STŘEDNÍ PRŮMYSLOVÁ ŠKOLA BRNO, PURKYŇOVA,
PŘÍSPĚVKOVÁ ORGANIZACE

PID TERMOSTAT

DAVID RADEK

V4B

PROFILOVÁ ČÁST MATURITNÍ ZKOUŠKY
MATURITNÍ PRÁCE

BRNO 2023

ZADÁNÍ MATURITNÍ PRÁCE

obhajované před zkušební komisí

Školní rok: 2022/2023

Předmět: Soubor odborných předmětů (zaměření na informační technologie)

Studijní obor: Informační technologie (18-20-M/01)

ŠVP: Informační technologie

Žák: **Radek David**

Třída: **V4B**

Vedoucí práce: Ing. Petr Pernes

Odborný konzultant:

Termín zadání MP: 1. prosince 2022

Termín odevzdání MP: 17. dubna 2023

Číslo zadání, verze zadání a název práce:

AUTO1c: ARDUINO – PID termostat

Zadání

Naprogramujte PID termostat včetně vytvoření HW. Systém se bude skládat z libovolného zobrazení, ovládání a teploměru. Případně další V/V dle potřeby. Dále naprogramujte vhodné funkce jako: PID, nastavení PID, zobrazování hodnot. Případně další funkce jako: max, min, avg, komunikace se serverem, uložení do EEPROM apod. Připravte si model, na kterém předvedete funkčnost PID termostatu. **Konstrukce bude provedena na platformě ARDUINO. Hlavní program pro ARDUINO musí být vytvořen studentem. V konstrukci použijte univerzální PCB, nebo lepší.**

Výstupem práce bude

Funkční systém dle zadání.

Program pro ARDUINO.

Schéma zapojení.

Postup instalace SW/HW.

Uživatelská příručka s popisem funkcí (může být součástí dokumentace).

Dokumentace a zdrojové kódy (včetně knihoven) v elektronické podobě.

Doporučené **prostředky pro řešení, technické požadavky**

- www.arduino.cc; www.arduino.cz

Součástí zadání je Příloha zadání maturitní práce.

V Brně dne 1. 12. 2022

---	---	---
Podpis žáka	Podpis vedoucího práce	Podpis předsedy PK

Prohlášení

Prohlašuji, že jsem maturitní práci na téma *PID termostat* vypracoval samostatně a použil jen zdroje uvedené v seznamu literatury.

Prohlašuji, že:

- Beru na vědomí, že zpráva o řešení maturitní práce a základní dokumentace k aplikaci bude uložena v elektronické podobě na intranetu Střední průmyslové školy Brno, Purkyňova.
- Beru na vědomí, že bude má maturitní práce včetně zdrojových kódů uložena v knihovně SPŠ Brno, Purkyňova, dostupná k prezenčnímu nahlédnutí.
- Beru na vědomí, že SPŠ Brno, Purkyňova má právo celou moji práci použít k výukovým účelům a po mém souhlasu nevýdělečně moji práci užít ke své vnitřní potřebě.
- Beru na vědomí, že pokud je součástí mojí práce jakýkoliv softwarový produkt, považují se za součást práce i zdrojové kódy, které jsou předmětem maturitní práce, případně soubory, ze kterých se práce skládá. Součástí práce není cizí ani vlastní software, který je pouze využíván za přesně definovaných podmínek, a není podstatou maturitní práce.

David Radek

Valtická 2, Brno

Dne: Podpis:

Poděkování

Děkuji vedoucímu maturitní práce Ing. Petru Pernesovi za metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé maturitní práce.

Dále děkuji mému otci, Pavlu Radkovi, za odbornou pomoc při tvorbě hardwaru. Zároveň i za poskytnutí materiálu pro tvorbu hardwaru.

Anotace

Čeština

V této práci popisuji svůj postup při řešení zadání na téma PID termostat. Jsou v ní obsáhlé poznatky, řešení problémů a ilustrace. Jejím účelem je také uvést čtenáře do problematiky termostatů, které využívají vývojovou desku ESP32 a vysvětlit jak fungují PID ovladače obecně.

Klíčové pojmy:
PID, Arduino, ESP32, DS1307, Flash paměť, termostat, PCB design, 3d tisk, CAD

English

In this work, I describe my process for solving a task on the topic of a PID thermostat. It contains extensive knowledge, problem-solving approaches, and illustrations. Its purpose is to introduce the reader to homemade thermostats that utilize the ESP32 development board and explain how PID controllers work in general.

Key terms:
PID, Arduino, ESP32, DS1307, Flash memory, thermostat, PCB design, 3D printing, CAD.

OBSAH

Seznam použitých zkratek

SPI - Sériové periferní rozhraní - Serial Peripheral Interface

I2C - Inter-Integrated Circuit (sběrnice)

USB - Univerzální sériová sběrnice - Universal Serial Bus

CPU - Centrální procesorová jednotka - Central Processing Unit

RAM - Paměť s náhodným přístupem – Random Access Memory

LED - Dioda s emitováním světla - Light Emitting Diode

LCD - Kapalinový displej - Liquid Crystal Display

OLED - Organický kapalinový displej - Organic Light Emitting Diode

SSID - Identifikátor sady služeb - Service Set Identifier

CAD - Počítačem podporovaný návrh - Computer-Aided Design

PCB - Tištěný spojový obvod - Printed Circuit Board

PID - Proportional Integral Derivative

RTC - Hodiny s reálným časem - Real-Time Clock

GND - Zemnící bod - Ground

VCC - Napájecí napětí - Voltage Common Collector

CACHE - Dočasná paměť - Temporary memory

1 TEORETICKÝ ÚVOD

1.1 Cíl práce

Cílem práce bylo vytvořit PID termostat založený na programovatelné vývojové desce Arduino.

Termostat měl umožnit uživateli jednoduše nastavit minimálně 3 proměnné a to Kp, Ki a Kd. Tyto 3 proměnné mají ovládat charakteristiku PID ovladače. Důležité také bylo, aby se nastavení nesmazalo po restartu zařízení.

Jako vedlejší funkci by měl obsahovat alespoň jednoduché plánování procedur.

1.2 Důvod výběru práce

Práci jsem si vybral, protože jsem nechtěl čistě jen naprogramovat software. Proto mi vyhovovala práce s ESP32 které od uživatele nevyžaduje velké znalosti v elektrotechnice a umožňuje opětovné nahrání jednoduchého programu. Už před výběrem práce jsem jej používal na malé projekty jako např. dálkově ovládané vypínání/zapínání PC, tudíž jsem měl představu o tom, v čem bude práce s ním spočívat.

Další důvod byl také ten, že jsem věděl, že se projekt využije v praxi. Moji rodiče se totiž hledaly řešení problému vytápění chalupy, která se v zimním období nenavštěvuje, ale zároveň je důležité, aby teplota v interiéru neklesla pod 0°C na dlouhou dobu. To se dost blížilo možnému zadání maturitní práce a mohl jsem tzv. „zabít dvě mouchy jednou ranou“.

1.3 Popis možných řešení zadání

Jako prvních bych rád stanovil své podmínky řešení, které jsem si přidal k původnímu zadání. Důvodem těchto podmínek je ten aby se dalo později používat v praxi.

1.3.1 Dodatečné podmínky pro řešení

1. Mimo automatický mód založený na plánování by měl program obsahovat i manuální nastavení teploty.
2. Program musí být dost jednoduchý na to, aby jej mohli ovládat starší generace
3. Zařízení se musí dát připevnit ke zdi

1.3.2 Ovládací Interface

Ovládat zařízení by se dalo například pomocí webového serveru, který by hostoval samotný ESP32 a na který by se uživatel připojil přes Wi-Fi síť kterou by mohl sám vysílat. Kdyby měl používat již existující Wi-Fi síť tak by byl problém v zadávání SSID a jména. Bez jeho připojení by se na něj totiž nedalo připojit. Jeden ze způsobů jak ošetřit tuto situaci by byl nastavit v programu základní SSID a heslo na které se má zařízení připojit po X neúspěšných pokusech připojení. Poté by uživatel jen na svém mobilním telefonu vytvořil hotspot se stejným SSID a heslem a tudíž byl schopný se na zařízení připojit a nastavit správná data.

ESP32 má schopnost výstupu a vstupu přes sériovou linku, toho by se dalo využít a ovládat zařízení čistě přes příkazy přes sériovou linku. Tohle řešení má ale značné nevýhody hlavně kvůli nutnosti používat počítač jako médium.

Poté nám zbývají ještě fyzické ovládací a zobrazovací prvky, těmi může být například 8mi segmentový, OLED, Alphabetic LCD, a mnoho dalších. Je důležité si ale umět vybrat správný display pro tohle použití.

LCD displej | 8mi segmentový displej | OLED displej

Obrázek 1 - foto variant displejů

- 8mi segmentový display
  - nízká spotřeba
  - nenáročný na výkon ESP32
  - jednoduše čitelný pokud vykresluje pouze velmi jednoduchá data
  - neschopný vykreslit složitější obrazce
  - levný, dlouhá životnost
- OLED display
  - Nízká spotřeba pokud zobrazuje z většiny černý obraz
  - Může zatěžovat ESP32
  - Větší rozměry jsou velmi drahé a menší jsou špatně čitelné
  - Schopný vykreslit složitý vícebarevný výstup
  - Trpí na tzv. burn in který se objevuje, když je na displeji dlouho stejný výstup (což je u termostatu vysoce pravděpodobné)
  - Drahý, krátká životnost
  - Dá se koupit s dotykovou plochou, která výrazně zjednodušuje navigaci
- Alphabetic LCD display

- Nízká spotřeba
- Nezatěžuje ESP32
- Úzký pozorovací úhel a vyžaduje podsvícení celé plochy
- Schopný vykreslit písmena, číslice a jiné znaky ale neschopný vykreslit obrazce
- Levný, velmi dlouhá životnost a odolnost

Z těchto 3 možností je pro tento projekt nejvhodnější LCD display. K němu se dá velmi dobře využít encoder s tlačítkem a samostatná tlačítka.

1.3.3 Získávání vstupních dat

Nás hlavně způsob, jakým bude získáván údaj o nynější teplotě, ale je možné do vstupních dat zahrnout i vlhkost nebo čas.

Teplotu lze získat pomocí tepelného čidla, ideálně by bylo dobré mít čidlo mimo zařízení a tím zvýšit jeho použitelnost (například může zařízení být v kuchyni a ovládat teplotu v boileru o patro níž). Některá čidla zároveň podporují i měření vlhkosti.

Čas můžeme získat pomocí tzv. RTC modulu (Real Time Clock). Ten je vybaven interní baterií, která jej udržuje pod napětím i při výpadku proudu a umožňuje tak neustále přičítat do registru (posunovat čas dopředu), z něj poté můžeme čas přečíst. Tyto moduly často podporují i dny v týdnu, přestupné roky a zimní a letní čas.

1.3.4 Propojení jednotlivých modulů – hardware

Propojení, se nejjednodušeji dá zprostředkovat pomocí nepájivého pole. Jeho hlavní výhodou je modulárnost a možnost kdykoli spojení změnit. Bohužel je ale velké a špatně se s ním manipuluje. Dál může být také problém jeho nespolehlivost oproti propájeným spojům.

Lepší možnost by mohla být prototypová deska, která umožňuje propájení modulů jak je potřeba. Její výhody jsou spolehlivost spojů, efektivnější rozměry a celkem jednoduchá úprava cest. Nevýhoda je, že samotná tvorba cest bývá občas velmi náročná a zdlouhavá.

Nejlepší je použít na míru vyrobenou desku plošných spojů (PCB). U ní je možnost přizpůsobit tvar a cesty co nejefektivněji to jde, nevýhoda ale je že když se vytvoří, tak už musí být design finální a jakékoli úpravy jsou velice náročné někdy až nemožné.

2 ROZBOR ŘEŠENÍ

2.1 Popis modulů

Většina modulů je propojena pomocí sběrnice I2C. V určité fázi vývoje mi přestala komunikace na sběrnici fungovat. Dlouho jsem nemohl najít důvod. Až teprve článek Issues with the I²C (Inter-IC) Bus and How to Solve Them (viz zdroje) mi vysvětlil, co jsem na sběrnici udělal špatně (jednalo se o problém s pull-up rezistory).

2.1.1 -Encoder

Veškeré vstupní ovládání uživatele je za pomocí rotačního encoderu pro který jsem nenašel dokumentaci ani modelové označení, ale je prakticky totožný s modelem Keyes KY-040. Modul má integrovaný mikrospínač a všechny piny jsou příhodně vyvedeny na kraj desky s roztečí 2.54mm.

Obrázek 2 - fotka encoderu

Uvnitř této součástky se nachází kotouč s magnety ve střídavé konfiguraci a dva senzory magnetického pole. Uvnitř tohoto encoderu se nachází disk, ve kterém jsou magnety. Tyto magnety mají všechny stejnou velikost a jsou od sebe stejně vzdáleny. Mimo disk jsou v encoderu ještě dva senzory magnetického pole které snímají ty magnety. Pomocí těchto senzorů se dá zjistit, který z nich se v přesnou chvíli nachází nad magnetem a protože jsou dva tak můžeme zjistit i kterým směrem se k němu magnet dostal. Následně může převést tyto data do digitální podoby pomocí esp32 a knihovny ESP32Encoder (viz zdroje). Existuje mnoho knihoven, ale já vybral tuto, protože má implementované interrupt requesty a tudíž je výpis otáček nezávislý na rychlosti kódu (interrupt request umožňuje ESP32 prioritně spustit kód nezávisle na zbytku kódu).

2.1.3 LCD

Nativní výstup je zařízen pomocí znakového LCD displeje IIC I2C LCD 1602. Můj model používá I2C převodník LCM1602 IIC V1. Jedná se o jednoduchý LCD display, schopný vykreslovat ASCII znaky a 8 uživatele, definovaných znaků. Tyto znaky se zobrazují v 16x2 mřížce. Display má pevně danou intenzitu podsvícení, které podsvěcuje celou plochu najednou. Podsvícení vyžaduje napájení 5V jinak poblikává ale logika převodníku je schopna operovat na 3.3V.

Obrázek 3 - obrázek LCD

Pro komunikaci využívá paralelní zapojení, které se pomocí přídavného I2C převodníku a vhodné knihovny dá ovládat přes I2C sběrnici. Knihovna, kterou využívám, se jmenuje LiquidCrystal_I2C (viz zdroje) a získal jsem ji ze stránek prodejce.

2.1.4 Interní a externí sensor

Interní senzor

Jedná se o modul LA131145A, využívající součástku SHT3X-DIS. Tento modul se vyznačuje hlavně kombinací měření teploty i vlhkosti. Vyžaduje napájení v rozsahu 5-3.3V, což je pro mé využití dostačující, a ve finálním zapojení jej napájím 3.3 volty.

Obrázek 4 - fotka interního senzoru tepla

Pro komunikaci využívá sběrnici I2C, pro kterou má zabudované 10KΩ odpory. Ty jsem ale odstranil, protože by ve finálním zapojení byli zapojené paralelně s ostatními pull-up odpory ostatních zařízení a snižovaly celkový odpor sběrnice pod minimální hranici 5KΩ. V programu je pro něj připojená knihovna Arduino-SHT (viz zdroje). Při přečtení neinicializovaného nebo nezapojeného senzoru knihovna vrací hodnotu *undefined* která se chová rozdílně než *null*.

Externí senzor

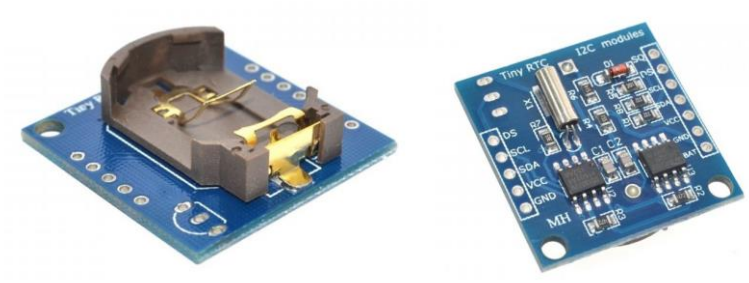
V případě, že by uživatel chtěl ovládat pomocí termostatu například kotel nebo nějakou jinou tekutinu, je zařízení schopné sbírat teplotu i z externího senzoru. Pro tohle využití se perfektně hodil senzor Dallas – DS18B20, zapouzdřený ve vodotěsné metalické sondě a 1 metr dlouhým kabelem. Jeho nevýhodou je nepřesnost, která je daná už při výrobě a v realitě pohybuje se kolem 1.5°C v obou směrech.

Obrázek 5 - fotka externího senzoru tepla

Senzor má 3 vývody Data, GND, a napájení. Důležité je napojit mezi napájení a data 4.7KΩ rezistor. Poté se se zařízením dá komunikovat přes sběrnici OneWire a za použití knihovny DallasTemperature (viz zdroje). Při přečtení neinicializovaného nebo nezapojeného senzoru knihovna vrací hodnotu -127°C. Senzor je schopný maximálně změřit teplotu -60°C takže je jednoduché zjistit, kdy není zapojený.

2.1.5 Real time clock

Poslední ale velmi důležitý interní modul je RTC neboli hodiny reálného času. Pro toto zařízení jsem vybral modul, který využívá součástku DS1307. Zajímavé na tomto modulu je že má velmi detailní a promyšlenou dokumentaci, která mi velmi pomohla při vývoji zařízení. Na desce tohoto modulu je přidaný ještě jeden modul a to 32kB non-volatilní paměti atmel332 který ale nevyužívám. Pro správnou funkčnost modulu je zapotřebí do něj zapojit ještě 3.3V baterii LIR2032. Je důležité do něj vložit dobíjitelnou baterii, protože se modul bude pokoušet zapojenou baterii, ať už to podporuje či ne a při nesprávném výběru by mohla baterie začít hořet. Modul vyžaduje napájení 5V ale jeho logika funguje i na 3.3V, v této konfiguraci ale není schopný dobíjet baterii, pouze ji držet na konstantním napětí.

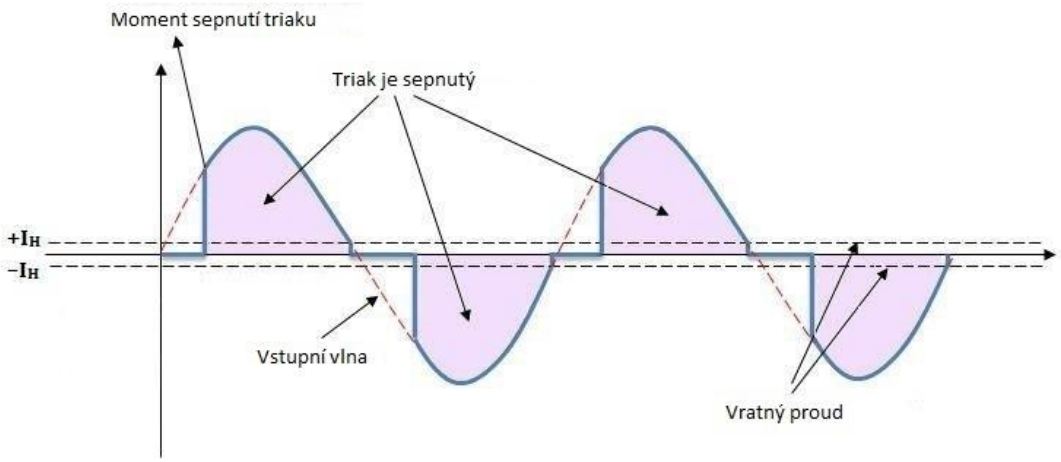


Obrázek 6 - fotka RTC

Pro komunikaci rovněž využívá I2C sběrnici pro kterou ale vyžaduje knihovnu „RTClib“ (viz zdroje) kterou jsem získal z návodu na použití od prodejce.

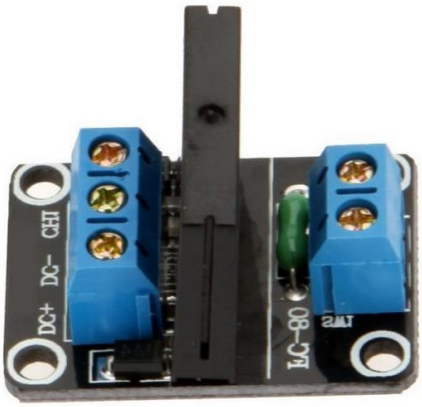
2.1.6 SSR

Modul SSR není velmoc ale relé. Jedná se o polovodičové relé OMRON G3MB-202P, schopné velmi rychle rozpojit či spojit dva vstupy pomocí fototriaku zabudovaného uvnitř. Tzv. triaková regulace, kterou relé využívá, je dobře vysvětlená na následujícím obrázku:



Obrázek 7 - vysvětlení triakové regulace

V zařízení jej používám k izolování ESP32 od ovládaného topného/chladícího zařízení. Ovládá se velmi jednoduše pomocí tří vstupů: GND, napájení a Data. Pro sepnutí relé je třeba na vodič data přivést logickou nulu (0-1.2V).



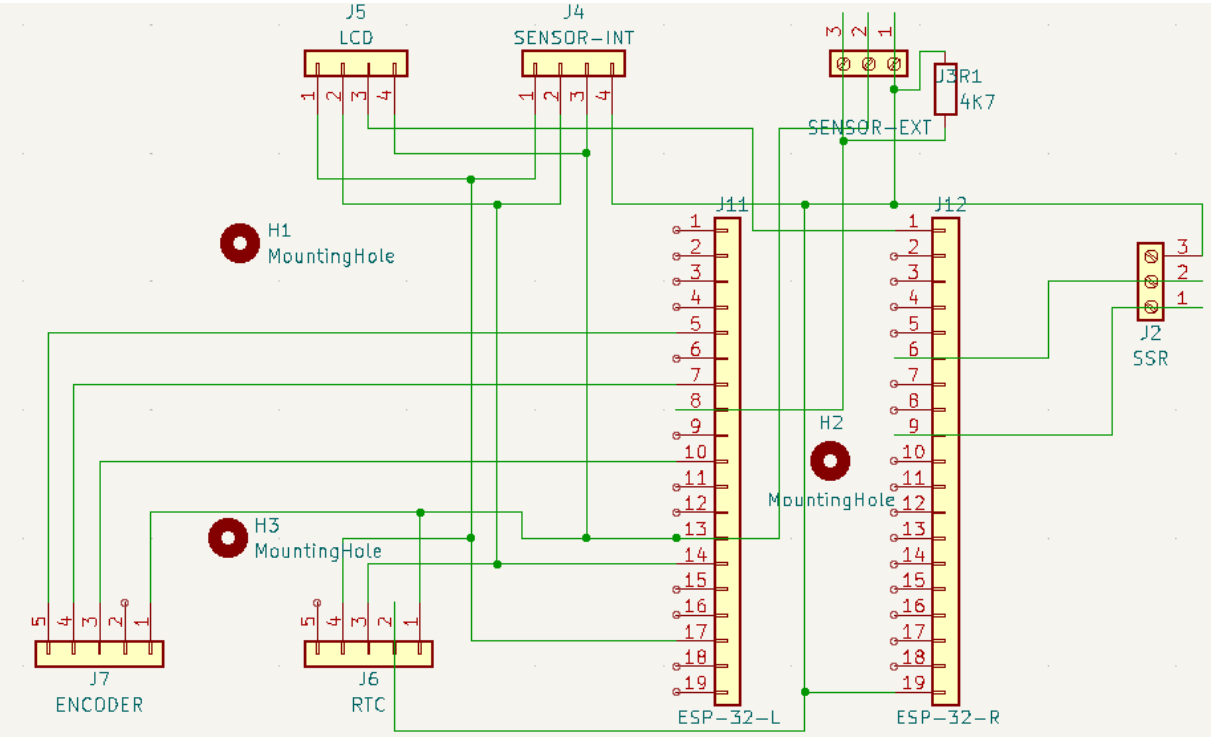
Obrázek 8 - fotka SSR

2.2 Deska plošného spoje

Pro spolehlivé spojení jednotlivých modulů je použitá na míru vyrobená PCB. Desku jsem kreslil v programu KiCad 6.0, který je obecně známý jako výborný open-source software pro navrhování plošných spojů.

2.2.1 Detaily schéma zapojení

Rozložení modulů na schéma reprezentuje první prototyp, který ještě byl na vývojovém nepájivém poli a tím pádem je pozice modulů odlišná od reálného zapojení. Další důležitá poznámka je že symboly reprezentují pouze konektory, ne moduly. Vybral jsem to takhle, protože reálná deska má taky napájené pouze konektory, do kterých se moduly zapojují (výjimka je jen u encoderu). Toto rozhodnutí bohužel znamená, že nemůžeme vidět názvy pinů.



Obrázek 9 - schéma zapojení

Při použití legendy pinů (další strana) je možné na schématu vidět, že všechna SDA a SCL jsou napojena do jednoho busu, GND vede ve dvou busech jeden pro moduly na levé straně desky druhý pro druhou stranu desky. Další důležité cesty jsou napájení rozdělené na 3.3v a 5v. Všechny moduly podporují 3.3V napájení až na LCD které vyžaduje 5V jinak poblikává podsvícení, jeho logika ale na 3.3V funguje bez problémů.

Pro piny modulů platí:

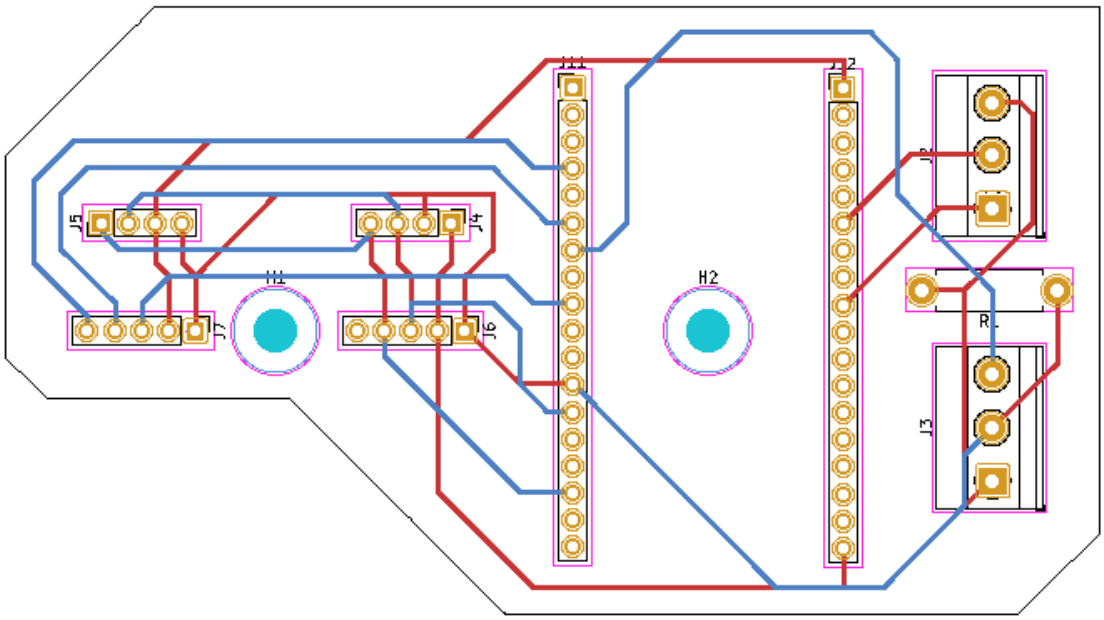
- **Encoder**
    1. **GND**
    2. **VCC+ (3V3)**
    3. **switch**
    4. **Data**
    5. **Clock**
- **LCD**
    1. **SCL**
    2. **SDA**
    3. **VCC+ (5V)**
    4. **GND**
- **RTC**
    1. **GND**
    2. **VCC+ (3V3)**
    3. **SDA**
    4. **SCL**
- **Internal sensor**
    1. **SCL**
    2. **SDA**
    3. **VCC+ (3V3)**
    4. **GND**
- **External sensor**
    1. **VCC+ (3V3)**
    2. **GND**
    3. **DATA**
- **Output**
    1. **DATA**
    2. **GND**
    3. **VCC+ (3V3)**
- **ESP32 levá strana**
    - **5 – GPIO2**
    - **7 – GPIO4**
    - **8 – GPIO16**
    - **10 – GPIO5**
    - **13 – GND**
    - **14 – SDA**
    - **17 – SCL**
- **ESP32 pravá strana**
    - **1 – VCC+ (5V)**
    - **6 – GND**
    - **9 – GPIO27**
    - **19 – VCC+ (3V3)**

2.2.2 Detaily PCB

K propojení součástek jsem už od začátku plánoval použít PCB desku. Vyskytl se ovšem problém, že vzhledem k mé neexistující zkušenosti se mi to nedařilo tak, jak jsem plánoval. Jinak řečeno jsem měl vše kontrolovat mnohem víc, než jsem kontroloval a z tohoto důvodu existují 3 verze PCB, avšak pouze 2 z nich jsem nechal vyhotovit. Všechny verze byli designované v KiCadu a vyhotovené firmou Plosnaky.cz.

Technické parametry 1. desky:

- Tloušťka desky 1.5mm
- Oboustranná
- Neprokovená
- Tloušťka mědi 0,035mm
- Šířka měděných cest 0.5mm
- Šířka desky 102,8700 mm
- Výška desky 57,1500 mm



Obrázek 10 - schéma PCB - prototyp

Na první pohled je velmi výrazná výseč na levé části desky, která je tam proto, abych mohl integrovat opěrky encoderu a RTC do pouzdra. Od tohoto návrhu jsem poté upustil, protože by bylo zbytečně náročné tak precizně vyrobit krabičku. To ale bylo až po vyrobení PCB a tudíž v návrhu výseč zůstala (na PCB je ale pouze vyznačená a nevyříznutá).

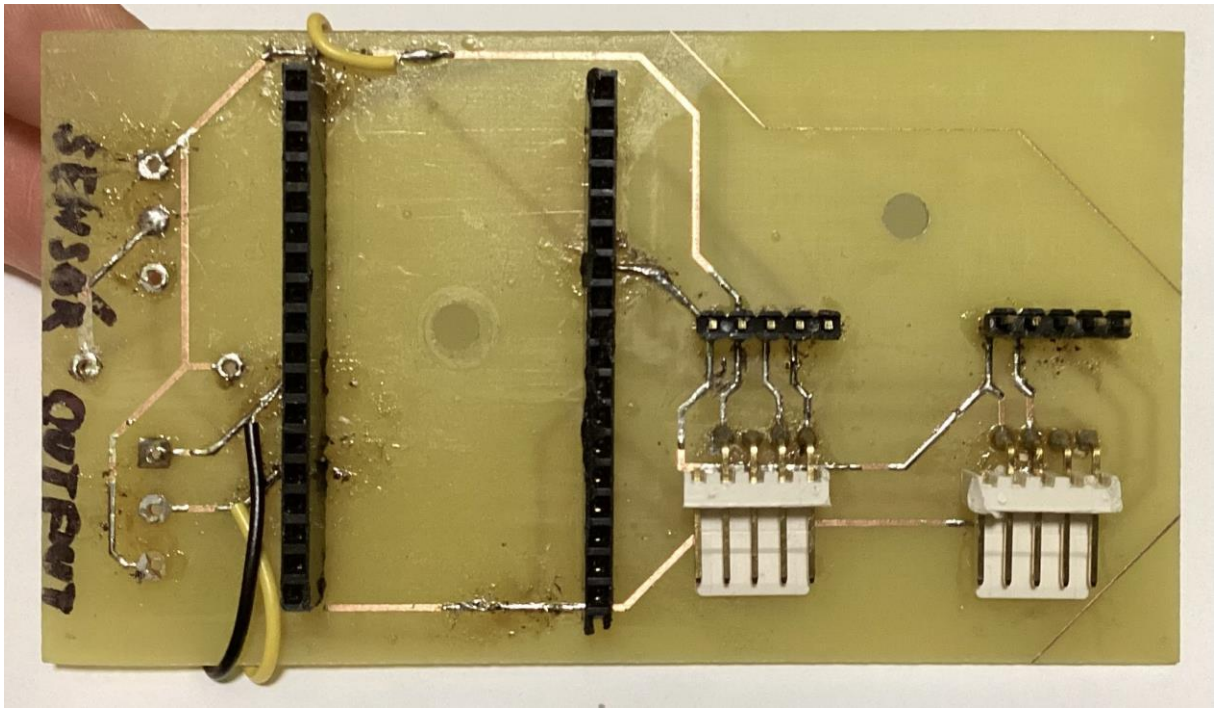
Deska je oboustranná, bez prokovení, protože ho firma neumí vytvořit. V tom spočívala moje první chyba, protože jsem si neuvědomil, že budu muset pájet součástky na obou stranách desky. Pájení na obou stranách se občas v PCB

deskách využívá, ale pouze u horizontálně upevněných součástek, ne u patic, které jsou dělané, aby dosedly rovnou na desku a z druhé strany se pevně připájely. K tomu jsou uzpůsobeny i délky jejich nožiček (asi 2mm). Aby je bylo možné připájet na obou stranách, musí se patice odsadit od desky, čímž nožičky vyčnívají asi jen o 0.1mm. To je **mnohem** míň, než je minimum pro správný spoj a způsobilo mi to řadu migrén, trápení a hlavně studených spojů. Studený spoj je spoj, na který se cín pouze přilepil za studena a nepropojil se s kovem pod zoxidovanou vrstvou. Tyto spoje mívají kontakt a občas i velmi nízký odpor, ale jsou extrémně náchylné na vše - od vlhkosti a teploty až po tlak. Jinými slovy se při měření chovají jako naprosto v pořádku a pak přes noc přestanou fungovat. Tyto spoje se objevovaly kvůli vrstvě roztaveného plastu, který jsem nechtěně odstranil z krajů patic, když jsem se k nim snažil dostat rozpáleným hrotem pájky.

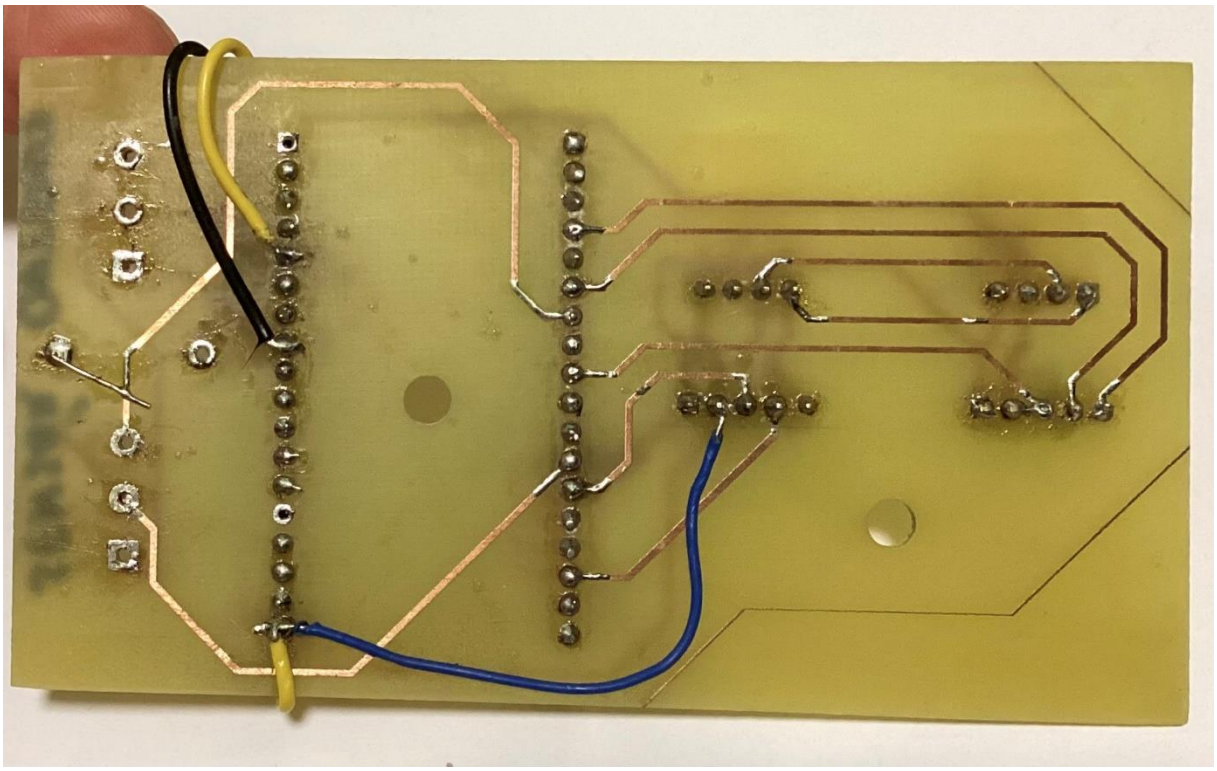
Druhý, velmi závažný problém, byl s paticí pro ESP32. Chybně jsem totiž spočítal počet spojů, kterých mělo být 19, ale na desce je jich 18. Naneštěstí jsem počítal každou stranu jinak, levou stranu zespoda a pravou seshora. To se na desce promítlo tak, že jsem si musel vybrat, kterou stranu přemostím na vedlejší (správný) pin. Rozhodl jsem se pro pravou stranu, protože měla míň spojů. Přemostění ovšem vydrželo pouze 2 měsíce, poté odpadlo a už jsem jej nikdy úspěšně nenapájel zpět.

Další problém byl v zapojení externího sensoru, ten totiž vyžaduje 10kΩ rezistor mezi napájením a daty. Já jej ale zapojil mezi napájení a zem, čímž jsem způsobil zkrat. Naštěstí byl odpor dost velký na to, aby se deska nepoškodila, a já si tudíž mohl této chyby všimnout až po dvou týdnech.

Bohužel deska měla ještě další problémy: měděné plošky byly moc malé a odpadávaly, cesty hlavních busů SCL a SDA byly propojené přes všechny součástky a střídaly vrstvy, takže když se objevil studený spoj na kterémkoli spoji, tak celá sběrnice vypověděla službu. Díry byly podle palcové mřížky, ale CAD program jsem měl nastavený na metrické jednotky. Tudíž rozměry nebyly přesné a ve skutečnosti deska neseděla v pouzdře.



Obrázek 11 - fotka PCB – prototyp zepředu



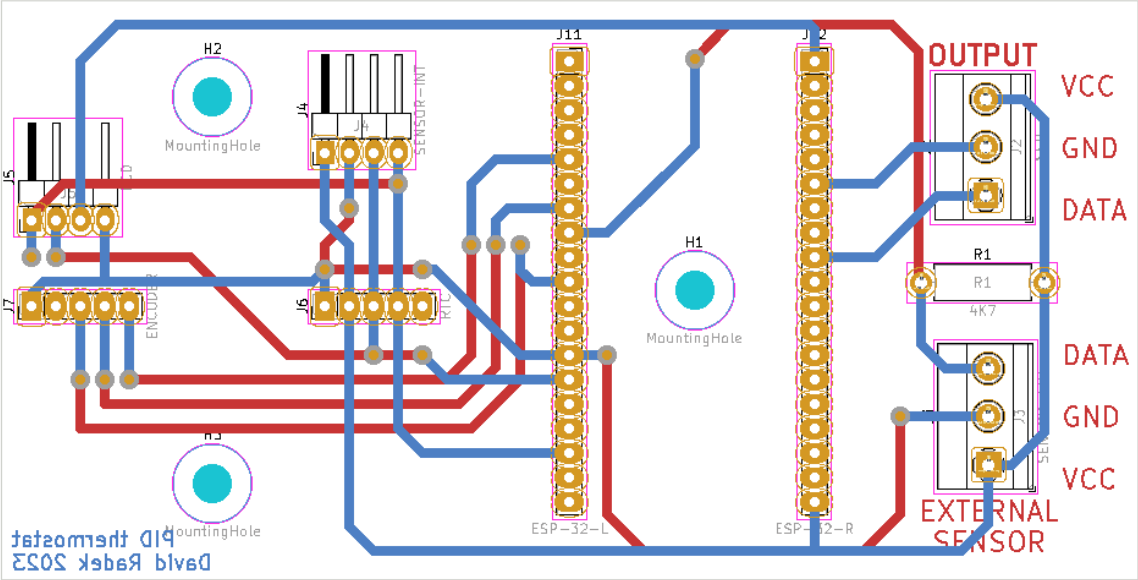
Obrázek 12 - fotka PCB - prototyp zezadu

Postupně jsem desku upravoval a spravoval, ale nikdy se mi nepovedlo ji uvést do stabilního stavu, kdy bych si mohl být jistý, že ji při příštím spuštění nebudu muset znovu přepájet (SCL a SDA busy jsem přepájel 9x, proto taky jejich měděné

plošky odpadly). Později jsem se tedy rozhodl, že vytvořím novou desku, která bude lepší než kterákoli před ní.

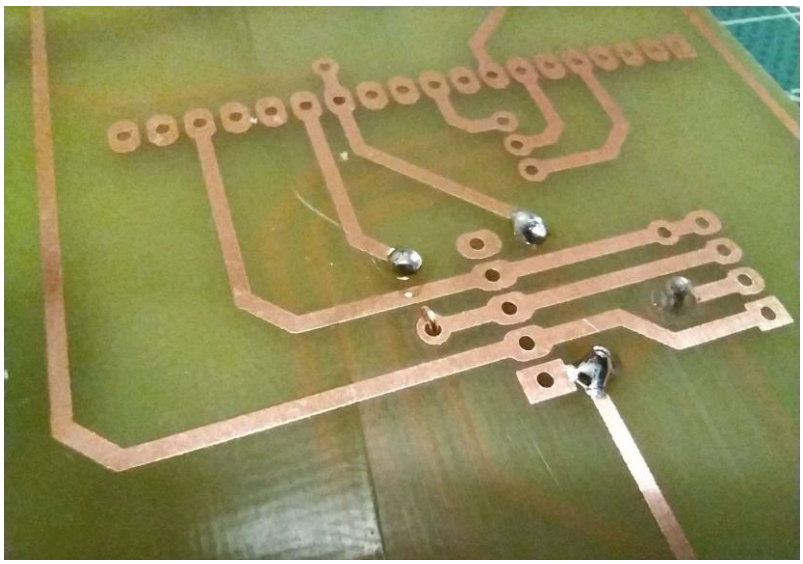
Technické parametry 2. desky:

- Tloušťka desky 1.5mm
- Oboustranná
- Neprokovená
- Tloušťka mědi 0,035mm
- Šířka měděných cest 1mm
- Šířka desky 118,0000 mm
- Výška desky 60,000 mm



Obrázek 13 - schéma PCB - finální

První, čeho se dá na novém designu všimnout, je jeho velikost. Desku jsem zvětšil, moduly odsadil o 2.54mm a na pravou stranu přidal popisy výstupů. Další důležitá změna je způsob jak přemostit cestu na druhou stranu. K tomu jsem použil tzv. prokovené díry (podle KiCadu), které



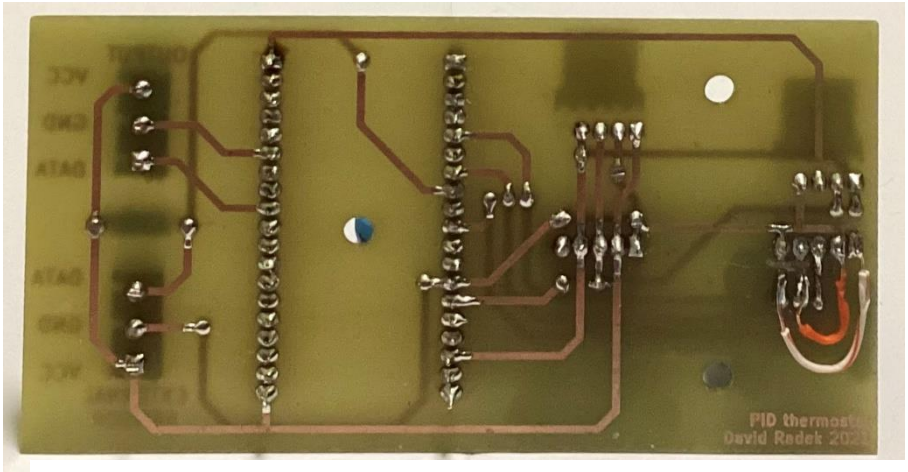
Obrázek 14 - fotka PCB - propojení vrstev

samozřejmě prokovené nebyly, ale daly se využít jako průchod pro malý drátek, který jsem připájel z obou stran (viz obrázek 14). Tímto způsobem jsem mohl velmi jednoduše vytvořit prokovené spoje s minimálním odporem a možností velmi dobře propájených spojů. Dále jsem zvětšil šířku cest (což je učinilo odolnější), zvětšil jsem také měděné plošky a změnil jejich tvar na oválný a snažil jsem se šetřit s množstvím přechodů mezi plochami.

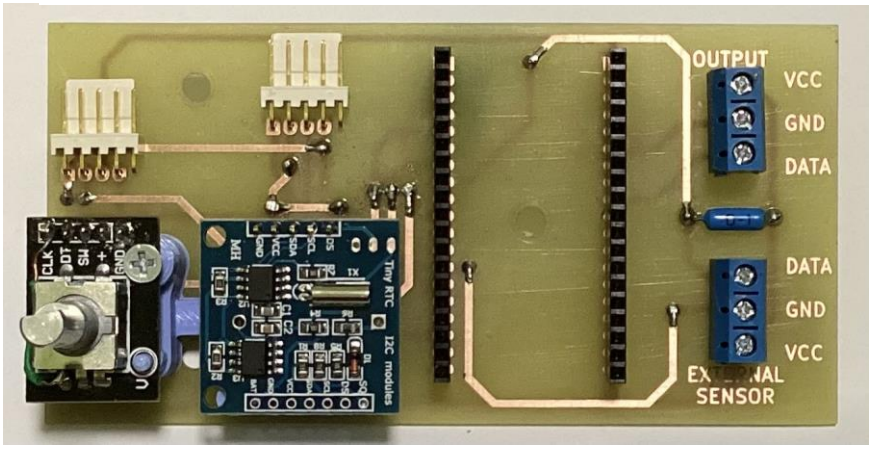
Avšak deska ani potom nebyla bez chyb.

V KiCadu je možnost pojmenovat konektory v jejich názvu (zobrazuje se pouze číslo). Tuto funkci jsem při designu první desky využil, a proto jsem napojil moduly z většiny správně. Ale u této desky jsem tak neučinil. Bez informace jakým směrem je modul otočený jsem je následovně zapojil naopak… Naštěstí jsem byl schopný desku opravit jen s malou úpravou cest pod encoderem. RTC jsem jen otočil hlavou dolů a kabely k LCD a internímu sensoru jsem nachystal v opačném zapojení.

Tohle byla jediná chyba desky a po opravě jsem se rozhodl ji použít jako finální. A jako malý bonus jsem ji popsal názvem projektu, svým jménem a rokem zhotovení.



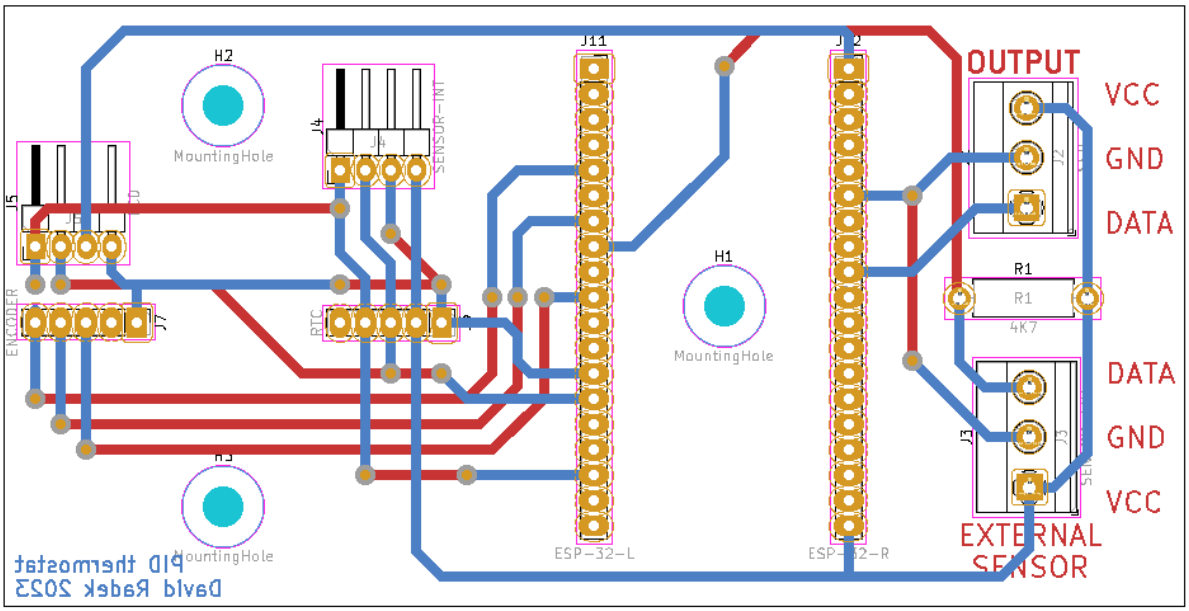
Obrázek 15 - fotka PCB - finální zezadu



Obrázek 16 - fotka PCB - finální zepředu

Technické parametry 3. desky:

- Tloušťka desky 1.5mm
- Oboustranná
- Neprokovená
- Tloušťka mědi 0,035mm
- Šířka měděných cest 1mm
- Šířka desky 118,0000 mm
- Výška desky 60,000 mm



Obrázek 17 - schéma PCB - návrh

Poslední návrh desky jsem nikdy nevyrobil. Tuto desku jsem navrhl ihned po lokalizaci chyby druhé desky a slouží pouze jako návrh finální desky bez chyb. Technické parametry jsou totožné s 2. návrhem avšak zapojení se v některých místech liší, například externí senzor je napojen na zem výstupu, abych jej nemusel vést na levou stranu desky. Dala by se dál zoptimalizovat, kdybych změnil piny zařízení na příhodnější, ale pro to bych musel přepsat kód a to nebylo cílem této desky.

2.3 Pouzdro

Aby se komponenty nepoškodily a zařízení bylo možné připevnit ke zdi, musí být umístěno v pouzdře. To jsem vymodeloval v programu Autodesk Inventor 2017 a vytisknul na své 3D tiskárně Creality Ender pro v2. Pouzdra jsem vytvořil dvě. První byl prototyp s mnoha problémy, ze kterých jsem se ale poučil a byl tak schopný vytvořit mnohem lepší, druhou verzi.

2.3.1 3d Modelování

Program Inventor 2017 je placený cadový program od společnosti Autodesk, který mám jako student zdarma na 4 roky. Bohužel se modely tvořené v něm, nedají otevřít v jakékoli jiné verzi, a tak jsou v přílohách uloženy ve formátu stl. Jako většina CAD programů se v Inventoru modeluje pomocí tzv. sketchů, což jsou 2D nákresy tělesa, které se poté vytáhnou do 3. dimenze pomocí jednoduchých nástrojů.

2.3.2 3D tisk

Tiskárna Ender 3 v2 kterou jsem použil, byla při tisku vybavená 0.4 mm tlustou tryskou. Samotný materiál, ze kterého je pouzdro vyrobeno je pak prusament PLA, který jsem dostal k tiskárně zdarma. K přípravě g-kódu (formát souboru který 3dTiskárna používá jako instrukce při tisku) jsem využil slicer Cura 5 ve kterém jsem nastavil následující parametry:

- bottom_layers = 2
- brim_width = 6
- infill_line_distance = 4
- infill_multiplier = 2
- infill_pattern = lines
- infill_sparse_density = 10
- line_width = 0.5
- material_print_temperature = 185.0
- raft_margin = 0.5
- speed_print = 80.0
- support_angle = 75.0
- support_tree_angle = 30
- top_bottom_thickness = =layer_height_0+layer_height*3
- top_layers = 2
- wall_thickness = 1
- z_seam_corner = z_seam_corner_inner
- z_seam_position = backleft
- adhesion_type = brim
- layer_height = 0.35

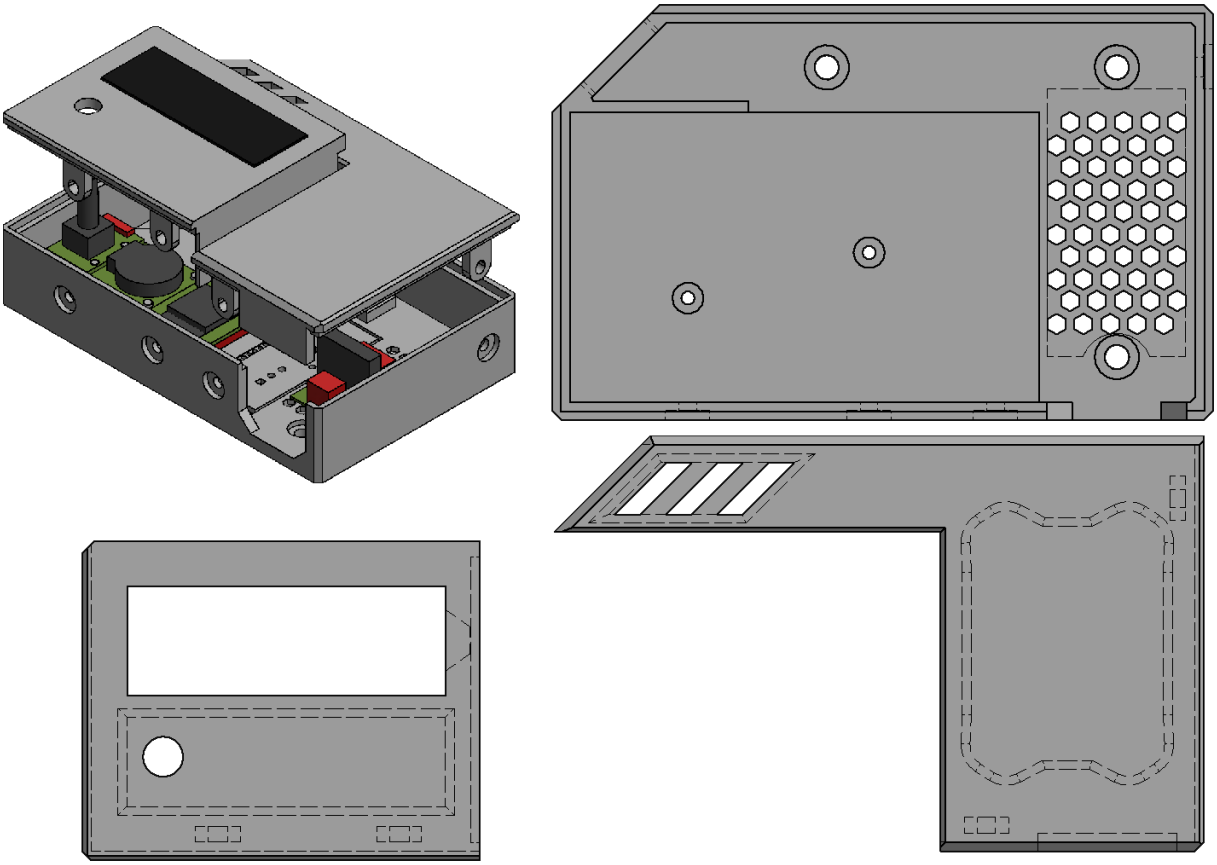
- material_bed_temperature = 40
- material_bed_temperature_layer_0 = 50
- smooth_spiralized_contours = False
- support_enable = True
- support_structure = tree

Tyto nastavení mám dobře odzkoušené na jiných výtiscích a vím, že se tisk s nimi povede. Zároveň je ale velmi rychlý se špatnými detaily (což v tomto případě nevadilo).

2.3.3 Design

Prototyp

Tato verze pouzdra mi sloužila jako zkušební vzorek, na kterém jsem si otestoval rozměry a design.



Obrázek 18 - náhled pouzdra - prototyp

Měla tvar obdélníku, se skoseným krajem. Tento kraj byl vyříznutý čistě z estetického hlediska. Zároveň jsem tuto stěnu proděravěl, aby se tato oblast řídila teplotou okolí a mohl v ní být umístěný senzor teploty. Níže, v levé spodní části pouzdra, byla umístěna PCB a všechny komponenty nutné pro funkčnost programu. Celá pravá strana je pak vyhrazena pouze pro SSR a kabeláž. Díry ve

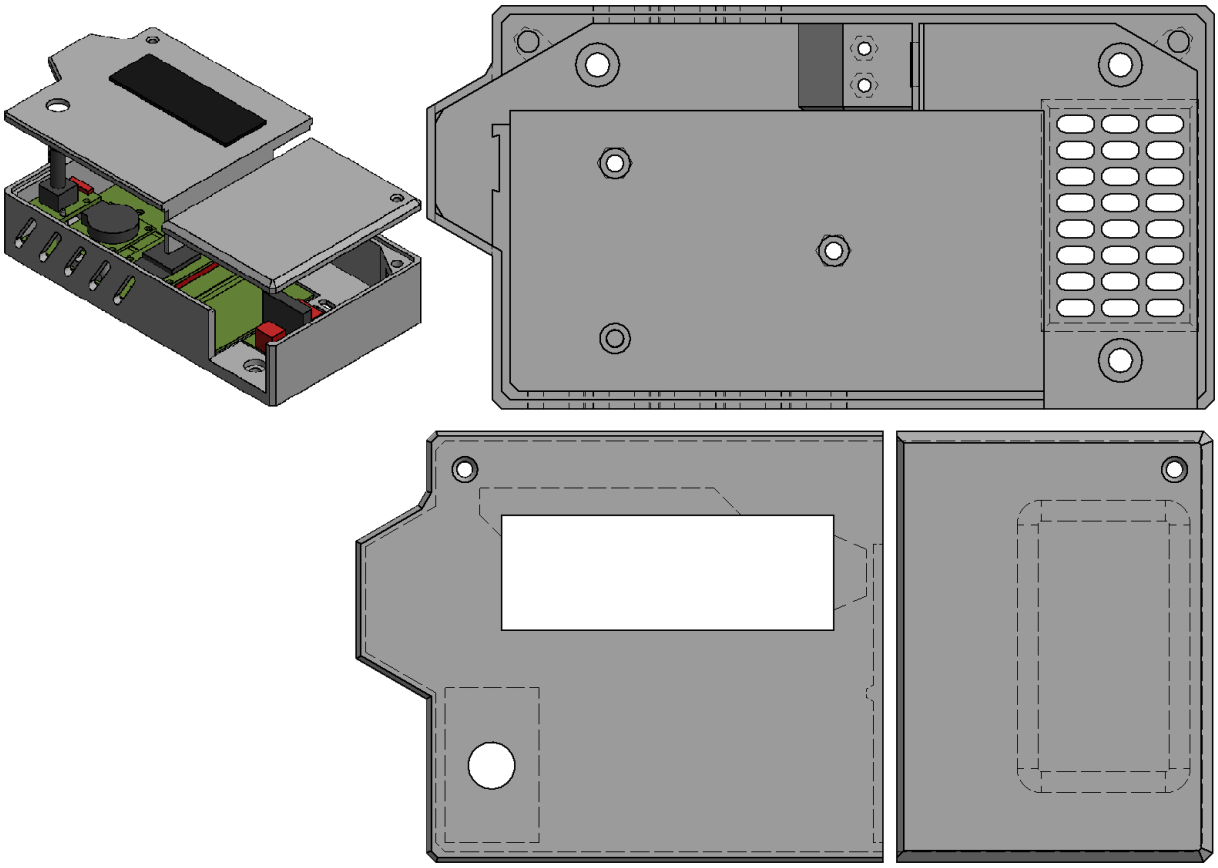
dnu pouzdra byly navrženy tak, aby se pomocí nich připevnilo co největší množství jednotlivých relé s různými roztečemi. Zároveň bylo v této oblasti dno vyříznuto ze zadní strany, aby se dala relé upevnit pomocí matek. Po dně jsou rozmístěné 3 M5 díry s místem pro zapuštění hlavy, aby se dalo zařízení připevnit na zeď. Na stěnách hlavní části jsou nachystané díry, které sloužily k upevnění krytů pouzdra. Ty mají pak na patřičném místě výstupky s oválnou dírou, aby kompenzovaly nepřesnost až 2 mm do obou směrů. První z těchto krytů je umístěn nad oblast s komponenty a byl koncipován tak, aby ho nebylo potřeba odebrat při případné manipulaci s kabeláží. Má v sobě dva výřezy. První je pro encoder, který by ale byl moc vysoký a nevlezl by se pod kryt. Proto je kolem něj kryt o pár milimetrů zúžený. Druhý výřez je pro display. Na pravé straně je ale také zúžení, kvůli podsvícení displeje. Pro překrytí zbývající oblasti pouzdra slouží třetí část, která připomíná velmi tlusté písmeno L. Tento tvar ji napomáhá upevnit a vytváří zajímavý vzhled. Na levé straně jsou další otvory pro senzor teploty a pravá strana je zúžená kvůli výšce SSR, které je umístěno ihned pod ní.

Nedostatky krabičky:

- Senzor se zahřívá od ESP32
- ESP32 se přehřívá
- Samořezné šrouby, které měly krabičku držet pohromadě, plast nevydrží.
- Nedostatek místa pro kabeláž
- Nedá se připevnit USB pinout modul
- Nevleze se do ní nová PCB deska

Pouzdro „Collosus“

Spolu s touto verzí pouzdra vznikla i nová PCB deska, která byla o asi 20 mm širší. Z tohoto důvodu je nové pouzdro větší. Zároveň pouzdro vyřešilo všechny nedostatky prototypu.



Obrázek 19 - náhled pouzdra - finální

Přehřívání senzoru jsem vyřešil oddělenou komorou na levé straně. Dál je nové i místo dedikované pro pinout modul a speciální přepážka tvořená ze dvou částí na krytu displeje a základu krabičky, která oddělila komoru pro SSR a kabeláž ještě o něco víc. Kryty jsou nově upevněné pomocí m2 šroubů, které se zašroubují z čela krabičky a jsou upevněny pomocí matek vlepených do předpřipravených komor ve zdech. Dále jsou optimalizované výřezy kolem displeje a encoderu (v pozdější fázi vývoje jsem musel encoder vyměnit za nový a ten už výřez nepotřebuje, v krabičce ale už zůstal). Obecně je tato krabičky mnohem pevnější a kvůli větracím dírám ve stěnách dole a nahoře se v ní ESP32 nepřehřívá. Dále jsou v krabičce menší změny, jakožto například výřezy pro matky ve dně, aby se daly PCB a pinout modul jednoduše upevnit.

2.4 PID-On/Off controller

PID je zkratka, která obsahuje jména všech 3 složek PID controlleru. Těmi jsou:

- Proporcionální
- Integrální
- Derivační

Všechny tři využívají neznámou err což je předpřipravený výpočet rozdílu od požadované hodnoty (požadovaná hodnota – aktuální hodnota).

Proporcionální výpočet je z těchto tří nejjednodušší. Jeho úkolem je předvídatelně se přibližovat k cíli přímo úměrnou sílou k jeho vzdálenosti. Jednoduše řečeno, čím dál je od cíle, tím víc se bude snažit k němu dostat. Jeho implementace v mém programu je znázorněna níže.

Integrální výpočet je složitější. V PID controlleru zajišťuje, aby se nikdy nezasekl. Jednoduše se to ukazuje na příkladu s kotlem na teplou vodu. Při použití jen Proporcionálního výpočtu se controller přiblíží k požadované hodnotě na tolik, že už nebude mít dostatečnou sílu překonat tendenci vody, vyrovnat svou teplotu s teplotou okolí. Tudíž se controller nikdy k požadované teplotě nedostane. Integrální výpočet ale vytváří tzv. kumulační rozdíl což je neznámá, do které každých x sekund přičítá rozdíl od požadované hodnoty. Pokud se tedy controller zasekne na dlouhou dobu kus od požadované hodnoty tak se k ní integrální výpočet bude snažit přibližovat čím dál silněji, až se mu to konečně podaří a dané hodnoty dosáhne. Nevýhoda integrálního výpočtu ale je jeho nestabilita. Může se stát, že vypadne proud, ale termostat bude dál fungovat. V tom případě se bude integrální výpočet pořád zesilovat a až proud zase naskočí tak způsobí přestřelení teploty o velký kus.

Proto využívám ve svém programu tzv. inteligentní integrální výpočet, který se liší následujícími vlastnostmi:

- Kumulativní výpočet nikdy nepřesáhne maximální výkon controlleru
- Integrální výpočet bude ignorován, pokud by měl zabraňovat ostatním výpočtům ve funkci (aktivně se vzdalovat od požadované hodnoty).
- Kumulativní výpočet počítá s časem (násobí výpočet počtem uběhlých milisekund)

Jeho implementace v mém programu je znázorněna níže.

Derivační výpočet zjišťuje, jak rychle se hodnota blíží k požadované. To vypočítá pomocí vzorce (err - last_err)/(time_passed). Err je aktuální rozdíl od požadované

hodnoty, last_err je naposledy naměřený rozdíl a time_passed je čas který uběhl od posledního měření.

Tyto komponenty jsou poté spojeny ve funkci **MyPID** která vypadá následovně:

```
float MyPID(float currPos, float target, float Kp_input, float Ki_input,
float Kd input, float maxStrength, bool printOut) {

  float err = target - currPos;

  float maxStrengthPos = maxStrength;
  float maxStrengthNeg = -maxStrength;

  float thoreticalPID_cumulativeERR = PID_cumulativeErr + (err / (millis()
- PID_lastMillis)) * Ki_input;

  if (thoreticalPID_cumulativeERR < maxStrengthPos &&
thoreticalPID_cumulativeERR > maxStrengthNeg) {
    PID_cumulativeErr = thoreticalPID_cumulativeERR;
  }

  float P = err * Kp_input;

  float I = PID_cumulativeErr;
  if ((I * err) < 0) {
    I = I * 0.1;
  }

  float D = ((err - PID_lastErr) / (millis() - PID_lastMillis)) * Kd_input
* 1000;


  if (printOut) {
    Serial.print(",P:" + String(P) + ",I:" + String(I) + ",D:" +
String(D));
  }

  PID_lastErr = err;
  PID_lastMillis = millis();
  return constrain((P + I + D) * Km, maxStrengthNeg, maxStrengthPos);
}
```

Jako prvních bych vysvětlil jaké vstupní hodnoty má.

- currPos je aktuální hodnota
- target je cílová/požadovaná hodnota
- Kp_input je násobič proporcionálního výpočtu
- Ki_input je násobič integrálního výpočtu
- Kd_input je násobič derivačního výpočtu
- maxStrength je maximální výstupní hodnota controlleru
- printOut řídí, jestli funkce vypíše své aktuální hodnoty přes sériové linky.

maxStrength je uschovaná do dvou neznámých pozitivní a negativní, kvůli bugu ve funkci constrain (není to ani moc bug, bohužel tato funkce není funkce ale tzv. makro a ty se chovají trochu jinak). Lze si také všimnout přidaných násobičů na Integrálním a derivačním výpočtu. Integrální jej má aby byl stabilnější (snížil se

tím jeho výkon) a derivační jej má aby se počítal každou sekundu (je vydělen milisekundami, což mu 1000-krát snížilo sílu).

Ne vždy je PID nejlepší způsob jak regulovat teplotu. Nevýhodou by mohla například být jeho závislost na analogovém výstupu. Tzn., že výstup nemůže být jen vypnuto/zapnuto, protože PID controller bude propouštět energii nonstop (aby automaticky kompenzoval chladnutí). Proto zařízení obsahuje dva controllery spojené do jednoho, které se přepínají podle nastavení v bajtu PIDsettings. Druhý controller je jednoduchý On/Off. Ten, má zabudovanou podporu hystereze, která umožňuje uživateli nastavit rozsah hodnot, které se berou jako cílové/požadované. Nevýhoda On/Off je že většinou přežene svůj výkon a tzv. „přestřelí“ požadovanou hodnotu.

2.5 Program

Mozek, ve kterém program běží, je v tomto zařízení deska ESP32. Tato deska má procesor s taktem 2.4Ghz, programovatelnou flash paměť s kapacitou 4MB a integrovanou Wi-Fi a bluetooth. Když se porovná s Arduinem uno (často využívaným programovatelným mikroprocesorem) tak je hned vidět že ESP32 je mnohem výkonnější. Její výkon mi umožnil vytvořit příjemné uživatelské rozhraní s rychlou odezvou. Tento modul má integrovaný i automatický bootloader, který zjednodušuje práci s mikroprocesorem. Pro tohle nahrávání jsem využil program Arduino IDE 2.0 ve kterém jsem zároveň kód psal. Program je napsaný v upravené verzi jazyka C, který je jediný podporovaný jazyk Arduino IDE. Při psaní programu jsem se snažil šetřit s prostředky, aby jej bylo možné spustit i na slabším hardwaru. Proto jsou některé části programu vysoce optimalizované.

Program se dělí do 3 souborů:

- **Main** - obsahuje kód pro všechny „stránky“ user interface, deklarace proměnných a kód pro různé periodické funkce.
- **Complementary functions** – obsahuje funkce pro všechny handlery, PID controller a rendering.
- **A0 symbols** – obsahuje deklarace mnou vytvořených symbolů

2.5.1 Struktura kódu

Program je napsaný tak, aby se opakovala funkce loop(), každému opakování říkám cyklus. Toto řešení ale vyžadovalo použití globálních proměnných, protože všechny proměnné vytvořené v jednom cyklu se smažou dřív, než se spustí další cyklus. Abych ušetřil paměť, kterou si každá globální proměnná zarezervuje jenom pro sebe, tak jsem vytvořil několik tzv. Cache. Každá je pro určitý datový typ a slouží k přeposílání informací do dalšího cyklu. Při psaní jsem si ale musel dávat pozor, abych nepoužil stejnou Cache pro dvě věci najednou, protože by to mohlo tvořit nepředvídatelné „bugy“. Některé Cache se používají nezávisle na currentMenu a tudíž se nesmí používat (platí například pro 1, 5 a 60 sekundové časovače).

Seznam obecně používaných cache:

- 3x bool
- 2x unsigned long
- 1x unsigned int
- 5x 64bit int
- 1x int
- 1x char
- 2x String

Výstup pro uživatele je zařízen pomocí stránek. Stránka je kód, obsažený ve společném přepínači v hlavním souboru Main.ino. Tento přepínač se ovládá pomocí globální proměnné currentMenu, která obsahuje integer, reprezentující ID stránky, která se zobrazí v příštím cyklu.

```
// základní struktura stránky
case <ID stránky> :
      {

        drawFrames = true; // použij vylepšené vykreslování

        if(editorjustFinished){ // odchytávač výsledků z editoru
          editorjustFinished = false;
          switch(editorIntCache){
            case 1:{
              //odchyt výsledku podle id výstupu editoru
              break;
            }
          }
        }

        if(encoderSwitchHoldOnce){
          break;
        }

        if(encoderSwitch){
          break;
        }

        break;
      }
```

Každá stránka má svůj vlastní index. Dál má na výběr, jestli chce používat vylepšené renderování. V případě, že využívá editor, tak je ještě potřeba jeho výsledek odchytnout a zpracovat. Na konci každé stránky se pak dají nastavit funkce pro encoderSwitchHoldOnce (stisk a podržení encoderu) a encoderSwitch (stisk encoderu). Většinou by mělo platit, že se při stisku potvrdí/nastaví hodnota a při podržení se uživatel vrátí zpět. To se liší pouze při vytváření a upravování plánovaného nastavení. Při použití jedné z těchto funkcí by se měla stránka

ukončit a program by se měl posunout na další cyklus. Všechny stránky by měly být napsané uvnitř přepínače stránek, s výjimkou překrytí.

Překrytí jsou speciální stránky, které se spouští prioritně nad všemi běžnými. Program obsahuje následující 3 překrytí: sleepMode, popup a editor. Může dojít k vykreslení dvou překrytí přes sebe. Dále se postupuje podle přednastavené priority, která je ve stejném pořadí, jak jsou vypsány. Překrytí by neměla používat Cache, protože není jisté, jestli tuto Cache nepoužívá stránka pod překrytím (stránky jsou pozastaveny, ale zůstávají vybrané v přepínači, tudíž by je mohla náhlá změna v odkladné paměti poškodit). To ale neplatí u editoru, který je velice odlišný od ostatních překrytí.

SleepMode, neboli úsporný režim, je implementovaný kvůli vylepšenému vykreslování, které vyžaduje vysoký výkon zařízení a tím vytváří uvnitř krabičky teplo. To následně i přes izolované prostředí zahřívá senzor a zkresluje regulaci teploty až o 3° Celsia. Aby se tomuto předešlo, tak se ve sleepModu překresluje obrazovka podle 5-ti sekundového časovače. Do tohoto módu zařízení přejde po jedné minutě neaktivity. Po stejné době se vypne podsvícení displeje, což také napomáhá nechtěnému zahřívání.

Popup je jednoduché upozornění, které může uživateli oznámit chybu nebo jen zpětnou vazbu na jeho akci. Toto upozornění je schopno zobrazit jakkoli dlouhý text i v případě, že by se text na obrazovku nevešel. Je-li tomu tak, jednoduše posune o 1 znak za půl sekundy. Až se tímto dostane na konec, posune ještě text o 3 prázdné pole pro snazší čitelnost a poté se vrátí na začátek textu. Tento proces se opakuje, dokud uživatel upozornění nevypne pomocí tlačítka OK.

Editor je překrytí s nejnižší prioritou, ale nejdůležitějším využitím. Umožňuje uživateli upravovat string pomocí jednoduchého ovládání: otáčením encoderu vybere znak, stiskem přejde do upravovacího módu, otáčením encoderu v tomto módu prochází možnostmi pro tento znak a stiskem jej potvrdí. Pro ukončení editoru uživatel přidrží tlačítko stisknuté. Handler (viz níže) pro editor má mnoho vstupů:

```
editorHandler(String variableValue, String variableName, int inputLength,
char placeHolderChar, String useableCharacters, int inputID)
```

1. počáteční hodnota upravovaného stringu
2. jméno hodnoty, která se zobrazí na displeji při upravování
3. maximální délka výstupního stringu
4. ascii hodnota znaku, kterým bude vstupní string doplněn, kdyby nebyl stejně dlouhý jako výstupní string
5. string obsahující všechny možné vstupní hodnoty
6. id výstupu, podle kterého stránka pozná, kterou hodnotu v minulém cyklu uživatel pomocí editoru nastavil.

Pozn., první znak nesmí být symbol. Jako kurzor ale využívám jeden ze symbolů. Proto je první znak na 2. řádku „=“ které odsadí zadávání právě o jeden znak.

Editor je ale pouze nástroj na úpravu stringů, tudíž samotné nastavení upravované hodnoty musí zprostředkovat stránka v sekci odchytávání výsledků.

Při přepínání mezi stránkami, či spouštění překrytí, je někdy potřeba nejdřív nachystat některé Cache, které daná stránka bude potřebovat. Na to slouží tzv. **Handlery**. Všechny handlery jsou uložené v souboru ComplemetaryFunc.h.

Ukázka Handleru pro menu nastavení teploty:

```
void tempSettingHandler(int sourceMenu) {
  currentMenu = 101;                //menu of temperature settings
  int64Cache = encoder.getCount();  //stores value of encoder before change
  int64Cache2 = sourceMenu;         //stores last menu to retun to (useless
in this case but it could fix some bugs)
  encoder.setCount(targetTemp * -2);
}
```

2.5.2 Vedlejší funkce

Ovládání zařízení je umožněno pomocí encoderu. Ten ale má pouze 3 možné akce: otočení po směru hodinových ručiček, otočení proti směru hodinových ručiček a zmáčknutí tlačítka. Abych rozšířil jeho možnosti, vytvořil jsem část kódu, která používá vstup z tlačítka a převádí jej na tři různé stavy, které ukládá do tří neznámých s hodnotou True nebo False:

- switchEncoder
- v tomto cyklu uživatel pustil tlačítko po tom, co jej krátce stiskl
- switchEncoderHold
- drží hodnotu True pokud uživatel drží tlačítko déle než 500 milisekund. V momentě, kdy uživatel tlačítko pustí, se vrátí na False
- switchEncoderHoldOnce
- v tomto cyklu uživatel začal držet tlačítko po dobu déle než 500 milisekund
- Tato hodnota může být znovu True až po té co uživatel tlačítko pustí a znovu jej podrží na 500 milisekund a déle.

Enum je integer, který má ke každé hodnotě přiřazený string. Bohužel tento datový typ v Arduinu chybí a tudíž jsem si musel vytvořit svoji variantu, kterou jsem nazval **getStringFromLib**

```
String getStringFromLib(int index, int_fast16_t nameArrayIndex)
{  //essentially mimics enum by letting you create an array of strings and
returning a string on given index

  switch (nameArrayIndex) {
    case 0:
      {
        String premadeNameArray[] = { "HEATER", "COOLER" };
        if (index > sizeof(premadeNameArray) || index < 0) {
          return "err1";
        } else {
          return premadeNameArray[index];
        }
      }
```

Tohle je kód, který jsem použil v nastavení hodnot. Jedná se o překlad hodnot 0 a 1 do stringů HEATER nebo COOLER. Tento systém ale umí až 256 možných názvů, které si člověk předdefinuje v této funkci a poté je jen zavolá pomocí indexu knihovny a čísla, které chce přeložit. Funkce mu pak vrátí příslušný string.

ConstrainEncoder je funkce, která umožňuje převést hodnotu encoderu na přepínač s daným rozsahem pozic. Tato funkce je potřeba z několika důvodů:

- Encoder počítá otáčky naopak (doprava je mínus) takže je potřeba hodnotu otočit
- Encoder počítá i do mínusu, což znemožňuje jednoduše použít zbytek po dělení (pozice by se vydělila počtem požadovaných pozic a zbytek by se nastavil v rozsahu mezi nulou a počtem pozic-1)
- I kdybych převedl hodnotu na unsigned integer (číslo které může být pouze v plusu) tak by byl problém se zbytkem po dělení kvůli převodu negativního čísla do plusu

Finální řešení jsem sestavil s pomocí pana Jílka (učitele C++). Toto řešení umí kompenzovat všechny nedostatky a používá se jednoduše pomocí funkce **ConstrainEncoder**(počet pozic). Případně lze nastavit jinou vstupní hodnotu funkcí **ConstrainEncoder**(počet pozic, speciální vstupní hodnota).

ProcessAutoMode je jednoduchá funkce která kontroluje, jestli není potřeba načíst do paměti nový plánovaný záznam. V případě, že tomu tak je, jej načte, případně upraví mód, v jakém se zařízení nachází (auto/manual).

scrollComponent a **scrollComponentspecific** jsou funkce, které zůstaly v programu po přepsání, které proběhlo hned na začátku vývoje. Původní myšlenka byla, že „ui“ bude tvořeno z tzv. „komponentů“, které by se volaly pomocí funkcí. Z tohoto záměru jsem později upustil a zůstal pouze tento komponent. Zobrazuje šipky na straně displeje, které uživateli značí, jestli má ještě kam scrollovat. Specific verze funkce se liší pouze v tom, že zabírá jen jeden řádek. Pozice šipek je daná na posledním znaku v řádku.

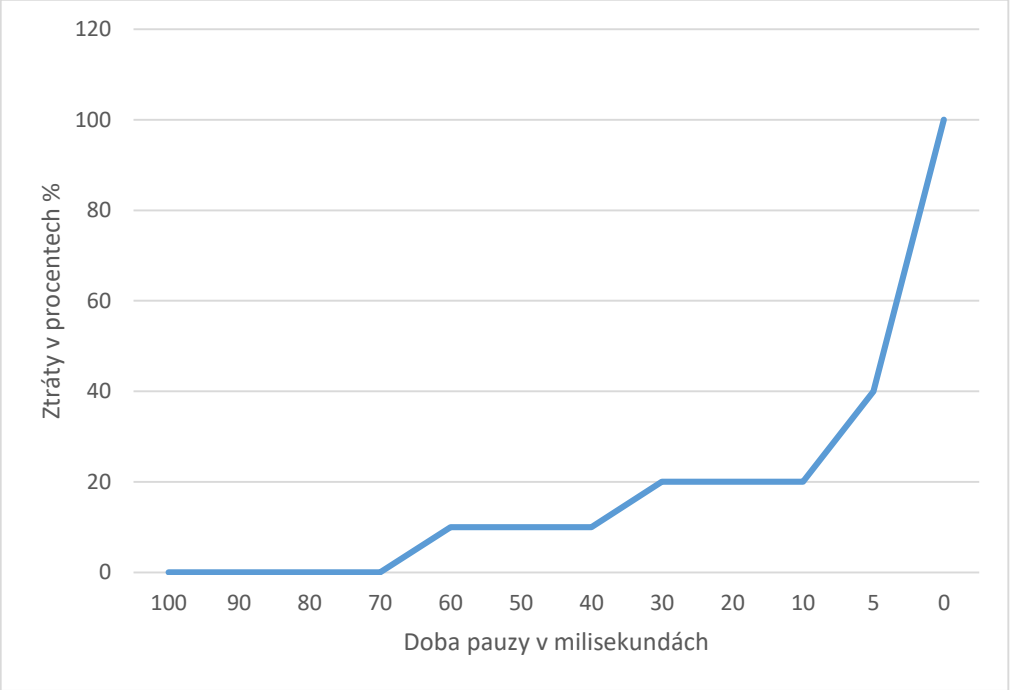
2.5.3 Flash paměť

Pro trvalé ukládání hodnot používám flash paměť ESP32, kterou má již předinstalovanou v sobě a knihovnu Preferences. Tato knihovna mi dává možnost spustit tzv. namespace v paměti v režimu ReadOnly nebo Read-Write. Každý namespace má oddělené hodnoty od ostatních namespace, aby mohl vývojář použít jedno ESP32 na více projektů bez konfliktů v paměti.

Čtení a zápis se poté řídí podle klíčů, což jsou stringy, pod kterými se ukládá hodnota do paměti. Rozdílné datové typy se dají ukládat pomocí připravených funkcí putFloat, putInt a podobné. Zbylé datové typy, které nemají speciální funkci, se dají uložit pomocí putBool(pointer na hodnotu, počet bajtů které hodnota má). Z paměti se dají vyčíst pomocí funkcí getFloat, getInt apod. stejným způsobem.

Bohužel jsem při vývoji zjistil, že flash paměť není konzistentní v čase, jak dlouho jí trvá požadovanou hodnotu získat. Tudíž se občas hodnota načte za 1 milisekundu a občas až po 60 milisekundách. Proto při každém vyčtení z paměti pozastavím na předdefinovanou dobu celý program a tím dám paměti čas na navrácení hodnoty. Každá tato pauza velmi zpomaluje program, takže jsem se ji snažil optimalizovat na co nejkratší dobu, což jsem nakonec pomocí jednoduchého programu nastavil na 10 milisekund. Program, kterým jsem testoval jaká prodleva bude nejbezpečnější, jen četl hodnotu z paměti, a poté ji zkontroloval. Pokaždé zopakoval tento proces 10x a poté snížil dobu pauzy.

Výsledky z programu vypadaly následovně:



Jak je vidět z grafu, tak je 10 milisekund přesně na hranici, kde se ještě hodnoty většinou zadaří odchytit. V reálném použití jsem zaznamenal ztráty kolem 5%. Abych předešel možným problémům, tak jsem do některých hodnot zakomponoval redundantní bity a checksumy.

2.5.4 Rendering

LCD displej má velmi příjemnou a jednoduchou knihovnu, která programátorovi pomůže velmi jednoduše posunout kursor na displeji a zapsat z toho místa string. Dál pak ještě podporuje jakýsi buffer, který uchovává text, co se nevešel na displej a pomocí příkazů scrollright() a scrollleft() posune všechna písmena doprava či doleva podle potřeby. Velmi zajímavá schopnost displeje je potom vykreslení až 8-mi speciálních předdefinovaných symbolů najednou. Tyto funkce ale sami o sobě nejsou dostatečné, protože pro vypsání textu potřebuje programátor 2 příkazy a další problém je že se „ui“ nevykresluje celé najednou, ale po segmentech a dochází pak k problikávání částí, které se překrývají. Další problém je se symboly, protože se najednou dá vykreslit pouze 8 rozdílných symbolů, které se manuálně přiřadí do jednoho z 8-mi slotů v paměti displeje. Proto jsem si upravil rendering tak, aby se mi s ním jednodušeji pracovalo.

První změna je v tom, že v průběhu programu přepisuji pouze string, který se potom zobrazuje až při zavolání příkazu refreshDisplay(); na konci cyklu. To odstranilo problikávání a zjednodušilo zápis na display na jeden příkaz *setStringAt(poziceY, poziceX, String)* a zvýšilo i stabilitu vykreslení textu. Mnohem větší změna ale nastala ve vykreslování speciálních symbolů. Výstup

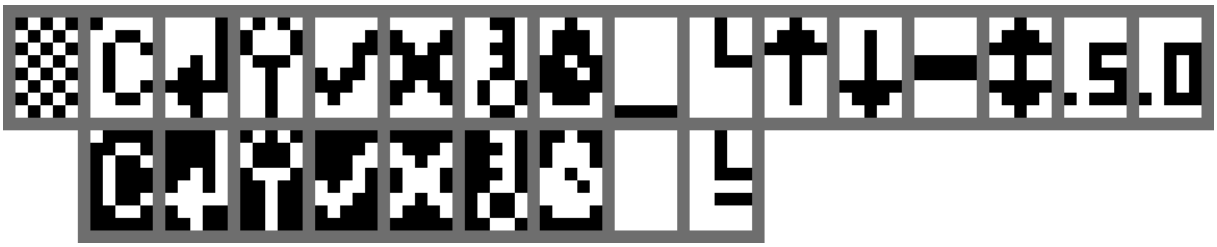
se totiž postupem programu skládá ve dvou stejně velkých polích, jedno obsahuje text a druhé symboly. Symboly se v něm ale ukládají jako index v paměti displeje. Každý symbol má zároveň přiřazené svoje ID, které se ukládá v dalším poli, jenž funguje jako hashmap, kde pozice v hashmapě určuje pozici v paměti displeje a hodnota určuje ID symbolu. Dál se ještě ukládá pozice posledního změněného indexu v paměti symbolů, která se používá při přepsání aktivních symbolů. Tam kde symbol není, má překrývací pole symbolů hodnotu -1. Když ale v buňce nastaví uživatel symbol, tak se do textového pole zapíše znak %, které ale není zobrazeno. Tento systém programátorovi umožní zobrazovat symboly velmi jednoduše pomocí příkazu setSymbolAt(poziceY, poziceX, IDsymbolu) bez toho aby musel manuálně přepisovat paměť displeje.

Dole vidíme reprezentaci všech těchto proměnných, kde symbol ☺ má ID 15 a symbol ® ID 28, výstup této konfigurace by vypadal takto: **AHOJ SVĚTE ☺®**

Tabulka 1 - vizualizace obsahu proměnných v paměti

text	A	H	O	J		S	V	Ě	T	E		%	%
symboly	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1
hashmap	15	28	0	0	0	0	0	0					
index	3												

V případě, že by programátor chtěl zobrazit více než 8 **různých** symbolů, by se tento systém pokusil vykreslit všechny najednou a neustále by přepisoval hashmap registrovaných symbolů. To by se na výstupu projevilo problikáváním všech symbolů. Avšak když by chtěl vypsat 9 stejných symbolů, tak se bezproblémově vypíšou. Další limitace, kterou ale nejsem schopen vyřešit je ta, že displej není schopen zobrazit symbol na 1. pozici řádku. Obě limitace jsou dané výrobcem.

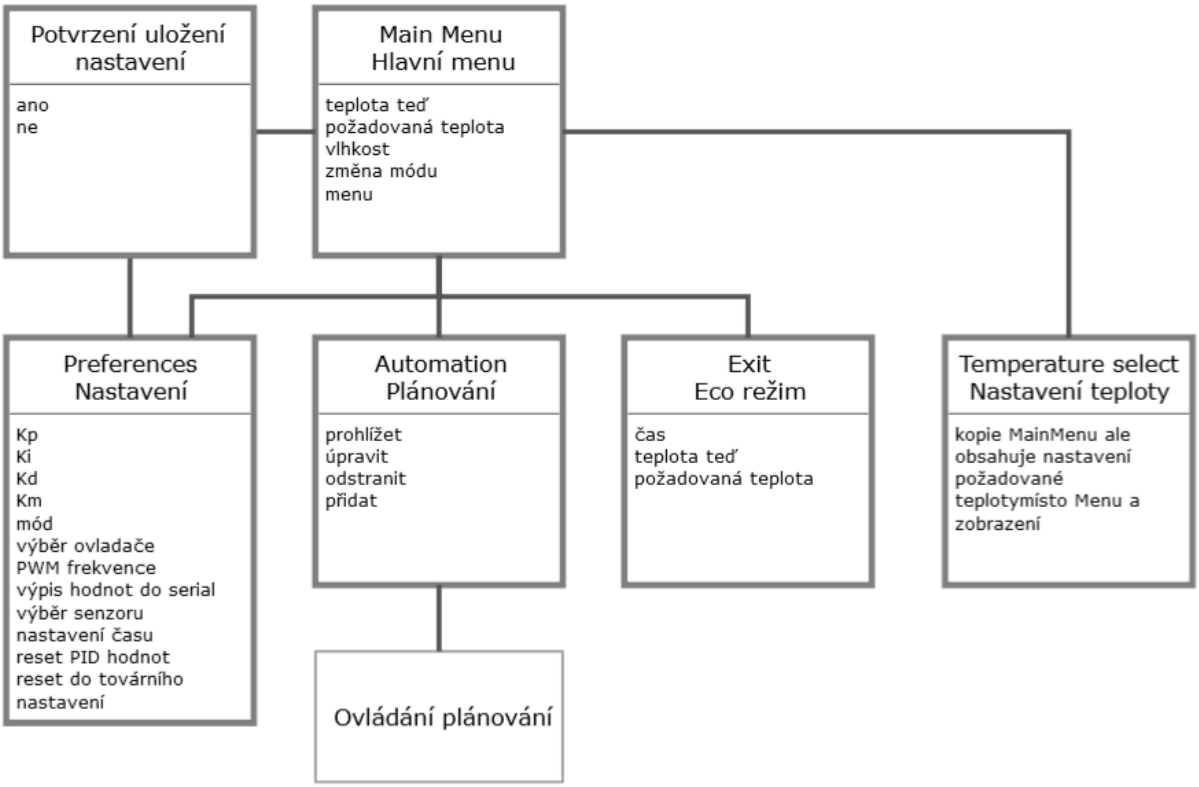


Obrázek 20 - spritesheet symbolů

Na obrázku 19 jsou předdefinované symboly, které jsem pro projekt nakreslil pomocí nástroje na stránce https://maxpromer.github.io/LCD-Character-Creator/. Některé symboly mají negovanou variantu (ta je pod nimi) a některé nejsou ve finálním řešení použité. První symbol je speciální v tom, že se zobrazuje při volání neexistujícího symbolu.

2.5.5 Stránky

Stránek nakonec ve finálním řešení zůstalo 16. V této části bych se ovšem zaměřil na prvních šest, protože zbytek stránek je použitý v nastavení plánování.



Obrázek 21 - UML diagram stránek - 1. část

main menu je po zapnutí zařízení první stránka, ve které se uživatel octne. Na levé straně je vždy zobrazena aktuální teplota senzoru (nahoře) a požadovaná teplota, na kterou se snaží controller dostat (dole). Pokud je v nastavení zvolen interní senzor, tak se vpravo zobrazí ještě vlhkost. Pomocí menu vpravo se poté dá zvolit jedna z těchto možností:

- Auto – přejde do nastavení plánování
- Pref – provede kontrolu správnosti uložených hodnot a otevře nastavení chování controlleru
- Exit – přejde do úsporného režimu bez vypnutí displeje
- Mode – přepne mezi automatickým a manuálním módem

Poté má ještě možnost podržet encoder, čímž se dostane do nastavení požadované teploty.

Preferences neboli nastavení, je stránka ve které uživatel může pomocí encoderu scrollovat nahoru a dolů a tím přepínat mezi hodnotami. Při stisknutí encoderu, poté přejde do nastavení jedné z těchto hodnot.

- Kp – násobič proporcionální hodnoty
- Ki – násobič integrální hodnoty
- Kd – násobič derivační hodnoty
- Km – násobič výstupu PID, jako jediný se nastavuje v násobku x100
- Mode – ovládá, jestli zařízení funguje jako chladič nebo zahřívač
- Control – typ controlleru. Možné hodnoty jsou PID nebo ON/OFF
- Headroom – nastavení hystereze ON/OFF controlleru. Je vidět pouze když je nastavený ON/OFF controller
- PWMfreq – nastavení frekvence PWM výstupu v Herzích. Je vidět pouze když je nastavený PID controller
- PrintOut – vypisování výstupu controlleru do sériové linky.
- Sensor – Nastavení používaného senzoru (Interní/externí)
- Time – nastavení času
- Date – nastavení data
- Reset PID cache – není nastavení, ale tlačítko, které resetuje hodnotu integrálního výpočtu
- Factory default – vymaže nastavení zařízení. Potřebuje potvrzení uživatelem a následný restart.

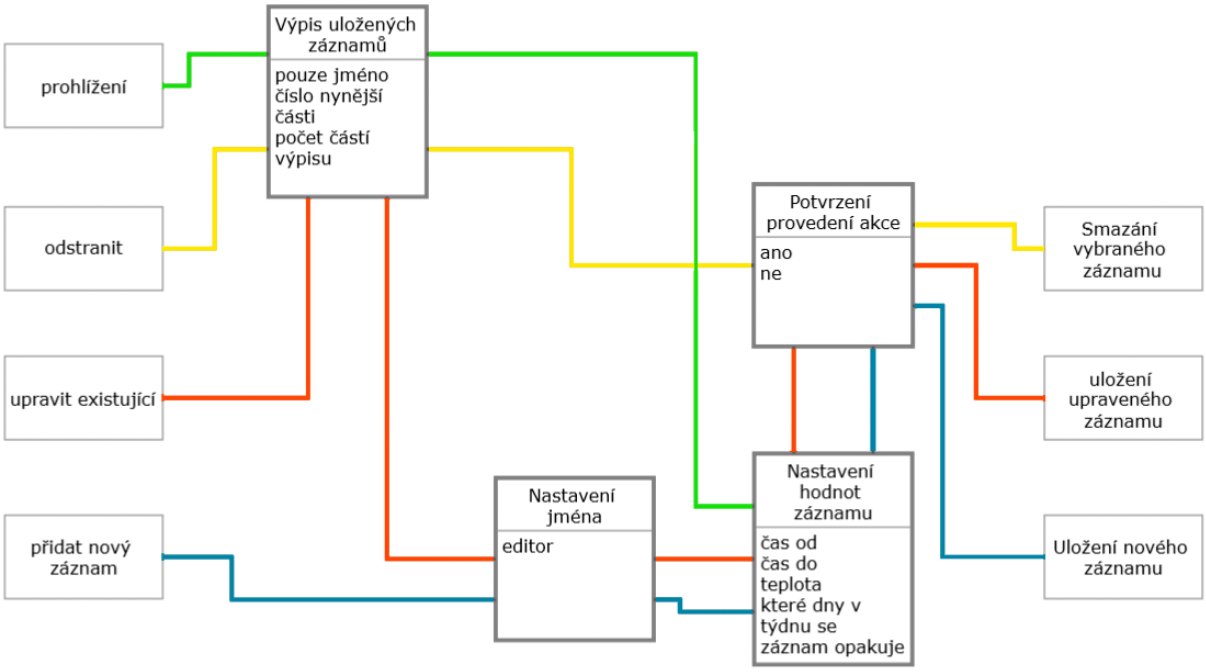
Bohužel je nutné před změnami kterékoli hodnoty zařízení odpojit od regulovaného spotřebiče, protože se nastavení provádí v reálném čase. To by se dalo spravit, kdybych vytvořil pro nastavení objekt, a uživatel by upravoval jeho duplikát, který by se následně přepsal. Bohužel jsem se v té fázi projektu domníval, že Arduino OOP nepodporuje vůbec a tudíž se toto řešení využít nedá. Když jsem se dozvěděl o objektech v Arduinu, tak už bylo nastavení plně implementované a přepisování by zabralo zbytečně moc času. Hodnoty, které mají předdefinované stavy, se nastavují rovnou v této stránce pomocí otáčení encoderu a následným stiskem jako potvrzení. Pro hodnoty, které mohou mít víc stavů je ale potřeba spustit editor. Jeho odchycení poté tvoří značnou část stránky hlavně kvůli odchycování výsledku času a data, které musí být rozděleny podle oddělující dvojtečky/tečky, parsovány a zkontrolována jejich platnost. Při přidržení encoderu se uživatel nevrátí ihned do hlavního menu, nejdřív se porovnají uložené hodnoty ve flash paměti s těmi v RAM a pokud se nějaká liší, přesměruje uživatele do potvrzení uložení nastavení.

potvrzení uložení nastavení se uživatele zeptá, zda chce uložit nastavené hodnoty do dlouhodobé flash paměti. Pokud stiskne „ano“, tak program hodnoty v flash přepíše. Pokud stiskne „ne“, načte zpět bývalé hodnoty a změny zahodí. V každém případě je ale uživatel přesměrován zpět do hlavního menu.

Exit pouze spouští saveScreenHandler.

Temperature select je zvláštní stránka v tom, že je pouze kopií hlavního menu bez spodní části. Ta je zaměněná za manuální ovládání požadované teploty. Teplota se dá nastavit mezi -30° až do 120°. Externí senzor nedokáže změřit vyšší teplotu než 120° a interní zase dokáže měřit pouze do -30°, takže tyto hodnoty jsou hraniční podle hardwaru. Pro potvrzení a návrat zpět do menu musí uživatel stisknout encoder.

Automation, k tomuto se schválně dostávám jako poslednímu. Prvotní menu plánování obsahuje pouze název a menu, kde uživatel může vybrat, jakou akci chce s plánováním provádět: prohlížení, upravení, odstranění, vytvoření nového záznamu. Každá z těchto akcí vyžaduje stránky, které se ale mezi akcemi kříží. Proto se právě v tomto menu po vybrání akce uloží ID akce, které se nemění při navigaci podstránkami této sekce.



Obrázek 22 - UML diagram stránek - 2. část

UML diagram (obrázek 22) dobře znázorňuje, jakým způsobem se propojují zmíněné stránky. Pro každou akci se stránka chová jinak. Například při prohlížení se sice uživatel dostane do stránky nastavení hodnot, ale nemůže nic měnit.

Výpis uložených záznamů umožňuje uživateli procházet seznam všech vytvořených plánovaných nastavení. Každý záznam je zastoupen svým jménem. Na pravé straně pak uživatel vidí svou pozici v seznamu. Abych zrychlil prohledávání a zároveň šetřil s pamětí, tak z paměti čtu pouze jméno záznamu, protože se zbylé vlastnosti v této stránce nevyužívá. Pokud se v příštím cyklu nezmění pozice v seznamu (ta se vypočítá pomocí zaokrouhlení poloviny hodnoty encoderu) tak obrazovku na místech, kde se nachází text, neobnovuju. Tímto způsobem se vyhnu ukládání jmen do paměti, protože je v sobě drží displej. Důležité ale je nezapomenout na manuální refresh při zobrazení překrytí, ale jinak je to velmi stabilní a efektivní cesta jak tyto jména zobrazit a zároveň šetřit s prostředky. Po vybrání jednoho ze záznamů se celý záznam načte do Cache „PROcache“, kde s ním potom mohou ostatní stránky pracovat.

Nastavení jména ani pořádně není stránka, protože pouze předpřipraví Cache pro editor a poté jej spustí. Když se uživatel vrátí z editoru zpět na stránku tak předpřipraví Cache pro Nastavení hodnot záznamu a poté jej na ni přesměruje. Při vytváření nového záznamu se jméno nastaví na „unnamed<číslo Záznamu>“

Nastavení hodnot záznamu je výkonově nejnáročnější stránka v celém programu. Nemá žádnou implementovanou optimalizaci. Pro uživatele je to způsob jak nastavit čas, od kdy do kdy má záznam být platný, nastavit jeho teplotu a vybrat, které dny v týdnu se má opakovat. V celé stránce se pohybuje pomocí otáčení encoderu a stiskem přechází mezi prohlížejícím a nastavovacím módem. Aby věděl, kde se zrovna nachází, tak hodnota, kterou zrovna upravuje, každých poblikává. Při procházecím módu poblikává každých 600 milisekund, ale i při každém pohybu, aby uživatel neztratil pojem o pozici kursoru. Naopak v nastavovacím módu, ve kterém kursor poblikává každých 400 milisekund, se při otáčení encoderu blikání vypne, aby uživatel viděl, co nastavuje. Pokud se na stránku uživatel dostal přes akci prohlížení, tak neuvidí blikání vůbec a ani nebude schopen jakkoliv záznam upravit.

Potvrzení provedené akce je stránka, která se liší pro každou akci. Všechny ji ale používají tak, aby uživatel mohl potvrdit, jestli akci opravdu chce provést. Pokud vybere „ano“, tak jej stránka přesměruje do patřičné stránky akce. Pokud vybere „ne“, tak jej vrátí zpět do menu plánování.

Na konci cesty každé akce se podle nastavených hodnot akce provede a přesměruje uživatele do prohlížeče nebo menu plánování. Ty fungují následovně:

- Prohlížení – nemá žádnou akci a slouží pouze jako readOnly nástroj
- Úprava – přepíše daný záznam tím v PROcache (kde je upravená kopie původního záznamu) pomocí funkce writeToFlash.
- Odebrání – Upraví místo v registru kde vybraný záznam byl uložený. Samotný záznam v paměti ale zůstává, aby se šetřila životnost flash.
- Přidání nového – vytvoří nový záznam v paměti pomocí funkce saveToFlash.

2.5.6 Plánování

Zařízení je schopné celkem solidního plánování. Tím mám na mysli možnost nastavit od kdy do kdy topit a na jakou teplotu. Navíc se dá nastavit, které dny v týdnu se má toto nastavení použít. Každý plánovaný záznam je instance objektu PlannedRecordObject, který má následující vlastnosti:

- Pole deseti charů (znaků), které reprezentují jeho jméno
- Jedno bitový integer obsahující dvojnásobek nastavené teploty
- Byte reprezentující, které dny v týdnu se má záznam opakovat
- Integer obsahující čas začátku
- Integer obsahující čas konce

Jméno jsem vybral jako hlavní způsob odlišení jednotlivých záznamů. Uživatel jej může nastavit při vytváření nového záznamu a může v něm použít následující znaky: „ abcdefghijklmnopqrstuvwxyz 1234567890 .-/()“. Mezera je v seznamu obsažena víckrát pro jednodušší zadávání. Teplotu schválně ukládám jako dvojnásobek. Umožňuje mi to přidat jedno desetinné místo bez použití doublu. Hodnota ale musí být dělitelná jednou polovinou, takže desetinné místo není plnohodnotné. Dny v týdnu se ukládají na bitové úrovni. To znamená, že s hodnotou 64 zacházím, jako kdyby reprezentovala pole bitů s hodnotami 01000000. Dále se v těchto hodnotách ukládá, který den v týdnu se má záznam opakovat tak, že je každý den zastoupen jedním bitem. Když je bit 1 tak se v daný den bude opakovat. Dny jsou poskládané podle RTC, které je řadí následovně: neděle, pondělí, úterý, středa, čtvrtek, pátek, sobota. Poslední bit nevyužívám, ale dal by se použít na ověření, zda je záznam platný (vždy by musel být 1 a když by nebyl, znamenalo by to, že došlo ke korupci dat). Poslední dvě vlastnosti jsou podobné v tom, že mají stejný formát. Čas do záznamu ukládám tak, aby byl

čitelný spíš pro mě než pro program. Převádím totiž například 12.48 na hodnotu 1248. Tento způsob jsem mohl použít, protože pouze kontroluje, jestli je hodnota větší nebo menší než jiný čas v tomto formátu a nepotřebuji z nich získat časový úsek. Jinak bych mohl čas ukládat jako počet minut uplynutých od půlnoci (ale to by se mi hůř četlo :D). Celkem každý zápis zabere pouze 16 bajtů v paměti (viz obr 23). Schválně jsem s touto velikostí šetřil, aby se jich dalo uložit co nejvíc (maximálně záznamy zaberou přesně 4kB paměti + 33 bajtů pro registr).

Obrázek 23 - znázornění využití paměti pro plánovaný záznam

Dále má objekt funkce: ObjectDetails (zobrazí detailní výpis o objektu), saveToFlashMem (uloží objekt do prvního volného místa v registru a danou pozici zaregistruje), Write to flashMem (zapíše objekt do přesně určeného místa v registru bez úprav registrů).

Abych urychlil prohledávání záznamů v paměti, vytvořil jsem ještě registr, který ukládá celkový počet záznamů a každému z nich přiřadí jednu pozici v registru, kde je reprezentován jedním bitem. Maximální počet pozic v registru je 255, protože se počet celkových záznamů dal jednoduše uložit jako bajt s hodnotou 0-255. Registr je v paměti uložen, jako pole 33 bajtů, z nichž jeden slouží pro uložení množství záznamů. Záznamy se ukládají v paměti podle klíče „pr<číslo pozice v registru>“, k tomuto klíči se přidávají klíče samotných vlastností jméno: „name<1-10>“, teplota: „temp“, dny v týdnu: „dayOW“, čas začátku: „sTime” ,čas konce: „eTime”.

Při následném prohledávání paměti aby program zjistil, jestli nemá být nějaký záznam aktivní, se nejdřív podívá, jestli jsou v registru vůbec nějaké záznamy. Pokud ano, postupně projde všemi a testuje jejich platnost následovně:

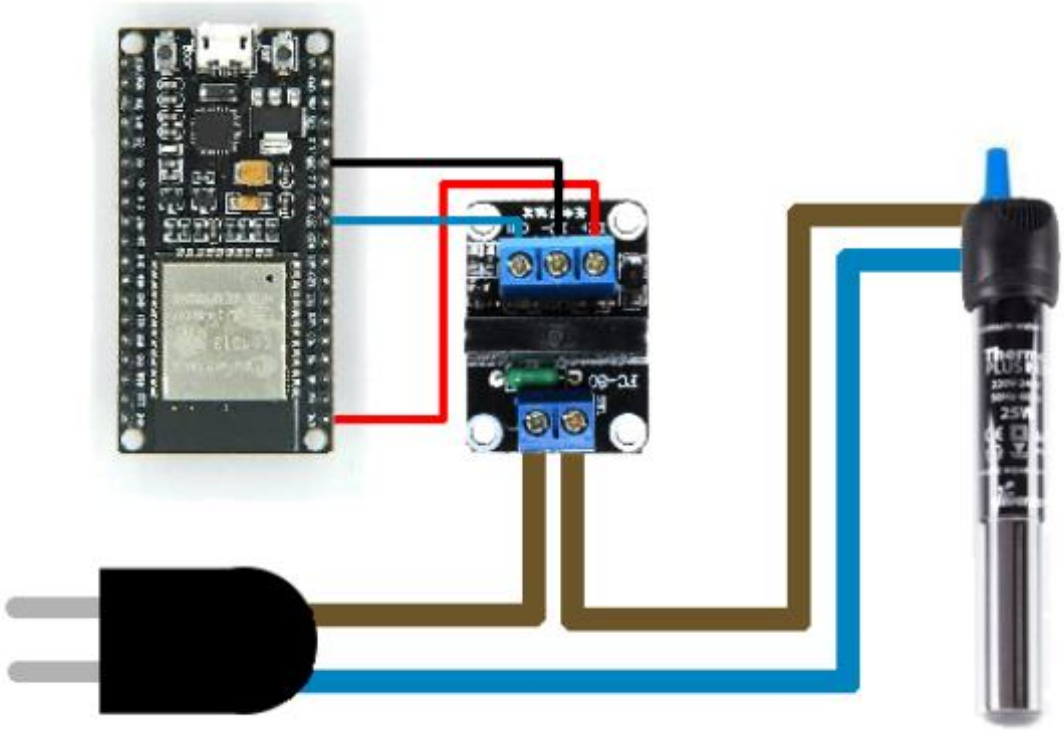
Den v týdnu> čas začátku> čas konce

Pokud se v kterémkoli kroku zjistí, že záznam není platný, postoupí hned na další. Kdyby vyčítal rovnou všechny vlastnost najednou tak by prohledávání mohlo trvat až 39 sekund (kvůli čekání na flash paměť). Když najde platný záznam, zkontroluje si, jestli nenašel už jiný, který by začínal dřív nebo stejně jako nově nalezený. Pokud ano tak nově nalezený záznam považuje za neplatný. Poslední podmínka je, aby nově nalezený záznam končil později než minulý platný. To vytváří prioritu záznamů a zajišťuje, aby se záznamy mohli překrývat bez problémů. Pokud nenalezne platný záznam dřív, než dojde na konec registru (nebo dosáhne počtu registrovaných záznamů), nastaví příští prohledání paměti za 1 minutu a přepne mód zařízení na „manual“. Když ale nalezne platný záznam, tak by bylo zbytečné prohledávat paměť dřív, než záznam skončí, a proto nastaví příští obnovení na čas konce načteného záznamu. Následně nastaví zařízení do automatického módu, čímž načte teplotu jako požadovanou teplotu.

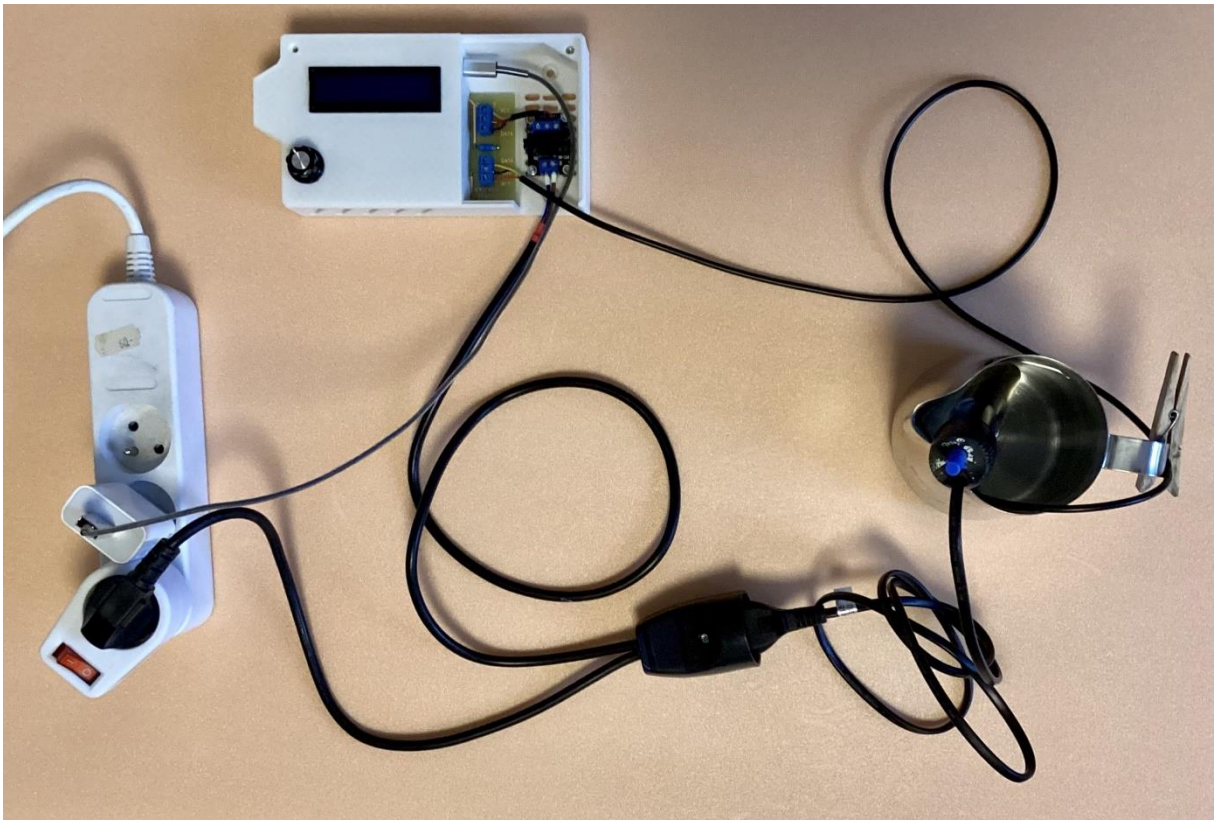
Registr i záznamy jsou všechny uloženy v dlouhodobé paměti. Při inicializaci zařízení se provede test registru, při kterém projde registrované pozice a zkontroluje jejich platnost. Následně zkontroluje, jestli počet registrovaných záznamů sedí s počtem záznamů, které našel a případně jej opraví.

3 EXPERIMENTÁLNÍ ČÁST

Pro testování zařízení jsem využil topení do akvárií Diversa Thermo plus 25 W 14cm, které už nebylo využívané, protože má rozbitý regulátor kvůli kterému může topit až na 50°C. Pro moje využití byl ale perfektní. Topítko pak ovládám pomocí SSR, které rozpojuje vodič fáze a tím reguluje vstupní napětí. Adaptér, který mi to umožňuje, mi vyrobil můj otec, protože má na rozdíl ode mě zkušenosti se silnoproudem.

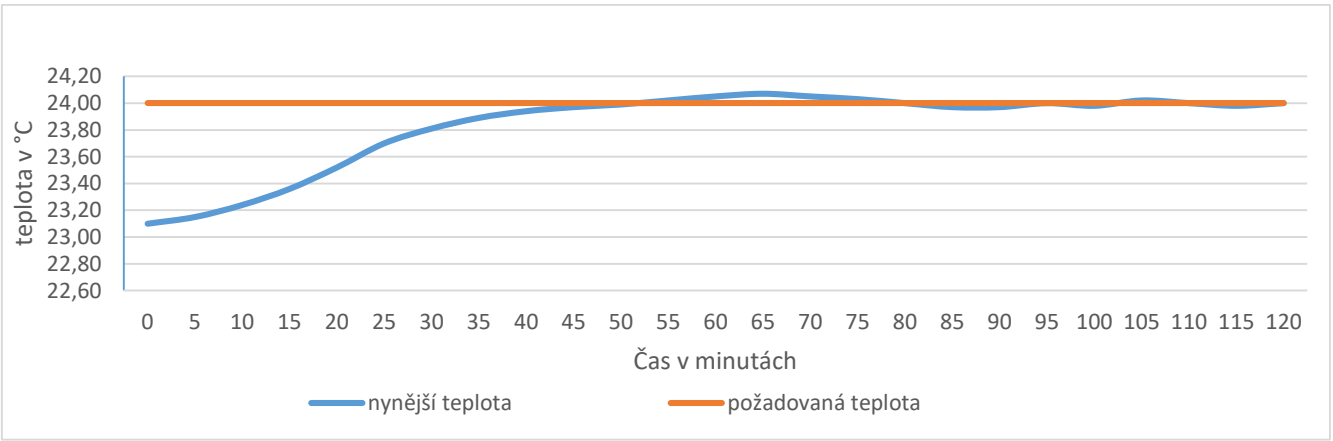


Obrázek 24 - Schéma připojení topítka



Obrázek 25 - fotka zapojení použitého při testování controlleru

Topítko je následně ponořené do nádoby s vodou do které jsem vložil i externí senzor. Program se podle senzoru potom řídí a zahřeje vodu na požadovanou teplotu. Jako nádobu jsem původně chtěl použít 3l plastovou nádobu na müsli. Později jsem ale zjistil že topítko, které používám, není schopné tuto nádobu vytopit. Proto jsem vyzkoušel několik dalších nádob, až jsem se ustálil na 1l plechové nádobce na vodu. Její hlavní výhodou je materiál, ze kterého je vyrobena, protože rychle chladne a umožnil mi tak rychleji testovat nastavení PID controlleru. Na obrázku 26 je vidět vývoj teploty při jednom z těchto testů. Výsledky jsou zapsány i v „tabulce vývoje teplot“ v přílohách.



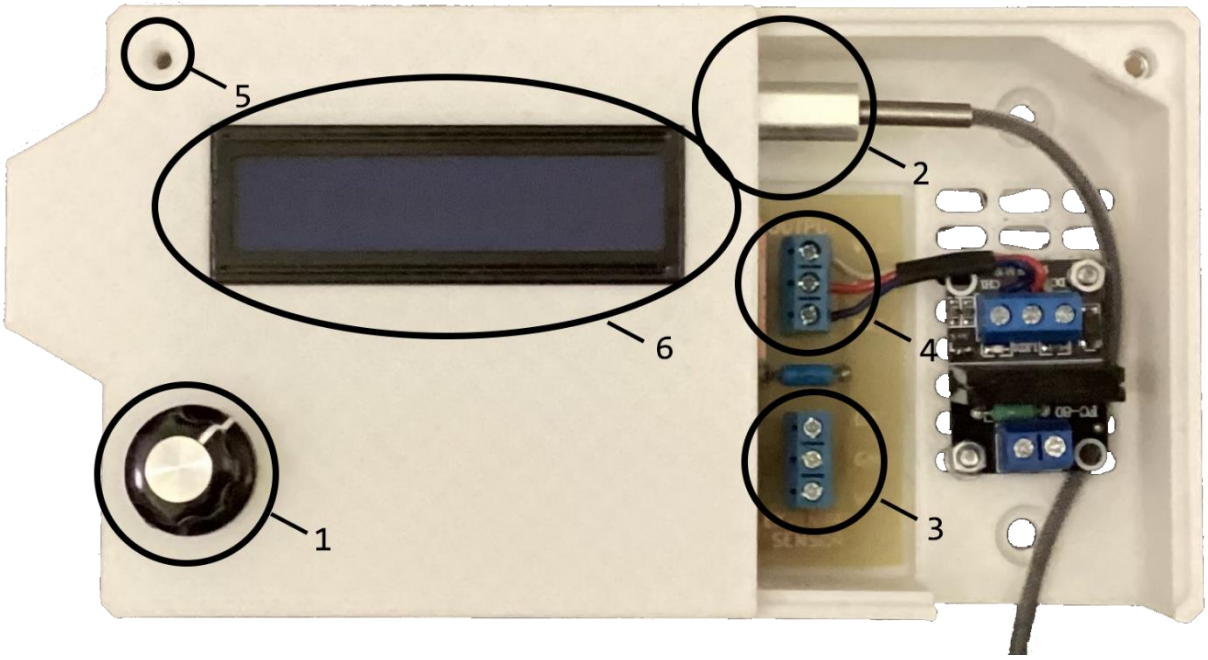
Obrázek 26 - graf vývoje teploty u jednoho z testů PID controlleru

4 UŽIVATELSKÁ PŘÍRUČKA

4.1 Bezpečnostní upozornění

- Před použitím si pozorně pročtěte návod
- Zařízení musí být umístěno mimo dosah dětí pod 15 let
- Zařízení nesmí být připojeno k regulovanému zařízení a nesmí být napájeno, pokud nejsou všechny kryty pevně přišroubovány na určeném místě.

4.2 Základní rozložení



Obrázek 27 - fotka zařízení s popisky

1. Navigační kolečko s tlačítkem
2. Napájení zařízení (micro-USB)
3. Svorky pro připojení externího teplotního senzoru
4. Svorky pro výstup (do relé)
5. Díra pro šroubek
6. Displej

4.3 Ovládání zařízení

Zařízení se ovládá pomocí navigačního kolečka s tlačítkem. Ovládání se v některých menu/stránkách může lišit.

Kurzor vypadá buďto jako úzké podtržítko [_] nebo znaménko „větší než“ [>].

Obecné ovládání

Potvrdit	Krátké stisknutí tlačítka
Zpět	Dlouhé stisknutí a přidržení tlačítka
Nahoru / doleva	Otočit kolečkem proti směru hodinových ručiček
Dolů / doprava	Otočit kolečkem po směru hodinových ručiček

Zadávání textu a čísel

- Při upravování čísel se s nimi zachází jako s textem
- POZOR – pokud nastavíte až moc vysokou hodnotu (1 miliarda a víc), tak se může zařízení zaseknout, a je ho poté potřeba restartovat
- Některé hodnoty (například čas plánování) se nenastavují v editoru ale jejich ovládání je stejné.

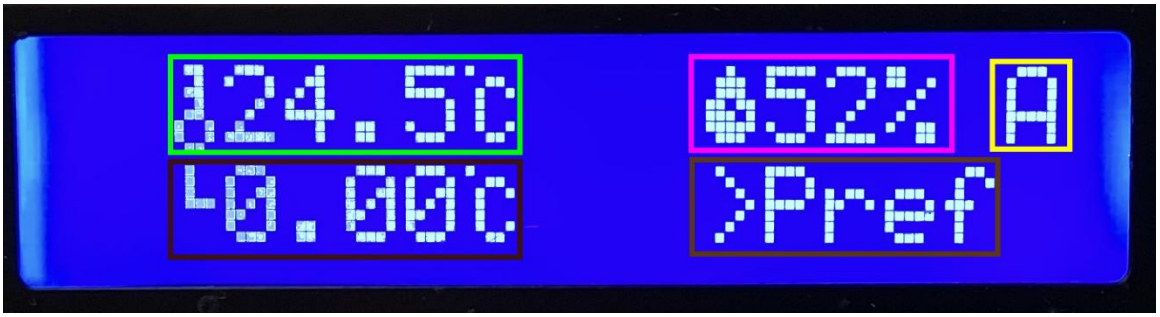
Procházecí mód (prochází se znaky v textu a kurzor nebliká, nebo bliká pomalu)

vybrat znak a přejít do upravovacího módu	Krátké stisknutí tlačítka
Uložit text a odejít z editoru	Dlouhé stisknutí a přidržení tlačítka
Posunout kurzor doleva	Otočit kolečkem proti směru hodinových ručiček
Posunout kurzor doprava	Otočit kolečkem po směru hodinových ručiček

Upravovací mód (prochází se možnosti pro vybraný znak a kurzor rychle bliká)

potvrdit znak a přejít do Procházecí módu	Krátké stisknutí tlačítka
Uložit text a odejít z editoru	Dlouhé stisknutí a přidržení tlačítka
Změnit znak na další možnost	Otočit kolečkem proti směru hodinových ručiček
Změnit znak na minulou možnost	Otočit kolečkem po směru hodinových ručiček

4.4 Hlavní menu



Obrázek 28 - fotka hlavního menu

- Aktuální teplota
  - Automaticky se přepne na zvolený senzor
- Požadovaná/cílová teplota
  - Ukazuje aktuální cílovou teplotu.
- Aktuální vlhkost
  - POZOR – pokud je aktuálně zvolený externí senzor tak se vlhkost neukazuje
- Aktuální mód ve kterém se zařízení nachází
  - A – automat (cílová teplota je automaticky načtená z plánování)
  - M – manual (cílová teplota je ovládaná manuálně)
- Menu
  - Ovládané pomocí obecného ovládání
  - Pref – preferences, přesměruje vás do nastavení zařízení
  - Auto – automation, přesměruje vás do nastavení plánování
  - Mode – přepíná mezi módy zařízení (automatický a manuální)
  - Exit – přepne zařízení do úsporného módu ale nevypne podsvícení

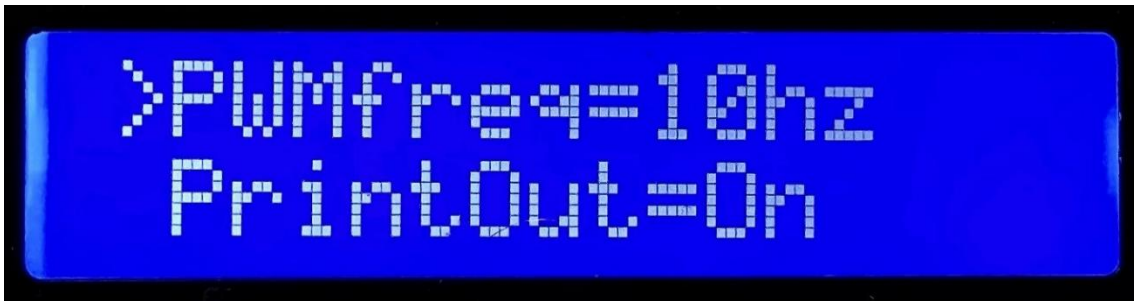
Pro zadání cílové teploty manualně stiskněte a přidržte tlačítko. Tím se otevře nastavení teploty. Cílová teplota může být maximálně v rozsahu -60° až 120°C.

POZOR 1 - pokud je zapnutý automatický mód tak se manuální teplota nenastaví. Aby se tak stalo tak musíte nejdříve přepnout zařízení do manuálního módu.

POZOR 2 - pokud nastavíte manuální teplotu, vydrží nastavená pouze do příštího plánovaného záznamu. Poté zůstane s novou hodnotou i po konci záznamu.

POZOR 3 - manuálně nastavená hodnota se neukládá do dlouhodobé paměti, tudíž je po restartu resetovaná

4.5 Nastavení - preferences

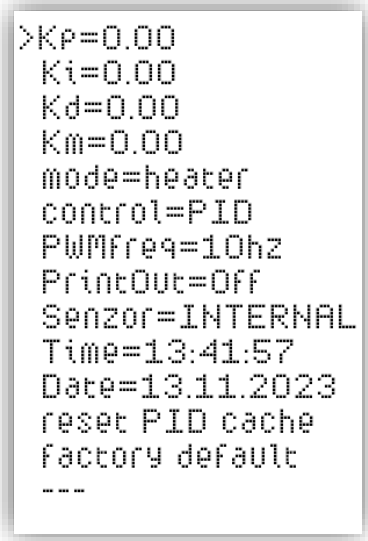


Obrázek 29 - fotka nastavení

Do nastavení se dá vejít z hlavního menu pomocí možnosti „Pref“ (preferences) viz vedle. V nastavení se naviguje pomocí obecného ovládání a kurzor má tvar „>“.

POZOR – pokud je vybrán PID výstup tak není možné nastavit hysterezi. Hystereze je přístupná pouze v On/Off výstupu a její nastavení je poté možné ovládat pomocí headroom hodnoty.

- Kp - je násobič proporcionálního výpočtu PID
- Ki - je násobič integrálního výpočtu PID
- Kd - je násobič derivačního výpočtu PID
- Km - je násobič výstupu PID
- Mode - nastavuje termostat na chlazení (Cooler) nebo ohřev (heater)
- Control - výběr mezi PID a On/Off výstupem
- PWMfreq - nastavuje frekvenci PWM výstupu
- PrintOut - vypisuje stav vybraného ovladače přes sériovou linku (více v kapitole o sériovém výstupu)
- Senzor – výběr mezi interním a externím senzorem (více v kapitole o zapojení příslušenství)
- Time – nastavení času, zadává se ve formátu hh:mm:ss
- Date – nastavení data, zadává se ve formátu DD.MM.RRRR
- Reset PID cache – výmaz kumulativního výpočtu PID
- Factory default – uvede zařízení do továrního nastavení. Vyžaduje další kroky které se ale zobrazí na displaji.



Obrázek 30 - vizualizace kompletní stránky nastavení

4.6 Plánování

Obrázek 31 - fotka menu plánování

Plánované záznamy se ovládají v menu, do kterého se dostanete pomocí menu v main menu. V něm vyberte možnost „Auto“. Plánování nabízí čtyři nástroje, pomocí kterých lze manipulovat se záznamy. Navigace v záznamech většinou využívá obecné nastavení.

- View
- Prohlížení
- umožnuje procházet záznamy a zobrazit hodnoty jednoho z nich
- Edit
- Úpravy
- umožňuje procházet záznamy a následně upravit hodnoty jednoho z nich.
- Nejdřív se upravuje jméno a po opuštění editoru textu vás přesměruje do editoru zbytku hodnot
- Remove
- Odstranění
- Umožňuje procházet záznamy a následně jeden vymazat
- Add
- Vytvoření
- Umožňuje vytvořit nový záznam, nastavit jeho jméno a hodnoty

Priority záznamů

Pokud zařízení zjistí, že je více než jeden záznam aktivní v jednu chvíli, vybere jeden z nich podle priority.

Vyšší prioritu má záznam, který začal být platný dřív a končí co nejdřív od aktuálního času.

Zadávání hodnot plánovaného záznamu

Obrázek 32 - fotka zadávání hodnot plánovaného záznamu

- Které dny v týdnu se má záznam opakovat
  - Křížek znamená že se tento den záznam nebude opakovat.
  - Fajka naopak znamená že záznam bude platný pokaždé když bude ten daný den v týdnu.
  - POZOR – pokud nevyberete žádny den v týdnu tak se záznam nikdy nespustí.
- Požadovaná/cílová teplota
  - Teplota se dá nastavit v rozsahu -60°C až 60°C.
  - Nastavitelné po 0.5°C krocích.
- Čas konce
  - POZOR – pokud je aktuálně zvolený externí senzor tak se vlhkost neukazuje.
  - M – manual (cílová teplota je ovládaná manuálně).
- Čas začátku
  - Ovládané pomocí obecného ovládání.
  - Pref – preferences, přesměruje vás do nastavení zařízení.
  - Auto – automation, přesměruje vás do nastavení plánování.
  - Mode – přepíná mezi módy zařízení (automatický a manuální).
  - Exit – přepne zařízení do úsporného módu ale nevypne podsvícení.

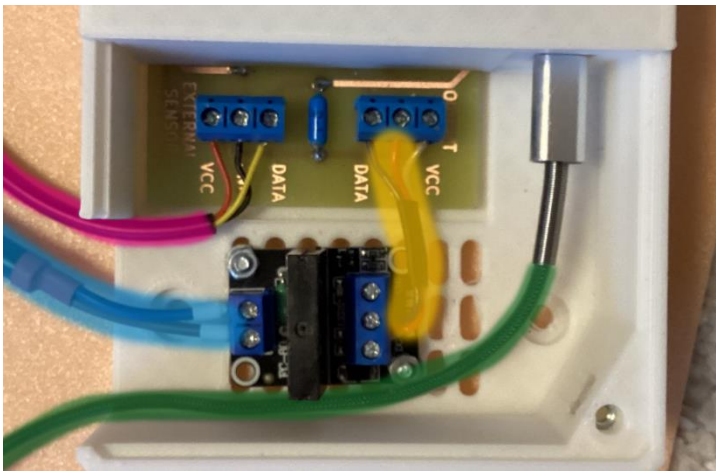
4.7 Zapojení příslušenství

POZOR 1 – Nemanipulujte s kabeláží, pokud je Spínací adaptér zapojený v elektrické síti. NIKDY SE VODIČŮ NEDOTÝKEJTE.

POZOR 2 – Šroubky na svorkách jsou propojeny s kabely v nich, tudíž se vyvarujte dotýkání těchto šroubků.

Popisky k obrázku číslo 33

- Napájení zařízení
- Externí senzor
    - Žlutá – data
    - Černá – GND
    - Červená – VCC
- Připojení Relé
    - Žlutá – data
    - Černá – GND
    - Červená – VCC
- Spínací adaptér
    - Není závislý na pořadí kabelu. Důležité je jen aby byl upevněný ve správné svorce a dostatečně přitažený pomocí šroubku.
    - Umožní relé odpojení regulovaného zařízení od elektrické sítě.



Obrázek 33 - fotka zapojení příslušenství

K připevnění a upevnění slouží pouze šroubová svorka. Při zapojení dbejte na to, abyste zapojily drátky tam, kam patří.

4.8 Sériový výstup

Pro přečtení sériového výstupu lze použít napájecí konektor micro-usb.

- Baud rate: 9600
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: None

Pokud je povolený výpis (hodnota PrintOut) tak zařízení každých 10 sekund vypíše aktuální hodnoty controlleru. Pokud je použitý PID controller tak vypíše i svoje vnitřní hodnoty (výstup každého výpočtu zvlášť). Pro vyfiltrování se dá využít tagu <OUT> ve kterém se výpis vždy nachází.

Příklad výpisu bez PID:

„<OUT>,controllerOutput:0.00,currentTemp:24.57,targetTemp:0.00</OUT>“

„<OUT>
,P:-12.1,I:-1.84,D:-0.03,controllerOutput:0.00,currentTemp:24.83,targetTemp:0.00
</OUT>“

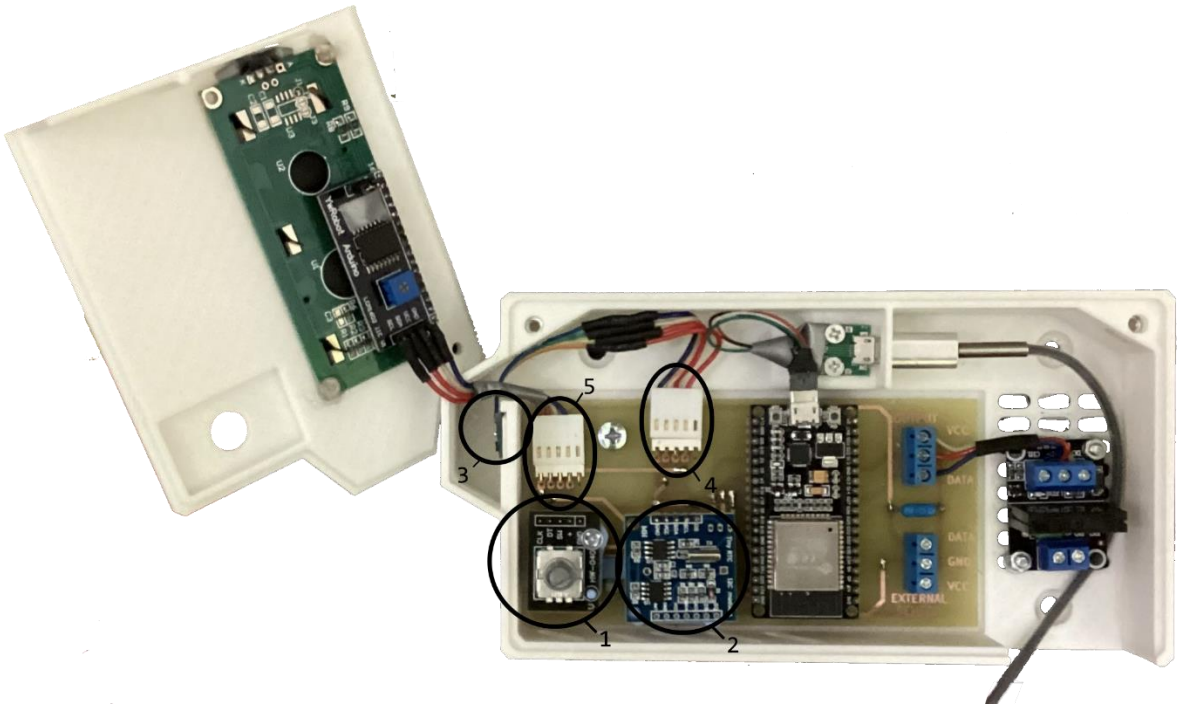
4.9 Výměna komponentů

V případě, že zařízení hlásí, že interní senzor nebo RTC nefungují správně. Nebo zařízení neuchovává správně čas. Je možná potřeba některý z modulů vyměnit.

Začneme tím, že odstraníme navigační kolečko. Na druhé straně od bílé čárky je ze strany malá dírka, ve které je šroubek s imbusovou hlavou velikosti 1.5. Pro jeho odstranění použijeme patřičný imbusový klíč a povolíme jej. *POZOR – neodšroubujte šroubek úplně, mohl by se ztratit.* Poté odšroubujeme šroubky krytů a kryty opatrně odstraníme.

Popisky k obrázku číslo 34

1. Encoder (navigační kolečko)
    - nedá se vyměnit
2. RTC (časový modul)
    - Dá se vyjmout z patice a vyměnit
    - Pokud se hodiny zpožďují nebo nedrží správně čas, stačí pouze dobít integrovanou baterii, která se nachází na zadní straně
3. Interní senzor teploty
    - Dá se vyměnit, stačí jen vytáhnout z konektoru (číslo 4) a odlepit ze stěny
4. Konektor od senzoru
5. Konektor od displeje



Obrázek 34 - fotka interního zapojení zařízení s popisky

5 ZÁVĚR

Vývoj tohoto projektu byl kvůli četným hardwarovým problémům zdlouhavý, ale i přesto jsem si jej užil. Naučil jsem se mnoho věcí o studených spojích a designu PCB. Při psaní softwaru jsem se naučil například vytvářet strukturu programu tak, aby byl jednoduše čitelný ale zároveň optimalizovaný.

Na zařízení asi budu dál pracovat ve volném čase. Například zkrátím délku názvu plánovaného záznamu na 9 znaků, aby mohla teplota být uložená na 2 bajty. Tudíž zvětším rozsah plánovatelných teplot nad 60°C. Mimo to vždy budou ještě další funkce, které by se v programu hodilo mít jako například synchronizace výstupu. Ta by se hodila protože PWM výstup není synchronizovaný s fází v sítí a kvůli triakové regulaci pak nedokáže pokrýt celé spektrum možných výstupných hodnot. Další funkci, kterou určitě implementuji, bude ovládání přes web. K tomu využiju Wi-Fi modul který je zabudovaný v ESP32. Zároveň by se také dal tentýž modul využít na dálkové ovládání topných hlavic a další. Poslední funkce, kterou mám v plánu je kalibrace senzorů. Každý senzor má od výroby nějakou odchylku a v tuto chvíli nemá zařízení jednoduchý způsob jak nastavit kompenzaci. To bych ale změnil a přidal do nastavení další hodnotu, kterou by se tato kompenzace ovládala. Zajímavé by mohlo být i automatické ladění PID controlleru ale to by vyžadovalo hodně zkušeností se zařízením v praxi a ty v tuto chvíli nemám. Na ladění PID controllerů existuje řada algoritmů, na které existují tutoriály na webu.

Nakonec bych ještě jednou chtěl poděkovat mému otci, který mi pomohl při realizaci hardwaru.

Seznam ilustrací a tabulek

Obrázky

Tabulky

Zdroje

Issues with the I²C (Inter-IC) Bus and How to Solve Them. In: Digi-key [online]. [cit. 2023-04-12]. < www.digikey.com/en/articles/issues-with-the-i2c-bus-and-how-to-solve-them >

SENSIRION. arduino-sht library . [online]. 2022, [cit. 2023-12-04]. <https://github.com/Sensirion/arduino-sht>

MADHEPHAESTUS. ESP32Encoder library. [online]. 2022, [cit. 2023-12-04]. <https://github.com/madhephaestus/ESP32Encoder>

VSHYMANSKYY . Preferences library. [online]. 2022, [cit. 2023-12-04]. <http://github.com/vshymanskyy/Preferences >

How To Control an I2C LCD with Arduino. In: Make-it.ca [online]. [cit. 2023-04-12]. Dostupné z: www.arduino.cc/reference/en/libraries/liquidcrystal-i2c/

SHENZEN EONE ELECTRONICS CO. Specification for LCD Module 1602A-1 (V1.2). [online]. 2015, [cit. 2023-12-04]. <https://www.laskakit.cz/user/related_files/eone-1602a1_datasheet.pdf>

HITACHI. HD44780U (LCD-II) DataSheet. [online]. 2015, [cit. 2023-12-04]. <https://www.laskakit.cz/user/related_files/hd44780_datasheet.pdf>

AKBARI, Mohammadreza. Interfacing SHT30 Temperature & Humidity Sensor with Arduino. [online]. 2020, [cit. 2023-12-04]. <https://electropeak.com/learn/interfacing-sht30-temperature-humidity-sensor-with-arduino/>

BDY, Simon. PID: The I, as in integral. [online]. 2019, [cit. 2023-12-04]. <https://medium.com/luos/pid-the-i-as-in-integral-4390c71db12e>

PIDEXPLAINED. How to Tune a PID Controller. [online]. -, [cit. 2023-12-04]. <https://pidexplained.com/how-to-tune-a-pid-controller/>

KOHANBASH, David. PID Control (with code), Verification, and Scheduling. [online]. 2014, [cit. 2023-12-04]. <https://www.robotsforroboticists.com/pid-control/>

RANDOM NERD TUTORIALS. ESP32 Save Data Permanently using Preferences Library. [online]. [cit. 2023-12-04]. <https://randomnerdtutorials.com/esp32-save-data-permanently-preferences/>

M., Luboš. Hodiny reálného času DS1307. [online]. [cit. 2023-12-04]. <https://navody.dratek.cz/navody-k-produktum/hodiny-realneho-casu-ds1307.html>

Triak. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-04-12]. Dostupné z: cs.wikipedia.org/wiki/Triak

Přílohy

Uživatelská příručka (strana 50)

Tabulka 2 - vývoj teplot u jednoho z testů controlleru.

čas	*nynější teplota*	*požadovaná teplota*	*controller output*
0	23,10	24,00	255,00
5	23,15	24,00	255,00
10	23,24	24,00	255,00
15	23,36	24,00	255,00
20	23,52	24,00	255,00
25	23,70	24,00	200,00
30	23,81	24,00	100,00
35	23,89	24,00	0,00
40	23,94	24,00	0,00
45	23,97	24,00	0,00
50	23,99	24,00	0,00
55	24,02	24,00	0,00
60	24,05	24,00	0,00
65	24,07	24,00	0,00
70	24,05	24,00	0,00
75	24,03	24,00	0,00
80	24,00	24,00	85,00
85	23,97	24,00	90,00
90	23,97	24,00	20,00
95	24,00	24,00	20,00
100	23,98	24,00	0,00
105	24,02	24,00	0,00
110	24,00	24,00	10,00
115	23,98	24,00	15,00
120	24,00	24,00	0,00