, kontaktní pole, termistor, servomotor, stejnosměrný motor, tranzistor, usměrňovací dioda, rezistor  $10\text{k}\Omega$ , vodiče typu zástrčka-zástrčka.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 6.
- ⑤ Pracovní listy pro studenty.
- ⑥ Poskládaný model skleníku.



Výrobu konstrukce stolního skleníku by měli studenti realizovat doma, podle přiloženého postupu, uvedeného na konci lekce. Konstrukce je opravdu jednoduchá a na hodinu by měli přinést hotový výrobek, do kterého zakomponují elektronické části pro jeho ovládání.

## 1. KROK 5 minut

Na začátku hodiny řekněte studentům, že se jedná zejména o opakování z přechozích lekcí a využijí vše co se již naučili. Rozdejte studentům sady Arduino a ať vyberou potřebné součástky.

## 2. KROK 5 minut

Prvním krokem bude sestavení obvodu s termistorem. Studenti by si měli najít schéma zapojení z předchozích hodin. To platí i pro další kroky týkající se zapojení el. komponent a programovacího kódu.



### ÚKOL PRO STUDENTY

- Sestavte první část ovládání skleníku. Zapojte termistor, který bude snímat teplotu ve skleníku.



### TIP

Termistor by měl být připojený pomocí vodičů mimo kontaktní pole, protože bude uvnitř skleníku. Termistor může být přilepený páskou ke květináči a vodiče vyvedeny mimo skleník.

## 3. KROK 5 minut



### ÚKOL PRO STUDENTY

- Implementujte program, který zajistí čtení teploty z termistoru ve stupních Celsia.

### TIP



Programový kód mohou studenti použít z předchozího příkladu. Tento kód obsahuje Steinhart-Hartovu rovnici. V tomto programu budou následně probíhat úpravy týkající se ovládání servomotoru a stejnosměrného motoru v závislosti na teplotě.

Ať studenti odzkouší zapojení a programový kód. Hodnoty naměřené termistorem budou zobrazovány v sériovém monitoru.

## 4. KROK 10 minut

V tomto kroku studenti zapojí servomotor, kterým se bude ovládat otevírání a zavírání střešního okna. Opět to může být samostatný úkol pro studenty.

### ÚKOL PRO STUDENTY

- Zapojte servomotor, který bude ovládat střešní okno skleníku. Servomotor je umístěn v konstrukci skleníku.
- Programový kód pro zjišťování teploty upravte tak, aby při dosažení vyšší teploty, než bude vámi definovaná, servomotor okno otevřel. Při poklesu teploty ve skleníku naopak servomotor okno zavře.



## 5. KROK 10 minut

### ÚKOL PRO STUDENTY

- Do stávajícího obvodu zapojte stejnosměrný motor, který bude sloužit ve skleníku jako větrák.
- Upravte programový kód tak, aby se větrák zapnul a vypnul při dosažení určité teploty. Bude zapínán ve stejnou chvíli jako servomotor pro otevírání okna.



## 6. KROK 10 minut

Zbytek hodiny ať studenti věnují plnému zprovoznění ovládání skleníku.



### OTÁZKA PRO STUDENTY

- Jak byste stávající skleník vylepšili? Míní se tím zejména ovládání elektronických komponent.

Viz níže v dalších nápadech.

### DALŠÍ NÁPADY



- V obvodu lze nahradit termistor za čidlo teploty a vlhkosti DHT11.
- Lze připojit také LCD displej pro zobrazení teploty a vlhkosti uvnitř skleníku.
- Otevírání střešního okna by mohlo být plynulé, nikoliv skokové.

# PRACOVNÍ LIST – SKLENÍK

VŠE, CO JSTE SE NAUČILI V PŘEDCHOZÍCH HODINÁCH, KTERÉ SE ZABÝVALY MĚŘENÍ A ZOBRAZOVÁNÍ TEPLITRY, NYNÍ VYUŽIJETE VE VELKÉM PROJEKTU. BUDETE OVLÁDAT SVŮJ VLASTNÍ SKELNÍK.

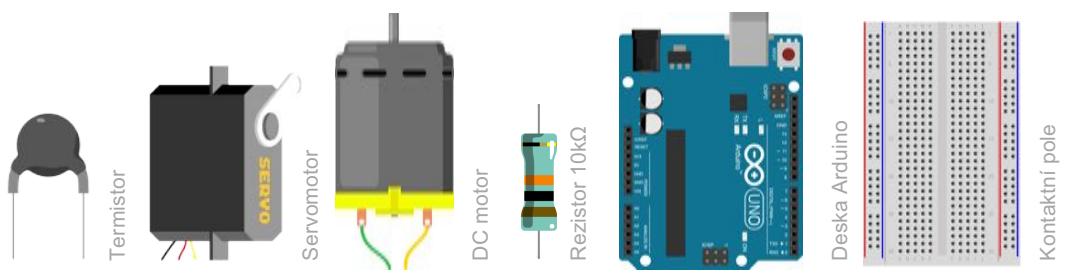
## CO SE NAUČÍTE

- ① Zapojit všechny komponenty pro měření teploty, ovládání střešního okna a větráku v jednom obvodu.
- ② Zkombinovat programový kód v jeden celek.



## CO BUDETE POTŘEBOVAT

- ① Termistor.
- ② Servomotor.
- ③ Stejnosměrný motor, tranzistor, usměrňovací dioda.
- ④ Rezistor  $10\text{k}\Omega$ .
- ⑤ Desku Arduino.
- ⑥ Kontaktní pole.
- ⑦ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...



V první řadě byste měli mít vyrobený model skleníku, podle návodu. Tento model jste si měli vyrobit například doma. Nyní již budete skleník ovládat pomocí komponent, které naprogramujete.

Tato hodina je především o samostatné práci. Využijete k tomu vše, co jste doposavad naučili. Při plnění úkolů můžete využít uložených příkladů z minulých hodin.



### ÚKOL PRO VÁS

- Sestavte první část ovládání skleníku. Zapojte termistor, který bude snímat teplotu ve skleníku.



### TYP

Termistor by měl být připojený pomocí vodičů mimo kontaktní pole, protože bude uvnitř skleníku. Termistor může být přilepený páskou ke květináči a vodiče vyvedeny mimo skleník.



### ÚKOL PRO VÁS

- Napište program, který zajistí čtení teploty z termistoru ve stupních Celsia.

### TYP



Můžete využít programový kód z již vypracovaného příkladu z předchozí hodiny.

Tento kód obsahoval Steinhart-Hartovu rovnici, pro převod naměřené změny napětí na teplotu. Programový kód budete v dalších úkolech pouze inovovat v závislosti na zapojení dalších komponent.

Po naprogramování vyzkoušejte zapojení a kód. Hodnoty si nechte zobrazit v sériovém monitoru.

Pokud vše funguje, jak má, a vidíte odpovídající teplotu prostřednictvím sériového monitoru, můžete přistoupit k dalšímu kroku. Tím bude automatické otevírání střešního okna skleníku v závislosti na teplotě uvnitř.

### ÚKOL VÁS

- Zapojte servomotor, který bude ovládat střešní okno skleníku.
- Programový kód z předchozího úkolu upravte tak, aby při dosažení vyšší teploty, než bude vámi definovaná, servomotor okno otevřel. Při poklesu teploty ve skleníku naopak servomotor okno zavře.



Servomotor umístěte do modelu skleníku. Měl by tam být připravený otvor. Jako táhlo servomotoru, můžete použít například kancelářskou sponku.



Po úspěšném vyzkoušení otevírání střešního okna, můžete přejít poslednímu úkolu. Opět se bude jednat o zapojení další el. komponenty a inovaci stávajícího programového kódu, která je velmi jednoduchá.



### ÚKOL PRO VÁS

- Do stávajícího obvodu zapojte stejnosměrný motor, který bude sloužit ve skleníku jako větrák.
- Upravte programový kód tak, aby se větrák zapnul a vypnul při dosažení určité teploty. Bude zapínán ve stejnou chvíli jako servomotor pro otevírání okna.

Zbytek hodiny se věnujte plnému zprovoznění ovládání skleníku.



### OTÁZKA PRO VÁS

- Jak byste stávající skleník vylepšili? Míni se tím zejména ovládání elektronických komponent.

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU, KTERÝ JE ZAMĚŘEN NA POUŽÍVÁNÍ SENZORŮ PRO MĚŘENÍ TEPLITY A VLHKOSTI. PRO ZOBRAZENÍ TĚCHTO VELIČIN SE VYUŽIJЕ LCD DISPLAY. ŘEŠENÉ ÚKOLY ZOHLEDŇUJÍ NOVĚ PROBRANÉ TÉMA A STAVÍ NA PŘEDCHOZÍCH ZNALOSTECH.

## OBSAH PRŮVODCE

- ① Získání dovedností při zapojování senzoru teploty.
- ② Naučení se pracovat se sériovým monitorem.
- ③ Naučit se zapojit LCD display a zobrazovat hodnoty.
- ④ Připojování externích knihoven pro snazší programování.
- ⑤ Naučit se zobrazovat hodnoty na LCD displeji.
- ⑥ Projekt skleníku.

## SPRÁVA KNIHOVEN V ARDUINO

Knihovny jsou soubory kódu, které usnadňují připojení různých senzorů a dalších modulů k desce Arduino. V této části využíváme jako senzor teplotní čidlo a pro zobrazení hodnot z tohoto senzoru LCD display. Aby se nám lépe pracovalo s těmi zařízeními, existují již hotové knihovny, které nám velmi zjednoduší jejich používání. Například pro LCD displej je již knihovna standardně nainstalovaná v prostředí Arduino IDE. Pro teplotní čidlo tomu tak není, proto si knihovnu musíme nainstalovat.



V Arduino jsou knihovny složky, které obsahují více souborů se zdrojovými kódy.

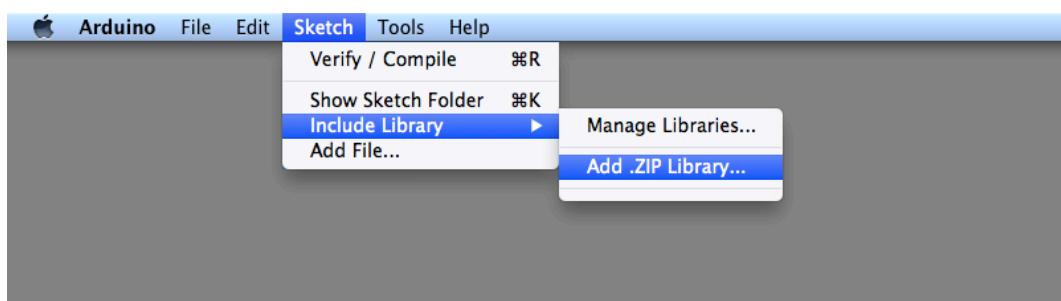
Tyto soubory jsou zpravidla napsány v jazyce C.

### JAK NAINSTALOVAT KNIHOVNU

Instalaci knihoven, lze provést několika způsoby.

#### POMOCÍ IMPORTU KNIHOVNY ZIP

Chcete-li nainstalovat novou knihovnu do Arduino IDE, můžete využít nástroj, který je k tomu určený. Ten je dostupný přímo v IDE. Otevřete Arduino IDE a klikněte na položku menu **Projekt** (Sketch). Potom na **Přidat knihovnu** (Import Library) a zvolte **Přidat .ZIP Knihovnu...**. Otevře se okno, ve kterém vyberte ZIP archív s knihovnou a potvrďte. Pokud je knihovna v pořádku, dojde k jejímu nainstalování a objeví se v seznamu knihoven v aktuálně otevřené nabídce menu.



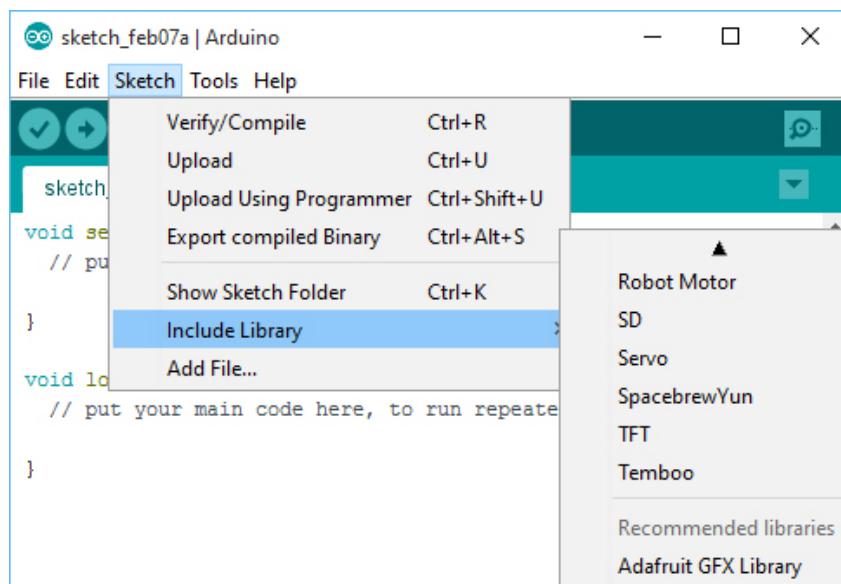
Obr. 1 - Instalace knihovny v Arduino IDE

## RUČNÍ INSTALACE

Knihovnu lze přidat také ručně. Stáhněte si knihovnu jako ZIP soubor a rozbalte jej. Rozbalený adresář, který nese jméno knihovny obsahuje všechny potřebné soubory, včetně vzorových příkladů, které autor poskytnul. Tento celý adresář nakopírujte do složky pro knihovny:

- ① V systému WINDOWS se adresář knihoven nachází: **Tento počítač > Dokumenty > Arduino > libraries.**
- ② V systému macOS se adresář knihoven nachází: **MyFolder > Dokumenty > Arduino > libraries.**

Po přesunutí adresáře knihovny do správného umístění, restartujte Arduino IDE a ověřte, zda je knihovna v seznamu knihoven Obr. 2 - Seznam knihoven.



Obr. 2 - Seznam knihoven

## SNÍMÁNÍ TEPLITÝ POMOCÍ TERMISTORU

Snímání teploty lze provádět několika různými senzory. Prvním takovým zástupcem je **termistor**.

### TERMISTOR

Termistory jsou jednoduché, levné a relativně přesné komponenty, které lze využít pro různé meteostanice, domácí automatizační systémy a obvody řízení a ochrany zařízení. Jsou to analogové senzory, které ve srovnání s digitálními jsou velmi jednouché a nevyžadují žádné knihovny.

Termistory jsou variabilní rezistory, které mění svůj odpor v závislosti na teplotě. Jsou klasifikovány podle toho, jak jejich odpor reaguje na změnu teploty:

- ① **Termistory s negativním teplotním koeficientem** (NTC) – odpor se snižuje s nárůstem teploty.
- ② **Termistory s kladným teplotním koeficientem** (PTC) – odpor se zvyšuje s nárůstem teploty.



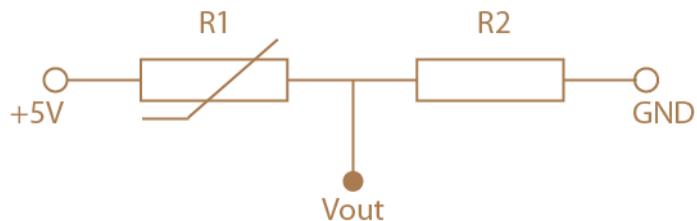
Běžnějšími termistory jsou NTC. Jsou vyrobeny z polovodičového materiálu, který je zahřát a následně stlačen, aby vytvořil teplotně citlivý vodivý materiál. Materiál obsahuje nosiče náboje, které dovolují protékání proudu. Vysoké teploty způsobují, že polovodivý materiál uvolňuje více nosičů náboje. V NTC termistorech vyrobených z oxidu železitého jsou nosičem náboje elektrony.

### JAK TO FUNGUJE S ARDUINEM?

Analogové piny desky Arduino čtou hodnoty napětí od 0V do 5V, jenomže termistor nám poskytuje změnu odporu a ten přímo deskou Arduino číst nemůžeme. Jak na to? Standardní způsob, jak číst změnu odporu termistoru v závislosti na změně napětí je vytvořit obvod děliče napětí – Obr. 3.

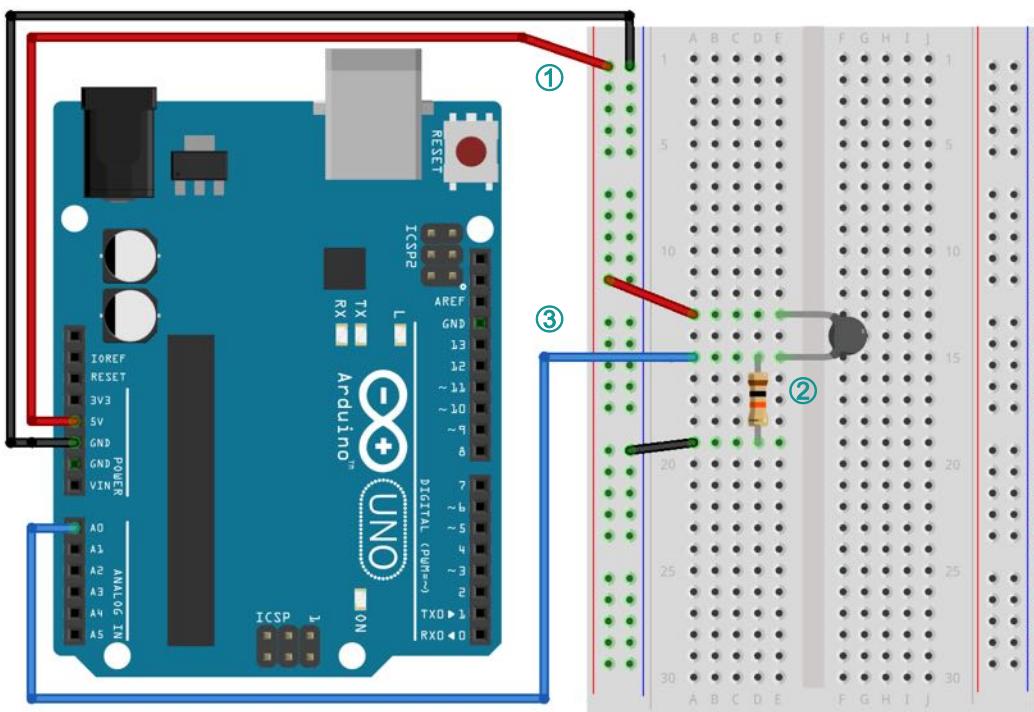
Je to velmi jednoduchý obvod. Při změně hodnoty odporu **R1** se změní hodnota **Vout**. Vzhledem k tomu, že odpor R1 je termistor a mění svůj odpor právě s teplotou, potom Vout může tvořit vstup do desky Arduino, který je pro ni čitelný.

Pro převod odporu termistoru na teplotu se využívá Steinhart-Hartova rovnice.



Obr. 3 - Dělič napětí.

## ZAPOJENÍ OBVODU S TERMISTOREM



Obr. 4 - Zapojení termistoru

- ① Vodič zemnění **GND** z desky Arduino zapojte do kontaktní desky k **modré** čáře. Vodič napájení připojte z Arduina z pinu **5V** do kontaktního pole.
- ② Rezistor  $10\text{k}\Omega$  a termistor jsou zapojeny podle obvodu děliče napětí.
- ③ Signální vodič je zapojen do analogového pinu **A0**.

## PROGRAMOVÝ KÓD

Programový kód obsahuje zejména matematické výpočty. Teplotu si zobrazíme v [Sériovém monitoru](#).

```
1 int termistorPin = 0;          ①
2 int Vout;                     ②
3 float R2 = 10000;            ③
4 float logR1, R1, T, Tc, Tf;  ④
5 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = ⑤
6 2.019202697e-07;
7
8 void setup() {
9     Serial.begin(9600);        ⑥
10 }
11
12 void loop() {
13     Vout = analogRead(termistorPin);    ⑦
14     R1 = R2 * (1023.0 / (float)Vout - 1.0);  ⑧
15     logR1 = log(R1);
16     T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1));  ⑨
17     Tc = T - 273.15;                ⑩
18     Tf = (Tc * 9.0) / 5.0 + 32.0;   ⑪
19
20     Serial.print("Teplota: ");
21     Serial.print(Tf);             ⑫
22     Serial.print(" F; ");
23     Serial.print(Tc);
24     Serial.println(" C");
25
26     delay(500);
27 }
```

- ① Deklarace proměnné **termistorPin** pro číslo analogového pinu desky Arduino, na který je připojen datový vodič z napěťového děliče.
- ② Deklarace proměnné **Vout**, která bude nabývat aktuální hodnoty v závislosti na teplotě.
- ③ Hodnota neměnného rezistoru **R2** u napěťového děliče, tj.  $10\text{k}\Omega$ .
- ④ Deklarace proměnných **logR2**, **R1**, **T**, **Tc**, **Tf**, které nabývají hodnot z výpočtů.
- ⑤ Deklarace konstant, které jsou určeny pro výpočet Steinhart-Hartovy rovnice. Tyto hodnoty jsou parametry, které se dají zjistit z katalogu součástky.

- ⑥ Nastavení rychlosti pro sériový přenos. Dílčí výsledky měření zobrazíme v sériovém monitoru.
- ⑦ Zjištění změny napětí, přečtené z analogového pinu desky Arduinoa uložení do proměnné **Vout**.
- ⑧ Výpočet aktuálního odporu **R1** v závislosti na změně **Vout**.
- ⑨ Steinhart-Hartova rovnice, která slouží k výpočtu poměru odpor-teplota. Výsledek v proměnné **T** je ve stupních Kelvinia.
- ⑩ Převod ze stupňů Kelvinia na Stupně Celsia - **Tc**.
- ⑪ Převod ze stupňů Celsia na stupně Fahreinheita - **Tf**.
- ⑫ Vypsání hodnot do sériového monitoru.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Otevřete sériový monitor. Pokud je vše v pořádku, měla by být vidět aktuální hodnota teploty naměřené termistorem.

## MĚŘENÍ TEPLITÝ A VLHKOSTI

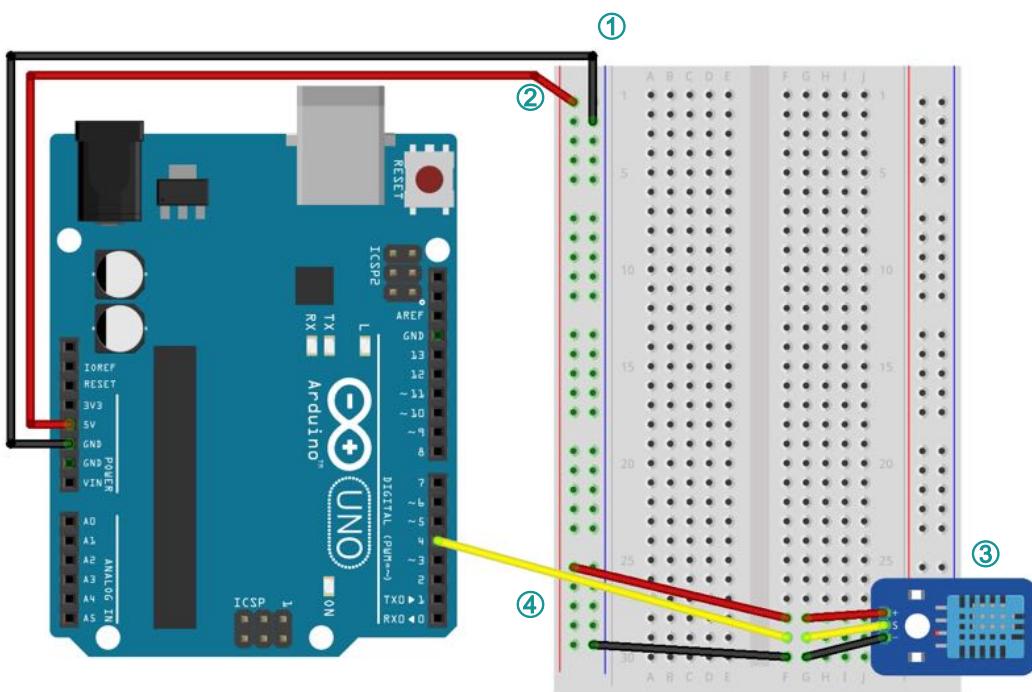
Zapojení obvodu se může jevit jako složitější, ale není tomu tak. Použité teplotní čidlo a LCD displej poskytuje rychlé řešení, které lze postavit za několik minut.

### TEPLOTNÍ ČIDLO

V projektu je použito teplotní čidlo, které je součástí sady Arduino. Jeho technické označení je DHT11. Čidlo je laboratorně kalibrované, přesné, stabilní a jeho signál je **digitální**. Čidlo má tři vývody: VCC (+) pro napájení, GND (-) je zemnění a S je signální pro připojení do pinu desky Arduino. Čidlo kromě teploty, zjišťuje také vlhkost.

### ZAPOJENÍ OBVODU

Nejdříve vytvoříme jednoduché zapojení pro teplotní čidlo. Naměřenou teplotu si necháme zobrazit pomocí sériového monitoru.



Obr. 5 - Zapojení teplotního čidla

- ① Vodič zemnění z desky Arduino zapojte do kontaktní desky k **modré** čáře. Tím se zpřehlední celé zapojení.
- ② Z desky Arduino, z pinu pro 5V přiveďte červený vodič do kontaktního pole k **červené** čáře.

- ③ Do kontaktní desky zapojte teplotní čidlo. Jeho konektory stačí zasunout do kontaktního pole. Teplotní čidlo má označené vývody **+**, **S**, **-**.
- ④ Jednotlivé vodiče propojte s teplotním čidlem. Vodič napájení (**červený**) přiveďte na vývod čidla „**+**“. Vodič zemnění (**černý**) přiveďte na vývod čidla „**-**“ a vodič signální (**žlutý**) přiveďte z pinu **4** desky Arduino do vývodu čidla **S**.

## PROGRAMOVÝ KÓD

Programový kód je díky knihovně pro čidlo velmi jednoduchý. Teplotu si zobrazíme opět v **Sériovém monitoru**.

```

1 #include <dht11.h>                                ①
2
3 dht11 cidlo;                                       ②
4
5 int dhtpin=4;                                      ③
6
7 void setup(){                                       ④
8     Serial.begin(9600);
9 }
10
11 void loop() {
12 {
13     cidlo.read(dhtpin);                            ⑤
14     Serial.print("Teplota = ");                   ⑥
15     Serial.println(cidlo.temperature);            ⑦
16     Serial.print("Vlhkost = ");                  ⑧
17     Serial.println(cidlo.humidity);              ⑨
18     delay(1000);                                ⑩
19 }
```

- ① Příkazem **#include** se připojují knihovny k projektu. Ve špičaté závorce je název hlavičkového souboru knihovny pro teplotní čidlo. Tuto knihovnu si lze stáhnout z adresy [https://github.com/Nowis75/PRIM/raw/master/Experiments/Arduino/06\\_THERMO\\_DI\\_SPLAY/04\\_Zdrojove\\_kody\\_prikladu/lib/Dht11.zip](https://github.com/Nowis75/PRIM/raw/master/Experiments/Arduino/06_THERMO_DI_SPLAY/04_Zdrojove_kody_prikladu/lib/Dht11.zip).
- ② Deklarace čidla **dht11** s názvem **cidlo**.
- ③ Deklarace proměnné **dhtpin**, což je číslo pinu na který je připojen signální vývod **S** čidla.

- ④ Ve funkci **setup()** se nastavuje přenosová rychlosť v bitech za sekundu (baud) pro sériový přenos. Nastavení probíhá pomocí **Serial.begin()**. Pro komunikaci s počítačem lze požít některou z rychlostí: 200, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 nebo 115200. Díky tomu lze na sériovém monitoru vypisovat různá hlášení.
- ⑤ Čtení dat z čidla s názvem **cidlo**. Funkce **read()** má jediný parametr a tím je číslo pinu, na který je připojen signální vývod čidla. V tomto příkladu je číslo pinu uloženo v proměnné **dhtpin**.
- ⑥ Do sériového portu se pomocí funkce **print()** vytiskne pro lidi čitelný ASCII text. V našem případě je to text „**Teplo**ta=“.
- ⑦ Pomocí funkce **cidlo.temperature**, zjistíme aktuální teplotu na čidle a zároveň ji vytiskneme do sériového monitoru pomocí funkce **println()**.



Všimněte si, že se k výpisu používají dvě funkce **print()** a **println()**. Jaký je mezi nimi rozdíl? Pokud použijeme funkci **print()**, tak další znaky se vytisknou za řetězec vypsáný touto funkcí. Když se použije funkce **println()**, tak další řetězec bude vytisknutý na novu řádku.

- ⑧ Vypsání řetězce „**Vlhkost**=“.
- ⑨ Zjištěn vlhkosti pomocí funkce **cidlo.humidity** a vypsání zjištěné hodnoty.
- ⑩ Zastavení běhu programu na jednu sekundu, aby bylo ve smyčce dostatek času na další zjištění aktuálních hodnot.

## OTEVŘENÍ SÉRIOVÉHO MONITORU

Nahrajte program do desky Arduino. V Arduino IDE klikněte na ikonu  pro otevření sériového monitoru. Otevře se nové okno, ve kterém zobrazí aktuální hodnoty teploty a vlhkosti Obr. 6 - Okno sériového monitoru.



The screenshot shows the Arduino Serial Monitor window titled '/dev/tty.usbmodem1421'. The window displays a series of text lines: 'Vlhkost = 30', 'Teplota = 24', 'Vlhkost = 30'. Below the text area are three buttons: 'Autoscroll' (checked), 'Newline', and '9600 baud'.

Obr. 6 - Okno sériového monitoru



## NEZOBRAZUJÍ SE HODNOTY TEPLITOY A VLHKOSTI

**Zapojení čidla** – zkontrolujte zapojení teplotního čidla, aby signální vývod byl opravdu připojen k digitálnímu pinu na desce Arduino.

Pozor na připojení vodičů pro napájení (5V) a zemnění (GND)

**Vodiče** – zkontrolujte, zda jsou vodiče opravdu dobře zasunuty do kontaktního pole desky Arduino.

**Přenosová rychlosť** – zkontrolujte přenosovou rychlosť sériového portu v okně sériového monitoru, že je stejná jako v kódu ve funkci

**Serial.begin(9600).**

## NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

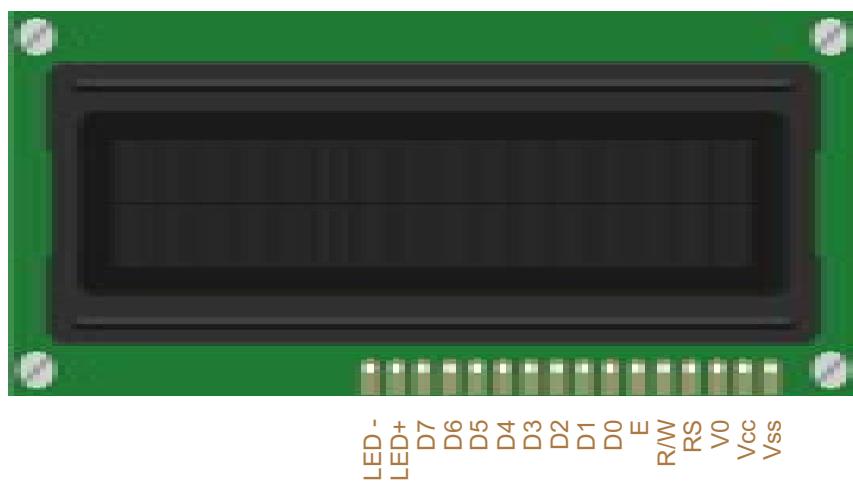


(Př. 1) Změňte programový kód předchozího příkladu tak, aby se kromě teploty ve stupních Celsia zobrazovala teplota i v Kelvinech a Fahrenheitech. Pro výpočet využijte vlastních funkcí.

## ZOBRAZENÍ TEPLOTY NA LCD displeji

### LCD displej

Použitý displej umožňuje zobrazovat alfanumerické znaky. Má 16 sloupců a dva řádky, tzn. dokáže zobrazit 32 znaků. Obsahuje velké množství pinů pro komunikaci s deskou Arduino a k napájení. Komunikační piny, ale není třeba zapojovat všechny.



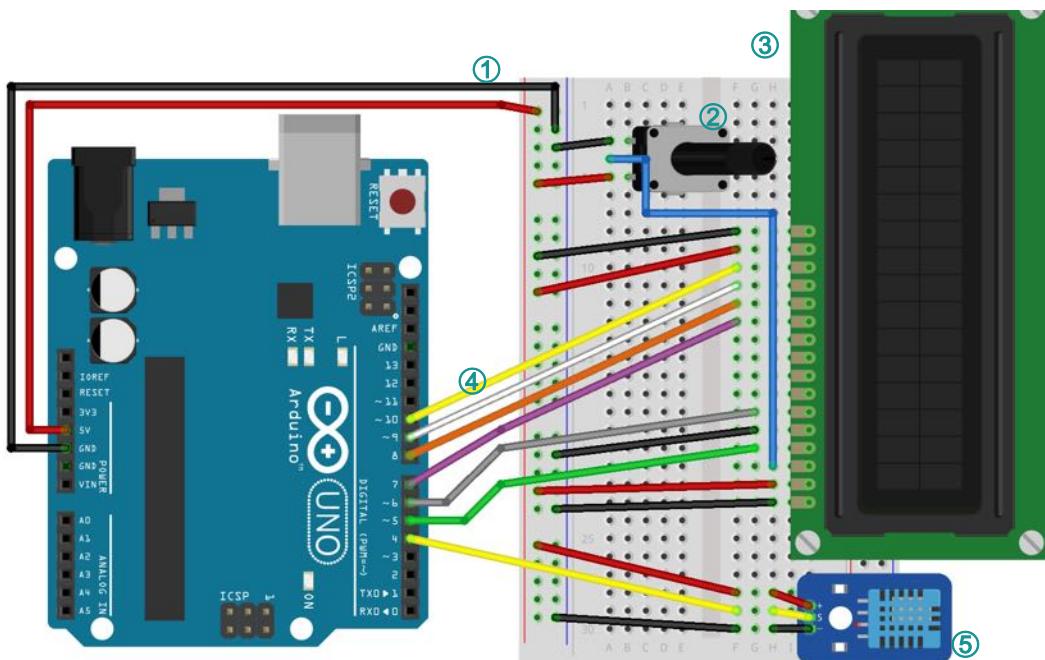
Vss	Připojení k zemi.	E	Arduino.
Vcc	Připojení k 5V.	D0 – D3	Datové piny, které se nepoužijí.
V0	Kontrast.	D4 – D7	Datové piny připojené k Arduino.
RS	Připojení k Arduinu.	A	Anoda podsvícení, 5V.
RW	Zem.	K	Katoda podsvícení, zem.

### POTENCIOMETR

Jedná se o proměnný odpor a v projektu je použit pouze pro nastavení kontrastu LCD displeje.

## ZAPOJENÍ OBVODU

Využijeme zapojení obvodu teplotního čidla a přidáme LCD displej pro zobrazení hodnot teploty a vlhkosti.



Obr. 7 - Zapojení teplotního čidla

- ① Vodič zemnění **GND** z desky Arduino zapojte do kontaktní desky k **modré** čáře. Vodič napájení připojte z Arduina z pinu **5V** do kontaktního pole. Tím se zpřehlední celé zapojení.
- ② Do kontaktního pole připojte potenciometr. Stačí zasunou vývody potenciometru přímo do pole. Jeden krajní vývod zapojte na zem a druhý krajní vývod na napájení 5V. Prostřední vývod připojte na vývod displeje **V0**. Bude sloužit k regulaci kontrastu LCD displeje.
- ③ LCD displej zasuňte do kontaktního pole.
- ④ Vývody z LCD displeje zapojeny následně: **Vss** na zem, **Vcc** na 5V, **V0** na prostřední vývod potenciometru, **RS** je připojen do pinu 5 na desce Arduino, **RW** je připojen k zemi, **E** je připojen na pin 6, **D4** je připojen na pin 7, **D5** je připojen na pin 8, **D6** na pin 9, **D7** na pin 10. Vývod **A** je připojen k 5V a **K** je připojen na zem.
- ⑤ Teplotní čidlo je připojeno stejně jako v přechozím příkladu, tj. vývod (+) je na napájení 5V, vývod (-) na zem a vývod (S) je připojen na pin desky Arduino **4**.

## PROGRAMOVÝ KÓD

Programový kód využívá knihovnu pro ovládání LCD displeje, která je v Arduino IDE již standardně nainstalovaná.

```
1 #include <dht11.h>
2 #include <LiquidCrystal.h>
3
4 int rsPin = 5;
5 int ePin = 6;
6 int d4Pin = 7;
7 int d5Pin = 8;
8 int d6Pin = 9;
9 int d7Pin = 10;
10 LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);
11
12 dht11 cidlo;
13 int dhtpin = 4;
14
15 void setup()
16 {
17     LCD.begin(16,2);
18     LCD.clear();
19     LCD.setCursor(0,0);
20     LCD.print("Teplota: ");
21
22     LCD.setCursor(0,1);
23     LCD.print("Vlhkost: ");
24 }
25
26 void loop()
27 {
28     cidlo.read(dhtpin);
29
30     LCD.setCursor(9,0);
31     LCD.print(cidlo.temperature);
32     LCD.setCursor(13,0);
33     LCD.print((char)223);
34     LCD.print("C");
35
36     LCD.setCursor(9,1);
37     LCD.print(cidlo.humidity);
38     delay(500);
39 }
```

- ① Příkazem `#include` se připojí knihovna pro teplotní čidlo.
- ② Stejným příkazem se připojí knihovna pro LCD displej.
- ③ Deklarace čísel pinů, které jsou využity pro připojení LCD displeje k desce Arduino. Displej můžeme připojit i na jiné piny, potom stačí změnit hodnoty těchto proměnných.
- ④ Funkce `LiquidCrystal` vytvoří objekt LCD displej o názvu LCD. Jeho parametry jsou deklarovány v proměnných viz přechozí krok. Proměnné bychom ani nemuseli používat, ale a musí být dodrženo uvedené pořadí.
- ⑤ Deklarace čidla `dht11` s názvem `cidlo`.
- ⑥ Deklarace proměnné `dhtpin` jako číslo pinu desky Arduino pro datový vývod tepelného. To je stejné jako v předchozím příkladu.
- ⑦ Spustí displej. Tento displej má 16 znaků v každém ze dvou řádků.
- ⑧ Funkce `clear()` preventivně vymaže display od starých znaků.
- ⑨ Funkce `setCursor()` nastaví kurzor na první znak a první řádek displeje. Číslování pozic LCD displeje začíná vždy od nuly.
- ⑩ Funkce `print()` napíše text „**Teplota:**“ na displej se začátkem definovaným v předchozím kroku.
- ⑪ Opět nastavíme pozici kurzoru, ale tentokrát na první pozici ve druhém řádku.
- ⑫ Na druhém řádku vypíšeme text „**Vlhkost:**“.
- ⑬ Přečteme hodnotu teploty.
- ⑭ Nastavíme kurzor na desátou pozici v prvním řádku, což je za textem „**Teplota:**“.
- ⑮ A vypíšeme naměřenou teplotu z čidla.
- ⑯ Nastavíme kurzor v prvním řádku na třináctou pozici. To je pozice pro vypsání jednotky teploty.
- ⑰ Vypíšeme znak stupňů celsia (`°`). To nám zajistí `(char)223`, což odpovídá ASCII hodnotě znaku stupňů. Pokud bychom napsali přímo znak stupňů, nemuselo by to fungovat.
- ⑱ Vypíšeme písmeno C pro. Výsledkem pak bude `°C`.
- ⑲ Nastavíme kurz v druhém řádku na devátou pozici za text „**Vlhkost:**“.
- ⑳ Vypíšeme aktuální hodnotu vlhkosti.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, na displeji se objeví teplota a vlhkost.



## NA LCD DISPLEJI NENÍ NIC VIDĚT

**Nastavení kontrastu** – ujistěte se, že kontrast displeje je v pořádku nastaven.

To provedete otočením potenciometru na jednu nebo druhou stranu.

**Zapojení LCD displeje** – zkontrolujte, zda jsou všechny vodiče opravdu správně zapojeny. Nezapomeňte, že piny v LCD displeji musí být v odpovídajícím pořadí.

## NEZOBRAZUJÍ SE HODNOTY NA LCD DISPLEJI

**Zapojení čidla** – zkontrolujte zapojení teplotního čidla, aby signální vývod byl opravdu připojen k digitálnímu pinu na desce Arduino.

Pozor na připojení vodičů pro napájení (5V) a zemnění (GND).

**Vodiče** – zkontrolujte, zda jsou vodiče opravdu dobře zasunuty do kontaktního pole desky Arduino.

## NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu. **Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.



(Př. 2) Změňte programový kód předchozího příkladu tak, aby se kromě teploty ve stupních Celsia, na LCD displeji, střídavě zobrazovala teplota v Kelvinech a Fahrenheitech.

## ŘEŠENÍ PŘÍKLADŮ

## PŘ. 1.

Řešení příkladu spočívá v přidání dvou vlastních funkcí, i když by se dal řešit i bez těchto funkcí. Pomocí nich se, ale program zpřehlední.

```
1 #include <dht11.h> // pripojení knihovny
2
3 dht11 cidlo;           // deklarace cidla
4
5 int dhtpin=4;          // deklarace signálního pinu
6
7 // funkce pro vypocet ve stupnich Fahrenheita
8 double Farenheit(double celsius){
9     return 1.8 * celsius + 32;
10 }
11
12 // funkce pro vypocet ve stupnich Kelvina
13 double Kelvin(double celsius){
14     return celsius - 273.15;
15 }
16
17 void setup(){
18     Serial.begin(9600);
19 }
20
21 void loop()
22 {
23     cidlo.read(dhtpin);
24
25     double celsia=cidlo.temperature; // zjistění stupnů Celsia
26     double kelviny=Kelvin(celsia); // zjistění stupnů Kelvina
27     double fahrenheity=Farenheit(celsia); // zjistění stupnů
28                                         // Fahrenheita
29
30     // Vypisy v seriovém monitoru
31     Serial.print("Teplota v Celsiech= ");
32     Serial.println(celsia);
33     Serial.print("Teplota v Kelvinech= ");
34     Serial.println(kelviny);
35     Serial.print("Teplota v Fahrenheitech= ");
36     Serial.println(fahrenheity);
37
38     Serial.print("Vlhkost = ");
39     Serial.println(cidlo.humidity);
40     delay(1000);
41 }
```

## PŘ. 2.

K vyřešení příkladu lze využít část kódu z příkladu (Př. 1). Jsou to zejména funkce pro výpočet teploty ve stupních Kevlina a Fahrenheita.

```
1 #include <dht11.h> // pripojeni knihovny
2 #include <LiquidCrystal.h>
3
4 int rsPin = 5;
5 int ePin = 6;
6 int d4Pin = 7;
7 int d5Pin = 8;
8 int d6Pin = 9;
9 int d7Pin = 10;
10 LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);
11
12 dht11 cidlo;
13 int dhtpin = 4;
14
15 // funkce pro vypocet ve stupnich Fahrenheita
16 double Farenheit(double celsius){
17     return 1.8 * celsius + 32;
18 }
19
20 // funkce pro vypocet ve stupnich Kelvina
21 double Kelvin(double celsius){
22     return celsius + 273.15;
23 }
24
25 void setup(){
26     LCD.begin(16,2);
27     LCD.clear();
28     LCD.setCursor(0,0);
29     LCD.print("Teplota: ");
30
31     LCD.setCursor(0,1);
32     LCD.print("Vlhkost: ");
33 }
34
35 void loop(){
36     cidlo.read(dhtpin);
37
38     double celsia=cidlo.temperature; // zjisteni stupnu Celsia
39     double kelviny=Kelvin(celsia);   // zjisteni stupnu Kelvina
40
41 }
```

```

42 double fahrenheity=Fahrenheit(celsia); // zjistění stupnů
43                                     // Fahrenheita
44
45 // v první rade se vypíše vlhkost
46 LCD.setCursor(9,1);
47 LCD.print(cidlo.humidity);
48
49 // vypis teploty ve stupních celsia
50 LCD.setCursor(9,0);
51 LCD.print(celsia);
52 LCD.setCursor(15,0); // musí se nastavit kurzor kde se
53 LCD.print(" ");     // zobrazuje staré hodnoty v K a F
54 LCD.setCursor(13,0);
55 LCD.print((char)223);
56 LCD.print("C");
57
58 delay(2000);
59
60 // vypis teploty ve stupních kelvina
61 LCD.setCursor(9,0);
62 LCD.print(kelviny);
63 LCD.setCursor(15,0);
64 LCD.print("K");
65
66 delay(2000);
67
68 // vypis teploty ve stupních fahrenheinta
69 LCD.setCursor(9,0);
70 LCD.print(fahrenheity);
71 LCD.setCursor(15,0);
72 LCD.print("F");
73
74 delay(2000);
75 }
76

```



Proč je v programu, na řádku **53** a **54** nastavení kurzoru a tisk mezer? Musíme si uvědomit, že program probíhá v neustálé smyčce, proto pokud bychom preventivně nevymazali staré hodnoty ze stupňů Kelvina a Fahrenheita, zůstávaly by tam staré znaky, protože ve stupních Celsia je řetězec kratší. Takto se konec displeje nahradí prázdným místem a teplota ve stupních Celsia se zobrazí v pořádku.

# AUTOMATICKÝ SKLENÍK

TATO ČÁST UKAZUJE, JAK SPOJIT PŘEDCHOZÍ NABYTÉ ZNALOSTI V JEDINÝ PROJEKT. TÍMTO PROJEKTEM JE VYTVOŘENÍ AUTOMATICKY OVLÁDANÉHO SKLENÍKU S VYUŽÍTM TEPLITNÍHO ČIDLA, SERVOMOTORU PRO OVLÁDÁNÍ VENTILACE A STEJNOSMĚRNÉHO MOTORU PRO VÉTRÁK.

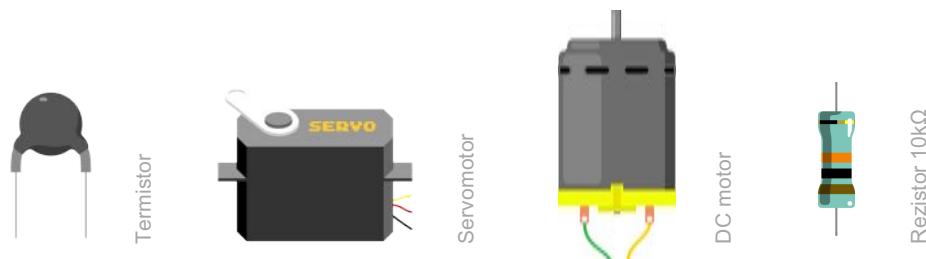
## CÍLE

- ① Upevnění nabytých znalostí pro práci s motory.
- ② Upevnění znalostí pro práci se senzory.
- ③ Zvládnutí práce s více komponentami najednou.
- ④ Spojení programátorských dovedností s konstrukčními dovednostmi.

Čas: **90 min**

Úroveň: ■■■■■

Vychází z: **5, 6**

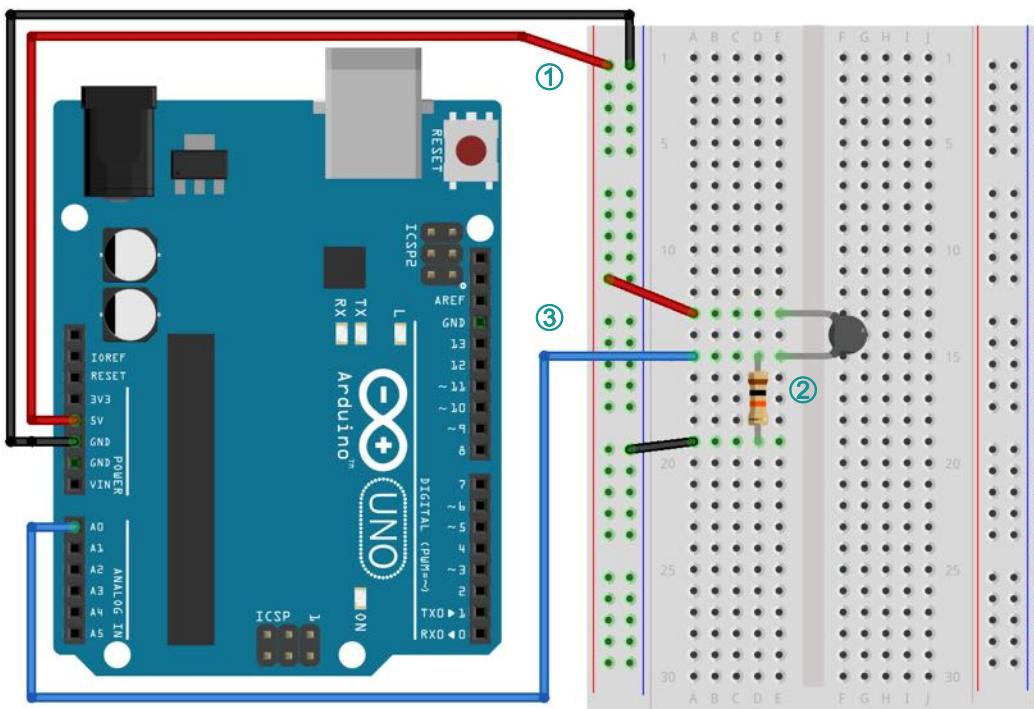


POUŽITÉ SOUČÁSTKY

Projekt automatického skleníku se skládá z několika oddělených částí:

- ① Snímání teploty pomocí termistoru.
- ② Ovládání střešního okna servomotorem.
- ③ Větrání prostřednictvím stejnosměrného motoru.

## ZAPOJENÍ OBVODU S TERMISTOREM



Obr. 8 - Zapojení termistoru

- ① Vodič zemnění **GND** z desky Arduino zapojte do kontaktní desky k **modré** čáře. Vodič napájení připojte z Arduina z pinu **5V** do kontaktního pole.
- ② Rezistor  $10\text{k}\Omega$  a termistor jsou zapojeny podle obvodu děliče napětí.
- ③ Signální vodič je zapojen do analogového pinu **A0**.

## PROGRAMOVÝ KÓD

Programový kód je díky knihovně pro čidlo velmi jednoduchý. Teplotu si zobrazíme v [Sériovém monitoru](#).

```
1 int termistorPin = 0;          ①
2 int Vout;                     ②
3 float R2 = 10000;            ③
4 float logR1, R1, T, Tc, Tf;  ④
5 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = ⑤
6 2.019202697e-07;
7
8 void setup() {
9     Serial.begin(9600);        ⑥
10 }
11
12 void loop() {
13     Vout = analogRead(termistorPin);      ⑦
14     R1 = R2 * (1023.0 / (float)Vout - 1.0); ⑧
15     logR1 = log(R1);
16     T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1)); ⑨
17     Tc = T - 273.15;                    ⑩
18     Tf = (Tc * 9.0)/ 5.0 + 32.0;       ⑪
19
20     Serial.print("Teplota: ");
21     Serial.print(Tf);                  ⑫
22     Serial.print(" F; ");
23     Serial.print(Tc);
24     Serial.println(" C");
25
26     delay(500);
27 }
```

- ① Deklarace proměnné **termistorPin** pro číslo analogového pinu desky Arduino, na který je připojen datový vodič z napěťového děliče.
- ② Deklarace proměnné **Vout**, která bude nabývat aktuální hodnoty v závislosti na teplotě.
- ③ Hodnota neměnného rezistoru **R2** u napěťového děliče, tj.  $10\text{k}\Omega$ .
- ④ Deklarace proměnných **logR2**, **R1**, **T**, **Tc**, **Tf**, které nabývají hodnot z výpočtu.
- ⑤ Deklarace konstant, které jsou určeny pro výpočet Steinhart-Hartovy rovnice. Tyto hodnoty jsou parametry, které se dají zjistit z katalogu součástky.

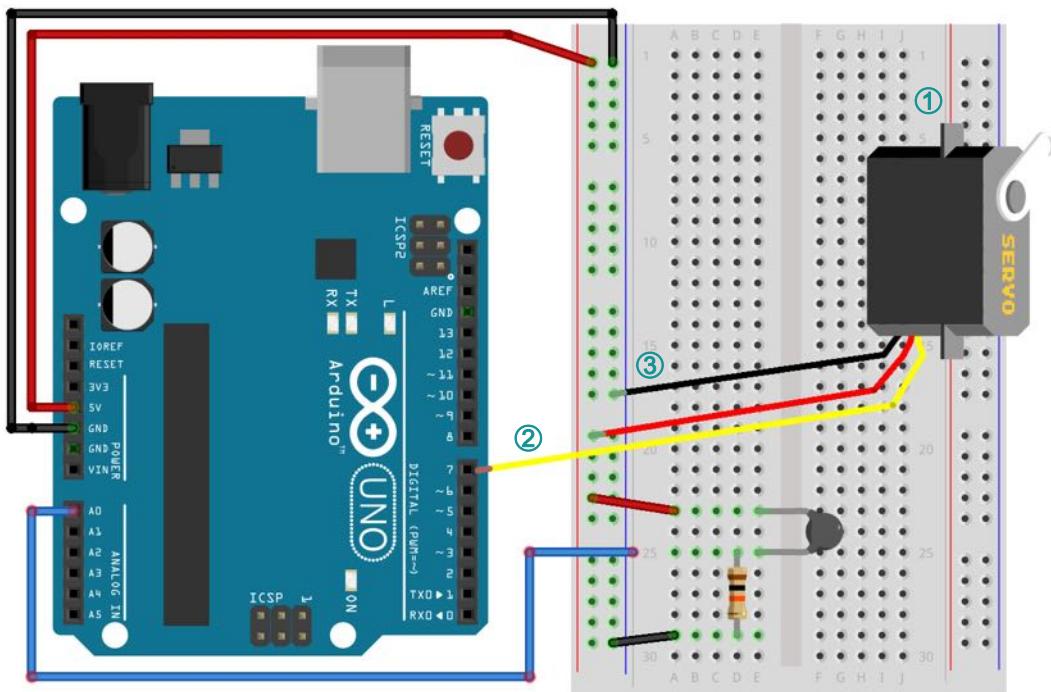
- ⑥ Nastavení rychlosti pro sériový přenos. Dílčí výsledky měření zobrazíme v sériovém monitoru.
- ⑦ Zjištění změny napětí, přečtené z analogového pinu desky Arduino a uložení do proměnné **Vout**.
- ⑧ Výpočet aktuálního odporu **R1** v závislosti na změně **Vout**.
- ⑨ Steinhart-Hartova rovnice, která slouží k výpočtu poměru odpor-teplota. Výsledek v proměnné **T** je ve stupních Kelvinia.
- ⑩ Převod ze stupňů Kelvinia na stupně Celsia - **Tc**.
- ⑪ Převod ze stupňů Celsia na stupně Fahreinheita - **Tf**.
- ⑫ Vypsání hodnot do sériového monitoru.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Otevřete sériový monitor. Pokud je vše v pořádku, měla by být vidět aktuální hodnota teploty naměřené termistorem.

## ZAPOJENÍ SERVOMOTORU

Zapojení servomotoru bude pokračovat z přechozího zapojení termistoru.



Obr. 9 - Zapojení servomotoru

- ① Servomotor, jak již známe, má tři vodiče. Napájení, zem a datový vodič. Konektor této vodičů typu **zásvuka**, musíme nastavit vodiči, pokud možná stejně barvy, typu **zástrčka**, aby šli zapojit do kontaktního pole a desky Arduino.
- ② Datový vodič zapojíme do digitálního pinu 7 desky Arduino.
- ③ Vodiče napájení a zemnění připojíme do kontaktního pole.

## PROGRAMOVÝ KÓD

Programový kód pro ovládání servomotoru je opět velmi jednoduchý. Servomotor bude pouze otevírat a zavírat střešní okno skleníku.

```

1 #include <Servo.h>                                ①
2
3 int termistorPin = 0;
4 int Vout;
5 float R2 = 10000;
6 float logR1, R1, T, Tc, Tf;
7 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 =
8 2.019202697e-07;
9
10 int servoPin = 7;                                 ②
11 int setTemp = 25;                                ③
12 int setTempRet=22;                               ④
13
14 Servo myservo;                                  ⑤
15
16 void setup() {                                   
17     Serial.begin(9600);
18
19     myservo.attach( servoPin );
20     myservo.write( 0 );
21 }
22
23 void loop() {                                   
24     Vout = analogRead(termistorPin);
25     R1 = R2 * (1023.0 / (float)Vout - 1.0);
26     logR1 = log(R1);
27     T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1));
28     Tc = T - 273.15;
29     Tf = (Tc * 9.0)/ 5.0 + 32.0;
30
31     Serial.print("Teplota: ");
32     Serial.print(Tf);
33     Serial.print(" F; ");
34     Serial.print(Tc);
35     Serial.println(" C");
36     delay(500);
37
38     if(T > setTemp ){
39         myservo.write(180);
40     }else if(T < setTempRet){
41         myservo.write(0);
42     }
43     delay(1000);
44 }
```

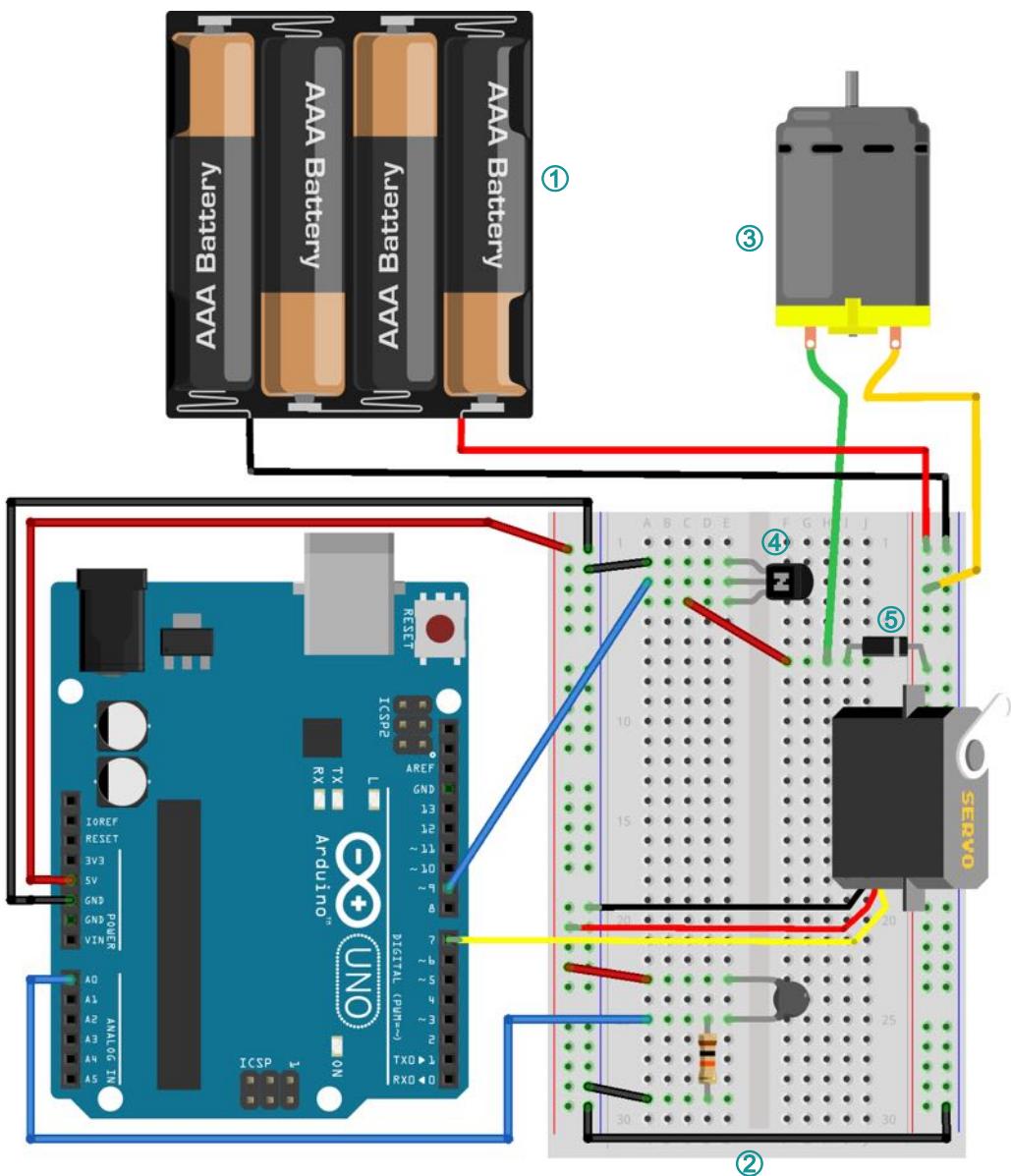
- ① Připojení knihovny pro ovládání servomotoru.
- ② Deklarace proměnné pro číslo pinu, na který je připojen datový vodič servomotoru.
- ③ Deklarace proměnné pro teplotu ve °C, při které dojde k **otevření** střešního okna.
- ④ Deklarace proměnné pro teplotu ve °C, při které dojde k **zavření** střešního okna.
- ⑤ Vytvoření instance **myservo** třídy pro servomotor.
- ⑥ Definice pinu, na který je motor připojen. Zde je na pinu **7**.
- ⑦ Nastavení výchozí pozice servomotoru. Při hodnotě **0** je střešní okno zavřené.
- ⑧ Testování pomocí podmínkového příkazu **if**, zda teplota nepřesáhla definovanou hodnotu v proměnné **setTemp**. Pokud ano, tak servomotor se nastaví do maximální pozice a střešní okno bude otevřené.
- ⑨ Druhá část podmínkového příkazu **if**, ve které se testuje, zda teplota neklesla pod stanovenou hodnotu uloženou v proměnné **setTempRet**. Pokud ano tak, se servomotor vrátí do pozice **0** a okno bude zavřené.



Opět nezapomeňte program zkompilovat a nahrát do desky Arduino. Zkuste rukou zahřívat termorezistor, nebo jej ochlazujte foukáním. V sériovém monitoru sledujte změnu teploty. Upravte případně hodnoty v proměnných **setTemp** a **setTempRet**, tak, aby došlo k aktualizaci pozice servomotoru.

## ZAPOJENÍ STEJNOSMĚRNÉHO MOTORU JAKO VĚTRÁKU

Zapojení DC motoru opět zakomponováno do existujícího zapojení s termistorem a servomotorem. Otáčky motoru, který je v projektu využitý jako větrák, budou závislé na aktuální teplotě.



Obr. 10 - Zapojení stejnosměrného motoru

- ① V první řadě přivedeme externí napájení do kontaktního pole. Napájení bude sloužit pouze k roztočení stejnosměrného motoru.

- ② V kontaktním poli propojíme země, které jsou přiveden z desky Arduino a z externího zdroje.
- ③ Stejnosměrný motor připojíme do kontaktního pole. Využijeme k tomu vodiče typu „zástrčka“. Vodiče k motoru připevníme buď připájením, nebo vytvořením oček a přelepením hmotou z tavné pistole nebo lepící páskou. Jeden vodič z motoru připojíme do kontaktního pole, k napájení. Druhý vodič připojíme do kontaktního pole, do střední části, ze které dále povedeme vodič k tranzistoru na **Kolektor**.
- ④ Tranzistor je vložen přímo do kontaktního pole. **Emitor** je připojený k zemnění. **Báze** je připojena k desce Arduino, do digitálního pinu **9**. Kolektor je připojen k usměrňovací diodě a k motoru.
- ⑤ Usměrňovací dioda je paralelně připojena k motoru a chrání obvod proti zpětnému proudu.

## PROGRAMOVÝ KÓD

Programový kód pro ovládání stejnosměrného motoru bude k regulaci otáček využívat aktuální teploty. Kód ovládání motoru je již zasazen do předchozího programu.

```

1 #include <Servo.h>
2
3 int termistorPin = 0;
4 int Vout;
5 float R2 = 10000;
6 float logR1, R1, T, Tc, Tf;
7 float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 =
8 2.019202697e-07;
9
10 int servoPin = 7;
11 int setTemp = 25;
12 int setTempRet=22;
13
14 Servo myservo;
15
16 int dcMotorPin=9;
17 int dcMotorSpeed=0;
18
19
20 void setup() {
21   Serial.begin(9600);

```



```

22
23     myservo.attach( servoPin );
24     myservo.write( 0 );
25
26     pinMode(dcMotorPin, OUTPUT)
27 }
28
29 void loop() {
30     Vout = analogRead(termistorPin);
31     R1 = R2 * (1023.0 / (float)Vout - 1.0);
32     logR1 = log(R1);
33     T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1*logR1));
34     Tc = T - 273.15;
35     Tf = (Tc * 9.0)/ 5.0 + 32.0;
36
37     Serial.print("Teplota: ");
38     Serial.print(Tf);
39     Serial.print(" F; ");
40     Serial.print(Tc);
41     Serial.println(" C");
42     delay(500);
43
44     if(T > setTemp ){
45         myservo.write(180);
46
47         dcMotorSpeed=map(T, setTemp, setTempRet, 32, 255); |—④
48         digitalWrite(dcMotorPin, dcMotorSpeed); |—⑤
49
50     }else if(T < setTempRet){
51         myservo.write(0);
52
53         digitalWrite(dcMotorPin,LOW); |—⑥
54
55     }
56     delay(1000);
57 }
```

- ① Deklarace proměnné **dcMotorPin**, která obsahuje číslo digitálního pinu pro připojení báze tranzistoru.
- ② Deklarace proměnné **dcMotorSpeed** pro uložení rychlosti otáček stejnosměrného motoru.
- ③ Definice pinu pro stejnosměrný motor.

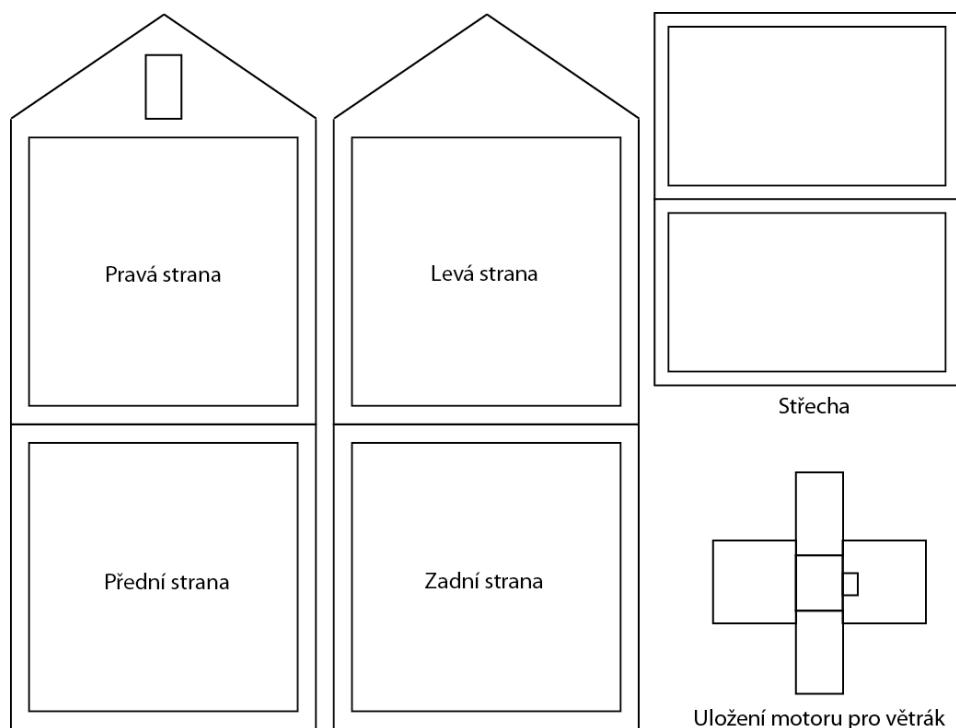
- ④ Namapování hodnot pro rychlosť motoru. Využívá se krajních hodnot teplot, pri ktorých dochází ke zmene otáček.
- ⑤ Funkcia **analogWrite()** posíla na výstup definovaný konštantou **dcMotorPin** nastavenou hodnotu rychlosťi **dcSpeedMotor**.
- ⑥ Pokud bude teplota nižšia než definovaná v konštante **setTempRet**, motor sa zastaví.



Pokud jste zvládli zapojit a naprogramovať všechny časti pro ovládání skleníku, můžete si takový model stolního skleníku vyrobit. Všechny elektronické časti pak do něj zabudovat.

## MODEL SKLENÍKU

Pro vytvoření skleníku budeme potřebovat: karton (stará krabice), průsvitnou fólii, tavnou pistoli, lepidlo, kancelářskou sponku, lepící pásku, nůžky.



Obr. 11 - Šablona skleníku

## PAPÍROVÁ KONSTRUKCE

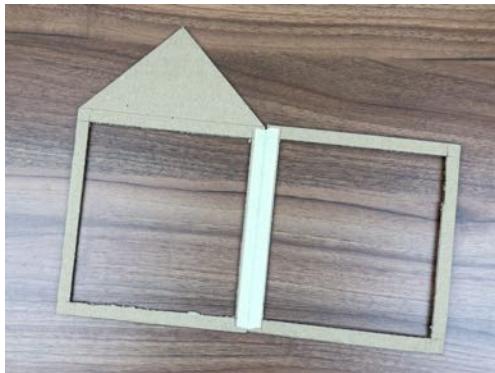
V této kapitole je ukázka. Jak sestrojit jednoduchý model skleníku, který lze po jeho sestavení umístit například na pracovní stůl.

V první řadě si překreslete na pevný karton model skleníku podle obrázku Obr. 11 - Šablona skleníku. Šablonu pak vystříhněte tak, aby vám vzniklo sedm samostatných dílů.



Obr. 12 – Vystřížená šablona

Nyní slepte pomocí lepící pásky vždy dvojici stěn: Pravá strana – Přední strana a Levá strana – Zadní strana, Obr. 13 – Slepění dílů stěn skleníku. Díly střechy také slepte k sobě.



Obr. 13 – Slepění dílů stěn skleníku



Obr. 14 – Nalepení fólie na stěny

Na všechny stěny, včetně střechy nalepte průhlednou fólii. Využijte k tomu obyčejného lepidla na papír, Obr. 14 – Nalepení fólie na stěny.

Dvojici dílů slepte páskou dohromady - Obr. 13.

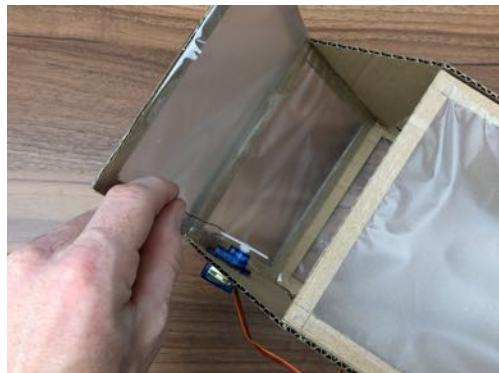


Obr. 135 – Sestavený skleník bez střechy



Obr. 16 – Přilepení střechy

Na stěny skleníku přilepte pomocí tavné pistole jednu polovinu střechy. Druhá polovina bude volná.



Obr. 17 – Připevnění sponky k servu

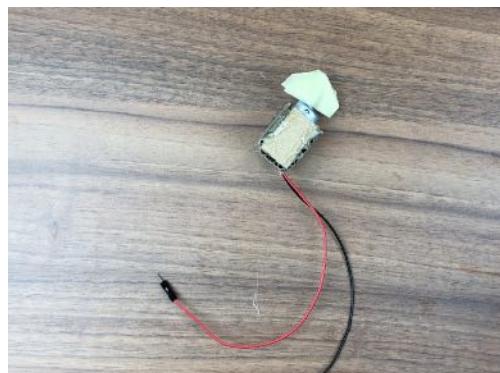


Obr. 18 – Spojení serva se střechou

Volná část střechy bude pomocí sponky spojená se servomotorem, který ji bude otevírat a zavírat. Sponku na ohýbejte podle obrázku Obr. 18 – Spojení serva se střechou.



Obr. 19 – Slepení pouzdra na motor



Obr. 20 – Motor s vrtulí

Poslední částí skleníku je větrák. Tavnou pistolí slepte úložnou kapsu, do které umístíte stejnosměrný motor. Na hřídel motoru přilepte lepící pásku jako vrtuli.

## 7. MATICOVÝ LED DISPLEJ

V TÉTO LEKCI SE ZAMĚŘÍME NA PRAKTICKÉ VYUŽITÍ MATICOVÉHO LED DISPLEJE. SEZNÁMÍME SE JAK TENTO displej ZAPojit A JEJ TAKÉ PROGRAMOVAT. TYTO ZDÁNLIVĚ JEDNODUCHÉ displeje mají i v dnešní době své využití pro svou čitelnost, technickou nenáročnost a poměrně snadné programování.

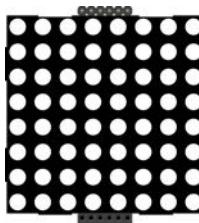
### CÍLE

- ① Zapojení maticového displeje 8x8.
- ② Programování maticového displeje pro zobrazení jednoduchých symbolů.
- ③ Zapojení a programování maticového displeje ve spojení s potenciometry.
- ④ Pro zručné studenty je určen úkol pro naprogramování hry Ping-Pong.
- ⑤ Seznámení s akcelerometrem.
- ⑥ Programování akcelerometru ve spojení s maticovým displejem.

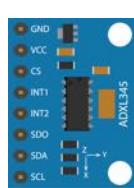
Čas: **5x45 min**

Úroveň: ■■■■■

Vychází z: **3, 4, 5**



Maticový displej 8x8



Akcelerometr



Potenciometr 2x

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví prakticky jediný obvod, do kterého budou v dalších hodinách pouze přidávat další komponenty. V této hodině bude hlavním úkolem pochopit princip maticového displeje. V programu využijí již získané vědomosti týkající se polí a jejich procházení. Součástí jsou jednoduché samostatné úkoly.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, maticový LED displej 8x8, vodiče.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 7.
- ⑤ Pracovní listy pro studenty.

## 1. KROK 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní vašeho kurzu bude si ukázat praktické možnosti využití maticového LED displeje.

### ZEPTEJTE SE STUDENTŮ

➔ **Kde jste se setkali s maticovým displejem?**

Např. ve veřejné dopravě, venkovní reklamě, na stadionu při zobrazení výsledků..

➔ **V čem byste spatřovali výhody maticového displeje?**

Jednoduchost, čitelnost, cena.



Studenti ať nejdříve zapojí displej pro jeho otestování. Zapojení je velice jednoduché. Ať využijí zobrazeného schématu, které je součástí pracovních listů nebo přiložené prezentace, kterou lze promítat pomocí dataprojektoru.

## 2. KROK 10 minut

Nyní studentům ukažte prostřednictvím dataprojektoru nebo pracovního listu základní kód, který zajistí blikání jediné diody na displeji.



### RYCHLÝ TIP

- Vysvětlete, princip maticového displeje tak, že se jedná o samostatné LED diody, které jsou vzájemně propojeny. Využijte k tomu přiložené schéma a tabulku zapojených pinů.

```
1 int pinA=2;
2 int pinB=6;
3
4 void setup() {
5     pinMode(pinA,OUTPUT);
6     pinMode(pinB,OUTPUT);
7     digitalWrite(pinA,HIGH);
8     digitalWrite(pinB,HIGH);
9 }
10
11 void loop() {
12     digitalWrite(pinB,LOW);
13     delay(200);
14     digitalWrite(pinB,HIGH);
15     delay(200);
16 }
```

Studenti ať program nahrají do desky a odzkouší, zda se dioda v horním levém rohu rozbliká.

#### RYCHLÝ TIP

→ Nezapomeňte, že při každé změně se musí program opět nahrát do desky.



### 3. KROK 20 minut

K objasnění principu maticového displeje ať studenti zkusí vyřešit následující úkol.



#### ÚKOL PRO STUDENTY

- A) Upravte obvod zapojení displeje a programový kód předchozího příkladu tak, aby blikaly i diody ve všech rozích stejně jako dioda první. Řešení je tohoto úkolu spočívá v zapojení odpovídajících výstupů displeje do desky Arduino.

### 4. KROK 15 minut

Pro stejné zapojení displeje z předchozího příkladu ať studenti vyřeší následující úkol.



#### ÚKOL PRO STUDENTY

- B) Změňte programový kód předchozího příkladu tak, aby diody v protilehlých rozích blikaly střídavě.

# PRACOVNÍ LIST – MATICOVÝ DISPLEJ

LED MATICOVÝ DISPLEJ JE SOUČÁSTKA, KTERÁ OBSAHUJE DIODY USPOŘÁDANÉ DO SLOUPCŮ A ŘÁDKŮ. TVOŘÍ DVOJROZMĚRNÉ POLE A VELIKOST ZÁVISÍ NA POČTU DIOD. NEJČASTĚJI JSOU POUŽÍVANÉ displeje o velikosti 8x8.

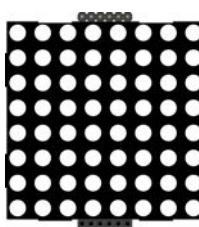
## CO SE NAUČÍTE

- ① Zapojit maticový displej.
- ② Zopakujete si zapojování LED diod.
- ③ Programovat maticový displej.

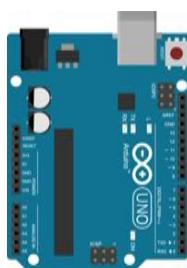


## CO BUDETE POTŘEBOVAT

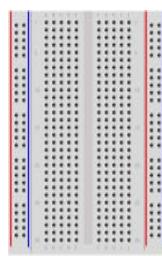
- ① Maticový displej.
- ② Desku Arduino.
- ③ Kontaktní pole.
- ④ Vodiče typu zásuvka-zásuvka.



Maticový displej 8x8



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

## OTÁZKY PRO VÁS

- Kde jste se setkali s maticovým displejem?
- V čem byste spatřovali výhody maticového displeje?



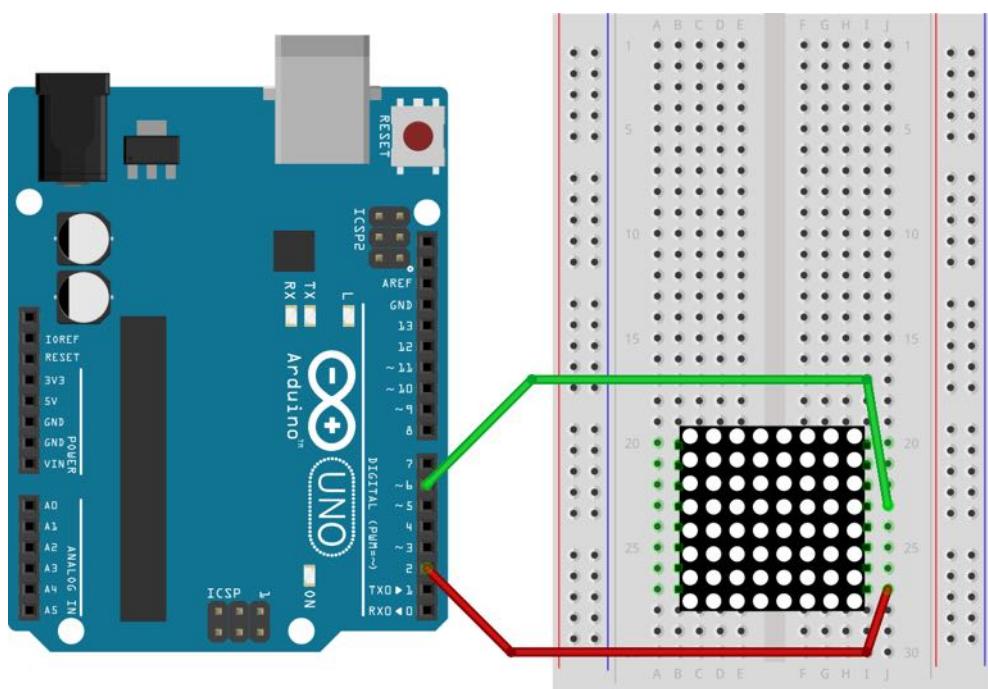
## DEJTE SI POZOR

- Pozor si dejte při vkládání displeje do kontaktního pole. Dejte pozor, abyste nožičky displeje zbytečně neohnuli. Všimněte si, že displej je na středu pole. Tím jsou jeho kontakty odděleny.



## A JDĚTE NA TO ...

- ① Podle přiloženého schématu zapojte obvod s maticovým displejem.



- ② Napište program, který rozbliká první diodu displeje.

```
1 int pinA=2;
2 int pinB=6;
3
4 void setup() {
5     pinMode(pinA,OUTPUT);
6     pinMode(pinB,OUTPUT);
7     digitalWrite(pinA,HIGH);
8     digitalWrite(pinB,HIGH);
9 }
10
11 void loop() {
12     digitalWrite(pinB,LOW);
13     delay(200);
14     digitalWrite(pinB,HIGH);
15     delay(200);
16 }
```



#### ÚKOL PRO VÁS

- A) Upravte obvod zapojení displeje a programový kód předchozího příkladu tak, aby blikaly i diody ve všech rozích stejně jako dioda první.

- ③ Pokud jste úkoly splnili a vše funguje, jak má zkuste si v rámci dalšího úkolu zapojit a naprogramovat ještě jeden obvod.



#### ÚKOL PRO VÁS

- B) Změňte programový kód předchozího příkladu tak, aby diody v protilehlých rozích blikaly střídavě.

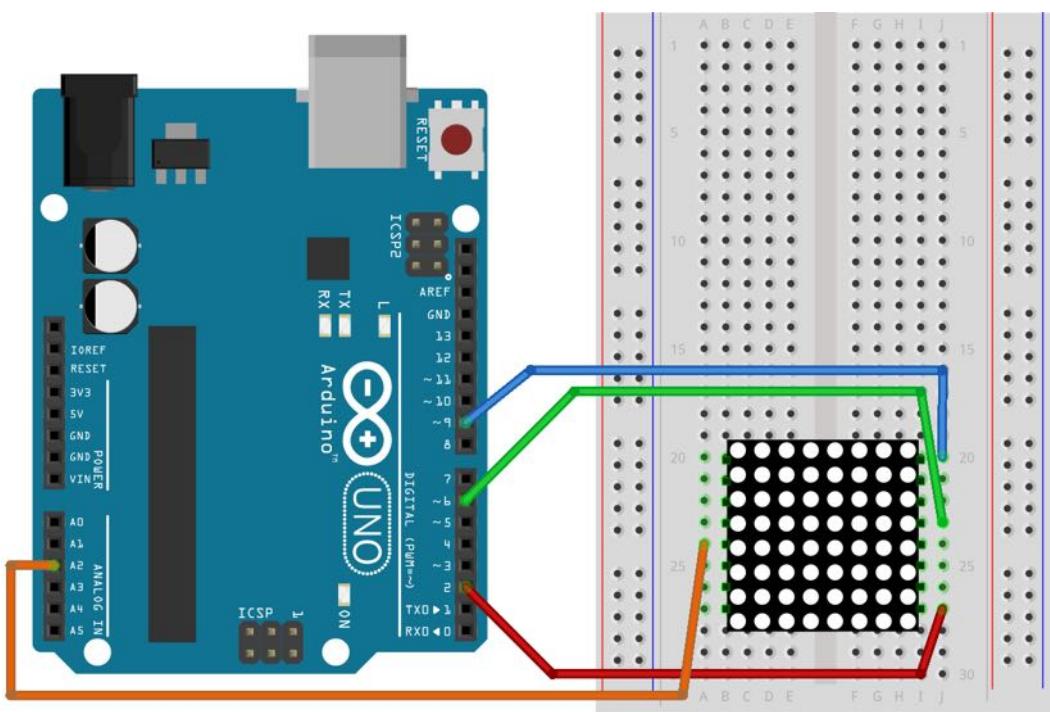


### NEZAPOMEŇTE

Nahrajte program do desky Arduino, kliknutím na ikonu ➔

# ŘEŠENÍ ÚLOH

Úkol A)



```
1 int pinA=2;
2 int pinB=6;
3
4 int pinC=9;
5 int pinD=A2;
6
7 void setup() {
8     pinMode(pinA,OUTPUT);
9     pinMode(pinB,OUTPUT);
10    digitalWrite(pinA,HIGH);
11    digitalWrite(pinB,HIGH);
12
13    pinMode(pinC,OUTPUT);
14    pinMode(pinD,OUTPUT);
15    digitalWrite(pinC,HIGH);
16    digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21     digitalWrite(pinB,LOW);
22     digitalWrite(pinD,HIGH);
23     delay(200);
24
25     digitalWrite(pinB,LOW);
26     digitalWrite(pinD,HIGH);
27     delay(200);
28 }
```

Úkol B)

```
1 int pinA=2;
2 int pinB=6;
3
4 int pinC=9;
5 int pinD=A2;
6
7 void setup() {
8     pinMode(pinA,OUTPUT);
9     pinMode(pinB,OUTPUT);
10    digitalWrite(pinA,HIGH);
11    digitalWrite(pinB,HIGH);
12
13    pinMode(pinC,OUTPUT);
14    pinMode(pinD,OUTPUT);
15    digitalWrite(pinC,HIGH);
16    digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21     digitalWrite(pinB,HIGH); // změna na HIGH
22     digitalWrite(pinD,LOW); // změna na LOW
23     delay(200);
24
25     digitalWrite(pinB,LOW);
26     digitalWrite(pinD,HIGH);
27     delay(200);
28 }
```

# PRŮVODCE HODINOU II



Tentokrát studenti budou pracovat s kompletně zapojeným maticovým displejem. Toto zapojení budou používat pro řešení několika příkladů. Naučí se zejména pracovat s vícerozměrným polem, jak jím procházet a přistupovat k hodnotám.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ④ Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, maticový LED displej 8x8, vodiče.
- ⑤ Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ⑥ Pokud je k dispozici, tak dataprojektor.
- ⑦ Prezentace k lekci 7.
- ⑧ Pracovní listy pro studenty.

## 1. KROK 🕒 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní hodiny bude si ukázat další možnosti ve využití maticového LED displeje.

### RYCHLÝ TIP

- Pro připomenutí ukažte studentům tabulku zapojení pinů a zeptejte se, zda byli schopní maticový displej zapojit sami.



## 2. KROK 🕒 10 minut

Ať studenti zapojí displej pro plnou funkcionalitu displeje podle přiloženého schématu v pracovním listu nebo promítaném prostřednictvím dataprojektoru.



### POZOR NA ZAPOJENÍ DISPLEJE

- Při zapojování displeje s větším množstvím vodičů upozorněte studenty, aby zapojení prováděli obzvláště pečlivě.

## 3. KROK 🕒 10 minut

Po zapojení obvodu mohou studenti začít psát programový kód. Uvedený kód postupně rozsvěcí v každém sloupci diody. Tím dojde ke kompletnímu otestování displeje.

```
1 const int row[8] = {2, 7, 19, 5, 13, 18, 12, 16};  
2 const int col[8] = {6, 11, 10, 3, 17, 4, 8, 9};  
3  
4 void setup(){  
5     for(int i = 0; i < 8; i++){  
6         pinMode(col[i], OUTPUT);  
7         pinMode(row[i], OUTPUT);  
8         digitalWrite(col[i], HIGH);  
9         digitalWrite(row[i], LOW);  
10    }  
11 }  
12  
13 void loop(){  
14     for(int j = 0; j<8;j++) {  
15         digitalWrite(col[j],LOW);  
16         for(int k = 0;k<8;k++){  
17             digitalWrite(row[k],HIGH);  
18             delay(200);  
19         }  
20         for(int i = 0;i<8;i++){  
21             digitalWrite(row[i],LOW);  
22             digitalWrite(col[i],HIGH);  
23         }  
24     }  
25 }
```



#### OTÁZKY PRO STUDENTY

- At' studenti po nahráti programu do desky, jak se chovají diody na displeji.
- Zeptejte se, při jaké kombinaci hodnot ve funkci `digitalWrite()` dioda na displeji svítí nebo je zhasnutá?

### 4. KROK 10 minut

Následující příklady upevňují znalosti týkající se principu programování maticového displeje.



#### ÚKOL PRO STUDENTY

- A) Upravte (optimalizujte) programový kód tak, aby se aktualizace a mazání displeje prováděla ve dvou vámí deklarovaných funkcích.

### 5. KROK 10 minut

V návaznosti na předchozí úkol, kdy by studenti měli vytvořit dvě funkce a tím tak optimalizovat kód i pro pozdější použití, stačí v následujícím úkolu provést změny v pořadí zapínání diod displeje.



#### ÚKOL PRO STUDENTY

- B) Upravte programový kód tak, aby se v celém, rozsvíceném displeji postupně posouval vypnutý sloupec a při tomto vypnutém sloupci projížděl vypnutý řádek.

# PRACOVNÍ LIST – MATICOVÝ DISPLEJ - II

V TÉTO ČÁSTI BUDETE POKRAČOVAT V ZAPOJOVÁNÍ A PROGRAMOVÁNÍ Maticového displeje. Tentokrát se již naučíte ovládat celý displej a vyzkoušíte si, jak pracovat s jednotlivými diodami.

## CO SE NAUČÍTE

- ① Zapojit celý maticový displej.
- ② Zopakujete cyklus **for**.
- ③ Programovat průchod polem pro rozsvícení diod maticového displeje.

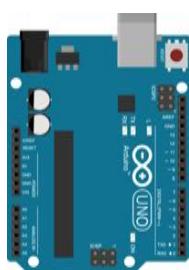


## CO BUDETE POTŘEBOVAT

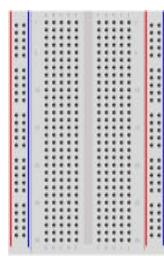
- ① Maticový displej.
- ② Desku Arduino.
- ③ Kontaktní pole.
- ④ Vodiče typu zásuvka-zásuvka.



Maticový displej 8x8



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

## ZOPAKUJTE SI ...

- ① Podívejte se na níže uvedenou tabulku *Tab. 1* a promyslete si, jak zapojit celý maticový displej.

Matrice pin	Řádek	Sloupec	Arduino pin
1	5	-	13
2	7	-	12
3	-	2	11
4	-	3	10
5	8	-	16
6	-	5	17
7	6	-	18
8	3	-	19
9	1	-	2
10	-	4	3
11	-	6	4
12	4	-	5
13	-	1	6
14	2	-	7
15	-	7	8
16	-	8	9

*Tab. 1 - Rozložení pinů*

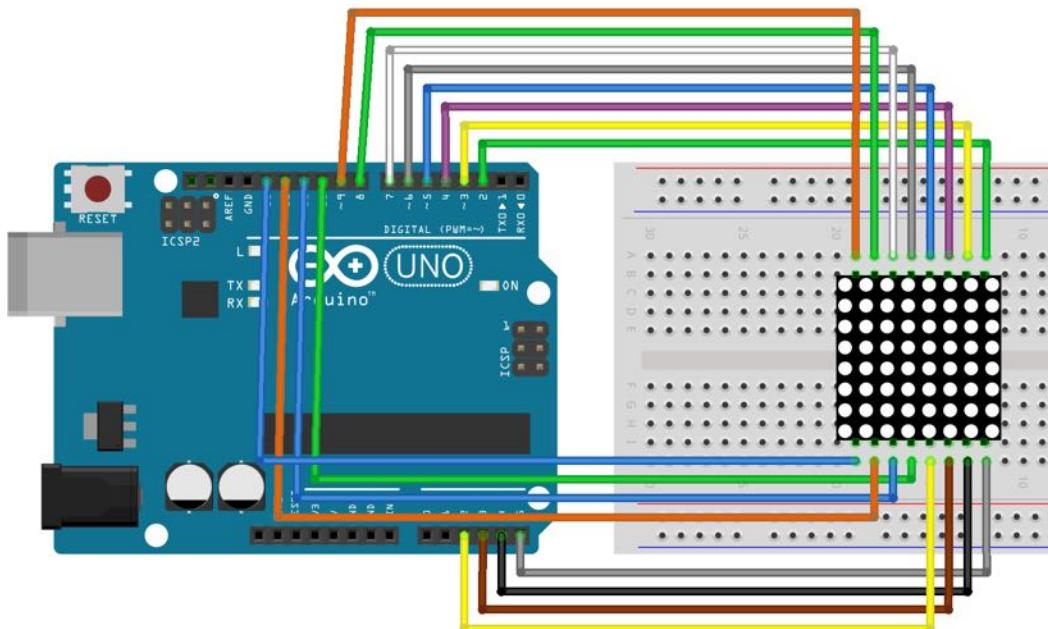
### DEJTE SI POZOR

➔ Všimněte si pořadí pinů ve sloupci Arduino pin. Přestože se používá označení analogových vstupů A2 – A5, lze je definovat jako číselné hodnoty 14-17.



## A JDĚTE NA TO ...

- ② Pokud si netroufnete zapojit displej podle tabulky pinů Tab. 1, využijte následující schéma.



### DEJTE SI POZOR

➔ Pozor si dejte při vkládání displeje do kontaktního pole. Dejte pozor, abyste nožičky displeje zbytečně neohnuli. Všimněte si, že displej je na středu pole. Tím jsou jeho kontakty odděleny.



- ③ Napište a nahrajte následující program do desky Arduino.

```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {
```

```

6   6, 11, 10, 3, 17, 4, 8, 9
7 };
8
9 void setup(){
10    for(int i = 0; i < 8; i++){
11        pinMode(col[i], OUTPUT);
12        pinMode(row[i], OUTPUT);
13        digitalWrite(col[i], HIGH);
14        digitalWrite(row[i], LOW);
15    }
16 }
17
18 void loop(){
19    for(int j = 0; j<8;j++) {
20        digitalWrite(col[j],LOW);
21        for(int k = 0;k<8;k++){
22            digitalWrite(row[k],HIGH);
23            delay(200);
24        }
25        for(int i = 0;i<8;i++){
26            digitalWrite(row[i],LOW);
27            digitalWrite(col[i],HIGH);
28        }
29    }
30 }

```

### OTÁZKA PRO VÁS

- ➔ Pokud jste v pořádku nahráli program do desky, popište, jak se chovají diody na displeji.
- ➔ Při jaké kombinaci hodnot ve funkci `digitalWrite()` dioda na displeji svítí nebo je zhasnutá?



- ④ Pokud se vám podařilo otestovat displej podle předchozího základního programu, vyřešte následující úkoly. Úkoly se týkají pouze úpravy programového kódu, není nutné měnit zapojení displeje.



#### ÚKOL PRO VÁS

- ➔ A) Upravte (optimalizujte) programový kód tak, aby se aktualizace a mazání displeje prováděla ve dvou vámi deklarovaných funkcích.
- ➔ B) Upravte programový kód tak, aby se v celém, rozsvíceném displeji postupně posouval vypnutý sloupec a při tomto vypnutém sloupci projížděl vypnuty řádek.

# ŘEŠENÍ ÚLOH

Úkol A)

```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {  
6     6, 11, 10, 3, 17, 4, 8, 9  
7 };  
8  
9 void setup(){  
10    for(int i = 0; i < 8; i++){  
11        pinMode(col[i], OUTPUT);  
12        pinMode(row[i], OUTPUT);  
13        digitalWrite(col[i], HIGH);  
14        digitalWrite(row[i], LOW);  
15    }  
16 }  
17  
18 void loop(){  
19    refreshScreen();  
20 }  
21  
22 void refreshScreen(){  
23    for(int j = 0; j<8;j++){  
24        digitalWrite(col[j], LOW);  
25        for(int k = 0; k<8; k++){  
26            digitalWrite(row[k], HIGH);  
27        }  
28        Clear();  
29    }  
30 }  
31  
32 void Clear(){  
33    for(int i = 0; i<8; i++){  
34        digitalWrite(row[i],LOW);  
35        digitalWrite(col[i],HIGH);  
36    }  
37 }
```

Úkol B)

```
1 const int row[8] = {
2     2, 7, 19, 5, 13, 18, 12, 16
3 };
4
5 const int col[8] = {
6     6, 11, 10, 3, 17, 4, 8, 9
7 };
8
9 void setup(){
10    for(int i = 0; i < 8; i++){
11        pinMode(col[i], OUTPUT);
12        pinMode(row[i], OUTPUT);
13        digitalWrite(col[i], HIGH);
14        digitalWrite(row[i], LOW);
15    }
16}
17
18 void loop(){
19     refreshScreen();
20 }
21
22 void refreshScreen(){
23    for(int j = 0; j<8;j++){
24        digitalWrite(row[j], LOW);
25        for(int k = 0; k<8; k++){
26            digitalWrite(col[k], HIGH);
27            delay(100);
28            digitalWrite(col[k], LOW);
29        }
30        digitalWrite(row[j], HIGH);
31    }
32 }
```

# PRŮVODCE HODINOU III



Studenti opět budou pracovat s kompletně zapojeným maticovým displejem. Tentokrát se naučí pracovat s vícerozměrným polem, kdy budou na displeji zobrazovat jednoduché symboly.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, maticový LED displej 8x8, vodiče.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 7.
- ⑤ Pracovní listy pro studenty.

## 1. KROK 🕒 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní hodiny bude si ukázat, jak se dají na maticovém displeji zobrazit jednoduché symboly.

### RYCHLÝ TIP

- ➔ Pokud mají studenti zapojený obvod s displejem z předchozí hodiny, můžete přistoupit rovnou k programovému kódu.
- ➔ V opačném případě si studenti v rámci opakování obvod sestaví podle schématu, které mají v pracovním listu nebo promítaného pomocí dataprojektoru.



## 2. KROK ⏳ 10 minut

Studenti ať napíší následující programový kód. Nevysvětlujte jim jeho princip, na to se zeptáte až jej spustí.



### RYCHLÝ TIP

- Ať studenti použijí některý z předchozích příkladů. Tzn. otevřou si jej v prostředí IDE a následně uloží pod novým jménem. Programový kód stačí pouze jednoduše inovovat.

## 3. KROK ⏳ 5 minut

Věnujte chvíliku času pro vysvětlení programového kódu formou otázek.



### OTÁZKY PRO STUDENTY

- V čem se liší programový kód pro zobrazení symbolu od kódu z předchozích kapitol?
- Jak si myslíte že vznikl tvar srdce na displeji. Kde je nadefinován?

## 4. KROK ⏳ 5 minut

Studenti budou řešit následující úkol. Je velmi jednoduchý, spočívá pouze v úpravě pole **image**.



### ÚKOL PRO STUDENTY

- A) Upravte programový kód tak, aby se na displeji zobrazil symbol smajlíku.



Definice tvaru symbolů je velmi snadné. Studenti mohou využít nástroj, pomocí něhož si symbol „naklikají“ a následně použijí vygenerované dvourozměrné pole vypnutých/zapnutých diod, které vloží do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>

## 5. KROK 🕒 15 minut



### ÚKOL PRO STUDENTY

- B) Změňte programový kód tak, aby se střídavě zobrazoval symbol velkého a malého srdce.

# PRACOVNÍ LIST – MATICOVÝ DISPLEJ - III

V TÉTO ČÁSTI BUDETE POKRAČOVAT ZEJMÉNA V PROGRAMOVÁNÍ MATICOVÉHO displeje. TENTOKRÁT SE NAUČÍTE PRACOVAT S VÍCEROZMĚRNÝM POLEM, POMOCÍ KTERÉHO SI ZOBRAZÍTE JEDNODUCHÉ SYMBOLY.

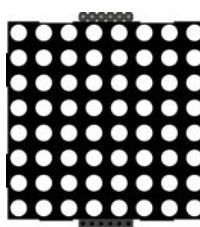
## CO SE NAUČÍTE

- ① Zopakujete si cyklus **for**.
- ② Pracovat s vícerozměrným polem.
- ③ Naučíte se princip zobrazování symbolů na maticovém displeji.

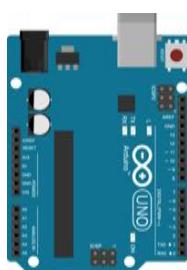


## CO BUDETE POTŘEBOVAT

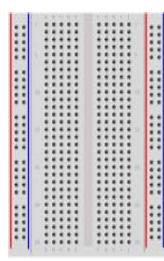
- ① Maticový displej.
- ② Desku Arduino.
- ③ Kontaktní pole.
- ④ Vodiče typu zásuvka-zásuvka.



Maticový displej 8x8



Deska Arduino

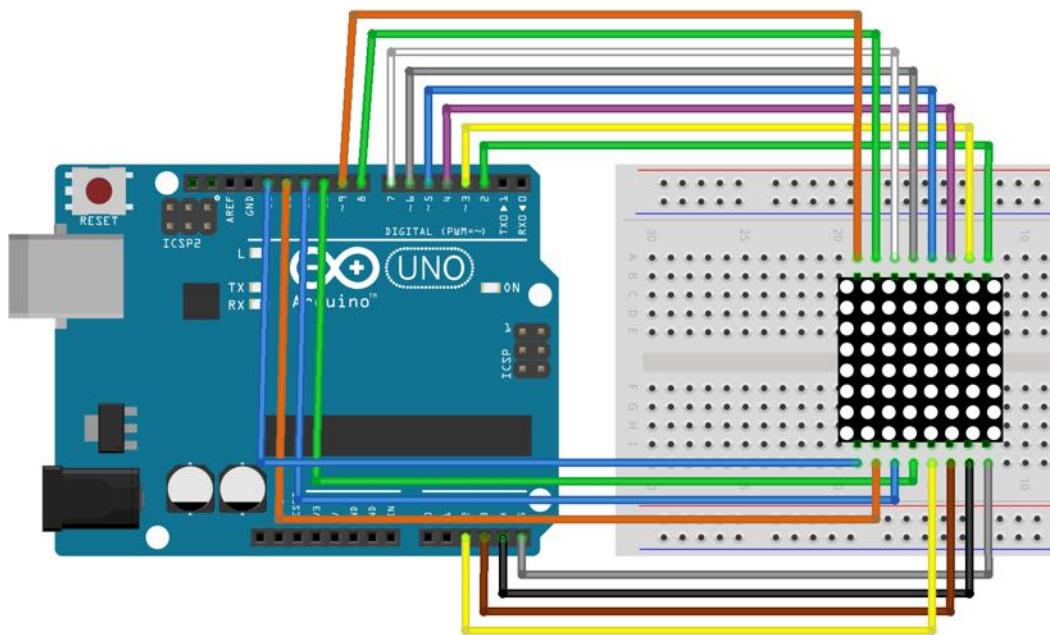


Kontaktní pole

POUŽITÉ SOUČÁSTKY

## RYCHLÝ ÚVOD ...

- ① Pokud nemáte sestavený obvod s maticovým displejem, tak v rámci opakování jej zapojte podle níže uvedeného schématu.



## A JDE SE PROGRAMOVAT ...

- ② Napište a nahrajte do desky Arduino následující programový kód.

### RYCHLÝ TIP

- ➔ Použijte některý z předchozích příkladů. Tzn. otevřete si jej v prostředí IDE a následně uložte pod novým jménem. Programový kód stačí pouze jednoduše inovovat.



```

1 const int row[8] = {2, 7, 19, 5, 13, 18, 12, 16};
2
3 const int col[8] = {
4     6, 11, 10, 3, 17, 4, 8, 9
5 };
6
7 byte image[8][8] = {
8     {0,0,0,0,0,0,0,0},
9     {0,1,1,0,0,1,1,0},
10    {1,0,0,1,1,0,0,1},
11    {1,0,0,0,0,0,0,1},
12    {1,0,0,0,0,0,0,1},
13    {0,1,0,0,0,0,1,0},
14    {0,0,1,0,0,1,0,0},
15    {0,0,0,1,1,0,0,0}};
16
17 void setup(){
18     for(int i = 0; i < 8; i++){
19         pinMode(col[i], OUTPUT);
20         pinMode(row[i], OUTPUT);
21         digitalWrite(col[i], HIGH);
22         digitalWrite(row[i], LOW);
23     }
24 }
25
26 void loop(){
27     refreshScreen();
28 }
29
30 void refreshScreen(){
31     for(int j = 0; j<8;j++){
32         digitalWrite(col[j], LOW);
33         for(int k = 0; k<8; k++){
34             digitalWrite(row[k], image[k][j]);
35         }
36         Clear();
37     }
38 }
39
40 void Clear(){
41     for(int i = 0; i<8; i++){
42         digitalWrite(row[i],LOW);
43         digitalWrite(col[i],HIGH);
44     }
45 }
46

```

- ③ Pokud se vám podařilo nahrát do desky Arduino programový kód, zkuste si odpovědět na následující otázky.



#### OTÁZKY PRO VÁS

- V čem se liší programový kód pro zobrazení symbolu od kódu z předchozích kapitol?
- Jak si myslíte že vznikl tvar srdce na displeji. Kde je nadefinován?

## VÝBORNĚ A JDE SE NA ÚKOLY

- ④ Pokud již chápete, jak se zobrazuje symbol srdce na displeji, vyřešíte následující úkol velmi rychle.



Definice tvaru symbolů je velmi snadné. Můžete využít nástroj, pomocí něhož si symbol „naklikáte“ a následně použijete vygenerované dvourozměrné pole vypnutých/zapnutých diod, které vložíte do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>



#### ÚKOL PRO VÁS

- ➔ A) Upravte programový kód tak, aby se na displeji zobrazil symbol smajlíku.

## A JEŠTĚ JEDEN ÚKOL



### ÚKOL PRO VÁS

- B) Změňte programový kód tak, aby se střídavě zobrazoval symbol velkého a malého srdce.

## ŘEŠENÍ ÚLOH

Úkol A)

```
1 // Srdce
2 byte image[8][8] = {
3     {0,0,0,0,0,0,0,0},
4     {0,1,1,0,0,1,1,0},
5     {1,0,0,1,1,0,0,1},
6     {1,0,0,0,0,0,0,1},
7     {1,0,0,0,0,0,0,1},
8     {0,1,0,0,0,0,1,0},
9     {0,0,1,0,0,1,0,0},
10    {0,0,0,1,1,0,0,0}};
11
12 // Smajlík
13 byte image[8][8] = {
14     {0,0,1,1,1,1,0,0},
15     {0,1,0,0,0,0,1,0},
16     {1,0,1,0,0,1,0,1},
17     {1,0,0,0,0,0,0,1},
18     {1,0,1,0,0,1,0,1},
19     {1,0,0,1,1,0,0,1},
20     {0,1,0,0,0,0,1,0},
21     {0,0,1,1,1,1,0,0}};
```

Úkol B)

```
1 const int row[8] = {
2     2, 7, 19, 5, 13, 18, 12, 16
3 };
4
5 const int col[8] = {
6     6, 11, 10, 3, 17, 4, 8, 9
7 };
8
9 // Velke srdce
10 byte image[8][8] = {
11     {0,0,0,0,0,0,0,0},
12     {0,1,1,0,0,1,1,0},
13     {1,0,0,1,1,0,0,1},
14     {1,0,0,0,0,0,0,1},
15     {1,0,0,0,0,0,0,1},
16     {0,1,0,0,0,0,1,0},
17     {0,0,1,0,0,1,0,0},
18     {0,0,0,1,1,0,0,0}};
19
20 // Male srdce
21 byte imageS[8][8] = {
22     {0,0,0,0,0,0,0,0},
23     {0,0,0,0,0,0,0,0},
24     {0,0,0,1,0,1,0,0},
25     {0,0,1,0,1,0,1,0},
26     {0,0,1,0,0,0,1,0},
27     {0,0,0,1,0,1,0,0},
28     {0,0,0,0,1,0,0,0},
29     {0,0,0,0,0,0,0,0}};
30
31 void setup(){
32     for(int i = 0; i < 8; i++){
33         pinMode(col[i], OUTPUT);
34         pinMode(row[i], OUTPUT);
35         digitalWrite(col[i], HIGH);
36         digitalWrite(row[i], LOW);
37     }
38 }
39
40 void loop(){
41     // Zobrazeni vždy po dobu 100 iteraci
42     for(int i = 0; i < 100; i++){
```

```
43     refreshScreen(image);
44 }
45
46 for(int i = 0; i < 100; i++){
47     refreshScreen(imageS);
48 }
49
50 }
51
52 void refreshScreen(unsigned char dat[8][8]){
53     for(int j = 0; j<8;j++){
54         digitalWrite(col[j], LOW);
55         for(int k = 0; k<8; k++){
56             digitalWrite(row[k], dat[k][j]);
57         }
58         delay(1);
59         Clear();
60     }
61 }
62
63 void Clear(){
64     for(int i = 0; i<8; i++){
65         digitalWrite(row[i],LOW);
66         digitalWrite(col[i],HIGH);
67     }
68 }
69 }
```

# PRŮVODCE HODINOU IV



Studenti opět budou pracovat s kompletně zapojeným maticovým displejem. Tentokrát se do obvodu přidají dva potenciometry. Tím získáme dva zdroje pro analogové vstupy, které využijme pro ovládání diod na maticovém displeji.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, maticový LED displej 8x8, vodiče, potenciometr (2x).
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 7.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⏳ 15 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní hodiny bude si ukázat, jak spojit displej s potenciometry tak, aby se dali jeho diody ovládat. Vytvoří se ovladač displeje.

### RYCHLÝ TIP

- ➔ Pokud mají studenti zapojený obvod s displejem z předchozí hodiny, tak je dobré jej využít. Nové schéma spočívá pouze v přidání dvou potenciometrů.
- ➔ V opačném případě studenti musí zapojit obvod celý.



### OTÁZKA PRO STUDENTY

- Zapojování potenciometrů byste již měli znát. Jak tedy zapojit potenciometry do obvodu s maticovým displejem?

Řešení je zcela standardní a studenti se pro nápovědu mohou podívat na řešení problému do kapitoly týkající se servomotorů.



### ÚKOL PRO STUDENTY

- A) Zapojte oba potenciometry do obvodu s maticovým displejem.  
Použijte analogové vstupy na desce Arduino A0 a 11.

## 2. KROK 15 minut

Ať si studenti otevřou programový kód naposledy realizované úlohy, týkající se zobrazování symbolů.

### OTÁZKY PRO STUDENTY

- Jak byste upravili kód, aby docházelo pomocí potenciometrů k posunu svíticí diody na displeji? Jak se čtou data z potenciometru a jakých nabývají hodnot?

Otázky by je měli přivést k řešení. Tzn. měli by již vědět, že použijí funkci `analogRead()`. Hodnoty získané z potenciometru jsou 0-1024. Toto rozmezí hodnot se musí rozložit do 8-mi diod v ose x a y.

- Jak se tyto hodnoty rozloží do 8-mi diod na displeji?

Použije se funkce `map()`.

- Když už víte, jak se čtou hodnoty z potenciometru a jak se dají rozložit do hodnot pro displej, jak byste řešili rozsvícení diody v závislosti na otočení potenciometru?

Nejfektivnější je vytvořit funkci určenou pro čtení hodnot z potenciometru.



### 3. KROK 🕒 10 minut

Ukažte studentům celý programový kód ať jej porovnají s kódem vlastním. Případně ať jej upraví a otestují.

Ve zbytku hodiny, můžete studentům ukázat ještě další aplikaci na spojení potenciometrů a maticového displeje. Je to hra pro dva PING-PONG.

Pro zručnější studenty by mohlo být naprogramování této hry i samostatným úkolem.

Programový kód hry je dostupný na GitHub.



# PRACOVNÍ LIST – MATICOVÝ DISPLEJ - IV

PŮVODNÍ ZAPOJENÍ MATICOVÉHO LED DISPLEJE ROZŠÍŘÍTE O DVA POTENCIOMETRY. TĚMITO POTENCIOMETRY BUDETE OVLÁDAT DIODY displeje NA KONKRÉTNÍCH POZICÍCH.

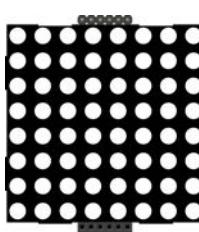
## CO SE NAUČÍTE

- ① Zopakujete si zapojení potenciometru.
- ② Spojení potenciometru a maticového displeje.
- ③ Zpracovávat hodnoty z potenciometru pro displej.

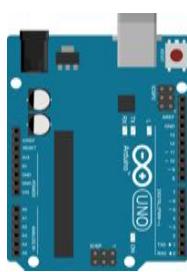


## CO BUDETE POTŘEBOVAT

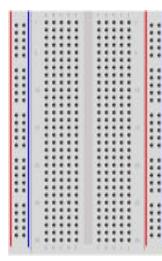
- ① Maticový displej.
- ② Desku Arduino.
- ③ Potenciometr – 2x
- ④ Kontaktní pole.
- ⑤ Vodiče typu zásuvka-zásuvka.



Maticový displej 8x8



Deska Arduino



Kontaktní pole

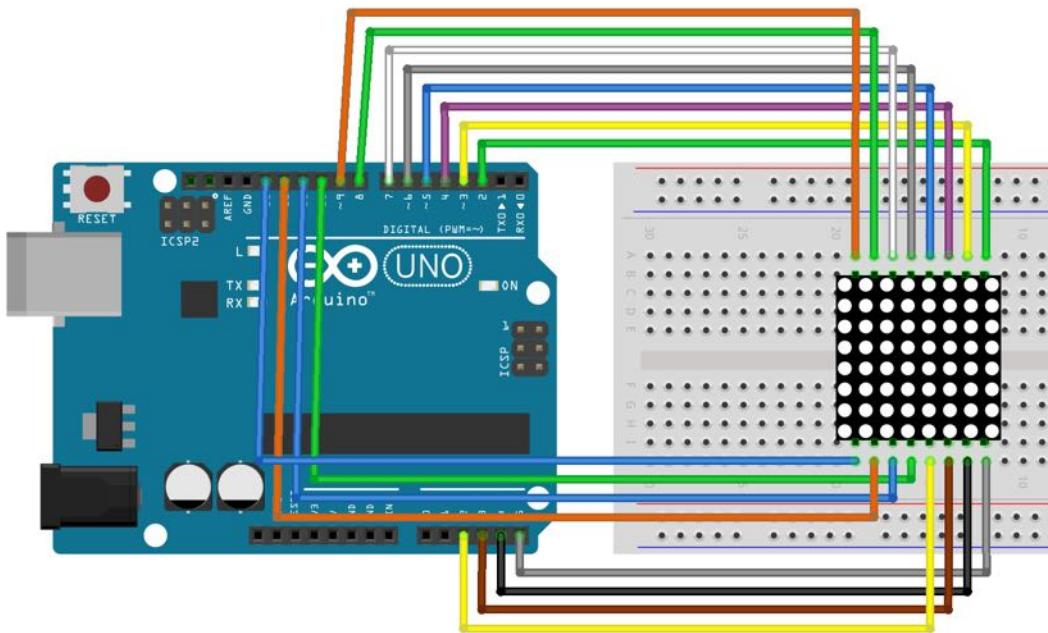


Potenciometr 2x

POUŽITÉ SOUČÁSTKY

## RYCHLÝ ÚVOD ...

- ① Pokud nemáte sestavený obvod s maticovým displejem, tak v rámci opakování jej zapojte podle níže uvedeného schématu.



### OTÁZKA PRO VÁS

→ Zapojování potenciometrů byste již měli znát. Jak tedy zapojit potenciometry do obvodu s maticovým displeje?



### ÚKOL PRO VÁS

→ A) Zapojte oba potenciometry do obvodu s maticovým displejem.  
Použijte analogové vstupy na desce Arduino A0 a 11.

- ② Po zapojení potenciometrů, přistupte k programovému kódu. Otevřete si některý z předchozích příkladů, týkajících zobrazování symbolů. Tento příklad uložte pod novým jménem.

### OTÁZKY PRO VÁS

- Jak byste upravili kód, aby docházelo pomocí potenciometrů k posunu svíticí diody na displeji? Jak se čtou data z potenciometru a jakých nabývají hodnot?
- Jak se tyto hodnoty rozloží do 8-mi diod na displeji?
- Když už víte, jak se čtou hodnoty z potenciometru a jak se dají rozložit do hodnot pro displej, jak byste řešili rozsvícení diody v závislosti na otočení potenciometru?



### A JDE SE PROGRAMOVAT ...

- ③ Zkuste porovnat vaše odpovědi z předchozích otázek, týkajících se programového kódu s přiloženým programem.

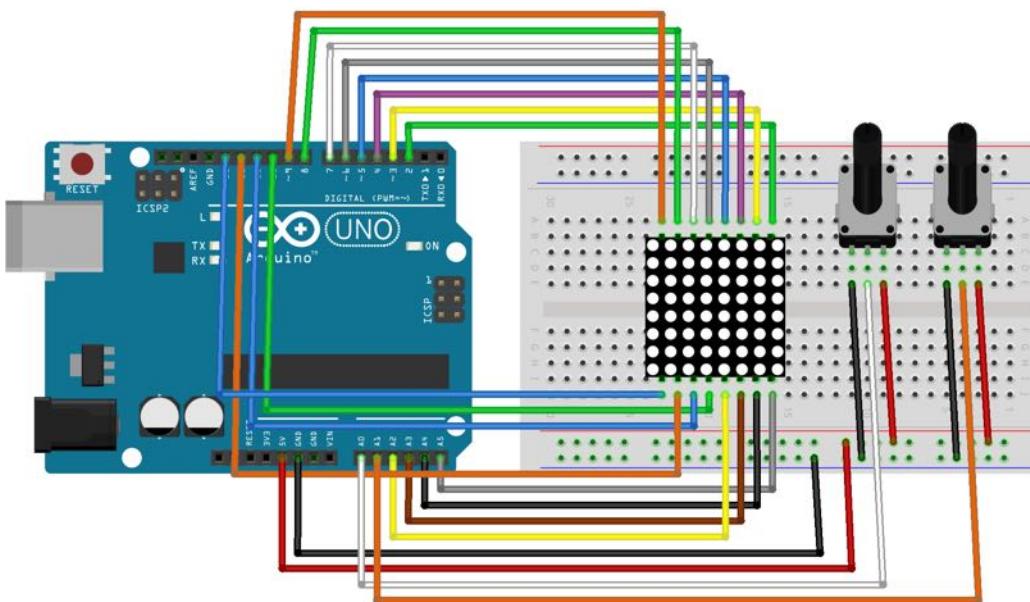
```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {  
6     6, 11, 10, 3, 17, 4, 8, 9  
7 };  
8  
9 int pixels[8][8];  
10  
11 int x = 5;  
12 int y = 5;  
13  
14 void setup(){  
15     for(int i = 0; i < 8; i++){  
16         pinMode(col[i], OUTPUT);  
17         pinMode(row[i], OUTPUT);  
18         digitalWrite(row[i], LOW);  
19     }  
20 }
```

```

19 }
20
21     for(int x = 0; x < 8; x++) {
22         for(int y = 0; y < 8; y++) {
23             pixels[x][y] = HIGH;
24         }
25     }
26 }
27
28 void loop(){
29     readSensors();
30     refreshScreen();
31 }
32
33 void readSensors(){
34     pixels[x][y] = HIGH;
35     x = 7 - map(analogRead(A0), 0, 1023, 0, 7);
36     y = map(analogRead(A1), 0, 1023, 0, 7);
37     pixels[x][y] = LOW;
38 }
39
40
41 void refreshScreen(){
42     for(int j = 0; j<8;j++){
43         digitalWrite(row[j], HIGH);
44         for(int k = 0; k<8; k++){
45             int thisPixel = pixels[j][k];
46             digitalWrite(col[k], thisPixel);
47             if (thisPixel == LOW) {
48                 digitalWrite(col[k], HIGH);
49             }
50         }
51         digitalWrite(row[j], LOW);
52     }
53 }
```

## ŘEŠENÍ ÚLOH

Úkol A)



# PRŮVODCE HODINOU V



Studenti opět budou pracovat s kompletně zapojeným maticovým displejem. Tentokrát se do obvodu přidá akcelerometr. Ten bude poskytovat vstupní hodnoty pro maticový displej.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, maticový LED displej 8x8, vodiče, akcelerometr.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 7.
- ⑤ Pracovní listy pro studenty.

## 1. KROK 🕒 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že náplní hodiny bude si ukázat, jak spojit maticový displej s akcelerometrem.

### OTÁZKY PRO STUDENTY

➔ **Víte kde se můžete setkat se zařízením akcelerometr?**

V dnešní době má akcelerometr takřka každý mobilní telefon. Dále jej nalezneme v automobilech, letadlech apod.

➔ **Víte, co akcelerometr měří?**

Měří pohybové zrychlení, a to nejlépe ve všech třech osách.



## 2. KROK 🕒 5 minut

Vysvětlete podrobněji princip akcelerometru.

## 3. KROK 🕒 10 minut

Ať studenti zapojí akcelerometr podle přiloženého nebo promítaného schématu.



### RYCHLÝ TIP

- ➔ Pokud mají studenti zapojený obvod s displejem z předchozí hodiny, tak je dobré jej využít. Nové schéma spočívá pouze v přidání akcelerometru.
- ➔ V opačném případě studenti musí zapojit obvod celý.



### NA CO SI DÁT POZOR

- ➔ Zaměřte se zejména na správné zapojení napájení akcelerometru a datových pinů SDA a SCL.

## 4. KROK 🕒 15 minut

Studenti by měli přistoupit k programování. Opět mohou použít předchozí programový kód vztahující se k potenciometrům. Ať si studenti otevřou předchozí program a uloží jej pod novým názvem.



Pro co možná nejjednodušší programování akcelerometru ADXL 345 je vhodné použít některou z knihoven. Proto ji studenti musí na začátku programového kódu připojit. Použitá knihovna pro ADXL 345 je k dispozici na GitHub.

Ukažte studentům výpočet úhlů **roll** a **pitch**.

```
1 roll = (atan2(-Yg, Zg)*180.0)/M_PI;  
2 pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg)))*180.0/M_PI;
```

#### ÚKOL PRO STUDENTY



- A) Inovujte programový kód otevřeného programu tak, abyste aplikovali uvedený vzorec pro výpočet úhlů **roll** a **pitch**.

Tento úkol je velmi jednoduchý. Stačí upravit funkci `readSensors()`. Pro správné namapování hodnot z akcelerometru by měli studenti ověřit, jaké hodnoty poskytuje. K tomu mohou využít sériový monitor. Následně podle získaných maxim upraví funkci `map()`.

# PRACOVNÍ LIST – MATICOVÝ DISPLEJ - V

PŮVODNÍ ZAPOJENÍ MATICOVÉHO LED DISPLEJE ROZŠÍŘÍTE O PŘIPOJENÍ AKCELEROMETRU. V ZÁVISLOSTI NA POLOZE BUDE POSKYTOVAT DATA PRO POZICI ROZSVÍCENÉ DIODY NA MATICOVÉM displeji.

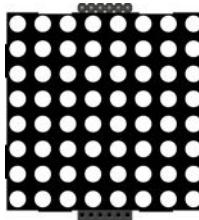
## CO SE NAUČÍTE

- ① Zapojovat akcelerometr.
- ② Spojení akcelerometru a maticového displeje.
- ③ Zpracovávat hodnoty z akcelerometru pro displej.

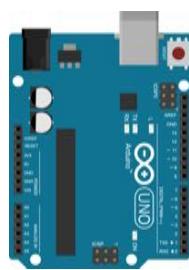


## CO BUDETE POTŘEBOVAT

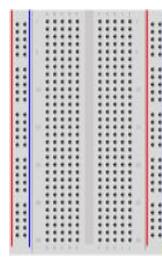
- ① Maticový displej.
- ② Desku Arduino.
- ③ Akcelerometr.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zásuvka-zásuvka.



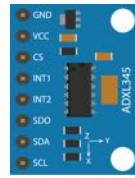
Maticový displej 8x8



Deska Arduino



Kontaktní pole

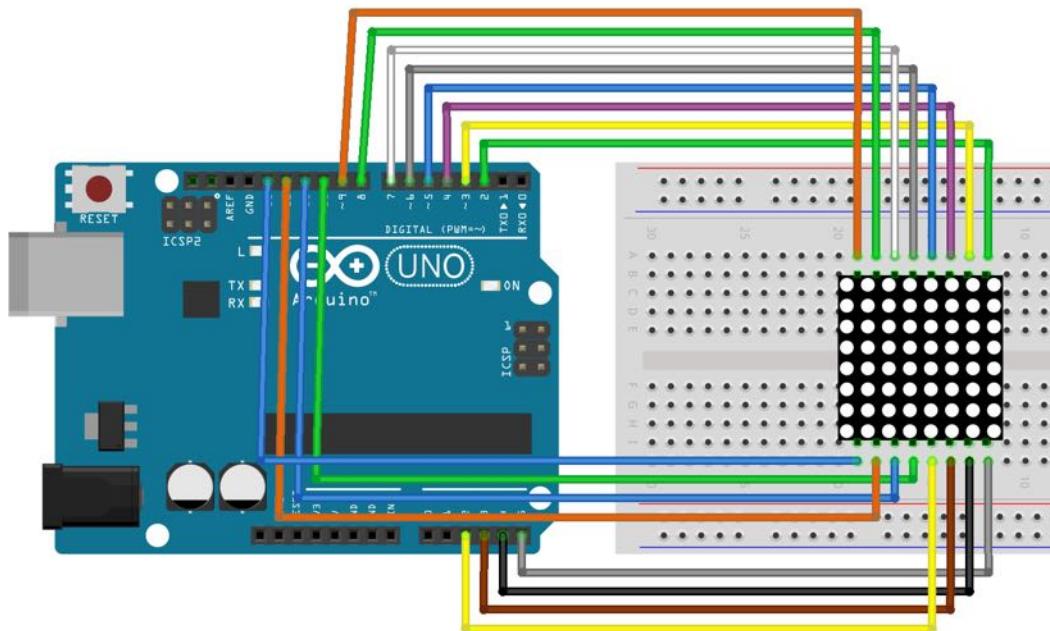


Akcelerometr

POUŽITÉ SOUČÁSTKY

## RYCHLÝ ÚVOD ...

- ① Pokud nemáte sestavený obvod s maticovým displejem, tak v rámci opakování jej zapojte podle níže uvedeného schématu.



### OTÁZKY PRO VÁS

- Víte kde se můžete setkat se zařízením akcelerometr?
- Víte, co akcelerometr měří?



## KRÁTCE O AKCELEROMETRU

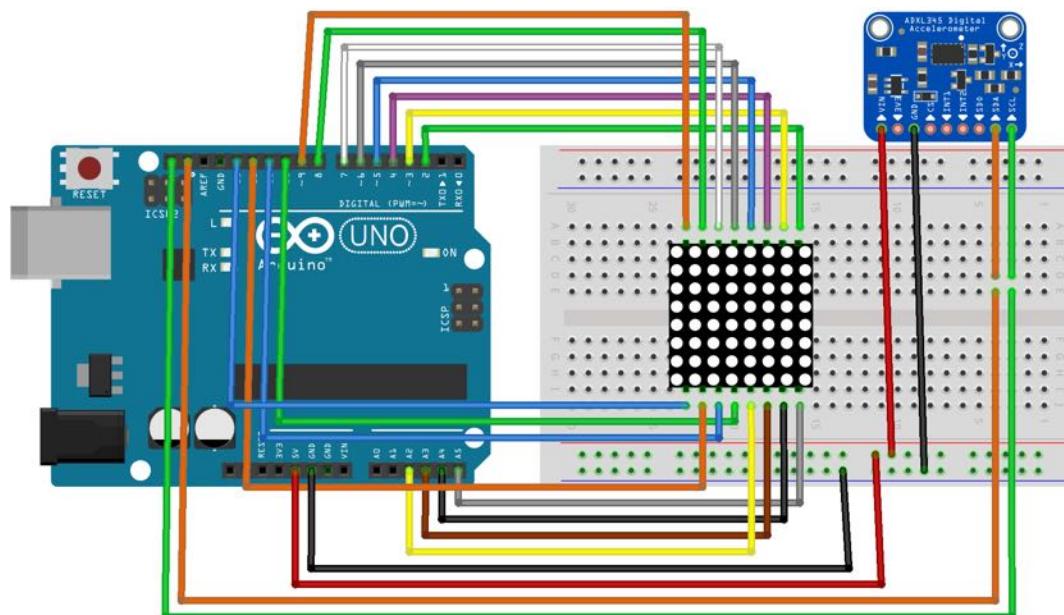
Akcelerometr je malé pohybové čidlo, které měří pohybové zrychlení a to nejlépe ve všech třech osách. Ze znalosti zrychlení a hmotnosti lze zjistit sílu působící na těleso.

Akcelerometry jsou vhodné nejen pro měření odstředivých a setrvačních sil, ale i pro určování pozice těla, jeho náklon nebo vibrace. Akcelerometry jsou dnes i v mobilních telefonech a využívají se v leteckém a automobilovém průmyslu.

Aby bylo možné definovat úhly akcelerometru ve třech rozměrech **pitch**, **roll** a **theta**, využívají se všechny tři výstupy akcelerometru. **Pitch** (ró), je definováno jako úhel vzhledem k ose X a země. **Roll** (fí) je definováno jako úhel vzhledem k ose Y a země. **Theta** je úhel vzhledem k ose Z - gravitace.

## JAK ZAPOJIT AKCELEROMETR

- ② Pokud máte zapojený maticový displej, připojte podle přiloženého schématu akcelerometr.



### DEJTE SI POZOR

- Zaměřte se zejména na správné zapojení napájení akcelerometru a datových pinů **SDA** a **SCL**.



## PROGRAMOVÁNÍ

- ③ Otevřete si předchozí programový kód a uložte jej jako nový soubor. Tím si ušetříte práci a čas.



Pro co možná nejjednodušší programování akcelerometru ADXL 345 je vhodné použít některou z knihoven. Proto si ji na začátku programového kódu připojte. Použitá knihovna pro ADXL 345 je k dispozici na GitHub ke stažení.

- ④ Připojení knihoven pro práci s akcelerometrem je následující:

```
1 #include <Wire.h>
2 #include <ADXL345.h>
3
4 acc.read(&Xg, &Yg, &Zg);
```

- ⑤ Pro výpočet úhlů **roll** a **pitch** využijte následující programový zápis.

```
1 roll = (atan2(-Yg, Zg)*180.0)/M_PI;
2 pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg)))*180.0/M_PI;
```



### ÚKOL PRO VÁS

- A) Inovujte programový kód otevřeného programu tak, abyste aplikovali uvedený vzorec pro výpočet úhlů **roll** a **pitch**. Nezapomeňte definovat všechny proměnné.

- ⑥ Hotový program nahrajte do desky Arduino. Pokud je vše v pořádku, tak na pohyb akcelerometru je znázorněn na maticovém displeji pohybujícím se světlem.

# ŘEŠENÍ ÚLOH

Úkol A)

```
1 #include <Wire.h>
2 #include <ADXL345.h>
3
4 ADXL345 acc;
5
6 const int row[8] = {2, 7, 19, 5, 13, 18, 12, 16};
7 const int col[8] = {6, 11, 10, 3, 17, 4, 8, 9};
8
9 int pixels[8][8];
10
11 int x = 5;
12 int y = 5;
13
14 void setup(){
15     acc.begin();
16
17     for(int i = 0; i < 8; i++){
18         pinMode(col[i], OUTPUT);
19         pinMode(row[i], OUTPUT);
20         digitalWrite(row[i], LOW);
21     }
22
23
24     for(int x = 0; x < 8; x++) {
25         for(int y = 0; y < 8; y++) {
26             pixels[x][y] = HIGH;
27         }
28     }
29 }
30
31 void loop(){
32     readSensors();
33     refreshScreen();
34 }
35
36 void readSensors(){
37     double pitch, roll, Xg, Yg, Zg;
38     acc.read(&Xg, &Yg, &Zg);
39
40     roll = (atan2(-Yg, Zg)*180.0)/M_PI;
41     pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg)))*180.0/M_PI;
```

```
42
43     pixels[x][y] = HIGH;
44     x = 7 - map(roll, -20, 20, 0, 7);
45     y = map(pitch, -20, 20, 0, 7);
46     pixels[x][y] = LOW;
47 }
48
49 void refreshScreen(){
50     for(int j = 0; j<8;j++){
51         digitalWrite(row[j], HIGH);
52         for(int k = 0; k<8; k++){
53             int thisPixel = pixels[j][k];
54             digitalWrite(col[k], thisPixel);
55             if (thisPixel == LOW) {
56                 digitalWrite(col[k], HIGH);
57             }
58         }
59         digitalWrite(row[j], LOW);
60     }
61 }
66
```

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU, KTERÝ JE ZAMĚŘEN NA POUŽÍVÁNÍ MATICOVÉHO LED displeje a akcelerometru. ŘEŠENÉ ÚKOLY ZOHLEDŇUJÍ NOVĚ PROBRANÉ TÉMA A STAVÍ NA PŘEDCHOZÍCH ZNALOSTECH.

## OBSAH PRŮVODCE

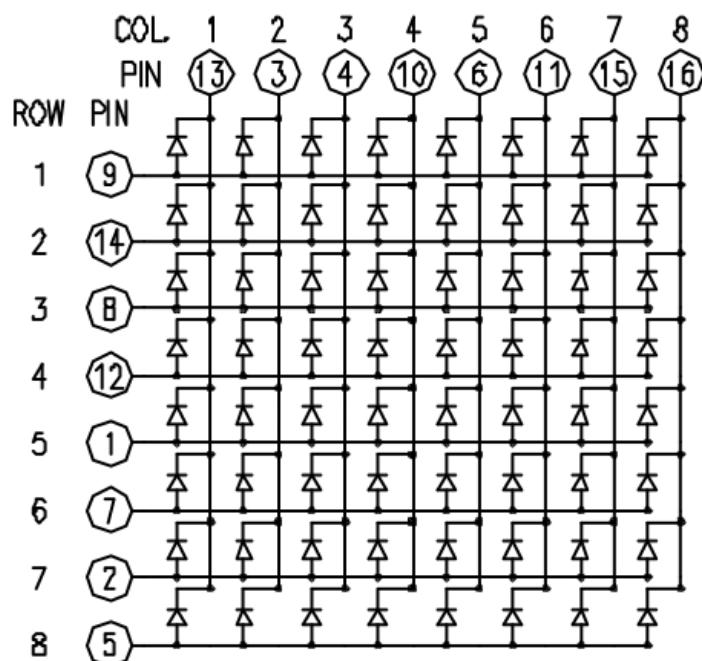
- ① Získání dovedností při zapojování maticového LED displeje.
- ② Naučení se zobrazovat symbolů na maticovém LED displeji.
- ③ Využití analogových hodnot pro ovládání diod maticového displeje.
- ④ Získání dovedností při zapojování akcelerometru.
- ⑤ Programování ovládání akcelerometru.
- ⑥ Získání dovedností při propojení maticového LED displeje a akcelerometru.

## ZOBRAZOVÁNÍ SYMBOLŮ NA MATICOVÉM LED DISPLEJI

### MATICOVÝ LED DISPLAY

Maticové LED displeje, i když mají jen malé rozlišení, se používají hlavě proto, že lze s nimi dosáhnout efektního vzhledu a zobrazení jednoduché grafiky, jsou relativně velké a čitelné z větší vzdálenosti, v šeru aktivním svitem poutají pozornost.

LED displeje jsou nejčastěji reprezentovány jako matice LED diod, uspořádaných v řadách a sloupcích. Řady představují běžné anody a sloupce společné katody nebo naopak.



Obr. 1 - Schéma zapojení diod maticového displeje

Chceme-li ovládat matici, musí se propojit její řady i sloupce s mikrokontrolérem. Sloupce jsou připojeny ke katodám LED (viz Obr. 1), takže pokud mají být zapnuty LED diody v konkrétním sloupci, musí být hodnota pro každou z diod nastavena na **LOW**. Řádky jsou připojeny k anodám LED diod, takže pro jejich zapnutí musí být nastavena hodnota na **HIGH**. Pokud jsou hodnoty pro řádky i sloupce nastaveny stejně na **LOW** nebo **HIGH**, dioda se na displeji nerozsvítí.

Chcete-li ovládat jednotlivé LED diody displeje, musí se nastavit ve jejím sloupci hodnota **LOW** a řádek na **HIGH**. Má-li být ovládáno několik LED diod za sebou, musí se nastavit

řádek na **HIGH**, poté konkrétní sloupec na **LOW** nebo **HIGH**; sloupec **LOW** zapne odpovídající LED a ve sloupci **HIGH** jej vypne.

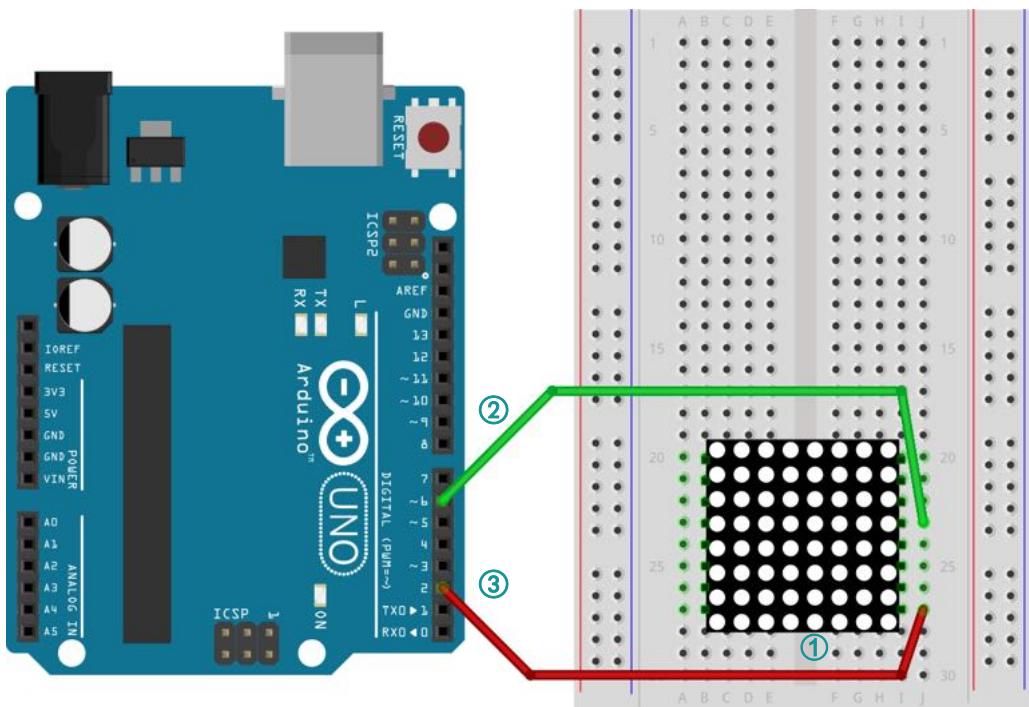


Pokud chceme rozsvítit LED na pozici [1,1], musíme připojit pin displeje 13 na + a 9 na GND. Pokud bychom ale chtěli rozsvítit zároveň bod [1,1] a [2,2], je situace trošku komplikovanější. Kdybychom totiž připojili 13 a 3 na + a 14 i 9 na GND současně, rozsvítíl by se celý čtverec ([1,1], [2,1], [1,2], [2,2]). Z tohoto důvodu se vždy pracuje jen s jednou řadou (atž už jde o řádek, nebo o sloupec), rozsvítí se všechny body, které se mají zobrazit, poté se napájení řady vypne, a to samé se opakuje se všemi dalšími řadami. Pokud toto „překreslování“ probíhá dostatečně rychle, lidské oko si ničeho nevšimne (kvůli jeho setrvačnosti). Anody jsou vypnuté, pokud je na jejich pinu stav **LOW**, u katod je tomu naopak – vypnuté jsou při stavu **HIGH**. Kombinace pinů a zapojení displeje je v následující tabulce.

Matrice pin	Řádek	Sloupec	Arduino pin
1	5	-	13
2	7	-	12
3	-	2	11
4	-	3	10
5	8	-	A2
6	-	5	A3
7	6	-	A4
8	3	-	A5
9	1	-	2
10	-	4	3
11	-	6	4
12	4	-	5
13	-	1	6
14	2	-	7
15	-	7	8
16	-	8	9

Tab. 2 - Rozložení pinů

## ZAPOJENÍ DISPLEJE PRO JEHO OTESTOVÁNÍ



Obr. 2 – Základní zapojení LED maticy

- ① Maticový displej je umístěn v kontaktním poli, tím se bude lépe propojovat s deskou Arduino.
- ② Zelený vodič připojte k pinu displeje číslo 13 a druhý konec vodiče k desce Arduino na pin 6.
- ③ Červený vodič připojte k pinu displeje číslo 9 a druhý konec vodiče k desce Arduino na pin 2.



Princip zapojení maticového displeje je patrný z výše uvedené tabulky kombinace pinů.

## PROGRAMOVÝ KÓD

Programový kód je velmi jednoduchý a je podobný jako při zapojení obyčejné LED diody.

```
1 int pinA=2;
2 int pinB=6;
3
4 void setup() {
5     pinMode(pinA,OUTPUT);
6     pinMode(pinB,OUTPUT);
7     digitalWrite(pinA,HIGH);
8     digitalWrite(pinB,HIGH);
9 }
10
11 void loop() {
12     digitalWrite(pinB,LOW);
13     delay(200);
14     digitalWrite(pinB,HIGH);
15     delay(200);
16 }
```



- ① Deklarace proměnné **pinA** definuje číslo pinu na desce Arduino. V maticovém displeji je připojen na katodu diody.
- ② Deklarace proměnné **pinB** definuje číslo pinu na desce Arduino. V maticovém displeji je připojen na anodu diody.
- ③ Vyhrazení pinu pro katodu.
- ④ Vyhrazení pinu pro anodu.
- ⑤ Nastavení hodnoty na **HIGH**.
- ⑥ Nastavení hodnoty na **HIGH**. Pokud je na anodě a katodě stejná hodnota, tak dioda nesvítí.
- ⑦ Na anodu je přivedena hodnota **LOW**, tím dojde k rozsvícení diody.
- ⑧ Dioda svítí 200ms.
- ⑨ Na diodu je přivedena hodnota **HIGH**, tím dioda opět zhasne.
- ⑩ Dioda zhasne na 200ms.



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, měla by blikat první dioda, tj. v levém horním rohu.



#### NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu. **Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

#### DIODA NA displeji NEBLIKÁ

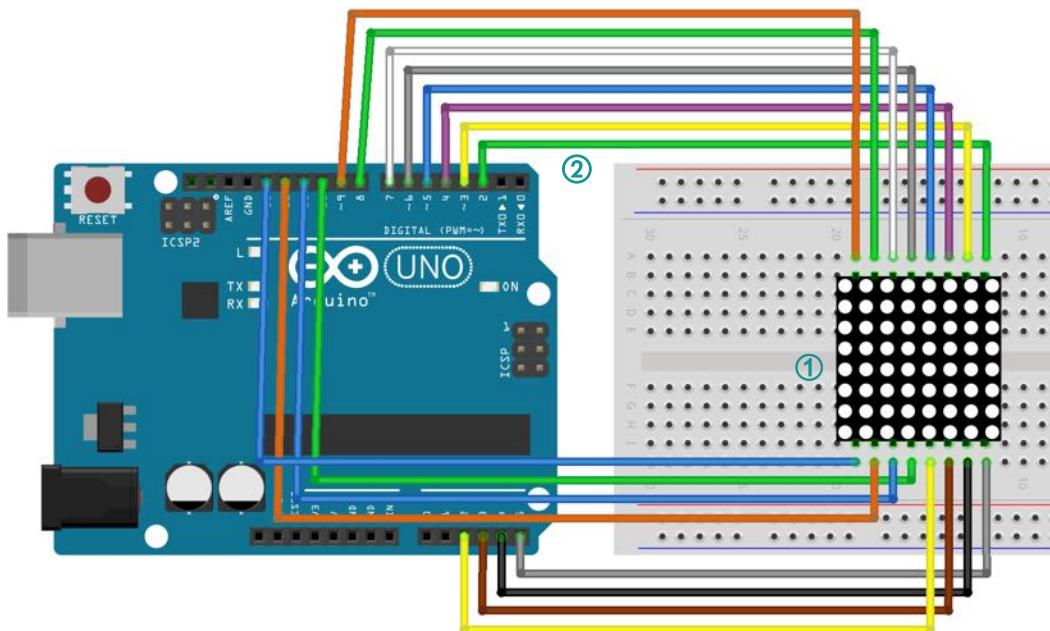
**Kontrola zapojení** – ujistěte se, že jsou vodiče opravdu zapojeny do správných pinů na kontaktním poli a desce Arduino.



(Př. 1) Upravte obvod zapojení displeje a programový kód předchozího příkladu tak, aby blikaly i diody ve všech rozích stejně jako dioda první.

(Př. 2) Změňte programový kód předchozího příkladu tak, aby diody v protilehlých rozích blikaly střídavě.

## ÚPLNÉ ZAPOJENÍ DISPLEJE



Obr. 3 – Základní zapojení LED matic

- ① Maticový displej je umístěn v kontaktním poli, tím se bude lépe propojovat s deskou Arduino.
- ② Pro propojení displeje s deskou Arduino je potřeba 16 vodičů. Zapojení je velmi jednoduché. Při zapojení si lze vzít na pomoc rozložení pinů v tabulce Tab. 2 - Rozložení pinů.



Na uvedeném zapojení lze provést celou řadu příkladů. Proto by bylo dobré, aby tento obvod mohl být složen i do příštích hodin a vy jste se věnovali pouze programování.

## PROGRAMOVÝ KÓD – POSTUPNÉ ROZSVĚCOVÁNÍ DIOD displeje

Rozsvěcování jednotlivých diod na maticovém displeji, lze udělat několika různými způsoby. Resp. lze vytvořit mnoho různých světelných kombinací. Zde je uvedena varianta, kdy se v každém řádku postupně rozsvěcují diody.

```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {  
6     6, 11, 10, 3, 17, 4, 8, 9  
7 };  
8  
9 void setup(){  
10    for(int i = 0; i < 8; i++){  
11        pinMode(col[i], OUTPUT);  
12        pinMode(row[i], OUTPUT);  
13        digitalWrite(col[i], HIGH);  
14        digitalWrite(row[i], LOW);  
15    }  
16}  
17  
18 void loop(){  
19    for(int j = 0; j<8;j++) {  
20        digitalWrite(col[j],LOW);  
21        for(int k = 0;k<8;k++){  
22            digitalWrite(row[k],HIGH);  
23            delay(200);  
24        }  
25        for(int i = 0;i<8;i++){  
26            digitalWrite(row[i],LOW);  
27            digitalWrite(col[i],HIGH);  
28        }  
29    }  
30}
```

- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.

- ③ Využití cyklu **for** k nastavení pinů pro sloupce a řádky jako výstupních.
- ④ Vymazání (zhasnutí) sloupců displeje.
- ⑤ Vymazání (zhasnutí) řádků displeje.
- ⑥ V cyklu **for** se postupně nastavují jednotlivé sloupce na hodnotu **LOW**.
- ⑦ V následujícím cyklu **for** se pro každý řádek v aktuálním sloupci nastavuje hodnota **HIGH**, čímž dojde k rozsvícení diody.
- ⑧ Aby bylo vidět postupné rozsvěcování diod je nastavena pauza na 200ms.
- ⑨ Ostatní sloupce a řádky se vypnou (zhasnou).

(Př. 3) Upravte (optimalizujte) programový kód tak, aby se aktualizace a mazání displeje prováděla ve dvou vámi deklarovaných funkcích.

(Př. 4) Upravte programový kód tak, aby se v celém, rozsvíceném displeji postupně posouval vypnutý sloupec a při tomto vypnutém sloupci projížděl vypnutý řádek.



## PROGRAMOVÝ KÓD – ZOBRAZOVÁNÍ SYMBOLŮ

Maticový displej, při vhodné kombinaci rozsvícených diod, může zobrazovat jednoduché symboly.

```
1 const int row[8] = {           └─①
2   2, 7, 19, 5, 13, 18, 12, 16
3 };
4
5 const int col[8] = {           └─②
6   6, 11, 10, 3, 17, 4, 8, 9
7 };
8
9 byte image[8][8] = {          └─③
10  {0,0,0,0,0,0,0,0},
11  {0,1,1,0,0,1,1,0},
12  {1,0,0,1,1,0,0,1},
13  {1,0,0,0,0,0,0,1},
14  {1,0,0,0,0,0,0,1},
15  {0,1,0,0,0,0,1,0},
16  {0,0,1,0,0,1,0,0},
17  {0,0,0,1,1,0,0,0};
18
19 void setup(){
20   for(int i = 0; i < 8; i++){
21     pinMode(col[i], OUTPUT);
22     pinMode(row[i], OUTPUT);
23     digitalWrite(col[i], HIGH);
24     digitalWrite(row[i], LOW);
25   }
26 }
27
28 void loop(){
29   refreshScreen();           └─⑥
30 }
31
32 void refreshScreen(){
33   for(int j = 0; j<8;j++){
34     digitalWrite(col[j], LOW);
35     for(int k = 0; k<8; k++){
36       digitalWrite(row[k], image[k][j]); └─⑨
37     }
38     clear();                  └─⑩
39   }
40 }
```

```

41
42 void Clear(){
43     for(int i = 0; i<8; i++){
44         digitalWrite(row[i],LOW);
45         digitalWrite(col[i],HIGH);
46     }
47 }
48

```

11

- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.
- ③ Deklarace dvourozměrného pole **image[8][8]**, které obsahuje definici řádků a sloupců displeje. Rozsvícené diody jsou reprezentovány hodnotou 1 a zhasnuté hodnotou 0.
- ④ Využití cyklu **for** k nastavení pinů pro sloupce a řádky jako výstupních.
- ⑤ Vymazání (zhasnutí) sloupců a řádků displeje.
- ⑥ Volání funkce **refreshScreen**, která vykresluje symbol na displej.
- ⑦ V cyklu **for** se postupně nastavují jednotlivé sloupce na hodnotu **LOW**.
- ⑧ V následujícím cyklu **for** se pro každý řádek v aktuálním sloupci.
- ⑨ Nastavuje se hodnota z pole **image[k][j]**, kde hodnota **HIGH = 1** a **LOW = 0**. Tím dojde k rozsvícení/zhasnutí aktuální diody.
- ⑩ Volání funkce **Clear**, která zhasne diody mimo symbol.
- ⑪ Deklarace funkce **Clear**.



Definice tvaru symbolů je velmi snadné. Můžete využít nástroj, pomocí něhož si symbol „naklikáte“ a následně použijete vygenerované dvourozměrné pole vypnutých/zapnutých diod, které vložíte do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>



Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, zobrazí se na displeji symbol srdce.

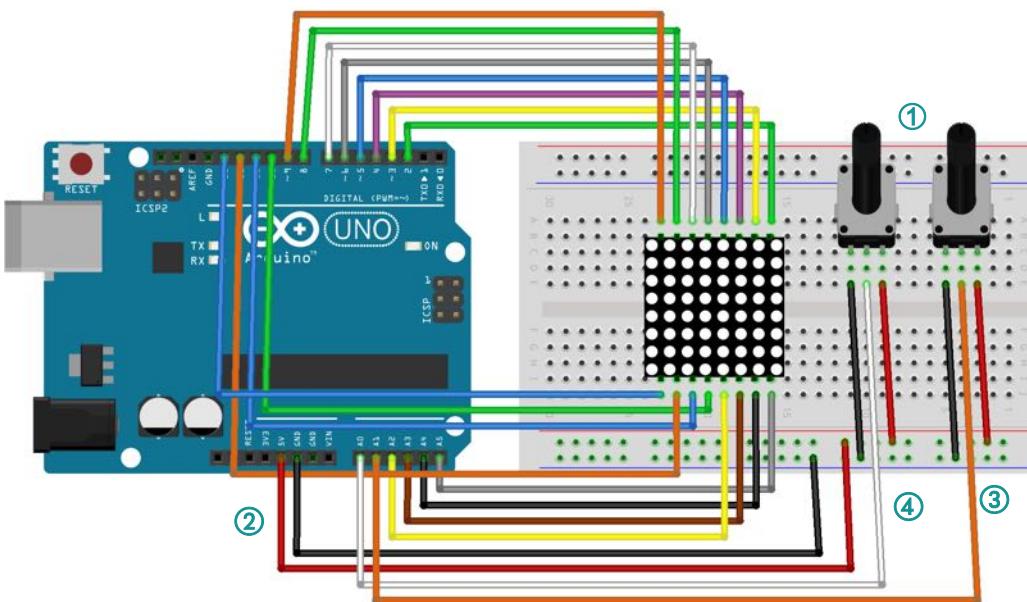


(Př. 5) Vyzkoušejte si zobrazit další symboly. Může to být smajlík, trojúhelník, dva kruhy v sobě atd.

(Př. 6) Změňte programový kód tak, aby se střídavě zobrazovalo velké a malé srdce.

## OVLÁDÁNÍ DIOD MATICOVÉHO displeje pomocí hodnot z analogového vstupu

Pro vstup se použijí dva potenciometry. Jeden bude ovládat pohyb rozsvícené diody v řádcích a druhý ve sloupcích.



Obr. 4 – Základní zapojení LED matic

- ① Potenciometry jsou připojeny přímo do kontaktní desky.
  - ② Na kontaktní desku je přivedeno napájení **5V** a zemnící vodič **GND** přímo z desky Arduino.
  - ③ Signál z prvního potenciometru je přiveden na analogový vstup desky Arduino – **A1**.
  - ④ Signál z druhého potenciometru je přiveden na analogový vstup desky Arduino – **A0**.



Zapojení maticového displeje je stejné, jako u předchozích příkladů, proto lze využít již zapojený obvod maticového displeje a přidat pouze potenciometry, což je již opakování. Studenti zapojení mohou provést v rámci samostatné úlohy.

## PROGRAMOVÝ KÓD – ANALOGOVÉ VSTUPY

Při programování ovládání maticového displeje pomocí analogových vstupů, lze opět vyjít z předchozích programů. Samotné doprogramování ovládání je pak velice jednoduché.

```
1 const int row[8] = {           └─①
2   2, 7, 19, 5, 13, 18, 12, 16
3 };
4
5 const int col[8] = {           └─②
6   6, 11, 10, 3, 17, 4, 8, 9
7 };
8
9 int pixels[8][8];            └─③
10
11 int x = 5;                  └─④
12 int y = 5;
13
14 void setup(){
15   for(int i = 0; i < 8; i++){
16     pinMode(col[i], OUTPUT);
17     pinMode(row[i], OUTPUT);
18     digitalWrite(row[i], LOW);
19   }
20
21   for(int x = 0; x < 8; x++) {
22     for(int y = 0; y < 8; y++) {
23       pixels[x][y] = HIGH;      └─⑤
24     }
25   }
26 }
27
28 void loop(){
29   readSensors();              └─⑥
30   refreshScreen();
31 }
32
33 void readSensors(){          └─⑦
34   pixels[x][y] = HIGH;
35   x = 7 - map(analogRead(A0), 0, 1023, 0, 7); └─⑧
36   y = map(analogRead(A1), 0, 1023, 0, 7);       └─⑨
37   pixels[x][y] = LOW;         └─⑩
38 }
```

```

41 void refreshScreen(){
42     for(int j = 0; j<8;j++){
43         digitalWrite(row[j], HIGH);
44         for(int k = 0; k<8; k++){
45             int thisPixel = pixels[j][k];
46             digitalWrite(col[k], thisPixel);
47             if (thisPixel == LOW) {
48                 digitalWrite(col[k], HIGH);
49             }
50         }
51         digitalWrite(row[j], LOW);
52     }
53 }
```

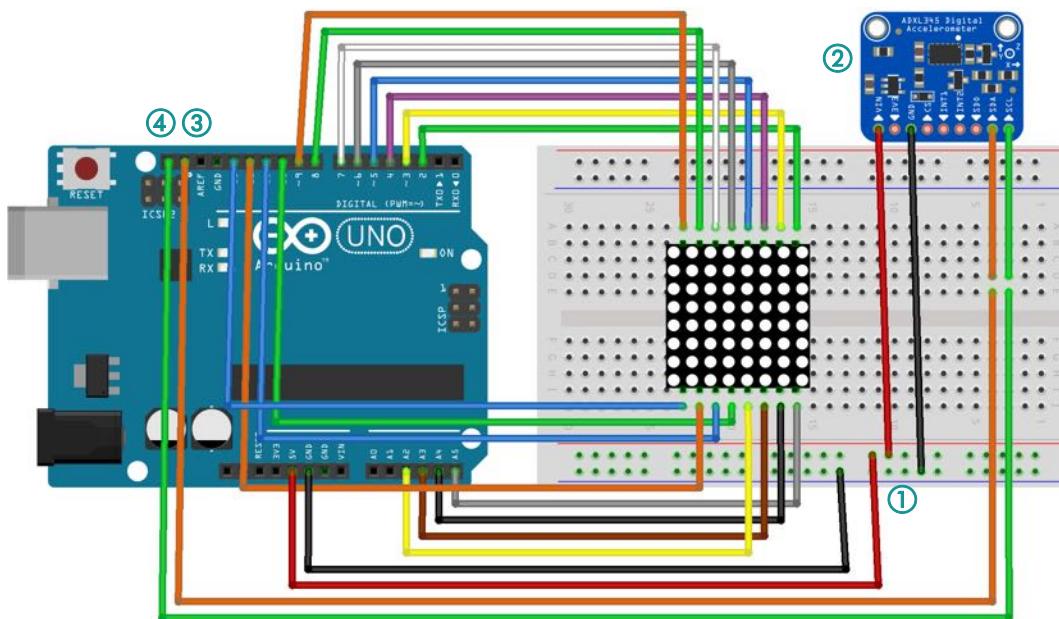
- ① Deklarace pole **row[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro řádky displeje.
- ② Deklarace pole **col[8]**, ve kterém jsou definovány čísla pinů na desce Arduino pro sloupce displeje.
- ③ Deklarace dvourozměrného pole **pixels[8][8]**, které obsahuje pozici rozsvícené diody.
- ④ Výchozí pozice rozsvícené diody při spuštění programu. Následuje již známá inicializace pinů pro výstup a vypnutí všech diod.
- ⑤ Inicializace matice **pixels**.
- ⑥ Volání funkce **readSensors()**, která čte hodnoty z potenciometrů.
- ⑦ Deklarace funkce **readSensors()**.
- ⑧ Namapování hodnoty z potenciometru pro souřadnici **x**.
- ⑨ Namapování hodnoty z potenciometru pro souřadnici **y**.
- ⑩ Nastavení nové pozice bodu tak, aby se LED dioda rozsvítila.
- ⑪ Projít přes řádky displeje.
- ⑫ Nastavení konkrétního bodu v řádku na hodnotu **HIGH**.
- ⑬ Získat hodnotu aktuálního bodu.
- ⑭ Pokud je hodnota aktuálního bodu pro řádek **HIGH** a pro sloupec **LOW**, dioda se rozsvítí.
- ⑮ Vypnutí bodu.
- ⑯ Vypnutí celého řádku.



(Př. 7) Dokázali byste pomocí potenciometrů a maticového displeje vytvořit klasickou hru Ping Pong? Pokud byste si nevěděli rady, tak vzorové řešení je k dispozici na GitHub.

## SPOJENÍ MATICOVÉHO DISPLEJE A AKCELEROMETRU

Pro spojení maticového displeje a akcelerometru lze využít z předchozího příkladu. Na základě předchozích kapitol již víte, jak maticový displej používat. Největším problémem je tak zapojení a programování akcelerometru, jako zdroje signálu.



Obr. 5 – Zapojení LED maticice a akcelerometru

- ① Na kontaktní desku je přivedeno napájení **5V** a zemníci vodič **GND** přímo z desky Arduino.
- ② Akcelerometr je připojen pouze na vodiče, aby se sním dalo snadno pohybovat. Z akcelerometru se připojí pin **VIN** (tento pin může být také značen jako VCC, VCC\_IN) do kontaktního pole k napájení. Pin **GND** se připojí do kontaktního pole na zem.
- ③ Z akcelerometru se dále připojí pin **SDA** do desky Arduino. Na desce Arduino tento pin bývá označen stejným názvem, na spodní straně desky.
- ④ Stejně se do desky Arduino připojí i druhý datový pin **SCL**.



Zapojení maticového displeje je stejné, jako u předchozích příkladů, proto lze využít již zapojený obvod maticového displeje a přidat pouze akcelerometr.

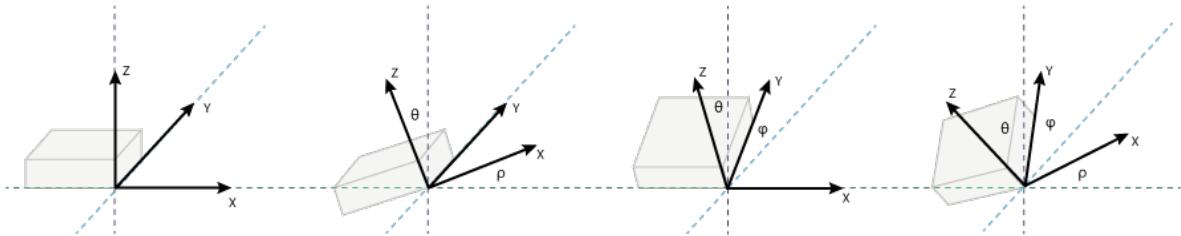
## CO TO JE AKCELEROMETR A JAK PRACUJE

Akcelerometr je malé pohybové čidlo, které měří pohybové zrychlení a to nejlépe ve všech třech osách. Ze znalosti zrychlení a hmotnosti lze zjistit sílu působící na těleso.

Akcelerometry jsou vhodné nejen pro měření odstředivých a setrvačních sil, ale i pro určování pozice tělesa, jeho náklon nebo vibrace. Akcelerometry jsou dnes i v mobilních telefonech a využívají se v leteckém a automobilovém průmyslu.

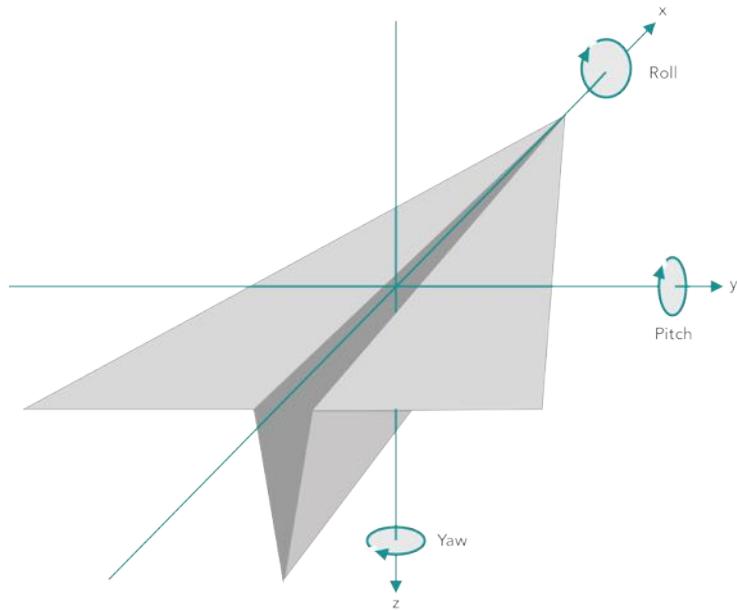
Aby bylo možné definovat úhly akcelerometru ve třech rozměrech **pitch**, **roll** a **theta**, využívají se všechny tři výstupy akcelerometru. **Pitch** (ró), je definováno jako úhel vzhledem k ose X a země. **Roll** (fí) je definováno jako úhel vzhledem k ose Y a země.

**Theta** je úhel vzhledem k ose Z - gravitace.



$$\rho = \arctan\left(\frac{Ax}{\sqrt{Ay^2 + Az^2}}\right) \quad \varphi = \arctan\left(\frac{Ay}{\sqrt{Ax^2 + Az^2}}\right) \quad \theta = \arctan\left(\frac{\sqrt{Ax^2 + Ay^2}}{Az}\right)$$

Je důležité poznamenat, že akcelerometr poskytuje poměrně přesné údaje úhlové orientace za předpokladu, že gravitace je jediná síla působící na snímač. Nicméně, při pohybu a otáčení senzoru, mohou působit jiné síly a dochází ke kolísání přesnosti. Výsledkem potom jsou údaje obsahující šum, který způsobuje sice krátkodobé, ale významné odchylky.



Pro uvedený příklad nás budou zajímat úhly **Pitch** a **Roll**. Zobecněný vzorec pro zrychlení z naměřených hodnot akcelerometru je:

$$G_{Accel} = Raw_{Accel} \frac{Range}{2^{Resolution-1}}$$

Jakmile jsou k dispozici odpovídající hodnoty z akcelerometru, lze pokračovat ve výpočtu úhlů pomocí následujících rovnic.<sup>1</sup>

$$pitch = \arctan\left(\frac{G_y}{\sqrt{G_x^2 + G_z^2}}\right) \quad roll = \arctan\left(\frac{-G_x}{G_z}\right)$$

Tyto vzorce lze v Arduino kódu dají přepsat v následujícím tvaru:

```

1 |   roll  = (atan2(-Yg, Zg)*180.0)/M_PI;
2 |   pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg)))*180.0/M_PI;
```

---

<sup>1</sup> NXP: Freescale Semiconductor [online]. 2013, 2013(6) [cit. 2018-11-15]. Dostupné z: [https://cache.freescale.com/files/sensors/doc/app\\_note/AN3461.pdf](https://cache.freescale.com/files/sensors/doc/app_note/AN3461.pdf)

## PROGRAMOVÝ KÓD – ČTENÍ HODNOT Z AKCELEROMETRU

Programové propojení akcelerometru a maticového displeje je velmi podobné jako při zapojení s potenciometry. Složitější je pouze v tom, že se musí provést čtení dat z akcelerometru. Složitost tohoto programování je závislá na zvolené knihovně, která se použije pro spojení akcelerometru a desky Arduino.



### KNIHOVNA ADXL345

→ Pro jednoduší práci a správnou funkcionalitu akcelerometru, musí být nainstalovaná podpůrná knihovna, kterou najeznete na GitHub.

```
1 #include <Wire.h>
2 #include <ADXL345.h>
3
4 ADXL345 acc;
5
6 const int row[8] = {
7     2, 7, 19, 5, 13, 18, 12, 16
8 };
9
10 const int col[8] = {
11     6, 11, 10, 3, 17, 4, 8, 9
12 };
13
14 int pixels[8][8];
15
16 int x = 5;
17 int y = 5;
18
19 void setup(){
20     acc.begin();
21
22     for(int i = 0; i < 8; i++){
23         pinMode(col[i], OUTPUT);
24         pinMode(row[i], OUTPUT);
25         digitalWrite(row[i], LOW);
26     }
27
28 }
```

```

29     for(int x = 0; x < 8; x++) {
30         for(int y = 0; y < 8; y++) {
31             pixels[x][y] = HIGH;
32         }
33     }
34 }
35
36 void loop(){
37     readSensors();
38     refreshScreen();
39 }
40
41 void readSensors(){
42     double pitch, roll, Xg, Yg, Zg;
43     acc.read(&Xg, &Yg, &Zg);
44
45     roll = (atan2(-Yg, Zg)*180.0)/M_PI;
46     pitch = (atan2(Xg, sqrt(Yg*Yg + Zg*Zg)))*180.0/M_PI;
47
48     pixels[x][y] = HIGH;
49     x = 7 - map(roll, -20, 20, 0, 7);
50     y = map(pitch, -20, 20, 0, 7);
51     pixels[x][y] = LOW;
52 }
53
54 void refreshScreen(){
55     for(int j = 0; j<8;j++){
56         digitalWrite(row[j], HIGH);
57         for(int k = 0; k<8; k++){
58             int thisPixel = pixels[j][k];
59             digitalWrite(col[k], thisPixel);
60             if (thisPixel == LOW) {
61                 digitalWrite(col[k], HIGH);
62             }
63         }
64         digitalWrite(row[j], LOW);
65     }
66 }
```

- ① Připojení knihovny **Wire.h**, která umožňuje komunikaci se zařízeními s I2C/TWI.
- ② Připojení knihovny **ADXL345.h**, zajišťuje jednoduší komunikaci mezi deskou Arduino a akcelerometrem.
- ③ Vytvoření instance třídy ADXL 345 pro práci s akcelerometrem.
- ④ Zahájení komunikace akcelerometru s deskou Arduino.
- ⑤ Deklarace funkce **readSensor**, která provádí čtení a výpočet dat z akcelerometru.

- ⑥ Deklarace proměnných pro výpočet klopení a klonění.
- ⑦ Čtení hodnot z akcelerometru.
- ⑧ Výpočet klopení.
- ⑨ Výpočet klonění.
- ⑩ Namapování hodnoty z akcelerometru pro **klopení**.
- ⑪ Namapování hodnoty z akcelerometru pro **klonění**.



Z programového kódu je patrné, že při využití funkcí je změna kódu minimální. Prakticky došlo pouze ke změně ve funkci pro čtení ze senzorů, která spočívala v převedení získaných hodnot z akcelerometru na hodnoty použitelné k namapování pro maticový displej.

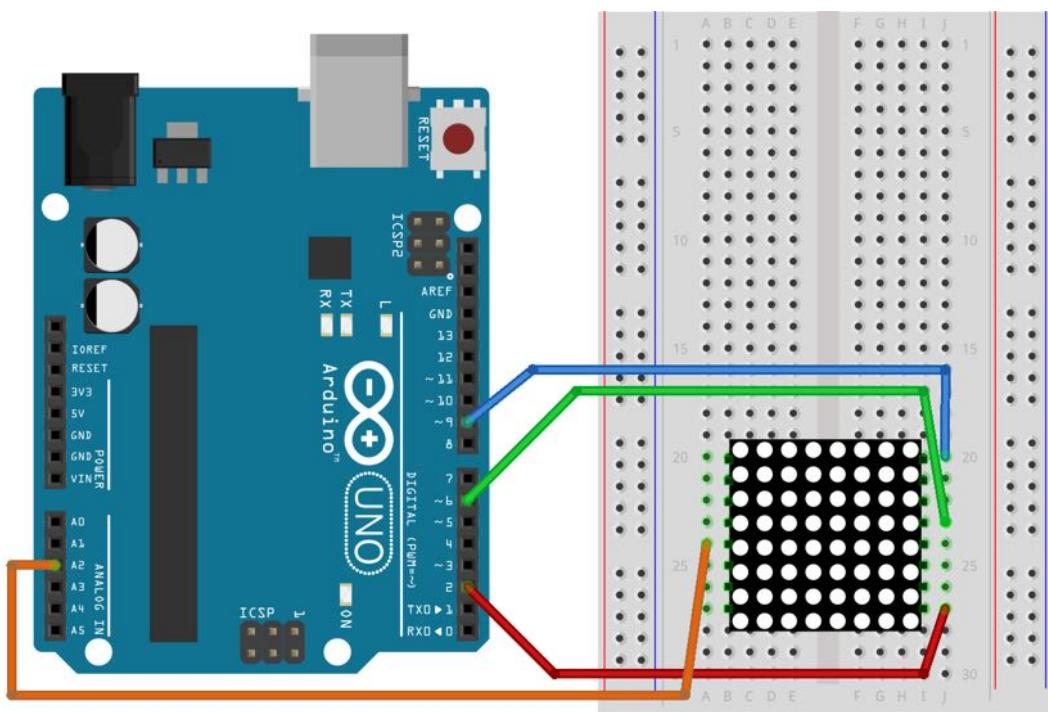


(Př. 8) Předchozí schéma a programový kód rozšiřte o zapojení dvou servomotorů, které budou reagovat na polohu akcelerometru.

## ŘEŠENÍ PŘÍKLADŮ

## PŘ. 1

Změna zapojení obvodu má vést k ujasnění si principu zapojení maticového displeje. Při řešení příkladu stačí využít tabulku pinů Tab. 2 - Rozložení pinů.



Obr. 6 – Základní zapojení LED maticy

```
1 int pinA=2;
2 int pinB=6;
3
4 int pinC=9;
5 int pinD=A2;
6
7 void setup() {
8     pinMode(pinA,OUTPUT);
9     pinMode(pinB,OUTPUT);
10    digitalWrite(pinA,HIGH);
11    digitalWrite(pinB,HIGH);
12
13    pinMode(pinC,OUTPUT);
```

```
14  pinMode(pinD,OUTPUT);
15  digitalWrite(pinC,HIGH);
16  digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21  digitalWrite(pinB,LOW);
22  digitalWrite(pinD,HIGH);
23  delay(200);
24
25  digitalWrite(pinB,LOW);
26  digitalWrite(pinD,HIGH);
27  delay(200);
28 }
```

## PŘ. 2

K vyřešení příkladu se využije stejné zapojení jako u příkladu 1. Programový kód bude takřka stejný. Pouze se změní pořadí zapínání diod.

```
1 int pinA=2;
2 int pinB=6;
3
4 int pinC=9;
5 int pinD=A2;
6
7 void setup() {
8     pinMode(pinA,OUTPUT);
9     pinMode(pinB,OUTPUT);
10    digitalWrite(pinA,HIGH);
11    digitalWrite(pinB,HIGH);
12
13    pinMode(pinC,OUTPUT);
14    pinMode(pinD,OUTPUT);
15    digitalWrite(pinC,HIGH);
16    digitalWrite(pinD,HIGH);
17
18 }
19
20 void loop() {
21     digitalWrite(pinB,HIGH); // změna na HIGH
22     digitalWrite(pinD,LOW); // změna na LOW
23     delay(200);
24
25     digitalWrite(pinB,LOW);
26     digitalWrite(pinD,HIGH);
27     delay(200);
28 }
```

### PŘ. 3

Původní programový kód stačí rozdělit do dvou funkcí. Např. **refreshScreen** a **Clear**.

```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {  
6     6, 11, 10, 3, 17, 4, 8, 9  
7 };  
8  
9 void setup(){  
10    for(int i = 0; i < 8; i++){  
11        pinMode(col[i], OUTPUT);  
12        pinMode(row[i], OUTPUT);  
13        digitalWrite(col[i], HIGH);  
14        digitalWrite(row[i], LOW);  
15    }  
16}  
17  
18 void loop(){  
19    refreshScreen();  
20}  
21  
22 void refreshScreen(){  
23    for(int j = 0; j<8;j++){  
24        digitalWrite(col[j], LOW);  
25        for(int k = 0; k<8; k++){  
26            digitalWrite(row[k], HIGH);  
27        }  
28        clear();  
29    }  
30}  
31  
32 void Clear(){  
33    for(int i = 0; i<8; i++){  
34        digitalWrite(row[i],LOW);  
35        digitalWrite(col[i],HIGH);  
36    }  
37}
```

## PŘ. 4

Postačí upravit pořadí zapínání diod ve funkci **refreshScreen**.

```
1 const int row[8] = {  
2     2, 7, 19, 5, 13, 18, 12, 16  
3 };  
4  
5 const int col[8] = {  
6     6, 11, 10, 3, 17, 4, 8, 9  
7 };  
8  
9  
10 void setup(){  
11     for(int i = 0; i < 8; i++){  
12         pinMode(col[i], OUTPUT);  
13         pinMode(row[i], OUTPUT);  
14         digitalWrite(col[i], HIGH);  
15         digitalWrite(row[i], LOW);  
16     }  
17 }  
18  
19 void loop(){  
20     refreshScreen();  
21 }  
22  
23 void refreshScreen(){  
24     for(int j = 0; j<8;j++){  
25         digitalWrite(row[j], LOW);  
26         for(int k = 0; k<8; k++){  
27             digitalWrite(col[k], HIGH);  
28             delay(100);  
29             digitalWrite(col[k], LOW);  
30         }  
31         digitalWrite(row[j], HIGH);  
32     }  
33 }
```

## PŘ. 5

V původním programovém kódu postačí změnit dvouozměrné pole image.



Definice tvaru symbolů je velmi snadné. Můžete využít nástroj, pomocí něhož si symbol „naklikáte“ a následně použijete vygenerované dvouozměrné pole vypnutých/zapnutých diod, které vložíte do programového kódu.

Odkaz: <https://www.prf.jcu.cz/generator-led-matrix/index.htm>

```
1 // Srdce
2 byte image[8][8] = {
3     {0,0,0,0,0,0,0,0},
4     {0,1,1,0,0,1,1,0},
5     {1,0,0,1,1,0,0,1},
6     {1,0,0,0,0,0,0,1},
7     {1,0,0,0,0,0,0,1},
8     {0,1,0,0,0,0,1,0},
9     {0,0,1,0,0,1,0,0},
10    {0,0,0,1,1,0,0,0}};
11
12 // Smajlík
13 byte image[8][8] = {
14     {0,0,1,1,1,1,0,0},
15     {0,1,0,0,0,0,1,0},
16     {1,0,1,0,0,1,0,1},
17     {1,0,0,0,0,0,0,1},
18     {1,0,1,0,0,1,0,1},
19     {1,0,0,1,1,0,0,1},
20     {0,1,0,0,0,0,1,0},
21     {0,0,1,1,1,1,0,0}};
```

## PŘ. 6

V příkladu je jednoduchá inovace, a to v podobě předávání parametru funkce **refreshScreen**. Tímto parametrem je pole s různým obrazcem.

```
1  const int row[8] = {
2      2, 7, 19, 5, 13, 18, 12, 16
3  };
4
5
6  const int col[8] = {
7      6, 11, 10, 3, 17, 4, 8, 9
8  };
9
10 // Velke srdce
11 unsigned char image[8][8] = {
12     {0,0,0,0,0,0,0,0},
13     {0,1,1,0,0,1,1,0},
14     {1,0,0,1,1,0,0,1},
15     {1,0,0,0,0,0,0,1},
16     {1,0,0,0,0,0,0,1},
17     {0,1,0,0,0,0,1,0},
18     {0,0,1,0,0,1,0,0},
19     {0,0,0,1,1,0,0,0}};
20
21 // Male srdce
22 unsigned char images[8][8] = {
23     {0,0,0,0,0,0,0,0},
24     {0,0,0,0,0,0,0,0},
25     {0,0,0,1,0,1,0,0},
26     {0,0,1,0,1,0,1,0},
27     {0,0,1,0,0,0,1,0},
28     {0,0,0,1,0,1,0,0},
29     {0,0,0,0,1,0,0,0},
30     {0,0,0,0,0,0,0,0}};
31
32 void setup(){
33     for(int i = 0; i < 8; i++){
34         pinMode(col[i], OUTPUT);
35         pinMode(row[i], OUTPUT);
36         digitalWrite(col[i], HIGH);
37         digitalWrite(row[i], LOW);
38     }
39 }
40
```

```
41 void loop(){
42     // Zobrazeni vždy po dobu 100 iteraci
43     for(int i = 0; i < 100; i++){
44         refreshScreen(image);
45     }
46
47     for(int i = 0; i < 100; i++){
48         refreshScreen(imageS);
49     }
50 }
51
52
53 void refreshScreen(unsigned char dat[8][8]){
54     for(int j = 0; j<8;j++){
55         digitalWrite(col[j], LOW);
56         for(int k = 0; k<8; k++){
57             digitalWrite(row[k], dat[k][j]);
58         }
59         delay(1);
60         Clear();
61     }
62 }
63
64 void Clear(){
65     for(int i = 0; i<8; i++){
66         digitalWrite(row[i],LOW);
67         digitalWrite(col[i],HIGH);
68     }
69 }
```

## 8. ROBOTICKÁ RUKA – JOYSTICK

VĚTŠINA INDUSTRIÁLNÍCH I DOMÁCÍCH SYSTÉMŮ MÁ NĚJAKÉ VSTUPNÍ ZAŘÍZENÍ – OVLADAČ. V TÉTO KAPITOLE SE SEZNÁMÍTE S JOYSTICKEM, KTERÝ ZNÁTE Z POČÍTAČOVÝCH HER, JAKO VSTUPNÍM ZAŘÍZENÍM PRO OVLÁDÁNÍ ROBOTICKÉ RUKY.

### CÍLE

- ① Pochopit princip joysticku a jeho využití jako vstupní zařízení pro konstrukce na bázi Arduina.
- ② Sestrojit jednoduchou robotickou ruku za pomocí krokového motorku, servo motorku a dílů vytiskněných na 3D tiskárně.
- ③ Tuto ruku naprogramovat a nastavit tak, aby uměla vymáchat pytlík čaje v připraveném hrníčku s vodou o požadované teplotě.

Čas: **135 min**

Úroveň: ■■■■■■■

Vychází z: **5, 6, 7**



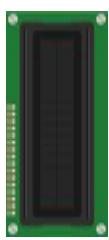
Servo motor



Krokový motor



PS2 Joystick



LCD display



Potenciometr

### POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU – JOYSTICK I



V této kapitole na sebe úlohy přímo navazují, a proto zejména mezi druhou a třetí hodinou, je-li to možné, nerozpojte obvod a ponechte pro další hodinu.



## PŘÍPRAVA

Co bude v této hodině potřeba?

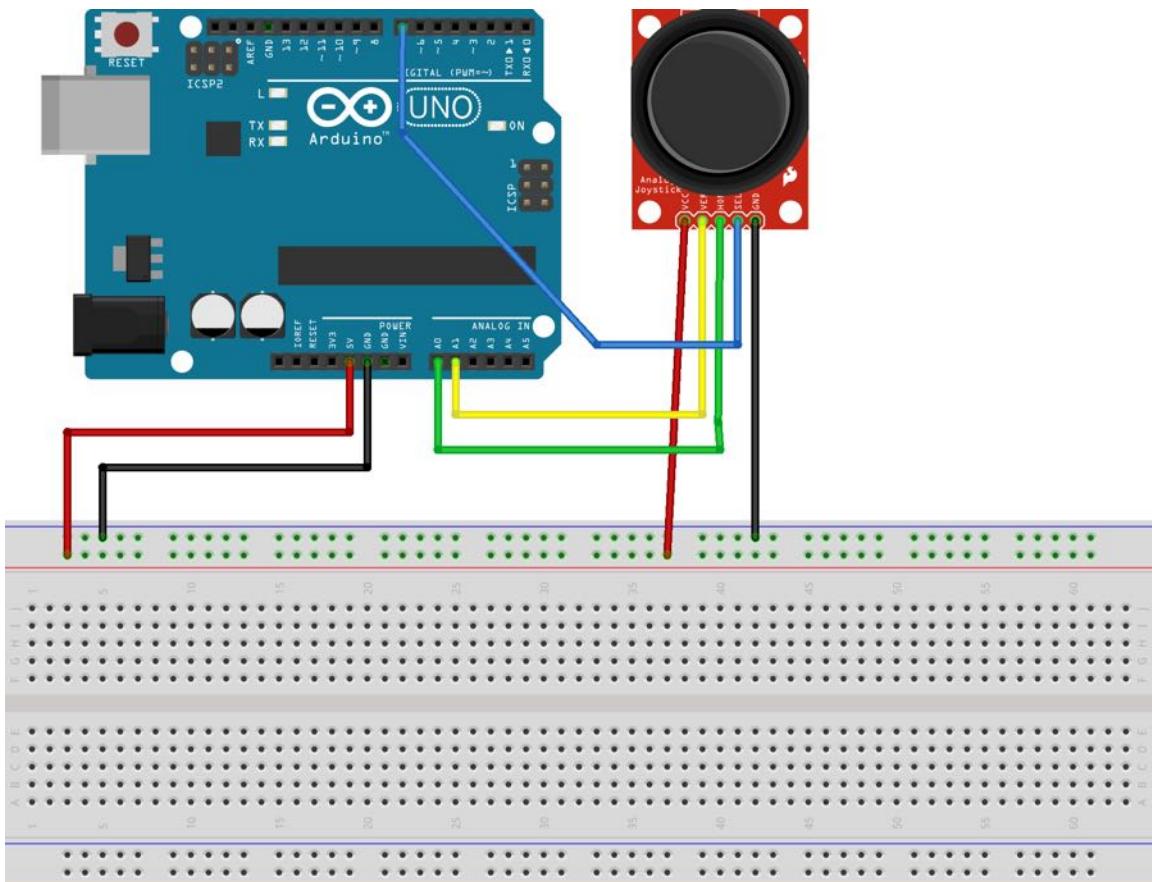
- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, vodiče, modul joysticku.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 8.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⏳ 10 minut

Na úvod rozdejte studentům sady Arduino. Pohovořte o různých dálkovém ovládání. Popište si princip joysticku a kde všude se joystick používá.

## 2. KROK 🕒 5 minut

Studenti si sestaví jednoduchý obvod s joystickem.



### 3. KROK 10 minut

Studenti přepíší (nahrají si) následující kód do programu Arduino IDE, přeloží jej a nahrají do Arduina:

```

1 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
2 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
3 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
4 int x,y,z;
5
6 void setup() {
7     Serial.begin(9600);
8     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
9 joysticku
10    Serial.println("Test joysticku");
11 }
12
13 void loop() {
14     x=analogRead(JoyStick_X);

```

```

15  y=analogRead(JoyStick_Y);
16  z=digitalRead(JoyStick_Z);
17  Serial.print("X = ");
18  Serial.print(x);
19  Serial.print(", Y = ");
20  Serial.print(y);
21  Serial.print(", Z = ");
22  Serial.println(z);
23  delay(500);
24 }

```

## 4. KROK 🕒 10 minut

Studenti si v Arduino IDE spustí sériový monitor a otestují chování joysticku.

## 5. KROK – volitelný

Zbyde-li čas nechte studenty připojit LCD panel (zapojení v úloze 3 této kapitoly) a nechte je vypisovat hodnoty joysticku na tento panel.

Alternativně lze tuto úlohu též řešit namísto úlohy 3, zejména pokud nemáte 3D tiskárnu.



### ➔ Nefunguje joystick

Zapojení v desce – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů desky Arduino.

Zapojení v joysticku – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů joysticku.

### ➔ Nejde nahrát kód do desky

USB kabel – ujistěte se, že máte desku Arduino připojenou k počítači.

Správný port – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

### ➔ Nefunguje Sériový monitor

Nezobrazuje se text – Zkontrolujte zapojení USB kabelu, zkontrolujte nastavení správné rychlosti – 9660 baudů.

Zobrazuje se nesmyslný text – Zkontrolujte nastavení správné rychlosti – 9660 baudů.

# PRACOVNÍ LIST - JOYSTICK I

PRVNÍ SEZNÁMENÍ S OVLÁDÁNÍM ARDUINA POMOCÍ JOYSTICKU. V TÉTO ČÁSTI SE SEZNÁMÍTE S JEHO ZAPOJENÍM A FUNKČNOSTÍ.

## CO SE NAUČÍTE

- ① Princip ovládání joysticku.
- ② Zapojení joysticku.
- ③ Naprogramování prvního programu pro ovládání pomocí joysticku.



## CO BUDETE POTŘEBOVAT

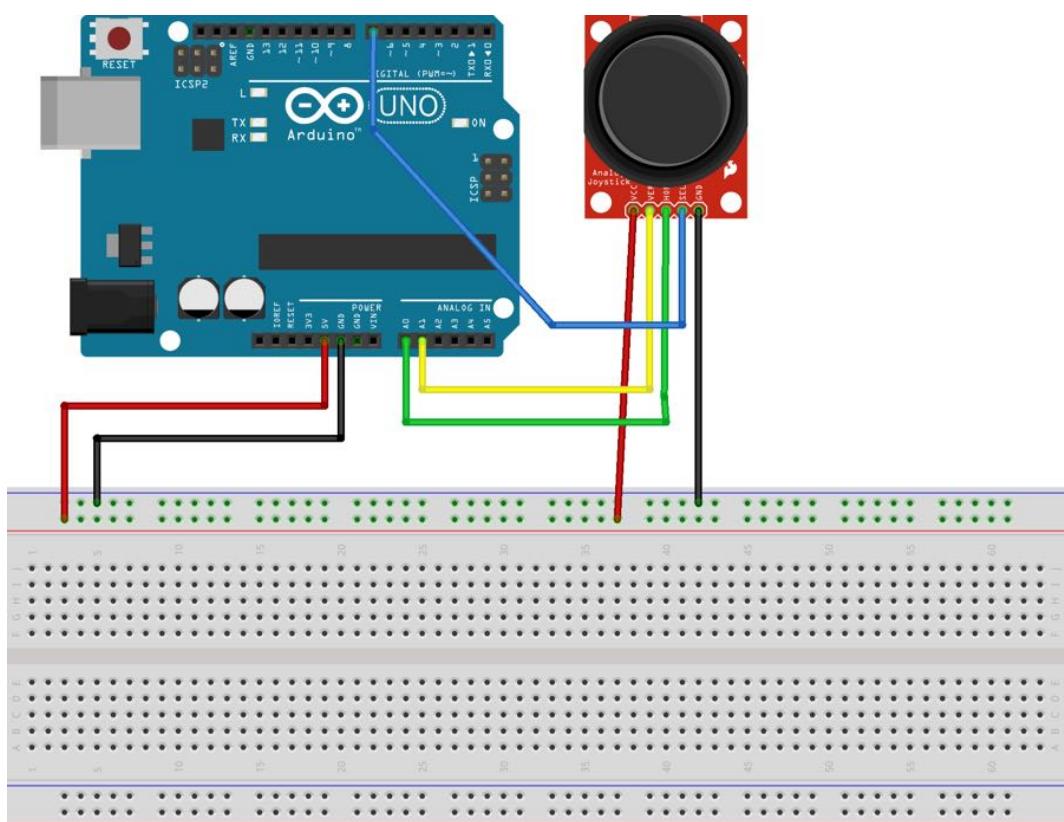
- ① Joystick.
- ② Desku Arduino.
- ③ Kontaktní pole.
- ④ Vodiče typu zástrčka- zástrčka.
- ⑤ Volitelně LCD panel a potenciometr.



POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

① Podle schématu zapojte elektronický obvod.



- ② Spusťte program Arduino IDE a napište následující programový kód.

```
1 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
2 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
3 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
4 int x,y,z;
5
6 void setup() {
7     Serial.begin(9600);
8     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
9 joysticku
10    Serial.println("Test joysticku");
11 }
12
13 void loop() {
14     x=analogRead(JoyStick_X);
15     y=analogRead(JoyStick_Y);
16     z=digitalRead(JoyStick_Z);
17     Serial.print("X = ");
18     Serial.print(x);
19     Serial.print(", Y = ");
20     Serial.print(y);
21     Serial.print(", Z = ");
22     Serial.println(z);
23     delay(500);
24 }
```

- ③ Přeložte program a nahrajte jej do Arduina.

- ④ Otevřete si v **Arduino IDE Sériový monitor**, kliknutím na ikonu 

- ⑤ Testujte joystick a sledujte odezvu v sériovém monitoru.



#### ÚKOL PRO VÁS

→ Zkuste zapojit LCD panel a zobrazovat stav joysticku na něm.

→ Použijte schéma z úlohy 3 této kapitoly.



# PRŮVODCE HODINOU – JOYSTICK II



Pro tuto hodinu musíte mít díly vytiskněny na 3D tiskárně. Pokud nemáte 3D tiskárnu nebo ji nechcete používat, vytvořte místo této a příští úlohy, úlohu se zapojením joysticku a LCD displeje.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, vodiče typu zástrčka-zástrčka, joystick, Servo, obvod L9110H (ovladač motoru) a DC motor.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.

## 1. KROK 5 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že v této hodině navází na předchozí příklad, který se týkal zapojení a ovládání obvodu s joystickem. Naučí se vzdáleně ovládat servo motor a krokový motor pomocí joysticku.

## 2. KROK 15 minut

Sestavte obvod dle schématu. Schéma jim promítněte na projektoru a naleznou jej i na pracovních listech.

### 3. KROK 10 minut

Studenti si zapíší kód do **Arduina IDE** a nahrají si jej do **Arduina**. Kód studentům vhodným způsobem připravte, ať jej nemusí opisovat, ale promítněte jej na projektoru a vysvětlete jej.

### 4. KROK 15 minut

Pokud studenti vše zvládli, mohou nyní testovat robotickou ruku pomocí joysticku.



#### ÚKOLY PRO STUDENTY

- ➔ A) Experimentujte s úhly, o které se otáčí servo.
- ➔ B) Experimentujte s rychlosí DC motoru.

## PRACOVNÍ LIST – JOYSTICK II

POKRAČOVÁNÍ V SEZNAMOVÁNÍ SE S JOYSTICKEM A JEHO POUŽÍVÁNÍM.  
TENTOKRÁT BUDEME POMOCÍ JOYSTICKU OVLÁDAT DVA MOTORKY – DC MOTOR  
A SERVO.

### CO SE NAUČÍTE

- ① Zopakujete si, zapojení joysticku a jeho použití.
- ② Zopakujete si zapojení DC motoru a serva.
- ③ Vytvoření programu pro vzdálené ovládání DC motoru  
a serva pomocí joysticku.
- ④ Vytvořené zapojení si otestujete.

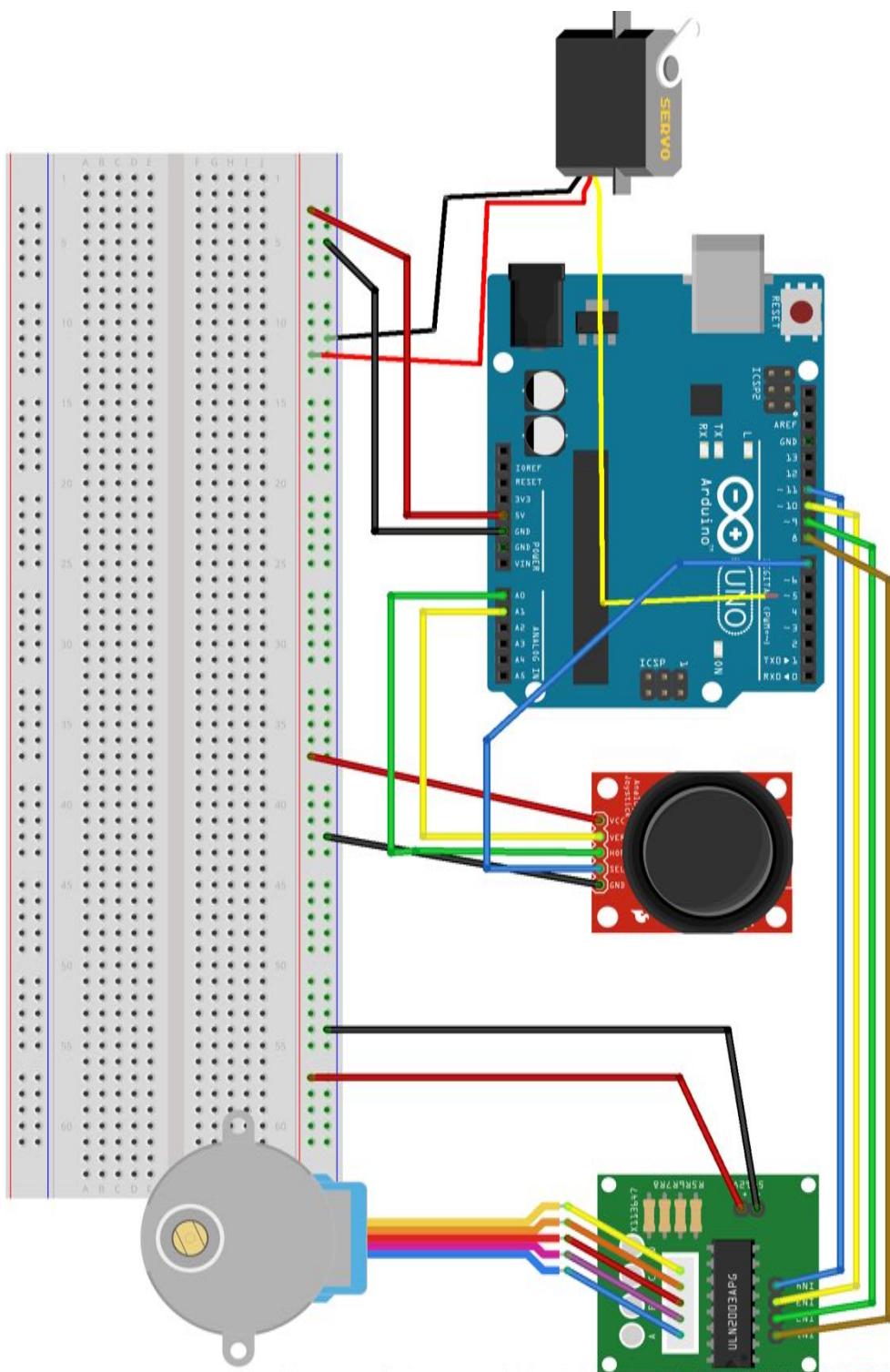


### CO BUDETE POTŘEBOVAT

- ① Deska Arduino s USB kabelem.
- ② Kontaktní pole.
- ③ Vodiče typu zástrčka-zástrčka.
- ④ Joystick.
- ⑤ Servo.
- ⑥ DC motor.

## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.



- ② Spusťte program Arduino IDE a napište následující programový kód.

```
1 #include <Servo.h>
2
3 Servo myservo;//Vytvoření objektu pro řízení krokového motoru
4 int poloha; //Svislá poloha ruky
5
6 // Piny pro krokový motor
7 const int in1 = 8;
8 const int in2 = 9;
9 const int in3 = 10;
10 const int in4 = 11;
11 // proměnná pro nastavení rychlosti,
12 // se zvětšujícím se číslem se rychlosť zmenšuje
13 int rychlosť = 8;
14
15 //Joystick
16 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
17 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
18 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
19 int x,y,z;
20
21 void setup() {
22     myservo.attach(6);//Servo motor je na pinu 6
23     myservo.write(0);//Ruka do výchozí polohy
24     poloha = 0; //Pamatuj si tuto polohu
25     // inicializace digitálních výstupů pro krokový motor
26     pinMode(in1, OUTPUT);
27     pinMode(in2, OUTPUT);
28     pinMode(in3, OUTPUT);
29     pinMode(in4, OUTPUT);
30     //inicjalizace Joysticku
31     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
32 joysticku
33 }
34
35 void loop() {
36     pohyb();
37     while(1);
38 }
39
40 void rotacePoSmeru(int uhel) {
41     for(int i=0;i<(uhel*64/45);i++){
```

```

42     krok(1,0,0,0);
43     krok(1,1,0,0);
44     krok(0,1,0,0);
45     krok(0,1,1,0);
46     krok(0,0,1,0);
47     krok(0,0,1,1);
48     krok(0,0,0,1);
49     krok(1,0,0,1);}
50 }
51
52 void rotaceProtismeru(int uhel) {
53     for(int i=0;i<(uhel*64/45);i++){
54         krok(1,0,0,1);
55         krok(0,0,0,1);
56         krok(0,0,1,1);
57         krok(0,0,1,0);
58         krok(0,1,1,0);
59         krok(0,1,0,0);
60         krok(1,1,0,0);
61         krok(1,0,0,0);}
62 }
63
64 void krok(int a, int b, int c, int d){
65     digitalWrite(in1, a);
66     digitalWrite(in2, b);
67     digitalWrite(in3, c);
68     digitalWrite(in4, d);
69     delay(rychlost);
70 }
71
72 void pohyb(){
73     int x,y,z;
74     z=1;
75     while (z) {
76         x=analogRead(JoyStick_X);
77         y=analogRead(JoyStick_Y);
78         z=digitalRead(JoyStick_Z);
79         if (x>550) { //doprava
80             rotacePoSmeru(5);
81         }
82         else if (x<480){ //doleva
83             rotaceProtismeru(5);
84         }
85         else if (y<480){ //dolu
86             if (poloha>=5) {
87                 poloha=poloha-5;
88                 myservo.write(poloha);
89                 delay(1000);

```

```

90     }
91   }
92   else if (y>550){ //nahoru
93     if (poloha<=170) {
94       poloha=poloha+5;
95       myservo.write(poloha);
96       delay(1000) ;
97     }
98   }
99 }
100 delay(100);
101 }

```

- ③ Program odladěte a nahrajte do Arduina.
- ④ Nyní vezměte váš joystick a vyzkoušejte program.
- ⑤ Pokud vše funguje, tak výborně. Můžete se vrhnout na samostatné úkoly.

#### RYCHLÝ TIP

→ Pokud budete řešit i následující úkol, ponechte si vše zapojené.



#### ÚKOL PRO VÁS

→ Experimentujte se změnou různých parametrů. S úhly o které se otáčí servo a rychlostí DC motoru.



#### VOLITELNÝ ÚKOL

→ Máte-li vytištěné díly pro stavbu robotické ruky, můžete jí nyní sestrojit a zkusit rozpohybovat.

# PRŮVODCE HODINOU – JOYSTICK III



## PŘÍPRAVA

Co bude v této hodině navíc proti minule potřeba?

- ① LCD displej.
- ② Potenciometr.
- ③ Libovolný šálek či hrníček, tak do 8 cm výšky a 8 cm průměru.
- ④ Pokud chcete opravdu vařit čaj, je třeba rychlovárná konvice a sáčky čaje.

### 1. KROK 15 minut

Sestavte obvod včetně robotické ruky.

### 2. KROK 15 minut

Nahrajte kód do Arduina. Vzhledem k rozsáhlosti tohoto kódu nelze doporučit jeho opisování z tabule ani z pracovních listů, proto kód studentům připravte ke stažení. Prodiskujte s nimi změny oproti minulému případu.

### 3. KROK 15 minut

Vyzkoušejte si ovládání systému nejprve s prázdným hrnkem a pokud vše půjde dobře zkuste si uvařit skutečně čaj.

# PRACOVNÍ LIST – JOYSTICK III

POKRAČOVÁNÍ V SEZNAMOVÁNÍ SE S JOYSTICKEM A JEHO POUŽÍVÁNÍM.  
TENTOKRÁT SESTROJÍME ROBOTICKOU RUKU PRO MÍCHÁNÍ PYTLÍKU S ČAJEM

## CO SE NAUČÍTE

- ① Zopakujete si, zapojení LCD panelu.
- ② Vyzkoušíte si práci s jednoduchou robotickou rukou.



## CO BUDETE POTŘEBOVAT

Oproti minulé hodině budeme navíc potřebovat

- ① LCD displej.
- ② Potenciometr.

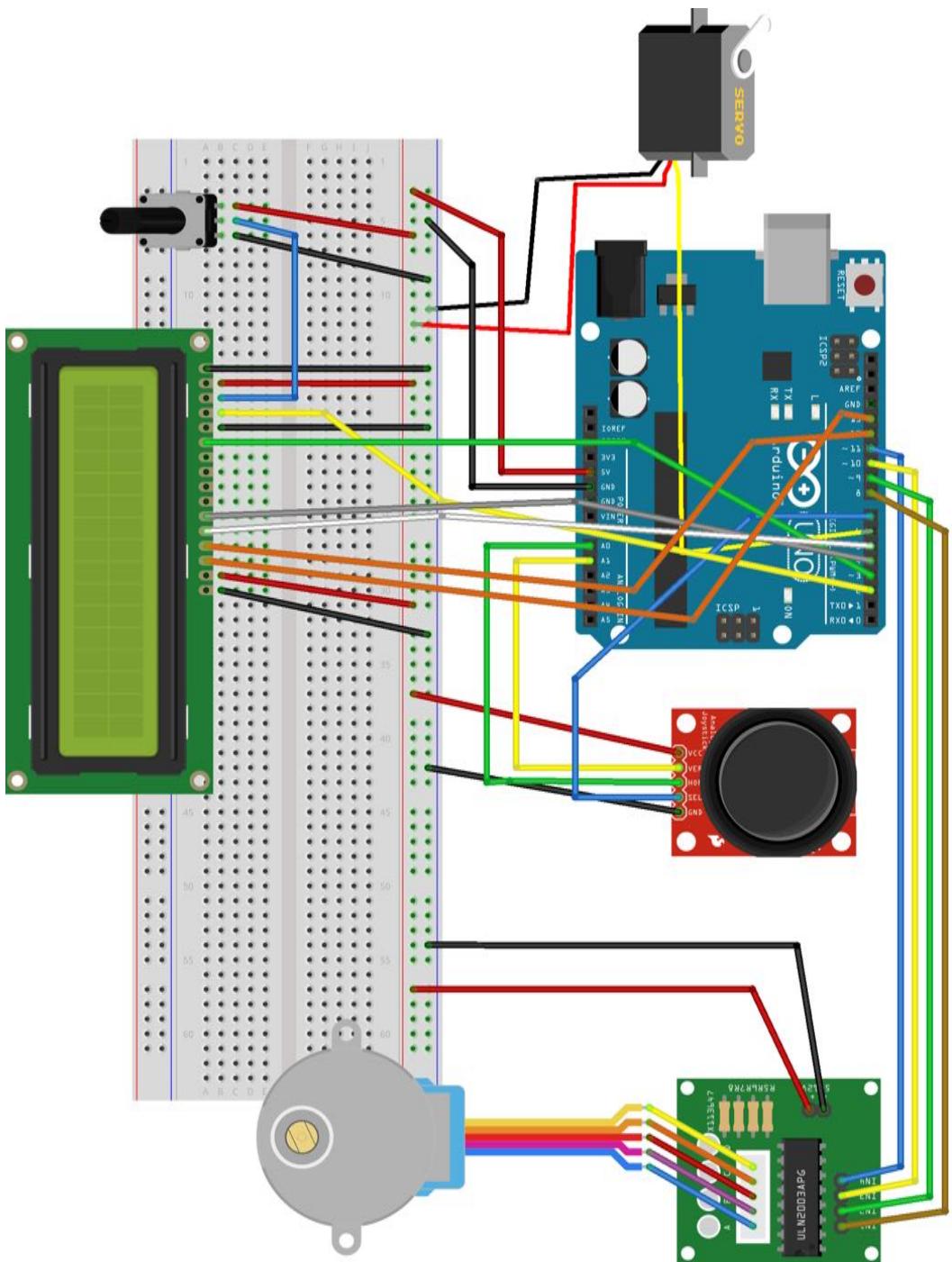


## A JDĚTE NA TO ...

- ① Pokud ještě nemáte, sestavte robotickou ruku, dle fotografie.



② Podle schématu zapojte elektronický obvod.



③ Spusťte program Arduino IDE a napište následující programový kód.

```
1 #include <LiquidCrystal.h>
2
3 #include <Servo.h>
4
5 Servo myservo; //Vytvoření objektu pro řízení krokového motoru
6
7 int poloha; //Svislá poloha ruky
8
9 // Piny pro krokový motor
10 const int in1 = 8;
11 const int in2 = 9;
12 const int in3 = 10;
13 const int in4 = 11;
14
15 // Proměnná pro nastavení rychlosti,
16 // se zvětšujícím se číslem se rychlosť zmenšuje
17 int rychlosť = 8;
18 int uhel1;
19 int x,y,z;
20 int i,j;
21
22 //Joystick
23 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
24 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
25 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
26
27 // Piny pro připojení displeje
28 LiquidCrystal lcd(2, 3, 4, 5, 12, 13);
29
30 int minut; //Počet minut pro máchání čaje
31
32 void setup() {
33     myservo.attach(6); //Servo motor je na pinu 6
34     myservo.write(0); //Ruka do výchozí polohy
35     poloha = 0; //Pamatuj si tuto polohu
36     // inicializace digitálních výstupů pro krokový motor
37     pinMode(in1, OUTPUT);
38     pinMode(in2, OUTPUT);
39     pinMode(in3, OUTPUT);
40     pinMode(in4, OUTPUT);
41     //inicjalizace Joysticku
42 }
```

```

43  pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
44  joysticku
45  lcd.begin(16, 2); // Počet sloupců a řádek LCD displeje
46
47 }
48
49 void loop() {
50     //main program
51     lcd.clear();
52     lcd.print("Pouzij joystick");
53     lcd.setCursor(0,2);
54     lcd.print("pro nastavení");
55     pocatecni_nastaveni(); //Nastavení polohy ruky "nad hrnek"
56     lcd.clear(); //Nastavení polohy pro připevnění pytlíku
57     myservo.write(poloha+15);
58     delay(1000);
59     rotaceProtismeru(90);
60     delay(1000);
61     lcd.print("Ruka pripravena");
62     lcd.setCursor(0,2);
63     lcd.print("připevní čaj a potvrd"); // Potvrdit stiskem
64                                     // joysticku
65     delay(1000);
66     z=1;
67     while (z) {
68         x=analogRead(JoyStick_X);
69         y=analogRead(JoyStick_Y);
70         z=digitalRead(JoyStick_Z);
71         delay(100);
72     }
73     delay(1000);
74     minut=pocetMinut(); //Nastavení počtu minut pro máchání
75     delay(1000);
76     rotacePoSmeru(90); //Najedeme nad čaj
77     delay(1000);
78     myservo.write(poloha-5); //Máchání čaje
79     for (i=minut;i;i--){
80         lcd.clear();
81         lcd.print("Zbyva:");
82         lcd.setCursor(0,1);
83         lcd.print(i);
84         lcd.print(" minut");
85         for (j=1;j<10;j++){
86             myservo.write(poloha-7);
87             delay(3000);
88             myservo.write(poloha+7);
89             delay(3000);
90     }

```

```

91     }
92     lcd.clear(); //Konec máchání
93     lcd.print("Hotovo");
94     myservo.write(poloha+20);
95     delay(10000);
96     rotacePoSmeru(90); //Odjezd doprava
97     myservo.write(15);
98     while(1) {} //Nekonečná smyčka
99 }
100 // zde následují funkce pro volání jednotlivých
101 // kroků pro otočení po či proti směru hodinových
102 // ručiček
103 void rotacePoSmeru(int uhel) {
104     for(int i=0;i<(uhel*64/45);i++){
105         krok(1,0,0,0);
106         krok(1,1,0,0);
107         krok(0,1,0,0);
108         krok(0,1,1,0);
109         krok(0,0,1,0);
110         krok(0,0,1,1);
111         krok(0,0,0,1);
112         krok(1,0,0,1);
113     }
114 }
115 void rotaceProtiSmeru(int uhel) {
116     for(int i=0;i<(uhel*64/45);i++){
117         krok(1,0,0,1);
118         krok(0,0,0,1);
119         krok(0,0,1,1);
120         krok(0,0,1,0);
121         krok(0,1,1,0);
122         krok(0,1,0,0);
123         krok(1,1,0,0);
124         krok(1,0,0,0);
125     }
126 }
127 // každý krok obsahuje výrobcem dané pořadí
128 // pro správné spínání motoru a následnou
129 // pauzu, kterou určujeme rychlosť otáčení
130 void krok(int a, int b, int c, int d){
131     digitalWrite(in1, a);
132     digitalWrite(in2, b);
133     digitalWrite(in3, c);
134     digitalWrite(in4, d);
135     delay(rychlosť);
136 }
137 void pocatecni_nastaveni(){
138     int x,y,z;

```

```

140 z=1;
141 while (z) {
142     x=analogRead(JoyStick_X);
143     y=analogRead(JoyStick_Y);
144     z=digitalRead(JoyStick_Z);
145     if (x>550) { //doprava
146         rotacePoSmeru(5);
147     }
148     else if (x<480){ //doleva
149         rotaceProtiSmeru(5);
150     }
151     else if (y<480){ //dolu
152         if (poloha>=5) {
153             poloha=poloha-5;
154             myservo.write(poloha);
155             delay(1000);
156         }
157     }
158     else if (y>550){ //nahoru
159         if (poloha<=170) {
160             poloha=poloha+5;
161             myservo.write(poloha);
162             delay(1000);
163         }
164     }
165     }
166     delay(100);
167 }
168 int pocetMinut(){
169     int m=2;
170     int x, y, z;
171     z=1;
172     lcd.setCursor(0,0);
173     lcd.print("Maximum 9 minut");
174     lcd.setCursor(0,1);
175     lcd.print("Louhovat: ");
176     lcd.print(m);
177     lcd.print(" min");
178     while (z) {
179         x=analogRead(JoyStick_X);
180         y=analogRead(JoyStick_Y);
181         z=digitalRead(JoyStick_Z);
182         if (y>550){ //dolu
183             if (m) {
184                 m=m-1;
185             }
186             lcd.setCursor(10,1);
187             lcd.print(m);

```

```

188     }
189     if (y<480){ //nahoru
190         if (m<9) {
191             m=m+1;
192             lcd.setCursor(10,1);
193             lcd.print(m);
194         }
195         delay(300);
196     }
197     return m;
}

```

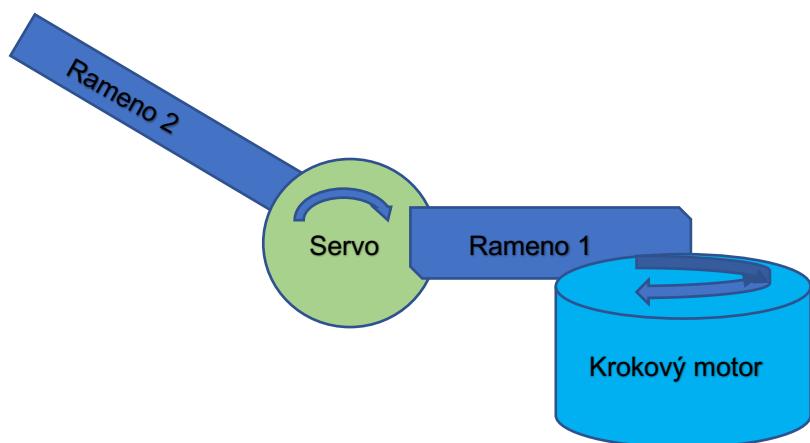
- ④ Program odladěte a nahrajte do Arduina.
- ⑤ Nyní zkuste nasimulovat vymáčhání pytlíku v čaji a až si budete jistí, můžete si opravdu zkusí uvařit čaj. Postupujte dle následujícího návodu.

## PRÁCE S ROBOTICKOU RUKOU

- ① Po spuštění programu je třeba jako první věc nastavit robotickou ruku nad prázdný hrnek. Doporučuji nastavovat na prázdný hrnek. Ruku nastavte trochu napravo od středu hrnku a částečně jí ponořte pod horní okraj (cca. 1 cm). Potvrďte stiskem joysticku.
- ② Ruka si nyní najede vlevo od hrnku na pozici, na které můžete připevnit pytlík s čajem. Snažte se při tom nepohybovat s rukou. Pokud se vám to nepodaří, připevněte pytlík, stiskněte reset Arduinu a vrátěte se na krok jedna. Alternativně začínejte rovnou s připevněným pytlíkem.
- ③ Připravte si talířek nebo nějakou podložku, nad kterou ruka donese vylouhovaný pytlík. Umístěte jí na pozici asi 90° napravo od hrnku.
- ④ Nastavte počet minut, po které se má čaj louhovat, nalejte do hrnku vodu s požadovanou teplotou a potvrďte joystickem.
- ⑤ Robotická ruka provede vymáčhání čaje a po nastavené době odnese pytlík napravo od hrnku nad připravenou podložku.
- ⑥ Tímto celý cyklus končí, pro jeho opakování je nutné stisknout reset na Arduinu.

# PODROBNÝ PRŮVODCE TEORIÍ

Robotická ruka, též zvaná **manipulátor**, patří mezi takzvané **sériové roboty** neboli řetězce. To jsou druhy robotů, charakteristické pravidelným střídáním pevných částí (zvaných **linky** nebo ramena) s pohyblivými částmi (zvanými **klouby**). Tyto klouby mohou být sférické, cylindrické nebo posuvné. V našem případě použijeme dva cylindrické (rotační) klouby sestavené za pomoci krokového motorku a serva známých z předchozích příkladů. Schéma viz následující obrázek.



Díly pro tuto robotickou ruku jsou vytiskněny na 3D tiskárně a je možné si je stáhnout z WWW stránek. Jsou velmi jednoduché a jejich sestavení je patrné z přiložených obrázků. Předpokládám, že 3D tiskárnu již má většina škol k dispozici. V případě nouze, lze ramena sestavit např. z několika spojených kusů lepenkového papíru. S takovou rukou však řešte pouze první dvě úlohy. Pokud chcete řešit i závěrečnou úlohu této kapitoly – máchání čaje pomocí robotické ruky, pak je zakončení ruky nutné udělat z nějakého vodě (čaji) odolného materiálu anebo ruku testujte pouze „nasucho“ – bez vody.

Počet kloubů u robotické ruky u ní určuje takzvaný stupeň volnosti. Tato ruka má tedy dva stupně volnosti. Pokud byste někdy chtěli sestrojit robotickou ruku, tak aby dokázala to, co umí ruka lidská potřebujete 6 stupňů volnosti. Průmyslově vyráběné a používané robotické ruce – manipulátory mají obvykle 5 až 7 stupňů volnosti.

## JOYSTICK

Zvaný též pákový nebo křížový ovladač. Je vstupní zařízení, které se sestává z páky připevněné na základně, která se dokáže pohybovat ve směru x a y, přičemž je snímán úhel a velikost vychýlení. Obvykle je rovněž vybaven jedním nebo více tlačítka, u kterých snímáno stlačení.

**Joystick** se používá pro hraní videoher, ale rovněž pro ovládání letadel, automobilů, robotických rukou atd.

**Joystick** obsažený ve vaší stavebnici má jedno tlačítko na vrcholu „kloboučku“. Pro připojení potřebuje pět vodičů. Dva jsou napájení. Jeden slouží jako indikace stlačení tlačítka a připojuje se na digitální vstup. Je nastaven tak, že implicitně proud prochází a při sepnutí nikoliv. Další dva vodiče se připojují na analogové vstupy a poskytují informaci o pohybu joysticku ve směru os x a y jako číslo v intervalu 0 až 1023. 512 znamená, že joystick je uprostřed a obě hraniční čísla pak znamenají maximální vychýlení. V následujícím příkladu se s joystickem blíže seznámíme.

## POZNÁMKA

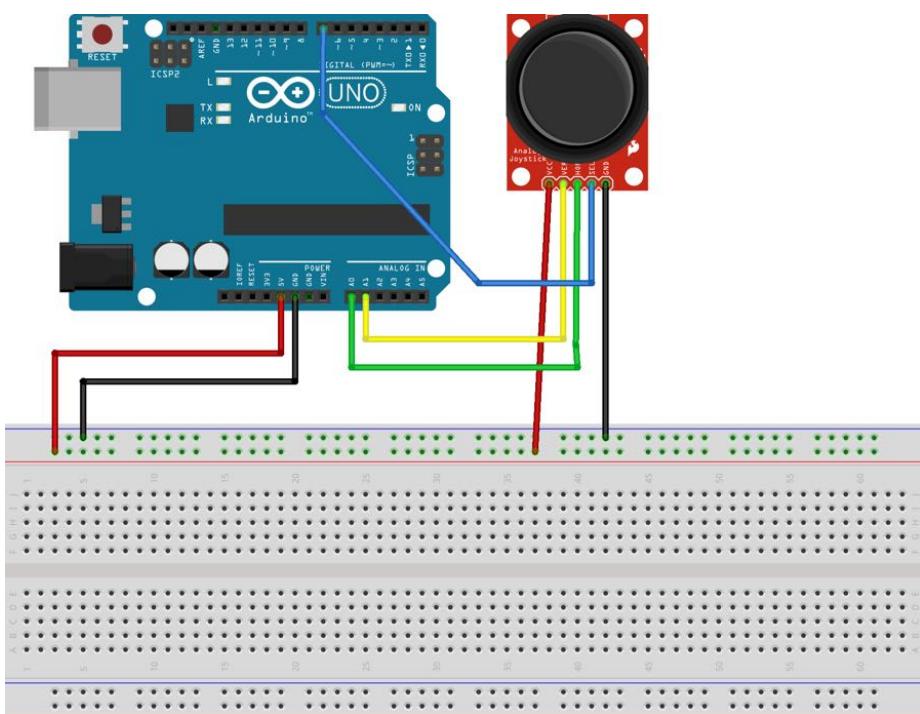
Následující úkoly na sebe navazují, a proto po vyřešení prvního úkolu není nutné obvod rozpojovat, ale naopak postupně k němu budete přidávat další součástky. Stejně tak i program je koncipován od jednoduššího ke složitějšímu.

## ÚKOL 1

Zapojte joystick a monitorujte jeho stavu pomocí sériové komunikace.

### ZAPOJENÍ OBVODU

Sestavte obvod dle následujícího obrázku:



## PROGRAMOVÝ KÓD

```
1 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
2 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
3 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
4 int x,y,z;
5
6 void setup() {
7     Serial.begin(9600);
8     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
9 joysticku
10    Serial.println("Test joysticku");
11 }
12
13 void loop() {
14     x=analogRead(JoyStick_X);
15     y=analogRead(JoyStick_Y);
16     z=digitalRead(JoyStick_Z);
17     Serial.print("X = ");
18     Serial.print(x);
19     Serial.print(", Y = ");
20     Serial.print(y);
21     Serial.print(", Z = ");
22     Serial.println(z);
23     delay(500);
24 }
```

①

②

③

- ① Nastavení připojení joysticku. Vodiče pro pohyb ve směru X a Y jsou připojeny na analogové piny na portech A0 a A1. Vodič pro osu Z (tlačítko joysticku) na pin 7 (digitální). Proměnné x, y a z jsou jejich instance v programu.
- ② Úvodní nastavení. Nastavení tlačítka joysticku a příprava sériového portu.
- ③ Hlavní program s cyklickým opakováním. Načtení hodnot z joysticku a jejich výpis na sériový port. Aby se výpis neopakoval příliš často, čeká se před následujícím načtením půl vteřiny

Program přeložte a nahrajte do Arduina. V programu Arduino si otevřete Sériový monitor. Pohybujte joystickem a sledujte, jak se při tom mění výpis.



#### → Nefunguje joystick

Zapojení v desce – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů desky Arduino.

Zapojení v joysticku – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů joysticku.

#### → Nejde nahrát kód do desky

USB kabel – ujistěte se, že máte desku Arduino připojenou k počítači.

Správný port – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

#### → Nefunguje Sériový monitor

Nezobrazuje se text – Zkontrolujte zapojení USB kabelu, zkontrolujte nastavení správné rychlosti – 9660 baudů.

Zobrazuje se nesmyslný text – Zkontrolujte nastavení správné rychlosti – 9660 baudů.



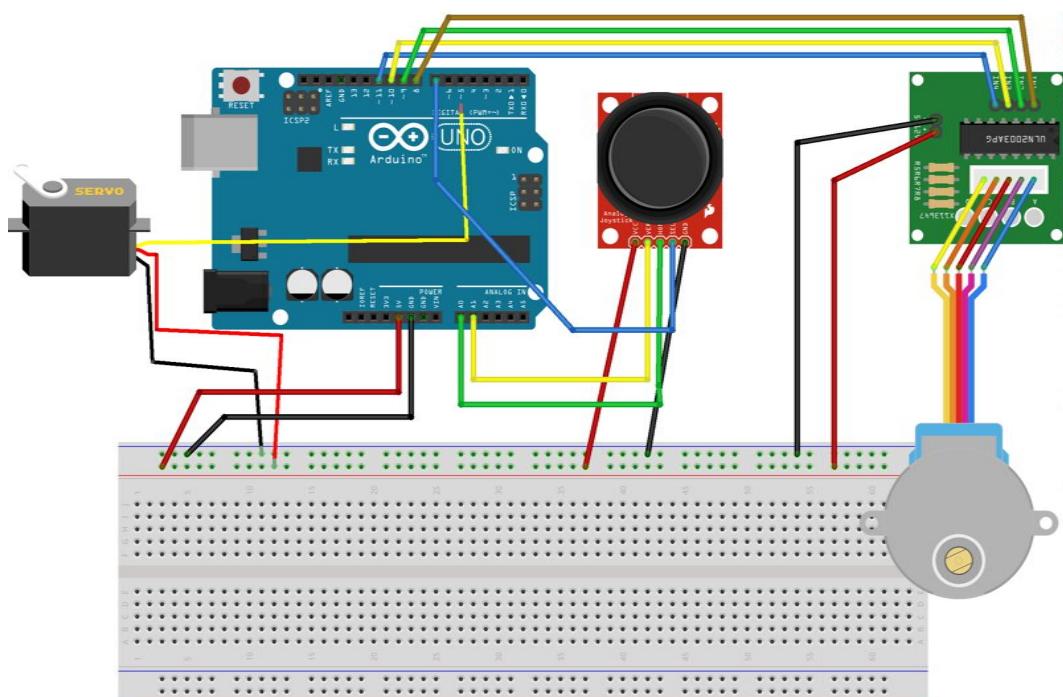
Zkuste úlohu řešit tak, že si místo sériového monitoru zapojíte LCD displej a pohyby joysticku si budete zobrazovat na něm. Můžete využít schéma zapojení a zdrojový kód z Úlohy 3 v této kapitole.

## ÚKOL 2

Sestrojte robotickou ruku ovládanou pomocí joysticku. Součástky na výrobu ruky vytiskněte na 3D tiskárně.

## ZAPOJENÍ OBVODU

Sestavte ze součástek robotickou ruku dle fotografie. Obvod zapojte dle následujícího schématu. K joysticku přibydou dva motorky. Krokový motor se zapojuje pomocí ovladače.



## PROGRAMOVÝ KÓD

```
1 #include <Servo.h>
2
3 Servo myservo;//Vytvoření objektu pro řízení krokového
4 motoru
5 int poloha; //Svislá poloha ruky
6
7 // Piny pro krokový motor
8 const int in1 = 8;
9 const int in2 = 9;
10 const int in3 = 10;
11 const int in4 = 11;
12 // proměnná pro nastavení rychlosti,
13 // se zvětšujícím se číslem se rychlosť zmenšuje
14 int rychlosť = 8;
15
16
17 //Joystick
18 int JoyStick_X = A0; //Xová osa - analogový pin 0
19 int JoyStick_Y = A1; //Yová osa - analogový pin 1
20 int JoyStick_Z = A7; //Tlačítko - pin 7
21 int x,y,z;
22
23
24
25 void setup() {
26     myservo.attach(6);//Servo motor je na pinu 6
27     myservo.write(0);//Ruka do výchozí polohy
28     poloha = 0; //Pamatuj si tuto polohu
29     // inicializace digitálních výstupů pro krokový motor
30     pinMode(in1, OUTPUT);
31     pinMode(in2, OUTPUT);
32     pinMode(in3, OUTPUT);
33     pinMode(in4, OUTPUT);
34     //inicjalizace Joysticku
35     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
36 joysticku
37 }
38
39 void loop() {
40     pohyb();
41     while(1);
42 }
43
44
45
46
```

The code is annotated with five callout numbers on the right side:

- ① Lines 3-5: Declaration of servo object and initial variable.
- ② Lines 12-14: Variable declaration for speed.
- ③ Lines 18-21: Variable declarations for joystick pins.
- ④ Lines 26-35: Setup function logic.
- ⑤ Lines 39-42: Loop function logic.

```

47 void rotacePoSmeru(int uhel) {
48     for(int i=0;i<(uhel*64/45);i++){
49         krok(1,0,0,0);
50         krok(1,1,0,0);
51         krok(0,1,0,0);
52         krok(0,1,1,0);
53         krok(0,0,1,0);
54         krok(0,0,1,1);
55         krok(0,0,0,1);
56         krok(1,0,0,1);
57     }
58     void rotaceProtismeru(int uhel) {
59         for(int i=0;i<(uhel*64/45);i++){
60             krok(1,0,0,1);
61             krok(0,0,0,1);
62             krok(0,0,1,1);
63             krok(0,0,1,0);
64             krok(0,1,1,0);
65             krok(0,1,0,0);
66             krok(1,1,0,0);
67             krok(1,0,0,0);
68     }
69
70     void krok(int a, int b, int c, int d){
71         digitalWrite(in1, a);
72         digitalWrite(in2, b);
73         digitalWrite(in3, c);
74         digitalWrite(in4, d);
75         delay(rychlost);
76     }
77
78     void pohyb(){
79         int x,y,z;
80         z=1;
81         while (z) {
82             x=analogRead(JoyStick_X);
83             y=analogRead(JoyStick_Y);
84             z=digitalRead(JoyStick_Z);
85             if (x>550) { //doprava
86                 rotacePoSmeru(5);
87             }
88             else if (x<480){ //doleva
89                 rotaceProtismeru(5);
90             }
91             else if (y<480){ //dolu
92                 if (poloha>=5) {
93                     poloha=poloha-5;

```

⑥

⑦

```

94         myservo.write(poloха);
95         delay(1000);
96     }
97 }
98 else if (y>550){ //nahoru
99     if (polоха<=170) {
100         polоха=polоха+5;
101         myservo.write(poloха);
102         delay(1000) ;
103     }
104 }
105 }
106 delay(100);
107 }

```

⑦

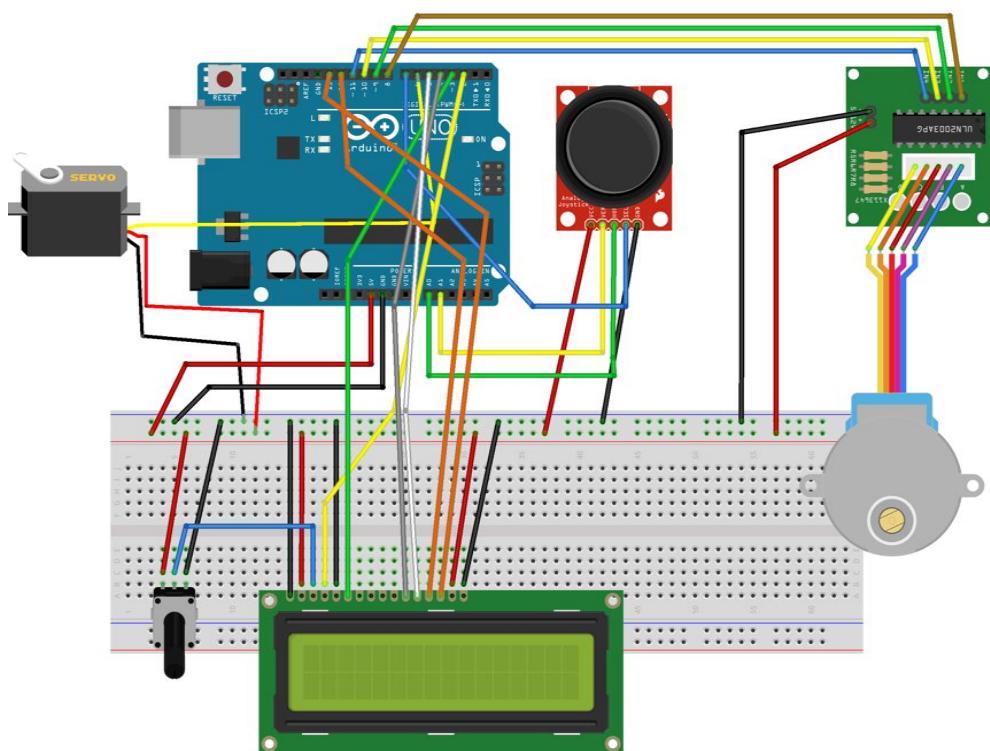
- ④ Nastavení servo motoru (svislý pohyb). Proměnná poloha slouží k zapamatování aktuální svislé polohy ruky.
- ⑤ Nastavení krokového motoru (vodorovný pohyb). Proměnná rychlosť slouží pro nastavení pauzy mezi jednotlivými pohyby motoru.
- ⑥ Nastavení pinů joysticku (viz předchozí příklad).
- ⑦ Inicializace potřebných proměnných a knihoven.
- ⑧ Hlavní program. Nedělá nic jiného, než spustí proceduru pohyb, která se stará o pohyb ruky, dle pohybu joysticku. Konstrukce while(1) je nekonečná smyčka, ve které skončí program po stisku tlačítka joysticku. Pro opětovné spuštění je nutné stisknout tlačítko reset na Arduinu.
- ⑨ Funkce řídící pohyb krokového motorku. Tyto funkce jsou opsané z manuálu výrobce motorku (dle manuálu k stavebnici, kterou máte k dispozici). Máte-li jiný motorek, než ten ze stavebnice, je možné (ale nepravděpodobné), že bude tyto funkce nutné přepsat dle manuálu vašeho motorku.
- ⑩ Hlavní funkce. Načítá pohyb joysticku do proměnných x a y a hlídá stisk tlačítka (proměnná z). Dle těchto hodnot pak řídí pohyb ruky pomocí obou motorků. Funkce končí stiskem tlačítka (v z je po stisku 0).

## ÚKOL 3

Sestrojte robotickou ruku pro máhání čaje, ovládanou pomocí joysticku. Tato ruka by měla umět připevněný čajový pytlík máhat předem danou dobu v hrnečku a pak jej přesunout mimo hrneček.

## ZAPOJENÍ OBVODU

Upravte zapojení z předchozího příkladu. Doplňte LCD panel, na který se budou zobrazovat informace a stavy ruky. Zkontrolujte opravdu pečlivě jeho zapojení vzhledem k velkému počtu vodičů.



## FOTOGRAFIE RUKY

Včetně dílů z 3D tiskárny



## PROGRAMOVÝ KÓD

Následující kód vychází z předchozího příkladu a popsány proto budou pouze odlišnosti.

```
1 #include <LiquidCrystal.h> ①
2
3 #include <Servo.h>
4
5 Servo myservo; //Vytvoření objektu pro řízení krokového
6 motoru
7
8 int poloha; //Svislá poloha ruky
9
10 // Piny pro krokový motor
11 const int in1 = 8;
12 const int in2 = 9;
13 const int in3 = 10;
14 const int in4 = 11;
15
16 // Proměnná pro nastavení rychlosti,
17 // se zvětšujícím se číslem se rychlosť zmenšuje
18 int rychlosť = 8;
19 int uhel1;
20 int x,y,z;
21 int i,j;
22
23 //Joystick
24 int JoyStick_X = A0; //Xová osa joysticku - analogový pin 0
25 int JoyStick_Y = A1; //Yová osa joysticku - analogový pin 1
26 int JoyStick_Z = A7; //Tlačítko joysticku - pin 7
27
28 // Piny pro připojení displeje
29 LiquidCrystal lcd(2, 3, 4, 5, 12, 13); ①
30
31 int minut; //Počet minut pro máchání čaje
32
33 void setup() {
34     myservo.attach(6); //Servo motor je na pinu 6
35     myservo.write(0); //Ruka do výchozí polohy
36     poloha = 0; //Pamatuj si tuto polohu
37     // inicializace digitálních výstupů pro krokový motor
38     pinMode(in1, OUTPUT);
39     pinMode(in2, OUTPUT);
40     pinMode(in3, OUTPUT);
41     pinMode(in4, OUTPUT);
42     //inicIALIZACE Joysticku
43 }
```

```

44     pinMode(JoyStick_Z, INPUT_PULLUP); //Nastavení tlačítka
45     joysticku
46     lcd.begin(16, 2); // Počet sloupců a řádek LCD displeje
47
48 }
49
50 void loop() {
51     //main program
52     lcd.clear();
53     lcd.print("Pouzij joystick");
54     lcd.setCursor(0,2);
55     lcd.print("pro nastavení");
56     pocatecni_nastaveni(); //Nastavení polohy ruky "nad hrnek"
57     lcd.clear(); //Nastavení polohy pro připevnění pytlíku
58     myservo.write(poloha+15);
59     delay(1000);
60     rotaceProtismeru(90);
61     delay(1000);
62     lcd.print("Ruka pripravena");
63     lcd.setCursor(0,2);
64     lcd.print("prievni caj a potvrd"); // Potvrdit stiskem
65                                     // joysticku
66     delay(1000);
67     z=1;
68     while (z) {
69         x=analogRead(JoyStick_X);
70         y=analogRead(JoyStick_Y);
71         z=digitalRead(JoyStick_Z);
72         delay(100);
73     }
74     delay(1000);
75     minut=pocetMinut(); //Nastavení počtu minut pro máchání
76     delay(1000);
77     rotacePoSmeru(90); //Najedeme nad čaj
78     delay(1000);
79     myservo.write(poloha-5); //Máchání čaje
80     for (i=minut;i;i--){
81         lcd.clear();
82         lcd.print("Zbyva:");
83         lcd.setCursor(0,1);
84         lcd.print(i);
85         lcd.print(" minut");
86         for (j=1;j<10;j++){
87             myservo.write(poloha-7);
88             delay(3000);
89             myservo.write(poloha+7);
90             delay(3000);

```

①

②

```

91     }
92   }
93   lcd.clear(); //Konec máchání
94   lcd.print("Hotovo");
95   myservo.write(poloха+20);
96   delay(10000);
97   rotacePoSmeru(90); //Odjezd doprava
98   myservo.write(15);
99   while(1) { } //Nekonečná smyčka
100 }
101
102 // zde následují funkce pro volání jednotlivých
103 // kroků pro otočení po či proti směru hodinových
104 // ručiček
105 void rotacePoSmeru(int uhel) {
106   for(int i=0;i<(uhel*64/45);i++){
107     krok(1,0,0,0);
108     krok(1,1,0,0);
109     krok(0,1,0,0);
110     krok(0,1,1,0);
111     krok(0,0,1,0);
112     krok(0,0,1,1);
113     krok(0,0,0,1);
114     krok(1,0,0,1);
115   }
116   void rotaceProtiSmeru(int uhel) {
117     for(int i=0;i<(uhel*64/45);i++){
118       krok(1,0,0,1);
119       krok(0,0,0,1);
120       krok(0,0,1,1);
121       krok(0,0,1,0);
122       krok(0,1,1,0);
123       krok(0,1,0,0);
124       krok(1,1,0,0);
125       krok(1,0,0,0);
126     }
127   }
128   // každý krok obsahuje výrobcem dané pořadí
129   // pro správné spínání motoru a následnou
130   // pauzu, kterou určujeme rychlosť otáčení
131   void krok(int a, int b, int c, int d){
132     digitalWrite(in1, a);
133     digitalWrite(in2, b);
134     digitalWrite(in3, c);
135     digitalWrite(in4, d);
136     delay(rychlosť);
137   }

```

②

```

138
139 int pocetMinut(){
140     int m=2;
141     int x, y, z;
142     z=1;
143     lcd.setCursor(0,0);
144     lcd.print("Maximum 9 minut");
145     lcd.setCursor(0,1);
146     lcd.print("Louhovat: ");
147     lcd.print(m);
148     lcd.print(" min");
149     while (z) {
150         x=analogRead(JoyStick_X);
151         y=analogRead(JoyStick_Y);
152         z=digitalRead(JoyStick_Z);
153         if (y>550){ //dolu
154             if (m) {
155                 m=m-1;}
156             lcd.setCursor(10,1);
157             lcd.print(m);
158         }
159         if (y<480){ //nahoru
160             if (m<9) {
161                 m=m+1;
162             }
163             lcd.setCursor(10,1);
164             lcd.print(m);
165         }
166         delay(300);
167     }
168     return m;
169 }
170
171 }
172
173 void pocatecni_nastaveni(){
174     int x,y,z;
175     z=1;
176     while (z) {
177         x=analogRead(JoyStick_X);
178         y=analogRead(JoyStick_Y);
179         z=digitalRead(JoyStick_Z);
180         if (x>550) { //doprava
181             rotacePoSmeru(5);
182         }
183         else if (x<480){ //doleva
184             rotaceProtiSmeru(5);

```

③

```

185 }
186 else if (y<480){ //dolu
187     if (poloha>=5) {
188         poloha=poloha-5;
189         myservo.write(poloha);
190         delay(1000);
191     }
192 }
193 else if (y>550){ //nahoru
194     if (poloha<=170) {
195         poloha=poloha+5;
196         myservo.write(poloha);
197         delay(1000);
198     }
199 }
200 }
201 delay(100);
}

```

- ① Nastavení LCD panelu včetně pinů, na které je zapojen.
- ② Vlastní program, tentokrát složitější. Postupně se prochází následující kroky:
  - a. Počáteční nastavení polohy ruky – nad hrnek.
  - b. Přesun ruky na pozici, kde se připevní pytlík čaje.
  - c. Nastavení doby pro máchání pytlíku.
  - d. Vlastní máchání pytlíku.
  - e. Posun ruky na místo, kde je možné pytlík sejmout.
  - f. Nekonečná smyčka na závěr.
- ③ Funkce pro nastavení počtu minut pro máchání čaje.

## PRÁCE S ROBOTICKOU RUKOU

- ① Po spuštění programu je třeba jako první věc nastavit robotickou ruku nad hrnek. Doporučuji nastavovat na prázdný hrnek. Ruku nastavte trochu napravo od středu hrnku a částečně jí ponořte pod horní okraj (cca. 1 cm). Potvrďte stiskem joysticku.
- ② Ruka si nyní najede vlevo od hrnku na pozici, na které můžete připevnit pytlík s čajem. Snažte se při tom nepohybovat s rukou. Pokud se vám to nepodaří, připevněte pytlík, stiskněte reset Arduinu a vraťte se na krok jedna. Alternativně začínejte rovnou s připevněným pytlíkem.

- ③ Připravte si talířek nebo nějakou podložku, nad kterou ruka donese vylouhovaný pytlík. Umístěte jí na pozici asi 90° napravo od hrnku.
- ④ Nastavte počet minut, po které se má čaj louhat, nalejte do hrnku vodu s požadovanou teplotou a potvrďte joystickem.
- ⑤ Robotická ruka provede vymáhání čaje a po nastavené době odnese pytlík napravo od hrnku nad připravenou podložku.
- ⑥ Tímto celý cyklus končí, pro jeho opakování je nutné stisknout reset na Arduinu.

## POZNÁMKY

Věnujte opravdovou pečlivost zapojení obvodu. Zejména zapojení LCD panelu je poměrně složité díky velkému počtu vodičů.

Vzhledem k tomu, že se zde pracuje s kapalinou (čajem) dávejte pozor, aby nedošlo k namočení Arduina, motorků nebo jiných elektronických součástek. Pokud k tomuto přece jen dojde, okamžitě odpojte Arduino od zdroje elektrického proudu a nechte vše dobře vyschnout, nejlépe do příštího týdne.

Můžete se setkat s nepřesnostmi v pohybu ruky, které jsou způsobeny tím, že ruka zavadí o nějaký předmět (nejčastěji hrnek). Pokud si tohoto všimnete v počátečních fázích, raději systém resetujte a začněte od začátku.

## ZÁVĚR

V tomto příkladu jste se naučili zapojit obvod sestávající z dvou motorků, displeje a joysticku. Dozvěděli jste se něco málo o robotice a vyzkoušeli si ovládání jednoduché robotické ruky. Je třeba si uvědomit, že robotické ruce v praxi mívají nejméně pět stupňů volnosti – pět motorků a jejich ovládání a programování je podstatně složitější.

## 9. VZDÁLENÉ OVLÁDÁNÍ POMOCÍ IR OVLADAČE

LEKCE JE ZAMĚŘENA NA SEZNÁMENÍ S PRINCIPY DÁLKOVÉHO OVLÁDÁNÍ POMOCÍ IR (INFRARED – INFRAČERVENÉHO) ZÁŘENÍ. JE SESTAVEN OBVOD PRO VZDÁLENÉ ŘÍZENÍ JEDNODUCHÉHO VOZÍTKA.

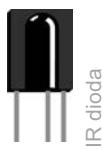
### CÍLE

- ① Co je to IR ovládání.
- ② Zapojení IR diody pro testování IR ovládání.
- ③ Konstrukce obvodu pro vzdálené ovládání dvou motorků pomocí IR protokolu.
- ④ Naprogramování kódu pro obsluhu obvodu z předchozího bodu.
- ⑤ Volitelně: využití obvodu k praktické konstrukci.

Čas: **2 (3) x 45 min**

Úroveň: ■ ■ ■ ■ ■

Vychází z: **5, 6**



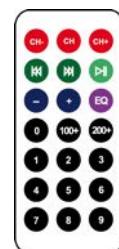
DC motor



L9110H



Servo motor



Dálkový ovladač

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I



Studenti sestaví základní obvod, ve kterém připojí k Arduinu IR diodu a budou v sériovém monitoru sledovat jaké kódy jsou přijaté při stisku různých tlačítek dálkového ovladače.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 1x IR dioda, dálkový IR ovladač. Studenti si mohou přinést z domova libovolný IR ovladač. **Nutno říci předem.**
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 9.
- ⑤ Pracovní listy pro studenty.

## 1. KROK ⏰ 15 minut

Na úvod rozdejte studentům sady Arduino. Pohovořte o různých typech dálkového ovládání. Popište si princip IR ovládání. Kde všude se IR používá.

Studenti si prohlédnou IR přijímač a IR ovladač. Pokud mají svůj z domova, je pravá chvíle, aby si jej připravili.

## 2. KROK ⏰ 5 minut

Studenti si sestaví jednoduchý obvod s IR přijímačem a připraví dálkový ovladač. Pokud v IR ovladači je dosud vložena folie u baterie, odstraňte jí.

## **ZEPEJTE SE STUDENTŮ**

### **→ Víte, co to je to infračervené světlo?**

Infračervené světlo – Infrared (IR) – jedná se o elektromagnetické záření s vlnovou délkou, která je větší než viditelné světlo a menší než mikrovlnné záření. Jedná se o záření lidským okem neviditelné.

### **→ Co je to IrDA?**

IrDA (Infrared Data Association) – komunikační infračervený port popisující bezdrátovou komunikaci mezi infračervenou LED diodou a fotodiodou. Pro komunikaci je nutná přímá viditelnost.

### **→ Kde se můžete potkat s infračerveným ovládáním?**

Televize, přehrávače a jiná audio video technika. Dálkově řízené hračky. Dálkové ovladače u herních konzolí.

### **→ Čím je dnes IrDA nahrazován?**

Většinou bluetooth technologií. Má větší dosah a není nutná přímá viditelnost.



## **3. KROK** 15 minut

Je-li třeba, je nutné provést následující krok:

Je nutné smazat adresář **RobotIRremote** v adresáři **Arduino IDE**. Najdete jej podle programu Arduino IDE – menu Soubor / Nastavení. Např.:

**C:\Program Files (x86)\Arduino\libraries**

a

**C:\Users\<Uživatel>\Dokumenty\Arduino\libraries**

Vymazání adresáře je nutné z důvodu použití jiné knihovny pro IR senzor.

Studenti si spustí Arduino IDE a napíší základní program.

Studentům vysvětlete programový kód, zejména pak základní strukturu programu a použité funkce pro zápis hodnot na pinu desky.

## 4. KROK 15 minut

Studenti si v Arduino IDE spustí sériový monitor namíří IR ovladač na přijímač a stisknou tlačítko. Nechte je experimentovat se stiskem různých tlačítek, případně s různými ovladači.

### ÚKOLY PRO STUDENTY

- Vysílají stejné typy ovladačů stejné signály?
- Jak je to s různými ovladači?
- Poznamenejte si kódy pro tlačítka, která chcete použít pro ovládání motorků v příští hodině. Potřebujete čtyři kódy (dvě strany dvou motorů) – např. funkce vpřed, vzad, vlevo a vpravo.



# PRACOVNÍ LIST I – IR DIODA

PRVNÍ SEZNÁMENÍ S DÁLKOVÝM OVLÁDÁNÍM ARDUINA POMOCÍ IR DIODY. V TÉTO ČÁSTI SE SEZNÁMÍTE S PRINCIPEM IR DIODY, JEJÍM ZAPOJENÍM A FUNKČNOSTÍ.

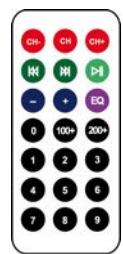
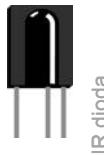
## CO SE NAUČÍTE

- ① Princip IR ovládání.
- ② Zapojení IR diody.
- ③ Naprogramování prvního programu pro ovládání IR diody.



## CO BUDETE POTŘEBOVAT

- ① IR diodu.
- ② Dálkový ovladač.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



Dálkový ovladač

---

### POUŽITÉ SOUČÁSTKY

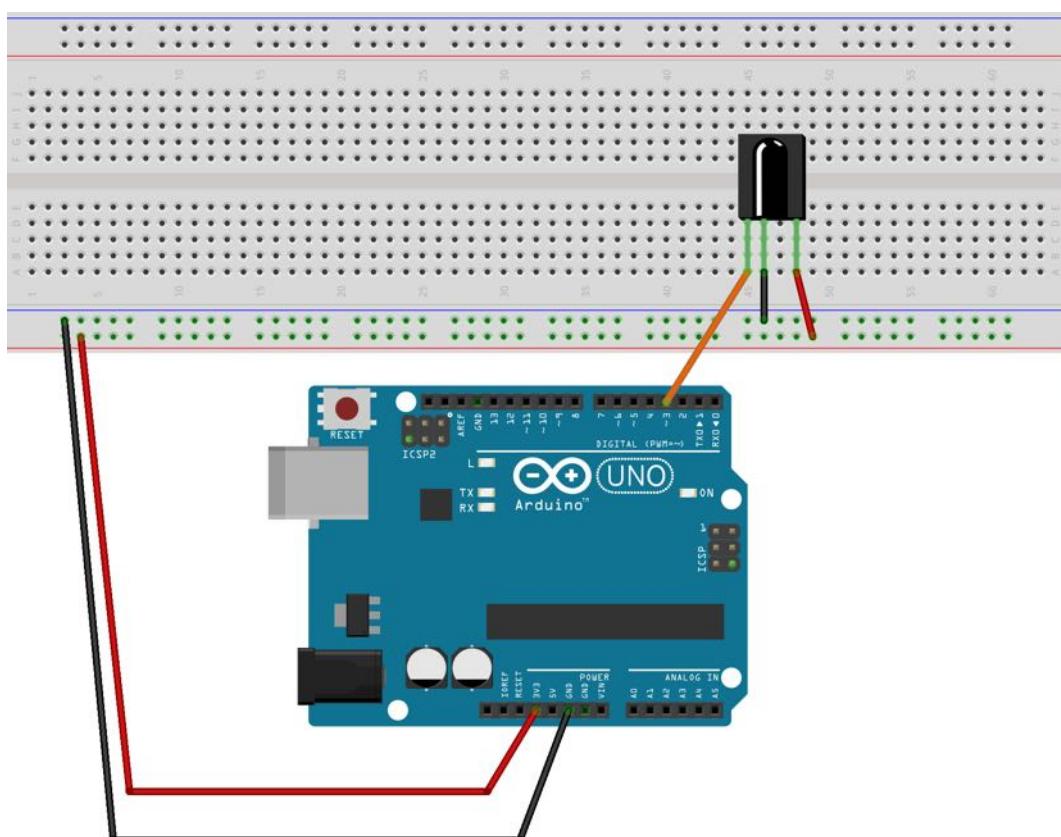
## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.



## DEJTE SI POZOR

- ➔ Dejte si pozor na to, jak zapojujete IR diodu. Díváte-li se proti diodě, pak vlevo je datový vodič, uprostřed zem (GND), vpravo 5 V.
  - ➔ Stavíte-li obvod pouze pro tuto hodinu, můžete vodiče připojit přímo k Arduinu. Zapojení s nepájivým polem, je i tak lepší, IR dioda je díky němu v pevné pozici.



- ② Spusťte program Arduino IDE a napište následující programový kód.

```
1 #include <IRremote.h>
2
3 int RECV_PIN = 3; // IR Dioda na pinu 3
4 IRrecv irrecv(RECV_PIN);
5 decode_results results;
6 String Vstup;
7
8 void setup() {
9     Serial.begin(9600);
10    irrecv.enableIRIn();
11 }
12
13 void loop() {
14     if (irrecv.decode(&results)) { //Dekóduj načtené
15         Vstup = String(results.value, HEX);
16         Serial.println(Vstup); //A zobraz na sériový
17     monitor
18     irrecv.resume(); //Načti další hodnotu
19 }
20 }
```

- ③ V programu Arduino IDE je nutné vymazat jednu z knihoven pro práci s IR. Zeptejte se učitele, zda je to na vašem počítači již hotové a pokud ne proveděte bod 4, jinak přejděte na bod 5.
- ④ Je nutné smazat adresář **RobotIRremote** v adresáři **Arduino IDE**. Najdete jej podle programu Arduino IDE – menu Soubor / Nastavení. Např.:  
C:\Program Files (x86)\Arduion\libraries  
a  
C:\Users<Uživatel>\Dokumenty\Arduino\libraries  
Vypněte a zapněte **Arduino IDE** (nejprve uložte svou práci).
- ⑤ V programu Arduino IDE nastavte odpovídající desku. V menu **Tools > Board > Arduino UNO**.
- ⑥ Dále nastavte port (v Menu **Tools > Seriál Ports > vyberte odpovídající port**).
- ⑦ Pro nahrání programu do desky Arduino, klikněte na ikonu 
- ⑧ Otevřete si v **Arduino IDE Sériový monitor**, kliknutím na ikonu 
- ⑨ Stiskněte dálkový ovladač a sledujte v sériovém monitoru, co se stane po stisku jednotlivých tlačítek.

Pokud vše funguje a vidíte zobrazené kódy tlačítek, můžete se věnovat samostatným úkolům. Všimněte si, že pokud tlačítka podržíte delší dobu, zobrazují se kódy „**FFFFFF**“, které nemají žádný další informační význam, kromě toho že uživatel stále drží předchozí stisknuté tlačítko.

### ÚKOLY PRO VÁS

- ➔ A) Zapište si kódy tlačítek, které hodláte použít v následující hodině pro ovládání dvou motorků. Potřebujete minimálně čtyři tlačítka (pro každý motor dvě – dva směry otáčení). Např. ve významu vpřed, vzad, vlevo, vpravo.
- ➔ B) Máte-li možnost vyzkoušejte si i jiný IR ovladač.
- ➔ C) Vyzkoušejte si, na jakou vzdálenost a přes jaké překážky IR ovladač funguje.



### DEJTE SI POZOR

- ➔ Jednotlivé ovladače nejsou „spárovány“ s konkrétním Arduinem. Stisk tlačítka na jednom ovladači mohou zaznamenat i IR diody vašich spolužáků a spolužáček, a naopak vaše IR dioda může zachytávat cizí dálkové ovladače.



# PRŮVODCE HODINOU II



Studenti naváží na minulou hodinu a doplní svůj obvod o servo motor a DC motor s ovladačem (můstkem) L9110H. Vyzkouší si vzdálené ovládání obou motorků pomocí IR diody a IR ovladače.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, vodiče typu zástrčka-zástrčka , IR diodu, IR ovladač, Servo, obvod L9110H (ovladač motoru) a DC motor.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 9, která je ke stažení na ...
- ⑤ Pracovní listy pro studenty (ke stažení na ...).

## 1. KROK 🕒 10 minut

Na úvod rozdejte studentům sady Arduino. Řekněte, že v této hodině naváží na předchozí příklad, který se týkal zapojení a ovládání obvodu s IR diodou. Naučí se vzdáleně ovládat DC a servo motor, které znají z předchozích příkladů.

Připomeňte si princip a vlastnosti IR ovládání.

## 2. KROK 🕒 10 minut

Sestavte obvod dle schématu. Schéma jim promítнete na projektoru a naleznou jej i na pracovních listech. DC motorky mohou studenti připojit např. tak, že provlíknete očky kontaktů ohnuté konce vodičů a připíchnete je k nepájivému poli. Ukažte jim to.

### 3. KROK 10 minut

Studenti si zapíší kód do **Arduina IDE** a nahrají si jej do **Arduina**. Kód studentům vhodným způsobem připravte, ať jej nemusí opisovat, ale promítnete jej na projektoru a vysvětlete jej. Upozorněte studenty ať si případně upraví kódy tlačítek dle svých zápisů z minulé hodiny. V ukázkovém kódu jsou použita tlačítka 2, 4, 6 a 8. Chtějí-li použít jiné, musí přepsat správné kódy (např. **Vstrup=="ff18e7"** pro tlačítko 2).

### 4. KROK 15 minut

Pokud studenti vše zvládli, mohou nyní zkoušet vzdálené ovládání obou motorků. Zbyde-li čas nechte je řešit následující úkoly.

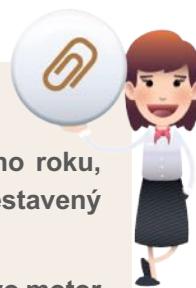
#### ÚKOLY PRO STUDENTY



- A) Co se stane, když zakomentujete pauzy (Delay(500))
- B) Experimentujte s úhly, o které se otáčí servo.
- C) Experimentujte s rychlosí DC motoru.

#### MOŽNÝ NÁPAD

- Máte-li ještě volné vyučovací hodiny do konce pololetí či školního roku, můžete věnovat jednu nebo dvě hodiny tomu, že studenti využijí sestavený obvod ke konstrukci nějakého zařízení. Nabízí se:
  - Dálkově ovládané vozítko – DC motor pohání kola a servo motor zatáčí s druhou nápravou
  - Dálkově ovládané vozítko-vznášedlo – DC motor pohání vrtuli, servo zatáčí.
  - Dálkově ovládaný ventilátor – DC motor pohání vrtuli, servo motor s ní otáčí.
- Pro konstrukci využijte kartóny, krabice, PET láhve, víčka od PET lahví, stará CD na kola atd. Máte-li k dispozici 3D tiskárnu, můžete využít i jí.



# PRACOVNÍ LIST II – IR DIODA POUŽITÍ PRO DÁLKOVÉ OVLÁDÁNÍ

POKRAČOVÁNÍ V SEZNAMOVÁNÍ SE S IR DIODOU A DÁLKOVÝM IR OVLÁDÁNÍM.  
TENTOKRÁT BUDEME POMOCÍ DÁLKOVÉHO OVLADAČE A IR DIODY OVLÁDAT DVA  
MOTORKY – DC MOTOR A SERVO.

## CO SE NAUČÍTE

- ① Zopakujete si, zapojení IR diody a její použití.
- ② Zopakujete si zapojení DC motoru a serva.
- ③ Vytvoření programu pro vzdálené ovládání DC motoru a serva pomocí IR.
- ④ Vytvořené zapojení si otestujete.

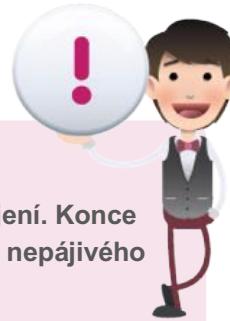


## CO BUDETE POTŘEBOVAT

- ① deska Arduino s USB kabelem.
- ② kontaktní pole.
- ③ vodiče typu zástrčka-zástrčka.
- ④ IR diodu.
- ⑤ IR ovladač.
- ⑥ Servo.
- ⑦ obvod L9110H (ovladač motoru).
- ⑧ DC motor.

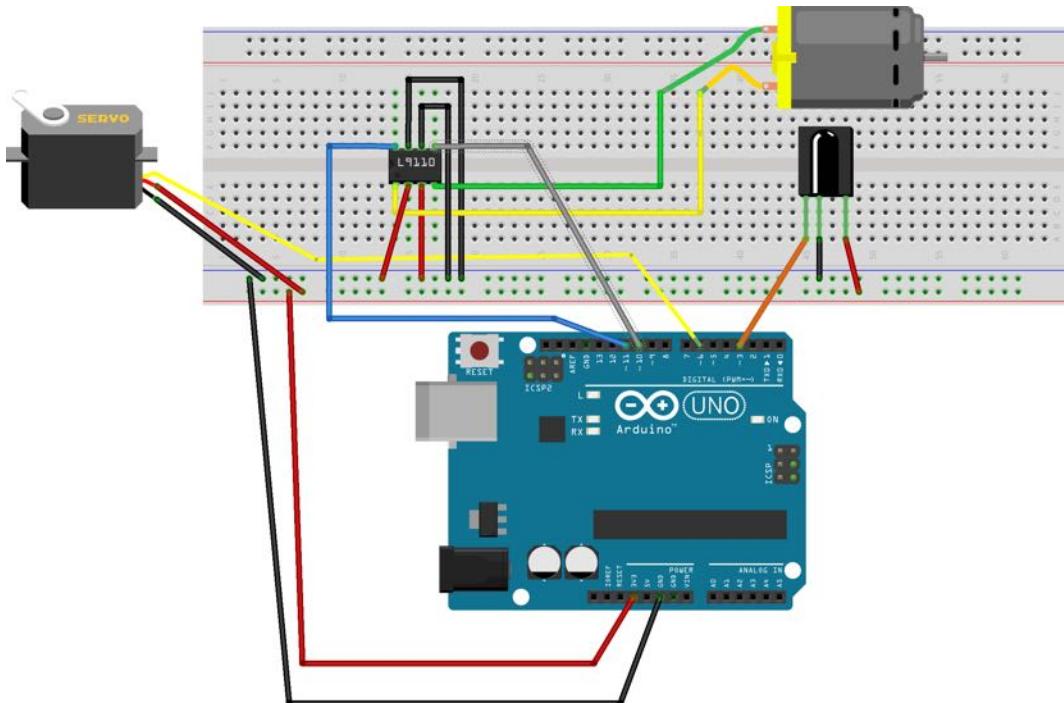
## A JDĚTE NA TO ...

① Podle schématu zapojte elektronický obvod.



## DEJTE SI POZOR

- Zapojení DC motoru. Můžete provizorně udělat následující zapojení. Konce vodičů protáhněte očky u DC motoru, ohnete je a zapíchnete do nepájivého pole.
  - Zapojení serva. Vodiče pro připojení na servo motoru mají následující význam – červený drát 5 V, hnědý drát GND, žlutý drát data.



② Spusťte program Arduino IDE a napište následující programový kód.

```

1 #include <IRremote.h>
2 #include <Servo.h>
3
4 int RECV_PIN = 8;           // IR Dioda na pinu 8
5 IRrecv irrecv(RECV_PIN);
6 decode_results results;
7 String Vstup;
8
9 Servo myservo;           //Vytvoření objektu pro řízení
10 krokového motoru
11 int poloha=90;
12
13 const int motorIn1 = 10;    //Piny pro DC motor na 10 a 11
14 const int motorIn2 = 11;
15 int rychlost = 255;        //Rychlost DC motoru
16
17 void setup() {
18     irrecv.enableIRIn();
19     pinMode(motorIn1,OUTPUT); //Inicializace DC motoru
20     pinMode(motorIn2,OUTPUT);
21     myservo.attach(3);      //Servo motor je na pinu 3
22     myservo.write(poloha); //Výchozí poloha
23 }
24
25 void loop() {
26     if (irrecv.decode(&results)) {
27         Vstup = String(results.value, HEX);
28         if (Vstup=="ff18e7") {
29             motor(rychlost,0); //DC motor směr vpřed
30             delay(500);
31         }
32         else if (Vstup=="ff4ab5") {
33             motor(0,rychlost); //DC motor směr vzad
34             delay(500);
35         }
36         else if (Vstup=="ff10ef") {
37             if (poloha>10){ //Hodnota 0 a menší není dobrá
38                 poloha=poloha-10;
39                 myservo.write(poloha); //Servo o 10 stupňů vlevo
40             }
41             delay(500);
42         }
43         else if (Vstup=="ff5aa5") {
44             if (poloha<170){ //Hodnota 180 stupňů a vyšší není dobrá
45                 poloha=poloha+10;
46                 myservo.write(poloha); //Servo o 10 stupňů vpravo
47             }

```

```

48     delay(500);
49 }
50 else{
51     motor(0,0);           //Zastav DC motor
52     delay(500);
53 }
54 irrecv.resume();          //Načti další hodnotu
55 }
56 }
57
58 void motor(int A, int B)    //Procedura pro obsluhu DC motoru
59 {
60     analogWrite(motorIn1,A);
61     analogWrite(motorIn2,B);
62 }
```

- ③ Po napsání programu připojte USB kabel k desce a k počítači.
- ④ V programu Arduino IDE nastavte odpovídající desku. V menu **Tools > Board > Arduino UNO**.
- ⑤ Dále nastavte port (v Menu **Tools > Seriál Ports > vyberte odpovídající port**).
- ⑥ Pro nahrání programu do desky Arduino, klikněte na ikonu 
- ⑦ Nyní vezměte váš dálkový ovladač a vyzkoušejte program. Dejte si opět pozor na vzájemné ovlivňování s ostatními.

Pokud vše funguje, tak výborně. Můžete se vrhnout na samostatné úkoly.

### ÚKOLY VÁS



- A) Upravte program tak, aby se servo otácelo o jiný úhel. Vytvořte si pro tyto účely novou proměnnou.
- B) Experimentujte s rychlosí DC motoru.



### VYSVĚTLENÍ

- Možná si všimnete, že na jakékoliv jiné tlačítko, než jsou čtyři zvolená se zastavuje DC motor. Je to proto, že v případě, že pokud je DC motor v činnosti vrací IR dioda někdy zcela jiný kód, než by měla.

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU A ŘEŠENÍ ÚKOLŮ A MOŽNÝCH PROBLÉMŮ PŘI NEFUNKČNOSTI OBVODŮ.

## OBSAH PRŮVODCE

- ① Princip IR ovládání.
- ② Podrobný popis zapojení obvodu s IR diodou.
- ③ Zdrojový kód programu pro IR ovládání.
- ④ Podrobný popis zapojení obvodu pro IR ovládání DC a servo motoru.
- ⑤ Zdrojový kód tohot zapojení.
- ⑥ Řešení možných potíží.
- ⑦ Další úkoly pro samostatnou práci.

## PRINCIP IR OVLÁDÁNÍ

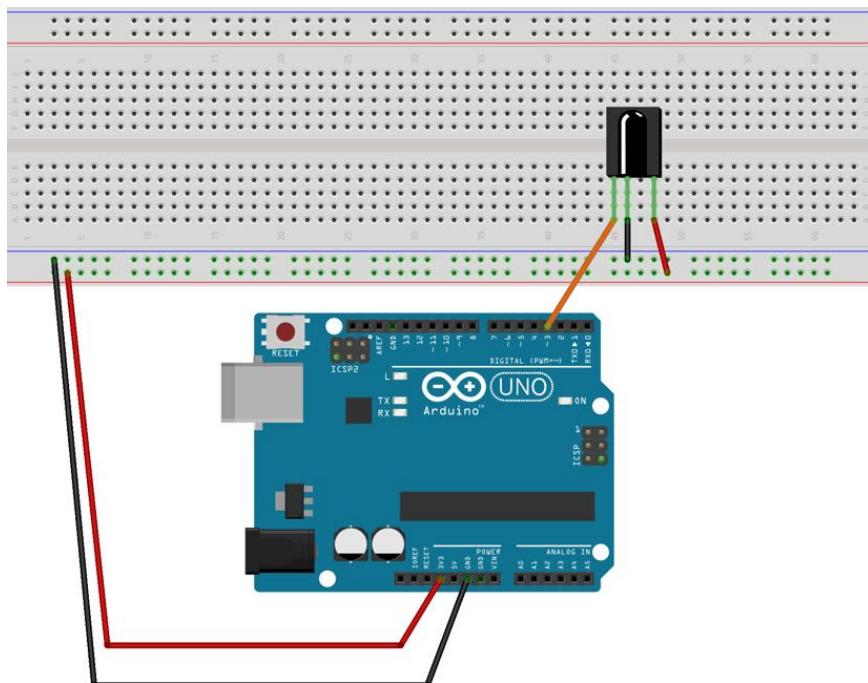
Infračervené ovládání (Infra Red – IR) je ovládání založené na šíření světla v infračerveném pásmu. Toto světlo je lidským okem neviditelné, ale pro jeho šíření platí stejná pravidla jako pro běžné světlo. V cestě signálu tedy nesmí stát žádná překážka. Signál je generován i přijímán IR diodou. Vysílač není přímo spárován s přijímačem jako u jiných typů dálkového ovládání a přenos lze tedy snadno ovlivnit jiný vysílačem.

Výhodou tohoto ovládání je jeho jednoduchost, jak uvidíme v této lekci, nevýhodou v podstatě vše ostatní. Pokud nám jde o bezpečnost, dosah a ovládání i přes překážky je lepší použít bluetooth nebo rádio (RF) ovládání.



Než si ukážeme konkrétní aplikaci programového kódu, vytvoříme elektronický obvod, ve kterém si pouze vyzkoušíme princip IR ovládání.

## ZAPOJENÍ OBVODU S IR DIODOU



Obr. 1 - Zapojení IR diody

Zapojení obvodu je velmi jednoduché. Na digitální pin **3** je připojen datový vodič IR diody. Běžným způsobem pak jsou připojeny další dva piny 3.3V a GND. Je možné zapojení bez nepajivého pole, ale toto zapojení má tu výhodu, že je IR dioda pevně umístěna.

## PROGRAMOVÝ KÓD

Pro správnou funkci je nutné smazat adresář **RobotIRremote** v adresáři **Arduino IDE**. Najdete jej podle programu Arduino IDE – menu Soubor / Nastavení. Např.:

```
C:\Program Files (x86)\Arduion\libraries  
a  
C:\Users<Uzivatel>\Dokumenty\Arduino\libraries
```

```
1 #include <IRremote.h>                                | ①  
2  
3 int RECV_PIN = 3;                                     | ②  
4 IRrecv irrecv(RECV_PIN);  
5 decode_results results;  
6 String Vstup;  
7  
8 void setup() {                                         | ③  
9     Serial.begin(9600);  
10    irrecv.enableIRIn();  
11 }  
12  
13 void loop() {                                         | ④  
14     if (irrecv.decode(&results)) {  
15         Vstup = String(results.value, HEX);  
16         Serial.println(Vstup);  
17         irrecv.resume();  
18     }  
19 }
```

- ① Zavedení knihovny pro práci s IR diodou.
- ② Nastavení proměnných. IR dioda je připojena na digitální PIN 3. Proměnná Vstup slouží k načtení hodnot z IR ovladače.
- ③ Ve funkci setup je inicializován IR vstup a současně i sériový port pro výstup na sériový monitor.
- ④ Ve funkci loop se cyklicky načítají přenesené hodnoty z IR ovladače a zobrazují na sériový monitor.



### NELZE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Chyba v kódu** – zkontrolujte, jestli je programový kód opravdu správně napsán.

Pokud bude existovat syntaktická chyba, kód se do desky nenahraje.

**Správná deska** – přesvědčte se, že máte správně zvolenou desku v nabídkách

**Tools>Board.**

### OVLADAČ NEFUNGUJE

**Zapojení IR diody** – zkontrolujte, zda je IR dioda správně zapojená, dle obrázku

1

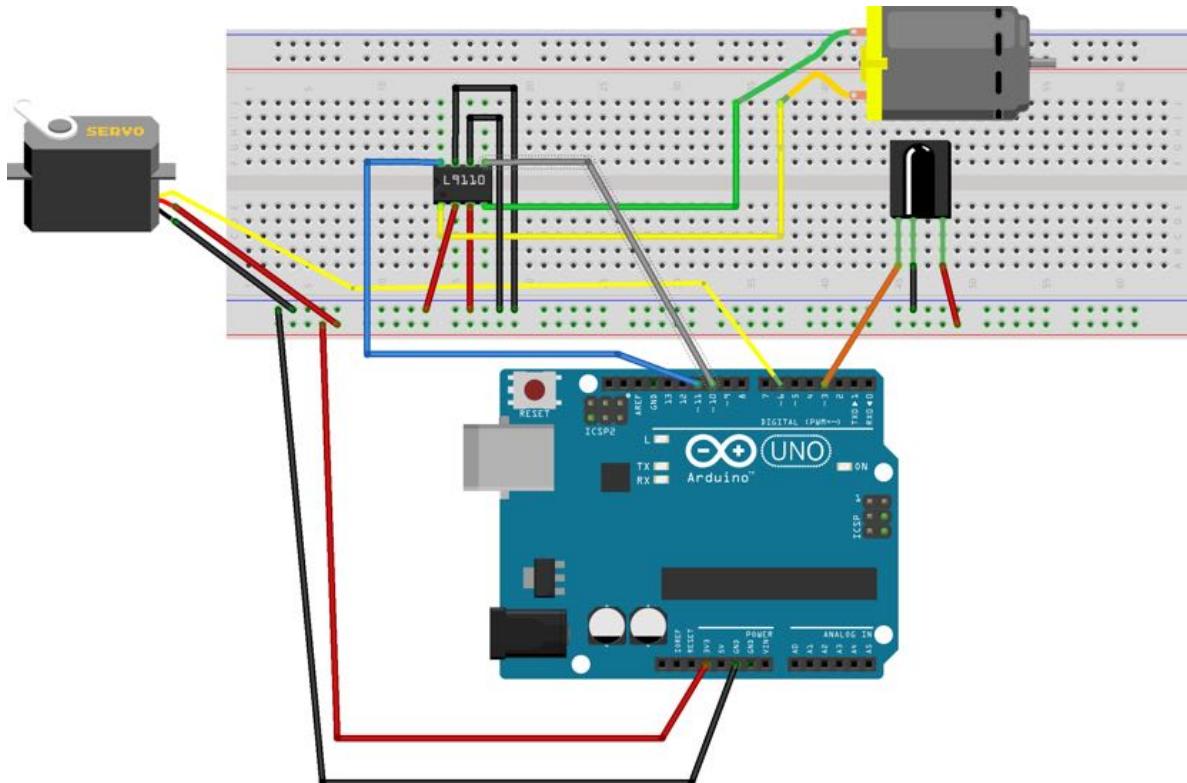
**Ovladač** – zkontrolujte, zda jsou v ovladači vložené baterie a zde je případně zapnutý.

### ÚKOLY PRO SAMOSTATNOU PRÁCI

- ➔ (Př. 1) Zapište si kódy ovladače, které budete potřebovat v dalším cvičení.  
Potřebujete nejméně čtyři tlačítka pro ovládání dvou motorků na obě strany.
- ➔ (Př. 2) Vyzkoušejte jiný ovladač.
- ➔ (Př. 3) Otestujte si vzájemné rušení (ovladač funguje na více Arduin).
- ➔ (Př. 4) Vyzkoušejte dosah ovladače.



## ZAPOJENÍ OBVODU S IR OVLÁDÁNÍM DVOU MOTORKŮ



Obr. 2 - Zapojení IR diody, DC motorku a serva

Jak si můžete všimnout, přibyly zde oproti předchozímu případu servo, DC motor a ovladač DC motoru (motor driver).

Vodiče můžete k DC motoru buď připájet anebo je protáhnout očky motoru zahnout a zapíchnout do podložky. Zvolíte-li druhou možnost, je třeba počítat s jistou nestabilitou zapojení, ale pro testování to plně postačí.

# PROGRAMOVÝ KÓD

```
1 #include <IRremote.h>
2 #include <Servo.h>
3
4 int RECV_PIN = 3; // IR Dioda na pinu 3
5 IRrecv irrecv(RECV_PIN);
6 decode_results results;
7 String Vstup;
8
9 Servo myservo;//Vytvoření objektu pro řízení
10 krokového motoru
11 int poloha=90;
12
13 const int motorIn1 = 10; //Piny pro DC motor na
14 10 a 11
15 const int motorIn2 = 11;
16 int rychlost = 255; //Rychlosť DC motoru
17
18 void setup() {
19     irrecv.enableIRIn();
20     pinMode(motorIn1,OUTPUT);      //Inicializace
21 DC motoru
22     pinMode(motorIn2,OUTPUT);
23     myservo.attach(8); //Servo motor je na pinu 8
24     myservo.write(poloha); //Výchozí poloha
25 }
26
27 void loop() {
28     if (irrecv.decode(&results)) {
29         Vstup = String(results.value, HEX);
30         if (Vstup=="ff18e7") { //DC motor 1. smer
31             motor(rychlost,0);
32             delay(500);
33         }
34         else if (Vstup=="ff4ab5"){ //DC motor 2. smer
35             motor(0,rychlost);
36             delay(500);
37         }
38         else if (Vstup=="ff10ef"){ //Servo 1. smer
39             if (poloha>10) {
40                 poloha=poloha-10;
41                 myservo.write(poloha);
42             }
43             delay(500);
44         }
45         else if (Vstup=="ff5aa5") { //Servo 2. smer
```

```

46     if (poloha<170){
47         poloha=poloha+10;
48         myservo.write(poloha);
49     }
50     delay(500);
51 }
52 else{
53     motor(0,0);
54     delay(500);
55 }
56 irrecv.resume(); //Načti další hodnotu
57 }
58 }
59
60 void motor(int A, int B) //Procedura pro ovládání
61 DC motoru
62 {
63     analogWrite(motorIn1,A);
64     analogWrite(motorIn2,B);
65 }
```

⑥

⑦

- ① Zavedení knihoven pro práci s IR diodou a servo motorem.
- ② Nastavení proměnných. IR dioda je připojena na digitální PIN 3. Proměnná Vstup slouží k načtení hodnot z IR ovladače.
- ③ Nastavení servo motoru. Proměnná poloha ukazuje polohu serva. Na začátku je nastavena na 90 stupňů, tedy na prostřední polohu servo motoru.
- ④ Narstavení DC motoru. Jsou použity digitální piny 10 a 11. Rychlosť motoru je nastavena na 255 – maximum.
- ⑤ Ve funkci setup je inicializován IR vstup, DC motor i servo na portu 8.
- ⑥ Ve funkci loop se cyklicky načítají přenesené hodnoty z IR ovladače a pokud se jedná o známé hodnoty, je provedena odpovídající akce.
- ⑦ Procedura pro ovládání DC motoru.



## NELZE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Chyba v kódu** – zkонтrolujte, jestli je programový kód opravdu správně napsán.

Pokud bude existovat syntaktická chyba, kód se do desky nenahraje.

**Správná deska** – přesvědčte se, že máte správně zvolenou desku v nabídkách

**Tools>Board.**

## OVLADAČ NEFUNGUJE

**Zapojení IR diody** – zkонтrolujte, zda je IR dioda správně zapojená, dle obrázku

1

**Ovladač** – zkонтrolujte, zda jsou v ovladači vložené baterie a zde je případně zapnutý.

## ZNÁMÉ PROBLÉMY

**Ovladač posílá neobvyklé kódy** – Pokud běží DC motor, přijímá IR dioda někdy **neobvyklé kódy o větší délce**. Kvůli této vlastnosti, nelze použít tlačítko pro STOP DC motoru, protože nelze určit spolehlivě kód, který bude při stisku konkrétního tlačítka v tomto případě přijat.

**Problémy serva v krajních polohách** – Něktetřá serva mají problémy v krajních polohách svého rozsahu zhruba od 0 do 5 a od 176 do 180 stupňů. Servo v těchto případech nedokáže zaujmout správnou polohu. Pokud zaznamenáte takovéto chování, upravte program tak, aby servo nemohlo těchto poloh dosáhnout.

## ÚKOLY PRO SAMOSTATNOU PRÁCI

- ➔ (Př. 2) Experimentujte s rychlosí DC motoru.
- ➔ (Př. 2) Upravte program tak, aby se servo otácelo o větší či menší úhel.



## CO DÁL

Nyní když máte sestaven tento obvod, můžete jej využít pro nějakou složitější konstrukci. Nabízí se například následující možnosti:

- ① Využít sestavený obvod pro dálkové ovládání autička. DC motor bude pohánět jednu z náprav a servo bude sloužit pro zatáčení druhé nápravy anebo řídícího kola, pokud se bude jednat o tříkolku. Konstrukci si můžete vytisknout na 3D tiskárně anebo použít kartonovou krabici a na kola např. víčka od PET lahví anebo stará CD. Pro převody můžete použít např. gumičky.
- ② Dálkově ovládaný ventilátor. DC motor bude sloužit k pohonu vrtule a servo k jejímu natáčení do stran. Konstrukci v tomto případě proveděte nejlépe na 3D tiskárně včetně vrtule.

## ZÁVĚR

V této kapitole jste poznali princip IR ovládání a naučili se jej využívat. Sestrojili jste si obvod včetně dvou motorků – serva a DC motoru s ovladačem.

# 10. OVLÁDÁNÍ SVĚTELNÉ KŘIŽOVATKY – SEMAFOR

VĚTŠINA Z VÁS DENNĚ PŘI CESTĚ DO ŠKOLY, NA NÁKUPY ATD. POTKÁVÁ SVĚTELNÉ KŘIŽOVATKY. JISTĚ ČASTO MÁTE POCIT, že INTERVALY SVITU ČERVENÉ JSOU DLOUHÉ A NAOPAK INTERVALY SVITU ZELENÉ ABNORMÁLNĚ KRÁTKÉ. V TÉTO KAPITOLE SI SESTROJÍTE MODEL KŘIŽOVATKY, KTEROU BUDETE ŘÍDIT POMOCÍ ARDUINA.

## CÍLE

- ① Sestavit složitější konstrukci z LED diod.
- ② Pochopit princip přerušení a způsob jakým se k němu v Arduinu přistupuje.
- ③ Od jednoduché křížovatky postupovat ke složitější.

Čas: **90 min**

Úroveň: ■■■■■

Vychází z: **1, 2**



Rezistor 11x 220Ω  
2x 10 kΩ



Tlačítko 2x

POUŽITÉ SOUČÁSTKY

# PRŮVODCE HODINOU I - SEMAFOR



Studenti sestaví model semaforu ze tří diod, které naprogramují tak, aby se systém choval jako skutečný semafor. Dále si obvod rozšíří o tlačítko pro chodce, na kterém si budou testovat přerušení.

U této úlohy je vhodné (je-li to možné), aby se jednalo spolu s následující hodinou o dvouhodinovku, neboť obvod postupně roste a studenti, tak nemusí znova sestavovat obvod z minulé hodiny.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 6 diod (2 x červená, 2 x zelená, 1 x žlutá, 1x modrá). Rezistory (6x  $220\Omega$ , 1x  $10\text{ k}\Omega$ ), tlačítko
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 10, která je ke stažení na ...
- ⑤ Pracovní listy pro studenty (ke stažení na ...).

## 1. KROK ⏰ 5 minut

Na úvod rozdejte studentům sady Arduino. Nechte je připravit si potřebné součástky, pohovořte si o stavech semaforu pro řidiče a jak se na něm obvykle mění barvy.

Studenti si prohlédnou IR přijímač a IR ovladač. Pokud mají svůj z domova, je pravá chvíle, aby si jej připravili.

## 2. KROK ⏰ 15 minut

Studenti si sestaví jednoduchý obvod se semaforem a naprogramují jeho stavy dle připraveného kódu. Nechte je vše dobře vyzkoušet a promyslet jednotlivé stavy a jejich pořadí.

### 3. KROK 5 minut

Vysvětlete, co je to přerušení a jaký je jeho princip. Popište možné typy přerušení u Arduina a rozdíly mezi nimi.

#### ZEPTEJTE SE STUDENTŮ

→ Jak se mění stavy semaforu pro řidiče?

Jednotlivé stavy jsou: červená, červená a oranžová (žlutá), zelená, oranžová, červená. Doba, po kterou se semafor nachází v konkrétním stavu se může významně měnit, dle důležitosti směru, hustoty provozu atd.

→ Uveděte příklady, kde se dá použít přerušení?

→ Který typ přerušení podporovaného Arduinem, zde můžeme použít?

De facto kterýkoliv, rozdíl by byl pouze v zapojení tlačítka. V našich příkladech používáme RISING – hlídající stisk tlačítka.



### 4. KROK 20 minut

Studenti si rozšíří obvod o druhý semafor pro chodce a tlačítko. Doplní si svůj program o obsluhu druhého semaforu a tlačítka pro chodce.

Nechte je vše důkladně vyzkoušet, zejména dbejte na pochopení přerušení.

#### ÚKOLY PRO STUDENTY

→ Jak naprogramovat naši úlohu bez použití přerušení



# PRACOVNÍ LIST – SEMAFOR

V TÉTO LEKCI SI SESTAVÍME MODEL SVĚTELNÉ KŘIŽOVATKY (SEMAFORU) A NAUČÍME SE JÍ OVLÁDAT. BUDEME POSTUPOVAT OD JEDNODUCHÉ K SLOŽITĚJŠÍ. SOUČASNĚ SI NA TOMTO PŘÍPADĚ VYSVĚTLÍME PRINCIP A POUŽITÍ PŘERUŠENÍ.

## CO SE NAUČÍTE

- ① Princip semaforu.
- ② Jak fungují světelné křižovatky.
- ③ Co je to přerušení a jak jej použít.



## CO BUDETE POTŘEBOVAT

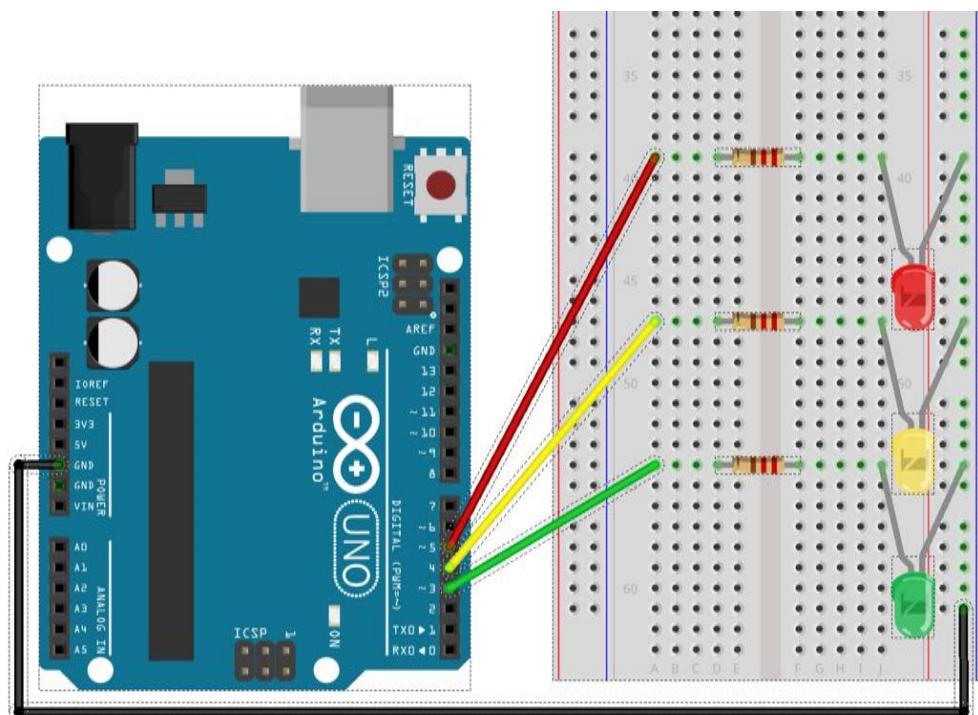
- ① LED diody (2 x červenou, 2 x zelenou, 1x žlutou, 1 x modrou).
- ② Tlačítko 2x.
- ③ Arduino.
- ④ Kontaktní pole.
- ⑤ Odpory 220  $\Omega$  (6x) a 10 k $\Omega$  (1x).
- ⑥ Vodiče typu zástrčka-zástrčka.



POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

- ① Podle schématu zapojte elektronický obvod.

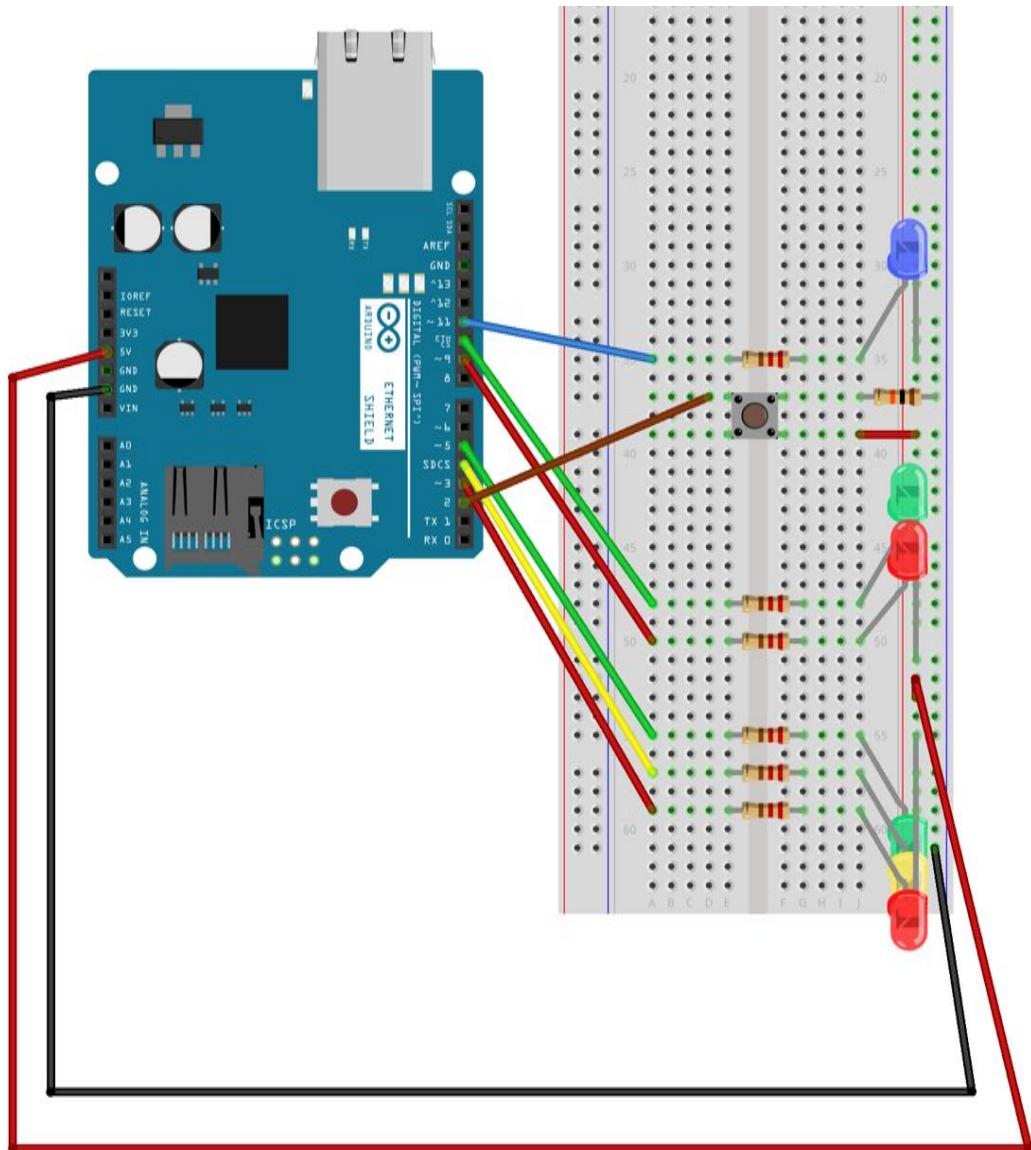


② Spusťte program Arduino IDE a napište následující programový kód.

```
1 int cervenal=3;
2 int oranzoval=4;
3 int zelenal=5;
4
5
6
7 void setup() {
8     pinMode(cervenal, OUTPUT);
9     pinMode(oranzoval, OUTPUT);
10    pinMode(zelenal, OUTPUT);
11 }
12
13
14 void loop() {
15     digitalWrite(cervenal,HIGH);
16     delay(1000);
17     digitalWrite(oranzoval,HIGH);
18     delay(1000);
19     digitalWrite(cervenal,LOW);
20     digitalWrite(oranzoval,LOW);
21     digitalWrite(zelenal,HIGH);
22     delay(2000);
23     digitalWrite(zelenal,LOW);
24     digitalWrite(oranzoval,HIGH);
25     delay(1000);
26     digitalWrite(oranzoval,LOW);
27     digitalWrite(cervenal,HIGH);
28     delay(1000);
29 }
```

Pokud vše funguje měli byste před sebou mít fungující semafory. Můžete experimentovat s dobou svícení jednotlivých světel. Až bude vše fungovat, jak si představujete, postupujte dál.

③ Nyní upravte a rozšiřte své zapojení dle následujícího schématu:



④ Do Arduina vložte následující kód:

```

1 int prepinac=2;
2 int tlacitko = 0;
3 int cervena1=3;
4 int oranzova1=4;
5 int zelena1=5;
6 int cervena3=9;
7 int zelena3=10;
8 int modra=11; //kontrolni dioda pro chodce
9
10 void setup() {
11     pinMode(prepinac, INPUT);
12     pinMode(cervena1, OUTPUT);
13     pinMode(oranzova1, OUTPUT);
14     pinMode(zelena1, OUTPUT);
15     pinMode(cervena3, OUTPUT);
16     pinMode(zelena3, OUTPUT);
17     pinMode(modra, OUTPUT);
18     digitalWrite(zelena1, HIGH);
19     digitalWrite(cervena3, HIGH);
20     attachInterrupt(digitalPinToInterrupt(prepinac),      zmena,
21 RISING);
22 }
23
24 void loop() {
25     delay(2000);
26     if (tlacitko)
27     {
28         digitalWrite(zelena1, LOW);
29         digitalWrite(oranzova1, HIGH);
30         delay(1000);
31         digitalWrite(oranzova1, LOW);
32         digitalWrite(cervena1, HIGH);
33         delay(500);
34         digitalWrite(zelena3, HIGH);
35         digitalWrite(cervena3, LOW);
36         digitalWrite(modra,LOW);
37         tlacitko=0;
38         delay(2000);
39         digitalWrite(zelena3, LOW);
40         digitalWrite(oranzova1, HIGH);
41         digitalWrite(cervena3, HIGH);
42         delay(1000);
43         digitalWrite(cervena1, LOW);
44         digitalWrite(oranzova1, LOW);
45         digitalWrite(zelena1, HIGH);
46     }
47 }
48

```

```
49 void zmena(){  
50     tlacitko=1;  
51     digitalWrite(modra, HIGH);  
52 }
```

Úloha nyní simuluje přechod pro chodce vybavený tlačítkem pro rozsvícení zelené na přechodu.

#### VYSVĚTLENÍ

- Asi nejdůležitější (a nové) pro vás v tomto případě je přerušení a jeho obsluha.
- Přerušení se nastavuje pomocí funkce `attachInterrupt` v části `setup`.
- Samotná obsluha přerušení je ve funkci `zmena`. Všimněte si, že jediné, co tato funkce udělá, je že při stisku tlačítka změní hodnotu proměnné. Dle její hodnoty pak program pozná, zda tlačítko bylo od minulého průchodu stisknuté.



#### UPOZORNĚNÍ

- Pokud nemusíte, pak obvod na konci hodiny nerozpojíte a ponechejte si jej zapojený pro příští hodinu.



#### ÚKOLY PRO VÁS

- A) Přemýšlejte, jak by bylo možné naprogramovat tuto úlohu bez použití přerušení.
- Která možnost je jednodušší
- Zkuste vymyslet další případy, kde lze s úspěchem použít přerušení.



# PRŮVODCE HODINOU II – SEMAFOR



Studenti naváží na minulou hodinu a doplní svůj obvod o dva další semafory. Vyzkouší si i změnu programu, kdy se bude jednat o křížovatku dvou cest.



## PŘÍPRAVA

Co bude v této hodině potřeba?

- ① Součásti obvodu – deska Arduino s USB kabelem, kontaktní pole, 11 diod (4 x červená, 4 x zelená, 2 x žlutá, 1x modrá). Rezistory (11x  $220\Omega$ , 2x  $10\text{ k}\Omega$ ), 2 x tlačítko.
- ② Osobní počítač pro studenty s nainstalovaným Arduino IDE.
- ③ Pokud je k dispozici, tak dataprojektor.
- ④ Prezentace k lekci 9, která je ke stažení na ...
- ⑤ Pracovní listy pro studenty (ke stažení na ...).

## 1. KROK ⏰ 20 minut

Studenti znova sestaví obvod z minulé hodiny (použijí obvod z minulé hodiny). Obvod rozšíří o další dva semafory. Naprogramují a odladí odpovídající program.

## 2. KROK ⏰ 20 minut

Studenti si opraví program, aby nyní odpovídal křížovatce. Diskutujte o změnách.

## 3. KROK ⏰ 5 minut

Diskutujte se studenty o typech světelných křížovatek.



### ÚKOLY PRO STUDENTY

- ➔ A) Šel by kód nějak zjednodušit, např. pomocí nějaké funkce?
- ➔ B) Čím budete limitování, když budete chtít simulovat nějakou rozsáhlejší křižovatku ve vašem městě?



### MOŽNÝ NÁPAD

- ➔ Máte-li ještě volné vyučovací hodiny do konce pololetí či školního roku, můžete věnovat jednu nebo dvě hodiny tomu, že studenti sestaví a naprogramují simulátor světelné křižovatky ve vašem městě
- ➔ Pozor vaše Arduino nemusí mít dostatek pinů. V takovémto případě je nutné použít Arduino Mega nebo spolu nějakým způsobem dvě Arduina synchronizovat (např. RX/TX nebo I2C)

# PRACOVNÍ LIST – SEMAFOR II

POKRAČOVÁNÍ V SEZNAMOVÁNÍ SE S MODELY SVĚTELNÝCH KŘIŽOVATEK A JEJICH OVLÁDÁNÍ.

## CO SE NAUČÍTE

- ① Zapojení složitějších typů světelních křížovatek.
- ② Zopakujete si přerušení a jak jej použít.



## CO BUDETE POTŘEBOVAT

- ① LED diody (4 x červenou, 4 x zelenou, 2x žlutou, 1 x modrou).
- ② 2 x tlačítko.
- ③ Arduino.
- ④ Kontaktní pole.
- ⑤ Odpory  $220\Omega$  (11x) a  $10\text{k}\Omega$  (2x).
- ⑥ Vodiče typu zástrčka-zástrčka.



LED diody



Rezistor 11x 220Ω  
2x 10 kΩ



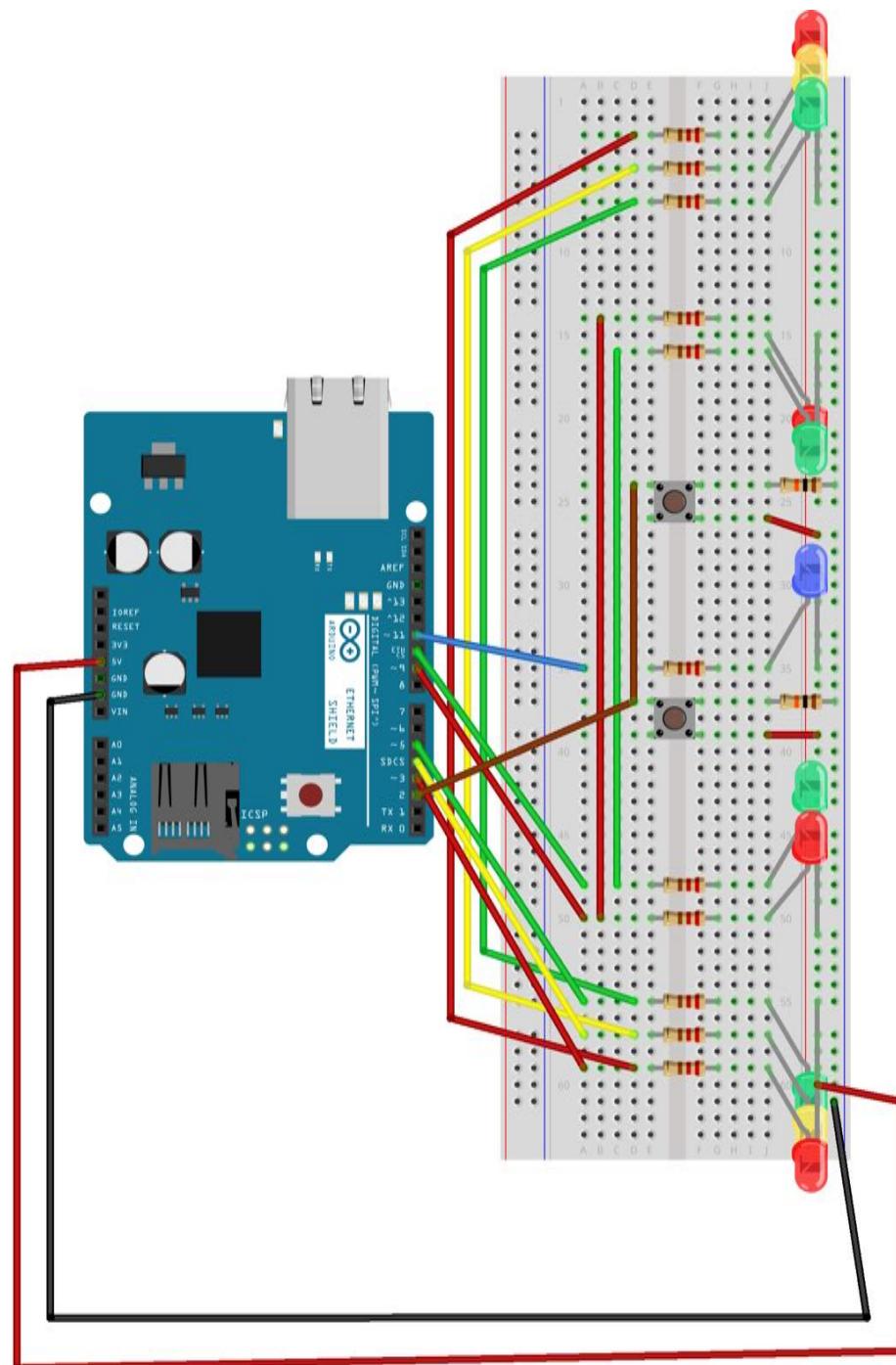
Tlačítko 2x

---

### POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

- ① Pokud nemáte zapojeno z minulé hodiny, pak schématu zapojte elektronický obvod.



Programový kód je shodný s kódem z minulé hodiny. Pokud jej máte v Arduinu stále nahraný, pak nemusíte dělat nic jiného než připojit Arduino ke zdroji. Jinak spusťte program Arduino IDE a napište programový kód z druhého příkladu z minulé hodiny.

Jedná se o zobecnění minulého příkladu. Opět se jedná o samostatný přechod pro chodce, ale tentokrát osazený semafory z obou stran silnice i přechodu a tlačítka z obou stran přechodu.

Pokud vše funguje, tak výborně. Můžete pokračovat dále Budeme se teď věnovat křižovatce dvou jednosměrných cest s jedním přechodem pro chodce, který je vybaven tlačítky pro přecházení.

- ② Nyní naopak ponechte zapojení, jak je a nahrajte následující programový kód:

```
1 int prepinac=2;
2 int tlacitko=0;
3 int cervena1=3;
4 int oranzova1=4;
5 int zelena1=5;
6 int cervena2=6;
7 int oranzova2=7;
8 int zelena2=8;
9 int cervena3=9;
10 int zelena3=10;
11 int modra=11;
12
13 void setup() {
14     pinMode(prepinac, INPUT);
15     pinMode(cervena1, OUTPUT);
16     pinMode(oranzova1, OUTPUT);
17     pinMode(zelena1, OUTPUT);
18     pinMode(cervena2, OUTPUT);
19     pinMode(oranzova2, OUTPUT);
20     pinMode(zelena2, OUTPUT);
21     pinMode(cervena3, OUTPUT);
22     pinMode(zelena3, OUTPUT);
23     pinMode(modra, OUTPUT);
24     attachInterrupt(digitalPinToInterrupt(prepinac),
25                     zmena, RISING);
26 }
27
```

```

28 void loop() {
29     digitalWrite(cervena1,HIGH);
30     digitalWrite(cervena2,HIGH);
31     digitalWrite(cervena3,HIGH);
32     delay(1000);
33     digitalWrite(oranzova1,HIGH);
34     delay(1000);
35     digitalWrite(cervena1,LOW);
36     digitalWrite(oranzova1,LOW);
37     digitalWrite(zelena1,HIGH);
38     delay(2000);
39     digitalWrite(zelena1,LOW);
40     digitalWrite(oranzova1,HIGH);
41     delay(1000);
42     digitalWrite(oranzova1,LOW);
43     digitalWrite(cervena1,HIGH);
44     delay(1000);
45     digitalWrite(oranzova2,HIGH);
46     delay(1000);
47     digitalWrite(cervena2,LOW);
48     digitalWrite(oranzova2,LOW);
49     digitalWrite(zelena2,HIGH);
50     delay(2000);
51     digitalWrite(zelena2,LOW);
52     digitalWrite(oranzova2,HIGH);
53     delay(1000);
54     digitalWrite(oranzova2,LOW);
55     digitalWrite(cervena2,HIGH);
56     delay(1000);
57     if (tlacitko)
58     {
59         tlacitko=0;
60         digitalWrite(zelena3,HIGH);
61         digitalWrite(cervena3,LOW);
62         digitalWrite(modra,LOW);
63         delay(2000);
64         digitalWrite(zelena3,LOW);
65     }
66 }
67
68 void zmena(){
69     tlacitko=1;
70     digitalWrite(modra,HIGH);
71 }
```

### ÚKOLY VÁS



- A) Šel by kód zjednodušit? Např. pomocí nějaké funkce.
- B) Dokázali byste si namodelovat světelnou křížovatku ve vašem okolí. Na jaké problémy narazíte? Jak byste jej řešili?

Poznámka: Arduino Mega má 64 vstupů a výstupů.

# PODROBNÝ PRŮVODCE TEORIÍ

PODROBNĚ ROZEPSANÉ PŘÍKLADY S POPISEM FUNKCIONALIT OBVODŮ A PROGRAMOVÉHO KÓDU A ŘEŠENÍ ÚKOLŮ A MOŽNÝCH PROBLÉMŮ PŘI NEFUNKČNOSTI OBVODŮ.

## OBSAH PRŮVODCE

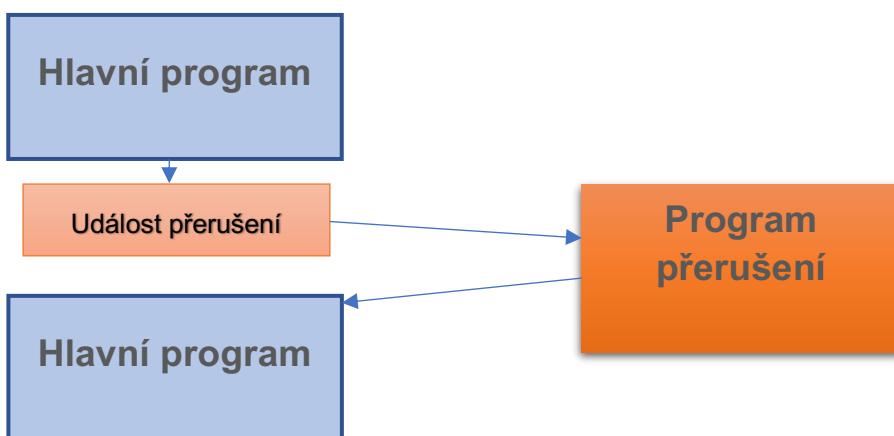
- ① Popis přerušení.
- ② Tlačítko a jeho zapojení.
- ③ Zapojení obvodu pro jednoduchý semafor.
- ④ Zdrojový kód pro jednoduchý semafor.
- ⑤ Řešení možných potíží.
- ⑥ Složitější úlohy pro semafor, včetně zapojení a zdrojového kódy.
- ⑦ Další úlohy pro samostatnou práci.

## PŘERUŠENÍ

Představme si, že je naše křižovatka umístěna na místě, kde se nepohybuje příliš chodců, ale nelze je zcela vyloučit. Je tedy zbytečné rozsvěct zelenou pro chodce, pokud žádný chodec nechce křižovatku přejít. V praxi se tato situace řeší umístěním tlačítka, které si musí chodci stisknout, pokud chtějí přecházet. Program cyklicky ošetruje jednotlivé stavy a v okamžiku, kdy dojde řada na chodce, tak si pouze ověří, zda od posledně bylo stisknuto tlačítko. Pokud ne, jedou auta z dalšího směru, pokud ano, naskočí nejprve zelená pro chodce.

Bylo by sice možné pravidelně kontrolovat stisk tlačítka – např. každou vteřinu, ale je to nepraktické a náročné na výpočetní prostředky. Navíc velmi krátký stisk (menší než 1 vteřina) nemusí být zaznamenán. Proto se tato situace řeší pomocí tzv. přerušení.

**Přerušení** si můžeme představit jako nezávisle běžící program, čekající na nějakou událost. Zde je to stisk tlačítka. Pokud tato situace nastane, pak hlavní program, přenechá výpočetní prostředky pro nezbytnou dobu programu pro obsluhu přerušení, ten vykoná, co potřebuje a opět vše vrátí hlavnímu programu.



V našem případě, se přerušení vyvolá stiskem tlačítka. Program přerušení, pak vykoná pouze to, že do **logické proměnné** uloží informaci o tom, že tlačítko bylo stisknuto (změní její hodnotu z nuly na jedničku). Pak se opět pokračuje vykonáváním hlavního programu.

V okamžiku, kdy se program rozhoduje, zda má nastavit zelenou pro chodce, se dotáže na hodnotu proměnné. Pokud je nula, semafor pro chodce se nenastavuje a pokračuje se vykonáváním programu. Pokud je jedna, zapíná se zelená na semaforu pro chodce (a všude jinde je samozřejmě červená). Současně se hodnota proměnné nastaví na nulu.

Pro obsluhu přerušení je zde použit jako vstupní pin 2. Kromě něj je na Arduinu ještě možné pro přerušení použít pin 3. Na jiných pinech nelze přerušení nastavit. Je zde použit typ přerušení **RISING**. To znamená, že se sleduje stav tohoto pinu, jakmile na něj přijde signál, zavolá se přerušení. Arduino zná tyto typy přerušení:

Typ	Význam
<b>LOW</b>	Přerušení je voláno, pokud na daném pinu není signál (logická 0).
<b>CHANGE</b>	Přerušení je voláno, pokud na daném pinu dojde ke změně signálu (změna logické 0 na 1 nebo naopak).
<b>RISING</b>	Přerušení je voláno, pokud na daný pin přijde signál (změna z logické 0 na 1).
<b>FALLING</b>	Přerušení je voláno, pokud na daném pinu je signál ukončen (změna z logické 1 na 0).

Kód pro informaci o nastavení přerušení píšeme do části setup a vypadá takto:

```
attachInterrupt(digitalPinToInterrupt(prepinac), zmena,
RISING);
```

Proměnná prepinac obsahuje číslo pinu, na kterém sledujeme přerušení. Funkce zmena je volána pro obsluhu přerušení, je umístěna na koci programu a vypadá takto:

```
void zmena(){
    tlacitko=1;
    digitalWrite(modra,HIGH);
}
```

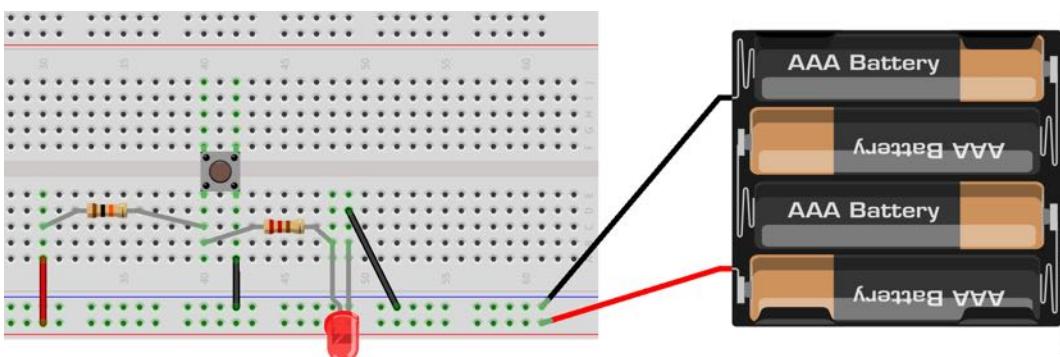
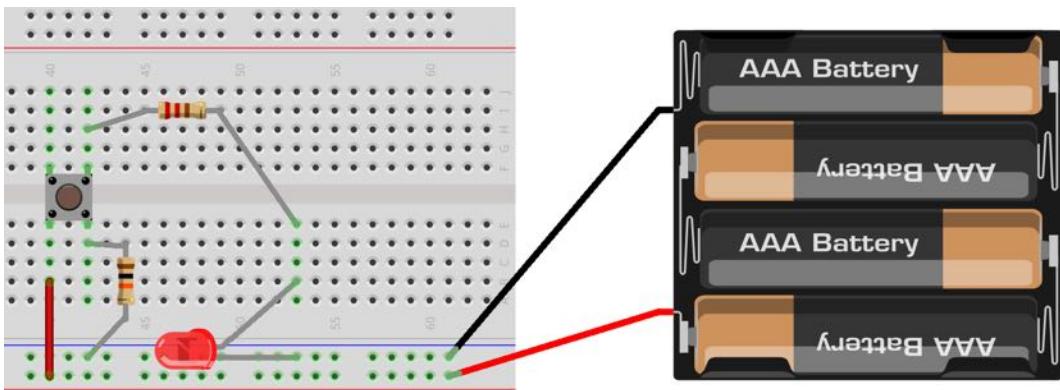
v našem příkladu, jako vedlejší efekt ještě rozsvítíme modrou diodu, která chodce upozorňuje, že jeho požadavek na umožnění přechodu byl přijat.

## TLAČÍTKO

Pasivní (pro svou činnost nemusí být přímo napájené) zařízení, které nám umožňuje naprogramovat nějakou událost, řízenou uživatelem – sepnutí nebo uvolnění tlačítka.

Vaše stavebnice obsahuje dva druhy tlačítek – malé a velké (které ještě lze ozdobit barevnou „čepičkou“). Jejich zapojení je totožné, takže si můžete vybrat dle nálady.

Tlačítka můžeme zapojit tak, že jejich sepnutím buď proud začne nebo naopak přestane procházet. Pro pochopení principu si zkuste sestavit dva následující obvody (bez Arduina) a vyzkoušejte si je. Potřebujete dva odpory (220 a 10k), tlačítko a libovolnou diodu. Pro napájení můžete s úspěchem použít součástku YwRobot z vaší stavebnice. Pozor na správné zapojení diody (delší nožička na + a kratší na -). U prvního proud prochází po sepnutí tlačítka, u druhého to je naopak proud prochází a po sepnutí tlačítka přestane.



V našem příkladu použijeme první možnost a namísto diody vyvedeme vodič, který připojíme do Arduina na port 2 a budeme pomocí přerušení sledovat, zda je pod napětím (stisknuté tlačítko) či nikoliv (normální stav).

## POZNÁMKA

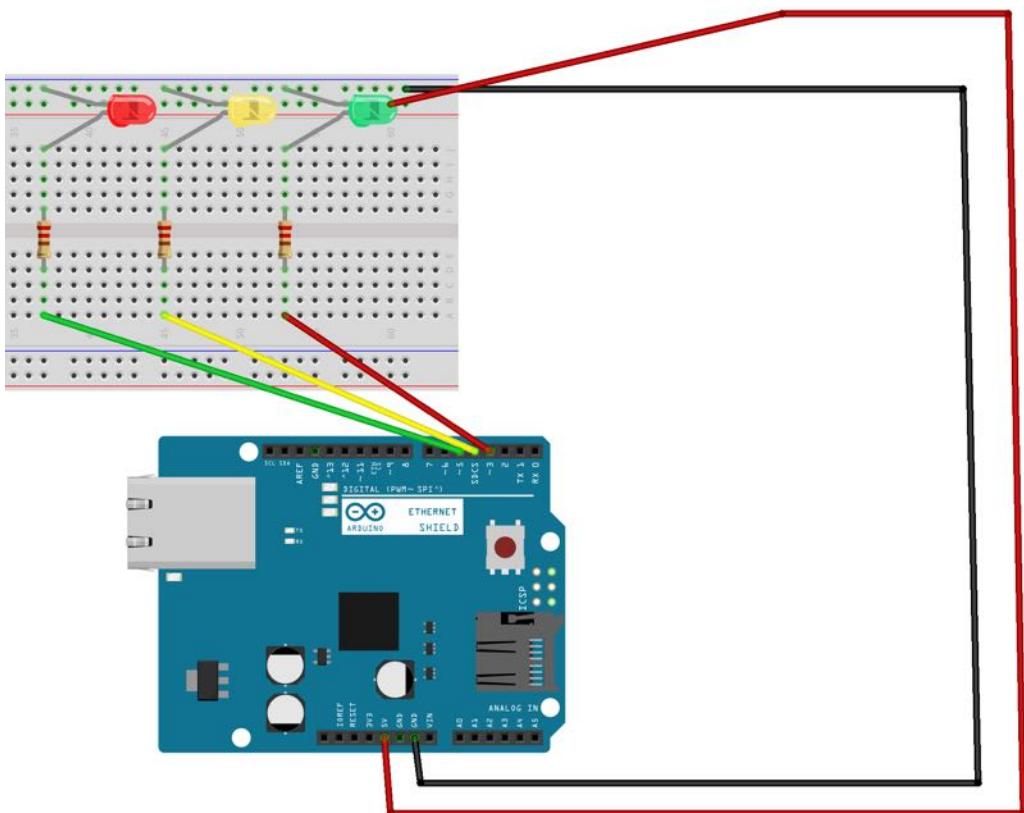
Následující úkoly na sebe navazují, a proto po vyřešení prvního úkolu není nutné obvod rozpojovat, ale naopak postupně k němu budete přidávat další součástky. Stejně tak i program je koncipován od jednoduššího ke složitějšímu.

## ÚKOL 1

Sestavte a naprogramujte model semaforu. V prvním úkolu se obejdeme bez přerušení.

## ZAPOJENÍ OBVODU

Sestavte obvod dle následujícího obrázku:



## PROGRAMOVÝ KÓD

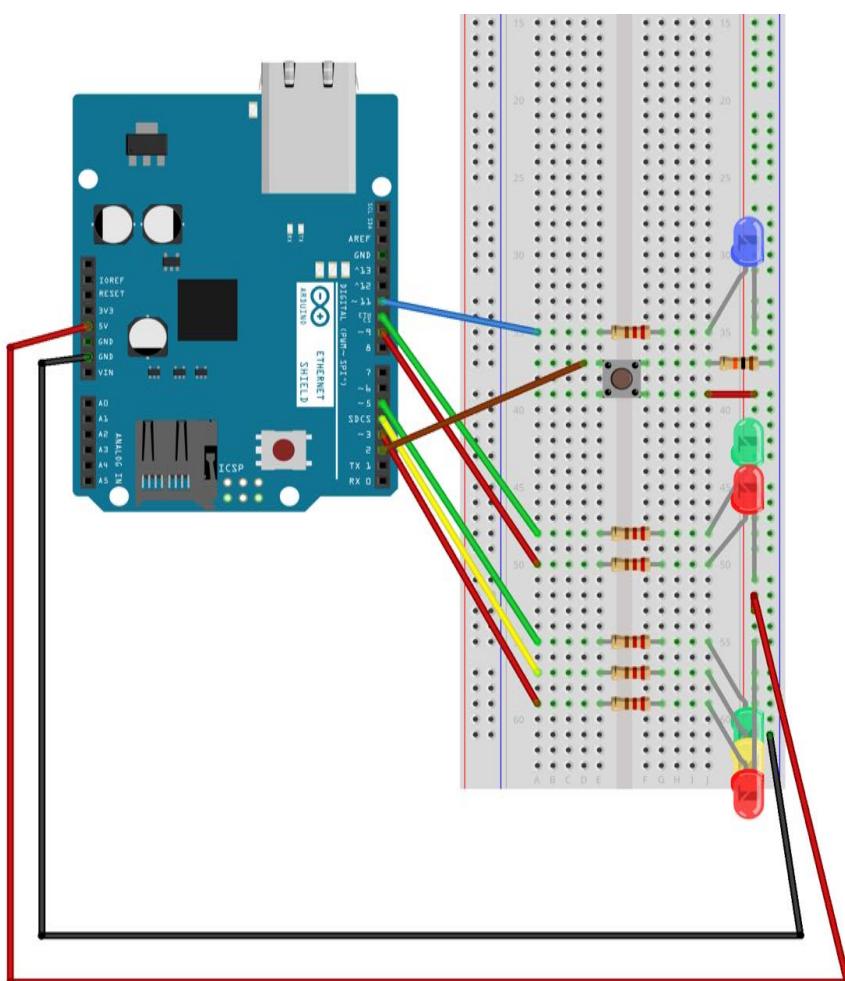
```
1 int cervena1=3;
2 int oranzova1=4;
3 int zelena1=5;
4
5
6 void setup() {
7     pinMode(cervena1, OUTPUT);
8     pinMode(oranzova1, OUTPUT);
9     pinMode(zelena1, OUTPUT);
10 }
11
12 void loop() {
13     digitalWrite(cervena1,HIGH);
14     delay(1000);
15     digitalWrite(oranzova1,HIGH);
16     delay(1000);
17     digitalWrite(cervena1,LOW);
18     digitalWrite(oranzova1,LOW);
19     digitalWrite(zelena1,HIGH);
20     delay(2000);
21     digitalWrite(zelena1,LOW);
22     digitalWrite(oranzova1,HIGH);
23     delay(1000);
24     digitalWrite(oranzova1,LOW);
25     digitalWrite(cervena1,HIGH);
26     delay(1000);
27 }
28 }
```

- ① Nastavení proměnných. Jsou použité piny 3 až 5, pin 2 bude později použit pro přerušení. Proměnné jsou nazvány cervena1, oranzova1 a zelena1, protože v dalších zadáních přibydou další semafory.
- ② Všechny piny budou výstupní.
- ③ Rozsvěcení světel dle obvyklého cyklu pro semafory pro řízení automobilového provozu.

## ÚKOL 2

Sestrojte dva semafory řídící přechod pro chodce. Jedná se o semafor pro automobily a semafor pro chodce. Navíc je zde tlačítko, kterým chodci dávají na vědomí svůj požadavek na přecházení. V případě tohoto požadavku rozsvíťte kontrolní diodu, aby chodci věděli, že jejich požadavek byl přijat. Kontrolujte jedenkrát za dvě sekundy, zda tlačítko bylo stisknuto. Pro obsluhu tlačítka použijte přerušení. Viz obrázek.

## ZAPOJENÍ OBVODU



## PROGRAMOVÝ KÓD

```
1 int prepinac=2;
2 int tlacitko = 0;
3 int cervena1=3;
4 int oranzova1=4;
5 int zelena1=5;
6 int cervena3=9;
7 int zelena3=10;
8 int modra=11; //kontrolni dioda pro chodce
9
10 void setup() {
11     pinMode(prepinac, INPUT);
12     pinMode(cervena1, OUTPUT);
13     pinMode(oranzova1, OUTPUT);
14     pinMode(zelena1, OUTPUT);
15     pinMode(cervena3, OUTPUT);
16     pinMode(zelena3, OUTPUT);
17     pinMode(modra, OUTPUT);
18     digitalWrite(zelena1, HIGH);
19     digitalWrite(cervena3, HIGH);
20     attachInterrupt(digitalPinToInterrupt(prepinac),
21 zmena, RISING);
22 }
23
24 void loop() {
25     delay(2000);
26     if (tlacitko)
27     {
28         digitalWrite(zelena1, LOW);
29         digitalWrite(oranzova1, HIGH);
30         delay(1000);
31         digitalWrite(oranzova1, LOW);
32         digitalWrite(cervena1, HIGH);
33         delay(500);
34         digitalWrite(zelena3, HIGH);
35         digitalWrite(cervena3, LOW);
36         digitalWrite(modra, LOW);
37         tlacitko=0;
38         delay(2000);
39         digitalWrite(zelena3, LOW);
40         digitalWrite(oranzova1, HIGH);
41         digitalWrite(cervena3, HIGH);
```

```

42     delay(1000);
43     digitalWrite(cervena1, LOW);
44     digitalWrite(oranzova1, LOW);
45     digitalWrite(zelena1, HIGH);
46   }
47 }
48
49 void zmena(){
50   tlacitko=1;
51   digitalWrite(modra, HIGH);
52 }
```

⑤

- ① Nastavení proměnných. Tlačítko (tlačítka) bude připojeno na pin 2 (viz kapitola o **přerušení**). Proměnná tlacitko udává, zda byl stisknut přepínač (hodnota 1) nebo ne (hodnota 0). Defaultní je nula. Následuje nastavení proměnných určujících, na které piny budou připojeny jednotlivé LED semaforů. Piny 3 až 5 jsou první semafor, 9 a 10 semafor pro chodce. Na pinu 11 je pak připojena modrá dioda pro informaci pro chodce, že jejich požadavek byl přijat.
- ② Určení, který z pinů bude vstupní (zde pouze pin 2) a které budou výstupní.
- ③ Informace, že pro pin 2 je nastaveno **přerušení**, o jaký typ přerušení se jedná a jaká funkce se bude zpracovávat při jeho vyvolání.
- ④ Základní část kódu. Program každé dvě vteřiny kontroluje, zda bylo stisknuto tlačítko. Pokud ano umožní chodcům přecházení. Stav se zjišťuje dle hodnoty tlačítka (1 = bylo stisknuto, 0 = nebylo stisknuto). Na konci této části se zhasne dioda a nastaví se hodnota tlačítka na nulu.
- ⑤ Obsluha přerušení. Nastavuje se hodnota proměnné tlacitko na jedna a současně se rozsvětí modrá dioda.

### OTÁZKY PRO VÁS

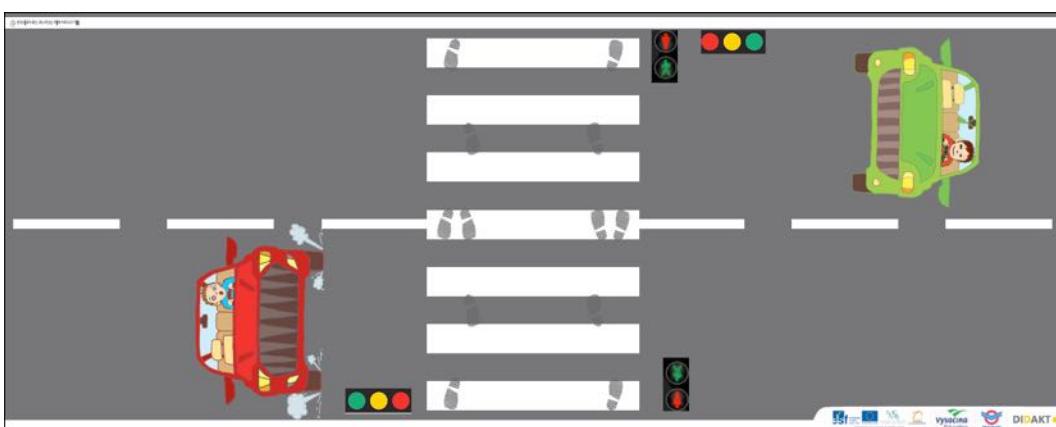
→ **Bylo by možné tuto úlohu řešit bez použití přerušení?**

Ano bylo by to poměrně jednoduše možné. Nejlepší řešení by bylo tak, že bychom zkrátili interval cyklu na cca. 0.2 sekundy a rovnou bychom kontrolovali, zda není stisknuto tlačítko. Museli bychom pravděpodobně přidat hlídání délky zelené pro automobily, aby nebyla příliš krátká. Také bychom obtížněji implementovali stavovou diodu.



## ÚKOL 3

Sestrojte skupinu semaforů řídících přechod pro chodce. Z obou stran je semafor pro automobily a rovněž tak z obou stran ulice je semafor pro chodce a tlačítko, kterým chodci dávají na vědomí svůj požadavek na přecházení. V případě tohoto požadavku rozsvíťte kontrolní diodu, aby chodci věděli, že jejich požadavek byl přijat. Kontrolujte jedenkrát za dvě sekundy, zda tlačítko bylo stisknuto. Pro obsluhu tlačítka použijte přerušení. Viz obrázek.



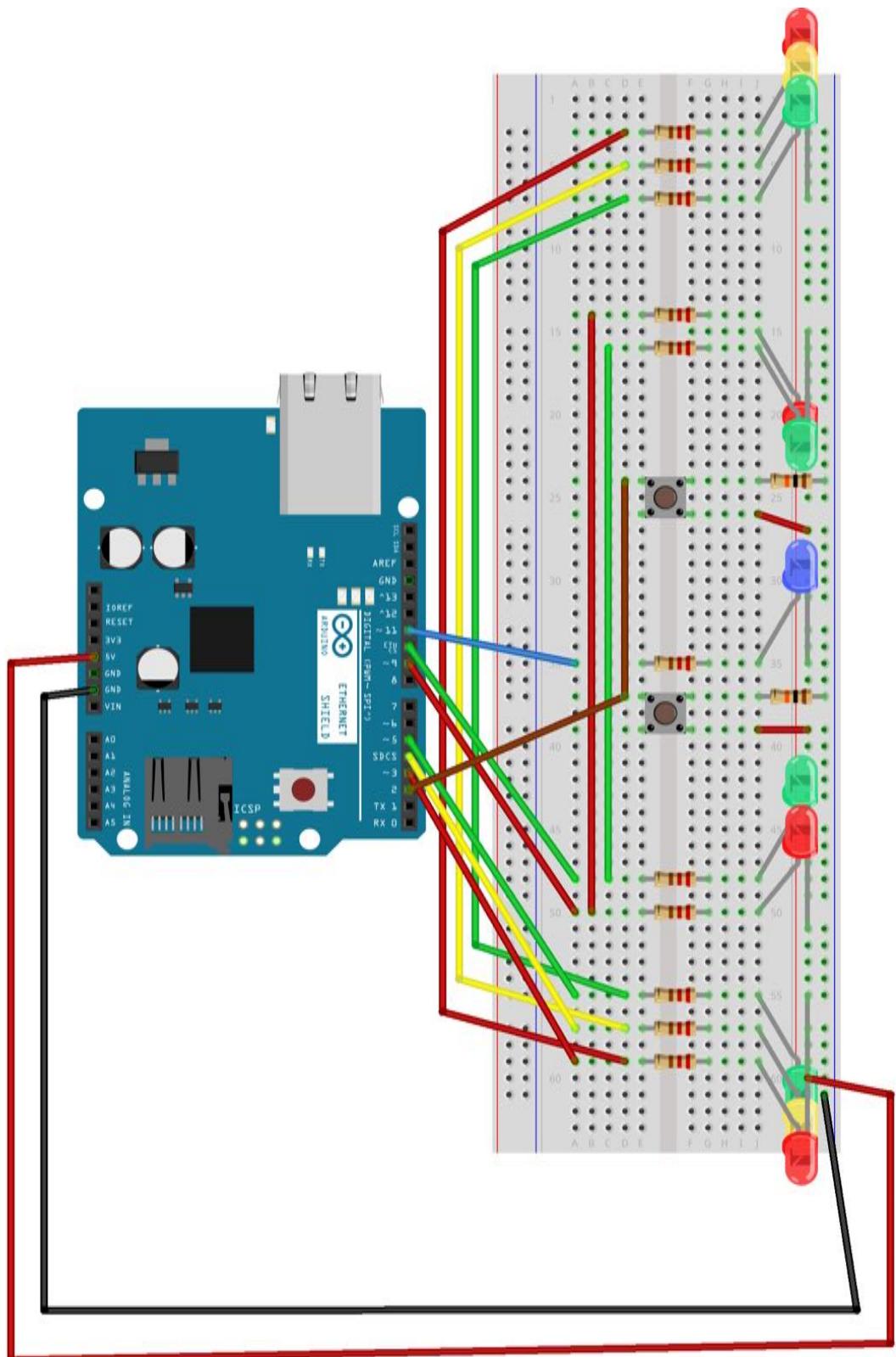
### ZAPOJENÍ OBVODU

Obvod vzniká rozšířením minulého obvodu o dva další semafory a jedno tlačítko.

Všimněte si, že jsou semafory propojené, neboť pro automobily z obou stran se musí následně rozsvítit jednotlivé barvy a stejně tak i pro chodce. Rovněž si všimněte, že jsou vynechány tři piny na Arduinu (6-8), které budou použity v dalším úkolu.

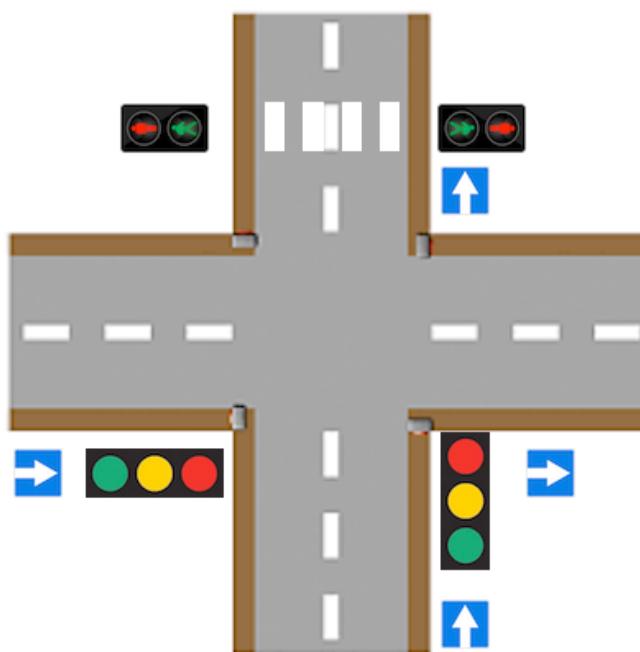
### PROGRAMOVÝ KÓD

Je shodný s předchozím příkladem, změna je pouze v zapojení.



## ÚKOL 4

Sestrojte model křižovatky dvou jednosměrných cest s jedním přechodem pro chodce řízený pomocí semaforu. Viz obrázek.



Jedná se vlastně o zobecnění předchozího případu. Jediný rozdíl bude v tom, že oba směry budou řešit dva nezávislé semafory. Při složitější křižovatce, bychom potřebovali více výstupů, než má Arduino k dispozici. (14 digitálních) Na takovouto křižovatku nám postačí deset pinů:

- Po třech na každý směr vozidel.
- Dva pro přechod pro chodce. Ty jsou pak rozvedeny do dvou semaforů.
- Jeden pro tlačítko pro přechod chodců. Správně bychom měli opět umístit dvě tlačítka na každou stranu jedno a výstup z nich spojit.
- Jeden pro diodu, která chodcům signalizuje, že byl přijat požadavek pro jejich přecházení.

Jedná se vlastně o systém se třemi následujícími stavami:

- ① Jedou auta ve směru „zdola“.

- ② Jedou auta ve směru „zleva“.
- ③ Přecházejí chodci, pokud bylo stisknuto tlačítko.

V rámci bezpečnosti při každém přechodu mezi stavy na určitou dobu (zde modelově 1s) musí svítit na všech semaforech červená.

Stavy u směrů automobilů a jejich časy jsou definovány dle následující tabulky:

červená	1s
Červená a oranžová	1s
Zelená	2s
Oranžová	1s

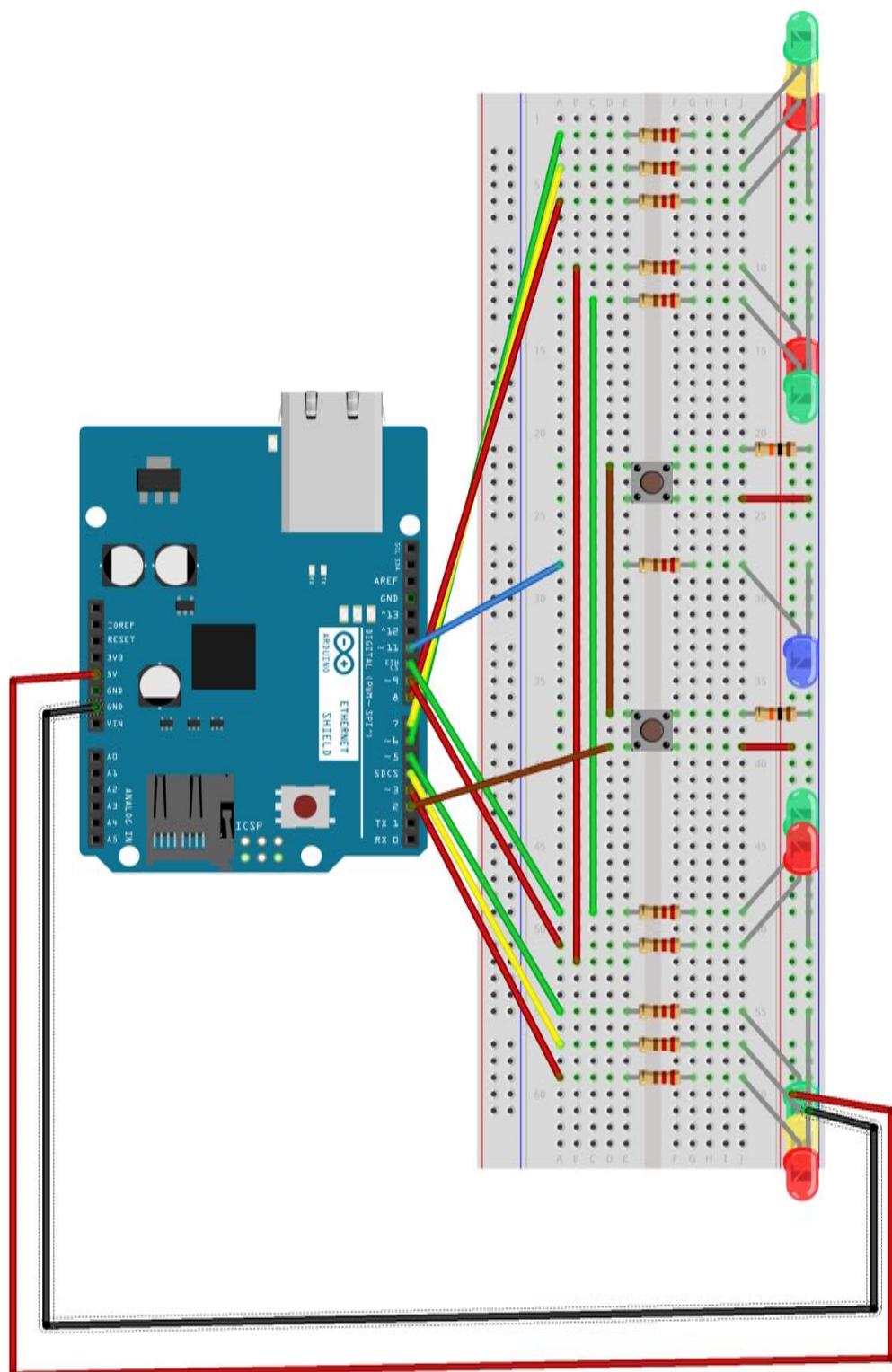
Stavy u přechodu pro chodce:

Červená	1s
Zelená	2s

Tento model křižovatky lze snadno převést na „klasickou“ křižovatku, kde auta jezdí z obou směrů a mají zelenou vždy oba směry proti sobě a chodci mohou přecházet po všech stranách. Postačí sestavit další semafory z diod a rozvést k nim vodiče. Budete rovněž potřebovat celkem osm tlačítek a pokud chceme dát ke všem tlačítkům informační diodu, pak i osm diod. To je však již množství, na které vám nepostačí součástky z jedné stavebnice. Můžete však pracovat ve větší skupině a sestrojit takovouto křižovatku.

Pro začátek ale zůstaneme pro jednoduchost u modelu se čtyřmi semafory (dva pro chodce) a dvěma tlačítky.

## ZAPOJENÍ OBVODU



## PROGRAMOVÝ KÓD

```
1 int prepinac=2;
2 int tlacitko=0;
3 int cervena1=3;
4 int oranzova1=4;
5 int zelena1=5;
6 int cervena2=6;
7 int oranzova2=7;
8 int zelena2=8;
9 int cervena3=9;
10 int zelena3=10;
11 int modra=11;
12
13 void setup() {
14     pinMode(prepinac, INPUT);
15     pinMode(cervena1, OUTPUT);
16     pinMode(oranzova1, OUTPUT);
17     pinMode(zelena1, OUTPUT);
18     pinMode(cervena2, OUTPUT);
19     pinMode(oranzova2, OUTPUT);
20     pinMode(zelena2, OUTPUT);
21     pinMode(cervena3, OUTPUT);
22     pinMode(zelena3, OUTPUT);
23     pinMode(modra, OUTPUT);
24     attachInterrupt(digitalPinToInterrupt(prepinac),
25                     zmena, RISING);
26 }
27
28 void loop() {
29     digitalWrite(cervena1,HIGH);
30     digitalWrite(cervena2,HIGH);
31     digitalWrite(cervena3,HIGH);
32     delay(1000);
33     digitalWrite(oranzova1,HIGH);
34     delay(1000);
35     digitalWrite(cervena1,LOW);
36     digitalWrite(oranzova1,LOW);
37     digitalWrite(zelena1,HIGH);
38     delay(2000);
39     digitalWrite(zelena1,LOW);
40     digitalWrite(oranzova1,HIGH);
41     delay(1000);
```

The code is annotated with four callout numbers:

- ① Points to the variable declarations at the top of the setup() function.
- ② Points to the pinMode() calls for pins 3 through 11.
- ③ Points to the attachInterrupt() call in the setup() function.
- ④ Points to the digitalWrite() loop in the loop() function.

```

42  digitalWrite(oranzova1,LOW);
43  digitalWrite(cervena1,HIGH);
44  delay(1000);
45  digitalWrite(oranzova2,HIGH);
46  delay(1000);
47  digitalWrite(cervena2,LOW);
48  digitalWrite(oranzova2,LOW);
49  digitalWrite(zelena2,HIGH);
50  delay(2000);
51  digitalWrite(zelena2,LOW);
52  digitalWrite(oranzova2,HIGH);
53  delay(1000);
54  digitalWrite(oranzova2,LOW);
55  digitalWrite(cervena2,HIGH);
56  delay(1000);
57  if (tlacitko)
58  {
59    tlacitko=0;
60    digitalWrite(zelena3,HIGH);
61    digitalWrite(cervena3,LOW);
62    digitalWrite(modra,LOW);
63    delay(2000);
64    digitalWrite(zelena3,LOW);
65  }
66 }
67
68 void zmena(){
69   tlacitko=1;
70   digitalWrite(modra,HIGH);
71 }
72

```

- ① Nastavení proměnných. Tlačítko (tlačítka) bude připojeno na pin 2 (viz kapitola o **přerušení**). Proměnná tlacitko udává, zda byl stisknut přepínač (hodnota 1) nebo ne (hodnota 0). Defaultní je nula. Následuje nastavení proměnných určujících, na které piny budou připojeny jednotlivé LED semaforů. Piny 3 až 5 jsou první semafor, 5 až 8 druhý semafor, 9 a 10 semafor pro chodce. Na pinu 11 je pak připojena modrá dioda pro informaci pro chodce, že jejich požadavek byl přijat.
- ② Určení, který z pinů bude vstupní (zde pouze pin 2) a které budou výstupní.
- ③ Informace, že pro pin 2 je nastaveno **přerušení**, o jaký typ přerušení se jedná a jaká funkce se bude zpracovávat při jeho vyvolání.

- ④ Procházení té části kódu, která se vykoná vždy. Jedná se o oba směry silničního provozu. Časy jsou nastaveny v přechozích tabulkách.
- ⑤ Část kódu, která obsluhuje semafor pro chodce a je aktivována na základě hodnoty proměnné tlacitko. Pokud je její hodnota rovna 1, pak se tato část kódu provede, jinak nikoliv. Během provádění této části kódu se hodnota proměnné opět nuluje. Rovněž se zhasíná modrá dioda.
- ⑥ Obsluha přerušení. Nastavuje se hodnota proměnné tlacitko na jedna a současně se rozsvěcí modrá dioda.



## NESVÍTÍ DIODA

**Zapojení diody** – zkontrolujte jejich zapojení v kontaktní poli. Zkuste vyměnit diodu.

**Zapojení v desce** – zkontrolujte, zda jsou vývody zapojeny do odpovídajících pinů desky Arduino.

**Rezistory** – zkontrolujte, zda jste použili správný rezistory.

## NEJDE NAHRÁT KÓD DO DESKY

**USB kabel** – ujistěte se, že máte desku Arduino připojenou k počítači.

**Správný port** – ujistěte se, že máte vybraný správný port pro připojení k desce Arduino pomocí USB kabelu.

## NEFUNGUJE TLAČÍTKO

**Zapojení tlačítka** – ověřte, zda máte správně připojené piny dle schématu. Tlačítko má zespod číslo. Na číslo 1 se připojuje napájení, číslo 2 propojíte přes odpor 10k se zemí a z čísla 3 jde výstup.

**Propojení tlačítek** – zkuste odebrat propojení se sekundárním tlačítkem (to od kterého nejde vodič k Arduinu) a zkuste nyní primární tlačítko.

## JINÉ

**Nesprávné pořadí rozsvěcení diod** – ověřte pečlivě zapojení diod (dle schématu i zdrojového kódu). Ověřte, zda nemáte překlep ve zdrojovém kódu.



Bylo by možné tuto úlohu řešit bez použití přerušení?

Zde již by to bylo významně složitější. Museli bychom neustále kontrolovat stisk tlačítka a v případě stisku volat funkci pro obsluhu přerušení.

## ZÁVĚR

Na tomto příkladu jste si ukázali sestrojení a naprogramování složitějšího obvodu z tlačítek a LED diod. Dříve jste pochopili princip přerušení a naučili se jej používat.

# PRŮVODCE HODNOCENÍM

TENTO PRŮVODCE JE SPÍŠE MOŽNÝM NÁVODEM A PŘEHLEDEM VZOROVÝCH OTÁZEK NEBO ÚLOH, KTERÉ MOHOU BÝT VYUŽITY K HODNOCENÍ ŽÁKŮ FORMOU ZNÁMKOVÁNÍ.

Jsou zde uvedeny příklady, které mohou být námětem pro otestování nabytých znalostí žáků se zaměřením zejména na programování. Studenti by neměli být primárně zkoušeni z elektroniky. Sestavování obvodů je sice důležitou součástí robotiky, ale v rámci výuky programování robotických nebo vestavěných (embedded) systémů by neměla být hlavním kritériem pro hodnocení.

Úkoly by měli být realizovány v praktické rovině, tzn. že by se nemělo jednat o pouhé vyplňování testů, ale žák by měl mít možnost si své řešení ověřit. V tomto ohledu se zde nabízí možnost praktického ověřování přímo na obvodech. Obvody může mít učitel připraveny, zejména pokud se bude jednat o složitější konstrukce. Jednodušší může žák sestavit sám, ale primárně podle schématu zapojení.

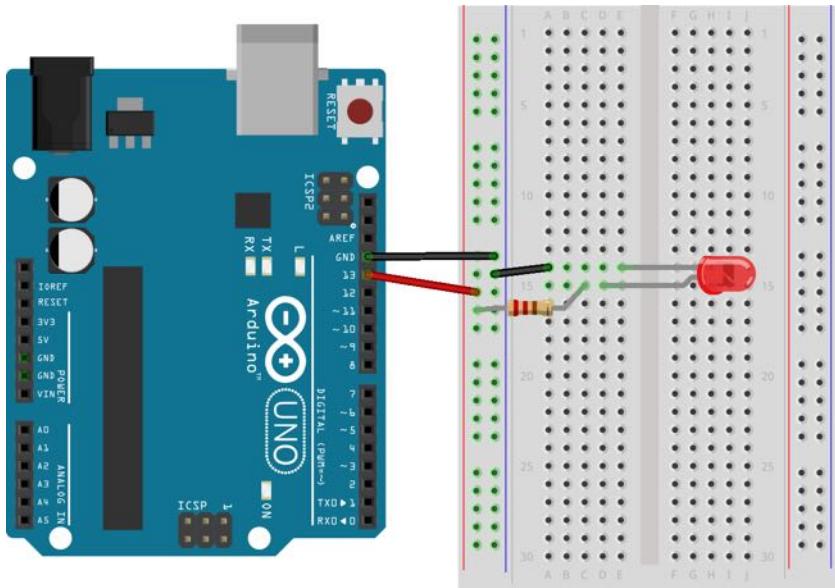
Otázky jsou rozděleny do témat, které kopírují strukturu učebnice. Z jednotlivých otázek lze poskládat rozsáhlejší dokument pro testování. Znalosti získané v jednotlivých témaitech lze dále testovat i s jinými elektronickými komponentami z jiných kapitol a uvedené otázky by měli sloužit jako návod, jak vytvořit další. Většinou se jedná o poměrně jednoduché otázky, které vedou k doplnění chybějícího fragmentu programového kódu, nebo otestování kódu za účelem zjištění funkcionality.

Otázka nebo skupina otázek je rozdělena na následující části:

1. Zadání – obsahuje schéma zapojení, programový kód nebo obojí.
2. Otázka – na základě zadání je formulována otázka, která pracuje s tímto zadáním pracuje. K jednomu zadání může být uvedeno více otázek.
3. Odpověď – vzorová řešení pro zodpovězení otázky.

# OTÁZKY LED

## ZADÁNÍ



```
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, HIGH);  
    delay(1000);  
}
```



### OTÁZKA

→ Z uvedeného obvodu a programového kódu zjistěte, zda bude dioda v obvodu blikat.



### ODPOVĚĎ

→ Dioda blikat nebude, protože v obou funkcích `digitalWrite` jsou uvedeny v druhém parametru hodnoty `HIGH` nebo `1`.



### OTÁZKA

→ Jak změníte uvedený kód, aby dioda začala blikat?

### ODPOVĚĎ

V jedné z funkcí **digitalWrite** musí být nastaven druhý parametr na hodnotu **LOW** nebo **0**.

```
void setup() {  
    pinMode(13, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(13, HIGH);  
    delay(1000);  
    digitalWrite(13, LOW);  
    delay(1000);  
}
```



### OTÁZKA

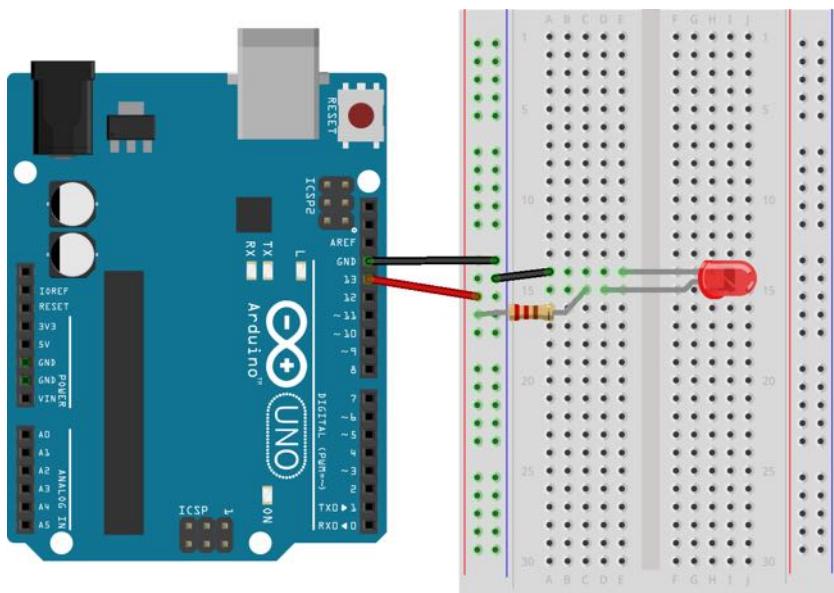
→ Jak upravíte program, aby dioda blikala pomaleji?



### ODPOVĚĎ

→ Dioda bude blikat pomaleji, jestliže se změní hodnota parametru funkce **delay** na vyšší číslo. Hodnota čísla je udávána v milisekundách.

## ZADÁNÍ



```
void setup() {  
    pinMode(12, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(12, HIGH);  
    delay(1000);  
    digitalWrite(12, LOW);  
    delay(1000);  
}
```

### OTÁZKA

→ Bude dioda blikat nebo svítit?



### ODPOVĚĎ

→ Dioda nebude svítit ani blikat, protože je zapojena na vstup 13, ale v programu je nastavena hodnota vstupního pinu na 12.





### OTÁZKA

→ Jak upravíte kód, aby byl příklad funkční?

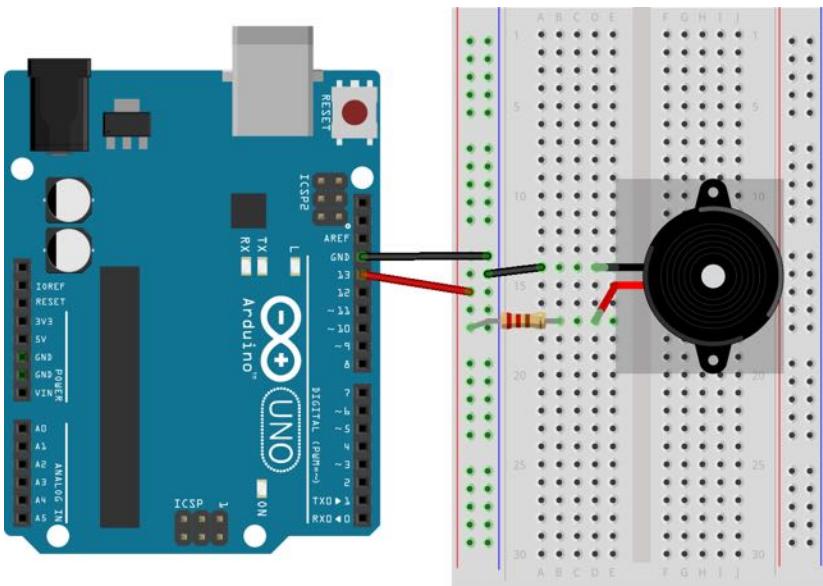


### ODPOVĚĎ

- Změní se číslo vstupního pinu na hodnotu 13.
- Druhým řešením může být připojení červeného vodiče v obvodu na pin 12.

# OTÁZKY PIEZO BZUČÁK

## ZADÁNÍ



```
const int bzucak=12;

void setup() {
    pinMode(bzucak, OUTPUT);
}

void loop() {
    tone(bzucak, 440);
    delay(1000);
    noTone(bzucak);
    delay(1000);
}
```

### OTÁZKA

→ Z uvedeného obvodu a programového kódu zjistěte, zda bude bzučák vydávat nějaký tón?



### ODPOVĚĎ

→ Bzučák nebude vydávat tón, protože vstupní pin je nastaven na číslo 12, ale v obvodu je zapojen na pin 13.





### OTÁZKA

→ Upravte programový kód tak, aby bzučák vydával nějaký zvuk.



### ODPOVĚĎ

→ V programovém kódu stačí změnit hodnotu konstanty bzucak na 13.

```
const int bzucak=13;
```



### OTÁZKA

→ Jak upravíte programový kód, aby bzučák vydával zvuk delší dobu?



### ODPOVĚĎ

→ V programovém kódu stačí změnit hodnotu funkce delay na hodnotu 3000.

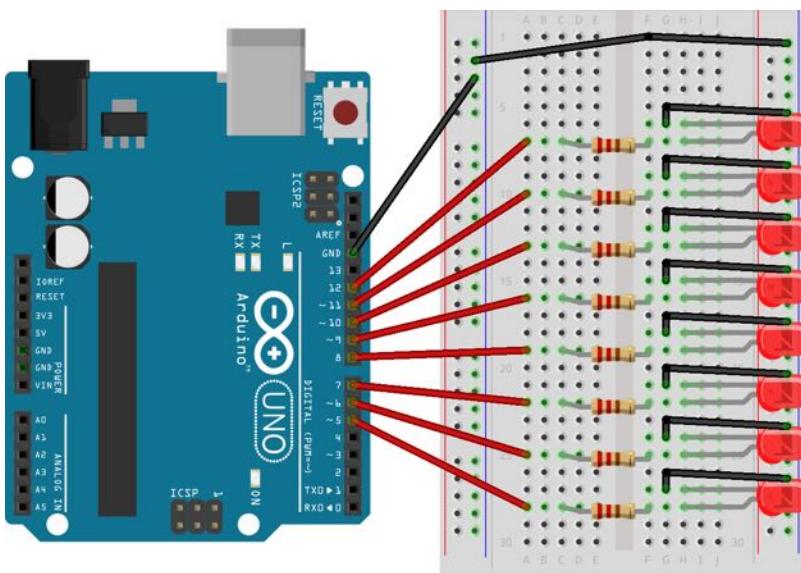
```
const int bzucak=13;
```

```
void setup() {
    pinMode(bzucak, OUTPUT);
}

void loop() {
    tone(bzucak, 440);
    delay(3000); // zde se zvýší hodnota
    noTone(bzucak);
    delay(1000);
}
```

# OTÁZKY LED ANIMACE

## ZADÁNÍ



## OTÁZKA

→ Doplňte programový kód tak, abyste využili níže uvedenou funkci a světlo opakovaně běželo z jedné strany na druhou?

```
void setup() {  
    pinMode(5, OUTPUT);      // dioda 1  
    pinMode(6, OUTPUT);      // dioda 2  
    pinMode(7, OUTPUT);      // dioda 3  
    pinMode(8, OUTPUT);      // dioda 4  
    pinMode(9, OUTPUT);      // dioda 5  
    pinMode(10, OUTPUT);     // dioda 6  
    pinMode(11, OUTPUT);     // dioda 7  
    pinMode(12, OUTPUT);     // dioda 8  
}  
  
void changeLED(int pin) {  
    digitalWrite(pin, HIGH);  
    delay(50);  
    digitalWrite(pin, LOW);  
    delay(50);  
}
```



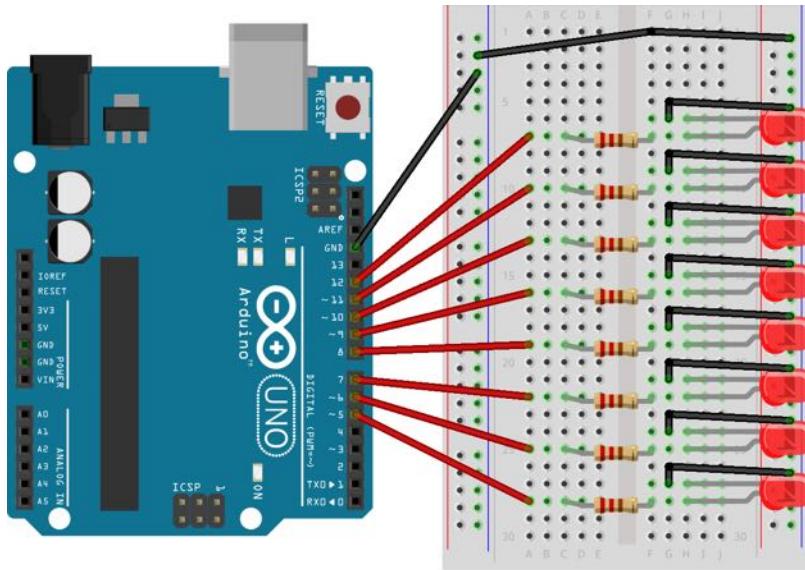


### ODPOVĚĎ

- V programovém kódu stačí opakovat volání deklarované funkce `changeLED`.

```
void loop() {  
    changeLED(5);  
    changeLED(6);  
    changeLED(7);  
    changeLED(8);  
    changeLED(9);  
    changeLED(10);  
    changeLED(11);  
    changeLED(12);  
}
```

## ZADÁNÍ



```
int pinArray[] = {5, 6, 7};  
int count = 0;  
int timer = 50;  
  
void setup() {  
    for (count=0;count<sizeof(pinArray);count++) {  
        pinMode(pinArray[count], OUTPUT);  
    }  
}  
  
void loop() {  
    for (count=0; count<sizeof(pinArray); count++) {  
        changeLED(pinArray[count]);  
    }  
}  
  
void changeLED(int pin) {  
    digitalWrite(pin, HIGH);  
    delay(timer);  
    digitalWrite(pin, LOW);  
    delay(timer);  
}
```

### OTÁZKA

→ Jak byste doplnili čísla pinů v poli pinArray tak, aby se diody opakovaně rozsvěcovali z prava do leva a zpět? Zapojení diod je vidět na schématu.

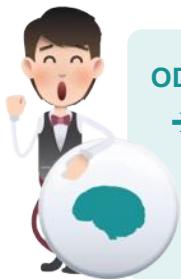
```
int pinArray[] = {5, 6, 7};
```



### ODPOVĚĎ

→ Pole pinArray se doplní o chybějící čísla pinů v následujícím pořadí.

```
int pinArray[] = {5,6,7,8,9,10,11,12,11,10,9,8,7,6 };
```



### OTÁZKA

→ Jak byste upravili programový kód, aby každá dioda v každém cyklu blikla 2x?



### ODPOVĚĎ

→ Prvním řešením je úprava pole pinArray.

```
int pinArray[] = {5,5,6,6,7,7};
```

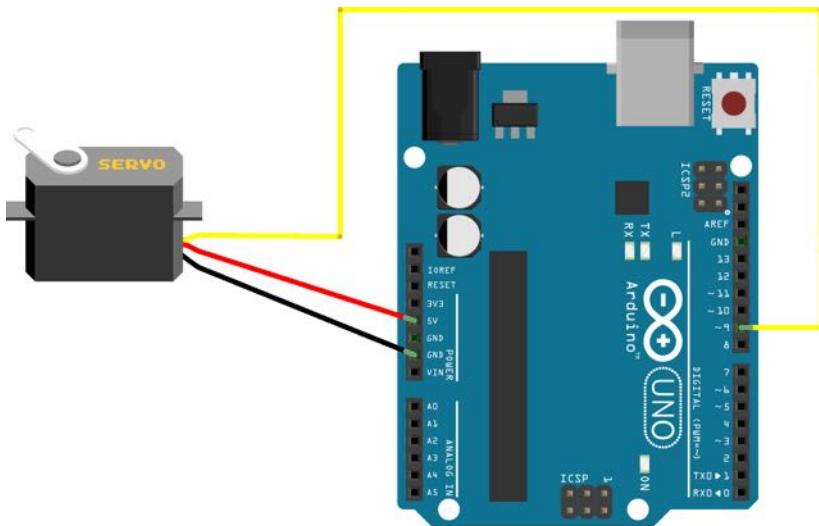
→ Druhým řešením je úprava funkce changeLED.

```
void changeLED(int pin) {
    digitalWrite(pin, HIGH);
    delay(timer);
    digitalWrite(pin, HIGH);
    delay(timer);
    digitalWrite(pin, LOW);
    digitalWrite(pin, LOW);
    delay(timer);
}
```



# OTÁZKY SERVO – FOR, IF

## ZADÁNÍ



```
#include <Servo.h>

Servo myservo;

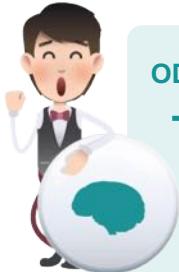
void setup()
{
    myservo.attach(9);
}

void loop()
{
    myservo.write(pos); // Nastavení pozice servomotoru
    delay(15);
}
```



### OTÁZKA

→ Nadefinujte proměnou pos tak, aby se osa servomotoru otočila do pozice 90°.



### ODPOVĚĎ

→ Stačí napsat jediný řádek, který definuje proměnou pos s hodnotou 90°.

`int pos = 90;`



### OTÁZKA

→ Jak byste upravili programový kód, aby se osa servomotoru natáčela postupně na 10°, 40°, 80°, 120° a 180°?

## ODPOVĚĎ

- ➔ Prvním řešením může být postupné natáčení s využitím volání metody `write` pro každý úhel.

```
#include <Servo.h>
Servo myservo;
void setup()
{
    myservo.attach(9);
}

void loop()
{
    myservo.write(10);
    delay(1000);
    myservo.write(40);
    delay(1000);
    myservo.write(80);
    delay(1000);
    myservo.write(120);
    delay(1000);
    myservo.write(180);
    delay(1000);
}
```

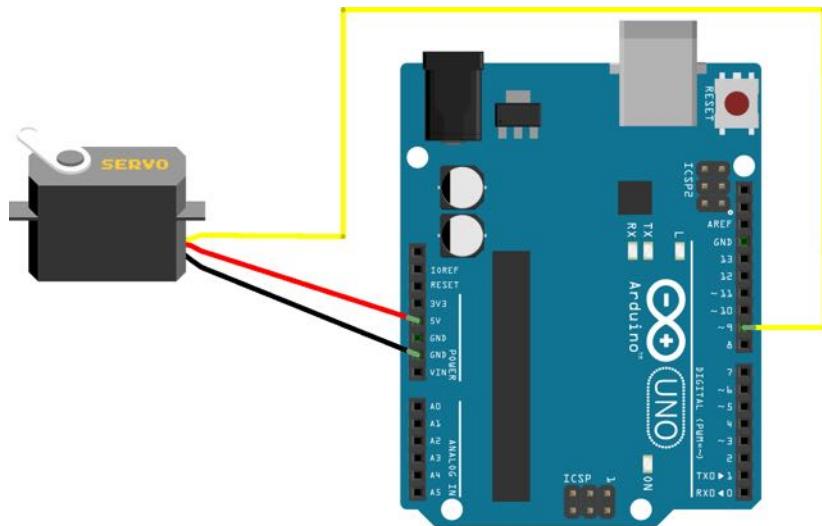
- ➔ Druhým řešením může být využití pole a cyklu `for`.

```
#include <Servo.h>
int degreeArray[] = {10,40,80,120,180};
Servo myservo;
void setup()
{
    myservo.attach(9);
}

void loop()
{
    for(pos = 0; pos <= 5; pos += 1){
        myservo.write(degreeArray[pos]);
        delay(1000);
    }
}
```



## ZADÁNÍ



```
#include <Servo.h>

Servo myservo;
void setup()
{
    myservo.attach(9);
}

void loop()
{
    for(pos = 0; pos <= 180; pos += 1){
        myservo.write(pos);
        delay(20);
    }
}
```



## OTÁZKA

→ Jak upravíte programový kód, aby se servomotor otácel i zpět?

## ODPOVĚĎ

→ Do programového kódu se přidá cyklus `for`, který zajistí opačné natáčení servomotoru.

```
#include <Servo.h>

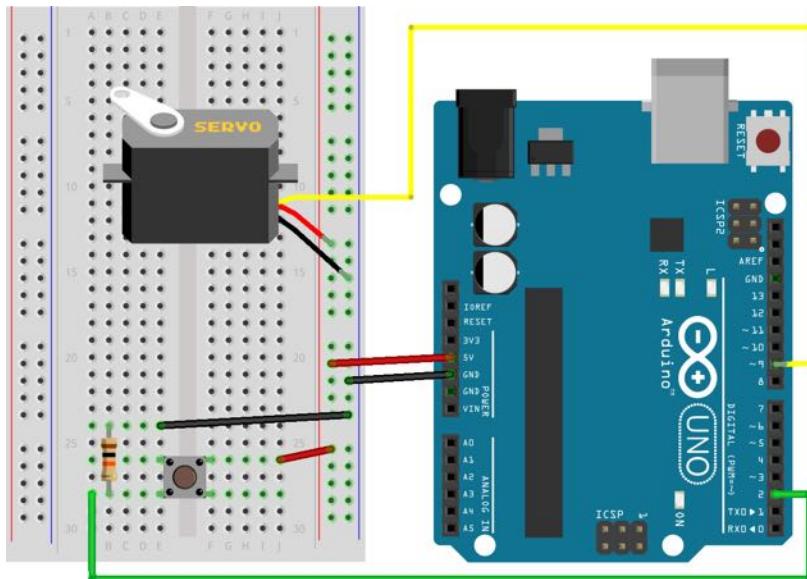
Servo myservo;
Int pos = 0;

void setup(){
    myservo.attach(9);
}

void loop(){
    for(pos = 0; pos <= 180; pos += 1){
        myservo.write(pos);
        delay(20);
    }
    for(pos = 180; pos >= 0; pos -= 1) {
        myservo.write(pos);
        delay(20);
    }
}
```



## ZADÁNÍ



```
#include <Servo.h>

int servoPin = 9;
int Button = 2;
int servoPos = 0;
int delayPeriod = 2;

Servo myservo;

void setup(){
    myservo.attach(servoPin);
    myservo.write(servoPos);
    pinMode(Button, INPUT);
}

void loop(){

    if(servoPos < 180){
        servoPos++;
    }
    myservo.write(servoPos);
    delay(delayPeriod);
}
```



### OTÁZKA

- ➔ Jak upravíte programový kód, aby se servomotor otáčel pouze při stisknutém tlačítku?

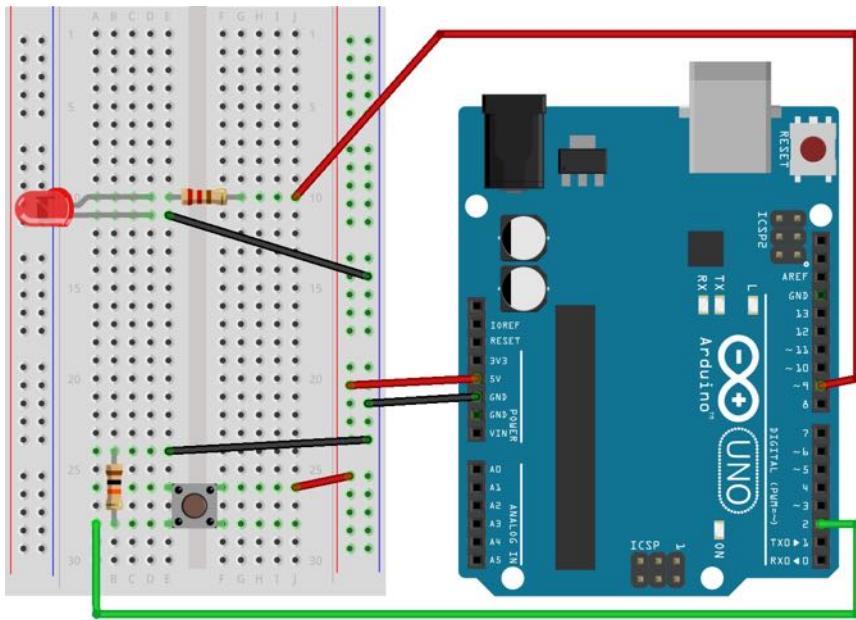
### ODPOVĚĎ

- ➔ Do funkce `loop` se přidá podmínka pro test stisknutého tlačítka.

```
void loop(){
    if(digitalRead(Button) == LOW){
        if(servoPos < 180){
            servoPos++;
        }
        myservo.write(servoPos);
        delay(delayPeriod);
    }
}
```



## ZADÁNÍ



```
int pinLed = 9;
int Button = 2;

void setup(){
    pinMode(Button, INPUT);
    pinMode(pinLed, OUTPUT);
}

void loop(){
    digitalWrite(pinLed, HIGH);
    delay(1000);
    digitalWrite(pinLed, LOW);
}
```

## OTÁZKA

➔ Jak upravíte programový kód, aby LED svítila pouze při stisknutém tlačítku?



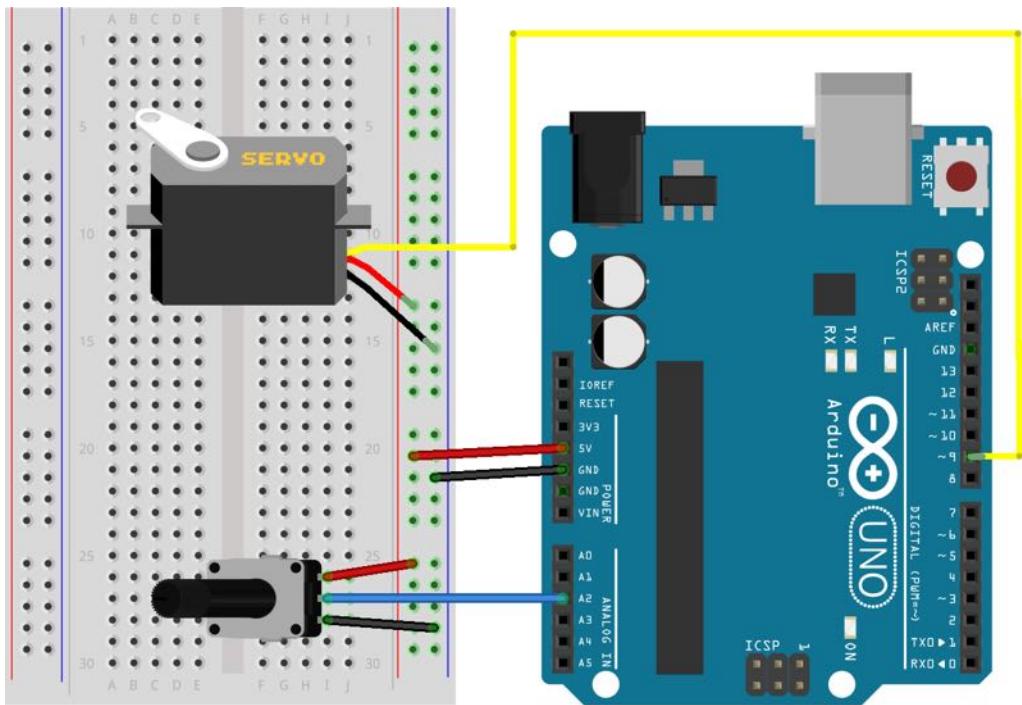
## ODPOVĚĎ

➔ Funkce `loop` se upraví tak, že se přidá podmínka pro test stisknutého tlačítka.



```
void loop(){
    if(digitalRead(Button) == LOW){
        digitalWrite(pinLed, HIGH);
    }else{
        digitalWrite(pinLed, LOW);
    }
}
```

## ZADÁNÍ



```
#include <Servo.h>

Servo myservo;
int pos = 0;

void setup()
{
  myservo.attach(9);
}

void loop()
{
  pos = analogRead(A2);
  pos = map(pos, W, X, Y, Z);
  myservo.write(pos);
  delay(5);
}
```



### OTÁZKA

→ Jaké hodnoty doplníte ve funkci `map` za písmena W, X, Y, Z, aby při otáčení potenciometru se natáčel i servomotor v celém svém rozsahu?



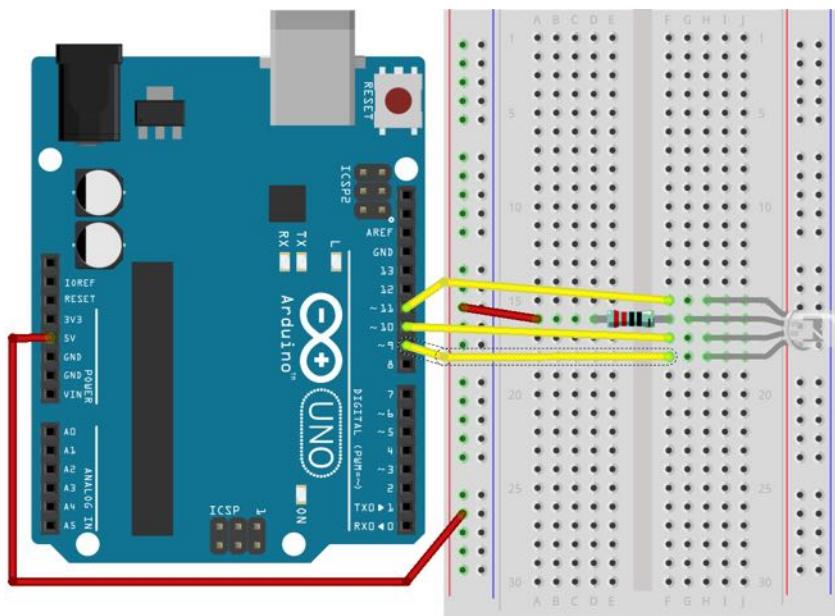
### ODPOVĚĎ

→ Parametry ve funkci `map` mohou vypadat následujícím způsobem.

```
pos = map(pos, 0, 1023, 0, 179);
```

# OTÁZKY RGB LED – VLASTNÍ FUNKCE

## ZADÁNÍ



```
void setup() {  
    pinMode(11, OUTPUT);          //červená  
    pinMode(10, OUTPUT);          //zelená  
    pinMode(9, OUTPUT);           //modrá  
}  
  
void loop() {  
    digitalWrite(11, X); //červená  
    digitalWrite(10, Y); //zelená  
    digitalWrite(9, Z); //modrá  
}
```



### OTÁZKA

- Jaké hodnoty doplníte ve funkcích digitalWrite za písmena X, Y, Z, aby dioda svítila modře?



### ODPOVĚĎ

- Pokud se jedná o RGB diodu se společnou anodou, tak řešení je následující.

```
void loop() {  
    digitalWrite(11, HIGH); //nebo místo HIGH je 255  
    digitalWrite(10, HIGH); //nebo místo HIGH je 255  
    digitalWrite(9, LOW); //nebo místo LOW je 0  
}
```



### OTÁZKA

- Jaké hodnoty doplníte ve funkcích digitalWrite za písmena X, Y, Z, aby dioda svítila zeleně?

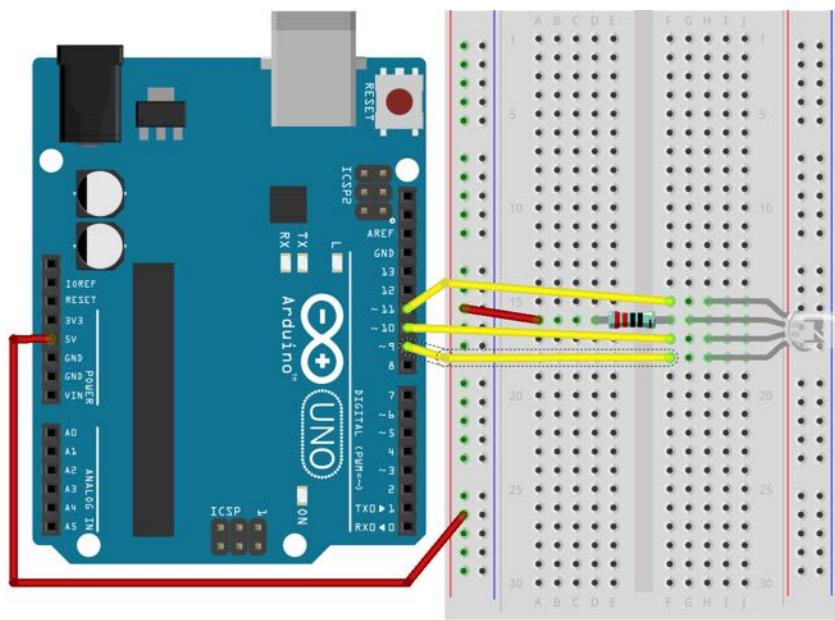


### ODPOVĚĎ

- Pokud se jedná o RGB diodu se společnou anodou, tak řešení je následující.

```
void loop() {  
    digitalWrite(11, HIGH); //nebo místo HIGH je 255  
    digitalWrite(10, LOW); //nebo místo HIGH je 0  
    digitalWrite(9, HIGH); //nebo místo LOW je 255  
}
```

## ZADÁNÍ



```
void setup() {
    pinMode(11, OUTPUT); //červená
    pinMode(10, OUTPUT); //zelená
    pinMode(9, OUTPUT); //modrá
}

void loop() {
    setColor(255,0,0);
}
```



### OTÁZKA

- Naprogramujte funkci `setColor` tak, aby se v uvedeném programu svítila RGB dioda definovanou barvou v zadaných parametrech volané funkce.

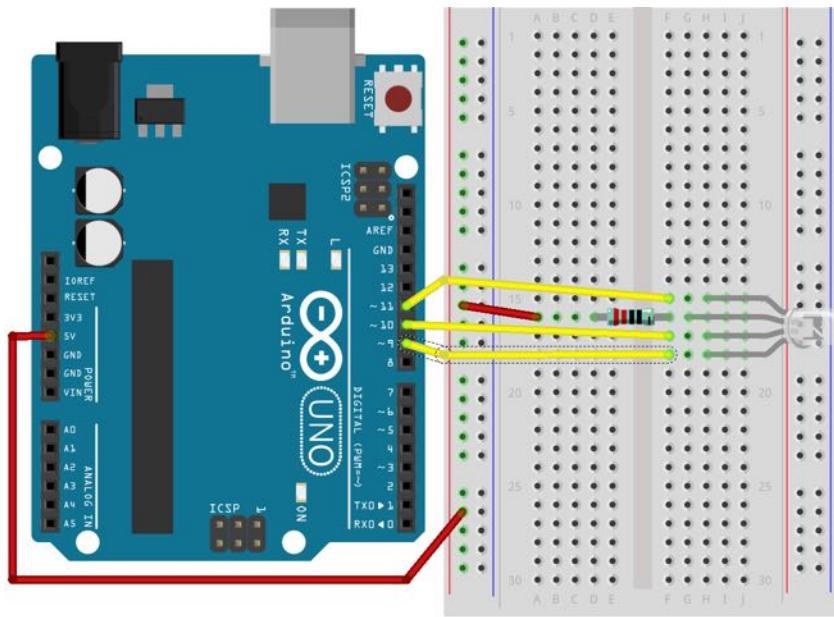


### ODPOVĚĎ

- Funkce `setColor` bude vypadat následovně.

```
void setColor(int redC, int greenC, int blueC ) {  
    digitalWrite(11, redC);  
    digitalWrite(10, greenC);  
    digitalWrite(9, blueC);  
}
```

## ZADÁNÍ



```
void setup() {
    pinMode(11, OUTPUT);          //červená
    pinMode(10, OUTPUT);          //zelená
    pinMode(9, OUTPUT);           //modrá
}

void loop() {
    setColor(255,0,0);
    delay(1000);
    setColor(0,0,255);
    delay(1000);
    setColor(0,250,0);
    delay(1000);
}
```



### OTÁZKA

- Naprogramujte funkci `setColor` tak, aby se v uvedeném programu střídaly barvy, podle zadaných parametrů.



### ODPOVĚĎ

- Funkce `setColor` bude vypadat následovně. Řešení pro diodu se společnou anodou.

```
void setColor(int redC, int greenC, int blueC) {  
    digitalWrite(11, redC);  
    digitalWrite(10, greenC);  
    digitalWrite(9, blueC);  
}
```



### OTÁZKA

- Jakými barvami bude RGB dioda svítit, při správně definované funkci `setColor`?

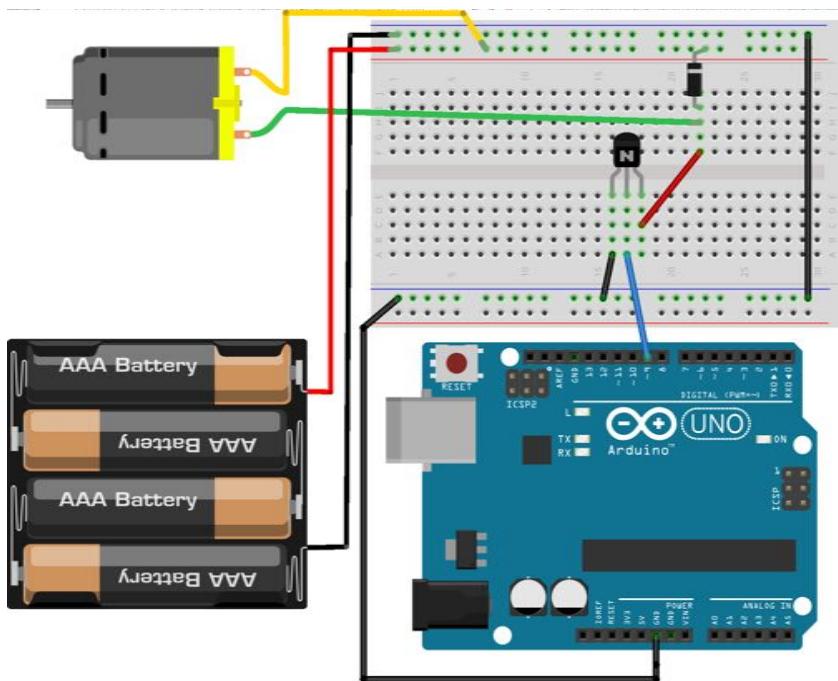


### ODPOVĚĎ

- Podle parametrů, které jsou ve funkci `setColor` bude RGB dioda postupně svítit barvami: tyrkysová, žlutá, fialová.

# MOTOR DC

## ZADÁNÍ



```
const int transistorPin = 9;
const int speedMotor = 200;

void setup() {
    pinMode(transistorPin, OUTPUT);
}

void loop() {
    analogWrite(transistorPin, speedMotor);
}
```

### OTÁZKA

- Jak upravíte program, aby se otáčky motoru postupně zrychlovaly od 0 do maximální rychlosti 255?



### ODPOVĚĎ

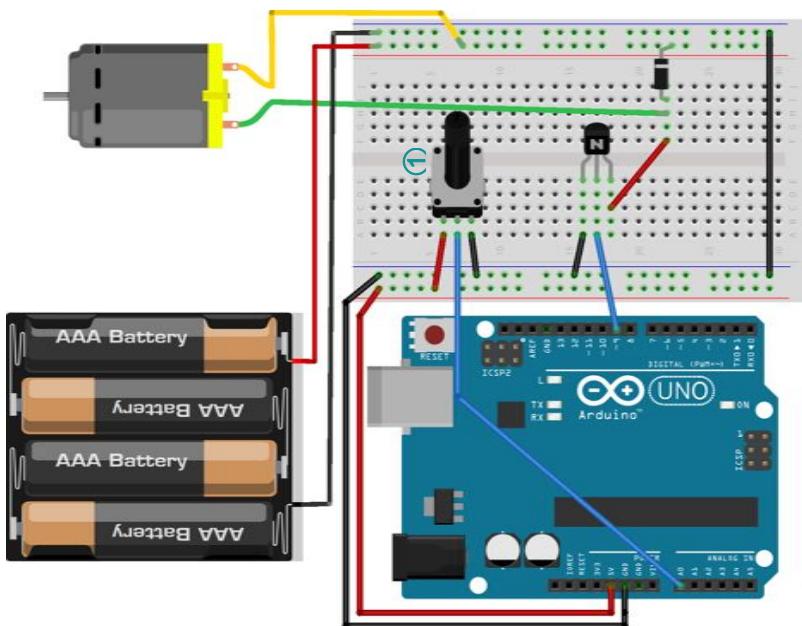
- Pro řešení lze použít příkaz cyklu `for`, který zajistí postupné zvyšování rychlosti v proměnné `speedMotor`.

A cartoon illustration of a person with brown hair, wearing a black suit and white shirt, holding a white circular object with a green brain icon on it.

```
void loop() {
    for(speedMotor = 0; speedMotor <= 255; speedMotor += 1){
        analogWrite(transistorPin, speedMotor);
    }
}
```

# MOTOR DC

## ZADÁNÍ



```
const int transistorPin = 9;

void setup() {
    pinMode(transistorPin, OUTPUT);
}

void loop() {
    int sensorValue = analogRead(A0);

    . . .

    analogWrite(transistorPin, outputValue);
}
```



### OTÁZKA

→ Jak upravíte program, aby se otáčky motoru regulovali pomocí připojeného potenciometru?



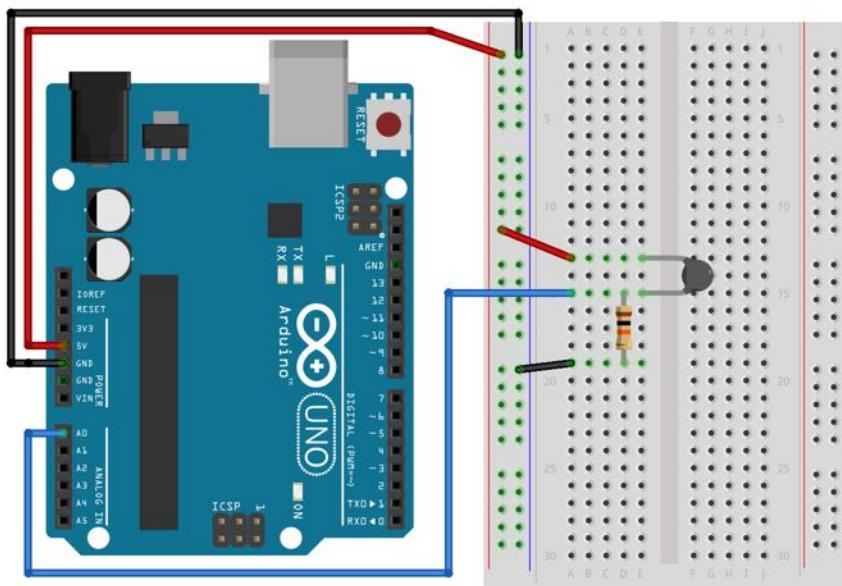
### ODPOVĚĎ

→ Pro řešení lze stačí doplnit funkci map, jejíž návratová hodnota se uložena do proměnné outputValue.

```
int outputValue = map(sensorValue, 0, 1023, 0, 255);
```

# MĚŘÍME TEPLITU

## ZADÁNÍ



```
int termistorPin = 0;
int Vout;
float R2 = 10000;
float logR2, R1, T;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04,
c3 = 2.019202697e-07;

void setup() {
    Serial.begin(9600);
}

void loop() {
    Vout = analogRead(termistorPin);
    R1 = R2 * (1023.0 / (float)Vout - 1.0);
    logR1 = log(R1);
    T = (1.0 / (c1 + c2*logR1 + c3*logR1*logR1));

    . . .

    delay(500);
}
```



### OTÁZKA

- Jak doplníte programový kód, abyste zjištěnou teplotu uloženo v proměnné T zobrazili pomocí sériového monitoru?



### ODPOVĚĎ

- Vzhledem k tomu, že ve funkci setup je inicializace pro sériovou komunikaci, měli by si žáci vzpomenout na funkce pro výpis v sériovém monitoru.

```
Serial.print("Teplota: ");
Serial.println(T);
```



### OTÁZKA

- Vytvořte dvě funkce, které budou zajišťovat výpočet teploty ve stupních Celsia a Fahreinhaita?

Výpočet stupňů Fahrenighthaita:  $(\text{Stupně celsia} * 9.0) / 5.0 + 32.0$   
Výpočet stupňů Celsia:  $T - 273.15$



### ODPOVĚĎ

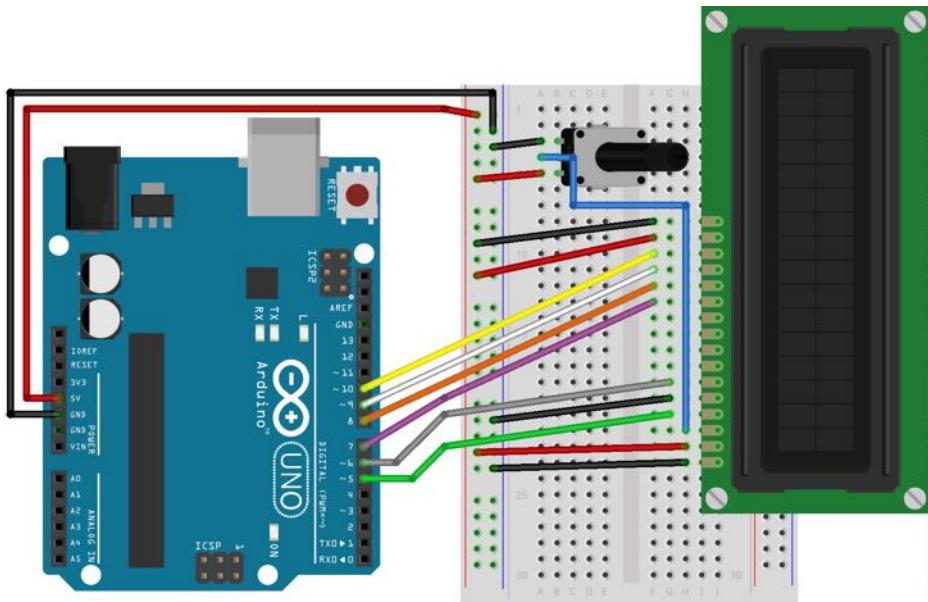
- Podle uvedených vzorců je výpočet pomocí funkcí velmi jednoduché.  
V první řadě se funkce musí deklarovat:

```
void getCelsius(T) {
    return T - 273.15;
}

void getFahrein(Tc) {
    return (Tc * 9.0) / 5.0 + 32.0;
}
```

Funkce se musí zavolat s odpovídajícími parametry:

## ZADÁNÍ



```
#include <LiquidCrystal.h>

int rsPin = 5;
int ePin = 6;
int d4Pin = 7;
int d5Pin = 8;
int d6Pin = 9;
int d7Pin = 10;
LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);

void setup(){
}

void loop(){}
```

### OTÁZKA

→ Jak doplníte programový kód, aby se na prvním řádku displeje zobrazilo slovo Dobrý a na druhém řádku den?



### ODPOVĚĎ

→ Pro zobrazení požadovaného textu lze využít metod z instance třídy LCD.

```
void setup(){
    LCD.begin(16,2);          // inicializace displeje
    LCD.clear();               // vymazání displeje
    LCD.setCursor(0,0);        // nastavení začátku kurzoru
    LCD.print("Dobrý");        // vypsání textu

    LCD.setCursor(0,1);        // nastavení kurzoru druhý řádek
    LCD.print("den");          // vypsání textu
}
```





### OTÁZKA

→ Jak upravíte programový kód, aby se na LCD displeji zobrazovalo počítadlo od 1 do 100? Po dosáhnutí hodnoty 100 se počítadlo vynuluje a začne opět od 1. Využijte podmínkový příkaz `if`.

### ODPOVĚĎ

→ Při využití podmínkového příkazu `if`, stačí zadat do funkce `loop` inkrementátor v podobě proměnné `i`. Upravený kód bude vypadat následovně.

```
int i = 0;

void setup(){
    LCD.begin(16,2);          // inicializace displeje
}

void loop(){
    LCD.clear();              // vymazání displeje
    LCD.setCursor(0,0);       // nastavení začátku kurzoru
    LCD.print(i);             // vypsání čísla
    delay(100);

    if(i==100){
        i=1;
    }else{
        i++;
    }
}
```





### OTÁZKA

- Jak upravíte programový kód, aby se na LCD displeji zobrazovalo počítadlo od 1 do 100? Po dosáhnutí hodnoty 100 se počítadlo vynuluje a začne opět od 1. Využijte příkaz cyklu **for**.

### ODPOVĚĎ

- Při využití příkazu cyklu **for**, stačí zadat vypisovat hodnotu proměnné **i**. Upravený kód bude vypadat následovně.

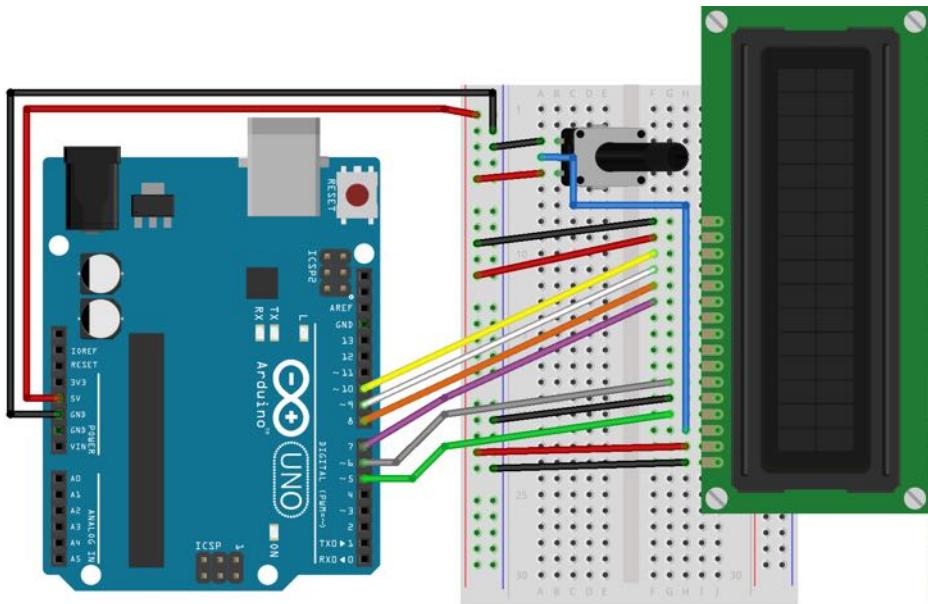
```
int i = 0;

void setup(){
    LCD.begin(16,2);          // inicializace displeje
}

void loop(){
    for(i = 0; i <= 100; i++){
        LCD.clear();           // vymazání displeje
        LCD.setCursor(0,0);    // nastavení začátku kurzoru
        LCD.print(i);          // vypsání čísla
        delay(100);
    }
    delay(5000);
}
```



## ZADÁNÍ



```
#include <LiquidCrystal.h>

int rsPin = 5;
int ePin = 6;
int d4Pin = 7;
int d5Pin = 8;
int d6Pin = 9;
int d7Pin = 10;
LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);

void setup(){
    LCD.begin(16,2);           // inicializace displeje
}

void loop(){
    for(i = 0; i <= 100; i++){
        LCD.clear();          // vymazání displeje
        LCD.setCursor(0,0);   // nastavení začátku kurzoru
        LCD.print(i);         // vypsání čísla
        delay(100);
    }
    delay(5000);
}
```

## OTÁZKA

→ Jak upravíte programový kód, aby se na LCD displeji zobrazovalo kromě počítadla od 1 do 100 i slovo Pocitadlo?



## ODPOVĚĎ

→ Do programového kódu se musí přidat nové nastavení pozice kurzoru. Jedno nastavení bude pro text Pocitadlo a druhé pro číslo v proměnné i.



```
int i = 0;

void setup(){
    LCD.begin(16,2);                  // inicializace displeje
}

void loop(){
    for(i = 0; i <= 100; i++){
        LCD.clear();                  // vymazání displeje

        LCD.setCursor(0,0);          // nastavení začátku kurzoru
        LCD.print("Pocitadlo");      // vypsání slova Pocitadlo

        LCD.setCursor(10,0);         // nastavení začátku kurzoru
        LCD.print(i);                // vypsání čísla
        delay(100);
    }
    delay(5000);
}
```

### OTÁZKA

- Jak upravíte programový kód, aby se na LCD displeji zobrazovalo počítadlo od 100 do 1?



### ODPOVĚĎ

- V cyklu `for` stačí provést obrácený odpočet proměnné `i`.



```
int i = 0;  
  
void setup(){  
    LCD.begin(16,2); // inicializace displeje  
}  
  
void loop(){  
    for(i = 100; i >= 1; i--){ // dekrementování proměnné i  
        LCD.clear(); // vymazání displeje  
        LCD.setCursor(0,0); // nastavení začátku kurzoru  
        LCD.print(i); // vypsání čísla  
        delay(100);  
    }  
    delay(5000);  
}
```

# PROGRAMOVACÍ JAZYK WIRING

REFERENČNÍ PŘÍRUČKA PROGRAMOVACÍHO JAZYKA WIRING, KTERÝ PŘEDSTAVUJE ZÁKLADNÍ RÁMEC PRO PROGRAMOVÁNÍ MIKROKONTROLÉRŮ BEZ SPECIFICKÝCH ZNALOSTÍ HARDWARU.

Programovací jazyk pro Arduino je odvozen z programovacího jazyka Wiring a je založen na klasickém programovacím jazyku C++, jehož základy položil Bjarne Stroustrup v roce 1983. Jedná se o relativně jednoduchý jazyk, jehož funkce i operátory (např. typu +, -, \*, /) jsou intuitivní a snadno pochopitelné. Pro navrhování aplikací pro Arduino byl jazyk navíc dále zjednodušen vynecháním relativně komplikované syntaxe objektového programování.

## STRUKTURA PROGRAMU

Základní struktura programovacího jazyka Arduino je poměrně jednoduchá a skládá se nejméně ze dvou částí, přesněji funkcí. Bloky příkazů v těle těchto dvou funkcí jsou ohraničeny složenými závorkami.

```
void setup()
{
    příkazy;
}

void loop()
{
    příkazy;
}
```

Funkce `setup()` je přípravná a provádí se jen jednou na začátku programu, funkce `loop()` je výkonná a provádí se neustále dokola. Pro správnou činnost programu je vždy nutné použít obě tyto funkce.

Funkce `setup()` by měla být volána až po deklaraci všech proměnných na začátku programu. Tato funkce se používá například k nastavení pinů Arduina na vstup nebo výstup, nastavení parametrů sériové komunikace a podobných jednorázových akcí.

Po funkci `setup()` následuje funkce `loop()`. Tělo této funkce obsahuje programový kód, který bude opakovaně prováděn v nekonečné smyčce, například čtení vstupů, nastavování

výstupů, výpočty atd. Tato funkce je jádrem všech programů Arduina a vykonává většinu činností.

## SETUP()

Funkce **setup()**, jak již bylo řečeno výše, se volá pouze jednou při spuštění programu. Používá se k inicializaci režimu jednotlivých pinů, k nastavení sériové komunikace apod. Tato funkce musí být v programu obsažena vždy, i když žádné inicializační příkazy neobsahuje.

```
void setup()
{
    pinMode(pin, OUTPUT); // nastav 'pin' na výstup
}
```

## LOOP()

Po dokončení funkce **setup()** se začne neustále dokola provádět funkce **loop()**, jak její název (loop =smyčka) ostatně napovídá. Příkazy, obsažené v těle této funkce, jsou určeny k provádění veškeré činnosti Arduina.

```
void loop()
{
    digitalWrite(pin, HIGH);      // nastav 'pin' na log. 1
    delay(1000);                // čekej jednu sekundu
    digitalWrite(pin, LOW);      // nastav 'pin' na log. 0
    delay(1000);                // čekej jednu sekundu
}
```

## SYNTAXE

### SLOŽENÉ ZÁVORKY { }

Složené závorky definují začátek a konec bloku kódu. Používají se ve funkcích i ve smyčkách.

```
type function()
{
    příkazy;
}
```

Za úvodní složenou závorkou [ { } ] musí vždy následovat závorka uzavírací [ } ]. Proto se často uvádí, že složené závorky musí být párovány. Samostatně umístěné závorky (úvodní bez uzavírací a naopak) mohou často vést k záhadným, špatně dohledatelným chybám kompilátoru.

Programové prostředí Arduino obsahuje praktickou funkci pro kontrolu párování složených závorek. Stačí vybrat závorku nebo kliknout myší bezprostředně pod závorku a související závorka bude zvýrazněna.

### ; STŘEDNÍK

Středníkem musí být ukončena deklarace i jednotlivé prvky programu. Středník je také používán k oddělení prvků ve smyčce.

```
int x = 13; // deklaruje proměnnou 'x' jako datový typ integer s
// hodnotou 13
```



Pokud zapomenete řádek programu středníkem ukončit, dojde k chybě kompilátoru. Z chybového hlášení může být zřejmé, že se jedná o zapomenutý středník, ovšem také nemusí. Pokud se objeví hlášení o záhadné nebo zdánlivě nelogické chybě kompilátoru, zkонтrolujte nejprve, zda v zápisu programu nechybí středník v blízkosti místa, kde kompilátor chybu ohlásil.

## **/\*... \*/ BLOKOVÉ KOMENTÁŘE**

Blokové komentáře nebo víceřádkové komentáře jsou oblasti textu, které jsou programem ignorovány. Jsou používány pro obsažnější komentování kódu nebo poznámky, které pomohou pochopit ostatním význam částí programu. Blokové komentáře začínají `/ *` a končí `*/` a mohou obsahovat více řádků textu.

```
/*
Toto je blokový komentář, nezapomeňte ho ukončit.
Znaky pro jeho začátek a konec musí být vždy v páru!
*/
```

Vzhledem k tomu, že komentáře jsou programem ignorovány, nezabírají žádný paměťový prostor; mohou tedy být hojně používány. Mohou být také použity k dočasnému znefunkčnění celých bloků kódu programu pro účely ladění.



Do blokového komentáře je možno vložit i jednořádkové komentáře (uvozené `//`), ale není možno do blokového komentáře vložit další blokový komentář.

## **// JEDNOŘÁDKOVÉ KOMENTÁŘE**

Jednotlivé řádky komentáře musí začínat `//` a končí na konci řádku. Stejně jako blokové komentáře jsou jednořádkové komentáře programem ignorovány a nezabírají žádný paměťový prostor.

```
// toto je jednořádkový komentář
```

Jednořádkové komentáře jsou často používány za příkazy k vysvětlení jejich funkce nebo jako poznámka pro další použití.

## PROMĚNNÉ

Proměnná je způsob pojmenování a uložení číselné hodnoty pro pozdější použití v programu. Jak jejich název naznačuje, proměnné jsou čísla, jejichž hodnota může být průběžně změněna, na rozdíl od konstant, jejichž hodnota se nikdy nemění. Proměnné musí být deklarovány a volitelně jim lze přiřadit hodnoty, které do nich mají být uloženy.

Následující kód deklaruje proměnnou názvem **inputVariable** a přiřadí jí hodnotu získanou čtením analogové hodnoty vstupního pinu 2:

```
int inputVariable = 0;           // deklaruje proměnnou  
                                // a přiřadí jí hodnotu 0  
inputVariable = analogRead(2);   // přiřadí proměnné hodnotu  
                                // podle analogové hodnoty pinu 2
```

Proměnnou pojmenujeme **InputVariable**. První řádek kódu deklaruje, že proměnná bude obsahovat datový typ **int** (zkratka pro název integer) a nastaví její hodnotu na 0. Druhý řádek nastaví hodnotu proměnné podle hodnoty analogového 2, která je dostupná jinde v programovém kódu.

Jakmile byla proměnné přiřazena nebo změněna hodnota, můžete použít její hodnotu přímo nebo testovat, zda tato hodnota splňuje určité podmínky.

Příklad ukazuje tři užitečné operace s proměnnými. Následující kód testuje, zda hodnota proměnné **inputVariable** je menší než 100, je-li to pravda, pak proměnné **inputVariable** přiřadí hodnotu 100, a pak nastaví na základě hodnoty proměnné **inputVariable** prodlevu, která je nyní minimálně 100:

```
if(inputVariable < 100)          // testuje, zda je hodnota proměnné méně než 100  
{  
    inputVariable = 100;         // je-li to pravda, je jí přiřazena hodnota 100  
}  
delay(inputVariable);           // použije hodnotu proměnné jako parametr funkce  
                                // delay
```



Proměnným bychom měli pro lepší orientaci v programovém kódu dávat popisné názvy. Jména proměnných, jako je **tiltSensor** nebo **pushButton** pomáhají programátorovi i komukoli jinému při čtení kódu snadněji poznat, co proměnná znamená. V našich příkladech ale naopak používáme krátká jména proměnných (například **var** nebo **value**), aby byl kód kratší a přehlednější. Proměnná může být pojmenována libovolným jménem, které nepatří mezi klíčová slova v jazyce Arduino.

## DEKLARACE PROMĚNNÝCH

Všechny proměnné musí být deklarovány ještě před jejich prvním použitím. Deklarace proměnné znamená, že definujete její typ (**int**, **long**, **float**, atd.), nastavíte její jméno, a případně přiřadíte počáteční hodnotu. Tyto deklarace postačí v programu provést jen jednou, hodnotu proměnné ale můžete v programu kdykoliv změnit.

Následující příklad deklaruje, že proměnná **inputVariable** je typu **int** neboli **integer**, a že její počáteční hodnota je rovna nule. To se nazývá jednoduché přiřazení.

```
int inputVariable = 0;
```

Proměnná může být deklarována kdekoli v programu a je použitelná od místa deklarace dále.

## PLATNOST PROMĚNNÝCH

Proměnná může být deklarována na začátku programu, ještě před funkcí **void setup()**, lokálně uvnitř funkce, a někdy v deklaraci bloku, jako je tomu u smyček. Místo, kde je proměnná deklarována, určuje její použitelnost pro některé části programu.

Globální proměnná je ta, ke které mají přístup a kterou mohou používat všechny funkce a deklarace v programu. Taková proměnná musí být deklarována na začátku programu, ještě před funkcí **void setup()**.

Lokální proměnná je ta, která je definována uvnitř funkce nebo jako součást smyčky. Je dostupná a může být použita pouze uvnitř funkce, uvnitř které byla deklarována. Je tedy

možné mít v různých částech téhož programu umístěné dvě nebo více proměnné stejného jména, které obsahují různé hodnoty. To, že přístup ke svým lokálním proměnným má pouze daná funkce, zjednodušuje program a snižuje možnost programových chyb.

Následující příklad ukazuje, jakým způsobem můžeme deklarovat různé typy proměnných a předvádí viditelnost každé z proměnných v programu:

```
int value;           // proměnná 'value' je viditelná
                     // pro všechny funkce
void setup()        // funkce neobsahuje žádné inicializační příkazy
{
}
void loop()
{
    for(int i=0; i<20;) // proměnná 'i' je viditelná
    {                   // jen uvnitř této smyčky
        i++;
    }
    float f;           // proměnná 'f' je viditelná
}                     // jen uvnitř této smyčky
```

## KONSTANTY

Jazyk Arduino má několik předdefinovaných hodnot, které se nazývají konstanty. Ty se používají k tvorbě přehlednějších programů. Konstanty jsou seřazeny do skupin.

## TRUE / FALSE

Jedná se o logické konstanty, které definují logické úrovně. FALSE je jednoduše definována jako 0 (nula), TRUE je často definována jako 1, ale může nabývat každou hodnotu kromě 0. V tomto smyslu je výsledek operace Boolean -1, 2 a -200 také definován jako TRUE.

```
if(b == TRUE)
{
    příkazy;
}
```

## HIGH / LOW

Tyto konstanty definují logickou úroveň pinů Arduina jako vysokou nebo nízkou a jsou používány při čtení nebo zápisu této hodnoty na digitální piny.

HIGH je definována jako logická úroveň 1, ON, nebo 5 voltů, zatímco LOW je logická úroveň 0, OFF nebo 0 voltů.

```
digitalWrite(13, HIGH);
```

## INPUT / OUTPUT

Konstanty používané ve funkci `pinMode()` pro přepnutí funkce digitálního pinu na vstup (INPUT) nebo výstup (OUTPUT).

```
pinMode(13, OUTPUT);
```

## DATOVÉ TYPY

### BYTE

Datový typ byte ukládá hodnoty jako 8-bitové číselné hodnoty bez desetinných míst. Ukládá hodnotu v rozsahu 0-255.

```
byte someVariable = 180;      // deklaruje proměnnou 'someVariable'  
                             // jako datový typ byte
```

### INT

Celočíselný datový typ **integer** je nejběžnějším způsobem ukládání celých čísel. Ukládá hodnotu jako 16-bitovou v rozsahu -32 768 až 32 767.

```
int someVariable = 1500;      // deklaruje proměnnou 'someVariable'  
                            // jako datový typ integer
```



Celočíselná proměnná přeteče (podteče), pokud dojde k překročení její maximální (minimální) hodnoty. Například, je-li  $x = 32\ 767$  a následná operace zvětší hodnotu o 1, (například  $x = x + 1$  nebo  $x++$ ), hodnota  $x$  je po přetečení rovna -32 768.

### LONG

Datový typ **long** (integer) je určen pro velká čísla bez desetinných míst. Ukládá hodnotu jako 32-bitovou v rozsahu -2 147 483 648 až 2 147 483 647.

```
long someVariable = 90000;    // deklaruje proměnnou 'someVariable'  
                           // jako datový typ long
```

## FLOAT

Datový typ **float** je určen pro operace s čísly s plovoucí desetinnou čárkou. Čísla qs plovoucí desetinnou čárkou mají větší rozlišení než celá čísla a jsou uložena jako 32-bitové hodnoty s rozsahem -3,4028235E38 k 3,4028235E38.

```
float someVariable = 90000; // deklaruje proměnnou 'someVariable'  
// jako datový typ float
```



Čísla s plovoucí desetinnou čárkou mají omezenou přesnost, díky níž může dojít k problémům při porovnávání dvou takových čísel. Matematické operace s plovoucí desetinnou čárkou jsou také mnohem pomalejší než celočíselná aritmetika a je vhodné – pokud je to možné – se jím vyhýbat.

## POLE

Polem se nazývá množina prvků, ke kterým je možno přistupovat prostřednictvím indexace. Libovolná hodnota v poli může být zpřístupněna uvedením názvu pole a indexu hodnoty. Pole jsou indexována od nuly, přičemž první hodnota v poli má index 0. Pole musí být deklarována a je jím možno přiřadit hodnoty dříve, než je můžeme použít.

```
int myArray[ ] = {hodnota0, hodnota1, hodnota2. ...}
```

Stejně tak je možné deklarovat pole tak, že mu určíme datový typ a velikost a později mu přiřadíme hodnoty na pozice dané indexem:

```
int myArray[5]; // deklaruje pole typu integer se šesti prvky  
myArray[3] = 10; // přiřadí čtvrtému prvku pole hodnotu 10
```

Chceme-li načíst do proměnné hodnotu z pole, přiřadíme proměnné název pole a index pozice:

```
x=myArray[3]; // 'x' se nyní rovná 10
```

Pole jsou často používána v cyklech **for**, kde je přírůstek čítače je zároveň použit jako index pozice pro každou hodnotu pole.

V následujícím příkladu definujeme pole pro blikání LED. Použijeme smyčku, jejíž čítač začíná na 0. Hodnota indexu na pozici 0 z pole **flicker[]** se zapíše do proměnné **ledPin** (v tomto případě 180). Na pinu 10, který je nastaven do analogového módu se objeví PWM modulace s parametry, určenými proměnnou **ledPin**, běh programu se pozastaví na 200 ms. Pak se smyčka znova opakuje, vybere se další prvek pole podle hodnoty proměnné čítače atd.

```
int ledPin = 10;           // LED na pinu 10
byte flicker[] = {180, 30, 255, 200, 10, 90, 150, 60} ;
                        // vytvořeno pole s osmi prvky různých hodnot
void setup()
{
    pinMode(ledPin, OUTPUT); // nastav pin jako VÝSTUP
}

void loop()
{
    for(int i=0; i<8; i++) // smyčka pokračuje, dokud se
                            // proměnná 'i' nerovná počtu prvků
                            // v poli
    {
        analogWrite(ledPin, flicker [i]); // zapiš do proměnné hodnotu indexu
                                         // z pole
        delay(200); // pauza 200 ms
    }
}
```

## ARITMETICKÉ OPERACE

Aritmetické operátory jsou: sčítání, odčítání, násobení a dělení.

Vrací součet, rozdíl, součin nebo podíl (v uvedeném pořadí) dvou operandů.

```
y = y + 3;  
x = x - 7;  
i = j * 6;  
r = r / 5;
```

Výsledek aritmetické operace je stejného datového typu jako operandy. Například 9/4 poskytne výsledek 2 namísto 2.25, protože operandy 9 a 4 jsou typu int, který není schopen vyjádřit číslice za desetinnou čárkou. To také znamená, že u aritmetické operace může dojít k přetečení, pokud výsledek je větší než maximum, které může být uloženo v použitém datovém typu.

Jsou-li operandy různých typů, je pro výpočet použit vždy větší typ. Například, jestliže jeden z operandů je typu **float** a druhý typu **int**, bude vypočet probíhat v typu **float**.

Pro své proměnné zvolte vždy takový datový typ, který dokáže uložit dostatečně velká čísla jako výsledky vašich výpočtů. Mějte na paměti, v kterém okamžiku vaše proměnná přeteče a také to, co se stane při přechodu v opačném směru, např. (0 -1) nebo (0 -32768). Pro matematiku, která vyžaduje zlomky, zvolte proměnné typu **float**, ale buďte si vědomi jejich nedostatků: zabírají více místa v paměti a výpočet je pomalejší.



Pomocí operátoru přetypování např. (**int**) **myFloat** lze převést jeden typ proměnné na jiný aniž by bylo nutné ukládat přetypovanou hodnotu do proměnné. Například, **i = (int) 3.6** nastaví i rovno 3.

## SLOŽENÉ PŘIŘAZENÍ

Složené přiřazení kombinuje aritmetické operace s přiřazením hodnoty proměnné. Tyto operace se běžně vyskytují v cyklech **for**, jak je popsáno dále.

Mezi nejčastější složená přiřazení patří:

```
x++ // je totéž jako x=x+1, tedy zvětšení hodnoty x o +1  
x-- // je totéž jako x=x-1, tedy zmenšení hodnoty x o -1  
x+=y // je totéž jako x=x+y, tedy zvětšení hodnoty x o hodnotu +y  
x-=y // je totéž jako x=x-y, tedy zmenšení hodnoty x o hodnotu - y  
x*=y // je totéž jako x=x*y, tedy vynásobení x y  
x/=y // je totéž jako x=x/y, tedy podělení x y
```



Například operace `x *= 3` ztrojnásobí původní hodnotu x a znovu jí přiřadí výsledné hodnotě x.

## RELAČNÍ OPERÁTORY

Porovnání jedné proměnné nebo konstanty s jinou se často používá v testech uvnitř příkazu `if`, kdy se porovnává, zda zadaná podmínka platí. Příklady naleznete na následujících stránkách.

Znak `??` zastupuje v dalším textu některou z následujících podmínek:

```
x==y // x je rovno y  
x!=y // x není rovno y  
x<y // x je menší než y  
x>y // x je větší než y  
x<=y // x je menší nebo rovno y  
x>=y // x je větší nebo rovno y
```

## LOGICKÉ OPERÁTORY

Logické operátory jsou obvyklým způsobem porovnávání dvou výrazů. Tyto operátory vrací hodnotu TRUE nebo FALSE v závislosti na typu operace.

K dispozici jsou tři logické operátory AND, OR a NOT, které jsou často používané v testu `if`:

### LOGICKÉ AND:

```
if (x>0 && x<5) // Vrací hodnotu TRUE, když oba  
                  // výrazy mají hodnotu TRUE
```

### LOGICKÉ OR:

```
if (x>0 || y>0) // Vrací hodnotu TRUE, když kterýkoli  
// z prvků má hodnotu TRUE
```

### LOGICKÉ NOT:

```
if (!x) // má za následek true jestliže operand je  
// false a naopak
```

## ŘÍZENÍ TOKU PROGRAMU

### IF

Příkaz **if** testuje, zda bylo dosaženo určité podmínky, třeba zda analogová hodnota je větší než zadaná, a provádí všechny příkazy uvnitř závorek, pokud tvrzení je pravdivé (TRUE).

Pokud tvrzení pravdivé není (FALSE), program příkazy uvnitř závorek přeskočí.

Formát příkazu **if** je:

```
if(someVariable ?? value)  
{  
    příkazy;  
}
```

Výše uvedený příklad porovnává hodnotu proměnné **someVariable** s jinou hodnotou, kterou může být opět buď proměnná nebo konstanta. Pokud je výsledek porovnání hodnot v závorce pravda (TRUE), jsou vykonány příkazy uvnitř složených závorek. Pokud ne, program je přeskočí a pokračuje za nimi.



Dejte si pozor na náhodné použití „=“ místo „==“ v příkazu **if(x = 10)**, zápis je syntakticky správný, ale nastavuje hodnotu proměnné x na hodnotu 10 a výsledkem je tedy vždy pro nenulovou hodnotu pravda (TRUE). Místo „=“ je nutno použít výraz „==“, tedy **(x == 10)**, který jen testuje, zda x se rovná hodnotě 10, nebo ne. Myslete na „=“ jako na "rovná se" na rozdíl od „==“, které znamená „se

## IF .. ELSE

Operace **if..else** umožňuje rozhodování stylem „buď – nebo“ a větvení programu podle výsledku operace. Například, pokud chcete otestovat stav digitálního vstupu a pak provést nějakou činnost, pokud je vstupní pin ve stavu HIGH nebo naopak provést něco jiného, pokud je vstupní úroveň nízká, můžete to zapsat tímto způsobem:

```
if(inputPin == HIGH)
{
    doThingA;
}
else
{
    doThingB;
}
```

nebo může také předcházet jiné **if**, pokud testujeme více vzájemně se vylučujících podmínek. Testy lze spustit současně. Je dokonce možné mít neomezený počet těchto větvení **else if**. Pamatujte si však, že je možné vždy spustit v závislosti na podmínkách testů jen jednu část kódu:

```
if(inputPin < 500)
{
    doThingA;
}
else if(inputPin >= 1000)
{
    doThingB;
}
else
{
    doThingC;
}
```



Příkaz **if** pouze testuje, zda je podmínka v závorce pravdivá nebo nepravdivá. Touto podmínkou může být jakýkoli platný příkaz jazyka C jako například, **if (inputPin == HIGH)**. V tomto příkladu 'if' pouze zkонтroluje, zda opravdu má zadáný vstup logickou úroveň HIGH neboli 5 V.

## FOR

Příkaz **for** se používá k několikanásobnému opakování bloku příkazů, uzavřených do složených závorek. Záhlaví smyčky se skládá ze tří částí, oddělených středníkem (:):

```
for(inicializace; podminka; výraz)
{
    příkazy;
}
```

Nejprve je deklarována lokální proměnná **inicializace** ve funkci přírůstkového čítače. Tato proměnná se deklaruje jen jednou. Za deklarací proměnné následuje podminka, která se testuje při každém průchodu smyčkou. Dokud je podminka pravdivá, jsou provedeny příkazy a výrazy uzavřené ve složených závorkách a výraz, která je součástí hlavičky for. Když se podmínka stane nepravdivou, smyčka se ukončí.

Následující příklad testuje hodnotu proměnné **i** typu integer, nastavené na 0 a zjišťuje, zda je její hodnota stále menší než 20. Je-li to pravda, provede se tělo cyklu a zvětší se hodnota proměnné **i** o 1:

```
for(int i=0; i<20; i++)      // deklaruje proměnnou 'i', testuje
                                // zda je menší než 20 a pokud
                                // ne, zvětší j i o 1
{
    digitalWrite(13, HIGH);    // nastaví pin 13 na HIGH
    delay(250);              // čeká 250 ms
    digitalWrite(13, LOW) ;   // nastaví pin 13 na LOW
    delay(250);              // čeká 250 ms
}
```



Programovací jazyk C má smyčky mnohem flexibilnější, než jsou podobné smyčky v jiných programovacích jazycích, včetně BASICu. Některé z prvků záhlaví nebo i všechny tři mohou být vynechány, středníky ale zůstávají povinné. Také pro inicializaci, podmínu i výraz může být použit jakýkoli platný příkaz jazyka C s nezávislými proměnnými. Tako definované záhlaví příkazu **if** může nabídnout řešení některých neobvyklých programových problémů.

## WHILE

Smyčka **while** se bude nekonečně cyklicky opakovat, dokud výraz uvnitř závorek nebude nepravdivý (FALSE). Pokud se hodnota testované proměnné nezmění, program smyčku **while** neopustí. Změnou může být ve vašem programu například inkrementace proměnné nebo změna externí podmínky, jakou může být výsledek testu senzoru.

```
while (someVariable ?? value)
{
    příkazy;
}
```

Následující příklad testuje, zda je hodnota proměnné **someVariable** menší než 200 a je-li to pravda, provede příkazy uzavřené mezi v závorkami. Tak se bude tato smyčka opakovat až do doby, kdy proměnná **someVariable** nabyde hodnoty 200 nebo větší.

```
while (someVariable < 200) // test, zda je proměnná menší než 200
{
    příkazy;           // vykonání příkazů
    someVariable++;   // zvětšení proměnné o 1
}
```

## DO .. WHILE

Tato smyčka pracuje stejně jako smyčka **while**, jen s tím rozdílem, že podmínka je testována nikoli na začátku, ale až na konci smyčky, takže smyčka proběhne vždy nejméně jednou.

```
do
{
    příkazy;
}
while (someVariable ?? value);
```

Následující příklad přiřadí funkci `readSensors()` hodnotu proměnné `x`, zastaví činnost na 50 milisekund, poté bude smyčku opakovat, dokud nebude `x` menší než 100:

```
do
{
    x = readSensors();      // přiřadí funkci readSensors()
                           // hodnotu proměnné 'x'
    delay(50);             // čekej 50 milisekund
} while (x < 100);        // znovu na začátek dokud 'x'
                           // není menší než 100
```

## FUNKCE

Funkce je část kódu, který je pojmenován a který ve svém těle obsahuje blok příkazů, které jsou provedeny při spuštění této funkce. Činnosti funkcí `void setup()` a `void loop()` již byly popsány výše a další vestavěné funkce budou popsány dále.

Vlastní funkce mohou vykonávat opakující se úkony a slouží také k udržení přehlednosti programu. Funkce musí být před použitím nejprve deklarovány čili musí být uvedeno, jakého jsou typu. To je datový typ hodnoty, kterou bude funkce vracet jako výsledek své činnosti, například `int` (zkratka pro integer), pokud funkce vrací celočíselnou hodnotu. Pokud se nemá vracet žádná hodnota, deklarujeme funkci typu `void(prázdná)`. Po uvedení typu následuje název funkce a v závorce seznam parametrů, předávaných funkci.

```
type functionName(parameters)
{
    příkazy;
}
```

Následující funkce `delayVal()` typu `integer` slouží v programu k nastavení velikosti zpoždění čtením hodnoty napětí z potenciometru. Nejprve deklarujeme lokální proměnnou `v`, vložíme do ní hodnotu přečtenou z potenciometru, která může nabývat hodnoty v rozsahu mezi 0 a 1023, pak se využije funkce `map`, abychom získali číslo v rozsahu mezi 0 a 255, a nakonec vrátíme tuto hodnotu zpět do hlavního programu.

```
int delayVal( )
{
    int v;                      // vytvoř í dočasnou proměnnou 'v'
    v = analogRead(pot);        // přečte hodnotu potenciometru
    v = map(v, 0, 1023, 0, 255); // konvertuje číslo z rozsahu 0 až
                                   // 1023 ma 0 - 255
    return v;                   // vrátí finální hodnotu 'v'
```

## DIGITÁLNÍ VSTUPY A VÝSTUPY

### PINMODE(PIN, MODE)

Tato konstanta se používá ve funkci `void setup()` a nastavuje určitý pin na vstup (INPUT) nebo výstup (OUTPUT).

```
pinMode(pin, OUTPUT); // nastav 'pin' jako výstupní
```

Digitální piny Arduino jsou standardně nastaveny jako vstupní, takže je není nutno do tohoto stavu znovu nastavovat pomocí funkce `pinMode()`. Piny nakonfigurované jako vstupní jsou ve stavu vysoké impedance.

Na piny ale mohou být připojeny zdvihací (pull-up) rezistory o velikosti 20 až 50 k $\Omega$ , které jsou obsaženy v interní struktuře mikrokontroléru. Ve výchozím nastavení mikrokontroléru jsou tyto zdvihací rezistory odpojeny, ale je možno je programově připojit.

Provádí se to následujícím způsobem:

```
pinMode(pin, INPUT);           // nastav 'pin' jako vstupní
digitalWrite(pin, HIGH);      // zapni pul l-up rezistory
```

Všimněte si, že ve výše uvedeném příkladu se nepřepíná pin na výstup, ale je to jen způsob aktivace vnitřního pull-up rezistoru. Pull-up rezistory se běžně používají, pokud na vstup připojujeme například spínač.



Piny, nakonfigurované jako výstup, mohou poskytnout proud až 40 mA do jiných zařízení či obvodů. Takový proud postačí pro jasné rozsvícení LED (nezapomeňte na předřadný rezistor), ale není to dostatečný proud pro sepnutí většiny relé, solenoidů nebo motorů.

Zkraty na pinech Arduino nebo jejich zatížení nadměrným proudem může obvod výstupního pinu poškodit nebo zničit, případně poškodit celý čip ATmega.

Proto je vhodné při připojování výstupního pinu k externímu zařízení použít rezistor s odporem 470  $\Omega$  nebo 1 k $\Omega$ , zapojeným v sérii s připojovaným zařízením.

## DIGITALREAD(PIN)

Funkce **digitalRead(pin)** přečte hodnoty z určeného digitálního pinu. Výsledkem je vysoká nebo nízká úroveň. Pin může být zadán buď jako proměnná nebo jako konstanta (0-13).

```
value=digitalRead(pin); // nastav proměnnou 'value' do stejného  
// stavu, jako má vstupní pin
```

## DIGITALWRITE(PIN, VALUE)

Nastaví zadaný digitální pin na vysokou (High) nebo nízkou (Low) úroveň. Číslo pinu může být zadáno buď jako proměnná nebo jako konstanta (0-13).

```
digitalWrite(pin, HIGH); // nastav 'pin' na HIGH
```

Následující příklad čte stav tlačítka, připojeného na digitální vstup a pokud je sepnuto, rozsvítí LED, připojenou k digitálnímu výstupu.

```
int led = 13; // připoj LED na pin 13  
int pin = 7; // připoj tlačítko na pin 7  
int value = 0; // proměnná pro uložení přečtené hodnoty  
void setup()  
{  
    pinMode(led, OUTPUT); // nastav pin 13 jako výstup  
    pinMode(pin, INPUT); // nastav pin 7 jako vstup  
}  
void loop()  
{  
    value=digitalRead(pin); // vlož do proměnné 'value' stav  
    // vstupního pinu  
    digitalWrite(led, value); // nastav hodnotu proměnné 'led' podle  
    // stavu tlačítka
```

## ANALOGOVÉ VSTUPY A VÝSTUPY

### ANALOGREAD(PIN)

Přečte hodnotu napětí z určeného analogového pinu v 10-bitovém rozlišení. Tato funkce pracuje pouze s analogovými piny (0-5). Výsledná hodnota je celočíselná s rozsahem od 0 do 1023.

```
value = analogRead(pin);      // nastav hodnotu proměnné 'value'  
                           // na hodnotu proměnné 'pin'
```



Analogové piny, na rozdíl od digitálních nemusí být nejprve deklarovány jako vstupní nebo výstupní.

### ANALOGWRITE(PIN, VALUE)

Zapíše pseudo-analogové hodnoty, generované pomocí hardwarově řízené pulzní šířkové modulace (PWM) na výstupní piny označené jako PWM. Na novějších Arduinech s čipy ATmega168 tato funkce pracuje na pinech 3, 5, 6, 9, 10 a 11. U starších Arduin s čipy ATmega8 jsou k dispozici pouze piny 9, 10 a 11.

Hodnota v rozsahu 0-255 může být zadána jako proměnná nebo konstanta.

```
analogWrite(pin, value); // zapíše 'value' analogově na 'pin'
```

Hodnota 0, zapsaná do **value**, nastavuje na zadáném výstupním pinu napětí 0 voltů (log.0), hodnota 255 nastavuje na zadáném výstupním pinu napětí 5 voltů (log.1). Aby bylo možno dosáhnout hodnot napětí mezi 0 a 5 volty, střídá se na výstupním pinu hodnota 0 a 1. Poměr tohoto střídání je určen hodnotou **value** (0 až 255) – čím vyšší hodnota, tím déle je na pinu vysoká logická úroveň (5 V). Například, pro hodnotu 64 je na pinu úroveň 0 tři čtvrtiny času, a 1 čtvrtinu času, pro hodnotu 128 bude na pinu úroveň 0 polovinu času a 1 druhou polovinu, a hodnota 192 znamená, že na pinu bude hodnota 0 čtvrtinu času a 1 tři čtvrtiny času.

Protože se jedná o hardwarovou funkci, bude pin po zavolání funkce **analogWrite** na pozadí běhu programu generovat nastavenou PWM až do příštího volání funkce

`analogWrite` (nebo do zavolání funkce `digitalRead` či `digitalWrite` na stejném pinu).



Analogové piny – na rozdíl od těch digitálních, nemusí být nejprve deklarovány jako vstupní nebo výstupní.

Následující příklad čte analogovou hodnotu ze vstupního pinu, upraví její hodnotu vydělením čtyřmi a pošle signál na výstup PWM:

```
int led = 10;           // LED s rezistorem 220 ohm na pinu 10
int pin = 0;            // potenciometer na analogový pin 0
int value;              // definice proměnné value
void setup()            // žádné nastavení není potřebné
{
}

void loop()
{
    value = analogRead(pin); // přepiš hodnotu 'value' do 'pin'
    value = map(pin, 0, 1023, 0, 255); // konvertuj rozsah 0-1023 na
                                         // /0-255
    analogWrite(led, value); // vyšli výstupní PWM signal na pin
                            // led
}
```

## ČASOVÁNÍ

### DELAY(MS)

Pozastaví program na dobu určenou v milisekundách, hodnota 1000 tedy odpovídá jedné sekundě.

```
delay(1000); // čekej jednu sekundu
```

### MILLIS()

Vrací počet milisekund od okamžiku rozběhu aktuálního programu na desce Arduino ve formátu unsigned long.

```
value = millis(); // nastav 'value' shodně s millis()
```



Hodnota této proměnné přeteče (přetočí se zpět na nulu), přibližně po 9 hodinách nepřetržitého běhu Arduina.

## MATEMATICKÉ FUNKCE

### MIN(X, Y)

Vypočítá poměr dvou čísel jakéhokoli datového typu a vrátí menší z nich.

```
value = min(value, 100); // nastaví 'value' na menší z hodnot  
// 'value' nebo 100, čímž zajistí, že  
// proměnná 'value' nepřekročí 100
```

### MAX(X, Y)

Vypočítá poměr dvou čísel jakéhokoli datového typu a vrátí větší z nich.

```
value = max(value, 100); // nastaví 'value' na větší z hodnot  
// 'value' nebo 100, čímž zajistí, že  
// proměnná 'value' neklesne pod 100
```

## NÁHODNÁ ČÍSLA

### RANDOMSEED(SEED)

Nastavuje hodnotu jako výchozí bod pro funkci `random()`.

```
randomSeed(value); // použij hodnotu proměnné 'value'
```

Arduino není schopno vytvořit skutečně náhodné číslo, funkce randomSeed umožňuje umístit proměnnou, konstantu, nebo jinou funkci do funkce random. Tento postup pomáhá vytvořit náhodnější "náhodná" čísla. Existuje celá řada různých funkcí, které mohou být použity jako základ funkce `random`, včetně funkce `millis()` nebo dokonce funkce `analogRead()`, která může číst elektrický šum přes analogový pin.



Random seed („náhodné semínko“) je náhodné číslo (nebo pole), které se používá při inicializaci generátoru pseudonáhodných čísel. Generátor při použití jiného semínka vrací jinou sekvenci pseudonáhodných dat.

### RANDOM(MAX), RANDOM(MIN, MAX)

Funkce `random` vrací pseudo-náhodná čísla v rozsahu stanoveném hodnotami min a max. Pokud je uveden pouze jeden parametr, bere se jako maximální hodnota.

```
value = random(100, 200);      // zapiš do 'value' náhodné
                                // číslo mezi 100-200
```



Nejprve použijte funkci `randomSeed()`.

Následující příklad vytvoří náhodnou hodnotu mezi 0-255 a použije ji pro řízení úrovně PWM signálu pinu PWM:

```
int randNumber;          // proměnná pro uložení náhodného čísla
int led = 10;            // LED s rezistorem 220 ohm na pin 10
void setup()             // nic se nenastavuje
{
}

void loop()
{
    randomSeed(millis()); // použij funkci millis() jako hodnotu
    randNumber = random(255); // ulož náhodné číslo mezi 0-255 do
                            // proměnné
    analogWrite(led, randNumber); // zapiš hodnotu proměnné na výstup
    delay(500);               // čekej 0,5 sekundy
}
```

## KOMUNIKACE

### SERIAL.BEGIN(RATE)

Otevře sériový port a nastaví komunikační rychlosť pro přenos dat. Typická přenosová rychlosť pro komunikaci s počítačem je 9600 bps, ale jsou podporovány i jiné přenosové rychlosti.

```
void setup()
{
    Serial.begin(9600); // otevří sériový port
} // nastav přenosovou rychlosť na 9600 bps
```



Při použití sériové komunikace nelze použít piny 0 (RX) a 1 (TX) současně jako digitální.

### SERIAL.PRINTLN(DATA)

Vytiskne data na sériový port jako čitelný ASCII text, následovaný znakem zalomení (ASCII 13 nebo '\r') a znakem nového řádku (ASCII 10 nebo '\n'). Tento příkaz má stejný tvar jako `Serial.print()`, ale jednodušeji se s ním pracuje, pokud zobrazujeme data v programu Serial Monitor.

```
Serial.println(analogValue); // odešli hodnotu
                            // promenné 'analogValue'
```



Další informace o různých možnostech funkcí `Serial.println()` a `Serial.print()` naleznete na webových stránkách Arduino.cc.

Následující jednoduchý příklad čte data z analogového pinu 0 a odesílá tato data jednou za sekundu do počítače.

```
void setup()
{
    Serial.begin(9600); // nastav komunikační rychlosť na 9600bps
}
void loop()
{
    Serial.println(analogRead(0)); // odesli analogovou hodnotu
    delay(1000); // čekej 1 sekundu
}
```