PODROBNÝ PRŮVODCE TEORIÍ

UKÁZKA PRÁCE JEDNÍM ZE ZÁKLADNÍCH SENZORŮ, KTERÝ LZE ZAPOJIT S DESKOU ARDUINO. JEDNÁ SEE O ULTRAZVUKOVÝ SENZOR VZDÁLENOSTI, KTERÝ POSKYTUJE VÝZNAMNÉ FUNKCIONALITĚ SPOJENÉ S AUTONOMIÍ ROBOTICKÝCH PROSTŘEDKŮ.

OBSAH PRŮVODCE

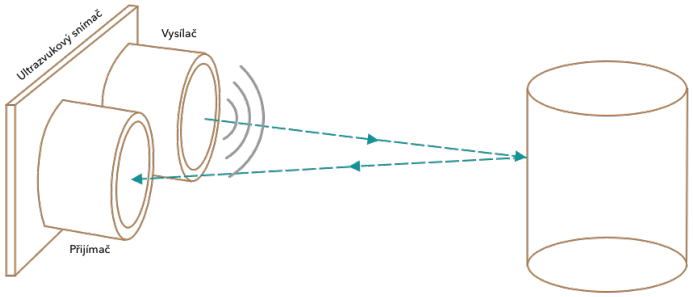
O ULTRAZVUKOVÉM SENZORU

Velmi častým a levným ultrazvukovým senzorem je typ HC-SR04. Jedná se o snadno použitelný ultrazvukový senzor vzdálenosti, s rozsahem od 2 do 400 cm. Běžně se používá při vyhýbání se překážkám robotům a automatizačním projektům.

JAK FUNGUJE ULTRAZVUKOVÝ SENZOR VZDÁLENOSTI?

Ultrazvukové senzory fungují tak, že vydávají zvukové vlny s frekvencí, která je pro člověka příliš vysoká. Tyto zvukové vlny cestují vzduchem rychlostí zvuku, zhruba 343 m / s. Pokud je před snímačem nějaký předmět, zvukové vlny se odrazí zpět a přijímač ultrazvukového snímače je detekuje. Měřením toho, kolik času uplynulo mezi odesláním a příjmem zvukových vln, lze vypočítat vzdálenost mezi senzorem a objektem. Obr. 1 - Princip .

Při 20 °C je rychlost zvuku zhruba 343 m/s nebo 0,034 cm/µs. Řekněme, že doba mezi



Obr. 1 - Princip ultrazvukového senzoru

odesláním a příjmem zvukových vln je 2 000 mikrosekund. Pokud se vynásobí rychlost zvuku časem, kdy zvukové vlny urazily, získá se vzdálenost, kterou zvukové vlny urazily.

Vzdálenost = rychlost x čas

Ale to není výsledek, který se hledá. Vzdálenost mezi senzorem a objektem je ve skutečnosti pouze poloviční, protože zvukové vlny putovaly ze senzoru do objektu a zpět z objektu do senzoru. Výsledek se musí vydělit dvěma.

Vzdálenost (cm) = rychlost zvuku (cm/µs) × čas (µs)/2

Příklad:

Vzdálenost (cm) = 0,0343 (cm/us) × 2000 (us)/2 = 34,3 cm

TEPLOTNÍ ZÁVISLOST RYCHLOSTI ZVUKU

Rychlost zvuku ve skutečnosti silně závisí na teplotě a v mnohem menší míře na vlhkosti vzduchu. Uvádí se, že rychlost zvuku stoupá zhruba o 0,6 m/s na stupeň Celsia. Pro většinu případů při 20 ° C lze použít pouze 343 m / s, ale pokud se má získat přesnější hodnoty, můžete vypočítat rychlost zvuku pomocí následujícího vzorce:

$$V\ (m / s) = 331{,}3 + (0{,}606 \times T)$$

V = rychlost zvuku (m / s), T = teplota vzduchu (° C)

Tento vzorec nezahrnuje vlhkost, protože jeho vliv na rychlost zvuku je jen velmi malý.

JAK FUNGUJE HC-SR04

Na přední straně senzoru HC-SR04 jsou dva stříbrné válce (ultrazvukové měniče), jeden je vysílačem zvukových vln a druhý přijímačem. Chceme-li, aby senzor generoval zvukový záblesk, musíme nastavit **Trig pin** na minimálně **10 µs**. Senzor poté vytvoří 8cyklový výbuch ultrazvuku při 40 kHz.

Tento zvukový záblesk se pohybuje rychlostí zvuku, odrazí se zpět a je přijímán přijímačem snímače. Pin **Echo** pak vydává čas, který zvukové vlny prošly v mikrosekundách.

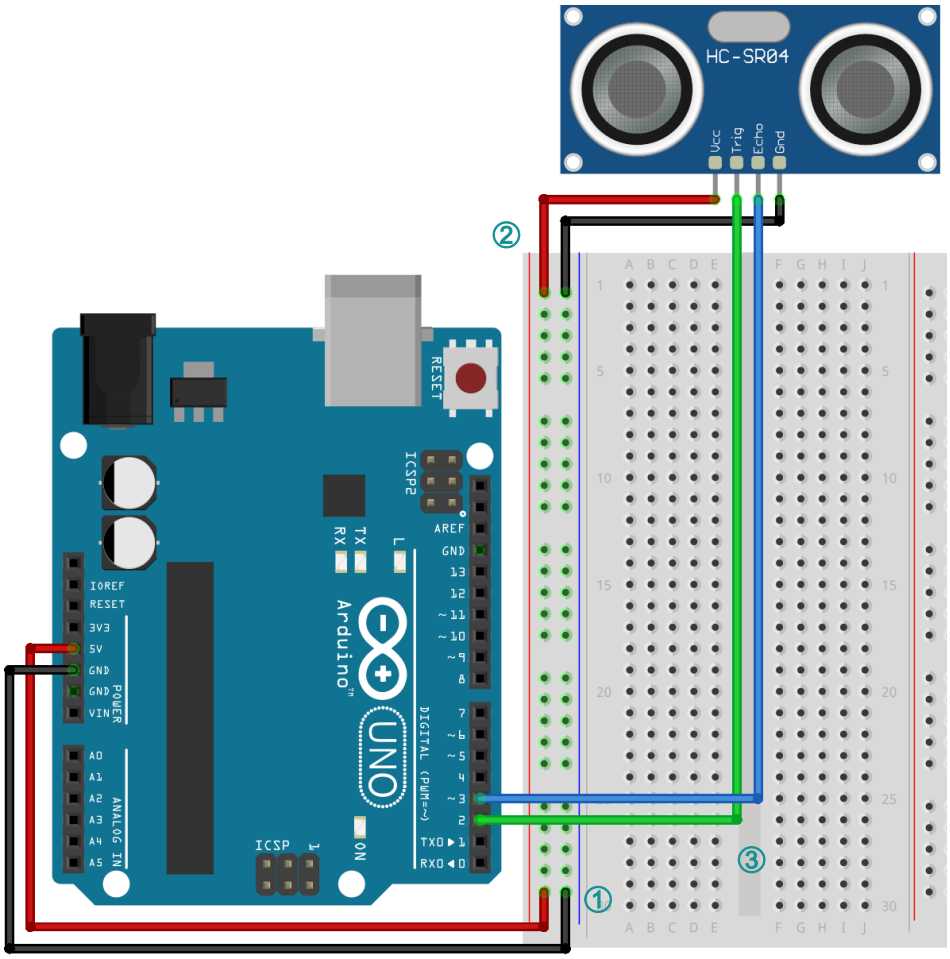
Funkcí `pulseIn()` v kódu Arduino se může použít ke čtení délky pulzu z pinu **Echo**. Poté můžeme pomocí vzorce uvedeného výše vypočítat vzdálenost mezi senzorem a objektem.

Obr. 2 – Ultrazvukový senzor HC-SR04

ZAPOJENÍ OBVODU

ZÁKLADNÍ PŘÍKLAD

Základní příklad představuje zapojení ultrazvukového senzoru pro jednoduchou ukázku odečítání vzdálenosti od překážek.



Obr. 3 - Zapojení ultrazvukového senzoru

① Do kontaktního pole, do druhé části, přivedeme zem z desky Arduino. Zem z externího napájecího zdroje a z desky Arduino jsou propojeny.

② Ze senzoru připojíme pin **VCC** na kladnou polaritu napájení na kontaktní desce a **GND** na zápornou polaritu.

③ Pin **ECHO** ze senzoru přivedeme na **pin 2** desky Arduino.

④ Pin **TRIG** ze senzoru přivedeme na **pin 3** desky Arduino.

PROGRAMOVÝ KÓD

```
1  #define trigPin 2
2  #define echoPin 3
3
4  long duration;
5  int distance;
6
7  void setup() {
8    pinMode(trigPin, OUTPUT);
9    pinMode(echoPin, INPUT);
10   Serial.begin(9600);
11 }
12
13 void loop() {
14   digitalWrite(trigPin, LOW);
15   delayMicroseconds(5);
16   digitalWrite(trigPin, HIGH);
17   delayMicroseconds(10);
18   digitalWrite(trigPin, LOW);
19   duration = pulseIn(echoPin, HIGH);
20   distance = duration * 0.034 / 2;
21   Serial.print("Distance = ");
22   Serial.print(distance);
23   Serial.println(" cm");
24   delay(50);
25 }
```

① Příkazy **`#define`** definuje konstantní hodnoty. Když je program kompilován, kompilátor nahradí všechny odkazy na tuto konstantu definovanou hodnotou. Takže kdekoli zmíníte **`trigPin`**, kompilátor jej při kompilaci programu nahradí hodnotou **2**. Stejně tomu je u **`echoPin`**, kde bude nastavena hodnota **3**.

② Dále jsem definovány dvě proměnné: **`duration`** což je doba trvání mezi odesláním a příjmem zvukových vln. Proměnná **`distance`** se používá k uložení vypočítané vzdálenosti.

③ Ve funkci **`setup()`** jsou nastaveny konstanty **`trigPin`** jako výstupu a **`echoPin`** jako vstup.

④ Inicializace sériové komunikace s přenosovou rychlostí 9600. Později lze zobrazit měřenou vzdálenost na sériovém monitoru, který se spustí pomocí **Ctrl+Shift+M** nebo

v nabídce **Nástroje > Sériový monitor**. Zkontrolujte, zda je v sériovém monitoru nastavena přenosová rychlost také na 9600.

⑤ Pro získání čistého signálu se začíná vymazáním **`trigPinu`** nastavením na hodnotu **`LOW`** na 5 mikrosekund.

⑥ Ve funkci **`loop()`** se spustí senzor nastavením **`trigPin`** na hodnotu **`HIGH`** na 10 µs.

⑦ Dále musíme přečíst délku pulzu odeslaného v **`echoPin`**. K tomu se používá funkce **`pulseIn()`**. Tato funkce čeká, až se pin přepne z **`LOW`** na **`HIGH`**, spustí časování, a poté čeká, až pin přejde na **`LOW`** a zastaví časování. Poté lze vzdálenost vypočítat pomocí vzorce uvedeného v úvodu.

⑧ Nakonec se vypočítaná vzdálenost vytiskne na sériovém monitoru.

Nezapomeňte program zkompilovat a nahrát do desky Arduino. Pokud je vše v pořádku, na sériovém monitoru, který spustíte pomocí **Ctrl+Shift+M** nebo v nabídce **Nástroje > Sériový monitor**. Zkontrolujte, zda je v sériovém monitoru nastavena přenosová rychlost také na **9600**.

NA SÉRIOVÉM MONITORU SE NEZOBRAZUJÍE ŽÁDNÝ TEXT

Nastavení přenosové rychlosti – zkontrolujte, zda je přenosová rychlost nastavena na 9600.

Zapojení senzoru – zkontrolujte zapojení pinu **ECHO** a **TRIG** na senzoru a desce 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.

ZÁKLADNÍ PŘÍKLAD – VYUŽITÍ KNIHOVNY NEWPING

Knihovnu **NewPing** lze použit s celou řadou ultrazvukových senzorů vzdálenosti. Můžete si všimnout, že níže uvedený kód, který používá knihovnu **NewPing**, je mnohem kratší než kód, který jsme použili dříve. Kromě toho, knihovna **NewPing** obsahuje některé další funkce. Umožňuje například nastavit maximální vzdálenost ke čtení a má integrovaný mediánový filtr, který výrazně zlepšit přesnost naměřených hodnot senzoru HC-SR04.

INSTALACE KNIHOVNY NEWPING

1. Stáhněte si knihovnu z repozitáře.
2. V nabídce IDE prostředí **Arduino Projekt > Přidat knihovnu zvolte Přidat ZIP knihovnu …**
3. Z disku vyberte stažený soubor knihovny. Knihovna se automaticky přidá do IDE Arduino.
4. Zda se knihovna přidala ověříte pohledem do nabídky **Soubor > Příklady > NewPing**.

Knihovna obsahuje příklady, které lze použít, ale musíme je upravit tak, aby odpovídaly našemu nastavení hardwaru.

PROGRAMOVÝ KÓD

```
1  #include <NewPing.h>                                          ①
2
3  #define trigPin  2                                           ②
4  #define echoPin  3
5  #define MAX_DISTANCE 350                                     ③
6  NewPing sonar(trigPin, echoPin, MAX_DISTANCE);               ④
7  float duration, distance;
8  void setup() {
9    Serial.begin(9600);
10 }
11
12 void loop() {
13   delay(50);                                                 ⑤
14
15   duration = sonar.ping();
16   distance = (duration / 2) * 0.0343;
17
18   Serial.print("Distance = ");
19   Serial.print(distance);                                    ⑥
20   Serial.println(" cm");
21 }
```

① Připojení knihovny **`NewPing`**.

② Příkazy **`#define`** definuje konstantní hodnoty. Když je program kompilován, kompilátor nahradí všechny odkazy na tuto konstantu definovanou hodnotou. Takže kdekoli zmíníte **`trigPin`**, kompilátor jej při kompilaci programu nahradí hodnotou **2**. Stejně tomu je u **`echoPin`**, kde bude nastavena hodnota **3**.

③ Maximální vzdálenost, na kterou chceme provést impulz (v centimetrech). Maximální vzdálenost snímače je dimenzována na 400-500 cm.

④ Nastavení pinů a maximální vzdálenosti pro **`NewPing`**.

⑤ Počkejte 50 ms mezi impulzy (asi 20 impulzů/s). Nejkratší zpoždění mezi impulzy by mělo být 29 ms.

⑥ Pokud bude limit snímané vzdálenosti mimo nastavené maximum, bude hodnota **`distance=0`**.

Můžeme také použít **`distance=sonar.ping_cm()`** nebo **`distance=sonar.ping_in()`**, který vrátí naměřenou vzdálenost v celých centimetrech nebo palcích. S touto funkcí se nemusí provádět měření délky a počítat vzdálenost.

JAK POUŽÍVAT DIGITÁLNÍ FILTR Z KNIHOVNY NEWPING

Velmi užitečná věc, kterou knihovna **NewPing** nabízí, je zabudovaný mediánový filtr. Tento filtr může výrazně zlepšit přesnost naměřených hodnot HC-SR04. Funkce **`ping_median()`** pracuje s několika měřeními v řadě, které.

PROGRAMOVÝ KÓD

```
#include <NewPing.h>

#define trigPin  2
#define echoPin  3
#define MAX_DISTANCE 350
NewPing sonar(trigPin, echoPin, MAX_DISTANCE);
float duration, distance;
void setup() {
  Serial.begin(9600);
}

void loop() {
  delay(50);

  int iterations = 5;                                    ①
  duration = sonar.ping_median(iterations);              ②
  distance = duration * 0.0343 / 2;

  Serial.print("Distance = ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

① Nastavení počtu měření.

② Volání funkce **`ping_median()`**.

(Př. 1) Do obvodu s ultrazvukovým senzorem zapojte dvě LED (červenou, zelenou). Červená dioda se rozsvítí, pokud bude překročena nastavená minimální vzdálenost např. 10 cm od překážky. Zelená dioda se rozsvítí při naměřené vzdálenosti větší jak 100 cm.

(Př. 2) K ultrazvukovému senzoru připojte LCD displej, na kterém se bude zobrazovat aktuální vzdálenost od překážky. Pro zapojení LCD displeje využijte kapitolu 6.

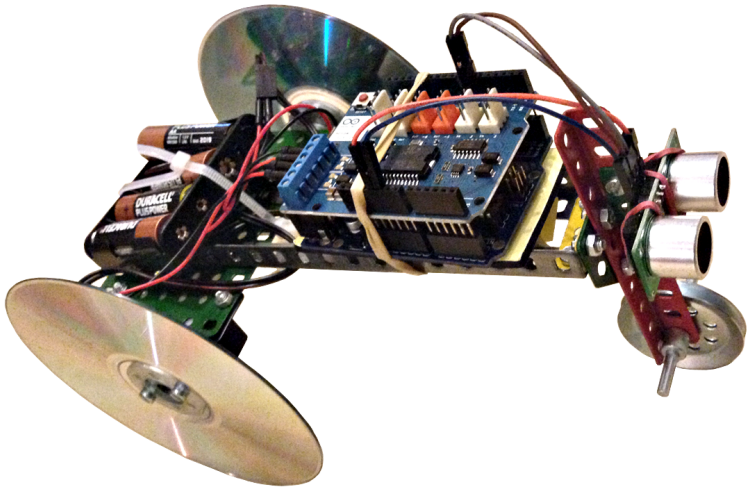
PROJEKT – ROBOTICKÉ VOZÍTKO

Pro vytvoření jednoduchého vozítka můžete využít vše co doma naleznete. Základem jsou dva stejnosměrné motory, které budou tvořit pohonný subsystém. Celková konstrukce může být tvořena například částmi různých stavebnic nebo může být složena z dřevěných prvků. V této části by měli žáci mít zcela volnou ruku. Je to ukázka STEM.

Základní komponenty: 2 stejnosměrné motory, ultrazvukový senzor, modul pro řízení motorů (L293D, nebo motor shield), baterie pro externí napájení, nepájivé pole a vodiče.

KONSTRUKCE

Ukázková konstrukce je složena z komponent stavebnice MERKUR. Jako kola jsou použity staré CD nosiče Obr. 4 – .

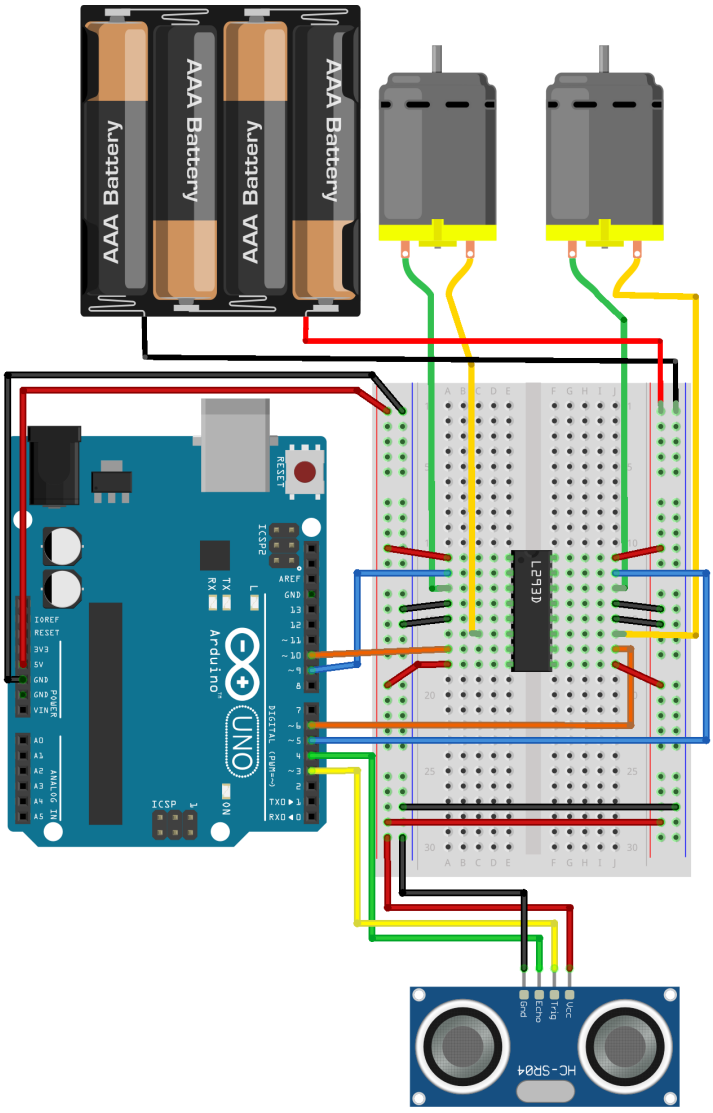


Obr. 4 – Robotický podvozek

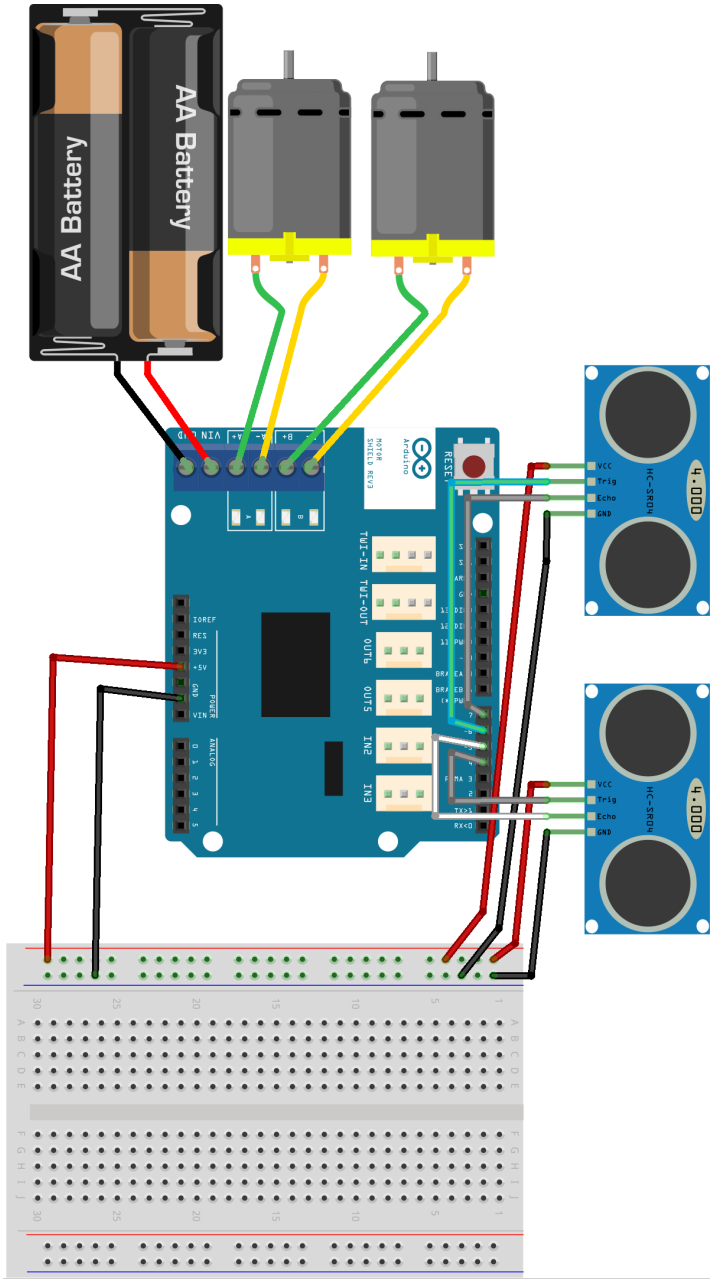
Robotický prostředek by se měl vyhýbat překážkám, které jsou detekovány v určité vzdálenosti pomocí ultrazvukového senzoru. Vyhýbání bude realizováno protisměrným otáčením motorů. Při využití jediného ultrazvukového senzoru se bude robotický prostředek uhýbat vždy na jednu stranu. Vylepšení pak spočívá ve využití dvou těchto senzorů, kdy lze zohlednit i umístění překážky a následně se rozhodnout v jakém směru se prostředek překážce vyhne.

SCHÉMA ZAPOJENÍ

Níže uvedené schéma využívá pro řízení motorů obvodu L293D. Zapojení je tím trochu složitější, ale finančně méně náročné Obr. 5. Při použití pole pro řízení motorů, které se přidává na desku Arduino formou přídavného modulu, se zapojení celého obvodu výrazně zjednoduší Obr. 6.



Obr. 5 – Schéma zapojení



Obr. 6 – Schéma zapojení s motorovým polem a dvěma senzory

PROGRAMOVÝ KÓD

V programovém kódu jsou použity pouze ty konstrukce, které jsou dostupné v předchozích kapitolách.

```
1   const int trigPin = 3;
2   const int echoPin = 4;
3   const int leftForward = 9;
4   const int leftBackward = 10;
5   const int rightForward = 5;
6   const int rightBackward = 6;
7
8   int duration = 0;
9   int distance = 0;
10
11  void setup() {
12    pinMode(trigPin, OUTPUT);
13    pinMode(echoPin, INPUT);
14    pinMode(leftForward, OUTPUT);
15    pinMode(leftBackward, OUTPUT);
16    pinMode(rightForward, OUTPUT);
17    pinMode(rightBackward, OUTPUT);
18    Serial.begin(9600);
19  }
20
21  void loop() {
21   digitalWrite(trigPin, HIGH);
23   delayMicroseconds(10);
24   digitalWrite(trigPin, LOW);
25
26   duration = pulseIn(echoPin, HIGH);
27   distance = duration * 0.034 / 2;
28
29   if (distance < 20) {
30     digitalWrite(leftForward, LOW);
31     digitalWrite(leftBackward, HIGH);
32     digitalWrite(rightForward, HIGH);
33     digitalWrite(rightBackward, LOW);
34     delay(100);
35   }else{
36     digitalWrite(leftForward, HIGH);
37     digitalWrite(leftBackward, LOW);
38     digitalWrite(rightForward, HIGH);
42     digitalWrite(rightBackward, LOW);
43   }
44  }
45
```

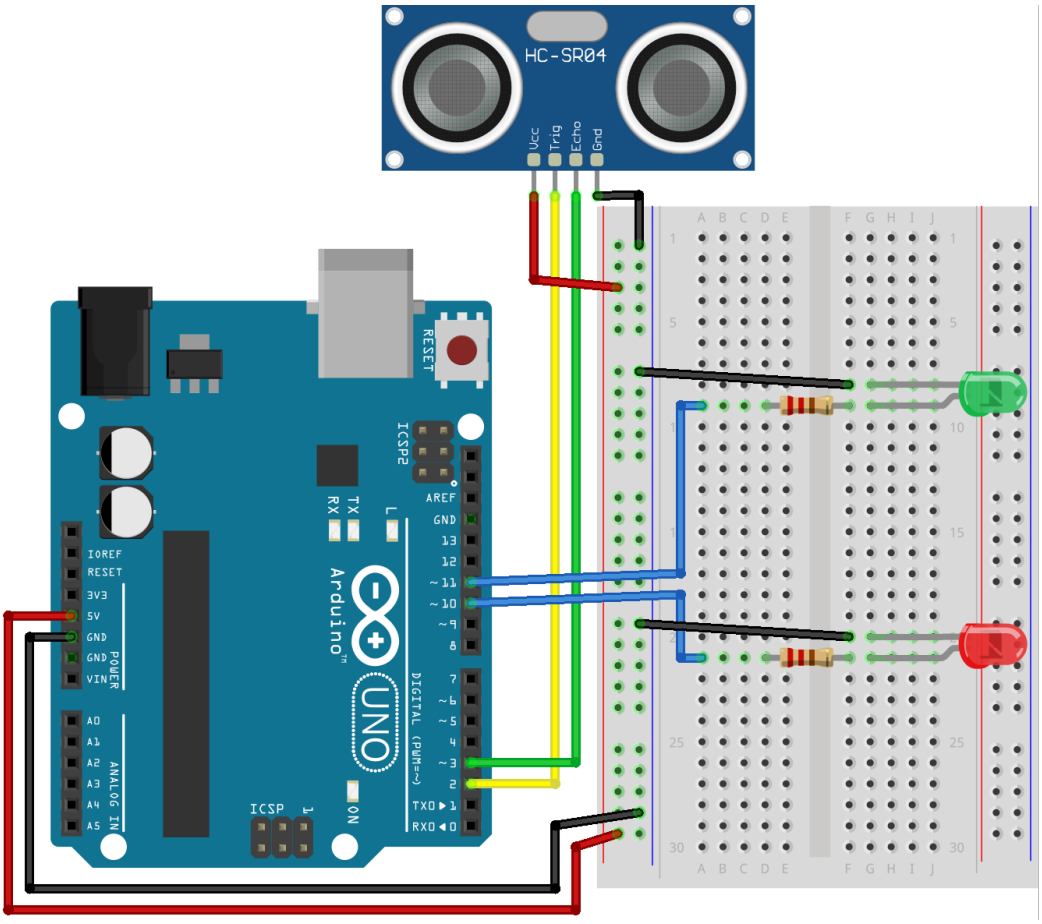
① ② ③ ④ ⑤ ⑥ ⑦ ⑧

① Definice pinů ultrazvukového senzoru.
② Definice pinů pro levý motor. Vždy se definují piny pro oba směry – dopřední a zpětný.
③ Definice pinů pro pravý motor.
④ Nastavení všech použitých pinů jako vstupní **`INPUT`** a výstupní **`OUTPUT`**.
⑤ Inicializace ultrazvukového senzoru.
⑥ Výpočet aktuální vzdálenosti předmětu před ultrazvukovým čidlem.
⑦ Na základě definované podmínky dochází ke změně směru otáčení kol. Pokud bude vzdálenost od překážky menší než vzdálenost uvedená v podmínce **`if`**, nastaví se hodnota pinů tak, aby se motory otáčely proti sobě. Tím dochází k otáčení robotického prostředku.
⑧ V opačném případě se robotický prostředek pohybu přímo v dopředném směru.

ŘEŠENÍ PŘÍKLADŮ

PŘÍKLAD 1

Obvod byl upraven tak, že byly přidány dvě diody a rezistory. Tento příklad může být mezistupněm před závěrečným projektem, protože si zde žáci procvičí podmínkový příkaz **`if`** a práci se vstupy.



Obr. 7 - Zapojení motoru s fotorezistorem

Programový kód využívá základního příkladu a může být pozměněn tak, že se využije externí knihovna **`NewPing`**. To může být dalším samostatným příkladem.

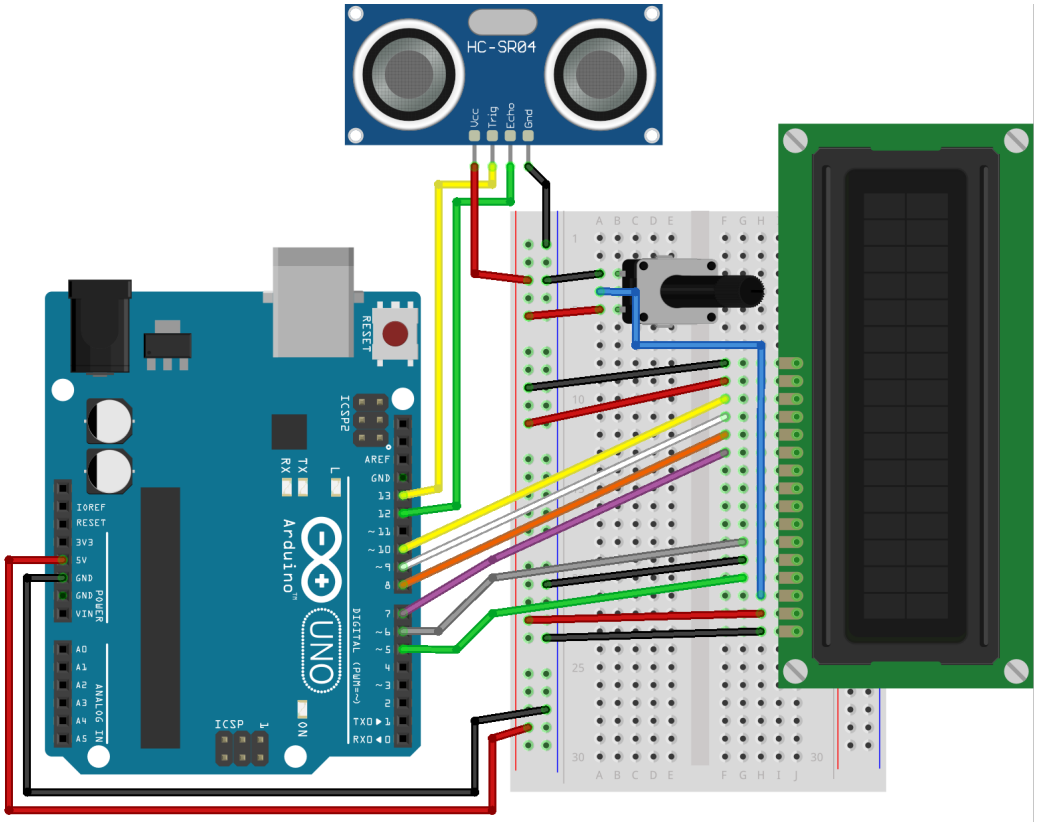
```
#define trigPin 2
#define echoPin 3
#define ledMin 10
#define ledMax 11
int maximumRange = 200;
int minimumRange = 0;
long duration, distance;

void setup(){
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledMin, OUTPUT);
  pinMode(ledMax, OUTPUT);
}

void loop(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034/2;

  if(distance < 10){
    digitalWrite(ledMin,HIGH);
    digitalWrite(ledMax,LOW);
  }else if(distance > 100){
    digitalWrite(ledMax,HIGH);
    digitalWrite(ledMin,LOW);
  }else{
    digitalWrite(ledMin,LOW);
    digitalWrite(ledMax,LOW);
  }
  if(distance>=maximumRange || distance<=minimumRange){
    Serial.println("Mimo dosah");
  }else{
    Serial.print(distance);
    Serial.println(" cm");
  }

  delay(500);
}
```

①
②
③
④
⑤
⑥
⑦
⑧
⑨

① Definice pinů pro ultrazvukový senzor.
② Definice pinů pro LED. Pin 10 je určen pro červenou a pin 11 pro zelenou diodu.
③ Definice proměnných, které vymezují minimální a maximální okruh snímání.
④ Přiřazení typu pinů pro ultrazvukový senzor a LED.
⑤ Zjištění dat pro výpočet vzdálenosti a samotný výpočet vzdálenosti překážky od ultrazvukového senzoru.
⑥ První část podmínky. Jestliže bude vzdálenost předmětu od překážky menší jak 10 cm, potom se rozsvítí červená dioda a zelená bude zhasnutá.
⑦ Pokud bude vzdálenost od překážky větší jak 100 cm, potom se rozsvítí zelená dioda a červená zhasne.
⑧ Zde se jedná o normální provoz, kdy se překážka pohybuje mezi stanovenými limity, tj. mezi 10 a 100 cm.
⑨ Pouze informativní věc, pokud se pohybuje překážka mimo stanovený rozsah.

PŘÍKLAD 2

Pro řešení tohoto příkladu lze vycházet z kapitoly zabývající se zobrazováním tepoty a vlhkosti na LCD displeji – lekce 6. Uvedená čidla budou nahrazena ultrazvukovým a na LCD displeji se bude zobrazovat vzdálenost od překážky.



Obr. 8 - Zapojení čidla a LCD displeje

Programový kód opět využívá základního příkladu a může být pozměněn tak, že se využije externí knihovna **`NewPing`**. To může být dalším samostatným příkladem.

```
1
2  #include <LiquidCrystal.h>                                          ①
3
4  const int trigPin = 3;
5  const int echoPin = 4;
6
7  const int rsPin = 5;
8  const int ePin = 6;
9  const int d4Pin = 7;
10 const int d5Pin = 8;                                               ②
11 const int d6Pin = 9;
12 const int d7Pin = 10;
13 LiquidCrystal LCD(rsPin,ePin,d4Pin,d5Pin,d6Pin,d7Pin);
14
15 void setup()
16 {
17   pinMode(trigPin, OUTPUT);                                        ③
18   pinMode(echoPin, INPUT);
19
20   LCD.begin(16,2);
21   LCD.clear();                                                     ④
22   LCD.setCursor(0,0);
23   LCD.print("Vzdálenost: ");
24 }
25
26 void loop()
27 {
28   digitalWrite(trigPin, HIGH);
29   delayMicroseconds(10);
30   digitalWrite(trigPin, LOW);                                      ⑤
31   duration = pulseIn(echoPin, HIGH);
32   distance = duration * 0.034 / 2;
33
34   LCD.setCursor(13,0);
35   LCD.print(distance);                                             ⑥
36   LCD.print(" cm");
37
38   delay(10);
39 }
```

① Připojení knihovny pro práci s LCD displejem.
② Definice pinů pro LCD displej a inicializace třídy LCD.
③ Nastavení pinů pro ultrazvukový senzor.
④ Prvotní inicializace LCD displeje.
⑤ Aktivace ultrazvukového čidla a výpočet vzdálenosti.
⑥ Nastavení kurzoru na LCD displeji pro výpis aktuální hodnoty vzdálenosti.