VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

MĚŘICÍ STANICE S PŘENOSEM DAT PŘES 4G SÍŤ
MEASURING STATION WITH DATA TRANSMISSION VIA 4G NETWORK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Michal Pernica

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jan Najman

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Michal Pernica**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jan Najman**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Měřicí stanice s přenosem dat přes 4G síť

Stručná charakteristika problematiky úkolu:

Vzdálený bezdrátový monitoring strojů a ukládání naměřených dat z mnoha stanic na server je v dnešní době velmi aktuálním tématem. Cílem této práce je vytvoření jednoduché měřicí stanice, která bude schopná odesílat naměřená data na server. Přenos dat bude realizován přes 4G síť a bude kladen důraz zejména na vysokou rychlost a nízkou latenci přenosu.

Cíle bakalářské práce:

1. Na vybraném modulu proveďte rešerši všech možností přenosu dat přes 4G síť.
2. Navrhněte a vyrobte desku s měřicí elektronikou:
– Obvody pro měření napěťových signálů.
– Řídicí člen (mikrokontroler).
– 4G modul + vhodné antény.
3. Vytvořte webovou aplikaci/server pro přijímání, ukládání a zobrazování naměřených dat.

Seznam doporučené literatury:

ZÁHLAVA, Vít. Návrh a konstrukce desek plošných spojů: principy a pravidla praktického návrhu. Praha: BEN - technická literatura, 2010. ISBN 978-80-7300-266-4.

VALÁŠEK, Michael. Mechatronika. Praha: České vysoké učení technické, 1995. ISBN 80-01-01276-X.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá přenosem dat přes 4G síť z měřící stanice na webový server. V první části práce jsou uvedené možnosti přenosu přes 4G síť pomocí modulu SIMCom 7600E-H. Dále jsou zde popsány způsoby, jak lze měřící stanici sestavit a jak využít webový server pro ukládání a zobrazování naměřených dat odeslaných z měřící stanice.
Ve druhé části práce jsou jednotlivé druhy přenosu otestovány a na základě výsledků je dále realizován živý přenos dat a intervalové odesílání dat. Součástí je také postup nastavení serveru a webové aplikace pro zobrazování dat či sestavení a naprogramování měřicí stanice.

Summary

The bachelor thesis is concerned with data transfer via 4G network from a measuring station to a web server. The first part presents the possibilities of transmission via 4G network using the SIMCom 7600E-H module. The ways how to assemble the measuring station or how to make a web server for saving and displaying the measured data are introduced.
In the second part, the individual types of transmission are tested and live data transmission and interval data transmission is also implemented based on the results from the first part. The thesis also notes the procedure for setting up a server and a web application for displaying data or assembling and programming a measuring station.

Klíčová slova

4G síť, LTE, SIMCom 7600E-H, TCP server, Python, Flask, webová stránka, Chart.js, ESP32, měřící stanice

Keywords

4G network, LTE, SIMCom 7600E-H, TCP server, Python, Flask, website, Chart.js, ESP32, measuring station

Bibliografická Citace

PERNICA, M. *Měřící stanice s přenosem dat přes 4G síť*. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132343. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce: Ing. Jan Najman.

Prohlašuji, že jsem bakalářskou práci na téma Měřící stanice s přenosem dat přes 4G síť vypracoval samostatně pod vedením Ing. Jana Najmana a uvedl v ní všechny použité literární a jiné odborné zdroje.

Michal Pernica

Brno . . . .

Rád bych tímto poděkoval Ing. Janu Najmanovi za věnovaný čas, vstřícný přístup a cenné připomínky, které mi v průběhu psaní bakalářské práce poskytl. Dále bych chtěl poděkovat mé rodině a kamarádům, kteří mě podporovali po celou dobu studia.

Michal Pernica

Obsah

1 Úvod

V dnešní době je velmi populární vzdálený bezdrátový monitoring, ať už v průmyslu pro monitorování strojů či v domácnostech např. pro sledování teploty v místnosti nebo množství dešťové vody v kádi. Ke sběru dat většinou slouží velké množství měřících stanic, které odesílají data na server. Naměřená data se ukládají do databázových úložišť a uživatel si je následně může zobrazovat na webové stránce nebo v mobilní aplikaci. Rozšířením těchto aplikací může být i případné odesílání příkazů zpět na měřící stanici, díky kterým se může měřený stroj či jiné zařízení např. vypnout nebo jiným způsobem ovládat a nastavovat.

Tato práce se zabývá přenosem naměřených dat z měřící stanice pomocí 4G sítě, která je dnes už velmi rozšířená a pokrývá skoro celé území ČR. V dnešní době se poměrně často mluví dokonce o síti 5G. Dostupnost této sítě je zatím omezená, především na větší města.
Přenos pomocí 4G síťě bude realizován na modulu SIMCom 7600E-H. Na tomto modulu nejprve provedeme řešerši všech možností přenosu dat pomocí 4G, jejich výhody a nevýhody. V praktické části práce budou jednotlivé možnosti otestovány a zhodnoceny pro další využití. Při hodnocení bude kladen důraz na nízkou latenci přenosu a vysokou přenosovou rychlost. Pomocí zvoleného přenosu bude následně realizována měřící stanice a server s webovou stránkou s grafickým zobrazením příchozích dat i historií měření.

2 Rešerše

2.1 Modul SIMCom A7600E-H

2.1.1 Specifikace modulu

A7600E-H od firmy SIMCom je modul LTE kategorie 4. Podporuje bezdrátovou komunikaci pomocí *LTE-TDD/LTE-FDD/HSPA+/GSM/GPRS/EDGE*, které zabezpečují rychlost stahování až 150 Mbit/s a rychlost nahrávání až 50 Mbit/s. Modul se ovládá pomocí AT příkazů a podporuje telefonování, přijímání/odesílání SMS a má již několik vestavěných síťových protokolů. Tyto protokoly budou podrobněji popsány v následující podkapitole. Modul dále nabízí určování geografické polohy pomocí GPS/Beidou/GLONASS nebo LBS. *A7600E-H* také disponuje základními rozhraními jako jsou USB, GPIO, I2C či UART s baudratem od 300 bits/s do 4 Mbits/s. V neposlední řadě může také bezdrátově aktualizovat firmware pomocí FOTA. [1]

Obrázek 2.1: Modul SIMCom 7600E-H [33]

2.1.2 AT příkazy

AT příkazy jsou dané pokyny zapsané v datasheetu modulu, pomocí kterých je modul ovládán přes sériový přenos dat. U sériového přenosu musí mít obě zařízení nastavena stejný baudrate a ukončovací znak. V tomto případě je to <CR>. Jak už název kapitoly napovídá, předpona každého příkazu je AT, po ní následují další požadavky. Než zadáme další příkaz, je nutné počkat na odpověď modulu o provedení či neúspěchu předchozího příkazu, jelikož do té doby modul nereaguje na žádné další příkazy.[2]

```
%priklady prikazu
AT+CRESET       %restart modulu
AT+CSQ?         %zjisteni kvality signalu
AT+IPR=921600   %nastaveni baudratu komunikace
```

2.1.3 Druhy přenosů

TCP

TCP (Transmission Control Protocol) je protokol transportní vrstvy a je spojově orientovaný. Před samotným odesláním dat musí být navázáno spojení mezi klientem a serverem. Spojení se potvrzuje tzv. trojcestným podáním ruky (Three-way handshake), podle kterého se následně číslují odesílané pakety, proto je protokol založen na spolehlivém přenosu dat. Přenášená data jsou před odesláním rozdělena do paketů a očíslována podle prvotního potvrzení spojení. Cílové zařízení potvrzuje příjem jednotlivých paketů. Díky číslování je schopno jednotlivé pakety seřadit. Jestliže zařízení, které odesílá pakety, do určité doby nepřijme potvrzení o příjmu, odešle daný paket znovu. Tímto je zabezpečen spolehlivý přenos pomocí TCP protokolu. [3]

Všechny příkazy pro TCP přenos jsou uvedeny na obr. 2.2. Pro odesílání dat z modulu budou dále využity jen některé, a to v následujícím pořadí uvedeném níže [4]. Mezi jednotlivými příkazy přichází odpovědi z modulu o potvrzení provedení či chybě. [2]

```
%TCP prenos
AT+NETOPEN              %zapne sluzbu TCP/IP
AT+CIPOPEN=<index(1-9)>,"TCP","<IP adresa serveru>",<port serveru(0-65535)>
%vytvori spojeni mezi servererem a modulem, index pro vyber pri odesilani
AT+CIPSEND=<index>,<velikost zpravy>  %odeslani zpravy na spojeni s indexem
                                      %max velikost zpravy 1500
> %po prichodu tohoto znaku nasleduje vlozeni zpravy o predem dane velikosti
AT+CIPCLOSE=<index>     %uzavreni spojeni se serverem na indexu
AT+NETCLOSE             %vypnuti sluzby TCP/IP
```

- **Výhody**
  - Spolehlivost protokolu;
  - pro odesílání dat je nutné navázat spojení pouze jednou;

  - možnost připojit modul až na 9 TCP serverů.

- **Nevýhody**
  - Vytvoření TCP serveru pro příjem a zpracování dat;
  - velikost zprávy je limitována maximálním počtem 1500 znaků.

Command	Description
AT+NETOPEN	Start TCPIP service
AT+NETCLOSE	Stop TCPIP service
AT+CIPOPEN	Setup TCP/UDP client socket connection
AT+CIPCLOSE	Destroy TCP/UDP client socket connection
AT+CIPSEND	Send TCP/UDP data
AT+CIPRXGET	Retrieve TCP/UDP buffered data
AT+IPADDR	Get IP address of PDP context
AT+CIPHEAD	Add an IP header when receiving data
AT+CIPSRIP	Show remote IP address and port
AT+CIPMODE	Select TCP/IP application mode
AT+CIPSENDMOE	Set sending mode
AT+CIPTIMEOUT	Set TCP/IP timeout value
AT+CIPCCFG	Configure parameters of socket
AT+SERVERSTART	Startup TCP server
AT+SERVERSTOP	Stop TCP server
AT+CIPACK	Query TCP connection data transmitting status
AT+CDNSGIP	Query the IP address of given domain name
AT+CDNSGHNAME	Query the domain name of given IP address
AT+CIPDNSSET	Set DNS query parameters
AT+CPING	Ping destination address
AT+CPINGSTOP	Stop an ongoing ping session

Obrázek 2.2: Příkazy pro TCP/UDP [2]

UDP

UDP (User Datagram Protocol) je také protokol transportní vrsty jako TCP, ale již není spojově orientovaný. Oproti TCP se též označuje jako nespolehlivý protokol, jelikož nezaručuje spolehlivé doručení ani správné pořadí všech odeslaných paketů. Protože ale nenavazuje spojení před odesíláním dat a pracuje na principu otázka-odpověď (request-response), je považován za jednoduchý a málo náročný protokol. UDP se využívá především u aplikací, kde je současně zapotřebí jednoduchost, nízká latence a neklade se důraz na příjem všech dat jako jsou např. online hry, VoIP nebo je nasazen na DNS serverech. [7]

Příkazy pro UDP jsou stejné jako pro TCP, uvedeny jsou na obr. 2.2. [2] Pro odesílání budou využity opět jen některé s následujícím nastavením. [4]

```
%UDP prenos
```

```
AT+NETOPEN                %zapne sluzbu TCP/IP
AT+CIPOPEN=<index(1-9)>,,"UDP",,,<local_port>
%pri nastaveni UDP nemusime zadat IP adresu a port serveru
%musime specifikovat mistni port
AT+CIPSEND=<index>,<velikost zpravy>,"<IP adresa serveru>",<port(0-65535)>
%odeslani zpravy pres port na indexu, max velikost zpravy 1500
> %po prichodu tohoto znaku nasleduje vlozeni zpravy o predem dane velikosti
AT+CIPCLOSE=<index>       %uzavreni portu na indexu
AT+NETCLOSE               %vypnuti sluzby TCP/IP
```

- **Výhody**
  - Vysoká rychlost - nemusí se navazovat spojení a potrvrzovat přijetí paketů.
- **Nevýhody**
  - Možná ztráta některých dat;
  - nutnost vytvořit UDP server;
  - velikost zprávy je maximálně 1500 znaků.

FTP

FTP (File Transfer Protocol) je síťový protokol, který se zaměřuje na přenos souborů mezi dvěma počítači. Je založen na bázi TCP spojení a na modelu klient-server. Obvykle má vyhrazeny TCP porty 20 a 21. Port 21 slouží pro navázání spojení a přenos příkazů. Přenos dat poté probíhá na portu 20. Uživatelé se ve většině případů přihlašují za pomoci uživatelského jména a hesla ve formátu prostého textu, ale mohou i anonymně, pokud je tak server nakonfigurován. Pro šifrování přenosu se využívá SSL/TLS (FTPS). [5]

```
%FTP prenos
AT+CFTPSSTART             %zapne sluzbu FTP
AT+CFTPSSINGLEIP=1        %port pro data je stejny jako pro prikazy
AT+CFTPSLOGIN="< IP adresa>",<port>,"<login>","<heslo>",<typ serveru>
%prihlaseni k serveru, typ serveru => FTP=0; FTPS(SSL)=1; FTPS(TLS)=2
AT+CFTPSPUTFILE="<nazev souboru>",<specifikace uloziste v modulu>
%specifikace uloziste: 1-F:/        2 - D:/(sd karta);          3 - E:/
AT+CFTPSPUT="<nazev souboru>",<pocet znaku>
%nahraje soubor na server skrz seriovou komunikaci, pocet znaku max 2048
AT+CFTPSLOGOUT        %odhlaseni ze serveru
AT+CFTPSSTOP          %vypnuti sluzby FTP
```

Všechny příkazy pro přenos dat pomocí FTP protokolu jsou uvedeny na obr. 2.3. [2] Pro nahrání souboru na FTP server stačí pouze příkazy uvedené výše. [4]

Command	Description
AT+CFTPSSTART	Start FTP(S) service
AT+CFTPSSTOP	Stop FTP(S) Service
AT+CFTPSLOGIN	Login to a FTP(S)server
AT+CFTPSLOGOUT	Logout FTP(S) server
AT+CFTPSMKD	Create a new directory on FTP(S) server
AT+CFTPSRMD	Delete a directory on FTP(S) server
AT+CFTPSDELE	Delete a file on FTP(S) server
AT+CFTPSCWD	Delete a file on FTP(S) server
AT+CFTPSPWD	Get the current directory on FTP(S) server
AT+CFTPSTYPE	set the transfer type on FTP(S) serve
AT+CFTPSLIST	List the items in the directory on FTP(S) server
AT+CFTPSGETFILE	Get a file from FTP(S) server to module
AT+CFTPSPUTFILE	Put a file from module to FTP(S) server
AT+CFTPSGET	Get a file from FTP(S) server to serial port
AT+CFTPSPUT	Put a file to FTP(S) server through serial port
AT+CFTPSSINGLEIP	Set FTP(S) data socket address type
AT+CFTPSCACHERD	Set FTP(S) data socket address type
AT+CFTPSABORT	Abort FTP(S) operations
AT+CFTPSSIZE	Get the File Size on FTP(S) server

Obrázek 2.3: Příkazy pro FTP [2]

- **Výhody**
  Nabízí nejen odesílání souborů uložených v paměti modulu nebo na SD kartě, ale je i možné přímo tento soubor vytvořit s maximálním počtem znaků 2048.

- **Nevýhody**
  Nutné vytvořit FTP server, obvykle ale bývá FTP server dostupný k online hostingům webových stránek jako je např. webzdarma.cz. K přihlášení se používají přihlašovací údaje, které umožní přístup k celé správě webu, tudíž se jedná v tomto ohledu jedná o méně bezpečné řešení.

HTTP

HTTP (Hyper Text Transfer Protocol) je síťový aplikační protokol, který slouží především k distribuci hypertextových dokumentů ve formátu HTML, XML. Protokol byl navržen pro komunikaci mezi webovými prohlížeči a webovými servery, ale může být použit také k jiným účelům. HTTP je protokol s modelem typu žádost-odpověď (request-response) v architektuře klient-server. Klient pomocí některé z metod posílá požadavek pro získání nebo změnu zdroje na server. Po dobu vyřizování požadavku je udržováno spojení, dokud server neodpoví zprávou se stavem zpracování požadavku, kterou obvykle následují vyžádaná data. Komunikace probíhá nad protokolem TCP a má rezervovaný port 80, ale může být zajištěna i jinými porty jako např. 8080 nebo 8008. [8]

- **HTTP metody modulu:**
  - GET - základní metoda, slouží k získání dokumentu, nejčastěji k HTML souboru;
  - POST - slouží k získání dokumentu, který umožňuje nahrání dat na server;
  - HEAD - velmi podobná metodě GET, ale získáme pouze informace o dokumentu, nikoliv samotný dokument;
  - DELETE - požadavek na odstranění dokumentu ze serveru.

Níže je uvedena syntaxe pro přenos pomocí HTTP protokolu [9], všechny použitelné příkazy jsou uvedeny na obr. 2.4.

```
%HTTP prenos
AT+HTTPINIT           %zapne sluzbu HTTP
AT+HTTPPARA="URL","<adresa prijimaci stranky>"
%nastaveni URL adresy s cilovym souborem pr. http://stranka.com/prijem.php
AT+HTTPPARA="CONTENT","application/x-www-form-urlencoded"
%nastaveni formatu odesilani
AT+HTTPDATA=<velikost zpravy>,<time (10-65535)>
%nacitadani dat na odeslani max velikost 153600, time - cas na vlozeni dat
DOWNLOAD %po prichodu teto zpravy lze nacitat data
AT+HTTPACTION=1 %potvrzeni odeslani dat, metoda POST = 1
AT+HTTPTERM     %zastavi sluzbu HTTP
```

Command	Description
AT+HTTPINIT	Sart HTTP(S) service
AT+HTTPTERM	Stop HTTP(S) service.
AT+HTTPPARA	Set HTTP(S) Parameter
AT+HTTPACTION	HTTP(S) Method Action
AT+HTTPHEAD	Read the HTTP(S) Header Information of Server Response
AT+HTTPREAD	Read the response Information of HTTP(S) Server
AT+HTTPDATA	Input HTTP(S) Data
AT+HTTPPOSTFILE	Send HTTP(S) Request to HTTP server by File
AT+HTTPREADFILE	Receive HTTP(S) Response Content to a file

Obrázek 2.4: Příkazy pro HTTP [2]

- **Výhody**
  - Maximální velikost zprávy je 153600 znaků;
  - k přenosu stačí použít metodu POST a připojit se na URL adresu s PHP skriptem, který danou zprávu zpracuje a uloží do textového souboru nebo databáze;
  - hosting webové stránky může být zdarma.

- **Nevýhody**
  - Omezení v podobě limitu přenesených dat či velikosti úložiště na bezplatném hostingu.

MQTT

MQTT (Message Queuing Telemetry Transport) je poměrně nový protokol hojně využívaný ke komunikaci mezi dvěma zařízeními (M2M), který vznikl díky rozvoji Internetu věcí (Internet of Things). Do té doby využívaný protokol HTTP nebyl přílíš vhodný z mnoha důvodů, např. kvůli modelu typu žádost-odpověď, náročnější implementaci a delšímu záhlaví zprávy. MQTT je tedy velmi jednoduchý a nenáročný protokol fungující na bázi TCP, nejčastěji na portu 1883 pro nezabezpečenou a 8883 pro zabezpečenou komunikaci (TLS). Využívá návrhového vzoru vydavatel–odběratel (publisher–subscriber) pomocí centrálního bodu tzv. brokera. Broker přijímá, ukládá a třídí zprávy pomocí tzv. témat (topic) a následně je odesílá klientům, kteří dané téma odebírají. Protokol nabízí tři stupně ověřování QoS (Quality of Service) a to 0, 1 a 2. [10]

- QoS 0 - Maximálně jednou
  Funguje podobně jako UDP, pošle zprávu a dál se nestará, jestli byla zpráva ztracena a nebo byla v pořádku doručena.
- QoS 1 - Alespoň jednou
  Je zajištěno, že zpráva dorazí, ale je možné, že bude přijata vícekrát než jednou.
- QoS 2 - Přesně jednou
  Je zajištěno, že zpráva dorazí, ale nestane se, že by mohla být duplikována.

Důležité je zmínit, že kromě QoS ještě může uživatel nastavit tzv. retain flag. Toto nastavení zajistí, že broker si danou zprávu uloží a při připojení subscribera odešle poslední zprávu s tímto nastavením. Broker tedy přeposílá zprávy stejným způsobem jako je přijímá, a to za pomoci stejného nebo nižšího QoS, v případě že subscriber nepožaduje nebo neumožňuje vyšší QoS.

Níže je opět uvedena implementace pro přenos [11], tentokrát za pomocí MQTT protokolu, všechny příkazy tohoto protokolu jsou uvedeny na obr. 2.5. [2]

```
%MQTT prenos
AT+CMQTTSTART       %zapne sluzbu MQTT
AT+CMQTTACCQ=<index>,"<nazev klienta>" %nastaveni jmena klienta pro server
%moznost nastaveni 2 klientu-index(0-1)=> vyuzivame dale prikazech - 0
AT+CMQTTCONNECT=0,"tcp://<adresa serveru>:<port>",<keepalive(0-64800s)>,1
%keepalive-po uplynuti doby bez zadne zpravy posle modul paket na server
AT+CMQTTTOPIC=0,<delka tematu> %po prichodu > vlozime tema o dane delce
AT+CMQTTPAYLOAD=0,<delka zpravy> %po prichodu > vlozime zpravu
AT+CMQTTPUB=0,<QoS>,<timeout>,<retainflag(0-1)
AT+CMQTTDISC=0,<timeout(0,60-180)>   %odhlaseni ze serveru
AT+CMQTTREL=0  %odstraneni klienta s indexem 0
AT+CMQTTSTOP   %vypnuti sluzby MQTT
```

Command	Description
AT+CMQTTSTART	Start MQTT service
AT+CMQTTSTOP	STOP MQTT service
AT+CMQTTACCQ	Acquire a client
AT+CMQTTREL	Release a client
AT+CMQTTSSLCFG	Set the SSL context
AT+CMQTTWILLTOPIC	Input the will topic
AT+CMQTTWILLMSG	Input the will message
AT+CMQTTCONNECT	Connect to MQTT server
AT+CMQTTDISC	Disconnect from server
AT+CMQTTTOPIC	Input the publish message topic
AT+CMQTTPAYLOAD	Input the publish message body
AT+CMQTTPUB	Publish a message to server
AT+CMQTTSUBTOPIC	Input a subscribe message topic
AT+CMQTTSUB	Subscribe a message to server
AT+CMQTTUNSUBTOPIC	Input a unsubscribe message topic
AT+CMQTTUNSUB	Unsubscribe a message to server
AT+CMQTTCFG	Configure the MQTT Context

Obrázek 2.5: Příkazy pro MQTT [2]

- **Výhody**
  - Maximální velikost zprávy až 10240 znaků;
  - open-source broker dostupný ke stažení na mosquitto.org;
  - možnost brokerů zdarma k použití jako maqiatto.com nebo hivemq.com, ale mají opět omezení např. maximální množství témat.
- **Nevýhody**
  - Před každým odesláním zprávy je nutné znovu nastavit topic.

2.2 Mikrokontroler

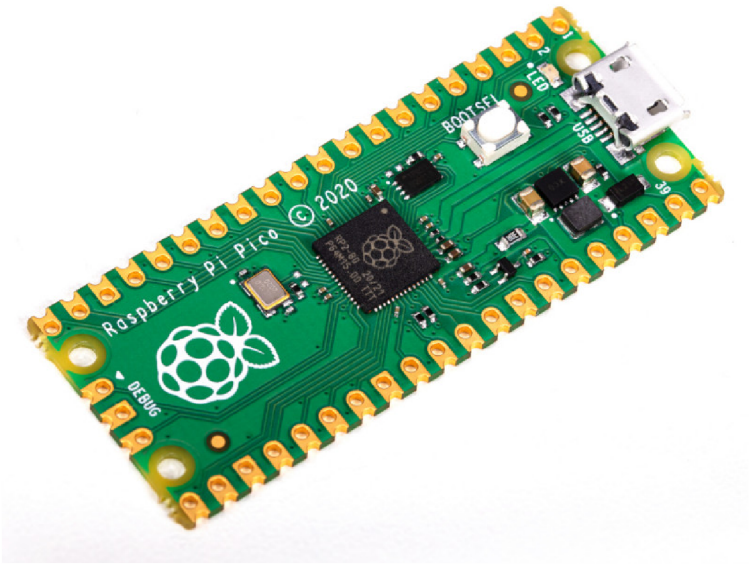
Jedním z cílů práce bylo navrhnout a vyrobit desku plošných spojů s měřící elektronikou, ale z důvodu pandemické situace byl omezen přístup do laboratoře. Místo toho byla po domluvě s vedoucím práce využita vývojová platforma. Těchto platforem je v dnešní době velký počet. Můžeme vybírat např. podle rychlosti procesoru, počtu vstupních nebo výstupních pinů (ADC/PWM), napájecího napětí nebo podle velikosti samotné desky. K těmto platformám je obvykle dostupné i samotné vývojové prostředí a spousta již hotových knihoven pro usnadnění práce. Hlavním kritériem při výběru bude maximální baudrate UARTu, protože je zásadní pro rychlost komunikace s 4G modulem. Pro praktickou část byl výběr uskutečněn mezi Raspberry Pi Pico a Esp32 DEVKitC.

- **ESP32 DevKitC-32D**
  Vývojová deska od společnosti Espressif osazená modulem Esp32-WROOM-32D. Tento modul je založen na dvoujádrovém procesoru Tensilica LX6 s nastavitelnou frekvencí od 80 do 240 MHz. Součástí je také také pamět RAM o velikosti 520 kB a SPI flash pamět s kapacitou 4 MB. Modul je také vybaven bezdrátovou komunikací v podobě rozhraní WIFI 2.4 GHz a Bluetooth 4.0. Má 34 multifunkčních pinů z toho 18 lze použít pro 12-bitový ADC převodník, 2x DAC, 16x PWM. Dále má tato komunikační rozhraní: 3x SPI, SDIO, 2x I2C, 2x I2S, IR a 3x UART s baudratem až 5 Mbit/s. Operační napětí je 3,3 V, ale může být napájeno i ze zdroje o napětí 5 V. Je kompatibilní s programováním v Arduino IDE, ale je možné programovat i v MicroPythonu. Implementován je také operační systém pro mikrokontroléry FreeRTOS, pomocí kterého můžeme na jednotlivých jádrech procesoru spoustět různé úlohy.[12]

Obrázek 2.6: ESP32 DevKitC V4 [34]

- **Raspberry Pi Pico**
  Jedná se o poměrně nový mikrokontrolér. Je zároveň poměrně levný a velmi výkonný, protože je osazen čipem RP2040 a dvoujádrovým procesorem Arm Cortex M0+

s frekvencí až 133 MHz, vestavěnou 246 kB pamětí SRAM a 2MB Flash pamětí. Má 30 multifunkčních pinů, z toho čtyři z nich lze použít pro 12-bitový ADC převodník. Dále nabízí tyto digitální periferie: 1x časovač se 4 alarmy, 1x čítač v reálném čase, 1x teplotní senzor, 2x I2C, 2x SPI, až 16 PWM kanálů a 2x UART s baudratem až 912600 bits/s. Napájen je pomocí spínaného integrovaného zdroje, který vytváří napětí 3,3V ze širokého pásma vstupních napětí (-1,8 až 5 V). Podporován je napříč vývojovými prostředími, ať už v jazyce C/C++ či MicroPythonu. [13]



Obrázek 2.7: Raspeberry Pi Pico [13]

2.3 Webová aplikace

Tvorba webové aplikace se obvykle rozděluje na 2 části, Frontend a Backend. Backend je serverová část aplikace a stará se o celou řadu věcí jako je správa databáze, různé výpočetní operace nebo příprava dat pro klienta. Zatímco Frontend je část aplikace, která běží ve webovém prohlížeči a jejím úkolem je připravená data zobrazit uživateli a odesílat požadavky zpět na server. [14]

2.3.1 Frontend

Webová stránka bude sloužit pro zobrazení právě příchozích dat a pro načtení starších dat z databáze. Proto bude rozdělena na dvě stránky. Pro obě tyto stránky budeme potřebovat knihovnu pro vykreslování grafů, dále nastavovací prvky jako např. výběr zobrazované hodnoty nebo data, pozastavení grafu a další. K vytvoření webové stránky budou použity jazyky HTML, CSS a Javascript, které se k tomuto účelu běžně využívají.

Knihovna grafů

Knihoven pro vykreslování grafů existuje celá řada. Některé knihovny jsou placené, ale lze využít i bezplatné. Placené varianty jsou někdy použitelné zdarma pouze pro nekormerční účely nebo mají omezenou funkčnost. Nabízejí mnoho druhů grafů jako např. sloupcový,

koláčový, spojnicový či bodový. Pro tuto práci bude vybrána zdarma použitelná knihovna, která podporuje dynamické obnovování grafů, nejlépe v reálném čase.

- **Highcharts.js**

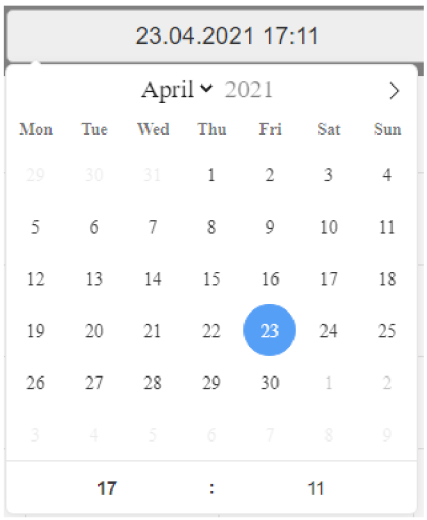
  Tato knihovna nabízí mnoho různých typů grafů, ať už pro zobrazování cen akcií nebo velkého množství datových bodů. Je volně k použití pro nekomerční účely. Při testování vykreslování grafu v reálném čase s frekvecí hodnot alespoň 100 Hz se objevovaly potíže s načítáním dat do grafu. [16]

- **Chart.js v2.8.0**

  Tato knihovna neobsahuje tolik druhů grafů jako knihovna Highcharts, ale její plná verze je k použití zcela zdarma. Na jejich oficiálních stránkách lze najít odkaz na Github, kde jsou dostupná rozšíření jejich knihovny. Jedno z nich přidává podporu pro živý přenos dat. Toto rozšíření má mnoho nastavení jako např. počet FPS, délka trvání, zpoždění grafu a další. Po otestování opět s frekvencí 100 Hz fungovala tato knihovna s rozšířením obstojně, a proto s ní budeme dále pracovat v praktické části této práce. [17] [18]

Výběr data a času

HTML nabízí několik druhů vkládacích elementů. Jeden z nich nabízí výběr data i času, a to konktrétně s typem "datetime-local". Tento typ ale bohužel není podporován v internetových prohlížečích Safari, Firefox ani Internetu Explorer 12 (nebo dřívější verze). [19] Proto bude využita další javascriptová knihovna a to konkrétně Flatpickr.



Obrázek 2.8: Ukázka knihovny Flatpickr

Knihovna podporuje vkládání dat mnoha časových formátů, deaktivaci konktrétních dat, vyobrazení data ve svém zvoleném formátu a mnoho dalších možností. [20]

Ostatní prvky

Zbylé prvky stránky jako tlačítka, jezdce pro zvolení doby trvání grafu, rozbalovací nabídka pro výběr zobrazované hodnoty, měnící texty budou využity prvky z HTML 5.0 a dostylovány pomocí CSS.

2.3.2 Backend

Při vývoji backendu webových aplikací se často využívá tzv. Web Application Framework. Jedná se o knihovnu, která usnadňuje a urychluje vývoj takovýchto aplikací. Mezi webové frameworky patří Flask a Django, Ruby on Rails, Node.js a další. Na základě předchozí zkušenosti s jazykem Python je výběr zaměřen pouze na Flask nebo Django. [15]

- **Flask**
  Microframework Flask je jednoduchý, nenáročný a nabízí větší flexibilitu a modifikovatelnost. Obsahuje pouze základní nástroje pro vývoj webových aplikací, ale zároveň nezakazuje přidání externích knihoven. Proto se hodí na menší projekty, které je nutné rychle vyvinout. Je také vhodný pro začátečníky a na prototypování. [21], [22]

- **Django**
  Framework Django je na trhu delší dobu, proto má širokou základnu uživatelů, kteří případně dokáží poradit. Je plně vybaven funkcemi a nemusí využívat externí knihovny, proto by měl být bezpečnější. Nabízí admin panel, podporu různých databázových systémů, adresářovou strukturu pro aplikace nebo předpřipravené šablony, které nám mohou ušetřit čas při vývoji. Je vhodnější pro větší a složitější projekty. [21], [22]

2.3.3 Ukládání dat

Po přijetí dat na serveru je potřeba tato data uložit pro případné načítání z historie. K uložení dat může sloužit i obyčejný textový soubor, ve kterém jsou data oddělena pomocí speciálních předem daných znaků. Toto řešení ale není přílíš vhodné pro další práci s takto uloženými daty. Proto se využívají databáze, které byly navrženy a vytvořeny speciálně pro práci s velkými objemy dat. Pomocí jejich systémů se jednoduše ukládají, mažou či vyhledávají určená data. Nejčastěji se dělí na modely Relační (SQL) a Nerelační (NoSQL) databáze. [23]

- **Relační (SQL) databáze**
  Tento typ databáze je založen na tabulkách, řádky chápeme jako jednotlivé záznamy. Primární klíč většinou definuje číslo záznamu a tzv. cizí klíče uchovávají informaci o vztazích mezi jednotlivými řádky (záznamy) nebo odkazují na záznamy v jiné tabulce. U tabulek je nutné při vytváření definovat pevnou strukturu, která se následně musí dodržovat. Tento druh databáze nabízí pouze vertikální rozšiřitelnost pro plnou funkčnost, což znamená, že výkon můžeme přidat pouze vylepšováním komponent serveru. Příklady známých databázových systémů jsou MySQL, PostgreSQL, SQLite. [24]

- **Nerelační (NoSQL) databáze**
  Tento druh databáze naopak nepotřebuje předem definovat schéma ukládaných dat, je možné snadno přidat další hodnotu k již vytvořenému dokumentu. Nabízí proto větší flexibilitu a je možné ukládat do tabulek, JSON dokumentů nebo grafů. Dále nabízí horizontální rozšiřitelnost, což znamená, že můžeme databázi bez problémů rozšířit na více serverů. Příklady známých databázových systémů MongoDB, Cassandra DB, Reddis. [24]

Pro realizaci praktické části bude vybrána volně dostupná relační databáze SQLite. Nejedná se o klasikou databázi typu klient-server, ale o databázi vestavěnou přímo do aplikace. Vybraná je z důvodu, že příchozí data budou strukturována, podporuje multitransakce a má menší výkonové požadavky na server než např. MySQL, která je typu klient-server.

2.3.4 Hosting serveru

Z mnoha důvodů se dnes nevyplatí provozovat vlastní server pro různé projekty. Společností, které nabízejí takovéto služby, je na trhu poměrně hodně a nabízejí různě výkonově odstupňované servery, jak pro menší, tak i pro větší a náročnější projekty. Pro menší projekty se rozhodně vyplatí takových služeb využít, jelikož jsou poměrně cenově dostupné a uživatel se nemusí fyzicky starat o server v případě nějakých hardwarových potíží.

- **Digital Ocean**
  Jedna z předních firem poskytující cloudové služby, nabízí také mnoho druhů virtuálních privátních serverů s názvem "Droplets" s cenou od 5$ měsíčně. V této nejlevnější konfiguraci nabízí:
  - 1 vCPU
  - 1 GB RAM
  - 25 GB SSD
  - 1000 GB přenosu dat

- **Linode**
  Je jednou z předních firem, které nabízí webhostingové služby. Kromě toho také nabízí virtuální privátní servery s názvem "Linode". Cena se pohybuje od 5$ do 960 $ měsíčně v závislosti na konfiguraci. V nejlevnější konfiguraci nabízí:
  - 1 vCPU
  - 1 GB RAM
  - 25 GB SSD
  - 1000 GB přenosu dat

K oběma hostingovým platformám se dají sehnat slevové a dárkové kódy pro nové uživatele. Tyto kódy nám darují kredit na placení jejich služeb.

2.4 Zhodnocení rešerše

Bylo zjištěno, že modul SIMCom 7600E-H nabízí 5 typů protokolů pro přenos dat pomocí 4G a byly popsány jejich možnosti implementace pro další použití. Konečný výběr použitého protokolu bude určen až po otestování všech dostupných protokolů v kapitole 3.1.

V kapitole 2.2 jsou uvedeny vybrané mikrokontroléry pro sestavení měřící stanice. Při realizaci bude použit mikrokontrolér ESP32 DevKitC-32D především z důvodu sériové komunikace s baudratem až 5 Mbit/s. Mimo jiné je také výkonnější, nabízí větší množství periferií a je dostupný v laboratoři.

Webová stránka bude zobrazovat data za pomoci knihovny grafů Chart.js a jejího rozšíření pro livestream dat. Dále budou použity standardní prvky HTML 5.0 a pro výběr data s časem bude využita knihovna Flatpickr.

Backend webové aplikace poběží na frameworku Flask, jelikož je vhodnější na menší projekty, prototypování a pro rychlý vývoj. Jako databáze bude použita relační databáze SQLite, protože data budou mít jasnou neměnnou strukturu, podporuje multitransakce při zápisu a je méně náročná na výkon oproti jiným databázím.

Při porovnání nejnižších konfigurací hostingových platforem bylo zjištěno, že obě nabízí stejné služby a dají se za pomocí kuponů využít zdarma. Nakonec bylo rozhodnuto pro využití platformy Linode.

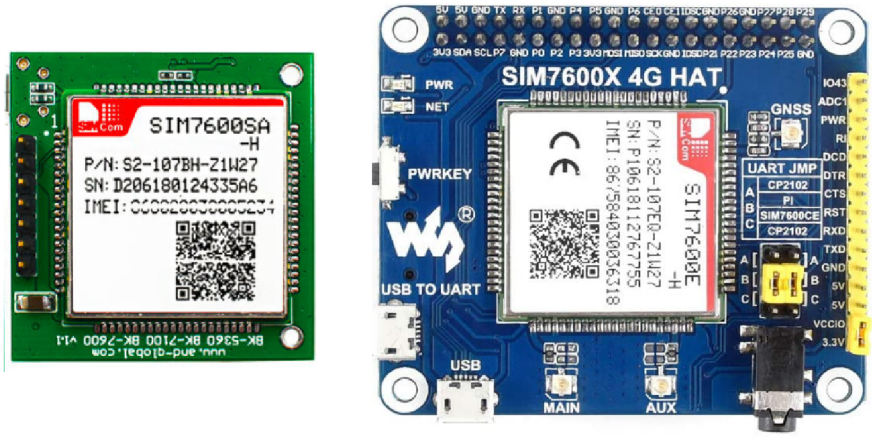
3 Praktická část

3.1 Testování přenosů - Matlab

Otestování přenosů bylo prováděno za pomoci programu MATLAB a jeho knihovny pro sériovou komunikaci *serialport*. Nakonfigurování sériové komunikace je uvedeno níže. Nejprve je ale potřeba vytvořit objekt, který nám definuje přenos pomocí COM port a baudrate. Následně je potřeba ještě k danému objektu nastavit ukončovací znak každé zprávy, v našem případě podle kapitoly 2.1.2 je to "CR". Dále jsou uvedeny příkazy pro vkládání a čtení zpráv sériové komunikace s vypsáním do příkazového řádku (Command Window). Čas byl měřen pomocí funkcí *tic* a *toc*.

```
modul = serialport("COM3",3000000);
configureTerminator(modul,"CR");
writeline(modul,'<prikaz>')
fprintf('%s \n',readline(modul))
```

K testování jsou k dispozici dvě různé desky osazené modulem *SIMCom A7600E-H*. Jedna od společnosti AND Technologies Co., ltd s názvem *BK-SIM7600* (obr. 3.1 vlevo) a druhá od společnosti Waveshare s názvem *SIM7600E-H 4G HAT for Raspberry Pi, LTE Cat-4 4G/3G/2G,GNSS, for Europe, Southeast Asia, West Asia, Africa* (obr. 3.1 vpravo). Výhodou u druhého zmíněného je, že nabízí slot pro MicroSD karty, proto bude možné otestovat přenos pomocí FTP i na větších souborech, které by se už nevešly do paměti modulu samotného.

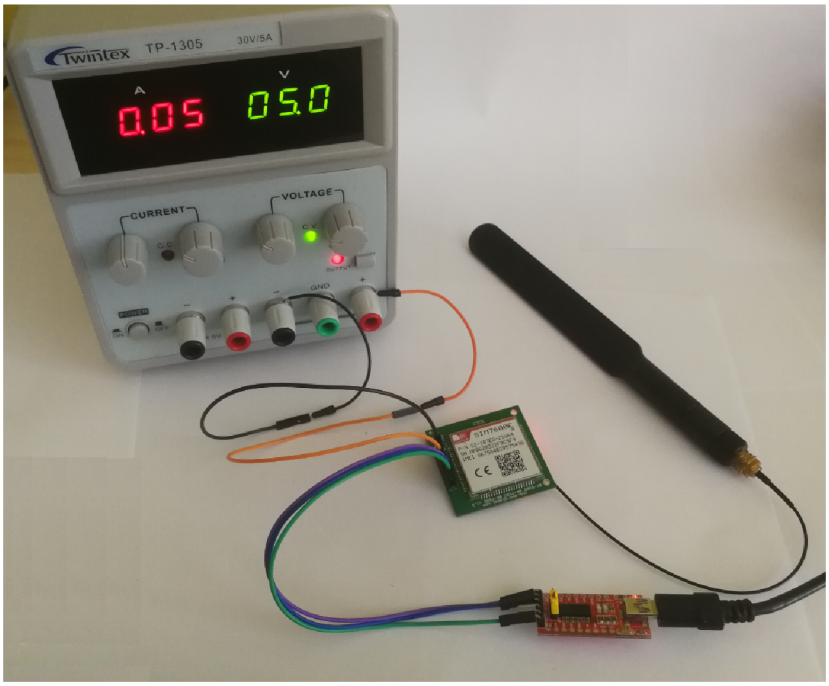


Obrázek 3.1: Ukázky vývojových platforem s modulem SIMCom A7600E–H [35] [36]

- **Zapojení modulu**
  Pro zapojení modulu BK-SIM7600 využijeme následující piny:

  – G - GND - uzemnění zdroje a sériové komunikace;
  – R - RX - příjem dat ze sériové komunikace;
  – T - TX - odesílání dat přes sériovou komunikaci;
  – V - VCC - připojení zdroje 5-10 V (doporučeno 5 V), maximální proud 1,5 A.

  Sériová komunikace využívá 3,3V TTL a pro komunikaci s počítačem je zapotřebí převodník do USB. Dále musíme zapojit anténu do konektoru pro LTE ze zadní strany modulu pro příjem signálu a vložit Micro SIM kartu. Celé zapojení je na obrázku 3.2.



Obrázek 3.2: Zapojení BK–SIM7600 ke zdroji a převodníku sériové komunikace do USB

3.1.1 TCP

Pro testování byla využita syntaxe uvedená v kapitole 2.1.3. Dále bylo potřeba vytvořit TCP server, ten byl realizován pomocí jazyka Python a knihoven *socket* [26] a *threading*. [27] Python byl vybrán z důvodu, že se jedná o interpretovaný vysokoúrovňový jazyk a nabízí dynamickou kontrolu datových typů, která nám usnadní práci při vývoji. Navíc bude použit i pro backend webové aplikace.

Z důvodu toho, že je TCP spojově orientováno, musíme využít i již zmíněné knihovny *threading*, která nám při vytvoření TCP spojení oddělí jednotlivá připojení do vláken, která jsou schopna fungovat paralelně.

Po vložení knihoven je dobré si definovat parametry jako IP adresu vytvářeného serveru, port a kodování zpráv.

```
PORT = xxxx
SERVER = 'xxx.xxx.xxx.xxx'
ADDR = (SERVER, PORT)
FORMAT = 'utf-8'
```

Dále vytvoříme soket, který musí obsahovat dva parametry. Prvním zadávaným parametrem je *socket.AF_INET*, který definuje typ IP adresy (IPv4). Druhým parametrem bude socket.SOCK_STREAM, který označuje spojitou cestu dat (TCP protokol). Následně připojíme vytvořený soket k definovaným parametrům serveru.

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(ADDR)
```

Poté bude spuštěna funkce pomocí které budeme vytvářet nová spojení. Nejprve je zapotřebí vytvořit frontu pro nová spojení. Následně přijmeme spojení a vytvoříme ho v novém vlákně pomocí knihovny *threading* a funkce, která se bude starat o příchozí zprávy.

```
def start():
    server.listen()
    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
```

U funkce *handle_client* sloužící pro správu jednotlivých spojení je nejdůležitější příkaz *conn.recv(1500).decode(FORMAT)*, pomocí kterého se přijímají zprávy a dekódují zprávy z modulu. Pokud není příchozí zpráva prázdná, vypíše se do terminálu. Pokud zpráva obsahuje předem domluvenou zprávu na ukončení spojení, tak se následně spojení ukončí.

```
def handle_client(conn, addr):
    connected = True
    while connected:
        msg = conn.recv(1500).decode(FORMAT)
        if msg:
            if msg == "!DISCONNECT":
                connected = False
            print(f"[{addr}] {msg}")
    conn.close()
```

Tento skript pro TCP server spustíme na hostingu serveru od Linode a můžeme testovat přenos.

3.1.2 UDP

Pro testování byly využity příkazy již uvedené v kapitole 2.1.3. Pro otestování bylo nutné vytvořit UDP server, ten byl realizován stejně jako v předchozí kapitole pomocí jazyka Python a jeho knihovny *socket*. [26]

Po definování parametrů (IP adresa, atd.) jako u serveru TCP následuje také vytvoření soketu, opět se dvěma parametry. První parametr zůstává stejný, ale u druhého musíme zadat socket.SOCK_DGRAM pro protokol UDP.

```
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind(ADDR)
```

Po připojení soketu k serveru už jen v cyklu přijímáme příchozí zprávy pomocí příkazu server.recvfrom(1500) s maximální délkou zprávy 1500 znaků. Po přijetí se zpráva dekóduje pomocí UTF-8 a vypíše se do terminálu.

```
while True:
    data, addr = server.recvfrom(1500)
    data = data.decode(FORMAT)
    print(f"{addr} {data}")
```

3.1.3 FTP

K otestování tohoto přenosu byl využit FTP server dostupný k webovému hostingu na www.webzdarma.cz. Bezplatně nabízený hosting disponuje úložným prostorem o velikosti 500 MB, což byla pro testovací účely dostatečná kapacita. K přihlášení k serveru je zapotřebí využít údaje, které přijdou uživateli emailem. Následně pomocí příkazů zmíněných v kapitole 2.1.3 byl vytvořen skript pro otestování v softwaru MATLAB.
Při tomto testování byl využit druhý modul od společnosti Waveshare, jelikož umožňuje vložení micro SD karty a tím i odeslání větších souborů. Modul byl nápajen i připojen k počítači pomocí micro USB kabelů.

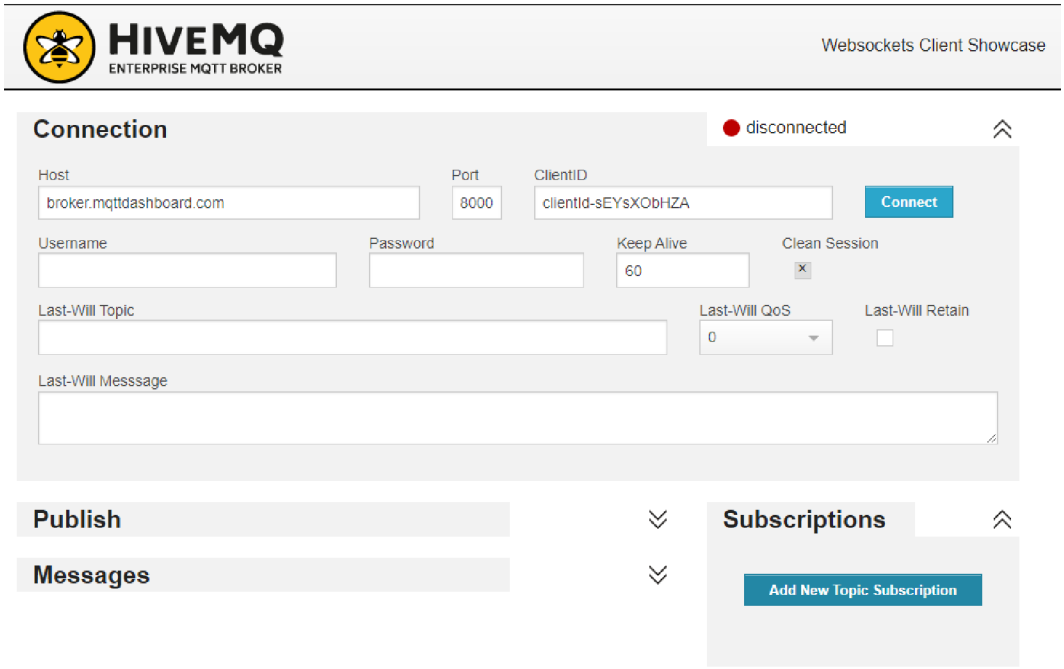
3.1.4 HTTP

Pro testování protokolu HTTP byl také využit hosting webzdarma.cz. K přijímání dat byl vytvořen PHP skript, pomocí kterého se příchozí zprávy ukládaly do textového souboru. Skript získává data pomocí příkazu *$_POST['msg']*. Je důležité, aby při tomto způsobu příjmu dat byla odeslaná zpráva ve tvaru *msg=<odesilana zpráva>*. Následně se otevře textový soubor a připraví data na zápis ve tvaru <čas> - <zpráva>. Poté se pomocí příkazu *file_put_contents* uloží na konec textového souboru. LOCK_EX uzamkne soubor při zapisování. Použité příkazy pro modul jsou okomentovány v kapitole 2.1.3.

```php
<?php
$time = time();
$value = $_POST['msg'];
$file = 'database.txt';
$data = $time." - ".$value."\n";
file_put_contents($file, $data, FILE_APPEND | LOCK_EX);
?>
```

3.1.5 MQTT

Protokol MQTT byl testován pomocí dostupného online brokera HiveMQ na adrese broker.hivemq.com na TCP portu 1883. Kontrolování příchodu zpráv je poté možné na stránce www.hivemq.com/demos/websocket-client viz obr. 3.3. Zde se stačí jen připojit a začít odebírat stejné téma (topic), pod kterým dané zprávy publikujeme. Příkazy pro modul byly již zmíněny v kapitole 2.1.3.



Obrázek 3.3: HiveMQ Websocket client

3.1.6 Výsledky a zhodnocení testování

Testování proběhlo na zprávách o velikosti 1500 znaků, případně 3200 znaků pro protokoly, které to umožňují. Měření byla provedena pro 100 a 1000 odeslaných zpráv, kdy každá z variant byla měřena 10x. Měření bylo také realizováno pro dva různé baudraty. Výpočet rychlosti přenosů dat v tabulkách níže vychází z velikosti odeslaných dat a naměřeného času, nezohledňuje přenos dat z hlavičky zprávy (overhead).

Po vyzkoušení jednotlivých protokolů a syntaxe z kapitoly 2.1 bylo zjištěno, že odesílání dat pomocí protokolu FTP je pro rychlý přenos malého množství dat nevhodný, jelikož odeslání jednoho souboru o velikosti 3200 znaků trvalo přes 11 s, viz tabulka 3.1. Naopak pro velké soubory dat je protokol FTP vyhovující, proto byl narozdíl od ostatních protokolů testován samostatně i pro větší soubory dat, které byly předem vytvořeny a nahrány na SD kartu. Z tabulky 3.1 je patrné, že dosáhl přenosové rychlosti až 918 kB/s při odesílání souboru o velikosti 100 MB.

Tabulka 3.1: Přenos pomocí FTP pro soubory od 3,2 kB do 100 MB

vel. souboru[kB]	3,2	5	10	50	100	500	1 000	10 000	100 000
průměr [s]	11,74	11,86	11,87	11,90	11,96	12,35	12,80	20,99	106,34
přenos [kbit/s]	2,13	3,29	6,58	32,82	65,31	316,31	610,13	3722,42	7346,59
přenos [kB/s]	0,27	0,41	0,82	4,10	8,16	39,54	76,27	465,30	918,32

Další testování bylo provedeno pro zjištění, jak je přenosová rychlost zavislá na velikosti baudratu. V tabulce 3.2 jsou zobrazeny výsledky měření protokolů HTTP a MQTT pro velikosti zprávy o 3200 znacích pro baudrate o velikosti 115200 bit/s a 3 Mbit/s. Z tabulky 3.2 vyplývá, že čím je baudrate větší, tím větší je i objem přenesených dat.

Tabulka 3.2: Měření pro zjištění závislosti na baudratu pro zprávu o 3200 znacích z 10 měření

Baudrate	115 200				3 000 000			
Počet opakování	100		1000		100		1000	
	HTTP	MQTT	HTTP	MQTT	HTTP	MQTT	HTTP	MQTT
průměr [s]	85,3	82,0	864,4	804,3	34,4	25,7	299,1	245,5
1 opakování [s]	0,9	0,8	0,9	0,8	0,3	0,3	0,3	0,2
přenos [kbit/s]	29,3	30,5	28,9	31,1	72,7	97,1	83,6	101,8
přenos [kB/s]	3,7	3,8	3,6	3,9	9,1	12,1	10,4	12,7

Další měření proto na základě předchozích výsledků proběhlo již jen s vyšším baudratem, konkrétně 3 Mbit/s. Testovány byly protokoly HTTP, MQTT, TCP a UDP. Pro lepší porovnání protokolů byla velikost zprávy v této části měření určena 1500 znaky, jelikož protokoly TCP a UDP mohou přenést maximálně 1500 znaků. Z tabulky 3.3 vychází nejlépe protokoly TCP a UDP s rychlostí odeslání jedné zprávy přibližně za 0,05 s.

Tabulka 3.3: Porovnání přenosů pro baudrate 3Mbit/s a zprávu o 1500 znacích z 10 měření

Počet opakování	1000			
	HTTP	MQTT	TCP	UDP
průměr [s]	269,49	314,47	47,37	48,10
1 opakování [s]	0,27	0,31	0,047	0,048
přenos [kbit/s]	43,49	37,27	247,38	243,61
přenos [kB/s]	5,44	4,66	30,92	30,45

S přihlédnutím k dosaženým výsledkům bylo vyhodnoceno, že následná aplikace bude rozdělena na dvě části. První částí aplikace bude přenos dat v reálném čase a druhou částí bude intervalový přenos dat. Pro každou z těchto částí budou vybrány vhodné protokoly.

- Přenos dat v reálném čase
  Pro tuto aplikaci byly vybrány protokoly TCP a UDP, protože dosahovaly podobných hodnot, a to nejmenší latence a největšího přenosu dat. Tyto dva protokoly byly ještě otestovány pro větší počet zpráv (až 150 tisíc) s maximální velikostí. Opět dosahovaly stejných výsledků, avšak při použití protokolu UDP některé zprávy nedorazily. Proto bylo rozhodnuto o použití protokolu TCP pro tuto aplikaci a bude popsána v kapitole 3.2.

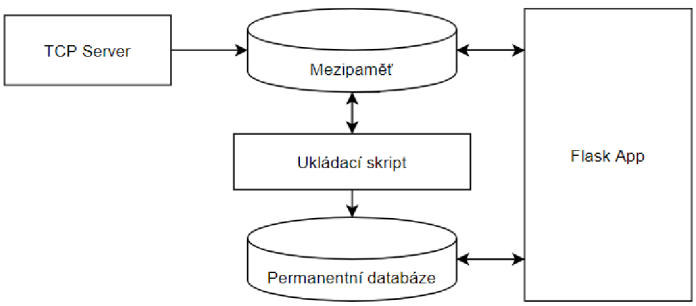
- Intervalové odesílání dat
  Pro tuto aplikaci byl vybrán protokol FTP, protože zvládl odesílat velké soubory dat a dosahuje největší přenosové rychlosti. Aplikace bude více popsána v kapitole 3.3.

3.2 Přenos dat v reálném čase – TCP

První část aplikace je zaměřena na přenos dat v reálném čase pomocí protokolu TCP. V kapitole bude popsána funkce backendu, frontendu a samotné měřící stanice.

3.2.1 Backend

Backend se skládá ze tří částí a funguje dle následujícího schématu na obr. 3.4, které popisuje, jak jsou data předávána mezi nimi. Jednotlivé části budou dále představeny.



Obrázek 3.4: Schéma backendu aplikace

Ukládání dat

Ukládání dat bylo rozděleno na dvě části:

- Mezipaměť - pro rychlé načítaní dat
  Ačkoliv server disponuje rychlými SSD disky, nestačilo to pro dostatečně rychlé načtení posledních hodnot z databáze. Operační systém Ubuntu ale nabízí vytvoření tzv. Ramdisku, což problém vyřešilo. Ramdisk je úložiště v paměti RAM, které je mnohonásobně rychlejší než SSD disk, ale její nevýhodou je, že při vypnutí serveru nebo počítače se z ní data smažou. To pro využití v této práci nevadí, jelikož se zde vždy bude ukládat jen posledních pár sekund dat, zbylá data budou již uložená v permanentní databázi. Ramdisk je při vytváření připojen ke složce na SSD disku, pomocí níž k Ramdisku přistupujeme a vkládáme do něho data. Data z TCP serveru byla ukládána rovnou do mezipaměti, které byly realizovány databází SQLite.
- Databáze pro historii
  Pro permanentní uložení dat byla také využita databáze SQLite. Pro ukládání do permanentní databáze z mezipaměti byl vytvořen skript. Ten čte záznamy z mezipaměti a jednotlivá data rozděluje podle roku a měsíce do tabulek v permanentní databázi.

K práci s databází se využívá standartní SQL syntaxe [28]. Ukázka některých příkazů je uvedena níže.

```
%pripojeni k databazi
db = sqlite3.connect('/database/data-history.db')
c = db.cursor()
%vyhleda v tabulce data hodnoty created time a valueA pro id = (500;600)
c.execute('SELECT created_time,valueA FROM data where id>500 AND id<600')
%ukladani zaznamu do tabulky data
Q1 = "INSERT INTO data (created_time,valueA,valueB) VALUES (?,?,?)"
c.execute(Q1,pub_msg)
db.commit()
```

TCP server

Pro tento účel byl využit již vytvořený server popsaný v kapitole 3.1.1. Bylo ho nutné ovšem modifikovat. Bylo zapotřebí jednak doplnit ukládání dat do mezipaměti, ale také předělat část pro příjem zprávy, z důvodu problémů s rozdělováním a ukončením příjmu zprávy dříve než dorazí celá. Příjem zprávy tedy bude probíhat po jednotlivých znacích, dokud server nepřijme ukončovací znak, kterým je #. Po navázání spojení je důležité vytvořit připojení k databázi a kurzoru pro práci s ní. Po prvotním přijetí zprávy je zjištěn čas a datum. Následně se zpráva rozdělí na jednotlivá měření podle znaku & a funkce *split()*. Dále bude následovat cyklus, ve kterém se jednotlivé podzprávy budou rozdělovat

pomocí znaku ; na jednotlivé měřené kanály. Na začátek se ještě přidá informace o čase a datu. Po dokončení celé zprávy se zápis potvrdí a z mezipaměti se odstraní starší data. Níže je uvedena část skriptu zabývající se zpracováním zprávy.

```
split_msg = msg.split('&')
for x in range(len(split_msg)-1):
    pub_msg = split_msg[x].split(';')
    pub_msg.insert(0,str(cas))
    pub_msg.insert(0,str(datum))
    cas += 10
    datum += timedelta(milliseconds=(10))
    c.execute(Q1,pub_msg)

db.commit()
c.execute('DELETE FROM data WHERE id<{}'.format(str(c.lastrowid-600)))
db.commit()
```

Flask

Po přidání potřebných knihoven (Flask, render_template, request, sqlite3 a další) je vytvořena instance Flasku. Následně byly definovány jednotlivé cesty pro různé URL, které se budou pomocí flasku zobrazovat nebo využívat. Na konci skriptu je potřeba ještě vytvořenou instanci flasku spustit. Příklad vytvoření a spuštění flasku s jednou stránkou s vrácením dokumentu HTML je ukázán níže.

```
app = Flask(__name__)

@app.route('/')
def main():
    return render_template('final.html')

if __name__ == "__main__":
    app.run(debug=True)
```

Pro účely aplikace jsou využity dohromady čtyři cesty pro URL. Dvě se budou starat o zaslání HTML dokumentu pro živý přenos a historii. Zbylé dvě budou obstarávat žádosti o data a a jejich přípravu z databáze a mezipaměti pro jednotlivé stránky. Žádané proměnné pro přípravu dat dostaneme pomocí příkazu *request.args['first']*, jejímž argumentem je název, pod kterým ji přiřazujeme na webové stránce. Žádaná data vracíme ve formátu json.

Nastavení serveru

Po vytvoření účtu na stránce Linode a serveru s operačním systémem Ubuntu 20.04 LTS musíme tento server nastavit. Pro připojení k serveru bude využit program Putty a pro správu disku a souborů program WinSCP.

- Apache2 - Flask
  Nejdříve je nutné nainstalovat program Apache2, který bude zajišťovat HTTP server. Po nainstalování je nutné vypnout firewall pro tento program.

```
sudo apt install apache2
sudo ufw allow 'Apache'
```

  Dále je nutné doinstalovat ještě další potřebné knihovny.

```
sudo apt install python3-pip
sudo apt install apache2-dev libapache2-mod-wsgi-py3
sudo pip3 install mod_wsgi
sudo pip3 install flask
sudo apt install sqlite3
```

  Po nainstalování vložíme složku webApp do /var/www/webApp. Tato složka musí obsahovat tři složky (static,templates,views) a soubor s názvem ___init___.py, který bude aplikaci flask spouštět. Do složky /etc/apache2/sites-available/ je potřeba vložit soubor webApp.conf, který konfiguruje server Apache2. Pro použití na serveru je nutné změnit IP adresu na prvním řádku. Všechny soubory jsou dostupné v příloze ve složce server. Pro povolení a spustění aplikace zadáme ještě následující příkazy.

```
sudo a2ensite webApp
systemctl reload apache2
sudo service apache2 restart
```

- Ramdisk
  Pro vytvoření Ramdisku je nutné nejdříve vytvořit složku s dostatečnými právy, ke které bude připojen. Tu vytvoříme pomocí následujicích příkazů. [30]

```
sudo mkdir /mnt/ramdisk          %vytvoreni slozky
sudo chmod 777 /mnt/ramdisk      %nastaveni prav
```

  Pro připojení ramdisku vždy při zapnutí serveru je potřeba do souboru s cestou /etc/fstab vložit následující příkaz, který při startu vytvoří ramdisk připojený k vytvořené složce o velikosti 50MB. Po restartování by se měl ramdisk automaticky připojit.

```
myramdisk  /mnt/ramdisk  tmpfs  defaults,size=50M,x-gvfs-show  0  0
```

- spustění skriptů - TCP server, přepis databáze
  O spuštění vytvotřených skriptů se stará program Cron. Náš požadavek vložíme do Cronu pomocí následujících příkazů. Prvním příkazem otevřeme soubor, do kterého nakonec zapíšeme kdy, v jakém jazyce a jaký skript má být spuštěn (musí být uvedená celá cesta k souboru). Pak už jen soubor uložíme a po restartování serveru by se měly skripty spustit. [31]

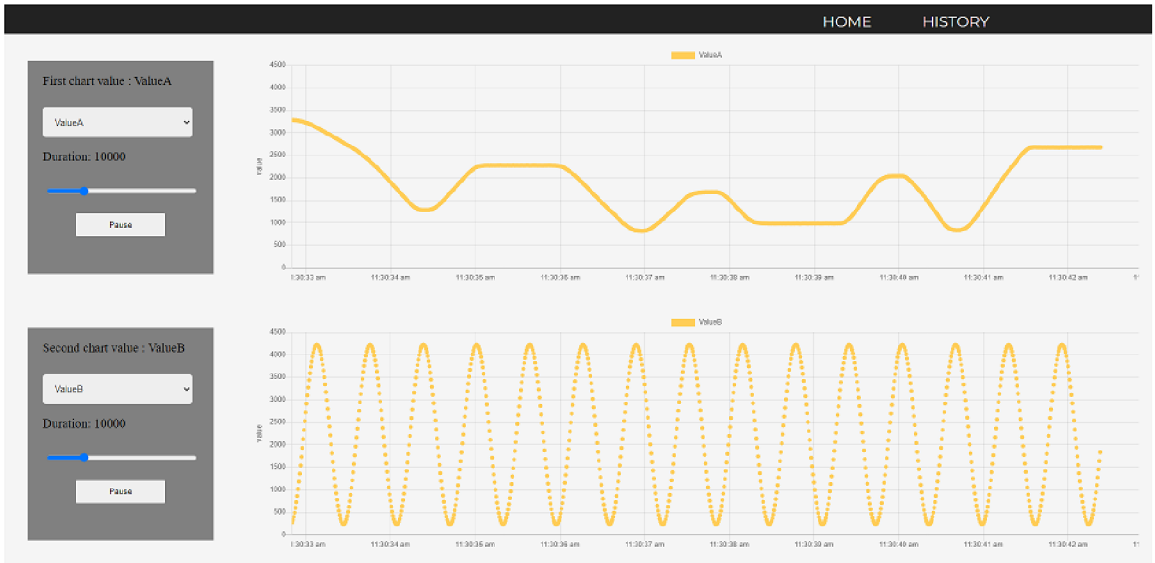
```
sudo crontab -e  %vypiseme do terminalu
%vlozit na konec souboru
@reboot python3 /root/python/saving.py &
@reboot python3 /root/python/server.py &
```

3.2.2 Frontend

Frontend je rozdělen na dvě části. První částí je stránka zobrazujucí příchozí data v reálném čase a druhou je stránka, kde se dají data zobrazit z historie.

Homepage

Na stránce pro živé vykreslení dat (obr. 3.5) jsou zobrazeny dva grafy, každý pro vykreslování jiného měřeného kanálu. K jednotlivým grafům je doděláno základní nastavení, kde se dá zvolit měřený kanál, pozastavit graf nebo nastavit délka grafu pomocí slideru.



Obrázek 3.5: Snímek domovské stránky

Jak již bylo zmíněno k vykreslování grafů bude využita knihovna Chart.js s rozšířením pro livestream dat. Je potřeba nejdříve importovat knihovny do HTML pomocí cdn. Dále

pak rozšíření do grafu přidáme nastavením typu v *options* a dalších nastavení jako je délka či zpoždění. Ještě je důležité přidat nastavení plugins, ve kterém zvolíme streaming a nastavíme počet překreslení za vteřinu (0-60), jak je uvedeno níže. [18]

```
options: {
    scales: {
        xAxes: [{
            type: 'realtime',
            realtime: {
                duration: 10000,
                ttl: 10000,
                delay: 500,
                }
        }],
    plugins: {
        streaming: {
            frameRate: 10
        }
    },
}
```

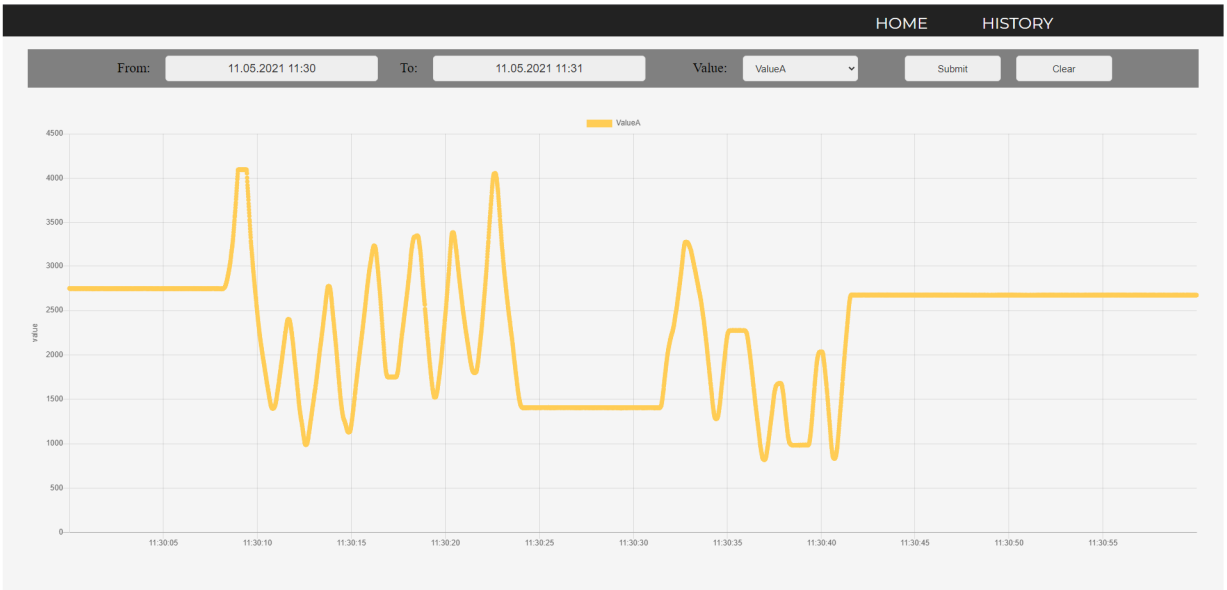
Takto máme vloženy dva grafy a k nim je doděláno nastavení grafů viz obr. 3.5. Nastavení je vytvořeno pomocí základních elementů HTML. Pomocí Javascriptu se mění nastavení grafů jako je pozastavení grafu, výběr zobrazované hodnoty či změna délky grafu.

Načítání dat na stránku je realizováno automatickým voláním vytvořené funkce *updating()*. Tato funkce pomocí knihovny ajax a její funkce *$.ajax()* odešle na server požadavek s hodnotami (posledním id načtených hodnot, hodnoty z selectboxů pro výběr zobrazovaných hodnot). Po úspěšném požadavku se vrácená data ve formátu json načtou do grafu.

Historie

Na stránce pro načítání a zobrazování dat z historie (obr. 3.6) je pouze jeden graf, ke kterému je opět doděláno potřebné nastavení. Můžeme nastavit vykreslení měřeného kanálu v časovém rozmezí pomocí dvou buněk pro výběr data a času, dále zvolit měřený kanál a dvěma tlačítky potvrdit a nebo smazat výběr.

Využita byla opět knihovna Chart.js s pluginem na zoomování v grafu. [29] Implementace pluginu je ukázána níže. Načítání dat je vyřešeno obdobným způsobem jako na hlavní stránce. Spustí se pouze po stiknutí tlačítka *Submit*.
Výběr data a času je vyřešen pomocí knihovny Flatpickr. Zde je za pomoci Javascriptu doděláno omezení výběru času a data a předvýběr data po načtení stránky. [20].



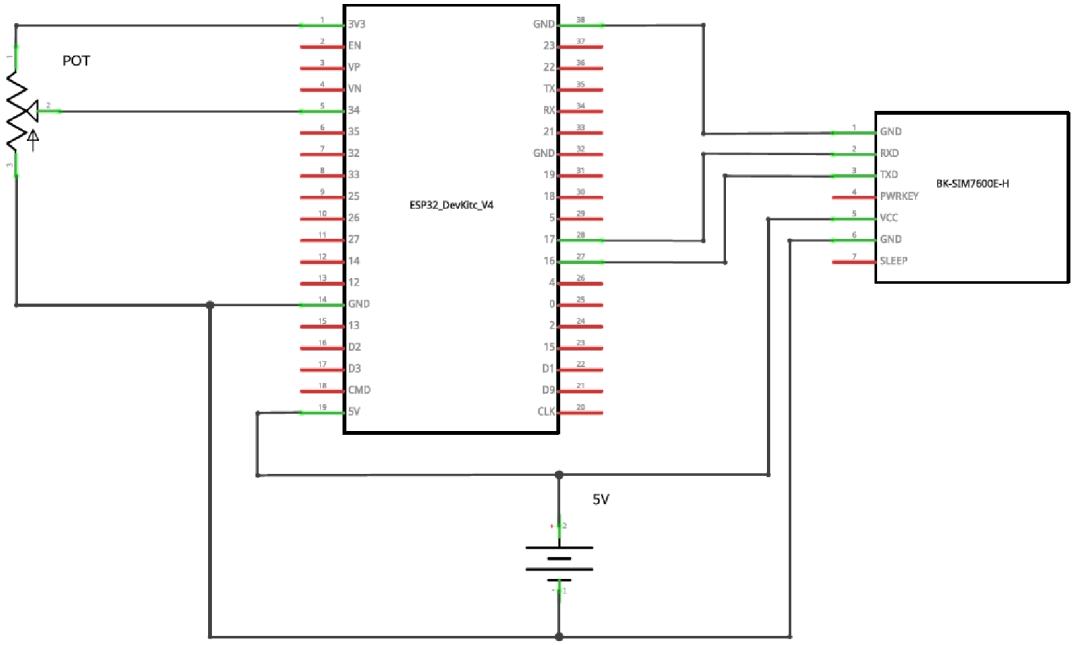
Obrázek 3.6: Snímek ze stránky historie

```
plugins: {
    zoom: {
        pan: {
            enabled: true,
            mode: 'x'
        },
        zoom: {
            enabled: true,
            mode: 'x',
        }
    }
}
```

3.2.3 Měřící stanice

Schéma zapojení

K sestavení měřící stanice bude využita deska *BK-7600*, řídícím členem bude mikrokontroler ESP32 DevKitC V4. Napájení zajišťuje napájecí modul pro nepájivé pole, který nabízí napájení pomocí 3,3 V nebo 5 V s maximálním proudem až 500 mA. Jako měřený obvod je zapojen potenciometr. Vše bude osazeno na nepájivém poli podle schéma na obr. 3.7.



Obrázek 3.7: Schéma zapojení měřící stanice

Naprogramování

Programování probíhalo v jazyce C v programu ARDUINO IDE, který nabízí podporu i pro platformy od firmy Espressif. Bylo využito několik knihoven totoho programu jako knihovna *Serial* nebo *AnalogRead*. Příklady příkazů pro sériovou komunikaci jsou uvedeny níže.

```
%spusteni seriove komunikace
Serial2.begin(115200, SERIAL_8N1, RXD2, TXD2);
%cteni seriove komunikace
content = Serial2.readStringUntil;
%poslani prikazu pro zmenu baudratu
Serial2.println("AT+IPR=3000000");
```

Jelikož při zvětšení frekvence ze 100 Hz na 500 Hz již mikrokontrolér nedosahoval potřebné rychlosti, bylo nutno využít i druhé jádro procesoru. Tudíž jedno jádro se stará o čtení hodnot z ADC převodníku a vytváří simulované hodnoty pomocí funkce *sin()* a druhé jádro se poté zabývá jen odesíláním zpráv. To bylo možné udělat pomocí systému FreeRTOS. [32] Nejdříve je nutné vytvořit proměnnou *TaskHandle_t Task1;*. Dále ve funkci setup, která se spouští na začátku programu, musí být definována tato funkce:

```
xTaskCreatePinnedToCore(
    codeForTask1, % funkce ulohy
    "Task_1",     % nazev ulohy
```

```
    10000,          % velikost ulohy
    NULL,           % parametr ulohy
    10,             % priorita ulohy
    &Task1,         % task handle
    0);             % jadro
```

Prioritu úlohy je vhodné zvolit větší z důvodů, že FreeRTOS potřebuje spouštět svoje vlastní úlohy pro správnou funkčnost. Jestliže se tyto úlohy za nějaký čas nespustí, mikrokontrolér vypíše chybovou hlášku a restartuje se. Jádro je zvoleno jako 0, protože funkce *void loop()* je spouštěna na jádře 1.
Následně už jen stačí vytvořit funkci *void codeForTask1(void * parameter)*, ve které bude vložen kód, podle kterého má daný úkol na druhém jádře pracovat.

Spotřeba

V této kapitole bude uveden naměřený proud, který dané zařízení odebíralo ze zdroje při daném úkolu. Měření bylo prováděno na laboratorním zdroji Twintex TP-1305 a uvedené hodnoty jsou spíše orientační. Napájecí napětí bylo vždy 5 V.

- Modul BK SIMCom7600
  V tabulce 3.4 jsou uvedené naměřené hodnoty odebíraného proudu ze zdroje.

Tabulka 3.4: Modul BK SIMCom7600

	I [mA]
Hledá síť	80-160
Připojování k síti	200-350
Idle – připojen k síti	40-50
Tcp – připojování	120-150
Připojen	40-50
Posílání (Počet hodnot – frekvence [Hz])	
5 - 100	220-240
29 – 100	220-240
5 – 500	220-240
2 - 1000	200-230

- Mikrokontroler ESP32 DevKitC v4
  V tabulce 3.5 jsou uvedené naměřené hodnoty odebíraného proudu ze zdroje.

Tabulka 3.5: Mikrokontroler ESP32 DevKitC v4

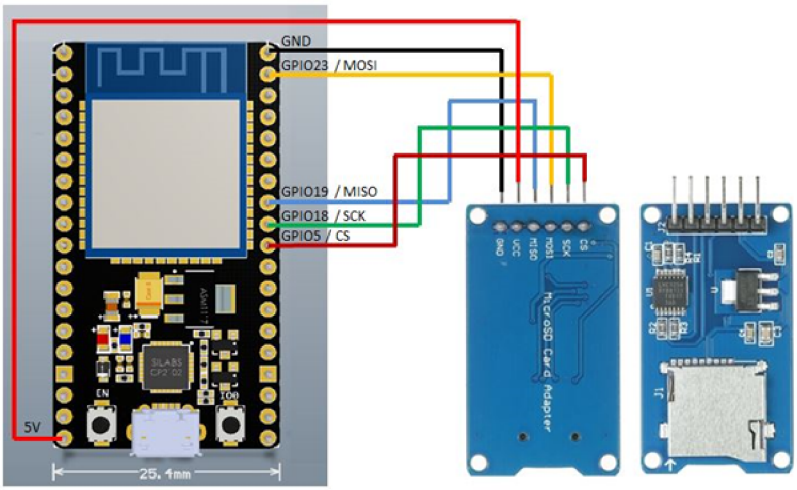
Posílání (Počet hodnot – frekvence [Hz])	I [mA]
5 -100	60
29 -100	70
5 -500	70
2 - 1000	70

3.3 Intervalové odesílání souboru dat – FTP

Jelikož se jednalo o vedlejší součást této práce, není řešení plně implementováno na hlavní webovou aplikaci a server.

3.3.1 Upravení zapojení měřící stanice

Z důvodu malé paměti mikrokontroleru ESP32 bylo nutné připojit externí úložiště, což bylo realizováno pomocí SD karty a její čtečky zapojené k mikrokontroléru podle následujícího schématu na obr. 3.8. Zapojení ostatních komponentů je uvedeno v obr. 3.7.



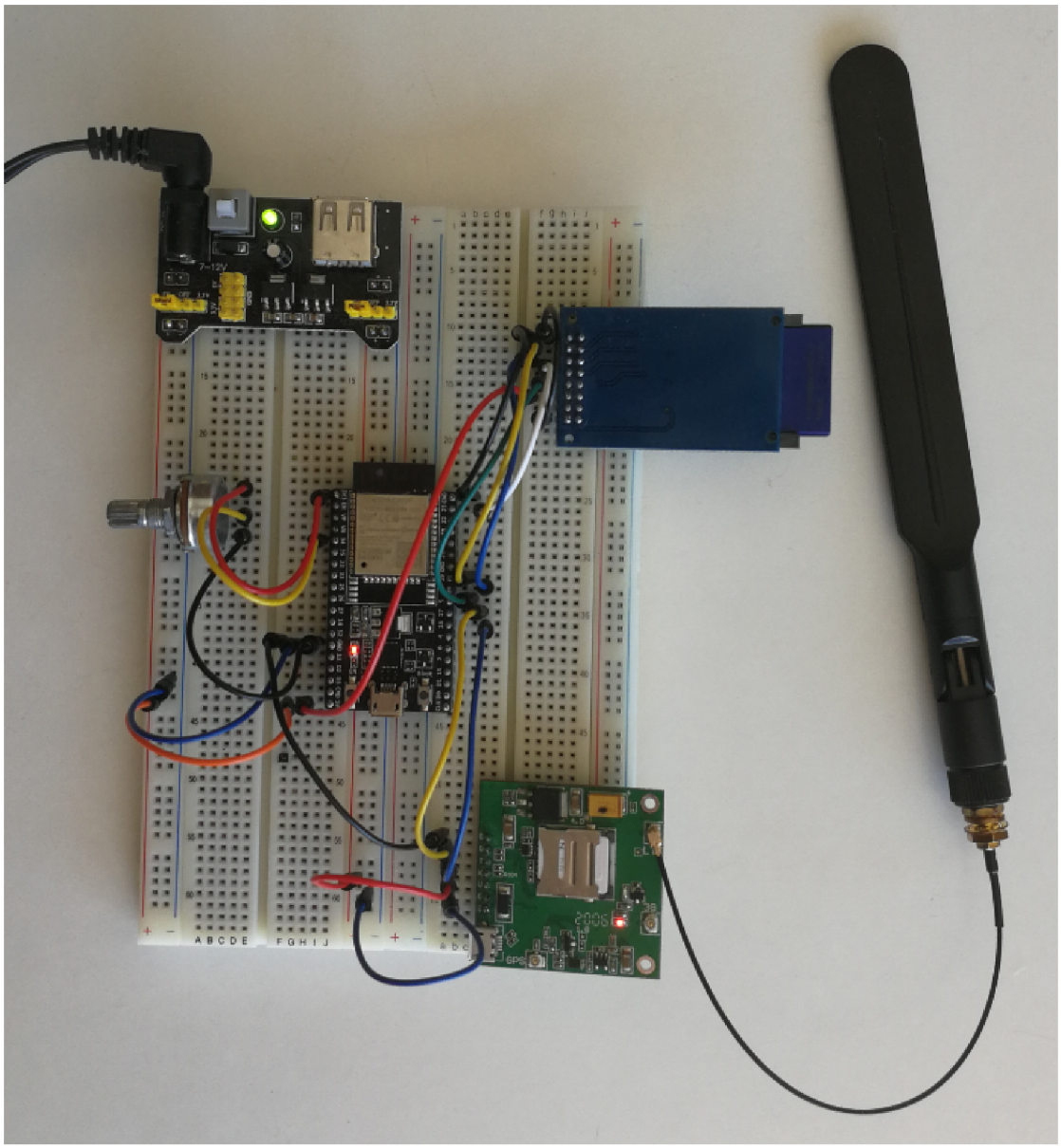
Obrázek 3.8: Schéma zapojení čtečky SD karet k měřící stanici [37]

3.3.2 Programování

Bylo opět využito systému FreeRTOS. Jedno jádro obstarává získávání hodnot a zápis do textového souboru na SD kartu a druhé se stará o přenos dat na FTP server. Pro práci se čtečkou SD karet byly využity knihovny FS.h, SD.h, SPI.h. Jako FTP server byl realizován stejným způsobem jako při testování přenosů v kapitole 3.3.
Po předem zvoleném čase, kdy se ukončí zápis do jednoho souboru, je nutné tento soubor poslat na server. Jelikož modul a mikrokontroler nemají přístup ke stejné SD kartě, je nutné přesunout soubor z SD karty do paměti modulu. To je vyřešeno pomocí následujících příkazů.[2].

```
AT+CFTRANRX="f:/<nazev souboru>",<velikost> %vlozeni souboru
% po znaku > vlozime data
AT+FSCD=F: % prechod do disku F
AT+FSDEL=<nazev> %odstraneni souboru
```

Na obr. 3.9 je vyfocena sestavená kompletní měřící stanice.



Obrázek 3.9: Měřící stanice rozšířená o čtečku SD karet

3.4 Zhodnocení a výsledky

Úkolem v praktické části bylo otestovat jednotlivé druhy protokolů pro přenos dat modulu SIMCom 7600E-H, dále pak sestavit měřící stanici se zobrazováním naměřených hodnot ve webové aplikaci.

Po otestování protokolů bylo rozhodnuto, že bude vhodné rozdělit aplikaci na dvě části, a to živý přenos dat a intervalové odesílání dat. Pro první část (živý přenos dat) byla klíčová nejnižší latence přenosu a největší přenos dat. Těchto vlastností dosahoval právě protokol TCP, a proto byl pro tutu aplikace zvolen. Pro druhou část (intervalové odesílání dat) bylo důležité rychlé odesílání velkých souborů dat. Toho dle výsledků z testování dosahoval protokol FTP, který je obecně pro přenosy velkých souborů nejlepší.

Po provedení nastavení serveru, webové aplikace a sestavení měřící stanice s mikrokotrolérem ESP32 probíhalo již testování v praxi. Podařilo se zajistit bezchybné odesílání až 29 měřených/simulovaných kanálů se vzorkovací frekvencí 100 Hz s periodou přenosu z modulu okolo 80 ms. Dále byla stanice a server otestovány pro odesílání 5 kanálů s frekvencí 500 Hz a 2 kanály s frekvencí 1000 Hz opět s periodou přenosu okolo 80 ms.
TCP server, po vložení mezipaměti do ramdisku, fungoval pro obě další konfigurace bez větších problémů. Ovšem při zobrazování hodnot na webové stránce se začaly vyskytovat problémy. Na některých klientských počítačích byl zaznamenán problém s plynulostí a nedostatkem výkonu pro vykreslování grafů pomocí zvolené knihovny Chart.js, ale pro frekvenci 100 Hz fungovala na všech testovaných počítačích Chart.js s pluginem pro livestreaming bez problémů.
Výsledky protokolu TCP jsou v praxi horší než při testování v kapitole 3.1, z důvodů nutnosti zápisu přijatých hodnot TCP serverem do databáze.

Ve druhé části byla využita měřící stanice z realizace živého přenosu rozšířená o čtečku SD karet. Bylo ověřeno, že měřící stanice dokáže naměřit, zapsat do souboru a odeslat tento soubor na FTP server. Testování bylo provedeno pro soubory do velikosti 20 MB, což zhruba odpovídá 2,5 hodinám zápisu 5 měřených/simulovaných kanálů o 100 Hz. Toto řešení ale nebylo z časových důvodů plně implementováno na hlavní webovou aplikaci a server. Předávání dat mezi mikrokontrolerem a modulem bylo vyřešeno kopírováním dat z SD karty do modulu a jejich následným odesláním. Pro další využití by bylo vhodné ještě prozkoumat, zda nelze připojit SD kartu zároveň k modulu a mikrokontroleru. Vzhledem k využití již hotových DPS nebylo možné takové připojení SD karet realizovat.
Rychlost celého procesu odesílání dat ve druhé části byla ovlivněna především kopírováním souboru z SD karty do modulu, což závisí na velikosti baudratu. Samotná rychlost přenosu dat při odesílání souboru z modulu na FTP server, by neměla být nijak ovlivněna.

4 Závěr

Cílem této práce bylo zjistit dostupné možnosti přenosu po síti 4G na modulu SIMCom 7600 E-H a následně vytvořit měřící stanici, ze které se budou data odesílat a zobrazovat ve webové aplikaci.

V rešerši jsou představeny druhy protokolů modulu SIMCom 7600E-H, přes které lze data odesílat pomocí 4G sítě. Součástí je také průzkum, jakým způsobem se dá realizovat měřící stanice a server s webovou stránkou.
Výstupem je přehled protokolů s popisem možností implementace. Dále na základě průzkumu bylo vyhodnoceno, že server s webovou aplikací poběží na frameworku Flask s hostingem u Linode. Knihovnou pro grafické zobrazení bude Chart.js a data se budou ukládat do relační databáze SQLite. Vývojovou platformou pro měřící stanici bude ESP32 DevKitC v4.

V praktické části byly nejprve všechny protokoly otestovány. Na základě poznatků z rešerše bylo rozhodnuto, že se aplikace rozdělí na dvě části, a to na živý přenos dat a intervalové odesílání dat. Pro první část byl vybrán protokol TCP z důvodů nejnižší latence přenosu a největšího přenosu dat. Pro druhou část (intervalové odesílání dat) byl zvolen protokol FTP, který dosahoval nejvyšší přenosové rychlosti pro velké soubory dat.

Dále byla sestavena měřící stanice s 4G modulem na desce BK-SIM7600 a čtečkou SD karet pro intervalový přenos dat. Řízena pomocí mikrokontroléru ESP32. K přijímání a zobrazování naměřených hodnot byla vytvořena webová aplikace pomocí jazyka HTML, CSS a javascriptu. Webová aplikace běžela na serveru Linode spolu s dvěma dalšími skripty: TCP serverem a skriptem na přepis dat z mezipaměti do permanentní SQLite databáze.

První částí aplikace bylo posílání dat v reálném čase. Podařilo se nám pomocí TCP protokolu zajistit přenos až 29 měřených/simulovaných kanálů při frekvenci 100 Hz, 5 kanálů při 500 Hz nebo 2 kanály při 1000 Hz. Všechny s periodou odesílání 80 ms z modulu na server. To byla oproti testování na začátku praktické části v kapitole 3.1 větší hodnota, ale byla způsobená časem nutným na zapsání přijatých dat do mezipaměti.
Při frekvenci 1000 Hz nastávaly u některých klientských počítačů problémy s nedostatkem výkonu při vykreslování grafů na stránce pomocí Chart.js, ale pro 100 Hz a 500 Hz fungovalo zobrazování bez větších problémů.

Druhá část aplikace nebyla z časových důvodů plně implementována na hlavní webovou aplikaci a server. Podařilo se ale realizovat přenos větších souborů (až 20 MB) pomocí protokolu FTP, což bylo zhruba 2,5 hodiny měření 5 kanálů se 100 Hz. Největším problémem u této aplikace bylo předávání dat mezi mikrokontrolérem a modulem z důvodu

využití již hotových platforem. Problém byl vyřešen kopírováním dat z SD karty do modulu, ale pro další využití by bylo vhodné prozkoumat jiné způsoby předávání dat.

Na práci by mohlo být dále navázáno rozšířením o správu většího počtu měřících stanic, stahování naměřených dat, vytvoření komunikace mezi webovou stránkou a měřící stanicí (např. odesílání požadavku na změnu vzorkovací frekvence z webové stránky na měřící stanici, poloha měřící stanice, atd.).

Literatura

[1] A7600E-H [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1290.html

[2] SIM7500_SIM7600 Series_ AT Command Manual [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1230.html

[3] BRZOBOHATÝ, Lukáš. Modul pro měření teploty a vlhkosti s rozhraním Ethernet [online]. 2012. Brno, [cit. 2021-5-20]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=118276

[4] SIM7500_SIM7600_SIM7800 Series_TCPIP_Application Note [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1211.html

[5] BRÁZDA, Libor. Instalace, konfigurace a testování serverových služeb typu DHCP, FTP, VPN, NAT a SNMP v prostředí IPv4 a IPv6 [online]. 2012. Brno, 2020 [cit. 2021-5-20]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=56911

[6] SIM7500_SIM7600 Series_FTP(S)_Application [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1220.html

[7] ROSENCRANCE, Linda, George LAWTON a Chuck MOOZAKIS. UDP (User Datagram Protocol) [online]. 2020. [cit. 2021-5-20]. Dostupné z: https://searchnetworking.techtarget.com/definition/UDP-User- Datagram-Protocol#:~:text=UDP%20(User%20Datagram%20Protocol)%20is,provided%20by%20the%20receiving%20party

[8] ZAPLETAL, Lukáš. Protokol HTTP 1.1 pod lupou [online]. 2001. [cit. 2021-5-20]. Dostupné z: https://www.root.cz/clanky/protokol-http-1-1-pod-lupou/

[9] SIM7500_SIM7600 Series HTTP(S)_Application Note [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1216.html

[10] MALÝ, Martin. Protokol MQTT: komunikační standard pro IoT [online]. 2016 [cit. 2021-5-20]. Dostupné z: https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/

[11] SIM7500_SIM7600_SIM7800 Series MQTT(S)_Application Note [online]. 2020 [cit. 2021-5-20]. Dostupné z: https://www.simcom.com/service-1214.html

[12] IoT ESP-WROOM-32 2.4GHz Dual-Mode WiFi+Bluetooth rev.1, CP2102 [online]. [cit. 2021-5-20]. Dostupné z: https://www.laskarduino.cz/iot-esp-32s-2-4ghz-dual-mode-wifi-bluetooth-rev-1–cp2102/

[13] Raspberry Pi Pico [online]. [cit. 2021-5-20]. Dostupné z: https://www.waveshare.com/wiki/Raspberry_Pi_Pico

[14] DOSTALOVÁ, Zuzana. Frontend vs. Backend [online]. 2014. [cit. 2021-5-20]. Dostupné z: https://www.czechitas.cz/cs/blog/zaciname-s-it/frontend-vs-backend

[15] What is a Web Framework? [online]. [cit. 2021-5-20]. Dostupné z: https://www.goodfirms.co/glossary/web-framework/

[16] Highcharts Documentation [online]. [cit. 2021-5-20]. Dostupné z: https://www.highcharts.com/docs/index

[17] Chart.js Documentation [online]. [cit. 2021-5-20]. Dostupné z: https://www.chartjs.org/docs/2.8.0/

[18] KUSANAGI, Akihiko a Alexandre GIRARD. Chartjs-plugin-streaming [online]. [cit. 2021-5-20]. Dostupné z: https://github.com/nagix/chartjs-plugin-streaming

[19] HTML Input Types [online]. [cit. 2021-5-20]. Dostupné z: https://www.w3schools.com/html/html_form_input_types.asp

[20] Flatpickr.js [online]. [cit. 2021-5-20]. Dostupné z: https://flatpickr.js.org/

[21] Flask vs Django: How to Understand Whether You Need a Hammer or a Toolbox [online]. 2019. [cit. 2021-5-20]. Dostupné z: https://steelkiwi.medium.com/flask-vs-django-how-to-understand-whether-you-need-a-hammer-or-a-toolbox-39b8b3a2e4a5

[22] DWYER, Gareth. Flask vs. Django: Why Flask Might Be Better [online]. 2017 [cit. 2021-5-20]. Dostupné z: https://www.codementor.io/@garethdwyer/flask-vs-django-why-flask-might-be-better-4xs7mdf8v

[23] TOMICZEK, Roman. Relační a NoSQL databáze pro sběr dat z IoT zařízení. Brno, 2018, 46 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. [online]. [cit. 2021-5-20]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=175480

[24] BERGA, Mariana a Tiago FRANCO. SQL VS NOSQL: WHEN TO USE? [online]. 2021. [cit. 2021-5-20]. Dostupné z: https://www.imaginarycloud.com/blog/sql-vs-nosql/

[25] CLARK, Jessica. Digitalocean vs Linode vs Heroku | Which is better? [online]. In: . [cit. 2021-5-20]. Dostupné z: https://blog.back4app.com/digitalocean-vs-linode-vs-heroku/

[26] Socket — Low-level networking interface [online]. [cit. 2021-5-20]. Dostupné z: https://docs.python.org/3/library/socket.html

[27] Threading — Thread-based parallelism [online]. [cit. 2021-5-20]. Dostupné z: https://docs.python.org/3/library/threading.html

[28] SQLite - Python [online]. [cit. 2021-5-20]. Dostupné z: https://www.tutorialspoint.com/sqlite/sqlite_python.htm

[29] Chartjs-plugin-zoom [online]. [cit. 2021-5-20]. Dostupné z: https://github.com/chartjs/chartjs-plugin-zoom

[30] GUOAN, Xiao. How to Easily Create RAM Disk on Debian, Ubuntu, Linux Mint, CentOS [online]. 2019. [cit. 2021-5-20]. Dostupné z: https://www.linuxbabe.com/command-line/create-ramdisk-linux

[31] A super-simple way to run scripts on boot [online]. [cit. 2021-5-20]. Dostupné z: https://learn.pimoroni.com/tutorial/sandyj/running-scripts-at-boot

[32] How to use ESP32 Dual Core with Arduino IDE [online]. [cit. 2021-5-20]. Dostupné z: https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/

[33] S2-107EQ-Z1W64 SIMCOM [online]. 2012. [cit. 2021-5-20]. Dostupné z: https://www.tme.eu/cz/details/sim7600e-h/moduly-m2m-gprs-hspa-lte/simcom/s2-107eq-z1w64/

[34] ESP32 DevKitC V4. [online]. 2012. [cit. 2021-5-20]. Dostupné z: https://sc04.alicdn.com/kf/HTB1CW9VefWG3KVjSZFgq6zTspXaj.jpg

[35] Breakout SIM7600 User Manual [online]. [cit. 2021-5-20]. Dostupné z: http://www.thebackshed.com/forum/uploads/Grogster/2018-07-24_122343_BK-SIM7600_Board_user_manual_V1.0.pdf

[36] SIM7600E-H 4G HAT for Raspberry Pi, LTE Cat-4 4G / 3G / 2G, GNSS, for Europe, Southeast Asia, West Asia, Africa [online]. [cit. 2021-5-20]. Dostupné z: https://www.waveshare.com/sim7600e-h-4g-hat.htm

[37] ESP32 - Working with SD card [online]. [cit. 2021-5-20]. Dostupné z: http://wei48221.blogspot.com/2018/11/esp32-working-with-sd-card.html

Seznam zkratek

TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
MQTT	Message Queuing Telemetry Transport
QoS	Quality of Service
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UART	Universal Asynchronous Receiver/Transmitter
M2M	Machine To Machine
ADC	Analog-to-Digital Converter
SQL	Structured Query Language

Seznam obrázků

Seznam příloh

Testovani	skripty k otestování přenosové rychlosti (MATLAB, Python)
Server	soubory k TCP serveru a webové stránce (Python, HTML, CSS)
Mikrokontroler	skripty pro mikrokontrolér ESP32 (Arduino)