**DLOUHODOBÁ MATURITNÍ PRÁCE**

**Obor: Automatizace a Robotika**

**Meteorologická stanice**

**Petr Franc**

**Praha**

**DLOUHODOBÁ MATURITNÍ PRÁCE**

**Obor: Automatizace a Robotika**

**Meteorologická stanice**

**Meteorology station**

**Autoři: Petr Franc**

**Škola: SPŠE Ječná**

**Kraj: Praha**

**Konzultant: Jura**

PRAHA 2023

**Prohlášení**

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

Prohlašuji, že tištěná verze a elektronická verze práce jsou shodné.

V Praze dne 2.4.2023 Petr Franc

**Poděkování**

Děkuji všem, kteří mi pomohli při tvorbě samotné práce nebo při zpracovaní dokumentace, především pak mému tátovi Ing.Pavlu Francovi.

**Anotace**

Tato práce popisuje návrh a vývoj meteorologické stanice s mikrokontrolérem ESP32 a e-ink displejem. Stanice je navržena pro sledování aktuálních meteorologických podmínek, jako jsou teplota a vlhkost. Stanice také nabízí uživatelské rozhraní, které umožňuje snadnou vizualizaci dat. Hlavním cílem práce je vytvořit spolehlivou a efektivní meteorologickou stanici, která bude vhodná pro použití v různých prostředích. Práce popisuje detailní postup návrhu hardwaru a programování mikrokontroléru, včetně použitých komponentů. Tato práce může být užitečná pro vývojáře a nadšence, kteří chtějí vytvořit vlastní meteorologickou stanici s ESP32 mikrokontrolérem a e-ink displejem.

**Klíčová slova**

ESP32, C++, IOT, E-INK, Wi-Fi konektivita, 3D tisk

**Annotation**

This work describes the design and development of a meteorological station with ESP32 microcontroller and e-ink display. The station is designed for monitoring current weather conditions, such as temperature, humidity. The station also offers a user interface that allows easy visualization of data. The main goal of the work is to create a reliable and efficient meteorological station suitable for use in various environments. The work describes a detailed process of hardware design and microcontroller programming, including the sensors and other components used. This work may be useful for developers and enthusiasts who want to create their own meteorological station with ESP32 microcontroller and e-ink display.

**Keywords**

ESP32, C++, IOT, E-INK, Wi-Fi connectivity, 3D print

**OBSAH**

[1 Úvod 7](#_Toc131461054)

[2 Jazyk, vývojové prostředí 7](#_Toc131461055)

[2.1 Jazyk 7](#_Toc131461056)

[2.2 Vývojové prostředí 8](#_Toc131461057)

[2.2.1 PlatformIO 8](#_Toc131461058)

[3 Co to je meteorologická stanice 10](#_Toc131461059)

[4 Technologie a vybavení 11](#_Toc131461060)

[4.1 ESP32 11](#_Toc131461061)

[4.2 E-ink 11](#_Toc131461062)

[4.3 Sensor HTU21D 12](#_Toc131461063)

[4.4 Baterie 13](#_Toc131461064)

[4.5 Krabička 14](#_Toc131461065)

[4.5.1 Co to je tisk na 3D tiskárně 14](#_Toc131461066)

[5 Repositář 15](#_Toc131461067)

[6 Architektura systému 16](#_Toc131461068)

[6.1 Schéma zapojení 16](#_Toc131461069)

[6.2 Komunikace se sensorem HTU21D 17](#_Toc131461070)

[6.3 Komunikace s e-ink displejem 18](#_Toc131461071)

[7 Uživatelské rozhraní 19](#_Toc131461072)

[7.1 Displej 19](#_Toc131461073)

[8 Popis programu 20](#_Toc131461074)

[8.1 Flow Diagram 20](#_Toc131461075)

[8.2 Pokus o připojení 20](#_Toc131461076)

[8.3 AP Mode 21](#_Toc131461077)

[8.4 Stáhnutí dat z internetu 21](#_Toc131461078)

[8.4.1 OpenWeatherMap API 21](#_Toc131461079)

[8.4.2 Použití OpenWeatherMap API 22](#_Toc131461080)

[8.5 Načtení dat ze sensoru 22](#_Toc131461081)

[8.6 Zobrazení dat na display 22](#_Toc131461082)

[8.7 Přechod do deepsleep 22](#_Toc131461083)

[9 Struktura programu 23](#_Toc131461084)

[9.1 Class Diagram 23](#_Toc131461085)

[9.2 Main 23](#_Toc131461086)

[9.3 Setup 24](#_Toc131461087)

[9.4 GxEPD display 24](#_Toc131461088)

[9.5 UIDocument 24](#_Toc131461089)

[9.5.1 Json document 25](#_Toc131461090)

[9.5.2 Element 25](#_Toc131461091)

[9.6 UIRenderer 26](#_Toc131461092)

[9.7 Wifi Manager 27](#_Toc131461093)

[9.8 File System 27](#_Toc131461094)

[9.9 Adafruit BME280 28](#_Toc131461095)

[9.10 HTTPClient 29](#_Toc131461096)

[9.10.1 REST API Request 29](#_Toc131461097)

[9.11 fileSystemUtils.hpp 30](#_Toc131461098)

[9.12 jsonUtils.hpp 30](#_Toc131461099)

[9.13 weatherStationDataTypes.hpp 30](#_Toc131461100)

[10 Popis vývojového procesu 31](#_Toc131461101)

[10.1 Vytvoření uživatelského rozhraní s e-ink displejem 31](#_Toc131461102)

[10.2 Integrace připojení na wifi pomocí ESP32 31](#_Toc131461103)

[10.3 Založení účtu na openweathermap.org 31](#_Toc131461104)

[10.4 Integrace ESP32 HttpClientu na stahování dat z internetu 31](#_Toc131461105)

[10.5 Integrace ESP32 WifiManageru 31](#_Toc131461106)

[10.6 Připojení senzoru HTU21D 31](#_Toc131461107)

[10.7 Vytvoření krabičky ve programu Fusion360 31](#_Toc131461108)

[10.8 Tisk na 3D tiskárně 32](#_Toc131461109)

[11 Výsledky a zhodnocení 33](#_Toc131461110)

[11.1 Shrnutí dosažených cílů a zhodnocení projektu 33](#_Toc131461111)

[11.2 Možná zlepšení 33](#_Toc131461112)

[12 Závěr 34](#_Toc131461113)

[13 Použitá literatura 35](#_Toc131461114)

[14 Seznam obrázků a tabulek 36](#_Toc131461115)

# Úvod

Tato dokumentace popisuje vývoj meteorologické stanice s mikrokontrolérem ESP32 a e-ink displejem. Tato stanice slouží k monitorování aktuálních meteorologických podmínek, jako jsou teplota, vlhkost. Stanice využívá Wi-Fi konektivitu k přenosu dat ze serveru a nabízí uživatelské rozhraní pro snadnou vizualizaci dat. Hlavním cílem tohoto projektu je vytvořit efektivní a spolehlivou meteorologickou stanici, která bude vhodná pro použití v různých prostředích.

Tento projekt vyžaduje znalosti v oblasti hardwaru, software a programování mikrokontrolérů. V této dokumentaci najdete informace o návrhu hardwaru, programování mikrokontroléru a rozhraní s uživatelem. Dokumentace by měla být užitečná pro vývojáře a nadšence, kteří chtějí postavit vlastní meteorologickou stanici s ESP32 mikrokontrolérem a e-ink displejem.

# Jazyk, vývojové prostředí

## Jazyk



[1] Obr.1: C++ Logo

Zvolil jsem jazyk C++ pro programování ESP32, protože ho na dobré úrovni ovládám a mám s ním zkušenosti. C++ je objektově orientovaný programovací jazyk, který je výkonný, flexibilní a používá se pro vývoj softwaru. Podporuje mnoho funkcí, jako jsou šablony, přetěžování operátorů a vícevláknové programování. Je známý pro svou rychlost a efektivitu, ale může být náročný na učení se.

## Vývojové prostředí



[2] Obr.2: VSCode Logo

Jako IDE používám Visual Studio Code s pluginem Platform.io, který umožňuje práci s mikrokontrolery. Visual Studio Code (VS Code) je open-source editor kódu, který se používá pro vývoj softwaru na různých platformách. Jeho flexibilita, možnosti rozšíření a snadná konfigurace ho dělají oblíbeným mezi programátory.

VS Code je open-source a zdarma ke stažení a používání na různých operačních systémech, jako jsou Windows, macOS a Linux. Některé z jeho klíčových funkcí zahrnují:

1. Editor kódu: VS Code obsahuje editor kódu s mnoha funkcemi, jako jsou například syntax highlighting, automatické doplňování kódu, refaktorování, debugování a integrace s Gitem.
2. Rozšiřitelnost: VS Code umožňuje uživatelům instalovat a používat různé rozšíření pro vylepšení funkcionality IDE. Tyto rozšíření lze nalézt v oficiálním rozšíření trhu nebo vytvořit vlastní.
3. Integrovaný terminál: VS Code má integrovaný terminál, který umožňuje uživatelům spouštět příkazy přímo z IDE, což usnadňuje práci s různými nástroji

### PlatformIO



[3] Obr.3: PlatformIO Logo

PlatformIO je plugin pro VS Code, který usnadňuje vývoj softwaru pro embedded systémy a IoT zařízení. Podporuje mnoho různých mikrokontrolérů a nabízí užitečné funkce, jako jsou inteligentní kódo-ové zvýraznění syntaxe, ladění kódu a správa knihoven. To usnadňuje programování pro embedded systémy a IoT aplikace, což je jinak náročné a časově náročné.

# Co to je meteorologická stanice

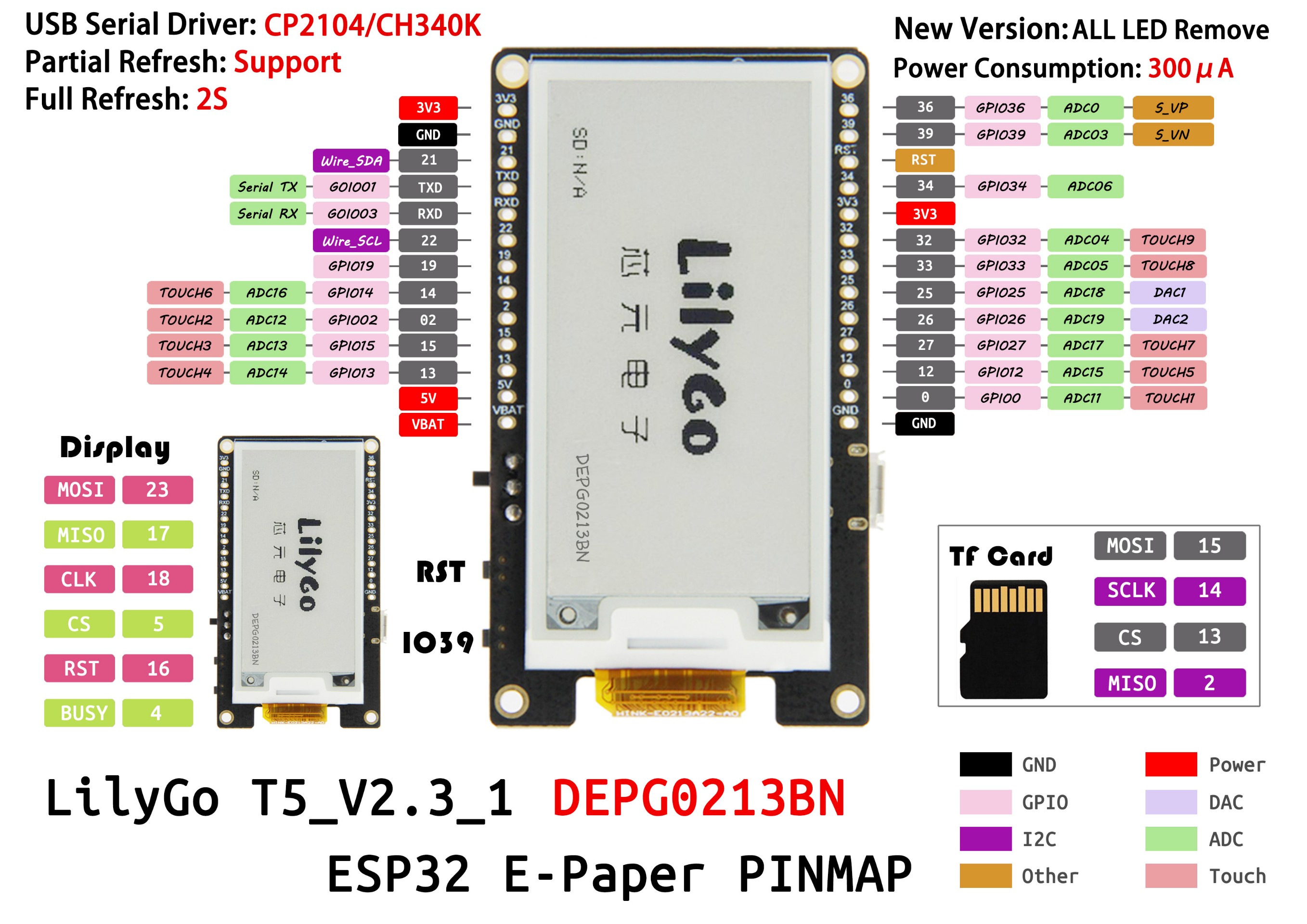


Obr.4: ukázka mé meteorologické stanice

Meteorologická stanice, kterou vytvářím informuje o teplotě a vlhkosti v Praze, stručný popis počasí, a teplota a vlhkost v místnosti. Ještě k tomu stanice vypíše na displej datum a čas. Teplota a vlhkost v místnosti se získá pomocí HTU21D sensoru. Teplota, vlhkost, popis počasí, datum a čas v Praze se získá přes internet pomocí http requestu na server „api.openweathermap.org“. Jelikož stanice získává data přes internet, musí být připojena přes wifi. Po načtení a zobrazení dat jde stanice do deepsleepu po dobu 15 minut, poté se proces opakuje. Pokud se stanici nepodaří připojit na internet. Přepne se do AP modu, ve kterém muže uživatel zadat wifi na kterou se má stanice připojit. Pokud i to selže stanice se podívá na poslední data, které stáhla a uložila do paměti a zobrazí je. Toto neplatí pro data o místní teplotě a vlhkosti, jelikož ta se získávají přes sensor HTU21D, který se dá použít nezávisle na připojení internetu.

# Technologie a vybavení

## ESP32



[4] Obr.5: EPS32 PINOUT

ESP32 je výkonný a všestranný mikrokontrolér, který se často používá pro vývoj IoT aplikací a embedded systémů. Obsahuje dvě jádra procesoru, Wi-Fi a Bluetooth modul, a mnoho periférií pro komunikaci a ovládání různých zařízení. ESP32 je také nízkopříkonový a relativně levný, což z něj činí ideální volbu pro vývoj aplikací pro IoT.

Já si koupil ESP32 LILYGO, které už má na sobě napájený e-ink displej.

[5]

## E-ink

E-ink display (též nazývaný jako elektronický papír) je typ displeje, který nevyžaduje podsvícení a spotřebovává energii pouze při změně zobrazovaného obsahu. To z něj činí ideální volbu pro aplikace, které vyžadují nízkou spotřebu energie, jako jsou například elektronické čtečky knih.

E-ink displeje mohou být snadno řízeny pomocí mikrokontroléru, jako je ESP32 nebo Arduino, a jsou k dispozici knihovny pro řízení těchto displejů. Tyto knihovny umožňují jednoduché zobrazování textu, obrázků a grafiky na displeji s nízkou spotřebou energie. E-ink displeje jsou tedy ideální volbou pro aplikace, které potřebují nízkou spotřebu energie a jednoduché zobrazování informací.

Tento displej už je integrovaný v desce.

## Sensor HTU21D



[6] Obr.6: Sensor HTU21D

HTU21D je digitální teplotní a vlhkostní senzor vyrobený společností Measurement Specialties, který je obvykle používán v embedded aplikacích a IoT zařízeních. Tento senzor měří relativní vlhkost a teplotu okolního prostředí a komunikuje pomocí digitálního rozhraní I2C. Senzor HTU21D je také známý pro svou vysokou přesnost a malé rozměry, což z něj činí ideální volbu pro aplikace, které vyžadují přesné měření teploty a vlhkosti s malou spotřebou energie. Senzor měří teplotu s přesností ±0,3°C a vlhkost s přesností ±2% relativní vlhkosti.

[7]

## Baterie



[8] Obr.7: GeB Li-Ion Baterie 3.7V

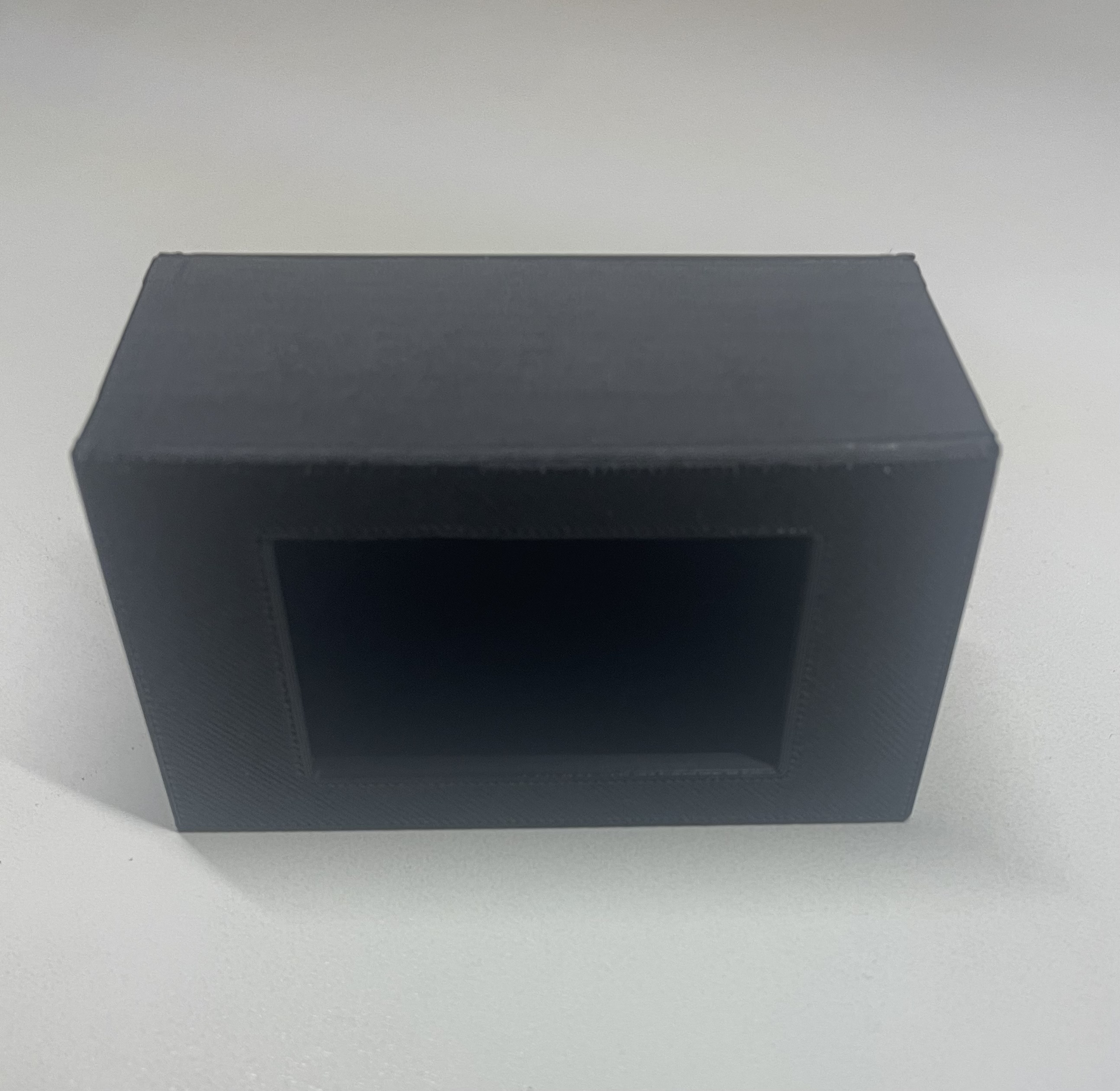
GeB Li-Ion Baterie 1x18650 1S1P 3.7V 3200mAh je dobíjecí lithium-iontová baterie navržená pro širokou škálu aplikací, včetně elektronických zařízení, robotiky, přenosných zdrojů energie, osvětlení a mnoha dalších. Tato baterie má kapacitu 3200mAh a napětí 3,7V. Její výstupní proud může dosáhnout až 10 A.

Baterie GeB Li-Ion 1x18650 je balena v kovovém pouzdře s rozměry 18 x 65 mm a hmotností 50 gramů. Je vybavena ochranou proti přepólování a přetížení. Baterie používá lithiové elektrody pro ukládání energie a má velmi dobrou výdrž, nízkou sebevybíjecí rychlost a vysoký počet cyklů nabíjení.

Tato baterie je určena pro profesionální a průmyslové aplikace, ale může být také vhodná pro použití v hobby projektech. Je důležité si uvědomit, že použití jakékoli lithium-iontové baterie vyžaduje dodržování bezpečnostních postupů, jako je používání správného nabíječe a skladování na bezpečném místě, aby se minimalizovalo riziko požáru nebo výbuchu.

[9]

## Krabička



Obr.8: Krabička

Model krabičky jsem vytvořil v programu Fusion360 a pak jsem model vytiskl na 3D tiskárně. Tento způsob jsem si vybral kvůli jeho jednoduchosti, a protože už mám tiskárnu doma. Více o tomto kroku je na straně 30 v kapitole „Vytvoření krabičky ve programu Fusion360“ a „Tisk na 3D tiskárně“.

### Co to je tisk na 3D tiskárně

3D tisk je proces výroby třírozměrných objektů na základě digitálního návrhu (CAD). Tento proces je realizován pomocí speciálních tiskáren nazývaných 3D tiskárny.

Proces 3D tisku probíhá postupným nanášením vrstev materiálu na základě digitálního návrhu. Tiskárna postupně tiskne vrstvy materiálu, dokud se nevytvoří celkový objekt. Použité materiály mohou být různé, včetně plastů, kovů, keramiky a dalších.

3D tisk se používá v mnoha různých oblastech, jako je průmyslové prototypování, lékařství, architektura a design. Umožňuje rychlé a relativně levné vytvoření prototypů a testování nových návrhů.

# Repositář

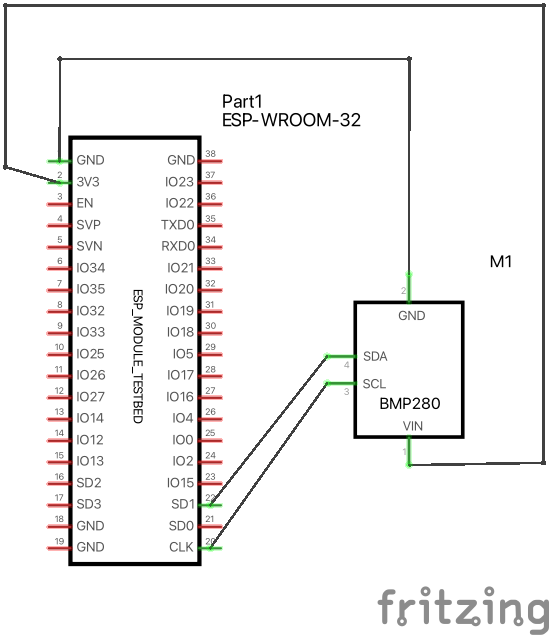


[10] Obr.9: GitHub Logo

Projekt má založený repositář na githubu. GitHub je webová platforma pro správu a sdílení verzovacího systému Git. Umožňuje uživatelům nahrávat, ukládat a spravovat svůj kód a projekty v cloudovém úložišti. GitHub poskytuje nástroje pro spolupráci a řízení verzí projektů, včetně správy úkolů, sledování chyb a řízení vývoje. Uživatelé mohou také sledovat a přispívat k projektům jiných uživatelů, sdílet svůj kód a diskutovat o něm s ostatními uživateli. GitHub se stal jedním z nejpopulárnějších nástrojů pro správu verzí a spolupráci na projektech v softwarovém vývoji. Link na repozitář: https://github.com/pixipit/Long-term

# Architektura systému

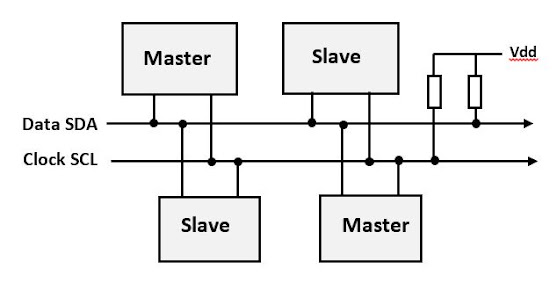
## Schéma zapojení



Obr. 10: Schéma zapojení

Schéma zapojení bylo uděláno za pomocí programu fritzing. Ve schématu je mikrokontroler ESP32, protože deska, kterou jsem použil, ESP32 LILYGO, v programu fritzing není ani se nedá stáhnout z internetu, kvůli tomuto schéma neodpovídá realitě. To samé platí pro sensor HTU21D. Sensor, který je ve schématu má ale stejný pinout a také snímá teplotu a vlhkost. Ve schématu lze vidět že sensor je napojen na piny 20 pro CLK, 22 pro SDA, 3V3 pro napájení a GND pro uzemnění. Toto je pravda i ve provedení s jediným rozdílem že na desce ESP32 LILYGO jsou tyto piny vedle sebe a zapojení je tím elegantnější.

## Komunikace se sensorem HTU21D



[12] Obr.11: Obecné I2C schéma

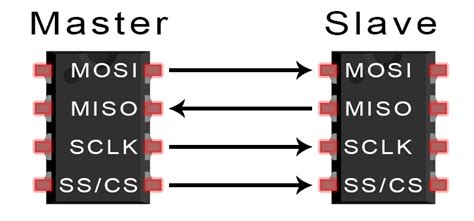
Komunikace mezi senzorem HTU21D a mikrokontrolerem ESP32 probíhá pomocí protokolu I2C (Inter-Integrated Circuit). I2C je sériový komunikační protokol, který umožňuje komunikaci mezi mikrokontrolery a dalšími periferiemi, jako jsou senzory, displeje nebo paměťová média.

Dotaz na senzor HTU21D se skládá z několika kroků. Nejprve musí být poslána zahajovací sekvence, která oznamuje senzoru, že bude následovat dotaz. Poté se posílá samotný dotaz, který obsahuje instrukce pro získání dat o teplotě a vlhkosti. Senzor HTU21D odpoví na tento dotaz přenosem dat o teplotě a vlhkosti zpět k mikrokontroleru ESP32. Na závěr se komunikace ukončí vysláním ukončovací sekvence.

Pro tyto kroky používáme knihovnu sensoru. Více informací o knihovně v kapitole

9.9Adafruit BME280 na straně 28.

## Komunikace s e-ink displejem



[13] Obr.12: Obecné SPI schéma

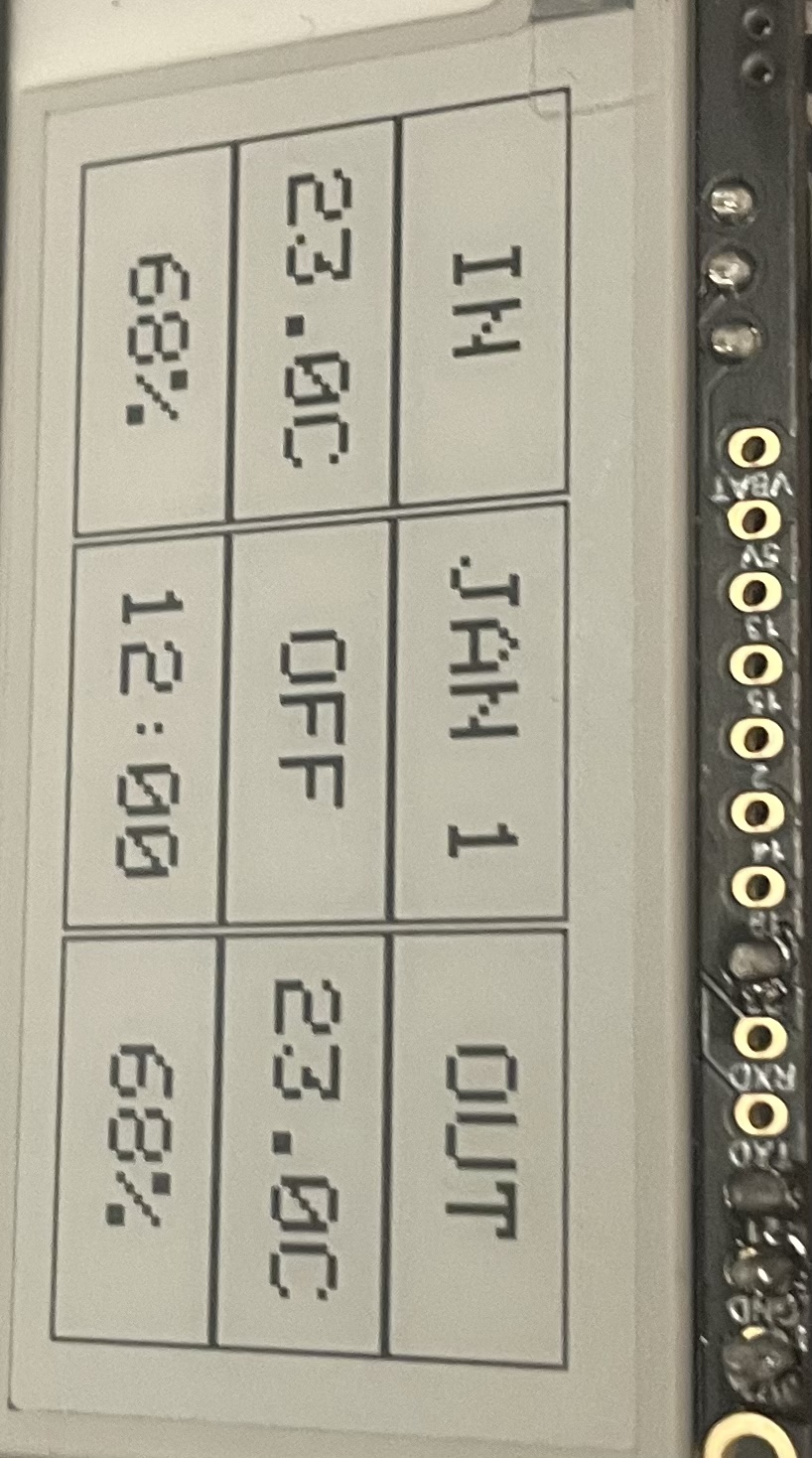
Komunikace mezi ESP32 a e-ink displejem pomocí knihovny GxEPD (Grayscale e-paper display) probíhá pomocí komunikačního rozhraní SPI (Serial Peripheral Interface), které je běžné u mnoha mikrokontrolérů, včetně ESP32.

ESP32 musí být propojen s displejem pomocí čtyř vodičů: SCK, MOSI, MISO a CS. Když ESP32 potřebuje ovládat displej, nastaví CS na LOW a začne zasílat příkazy pomocí SPI. Poté, co jsou příkazy odeslány, nastaví se CS zpět na HIGH.

Komunikace s e-ink displejem pomocí knihovny GxEPD je tedy poměrně jednoduchá a snadno se dá integrovat do kódu programu na ESP32.

# Uživatelské rozhraní

## Displej



Obr.13: Displej

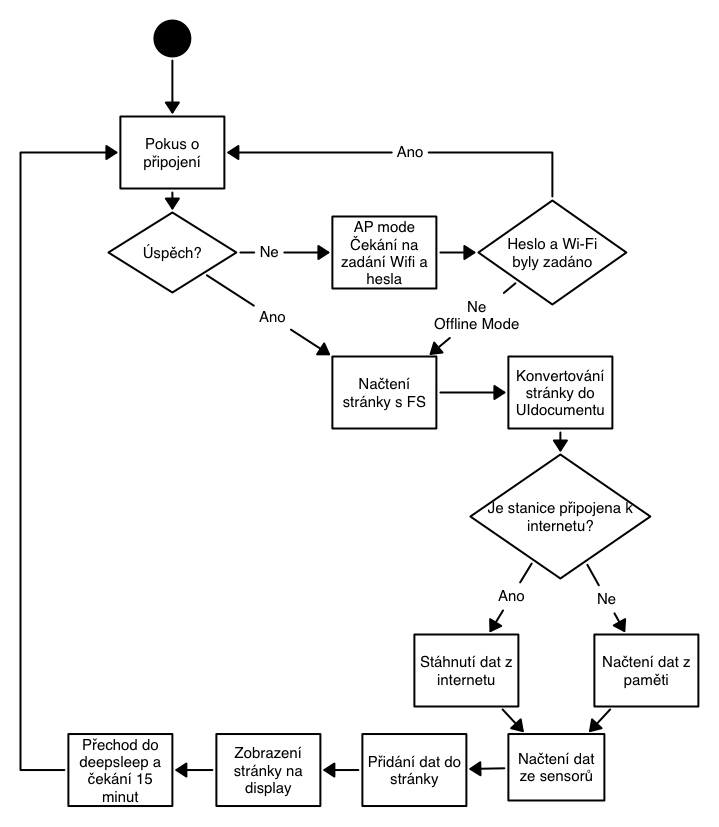
Na displeji s e-ink technologií se zobrazují tři hlavní informační bloky. V levé části displeje je zobrazena aktuální teplota a vlhkost v místnosti. V pravé části displeje se zobrazují informace o venkovní teplotě a vlhkosti. Prostřední část displeje zobrazuje datum, čas a popis aktuálního počasí.

Celkově je tedy možné na displeji rychle a snadno získat nejdůležitější informace o počasí a klimatu jak v místnosti, tak i venku. Díky jednoduchému a intuitivnímu rozvržení informací na displeji je přehlednost zajištěna a uživatel se tak může rychleji rozhodovat, jak se vybavit pro aktuální počasí a jak přizpůsobit své aktivity dle této informace.

Na obrázku lze vidět co displej ukáže poprvé co ho zapneme a on není připojen k internetu ani nemá data uložená.

# Popis programu

## Flow Diagram



Obr.14: Flow diagram

Tento diagram popisuje chod meteorologické stanice.

## Pokus o připojení

Připojení k Wifi sítím probíhá pomocí knihovny Wifi Manager pro ESP32 mikrokontroléry a probíhá následovně:

1. Pokud ESP32 mikrokontrolér nenajde žádnou konfigurovanou Wifi síť, automaticky se spustí Wifi Manager a začne nabízet možnost nastavení nové Wifi sítě.
2. V případě, že je již Wifi síť nastavena, ESP32 se připojí k této Wifi síti pomocí uložených přihlašovacích údajů.
3. Pokud se ESP32 mikrokontrolér nedokáže připojit k Wifi síti pomocí uložených údajů, automaticky se spustí Wifi Manager a nabídne možnost zadání nových přihlašovacích údajů pro Wifi síť.

Připojení k Wifi síti pomocí knihovny Wifi Manager je tedy velmi jednoduché a uživatelsky přívětivé. Uživatel nemusí ručně zadávat konfigurační údaje pro Wifi síť, což šetří čas a minimalizuje chyby při nastavování Wifi připojení.

## AP Mode

AP (Access Point) mód v rámci Wifi Manageru pro ESP32 mikrokontroléry umožňuje vytvoření vlastní Wifi sítě, ke které se mohou další zařízení připojit a nakonfigurovat Wifi Manager. V AP módu se ESP32 mikrokontrolér chová jako Wifi router a vytváří svou vlastní Wifi síť s předem definovaným názvem a heslem.

Pro nastavení Wifi Manageru v AP módu je potřeba nejprve zavolat funkci wifiManager.autoConnect() a připojit se k Wifi síti vytvořené ESP32 mikrokontrolérem. Poté lze přistoupit ke konfiguraci Wifi Manageru přes webové rozhraní, které je dostupné na adrese 192.168.4.1.

V AP módu lze konfigurovat Wifi Manager pomocí jakéhokoli zařízení, které podporuje Wifi připojení, jako jsou například chytré telefony, počítače nebo tablety. Po dokončení konfigurace se ESP32 mikrokontrolér automaticky připojí k nové Wifi síti a přestane fungovat jako Wifi router v AP módu.

AP mód je užitečný v situacích, kdy není k dispozici žádná jiná Wifi síť pro připojení a konfiguraci Wifi Manageru.

## Stáhnutí dat z internetu

Stanice stahuje meteorologická data ze serveru "api.openweathermap.org" pomocí třídy HttpClientu, která je k dispozici v knihovně pro ESP32. HttpClient poskytuje prostředky pro komunikaci s webovými servery pomocí protokolu HTTP. Stanice se připojí k serveru a odešle požadavek na získání dat. Po obdržení odpovědi od serveru analyzuje stanice data a ukládá je do paměti.

### OpenWeatherMap API



[14] Obr.15: OpenWeatherMap Logo

api.openweathermap.org je veřejné API pro získání aktuálního a předpovědního meteorologického obsahu, jako jsou teplota, vlhkost, rychlost větru, atmosférický tlak a mnoho dalšího pro libovolné geografické místo na světě. Toto API je poskytováno společností OpenWeatherMap, která sbírá a zpracovává data od meteorologických stanic a dalších zdrojů. Tato data jsou pak poskytována v různých formátech, jako jsou JSON, XML nebo HTML. API je běžně používáno pro vývoj mobilních aplikací, webových stránek a IoT projektů, které potřebují aktuální meteorologické informace.

### Použití OpenWeatherMap API

Pro použití API openweather.org je potřeba nejprve získat svůj API klíč, který se používá pro autorizaci při každém dotazu na API.

Po získání API klíče je možné vytvořit dotaz na API pomocí HTTP protokolu. Dotaz je tvořen URL adresou, na kterou se odesílá dotaz s parametry.

Například pro získání aktuálního počasí pro město Praha v jednotkách Celsius je URL adresa:

http://api.openweathermap.org/data/2.5/weather?q=Praha&units=metric&appid={API\_KEY}

V této adrese jsou tři parametry:

* q - označuje město, pro které se má počasí získat
* units - označuje jednotky, ve kterých se má počasí zobrazit
* appid - API klíč pro autorizaci

Po odeslání dotazu na API se v odpovědi vrátí JSON objekt s požadovanými informacemi o počasí. Tento objekt je nutné zpracovat a případně zobrazit na webové stránce nebo v aplikaci.

## Načtení dat ze sensoru

Stanice získává data o teplotě a vlhkosti v místnosti pomocí senzoru HTU21D, který je připojen k ESP32 mikrokontroléru. Senzor HTU21D je digitální teploměr a hygrometr, který poskytuje přesné měření teploty a vlhkosti.

## Zobrazení dat na display

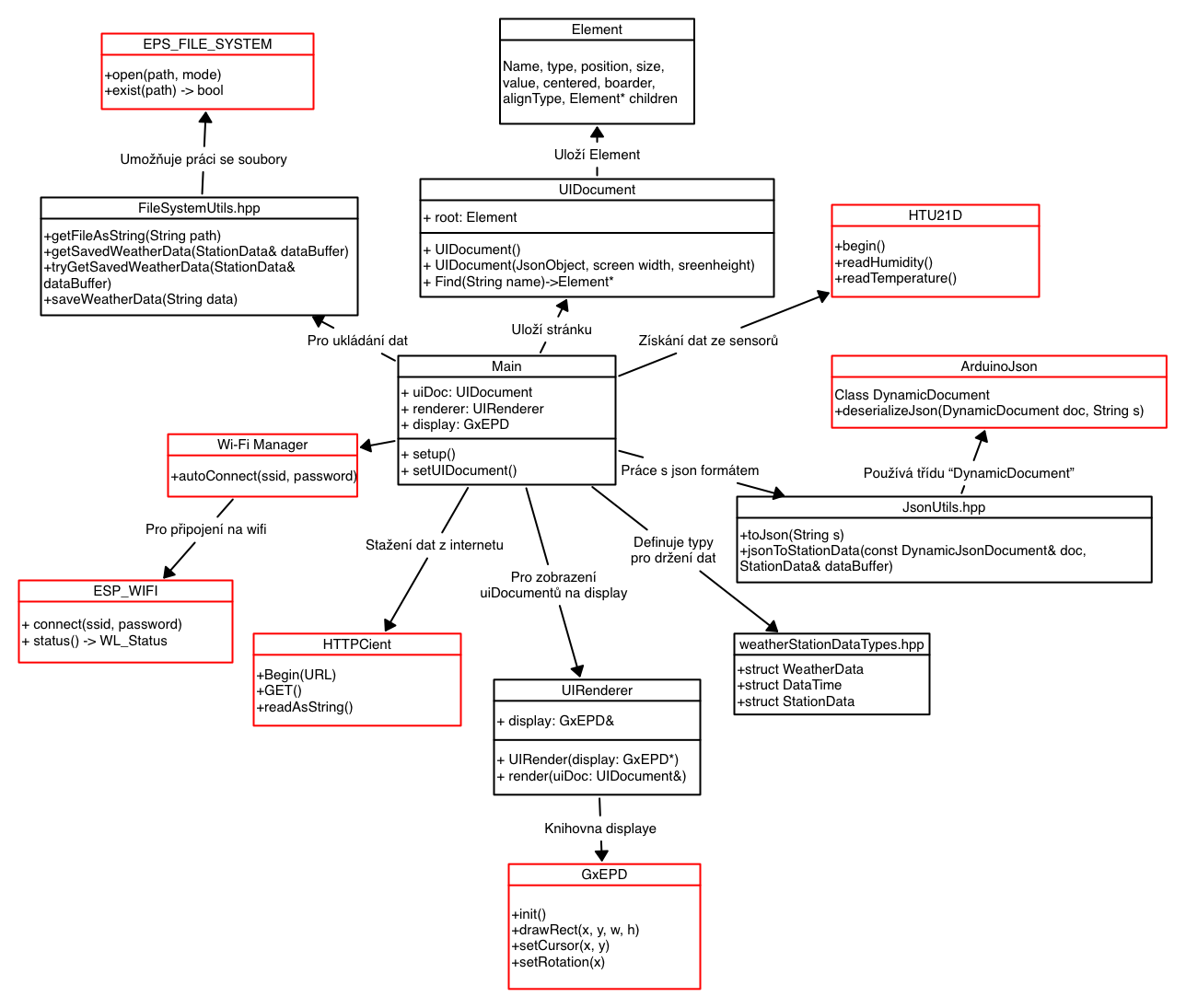
Data se na display zobrazují díky třídě UIRenderer.

## Přechod do deepsleep

Po zobrazení dat na displeji meteorologické stanice se stanice přepne do režimu hlubokého spánku, zvaného "deepsleep", po dobu 15 minut. Během této doby jsou vypnuty všechny nezbytné komponenty stanice, s výjimkou ESP32 mikrokontroléru, který zůstává v režimu hlubokého spánku, přičemž spotřeba energie se minimalizuje. Tento krok je důležitý pro prodloužení životnosti baterie stanice a zabraňuje zbytečné spotřebě energie, když stanice nevykonává žádnou funkci. Po uplynutí 15 minut se stanice probudí z režimu hlubokého spánku a opakuje cyklus.

# Struktura programu

## Class Diagram



Obr.16: Diagram tříd

Diagram ukazuje použité třídy a knihovny v programu. Červeně jsou označeny třídy, které jsem nenapsal, ale používám je v programu.

## Main

Hlavní část programu, kde se nachází funkce „setup“ a „loop“. Ve funkci „setup“ se odehrává všechen kód a po ukončení funkce jde stanice do deepsleepu.

Funkce v souboru umožňují:

* Připojení k WiFi a automatické připojení k předchozí síti při spuštění.
* Dotazování se na OpenWeatherMap API a získávání aktuálního počasí. Pokud dotaz selže, použije se uložené lokální počasí.
* Načítání a zobrazování dat na displeji pomocí knihoven GxEPD, UIRenderer a UIDocument
* Ukládání a čtení dat pomocí SPIFFS (SPI Flash File System).
* Používání HTU21D senzoru pro měření teploty a vlhkosti uvnitř.

## Setup

void setup()

{

init();

run();

Serial.println("Done going to sleep");

esp\_deep\_sleep\_start();

}

Obr.17: Funkce „setup“

Funkce "setup" se spouští na začátku programu a připravuje zařízení pro další běh. Spustí se funkce "init", která inicializuje zařízení, a nakonec se spustí funkce "run", která provede hlavní běh programu. Funkce "run" je hlavní funkcí programu, která řídí celý průběh aplikace. Nejprve inicializuje objekt "StationData", který slouží pro ukládání dat o počasí. Poté se inicializuje a vykreslí se stránka pro připojení k Wi-Fi sítím. Pokud se podaří připojení k internetu, tak se načtou data o počasí z webového API. Pokud k připojení nedojde, tak se načtou data o počasí z paměti. Nakonec se vykreslí hlavní menu aplikace s aktuálními daty o počasí. Po dokončení běhu funkce "run" se na konzoli vypíše zpráva "Done going to sleep" a zařízení přejde do hlubokého spánku pomocí funkce "esp\_deep\_sleep\_start".

## GxEPD display

GxEPD je knihovna pro Arduino a ESP32 platformy, která umožňuje snadné ovládání E-ink displejů (elektronických papírů) vytvořených společností Good Display. Tato knihovna podporuje různé velikosti a typy displejů, včetně barevných a černobílých, a umožňuje snadnou a rychlou integraci displeje do projektu s ESP32 mikrokontrolérem.

Knihovna GxEPD poskytuje funkce pro inicializaci displeje, vykreslování textu a grafiky, aktualizaci displeje a další. Podporuje také nízkou spotřebu energie a umožňuje zobrazovat informace na displeji po dlouhou dobu bez nutnosti častého obnovování.

Díky svým vlastnostem se GxEPD knihovna stala oblíbenou volbou pro vývojáře, kteří chtějí použít E-ink displeje v různých projektech, jako jsou například nízkopříkonové zařízení pro IoT aplikace, čtečky e-knih, nebo displeje pro náramkové hodinky a další podobné zařízení.

[15]

## UIDocument

Třída UIDocument je součástí knihovny pro tvorbu uživatelského rozhraní v prostředí Arduino a umožňuje vytvoření hierarchického stromu elementů, které tvoří uživatelské rozhraní. Tyto elementy jsou definovány v JSON formátu.

UIDocument::UIDocument(const JsonObject& jObject, int screenWidth, int screenHeight, const JsonArray\* classArray = nullptr): classes(classArray){

Serial.println("converting json to UIDoc");

root = jsonToElement(jObject);

Serial.println("json converted");

Element parent;

parent.posX = 0;

parent.posY = 0;

parent.sizeX = screenHeight;

parent.sizeY = screenWidth;

root = toRealCoord(root, parent);

alignChildren(root);

Serial.println("converted to UIdoc");

}

Obr.18: Konstruktor třídy UIDocument

Konstruktor třídy přijímá JSON objekt, šířku a výšku displeje a pole JSON tříd, které mohou být použity k definování vzhledu elementů. Metoda jsonToElement(), která je rekursivní, převádí JSON objekty na objekty typu Element, které jsou použity pro reprezentaci uživatelského rozhraní. V programu je použit druhý konstruktor, který si bere jako parametr cestu k soubory typu json. Tento soubor načte a konvertuje do třídy typu JsonObject a potom zavolá tento konstruktor.

Další metody třídy UIDocument jsou použity pro výpočet rozmístění elementů vzhledem k ostatním elementům v hierarchii, zarovnání elementů a nalezení konkrétních elementů pomocí jejich názvu. Třída obsahuje taku metodu printElement(), která slouží k výpisu hierarchie elementů v konzoli pro účely debugování.

### Json document

JSON (JavaScript Object Notation) je jednoduchý formát pro výměnu dat. Jedná se o textový formát, který slouží k reprezentaci strukturovaných dat. JSON dokument se skládá z kolekcí párů klíč-hodnota, které mohou obsahovat různé datové typy, jako jsou řetězce, čísla, pole a další objekty. Dokument začíná a končí složenými závorkami a každý pár klíč-hodnota je oddělen čárkou. JSON se používá v mnoha aplikacích, protože je snadno čitelný a zapisovatelný pro člověka i stroj.

### Element

Třída Element obsahuje různé vlastnosti, které popisují element v uživatelském rozhraní. Má proměnnou border, která určuje, zda má element okraj, výčtový typ ElementType, který popisuje typ elementu, proměnnou name, která udává název elementu, proměnné posX a posY, které určují pozici elementu na ose X a Y, a proměnné sizeX a sizeY, které určují šířku a výšku elementu.

Dále má proměnnou value, která udává hodnotu elementu, proměnnou textSize, která určuje velikost textu, výčtový typ AlignType, který určuje typ zarovnání, počet potomků (nOfChildren) a ukazatel na pole potomků (children) typu Element.

## UIRenderer

void UIRenderer::drawElement(const Element& e){

if(e.border){

drawBoarder(e);

}

if(e.type == ElementType::Text){

drawText(e);

}

for(int i = 0; i < e.nOfChildren; i++){

drawElement(e.children[i]);

}

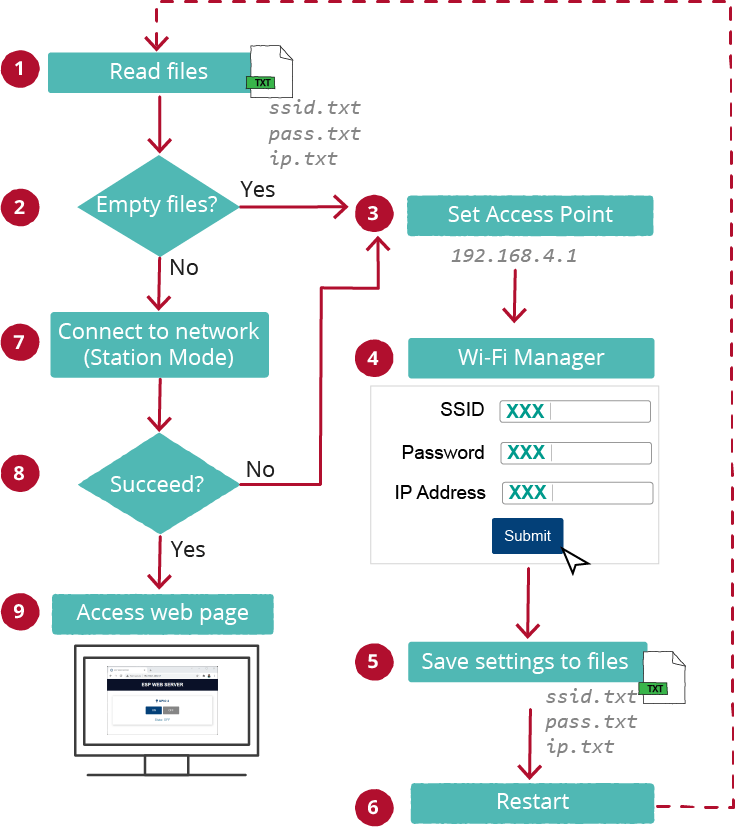
}

Obr.19: Ukázka funkce drawElement ve třídě UIRenderer

Třída UIRenderer slouží k vykreslování prvků uživatelského rozhraní na displej. Obsahuje metody pro vykreslení rámce prvku, textu a pro rekurzivní vykreslení celého dokumentu.

Konstruktor třídy "UIRenderer" přijímá ukazatel na displejový objekt typu "GxEPD\_Class", který je používán k vykreslení prvků na displej. Tato třída obsahuje tři funkce pro vykreslení různých prvků UI: "drawBorder" pro vykreslení rámečku prvku, "drawText" pro vykreslení textového prvku a "drawElement" pro rekurzivní vykreslení celého stromu prvků. Funkce "drawElement" je hlavní funkcí, která vykresluje celý strom prvků. Tato funkce rekurzivně prochází všechny prvky ve stromu a pro každý prvek volá příslušnou vykreslovací funkci. Pokud má prvek nastavenou vlastnost "border" na hodnotu "true", funkce "drawBorder" vykreslí rámeček kolem prvku. Pokud se jedná o textový prvek, funkce "drawText" vykreslí text v souladu s pozicí a velikostí prvku. Celý dokument reprezentovaný instancí třídy "UIDocument" lze poté vykreslit na displej voláním funkce "render" s ukazatelem na instanci "UIDocument" jako parametrem. Tato funkce zavolá funkci "drawElement" s kořenovým prvkem stromu prvků jako parametrem, která nakonec vykreslí celý dokument na displej.

## Wifi Manager



[16] Obr.20: Flow diagram wifiManageru

Wifi Manager je knihovna pro ESP32, která usnadňuje konfiguraci Wi-Fi připojení z mikrokontroléru. Tato knihovna umožňuje uživateli konfigurovat Wi-Fi připojení přímo z aplikace, což eliminuje potřebu ruční konfigurace pomocí sériového monitoru.

Wifi Manager funguje tak, že vytvoří vlastní Wi-Fi síť, ke které se uživatel může připojit a zadat své přihlašovací údaje pro připojení k cílové Wi-Fi síti. Tento proces se odehrává v prostředí, které je uživatelsky přívětivé a jednoduché na použití.

Po úspěšném připojení k cílové Wi-Fi síti se Wifi Manager ukončí a ESP32 se připojí k internetu. Toto usnadňuje konfiguraci Wi-Fi připojení z mikrokontroléru a umožňuje rychlé a snadné vytvoření aplikací pro IoT, které vyžadují připojení k internetu.

[17]

## File System

ESP32 obsahuje souborový systém, který umožňuje ukládání a správu dat na flash paměti mikrokontroléru. Tento souborový systém se nazývá SPIFFS (SPI Flash File System) a je obvykle používán pro ukládání konfiguračních souborů, webových stránek a jiných dat v aplikacích pro IoT.

SPIFFS je implementován jako virtuální souborový systém, který se nachází na vrcholu flash paměti a používá jednoduché rozhraní API pro práci se soubory. Tento souborový systém umožňuje snadné ukládání a načítání dat, a to i při běhu aplikace.

SPIFFS je velmi užitečný pro vývoj aplikací pro IoT, které vyžadují ukládání dat, a to zejména v případech, kdy nejsou k dispozici jiné úložné médium, jako například microSD karty.

String getFileAsString(String path){

if(!SPIFFS.exists(path)){

Serial.println("file not found! path:" + path);

return "";

}

File f = SPIFFS.open(path, "r");

if (!f) {

Serial.println("file open failed");

}

String s = f.readString();

f.close();

return s;

}

Obr.21: Ukázka použití ve funkci getFileAsString

## Adafruit BME280

Knihovna Adafruit BME280 umožňuje snadné a pohodlné čtení dat z digitálního senzoru tlaku, teploty a vlhkosti vzduchu BME280. Knihovna poskytuje metody pro inicializaci senzoru, kalibraci a čtení dat. Data jsou k dispozici ve formě numerických hodnot, které se snadno zpracovávají v kódu. Knihovna podporuje i pokročilejší funkce, jako například filtrování dat nebo změnu měřeného rozsahu.

Knihovna Adafruit BME280 je zdarma dostupná na webových stránkách společnosti Adafruit a může být snadno instalována do Arduino IDE pomocí správce knihoven. Knihovna byla vyvinuta pro platformu Arduino, ale může být také použita s jinými mikrokontroléry.

Použití v programu je jednoduché stáčí vytvořit instanci třídy HTU21D, zavolání funkce „begin“. Pro získání teploty použijeme funkci třídy „readTemperature“, která nemá žádný vstupní parametr a vrací hodnotu typu float, která už je převedena na stupně celsia. Pro získání vlhkosti použijeme funkci třídy „readHumidity“, která nemá žádný vstupní parametr a vrací hodnotu typu float, která už je převedena na procenta vlhkosti.

Ukázka použití v programu:

//reading data from sensor

float humd = indoorSensor.readHumidity();

float temp = indoorSensor.readTemperature();

Obr.22: příklad použití sensoru

[18]

## HTTPClient

httpClient je knihovna pro ESP32, která umožňuje mikrokontroléru komunikovat s webovými servery pomocí protokolu HTTP. Tato knihovna umožňuje mikrokontroléru odesílat HTTP požadavky a zpracovávat HTTP odpovědi.

HttpClient podporuje standardní metody HTTP, jako jsou GET, POST, PUT a DELETE, a také umožňuje nastavit vlastní hlavičky HTTP. Tato knihovna také umožňuje použití HTTPS připojení pomocí certifikátů.

HttpClient je užitečný pro vývoj aplikací pro IoT, které vyžadují přístup k internetu a komunikaci s webovými servery. Tato knihovna umožňuje mikrokontroléru získat data z webových stránek, odesílat data na webové servery a využívat různé webové API.

[19]

### REST API Request

REST API request je zpráva, kterou klient odesílá serveru, a obsahuje několik klíčových prvků, jako je URL adresa požadovaného zdroje, HTTP metoda, a možná také tělo požadavku v případě, že klient odesílá data na server.

**Metoda** - HTTP metoda určuje akci, kterou klient požaduje na zdroji na serveru. Nejběžnějšími metodami jsou GET, POST, PUT a DELETE.

**Cílová adresa** - cílová adresa (URL) specifikuje, na jaký zdroj na serveru se požadavek vztahuje. Adresa se skládá z protokolu, domény, portu (nepovinné) a cesty.

**Hlavičky** - hlavičky HTTP umožňují klientovi a serveru vyměňovat metadata spojená s požadavkem. Hlavičky obsahují informace o typu dat, jazyku, autentizaci a dalších.

**Tělo požadavku** - tělo požadavku obsahuje data, která klient odesílá na server. Toto tělo je nepovinné a může obsahovat různé typy dat, jako jsou JSON, XML nebo textové řetězce.

Ukázka použití HTTPClientu v programu:

HTTPClient client;

client.begin(PROTOKOL + HOST\_NAME + URL + APPID);

int http\_code = client.GET();

if(http\_code > 0){

String payload = client.getString();

//process payload

//...

}

Obr.23: příklad GET dotazu

## fileSystemUtils.hpp

Soubor "fileSystemUtils.hpp" obsahuje funkce pro práci souborovým systémem na platformě Arduino pomocí knihovny "FS" a "SPIFFS". Soubor slouží jako pomocník pro ukládání a načítání dat spojených s meteorologickou stanicí.

Funkce "getFileAsString" slouží k načtení obsahu souboru na dané cestě jako řetězec. Funkce "getSavedWeatherData" načítá uložená data o počasí a ukládá je do předané struktury "StationData". Pokud tato data nejsou k dispozici, funkce "tryGetSavedWeatherData" se pokusí o načtení dat a uložení do "dataBuffer", ale v případě neúspěchu nevyvolá chybu. Funkce "saveWeatherData" slouží k uložení dat o počasí do souboru na předané cestě.

Celkově tedy soubor poskytuje užitečné funkce pro ukládání a načítání dat o počasí, což může být pro meteorologickou stanici velmi užitečné.

void getSavedWeatherData(StationData& dataBuffer){

String s = getFileAsString(SAVED\_WEATHER\_DATA\_PATH);

DynamicJsonDocument doc = toJSON(s);

jsonToStationData(doc, dataBuffer);

}

Obr.24: Ukázka funkce getSavedWeatherData

## jsonUtils.hpp

Tento header file obsahuje deklarace funkcí pro práci s JSON formátem dat. Konkrétně jsou zde deklarovány funkce pro konverzi řetězce ve formátu JSON na dynamický objekt JSON a pro konverzi dynamického objektu JSON na instanci struktury "StationData", která je používána v projektu jako reprezentace dat ze stanice.

Tyto funkce jsou důležité pro práci s daty v projektu, protože formát JSON se používá pro ukládání a zpracování dat ze stanice a struktura "StationData" obsahuje informace o aktuálních meteorologických podmínkách na stanici.

## weatherStationDataTypes.hpp

Tato knihovna v programu definuje tři struktury: "WeatherData", "DataTime" a "StationData". Struktura "WeatherData" obsahuje dva členské proměnné typu "float" a "int", které reprezentují teplotu a vlhkost. Konstruktor této struktury inicializuje tyto hodnoty pomocí parametrů, které jsou předány do konstruktoru. Struktura "DataTime" obsahuje dvě členské proměnné typu "String", které reprezentují datum a čas. Struktura "StationData" obsahuje dvě instance struktury "WeatherData" pro online a offline data, členskou proměnnou typu "String" pro popis a instanci struktury "DataTime" pro časové údaje. Konstruktor této struktury inicializuje výchozí hodnoty pro všechny členské proměnné.

# Popis vývojového procesu

Vývojový proces meteorologické stanice s mikrokontrolerem ESP32 a e-ink displejem lze rozdělit do několika kroků:

## Vytvoření uživatelského rozhraní s e-ink displejem

* Návrh uživatelského rozhraní
* Vytvoření funkcí, které pracují s formátem json
* Implementace grafického rozhraní pomocí knihovny pro ovládání e-ink displeje
* Použití knihovny GxEPD

## Integrace připojení na wifi pomocí ESP32

* Implementace kódu pro inicializaci a připojení k wifi sítím

## Založení účtu na openweathermap.org

* Registrace na webových stránkách openweathermap.org
* Získání API klíče pro přístup k datům

## Integrace ESP32 HttpClientu na stahování dat z internetu

* Implementace kódu pro stahování dat z openweathermap.org pomocí HTTP GET požadavků
* Zpracování dat (např. parsování JSON)

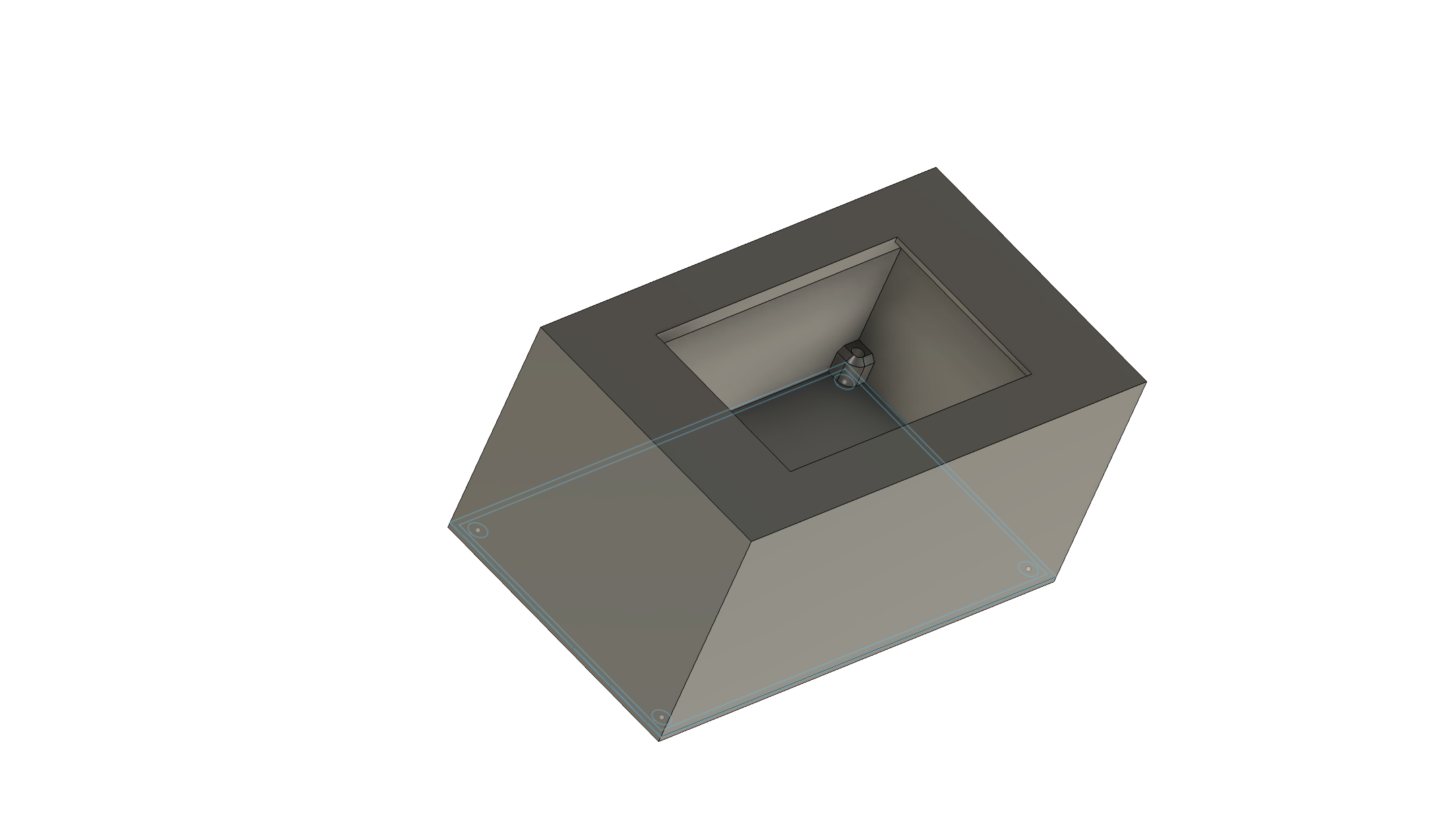
## Integrace ESP32 WifiManageru

* Implementace kódu pro automatickou konfiguraci wifi sítě pomocí webového rozhraní
* Umožnění uživateli změnit wifi síť a heslo bez nutnosti znalosti kódu

## Připojení senzoru HTU21D

* Implementace kódu pro čtení dat z senzoru HTU21D pomocí I2C rozhraní
* Využití dat z senzoru pro zobrazení aktuálního stavu vlhkosti a teploty v uživatelském rozhraní

## Vytvoření krabičky ve programu Fusion360



Obr.25: Ukázka modelu krabičky

* Vytvoření nového projektu, nastavení jednotek a měřítek.
* Tento tvar se dá vytvořit několika způsoby, např. vytvořením pravoúhlého tělesa a následným upravením rozměrů.
* Vytvoření díry v krabičce s rozměry odpovídajícími rozměrům displeje.
* Přetáhnutí stran do třetí dimenze.
* Exportování modelu: Hotový model se poté exportuje do souboru, který je možné otevřít v programu pro řízení 3D tiskárny a připravit tak model pro samotný tisk. V našim případě převedeme soubor z typu STL na GCODE. Tento krok se dělá pomocí programu Prusa Slicer.

## Tisk na 3D tiskárně

* Jakmile je model ve formátu GCODE, můžeme jej poslat do tiskárny.
* Doma mám tiskárnu „Prusa mk3“, na které jsem to vytiskl.
* Doba tisku byla 1 hodina 40 minut.

# Výsledky a zhodnocení

## Shrnutí dosažených cílů a zhodnocení projektu

Cílem této práce bylo vytvořit meteorologickou stanici, která bude měřit teplotu a vlhkost pomocí sensoru HTU21D a zobrazovat aktuální meteorologická data na e-ink displeji. Stanice by měla také být schopna připojit se na internet a stahovat aktuální meteorologická data z openweathermap.org. Pro připojení na internet byl použit mikrokontroler ESP32 s integrovaným e-ink displejem.

V první fázi bylo vytvořeno uživatelské rozhraní pro e-ink displej, které zobrazuje aktuální teplotu a vlhkost získanou ze senzoru HTU21D. Poté bylo integrováno připojení na Wi-Fi pomocí ESP32, aby bylo možné připojení k internetu.

Další krok byl založení účtu na openweathermap.org a následné použití ESP32 HTTP klienta pro stažení aktuálních meteorologických dat.

Byl také integrován Wi-Fi manažer pro snadné nastavení a připojení k síti. Poté byl sensor HTU21D připojen k mikrokontroleru a byl použit pro měření teploty a vlhkosti v reálném čase.

Nakonec jsem vytvořil a vytiskl krabičku pro stanici. Výsledkem je kompletní meteorologická stanice, která zobrazuje aktuální meteorologická data z internetu a měří teplotu a vlhkost pomocí senzoru HTU21D.

Celkově byla tato práce úspěšná a byly dosaženy všechny cíle. Meteorologická stanice je funkční a poskytuje uživateli aktuální meteorologická data a měří teplotu a vlhkost v reálném čase. Byly použity moderní technologie a nástroje, jako je mikrokontroler ESP32, e-ink displej a senzor HTU21D, což zajišťuje spolehlivý a přesný výsledek.

## Možná zlepšení

Zde jsou možná zlepšení práce:

* Implementace více typů senzorů pro získávání dalších meteorologických dat, jako je například měření srážek, vítr, sluneční záření a další.
* Integrace výstražného systému, který by uživatele upozornil na extrémní povětrnostní podmínky jako jsou bouřky, tání sněhu nebo silný vítr.
* Vylepšení uživatelského rozhraní, včetně přidání více funkcí, jako například zobrazování grafů nebo historických dat o počasí.
* Implementace automatického uploadu dat na cloudové úložiště, aby bylo možné sledovat data na více zařízeních nebo sdílet data s ostatními uživateli.
* Integrace GPS modulu pro získání geografických souřadnic a zobrazení aktuální polohy stanice na mapě.
* Vylepšení konstrukce a vzhledu stanice, aby byla odolnější a esteticky přitažlivější.

# Závěr

Závěr této práce je, že se mi podařilo úspěšně vyvinout meteorologickou stanici s mikrokontrolerem ESP32 a e-ink displejem. Stanice umožňuje sledování teploty a vlhkosti pomocí senzoru HTU21D, zobrazování aktuálního počasí a předpovědi počasí pomocí openweathermap.org API a zobrazení dat na e-ink displeji.

V rámci práce jsem se naučil programovat v jazyce C++ pro mikrokontroler ESP32 a pracovat s různými periferiemi, jako jsou e-ink displej a senzor HTU21D. Také jsem se seznámil s použitím openweathermap.org API a práci s HTTPClient knihovnou pro stahování dat z internetu.

Celkově lze tuto práci považovat za úspěšnou a má potenciál být inspirací pro další vývoj podobných zařízení v budoucnu.

# Použitá literatura

[1] OBR.1: C++ LOGO. Dostupné z:

https://www.ujudebug.com/wp-content/uploads/2021/03/c-plus-plus-Logo-ujudebug-e1617611636592.png

[2] OBR.2: VSCODE LOGO. Dostupné z: https://miro.medium.com/max/1400/1\*0LS0sRb2kjHtIIWtKAt-iw.png

[3] OBR.3: PLATFORMIO LOGO. Dostupné z:

https://pbs.twimg.com/profile\_images/614811335533199360/W75l6K75\_400x400.png

[4] OBR.5: EPS32 PINOUT. Internetový portál LYLIGO. Dostupný z: http://www.lilygo.cn/prod\_view.aspx?TypeId=50031&Id=1393

[5] EPS32. Internetový portál LYLIGO. Dostupný z: http://www.lilygo.cn

[6] OBR.6: SENSOR HTU21D. Internetový obchod Drátek. Dostupné z: https://dratek.cz/photos/produkty/f/1/1487.jpg?m=1644828724

[7] SENSOR HTU21D. Internetový obchod Drátek. Dostupné z:

https://dratek.cz/arduino/1487-iic-i2c-senzor-teploty-a-vlhkosti-htu21d.html

[8] OBR.7: GEB LI-ION BATERIE 3.7V. Internetový obchod Laskakit. Dostupné z:

https://cdn.myshoptet.com/usr/www.laskakit.cz/user/shop/big/8895\_geb-li-ion-baterie-1x18650-1s1p-3-7v-3200mah.jpg?63500064

[9] GEB LI-ION BATERIE 3.7V. Internetový obchod Laskakit. Dostupné z:

https://www.laskakit.cz/geb-li-ion-baterie-1x18650-1s1p-3-7v-3200mah/

[10] OBR.9: GITHUB LOGO. Dostupné z: https://external-content.duckduckgo.com/iu/?u=https%3A%2F%2Ftse2.mm.bing.net%2Fth%3Fid%3DOIP.sRloc7wLVwHYvX-cwrNBywHaEK%26pid%3DApi&f=1&ipt=d20fa564d6513a6b20c7e1e049563404ffb85e60a66749f85d0330d748378ff8&ipo=images

[12] OBR.11: OBECNÉ I2C SCHÉMA. Dostupné z: https://1.bp.blogspot.com/-P7K0A53gg3c/YMsFMbtJAqI/AAAAAAAABxA/gvom2kbI7TIB9aKLhtpA1DqhOIfRYhrPwCLcBGAsYHQ/w557-h283/I2C-Wiring.jpg

[13] OBR.12: OBECNÉ SPI SCHÉMA. Dostupné z: https://i.pinimg.com/originals/dd/23/7d/dd237d1c7d04fb77a949c93b6137fc7f.png

[14] OBR.15: OPENWEATHERMAP LOGO. Dostupné z:

https://brands.home-assistant.io/\_/openweathermap/logo@2x.png

[15] GxEPD. Knihovna na githubu. Dostupná z: https://github.com/ZinggJM/GxEPD

[16] OBR.20: FLOW DIAGRAM WIFIMANAGERU. Portál Random Nerd Tutorial, kapitola ESP32: Create a Wi-Fi Manager (AsyncWebServer library). Dostupný z:

https://randomnerdtutorials.com/esp32-wi-fi-manager-asyncwebserver/

[17] WifiManager Portál Random Nerd Tutorial, kapitola ESP32: Create a Wi-Fi Manager (AsyncWebServer library). Dostupný z:

https://randomnerdtutorials.com/esp32-wi-fi-manager-asyncwebserver/

[18] BME280. Knihovna na githubu. Dostupná z: https://github.com/adafruit/Adafruit\_BME280\_Library

[19] HTTPCLIENT. Knihovna na githubu. Dostupná z: https://github.com/espressif/arduino-esp32/tree/master/libraries/SPIFFS

# Seznam obrázků a tabulek

[[1] Obr.1: C++ Logo 7](#_Toc131462849)

[[2] Obr.2: VSCode Logo 8](#_Toc131462850)

[[3] Obr.3: PlatformIO Logo 8](#_Toc131462851)

[Obr.4: ukázka mé meteorologické stanice 10](#_Toc131462853)

[[4] Obr.5: EPS32 PINOUT 11](#_Toc131462854)

[[6] Obr.6: Sensor HTU21D 12](#_Toc131462856)

[[8] Obr.7: GeB Li-Ion Baterie 3.7V 13](#_Toc131462858)

[Obr.8: Krabička 14](#_Toc131462860)

[[10] Obr.9: GitHub Logo 15](#_Toc131462861)

[Obr. 10: Schéma zapojení 16](#_Toc131462862)

[[12] Obr.11: Obecné I2C schéma 17](#_Toc131462863)

[[13] Obr.12: Obecné SPI schéma 18](#_Toc131462864)

[Obr.13: Displej 19](#_Toc131462865)

[Obr.14: Flow diagram 20](#_Toc131462866)

[[14] Obr.15: OpenWeatherMap Logo 21](#_Toc131462867)

[Obr.16: Diagram tříd 23](#_Toc131462868)

[Obr.17: Funkce „setup“ 24](#_Toc131462869)

[Obr.18: Konstruktor třídy UIDocument 25](#_Toc131462871)

[Obr.19: Ukázka funkce drawElement ve třídě UIRenderer 26](#_Toc131462872)

[[16] Obr.20: Flow diagram wifiManageru 27](#_Toc131462873)

[Obr.21: Ukázka použití ve funkci getFileAsString 28](#_Toc131462875)

[Obr.22: příklad použití sensoru 28](#_Toc131462876)

[Obr.23: příklad GET dotazu 29](#_Toc131462879)

[Obr.24: Ukázka funkce getSavedWeatherData 30](#_Toc131462880)

[Obr.25: Ukázka modelu krabičky 32](#_Toc131462881)