

Střední průmyslová škola strojní   
a elektrotechnická a Vyšší odborná škola, Liberec 1, Masarykova 3

Sběr atmosférických dat

Maturitní práce

Autor **Martin Těhník**

Obor **Elektronické systémy, automatizace a sdělovací technika**

Vedoucí práce **Ing. Petr Zenkl**

Školní rok **2022/2023**

Anotace

Má práce je zaměřená na domácí automatizaci. Pomocí modulů M5stack je zaručen sběr dat (teplota, tlak a vlhkost). Vyznačuje se prácí s daty ze senzorů pomocí sběrnice UART a následné posílání dat pomocí MQTT protokolu za pomocí modulu ATOM-LITE DTU NB IoT od firmy M5stack, z kterého budou data vizualizována pomocí ve webovém rozhraní. Celý proces bude plně automatický, uživatel nebude nucen nic dělat. Výhodou bude snadná flexibilita a malý nárok na prostor. Výsledek mé maturitní práce bude použit týmem lidí, kteří mají na starosti testování výrobků, na monitoring referenčních hodnot, popřípadě na jednoduché ovládání osvětlení skrze relé.

Summary

My work is focused on home automation. Using M5stack modules data collection (temperature, pressure and humidity) is guaranteed. It is characterized by work with data from sensors using the UART bus and then sending data using the MQTT protocol using the ATOM-LITE DTU NB IoT module from M5stack, from which the data will be visualized using the REST API on the web interface. The whole process will be fully automatic, the user will not be forced to do anything. The advantage will be easy implementation and compliance. The result of the graduation thesis will be used by a team of people who are in charge of product testing, to measure reference values or to simply control lighting through a relay.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní práci vypracoval sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne

Martin Těhník

Poděkování

V první řadě bych chtěl poděkovat Ing. Petrovi Duňkovi za možnost vykonání dlouhodobé maturitní práce ve firmě Jablotron Controls s.r.o. Poté bych chtěl poděkovat Janovi Tichému za čas a rady, které mi poskytoval během vypracovávání mé maturitní práce. Dále bych chtěl poděkovat Ing. Petru Zenklovi za rady ohledně dokumentace k maturitní práci a v neposlední řadě bych chtěl poděkovat rodině za podporu.

Obsah

[Úvod 1](#_Toc127430354)

[1 Teoretická část 2](#_Toc127430355)

[1.1 Hardware 2](#_Toc127430356)

[1.1.1 UART 3](#_Toc127430357)

[1.1.2 Sériové komunikační protokoly 3](#_Toc127430358)

[1.1.3 Asynchronní komunikační protokoly 4](#_Toc127430359)

[1.1.4 RS 485 4](#_Toc127430360)

[1.1.5 Modbus 5](#_Toc127430361)

[1.2 Software 5](#_Toc127430362)

[1.2.1 MQTT protokol 5](#_Toc127430363)

[1.2.2 MQTT broker 6](#_Toc127430364)

[1.2.3 Easy MQTT 6](#_Toc127430365)

[1.2.4 WebSocket 7](#_Toc127430366)

[1.2.5 SSL protokol 7](#_Toc127430367)

[1.2.6 TLS (Transport Layer Security) 7](#_Toc127430368)

[1.3 Programovací jazyk 8](#_Toc127430369)

[1.3.1 Python 8](#_Toc127430370)

[1.3.2 Micropython 8](#_Toc127430371)

[1.3.3 Visual Studio Code 9](#_Toc127430372)

[1.3.4 UIflow 9](#_Toc127430373)

[1.3.5 AT commandy 10](#_Toc127430374)

[1.4 Rest API 10](#_Toc127430375)

[1.4.1 JavaScript Object Notation 11](#_Toc127430376)

[1.4.2 Postman 13](#_Toc127430377)

[2 Praktická část 14](#_Toc127430378)

[2.1 Pochopení potřebného HW a SW 14](#_Toc127430379)

[2.1.1 Osvojení M5stack modulů 14](#_Toc127430380)

[2.1.2 Reléový modul SDM-6RO 15](#_Toc127430381)

[2.1.3 Zálohování dat a využití GitHub repositáře 15](#_Toc127430382)

[2.1.4 Praktická zkouška REST API a JSON 16](#_Toc127430383)

[2.2 Tvorba SW pro M5stack moduly 17](#_Toc127430384)

[2.2.1 Ovládání Atom Lite pomocí AT commandů 17](#_Toc127430385)

[2.2.2 Volba MQTT serveru 17](#_Toc127430386)

[2.2.3 Tvorba spojení s MQTT serverem 18](#_Toc127430387)

[2.2.4 Vytváření základního UI 19](#_Toc127430388)

[2.3 Ovládání reléového modulu SDM-6R0 20](#_Toc127430389)

[2.3.1 Realizace ovládání v rámci jazyku Python 20](#_Toc127430390)

[2.3.2 !Ovládání! relé pomocí M5stack-core 20](#_Toc127430391)

[2.4 Test dosavadní práce 20](#_Toc127430392)

[2.5 Vizualice dat ve webovém rozhraní 20](#_Toc127430393)

[2.5.1 Úprava Homeassistant UI 20](#_Toc127430394)

[Závěr 21](#_Toc127430395)

[Seznam zkratek a odborných výrazů 22](#_Toc127430396)

[Seznam obrázků 23](#_Toc127430397)

[Použité zdroje 24](#_Toc127430398)

[A. Seznam přiložených souborů I](#_Toc127430399)

Úvod

Naše škola umožňuje, v rámci praktické maturitní zkoušky, volbu mezi dlouhodobou maturitní prací nebo laboratorním měření.

Zvolil jsem si dlouhodobou maturitní práci, protože vidím příležitost osobního rozvoje a možném kariérním postupu v dané firmě. K tomu přispívá i vlastní organizace maturitní práce, respektive zlepšení své disciplíny a organizace času.

Na maturitní práci jsem pracoval pravidelně každý čtvrtek ve firmě od 8. 9. 2022 do ……… a většinou i ve svém volném čase.

Cílem maturitní práce bylo měření atmosférických hodnot v těžko přístupných místech s požadavkem na snadnou a rychlou instalaci a bezdrátové odesílání naměřených dat do cloudu/webového prostředí.

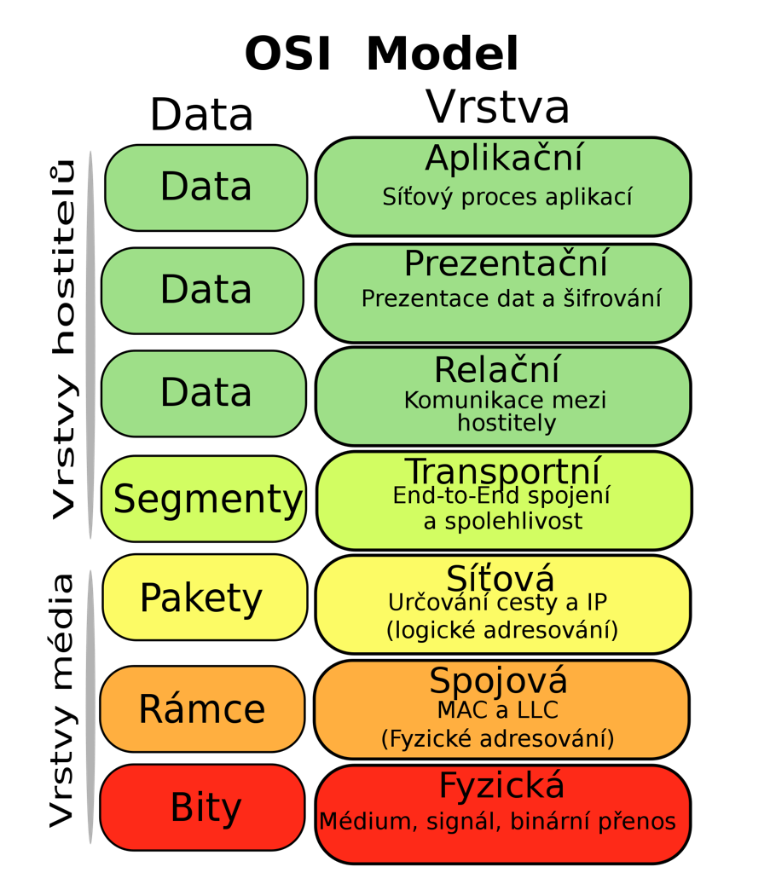
Pro realizaci cíle maturitní práce jsem použil ATOM NB-Iot module od firmy M5stack pro komunikaci se senzory atmosférických dat (teplota, tlak a vlhkost vzduchu), protože se jedná o malé a velmi schopné, programovatelné zařízení v poměru cena/výkon. Komunikace s těmito senzory probíhala na rozhraní UART. Dále se naměřená data posílají na MQTT server, ze kterého budou vyčítána a vizualizována ve webové aplikaci.

# Teoretická část

Jedná se o popis teoretické části. Popisuji a vysvětluji zde technologie, které jsem při práci použil nebo jsem s nimi pouze přišel do styku v podobě dalších alternativ.

## Hardware

V této kapitole se věnuji použitému Hardwaru neboli fyzickým komponentům, které jsem použil v rámci mé maturitní práce. Převážně popisuji komunikační protokoly, proto je důležité zmínit vrstvy do kterých se komunikace dělí v rámci ISO/OSI. Které mají pevně dané pořadí.

1. Fyzická
2. Spojová
3. Síťová
4. Transportní
5. Relační
6. Prezentační
7. Aplikační



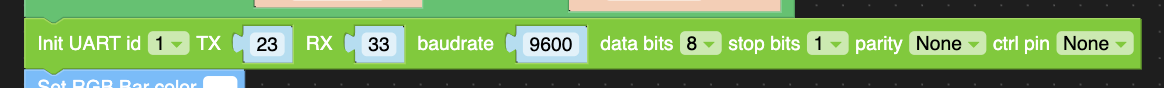
Obrázek: Schéma OSI modelu

### UART

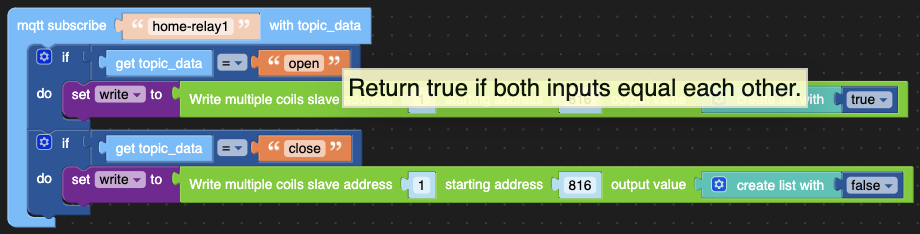
Zkratka UART znamená “Universal Asychronous Receiver-Transmitter”, v češtině to znamená “Univerzální Asynchronní Přijímač-Vysílač”. Tento komunikační protokol patří do sériových komunikací.

Na vysílači musí být stejné nastavení jako na přijímači, aby přenos dat byl funkční. Bity jsou posílány v tzv. “rámcích”. Každý rámec je ošetřen start bitem a stop bitem, který slouží k indikaci začátku a konce vysílané zprávy. Ke kontrole je potřeba paritní bit, který pozná neúplnost přenesených dat nebo chybu v jejich přenosu. Sběrnice UART zastřešuje například RS-232(12V) a RS-485(2, 5, 7, 12 a 24V).

Pro přenos je důležité stanovit rychlost přenosu, paritní bit. Sériové komunikace jsou daleko výhodnější než paralelní, díky své rychlosti přenosu a odolnosti vůči okolnímu rušení.



Obrázek : Ukázka mého nastavení pro UART



Obrázek : Modbus v rámci UART

### 

### Sériové komunikační protokoly

Pro posílání dat slouží jednotlivé bity a posílají jeden po druhém za sebou v tzv. rámcích. Sériové komunikace mají výhodu téměř “neomezené“ vzdálenosti, jelikož se dá jejich signál neustále reprodukovat pomocí opakovačů, ale dochází tím ke zpoždění a deformaci signálu.

Jsou odolné vůči rušení a přeslechům, což můžeme ocenit v zarušených prostředích (např. průmyslových halách). Sériová komunikace má bohužel i své špatné vlastnosti, a to je větší náročnost na hardware i software. Důležitá je také synchronizace dat a té je možné docílit pomocí přesného časování. Příklady sériových komunikačních protokolů známe v podobě ethernetu nebo SATA rozhraní.

|  |  |
| --- | --- |
| Sériová komunikace | Paralelní komunikace |
| Data se posílají za sebou (sekvenčně) | Data se posílají ve více datech zároveň |
| Vysoká přenosová rychlost na dlouhé trasy | Vysoká přenosová rychlost na krátké trasy |
| Jedna přenosová linka | Více přenosových linek |
| Minimální výskyt přeslechů na trase | Větší přeslechovost |

Tabulka 1: Porovnání Sériové a Paralelní

### Asynchronní komunikační protokoly

Signál, na rozdíl od synchronních protokolů, nenese hodinový signál pro synchronizaci. To zajišťuje start bit a stop bit (někdy 2 stop bity), které jsou vyslány společně s daty (data bit). Start a stop bit se vyskytují na začátku a konci každého rámce (jedné zprávy). Start dá přijímači signál pro příchod dat a stop ukončí přenos (proces synchronizace), aby se mohlo začít nové spojení. V rámci asynchronního vysílání nebere vysílač ohled na stav přijímače, protože se v rámci signálu neposílá hodinový signál. Asynchronní komunikační protokoly se vyskytují například v Ethernetu nebo například v USB rozhraních, popřípadě i e-mail je asynchronní komunikace.

### RS 485

Spadá do sériových komunikací a používá se především v průmyslu. Je navržen jako dvouvodičový half-duplexní sériový přenos. Half-duplex znamená, že obě strany spojení mohou přijímat i vysílat data, ale ne současně. Přenos probíhá vždycky jedním směrem. Příkladem této komunikace jsou třeba vysílačky. Maximální délka kabelu při použití této sběrnice činí 1200 metrů může mít maximálně 32 uzlů na trase (s opakovači jich může být více). Má velkou podobnost s RS-232, liší se především v napěťových úrovních určených pro ovládání přenosu. Mezi další výhody patří složení sběrnice RS485 z rozšířenější RS-232. Je ovšem nutné využít převodníky napěťových úrovní, jelikož se obě technologie liší.

Obrázek 1: UART přenos

### Modbus

Modbus je aplikační protokol především průmyslové řešení pro komunikaci různých, ale dá se použít i třeba v domácnosti. V průmyslu se používá převážně pro PLC, dotykové displeje nebo I/O rozhraní. Funguje na principu master a slave. Master je zařízení (může jich být více), které vysílá nějaký pokyn popřípadě zprávu a slave slouží pro vykonání daného pokynu nebo odpovídá na zprávu. Na modbusu funguje řada dalších verzí protokolů, například RS-232, RS-485 nebo TCP/IP. Reálným příkladem může být třeba PLC, které má periferie, jež ovládá. V tomto příkladu je PLC masterem a periferie slave.

### M5stack

M5stack je firma zabývající se výrobou programovatelných mini-počítačů postavených na principu ESP32, které slouží pro rychlou a jednoduchou aplikaci v těžce dostupných místech z hlediska prostoru, a hlavně dostupnosti internetového připojení.

ESP32 jsou mini-počítače s nízkým výkonem, díky tomu jsou idální jednoduché instalace. Jsou programovatelné několika mo*ž*nými aplikacemi např. VScode, Arduino IDE nebo blokově pomocí UIflow. Využívají programovacího jazyku Javascript/Micropython, které umožňují aplikování mnoha knihoven pro komunikaci mezi zařízeními, což značně usnadní práci s těmito moduly. Některé moduly mají programovatelný displej a tlačítka, což umožňuje jednoduchou vizualici přijatých dat nebo tvorbu jednoduchých aplikací.

V mém případě se jedná o verzi M5stack core, která je vybavena displejem a tlačítky. Druhý modul je Atom Lite s NB IoT rozšířením, které slouží ke komunikaci po NB (narrow band) síti, která využívá strukturu mobilních operátorů a využívá pásmo o velikosti 200 kHz. Převážně se tyto moduly používají v chytrých domácnostech nebo mezi domácími kutily.

### Reléový modul SDM-6RO

Představuje soubor více relé pohromadě od firmy Aspar z Polska, které jsou ovladatelné především pomocí PLC s využitím sběrnice RS485, s kterou je propojen pomocí twistovaného páru, nebo pomocí PC s využitím Modbus protokolu. Tento modul je ovladatelný pomocí softwaru od výrobce, který slouží ke konfiguraci. Je uzpůsoben pro uchycení na DIN lištu nebo může být pověšen na zdi. Každé relé má tři výstupy, NC (normally closed), NO (normally open) a common. Výstupy jsou z důvodu vyšší bezpečnosti galvanicky oddělené. Relé modul je doplněn o indikační ledku, která indikuje komunikaci po RS485.

## Software

V této kapitole se zabývám Softwarem neboli vyvinutými programy/použitými aplikacemi. Popisuji zde i TLS a SSL, které jsem sice nevyužil, jelikož využívám veřejný MQTT server, ale je to nedílná součást v rámci identifikace v protokolu MQTT.

### MQTT protokol

MQTT neboli Message Queue Telemetry Transport je postaven na protokolu TCP/IP. Využívá se pro přenos malého objemu dat (MQTT je limitováno do 260MB), což je právě případ mé maturitní práce. MQTT spadá do asynchronních komunikačních protokolů. Při navázání MQTT spojení je také potřeba definovat port, na který budou data adresována a následně vyčítána.

Posílání dat se dále dá ošetřit pomocí QoS (quality of service), který má tři úrovně 0,1,2. QoS 0 znamená, že se zpráva pošle bez potvrzení o doručení a není jisté její doručení. QoS 1 říká, že zpráva je doručena aspoň jednou a QoS 2 značí doručení zprávy právě jednou.

Propojení může být realizováno i pomocí websocketů, které umožňují obousměrnou komunikaci s webovými servery. Nabízí se použití protokolu HTTP, který je však více náročný na použití



Obrázek : Schéma propojení pomocí MQTT

### MQTT broker

Broker je software, který spustí prostředníka (server) při komunikaci přes MQTT protokol. Komunikace probíhá mezi publisherem – serverem –subscriberem.

Publisher je zařízení, které publikuje uživatelem danou zprávu na daný server. Subscriber je zařízení, které publikovaný obsah sbírá a dále ho může zpracovávat.

Publikovaný obsah se dále dělí podle tzv. “topic“, který slouží k rozdělení poslaných dat do různých témat. Broker není omezen počtem připojených zařízení, ani počtem zařízení, která publikují/odebírají daná témata.

Zde uvádím příklady MQTT brokerů

* HIVEMQ (Open source, ale i placená privátní verze)
* Mosquitto (Může běžet na vlastním zařízení nebo u brokera, Open source a placená verze)
* EMQ X (Opět open source i placená verze, nevýhodou je poměrně vysoká cena u placených verzí)



Obrázek : Dělení dle Topic

### Easy MQTT

Tento program jsem používal v rámci testování MQTT spojení mezi zařízeními.

Uživatel musí zadat jméno, adresu serveru, port, na který se připojit. V případě, že server vyžaduje autentifikaci pomocí uživatelských údajů, tak i ty se musí vyplnit. Některé servery, speciálně ty privátní placené, vyžadují autentifikaci pomocí brokerem generovaného certifikátu. Pak si uživatel vybere pouze jestli chce “Subscribe” nebo “Publish” a může začít testovat. Výhodou je nenáročnost na znalost programování.

### WebSocket

Jedná se o počítačový protokol, který funguje jako full-duplex díky TCP spojení. TCP je nejpoužívanějším protokolem transportní vrstvy v sadě protokolů TCP/IP, které se používají ke komunikaci přes internet. TCP protokol zajišťuje také správně doručení a pořadí dat.

Full – duplex znamená obousměrný přenos dat v reálném čase. Websocket je určen primárně pro použití ve webových stránkách, prohlížečích a serverech, ale dá se použít i v jiných aplikacích. Umožňuje spojení s prohlížečem a webovým serverem s menší režií a tím usnadňuje přenos dat mezi servery.

### SSL protokol

SSL (Secure Socket Layer) je vrstva vložená mezi transportní (TCP/IP) a vrstvu aplikační, která poskytuje zabezpečení komunikace a její šifrování. SSL dělá například z HTTP zabezpečený protokol.

Každá strana vzájemné komunikace má dvojici šifrovacích klíčů (veřejný a soukromý). Veřejný klíč se dá zveřejnit a pokud je tímto klíčem zašifrovaná zpráva, tak ji může rozšifrovat pouze majitel použitého veřejného klíče svým soukromým klíčem.

### TLS (Transport Layer Security)

Je nástupce SSL protokolu a zabraňuje odposlouchávání a falšování zpráv. Pomocí šifrování poskytuje TLS uživatelům soukromí při komunikaci s použitím Internetu. Šifrován je pouze server, uživatelé však nikoliv. Další vrstva zabezpečení, kdy oba uživatelé mají ověření s kým komunikují (vzájemná autentizace).

Dělí se na tři základní fáze. První je dohoda účastníků na podporovaných algoritmech. Druhá fáze obstarává výměnu klíčů na šifrování s veřejným klíčem. Třetí šifruje provoz symetrickou šifrou.

## Programovací jazyk

Slouží k syntaktickému zápisu programů/algoritmů, které může počítač nebo mikroprocesor vykonávat. Výsledkem programování je nějaký program, který je spustitelný. Programovací jazyky se dělí na kompilované a interpretované.

Kompilované jazyky se musí nejprve přeložit z kódu zdrojového na strojový a pak je možné daný program spustit.

Interpretované jazyky potřebují zdrojový kód, který je spustitelný za pomocí interpreteru, což je součást většiny editorů kódu.

V podstatě se jedná o komunikační prostředek mezi programátorem a počítačem. Při psaní programu je nutné se držet pravidly (odborně-syntaxí) podle používaného jazyka.

Například:

* V jazyce C# musí každý řádek, kromě definování funkcí, končit středníkem.
* V jazyce Python se naopak řádek nijak neukončuje.
* V jazyce C# je potřeba definovat datový typ proměnné. (integer, string, var…)
* Jazyk Python je v tomto ohledu daleko jednodušší a definování typu proměnné nevyžaduje

### Python

Python je z rodiny interpretovaných jazyků a byl navržen v roce 1991. V současnosti patří mezi nejpoužívanější jazyk, především pro svou univerzálnost. S tímto programovacím jazykem je možné tvořit ať už nějaké backend mechanismy nebo celé počítačové/mobilní aplikace. Backend je typ programu nebo jeho části, která není vidět.

Typicky to může být například umělá inteligence nebo výpočetní algoritmy. Jedná se o open source projekt, což s sebou nese spoustu výhod, jako například spousty zdarma dostupných instalačních balíčků.

### Micropython

Micropython je velmi zjednodušený oproti klasickému Pythonu. Je optimalizován tak že, se dá použít při programování microkontrolerů.

Jednoduchost syntaxe z něj dělá ideální programovací jazyk pro začátečníky. I přes jeho jednoduchost je možné v něm dělat pokročilejší algoritmy a programy. ////////////

### Visual Studio Code

Jedná se o editor zdrojového kódu od firmy Microsoft, ve kterém jsme schopni editovat a psát zdrojový kód. Je to rozhraní, kde je uživatel schopen psát kód pomocí textových příkazů na řádcích.



Výhodou VSC = Visual Studio Code je veliká univerzálnost, jelikož tento program slouží pro programování v téměř všech programovacích jazycích, což z něj dělá komplexní nástroj pro tvorbu kódu.

Obsah obrázku text

Popis byl vytvořen automaticky

Obrázek: Ukázka VSC rozhraní

### UIflow

Jedná se o prostředí vytvořené pro programování modulů od firmy M5stack ve webovém prostředí, což značně usnadňuje přístupnost. Programování je možné jak ve formě bločků, které zastávají nějakou funkci, tak pomocí ručně psaného kódu. Ruční psaní v tomto rozhraní není až tak dobrá volba, jelikož psaný kód musí být zpětně kompatibilní s blokovým kódem, a to vytváří jistá omezení v kreativitě.

### AT commandy

AT příkazy jsou příkazy pro modemy a jiné zařízení, které jsou schopny komunikovat přes sériovou linku nebo síťový port. Příkazy AT jsou zkratkou pro Attention a jsou používány k ovládání modemů a dalších zařízení přes sériové rozhraní. Například příkaz „AT+CSQ” zjišťuje kvalitu WiFi signálu nebo AT+CGMR, který slouží pro kontrolu firmwaru. Existuje mnoho dalších příkazů AT, které jsou používány k ovládání modemů a dalších zařízení. Každé zařízení by mělo mít svou tabulku AT commandů, které se pro dané zařízení používají.

## Rest API

Slouží jako veřejné rozhraní pro vzdálenou komunikaci pomocí jakékoliv aplikace či zařízení. To v praxi znamená, že jakákoliv aplikace z jakéhokoliv jazyka by měla být schopná s REST API pracovat a připojit se na něj. Dělí se na čtyři úrovně.

Počínaje nultou úrovní, která má na starosti přenos pomocí protokolu http (hyper text transfer protocol), který je dnes v tomto ohledu nejpoužívanější.

První úroveň slouží k rozlišení poslaných dat, aby se neposílaly na jeden hlavní bod. Každý zdroj je dále strukturován dle obsahu do dalších “záložek”, například “GET/cities” nám vrátí seznam měst.

Druhá úroveň má na starosti tzv. “http verbs”, což jsou vlastně metody, které určují co se stane. Nejznámější je metoda GET, pomocí které jsme schopni z nějakého API dostat data a dále s nimi pracovat. Dále jsou to například POST, DELETE, OPTIONS nebo PATCH. Důležité jsou také stavové kódy, které signalizují co se děje, respektive jsou posílány jako odpověď na nějaký REQUEST. Důležitým požadavkem je udržet REST API bezstavové, což v praxi znamená, že například ověření uživatele nebude podléhat cookies.

Třetí úroveň je známá jako “HATEOAS” (Hypertext as the Engine of Application State). Její využití je v případě navrácení dat, kdy se vrací společně s odkazy na další zdroje, které se následně řetězí. Díky tomu klient není závislý na URL, respektive mu stačí pouze ta základní. Vyčítaná data z REST API se posílají nejčastěji v textovém formátu JSON, se kterým umí pracovat většina jazyků. Dají se použít i jiné formáty, jako třeba XML.

### JavaScript Object Notation

Ve zkratce “JSON“ je způsob zápisu dat, který je nezávislý na platformě. JSON je určený pro přenos dat, která mohou být organizována v polích nebo třeba zasazená v objektech. Vstupem JSONu může být prakticky cokoliv, například číslo, objekt nebo pole, a jeho výstupem je vždy řetězec dat. Vstup JSONu je tedy téměř neomezený, což umožňuje pracovat opravdu s jakýmikoliv daty. Výsledný text je kódován ve formátu UTF-8. Toto je soubor JSON z rest API od firmy Accuweather, která vyvíjí aplikaci o počasí. Tento soubor

},

"GeoPosition": {

"Latitude": 50.088,

"Longitude": 14.421,

"Elevation": {

"Metric": {

"Value": 197.0,

"Unit": "m",

"UnitType": 5

},

"Imperial": {

"Value": 647.0,

"Unit": "ft",

"UnitType": 0

}

}

},

### Postman

Postman je program pro testovaní rest API viz. ukázka JSON. Je podoný programům pro testování MQTT jako je například MQTTX.

Uživatel zadá pouze API key generované zpravidla danou stránkou, popřípadě zadá nějaké specifické slovo pro vyhledávání. Například v případě JSON ukázky bylo potřeba zadat API key a k tom quarry parametr, který byl v mém případě „prague“. Prague proto abych našel údaje k městu Praha, bez zadání tohoto parametru by to nemohlo fungovat.

Výhodou je uživatelská nenáročnost, protože uživatel nemusí umět programovat, aby se dostal k JSON souboru. Dále je to rychlejší než tvorba programu, který by dělal to samé.

# Praktická část

Praktická část se věnuje popisu jednotlivých kroků, které jsem musel podniknout pro dokončení maturitní práce. V praktické části dochází ke skloubení teoretických vědomostí a aplikací daný vědomostí.

## Pochopení potřebného HW a SW

Pro pochopení daného hardwaru a softwaru jsem ze začátku používal především dokumentace výrobců, odborné články nebo edukativní youtube videa.

### Osvojení M5stack modulů

Seznámení s M5stack moduly jsem praktikoval pomocí jednoduchých programů v rámci UIflow. Zkoušel jsem jednoduché instrukce na M5stack core díky možnosti vizualizace dat na displeji.

V rámci M5stack core jsem začal jednoduchou funkcí IF, která při stisku tlačítka zobrazila předem definovaný text na displeji. Pokračoval jsem dále různými vizualizacemi na displeji, jelikož UIflow obsahuje knihovny pro jednoduchou vizualizaci.

Na modulu ATOM LITE jsem si vyzkoušel jeho ovládání pomocí AT commandů po sběrnici UART. Toto řešeni mi nevyhovovalo pro jeho omezenou využitelnost vlivem přesné formulace AT commandů ve kterých mi vždy něco chybělo, například při komunikaci s MQTT serverem jsem nebyl schopen pracovat s přečtenou zprávou dál v programu.

Obsah obrázku interiér, kuchyňské spotřebiče

Popis byl vytvořen automatickyObsah obrázku zeď, interiér

Popis byl vytvořen automaticky

Obrázek : Atom Lite+NB Iot modul

Obrázek : M5stack core modul

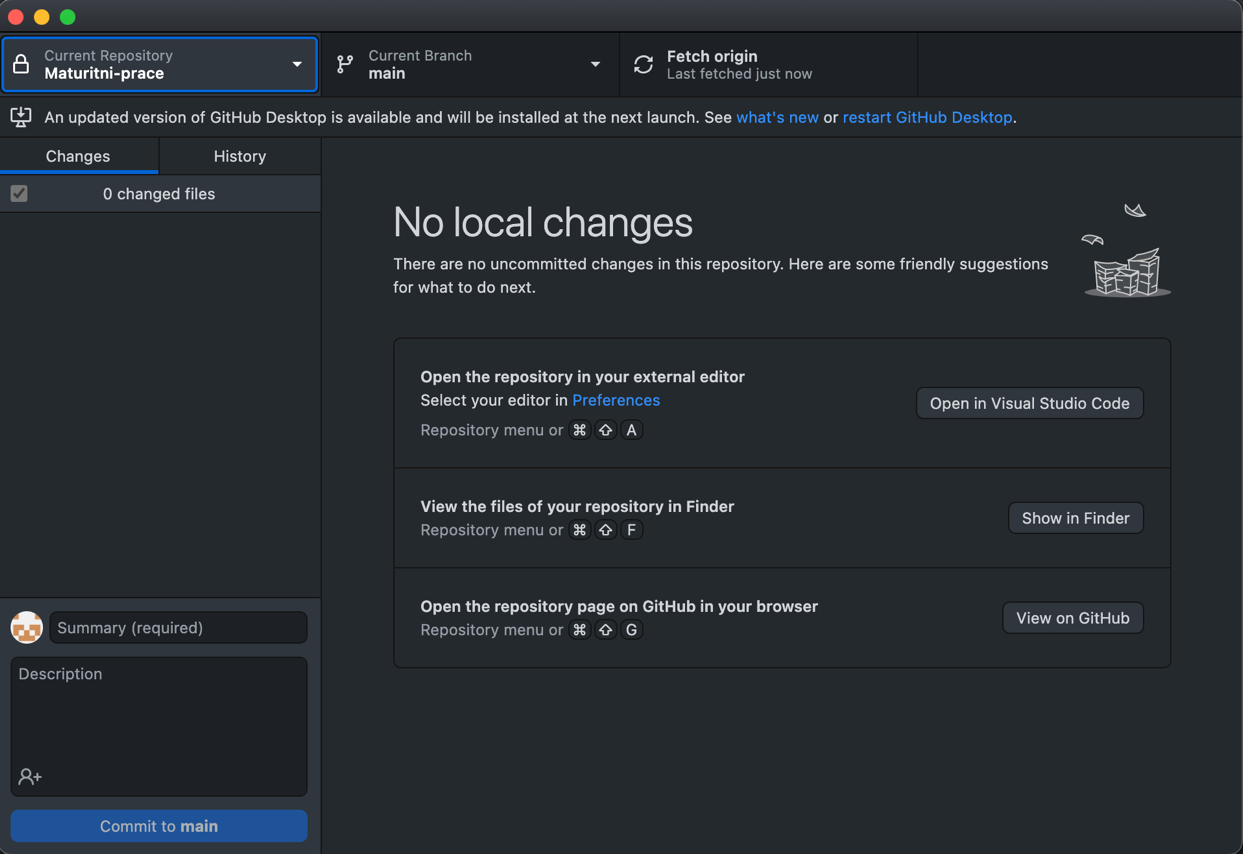
### Práce s relé modulem

Nejprve jsem musel upravit kabeláž, abych mohl relé modul ovládat pomocí počítače nebo M5 modulů. Připojil jsem 3 vodiče do svorek pro RS485 a ty jsem následně zapojil do převodníku z USB na RS485. Prostudoval jsem si dokumentaci a následně vyzkoušel jeho funkci skrze program tvořený pro tento modul.

### Zálohování dat a využití GitHub repositáře

Zálohování je nejvíce podceňovaným aspektem dlouhodobých prací. Proto jsem se tomu nějaký čas věnoval, abych o svá data v budoucnu nepřišel. Existuje mnoho možností zálohování ať už se jedná o fyzickou (flash disky, externí disky) nebo cloudové řešení (Onedrive, Google drive nebo právě GitHub).

Já si vybral GitHub především pro jeho jednoduchost, popularitu mezi programátory a na doporučení konzultanta. Funguje na principu správy sdílených složek, které jsou dostupné na vybraných zařízeních a webovém rozhraní GitHubu, které mají přístup k těmto složkám. Uživatel upraví soubor ve sdílené složce a následně musí použít tzv. “PUSH” aby se změna souborů v této složce mohla objevit na webovém rozhraní nebo na jiném zařízení ve stejné složce. V případě přeposílání souborů mezi zařízeními je nutné, aby zařízení, co má změnu souboru přijmout, použilo příkaz “PULL”. Pull slouží pro stažení dat do lokálního úložiště.



Obrázek: Rozhraní GitHub aplikace

### Praktická zkouška REST API a JSON

Praktickou zkoušku Rest API jsem prováděl pomocí veřejných služeb, které poskytují svoje API pro testovací účely nebo pro open source řešení. V mém případě se jednalo o službu Accuweather, která spravuje údaje o počasí z celého světa. Vytvořil jsem si jednoduchou terminálovou aplikaci v pythonu pro čtení atmosferických hodnot o daném místě pomocí rest API ze serveru Accuweather.

## Tvorba SW pro M5stack moduly

Tvorba SW probíhala v oficiálním webovém rozhraní UI flow od výrobce modulů M5stack, každá funkce modulů byla zároveň naprogramována pomocí Visual Studio Code a otestována na PC. Tudíž dané funkce a ovládání je aplikovatelné i na jiné systémy, které jsou schopné spustit programy vytvořené v jazyku Python.

### Ovládání Atom Lite pomocí AT commandů

Toto byl jeden z prvních kroků pro ovládání Atom modulu. Pomocí AT commandů jsem zjistil základní vlastnosti Atom modulu jako například stav WiFi signálu nebo jsem mohl spustit kontrolu firmwaru daného modulu. Následně jsem se pokusil zrealizovat MQTT most. AT commandy se dají použít i pro složitější funkce, ale jsou poměrně nepřehledné a omezené viz. Příklad připojení k Vodafone MQTT serveru.

uart1 = machine.UART(1, tx=22, rx=19)

uart1.init(9600, bits=8, parity=None, stop=1)

uart1.write('AT+CMQNEW="mqtts://IEZ000246.mqtt.ioteasyconnect.cz","1883",12000,100'+"\r\n")

uart1.write('AT+CMQCON=0,3,"myclient",600,0,0[IEZ000246:IEZ000246,Encantr1892!]'+"\r\n")

uart1.write('AT+CMQSUB=0,"mytopic",1'+"\r\n")

uart1.write('AT+CMQPUB=0,"mytopic",1,0,0,8,"31323334"'+"\r\n")

print\_uart()

### Volba MQTT serveru

Původně bylo v plánu využít služby Vodafone, jejíž produkty využívá Jablotron. Vlivem nedostatečné dokumentace této služby. Jsem při vývoji aplikace používal open source řešení od firmy Eclipse s názvem Mosquitto a spouštěl jsem svého MQTT brokera na svém PC. Aby řešení bylo aplikovatelné v praxi, bylo potřeba brokera změnit na funkčního a vybral jsem HIVEMQ.

### Tvorba spojení s MQTT serverem

Při tvorbě spojení s MQTT serverem bylo potřeba využít jednu z knihoven od M5stack, AT commandy nebo knihovnu přímo pro modul NB-IoT. Při použití AT commandů jsem narazil na již zmiňovanou nedokonalost tohoto řešení. Problémem bylo další použivání proměnné s hodnotami poslanými přes MQTT server. Od tohoto řešení jsem tedy upustil a využil jsem dočasně řešení od firmy Eclipse, jménem Mosquitto. Nakonec jsem použil integrovanou knihovnu v UIflow prostředí pro připojení k MQTT brokeru viz. ukázka kódu, kde první část je inicializace MQTT spojení + odebíraní tématu home-relay, které slouží k bezdrátovému ovládání relátek a druhá část slouží k posílání hodnot ze senzorů na centrální jednotku, kterou je M5stack modul s displejem.

m5mqtt = M5mqtt('mqtt', '192.168.229.212', 1883, '', '', 300)

m5mqtt.subscribe(str('home-relay'), fun\_home\_relay\_)

m5mqtt.start()

while True:

m5mqtt.publish(str('home-temp'), str(home\_temp\_value), 0)

m5mqtt.publish(str('home-press'), str(home\_press\_value), 0)

m5mqtt.publish(str('home-hum'), str(home\_hum\_value), 0)

wait(10)

wait\_ms(2)

### Vytváření základního UI

Dále jsem se vrhnul na tvorbu základního UI pro M5stack core. M5stack core v podstatě zastupuje funkce termostatu, jelikož zobrazuje atmosférická data (teplotu, vlhkost a tlak) a stav relé v rámci relé modulu. Je možné jím ovládat relé moduly, které mohou spínat například topení.

Vizualice teploty, vlhkosti a tlaku byla udělána pomocí funkce „Subscribe“ v rámci MQTT serveru. Ovládání relé modulu pomocí tlačítek, které díky příkazu „Publish“ posílají zprávu open/close na M5 Atom a tím je řízen relé modul. Dále jsem přidal indikátory stavu relé, které jsou závislé na zprávě poslané přes MQTT, tím se eliminuje milná indikace stavu. Přidal jsem i indikátor napájení v pravém horním rohu obrazovky s animací nabíjení.

Obsah obrázku text, zeď, monitor

Popis byl vytvořen automaticky

Obrázek: Uživatelské rozhraní M5stack

## Ovládání reléového modulu SDM-6R0

V případě relé modulu jsem si musel nastudovat dokumentaci, abych mohl ho mohl resetovat, jelikož už by použitý. Po provedení prvotního nastavení jsem se mohl vrhnout na zprovoznění.

Pro základní otestování funkce jsem si udělal jednoduchý script s využitím „pymodbus“ knihovny, který pomocí USB převodníku na RS485 ovládal relé, které bylo dané specifickou adresou. Adres byly vypsané v dokumentaci relé modulu.

Tím jsem si ověřil, že relé modul je funkční a mohl jsem jít tvořit software pro M5 moduly.

from pymodbus.client import ModbusSerialClient as ModbusClient

#digital\_inputs = 817

client = ModbusClient(method='rtu', port='/dev/tty.usbserial-0001', baudrate=19200, parity="N", stopbits=1, timeout=2)

client.connect()

"""counter\_err = 0

counter\_read = 0"""

read = client.write\_coil(address=0x330, value=False, slave=1)

print(read)

Touto částí jsem se zabýval asi nejdéle ze všech v rámci maturitní práce. V UIflow existuje mnoho knihoven, které se dají použít, ale žádná mi nefungovala, jelikož jsem nebyl schopný změnit stav relé. Indikační LEDka nevykazovala žádný přenos dat. Zkusil předem definované bloky z knihoven pro UART, knihovnu pro rozšíření DTU NB IoT, nebo knihovnu pro Modbus a ani jedna neposílala signál za použití bloku „write coils“. Zkoušel jsem to ověřit i pomocí osciloskopu a skutečně vodičem nešel žádný signál.

### Ovládání relé pomocí M5stack-core

V této části jsem strávil nejvíce času, díky nejasnosti dokumentace UIflow od výrobce, a navíc díky nefunkčnosti některých bloků jsem byl téměř odepsaný.

Začal jsem studováním dokumentace MODBUS knihovny, která je přímo implementovaná v rámci M5 UIflow. Ta byla bohužel zastaralá o několik verzí dozadu a nereflektovala současný stav této knihovny. Zkusil jsem to po svém a dal se na vlastní pokusy, které však skončili neúspěšně.

Díky předchozím pokusům v rámci VSC a díky dokumentaci modulu SDM-6RO jsem zjistil, že adresy modulu jsou typu coil.

Zkoušel jsem tedy bloky, které zapisovaly do coilů viz. obrázek. Zde jsem vyplnil vše dle dokumentace k modulu, a i vyzkoušeného kódu a bohužel nic nefungovalo, ani indikační LED na modulu nevykazovala přenesená data.

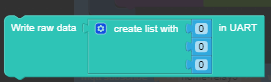


Obrázek: Ukázka bloku pro MODBUS

Zkusil jsem i dokumentaci k druhé knihovně v rámci modulu DTU NB-IoT, kde dokumentace byla kupodivu lepší, ale opět nefunkční. Adresy jsem měl nastavené správně akorát „output value“ měla být typu bool, tedy TRUE nebo FALSE. Zkusil jsem tedy zaměnit integer za bool a stále nebyly vidět žádné známky přenosu dat.

Šel jsem na to tedy od nejjednodušších věcí, zkusil jsem ověřit, jestli vodiči vůbec jde nějaké napětí. Připojil jsem na ně sondy od osciloskopu a zjistil jsem, že vodiči nic neprochází. Problém tedy musel být v kódu.

Všiml jsem si, že se v obou knihovnách vyskytuje blok „write raw data“, který zapisuje data jako list s hodnotou danou proměnnou typu bool.



Obrázek: Blok pro write raw data

Nutné tedy bylo !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

## Snímání atmosférických hodnot

Snímání teploty, vlhkosti a tlaku jsem zařídil díky modulu od firmy M5stack, který je určený přesně pro toto měření. Připojil jsem ho k jednotce ATOM Lite a zároveň jsem ho přidal v rámci UIflow prostředí jako knihovnu. To mi umožnilo s tímto modulem pracovat.

Sestrojil jsem jednoduchý kód, který data ze senzoru posílal přes MQTT server na jednotku M5 core s displejem. Data se zobrazovala na displeji včetně stavů relé v rámci relé modulu.

## Test dosavadní práce

## Vizualice dat ve webovém rozhraní

### Úprava Homeassistant UI

Závěr

Tak jsem se dostal až na konec.

Seznam zkratek a odborných výrazů

Seznam obrázků

[Obrázek 1 . 3](#_Toc86059903)

[Obrázek 2 3](#_Toc86059904)

Použité zdroje

1. UART (USART) – komunikujte sériově po dvou vodičích – ZAVAVOV.  [online]. Copyright © 2023 [cit. 02.02.2023]. Dostupné z: <http://www.zavavov.cz/cz/elektrotechnika/komunikacni-sbernice/67-uart-usart-komunikujte-seriove-po-dvou-vodicich/>
2. Komunikační protokol universální sériové sběrnice - Root.cz.  [online]. Copyright © 1997 [cit. 10.01.2023]. Dostupné z: <https://www.root.cz/clanky/komunikacni-protokol-universalni-seriove-sbernice/>
3. Sériová komunikace – Wikipedie. [online]. Dostupné z: <https://cs.wikipedia.org/wiki/S%C3%A9riov%C3%A1_komunikace>
4. Arytmický sériový přenos – Wikipedie. [online]. Dostupné z: <https://cs.wikipedia.org/wiki/Arytmick%C3%BD_s%C3%A9riov%C3%BD_p%C5%99enos>
5. Sběrnice RS-422, RS-423 a RS-485 - Root.cz.  [online]. Copyright © 1997 [cit. 02.02.2023]. Dostupné z: <https://www.root.cz/clanky/sbernice-rs-422-rs-423-a-rs-485/>
6. Protokol MQTT: komunikační standard pro IoT - Root.cz.  [online]. Copyright © 1997 [cit. 02.02.2023]. Dostupné z: <https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>
7. Co je to WebSocket? | Michal Strelec.  [online]. Copyright © 2023 Michal Strelec [cit. 02.02.2023]. Dostupné z: <https://www.strelec.pro/slovnik-vyvojare/co-je-to/websocket>
8. Co je SSL.  [online]. Copyright © SSLS.CZ [cit. 02.02.2023]. Dostupné z: <https://www.ssls.cz/slovnik/ssl.html>
9. Co je TLS.  [online]. Copyright © SSLS.CZ [cit. 02.02.2023]. Dostupné z: <https://www.ssls.cz/slovnik/tls.html>
10. Stopařův průvodce REST API.  [online]. Copyright © 2023 itnetwork.cz. Veškerý obsah webu [cit. 02.02.2023]. Dostupné z: <https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api>
11. JavaScript Object Notation – Wikipedie. [online]. Dostupné z: <https://cs.wikipedia.org/wiki/JavaScript_Object_Notation>
12. Seznam přiložených souborů