

## 2. LEKCE: 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 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.
- 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 a druhé zprava). Po dobehnutí 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

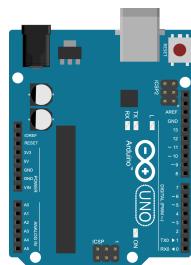
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8x.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu zástrčka-zástrčka.



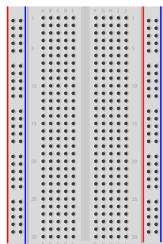
RGB led – 8 kusů



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.



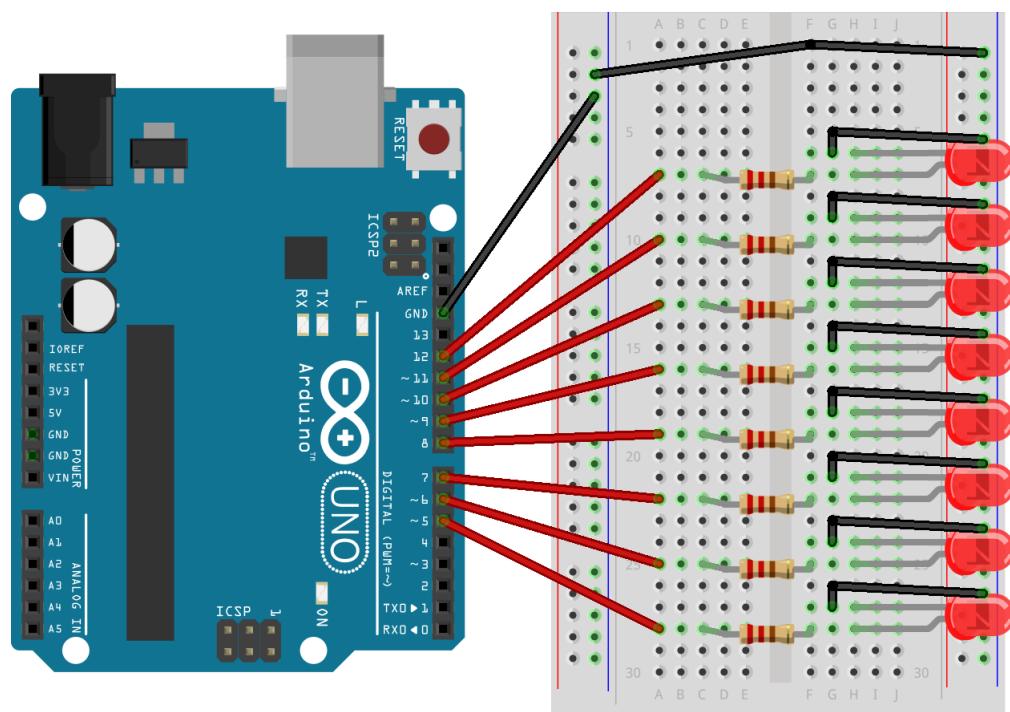
**DEJTE SI POZOR**

- ➔ 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 diod
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 samec-samec.
- 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

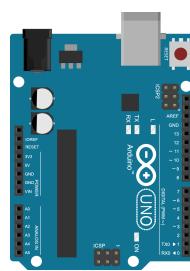
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8x.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu samec-samec.



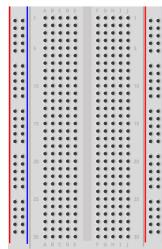
RGB led – 8 kusů



Rezistor  $220\Omega$  – 8 kusů



Deska Arduino

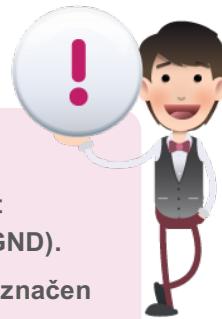


Kontaktní pole

POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

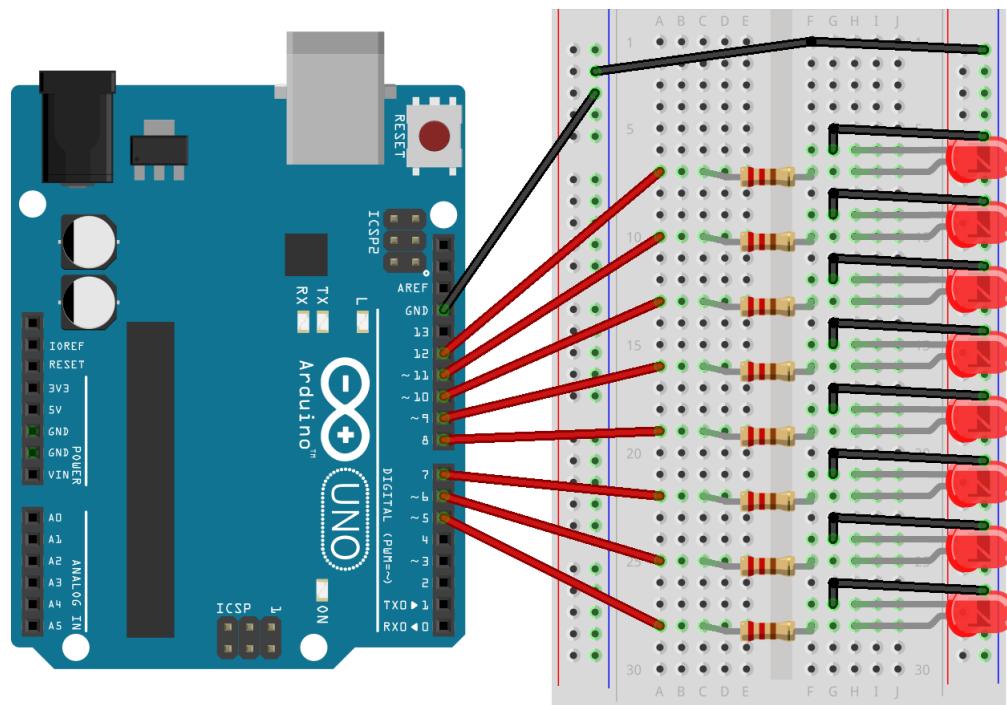
- ① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do programování. V opačném případě obvod musíte opět složit podle přiloženého schématu.



### 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[6] = {2, 4, -8, 3, 2};
4 char message[6] = "hello";
```

- ① **myInt[6]** – deklarace pole bez jeho inicializace.
- ② **myPins[]** – deklarace pole bez uvedení jeho konkrétní velikosti, komplilátor nejdříve spočítá prvky a pak pole vytvoří o odpovídající velikosti.
- ③ **mySendVals[6]** – deklarace pole s jeho přesným počtem prvků.
- ④ **message[6]** – u pole datového typu **char** musí být provedena jeho inicializace minimálně jedním prvkem.

## PŘÍSTUP K POLÍ

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 diod
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 diod
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 struktur – 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 samec-samec.
- 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 0-244.

- 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

- C) 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.

- D) 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

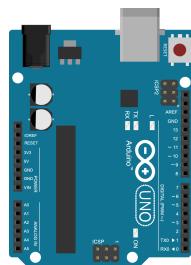
- ① LED diodu - 8x.
- ② Rezistor  $220\Omega$  – 8 kusů.
- ③ Desku Arduino.
- ④ Kontaktní pole.
- ⑤ Vodiče typu samec-samec.



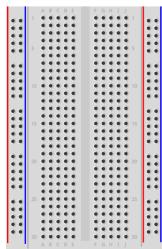
RGB led – 8 kusů



Rezistor  $220\Omega$  – 8 kusů



Deska Arduino

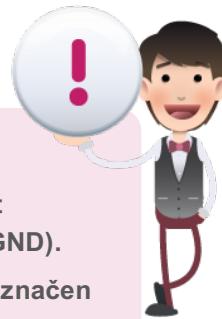


Kontaktní pole

POUŽITÉ SOUČÁSTKY

## A JDĚTE NA TO ...

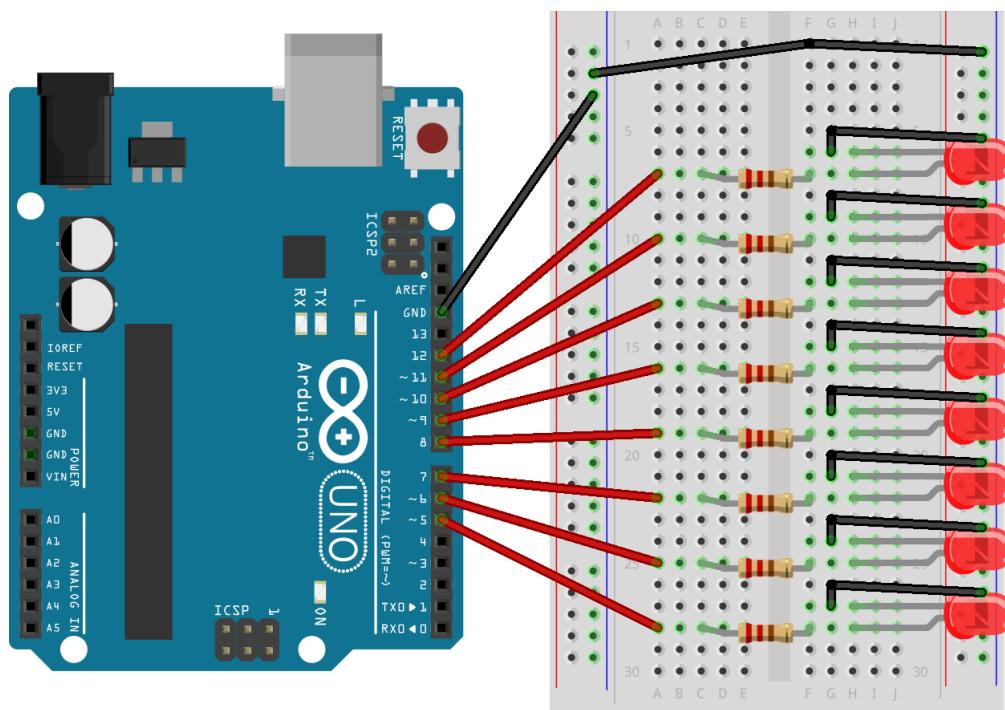
- ① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do programování. V opačném případě obvod musíte opět složit podle přiloženého schématu.



### 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 count=0; count<255; count++) {
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 umožnuje 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 navratovytyp;  
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[6] = {2, 4, -8, 3, 2};
4 char message[6] = "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]** – u pole datového typu **char** musí být provedena jeho inicializace minimálně jedním prvkem.

### 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 jakýkoliv 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 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 **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(i);
5         delay(10);
6     }
7 }
8
9 void loop() {
10}
11}
```

①

②

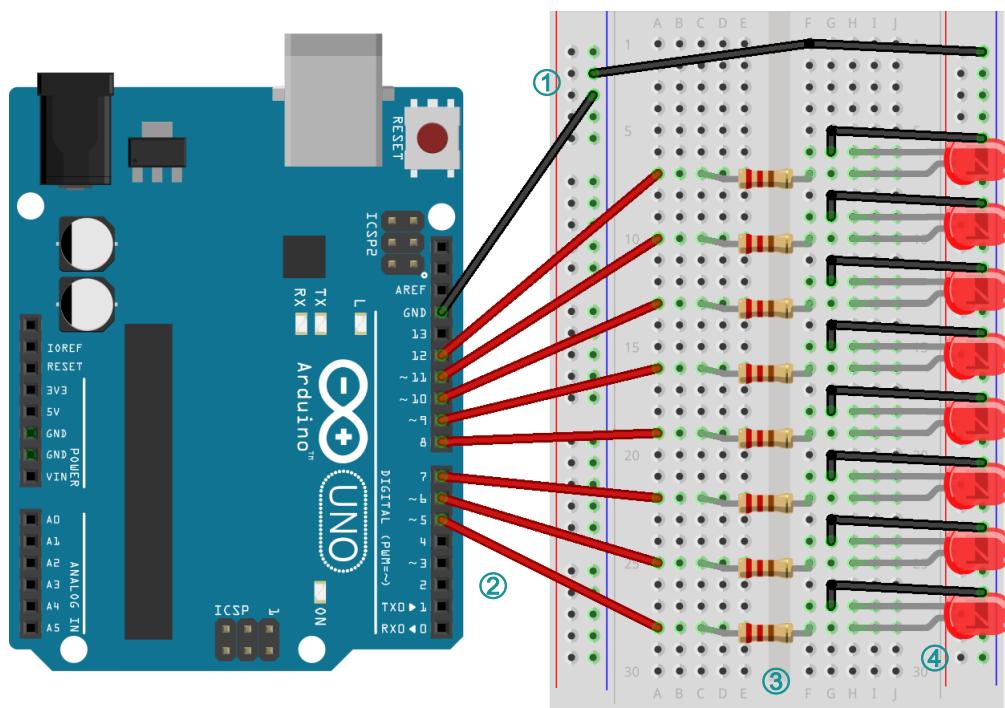
- ① Pokud se bude využívat sériové komunikace, je nutné nastavit rychlosť prenosu.
- ② Výpis hodnôt v sériovom monitoru. V tomto prípade se bude vypisovať hodnota premennej **i**. Každá hodnota bude na novom riadku, ak boli chteli hodnoty všetky vedľa seba v jednom riadku, použijeme funkciu **Serial.print(i)**.

# 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 diod osm. 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 Najdete 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. Krátký 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 zhasní 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. Návodou 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 nebo jej rozšířit o dvourozměrné pole a vytvořit tak kombinaci několik různých animací.



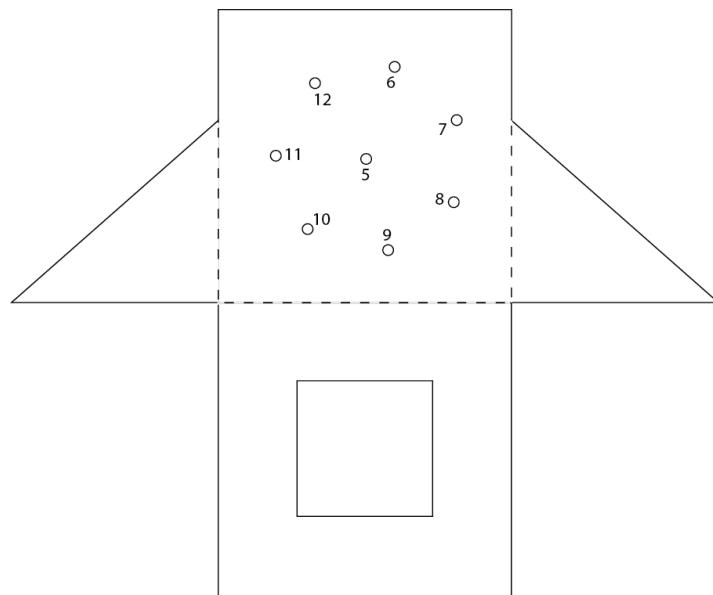
Pokud budete mít připravenou a funkční programovou část, můžete vytvořit stojánek pro LED diody. Tím animace bude efektnější. Návod, je uveden v následující kapitole.

## STOJAN PRO LED DIODY

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

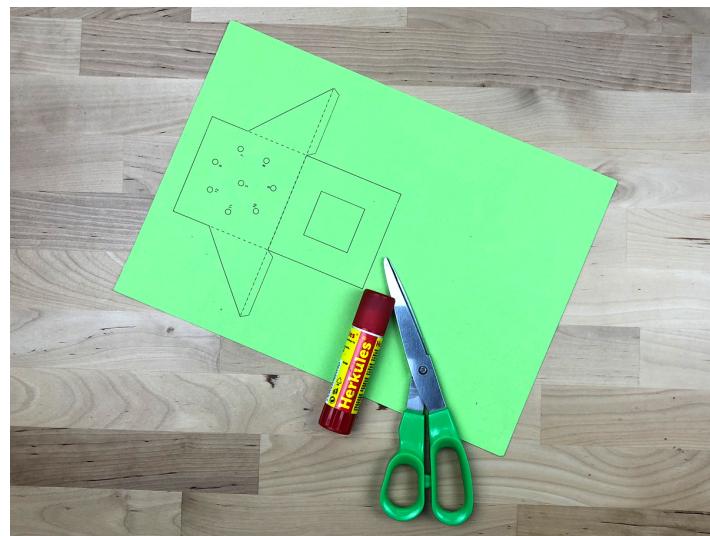
## PAPÍROVÁ KONSTRUKCE

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

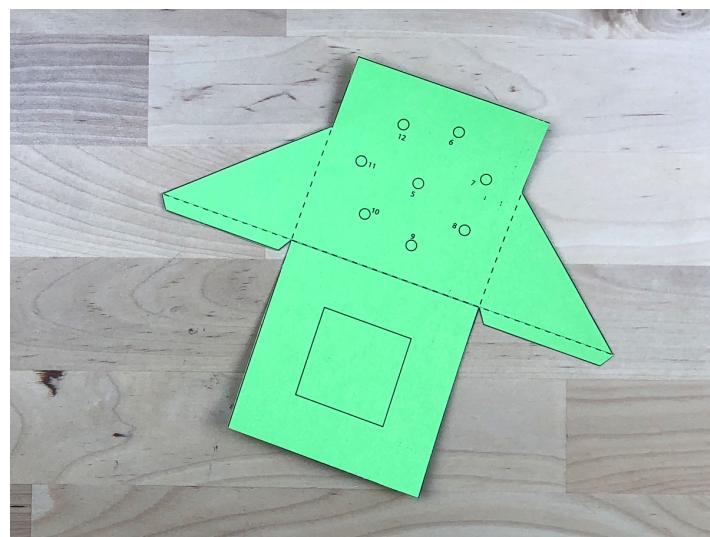


Obr. 2 - Šablona stojanu

Vytištěnou šablonu vystříhněte podle plných čar.



Obr. 3 - Vytisknuta šablona



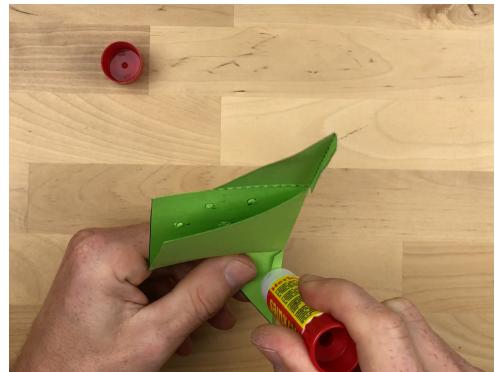
Obr. 4 - Vyštřízená šablona

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 na ohýbejte šablonu podle přerušovaných čar. Na lepící hrany naneste lepidlo  
Obr. 8 – připojení LED.

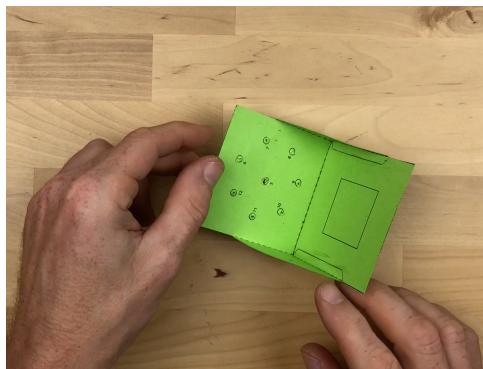


Obr. 5 – Ohýbání lepících hran

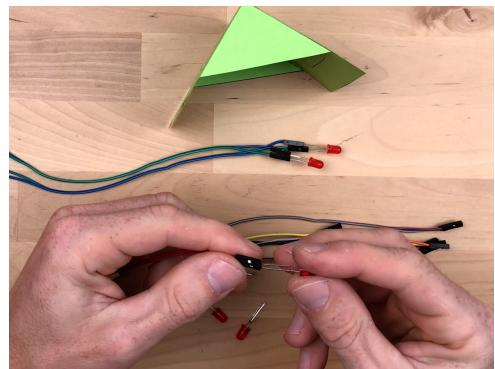


Obr. 6 – Lepení hran

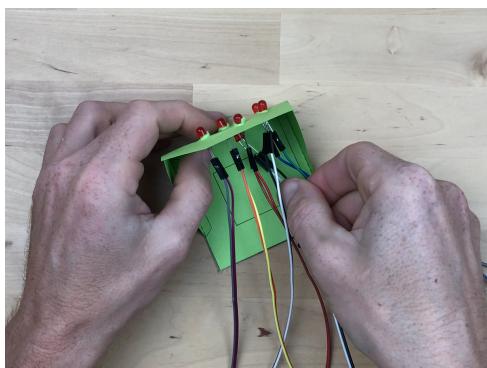
Lepící hrany přilepte k základně stojanu Obr. 6 – Lepení hran. Než zasunete LED diody do stojanu, připojte k nim vodiče Obr. 8 – Připojení LED diod.



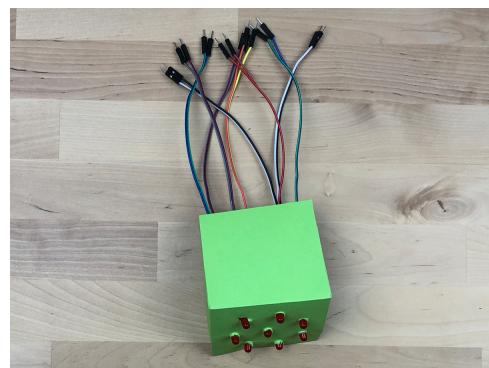
Obr. 7 – Ohýbání lepících hran



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(13, HIGH);   // rozsvícení diody 9
30     delay(50);
31     digitalWrite(5, LOW);     // zhasnutí diody
32     delay(50);
33     digitalWrite(6, LOW);     // zhasnutí diody
34     delay(50);
35     digitalWrite(7, LOW);     // zhasnutí diody
36     delay(50);
37     digitalWrite(8, LOW);     // zhasnutí diody
38     delay(50);
39     digitalWrite(9, LOW);     // zhasnutí diody
40     delay(50);
41     digitalWrite(10, LOW);    // zhasnutí diody
42     delay(50);
43     digitalWrite(11, LOW);    // zhasnutí diody
44     delay(50);
```

```
44 |     digitalWrite(11, LOW);      // zhasnutí diody
45 |     delay(50);
46 |     digitalWrite(12, LOW);      // zhasnutí diody
47 |     delay(50);
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 ,7  
2 ,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[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](#)).