

LasKKit LBot

Obsah

Úvod.....	2
Sestavení	3
Příprava na programování	3
Lekce 1 – Sériový monitor	4
Lekce 2 – Inteligentní RGB LED WS2812	5
Lekce 3 – Snímač okolního osvětlení	5
Lekce 4 – Bzučák	6
Lekce 5 – Tlačítko	6
Lekce 6 – IR přijímač.....	6
Lekce 7 – IR vysílač.....	7
Lekce 8 - Motory	8
Lekce 9 – Ultrazvukový měřič vzdálenosti	9
Lekce 10 – Čidla pro sledování čáry	10
Lekce 11 – Bluetooth modul	11
Demo software.....	13
Schéma zapojení základní desky	14

Úvod

LasKKit LBot je sada robotického vozítka pro začátečníky, na kterém se naučíš hardwarové a softwarové základy tvorby interaktivních zařízení.


Sada obsahuje:

- 1ks – Robotický podvozek s DC motory a kolečky
- 1ks – Držáku na 4xAA baterie s konektorem
- 1ks – Základní deska LBot V1.0
- 1ks – Arduino NANO
- 1ks – Motorový driver TB6612
- 1ks – Ultrazvukového čidla pro měření vzdálenosti od překážky HC-SR04
- 1ks – Dvoukanálového čidla pro sledování čáry
- 1ks – Bluetooth modul HC-06
- 1ks – miniUSB kabel do počítače
- Veškerý potřebný spojovací materiál
- Propojovací vodiče s Dupont konektory

Na základní desce jsou integrovány tyto komponenty:

- Patice pro osazení vývojové desky Arduino NANO
- Patice pro osazení modulu motorového driveru TB6612
- Bzučák
- Tlačítko
- Senzor okolního osvětlení – fototranzistor 5800B
- IR přijímač – VS1838B
- IR vysílač – IR LED dioda
- 2x inteligentní RGB LED WS2812
- 4x port pro připojení dalších čidel a externích zařízení
- Konektor pro připojení Bluetooth modulu
- Konektor pro připojení napájení
- Posuvný přepínač, pro zapnutí/vypnutí

Sestavení

 Při zapojování propojovacích kablíků dávej pozor na správné zapojení! Vždy je hnědý vodič v kablíku připojen na GND!

1. Pomocí dvou šroubů M3x25mm a dvěma maticemi M3 na každý motorek přišroubuj motorky na určené místo na vnitřních stranách boků podvozku tak, aby vodiče byly směrem doprostřed podvozku. Vodiče s konektory provlékni kruhovými otvory v přední části na horní stranu podvozku. Na hřídele motorků nasad' obě kolečka a přišroubuj je šrouby do plastu $\varnothing 2,2 \times 16$ mm.
2. Pomocí kuličky, která slouží jako přední kolečko, přišroubuj čidlo sledování čáry dvěma imbus šrouby M4x8 na spodní stranu přední části podvozku. Propoj čidlo přiloženým kablíkem s 4pin Dupont konektorem a druhý konec kablíku provlékni čtvercovým otvorem v přední části na horní stranu podvozku.
3. Zasad' ultrazvukové čidlo HC-SR04 do tištěného držáku a přišroubuj ho dvěma šroubky M4x6mm na přední čelo podvozku. Připoj kablík s 4pin Dupont konektorem.
4. Na horní stranu podvozku našroubuj čtyři distanční sloupky M-F M3x25mm.
5. Mezi distanční sloupky přilep oboustrannou lepící páskou držák se 4ks AA bateriemi.
6. Základní desku přišroubuj pomocí čtyř šroubů M3x6mm na distanční sloupky. Kablík od ultrazvukového čidla a čidel sledování čáry veď pod základní deskou mezi pravým distančním sloupkem a držákem baterií směrem k portu 1 a 2. Do portu č. 1 zapoj ultrazvukové čidlo, do portu č. 2 čidla sledování čáry.
7. Do konektorů MR a ML zapoj kablíky od motorků – MR = pravý motorek, ML = levý motorek.
8. Do patič v základní desce osad' Arduino NANO a řadič motorů.
9. Připoj konektor z bateriového držáku do konektoru na základní desce.

Příprava na programování

Pro programování budeš potřebovat vývojové prostředí **Arduino IDE**, ve kterém budeš vyvíjet vlastní software a pomocí něho budeš software do vývojové desky nahrávat.

Pokud ještě Arduino IDE nemáš, můžeš si ho nainstalovat podle našeho návodu -

<https://blog.laskarduino.cz/zaciname-s-arduinoem/>.

Jako další bude potřeba nainstalovat ovladače USB převodníku pro nahrávání softwaru osazený na vývojové desce Arduino NANO. V sadě dodávané desce Arduino NANO je převodník založen na čipu CH340 a pro instalaci můžeš opět využít našeho návodu - <https://blog.laskarduino.cz/instalace-ovladace-prevodniku-usb-na-uart-ch340/>.

Lekce 1 – Sériový monitor

Sériový monitor se používá ke komunikaci mezi Arduino deskou a počítačem nebo jinými přístroji. Je také důležitým nástrojem při ladění programů. Když něco nefunguje, necháš si vypisovat hodnotu proměnné, nebo určitý text na sériovou konzoli a pomocí těchto informací odhalíš chybu.

Následující kód ti bude každou 0,5s na sériovou konzoli vypisovat hodnotu proměnné **i**, ve které je každým cyklem číselná hodnota proměnné zvětšena o 1.


```
int i;

void setup() {
  Serial.begin(115200); // Nastavení a spuštění sériové komunikace s přenosovou rychlostí 115200baud
}

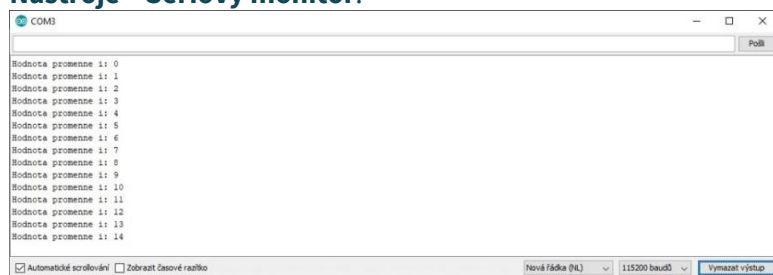
void loop() {
  Serial.print("Hodnota promenne i: "); // Vypise text na sériovou konzoli, ale neposílá konec řádku - další údaje se vypisují na stejný řádek za tento text
  Serial.println(i); // Vypsání hodnoty promenne i a ukončení řádku - další údaje se budou vypisovat na nový řádek

  i++; // Zvětšení hodnoty promenne i o 1

  delay(500); // Prodleva 0,5s před dalším opakováním
}
```

Nahraj program do desky tlačítkem **Nahrát** . Po úspěšném nahrání programu do desky se program automaticky spustí a hodnota proměnné se začne posílat na sériovou konzoli.

Okno sériové konzole si otevřeš klepnutím na ikonu  v pravém horním rohu Arduino IDE, nebo v menu **Nástroje – Sériový monitor**.



Lekce 2 – Inteligentní RGB LED WS2812

Na základní desce jsou osazeny dvě inteligentní RGB LEDky WS2812. V této lekci se je naučíš ovládat.

Jako první je potřeba nainstalovat potřebnou knihovnu do Arduino IDE. Knihovna, kterou k tomu budeš potřebovat se jmenuje **Adafruit NeoPixel** by **Adafruit**. Dostupná je ve Správci knihoven.

Pokud nevíš, jak takovou knihovnu nainstalovat, můžeš si přečíst náš článek

<https://blog.laskarduino.cz/instalace-knihoven-do-arduino-ide/>, kde popisujeme, jak se knihovny instalují.

Na začátek zdrojového kódu vlož řádek pro import knihovny:

```
#include <Adafruit_NeoPixel.h>
```

Nejdříve si vytvoř pomocnou proceduru, ve které se bude provádět vlastní změna barvy LEDky:

```
void nastavBarvu(byte r, byte g, byte b, int id) { // r = hodnota cervene, g = hodnota zelene, b = hodnota modre, id = cislo LED v poradí, kterou budeme nastavovat(1 = 1. LED, 2 = 2. LED atd.)
    uint32_t rgb; // Pomocna promenna, do ktere se bude ukladat hodnota barvy

    rgb = rgbWS.Color(r, g, b); // Konverze vstupnich hodnot R, G, B do pomocne promenne

    rgbWS.setPixelColor(id - 1, rgb); // Nastavi pozadovanou barvu pro konkretni led = pozice LED zacinaji od nuly
    rgbWS.show(); // Zaktualizuje barvu
}
```

Kompletní zdrojový kód. Po nahrání programu ti budou LEDky blikat jako policejní maják.

```
#include <Adafruit_NeoPixel.h> // Nacteni knihovny

Adafruit_NeoPixel rgbWS = Adafruit_NeoPixel(2, 13, NEO_GRB + NEO_KHZ800); // Inicializace LED - pocet_LED, pin_na_kterem_jsou_pripojeny_LED, parametry_komunikace

void setup() {
    rgbWS.begin(); // Spusteni komunikace s LED
    rgbWS.setBrightness(10); // Nastaveni jasu LED - 0 = minimalni jas (nesviti), 255 = maximalni jas (sviti naplno)
}

void loop() {
    nastavBarvu(255, 0, 0, 1); // Nastavi barvu 1. LED na červenou
    nastavBarvu(0, 0, 255, 2); // Nastavi barvu 2. LED na modrou
    delay(250); // Prodleva 250ms
    nastavBarvu(0, 0, 255, 1); // Nastavi barvu 1. LED na modrou
    nastavBarvu(255, 0, 0, 2); // Nastavi barvu 2. LED na červenou
    delay(250); // Prodleva 250ms
}

void nastavBarvu(byte r, byte g, byte b, int id) { // r = hodnota cervene, g = hodnota zelene, b = hodnota modre, id = pozice LED pro nastaveni
    uint32_t rgb;
    rgb = rgbWS.Color(r, g, b); // Konverze vstupnich hodnot R, G, B do jedne promenne
    rgbWS.setPixelColor(id - 1, rgb); // Nastavi pozadovanou barvu pro konkretni led = pozice LED zacinaji od nuly
    rgbWS.show(); // Zaktualizuje barvu
}
```

Lekce 3 – Snímač okolního osvětlení

Další integrovanou součástí je senzor okolního osvětlení. Jedná se o fototranzistor, který mění svůj odpor v závislosti na intenzitě dopadajícího světla. Pro zjištění hodnoty okolního osvětlení ti tedy stačí načíst napěťovou úroveň na analogovém pinu, na který je senzor připojen. V následujícím kódu načteš hodnotu na analogovém pinu kam je senzor připojen a následně ji necháš vypsát na sériovou konzoli.

```
const unsigned int svetelny_senzor = A6; // Senzor pripojen na pin A6
int val = 0; // Promenna, do ktere se bude ukladat hodnota okolního osvetleni

void setup() {
    Serial.begin(115200);
}

void loop() {
    val = analogRead(svetelny_senzor); // Nacteni hodnoty na analogovem pinu, kde je pripojen senzor. Hodnota v rozsahu 0 - 1024
    Serial.println(val); // Vypsani hodnoty na seriovou konzoli
    delay(250);
}
```

Lekce 4 – Bzučák

Další součástka na desce je bzučák. Pomocí něho můžeš například signalizovat různé stavy robota. V příkladu robot pípne po zapnutí napájení.

```
const unsigned int bzucak = 8; // Bzucak pripojen na pin D8

void setup() {
  pinMode(bzucak, OUTPUT); // Nastaveni pinu, kam je bzucak pripojen, jako vystupni

  tone(bzucak, 1000); // Spusteni bzucaku s frekvenci 1kHz
  delay(1000); // Pauza 1s
  noTone(bzucak); // Zastaveni bzucaku
}

void loop() {
}
```

Lekce 5 – Tlačítko

Deska obsahuje jedno tlačítko, které také můžeš ve svém programu využít. Například při stisknutí tlačítka bude pískat bzučák. Z důvodů nedostatku digitálních pinů je tlačítko připojeno na pin analogový, konkrétně na pin A7. U desky Arduino NANO se na tomto pinu nedá číst logická úroveň příkazem `digitalRead()`. Bude se tedy číst analogová úroveň na pinu (jako třeba u senzoru okolního osvětlení) a pomocí jednoduché podmínky se převede na 1bit informaci – 0 / 1.

```
const unsigned int tlacitko = A7; // Tlacitko pripojeno na pin A7
bool stav_tlacitka = 0; // Promenna, kam se bude ukladat aktualni stav tlacitka

void setup() {
  Serial.begin(115200);
  pinMode(bzucak, OUTPUT); // Nastaveni pinu, na který je pripojen bzucak, jako vystupni
  pinMode(tlacitko, INPUT); // Nastaveni pinu, na který je pripojeno tlacitko, jako vstupni
}

void loop() {
  stav_tlacitka = !(analogRead(tlacitko) >= 512); // Nacteni analogove hodnoty na pinu a pokud je hodnota vetsi, nebo rovna 512 vrati hodnotu 1. Jinak vraci hodnotu 0. Vysledek je negovan, takže hodnota zapsana do promenne je: 1 = tlacitko sepnuto, 0 = tlacitko rozepnuto
  Serial.println(stav_tlacitka);
  if (stav_tlacitka) { // Pokud je tlacitko stisknuto
    tone(bzucak, 1000); // Spusti bzucak s frekvenci 1kHz
  } else {
    noTone(bzucak); // Vypne bzucak
  }
  delay(250);
}
```

Lekce 6 – IR přijímač

V této lekci zprovozniš další integrovanou součást a tou je IR přijímač. Načteš a dekoduješ přijatý signál z dálkového ovladače, výsledek vypíšeš na sériovou konzoli a blikneš LEDkou na Arduino.

Pro zprovoznění přijímače nainstaluj ve Správci knihoven knihovnu **IRremote** by **shirriff**.

```
#include <IRremote.h>

const unsigned int ir_prijimac = 2; // IR prijimac na pinu D2
IRrecv irrecv(ir_prijimac); // Inicializace prijimace
unsigned long kod = 0; // Pomocna promenna, do ktere se bude ukladat prijaty kod

void setup() {
  Serial.begin(115200);
  pinMode(LED_BUILTIN, OUTPUT); // Pin integrovane LED na desce Arduino NANO jako vystup
  irrecv.enableIRIn(); // Start komunikace s prijimacem
  irrecv.blink13(true); // Pri prijmu kodu bliká integrovana LED na desce Arduino NANO
}

void loop() {
  decode_results prijaty_kod; // Nastaveni promenne, do ktere se bude ukladat prijaty kod

  if (irrecv.decode(&prijaty_kod)){ // Poku je prijaty kod
    if (prijaty_kod.value == 0xFFFFFFFF) prijaty_kod.value = kod; // Pokud prichazi kod opakovane (pri delsim stisku tlacitka) priradi do promenne kod predchozi
    Serial.println(prijaty_kod.value, DEC); // Vypise kod na seriovou konzoli v ciselne(desitkove) soustavě
    irrecv.resume(); // Nacte novou hodnotu
    kod = prijaty_kod.value; // Priradi prijaty kod do promenne
    delay(150); // Pauza 150ms
  }
}
```

i Pokud budeš chtít používat bzučák a IR přijímač v jednom kódu současně, je potřeba upravit soubor `IRremoteBoardDefs.h` ve složce `C:\Users\tve_jmeno\Documents\Arduino\libraries\IRremote\src\private`. Knihovna `IRremote` v základním nastavení používá stejný časovač jako funkce `tone()`, použitá u ovládání bzučáku. Bez úpravy by zkompileování programu skončilo chybou!

Úprava spočívá v zakomentování(vložení znaků // na začátek řádku) řádku obsahující `IR_USE_TIMER2` a odkomentování(smazání znaků // na začátku řádku) řádku obsahující `IR_USE_TIMER1`:

```

/*****
 * ARDUINO Boards
 *****/
// Arduino Duemilanove, Diecimila, LilyPad, Mini, Fio, Nano, etc
// ATmega48, ATmega88, ATmega168, ATmega328
#elif defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__) // old default clause
# if !defined(IR_USE_TIMER1) && !defined(IR_USE_TIMER2)
#define IR_USE_TIMER1 // tx = pin 9      (nove nastaveni)
#define IR_USE_TIMER2 // tx = pin 3      (puvodni nastaveni)
# endif

```

Úpravu lze udělat v jakémkoli textovém editoru, např. Poznámkový blok, PSPad, nebo kterýkoli jiný. Uvedené změny pak stačí jen uložit.

Lekce 7 – IR vysílač

Další integrovaná součást desky je IR vysílač. Pomocí něho můžeš ovládat prakticky jakékoli zařízení s IR dálkovým ovládáním, např. televizi. Musíš pouze znát jeho způsob komunikace. Pro Arduino jsou již některé protokoly zpracovány v knihovně, kterou jsme používali v minulé lekci pro příjem kódů.

V knihovně jsou zpracovány protokoly těchto výrobců:

Aiwa, BoseWave, Denon, Dish, JVC, Lego PF, LG, MagiQuest, Mitsubishi, NEC, Panasonic, RC5, RC6, Samsung, Sanyo, Sharp, Sony a Whynter.

V ukázce si vyzkoušíš poslat kód pro zařízení např. od Sony. Přednastavený kód se pošle stiskem tlačítka na základní desce.

```

#include <IRremote.h>

IRsend irsend; // Inicializace vysilace

const unsigned int tlacitko = A7;
bool stav_tlacitka = 0;

void setup() {
}

void loop() {
  stav_tlacitka = !(analogRead(tlacitko) >= 512);
  if (stav_tlacitka) {
    unsigned long tData = 0x41; // Data pro odeslani - ASCII znak A
    irsend.sendSony(tData, 16); // Odesleme data s kodovanim SONY - data, pocet_bitu_k_odeslani
    delay(200);
  }
}

```

 I zde platí informace z předchozí lekce, že pokud budeš používat IR vysílač současně s bzučákem, je nutná výše popsaná úprava.

Lekce 8 - Motory

Jako poslední je na desce osazena patice pro driver dvou DC motorů. Pomocí něho robota rozpohybuješ. V tomto příkladu využiješ kód z předchozí lekce č. 6 – motory a tím i robota budeš ovládat pomocí dálkového ovládání. Budeš moci s robotem jezdit dopředu, dozadu, zatáčet doprava a doleva.

```
#include <IRremote.h>

// Motor A, Leva strana
const unsigned int pwmLeft = 6; // PWM pin pro motor A
const unsigned int left = 7; // Dir pin, který určuje smer pro motor A
unsigned int korekce_L = 0; // Promenna pro nastaveni korekce rychlosti leveho motoru

// Motor B, Prava strana
const unsigned int pwmRight = 5; // PWM pin pro motor B
const unsigned int right = 4; // Dir pin, který určuje smer pro motor B
unsigned int korekce_R = 0; // Promenna pro nastaveni korekce rychlosti praveho motoru

unsigned int rychlost = 200; // Rychlost jizdy v rozsahu 0 - 255, 0 = stop, 255 = rychlost 100%

void setup() {
  pinMode(pwmLeft, OUTPUT); // PWM pin A jako vystup
  pinMode(left, OUTPUT); // Dir pin A jako vystup
  pinMode(pwmRight, OUTPUT); // PWM pin B jako vystup
  pinMode(right, OUTPUT); // Dir pin B jako vystup

  motoryStop(); // Motory stop
}

void loop() {
  decode_results prijaty_kod; // Nacteny kod z prijimace ulozone do promenne

  if (irrecv.decode(&prijaty_kod)){ // Poku je prijaty kod
    irrecv.resume(); // Nacte novou hodnotu

    if (prijaty_kod.value == 0xFFFFFFFF) prijaty_kod.value = kod; // Pokud prichazi kod opakovane (pri delsim stisku tlacitka na ovladaci)

    switch (prijaty_kod.value) { // Podle prijateho kodu ovladame motory
      case 16712445:
        jedDopredu();
        break;
      case 16758695:
        jedDozadu();
        break;
      case 16769055:
        otocDoleva();
        break;
      case 16748655:
        otocDoprava();
        break;
      default:
        //
        break;
    }

    kod = prijaty_kod.value;
  }else{ // Pokud neprichazi zadny kod tlacitek dalkoveho ovladace, zastav motory
    motoryStop();
  }
}

// Pomocna procedura pro vypocet korigovane rychlosti otaceni leveho motoru
unsigned int speed_L(unsigned int rychlost){
  unsigned int rychlost_L = rychlost + map(korekce_L, 0, 100, 0, rychlost);
  if (rychlost_L > 255) rychlost_L = 255;
  if (rychlost_L < 0) rychlost_L = 0;
  return rychlost_L;
}

// Pomocna procedura pro vypocet korigovane rychlosti otaceni praveho motoru
unsigned int speed_R(unsigned int rychlost){
  unsigned int rychlost_R = rychlost + map(korekce_R, 0, 100, 0, rychlost);
  if (rychlost_R > 255) rychlost_R = 255;
  if (rychlost_R < 0) rychlost_R = 0;
  return rychlost_R;
}

// Zastavi motory
void motoryStop() {
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, 0); // PWM motoru A na 0%
  analogWrite(pwmRight, 0); // PWM motoru B na 0%
}

// Jizda dopredu
void jedDopredu() {
  korekce_L = 7; // Korekce rychlosti leveho motoru 7% nastavene rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypocita rychlost leveho motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypocita rychlost praveho motoru
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Jizda dozadu
void jedDozadu() {
  korekce_L = 2; // Korekce rychlosti leveho motoru 2% nastavene rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypocita rychlost leveho motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypocita rychlost praveho motoru
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Otoci doleva
void otocDoleva() {
  unsigned int rychlost_L = map(50, 0, 100, 0, speed_L(rychlost)); // 50% rychlosti praveho kola
  unsigned int rychlost_R = speed_R(rychlost);
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Otoci doprava
void otocDoprava() {
  unsigned int rychlost_L = speed_L(rychlost);
  unsigned int rychlost_R = map(50, 0, 100, 0, speed_R(rychlost)); // 50% rychlosti leveho kola
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}
```

Lekce 9 – Ultrazvukový měřič vzdálenosti

Jako přídatný modul je v sadě přiložen ultrazvukový snímač vzdálenosti od překážky HC-SR04. Toto čidlo tvému robotovi umožní „vidět“ překážky před sebou a při vhodném programu se jim třeba vyhnout, nebo před překážkou zastavit.

Pro zprovoznění čidla je potřeba doinstalovat ve Správci knihoven knihovnu **HCSR04** by **Martin Sosic**.

V příkladu si změříš vzdálenost a její hodnotu vypíšeš na sériovou konzoli.

```
#include <HCSR04.h> // Nactení knihovny pro práci s ultrazvukovým čidlem

const unsigned int triggerPin = 11; // Trigger pin čidla na pinu D11
const unsigned int echoPin = 12; // Echo pin čidla na pinu D12

double vzdalenost; // Promenna, do které se bude ukladat zmerena vzdalenost

UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin); // Inicializace čidla

void setup () {
  Serial.begin(115200); // Spuštění sériové komunikace
}

void loop () {
  vzdalenost = distanceSensor.measureDistanceCm(); // Změření a uložení hodnoty vzdálenosti do promenne. Hodnota vzdálenosti je v cm
  if (vzdalenost < 0) vzdalenost = 0; // Knihovna vrací -1, pokud je zmerený rozsah mimo pracovní oblast čidla = < 0 a > 400cm
  Serial.println(distance, 0); // Vypíše na sériovou konzoli hodnotu zmerené vzdálenosti zaokrouhlenou na celé číslo

  delay(500);
}
```

Lekce 10 – Čidla pro sledování čáry

Další přídatný modul sada obsahuje dvě jednobanálová čidla pro sledování čáry. S těmito čidly bude tvůj robot po stisku tlačítka sám jezdit po vyznačené trase. Dalším stiskem tlačítka ukončíš režim sledování čáry a robot se zastaví.

```
const unsigned int cidlo_L = 9; // Leve cidlo na pinu 9
const unsigned int cidlo_R = 10; // Prave cidlo na pinu 10

// Motor A, leva strana
const unsigned int pwmLeft = 6; // PWM pin pro motor A
const unsigned int left = 7; // Dir pin, který určuje směr pro motor A
unsigned int korekce_L = 0; // Promenna pro nastavení korekce rychlosti levého motoru

// Motor B, prava strana
const unsigned int pwmRight = 5; // PWM pin pro motor B
const unsigned int right = 4; // Dir pin, který určuje směr pro motor B
unsigned int korekce_R = 0; // Promenna pro nastavení korekce rychlosti pravého motoru

unsigned int rychlost = 70; // Rychlost jízdy v rozsahu 0 - 255, 0 = stop, 255 = rychlost 100%

const int tlacitko = A7;
bool stav_tlacitka = 0;
bool on = 0;

void setup() {
  Serial.begin(115200);
  pinMode(pwmLeft, OUTPUT); // PWM pin A jako výstup
  pinMode(left, OUTPUT); // Dir pin A jako výstup
  pinMode(pwmRight, OUTPUT); // PWM pin B jako výstup
  pinMode(right, OUTPUT); // Dir pin B jako výstup
  motoryStop();
}

void loop() {
  stav_tlacitka = (analogRead(tlacitko) >= 512);
  if (stav_tlacitka) {
    if (on) { // Pokud byl spuštěn režim sledování čáry
      on = 0; // Vypni režim sledování čáry
      Serial.println("OFF");
    } else { // Pokud nebyl spuštěn režim sledování čáry
      on = 1; // Zapni režim sledování čáry
      Serial.println("ON");
    }
    delay(300);
  }

  if (on) { // Pokud byl zapnut režim sledování čáry
    bool s1 = digitalRead(cidlo_L); // Nactení logické úrovně z levého čidla sledování čáry
    bool s2 = digitalRead(cidlo_R); // Nactení logické úrovně z pravého čidla sledování čáry

    if (s1 && s2) {
      jedDopredu();
      Serial.println("Dopredu");
    } else if (!s1 && !s2) {
      motoryStop();
      Serial.println("Stop");
    } else if (s1 && !s2) {
      otocDoprava();
      Serial.println("Doprava");
    } else if (!s1 && s2) {
      otocDoleva();
      Serial.println("Doleva");
    }
  } else { // Pokud byl vypnut režim sledování čáry
    motoryStop();
  }
}

// Pomocná procedura pro výpočet korigované rychlosti otáčení levého motoru
unsigned int speed_L(unsigned int rychlost) {
  unsigned int rychlost_L = rychlost + map(korekce_L, 0, 100, 0, rychlost);
  if (rychlost_L > 255) rychlost_L = 255;
  if (rychlost_L < 0) rychlost_L = 0;
  return rychlost_L;
}

// Pomocná procedura pro výpočet korigované rychlosti otáčení pravého motoru
unsigned int speed_R(unsigned int rychlost) {
  unsigned int rychlost_R = rychlost + map(korekce_R, 0, 100, 0, rychlost);
  if (rychlost_R > 255) rychlost_R = 255;
  if (rychlost_R < 0) rychlost_R = 0;
  return rychlost_R;
}

// Zastaví motory
void motoryStop() {
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, 0); // PWM motoru A na 0%
  analogWrite(pwmRight, 0); // PWM motoru B na 0%
}

// Jízda dopředu
void jedDopredu() {
  korekce_L = 7; // Korekce rychlosti levého motoru 7% nastavené rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypočítá rychlost levého motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypočítá rychlost pravého motoru
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Jízda dozadu
void jedDozadu() {
  korekce_L = 2; // Korekce rychlosti levého motoru 2% nastavené rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypočítá rychlost levého motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypočítá rychlost pravého motoru
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Otáčí doleva
void otocDoleva() {
  unsigned int rychlost_L = map(50, 0, 100, 0, speed_L(rychlost)); // 50% rychlosti pravého kola
  unsigned int rychlost_R = speed_R(rychlost);
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Otáčí doprava
void otocDoprava() {
  unsigned int rychlost_L = speed_L(rychlost);
  unsigned int rychlost_R = map(50, 0, 100, 0, speed_R(rychlost)); // 50% rychlosti levého kola
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}
```

Lekce 11 – Bluetooth modul

Jako poslední je v sadě obsažen Bluetooth modul HC-06, se kterým budeš moci robota ovládat na dálku např. z aplikace v mobilním telefonu.

Jako ovládací aplikaci pro mobilní telefony Android nainstaluj [Bluetooth Controller for Arduino](#). Jedná se o velice jednoduchou aplikaci, ve které si můžeš vybrat z několika typů ovládání vzdáleného zařízení. V našem příkladu budeš používat volbu **Game Controller**. Tato varianta ovládání ti umožní pomocí tlačítek na displeji jezdit s robotem dopředu, dozadu, zatáčet doprava a doleva. Způsob je obdobný, jako u ovládání IR dálkovým ovládáním. Kód tohoto příkladu je tedy velmi podobný kódu z lekce 8.

Jelikož bluetooth modul komunikuje s Arduino NANO pomocí UARTu, je potřeba mít modul při nahrávání programu do Arduina odpojený z konektoru, jinak nahrávání skončí chybou. Arduino NANO má pouze jeden hardwarový UART a ten se používá i pro nahrávání softwaru.

```
// Motor A, Leva strana
const unsigned int pwmLeft = 6; // PWM pin pro motor A
const unsigned int left = 7; // Dir pin, který určuje smer pro motor A
unsigned int korekce_L = 0; // Promenna pro nastaveni korekce rychlosti leveho motoru

// Motor B, Prava strana
const unsigned int pwmRight = 5; // PWM pin pro motor B
const unsigned int right = 4; // Dir pin, který určuje smer pro motor B
unsigned int korekce_R = 0; // Promenna pro nastaveni korekce rychlosti praveho motoru

unsigned int rychlost = 200; // Rychlost jizdy v rozsahu 0 - 255, 0 = stop, 255 = rychlost 100%

char prijata_data; // Promenna, do ktere se nactou prijata data z bluetooth

void setup() {
  pinMode(pwmLeft, OUTPUT); // PWM pin A jako vystup
  pinMode(left, OUTPUT); // Dir pin A jako vystup
  pinMode(pwmRight, OUTPUT); // PWM pin B jako vystup
  pinMode(right, OUTPUT); // Dir pin B jako vystup
  Serial.begin(9600); // Spusteni komunikace prez seriový port
}

motoryStop();

void loop() {
  if (Serial.available() > 0) { // Poku jsou prijaty data z Bluetooth
    prijata_data = Serial.read(); // Nacte data z Bluetooth
    delay(2);

    switch (prijata_data) {
      case 'u':
        jedDopredu();
        break;
      case 'd':
        jedDozadu();
        break;
      case 'l':
        otocDoleva();
        break;
      case 'r':
        otocDoprava();
        break;
      case 'o':
        motoryStop();
        break;
      default:
        //
        break;
    }
  }
}

// Pomocna procedura pro vypocet korigovane rychlosti otaceni leveho motoru
unsigned int speed_L(unsigned int rychlost){
  unsigned int rychlost_L = rychlost + map(korekce_L, 0, 100, 0, rychlost);
  if (rychlost_L > 255) rychlost_L = 255;
  if (rychlost_L < 0) rychlost_L = 0;
  return rychlost_L;
}

// Pomocna procedura pro vypocet korigovane rychlosti otaceni praveho motoru
unsigned int speed_R(unsigned int rychlost){
  unsigned int rychlost_R = rychlost + map(korekce_R, 0, 100, 0, rychlost);
  if (rychlost_R > 255) rychlost_R = 255;
  if (rychlost_R < 0) rychlost_R = 0;
  return rychlost_R;
}

// Zastavi motory
void motoryStop() {
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, 0); // PWM motoru A na 0%
  analogWrite(pwmRight, 0); // PWM motoru B na 0%
}

// Jizda dopredu
void jedDopredu() {
  korekce_L = 7; // Korekce rychlosti leveho motoru 7% nastavene rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypocita rychlost leveho motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypocita rychlost praveho motoru
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Jizda dozadu
void jedDozadu() {
  korekce_L = 2; // Korekce rychlosti leveho motoru 2% nastavene rychlosti
  korekce_R = 0;
  unsigned int rychlost_L = speed_L(rychlost); // Vypocita rychlost leveho motoru
  unsigned int rychlost_R = speed_R(rychlost); // Vypocita rychlost praveho motoru
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

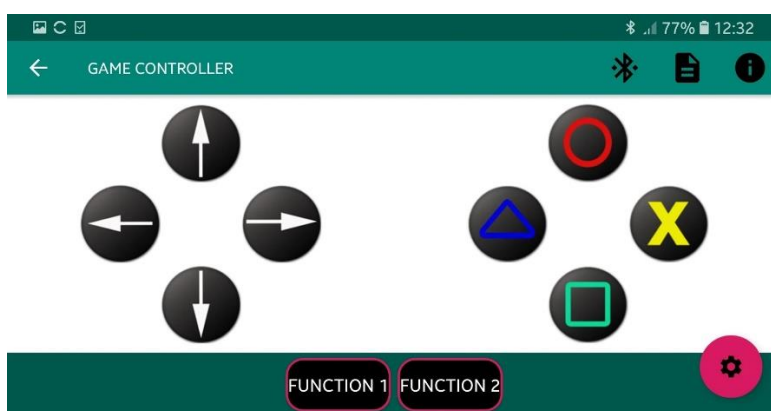
// Otoci doleva
void otocDoleva() {
  unsigned int rychlost_L = map(50, 0, 100, 0, speed_L(rychlost)); // 50% rychlosti praveho kola
  unsigned int rychlost_R = speed_R(rychlost);
  digitalWrite(left, LOW); // Dir pin motoru A na log. 0
  digitalWrite(right, HIGH); // Dir pin motoru B na log. 1
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}

// Otoci doprava
void otocDoprava() {
  unsigned int rychlost_L = speed_L(rychlost);
  unsigned int rychlost_R = map(50, 0, 100, 0, speed_R(rychlost)); // 50% rychlosti leveho kola
  digitalWrite(left, HIGH); // Dir pin motoru A na log. 1
  digitalWrite(right, LOW); // Dir pin motoru B na log. 0
  analogWrite(pwmLeft, rychlost_L); // PWM motoru A na nastavenou rychlost
  analogWrite(pwmRight, rychlost_R); // PWM motoru B na nastavenou rychlost
}
```

Aby fungovala komunikace s mobilním telefonem, je nejprve nutné modul HC-06 s telefonem spárovat. Pro párování použij klíč **1234**. Po spárování modulu spusť aplikaci Bluetooth Controller for Arduino a na úvodní obrazovce vyber zařízení HC-06:



Potom z nabídky vyber **Game Controller**:



Nyní můžeš šipkami s robotem jezdit.

V našem příkladu je modul HC-06 v základním nastavení se kterým je v sadě dodáván, takže sériová komunikace probíhá rychlostí 9600baud, párovací klíč je 1234 a název modulu je HC-06. Pokud budeš potřebovat některé z parametrů nastavení modulu změnit, přečti si náš článek: <https://blog.laskarduino.cz/bluetooth-modul-hc-06/>, kde se dozvíš podrobnosti o možnostech nastavení tohoto modulu.

Demo software

Z těchto příkladů programování jednotlivých komponent jsme vytvořili jednoduchý demo program, se kterým je LBot dodáván.

S tímto programem můžeš ihned po sestavení robota jezdit pomocí infračerveného dálkového ovládání, nebo z mobilního telefonu přes Bluetooth a nikdy nenarazíš do žádné překážky, protože program sleduje ultrazvukovým čidlem překážky před sebou, v bezpečné vzdálenosti před překážkou zastaví a pro snadnější objetí překážky i kousek couvne. Další funkcí v tomto základním režimu je, že pokud klesne intenzita okolního osvětlení, rozsvítí obě tříbarevné LED diody bílou barvou a pokud zase intenzita stoupne, automaticky je zhasne.

Tlačítkem na základní desce, nebo tlačítkem **O** v aplikaci na mobilním telefonu/tabletu lze robota přepnout do režimu sledování čáry a LBot začne jezdit tvoji připravenou trasu. Opakovaným stiskem tlačítka na základní desce, nebo tlačítka **O** v aplikaci na mobilním telefonu/tabletu se režim sledování čáry ukončí a LBot se vrátí do základního režimu. Stejnou funkci má i tlačítko **O** na infračerveném dálkovém ovladači. Zapnutí a vypnutí tohoto režimu dá LBot vědět i na sériovou konzoli.

Zdrojové kódy našeho demo softwaru si můžeš prohlédnout a stáhnout na našem GitHubu:

<https://github.com/LasKKit/LBot>

Schéma zapojení základní desky

