Středoškolská technika 2018

Setkání a prezentace prací středoškolských studentů na ČVUT

Víceúčelový LED display

Karel Čtvrtečka

Střední průmyslová škola elektrotechnická a Vyšší odborná škola Pardubice

Karla IV. 13, 530 02 Pardubice

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 18.4.2018 ……

Poděkování

Tímto bych rád poděkoval svému třídnímu učiteli, Ing. Miroslavu Kouckému, za jeho připomínky a užitečné rady

Anotace

Práce se zabývá vývojem, stavbou a programováním víceúčelového LED displeje, který primárně slouží jako ukazatel času a dalších hodnot, které sbírají připojitelná čidla. Nastavení displeje lze měnit pomocí externího terminálu.

Klíčová slova:

Display, LED, senzory, hodiny, Arduino

Annotation

The work deals with development, constructing and programming multipurpose LED display which is primarily used for displaying time and other values. Data will be captured by external sensor modules. Display settings will be changeable via external terminal.

Keywords:

Display, LED, sensors, time, Arduino

Obsah

1 SEZNAM POUŽITÝCH ZNAČEK A SYMBOLŮ 8

2 ÚVOD 9

3 HARDWARE 10

3.1 DISPLAY 10

3.2 LED PÁSEK 11

3.3 MŘÍŽKA 12

3.4 NAPÁJENÍ 13

3.4.1 Zdroj 13

3.4.2 DC-DC Měnič 13

3.4.3 Řídící elektronika zdroje 14

3.5 ŘÍDÍCÍ ELEKTRONIKA 15

3.5.1 Arduino MEGA 15

3.5.2 RTC Modul 16

3.6 ČIDLA 17

3.6.1 DHT11 17

3.7 OVLÁDACÍ TERMINÁL 18

3.7.1 Arduino Pro Mini 18

3.7.2 LCD Display 19

3.7.3 Rotační enkodér 20

3.7.4 Krabička 21

3.7.5 Připojení 21

3.8 KONEKTORY 22

3.8.1 Napájecí 22

3.8.2 Konektor terminálu 22

3.8.3 Konektor čidla 23

3.8.4 Přídavný konektor 23

4 FIRMWARE 24

4.1 HLAVNÍ ČIP 24

4.1.1 Použité knihovny 24

4.1.2 Struktura programu 25

4.2 TERMINÁL 30
4.2.1 Použité knihovny 30
4.2.2 Struktura programu 31
4.3 ELEKTRONIKA ZDROJE 34
4.3.1 Použité knihovny 34
4.3.2 Struktura programu 34
5 ZÁVĚR 36
6 ZDROJE 37
7 SEZNAM OBRÁZKŮ 38
8 PŘÍLOHY 41

1 Seznam použitých značek a symbolů

LED polovodičová dioda vyzařující světlo

DPS deska plošných spojů, slouží k propojení jednotlivých součástek obvodu

I^2C sériová synchronní sběrnice

1-Wire Sériová asynchronní sběrnice po jednom vodiči

RTC modul reálného času

IDE vývojové prostředí

NTP síťový protokol pro synchronizaci času

2 Úvod

Na nápad vytvořit display z LED pásku jsem přišel, když jsem se teprve učil programovat Arduino a přišly mi do ruky tyto řiditelné LED pásky. Díky tomu, že umožnují rozsvítit jakýkoliv pixel po celé své délce jakoukoli barvou, jsou pro takové využití dokonalé. Při použití mřížky na oddělení jednotlivých pixelů v kombinaci s bílým sklem, které rozprostírá světlo, působí display velmi moderně. K realizaci tohoto projektu jsem se dostal až na střední škole, když jsme vymýšleli, jaké hodiny si pořídíme do třídy.

Primární funkcí je zobrazování času. Dále je displej vybaven konektory pro připojení dalších periferií. Díky dostatečné hardwarové výbavě je displej připravený na rozšíření. Základní firmware je pro středně pokročilého programátora lehké upravit a možnost přidat nové vlastnosti tak záleží už pouze na fantazii.

Využití najde tento display jako moderní digitální nástěnné hodiny doplněné o meteostanici. Při přidání Wi-Fi nebo ethernet modulu se může display připojit do internetu věcí a zobrazovat z něj data, jako například aktuální stavy na burze a mnoho dalších. Další využití by mohl najít ve zdravotnictví – pomáhat lidem se zrakovými problémy. Velké využití najde také ve školách. Může studentům ukazovat čas do zvonění nebo konce přestávky či zobrazovat aktuální číslo hodiny.

3 HARDWARE

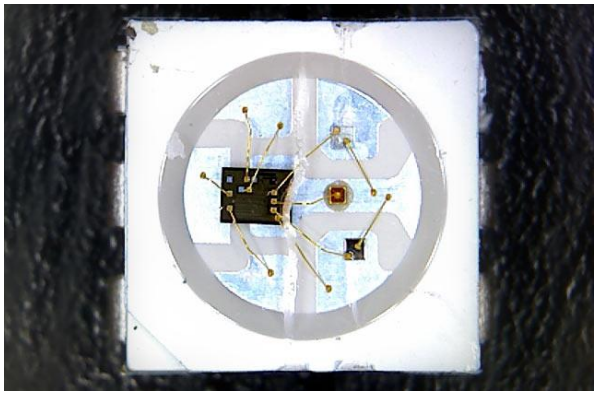
3.1 Display

Základem displeje je 2,5 mm tlustý hliníkový plech o rozměrech 473x180 mm, který slouží jako opora a zároveň chladič LED pásku. Přímo na něm je přilepený samotný LED pásek v osmi řadách, po 26 diodách každá, který překrývá mřížka. Po obvodu je k plechu samořeznými šrouby přimontován hliníkový U profil s rozměry 15x10x1,5 mm na který je přilepeno 4 mm sklo s bílým lakem.

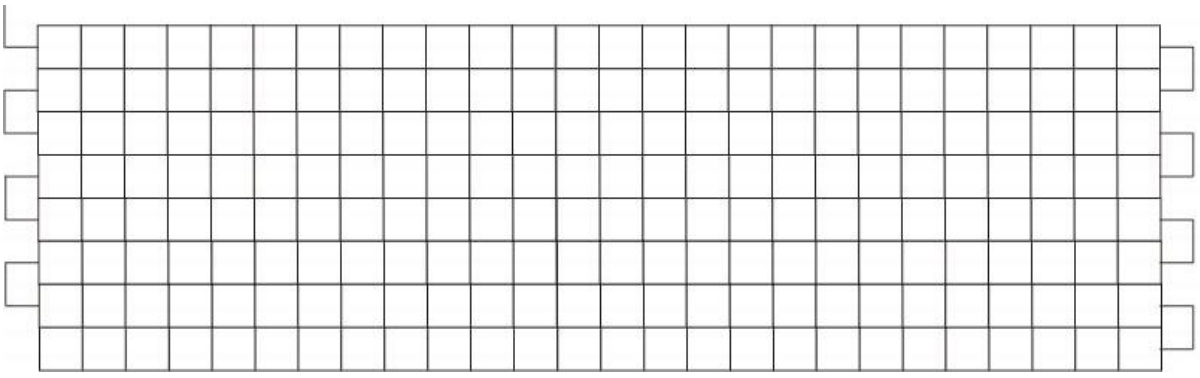
Obrázek 1 Základ displeje

3.2 LED Pásek

LED pásek je složený z RGB čipů WS2812[1]. Každý čip funguje jako posuvný registr. Sám o sobě se po připojení nerozsvítí. K rozsvícení potřebuje data o čísle diody a barvě, jakou se má rozsvítit. Každou barvu lze rozsvítit v 256 úrovních. Napájecí napětí je 5 V a odběr cca 60 mA při plném rozsvícení bílé.Komunikační protokol neodpovídá žádnému standardnímu protokolu (i2C, SPI, RS232). Pro ovládání Arduinem se používá knihovna NeoPixel od Adafruitu[2]. Komunikace s páskem je velmi náročná na přesnost časování (desítky nanosekund). Zapojení jednotlivých řad je realizováno do jakéhosi „hada“, kdy se začíná v levém horním rohu a na konci je propojení s nižší řadou, jak je patrné z obrázku 3.



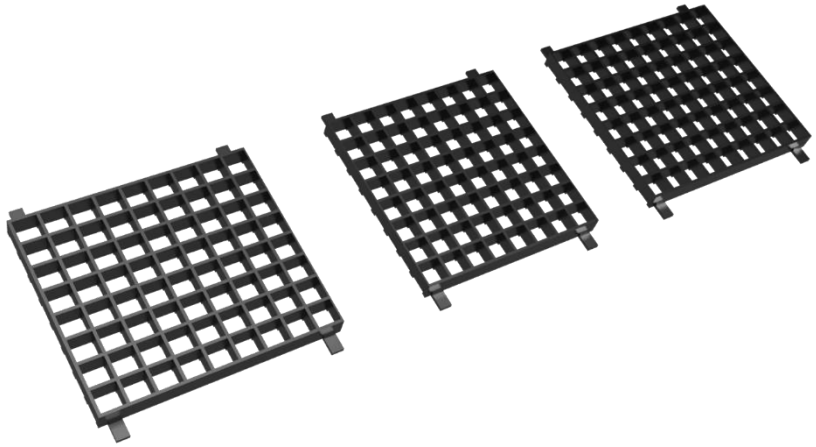
Obrázek 2 Čip WS2812 [3]



Obrázek 3 Zapojení pásku

3.3 Mřížka

Pro oddělení jednotlivých pixelů je použita mřížka o rozměrech buňky 14,9 x 14,9 mm a šířce stěny buňky 0,9 mm, která je vytisknuta na 3D tiskárně. Kvůli rozměrům tiskové plochy tiskárny je rozdělena na díly. Použitý materiál pro tisk je PET-G plast kvůli své odolnosti a větší životnosti a minimální změně objemu při změně teploty.



Obrázek 4 Překryvná mřížka

3.4 Napájení

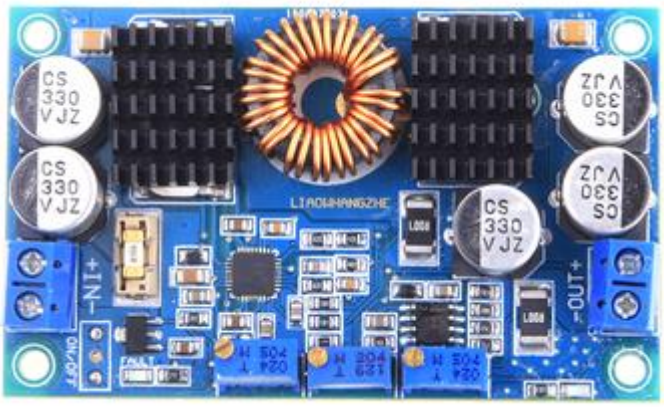
3.4.1 Zdroj

Pro napájení je použit 90 W zdroj od notebooku s výstupním napětím 19 V. Díky vyššímu napětí protéká přívodním vodičem menší proud, což snižuje ztráty. Jelikož je toto napětí pro LED pásky moc vysoké, je použit DC-DC měnič.

Obrázek 5 Notebookový adaptér

3.4.2 DC-DC Měnič

Pro vytvoření vhodného napájecího napětí pro LED pásky byl použit modul DC-DC měniče založený na čipu LTC3780[4]. Tento modul umožňuje nastavit výstupní napětí v rozmezí 0–30 V a umožňuje regulaci proudu. Na vstupu vyžaduje napětí 5–35 V. Vstup je opatřen tzv. „undervoltage protection“ což znamená,že při poklesu vstupního napětí pod nastavenou hodnotu se celý modul deaktivuje. Tato vlastnost se velmi hodí při napájení z baterie. Tyto měniče se vyznačují vysokou účinností a malými rozměry

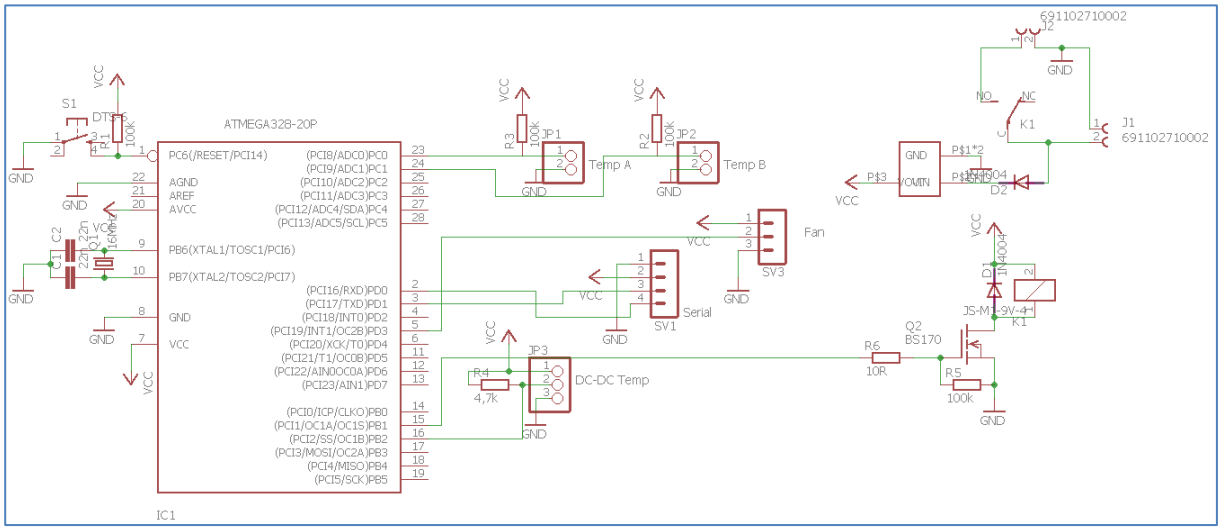


Obrázek 6 DC-DC Měnič [5]

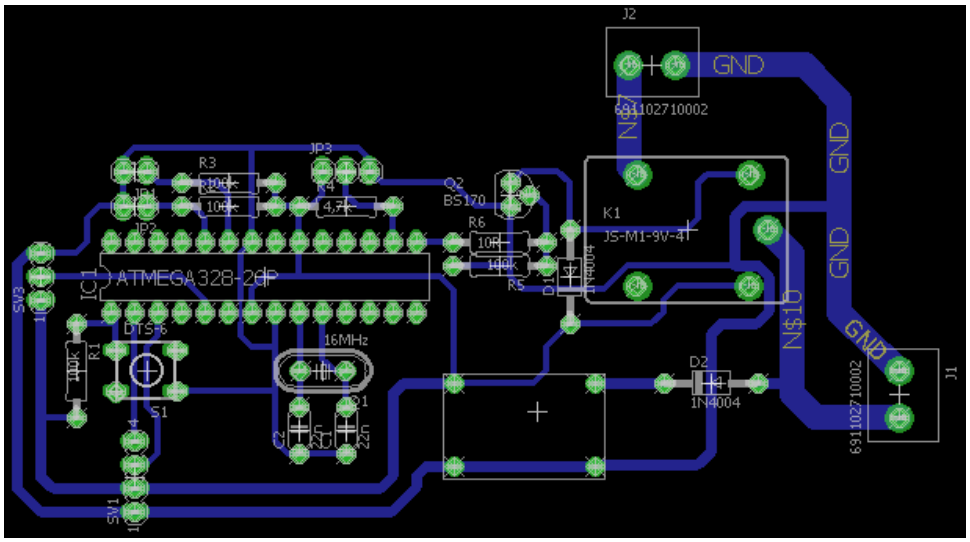
3.4.3 Řídící elektronika zdroje

Elektronika zdroje monitoruje teplotu hlavního DC-DC měniče, teplotu celého displeje, reguluje otáčky ventilátoru a obsahuje malý DC-DC měnič pro napájení veškeré elektroniky. Základem je mikročip ATMega328p[6], který je doplněn o oscilátor složený z 16 MHz krystalu a dvou 22 nF keramických kondenzátorů. Pro případné restartování je na první pin mikročipu připojeno tlačítko s pullup rezistorem. Pro připojení analogových čidel jsou zde konektory JP1 a JP2, u kterých se nachází 100 kΩ rezistor, který tvoří s termistorem dělič napětí. Digitální čidlo se připojuje přes konektor JP3, mezi jehož piny 1 a 2 je připojen 4,7 kΩ rezistor, který slouží jako pullup nebo v případě dvou pinového čidla se skrz něj čidlo napájí.

Dále deska obsahuje relé, přes které je připojen hlavní měnič. Relé je spínáno pomocí n-kanálového mosfetu BS170 doplněného o R5, který slouží jako pulldown. V případě překročení 90 stupňů celsia se relé rozpojí, aby nedošlo k poškození měniče.



Obrázek 7 Schéma elektroniky zdroje



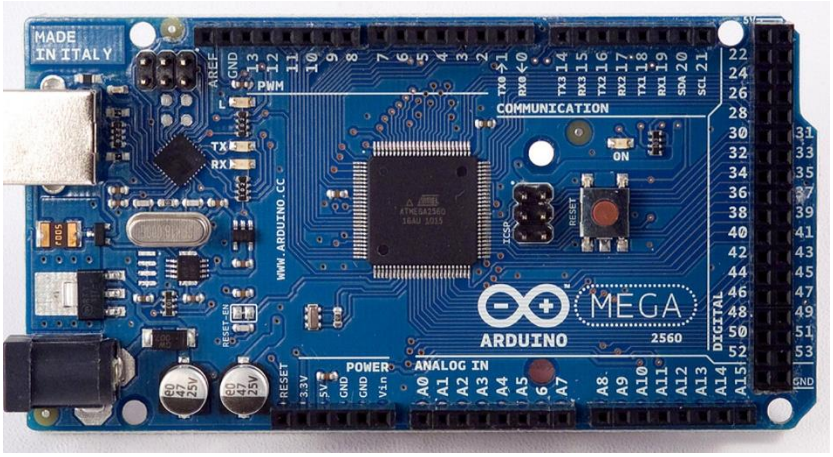
Obrázek 8 Schéma DPS

3.5 Řídící elektronika

Mozkem celého displeje je Arduino MEGA s mikročipem Atmel ATMega2560[7] doplněným o RTC (modul reálného času).

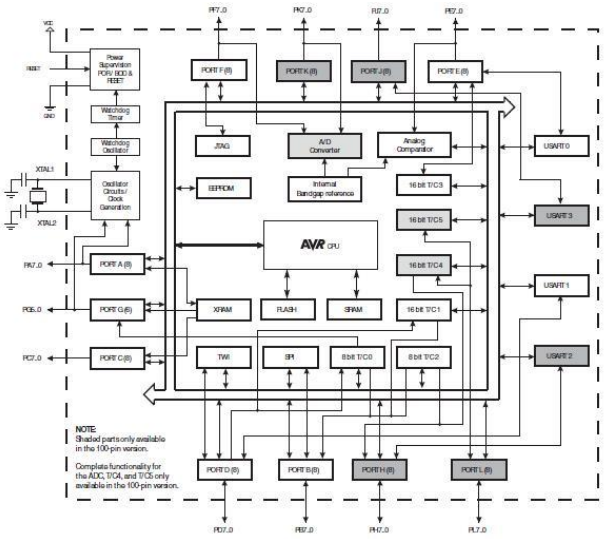
3.5.1 Arduino MEGA

Řízení celého projektu zajišťuje právě Arduino MEGA, které jsem chtěl původně nahradit samotným mikročipem Atmel ATMega2560, ale z důvodu zjednodušení použiji přímo celý modul od Arduina. Obsahuje již FTDI převodník, díky kterému je jednodušší nahrávání nového software či případná komunikace s PC. Arduino MEGA má 54 digitálních vstupně-výstupních pinů. Mezi nimi jsou 4 hardwarové sériové porty, 14 PWM pinů a I^2C[8] rozhraní. Navíc k těmto 54 pinům je možné využít 16 vstupních analogových pinů pro připojení nejrůznějších senzorů. Dalo by se využít i Arduino Uno, ale při velkém počtu diod by mohl nastat nedostatek operační paměti, a proto byl zvolen jeho větší bratr.



Obrázek 9 Arduino MEGA [9]

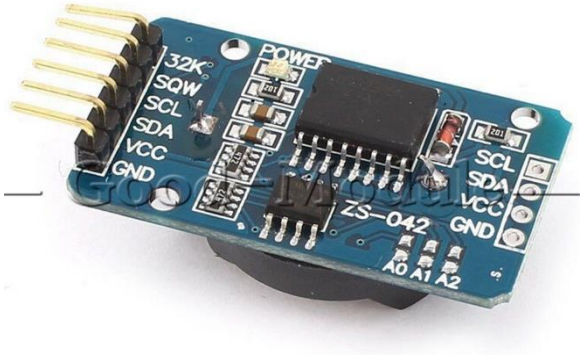
Obsažený mikročip na desce je ATMega2560 taktovaný na 16 MHz. Jedná se o nízko výkonový, osmi bitový CMOS mikrokontrolér založený na AVR rozšířené RISC architektuře. Vyznačuje vysokým výpočetním výkonem, dosahujícím až 1 MIPS při taktovací frekvenci 1 MHz, velkou kapacitou interní FLASH a SRAM paměti.



Obrázek 10 Diagram čipu

3.5.2 RTC Modul

Jelikož použitý mikročip by nebyl dostatečně přesný pro měření času, je použit externí RTC modul s hodinovým čipem DS3231[10] a pamětí AT24C32 o velikosti 32 K. Využívá velmi přesného krystalového oscilátoru. Odchylka může činit maximálně ±2ppm (±0.432 sekund/den). Obsahuje předprogramovaný přesný kalendář do roku 2100. Je doplněn o držák na lithiovou baterii, díky které dokáže udržet čas až 3 roky. Pro komunikaci využívá protokol I^2C.



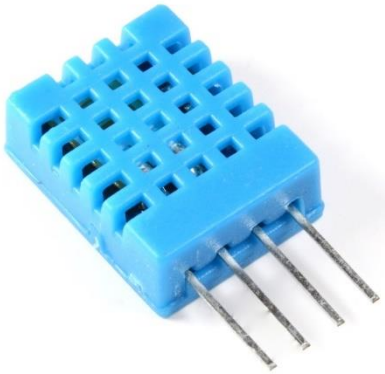
Obrázek 11 RTC Modul [11]

3.6 Čidla

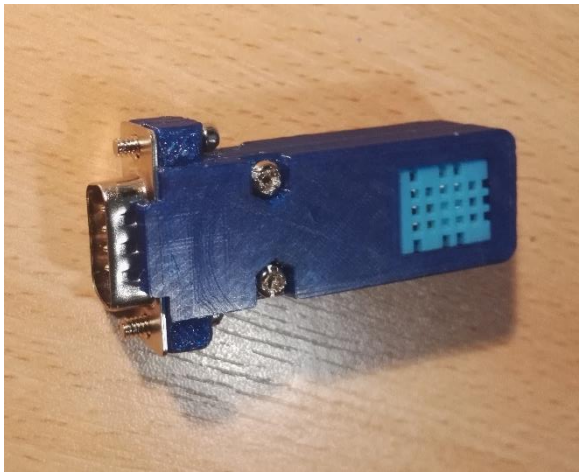
K displeji je možné připojovat nejrůznější čidla, která se dají připojit na sběrnice I^2C nebo 1-Wire[12]. Dále je možné připojit i analogová a jednoduchá digitální čidla. Jednotlivá čidla je možné také kombinovat do modulů.

3.6.1 DHT11

Jedná se o digitální čidlo teploty a vlhkosti, které měří s rozlišením 16 bitů s přesností ± 5 % při měření vlhkosti a ± 2 °C při měření teploty. Napájí se napětím v rozmezí 3,5–5,5 V. K odesílání dat používá jednoduchou sériovou komunikaci přes jeden vodič[13].



Obrázek 12 Čidlo DHT11 [14]



Obrázek 13 Senzor s čidlem DHT11

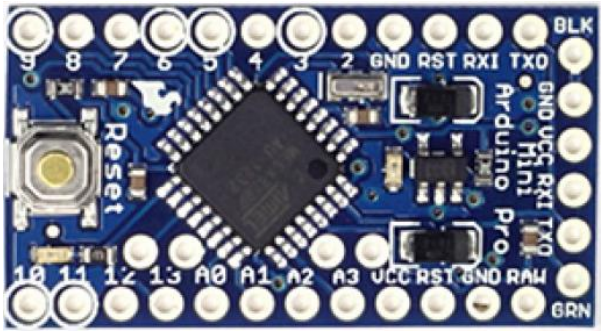
3.7 Ovládací terminál

Slouží k ovládání displeje a ke změnám v nastavení.

3.7.1 Arduino Pro Mini

Základ ovládacího terminálu tvoří Arduino Pro Mini, nejmenší z rodiny arduin. Je založeno na čipu Atmel ATMega328, a jelikož neobsahuje žádný FTDI převodník či programátor, je třeba využít externí.

ATMega328 od společnosti Atmel je 8-bitový AVR mikrořadič založený na RISC architektuře. Obsahuje 32 kB programové flash paměti se schopností čtení při zápisu, 1 kB EEPROM, 2 kB SRAM. Dále obsahuje 23 vstupně výstupních pinů, 32 univerzálních registrů, interní i externí přerušení, programovatelní USART pro seriovou kominikaci, I^2C sběrnici, SPI sběrnici, 6 kanálový 10 bitový A/D převodník. Mikročip může pracovat v rozmezí 1,8–5,5 voltů.

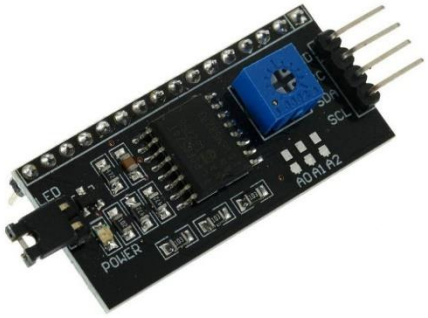


Obrázek 14 Arduino Pro Mini [15]

3.7.2 LCD Display

Jedná se o dvouřádkový alfanumerický LCD displej MC1602E-SBL/H[16] s řadičem, modrým podsvícením a šestnácti znaky na řádek. Je doplněn o kontrolér PCF8574[17] s rozhraním I^2C, který usnadňuje zapojení a eliminuje nutnost použití dalších pasivních součástek.

Obrázek 15 LCD Display [18]



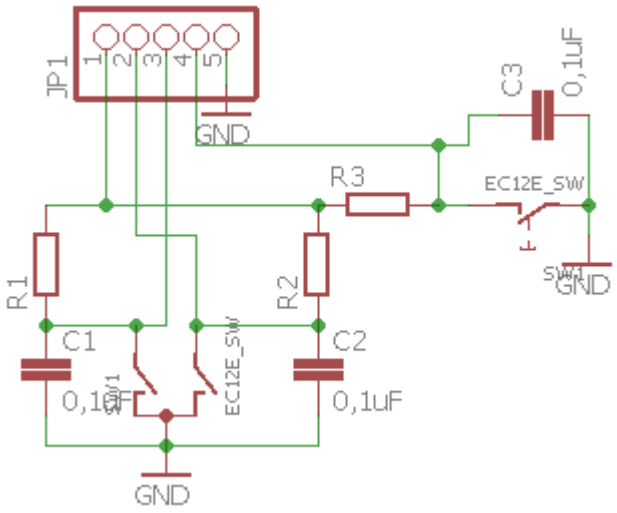
Obrázek 16 I2C Modul [19]

3.7.3 Rotační enkodér

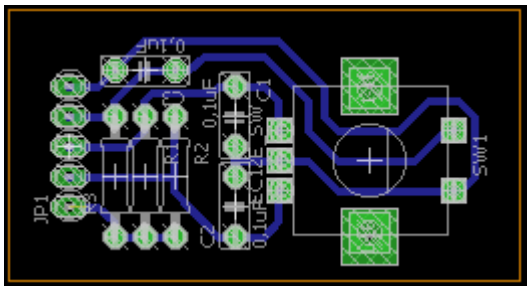
Jako jediný ovládací prvek je použit rotační enkodér se zabudovaným tlačítkem. Rotační enkodér při svém otáčení generuje informaci o rotaci a jejím směru. Výstupní piny A a B se při otáčení spojují s prostředním pinem. Generuje tedy dva obdélníkové signály. Pokud se prvně sepne A, poté B enkodér se otáčí proti směru hodinových ručiček. Pokud se prvně spíná B otáčí se po směru ručiček[20].

Enkodér je připájen na desce, která je vyrobena dle schématu. Pro každý kanál je zde pull up rezistor, který zvedá výstup do logické jedničky. Dále jsou zde 0,1uF svitkové kondenzátory, které slouží k potlačení zákmitů, které při otáčení vznikají. Jedná se sice o malé pulzy, ale mikročip je zvládne detekovat a vyhodnotit jako otáčení.

Obrázek 17 Rotační enkodér [21]



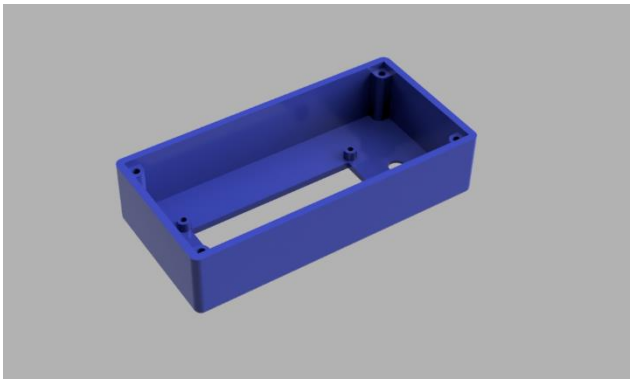
Obrázek 18 Schéma desky enkodéru



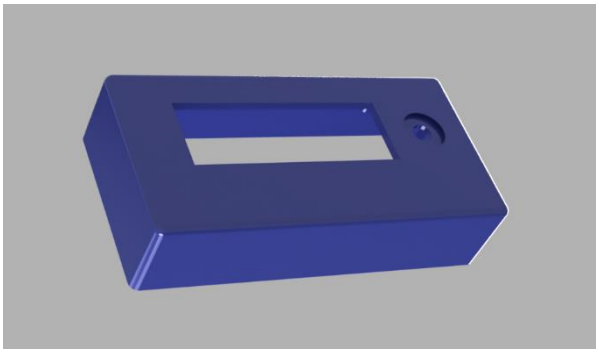
Obrázek 19 Schéma DPS enkodéru

3.7.4 Krabička

Krabička terminálu je vymodelována v programu Autodesk Fusion 360[22] a vytisknuta na 3D tiskárně stejně jako mřížka z plastu PET-G. Je tvořena ze dvou dílů, které jsou sešroubované M3 šrouby.



Obrázek 21 Model krabičky



Obrázek 20 Model krabičky

3.7.5 Připojení

Připojení terminálu k displeji je realizováno pomocí čtyřžilového telefonního kabelu s konektorem RJ-11. Komunikace je realizována pomocí sériové linky RS232 a terminál je napájen skrz tento kabel pěti volty.

3.8 Konektory

Na zadním panelu se nacházejí 4 konektory. Jeden napájecí a tři datové.

Obrázek 22 Zadní panel

3.8.1 Napájecí

K napájení celého displeje slouží souosý konektor 5,5x2,5 mm. Na střední pól se připojuje kladné napětí a na vnější pól zem.

3.8.2 Konektor terminálu

Na připojení terminálu k displeji slouží konektor typu RJ-11, který se nachází na spodní straně krabičky s elektronikou.

Obrázek 23 Konektor RJ-11 [23]

Rozložení pinů

Pin	Název
1	VCC
2	Tx
3	Rx
4	GND

3.8.3 Konektor čidla

Pro připojení čidel je použit konektor CANON 9. Jedná se o konektor, který je ve výpočetní technice hojně využíván pro připojení sériové linky. Nachází se na spodní straně krabičky s elektronikou.

Obrázek 24 Konektor CANON 9

Rozložení pinů

Pin	Název
1	VCC
2	SDA
3	SCL
4	A1
5	A0
6	2
7	3
8	4
9	GND

3.8.4 Přídavný konektor

Tento konektor slouží k připojení jednoduchých zařízení pomocí analogových a digitálních pinů jako například ovladač.

Obrázek 25 Konektor CANON 15

Rozložení pinů

Pin	Název
1	D23
2	D25
3	D27
4	D29
5	D31
6	D33
7	D35
8	D37
9	GND
10	A15
11	A14
12	A13
13	A12
14	A11
15	VCC

4 FIRMWARE

Jelikož se celý projekt skládá ze 3 mikroprocesorů firmware, je rozdělen na 3 části, které spolu komunikují. Firmware všech 3 mikročipů je napsán v jazyce C ve vývojovém prostředí Arduino IDE, jelikož je to nejjednodušší varianta na programování těchto čipů[24].

Každý firmware musí obsahovat dvě základní části: *void setup()* a *void loop()*. Část *setup* se spouští po startu mikrokontroleru a provede se pouze jednou. V zásadě se používá k nastavení, od toho také název *setup*. Můžeme například definovat funkci pinu (vstupní/výstupní) či inicializovat knihovnu. Po dokončení funkce *setup* se spustí *void loop()*, která se poté opakuje donekonečna. Obsahuje tedy kód, který bude mikrokontrolér vykonávat, dokud nedojde k odpojení napájení nebo restartu.

4.1 Hlavní čip

Firmware hlavní ho čipu se stará o ovládání LED pásku, počítání dat k zobrazení, komunikaci s terminálem a čtení dat ze senzorů.

4.1.1 Použité knihovny

Pro zjednodušení firmwaru jsou použity knihovny pro komunikaci s pásky, RTC modulem a čidly.

- Adafruit_NeoPixel – Jedná se o knihovnu od firmy Adafruit, která obsahuje kompletní komunikační protokol pro odesílání dat do LED pásku. Před odesláním si musíme připravit data v paměti mikrokontroleru. Pomocí funkce *setPixelColor(n, r, g, b)* nastavíme barvu daného pixelu. První argument funkce znamená číslo diody, které budeme barvu nastavovat. Počítáme od nuly kdy 0 je pixel, který je připojen k mikrořadiči. Další tři argumenty jsou složky RGB modelu. Další možnost jak namíchat barvu je použití funkce *Color(r, g, b)*, která vrací 32 bitové číslo. Poté co jsou data připravena odešleme je funkcí *show()*.

- Adafruit_NeoMatrix – Knihovna slouží k přepočtu souřadnic pixelů na čísla diod, aby je bylo možné rozsvítit pomocí *NeoPixel* knihovny. Při inicializaci knihovny musíme dodat argumenty, kterými popíšeme, jak je matrix sestaven a zapojen. Například Adafruit_NeoMatrix (MATRIXWIDTH, MATRIXHEIGH, MATRIXPIN, NEO_MATRIX_TOP + NEO_MATRIX_LEFT + NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG, NEO_GRB + NEO_KHZ800) první dva argumenty určují šířku a výšku matrixu. Poté pin mikrokontroleru, na který je matrix připojen. Další část určuje uspořádání:
    - NEO_MATRIX_TOP – první pixel nahoře
    - NEO_MATRIX_LEFT – první pixel je vlevo
    - NEO_MATRIX_ROWS – zapojeno po řadách
    - NEO_MATRIX_ZIGZAG – jednotlivé řady jsou vždy na konci spojeny se spodní řadou viz Obrázek 3

  Poté už jen zvolíme typ diod, ze kterých je matrix sestaven.
- Adafruit_GFX – Grafická knihovna k vytváření nejrůznějších obrazců a psaní textů. Podporuje i nejrůznější fonty. Je velmi univerzální a dá se také využít s grafickými LCD displeji.
- Wire[25] – Standartní Arduino knihovna, která je obsažena v IDE
- RTClib[26] – Slouží ke komunikaci s modulem reálného času, je použita ke zjednodušení ovládání modulu. Její dvě základní funkce jsou *now()* a *adjust*(). Funkce *now* vrací *DateTime* objekt obsahující aktuální datum a čas získaný z modulu. Funkce *adjust* slouží k úpravě aktuálního času modulu.
- DHT[27] – Knihovna, která je použita pro sbírání dat z čidla DHT-11, které snímá teplotu a vlhkost.

4.1.2 Struktura programu

Program se dělí na čtyři hlavní části. V první části jsou připojené výše popsané knihovny, definována čísla hardwarových pinů. Dále jsou ty vytvořeny nutné globální proměnné. Druhou částí je funkce *setup,* kde je spuštěna komunikace a nastavené mody pinů. Ve třetí části se nachází funkce *loop* a v ní jednotlivé programy. Poslední je obsluha vnitřního přerušení.

4.1.2.1 Knihovny, definice, proměnné

Pomocí příkazu *include* jsou připojeny všechny potřebné knihovny. Následují inicializace jednotlivých knihoven s dodáním veškerých parametrů. Dále jsou globální proměnné, z nichž jsou některé deklarované jako *volatile* což sděluje kompilátoru, že s touto proměnnou mohou manipulovat i jiné funkce nebo obsluhy přerušení. Nejdůležitější z globálních proměnných je *taskSwitch*, která slouží k přepínání programů. Dále proměnná *colors* obsahuje barvu, která je využita při zobrazování znaků a základních tvarů na displej. Poslední proměnná *firstRun* je využita na vytvoření jakýchsi *setup* funkcí pro jednotlivé programy, pokud je třeba něco spustit jen jednou po startu.

```
#include <Adafruit_NeoPixel.h>
#include <Adafruit_GFX.h>
#include <Adafruit_NeoMatrix.h>
#include <Wire.h>
#include "RTClib.h"
#include "display.h"
#include "snake.h"
// SENSOR LIBS
#include <DHT.h>

//Pin definitions
#define MATRIXPIN 6
#define UP 33
#define DOWN 31
#define LEFT 35
#define RIGHT 37
#define Abutton 29
#define Bbutton 27
#define DHTPIN 2

//Matrix parameters
#define MATRIXWIDTH 26
#define MATRIXHEIGH 8

//Initalize matrix
Adafruit_NeoMatrix matrix = Adafruit_NeoMatrix(MATRIXWIDTH, MATRIXHEIGH, MATRIXPIN,
                            NEO_MATRIX_TOP     + NEO_MATRIX_LEFT +
                            NEO_MATRIX_ROWS + NEO_MATRIX_ZIGZAG,
                            NEO_GRB            + NEO_KHZ800);

RTC_DS3231 rtc;
Display display(&matrix);
Snake snake(&matrix, MATRIXWIDTH, MATRIXHEIGH);
DHT dht(DHTPIN, DHT11);

// GLOBAL VARIABLES
volatile byte taskSwitch = 1;
volatile uint16_t colors = matrix.Color(255, 255, 255);
volatile boolean firstRun = true;

// Programs variables
int posX = 5;
int posY = 10;
```

Obrázek 26 Hlavní čip – Knihovny, definice, proměnné

4.1.2.2 Funkce setup

Na začátku se pomocí příkazu *pinMode* nastaví piny použité pro ovladač přípojený přes pomocný konektor. Příkaz *pinMode* má dva argumenty. První určuje číslo pinu který budeme nastavovat. Druhý argument určuje mod. Existují 3 mody: OUTPUT – pin je výstupní, INPUT – pin je vstupní, INPUT_PULLUP – pin je vstupní a navím se připojí vnitřní *pullup* resistor, který nastavuje pin do logické jedničky.

Dále se aktivuje RTC modul a sériová komunikace na obou linkách rychlostí 9600 bitů za sekundu. Poté následuje aktivace knihovny NeoMatrix a nastavení základního jasu a barvy a pomocí fillScreen(0) je display zhasnut.

Poslední část funkce setup nastavuje *timer interrupt* přímou manipulací s řídícími registry mikrokontroléru.

```
52 void setup() {
53   pinMode(UP, INPUT_PULLUP);
54   pinMode(DOWN, INPUT_PULLUP);
55   pinMode(LEFT, INPUT_PULLUP);
56   pinMode(RIGHT, INPUT_PULLUP);
57   pinMode(Abutton, INPUT_PULLUP);
58   pinMode(Bbutton, INPUT_PULLUP);
59
60
61   rtc.begin();
62   Serial.begin(9600);
63   Serial1.begin(9600);
64   Serial1.setTimeout(2000);
65   matrix.begin();
66   matrix.setTextWrap(false);
67   matrix.setBrightness(50);
68   matrix.setTextColor(colors);
69   matrix.fillScreen(0);
70
71   TCCR1A = 0;
72   TCCR1B = 0;
73   TCCR1B |= (1 << CS10) | (1 << CS11);
74   TIMSK1 |= (1 << TOIE1);
75 }
```

Obrázek 27 Hlavní čip – Setup

4.1.2.3 Funkce loop

V *loop* se nacházejí jednotlivé programy oddělené v cyklech, mezi kterými se přepíná proměnou *taskSwitch*. Aktuální kód obsahuje 5 programů.

1. Hodiny – Program, který zobrazuje aktuální čas získaný z RTC modulu. K zobrazení se používá funkce *showTime*, která je obsažena v souboru funkcí *Display*, který vytvořil autor projektu pro zjednodušení zobrazování dat na displeji.
2. Čtení dat ze senzorů – Tento program sbírá data z připojeného senzoru, v tomto případě DHT-11, které zobrazuje pomocí funkcí *showTemperature* a *showHumidity*. Tyto příkazy také spadají do objektu Display.
3. Bouncing pixel – Jedná se o jednoduchou animaci, kdy po displeji „lítá“ pixel, který se odráží od hran displeje.
4. Hra Snake – Velmi populární hra, která je hlavně známa ze starých mobilních telefonů. Ukazuje, že je takový display schopen i hraní jednoduchých her.
5. Lampa – Slouží pouze k demonstračním účelům. Rozsvítí všechny diody přednastavenou barvou.

```
78  void loop() {
79    while (taskSwitch == 1) {
80      if (millis() % 100 == 0) {
81        DateTime now = rtc.now();
82        byte hour = now.hour();
83        display.showTime(hour, now.minute(), colors);
84        delay(5);
85      }
86    }
87    while (taskSwitch == 2) {
88      if (firstRun) {
89        dht.begin();
90      } else {
91        float h = dht.readHumidity();
92        float t = dht.readTemperature();
93        if (isnan(h) || isnan(t)) {
94          display.showText("No sensor");
95        }
96        if (millis() % 6000 >= 0 && millis() % 6000 <= 3000) {
97          showTemperature(t);
98        }
99        if (millis() % 6000 > 3000 && millis() % 6000 <= 6000) {
100         showHumidity(h);
101       }
102     }
103
104   }
105   while (taskSwitch == 3) {
106     if (dirX) {
107       posX++;
108     } else {
109       posX--;
110     }
```

Obrázek 28 Hlavní čip – Loop

4.1.2.4 Obsluha přerušení

Timer interrupt je vnitřní přerušení, které se spíná podle nastavení registrů. Každý hodinový cyklus se inkrementuje hodnota registru TCNT1. Jelikož je ten to registr 16-bitový přeteče přibližně za 4 milisekundy a nastaví se příznak přerušení. Nejpoužívanější jsou dva módy. Normal mode, kdy se s každým přetečením spustí obsluha přerušení, která vykoná nějaký kód. Druhý je CTC mód, který porovnává registr TCNT1 s registry OCR1A a OCR1B. Jakmile se registry shodují, je vyvoláno přerušení. V kódu je využit Normal mode, takže prvně vynulujeme řídící registry Timer1 a poté v registru TCCR1B nastavíme děličku frekvence na 64 abychom zvýšili čas přerušení na přibližně 260 ms. Bit TOIE1 v registru TIMSK1 povoluje přerušení při přetečení registru TCNT1. Obsluhu tohoto přerušení zapisujeme do ISR(TIMER1_OVF_vect) {}[28].

Toto přerušení se využívá ke zpracování komunikace po sériové lince nezávisle na aktuálně prováděném kódu. Komunikaci vždy uvozuje příkazový bajt. Podle tohoto bajtu se očekávají data nebo se data odešlou. Seznam příkazů je uveden v tabulce.

Hodnota bajtu	Příkaz	Popis
0xE0	Změna programu	Mění aktuálně běžící program. Následuje 1 bajt s číslem programu.
0xE1	Změna barvy	Mění aktuálně používanou barvu pro zobrazování znaků. Následují 3 bajty s hodnotami složek RGB.
0xE2	Změna jasu	Mění aktuální jas displeje. Následuje 1 bajt s hodnotou jasu.
0xE3	Změna data a času	Mění aktuální čas v RTC modulu. Následuje 6 bajtů ve formátu YY-MM-DD-HH-MM-SS kdy rok je o 2000 menší
0xD0	Aktuální jas	Slouží ke zjištění aktuálního jasu. Odpovídá se jedním bajtem s aktuální hodnotou jasu
0xD1	Aktuální datum a čas	Slouží ke zjištění aktuálního data a času. Odpovídá se šesti bajty ve formátu HH-MM-SS-DD-MM-YY kdy rok je o 2000 menší

```
171 ISR(TIMER1_OVF_vect) {
172   if (Serial1.available() > 0) {
173     byte incoming = Serial1.read();
174     switch (incoming) {
175       case 224: // change program
176         byte program[1];
177         Serial1.readBytes(program, 1);
178         firstRun = true;
179         taskSwitch = program[0];
180         break;
181       case 225: //setColor
182         byte colorData[3];
183         Serial1.readBytes(colorData, 3);
184         colors = matrix.Color(colorData[0], colorData[1], colorData[2]);
185         matrix.setTextColor(colors);
186         break;
187       case 226: //setBrightness
188         byte brightness[1];
189         Serial1.readBytes(brightness, 1);
190         matrix.setBrightness(brightness[0]);
191         break;
192       case 227: //setDateTime
193         byte dateTime[6];
194         Serial1.readBytes(dateTime, 6);
195         rtc.adjust(DateTime(((int)dateTime[0] + 2000), dateTime[1], dateTime[2], dateTime
196         //                  year             month         day             hour        m
197         break;
198       case 208: //0xD0
199         Serial1.write(matrix.getBrightness());
200         break;
201       case 209:
202         DateTime now = rtc.now();
203         byte dataOut[6] = {(byte)now.hour(), (byte)now.minute(), (byte)now.second(), (byt
204         Serial1.write(dataOut, 6);
```

Obrázek 29 Hlavní čip – Obsluha přerušení

4.2 Terminál

Firmware terminálu obsahuje velké menu, které slouží k nastavení parametrů hlavního čipu.

4.2.1 Použité knihovny

Pro zjednodušení firmwaru jsou použity knihovny pro komunikaci s pásky, RTC modulem a čidly.

- Wire – Standartní Arduino knihovna pro I^2C komunikaci, která je obsažena v IDE
- LiquidCrystal_I2C[29] – Knihovna pro komunikaci s LCD displejem, který je doplněn o I^2C modul s čipem PCF8574.
- Rotary[30] – Knihovna od brianlow, která slouží ke snímání rotačního enkodéru. Obsahuje algoritmy pro přesné snímání s potlačením zákmitů.

4.2.2 Struktura programu

Program se dělí na tři hlavní části. V první části jsou připojené výše popsané knihovny definovaná čísla hardwarových pinů. Dále jsou ty vytvořeny nutné globální proměnné. Druhou částí je funkce *setup,* kde je spuštěna komunikace a nastavené mody pinů. Ve třetí části se nachází funkce *loop* a v ní celá struktura menu.

4.2.2.1 Knihovny, definice, proměnné

Pomocí příkazu include jsou připojeny všechny potřebné knihovny. Následují inicializace jednotlivých knihoven s dodáním veškerých parametrů. Dále jsou globální proměnné, z nichž jsou některé deklarované jako volatile, což sděluje kompilátoru, že s touto proměnnou mohou manipulovat i jiné funkce nebo obsluhy přerušení. Nejdůležitější z globálních proměnných je TurnDetected, která signalizuje, že bylo otočeno enkodérem. Proměnná up poté signalizuje, jestli bylo otočeno po směru hodinových ručiček. Dále se zde nachází pole bajtů, ve kterém je vytvořen symbol šipky pro menu. Poslední část je obsluha vnějšího přerušení, která detekuje otočení enkodéru.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Rotary.h>


#define rotaryA 2
#define rotaryB 3
#define rotaryButton 4

LiquidCrystal_I2C lcd(0x27, 16, 2);
Rotary rotary = Rotary(rotaryA, rotaryB);
long timeout = 0;
volatile boolean TurnDetected = false;
volatile boolean up = false;
volatile byte mode = 0;
volatile int menuPosition = 0;
volatile int cursorPosition = 0;
String options[6] = {"Select Program", "Change Color", "Set Brightness", "Set Time","Exit"};

byte arrow[8] = {
  B01000,
  B01100,
  B01110,
  B01111,
  B01111,
  B01110,
  B01100,
  B01000
};

void isr0 ()  {
  if (mode > 0) {
    unsigned char result = rotary.process();
    if (result == DIR_CW) {
      TurnDetected = true;
      up = false;
    } else if (result == DIR_CCW) {
      TurnDetected = true;
      up = true;
    }
  }
}
```

Obrázek 30 Terminál – Knihovny, definice, proměnné

4.2.2.2 Funkce setup

Na začátku se pomocí příkazu *pinMode* nastaví piny použité pro rotační enkodér.

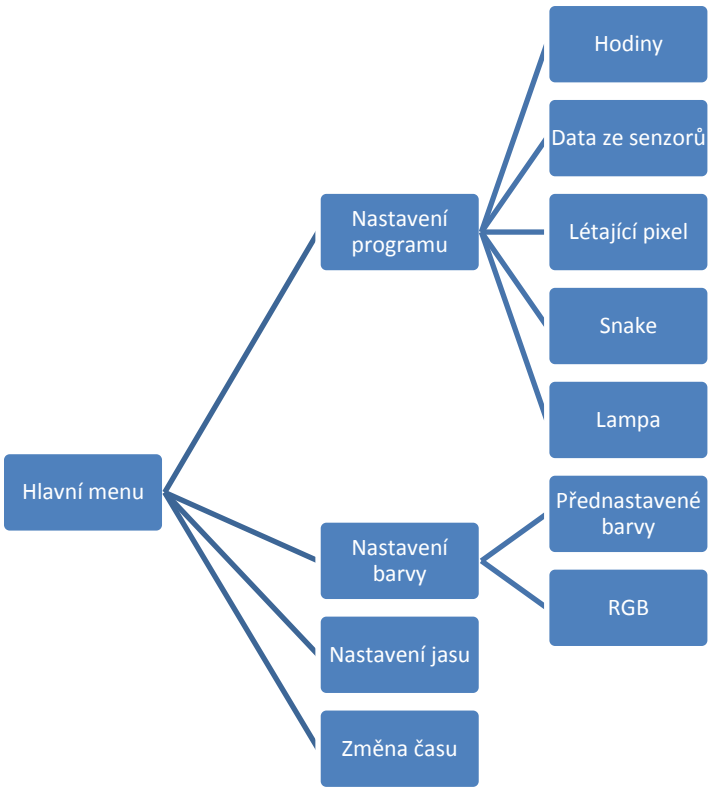
Dále se aktivuje sériová komunikace rychlostí 9600 bitů za sekundu a nastaví se diplay. Poté následuje definice externích přerušení na pinech 2 a 3, kde je připojen rotační enkodér. Obsluhou těchto přerušení je funkce isr0(), která signály z enkodéru zpracuje.

```
44 void setup() {
45   pinMode(rotaryA, INPUT);
46   pinMode(rotaryB, INPUT);
47   pinMode(rotaryButton, INPUT);
48   Serial.begin(9600);
49   lcd.begin();
50   lcd.backlight();
51   lcd.createChar(0, arrow);
52   attachInterrupt(digitalPinToInterrupt(rotaryA), isr0, CHANGE);
53   attachInterrupt(digitalPinToInterrupt(rotaryB), isr0, CHANGE);
54   displayData();
55 }
```

Obrázek 31 Terminál – Setup

4.2.2.3 Funkce loop

V loop se nachází menu, které se dále větví do několika podmenu viz diagram.



Obrázek 32 Terminál – Struktura menu

Hlavní menu se dělí na 4 podmenu. V nastavení programu zvolíme, který program chceme spustit. Dále je zde nastavení barvy, ve kterém se nachází deset přednastavených barev a možnost vytvořit si vlastní barvu pomocí RGB modelu. V nastavení jasu je možné procentuálně nastavit jas displeje. Změna času umožňuje upravit aktuální čas RTC modulu.

```
57 void loop() {
58   if (!(digitalRead(rotaryButton))) {
59     timeout = millis();
60     while (!(digitalRead(rotaryButton))) {}
61     switch (mode) {
62       case 0:
63         mode = 1;
64         break;
65       case 1:
66         switch ((menuPosition + cursorPosition + 1)) {
67           case 1:
68             changeProgram();
69             break;
70           case 2:
71             changeColor();
72             break;
73           case 3:
74             changeBrightness();
75             break;
76           case 4:
77             changeTime();
78             break;
79           case 5:
80             mode = 0;
81             menuPosition = 0;
82             cursorPosition = 0;
83             break;
84         }
85         break;
86     }
87     displayData();
88     delay(50);
89   }
90   if (mode == 1 && TurnDetected) {
91     timeout = millis();
92     if (up) {
93       if (cursorPosition == 1) {
94         menuPosition++;
95         if (menuPosition > 3) {
96           menuPosition = 0;
97           cursorPosition = 0;
98         }
99       } else {
```

Obrázek 33 Terminál – Loop

4.3 Elektronika zdroje

Firmware mikročipu v elektronice zdroje slouží ke kontrole teploty měniče, podle které reguluje ventilátor. V případě překročení nastavené hranice odpojí napájení.

4.3.1 Použité knihovny

Pro zjednodušení firmwaru jsou použity knihovny pro komunikaci s pásky, RTC modulem a čidly.

- OneWire – Knihovna od Paula Stoffregena slouží k simulaci sběrnice 1-Wire na některém z pinů Arduina.
- DS18B20[31] – Jedná se o knihovnu pro komunikaci se sériovým teplotním čidlem DS18B20 od uživatele nettigo.

4.3.2 Struktura programu

Program se dělí na tři hlavní části. V první části jsou připojené výše popsané knihovny definovaná čísla hardwarových pinů. Dále jsou ty vytvořeny nutné globální proměnné. Druhou částí je funkce *setup*, kde je spuštěna komunikace a nastavené mody pinů. Ve třetí části se nachází funkce *loop* a v ní celá struktura menu.

4.3.2.1 Knihovny, definice, proměnné

Pomocí příkazu include jsou připojeny všechny potřebné knihovny. Následují inicializace jednotlivých knihoven s dodáním veškerých parametrů. Dále je zde pole bajtů obsahující adresu teplotního čidla.

```
1  #include <DS18B20.h>
2  #include <OneWire.h>
3
4  #define RELAY 9
5  #define FAN 3
6  #define TEMPA A0
7  #define TEMPB A1
8  #define TEMP 10
9
10 byte sensorAddress[8] = {0x28, 0xFF, 0xBC, 0x6C, 0xC2, 0x16, 0x3, 0x1C};
11 OneWire oneWire(TEMP);
12 DS18B20 tempSensor(&oneWire);
```

Obrázek 34 Elektronika zdroje – Knihovny, definice, proměnné

4.3.2.2 Funkce setup

Na začátku se pomocí příkazu *pinMode* nastaví piny použité pro větráček a relé.

Dále se sepne relé a aktivuje komunikace s teplotním čidlem.

```
13 void setup() {
14   pinMode(RELAY, OUTPUT);
15   pinMode(FAN, OUTPUT);
16   digitalWrite(RELAY, HIGH);
17   tempSensor.begin();
18   tempSensor.request(sensorAddress);
19 }
```

Obrázek 35 Elektronika zdroje – Setup

4.3.2.3 Funkce loop

V loop se kontroluje teplota na desce DC-DC měniče, a jakmile přesáhne 30 stupňů celsia spustí se ventilátor na nízké otáčky. S přibývající teplotou je větráček plynule regulován. Pokud teplota přesáhne 90 °C, je modul odpojen a ventilátor spuštěn na plný výkon.

```
21 void loop() {
22   if (tempSensor.available())
23   {
24     float temp = tempSensor.readTemperature(sensorAddress);
25     if (temp > 30) {
26       analogWrite(FAN, map(temp, 30, 65, 50, 255));
27     } else {
28       digitalWrite(FAN, LOW);
29     }
30     if (temp > 90){
31         digitalWrite(RELAY, LOW);
32         analogWrite(FAN, 255);
33         delay(30000);
34       }
35     tempSensor.request(sensorAddress);
36   }
```

Obrázek 36 Elektronika zdroje – Loop

5 Závěr

Při realizaci projektu jsem získal mnoho nových poznatků v oblasti elektrotechniky a programování. V módu hodin je měření času velmi přesné a dosahuje skoro neznatelných odchylek. Celý display má velmi dobré pozorovací úhly a je dobře čitelný i při velké intenzitě okolního světla. Konfigurace přes připojitelný terminál je velmi jednoduchá a intuitivní. Zvládne ji i technicky nezaložený člověk.

Do budoucna plánuji o rozšíření Wi-Fi modulem ESP8266, díky kterému by bylo možné získávat data na zobrazení z internetu nebo synchronizace RTC modulu s internetovým časem přes protokol NTP. Dále bych chtěl rozšířit portfolio podporovaných senzorů.

Jednou z těžších částí byla komunikace mezi terminálem a displejem, jelikož občas část příkazu nedorazí a čip čeká na něco, co už nikdy nepřijde.

Uplatnit by se takový display mohl jako moderní hodiny, informační display či jednoduchá herní konzole. Využití najde, kdekoliv je potřeba zobrazit RGB grafiku bez nutnosti velkého rozlišení.

6 Zdroje

[1] *WS2812 datasheet* [online]. WORLDSEMI [cit. 2018-03-20]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

[2] *Adafruit Überguide* [online]. Adafruit [cit. 2018-03-20]. Dostupné z: https://learn.adafruit.com/adafruit-neopixel-uberguide/the-magic-of-neopixels

[3] *Obrázek WS2812* [online]. In: Phillip Burgess [cit. 2018-03-20]. Dostupné z: https://learn.adafruit.com/assets/10668

[4] *LTC3780 datasheet* [online]. Linear Technology [cit. 2018-03-20]. Dostupné z: http://www.analog.com/media/en/technical-documentation/data-sheets/3780ff.pdf

[5] *Obrázek DC-DC měnič* [online]. In: [cit. 2018-03-20]. Dostupné z: http://cdn.shopify.com/s/files/1/1978/9859/products/01_595_17_13_1024x1024.jpg?v=1499267138

[6] *Atmel ATmega328 datasheet [online]. Atmel [cit. 2018-03-20]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf*

[7] *Atmel ATmega2560 datasheet* [online]. Atmel [cit. 2018-03-20]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

[8] *I^2C specifikace* [online]. [cit. 2018-03-20]. Dostupné z: http://i2c.info/i2c-bus-specification

[9] *Obrázek Arduino Mega 2560* [online]. In: . [cit. 2018-03-20]. Dostupné z: https://images-na.ssl-images-amazon.com/images/I/81u3Xz46PKL._SL1297_.jpg

[10] *DS3231 datasheet* [online]. Maxim Integrated, 2015 [cit. 2018-03-20]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

[11] *Obrázek RTC modul* [online]. In: . [cit. 2018-03-20]. Dostupné z: https://i.ebayimg.com/images/g/ntkAAOSwI59aQHyf/s-l1600.jpg

[12] *1-Wire specifikace* [online]. Maxim Integrated [cit. 2018-03-20]. Dostupné z: https://www.maximintegrated.com/en/app-notes/index.mvp/id/1796

[13] *DHT11 datasheet* [online]. Aosong Electronics [cit. 2018-03-20]. Dostupné z: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf

[14] *Obrázek DHT11* [online]. In: . [cit. 2018-03-20]. Dostupné z: http://www.hobbyandyou.com/content/images/thumbs/0002132_dht11-digital-temperature-and-humidity-sensor-sensor-arduino_100.jpeg

[15] *Obrázek Arduino Pro Mini* [online]. In: . [cit. 2018-03-20]. Dostupné z: https://store-cdn.arduino.cc/uni/catalog/product/cache/1/image/520x330/604a3538c15e081937dbfbd20aa60aad/e/0/e000025_featured.jpg
[16] *MC1602E-SBL/H datasheet* [online]. [cit. 2018-03-20]. Dostupné z: https://www.gme.cz/data/attachments/dsh.513-128.1.pdf
[17] *PCF8574 datasheet* [online]. NXP, 2013 [cit. 2018-03-20]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf
[18] *Obrázek display 16x2* [online]. In: . [cit. 2018-03-20]. Dostupné z: https://www.gme.cz/data/product/480_480/pctdetail.775-068.1.jpg?ts=1520575944
[19] *Obrázek I^2C display driver* [online]. In: . [cit. 2018-03-20]. Dostupné z: https://i.ebayimg.com/images/g/J0YAAOSwcu5UNp~I/s-l1600.jpg
[20] *Rotary Encoder datasheet* [online]. 2017 [cit. 2018-03-20]. Dostupné z: https://www.gme.cz/data/attachments/dsh.532-110.1.pdf
[21] *Obrázek Rotační enkodér* [online]. [cit. 2018-03-20]. Dostupné z: https://media.rs-online.com/t_large/F6234136-01.jpg
[22] *Autodesk Fusion 360* [online]. [cit. 2018-03-20]. Dostupné z: https://www.autodesk.com/products/fusion-360/students-teachers-educators
[23] *Obrázek Konektor RJ-11* [online]. [cit. 2018-03-20]. Dostupné z: https://www.gme.cz/data/product/480_480/pctdetail.833-024.1.jpg?ts=1504876085
[24] *Arduino reference* [online]. Arduino [cit. 2018-03-20]. Dostupné z: https://www.arduino.cc/reference/en/
[25] *Knihovna Wire* [online]. [cit. 2018-03-20]. Dostupné z: https://www.arduino.cc/en/Reference/Wire
[26] RTC library. *Github* [online]. Adafruit [cit. 2018-03-20]. Dostupné z: https://github.com/adafruit/RTClib
[27] DHT library. *Github* [online]. Adafruit [cit. 2018-03-20]. Dostupné z: https://github.com/adafruit/DHT-sensor-library
[28] Electronic Basics #30: Microcontroller (Arduino) Timers. In: *YouTube* [online]. GreatScott! [cit. 2018-03-20]. Dostupné z: https://www.youtube.com/watch?v=IdL0_ZJ7V2s&t=263s
[29] *LCD Displays (Blue and Yellow) with I2C/TWI Interface* [online]. Terry King [cit. 2018-03-20]. Dostupné z: https://arduino-info.wikispaces.com/LCD-Blue-I2C
[30] Rotary Encoder Arduino Library. *Github* [online]. brianlow [cit. 2018-03-20]. Dostupné z: https://arduino-info.wikispaces.com/LCD-Blue-I2C*How*
[31] DS18B20 sensor library for Arduino. *Github*[online]. nettigo [cit. 2018-03-20]. Dostupné z: https://github.com/nettigo/DS18B20

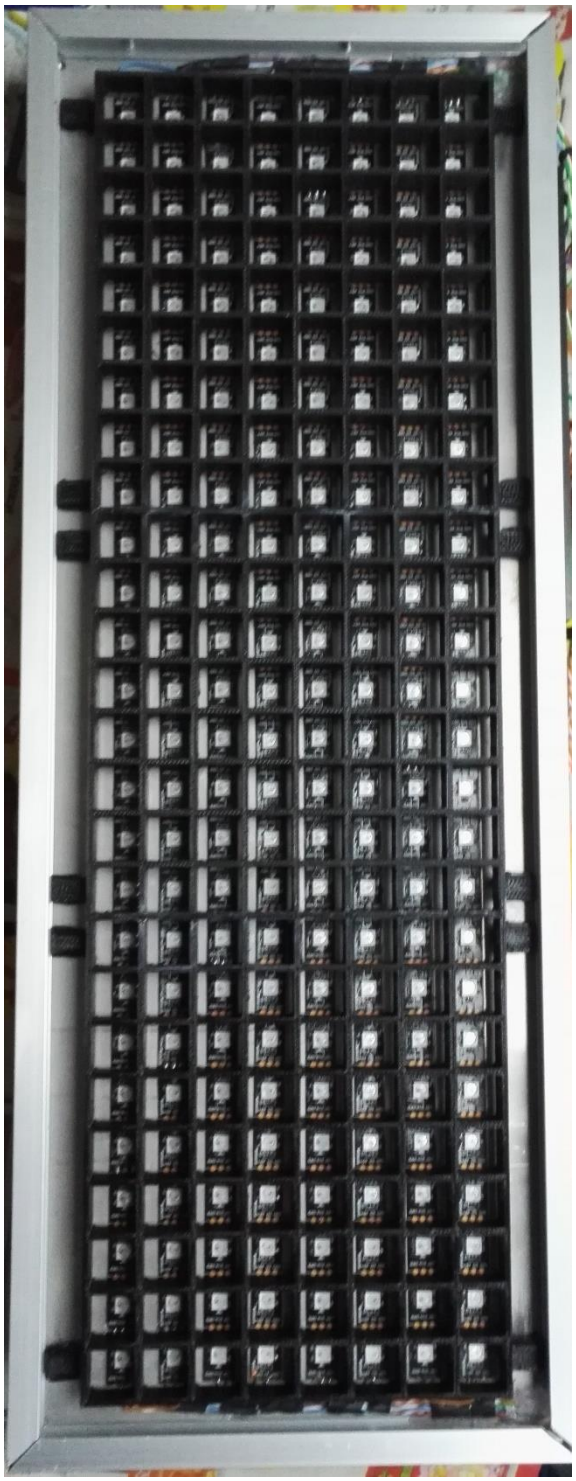
7 Seznam obrázků

Obrázek 1 Základ displeje.....10

Obrázek 2 Čip WS2812 [3] 11
Obrázek 3 Zapojení pásku 11
Obrázek 4 Překryvná mřížka 12
Obrázek 5 Notebookový adaptér 13
Obrázek 6 DC-DC Měnič [5] 13
Obrázek 7 Schéma elektroniky zdroje 14
Obrázek 8 Schéma DPS 14
Obrázek 9 Arduino MEGA [9] 15
Obrázek 10 Diagram čipu 16
Obrázek 11 RTC Modul [11] 16
Obrázek 12 Čidlo DHT11 [14] 17
Obrázek 13 Senzor s čidlem DHT11 17
Obrázek 14 Arduino Pro Mini [15] 18
Obrázek 15 LCD Display [18] 19
Obrázek 16 I2C Modul [19] 19
Obrázek 17 Rotační enkodér [21] 20
Obrázek 18 Schéma desky enkodéru 20
Obrázek 19 Schéma DPS enkodéru 20
Obrázek 21 Model krabičky 21
Obrázek 20 Model krabičky 21
Obrázek 22 Zadní panel 22
Obrázek 23 Konektor RJ-11 [23] 22
Obrázek 24 Konektor CANON 9 23
Obrázek 25 Konektor CANON 15 23
Obrázek 26 Hlavní čip – Knihovny, definice, proměnné 26
Obrázek 27 Hlavní čip – Setup 27

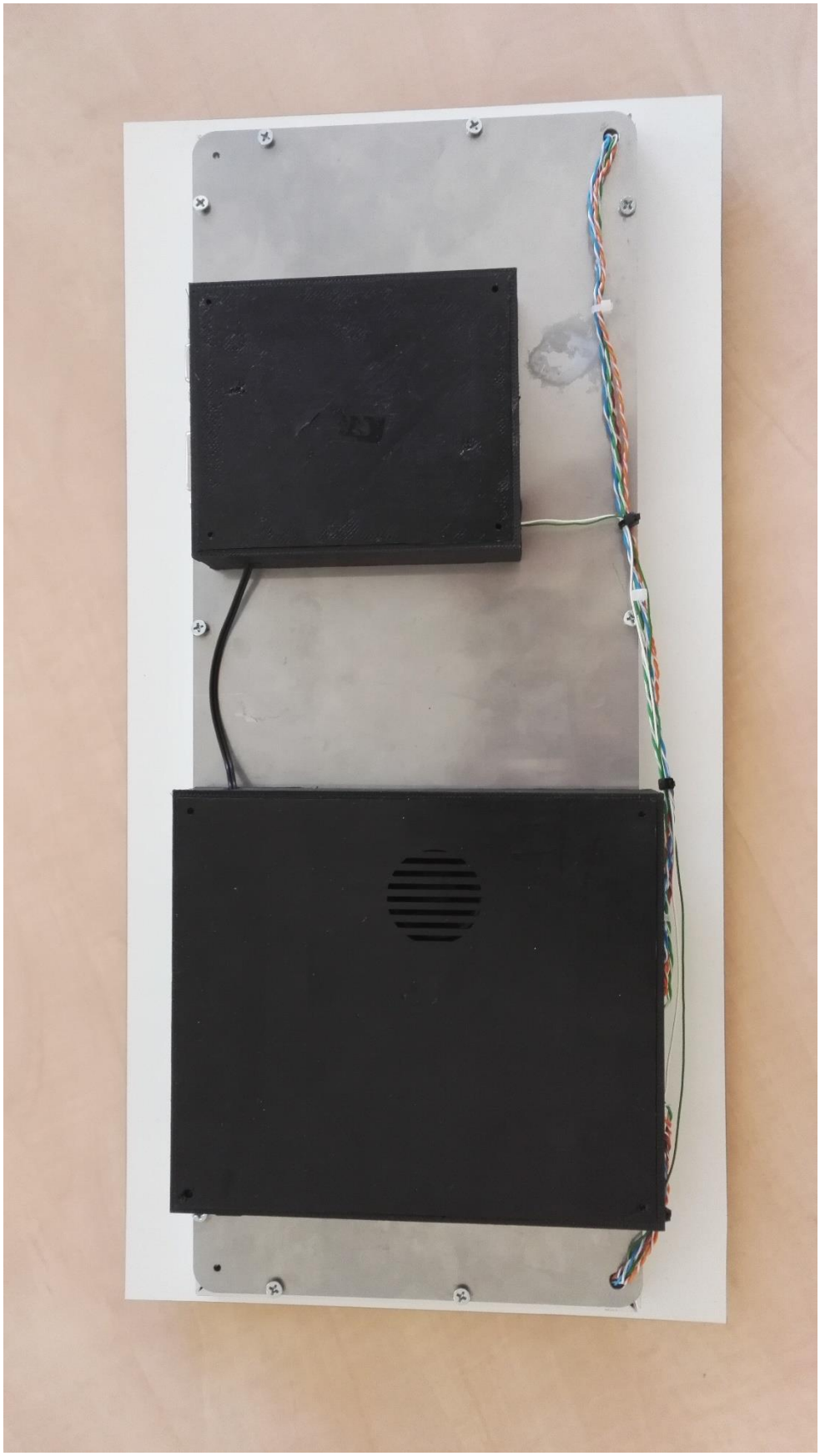
Obrázek 28 Hlavní čip – Loop 28
Obrázek 29 Hlavní čip – Obsluha přerušení 30
Obrázek 30 Terminál – Knihovny, definice, proměnné 31
Obrázek 31 Terminál – Setup 32
Obrázek 32 Terminál – Struktura menu 32
Obrázek 33 Terminál – Loop 33
Obrázek 34 Elektronika zdroje – Knihovny, definice, proměnné 34
Obrázek 35 Elektronika zdroje – Setup 35
Obrázek 36 Elektronika zdroje – Loop 35
Obrázek 37 Display s hliníkovým rámem 41
Obrázek 38 Předek displeje 42
Obrázek 39 Zadní část displeje 42
Obrázek 40 Zadní část odhalená elektronika 42
Obrázek 41 Terminál 42
Obrázek 42 Terminál odhalená elektronika 42

8 Přílohy

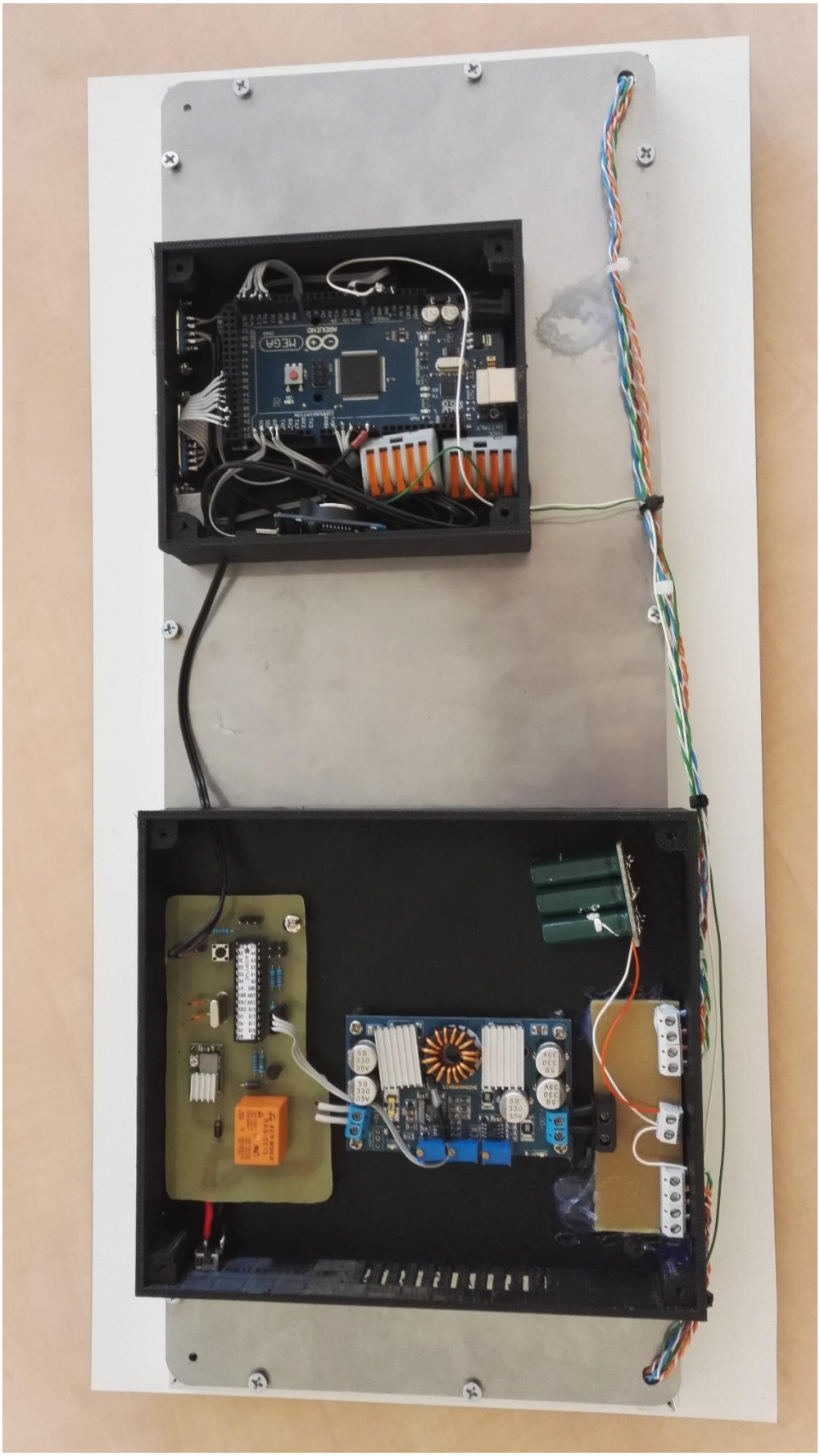


Obrázek 37 Display s hliníkovým rámem

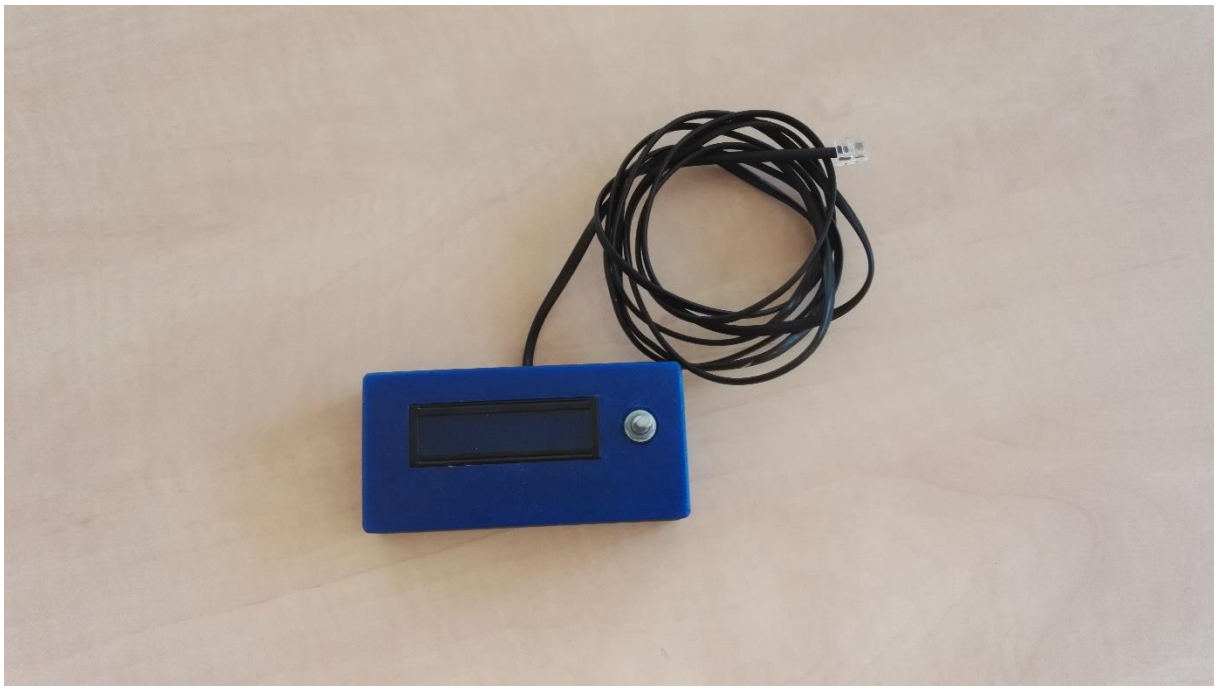
Obrázek 38 Předek displeje



Obrázek 39 Zadní část displeje



Obrázek 40 Zadní část odhalená elektronika



Obrázek 41 Terminál

Obrázek 42 Terminál odhalená elektronika