

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 se zabývá sběrem atmosférických dat v terénu, měřených pomocí senzorů teploty, tlaku a vlhkosti. Prací 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 REST API ve webovém rozhraní. Celý proces bude plně automatický, uživatel nebude nucen nic dělat. Výhodou bude snadná realizace a skladnost. Výsledek mé maturitní práce bude použit týmem lidí, kteří mají na starosti testování výrobků, na měření referenčních hodnot, popřípadě na jednoduché ovládání osvětlení skrze relé.

Summary

My work deals with the collection of atmospheric data in the field, measured using temperature, pressure and humidity sensors. It works 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](#_Toc124313618)

[1 Teoretická část 2](#_Toc124313619)

[1.1 UART 2](#_Toc124313620)

[1.1.1 Sériové komunikační protokoly 2](#_Toc124313621)

[1.1.2 Asynchronní komunikační protokoly 3](#_Toc124313622)

[1.1.3 RS 485 3](#_Toc124313623)

[1.1.4 Modbus 4](#_Toc124313624)

[1.2 MQTT protokol 4](#_Toc124313625)

[1.2.1 MQTT broker 4](#_Toc124313626)

[1.2.2 WebSocket 5](#_Toc124313627)

[1.3 Programovací jazyk 5](#_Toc124313628)

[1.3.1 Python 5](#_Toc124313629)

[1.3.2 Micropython 6](#_Toc124313630)

[1.3.3 Visual Studio Code 6](#_Toc124313631)

[1.3.4 AT commandy 6](#_Toc124313632)

[1.4 6](#_Toc124313633)

[1.5 6](#_Toc124313634)

[2 Seznámení s Hardwarem 7](#_Toc124313635)

[2.1.1 Osvojení M5stack modulů 7](#_Toc124313636)

[2.1.2 Reléový modul SDM-6RO 8](#_Toc124313637)

[2.2 Tvorba SW pro M5stack moduly 8](#_Toc124313638)

[2.2.1 Ovládání Atom Lite pomocí AT commandů 8](#_Toc124313639)

[2.2.2 Tvorba spojení s MQTT serverem 9](#_Toc124313640)

[3 Studium REST API a JSON 10](#_Toc124313641)

[3.1 REST API 10](#_Toc124313642)

[3.2 JavaScript Object Notation 11](#_Toc124313643)

[4 12](#_Toc124313644)

[Závěr 13](#_Toc124313645)

[Seznam zkratek a odborných výrazů 14](#_Toc124313646)

[Seznam obrázků 15](#_Toc124313647)

[Použité zdroje 16](#_Toc124313648)

[A. Seznam přiložených souborů I](#_Toc124313649)

Úvod

V rámci praktické maturity je možné si zvolit dlouhodobou maturitní práci nebo praktickou maturitu ve škole. Zvolil jsem si dlouhodobou maturitní práci, protože v ní vidím příležitost osobního rozvoje a možném kariérním postupu v dané firmě. Kreativní tvorba mě může hodně posunout. K tomu přispívá i vlastní organizace maturitní práce, respektive zlepšení své disciplíny a organizace času.

Na maturitní práci jsem pracoval 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 snadné a rychlé instalace a bezdrátového odesílání naměřených dat.

Pro realizaci cíle maturitní práce jsem použil ATOM NB-Iot module od firmy M5stack pro komunikaci se senzory atmosférických dat. 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

## UART

Zkratka UART znamená “Universal Asychronous Receiver-Transmitter”, v češtině se jedná o “Univerzální Asynchronní Přijímač-Vysílač”. Tento komunikační protokol spadá 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 (mohou být i 2), který slouží k indikaci začátku a konce vysílané zprávy. Ke kontrole slouží paritní bit, který slouží k odhalení chyb v přenosu dat. Pod UART spadá i RS-232(12V) a RS-485(5V). Pro přenos je důležité stanovit rychlost přenosu, paritní bit. Sériové komunikace jsou daleko výhodnější než paralelní.

### Sériové komunikační protokoly

Posílají informace tak, že nejdřív převedou bajty na bity a poté bity posílají jeden po druhém za sebou v tzv. rámcích. Sériové komunikace mají výhodu téměř “neomezené“ vzdálenosti. Jsou odolné vůči rušení a přeslechům, což můžeme ocenit v zarušených prostředích. Sériová komunikace má bohužel i své neduhy, 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. Dnes je už nahrazována paralelní komunikací, jelikož paralelní komunikace je rychlejší.

### Asynchronní komunikační protokoly

Signál, na rozdíl od synchronních protokolů, nenese hodinový signál pro synchronizaci. Synchronizaci zajišťuje start bit a stop bit (někdy 2 stop bity), které jsou vyslány společně s daty. Start a stop bit se vyskytují na začátku a konci každého rámce (jedné zprávy). Start dá přijímači signál pro příchod dat a stop z resetuje přijímač na počáteční nastavení, aby se mohlo začít nové spojení. Závisí na kvalitě hodinového signálu, který způsobuje změnu preiodickou změnu stavu. Je generován zpravidla pomocí RC nebo RL obvody nebo krystalem v daném zařízení. 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 nemohou současně. Přenos probíhá vždycky jedním směrem. Příkladem této komunikace jsou například 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, délce kabelů a maximálním počtu uzlů. Mezi další výhody patří složení sběrnice RS485 z rozšířenější RS-232. Je ovšem nutné využít převodníky napěťových úrovní, jelikož se obě technologie liší

Obrázek 3: UART přenos

### Modbus

Modbus je 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.

## MQTT protokol

MQTT neboli Message Queue Telemetry Transport je postaven na protokolu TCP/IP patří mezi méně náročnější protokoly. To umožňuje použít méně výkonný hardware. Využívá se pro přenos malého objemu dat, 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

### 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 nějakou zprávu na daný server. Subscriber je zařízení, které publikovaný obsah sbírá a dále s ním může nějak nakládat. Publikovaný obsah se dále dělí podle tzv. “topic“, který slouží k rozdělení poslaných dat do různých témat.

### 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 jakýkoliv aplikacích. Umožňuje spojení s prohlížečem a webovým serverem s menší režií, usnadňuje přenos dat mezi servery.

## Programovací jazyk

Slouží k zápisu programu/algoritmů, které mohou být provedeny na celém počítači nebo v rámci jedné výpočetní jednotky. Výsledkem programování je nějaký program, který je spustitelný. V podstatě se jedná o komunikační prostředek mezi programátorem a počítačem, který daný program vykonává. Při psaní programu je nutné se držet určitými pravidly podle používaného jazyka.

### Python

Tento programovací jazyk 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 včetně spousty dostupných instalačních balíčků.

### Micropython

Micropython je zjednodušená verze hojně používaného jazyka python. Liší se pouze v náročnosti na výpočetní výkon. Micropython je velmi zjednodušený oproti klasickému Pythonu a díky tomu se dá použít při programování microkontrolerů. Python, popřípadě Micropython je velmi rozšířený hlavně díky své jednoduchosti a spoustě knihoven, které vyřeší spoustu věcí za nás. Nemusíme vymýšlet různé řešení, jelikož jsou řešena v samotných knihovnách.

### Visual Studio Code

Jedná se o program 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 možnost importovat například knihovny, což v počátcích programování nebylo běžné. Tento program slouží pro programování v téměř všech programovacích jazycích, což patří mezi jednu z jeho dalších výhod.

### 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 nebo “GET/cities/eu” nám vrátí evropská města.

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. Stavové kódy se posílají ve formátu stovek a každý kód má nějakou vlastnost, kterou signalizuje. 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. Vstupen 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.

## 

# Seznámení s Hardwarem

### Osvojení M5stack modulů

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.

Obsah obrázku zeď, interiér

Popis byl vytvořen automaticky

Obsah obrázku interiér, kuchyňské spotřebiče

Popis byl vytvořen automaticky

Obrázek : Atom Lite+NB Iot modul

Obrázek 2: M5stack core modul

### 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.

## 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 v jazyku Python a otestována na PC. Tudíž dané funkce a ovládání je aplikovatelné i na Windows systémy.

### 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 a nedostatečného popsání funkce 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.

### 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, které jsem později nahradil jiným brokerem “”””””””””. Nakonec jsem použil integrovanou knihovnu v UIflow prostředí pro připojení k MQTT brokeru.

### Ovládání pomocí modulu

### 

# 

Závěr

Zatím mě maturitní práce stála 120 hodin a k tomu nějaký volný čas.

Seznam zkratek a odborných výrazů

Seznam obrázků

[Obrázek 1. 3](#_Toc86059903)

[Obrázek 2 3](#_Toc86059904)

Použité zdroje

1.

1. Seznam přiložených souborů

Na přiloženém datovém nosiči se nacházejí následující soubory a složky:

* **Výkresy** – kompletní výkresová dokumentace
* **Aplikace** – zdrojové kódy