**

**Vyšší odborná škola   
a Střední průmyslová škola elektrotechnická   
Plzeň, Koterovská 85**

**Autor práce: Ernst Christoph Leschka**

**Třída: IV.H**

**Dne: 05/08/2024**

Téma: Vlastní shield pro ESP

DOKUMENTACE K PROJEKTU IOT

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně a použil(a) jsem literární prameny a informace, které cituji a uvádím v seznamu použité literatury.

Souhlasím s využitím mé práce učiteli VOŠ a SPŠE Plzeň k výuce.

V Plzni dne: …..................... Podpis: …..........................

Poděkování

Děkuji Mgr. Jiřímu Hautovi za užitečnou zpětnou vazbu k dokumentaci a softwarovému řešení projektu. Také děkuji Ing. Monice Skřivanové za výbornou přípravu na konci třetího ročníku k vyhotovení projektu.

Obsah

[Úvod 6](#_Toc178454167)

[1 WeMos D1 R2 UNO ESP8266 7](#_Toc178454168)

[1.1 Využití I2C 7](#_Toc178454169)

[2 Hardware řešení 8](#_Toc178454170)

[2.1 Seznam komponent 8](#_Toc178454171)

[2.2 Prototype Shield 9](#_Toc178454172)

[2.3 LCD Displej 16x2 (I2C) 9](#_Toc178454173)

[2.3.1 Schéma zapojení a ukázkový kód I2C 1602 LCD Displeje 10](#_Toc178454174)

[2.4 DS18B20 Digitální čidlo teploty 11](#_Toc178454175)

[2.4.1 Schéma zapojení a ukázkový kód DS18B20 11](#_Toc178454176)

[2.5 RGB LED NeoPixel pásek 13](#_Toc178454177)

[2.5.1 Schéma zapojení a ukázkový kód NeoPixel RGB pásku 13](#_Toc178454178)

[2.6 KY-040 Rotační enkodér 15](#_Toc178454179)

[2.6.1 Schéma zapojení a ukázkový kód enkodéru 15](#_Toc178454180)

[2.7 Finální hardware řešení shieldu 17](#_Toc178454181)

[2.7.1 Vysvětlení finálního HW řešení 18](#_Toc178454182)

[3 Software řešení 19](#_Toc178454183)

[3.1 Vlastní knihovna „LeschkaShield“ 19](#_Toc178454184)

[3.1.1 Funkce „void setStatusLEDs(bool led1State, bool led2State)“ 21](#_Toc178454185)

[3.1.2 Funkce „int readEncoder()“ 21](#_Toc178454186)

[3.1.3 Funkce „void clearLEDStrip()“ 21](#_Toc178454187)

[3.1.4 Funkce „void setLEDsForLight(int value)“ 22](#_Toc178454188)

[3.1.5 Funkce „void setLEDsForTemperature(float temp)“ 22](#_Toc178454189)

[3.1.6 Funkce „float readTemperature()“ 23](#_Toc178454190)

[3.1.7 Funkce „int readLDR()“ 23](#_Toc178454191)

[3.1.8 Funkce „bool readButton()“ 24](#_Toc178454192)

[3.2 Hlavní program 25](#_Toc178454193)

[3.2.1 Funkce „void buttonPress(bool& inMenu, int& menuIndex)“ 25](#_Toc178454194)

[3.2.2 Funkce „void displayMenu(int menuIndex)“ 25](#_Toc178454195)

[3.2.3 Funkce „void menuSelection(int menuIndex)“ 26](#_Toc178454196)

[3.2.4 Funkce „void measureLight()“ 26](#_Toc178454197)

[3.2.5 Funkce „void measureTemp()“ 27](#_Toc178454198)

[3.2.6 Funkce „void setup()“ 27](#_Toc178454199)

[3.2.7 Funkce „void loop()“ 28](#_Toc178454200)

[4 Finální podoba řešení projektu vlastní shield 29](#_Toc178454201)

[Závěr 30](#_Toc178454202)

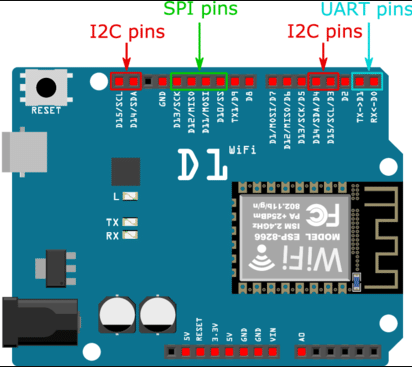
[Seznam použité literatury 31](#_Toc178454203)

[Seznam obrázků, tabulek a grafů 33](#_Toc178454204)

# Úvod

Dokumentace se týká custom shieldu na ESP8266, který měří teplotu a taktéž měří svítivost pomocí LDR (Light Dependent Resistor – čili fotorezistor). Cílem této dokumentace je provést čtenáře funkčností jednotlivých komponent a následně řešením, funkcemi a fungováním shieldu, aby tedy bylo jasno co se děje a jak jej použít.

# WeMos D1 R2 UNO ESP8266

Shield je zpracován na mikrokontroler (dále jen MCU) ESP8266, v mém případě specificky na model WeMos D1 R2 UNO ESP8266. Tato MCU mimo jiné na rozdíl například od Arduina umožňuje připojení k Wi-Fi. Jistě si každý povšimne, že na rozdíl od Arduina má méně využitelných pinů, k vyřešení problému nedostatku pinů lze využít expander jako je například PCF8575, který vám pomocí I2C přidá o 16 pinů navíc.

Obrázek 1: Zobrazení ESP8266 v software Fritzing s popsanými piny [1]

## Využití I2C

Neodlučitelnou součástí řešení mého shield je I2C, což je sériový komunikační protokol, který využívá seriál data line (SDA) a seriál clock line (SCL) [2] – viz. Obrázek 1. Nedílná součást shieldu je 16x02 LCD Displej, který komunikuje právě přes I2C. Měl jsem v plánu se taktéž pokusit o vykreslení změny hodnot v čase pomocí grafu na 128x32 OLED displeji, který taktéž komunikuje přes I2C, nicméně se mi bohužel nepodařilo jej zprovoznit. Podobně jsem měl v plánu využít 16bitový expandér pinů PCF8575, jenž taky komunikuje přes I2C, ten jsem ale nakonec nepotřeboval, tak jsem ho z mého řešení eliminoval.

# Hardware řešení

Všechny komponenty jsou na prototypovém shieldu, jelikož si nedokážu představit, jak to zhotovit jinak. Na shieldu jsou přímo připájené LEDky pro zobrazení stavu, enkodér, tlačítko a fotorezistor. Přímo mimo shield leží ještě tepelné čidlo, LCD displej a LED pásky. V podkapitolách bude popis unikátních komponent.

## Seznam komponent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Komponenta** | **Množství** | **Cena/kus** | **Link** | **Cena** |
| Arduino UNO prototype Shield | 1 | 54 Kč | [laskakit](https://www.laskakit.cz/arduino-uno-prototype-shield-mini-breadboard/) | 54 Kč |
| 16x2 LCD displej 1602 modrý + I2C převodník | 1 | 98 Kč | [laskakit](https://www.laskakit.cz/16x2-lcd-displej-1602-i2c-prevodnik/) | 98 Kč |
| DS18B20 Digitální vodotěsné čidlo teploty | 1 | 48 Kč | [laskakit](https://www.laskakit.cz/dallas-digitalni-vodotesne-cidlo-teploty-ds18b20-1m/) | 48 Kč |
| 8x inteligentní RGB LED NeoPixel pásek, WS2812B | 2 | 28 Kč | [laskakit](https://www.laskakit.cz/8x-inteligentni-rgb-led-neopixel-pasek--ws2812b--5050--5v/) | 56 Kč |
| Fotorezistor 5 mm GL5516 | 1 | 3 Kč | [laskakit](https://www.laskakit.cz/fotorezistor-5mm-gl5516/) | 3 Kč |
| Keyes KY-040 Rotační encoder | 1 | 34 Kč | [laskakit](https://www.laskakit.cz/keyes-ky-040-rotacni-encoder-s-tlacitkem/) | 34 Kč |
| Rezistor 10kΩ | 3 | 1 Kč | [laskakit](https://www.laskakit.cz/futaba-1-4w-odpor-10k-5/) | 3 Kč |
| Rezistor 220Ω | 2 | 1 Kč | [laskakit](https://www.laskakit.cz/1-4w-odpor-220r-5/) | 2 Kč |
| Tlačítko | 1 | 2 Kč | [laskakit](https://www.laskakit.cz/tlacitko-6x6x5mm/) | 2 Kč |
| LED 5 mm | 2 | 1 Kč | [laskakit](https://www.laskakit.cz/led-dioda-5mm/?variantId=1130) | 2 Kč |
| WeMos D1 R2 UNO ESP8266 [[1]](#footnote-1) | 1 | 178 Kč | [laskakit](https://www.laskakit.cz/wemos-d1-r2-uno-esp8266/) | 178 Kč |
| ~~PCF8575 I2C 16 bit I/O Expander~~ [[2]](#footnote-2) | 1 | 118 Kč | [laskakit](https://www.laskakit.cz/pcf8575-i2c-16-bit-i-o-expander/) | 118 Kč |
| ~~0.91" 128x32 OLED displej~~ | 1 | 88 Kč | [laskakit](https://www.laskakit.cz/oled-displej-modry-128x32-0-91--i2c/) | 88 Kč |
|  |  |  |  | 686 Kč [[3]](#footnote-3) |

## Prototype Shield

Prototype shield je vlastně základ celého shieldu, zjednodušuje konstrukci a po připojení součástek je, lze říci, „Plug-and-Play“. Na ploše shieldu jsou vyhrazená místa pro připojení napájení (5 V) a uzemnění (GND). Taky zde najdeme už připravené tlačítko na reset.

Obsah obrázku Obvodoví součástka, Elektronická součástka, elektronika, Počítačová komponenta

Popis byl vytvořen automaticky

Obrázek 2: Arduino UNO prototype Shield [3]

## LCD Displej 16x2 (I2C)

Tento LCD displej má, jak lze poznat podle názvu, rozlišení 16x2, tedy 16 sloupců na dva řádky. Tento displej se primárně používá pro zobrazení znaků či nějakého krátkého textu. V mém případě je to pro zobrazení menu, kde jsou dvě možnosti a poté pro zobrazení hodnoty toho, co právě měříme. Zezadu je potenciometr k úpravě kontrastu.

Obsah obrázku elektronika, Elektronická součástka, Počítačová komponenta, Obvodoví součástka

Popis byl vytvořen automaticky

Obrázek 3: 16x2 LCD displej 1602 modrý [4]

### Schéma zapojení a ukázkový kód I2C 1602 LCD Displeje

V případě I2C verze tohoto displeje je zapojení triviální – uzemnění do GND, napájení VCC do 5 V a následně SDA a SCL do I2C pinů na desce. Na obrázku níže je sice použito Arduino, ale v našem případě zapojení vypadá stejně.

Obsah obrázku elektronika, Elektronické inženýrství, Elektronická součástka, Obvodoví součástka

Popis byl vytvořen automaticky

Obrázek 4: I2C zapojení 1602 LCD k Arduinu [5]

Abychom mohli opravdu použít LCD displej musíme použít knihovnu „LiquidCrystal\_I2C“, která zahrnuje mnoho funkcí na jeho ovládání. Na obrázku níže vidíme kód, který na první řádce displeje zobrazí text „Hello World!“ a na druhé řádce „LCD Tutorial“. [6]

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek 5: Ukázka kódu k LCD [6]

## DS18B20 Digitální čidlo teploty

Jedna ze dvou měřených jednotek na shieldu je teplota, tu budeme měřit za pomocí digitálního vodotěsného čidla teploty DS18B20. Čidlo využívá 1wire protokol, který je v principu podobný I2C, ale s nižší rychlostí přenosu dat a delší funkční vzdáleností. [7]

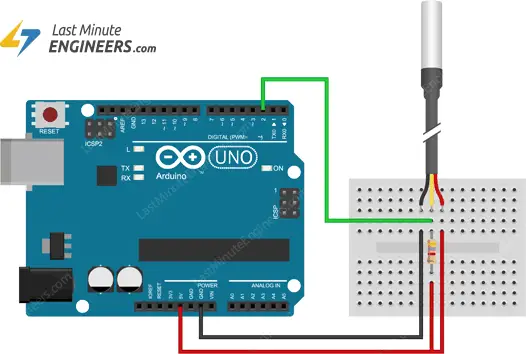
Obsah obrázku kabel

Popis byl vytvořen automaticky se střední mírou spolehlivosti

Obrázek 6: čidlo teploty DS18B20 [8]

### Schéma zapojení a ukázkový kód DS18B20

Zapojení je intuitivní, uzemnění do GND, napájení VDD do 5 V a datový kabel do digitálního pinu na desce (v případě obrázku D2). Aby přenos dat byl stabilní, přidáme ještě 10kΩ pull-up rezistor mezi datový kabel a napájení.



Obrázek 7: Zapojení čidla teploty k Arduinu [9]

Abychom mohli pomocí čidla měřit, tak budeme potřebovat dvě knihovny – tj. „OneWire“ a „DallasTemperature“. Knihovna OneWire přidává podporu pro 1wire protokol (podobně jako knihovna Wire pro I2C) a knihovna DallasTemperature přidává funkce k ovládání tepelného čidla.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 8: Ukázka kódu k čidlu teploty [10]

## RGB LED NeoPixel pásek

Tento LED pásek je tvořen LED diodami WS2812B zapojenými do série. Tyto LED diody mají vestavěný čip s pamětí 3 bajty. Těmi se nastavuje intenzita svitu každé ze tří barev. To umožňuje komunikaci pomocí jednokanálového rozhraní. [11] Pro zobrazení „velikosti“ naměřených hodnotu využijme RGB LED pásku. Při měření světla tedy svítí 0-8 „pixelů“ na pásce podle toho, jakou hodnotu od 0-1024 fotorezistor naměří. Stejně tak u teploty svítí 0-8 „pixelů“ podle toho, jakou hodnotu od -10°C do 70°C naměří tepelné čidlo.



Obrázek 9: RGB LED NeoPixel pásek [11]

### Schéma zapojení a ukázkový kód NeoPixel RGB pásku

Opětně triviální zapojení – uzemnění do GND, napájení VCC do 5V a DIN do digital pinu na desce. Chceme-li zapojit více pásek sériově, tak z druhé strany první pásky zapojíme DOUT do druhé pásky DIN.

Obsah obrázku text, diagram, Paralelní, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek 10: Zapojení WS2812 LED Pásky [12]

Nyní k ukázce kódu – tento program vytvoří efekt pomalého „nalití“ barvy do pásku dlouhého 1 m s 30 LED/m, bude se cyklicky střídat sedm barev. Po rozsvícení celý pásek plynule pohasne [13]

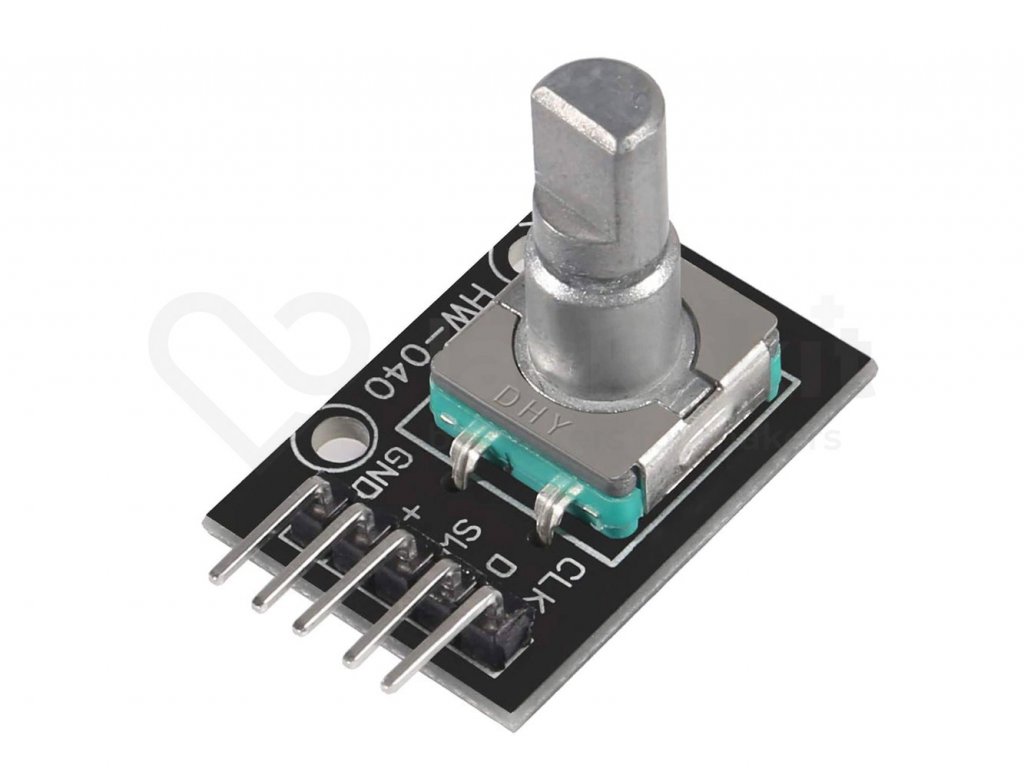
Obsah obrázku text, snímek obrazovky, číslo, Písmo

Popis byl vytvořen automaticky

Obrázek 11: Ukázka kódu k LED pásce [13]

## KY-040 Rotační enkodér

Otočný enkodér Keyes KY-040 je rotační vstupní zařízení, které ukazuje, jak moc byl knoflík otočen a jakým směrem se otáčí. Je to skvělá komponenta pro řízení krokových a servomotorů. [14] V našem případě ho využijeme k výběru měřené jednotky v menu.



Obrázek 12: KY-040 Rotační enkodér [15]

### Schéma zapojení a ukázkový kód enkodéru

Opět nic složitého, uzemnění do GND, napájení do 5V a poté CLK a DT do digital pinů na desce. SW nevyužijeme.

Obsah obrázku text, Obdélník, snímek obrazovky, design

Popis byl vytvořen automaticky

Obrázek 13: Zapojení enkodéru KY-040 do Arduina [14]

Kód testuje klíčové funkce enkodéru, tedy pozici, ve které aktuálně je a možnost zjistit, zda poslední rotace byla ve směru hodinových ručiček či v proti směru. V řešení shieldu takto „komplikovaný“ kód ale nebyl potřeba, jelikož nám stačí dvě hodnoty, tedy např. 1 a 0.

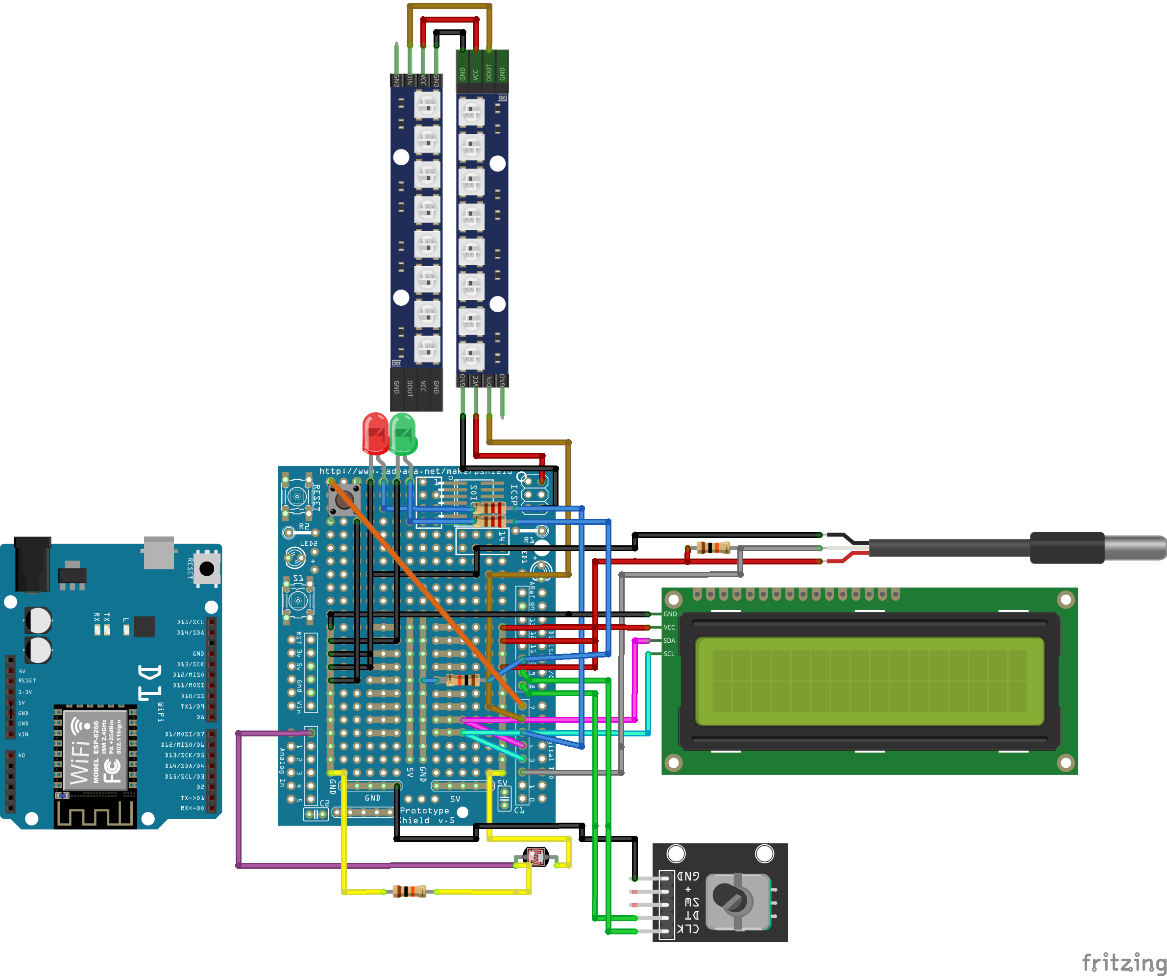
Obsah obrázku text, snímek obrazovky, dokument

Popis byl vytvořen automaticky

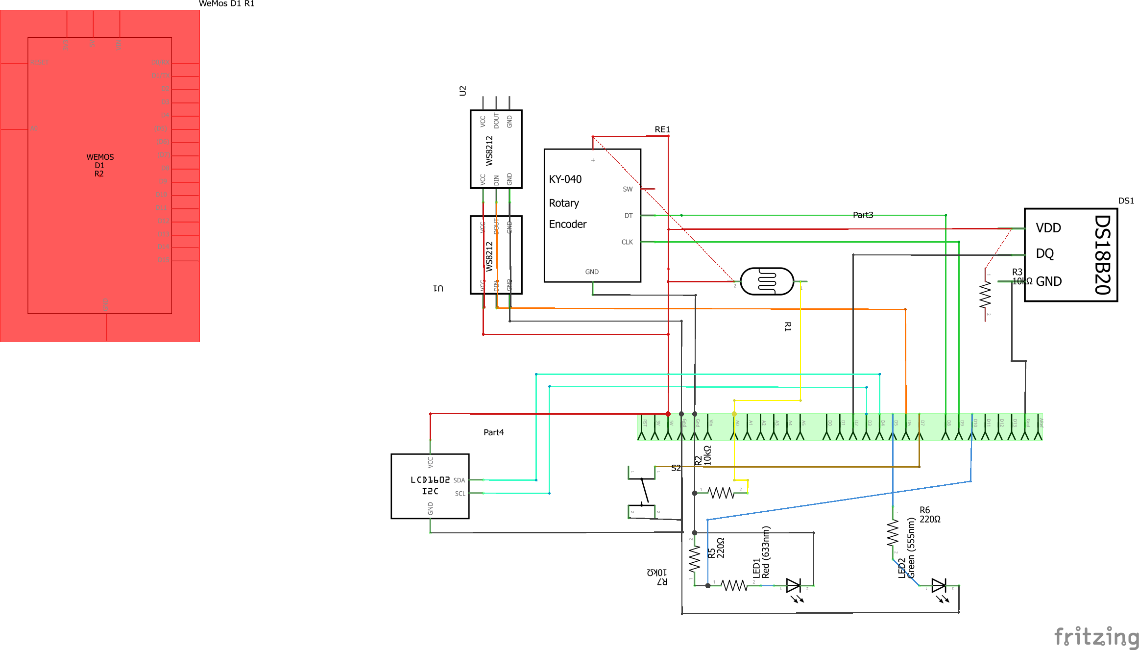
Obrázek 14: Ukázka kódu enkodéru KY-040 [14]

## Finální hardware řešení shieldu

Poskládáme-li všechny zmíněné komponenty dohromady, vznikne tato finální verze zapojení.



Obrázek 15: Zapojení shieldu ve Fritzingu [16]



Obrázek 16: Schéma zapojení shieldu ve Fritzingu [16]

### Vysvětlení finálního HW řešení

|  |  |
| --- | --- |
| **Komponenta** | **Pin** |
| Červená status LED | D8 |
| Zelená status LED | D3 |
| Enkodér pin 1 | D6 |
| Enkodér pin 2 | D7 |
| LED Pásek | D4 |
| Protokol OneWire  (datový pin tepelného čidla) | D0 |
| Tlačítko | D5 |
| Fotorezistor | A0 |

* Vše spojeno na prototype shieldu, který se zapojí do MCU.
* Dvě LED pásky o 8 LED „pixelech“ spojené v sérii, obě tedy dávají součet 16 LED „pixelů“.
* 10kΩ rezistor u LDR.
* U tlačítka rezistor nevyužijeme, jelikož použijeme zabudovaný pull-up rezistor na desce.
* U červené status LED využijeme navíc 10kΩ rezistor, jelikož je zapojena do digitálního pinu 8, který má speciální vlastnosti.
* Jak jsme již zmínili v kapitole o tepelném čidle, využijeme 10kΩ rezistor mezi datovým pinem a napájením.
* LCD displej je na I2C sběrnici, D2 = SDA; D1 = SCL

# Software řešení

Jelikož v zadání bylo nutností, aby kód byl modulární, tak jsem můj kód modularizoval, jak jen to šlo. Nejprve jsem využil možnosti vytvořit si vlastní knihovnu, ve které budou funkce pro každou komponentu. A v hlavním programu jsem kód modularizoval do funkcí.

## Vlastní knihovna „LeschkaShield“

Nejprve začněme vytvořením header souboru, ve kterém budou definované využité potřebné knihovny pro komponenty, konstanty (piny, počet LED na pásce) a objekty (LCD, LED Pásek, tepelné čidlo) včetně deklarace funkcí, které využijeme v C++ souboru knihovny.

Obsah obrázku text, snímek obrazovky, Písmo, dokument

Popis byl vytvořen automaticky

Obrázek 17: Kód v header souboru "LeschkaShield.h" [16]

V C++ souboru nejprve přiřadíme hodnoty objektům inicializovaným v header souboru.

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek 18: Objekty v LeschkaShield.cpp [16]

Poté vytvoříme všechny funkce, které chceme, aby knihovna zahrnovala. V mém případě jsem vytvořil nejprve setup funkci ke každé komponentě, tj. „setupStatusLEDs()“, „setupLCD()“, „setupLEDStrip()“, „setupTemperatureSensor()“ a „setupButton()“, tyto funkce v hlavním programu vložíme do „setup()“ a máme to z krku.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 19: setup funkce v LeschkaShield.cpp [16]

A následně jsem vytvořil již funkce, které „něco dělají“, tj. „setStatusLEDs()“, „readEncoder()“, „clearLEDStrip()“, „setLEDsForLight()“, „setLEDsForTemperature()“, „readTemperature()“, „readButton()“ a „readLDR()“. Tyto funkce si probereme detailněji.

### Funkce „void setStatusLEDs(bool led1State, bool led2State)“

Funkce bere dva bool parametry, tj. led1State a led2State, které určí, zda má LED svítit či nikoliv.

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Popis byl vytvořen automaticky

Obrázek 20: Funkce setStatusLEDs [16]

### Funkce „int readEncoder()“

Funkce int readEncoder() vrací z pinu enkodéru buď 1 nebo 0, jelikož máme v menu jen dvě možnosti, tak jiné hodnoty nepotřebujeme.

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Popis byl vytvořen automaticky

Obrázek 21: Funkce readEncoder [16]

### Funkce „void clearLEDStrip()“

Funkce, jak lze poznat podle názvu, jenom „vyčistí“, čili vypne LED pásek pomocí funkcí z knihovny „Adafruit\_NeoPixel.h“.

Obsah obrázku text, Písmo, design

Popis byl vytvořen automaticky

Obrázek 22: Funkce clearLEDStrip [16]

### Funkce „void setLEDsForLight(int value)“

Funkce rozsvítí x LED na pásku podle toho, jakou hodnotu naměří LDR (parametr value). Namapuje tedy hodnotu z LDR mezi 0–8, nastaví barvu a poté pomocí cyklu for na obou páscích zároveň rozsvítí 0-8 LED podle naměřené hodnoty.

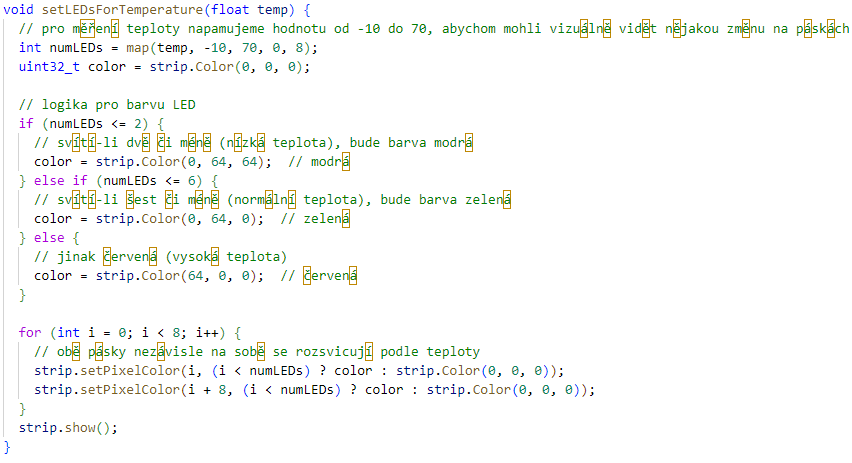
Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Popis byl vytvořen automaticky

Obrázek 23: Funkce setLEDsForLight [16]

### Funkce „void setLEDsForTemperature(float temp)“

Funkce rozsvítí x LED na pásku podle toho, jakou hodnotu naměří tepelné čidlo (parametr temp). Namapuje tedy hodnotu z čidla (-10 °C–80 °C) mezi 0–8, barvu nastaví podle počtu zapnutých LED – modrou, svítí-li dvě či méně, zelenou svítí-li šest či méně a červenou, svítí-li sedm či více. Stejně jako u předchozí funkce pomocí cyklu for na obou páscích zároveň rozsvěcuje 0–8 LED podle naměřené hodnoty.



Obrázek 24: Funkce setLEDsForTemperature [16]

### Funkce „float readTemperature()“

Pomocí funkcí z knihovny „DallasTemperature.h“ vrátí desetinnou hodnotu teploty.

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Popis byl vytvořen automaticky

Obrázek 25: Funkce readTemperature [16]

### Funkce „int readLDR()“

Funkce vrací hodnotu načtenou z pinu LDR.

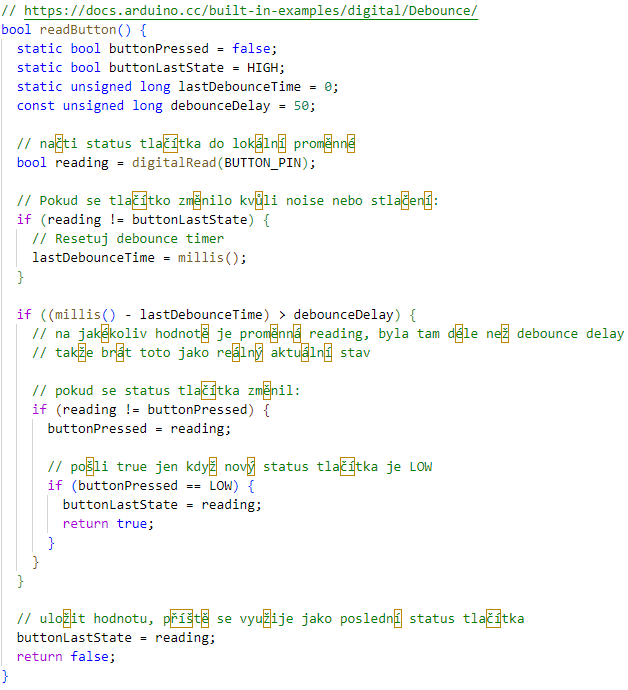
Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Popis byl vytvořen automaticky

Obrázek 26: Funkce readLDR [16]

### Funkce „bool readButton()“

Tato funkce zajišťuje správnou funkčnost tlačítka.



Obrázek 27: Funkce readButton [17]

## Hlavní program

Jelikož jsme si vytvořením knihovny zjednodušili hlavní sketch, tak nám stačí importovat knihovnu, vytvořit pár QoL funkcí (modularizace), zajistit logiku v setup a loop funkcích a nahrát.

### Funkce „void buttonPress(bool& inMenu, int& menuIndex)“

Funkce mění stav proměnné inMenu, zajišťuje správné zobrazení na LCD displeji a vypíná LED pásek, pokud jsme v menu.

Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Popis byl vytvořen automaticky

Obrázek 28: Funkce buttonPress [16]

### Funkce „void displayMenu(int menuIndex)“

Funkce čte hodnotu z enkodéru, dle které vizuálně na displeji ukáže, zda je právě zvolena možnost měření světla či teploty. Taktéž aktualizuje LED statusu.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 29: Funkce displayMenu [16]

### Funkce „void menuSelection(int menuIndex)“

Funkce zobrazí na monitoru právě měřenou veličinu podle hodnoty z enkodéru, taky aktualizuje LED statusu.

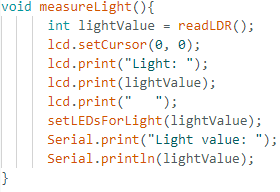
Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 30: Funkce menuSelection [16]

### Funkce „void measureLight()“

Funkce načte hodnotu z LDR, zobrazí jej na LCD a rozsvítí adekvátní počet LED na pásce.



Obrázek 31: Funkce measureLight [16]

### Funkce „void measureTemp()“

Funkce načte hodnotu z tepelného čidla, zobrazí jej na LCD a rozsvítí adekvátní počet LED na pásce.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 32: Funkce measureTemp [16]

### Funkce „void setup()“

Konečně se dostáváme k hlavním funkcím hlavního programu, v setup jen zavoláme setup funkce z naší knihovny.

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek 33: Funkce setup [16]

### Funkce „void loop()“

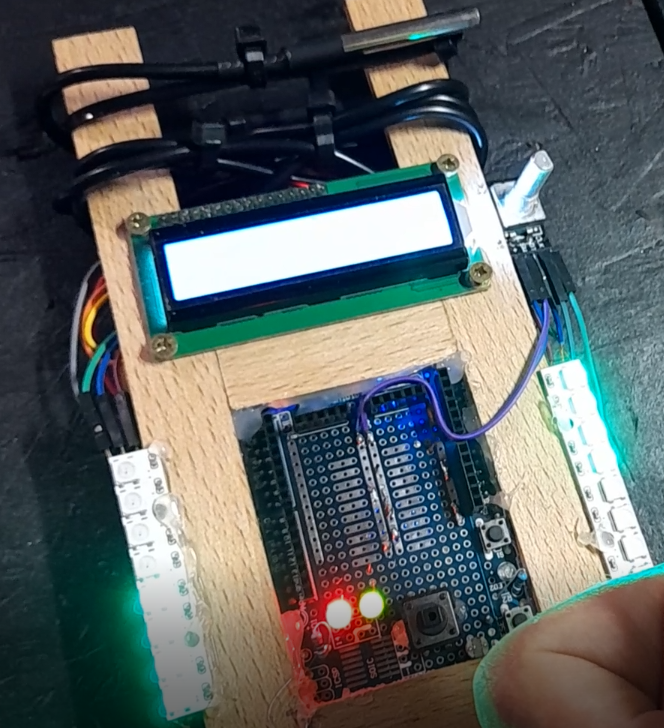
A nakonec, jelikož jsme si kód modularizovali i v hlavním programu, tak je loop velice zjednodušený.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 34: Funkce loop [16]

# Finální podoba řešení projektu vlastní shield

Díky správnému návrhu schéma zapojení a modularizaci kódu skrz knihovnu a funkce vše fungovalo tak, jak má.

Obrázek 35: Finální řešení [16]

# Závěr

Tento projekt mě naučil mnohého – mezi hlavní poznatky patří primárně vytvoření knihovny a práce s I2C (přestože jsem I2C již využil v projektu „Model Dopravní Situace“, tak jsem se naučil podrobněji, jak funguje). Samozřejmě nemohu opominout ten fakt, že jsem hlouběji porozuměl komponentám a hlavně ESP8266 (neměl jsem možnost ve škole, jelikož jsem byl na stáži v Německu). Naopak se mi bohužel nepovedlo zprovoznit OLED displej, který měl na sobě míti graf.

# Seznam použité literatury

[1] - DEBAJYOTI, Biswas. *A WiFi-based Self-Organizing Multi-Hop Sensor Network for Internet of Things*. Online. In: Research Gate. 2021. Dostupné z: <https://www.researchgate.net/figure/The-UART-I2C-and-SPI-pins-of-a-wemos-D1-R2-ESP8266_fig3_350940118>. [cit. 2024-08-09].

[2] - TEXAS INSTRUMENTS. *A Basic Guide to I2C*. Online. 2022. Dostupné z: <https://www.ti.com/lit/an/sbaa565/sbaa565.pdf?ts=1723186653015&ref_url=https%253A%252F%252Fwww.google.com%252F#:~:text=I2C%20is%20a%20two%2Dwire,and%20receive%20commands%20and%20data.>. [cit. 2024-08-09].

[3] - *Arduino UNO prototype Shield + mini BreadBoard.* Online. C2024. Dostupné z: <https://www.laskakit.cz/arduino-uno-prototype-shield-mini-breadboard/>. [cit. 2024-08-10].

[4] - *16x2 LCD displej 1602 modrý*. Online. C2024. Dostupné z: <https://www.laskakit.cz/16x2-lcd-displej-1602-i2c-prevodnik/>. [cit. 2024-08-10].

[5] - *Wiring an I2C LCD Display to an Arduino*. Online. C2024. Dostupné z: <https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/>. [cit. 2024-08-10].

[6] - *Basic Arduino Sketch – Hello World*. Online. C2024. Dostupné z: <https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/>. [cit. 2024-08-10].

[7] - *1-Wire*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2001-. Dostupné z: <https://en.wikipedia.org/wiki/1-Wire>. [cit. 2024-08-10].

[8] - *DS18B20 Digitální vodotěsné čidlo teploty 1m*. Online. C2024. Dostupné z: <https://www.laskakit.cz/dallas-digitalni-vodotesne-cidlo-teploty-ds18b20-1m/#relatedFiles>. [cit. 2024-08-10].

[9] - *Wiring a DS18B20 Temperature Sensor to an Arduino*. Online. C2024. Dostupné z: <https://lastminuteengineers.com/ds18b20-arduino-tutorial/>. [cit. 2024-08-10].

[10] - *Arduino Example Code*. Online. C2024. Dostupné z: <https://lastminuteengineers.com/ds18b20-arduino-tutorial/>. [cit. 2024-08-10].

[11] - *8x inteligentní RGB LED NeoPixel pásek, WS2812B, 5050, 5V*. Online. C2024. Dostupné z: <https://www.laskakit.cz/8x-inteligentni-rgb-led-neopixel-pasek--ws2812b--5050--5v/#ratingTab>. [cit. 2024-08-10].

[12] - *Wiring WS2812 8-Bit RGB LED Strip with Arduino*. Online. C2024. Dostupné z: <https://www.14core.com/wiring-ws2812-8-bit-rgb-led-strip-with-arduino/>. [cit. 2024-08-10].

[13] - *LED NeoPixel – jak na ně s Arduinem*. Online. C2024. Dostupné z: <https://robodoupe.cz/2015/led-neopixel-jak-na-ne-s-arduinem/>. [cit. 2024-08-10].

[14] - *KY-040 Arduino Rotary Encoder User Manual*. Online. C2024. Dostupné z: <https://www.epitran.it/ebayDrive/datasheet/25.pdf>. [cit. 2024-08-11].

[15] - *Keyes KY-040 Rotační encoder s tlačítkem bez závitu*. Online. C2024. Dostupné z: <https://www.laskakit.cz/keyes-ky-040-rotacni-encoder-s-tlacitkem-bez-zavitu/>. [cit. 2024-08-11].

[16] – Vlastní zdroj

[17] - *Debounce on a Pushbutton*. Online. 2015, 09/02/2022. Dostupné z: <https://docs.arduino.cc/built-in-examples/digital/Debounce/>. [cit. 2024-08-10].

# Seznam obrázků, tabulek a grafů

[Obrázek 1: Zobrazení ESP8266 v software Fritzing s popsanými piny [1] 7](#_Toc178533525)

[Obrázek 2: Arduino UNO prototype Shield [3] 9](#_Toc178533526)

[Obrázek 3: 16x2 LCD displej 1602 modrý [4] 9](#_Toc178533527)

[Obrázek 4: I2C zapojení 1602 LCD k Arduinu [5] 10](#_Toc178533528)

[Obrázek 5: Ukázka kódu k LCD [6] 10](#_Toc178533529)

[Obrázek 6: čidlo teploty DS18B20 [8] 11](#_Toc178533530)

[Obrázek 7: Zapojení čidla teploty k Arduinu [9] 11](#_Toc178533531)

[Obrázek 8: Ukázka kódu k čidlu teploty [10] 12](#_Toc178533532)

[Obrázek 9: RGB LED NeoPixel pásek [11] 13](#_Toc178533533)

[Obrázek 10: Zapojení WS2812 LED Pásky [12] 14](#_Toc178533534)

[Obrázek 11: Ukázka kódu k LED pásce [13] 14](#_Toc178533535)

[Obrázek 12: KY-040 Rotační enkodér [15] 15](#_Toc178533536)

[Obrázek 13: Zapojení enkodéru KY-040 do Arduina [14] 15](#_Toc178533537)

[Obrázek 14: Ukázka kódu enkodéru KY-040 [14] 16](#_Toc178533538)

[Obrázek 15: Zapojení shieldu ve Fritzingu [16] 17](#_Toc178533539)

[Obrázek 16: Schéma zapojení shieldu ve Fritzingu [16] 17](#_Toc178533540)

[Obrázek 17: Kód v header souboru "LeschkaShield.h" [16] 19](#_Toc178533541)

[Obrázek 18: Objekty v LeschkaShield.cpp [16] 20](#_Toc178533542)

[Obrázek 19: setup funkce v LeschkaShield.cpp [16] 20](#_Toc178533543)

[Obrázek 20: Funkce setStatusLEDs [16] 21](#_Toc178533544)

[Obrázek 21: Funkce readEncoder [16] 21](#_Toc178533545)

[Obrázek 22: Funkce clearLEDStrip [16] 21](#_Toc178533546)

[Obrázek 23: Funkce setLEDsForLight [16] 22](#_Toc178533547)

[Obrázek 24: Funkce setLEDsForTemperature [16] 22](#_Toc178533548)

[Obrázek 25: Funkce readTemperature [16] 23](#_Toc178533549)

[Obrázek 26: Funkce readLDR [16] 23](#_Toc178533550)

[Obrázek 27: Funkce readButton [17] 24](#_Toc178533551)

[Obrázek 28: Funkce buttonPress [16] 25](#_Toc178533552)

[Obrázek 29: Funkce displayMenu [16] 25](#_Toc178533553)

[Obrázek 30: Funkce menuSelection [16] 26](#_Toc178533554)

[Obrázek 31: Funkce measureLight [16] 26](#_Toc178533555)

[Obrázek 32: Funkce measureTemp [16] 27](#_Toc178533556)

[Obrázek 33: Funkce setup [16] 27](#_Toc178533557)

[Obrázek 34: Funkce loop [16] 28](#_Toc178533558)

[Obrázek 35: Finální řešení [16] 29](#_Toc178533559)

1. MCU není nedílnou součástí shieldu, ale shield je na tuto MCU, tudíž dává smysl ho dát do ceny [↑](#footnote-ref-1)
2. Přeškrtnutý text = v konečném řešení součástka není, avšak byla v plánu a je zmíněna v textu [↑](#footnote-ref-2)
3. Cena včetně komponent, které buď nejsou nedílnou součástí shieldu nebo nejsou součástí vůbec [↑](#footnote-ref-3)