Pedagogická fakulta
Faculty of Education
Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Pedagogická fakulta
Katedra informatiky

Soubor videotutoriálů pro práci s platformou Arduino UNO

A set of video tutorials for working with the Arduino UNO platform

Bakalářská práce

Vypracoval: Lukáš Navrátil
Vedoucí práce: Michal Šerý Ing. Ph.D.

České Budějovice 2024

Prohlášení

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne:

_____

Lukáš Navrátil

Poděkování

Chtěl bych vyjádřit hluboké díky panu Ing. Michalu Šerému, Ph.D., za jeho neocenitelnou podporu a odborné vedení, které mi poskytl během psaní této bakalářské práce. Jeho odborné znalosti a cenné rady byly klíčové pro úspěšné dokončení mé práce.

Dále bych rád poděkoval své rodině a přátelům, kteří mi během studia poskytovali neustálou podporu a porozumění. Vaše morální podpora a praktická pomoc byly pro mě neocenitelné, obzvláště ve chvílích, kdy jsem se potýkal s výzvami.

Abstrakt/Anotace

Tato bakalářská práce se zaměřuje na popis a natáčení série video tutoriálů, které mají za úkol představit technologii Arduino UNO. Je primárně určena začátečníkům a mírně pokročilým uživatelům, kteří chtějí získat základní znalosti o této jednočipové platformě a problematice jejích senzorů. Obsah práce je rozdělen do několika videotutoriálů, z nichž každý trvá přibližně tři minuty, přičemž celková délka všech videí dosahuje zhruba šedesáti minut. Tutoriály byly natočeny na mobilní telefon a upraveny v programu Adobe Premiere Pro. Jsou strukturovány tak, že postupně představují jednotlivé komponenty desky Arduino a její zapojení, a dále pak přecházejí ke složitějším obvodům, jako je napájení, snímání světla, měření vlhkosti půdy, teploty a vlhkosti vzduchu, osvětlení a řízení krokových motorů.

Teoretická část práce poskytuje vysvětlení základních pojmů a podrobný popis všech komponent desky. Praktická část obsahuje jednotlivé tutoriály, které zahrnují úvodní seznámení s moduly a pojmy, přehled potřebných komponent a finální zapojení a programování v Arduino IDE a Tinkercadu.

Klíčová slova

Ardunio UNO, Adobe premiere pro

Abstract

This bachelor's thesis focuses on the description and production of a series of video tutorials, which aim to introduce the Arduino UNO technology. It is primarily intended for beginners and moderately advanced users who want to acquire basic knowledge of this single-chip platform and the issues related to its sensors. The content of the work is divided into several video tutorials, each lasting approximately three minutes, with the total length of all videos reaching about sixty minutes. The tutorials were recorded on a mobile phone and edited in Adobe Premiere Pro. They are structured in a way that gradually introduces the individual components of the Arduino board and its connections, and then progresses to more complex circuits such as power supply, light detection, soil moisture measurement, temperature and humidity sensing, lighting, and control of stepper motors.

The theoretical part of the thesis provides explanations of basic concepts and a detailed description of all the components of the board. The practical part contains individual tutorials that include an introductory acquaintance with modules and terms, an overview of the necessary components, and the final assembly and programming in Arduino IDE and Tinkercad.

Key word

Ardunio UNO, Adobe premiere pro, Arduino IDE, Tinkercad

OBSAH

1 Úvod

Tato bakalářská práce se zaměřuje na tvorbu série video tutoriálů, které mají za cíl podpořit studenty předmětu Arduino v kurzu vedeném panem doktorem Šerým. Tutoriály jsou navrženy tak, aby poskytovaly přehled o technologii Arduino UNO a byly vhodné pro začátečníky i mírně pokročilé uživatele. Obsah je strukturován do několika krátkých videí s délkou přibližně 3 minut každé (celková délka 60 minut). Tyto tutoriály jsou natáčeny pomocí mobilního telefonu a upravovány v programu Adobe Premiere Pro. Jednotlivé lekce postupně představují různé komponenty desky Arduino a jejich propojení a zabývají se i složitějšími obvody, jako je řízení napájení, detekce světla a řízení krokových motorů.

1.1 Východiska práce

Teoretická část této práce spočívá v detailním vysvětlení základních pojmů a podrobném popisu všech komponent desky Arduino UNO. Porozumění těmto základům je klíčové pro uživatele, kteří se chtějí efektivně seznámit s touto platformou. Praktická část práce je pak založena na vytvoření ukázek, které nejen uživatele zavádějí do modulů a jejich funkcí, ale také poskytují přehled o potřebných komponentech pro každou ukázku. Tím je zajištěno, že uživatelé mají solidní základy nezbytné k dalšímu praktickému využití Arduino.

Jednotlivé lekce jsou k dispozici na online webové stránce: http://home.pf.jcu.cz/~kyklop/SERYM/Ardu/videotut/index.html.

1.2 Cíle práce

Hlavním cílem této bakalářské práce je vybavit uživatele znalostmi a praktickými dovednostmi potřebnými k efektivní práci s Arduinem UNO a současně poukázat na problematiku některých senzorů a akčních členů. Dále se práce zaměřuje na seznámení komponentů s bezpečným zapojením obvodů na desce, přičemž minimalizuje riziko zkratu. Uživatelé budou také seznámeni s různými prostředími pro programování Arduina, jako jsou Arduino IDE, Tinkercad a Fritzing. Jednočipový počítač Arduino vytváří systém, který vyhodnocuje a porovnává data z různých senzorů a poté provádí další operace, definované uživatelem.

Bakalářská práce se skládá z teoretické a praktické části. V teoretické části popisujeme jednotlivé prvky použité v různých obvodech, které byly využity při tvorbě návodů, jako jsou napájecí modul, modul snímání světla, zalévání, měření teploty a vlhkosti vzduchu, osvětlení, měření vlhkosti půdy a řízení krokových motorů. Praktická část obsahuje video návody na moduly již zmíněné v teoretické části.

V závěru práce se zaměřujeme na představení možností rozšíření a dalšího využití získaných znalostí a dovedností v oblasti práce s Arduinem UNO. Diskutujeme o potenciálních aplikacích v různých odvětvích, jako je automatizace, robotika, Internet věcí (IoT) a další. Dále naznačujeme možnosti dalšího studia nebo specializace v této oblasti, aby uživatelé mohli rozvíjet své dovednosti a dosáhnout pokročilejších úrovní znalostí v oblasti hardwarového a softwarového vývoje s Arduinem.

2 Arduino

2.1 Historie Arduino

Arduino je vnímáno jako fenomén nebo obchodní značka spíše než jako samostatný mikrokontrolér. Začátky projektu Arduino sahají do Itálie, kde byl projekt zahájen v roce 2005 s úmyslem vyvíjet dostupný hardware pro interakční design. Tento projekt iniciovali Massimo Banzi a David Cuartielles, kteří úspěšně navázali na platformu „Wiring“, která vznikla o dva roky dříve, v roce 2003. Arduino představuje otevřenou hardwarovou platformu, která nabízí otevřené vývojové prostředí a snadno naučitelný programovací jazyk inspirovaný jazykem Wiring. Arduino disponuje integrovaným vývojovým prostředím založeným na programovacím prostředí Processing, přičemž hardware je otevřený pro uživatele operačních systémů Windows, Mac i Linux. [1]

Od zahájení projektu Arduino v roce 2005 až do roku 2013 se prodalo přes 700 000 oficiálních desek. Počet neoficiálních kopií však překonal tuto hodnotu, odhaduje se dokonce až dvojnásobek. Z toho vyplývá, že bylo prodáno přes několik milionů kusů desek Arduino. [2]

V současné době je téměř nemožné odhadnout přesný počet prodaných desek Arduino kvůli rozsáhlé výrobě neoficiálních kopií v Číně. Nicméně popularita Arduina stále roste, a to především díky snadnému a intuitivnímu užívání a rozsáhlé komunitě, která tento program využívá k realizaci široké škály projektů.

Arduino vzniklo díky úsilí Massima Banziho v institutu IVRAE v roce 2005. Původně šlo o desku spojenou s mikrokontrolérem, která umožňovala připojení pouze rezistorů a LED diod a nebyla vybavena žádným speciálním programovacím jazykem. Jeho vznik byl reakcí na nedostatek dostupného výukového materiálu pro studenty institutu, kteří čelili vysokým cenám běžných desek. Po úspěchu se k týmu připojili Harmando Barragán a David Mellis, kteří společně vyvinuli programovací prostředí pro desku. Dále se na vývoji podíleli i David Cuartielles a Tom Igoe, kteří vylepšili hardwarové rozhraní, přidali USB porty a mikrokontroléry, které umožňovali interpretaci a ukládání instrukcí odeslaných programem Wiring. [3]

3 Platformy Arduino

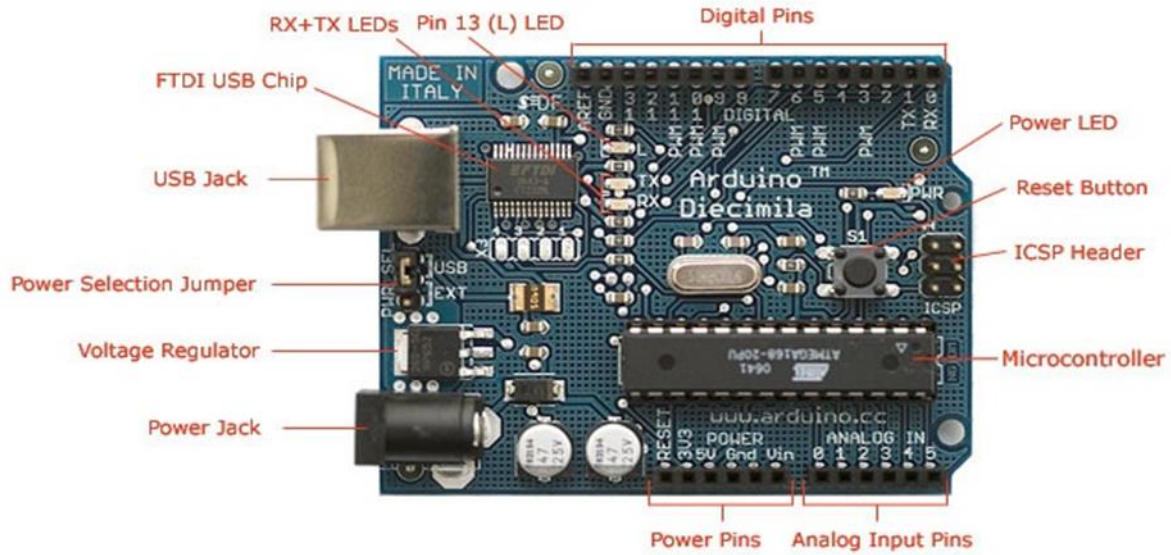
V rámci platformy Arduino existuje několik typů, které se od sebe více či méně liší. V této části práce se zaměříme pouze na jejich kategorizaci, přičemž v následujících částech bakalářské práce se podrobněji podíváme na jednotlivé typy. Platformy rozdělujeme na:

- Arduino Mega 2560
- Arduino ADK
- Arduino YUN
- Arduino UNO
- Arduino Duemilanove
- Arduino NANO
- Arduino Leonardo
- Arduino Micro
- Arduino Esplora
- Arduino Mini
- Arduino BT
- Arduino PRO
- Arduino Pro Mini
- Arduino YUN Linux
- Arduino Duemilanove ATmega168

- Arduino Diecimila
- Arduino NANO ATmega168
- Arduino LilyPad ATmega328
- Arduino Fio
- Arduino Pro ATmega328
- Arduino Gemma

3.1 Arduino UNO

Tato deska využívá mikrokontrolér ATMEL ATmega328 s frekvencí 16 MHz. Arduino UNO je nejnovější a zároveň nejrozšířenější verzí. Disponuje 32 kB programové paměti FLASH a 2 kB datové paměti RAM. Arduino poskytuje dvacet pinů mikrokontroléru, z nichž čtrnáct slouží jako digitální vstupy/výstupy a šest jako analogové vstupy. Mikrokontrolér Arduino obsahuje vestavěné periferie, jako jsou sériová linka, sběrnice I2C a čítače. Arduino je navrženo tak, aby bylo snadno rozšiřitelné pomocí dalších modulů. [1] [3]



Obrázek 1 – Arduino UNO komponenty

- **Digitální piny (digital pins)** - slouží k připojení různých periferií
- **Indikační LED dioda (Power LED)** - svítí, když je připojeno napájení

- **Resetovací tlačítko (Reset Button)** - slouží k resetu program, když chceme program spustit od začátku
- **ICSP hlavice (ICSP Header)** - externí programování hlavního čipu
- **Mikrokontroler (Microcontoller)** - hlavní čip desky
- **Analogové vstupy (Analog Input Pins)** - připojení vodičů, nebo možno využít jako digitální vstupy a výstupy
- **Napájecí piny (Power Pins)** - Napájecí výstupy Arduino
- **Napájecí konektro (Power Jack)** - Napájení pomocí USB
- **Regulátor napětí (Voltage Regulator)** - kontrola napětí, aby nedocházelo k přetížení
- **Power Selection Jumper** - výběr způsobu napájení (VIN/USB)
- **USB konektor** - napájení, tvorba sériové komunikace nebo nahrání kódů na desku
- **Sériový převodník (FTDI USB Chip)** - komunikace mezi hlavním čipem a PC
- **Indikační LED diody Rx a Tx (RX + TX LEDS)** - blikají v případě, že probíhá komunikace přes sériovou linku
- **Indikační LED dioda L (PIN 13 (L) LED)** - připojená k výstupu číslo 13, je s ním možné vyzkoušet blikání bez připojené externí LED diody

3.2 Arduino NANO

Arduino Nano je kompaktní deska, která si zachovává všechny své komponenty bez nutnosti obětovat jejich velikost. Tato vlastnost usnadňuje integraci desky do různých projektů a umožňuje její použití na prkénkách bez potřeby zbytečného zvětšování.

Parametry desky Arduino NANO:

- **Mikrokontrolér:** ATmega328
- **Vstupní napětí:** 5 V

- **Počet digitálních pinů:** 14
- **Počet analogových vstupů:** 8
- **Maximální proud na 1 pin:** 40 mA
- **Flash paměť:** 32 KB
- **SRAM paměť:** 2 KB
- **EEPROM paměť:** 1 KB

3.3 Arduino MEGA

Arduino mega 2560 je deska největším počtem vstupů a výstupů (I/O), což umožňuje tvorbu velmi rozsáhlých elektronických obvodů. Tato deska rovněž poskytuje podporu pro většinu rozšíření (shieldů) určených pro rodinu desek Arduino. Arduino mega 2560 představuje pokročilou verzi tradičního modelu Arduino mega. [3]

Parametry desky Arduino MEGA:

- Mikrokontrolér: ATmega2560
- Vstupní napětí: 5 V
- Počet digitálních pinů: 28
- Počet analogových vstupů: 11
- Maximální proud na 1 pin: 20 mA
- Maximální proud na 1 pin 3,5 V: 50 mA
- Flash paměť: 256 KB
- SRAM paměť: 256 KB
- EEPROM paměť: 4 KB

4 Vývojové prostředí

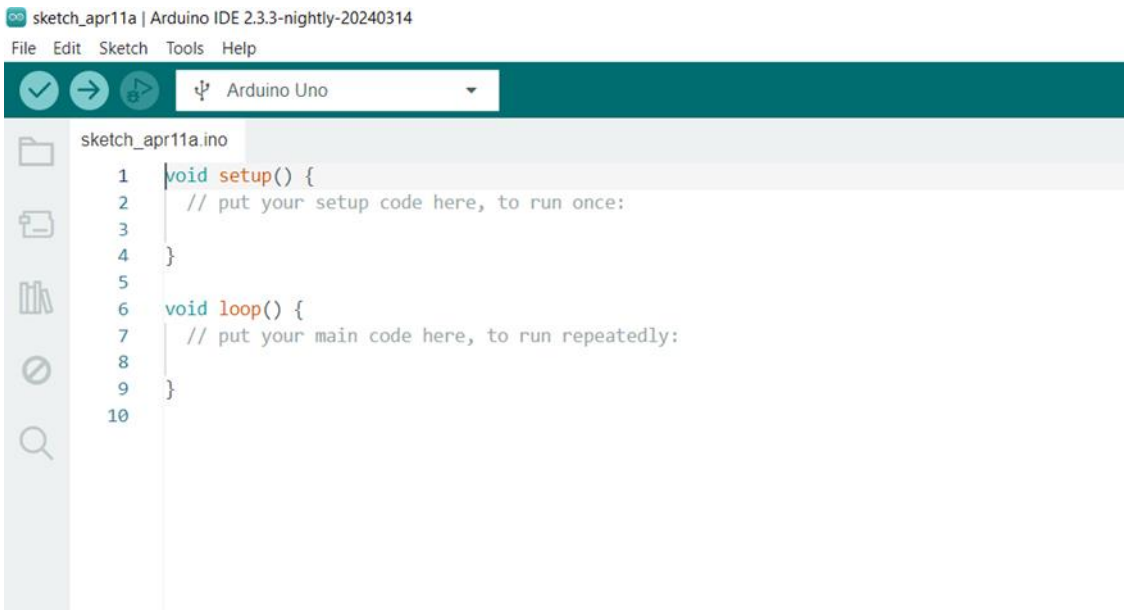
4.1 Arduino IDE

Pro dosažení našich cílů s deskou Arduino UNO je potřeba nahrát na desku kód, který se nazývá sketch (česky skica). Tento kód lze vytvořit pomocí vývojového prostředí

Arduino IDE, které poskytuje uživatelům kompletní prostředí pro vytváření kódu pro jejich projekty. Obsahuje textový editor, oblast pro zprávy a chybové hlášky a umožňuje snadné propojení s Arduinem. Programy vytvořené v Arduino IDE jsou uloženy s příponou .ino.

Arduino IDE je vývojové prostředí umožňující psát a nahrávat kód do paměti Arduina. Je dostupné pro nejběžnější operační systémy – Windows, Linux a macOS. Pro usnadnění programování Arduina je k dispozici knihovna Wiring, která rozšiřuje možnosti programovacího jazyka C++. Podrobný návod na instalaci najdete na oficiálních stránkách Arduino.cc. [6]

Klíčovou funkcí, která nám rozděluje kód na dva bloky, je void setup(), který se po spuštění programu provede pouze jednou. Následuje druhý blok, void loop(), který se neustále opakuje. Příkazy, které mají být provedeny, se nacházejí ve složených závorkách. [7]



Obrázek 2 – Arduino IDE Prostředí

4.2 Funkce

Deska Arduino umožňuje realizaci různých automatizačních úkolů. Pro svou činnost vyžaduje senzory za pomocí, kterých získává data a funguje na základě uživatelského algoritmu, který automaticky řídí připojená vstupní zařízení. Tato deska podporuje širokou škálu nejnovějších technologií rozhraní a senzorů, které jsou dostupné na trhu. [1]

Existuje široká škála využití Arduina jako jsou:

- Automatizace – domácnost, kancelář, průmysl
- Vývoj robotiky – drony, humanoidi
- Lékařské přístroje – analyzátor dechu, monitor srdeční frekvence
- Produkty související s internetem – webem ovládaná zařízení, servery
- 3D tisk
- Videohry

4.3 Wiring

Setup() x loop()

Mezi dvě základní funkce Arduina řadíme setup() a loop(). Setup() umožňuje vykonat příkaz pouze jednou, zatímco funkce loop() vykonává příkaz ve stále smyčce, kterou lze přerušit až vypnutím nebo odpojením od elektrického zdroje. [4]

Uživatelské funkce

Arduino poskytuje mnoho užitečných funkcí, které umožňují uživatelům dosáhnout požadovaného výsledku svých projektů. Mezi tyto uživatelské funkce řadíme:

- PinMode()
- DigitalWrite()
- DigitalRead()
- AnalogRead()
- AnalogWrite()
- Delay()
- Serial.begin()
- Serial.println()
- Random()

PinMode()

Funkce pinMode() je základní funkcí v programování mikrokontrolérů Arduino. Používá se k nastavení režimu pinu jako vstupu nebo výstupu. Je nástrojem pro správné nastavení pinu. U funkce pinMode() rozlišujeme tyto parametry:[4]

- Pin – Číslo pinu, který cheme nastavit
- Mode – Režim pinu, které chcete nastavit. Může být buď "INPUT" pro nastavení pinu jako vstupní, nebo "OUTPUT" pro nastavení pinu jako výstupního.

```
1   void setup() {
2
3     pinMode(13, OUTPUT); //Nastav PIN 13 jako výstup
4   }
5
6   void loop() {
7
8     digitalWrite(13, HIGH); //HIGH = zapni LEDku
9     delay(1000); //Počkej 1 sekundu
10  }
```

Obrázek 3 – Arduino IDE - Kód – PinMode()

DigitalWrite()

Funkce **digitalWrite()** slouží v programování mikrokontrolérů Arduino k ovládání digitálních výstupních pinů. S její pomocí lze nastavit pin na **HIGH** (logická hodnota 1) nebo **LOW** (logická hodnota 0), což umožňuje ovládat různá zařízení, jako jsou LED diody, relé, nebo motory. Je to nezbytný nástroj pro kontrolu periferií v elektronických projektech.

U funkce **digitalWrite()** rozlišujeme tyto parametry:

- **Pin** – číslo pinu, který chceme ovládat.
- **Value** – hodnota, kterou chceme na pin nastavit. Může být buď **HIGH** nebo **LOW**.

Funkce **digitalWrite()** je důležitým nástrojem v programování Arduino a často se používá pro ovládání různých periferií v elektronických projektech. [4]

DigitalRead()

Funkce **digitalRead()** je důležitou součástí programování mikrokontrolérů Arduino, která slouží k čtení stavu digitálního vstupního pinu. Umožňuje získávat informace o stavu připojených zařízení, jako jsou tlačítka nebo senzory. Tato funkce je často využívána k čtení stavu tlačítek nebo digitálních senzorů připojených k digitálním vstupním pinům. Funkce vrací **HIGH**, pokud je na pinu detekována napěťová úroveň vyšší než určitá hodnota; jinak vrací **LOW**. Před použitím **digitalRead()** je důležité nastavit pin jako vstup pomocí funkce **pinMode()**.

Parametry funkce **digitalRead()**:

- **Pin** – číslo pinu, který chceme číst.

Návratové hodnoty funkce **digitalRead()**:

- Vrátí hodnotu **HIGH** (logická hodnota 1), pokud je na pinu detekována napěťová úroveň odpovídající logické jedničce (např. 5 V).
- Vrátí hodnotu **LOW** (logická hodnota 0), pokud je na pinu detekována napěťová úroveň odpovídající logické nule (např. 0 V). [4]

```
1  const int buttonPin = 2; //Pin ke kterému je připojeno tlačítko
2  const int ledPin = 13;//Pin ke kterému je připojena LEDKA
3
4  void setup() {
5    pinMode(buttonPin, INPUT);//Nastavit tlačítko jako vstup
6    pinMode(ledPin, OUTPUT);//Nastavit LED jako výstup
7  }
8
9  void loop() {
10   if (digitalRead(buttonPin) == HIGH) { //Pokud držíme tlačítko, tak:
11     digitalWrite(ledPin, HIGH); //LED svítí
12   } else {
13     digitalWrite(ledPin, LOW); //Pokud není nesvítí
14 }
15
```

Obrázek 4 Arduino IDE – Kód – DigitalRead()

AnalogWrite()

Funkce analogWrite() v Arduino je funkce, která nám generuje modulovaný signál PWM na výstupních pinech. Tento signál nám umožňuje ovládat různé periferie, jako jsou LED diody, motory a serva, změnou jejich průměrného napětí.

U AnalogWrite() rozlišujeme tyto parametry:

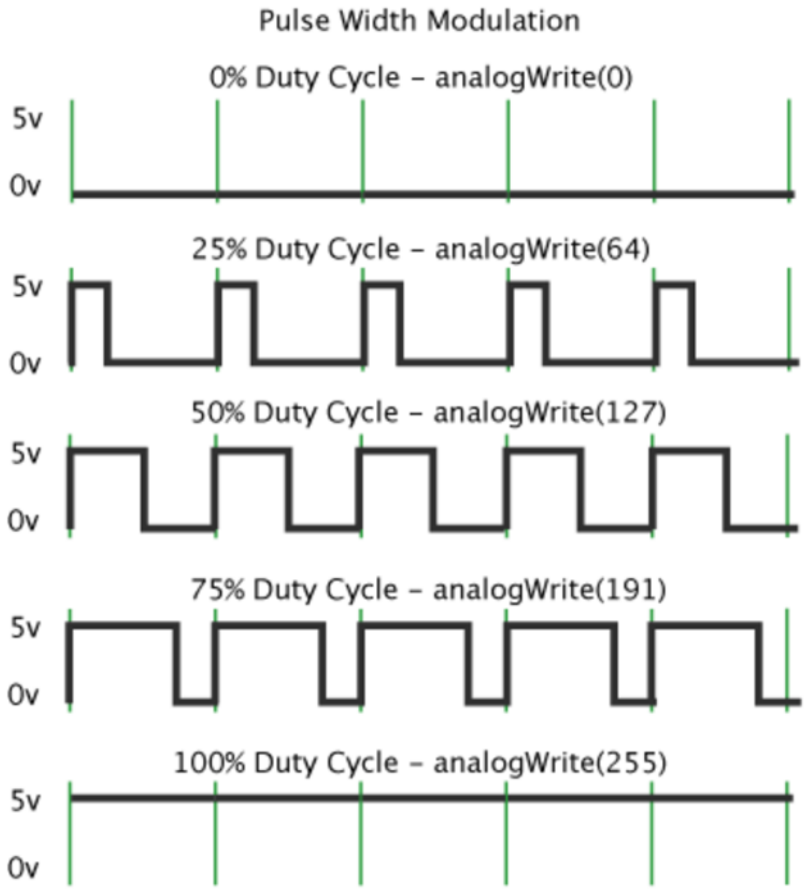
- Pin – Číslo digitálního pinu, který chceme ovládat. Na desce Arduino UNO jsou digitální hodnoty pinu označeny 0 až 13.
- Value – Hodnota, kterou chceme nastavit pro modulovaný signál. Může být v rozmezí 0 (žádný signál) až 255 (plný signál).

```
1  const int ledPin = 9; //Pin ke kterému je připojena LEDKA
2  void setup() {
3    pinMode(ledPin, OUTPUT); //Nastavit LEDKU jako výstup
4  }
5  void loop() {
6    analogWrite(ledPin, 128); // Nastavení jasu
7  }
```

Obrázek 5 Arduino IDE – Kód – AnalogWrite()

PWM (Pulsně šířková modulace)

Signál PWM moduluje úroveň výstupního signálu mezi logickou nulou a jedničkou s různými poměry zapnutí a vypnutí. To umožňuje sledovat změny průměrného výstupního napětí, což vede k různým efektům, jako je regulace jasu LED diod nebo rychlosti motorů. [7][13]



Obrázek 6 - PWM

AnalogRead()

Funkce **analogRead()** je základním nástrojem v programování mikrokontrolérů Arduino a slouží k čtení analogových hodnot z vstupních analogových pinů. Tato funkce umožňuje uživatelům získávat hodnoty z analogových senzorů, jako jsou teplotní senzory, potenciometry nebo fotoodpory.

Funkce **analogRead()** se používá pro čtení analogových hodnot ze senzorů, které poskytují spojitou škálu hodnot, na rozdíl od jednoduchého zapnutí nebo vypnutí.

Pro čtení analogových hodnot je nutné využít analogové piny na desce Arduino, které jsou speciálně určené pro tento účel. Před použitím funkce **analogRead()** je nezbytné nastavit pin jako vstupní pomocí funkce **pinMode()**.

Parametry:

- **Pin**: Číslo analogového pinu, který chceme číst. Na desce Arduino UNO jsou analogové piny označeny čísly A0 až A5. [4]

```arduino
const int sensorPin = A0; //Pin ke kterému je připojen sensor
const int ledPin = 9;     //Pin ke kterému je přiřazena LEDKA
void setup() {
  pinMode(ledPin, OUTPUT); //Nastavit Ledku jako výstup
}
void loop() {
  int sensorValue = analogRead(sensorPin); //Snímání senzoru

//Když hodnota senzoru je větší než 256, zapneme LEDKU, jinak vypneme
  if (sensorValue > 512) {
    digitalWrite(ledPin, HIGH);//LED ON
  } else {
    digitalWrite(ledPin, LOW);//LED OFF
  }
}
```

Obrázek 7 Arduino IDE – Kód –AnalogRead()

delay()

Funkce **delay()** je základní funkcí, která vytváří pauzu v kódu. Umožňuje uživatelům zpomalit běh programu a čekat na určitý časový interval před pokračováním v dalších instrukcích. Tato funkce zastaví vykonávání programu na dobu určenou parametrem **ms**, který musí být kladné celé číslo. Během této pauzy mikrokontrolér nevykonává žádnou funkci a čeká na uplynutí tohoto časového intervalu.

Parametry

- **ms** - Délka pauzy v milisekundách. Jedna sekunda (1000 ms) odpovídá hodnotě 1000.

```arduino
void setup() {

    pinMode(13, OUTPUT); //Nastav PIN 13 jako výstup
  }

  void loop() {

    digitalWrite(13, HIGH); //HIGH = zapni LEDku
    delay(1000); //Počkej 1 sekundu
  }
```

Obrázek 8 – Arduino IDE – Kód – delay()

Serial.begin()

Funkce Serial.begin() zajišťuje sériovou komunikaci mezi Arduinem a dalším zařízením prostřednictvím sériového portu. Parametr "speed" určuje rychlost přenosu dat (baudovou rychlost). Tato funkce představuje klíčový nástroj pro vytváření spojení pro výstup a vstup dat mezi Arduinem a počítačem nebo jinými zařízeními. Při inicializaci je nutné, aby Arduino bylo připojeno k počítači nebo jinému zařízení pomocí USB kabelu nebo sériového portu.

Parametry:

- **speed** - Baudová rychlost sériové komunikace v baudových bodech. Baudová rychlost určuje, jak rychle jsou data přenášena mezi Arduinem a jiným zařízením.

```
1  void setup() {
2    Serial.begin(9600);//Nastavení sériové komunikace
3  }
4
5  void loop() {
6    Serial.println("Ahoj čtenáři"); //Poslání zprávy "Ahoj čtenáři" po sériové komunikaci
7    delay(1000); //1 sekund počkat
8  }
```

Obrázek 9 – Arduine IDE – Kód – Seriál.begin()

Serial.println()

Funkce Serial.println() funguje jako vysílač pro Arduino. Je používána k vypisování informací a dat na sériový port, který lze přirovnat k virtuálnímu drátu mezi Arduinem a počítačem. Pokud chcete sledovat, co Arduino provádí, můžete pomocí Serial.println() zadat, aby vám poslalo zprávy. Toto vám umožní lépe porozumět dění ve vašem programu a chování vašeho Arduina.

Parametry:

- **data** – Data, která chcete vypsat na sériový port. Může to být textový řetězec, celé číslo, desetinné číslo nebo jiný datový typ. [4]

```
1  void setup() {
2  }
3  void loop() {
4    Serial.println("Hello, Arduino!");
5    delay(1000);
6  }
```

Obrázek 10 – Arduine IDE - Seriál.Println()

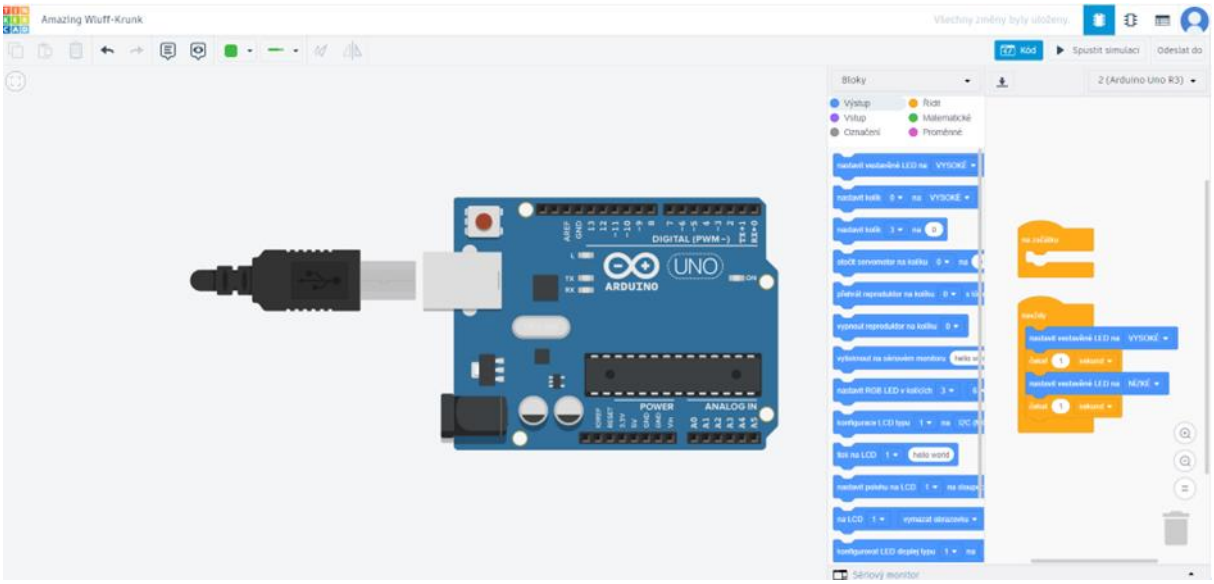
Ovládání

Pro ovládání desky Arduino musí uživatel vytvořit instrukce ve formě lidsky čitelného kódu na hostitelském počítači, které následně převede do strojového jazyka a nahraje na desku. Tento proces se provádí pomocí jednoduchého a uživatelsky přívětivého vývojového prostředí IDE (Integrated Development Environment), které obsahuje textový editor, kompilátor, uploader, monitor sériového portu a debugger. Programovací jazyk Arduino je založen na zjednodušené verzi jazyků C/C++. Po nahrání programu do paměti čipu uvnitř desky bude program spuštěn při každém zapnutí nebo resetování desky. Uživatel má možnost program snadno měnit nebo aktualizovat podle svých potřeb. [1][2]

4.4 TinkerCAD

Pokud nemáme fyzicky Arduino UNO desku, tento program je řešením. Tinkercad je webovým nástrojem pro 3D modelování, simulaci obvodů a blokové kódování, který je zdarma prostřednictvím webového prohlížeče. Jeho jednoduché uživatelské rozhraní a bezplatná dostupnost přes PC nebo tablet přitahují spousty nadšenců. Společnost Autodesk získala Tinkercad v roce 2013 a od té doby přidala spousty nových funkcí včetně kódování.

Tinkercad nabízí uživatelům jednoduchý nástroj pro blokové kódování, který umožňuje tvorbu interaktivních projektů bez nutnosti psát složité kódy. Tento nástroj je ideální pro začátečníky v programování, kteří se chtějí naučit základy bez potřeby hlubších znalostí programování. Můžeme snadno přidávat logické bloky, jako jsou podmínky a smyčky a pomocí intuitivního rozhraní drag-and-drop bloky přesouvat. Dále podporuje funkce, jako jsou proměnné, pole a vstupně-výstupní bloky. [8][9]

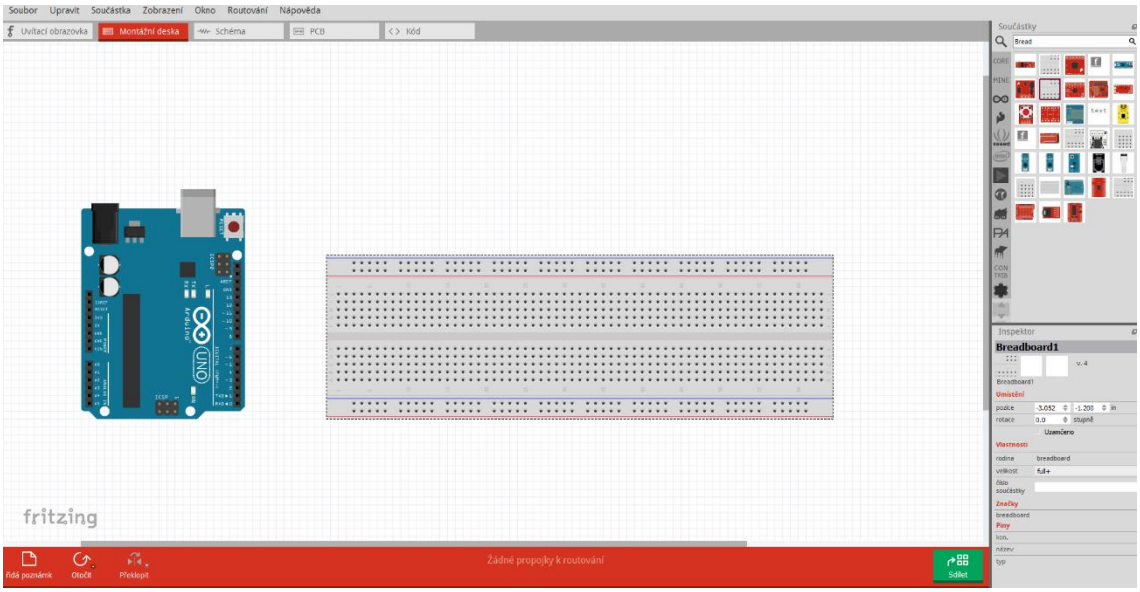


Obrázek 11 – Tinkercad - Prostředí

Jak můžeme vidět na obrázku 11, v pravé části obrázku na začátku je void setup() a následně void loop().

4.5 Fritzing

Fritzing je open-source EDA software vyvinutý Univerzitou v Potsdamu, který je navržený pro neinženýry, jako jsou designéři, vynálezci a vzdělavatelé. Tento nástroj umožňuje snadné prototypování a tvorbu tisknutých plošných spojů (PCB) bez potřeby hlubokých technických znalostí. Fritzing je ideální pro tvorbu elektronických schémat a vývojových desek a nabízí intuitivní rozhraní s možnostmi, jako je práce s breadboardem, schématy a programováním. Platforma podporuje široké spektrum komponent, které mohou uživatelé snadno integrovat do svých projektů.[5]



Obrázek 12 Fritzing - Prostředí

4.6 Rozdíly mezi Arduino IDE a TinkerCAD

TinkerCad vyniká v uživatelské přívětivosti a nabízí přehledné rozhraní, zejména pro začátečníky, zatímco Arduino IDE vyžaduje více odborných znalostí. Z hlediska ceny, Arduino IDE je zdarma, zatímco TinkerCAD vyžaduje předplatné pro plný přístup k funkcím, přestože pro začátečníky poskytuje mnoho bezplatných nástrojů. TinkerCAD nabízí širší výběr komponent a podporu pro více desek než Arduino IDE. Zatímco Arduino IDE používá programovací jazyk C/C++, TinkerCAD spoléhá na vizuální programovací jazyk založený na blocích. Ohledně simulace, TinkerCAD se spustí virtuálně na jejich platformě oproti Arduinu, který potřebujeme mít fyzicky. [10]

5 Editace videí v Adobe Premiere PRO

5.1 Adobe Premiere Pro

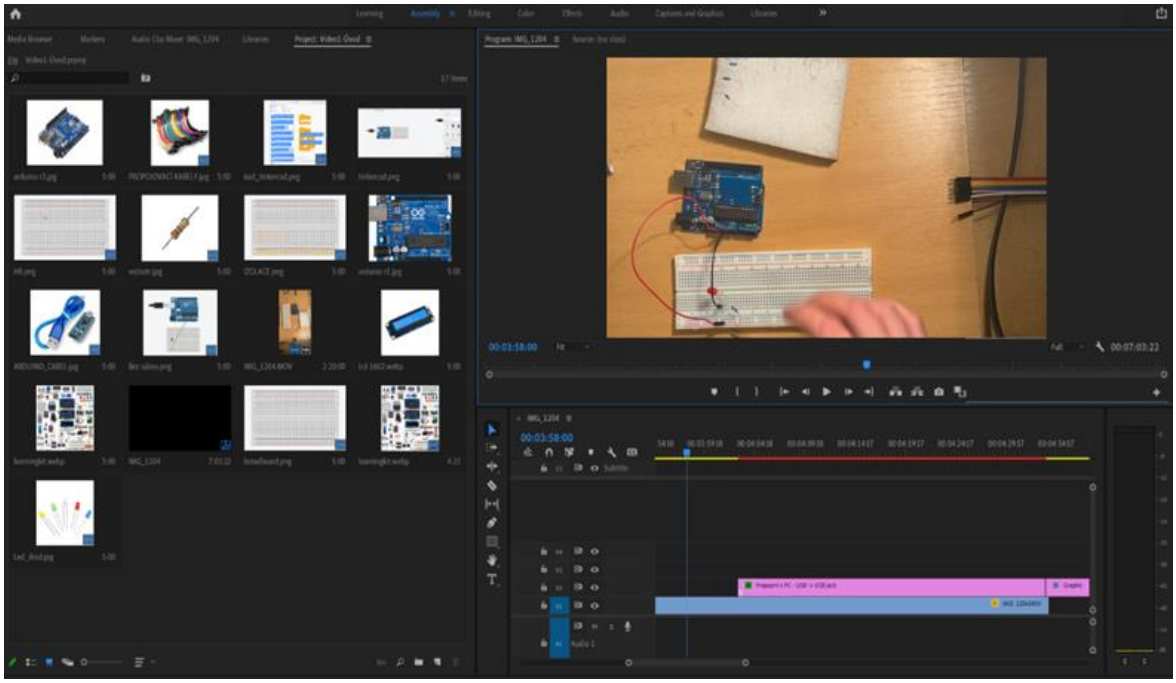
Pro kvalitní střih používáme program Adobe Premiere Pro, jak už název napovídá, software byl vyvinut společností Adobe. Tato společnost nabízí řadu dalších programů pro zpracování videí, grafickou práci a úpravy zvuku. Adobe zvolilo odlišný přístup k licencování Premiere Pro; nabízí jej jako součást balíčku Adobe Creative Cloud, který se platí měsíčními poplatky a může obsahovat i další programy. Premiere Pro je široce používán v různých odvětvích, včetně tvorby videí pro YouTube, televizních reklam a filmové tvorby.

5.2 Porovnání s ostatními programy

Hlavním konkurentem Premiere PRO je Final Cut Pro, který je dostupný pouze pro platformu macOS. Ačkoliv oba programy nabízejí podobné funkce, Premiere Pro je dostupné na více platformách, což představuje výhodu pro uživatele pracující s různými operačními systémy. Premiere Pro je vybaveno širokou škálou funkcí, včetně pokročilé práce se zvukem a podpory HDR pro kvalitní barevnou korekci. Tento program umožňuje editaci 360° videí a pracuje s rozlišením až do 8k. Uživatelé mohou snadno používat soubory z dalších programů Adobe, jako je Photoshop, což zvyšuje efektivitu práce. [12][13]

5.3 Výhody Adobe Premiere Pro

Software sice vyžaduje placené měsíční nebo roční předplatné a může být náročný na hardware, učení se s ním může být pro začátečníky obtížné kvůli jeho rozsáhlým funkcím. Dále vyžaduje připojení k internetu pro aktivaci a aktualizaci, což může představovat nevýhodu pro uživatele s nestabilním připojením k internetu. Nicméně díky svým výhodám zůstává Adobe Premiere Pro oblíbeným nástrojem pro profesionální úpravu videí. [11][12]



Obrázek 13 Adobe Premiere PRO - prostředí

6 Praktická část

6.1 Cíl praktické části

Tato práce si klade za cíl vytvořit interaktivní webovou platformu, která bude obsahovat dvacet lekcí zaměřených na výuku platformy Arduino UNO. Každá z těchto lekcí byla navržena jako pracovní list, který postupně provádí studenty od základních po pokročilejší projekty. Hlavním prvkem každé lekce jsou videa, která demonstrují studentům fyzické zapojení a programování v prostředí Arduino IDE. Pro lepší přehled je také představeno schéma v TinkerCadu.

6.2 Přehled o tvorbě videotutoriálů na platformu Arduino UNO

Tato práce je založena na využití platformy Arduino UNO, která je nejpopulárnějším modelem v oblasti vývoje projektů s Arduino. Pro tvorbu interaktivní webové stránky jsem zvolil HTML a CSS, což jsou nejznámější jazyky pro tvorbu internetových stránek. Tato volba umožňuje snadný přístup k materiálům.

Pro tvorbu videí jsem použil Adobe Premiere Pro, profesionální software pro úpravu videí. Při výběru softwaru jsem kladl důraz na jeho dostupnost a finanční nenáročnost, s výjimkou Adobe Premiere Pro.

6.3 Proces tvorby vlastního webu

Prvním krokem v procesu tvorby webové platformy byla analýza potřeb a požadavků. Bylo nezbytné určit, co všechno platforma musí umět, aby efektivně podporovala učení. Následně jsem přešel k návrhu, abych vytvořil plán, jak platformu navrhnout tak, aby byla snadno použitelná a zároveň vypadala moderně. Poté jsem začal s implementací pomocí HTML a CSS, abych vytvořil stránky, které se přizpůsobí různým zařízením a byly tak přístupné každému.

6.4 Podrobnosti každé lekce a účel

Na této vzdělávací platformě je každá lekce pečlivě navržena s jasně stanovenými cíli a obsahem. Podrobnosti o obsahu každé lekce poskytují detailní přehled probíraných témat a definují očekávané výstupy pro studenty. Hlavním účelem každé lekce je poskytnout užitečné informace a praktické dovednosti prostřednictvím webového rozhraní, což umožňuje lepší porozumění danému tématu.

6.5 Použité technologie

Při vytváření webové platformy jsem využíval široké spektrum technologií a nástrojů. Pro strukturu obsahu bylo použito HTML a CSS, které sloužilo k úpravě vzhledu. Pro úpravu kódu byla využívána integrovaná vývojová prostředí, jako je Visual Studio Code.

6.6 Struktura a obsah lekcí

6.6.1 Úvodní strana

Na úvodní straně naší platformy jsou zobrazeny všechny lekce dostupné uživatelům. Tato stránka slouží jako navigační centrum, kde si uživatelé mohou jednoduše vybrat, které téma je zajímá, a přejít přímo k příslušné lekci. Každá lekce je prezentována jako samostatný blok, obsahující název lekce a krátký popis tématu.

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Arduino UNO</title>
<link rel="stylesheet" href="styles.css">
</head>
<body>
<header>
<h1>Arduino UNO</h1>
</header>
<main>
<section class="project">
<h2><a href="uvodarduino.html">Úvod do Arduina    </a></h2>
<p>V této lekci si ukážeme prezentaci ohledně základního fungování
Arduina a představení hlavních komponent pro tvorbu elektronických
projektů.
</p>
</section> <section class="project">
<h2><a href="tinkercad.html">TinkerCAD   </a></h2>
<p>V této lekci si představíme virtuální prostředí TinkerCADu, které
slouží jako nástroj pro simulaci a tvorbu elektronických obvodů bez
fyzických komponent.</p>
</section>
<section class="project">
<h2><a href="arduinoide.html">ArduinoIDE    </a></h2>
<p>V této lekci se zaměříme na představení Arduino IDE, integrovaného
vývojového prostředí pro programování Arduina,
a naučíme se základní kroky pro psaní a nahrávání kódu do Arduina.</p>

</main>
<footer>
<p>&copy; 2024 Moje Projekty</p>
</footer>
</body>
</html>
```

Příklad 1 HTML – Úvodní strany

V našem HTML kódu jsou meta znaky pro nás důležité, protože definují typ dokumentu a verzi HTML, kterou používáme. Taktéž určují, jak má být obsah zobrazen na různých zařízeních s různou šířkou obrazovky. Nadto pomocí elementu <title> nastavujeme název stránky, který se zobrazuje v záhlaví okna prohlížeče. Odkazujeme také na externí CSS soubor pomocí elementu <link rel="stylesheet" href="styles.css">, kde jsou definovány styly pro vizuální formátování stránky. Struktura stránky je dále definována pomocí elementů <header>, <main> a <footer>, které označují záhlaví, hlavní

obsah a zápatí stránky. Odkazy na jiné stránky jsou vytvořeny pomocí elementu <a href="url">, který umožňuje navigaci mezi různými částmi webu.

6.6.2 Detailní popis postupu tvorby webové platformy

Struktura jednotlivých lekcí:

Úvod

Teoretická část

Praktická část

Závěr

6.6.3 Úvod

Každá lekce začíná poutavým úvodem, který studenty seznámí s tématem. Úvod může obsahovat zajímavé anekdoty, citace nebo otázky, které vyvolají zájem a připraví studenty na nadcházející obsah.

```
<h2><p>Úvod</p></h2>
<p>V dnešní lekci se zaměříme na základy zapojení první LED diody s
Arduino UNO. Budeme postupovat krok za krokem a naučíme se správně
připojit LED diodu k Arduinu, abychom mohli začít s prvním jednoduchým
projektem. Tato lekce poskytne solidní základy pro budoucí práci s LED
diodami a Arduino platformou. A teď pojďme do toho!  </p>
```

Příklad 2 Ukázka HTML-Úvod

Ve výše uvedeném příkladu jsme využili HTML značky <H2> a <p>. Značku <h2> jsme použili pro vytvoření nadpisu druhé úrovně, což je titulek, který nám pomáhá rozdělovat text na jednotlivé části. Naopak <p> je určena pro vytvoření odstavce textu, podobně jako když píšeme běžně do knihy. Tyto značky nám pomáhají organizovat text.

6.6.4 Teoretická část

Tato část poskytuje ucelený přehled tématu lekce. Začínáme popisem nezbytných komponentů, které budou v lekci použity. Uživatelé obdrží detailní informace o každém komponentu a jeho funkci.

Teoretická část se následně dělí na další dvě části:

- **Komponenty** – V této části vypíšeme všechny potřebné komponenty, které budou k projektu potřeba.
- **Popis** – Podrobně popisujeme jednotlivé komponenty použité v projektu.

```
•
• <section id="teoreticka_cast">
• <h2>Komponenty</h2>
• <p>- 1 * Arduino UNO</p>
• <p>- 1 * USB Cable</p>
• <p>- 1 * Resistor</p>
• <p>- 1 * LED</p>
• <p>- 1 * Breadboard</p>
• <p>- 2 * Proudové kabely</p>
• </section>
•
```

Příklad 3 HTML teoretická část – komponenty

Jak můžeme vidět, značky <h2> a <p> se opakují, ale v této části byl přidán příkaz <section>, který slouží k zpřehlednění částí kódu. Tímto je definováno, že je součástí teoretické části.

```
<section id="teoreticka_cast">
  <h3>Co je to LED?</h3>
  <p>LED je zkratka pro light emitting diode (světlo emitující
dioda).</p>
  <p>Obvykle je vyrobena z polovodičových materiálů gallium arsenidu a
gallium fosfidu.</p>
  <p>LED má dvě elektrody: kladnou a zápornou.</p>
  <p>Své světlo vydává pouze tehdy, když jí prochází přední proud.</p>
  <p>Obvyklý řídící proud pro LED je v rozmezí 5-20 mA.</p>
  <img src="Led1.png" alt="Popis fotografie" width="400" height="500">>
  <h3>Co je to rezistor?</h3>
  </section>
```

Příklad 4 HTML teoretická část – detail komponent

Výskyt značek HTML jako <h2>, <p> a <section> se v našem kódu opakuje. Tentokrát jsme však přidali několik dalších prvků, které zlepšují prezentaci našich informací. Prvky jako <img src>, <alt>, width a height nám umožňují efektivněji pracovat s obrázky. Atribut <img src> určuje zdrojový soubor obrázku, <alt> poskytuje alternativní text pro případ, že obrázek není načten nebo pro uživatele s omezeným

zrakem. Atributy width a height určují rozměry obrázku, což napomáhá responzivnímu designu a přizpůsobení obsahu různým zařízením.

6.6.5 Praktická část

V praktické části přecházíme k aplikaci teorie v praxi. Setkáme se zde s popisem videa a konkrétního videa, které demonstruje postupy práce s jednotlivými komponenty nebo realizaci specifického projektu. Uživatelé mají možnost sledovat, jak se teoretické znalosti transformují do praktických úkonů.

```
<section id="prakticka_cast">
  <h2>Praktická část</h2>
  <p>V dnešní lekci se zaměříme na základy zapojení první LED diody s
Arduino UNO. Budeme postupovat krok za krokem a naučíme se správně
připojit LED diodu k Arduinu, abychom mohli začít s prvním jednoduchým
projektem. Tato lekce poskytne solidní základy pro budoucí práci s LED
diodami a Arduino platformou. A teď pojďme do toho!</p>
  </section>
  <section id="video">
  <h2>Video</h2>
  <div class="video-container">
  <video controls width="640" height="360">
  <source src="video1.mp4" type="video/mp4">
  </video>
  </div>
```

Příklad 5 Html teoretická část – video

Ve svém kódu jsme znovu využili HTML značky jako <h2>, <p>, <section> a nově jsme přidali další prvky pro lepší organizaci obsahu a integraci videí. Značka <div class> nám umožňuje vytvářet skupiny prvků, které poté můžeme stylovat pomocí CSS. Prvek <video controls> slouží k zobrazení videa na stránce a obsahuje ovládací prvky pro přehrávání, jako jsou tlačítka play, pause a volume. Uvnitř <video> používáme značku <source src type>, abychom specifikovali zdrojový soubor videa a jeho formát.

6.6.6 Závěr

Tato část slouží k shrnutí klíčových poznatků z lekce. Budeme rekapitulovat hlavní body, které jsme probrali, a upozorníme na možné chyby nebo aspekty, na které by si uživatelé měli dávat pozor při dalším praktickém využití.

```
<h2>Závěr</h2>
<p>V dnešní lekci jsme si zapojili naší první diodu.</p>
</section>
<section id="zaver" style="background-color: red; color: white;">
<p><h3>Na co si dát pozor</h3></p>
<p>Správně otočit Led diodu.</p>
<p>5V Napájet + na breadboardu - PROUD</p>
<p>GND(Ground) uzemnit katodu Ledky.</p>
</section>
```

Příklad 6 Teoretická část – závěr

6.7 Podrobný přehled lekcí

V této části si představujeme podrobný přehled lekcí, který se rozvíjí od základů až po pokročilé techniky spojené s platformou Arduino. Obsahuje široké spektrum témat, od úvodního seznámení s Arduino a TinkerCAD po pokročilé projekty, které využívají různé senzory a moduly. Tato příprava umožňuje hluboký ponor do světa elektroniky a programování v rámci bakalářské práce.

Přehled lekcí:

1. Představení platformy Arduino
2. TinkerCAD
3. Arduino IDE
4. Struktura programu
5. Přerušení
6. Analogový vstup
7. Funkce
8. Podprogramy
9. Breadboard
10. Připojení desky k počítači
11. LED dioda
12. Tlačítko
13. Snímání světla

14. Měření vlhkosti půdy
15. Modul zalévání
16. Modul měření teploty a vlhkosti půdy
17. Modul vlhkosti vzduchu
18. Modul osvětlení
19. Krokové motory
20. LCD displej

Každá lekce poskytuje strukturovaný přehled konkrétních aspektů platformy Arduino. V úvodu se studenti seznamují s tématem, které je motivuje k dalšímu studiu. Teoretická část podrobně vysvětluje komponenty a programování. V praktické části mají studenti možnost aplikovat své znalosti v reálném prostředí. Závěrečná část reflektuje dosažený pokrok.

6.7.1 Představení Platformy ARDUINO

V úvodní lekci se seznámíme s platformou Arduino a jejími základními komponentami. Arduino je open-source platforma s mikrokontrolérem, která umožňuje vytvářet interaktivní elektronická zařízení. Tato lekce se zaměří na Learning Kit pro začátečníky, který zahrnuje klíčové komponenty, jako jsou Arduino deska, breadboard, LED diody, rezistory a kabely.

Ve videu se podíváme na praktické využití těchto komponent. Dozvíme se více o jednotlivých prvcích, ale především se podrobně seznámíme s Learning Kitem a jeho komponentami.

6.7.2 TincerCAD

V druhé lekci se seznámíme s platformou Tinkercad, který poskytuje prostředí pro programování v jednoduchém uživatelském rozhraní. Tato platforma nabízí širokou škálu funkcí pro programování pomocí bloků, psaní vlastního kódu a především umožňuje simulovat práci s fyzickou deskou ve virtuálním prostředí, aniž bychom ji museli vlastnit.

Ve videu jsme se zaměřili na nejdůležitější funkce TinkerCadu a předvedli praktické ukázky jeho využití. Pro začátečníky jsme zdůraznili základní kroky a nástroje, které jsou nezbytné pro intuitivní používání této platformy.

6.7.3 Arduino IDE

V třetí lekci se seznámíme s platformou Arduino IDE, která nám poskytuje prostředí pro programování mikrokontrolérů Arduina. Arduino IDE je navrženo tak, aby bylo jednoduché a přístupné začátečníkům, ale zároveň poskytovalo pokročilé funkce pro zkušenější uživatele.

Ve videu jsme se zaměřili na seznámení s funkcemi Arduino IDE, které umožňuje psát a nahrávat programy do desek Arduino. Důkladně jsme prozkoumali textový editor.

6.7.4 Struktura programu

V čtvrté lekci se seznámíme s programovou strukturou Arduino. Tato struktura zahrnuje hlavičkový soubor, funkce setup() a loop(), proměnné a konfiguraci pinů. Proměnné a konstanty jsou používány pro ukládání dat a hodnot. Video demonstruje první použití funkcí setup() a loop().

```
void setup() {
  // Pouze jednou při spuštění

  }

  void loop() {
  // Provádí se opakovaně

  }
```

Příklad 7 Arduino IDE – void(),setup()

6.7.5 Přerušení

V paté lekci si představíme přerušení v Arduinu. Přerušení umožňuje okamžitě reagovat na události, jako je stisk tlačítka nebo příjem dat ze senzoru. Existuje několik typů přerušení, které jsou podrobně popsány v této lekci. Při implementaci přerušení je klíčové dbát na délku a složitost přerušovací rutiny, aby byla zajištěna rychlá reakce. Ve videu je demonstrováno první přerušení.

```
const int ledPin = 2;

void loop() {
  // Nic nedělej - čekáme na přerušení
}

void buttonISR() {
  // Obsluha přerušení - Invertujeme stav LED
  digitalWrite(ledPin, !digitalRead(ledPin));
}
```

Příklad 8 Arduino IDE – Přerušení

6.7.6 Analogový vstup

V úvodní části této lekce jsme se zaměřili na analogové vstupy a jejich důležitost v elektronických projektech. Diskutovali jsme o tom, jak analogové vstupy umožňují Arduino číst analogová data, jako jsou napětí nebo intenzita světla, což rozšiřuje možnosti senzorického vnímání a monitorování prostředí.

V teoretické části jsme prozkoumali princip fungování analogových vstupů a převodníku ADC (Analog to Digital Converter) v Arduino. Vysvětlili jsme, jak ADC převádí analogová data na digitální hodnoty a jaký vliv má rozlišení ADC na přesnost měření.

V praktické části jsme provedli experimenty s různými analogovými senzory, jako jsou senzory světla, teploty a vlhkosti, a připojili je k Arduino. Demonstrujeme, jak číst hodnoty z analogových senzorů pomocí analogových vstupů Arduino a jak tyto hodnoty využít k řízení různých akcí nebo reakcí v našem programu.

V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití analogových vstupů v různých projektech. Potvrdili jsme, že jsme získali dostatečné znalosti pro úspěšné použití analogových senzorů a vstupů v našich budoucích projektech s Arduino.

6.7.7 Funkce

Tato lekce se zaměřuje na koncept funkcí v programování. Vysvětlíme si základní funkce:

pinMode ()

```
void setup() {
  pinMode(13, OUTPUT); //Nastavit pin 13 Jako výstup
}
void loop() {
  digitalWrite(13, HIGH); //zapnout LED
  delay(500); //Počkat půl vteřiny
  digitalWrite(13, LOW); // Vypnout LED
  delay(500); //Počkat půl vteřiny
}
```

Příklad 9 Arduino IDE

V tomto příkladu funkce pinMode() nastavuje režim pinu 13 na Arduinu. Specificky, pin 13 je konfigurován jako výstupní pin s použitím pinMode(13, OUTPUT) v rámci funkce setup(). Tato konfigurace instruuje Arduino, aby pin 13 využívalo jako výstup pro ovládání LED. Díky tomu můžeme následně využít funkci digitalWrite() pro zapínání a vypínání LED připojené k tomuto pinu.

digitalWrite(), delay()

```
const int LED_PIN = 13;
void setup() {
  pinMode(LED_PIN, OUTPUT); //Nastavení pinu LED jako výstup
}
void loop() {
  digitalWrite(13, HIGH); //zapnout LED
  delay(1000); //Počkat vteřinu
  digitalWrite(13, LOW); // Vypnout LED
  delay(1000); //Počkat vteřinu
}
```

Příklad 10 Arduine IDE – digitalWrite()

V tomto kódu je využita funkce digitalWrite(), která umožňuje ovládat digitální piny na platformě Arduino. Tato funkce přijímá dva parametry: číslo pinu a stav, do kterého má být pin nastaven. Když je funkce digitalWrite(13, HIGH) volána, pin 13 je nastaven na logickou hodnotu HIGH, což zapne LED diodu připojenou k tomuto pinu. Opakem je volání digitalWrite(13, LOW), které nastaví pin 13 na logickou hodnotu LOW a vypne LED diodu. Pomocí funkce delay(1000) mezi těmito dvěma voláními digitalWrite() se zajišťuje, že LED dioda zůstane buď zapnutá, nebo vypnutá po dobu jedné sekundy, což vede k blikání LED diody.

digitalRead()

```
const int BUTTON_PIN = 2;
const int LED_PIN = 13;
void setup() {
  pinMode(BUTTON_PIN, INPUT); // Nastavení pinu tlačítka jako vstup
  pinMode(LED_PIN, OUTPUT); //Nastavení pinu LED jako výstup
}
void loop() {
  int button = digitalRead(BUTTON_PIN); // Čtení stavu tlačítka

  if (button == LOW){
    digitalWrite(LED_PIN, HIGH); // Zapnutí LED
  } else {
    digitalWrite(LED_PIN, LOW); // Vypnutí LED
  }
}
```

Příklad 11 Arduino IDE – digitalRead()

V našem kódu je funkce digitalRead() použita k čtení stavu tlačítka, které je připojeno k pinu 2 na Arduinu. Když je tlačítko stisknuto a tedy na pinu 2 je logická hodnota LOW, funkce digitalRead() vrátí hodnotu LOW, která je uložena do proměnné button. Poté, když je proměnná button rovna LOW, Arduino zapne LED připojenou k pinu 13 pomocí funkce digitalWrite(LED_PIN, HIGH);. Naopak, pokud tlačítko není stisknuto a na pinu 2 je přítomen logický HIGH, funkce digitalRead() vrátí hodnotu HIGH, která se opět uloží do proměnné button. V tomto případě, když je proměnná button rovna HIGH, Arduino vypne LED na pinu 13 pomocí funkce digitalWrite(LED_PIN, LOW);.

analogRead(), map()

```
const int POTENTIOMETER_PIN = A0; //Definice čísla analogového pinu pro
potenciometr
const int LED_PIN = 9; // Definice čísla pinu pro LED diodu
void setup() {
  pinMode(LED_PIN, OUTPUT); //Nastavení pinu LED jako výstup
}
void loop() {


int sensorValue = analogRead(POTENTIOMETER_PIN); // čtení hodnoty z
potenciometru
int brightness = map(sensorValue, 0, 1023, 0, 255); // Převod čtené
hodnoty na jas LED (0-255)

analogWrite(LED_PIN, brightness); // Nastavení jasu LED diody na základě
hodnoty z potenciometru
}
```

Příklad 12 Arduino IDE – analogRead()

V tomto kódu jsem použil funkci analogRead(), která mi umožňuje číst hodnoty z analogového pinu na Arduinu. Tato funkce přijímá jeden parametr - číslo analogového pinu, ze kterého chci číst hodnotu. V tomto případě jsem si zvolil číslo pinu pro potenciometr a nastavil ho jako POTENTIOMETER_PIN, kterým je A0, což je analogový pin 0 na Arduinu. Když jsem zavolal funkci analogRead(POTENTIOMETER_PIN);, Arduino přečetlo hodnotu napětí na pinu A0 a převedlo ji na číselnou hodnotu mezi 0 a 1023, kde 0 odpovídá nulovému napětí a 1023 odpovídá maximálnímu napětí.

Dále jsem v kódu použil funkci map(), která slouží k převodu hodnoty z jednoho rozsahu na jiný. V mém případě jsem potřeboval převést hodnotu z potenciometru, která je v rozsahu 0 až 1023, na hodnotu jasu LED, která je v rozsahu 0 až 255. To jsem provedl pomocí funkce map(sensorValue, 0, 1023, 0, 255);, kde sensorValue je hodnota přečtená z potenciometru.

A nakonec jsem využil funkci analogWrite(), abych řídil jas LED. Tato funkce přijímá dva parametry - číslo pinu pro LED a hodnotu jasu. V mém kódu jsem nastavil hodnotu jasu na brightness, která byla předtím vypočítána funkcí map().

AnalogWrite()

```
const int LED_PIN = 9; // Definice čísla pinu pro LED diodu
void setup() {
  pinMode(LED_PIN, OUTPUT); //Nastavení pinu LED jako výstup
}
void loop() {
  for (int brightness = 0; brightness < 255; brightness++){
    analogWrite(LED_PIN, brightness); //Nastavení jasu LED diody
    delay(10); // Počkání 10ms
  }

for (int brightness = 255; brightness > 0; brightness--){
  analogWrite(LED_PIN, brightness); //Nastavení jasu LED diody
  delay(10); // Počkání 10ms
}
}
```

Příklad 13 Arduino IDE – AnalogWrite()

V tomto kódu používám funkci analogRead(), která mi umožňuje číst hodnoty z analogového pinu na Arduino desce. Tato funkce přijímá jeden parametr – číslo analogového pinu, ze kterého chci číst hodnotu. V tomto případě jsem zvolil číslo pinu pro potenciometr a definoval jej jako POTENTIOMETER_PIN, což je A0, tedy analogový pin 0 na Arduino. Když zavolám funkci analogRead(POTENTIOMETER_PIN), Arduino přečte hodnotu napětí na pinu A0 a převede ji na číselnou hodnotu v rozsahu 0 až 1023, kde 0 odpovídá nulovému napětí a 1023 maximálnímu napětí.

Dále v kódu využívám funkci map(), která slouží k převodu hodnoty z jednoho rozsahu na jiný. V mém případě potřebuji převést hodnotu z potenciometru, která je v rozsahu 0 až 1023, na hodnotu jasu LED diody, která je v rozsahu 0 až 255. Toto provedu pomocí funkce map(sensorValue, 0, 1023, 0, 255), kde sensorValue je hodnota přečtená z potenciometru.

Nakonec využívám funkci analogWrite(), abych řídil jas LED diody. Tato funkce přijímá dva parametry – číslo pinu pro LED a hodnotu jasu. V mém kódu nastavuji hodnotu jasu na proměnnou brightness, která byla předtím vypočítána funkcí map().

Tedy výstup tohoto projektu je, že se LED postupně rozsvítí a zase zhasne.

Seriál.begin()

```
void setup() {
  Serial.begin(9600); // zahájení sériové komunikace s rych. 9600 baudů
}

void loop() {
  Serial.println("Ahoj, všichni"); // Odešle zprávu "Ahoj, světe!" přes
sériovou linku
  delay(1000); // Počká 1 sekundu
}
```

Příklad 14 Arduine IDE – Seriál.Begin()

V tomto kódu se funkce Serial.begin(9600); používá k zahájení sériové komunikace s rychlostí 9600 baudů.

Serial.print(), Serial.println()

```
void setup() {
  Serial.begin(9600); // zahájení sériové komunikace s rych. 9600 baudů
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.print("Analogová hodnota");
  Serial.println(sensorValue); // Odešle zprávu "Ahoj, světe!" přes
sériovou linku
  delay(1000); // Počká 1 sekundu
}
```

Příklad 15 Arduine IDE – Serial.print(), Serial.println()

Na začátku programu jsem použil funkci Serial.begin(9600);, která zahajuje sériovou komunikaci mezi Arduinem a počítačem. Tato funkce je klíčová pro monitorování výstupů programu a jeho ladění v Arduino IDE. V hlavní smyčce loop() jsem pak využil funkce Serial.print() a Serial.println(), abych vypsal textový popis "Analogová hodnota" a hodnoty přečtené z analogového pinu A0, uložené ve proměnné sensorValue.

Random()

```
void setup() {
  Serial.begin(9600); // Zavedení sériové komunikace
  randomSeed(analogRead(0)); // generátor náhodných čísel
}

void loop() {
  int randomNumber = random(0, 10); // náhodná čísla 0,9
  Serial.print("Náhodné číslo: ");
  Serial.println(randomNumber); // Vypsije náhodné číslo do Sériové
komunikace
  delay(1000); // Čeká sekundu
}
```

Příklad 16 Arduino IDE – random()

V části setup() jsem použil funkci randomSeed(analogRead(0)), která nastavuje základní hodnotu pro generování náhodných čísel z analogového pinu A0. V hlavní smyčce loop() je využita funkce random(0, 10), generující náhodná čísla v rozsahu od 0 do 9. Tato čísla jsou následně vypsána na sériovou linku prostřednictvím funkcí Serial.print() a Serial.println().

Ve videu demonstrujeme správné používání parametrů a návratových hodnot. Procházíme všechny funkce, které jsme výše zmínili, a implementujeme je do základních projektů. Tímto způsobem uživatelé získávají porozumění fungování těchto funkcí a jejich praktickém využití.

6.7.8 Podpogramy

V této lekci jsme se zaměřili na důležitost a využití podprogramů ve vývoji softwaru. Na úvod jsme zdůraznili, že vytváření a používání funkcí v Arduinu lze přirovnat ke skládání stavebnice, kde každá funkce představuje jedinečný dílek přispívající k vytvoření rozsáhlejšího celku projektu.

V teoretické části jsme se podrobněji zaměřili na základní koncepty funkcí v Arduino, což zahrnovalo syntaxi pro definici funkcí, metody volání funkcí, práci s parametry a návratovými hodnotami, rozlišení mezi lokálními a globálními proměnnými, a také na rozdíl mezi uživatelsky definovanými funkcemi a funkcemi poskytovanými knihovnami.

Praktická část lekce obsahovala video s praktickými příklady, které ilustrovaly jednotlivé koncepty a demonstrovaly, jak je aplikovat ve vlastních projektech.

V závěru lekce jsme zdůraznili, že správně navržené a implementované podprogramy mohou výrazně zjednodušit a urychlit vývoj projektů v Arduino IDE. Upozornili jsme také na důležitost správného zacházení s návratovými hodnotami, parametry a chybami při práci s funkcemi. Dále jsme vyzdvihli význam optimalizace výkonu a dokumentace kódu jako klíčových aspektů při práci s podprogramy.

6.7.9 Bredboard

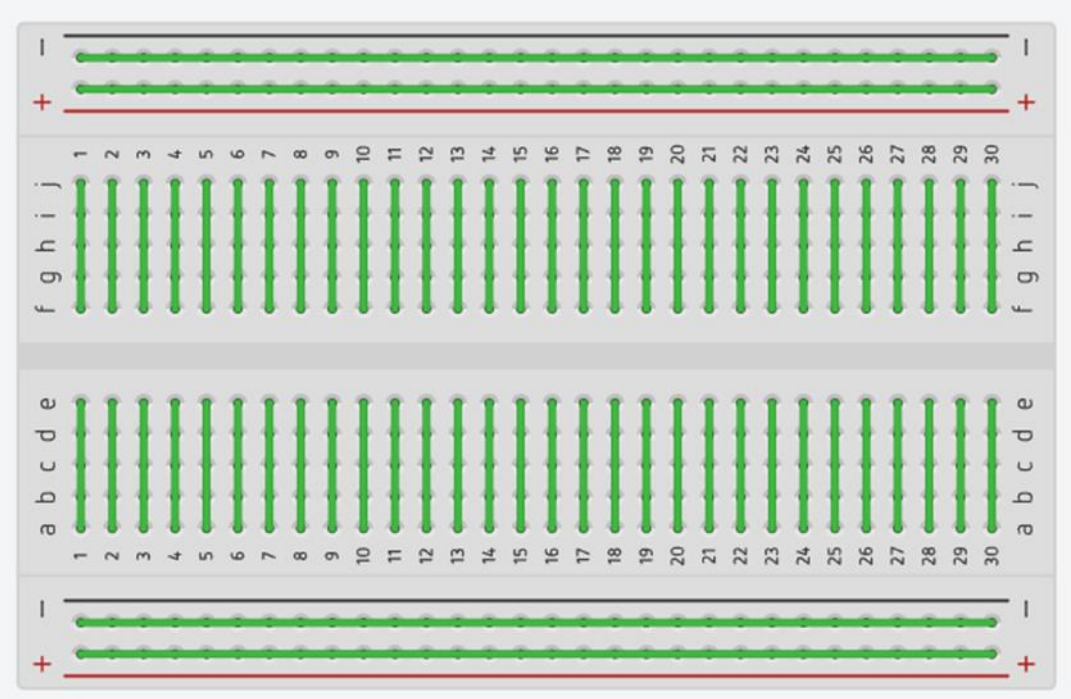
V dnešní lekci jsme si vysvětlili, co představuje breadboard a jak jej používat. Na úvod jsme zdůraznili, že breadboard je jako vaš partner při tvorbě s Arduinem UNO, poskytující bezpečné a pohodlné prostředí pro rychlé prototypování a testování nápadů bez ohledu na úroveň vaší zkušenosti.

V teoretické části jsme podrobně prozkoumali strukturu breadboardu a jeho využití. Zjistili jsme, že breadboard nabízí mřížku otvorů, do kterých lze zasunout nožičky elektronických součástek a propojit je pomocí propojovacích vodičů. Tato vlastnost umožňuje rychlé a snadné sestavování a testování různých obvodů bez nutnosti trvalého pájení.

V praktické části lekce jsme měli k dispozici video s praktickými příklady, které ilustrovalo použití breadboardu a ukázalo, jak jej efektivně využít při tvorbě elektronických projektů.

Závěr lekce byl zaměřen na důležité aspekty, na které je třeba si dávat pozor při práci s breadboardem. Zahrnovaly správné zapojení pinů, ochranu součástek a výběr kvalitního breadboardu pro spolehlivé spojení.

Bradboard funkčnost



Obrázek 14 - Breadboard

Abychom lépe pochopili fungování breadboardu, můžeme se podívat na obrázek, který ukazuje spolupráci jednotlivých komponent. Vezměme si jako příklad první řadu zdířek, kterou označíme jako řadu 1 a sloupec A. Přivedeme-li napájecí napětí do díry v řadě 1 a sloupci A, toto napětí se rozšíří do všech zdířek v této řadě. To znamená, že všechny zdířky v řadě 1 jsou elektricky propojené a mohou sdílet připojené napětí. Chceme-li propojit dva různé sloupce, například sloupec A a B, můžeme toho dosáhnout pomocí propojovacího drátu. Připojením díry 1A ke zdířce 1B drátem vytvoříme elektrické spojení mezi sloupci, což umožňuje přenos napětí mezi různými částmi breadboardu.

6.7.10 Připojení desky k počítači

V této lekci jsme se zaměřili na téma připojení desky k počítači a zdůraznili jsme důležitost tohoto kroku pro úspěšný vývoj našich projektů. Upozornili jsme také na potřebu postupovat opatrně a správně.

V teoretické části jsme se zaměřili na proces fyzického připojení desky k počítači. Vysvětlili jsme, jak připravit desku a potřebné kabely a jak je následně fyzicky propojit s počítačem pomocí USB kabelu.

V praktické části jsme začali videm, jak správně zapojit desku k počítači. Během videa jsme detailně ukázali každý krok procesu a zdůraznili jsme důležité body, na které je třeba během připojování desky k počítači dbát.

V závěru jsme shrnuli nejdůležitější body lekce. Zopakovali jsme klíčové informace o připojení desky k počítači a zdůraznili jsme důležitost správného postupu a opatrnosti. Také jsme si připomněli, na co si během procesu připojování desky k počítači dát pozor, abychom zajistili bezproblémový průběh a úspěšný vývoj našich projektů.

6.7.11 LED dioda

V úvodu jsme se zaměřili na vysvětlení prvního zapojení naší LED diody s Arduinem. Prošli jsme postupně procesem připojení LED diody k Arduinu s využitím rezistoru, přičemž jsme se soustředili na správné zapojení anody a katody diody.

Teoretická část se zabývala podrobným rozborem konceptu LED diody a jejího fungování. Vysvětlili jsme principy anody a katody, stejně jako účel a význam použití rezistoru k omezení proudu. Diskutovali jsme také o důležitosti správného zapojení diody v elektrickém obvodu.

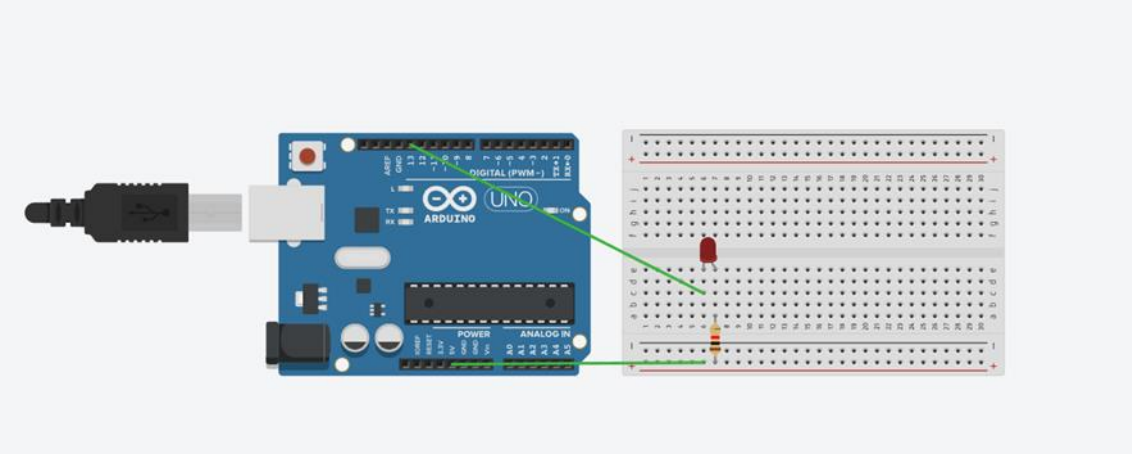
V praktické části jsme demonstrativně ukázali postup připojení LED diody k Arduinu a její ovládání. Skrze praktický experiment jsme si osvojili základní dovednosti v práci s LED diodami a Arduinem, což nám umožní úspěšně pracovat na dalších projektech.

Závěrem jsme shrnuli klíčové body této lekce a zdůraznili důležitost správného připojení LED diody a použití rezistoru. Potvrdili jsme, že jsme získali pevné základy pro úspěšné experimentování s LED diodami a Arduinem.

```
const int LED_PIN = 13;
void setup() {
  pinMode(LED_PIN, OUTPUT); //Nastavení pinu LED jako výstup
}
void loop() {
  digitalWrite(13, HIGH); //zapnout LED
  delay(1000); //Počkat vteřinu
  digitalWrite(13, LOW); // Vypnout LED
  delay(1000); //Počkat vteřinu
}
```

Příklad 17 Arduino IDE – blikání LED

Nejprve jsme definovali proměnnou LED_PIN s hodnotou 13, která označuje pin pro připojení LED diody. V rámci funkce setup() jsme tento pin nastavili jako výstupní. Samotné blikání LED probíhá v hlavním těle programu, v rámci funkce loop(). Zde se nejprve LED dioda zapne nastavením výstupního napětí na HIGH pomocí příkazu digitalWrite(13, HIGH). Poté program čeká jednu sekundu (1000 milisekund) díky funkci delay(1000). Následuje vypnutí LED diody příkazem digitalWrite(13, LOW) a program opět čeká jednu sekundu. Tento cyklus se neustále opakuje.



Obrázek 15 – Tinkercad – LED dioda

6.7.12 Tlačítko

V teoretické části jsme tlačítko podrobněji představili jako základní elektronickou součástku. Diskutovali jsme o tom, jak tlačítka fungují a jaké principy ovládání mají. Dále jsme se věnovali různým typům tlačítek a jejich využití v elektronických projektech.

V praktické části jsme demonstrovali správné připojení tlačítka k Arduinu a jeho využití pro interakci s naším programem. Prostřednictvím praktického experimentu jsme získali zkušenosti s tlačítkem a pochopili jeho aplikace v reálných projektech.

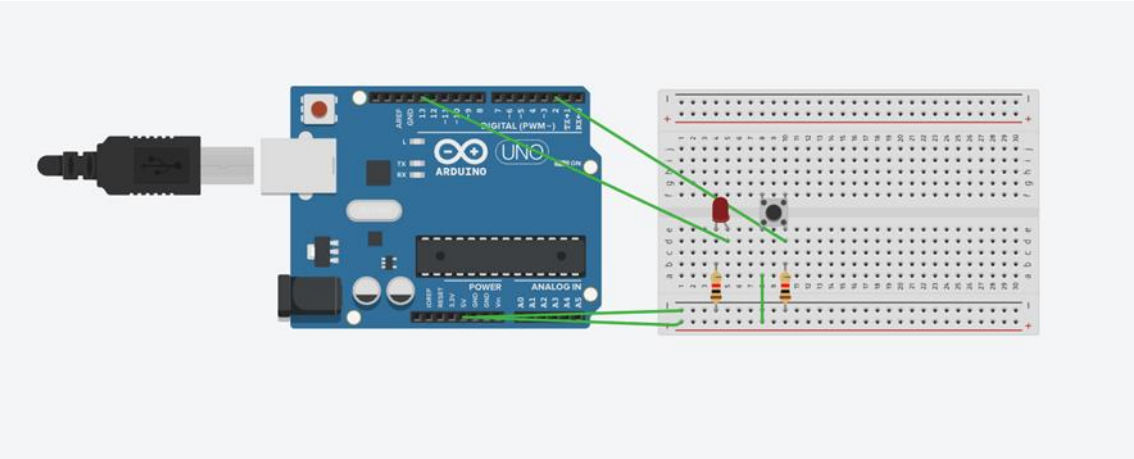
Závěrem jsme shrnuli klíčové body této lekce a zdůraznili význam tlačítka jako komponenty pro interakci s Arduinem. Potvrdili jsme, že jsme získali nezbytné znalosti a dovednosti pro úspěšné začlenění tlačítka do našich budoucích projektů s Arduinem.

```
const int BUTTON = 2; // Nastavení tlačítka
const int LEDPIN = 13;  //Nastavení Ledky
void setup() {
  pinMode(BUTTON, INPUT); // Nastavení tlačítka jako vstup
  pinMode(LEDPIN, OUTPUT);   // Nastavení LED jako výstup
}
void loop() {
  int button1 = digitalRead(BUTTON); // čtení z tlačítka
  if (button1 == LOW) {
    digitalWrite(LEDPIN, HIGH); // ON LED
  } else {
    digitalWrite(LEDPIN, LOW);  // OFF LED
  }
}
```

Příklad 18 Arduino IDE – tlačítko

Tento kód umožňuje ovládání LED diody pomocí tlačítka připojeného k Arduinu. Nejprve definujeme proměnné pro piny: BUTTON pro tlačítko a LEDPIN pro LED diodu. V rámci funkce setup() nastavujeme, který pin bude sloužit jako vstup a který jako výstup. Pin tlačítka je nastaven jako vstup a pin LED jako výstup.

V hlavní smyčce loop() se průběžně kontroluje stav tlačítka. Pokud je tlačítko stisknuto, LED dioda se rozsvítí; není-li stisknuto, LED dioda zhasne. Tento mechanismus umožňuje ovládat LED prostřednictvím stisku a uvolnění tlačítka.



Obrázek 16 – Tinkercad – Button

6.7.13 Snímaní světla

Na úvod jsme se zaměřili na základní principy snímání světla pomocí Arduina. Objasnili jsme, které součástky jsou potřebné k měření intenzity světla a jak lze tuto funkcionalitu využít v našich projektech.

V teoretické části jsme se podrobněji zabývali fungováním světelných senzorů a jejich různými typy. Diskutovali jsme o principu, na kterém světelné senzory pracují, a o faktorech, které ovlivňují jejich citlivost a přesnost.

Praktická část zahrnovala demonstraci, jak připojit světelný senzor k Arduinu a jak využít naměřené hodnoty intenzity světla v našem kódu. Provedli jsme experimenty s různými světelnými podmínkami a analyzovali jsme výsledky.

V závěrečné části jsme shrnuli klíčové poznatky z této lekce a zdůraznili význam snímání světla jako důležité funkce v mnoha projektech s Arduinem. Potvrdili jsme, že jsme získali potřebné dovednosti pro úspěšné využití světelných senzorů v našich budoucích projektech.

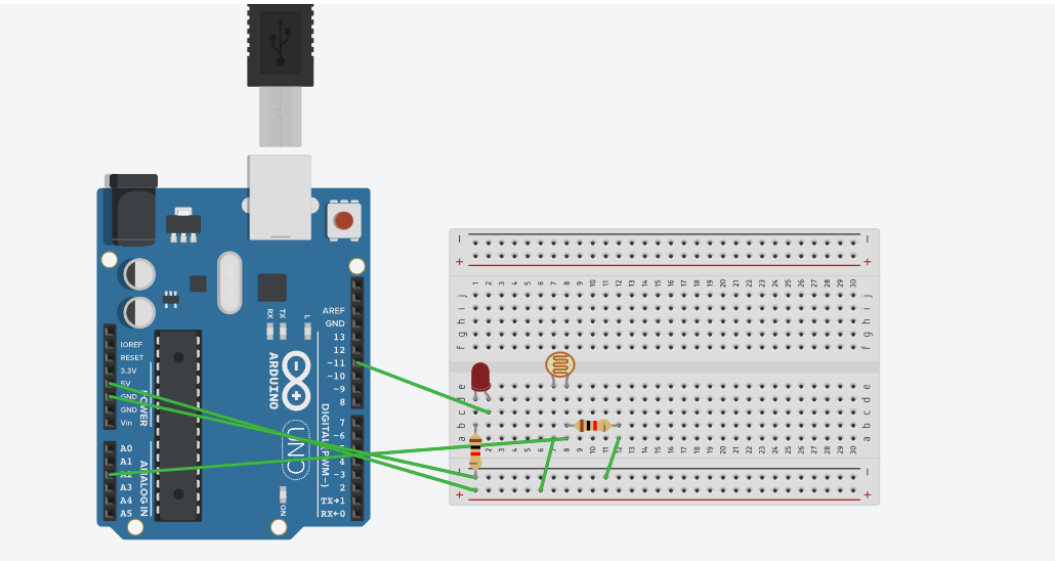
```
int ldrPin = A2;
int ledPin = 11;

void setup()  {
  Serial.begin(9600);
  pinMode(ldrPin, INPUT_PULLUP);
  pinMode(ledPin,  OUTPUT);
}

void loop() {
  int analogValue = analogRead(A2);

  Serial.println("Analogová hodnota = ");
  Serial.print(analogValue);

  if  (analogValue < 80) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin,  LOW);
  }
  delay(500);
}
```

Příklad 19 Arduino IDE – Snímání světla

V tomto příkladu jsme nejprve definovali proměnné ldrPin a ledPin, které určují piny připojené k LDR a LED. Ve funkci setup() jsou tyto piny nastaveny takto: pin pro LDR je konfigurován jako vstup s pull-up rezistory (INPUT_PULLUP) a pin pro LED jako výstup (OUTPUT). Pull-up rezistory pomáhají udržet stabilní stav pinu, když není aktivní žádný vstup.

V hlavní smyčce loop() se neustále provádí čtení hodnoty z LDR pomocí funkce analogRead(A2), která čte analogovou hodnotu z pinu A2, kde je připojen LDR. Tato hodnota je poté zobrazena v sériovém monitoru připojeném k Arduinu pomocí Serial.println().

Následně se hodnota porovnává s předem definovanou mezí (v tomto případě 80). Pokud je naměřená hodnota menší než 80, což naznačuje, že okolní světlo je pod určitou úrovní, zapne se LED dioda pomocí funkce digitalWrite(ledPin, HIGH). Pokud je hodnota větší nebo rovna 80, LED se vypne pomocí digitalWrite(ledPin, LOW).



Obrázek 17 – Tinkercad – snímání světla

6.7.14 Měření vlhkosti půdy

V úvodu této lekce jsme se zaměřili na pochopení základních principů měření vlhkosti půdy pomocí Arduina. Diskutovali jsme o důležitosti monitorování vlhkosti půdy pro úspěšný růst rostlin a o tom, jak může Arduino poskytnout efektivní řešení pro tuto úlohu.

V teoretické části jsme prozkoumali fungování senzoru vlhkosti půdy a jeho principy. Zjistili jsme, jakým způsobem senzor detekuje vlhkost půdy a jaké faktory mohou ovlivnit jeho výkon. Dále jsme se seznámili s různými typy senzorů dostupných na trhu a jejich vlastnostmi.

Praktická část lekce zahrnovala demonstraci připojení senzoru vlhkosti půdy k Arduinu a programování Arduina k měření a zobrazení hodnot vlhkosti půdy. Provedli jsme experimenty s různými typy půdy a analyzovali naměřené hodnoty.

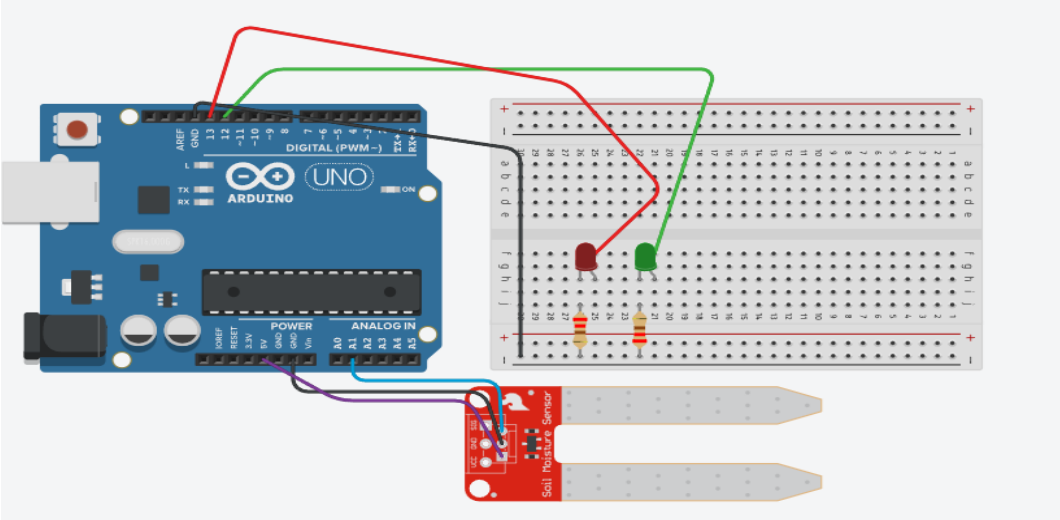
V závěrečné části jsme zhodnotili výsledky našich experimentů a zdůraznili důležitost správného monitorování vlhkosti půdy pro zdravý růst rostlin. Potvrdili jsme, že jsme získali potřebné znalosti k úspěšnému využití senzoru vlhkosti půdy v našich budoucích projektech s Arduino.

```
int vlhkost = 0;
void setup()
{
  pinMode(A1, INPUT);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop()
{
  vlhkost = analogRead(A1);
  Serial.println(vlhkost);
  delay(500); // Počkej půl vtěřiny
  if (vlhkost <= 500) {
    digitalWrite(13, HIGH);
    digitalWrite(12, LOW);
  } else {
    digitalWrite(13, LOW);
    digitalWrite(12, HIGH);
  }
}
```

Příklad 20 Arduino IDE – Vlhkost

Tento kód slouží k měření úrovně vlhkosti pomocí senzoru vlhkosti připojeného k pinu A1 Arduina. Nejprve jsou nastaveny piny A1, 13 a 12. Pin A1 je nastaven jako vstup pro čtení hodnoty vlhkosti ze senzoru, zatímco piny 13 a 12 jsou nastaveny jako výstupy pro ovládání LED diod.

V hlavní části programu je neustále čtená hodnota vlhkosti ze senzoru vlhkosti. Tato hodnota je pak poslána do sériového monitoru pro zobrazení. Podle hodnoty vlhkosti se poté buď LED na pinu 13 zapne a LED na pinu 12 vypne, pokud je vlhkost menší nebo rovna 500 (což znamená nižší vlhkost), nebo se LED na pinu 13 vypne a LED na pinu 12 zapne, pokud je vlhkost vyšší než 500.



Obrázek 18 – Tinkercad – Vlhkost půdy

6.7.15 Modul zalévání

V úvodu této lekce jsme se zaměřili na koncept automatického zalévání rostlin pomocí modulu připojeného k Arduinu. Diskutovali jsme o důležitosti udržení správné úrovně vlhkosti půdy pro zdravý růst rostlin a o výhodách automatického zalévání.

V teoretické části jsme prozkoumali základní principy fungování modulu zalévání a jeho komponent. Zjistili jsme, jak modul detekuje úroveň vlhkosti půdy a jaké mechanismy řídí přívod vody k rostlinám. Dále jsme se seznámili s různými typy modulů zalévání dostupných na trhu a jejich vlastnostmi.

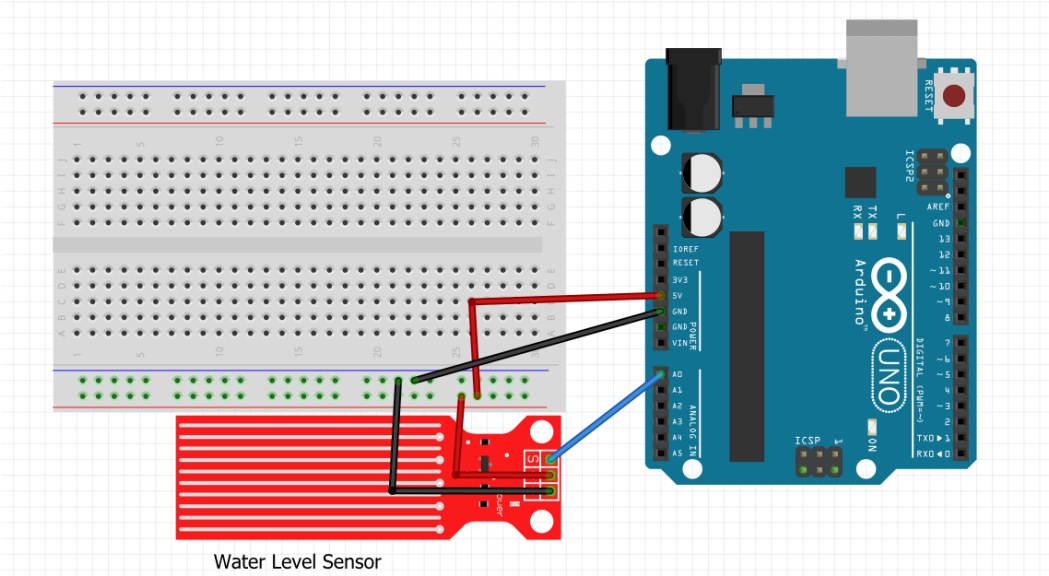
Praktická část lekce zahrnovala demonstraci připojení modulu zalévání k Arduinu a programování Arduina pro řízení zalévání na základě naměřených hodnot vlhkosti půdy. Provedli jsme experimenty s různými nastaveními zalévání a analyzovali jejich účinnost.

V závěrečné části jsme zhodnotili výsledky našich experimentů a zdůraznili význam automatického zalévání pro udržení zdraví a vitality rostlin. Potvrdili jsme, že jsme získali dostatečné znalosti k úspěšnému využití modulu zalévání v našich budoucích projektech s Arduino.

```
int vlhkost = 0;
void setup()
{
  pinMode(A1, INPUT);
  Serial.begin(9600);

}
void loop()
{
  vlhkost = analogRead(A0);
  Serial.println(vlhkost);
  delay(1000);
  if (vlhkost < 25){
    Serial.println("Senzor není ve vodě");
    }else{
      Serial.println("Je ve vodě");
    }
}
```

Příklad 21 Arduine IDE – Senzor vody

Tento kód umožňuje čtení úrovně vlhkosti za pomocí analogového senzoru, který je připojen k pinu A0 na desce Arduino. Proměnná "vlhkost" uchovává hodnotu získanou z tohoto senzoru. Ve funkci setup() dochází k inicializaci pinů, přičemž pin A1 je nastaven jako vstup pro připojení senzoru. Zároveň je zahájena sériová komunikace s rychlostí 9600 bps pro komunikaci s počítačem. V hlavní smyčce (loop()) Arduino opakovaně čte hodnotu vlhkosti za použití funkce analogRead(A0), zobrazuje ji na sériový monitor a poté čeká jednu sekundu s použitím funkce delay(1000). Pokud je hodnota vlhkosti nižší než 25, Arduino vypíše zprávu "Senzor není ve vodě"; pokud je vyšší, vypíše "Je ve vodě".



Obrázek 19 – Fritzing – Vodní senzor

6.7.16 Modul měření teploty

V úvodní části této lekce jsme se zaměřili na modul měření teploty a jeho důležitost v elektronických projektech. Diskutovali jsme, jak modul měření teploty umožňuje Arduino snímat teplotní údaje ze svého okolí, a jak lze tuto informaci využít k různým účelům, jako je monitorování prostředí či řízení klimatizace.

V teoretické části jsme prozkoumali princip fungování modulu měření teploty a různé typy teplotních senzorů, které lze použít s Arduinem. Vysvětlili jsme, jak Arduino komunikuje s teplotním senzorem a jak získávat teplotní data ve formátu digitálních dat.

V praktické části jsme provedli experimenty s konkrétním teplotním senzorem a připojili jsme jej k Arduinu. Ukázali jsme, jak načítat teplotní hodnoty z modulu měření teploty pomocí Arduina a jak tyto hodnoty využívat k různým účelům, jako je zobrazování teploty na displeji nebo ovládání zařízení v závislosti na teplotních podmínkách.

V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití modulu měření teploty v různých projektech. Potvrdili jsme, že jsme získali dostatečné znalosti k úspěšnému použití teplotních senzorů s Arduinem a že tyto senzory představují užitečný nástroj pro monitorování a řízení teploty v elektronických zařízeních.

```
const int sensorPin = A0;

void setup() {
  Serial.begin(9600);
  delay(1000);
}

void loop() {
  int sensorValue = analogRead(sensorPin); // hodnota ze senzoru

  float teplota = sensorValue ;

  Serial.print("Teplota: ");
  Serial.print(teplota);
  Serial.println("°C");

  delay(3000); // Počkat 3 sekundy
}
```

Příklad 22 Arduino IDE – Teplota

V dnešní lekci jsme se zaměřili na práci s tepelným senzorem pomocí desky Arduino Uno. Tepelný senzor je připojen k desce prostřednictvím analogového vstupu A0. Nejprve jsme definovali proměnnou sensorPin, která specifikuje pin, ke kterému je tepelný senzor připojen. V rámci funkce setup() jsme inicializovali sériovou komunikaci s rychlostí 9600 baudů a vyčkali jsme jednu sekundu na dokončení inicializace. V hlavní smyčce loop() jsme neustále četli hodnotu z tepelného senzoru pomocí funkce analogRead(sensorPin). Tato hodnota byla uložena do proměnné sensorValue a interpretována jako teplota, ačkoliv ve skutečnosti představuje pouze analogovou hodnotu, kterou senzor poskytuje.

6.7.17 Modul vlhkosti vzduchu

Na začátku této lekce jsme se věnovali významu monitorování vlhkosti vzduchu a jeho vlivu na lidské zdraví a pohodu. Diskutovali jsme o faktorech ovlivňujících úroveň vlhkosti vzduchu a důležitosti udržování této úrovně v optimálním rozmezí.

V teoretické části jsme prozkoumali jsme principy fungování modulu pro měření vlhkosti vzduchu a jeho jednotlivé komponenty. Zjistili jsme, jak modul snímá vlhkost vzduchu a jaké senzory jsou k tomuto účelu nejčastěji používány. Dále jsme se seznámili s různými metodami měření vlhkosti vzduchu a s faktory, které je při měření nutné zohlednit.

V praktické části lekce jsme provedli demonstraci připojení modulu pro měření vlhkosti vzduchu k Arduinu a programování Arduina pro monitorování vlhkosti vzduchu v reálném čase. Experimentovali jsme s různými úrovněmi vlhkosti a analyzovali vliv různých faktorů na měřené hodnoty.

V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití modulu pro měření vlhkosti vzduchu v různých aplikacích, jako jsou automatizace klimatizace, monitorování kvality vzduchu v interiérech a řízení zavlažovacích systémů. Potvrdili jsme, že jsme získali dostatečné znalosti pro úspěšné nasazení tohoto modulu v našich budoucích projektech s Arduino.

```
const int sensorPin = A0;

void setup() {
  Serial.begin(9600);
  delay(1000);
}
void loop() {
  int sensorValue = analogRead(sensorPin); // hodnota ze senzoru

  float percentvlhkosti = map(sensorValue, 0, 1023, 0, 100);

  Serial.print("Vlhkost: ");
  Serial.print(percentvlhkosti);
  Serial.println("%");


  delay(3000); // Počkat 3 sekundy
}
```

Příklad 23 Arduino IDE – Vlhkost

V dnešní lekci jsme se zaměřili na měření vlhkosti pomocí desky Arduino Uno a senzoru vlhkosti, který byl připojen k analogovému vstupu A0. Nejprve jsme definovali proměnnou sensorPin, která určuje pin, k němuž je senzor připojen. Ve funkci setup() jsme inicializovali sériovou komunikaci s rychlostí 9600 baudů a vyčkali jsme jednu sekundu, aby se inicializace úspěšně dokončila. V hlavní smyčce loop() jsme opakovaně četli hodnotu z senzoru vlhkosti pomocí funkce analogRead(sensorPin). Tato hodnota byla uložena do proměnné sensorValue a následně jsme ji pomocí funkce map() převedli na procentuální vlhkost, která byla uložena do proměnné percentVlhkosti.

6.7.18 Modul osvětlení

V úvodní části této lekce jsme se zaměřili na význam monitorování osvětlení a jeho dopady na životní prostředí i lidské zdraví. Diskutovali jsme o důležitosti adekvátního osvětlení v různých situacích a prostředích, jak v interiérech, tak v exteriérech.

V teoretické části jsme prozkoumali principy fungování modulu pro měření osvětlení a jeho komponenty. Popsali jsme, jak senzor osvětlení detekuje okolní světlo a jaké parametry jsou přitom měřeny. Seznámili jsme se s různými jednotkami měření osvětlení a faktory ovlivňujícími úroveň osvětlení v daném prostoru.

V praktické části lekce jsme demonstrovali připojení modulu pro měření osvětlení k Arduinu a programování Arduina pro monitorování osvětlení v reálném čase. Provedli jsme experimenty s různými intenzitami osvětlení a analyzovali jsme vliv různých zdrojů světla na měřené hodnoty.

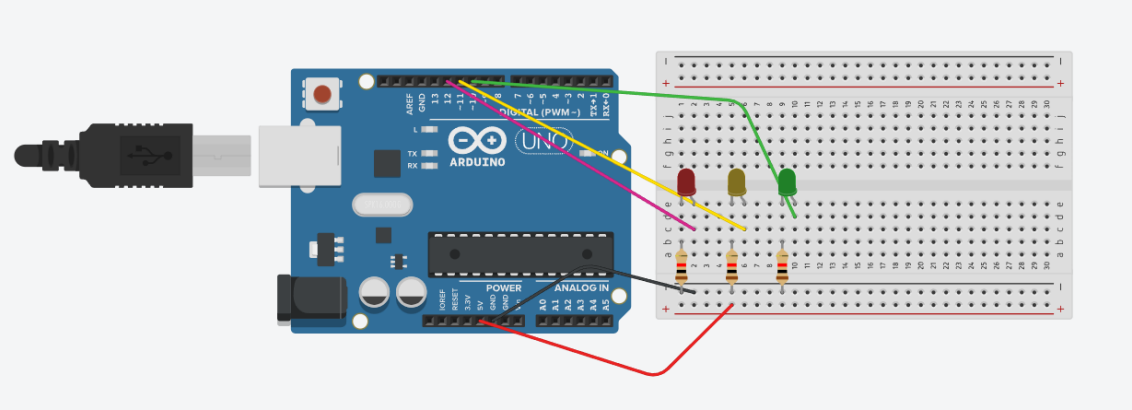
V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití modulu pro měření osvětlení v různých aplikacích. Zahrnuli jsme automatické řízení osvětlení v interiérech, monitorování denního osvětlení ve venkovních prostředích a optimalizaci spotřeby energie v osvětlovacích systémech. Potvrdili jsme, že jsme získali dostatečné znalosti pro úspěšné nasazení modulu pro měření osvětlení v našich budoucích projektech s Arduino.

```
int red = 12;
int yellow = 11;
int green = 10;

void setup(){

  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green,  OUTPUT);

}
void loop(){
digitalWrite(red, HIGH);
 delay(3000);
digitalWrite(red,  LOW);

  digitalWrite(yellow, HIGH);
delay(3000);
  digitalWrite(yellow,  LOW);

digitalWrite(green, HIGH);
delay(3000);
digitalWrite(green,  LOW);

  digitalWrite(yellow, HIGH);
delay(3000);
  digitalWrite(yellow,  LOW);

  digitalWrite(red, HIGH);
 delay(3000);
digitalWrite(red,  LOW);

}
```

Příklad 24 Arduino IDE – osvětlení

V dnešní lekci jsme se zaměřili na praktickou stránku elektroniky. Konkrétně jsme se naučili, jak zapojit semafor, což je zařízení sloužící k řízení dopravy, které signalizuje řidičům, kdy mají zastavit a kdy mohou pokračovat. Začali jsme tím, že jsme definovali piny pro červenou, žlutou a zelenou LED diodu. Tyto LED diody jsme pak připojili k Arduinu. Poté jsme ve funkci setup() nastavili tyto piny jako výstupy, což umožní ovládat LED diody pomocí Arduina. V hlavní smyčce loop() jsme vytvořili sekvenci pro simulaci chování semaforu. Nejprve jsme rozsvítili červenou LED diodu, což signalizuje řidičům, že mají zastavit. Po krátké prodlevě jsme červenou LED diodu zhasli a rozsvítili žlutou LED diodu, což označuje připravenost k zastavení. Po další krátké prodlevě jsme

zhasli žlutou LED diodu a rozsvítili zelenou LED diodu, což řidičům signalizuje, že mohou pokračovat. Nakonec jsme ještě jednou zhasli zelenou LED diodu a vrátili se zpět na červenou, čímž jsme vytvořili kruhový cyklus chování semaforu.



Obrázek 20 Tinkercad – semafor

6.7.19 Krokové motory

V úvodu této lekce jsme se zaměřili na krokové motory a jejich využití v elektronických projektech. Diskutovali jsme o tom, jak krokové motory umožňují přesné ovládání polohy a jsou vhodné pro různé aplikace, včetně 3D tiskáren, CNC strojů, robotů a automatizace procesů.

V teoretické části jsme podrobně prozkoumali princip fungování krokových motorů a jejich základní typy, mezi které patří unipolární a bipolární motory. Popsali jsme, jak krokové motory pracují na základě pulzního signálu a jak různé konfigurace vinutí motoru ovlivňují jejich chování a výkon.

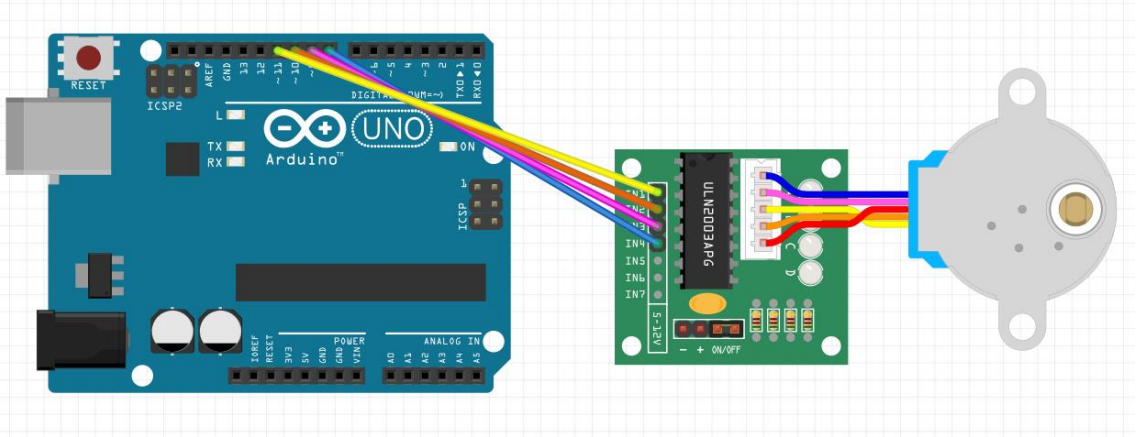
V praktické části lekce jsme provedli demonstraci zapojení krokového motoru k Arduinu a jeho programování pro ovládání polohy motoru. Experimentovali jsme s různými režimy pohybu motoru, včetně jednofázového, dvoufázového a mikrokrokového režimu, a analyzovali jsme jejich vliv na přesnost a plynulost pohybu.

V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití krokových motorů v různých aplikacích. Potvrdili jsme, že jsme získali dostatečné znalosti k úspěšnému nasazení krokových motorů v našich budoucích projektech s Arduino.

```
#include <Stepper.h> // Nutná knihovna

  #define KROKY 100 //počet kroků motoru

  Stepper krokovyMotor(KROKY, 8, 9, 10, 11);//Vytvoření třídy Stepper,
kde definujeme počet kroků motoru a připojené piny

  int predchozi = 0; //Předchozí odečtení z analagového pinu

  void setup() {
    krokovyMotor.setSpeed(99); // Nastavení rychlosti motoru na 99 otáček
za minutu (RPM)
  }
  void loop() {
    int hodnota = analogRead(0); //zisk hodnoty ze senzoru
    krokovyMotor.step(hodnota - predchozi); //počet kroků po změně v
hodnotě senzoru
    predchozi = hodnota; //Zapamatování předchozí hodnoty
  }
```

Příklad 25 Arduino IDE – Krokové motory

V dnešní lekci jsme se zaměřili na náš první projekt s krokovými motory. Na začátku jsme do projektu zahrnuli knihovnu Stepper.h, která nám umožňuje snadno ovládat krokové motory. Poté jsme definovali proměnnou STEPS, určující počet kroků, které krokový motor vykoná při jednom otočení. Následně jsme vytvořili instanci krokového motoru s použitím konstruktoru Stepper, kde jsme specifikovali počet kroků motoru a piny, ke kterým je motor připojen. Ve funkci setup() jsme nastavili rychlost motoru na 99 otáček za minutu (RPM). V hlavní smyčce loop() jsme četli hodnotu ze senzoru připojeného k pinu A0 pomocí funkce analogRead(). Poté jsme vypočítali rozdíl mezi aktuální a předchozí hodnotou senzoru a tento rozdíl použili k ovládání krokového motoru funkcí step().



Obrázek 21 – Fritzing – Krokový motor

6.7.20 LCD display

V úvodní části této lekce jsme se zaměřili na význam a využití LCD displejů v elektronických projektech. Diskutovali jsme o jejich flexibilitě, která umožňuje zobrazovat různé typy informací, a o jejich širokém rozšíření v různých zařízeních, od digitálních hodin po měřicí přístroje a zobrazovače informačních panelů.

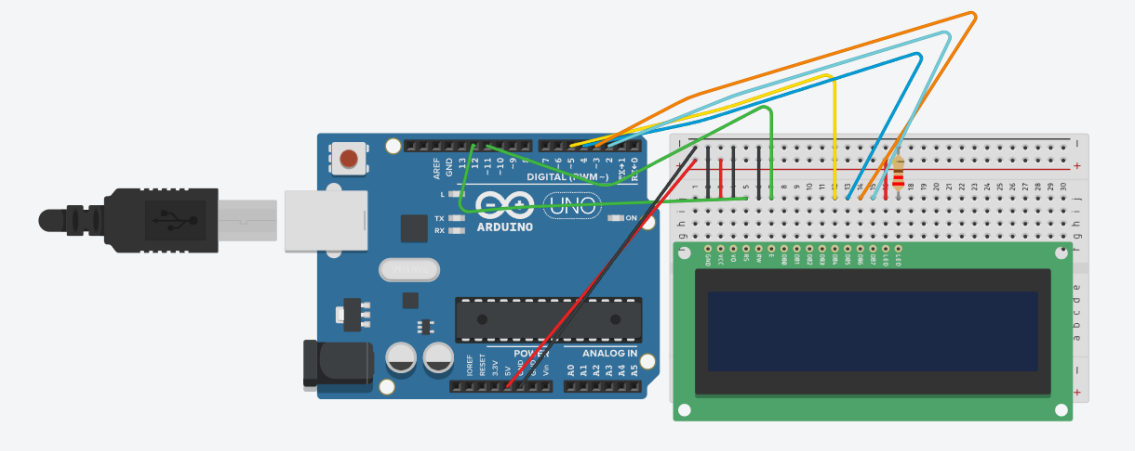
V teoretické části jsme se podrobně seznámili s principem fungování LCD displejů a jejich základními komponentami. Popsali jsme, jak LCD displeje zobrazují text a grafiku pomocí segmentů a jak jsou řízeny pomocí mikrokontrolérů. Dále jsme prozkoumali různé typy LCD displejů, jako jsou znakové, grafické a dotykové displeje, a diskutovali o jejich výhodách a nevýhodách.

V praktické části lekce jsme provedli demonstraci připojení LCD displeje k Arduinu a programování Arduina pro zobrazení textu a jednoduchých grafických prvků na displeji. Provedli jsme experimenty s různými typy zobrazení a analyzovali jejich vliv na čitelnost a efektivitu komunikace informací.

V závěrečné části jsme zhodnotili výsledky našich experimentů a diskutovali o praktickém využití LCD displejů v různých aplikacích, jako je zobrazování senzorických dat, uživatelské rozhraní pro ovládání elektronických zařízení nebo zobrazení informací v rámci vzdělávacích a zábavních projektů. Potvrdili jsme, že jsme získali dostatečné znalosti k úspěšnému nasazení LCD displejů v našich budoucích projektech s Arduino.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
}

void loop() {
  lcd.setCursor(0, 0);
  lcd.print("Ahoj kamaradi");
  lcd.setCursor(2, 1);
  lcd.print("Jak se mmte?");
}
```

Příklad 26 Arduino IDE – LCD display



Obrázek 22 – Tinkercad – LCD display

V dnešní lekci jsme se naučili, jak rozsvítit náš první LCD displej pomocí knihovny LiquidCrystal pro Arduino. Tato knihovna nám umožňuje snadné ovládání LCD displejů a bez ní by bylo psaní kódu pro ovládání displeje mnohem složitější a méně přehledné.

Na počátku našeho kódu jsme inicializovali objekt lcd pomocí konstruktoru LiquidCrystal. Specifikovali jsme piny na Arduinu, které jsou připojeny k řídícím signálům LCD displeje. Následně jsme ve funkci setup() inicializovali samotný displej metodou begin(), kde jsme také určili jeho rozměry (počet sloupců a řádků).

V hlavní smyčce programu, obsažené ve funkci loop(), jsme určili pozici kurzoru na displeji metodou setCursor(). Následně jsme text "Ahoj kamarádi!" a "Jak se máte?"

vypsali na displej metodou print(), čímž jsme dosáhli zobrazení textu v prvním a druhém řádku našeho LCD displeje.

Použití objektu lcd(12, 11, 5, 4, 3, 2, 1) je příkladem konstruktoru, speciální funkce vytvářející objekt typu LiquidCrystal. Tento způsob inicializace umožňuje efektivní využití knihovny pro ovládání LCD. Pin 12 slouží pro řízení RS (Register Select), pin 11 pro řízení E (Enable) a piny 5, 4, 3, a 2 řídí komunikaci přes datové linky D4 až D7 displeje

6.8 Proces tvorby videa

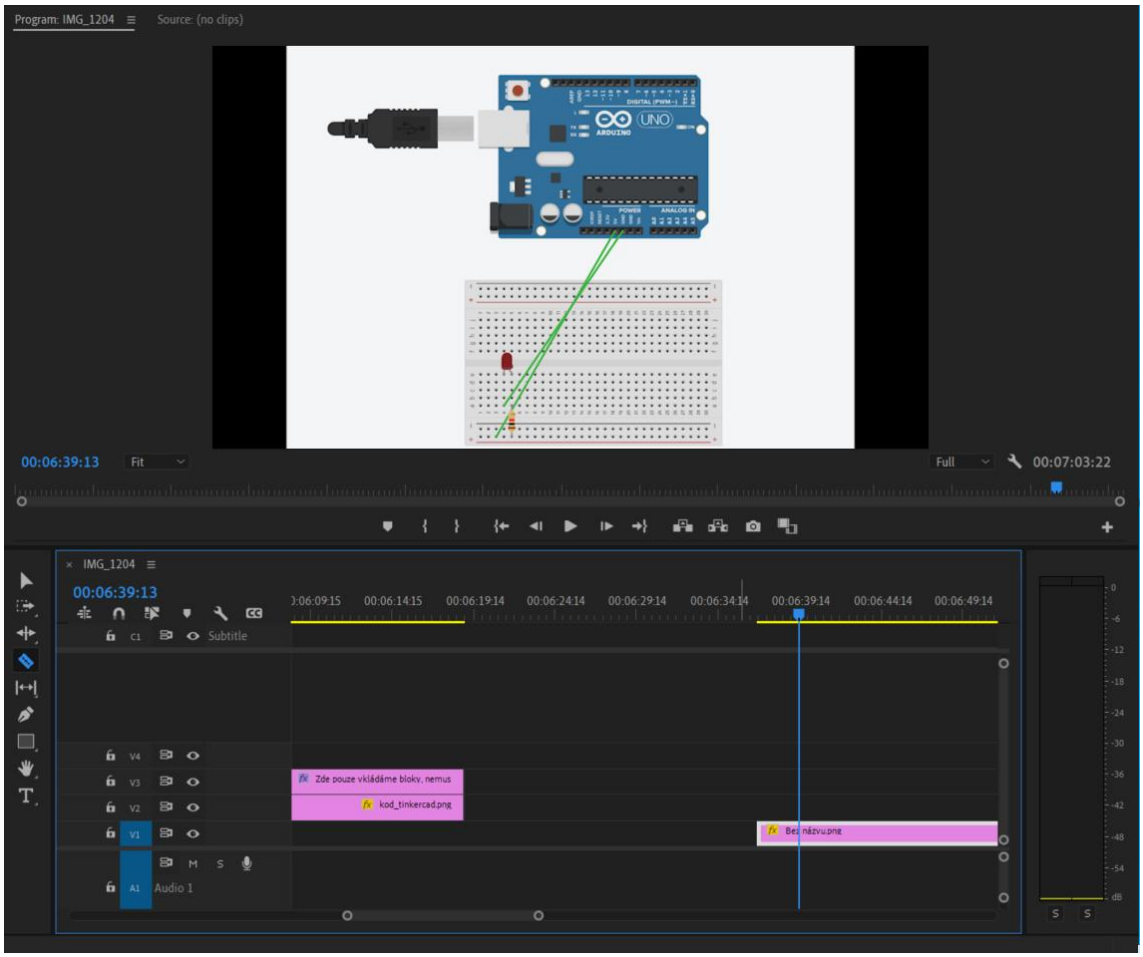
V této části jsem se zaměřil na celkový proces tvorby videa pro naše lekce, od natočení až po výstup hotového videa.

6.8.1 Natočení videa

Prvním krokem v procesu tvorby bylo natočení videa. K natáčení jsem použil iPhone 15 Pro, který poskytuje vynikající kvalitu záběrů. Pro zajištění dostatečného osvětlení jsem použil lampičku. Kamera byla umístěna na vysokém stativu, směřujících dolů na dřevěnou desku, což zajistilo stabilní podklad pro natáčení. Je důležité zmínit, že mobilní telefon sloužil pouze k natočení videa, zatímco veškeré další úpravy a editace probíhaly v programu Adobe Premiere Pro.

6.8.2 Střih

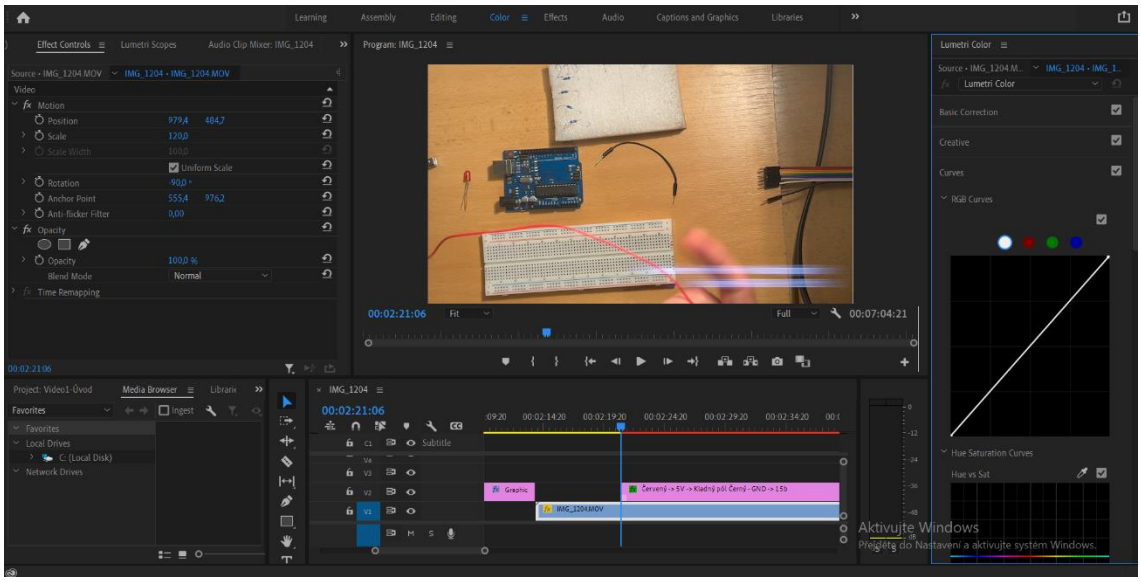
Po importování videa do Adobe Premiere Pro jsem přistoupil k procesu střihu. Využil jsem širokou škálu nástrojů a funkcí dostupných v tomto programu k ořezání, spojení, rozdělení a úpravám videa podle mých požadavků. Měl jsem k dispozici nástroje jako břitva ("Razor tool"), která mi umožnila přesně definovat místa střihu. Všechny úpravy jsem prováděl přímo v Adobe Premiere Pro, kde jsem měl plnou kontrolu nad každým aspektem svého projektu. Obrázek náhledu videa a časové osy mi poskytoval přehled o struktuře a délce jednotlivých záběrů, což mi usnadnilo práci při střihu. Občas se mi však stávalo, že se počítač zasekl kvůli náročnosti programu, zejména při práci s velkými a složitými projekty.



Obrázek 23 – Adobe Premiere PRO – Střih

6.8.3 Úprava videa

Po dokončení základního střihu jsem přešel k úpravám videa. Hlavním cílem bylo zajistit, aby uživatel měl pozitivní zážitek z prohlížení videa. Pracoval jsem na úpravě barev, aby byl obraz atraktivní a poutavý. Například jsem upravil kontrast a sytost barev, abych zvýraznil důležité prvky ve videu. Dále jsem se zaměřil na nastavení čitelných titulků, které umožní divákovi snadněji sledovat obsah a porozumět příběhu. Titulky jsem například vyvedl do kontrastní barvy, aby se lépe vyznačovaly na pozadí. Důraz byl kladen také na správné tempo a rytmus úprav, aby se divák mohl pohodlněji orientovat v průběhu sledování. Celkově jsem se snažil vytvořit atraktivní a profesionální vzhled videa, který osloví cílovou skupinu diváků.

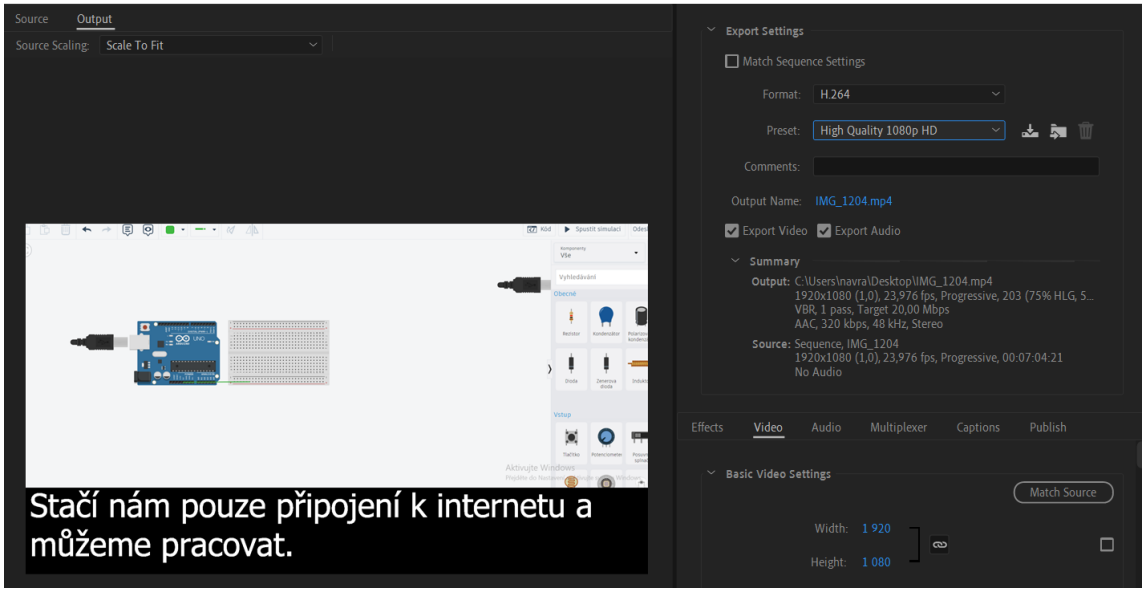


Obrázek 24 – Adobe Premiere PRO – Úprava videa

6.8.4 Výstup videa

Po dokončení veškerých úprav jsem přistoupil k výstupu finálního videa. Mým hlavním cílem bylo zajistit, aby výsledné video bylo připraveno k sdílení a mohlo být prezentováno cílovému publiku. Pro tento účel jsem využil nástroje Adobe Premiere Pro k exportu videa ve vhodném formátu a rozlišení, které odpovídalo požadavkům mého projektu. Zvolil jsem formát H.264, neboť je široce podporován na většině platform, což zaručuje kompatibilitu s různými zařízeními a webovými službami.

Při volbě Presetu jsem se rozhodl pro HIGH QUALITY 1080p HD, protože tento preset poskytuje optimální kombinaci kvality a velikosti souboru pro videa v rozlišení 1080p. S tímto nastavením jsem dosáhl vysoké kvality videa a současně jsem udržel rozumnou velikost souboru, což je důležité pro rychlé načítání a plynulé přehrávání na různých zařízeních a platformách. Díky pečlivému výběru nastavení jsem mohl zaručit, že finální video bude vypadat profesionálně a bude připraveno k publikaci na různých platformách, jako jsou sociální sítě, webové stránky nebo televizní vysílání.



Obrázek 25 – Adobe Premiere PRO – export

ZÁVĚR

V této bakalářské práci jsem se zaměřil na tvorbu a popis série videotutoriálů pro platformu Arduino UNO, které jsou určeny začínajícím a mírně pokročilým uživatelům. Tutoriály, natočené s využitím mobilního telefonu a editované v Adobe Premiere Pro, jsou strukturované tak, aby postupně představily základní i složitější koncepty zapojení a programování s Arduinem.

Hlavním přínosem této práce je vývoj uceleného vzdělávacího materiálu, který efektivně zavádí studenty do světa Arduina a elektronických projektů. Díky detailně zpracovaným videím a důkladnému popisu každého kroku, jsou uživatelé schopni rychle získat nejen teoretické znalosti, ale i praktické dovednosti v oblasti hardware a softwarového vývoje.

Klíčovým aspektem je také zpřístupnění všech lekcí na online webové stránce, která poskytuje snadný přístup k učebním materiálům, čímž se zvyšuje dostupnost a interaktivita vzdělávacího procesu. Tato platforma umožňuje studentům pohodlně navigovat mezi jednotlivými lekcemi a efektivně využívat získané informace pro své projekty.

Tato bakalářská práce nejenže poskytuje cenné zdroje pro začínající uživatele Arduina, ale také otevírá možnosti pro další výzkum a rozvoj v oblasti vzdělávacích technologií a elektronických nástrojů. Díky zpětné vazbě od uživatelů a dalšímu vývoji projektu můžeme očekávat, že dostupnost a kvalita vzdělávacích materiálů se bude nadále zlepšovat, což přispěje k rozšíření praktických znalostí a dovedností potřebných pro inovativní práci v technologických oborech.

SEZNAM POUŽITÉ LITERATURY A ZDROJŮ

[1] KULKARNI, Udayakumar. Arduino: A Begineer's Guide. [Online]. Udayakumar G.Kulkarni, 2017 [cit. 01-04-2024]. Dostupné z: https://play.google.com/books/reader?id=dkksDwAAQBAJ&pg=GBS.PA1&hl=cs

[2] SELECKÝ, Matúš. Arduino: uživatelská příručka. Brno: Computer Press, 2016. ISBN 9788025148402.

[3] NOVILLO-VICUñA, Johnny, Dixys Hernández Rojas, Bertha Mazón Olivo, Jimmy Molina Ríos, Oscar Cárdenas Villavicencio. Arduino y el internet de las cosas. 2018. ISBN 978-84-949151-8-5.

[4] JEDE ROBOT S.R.O. Arduino – příručka programátora. [Online]. Hobbyrobot.cz, 2020 [cit. 01-04-2024]. Dostupné z: http://www.hobbyrobot.cz/wp-content/uploads/ArduinoPriruckaProgramatora.pdf

[5] SOLDERED. What is fritzing, how does it work and how to use it? [online]. 2023 [cit. 01-04-2024]. Dostupné z: https://soldered.com/learn/what-is-fritzing-how-does-it-work-and-how-to-use-it/

[6] PEñA, Claudio. Arduino IDE: Domina la programación y uso. Buenos Aires: Plandos, 2020. ISBN 978-987-47579-7-5.

[7] BASTLÍRNA HW KITCHEN. Programujeme Arduino. BASTLÍRNA HW KITCHEN [online]. Zbyšek Voda, 2014 [cit. 01-04-2024]. Dostupné z: https://bastlirna.hwkitchen.cz/programujeme-arduino/

[8] All3DP. What Is Tinkercad? – Simply Explained. All3DP [online]. Andreas Giencke, 2024 [cit. 01-04-2024]. Dostupné z: https://all3dp.com/2/what-is-tinkercad-simply-explained/#i-28-user-experience

[9] AUTODESK Tinkercad. Basics of Arduino (TINKERCAD). AUTODESK Tinkercad [online]. Mukesh Sankhla, 2023 [cit. 05-04-2024]. Dostupné z: https://www.tinkercad.com/projects/Basics-of-Arduino-TINKERCAD

[10] linuxhint. Difference between Arduino IDE and TinkerCAD Simulator. Linuxhint [online]. Kashif, 2023 [cit. 05-04-2024]. Dostupné z: https://linuxhint.com/arduino-ide-tinkercad-simulator-difference/

[11] PCMAG. Adobe Premiere Pro Review. PCMAG [online]. Michael Muchmore, 2023 [cit. 05-04-2024]. Dostupné z: https://www.pcmag.com/reviews/adobe-premiere-pro

[12] Redresscompliance. ADOBE PREMIERE: THE PROS AND CONS FOR VIDEO EDITING. Redresscompliance [online]. Fredrik Filipsson, 2024 [cit. 05-04-2024] Dostupné z: https://redresscompliance.com/adobe-premiere-the-pros-and-cons-for-video-editing/

[13]Zapier. The best video editing software in 2024. Zapier [online]. Tim Brookes, 2023 [cit. 05-04-2024]. Dostupné z: https://zapier.com/blog/best-video-editing-software/

[14] DOCS Arduino.Cc. Basics of PWM (Pulse Width Modulation). DOCS Arduino.Cc [online]. Timothy Hirzel, 2022 [cit. 05-04-2024] Dostupné z: https://docs.arduino.cc/learn/microcontrollers/analog-output/

SEZNAM PŘÍKLADŮ

SEZNAM OBRÁZKŮ

A PŘÍLOHA

Web - http://home.pf.jcu.cz/~kyklop/SERYM/Ardu/videotut/index.html