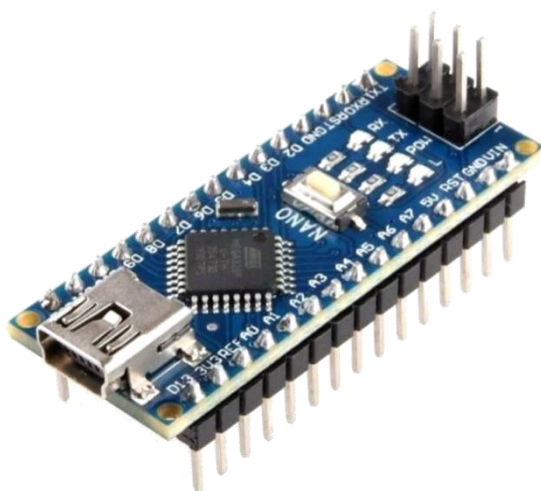




PRVNÍ PROJEKT



Obsah

1. Úvod.....	3
1.1. Co je to Arduino?	3
1.2. Existuje jen jedno Arduino?	3
1.3. Jak přimět Arduino pracovat?	3
1.4. Arduino IDE	3
1.5. První program	4
1.6. Externí LED	5
2. Základy programování.....	6
2.1. Struktura Arduino kódu.....	6
2.2. Základní pravidla psaní kódu v Arduino	7
3. Úlohy	8
3.1. Semafor	8
3.2. Čtení hodnot z potenciometru	8
3.2.1. Analogové/digitální piny; datové typy	8
3.2.2. Potenciometr a sériový monitor.....	9
3.2.3. Zapojení úlohy	9
3.2.4. Nastavení okna Serial monitor	10
3.3. Nastavení jasu LED, PWM.....	11
3.3.1. Co je to PWM?	11
3.3.2. Ovládání jasu LED potenciometrem.....	11
3.3.3. Mapování (map()) – Co to je a proč ho používat?	12
3.4. Řízení servomotoru	13
3.4.1. Knihovna Servo.h	13
3.4.2. Ovládání servomotoru na pevné úhly (0°, 90°, 180°).....	13
3.4.3. Ovládání serva potenciometrem.....	14
4. Řešené úlohy	15



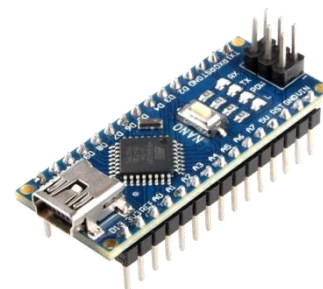
1. Úvod

1.1. Co je to Arduino?

Arduino je malý programovatelný počítač, tzv. mikrokontroler, který umožňuje číst hodnoty ze senzorů a potenciometrů na vstupech a řídit výstupy, jako jsou LEDky nebo motory. V jednoduchých aplikacích například ovládá světla, měří teplotu nebo reguluje otáčky motoru. Ve složitějších systémech může řídit nabíjení fotovoltaických panelů, měřit výkon a spotřebu energie, odesílat data přes internet, zpracovávat obraz či video nebo dálkově ovládat různá zařízení. Možnosti využití jsou téměř neomezené

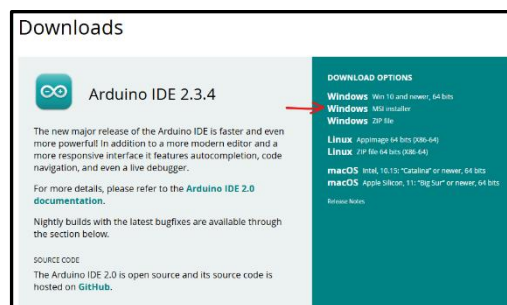
1.2. Existuje jen jedno Arduino?

Arduino existuje v mnoha variantách. My budeme používat Arduino Nano, které nabízí kompaktní rozměry, dostatek vstupních a výstupních pinů a snadné připojení k počítači. Arduino je open-source platforma, což znamená, že jeho návrh je veřejně dostupný. Díky tomu si kdokoli může navrhnout a vyrobit vlastní verzi Arduino. Každopádně proto existuje mnoho klonů, které jsou levnější než originální desky, i na tento kurz bude využit klon, který se však funkčně nijak neliší od originální verze.



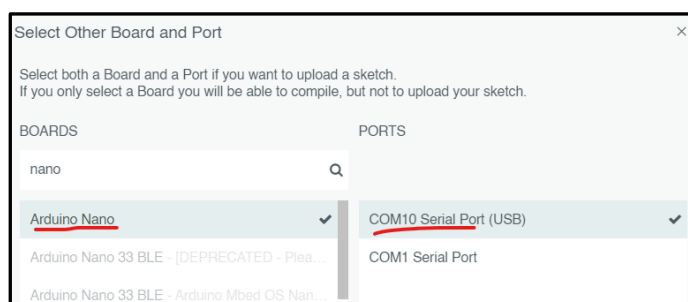
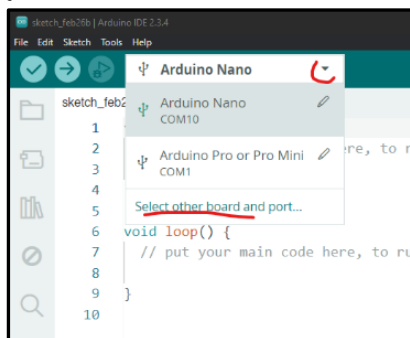
1.3. Jak přimět Arduino pracovat?

Aby Arduino vykonávalo požadované úlohy, je nutné napsat program (software). Programovací jazyk Arduino je odvozen od C++, což je výkonný, ale poněkud složitější jazyk. Naštěstí díky moderním nástrojům, jako je ChatGPT, se programování stává dostupné i pro začátečníky bez předchozích zkušeností. Programy pro Arduino se vytvářejí v prostředí Arduino IDE (Integrated Development Environment – Integrované vývojové prostředí). Arduino IDE lze stáhnout z oficiálních stránek: <https://www.arduino.cc/en/software>

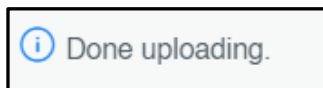
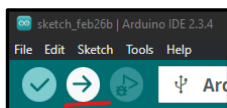


1.4. Arduino IDE

Připojte Arduino k PC pomocí USB kabelu. Spusťte Arduino IDE. V hlavním okně se zobrazí kostra programu obsahující dvě funkce void setup a void loop, kostru zatím nemažte bude se za chvíli hodit. Nyní v programu otevřete lištu pro vybrání typu Arduino a portu na které je Arduino připojené.



Pro ověření připojení k PC, stačí zmáčknout tlačítko Upload (šipku), která slouží k nahrání programu do Arduino. Pokud je připojení správné, zobrazí se v pravém dolním rohu zpráva Done uploading.

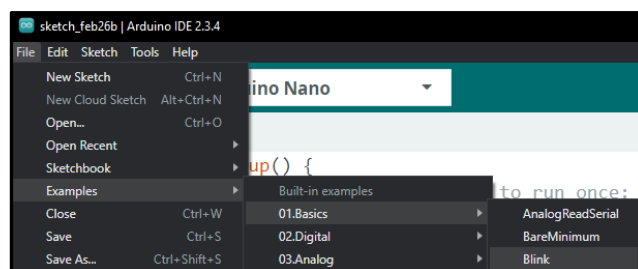


Verify – slouží ke kontrole kódu bez odeslání do Arduino

1.5. První program

Postupně se budeme učit jazyk Arduino, abyste si mohli psát vlastní programy. Než se do toho ale pustíme, vyzkoušíme si existující příklady, které Arduino IDE nabízí. Jako první úlohu si rozblíkáme vestavěnou LED na desce Arduino. Tento experiment nevyžaduje žádný další hardware, protože LED je přímo součástí desky. Příklad najdete v Arduino IDE:

File > Examples > 01.Basic > Blink



Otevře se nová skica s programem Blink. Nyní ji nahrajeme do Arduino. Dáme upload a sledujeme desku Arduino. Pokud je vše správně, jedna z LED bude pravidelně blikat. LED se rozsvítí na 1 sekundu a poté zhasne na 1 sekundu – tento cyklus se stále opakuje. Pokud se LED nerozblíká: Zkontrolujte, že máte správně vybranou desku a port v Tools > Board / Port. Pokud se objeví chyba při nahrávání, zkuste odpojit a znovu připojit Arduino a zopakovat nahrání.

Ted' si vyzkoušíme upravit kód, ale protože příklady (Examples) jsou jen pro čtení (Read-Only), nemůžeme je upravovat přímo. Proto si vytvoříme vlastní kopii.

Otevřete novou skicu

- Klikněte na File > New Sketch (Nová skica).

Zkopírujte kód z příkladu Blink

- Stiskněte Ctrl+A (vybrat vše), poté Ctrl+C (kopírovat).

Vložte kód do nové skici

- Přepněte se do nově otevřené skici.
- Stiskněte Ctrl+V (vložit).

Vytvořte složku pro úlohy

- Na svém disku vytvořte složku, kam budete ukládat všechny Arduino projekty (např. Arduino_úlohy).

Uložte skicu do složky

- Klikněte na File > Save As... a uložte skicu do složky s úlohami.
- Pojmenujte ji například 01_Blink.

Najděte v kódu řádky:

```
delay(1000);           // wait for a second
```

Změňte v obou řádcích hodnotu 1000 na 200 a nahrajte program znovu. Co se změnilo? LED teď bliká rychleji!

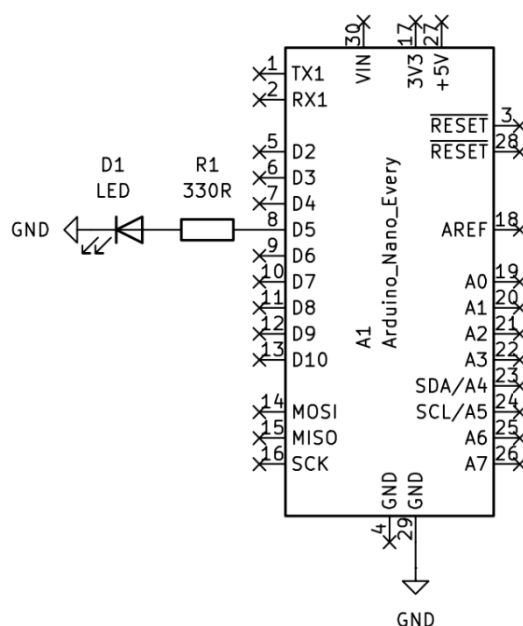
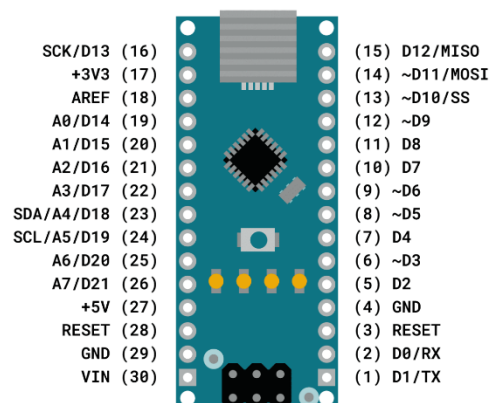
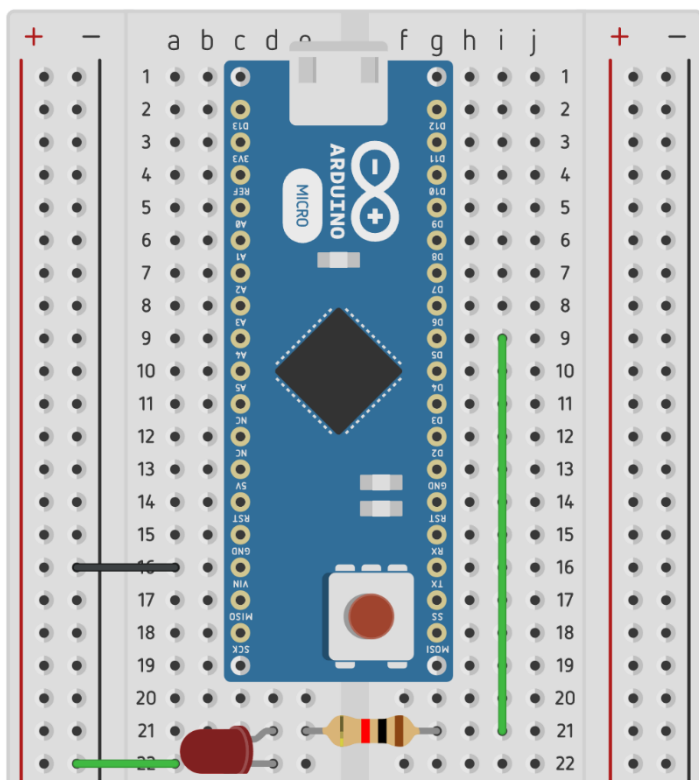
Nyní zkuste kód upravit tak aby se LED rozsvítila na 1s zhasla na 1s, pak se rozsvítila na 3s a zhasla na 3s.



1.6. Externí LED

Ted' si místo vestavěné LED zkusíme rozblikat externí LED. Budeme potřebovat

- 1× LED
- 1× rezistor 330 Ω
- Propojovací vodiče
- Nepájivé pole (Breadboard)
- Zapojte dle schématu



V předchozí úloze jsme ovládali vestavěnou LED, nyní budeme ovládat externí LED připojenou k pinu D5. Pro omezení proudu LED vložíme předřadný rezistor 330R. Nyní je nutné softwaru říct, který pin má ovládat. Pin má sice číslo 9, ale abychom psali co nejčitelnější kód, je dobré používat výstižné názvy. Proto si nadefinujeme nový název pro pin 9. Na začátek kódu přidáme řádek:

```
#define LED_PIN 9
```

Díky tomu můžeme v programu psát LED_PIN místo čísla 9, což kód zpřehlední. Protože nyní má software ovládat externí LED místo vestavěné, je třeba v celém kódu nahradit LED_BUILTIN na LED_PIN. Pro lepší přehlednost je také vhodné smazat všechny šedé komentáře, které nejsou nezbytné k pochopení kódu.

```
1  #define LED_PIN 9
2
3  void setup() {
4
5      pinMode(LED_PIN, OUTPUT);
6  }
7
8  // the loop function runs over
9  void loop() {
10     digitalWrite(LED_PIN, HIGH);
11     delay(1000);
12     digitalWrite(LED_PIN, LOW);
13     delay(1000);
14     digitalWrite(LED_PIN, HIGH);
15     delay(3000);
16     digitalWrite(LED_PIN, LOW);
17     delay(3000);
18 }
```



2. Základy programování

2.1. Struktura Arduino kódu

Než se pustíte do následující kapitoly, mějte na paměti, že programování je jako učení se novému jazyku – chce to čas a praxi. Je naprosto normální, pokud nepochopíte všechno hned. Doporučuji kapitolu nejprve projít a pak se k ní vracet během praktických úloh. Čím více času s programováním strávíte, tím si přirozeně všechno sedne a začne dávat smysl!

Každý program pro Arduino má tři hlavní části:

Deklarace

Tato úvodní část kódu se používá se pro definování smysluplnější názvy pinů a proměnných: Příklad deklarace:

```
1  #define LED_PIN 9
```

Nyní se pin 9 jmenuje výstižněji LED_PIN

Nastavení programu - setup()

Tato funkce se spustí pouze jednou po zapnutí Arduina. Používá se pro nastavení pinů, inicializaci sériové komunikace, další jednorázová nastavení

Příklad setup():

```
void setup() {  
  |  pinMode(LED_PIN, OUTPUT);  
}
```

Funkce pinMode nastaví, že pin který se jmenuje LED_PIN, bude fungovat jako výstup signálu.

Hlavní smyčka programu – loop()

Funkce loop() se spouští stále dokola, dokud je Arduino zapnuté. Čte hodnoty ze senzorů, ovládá výstupy (LED, motory...), vykonává opakující se úlohy

Příklad loop():

```
void loop() {  
  digitalWrite(LED_PIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_PIN, LOW);  
  delay(1000);  
  digitalWrite(LED_PIN, HIGH);  
  delay(3000);  
  digitalWrite(LED_PIN, LOW);  
  delay(3000);  
}
```

LED bliká, protože loop() běží neustále dokola. Funkce digitalWrite zapíše, že pin s názvem LED_PIN je buď zapnutý (HIGH) nebo vypnutý (LOW). Funkce delay slouží jako zpoždění, kdy kód nic nedělá, pouze čeká.

Také by mělo být nyní jasné, že základní časový krok, v kterém Arduino pracuje je 1ms.



2.2. Základní pravidla psaní kódu v Arduino

Kód pro Arduino je napsaný v jazyce C/C++, který má několik klíčových pravidel:

Velikost písmen –

Veškeré funkce musí být správně nazvané včetně velikosti písmen

Správně

Špatně

```
digitalWrite(LED_PIN, LOW); digitalWrite(LED_PIN, LOW);
```

Středníky ; – Označení konce příkazu

V jazyce C/C++ se každý příkaz ukončuje středníkem ;.

Správně

Špatně

```
digitalWrite(LED_PIN, LOW); digitalWrite(LED_PIN, LOW)
```

Poznámka: Středníky se nepišou za názvy funkcí (void setup() {}) ani za složené závorky {}.

Složené závorky {} – Bloky kódu

Složené závorky označují začátek a konec bloku kódu, například funkcí nebo podmínek.

Správně

Špatně

```
void setup() { void setup()
| pinMode(LED_PIN, OUTPUT); | pinMode(LED_PIN, OUTPUT);
}
```

Tip: Každá { musí mít svou }. Pokud chybí, kód nebude fungovat.

Struktura funkcí – Jak fungují?

Funkce v Arduino (např. setup() a loop()) mají následující strukturu:

```
navratovy_typ nazev_funkce() {
    // Tady je obsah funkce
}
```

- Návrátový typ – Říká, co funkce vrací (void = nic nevrací).
- Název funkce – Může být cokoliv, co dává smysl.
- Závorky () – Uvnitř mohou být parametry (např. čísla).
- Složené závorky {} – Obsahují kód, který funkce vykonává.

Příklad vlastní funkce:

```
void rozsvitLED() {
    digitalWrite(LED_PIN, HIGH);
}
```

Tato funkce rozsvítí LED, když ji zavoláme:

```
rozsvitLED(); // Zavolání funkce v loop()
```

Komentáře – Jak psát poznámky do kódu?

Komentáře jsou části kódu, které se neprovádějí, ale slouží jako vysvětlující poznámky

Jednořádkový komentář (//)

```
digitalWrite(LED_PIN, HIGH); // Zapnutí LED
```

Víceřádkový komentář (/* ... */)

```
/*
Tento kód rozsvítí LED na 1 sekundu,
poté ji zhasne na 1 sekundu.
*/
void loop() {
    digitalWrite(LED_PIN, HIGH);
    delay(1000);
    digitalWrite(LED_PIN, LOW);
    delay(1000);
}
```



3. Úlohy

3.1. Semafor

Cíl úlohy:

Vytvořte jednoduchý semafor pro auta, který bude řídit tři LED diody – červenou, oranžovou a zelenou. LED se budou automaticky přepínat v daném pořadí a s odpovídajícími časovými intervaly.

LED budou svítit v tomto pořadí:

- Zelená svítí 5 sekund
- Oranžová svítí 2 sekundy
- Červená svítí 5 sekund
- Oranžová svítí 2 sekundy
- Cyklicky se opakuje

To-Do List

Definovat piny (#define)

setup() – Nastavení pinů (pinMode()),

loop() – Hlavní část programu, kde běží logika.

Komentáře k důležitým částem kódu.



ChatGPT-4o

Zkuste na řešení přijít sami, pokud však nevíte nebo program nefunguje, ChatGPT je skvělý partner pro jednoduché programovací úlohy. Můžete AI rovnou požádat ať vám kód vytvoří. Nebo nakopírujete svoji verzi kódu a požádejte ji o radu bez toho aby vám AI kód vyřešila.

3.2. Čtení hodnot z potenciometru

3.2.1. Analogové/digitální piny; datové typy

Doposud jsme ovládali LED, která měla dva stavy – buď HIGH, nebo LOW. Pokud zapisujeme nebo čteme pouze dva stavy, používáme digitální piny. Pokud bychom však chtěli číst hodnoty z teploměru nebo potenciometru, je třeba využít analogové piny. Na nich je Arduino schopno číst široký rozsah hodnot v rozmezí 0 až 1023, odpovídající napětí 0–5V.

```
int stavTlacitka = digitalRead(9);
```

Čtení hodnoty (LOW/HIGH) z tlačítka na digitálním pinu 9

```
int Potenciometr = analogRead(A0);
```

Čtení hodnoty (0 až 1023) z analogového pinu A0

Co je to **int** ?

V programování se hodnoty ukládají do proměnných. Každá proměnná má typ, který říká, jaký druh dat může obsahovat.

int (integer = celé číslo) se používá pro celá čísla (např. -100, 0, 255, 1023).

```
int Potenciometr = analogRead(A0);
```

int Potenciometr - vytvoří proměnnou s názvem Potenciometr, která uchová celé číslo
analogRead(A0) - Arduino přečte hodnotu z potenciometru (0–1023)
= znamená, že hodnota (0–1023) se uloží do proměnné



Arduino je schopno pracovat jak s digitálními vstupy i výstupy, ale analogové hodnoty je schopno pouze číst. Jaké piny jsou tedy vstupy a výstupy?

Typ pinu	Použití	Typ signálu	Typická funkce
Digitální (D0–D13)	Vstup/Výstup	Digitální (0/1)	digitalRead(), digitalWrite()
Analogové (A0–A5)	Pouze vstup	Analogový (0–1023)	analogRead()

3.2.2. Potenciometr a sériový monitor

Potenciometr je nastavitelný rezistor, který při zapojení na napájení funguje jako dělič napětí. Proměnná napětí se typicky mění na prostřední nožičce potenciometru, tuto hodnotu je možné číst na analogovém pinu, například A0.

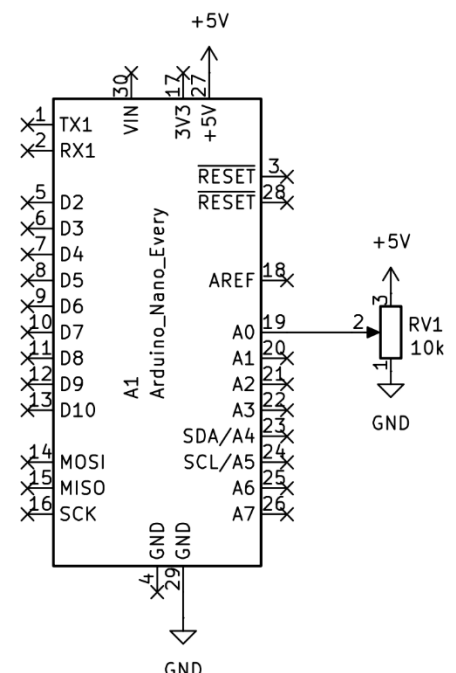
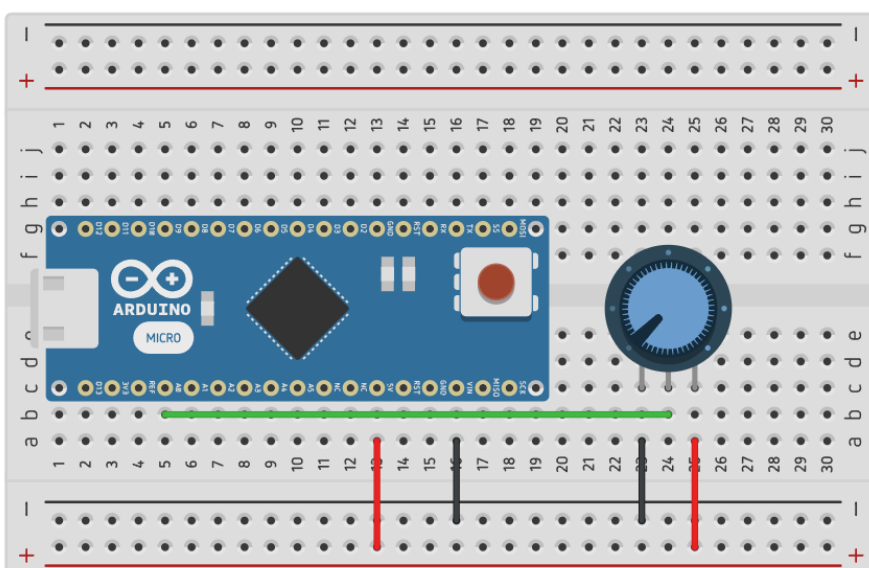
Aby bylo možné vyčítat hodnoty z Arduino v prostředí Arduino IDE k tomu je určená vestavěná pro komunikaci s počítačem, která se nazývá Sériový monitor. Pomocí něj můžeme zobrazovat hodnoty ze senzorů, zjišťovat co se v Arduino děje, posílat příkazy z počítače do Arduino. Aby Arduino posílalo zprávy do sériového monitoru, musíme v setup() zapnout sériovou komunikaci:

```
void setup() {
  Serial.begin(9600); // Spustí sériovou komunikaci na rychlosti 9600 baudů
}
```

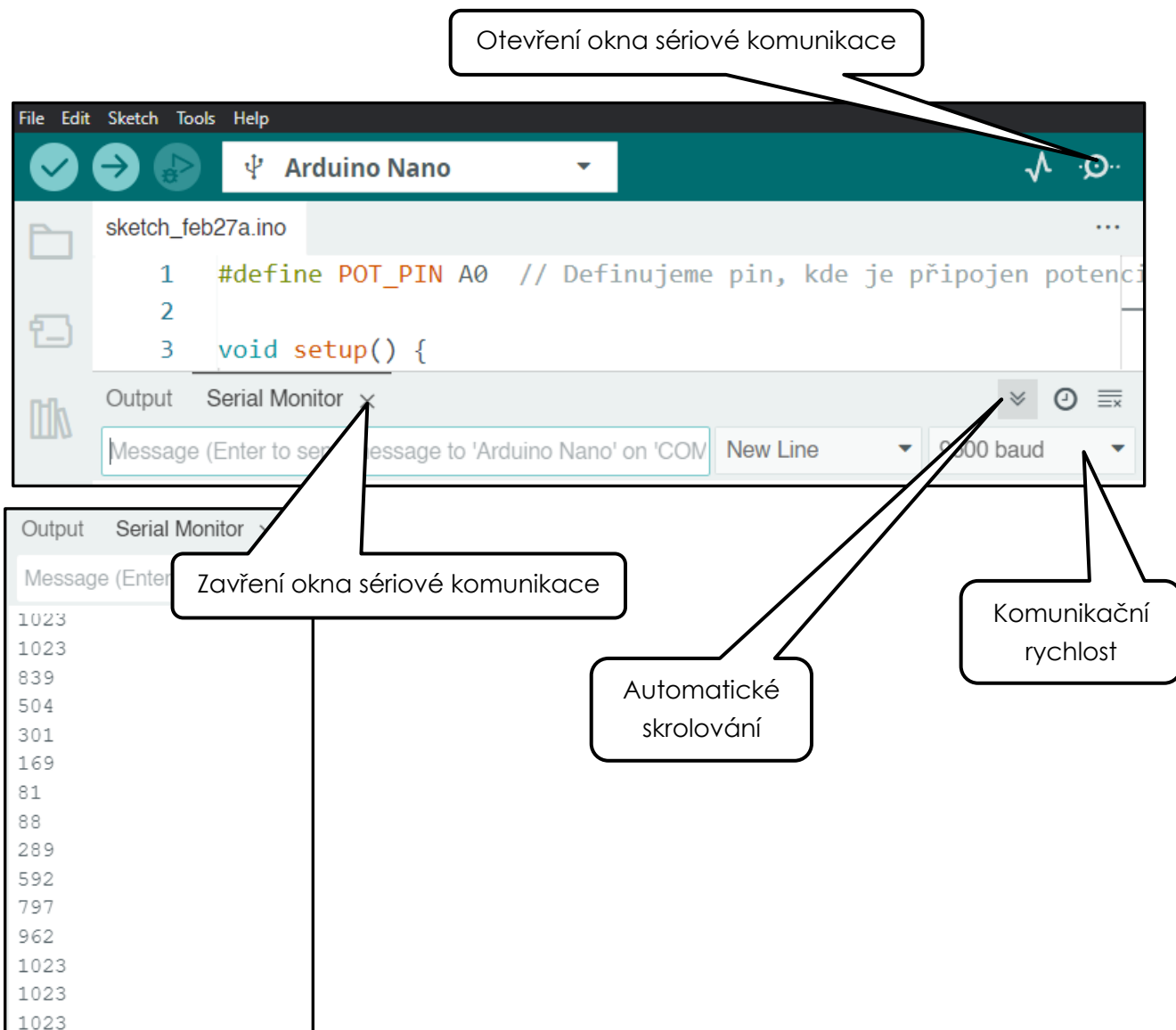
Když je funkce pro komunikaci zapnutá, musíme Arduino říct, kdy a jaká data má do posílat. To se provádí buď funkcí Serial.print nebo Serial.println. Serial.print umožňuje vypisovat více hodnot do jednoho řádku, println ukončuje řádek. Pokud chci vypsát jakýkoliv text, stačí ho dát do uvozovek např.: Serial.print("Potenciometr: ");

```
void loop() {
  int Potenciometr = analogRead(A0); // Čtení hodnoty potenciometru
  Serial.println(hodnota);           // Výpis do sériového monitoru
  delay(500);                       // Pauza pro lepší čitelnost výpisu
}
```

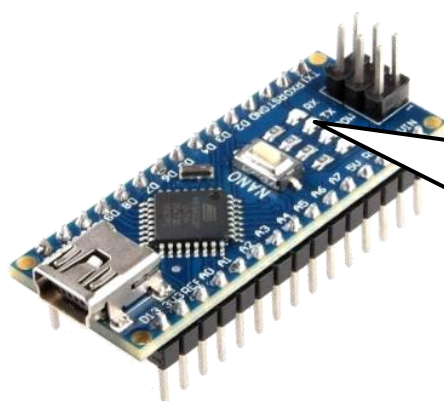
3.2.3. Zapojení úlohy



3.2.4. Nastavení okna Serial monitor



Výpis analogových hodnot v sériovém monitoru při točení s potenciometrem.



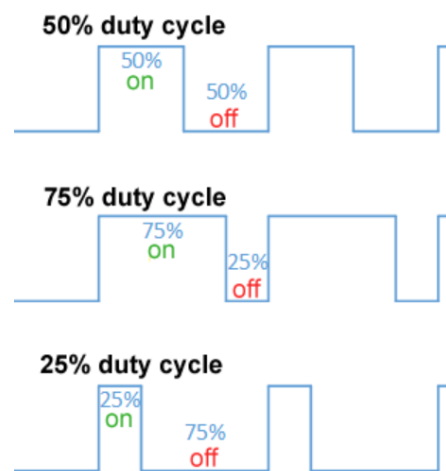
Měla by velmi rychle blikat vestavěná LED označená TX, (T – transmit, vysílání). Lze tedy vidět, že každých 100ms, Arduino odešle data do PC.



3.3. Nastavení jasu LED, PWM

3.3.1. Co je to PWM?

Doposud jsme LED pouze zapínali (HIGH) nebo vypínali (LOW), ale co když chceme nastavit její jas plynule? Řešením je PWM (Pulse Width Modulation, Pulzně šířková modulace). PWM funguje tak, že LED velmi rychle bliká mezi 0 a 1, ale tak rychle, že to lidské oko nevnímá. Místo toho vidíme průměrný jas, který závisí na poměru doby, kdy je LED zapnutá a vypnutá neboli jaká je střída signálu (duty cycle). Duty cycle 75 % LED vypadá jasnější. Duty cycle 25 % LED vypadá tlumenější. I když Arduino nemá skutečné analogové výstupy, umí to obejít právě pomocí PWM. Tímto způsobem dokáže napodobit plynulou změnu napětí a umožňuje řídit jas LED nebo výkon motoru. Piny, které umí PWM jsou D3, D6, D9, D10, D11. Rozsah hodnot, které je možné pomocí funkce `analogWrite` do pinu zapisovat je v rozsahu 0 až 255.

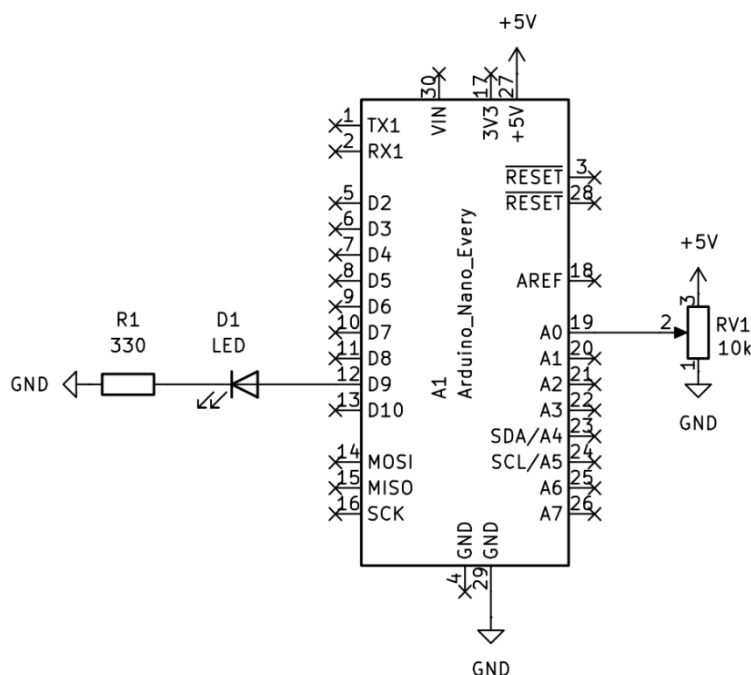


Arduino nemá skutečné analogové výstupy, které by uměli plynule měnit hodnotu napětí 0 až 5V. Důvodem je, že Mikrokontrolery v Arduino (např. ATmega328) neobsahují digitálně-analogový převodník (DAC), který by umožňoval vytvářet skutečně plynulé napětí. DAC je složitější obvod, který vyžaduje další hardware a Arduino je navrženo tak, aby bylo jednoduché a levné.

3.3.2. Ovládání jasu LED potenciometrem

Cíl úlohy:

V této úloze vytvoříte stmívač LED, kde se jas LED diody bude plynule měnit podle polohy potenciometru. LED bude připojena na PWM pin a její jas se bude řídit potenciometrem. Potenciometr bude připojen k analogovému vstupu (A0), aby Arduino mohlo číst jeho hodnotu. Hodnoty potenciometru (0–1023) se převedou na rozsah vhodný pro řízení jasu LED (0–255).



To-Do List

Definování pinů (#define)

- Definovat pin 9 jako LED_PIN (PWM výstup pro LED).
- Definovat pin A0 jako POT_PIN (analogový vstup pro potenciometr).

setup()

- Nastavit LED_PIN jako výstup (OUTPUT) pomocí pinMode().
- Spustit sériový monitor Serial.begin(9600); pro výpis hodnot.

loop()

- Odečíst analogovou hodnotu z potenciometru pomocí analogRead(POT_PIN)
- Toto celé číslo uložit do proměnné int hodnota.
`int hodnota = analogRead(POT_PIN);`
- proměnná hodnota je nyní v rozsahu 0-1023, jenže PWM funguje v rozsahu 0-255. Výhodou je že to je přesně 4x méně. Proto vytvoříme novou celočíselnou (int) proměnnou s názvem jas, která bude 4x menší než proměnná hodnota
`int jas = hodnota / 4;`
- Zapsat do LED_PIN, proměnnou jas
`analogWrite(LED_PIN, jas);`
- Poslat do Arduino IDE hodnoty proměnných hodnota a jas.
`Serial.print("Potenciometr: ");`
`Serial.print(hodnota);`
`Serial.print(" Jas LED: ");`
`Serial.println(jas);` // Všimněte si, že je rozdíl Serial.print vs Serial.println „ln“ je pro ukončení řádku, pak data v sériovém monitoru budou vypadat takto
`Potenciometr: 606 Jas LED: 151`
- Přidat krátké 10ms zpoždění (delay(10);) pro stabilní chod.

3.3.3. Mapování (map()) – Co to je a proč ho používat?

Potenciometr vrací hodnoty 0–1023, ale PWM pro LED pracuje s 0–255. V první verzi jsme to vyřešili ručním dělením / 4, ale co když bychom chtěli jiný rozsah? Nebo kdyby potenciometr měl jiný výstup než 0-1023?

Řešení: Funkce map() ta automaticky převede hodnotu z jednoho rozsahu do jiného.

Obecná syntaxe:

```
map(hodnota, vstup_min, vstup_max, výstup_min, výstup_max);
```

V našem případě:

```
int jas = map(hodnota, 0, 1023, 0, 255);
```

Tento příkaz zajistí, že hodnota 0 z potenciometru bude 0 pro LED a hodnota 1023 bude 255 pro LED. Vše mezi tím se přepočítá automaticky.

Kdy je lepší map() a kdy ruční výpočet?

Metoda	Výhody	Nevýhody
Ruční dělení (/4)	Snadné, přesné pro tento konkrétní případ	Funguje jen u přesného poměru (např. 1023/4 = 255)
map()	Univerzální, přizpůsobitelné pro různé rozsahy	Nepodporuje desetinná čísla

Úkol: Nahradte ruční dělení funkce jas pomocí mapování



3.4. Řízení servomotoru

3.4.1. Knihovna Servo.h

Servomotor je druh motoru, který se neotáčí dokola, ale natáčí se pouze v určitém úhlu v našem případě 0°-180°. Pokud jste zvládli předchozí úlohy s PWM, máte už to nejtěžší za sebou! Servomotor se totiž také řídí pomocí PWM signálu, ale s tím rozdílem, že PWM neurčuje výkon (jako u LED), ale úhel natočení serva. Malý problém je v tom, že frekvence, kterou nyní PWM má není vhodná pro správnou funkci servomotoru. Snadné řešení nabízí použití knihovny pro servomotory, která automaticky upraví PWM signál tak, aby servomotor správně fungoval.



3.4.2. Ovládání servomotoru na pevné úhly (0°, 90°, 180°)

Cíl úlohy:

V této úloze budeme otáčet servomotor mezi třemi pevnými úhly – 0°, 90° a 180°. Servomotor se bude vždy natočen do jednoho z těchto úhlů, chvilku počká a poté se přesune do dalšího.

To-Do List

Načíst knihovnu, vytvořit proměnou

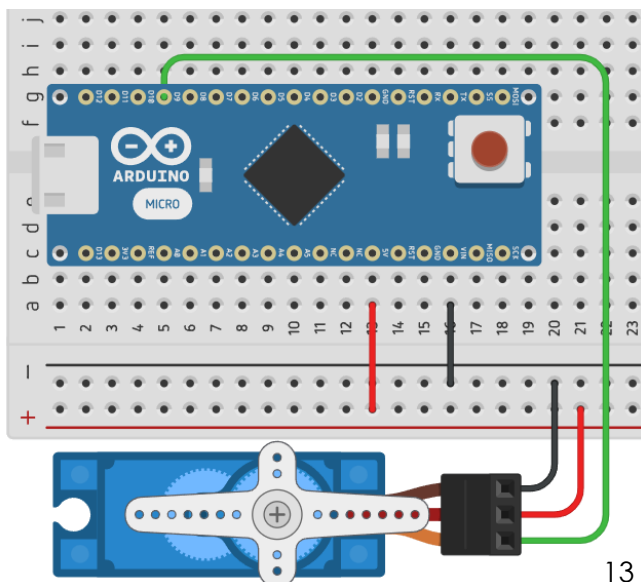
- Na začátek kódu načíst knihovnu
`#include <Servo.h>`
- Vytvořit proměnnou, která bude řídit servomotor
`Servo servoMotor;`

setup()

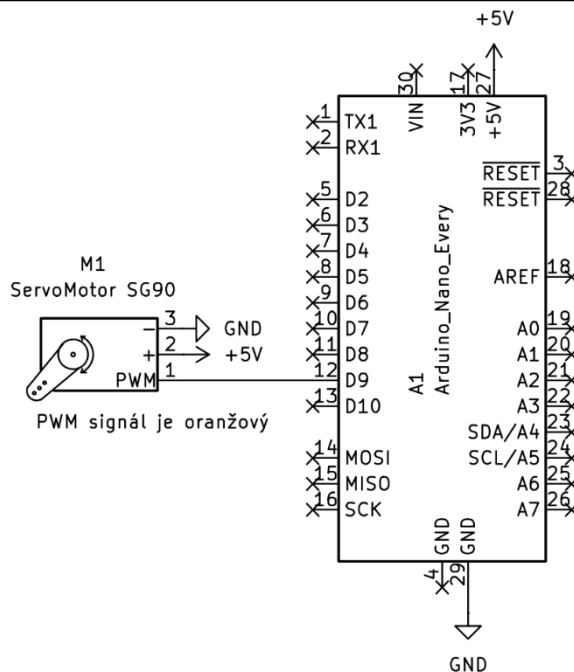
- Místo pinMode, použijeme pro přiřazení serva k pinu D9 funkci:
`servoMotor.attach(9);`

loop()

- Nastavit servo na 0°
`servoMotor.write(0);`
- Počkat 1 sekundu
`delay(1000);`
- Nastavit servo na 90°
- Počkat 1 sekundu.
- Nastavit servo na 180°
- Počkat 1 sekundu

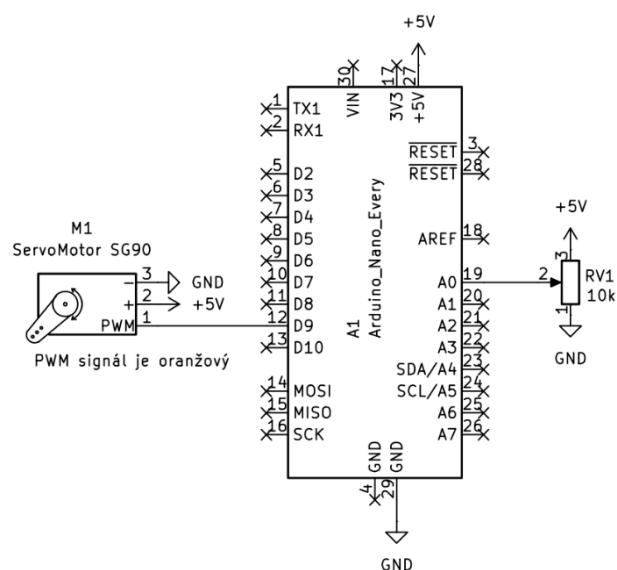


13



3.4.3. Ovládání serva potenciometrem

V předchozích úlohách jsme se naučili číst hodnoty z potenciometru a ovládat servomotor na pevné úhly (0°, 90°, 180°). Teď tyto znalosti spojíme – budeme plynule řídit úhel natočení servomotoru potenciometrem. Potenciometr bude sloužit jako řídicí prvek – otočením knoflíku budeme nastavovat úhel serva. Servo se bude otáčet v rozsahu 0°–180° podle hodnoty potenciometru. Použijeme `map()`, abychom převedli rozsah hodnot potenciometru (0–1023) na úhel serva (0–180°). Po dokončení této úlohy budete umět řídit servomotor intuitivně – otáčením potenciometru!



To-Do List

Načíst knihovnu, vytvořit proměnou

- Na začátek kódu Načíst knihovnu `Servo.h`
`#include <Servo.h>`
- Definovat piny pomocí `#define`
`POT_PIN = A0` //(analogový vstup pro potenciometr).
`SERVO_PIN = D9` //(PWM výstup pro servomotor).
- Vytvořit proměnnou, `;`, která bude řídit servomotor
`Servo servoMotor;`

setup()

- připojit servo na pin D9.
`servoMotor.attach(SERVO_PIN);`
- Spustit sériovou komunikaci připojit servo na pin D9.
`Serial.begin(9600);`

loop()

- přečíst hodnotu potenciometru (rozsah 0–1023).
`analogRead(POT_PIN);`
- Převést hodnotu potenciometru na úhel serva (0–180°) pomocí `map()`.
`int uhel = map(hodnota, 0, 1023, 0, 180);`
- nastavit servo na odpovídající úhel.
`servoMotor.write(uhel);`
- Vypsát hodnotu potenciometru a úhel serva do sériového monitoru (`Serial.print()` a `Serial.println()`).
- Přidat krátké zpoždění pro plynulejší chod."
`delay(10);`



4. Řešené úlohy

Úloha 3.2.2 Potenciometr a sériový monitor

```
#define POT_PIN A0 // Definujeme pin pro potenciometr

void setup() {
  Serial.begin(9600); // Spustíme sériovou komunikaci
}

void loop() {
  int hodnota = analogRead(POT_PIN); // Čtení hodnoty potenciometru (0–1023)
  Serial.print("Hodnota potenciometru: ");
  Serial.println(hodnota); // Výpis hodnoty do sériového monitoru
  delay(100); // Krátká pauza pro lepší čitelnost výpisu
}
```

Jak to funguje?

- #define POT_PIN A0; Definuje analogový pin A0, kde je připojen potenciometr.
- Serial.begin(9600); Spustí sériovou komunikaci na rychlosti 9600 baudů.
- analogRead(POT_PIN); Arduino přečte hodnotu napětí na pinu A0 (v rozsahu 0–1023).
- Serial.print("Hodnota potenciometru: "); Vypíše textový popis do sériového monitoru.
- Serial.println(hodnota); Vypíše přečtenou hodnotu potenciometru na nový řádek.
- delay(100); Přidá krátké zpoždění, aby výpis do monitoru nebyl příliš rychlý.



Úloha 3.3.2 Ovládání jasu LED potenciometrem

```
#define LED_PIN 9      // LED připojena na PWM pin
#define POT_PIN A0     // Potenciometr připojen na analogový vstup

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600); // Spustíme sériovou komunikaci pro kontrolu hodnot
}

void loop() {
  int hodnota = analogRead(POT_PIN); // Čtení potenciometru (0-1023)
  int jas = hodnota / 4; // Ruční převod rozsahu 0-1023 na 0-255

  analogWrite(LED_PIN, jas); // Nastavení jasu LED
  Serial.print("Potenciometr: ");
  Serial.print(hodnota);
  Serial.print("  Jas LED: ");
  Serial.println(jas); // Výpis hodnot do sériového monitoru

  delay(10); // Krátká pauza pro stabilní chod
}
```

Jak to funguje?

- #define LED_PIN (D9); Digitální výstup s podporou PWM (pro regulaci jasu LED).
- #define POT_PIN (A0); Analogový vstup, pro čtení hodnot z potenciometru.
- pinMode(LED_PIN, OUTPUT); Nastavíme pin 9 jako výstup pro LED.
- Serial.begin(9600); Spustí komunikaci pro výpis hodnot potenciometru a jasu LED.
- int hodnota = analogRead(POT_PIN); Přečte hodnotu potenc. v rozsahu 0-1023.
- int jas = hodnota / 4; Převod na rozsah 0-255 (protože $1023/4 \approx 255$).
- analogWrite(LED_PIN, jas); Ovládá jas LED tím, že nastaví PWM signál v rozsahu 0-255.
- Serial.print(); Vypisuje hodnoty potenciometru a jasu LED do sériového monitoru.
- Krátká 10ms pauza, aby výpis do sériového monitoru nebyl příliš rychlý.



Úloha 3.3.3 Mapování

```
#define LED_PIN 9
#define POT_PIN A0

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int hodnota = analogRead(POT_PIN); // Čtení potenciometru (0-1023)
  int jas = map(hodnota, 0, 1023, 0, 255); // Automatický převod rozsahů

  analogWrite(LED_PIN, jas);
  Serial.print("Potenciometr: ");
  Serial.print(hodnota);
  Serial.print("  Jas LED: ");
  Serial.println(jas);

  delay(10);
}
```

Úloha 3.4.2. Ovládání servomotoru na pevné úhly (0°, 90°, 180°)

```
#include <Servo.h> // Načtení knihovny pro servo

Servo servoMotor; // Vytvoření objektu pro servo

void setup() {
  servoMotor.attach(9); // Připojení serva na pin 9
}

void loop() {
  servoMotor.write(0); // Natočení serva na 0°
  delay(1000);         // Počkat 1 sekundu

  servoMotor.write(90); // Natočení serva na 90°
  delay(1000);         // Počkat 1 sekundu

  servoMotor.write(180); // Natočení serva na 180°
  delay(1000);         // Počkat 1 sekundu
}
```

Jak to funguje?

- #include <Servo.h> – Načte knihovnu pro servomotory.
- Servo servoMotor; – Vytvoříme objekt pro řízení serva.
- servoMotor.attach(9); – Řekneme Arduino, že servo je připojeno na pin D9.
- servoMotor.write(úhel); – Pošle servu signál, aby se otočilo na zadaný úhel.
- delay(1000); – Počkáme 1 sekundu mezi změnami úhlu.



Úloha 3.4.3 Řízení serva potenciometrem

```
#include <Servo.h> // Načtení knihovny pro servo

#define POT_PIN A0 // Potenciometr na analogovém pinu A0
#define SERVO_PIN 9 // Servo připojené na digitální pin D9 (PWM)

Servo servoMotor; // Vytvoření objektu pro servo

void setup() {
  servoMotor.attach(SERVO_PIN); // Připojení serva na pin 9
  Serial.begin(9600); // Spuštění sériové komunikace pro kontrolu hodnot
}

void loop() {
  int hodnota = analogRead(POT_PIN); // Čtení hodnoty potenciometru (0-1023)
  int uhel = map(hodnota, 0, 1023, 0, 180); // Převod rozsahu na úhel 0-180°

  servoMotor.write(uhel); // Nastavení serva na odpovídající úhel
  Serial.print("Potenciometr: ");
  Serial.print(hodnota);
  Serial.print(" Úhel serva: ");
  Serial.println(uhel); // Výpis hodnot do sériového monitoru

  delay(10); // Krátká pauza pro stabilní chod
}
```

Jak to funguje?

- `#include <Servo.h>` Načte knihovnu pro servomotor.
- `#define POT_PIN A0` a `#define SERVO_PIN 9`; Pojmenuje piny pro potenc. a servo.
- `Servo servoMotor`; Vytvoří objekt pro řízení serva.
- `servoMotor.attach(SERVO_PIN)`; Připojí servo na pin D9.
- `analogRead(POT_PIN)`; Přečte hodnotu potenciometru (0-1023).
- `map(hodnota, 0, 1023, 0, 180)`; Převeď rozsah potenciometru na úhel serva (0-180°).
- `servoMotor.write(uhel)`; Pošleme servu signál k otočení na daný úhel.
- `Serial.print()`; Vypíšeme hodnoty potenciometru a úhlu serva do sériového monitoru
- `delay(10)`; Přidá krátké zpoždění pro plynulejší chod.

