STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č.10: elektrotechnika, elektronika a komunikace

Sběr atmosférických dat

Martin Těhník
Liberecký kraj

Liberec 15.3.2023

Prohlášení

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

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

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

V Liberci dne 15.3.2023 ……

Martin Těhník

Anotace

Má práce je zaměřená na domácí automatizaci. Pomocí modulů od firmy 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, z kterého budou data vizualizována pomocí MQTT ve webovém rozhraní. Celý proces bude plně automatický, bez zásahu uživatele. Výhodou bude velká flexibilita a malý nárok na prostor. Výsledek mé maturitní práce bude použit týmem lidí, kteří mají na starosti testování výrobků, na monitoring přibližných 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 15.03.2023 .....

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. Petrovi Zenklovi za rady ohledně dokumentace k maturitní práci a v neposlední řadě bych chtěl poděkovat rodině za podporu.

Obsah

Úvod

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

Zvolil jsem si dlouhodobou maturitní práci, protože v ní 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 15.3. 2023 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.

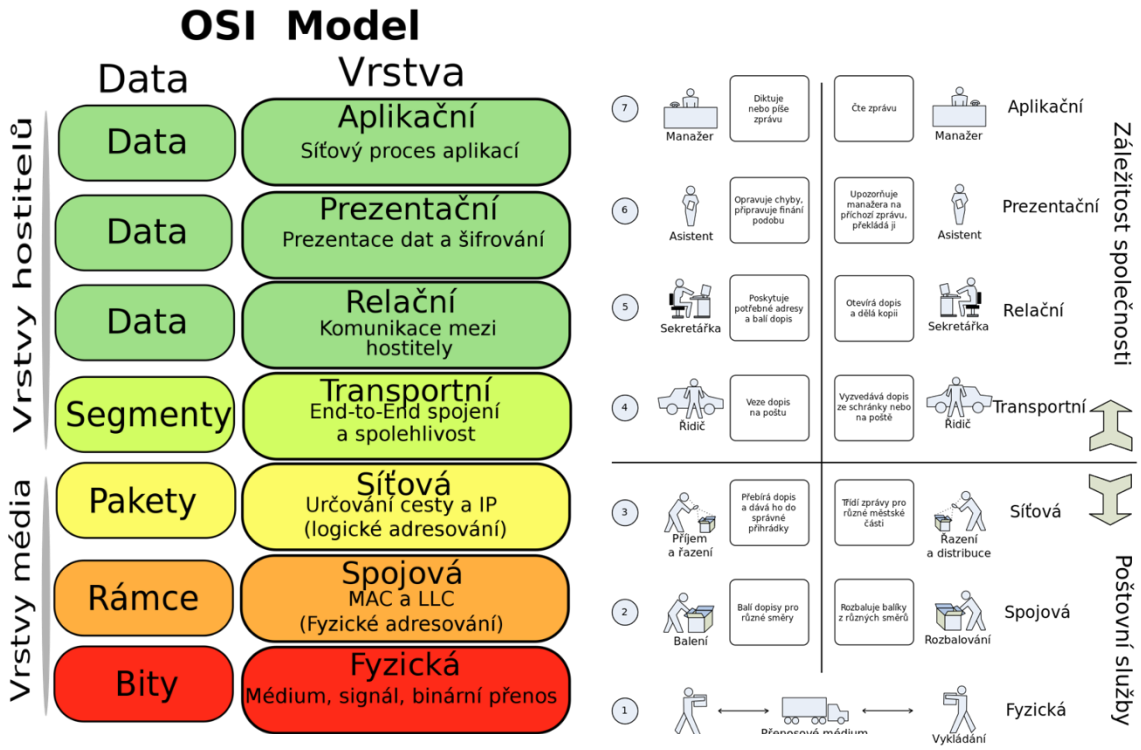
1 Teoretická část

Jedná se o popis teoretické části. Popisuji a vysvětluji zde technologie, které jsem při práci použil nebo jsem je mohl použít a zvážit v podobě dalších alternativ.

1.1 Hardware

V této kapitole se věnuji použitému Hardwaru neboli fyzickým komponentům, které jsem použil v rámci mé maturitní práce. Popisuji komunikační protokoly, proto zde uvedu vrstvy ISO/OSI 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**



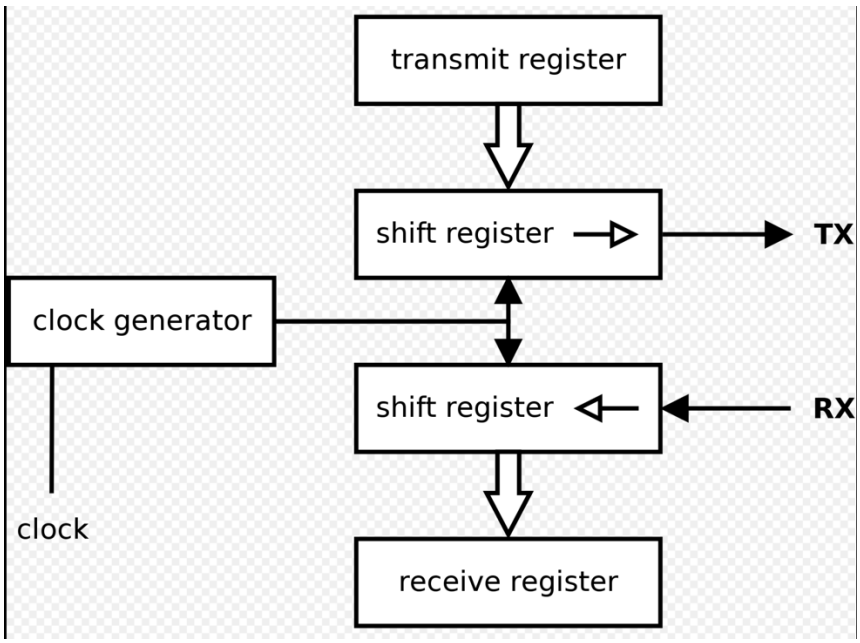
Obrázek 1: Schéma ISO modelu (24)

1.1.1 UART

Zkratka UART znamená v češtině 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 a paritní bit. Sériové komunikace jsou daleko výhodnější než paralelní, díky své rychlosti přenosu a odolnosti vůči okolnímu rušení. (1) (2)



Obrázek 2: Schéma UART (2)

	RX (číslo pinu)	TX (číslo pinu)	Baudrate	data bits	stop bits	parity	ctrl pin
Mé nastavení	23	33	9600	8	1	none	none

Tabulka 1: Nastavení UART nastavení

1.1.2 Sériové komunikační protokoly

Pro přenos dat slouží jednotlivé bity a jsou posílány 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. (3)

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

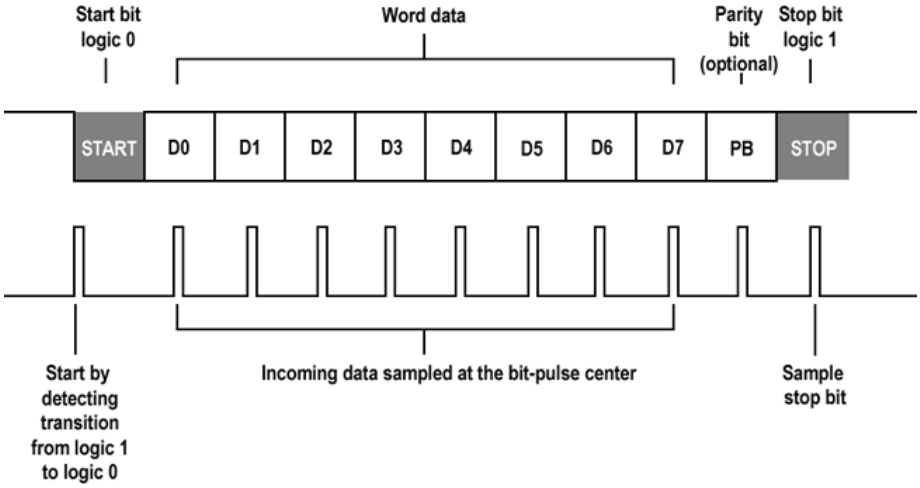
Tabulka 2: Srovnání sériové a paralelní komunikace

1.1.3 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. (4)

1.1.4 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. (5)



Obrázek 3: RS485 posílání dat (25)

1.1.5 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.

Označení	Význam
Discrete Input	1bitový registr, pouze čtení
Coil	1bitový registr, čtení/zápis
Input Register	16bit registr, pouze čtení
Holding Register	16bit. registr, čtení i zápis

Tabulka 3: Datové registry Modbus protokolu

1.1.6 M5stack

M5stack je firma zabývající se výrobou programovatelných mikrocontrollerů 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. Moduly M5stack jsou programovatelné několika možnými 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. (6) (7) (8)

1.1.7 Reléový modul SDM-6RO

Představuje soubor více relé pohromadě od firmy Aspar z Polska, jež 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, sloužící ke konfiguraci. Je uzpůsoben pro uchycení na DIN lištu nebo může být pověšen na zdi. Každé relé má tři výstupy, NC, NO a common. Výstupy jsou z důvodu vyšší

bezpečnosti galvanicky oddělené. Relé modul je doplněn o indikační LED, která indikuje komunikaci po RS485. (9)

power supply	Voltage	10-30 VDC; 10-28VAC
	Maximum current	DC: 200 mA , 24VDC AC: 250 mA , 24VAC
	Maximum power consumption	DC: 4.8W; AC: 6VA
relay outputs	No oF outputs	6
		5A 250V AC
temperature	The maximum current and voltage	10A 24V DC
	Work	-10 °C - +50°C
	Storage	-40 °C - +85°C
Connectors	Power supply	2 pins
	Communication	3 pins
	Outputs	2x 10 pins
	Configuration	Mini USB

Tabulka 4: Dokumentace relé modulu

1.1.8 ENV II – senzor atmosférických hodnot

ENV II je senzor, který umí měřit teplotu, tlak a vlhkost. Je složen ze dvou energeticky úsporných senzorů SHT30 a BMP280, které měří jednotlivé hodnoty a využívají sběrnici I2C.

SHT30 funguje jako digitální senzor teploty a vlhkosti v jednom. Jeho teplotní měřící rozsah je od -40 až 120 stupňů s tolerancí 0,2 stupně. Vlhkostní měřící rozsah činí 10 až 90 procent s tolerancí +- 2 procenta.

BMP280 je barometrický tlakový senzor, který byl navržen pro využití v mobilních/přenosných zařízeních. Rozsah tlakových měření s použitím tohoto senzoru je 300–1100 hPa s tolerancí +- 1 hPa. (10)

	Typ měření	Rozsah	Tolerance
SHT	Teplota	(-40) až 120 stupňů	+- 0,2
	Vlhkost	10–90 procent	+- 2 procenta
BMP280		300–1100 hPa	+- 1 hPa

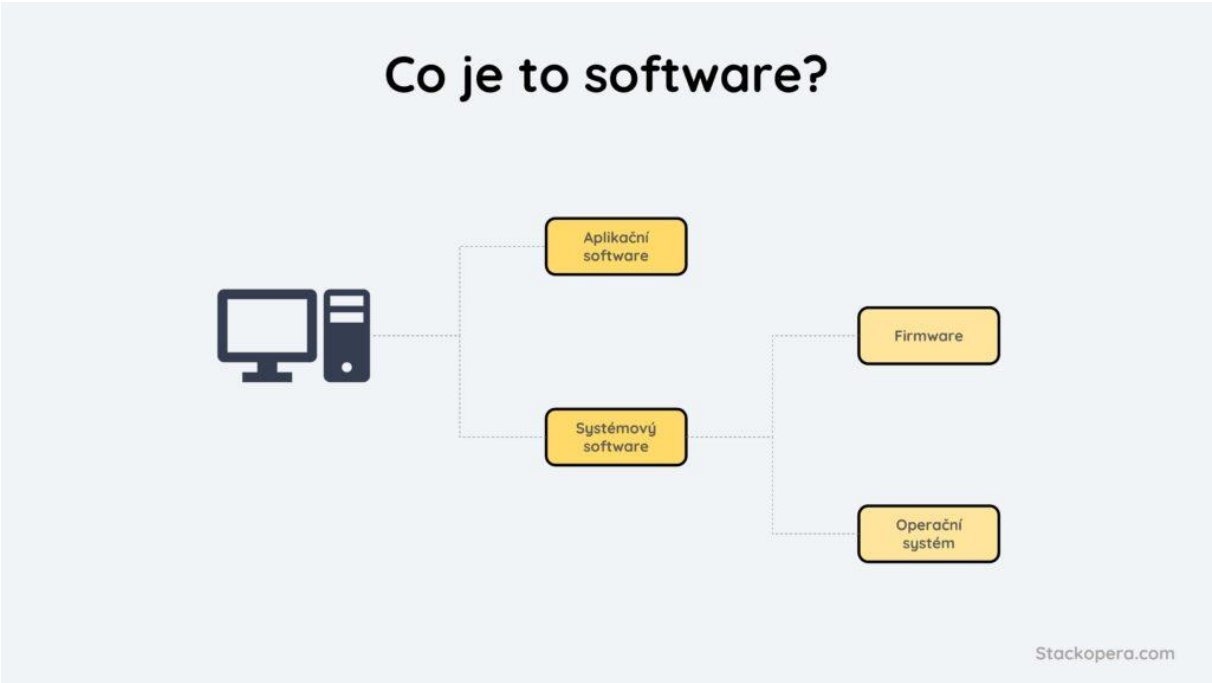
Tabulka 5: Vlastnosti senzorů

1.2 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.

Uvádím zde také REST API a JSON, jenž jsem nevyužil, ale byli v původním návrhu a později se od nich upustilo z důvodu snazšího a přehlednějšího řešení v podobě Homeassistantu.



Obrázek 4: Co je to software? (11)

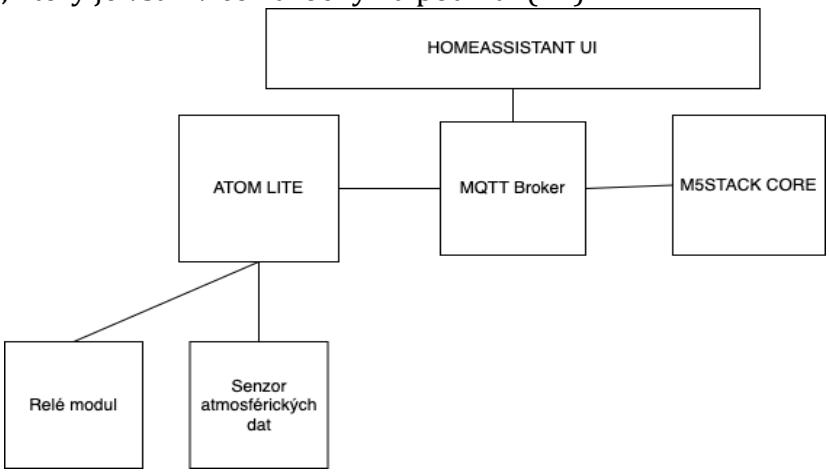
1.2.1 MQTT protokol

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.

Přenos dat se dále lze 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. (12)



Obrázek 5: MQTT schéma MP

1.2.2 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 být spuštěn na vlastním lokálním 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 (12)

Obrázek 6: Publish dat skrze MQTT

1.2.3 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.

1.2.4 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. (13)

1.2.5 SSL protokol

SSL 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. (14)

1.2.6 TLS protokol

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. (15)

SSL	TLS
Navrženo firmou Netscape	Navrženo firmou IETF
První verze v roce 1995	První verze v roce 1999
Už se téměř nepoužívá	Používají se verze TLS 1.2 a 1.3
Využívá port pro explicitní připojení	Využívá protokol pro implicitní připojení
Používá MAC pro ověření zpráv	TLS používá HMAC pro ověření zpráv

Tabulka 6: SSL vs. TLS

1.2.7 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 část, 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.

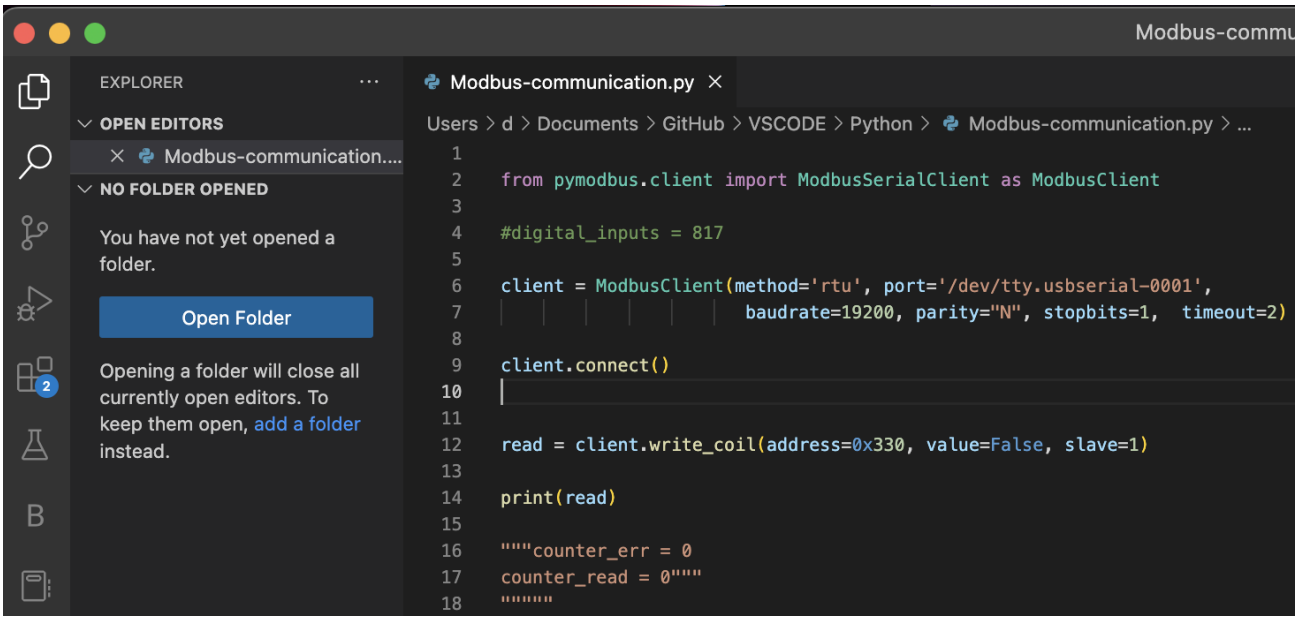
Jednoduchost syntaxe z něj dělá ideální programovací jazyk pro začátečníky. I přes jeho jednoduchost je možné v něm dělat pokročilejší algoritmy a programy.

Lidé si ho volí především díky nízkým nárokům na Hardware, 256KB ROM úložiště a 16KB RAM. (16)

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



Obrázek 7: Ukázka VSC

1.2.8 Ulflow

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. (6)

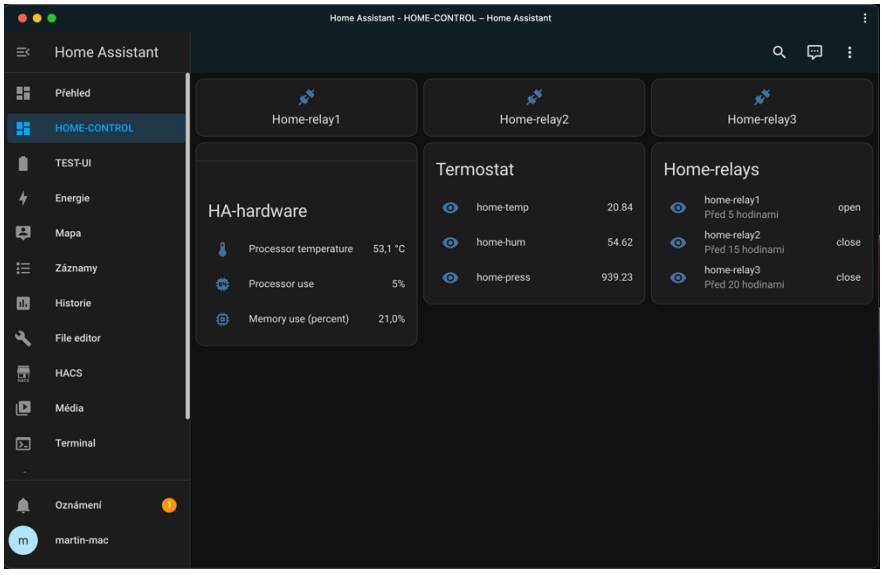
1.2.9 Homeassistant

Homeassistant je služba pro vizualizaci dat a různých grafů v rámci chytré domácnosti.

Je upravitelná, uživatel má velmi velké pravomoci v rámci úpravy svých grafů a tlačítek. Je zde možnost využít přednastavené grafy, tabulky a tlačítka. Na druhou stranu je zde i možnost si takové grafy, tabulky a tlačítka "naprogramovat" v jazyku YAML. Jeho velkou výhodou bylo sjednocení produktů různých značek s využitím různých protokolů.

To už nebude ten hlavní smysl využití této služby, jelikož na trh začínají přicházet "smarthome" zařízení, která podporují protokoly Thread a Matter.

Tyto protokoly zařídí propojení všech nově vyrobených zařízení mezi sebou. Jedná se totiž o nový standard. (17)



Obrázek 8: Prostředí Homeassistant

1.2.10 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. (18)

AT command	Meaning
+CMGS	Send message
+CMSS	Send message from storage
+CMGW	Write message to memory
+CMGD	Delete message
+CMGC	Send command
+CMMS	More messages to send

Obrázek 9: Tabulka AT commandů

1.2.11 Rest API

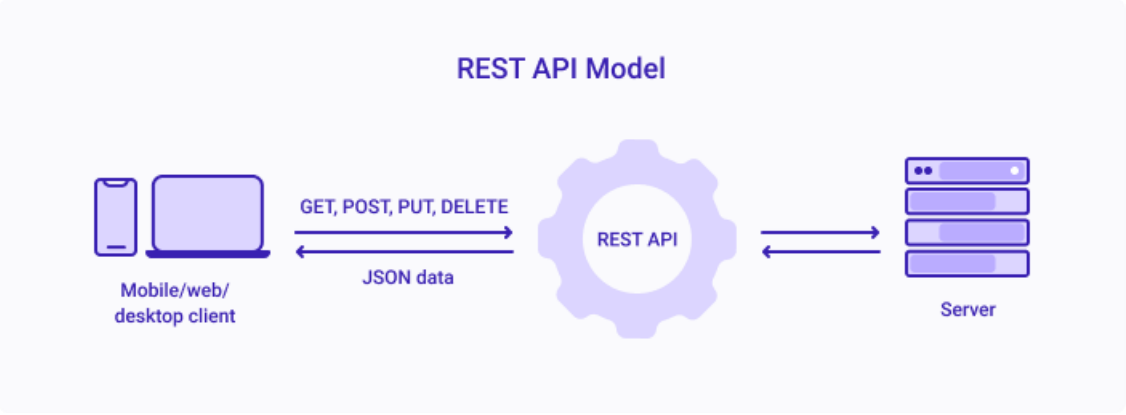
Slouží jako veřejné rozhraní pro vzdálenou komunikaci pomocí jakékoliv aplikace či zařízení. To v praxi znamená, že jakákoliv aplikace z jakéhokoliv jazyka by měla být schopná s REST API pracovat a připojit se na něj. Dělí se na čtyři úrovně.

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" 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. (19)



Obrázek 10: Rest API schéma

1.2.12 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 ukazuje data o zeměpisné pozici Prahy. (20)

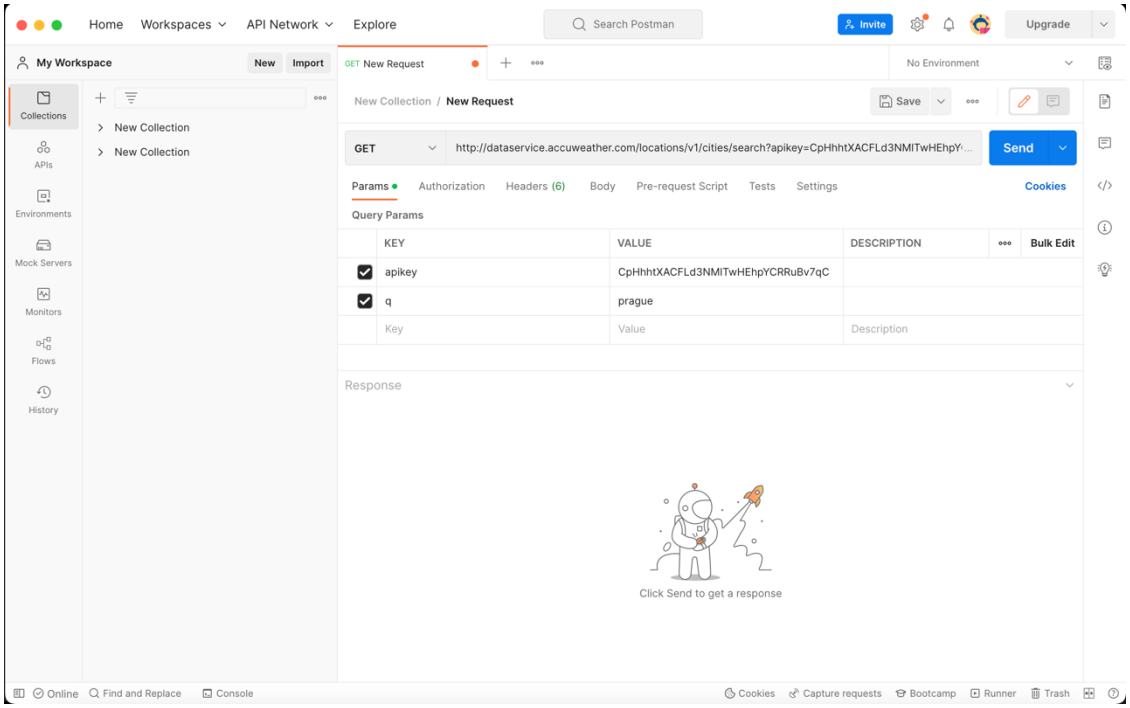

```
"Version": 1,
        "Key": "125594",
        "Type": "City",
        "Rank": 20,
        "LocalizedName": "Prague",
        "EnglishName": "Prague",
        "PrimaryPostalCode": "",
        "Region": {
            "ID": "EUR",
            "LocalizedName": "Europe",
            "EnglishName": "Europe"
        },
        "Country": {
            "ID": "CZ",
            "LocalizedName": "Czechia",
            "EnglishName": "Czechia"
        },
        "AdministrativeArea": {
            "ID": "10",
            "LocalizedName": "Prague",
            "EnglishName": "Prague",
            "Level": 1,
            "LocalizedType": "Capital City",
            "EnglishType": "Capital City",
            "CountryID": "CZ"
        },
```

1.2.13 Postman

Postman je program pro testovaní rest API viz. ukázka JSON. Je podobný 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.



Obrázek 11: Prostředí Postman

2 Praktická část

Praktická část se věnuje popisu jednotlivých kroků, které jsem musel podniknout pro dokončení maturitní práce. Dochází k průniku teoretických vědomostí a aplikací daný vědomostí.

2.1 Pochopení potřebného HW a SW

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

2.1.1 Osvojení M5stack modulů

Při seznamování s M5 moduly byly použity ukázkové programy od výrobce. Poté jsem přešel na jednoduché instrukce na M5stack core, protože disponuje displejem a tlačítky, kde se dají zobrazit různá data.

V rámci M5stack core jsem začal jednoduchou podmínkou IF, která při stisku tlačítka zobrazila předem definovaný text na displeji. Osvojení dále pokračovalo různými vizualizacemi na displeji, jelikož UIflow obsahuje knihovny pro jednoduché zobrazení dat.

Na modulu ATOM LITE bylo vyzkoušeno jeho ovládání pomocí AT commandů po sběrnici UART. Toto řešení bylo nevyhovující pro jeho omezenou využitelnost vlivem přesné formulace AT commandů, které nejsou nijak upravitelné, například při komunikaci s MQTT serverem jsem nebylo možné dále pracovat s přečtenou zprávou dál v programu.

Obrázek 12: M5stack core

Obrázek 13: ATOM LITE

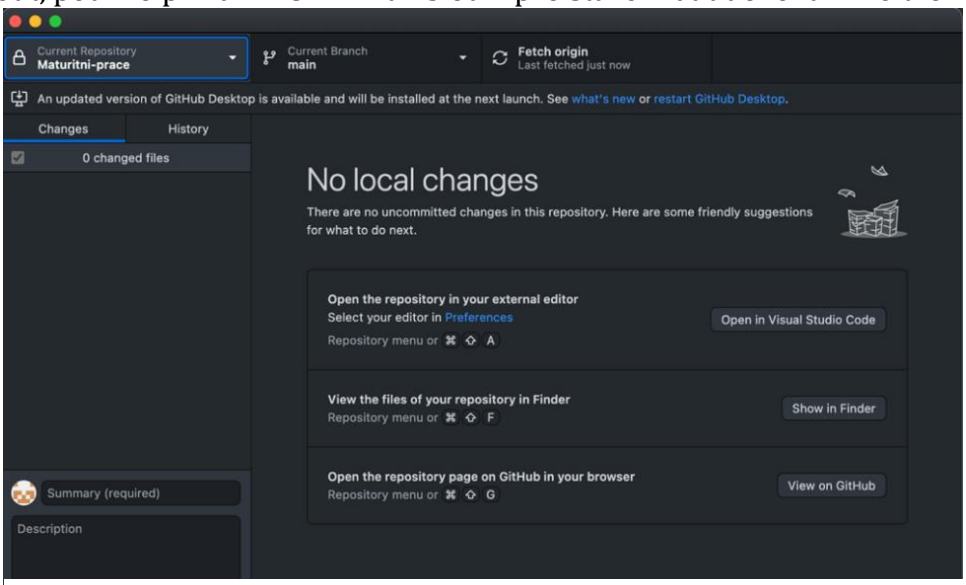
2.1.2 Práce s relé modulem

Nejprve bylo potřeba upravit kabeláž, abych mohl relé modul ovládat pomocí počítače nebo M5 modulů. Pro zapojení bylo nejdříve nutné se podívat na dokumentaci. Bylo nutné připojit 3 vodiče do svorek pro RS485 podle dokumentace výrobců a ty jsem následně zapojil do převodníku z USB na RS485. Po prostudování dokumentace a následně vyzkoušel jeho funkci skrze program tvořený pro tento modul.

2.1.3 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 14: Rozhraní aplikace GitHub

2.1.4 Praktická zkouška REST API a JSON

Poznávání Rest API bylo provedeno 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 atmosférických hodnot o daném místě pomocí rest API ze serveru Accuweather.

2.2 Tvorba SW pro M5stack moduly

Tvorba SW probíhala v oficiálním webovém rozhraní UI flow od výrobce modulů M5stack, jednotlivé funkce modulů byly zároveň naprogramovány 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.

2.2.1 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.ioteasyconnec
t.cz","1883",12000,100'+"\r\n")
uart1.write('AT+CMQCON=0,3,"myclient",600,0,0[IEZ000246:IEZ0
00246,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()
```

2.2.2 Volba MQTT serveru

Původně bylo v plánu využít služby firmy 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 lokálně na svém PC. Aby řešení bylo aplikovatelné v praxi a dostupné odkudkoliv, bylo potřeba brokera změnit na funkčního a vybral jsem HIVEMQ.

2.2.3 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 používání 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)
```

2.2.4 Vytváření základního UI

Dále jsem se věnoval tvorbě 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.

Obrázek 15: UI M5stack Core

2.3 Ovládání reléového modulu SDM-6R0

Touto částí jsem se zabýval asi nejdéle ze všech v rámci maturitní práce. Toto ovládání mi dělalo největší problém v rámci M5stacku, jelikož se v něm vyskytují nefunkční knihovny, a špatné dokumentace k nim.

V případě relé modulu bylo potřeba si nastudovat dokumentaci, abych mohl ho mohl resetovat, jelikož už byl 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. Adresy 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)
```

2.3.1 Ovládání relé pomocí M5stack-core

Tato část zabrala nejvíce času, díky nejasnosti dokumentace UIflow od výrobce, a navíc díky nefunkčnosti některých bloků jsem byl téměř odepsaný.

Počínaje 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.

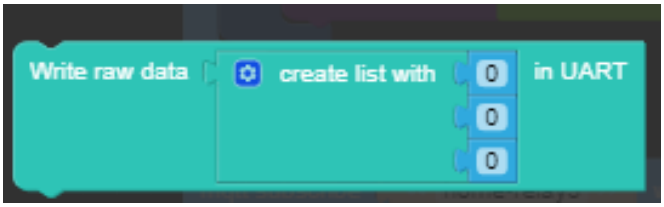
Díky předchozím pokusům v rámci VSC a díky dokumentaci modulu SDM-6RO jsem zjistil, že adresy modulu jsou typu coil.

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

Obrázek 16: Ukázka zápisu více coilů (22)

Pokusil jsem se využít i dokumentaci k druhé knihovně v rámci modulu DTU NB-IoT, kde dokumentace byla lepší, ale opět nefunkční. Adresy byly nastavené správně, akorát „output value“ měla být typu bool, tedy TRUE nebo FALSE, zápis mohl být i 1 nebo 0. Nepomohla ani záměna integeru 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.



Obrázek 17: využití zapisování hodnot v listu

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

bylo zadávat hodnotu pro otevření a zavření relé v listu, což nikde v dokumentaci zmíněno není a nikde na internetu taky ne. Je možné, že to způsobila nová verze UIflow, ale dokumentace byla postavena na generaci, která je minimálně 2 roky stará.

To samé se opakovalo v případě knihovny k DTU NB IoT modulu, jenž byla téměř identická s defaultní knihovnou pro Modbus. Uživatel je tedy odkázán na radu třetích stran nebo internetové články/videa.

Zkoušel jsem použít i svůj kód vytvořený v rámci VSC, který byl napsán v jazyku python a jeho funkčnost byla otestována na počítači. Pokusil jsem se toto řešení aplikovat i v rámci M5, ale toto řešení bylo opět nefunkční.

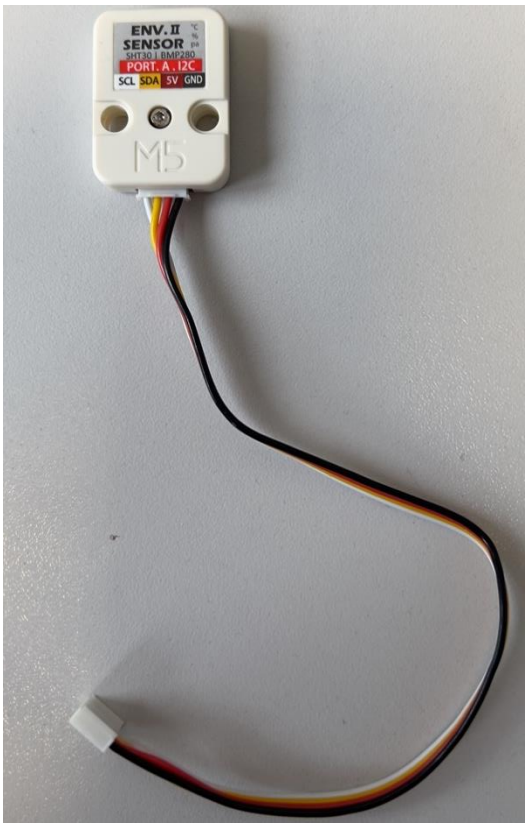
Ve výsledku jsem tento problém reportoval podpoře, aby tento demotivující proces hledání chyby nemuseli absolvovat další uživatelé, ale podpora mi do dnes neodpověděla.

Obrázek 18: Výsledné ovládání relé modulu

2.4 Snímání atmosférických hodnot

Snímání teploty, vlhkosti a tlaku bylo zprostředkováno 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. Data bylo nutné formátovat pro dodržení smysluplnosti zobrazených informací a adekvátního zobrazení, hodnoty jsou omezeny na 2 desetinná čísla. Na obrázku je čidlo, které používám v rámci své práce.



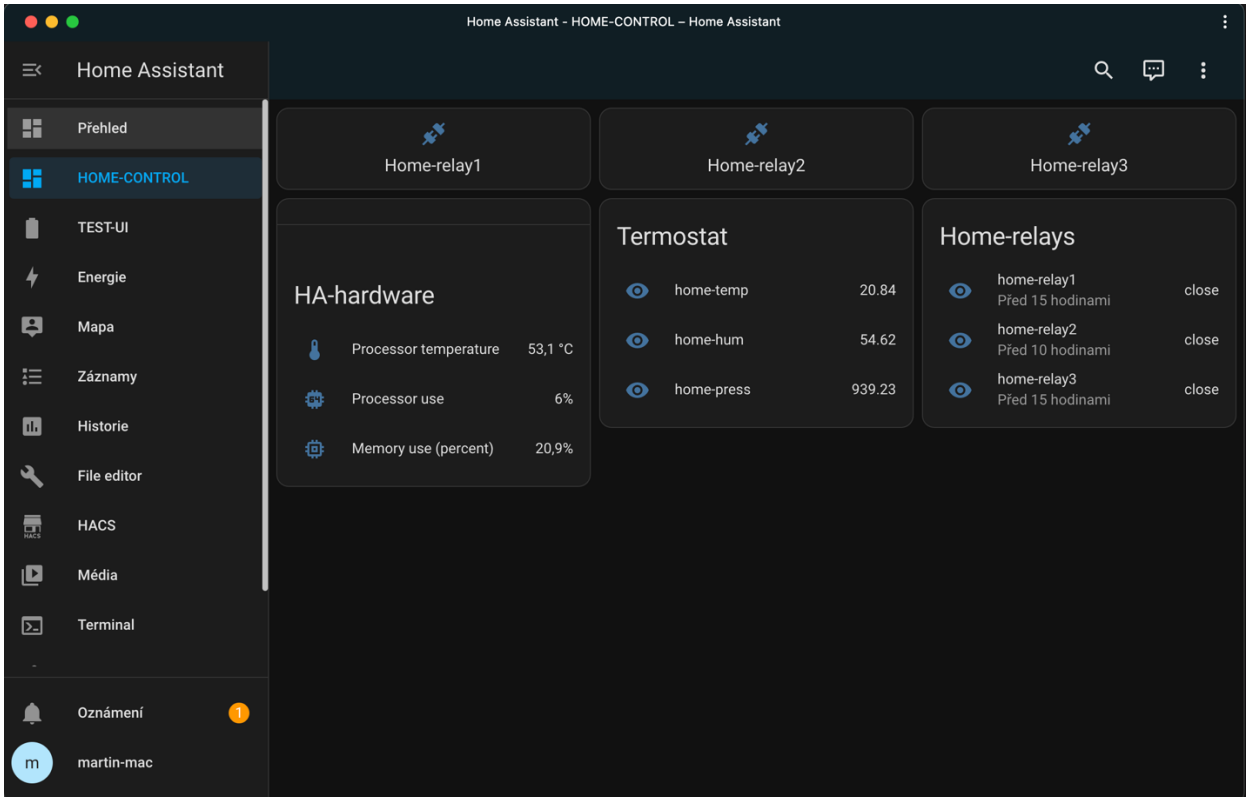
Obrázek 19: Senzor atmosferických hodnot

2.5 Vizualice dat ve webovém rozhraní

Jedna z posledních částí se zabývá vizualizací naměřených hodnot a stavů relé. Cílem bylo dosáhnout přístupu k tomuto zapojení odkudkoliv a ovládat ho.

Variant bylo mnoho, jednou z nich byla webová stránka, která by běžela na Raspberry PI. Od tohoto jsem upustil, jelikož je to složité, kvůli umístění Raspberry PI ve veřejné síti. Další variantou byl webhosting, což s sebou nese další finanční náklady a není tolik variabilní.

Vybral jsem si službu Homeassistant, která běží na Raspberry PI, k tomu se platí cloud a díky němu je toto rozhraní přístupné odkudkoliv pomocí internetu. Přidal jsem tam všechny prvky, které jsou zobrazené na M5stack core a dále jsem do aplikace přidal termostaty JAC (Jablotron Controls). Přidal jsem taky tabulku s real-time údaji o Raspberry PI (teplota CPU, využití pamětí atd.).



Obrázek 20: Vizualizace dat v HA

2.5.1 Úprava Homeassistant UI

Jelikož Homeassistant disponuje textovým editorem pro úpravu karet, tak otvírá další možnosti. Například animace se dají použít v rámci snímání větráku, kde může být animace točícího větráku, což však nebyl můj případ. Já se snažil o logické rozložení ovládacích prvků, což se mi podařilo a pokročilejší vizualizaci dat. Šlo mi o pestrost a různou vizualizaci dat, která se mi nakonec podařila pomocí obyčejných bloků v HA, které byli poskládány tak aby byly přehledné. Toto řešení je dále editovatelné pomocí pluginů, takže to je všestranné řešení.

2.6 Test funkčnosti

Tento test jsem provedl tak, že jsem celý okruh zapojil, zprovoznil a týden testoval u sebe doma a pozoroval hodnoty, které měření vykazovalo. Byla testována především stálost, responzivita MQTT serveru a různé bugy v UI. Při testování se občas objevila delší prodleva mezi stisknutím tlačítka pro změnu stavu relé a reálnou změnou stavu. To mohlo být způsobeno dvěma způsoby.

První se skrývá v pomalém internetovém připojení, což je v tomto případě nepravděpodobné.

Druhá možnost byla zpomalení procesu MQTT brokerem HiveMQ, jelikož využívám veřejné řešení, tak se tímto serverem posílají data o objemu v řádech Terabytů. Celkový počet MQTT přenosů čítá okolo 81 tisíc realizovaných přenosů.

To může pravděpodobně za zpoždění v přenosu, které je sice zanedbatelné ve většině případů, ale občas je tato prodleva velmi cítit (1-3 sekundy).

Závěr

Cílem mé maturitní práce bylo měření atmosférických dat a ovládání relé pomocí webového rozhraní. Cílů jsem dosáhl, avšak práce se neobešla bez změn ve využitých technologiích.

Původně jsem měl využít MQTT brokera od firmy Vodafone, který však disponoval nepřehlednou a krátkou dokumentací. Vodafone broker byl dočasně po dobu práce na tvorbě kódu nahrazen brokerem Mosquitto a konečné řešení bylo HiveMQ.

Dále jsem plánoval použít webové rozhraní, které by s M5 moduly komunikovalo pomocí REST API, od kterého jsem upustil, kvůli přechodu firmy Jablotron Controls na řešení HomeAssistant a tak bylo vhodné využít tuto službu také, aby byla všechna měření pohromadě.

Výhodou tohoto řešeni je velmi velká variabilita v rozšíření, což z něj dělá velmi komplexní cloudovou službu. Díky využití HomeAssistantu je možné přidat jak komerční řešení pro chytrou domácnost (Homekit/Google home) tak i průmyslové řešení (využití například PLC). Díky relé modulu je možnost spínat i obyčejná zařízení jako např. větráky, LED diody a další. Použití MQTT protokolu dává další možnosti pro jednoduché skripty a aplikace na dalších platformách, neboť MQTT se má širokou podporu.

Díky volbě dlouhodobé maturitní práci jsem se spoustu věcí naučil a posunul se dál. Tato volba mi dala možnost pracovat na tom co mě baví a rozvíjet znalosti v mém oboru. Velkou výhodou byla možnost upravit si zadání dle svých představ a pracovat na něm dlouhodobě, což mě naučilo organizovat svůj čas.

Seznam zkratek a odborných výrazů

- Arduino IDE – Arduino Integrated Development Environment
- CPU – Central Processing Unit
- Ctrl pin – control pin
- DIN – Deutsches Institut für Normung
- DTU NB IoT – Data Transmission Unit Narrow Band
- ESP32 – levné mikrokontrolery
- Full-duplex – Obousměrný provoz
- HATEOAS – Hypertext As The Engine OF Application State
- HMAC – Hash-Based Message Authentication Code
- HTTP – HyperText Transfer Protocol
- HW a SW – Hardware a Software
- I/O – Input/Output
- IETF – Internet Engineering Taskforce (název firmy, která přišla s TLS)
- IF – podmínková funkce
- ISO/OSI – ISO = jméno organizace, OSI = Open Systém Interconnection (pojemnování standardizace)
- JSON – Java Script Object Notation
- LED – Light Emiting Diode
- MAC – Message Authentication Code
- MQTT – Message Queing Telemetry Transport
- NC/NO – Normally Closed/Normally Open
- PC – Personal Computer
- PLC – Programmable Logic Controller

- QoS – Quality of Service
- RAM – Random Access Memory
- REST API – REpresentational State Transfer Application Programming Interface
- ROM – Read Only Memory
- RS (např. RS-485) – Recommended Standard
- RX – Receive
- SATA – Serial Advanced Technology Attachment
- SSL – Secure Sockets Layer
- TCP/IP – Transmission Control Protocol/Internet Protocol
- TLS – Transport Layer Security
- TX – Transmit
- UART – Universal Asynchronous Receiver Transmitter
- URL – Unifrom Resource Locator
- USB – Universal Serial Bus
- UTF-8 – Unicode Transformation Format
- VSC – Visual Studio Code
- WiFi – Wireless Fidelity
- XML – eXtensible Markup Language

Seznam obrázků

Seznam tabulek

3 Bibliografie

1. **Wikipedie, Přispěvatelé.** Sériová komunikace. *cs.wikipedia.org.* [Online] 2023. https://cs.wikipedia.org/wiki/S%C3%A9riov%C3%A1_komunikace.

2. —. Universal asynchronous receiver-transmitter. *en.wikipedia.org.* [Online] 2023. https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter.

3. **Tišnovský, Pavel.** Komunikační protokol universální sériové sběrnice. *root.cz.* [Online] 2009. https://www.root.cz/clanky/komunikacni-protokol-universalni-seriove-sbernice/.

4. **Wikipedie, Přispěvatelé.** Arytmický sériový přenos. *cs.wikipedia.org.* [Online] 2021. https://cs.wikipedia.org/wiki/Arytmick%C3%BD_s%C3%A9riov%C3%BD_p%C5%99enos.

5. **Tišnovský, Pavel.** Sběrnice RS-422, RS-423 a RS-485. *root.cz.* [Online] 2008. https://www.root.cz/clanky/sbernice-rs-422-rs-423-a-rs-485/.

6. **M5stack.** Basic. *docs.m5stack.com.* [Online] 2021. https://docs.m5stack.com/en/core/basic.

7. —. ATOM Lite. *docs.m5stack.com.* [Online] 2021. https://docs.m5stack.com/en/core/atom_lite.

8. —. ATOM DTU NB. *docs.m5stack.com.* [Online] 2021. https://docs.m5stack.com/en/atom/atom_dtu_nb.

9. **ASPAR.** SDM-6RO. *www.aspar.com.* [Online] 2022. https://www.aspar.com.pl/katalogi/IOMODULES/SDM_6RO_en.pdf.

10. **M5stack.** ENV II. *docs.M5stack.com.* [Online] 2022. https://docs.m5stack.com/en/unit/envII.

11. **Stackopera.** Co je to software? *Stackopera.* [Online] 2023. https://blog.stackopera.com/co-je-to-software/.

12. **Malý, Martin.** Protokol MQTT: komunikační standard pro IoT. *root.cz.* [Online] 2016. https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/.

13. **Strelec, MIchal.** Co je to WebSocket?strelec.pro. *strelec.pro.* [Online] 2023. https://www.strelec.pro/slovnik-vyvojare/co-je-to/websocket.

14. **SSLS.** SSL protokol. *ssls.cz.* [Online] 2006-2023. https://www.ssls.cz/slovnik/ssl.html.

15. —. TLS. *ssls.cz.* [Online] 2006-2023. https://www.ssls.cz/slovnik/tls.html.

16. **MicroPython.** MicroPython. *micropython.org.* [Online] 2023. https://micropython.org/.

17. **HomeAssistant.** HomeAssistant Documentation. *home-assistant.io.* [Online] 2022. https://www.home-assistant.io/installation/.

18. **ESPRESSIF.** AT Command Set. *docs.espressif.com.* [Online] 2016-2023. https://docs.espressif.com/projects/esp-at/en/latest/esp32/AT_Command_Set/index.html.

19. **Hanák, Drahomír.** Stopařův průvodce REST API. *itnetwork.cz.* [Online] 2023. https://www.itnetwork.cz/programovani/nezarazene/stoparuv-pruvodce-rest-api.

20. **WIkipedie, Přispěvatelé.** JavaScript Object Notation. *cs.wikipedia.org.* [Online] 2022. https://cs.wikipedia.org/wiki/JavaScript_Object_Notation.

21. **M5stack.** UIFlow - M5Core. *docs.m5stack.com.* [Online] 2021. https://docs.m5stack.com/en/quick_start/m5core/uiflow.

22. —. Modbus Master. *docs.m5stack.com.* [Online] 2021. http://docs.m5stack.com/en/uiflow/advanced/modbus.

23. **Zavavov.** UART (USART) – komunikujte sériově po dvou vodičích. *Zavavov.cz.* [Online] 2014. http://www.zavavov.cz/cz/elektrotechnika/komunikacni-sbernice/67-uart-usart-komunikujte-seriove-po-dvou-vodicich/.

24. **Brodský, Vojtěch.** *03 - Referenční modely a protokoly.* Liberec : autor neznámý, 31. 10 2022.

25. **ElectricImp.** UART Explained. *developer.electricimp.com.* [Online] 2023. https://developer.electricimp.com/resources/uart.

A. Seznam přiložených souborů

– **MP2022-2023-E4A-Těhník-Martin.docx** – editovatelná verze dokumentace maturitní práce
– **MP2022-2023-E4A-Těhník-Martin.pdf** – tisknutelná verze dokumentace
maturitní práce
– **SOČ-2023-E4A-Těhník-Martin.docx** - editovatelná verze dokumentace středoškolské odborné činnosti
– **SOČ-2023-E4A-Těhník-Martin.pdf** - tisknutelná verze dokumentace středoškolské odborné činnosti
– **Aplikace** – zdrojové kódy, funkční software, celý projekt.zip