

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

IoT projekt pre stredné školy II

Bakalárska práca

2023

Denys Lashuk

**Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky**

IoT projekt pre stredné školy II

Bakalárska práca

Študijný program: Informatika
Študijný odbor: 9.2.1. Informatika
Školiace pracovisko: Katedra počítačov a informatiky (KPI)
Školiteľ: Miroslav Biňas

Košice 2023

Denys Lashuk

Abstrakt v SJ

Cieľom tejto bakalárskej práce je vytvoriť vzdelávací projekt v oblasti internetu vecí (IoT), ktorý môže byť ďalej implementovaný do študijného kurzu. Cieľovou skupinou tohto projektu sú študenti stredných škôl, ktorí už majú základné znalosti programovania. Projekt však môže byť užitočný aj pre dospelé osoby, ktoré nemajú skúsenosti s IoT a chcú pochopiť jeho základy.

Počas vývoja tohto projektu sa študenti oboznámia so základmi internetu vecí a elektroniky.

Táto práca má praktické využitie a môže byť zaradená do školských učebných plánov ako doplnkový kurz. To umožní študentom prehľbiť si vedomosti z programovania a rozvíjať svoje schopnosti pri tvorbe IoT projektov.

Cieľom tejto bakalárskej práce je teda pomôcť mladej generácii programátorov pochopiť základy internetu vecí a sprostredkovať im praktické skúsenosti pri vývoji IoT projektov.

Klúčové slová v SJ

IoT, Internet vecí, kurz, projekt

Abstrakt v AJ

The aim of this bachelor thesis is to create an educational project in the field of Internet of Things (IoT) that can be further implemented in a course of study. The target audience of this project is high school students who already have a basic knowledge of programming. However, the project can also be useful for adults who have no experience with IoT and want to understand its basics.

During the development of this project, students will be introduced to the basics of IoT and electronics.

This work has practical use and could be included in school teaching plans as a supplementary course. It will allow students to deepen their knowledge of programming and develop their skills in creating IoT projects.

Therefore, the aim of this bachelor project is to help the young generation of programmers to understand the fundamentals of IoT and to provide them with practical experience in developing IoT projects.

Klúčové slová v AJ

IoT, Internet of Things, course, project

Bibliografická citácia

LASHUK, Denys. *IoT projekt pre stredné školy II*. Košice: Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky, 2023. 40s. Vedúci práce: Miroslav Biňas

TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
Katedra počítačov a informatiky

**ZADANIE
BAKALÁRSKEJ PRÁCE**

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

IoT projekt pre stredné školy II.

IoT Project for High Schools II.

Študent: **Denys Lashuk**
Školiteľ: **Ing. Miroslav Biňas, PhD.**
Školiace pracovisko: **Katedra počítačov a informatiky**
Konzultant práce:
Pracovisko konzultanta:

Pokyny na vypracovanie bakalárskej práce:

1. Preskúmať spôsoby vyučby Internetu vecí (IoT) na stredných školách.
2. Navrhnúť a predstaviť IoT projekt, jeho architektúru a jeho pridanú hodnotu, vhodný pre vyučbu problematiky na stredných školách.
3. Navrhnúť hardvér "vecí", ktoré budú súčasťou projektu s dôrazom na požiadavky pre tvorbu vecí v IoT.
4. Implementovať a experimentálne overiť navrhnuté riešenie.
5. Dokumentačnú časť práce vypracovať v typografickom jazyku LaTeX.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 26.05.2023
Dátum zadania bakalárskej práce: 31.10.2022



.....
prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty



Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 13.5.2023

.....

Vlastnoručný podpis

Podčakovanie

Na tomto mieste by som rád podčakoval svojmu vedúcemu práce Miroslavu Biňasu za jeho čas a odborné vedenie počas riešenia mojej práce.

Chcem podčakovať všetkým svojim blízkym a priateľom, ktorí ma podporovali. iež by som chcel podčakovať chatu GPT a prekladateľovi DeepL za pomoc pri písaní tejto práce.

Obsah

Úvod	1
1 Analytická časť	3
1.1 Čo je to Internet vecí	3
1.2 Moje skúsenosti s vyučovaním IoT a prácou s elektronikou	5
1.3 Analýza podobných riešení a cieľovej skupiny projektu.	6
1.3.1 Analýza riešení na Slovensku	6
1.3.2 Analýza medzinárodných riešení	6
1.3.3 Analýza znalostí študentov stredných škôl na Slovensku . .	7
1.4 Podrobne plánovanie projektu a jeho jednotlivých častí	8
1.4.1 Základné princípy	8
1.4.2 Moja vízia projektu a jeho plánovania	9
1.4.3 Výber témy pre projekt	10
1.4.4 Výber softvéru a hardvéru na školenie	12
1.4.5 Výber modelu mikrokontroléra	13
1.4.6 Dodatočné periférie pre Raspberry Pi Pico W	15
1.4.7 Výber MQTT brokera	15
1.4.8 Prečo NodeRed?	16
2 Syntetická časť	17
2.1 Počiatocný vývoj meteostanice	17
2.1.1 Prvé pokusy	17
2.1.2 Zásady vývoja	18
2.2 Hardvérová časť meteostanice	19
2.2.1 Schéma zapojenia	19
2.2.2 Príklady realizácie schémy zapojenia	21
2.3 Vývoj softvérovej časti meteostanice	21
2.4 Hlavná časť vývoja (Prvá iterácia)	23
2.4.1 Postup vývoja	23

2.4.2	Pomocná knižnica	25
2.4.3	Kritické problémy, s ktorými som sa stretol počas vývoja. . .	26
2.4.4	Výsledok vývoja a potreba refaktoringu.	26
2.5	Refaktoring (Druhá iterácia)	27
2.5.1	Konečný stavový automat v projekte	28
2.5.2	Výsledok refaktorovania kódu.	28
2.6	Testovanie (Tretia iterácia)	29
2.6.1	Ciele	29
2.6.2	Príprava	30
2.6.3	Úroveň znalostí študentov	31
2.6.4	Výsledky	31
2.6.5	Záver	31
2.7	Zhromažďovanie údajov na bráne a ich spracovanie	31
2.8	Analiza pridanej hodnoty projektu	33
2.8.1	Hlavná časť analýzy	33
2.8.2	Záver analýzy	34
3	Vyhodnotenie	35
3.1	Téma a architektúra projektu	35
3.2	Softvér meteostanice	35
3.3	Program na bráne	36
3.4	Testovanie	36
3.5	Zhrnutie	37
4	Záver	38
Literatúra		39
Slovník		41
Zoznam príloh		42
A	Štruktúra projektu	43
A.1	Súčasti meteostanice	43
A.1.1	Hardvér meteostanice	43
A.1.2	Softvér meteostanice	44
A.2	Súčasti IoT brány	45
A.2.1	Hardvér	45
A.2.2	Softvér	45

A.2.3 Alternatívny postup	45
-------------------------------------	----

Zoznam obrázkov

1.1	Vrstvy architektúry IoT riešení [3]	4
1.2	Koncepcia architektonického modelu projektu	10
1.3	Zoznam platforiem skupiny Arduino, ktoré podporujú Micropython [12]	13
1.4	Vihľad <i>Cytron Maker Pi Pico Base</i> [13]	15
2.1	Vzhľad breadboardu z oboch strán [15]	18
2.2	Rozloženie pinov na doske <i>Raspberry pi pico W</i> [16]	20
2.3	Chéma pripojenia k doske <i>Raspberry pi pico W</i>	21
2.4	Fyzický príklad schémy zapojenia 2.3 pomocou <i>Raspberry Pi Pico W</i>	22
2.5	Fyzický príklad schémy zapojenia 2.3 pomocou <i>Raspberry Pi Pico W</i> s doplnkovou doskou <i>Cytron Maker Pi Pico Base</i>	22
2.6	Fyzický príklad schémy zapojenia 2.3 pomocou <i>ESP32</i>	23
2.7	Práca meteostanice vo forme konečno stavobehu automatu.	28
2.8	Logika aplikácie na vizualizáciu dát vytvorených v <i>Node Red</i>	32
2.9	Vizualizácia aplikácie vytvorených na <i>Node Red</i>	32
A.1	Chéma pripojenia k doske <i>Raspberry pi pico W</i>	44
A.2	Vizualizácia aplikácie vytvorených na <i>Node Red</i>	46

Zoznam tabuliek

Úvod

V dnešnom svete, kde technológie sú centrom nášho každodenného života, je rozvoj internetu vecí čoraz dôležitejší. Internet vecí otvára veľké možnosti komunikácie medzi zariadeniami a predmetmi okolo nás a má veľký potenciál ovplyvňovať na rôzne oblasti nášho života vrátane školstva. Preto je podpora internetu vecí v školách mimoriadne dôležitá. Zamyslime sa nad tým, aký význam má implementácia internetu vecí do vyučovacieho procesu a aké sú jeho výhody pre študentov a učiteľov.

Implementácia IoT do školského vyučovania dáva študentom možnosť rozvíjať schopnosti, ktoré budú potrebovať v budúcnosti. prostredníctvom internetu vecí sa žiaci môžu naučiť pracovať so senzormi, sieťami, aktuátormi a inými zariadeniami, ktoré si vyžadujú programátorské schopnosti a pochopenie technológií. Tým sa rozvíja ich kreativita, schopnosť riešiť problémy a tímová práca.

Podpora internetu vecí v školách pomáha zvyšovať záujem študentov o vedu a techniku. Vďaka IoT môžu žiaci vidieť konkrétné príklady použitia technológií v reálnom svete. Môžu vytvárať projekty, ktoré si vyžadujú pochopenie fyziky, matematiky, programovania a ďalších predmetov. To im pomáha pochopiť prepojenie medzi teoretickým učením a praktickým využitím vedomostí.

Implementácia internetu vecí do školského vzdelávania môže výrazne zlepšiť učenie a rozvoj žiakov. Okrem toho je podpora internetu vecí v školách dôležitá aj pre rozvoj technologickej gramotnosti učiteľov. Učitelia musia byť dobre oboznámení s novými technológiami, aby ich mohli efektívne využívať v školských hodinách. Získanie zručností v oblasti internetu vecí umožní učiteľom používať inovatívne vyučovacie metódy, zapojiť žiakov do aktívnej účasti a vytvoriť dynamické vzdelávacie prostredie.

Implementácia internetu vecí do škôl má veľký význam. Pomáha rozvíjať zručnosti budúcnosti, zvyšovať záujem študentov o vedu a technológie, zlepšovať učenie a rozvoj žiakov a rozvíjať technologickú gramotnosť učiteľov. Školy by mali aktívne podporovať internet vecí a začleniť ho do vzdelávacieho procesu, aby pripravili mladú generáciu na život v modernom technologickom svete.

Formulácia úlohy

Mojím cieľom je vytvoriť projekt internetu vecí, ktorý by sa dal jednoducho implementovať do kurzu pre stredné školy na Slovensku, počas ktorého by sa študenti oboznámili so základmi internetu vecí.

Preto som si v rámci tejto bakalárskej práce stanovil cieľ:

1. Vytvoriť projekt internetu vecí, ktorý je zameraný na vzdelávací stupeň študentov stredných škôl a obsahuje aspoň základné funkcie štandardného produktu internetu vecí.
2. Zamyslieť sa nad tým, ako budú študenti tento projekt vyvíjať a vytvoriť kostru projektu, ktorá by tento vývoj uľahčila.
3. Otestujte svoj projekt na cieľovom publiku projektu a v prípade potreby vykonajte úpravy konečného riešenia.
4. Odhadnúť cenu projektu.
5. Vytvoriť kvalitnú dokumentáciu projektu, aby ho učiteľ mohol ľahko implementovať do svojho kurzu.

1 Analytická časť

1.1 Čo je to Internet vecí

Skôr ako začnete navrhovať a plánovať projekt internetu vecí, musíte najprv pochopiť definíciu internetu vecí a jeho základné princípy.

Takže, čo je to internet vecí? Ak sa spoľahneme na málo spoľahlivý zdroj, konkrétnie na Wikipédiu, nájdeme nasledujúcu definíciu:

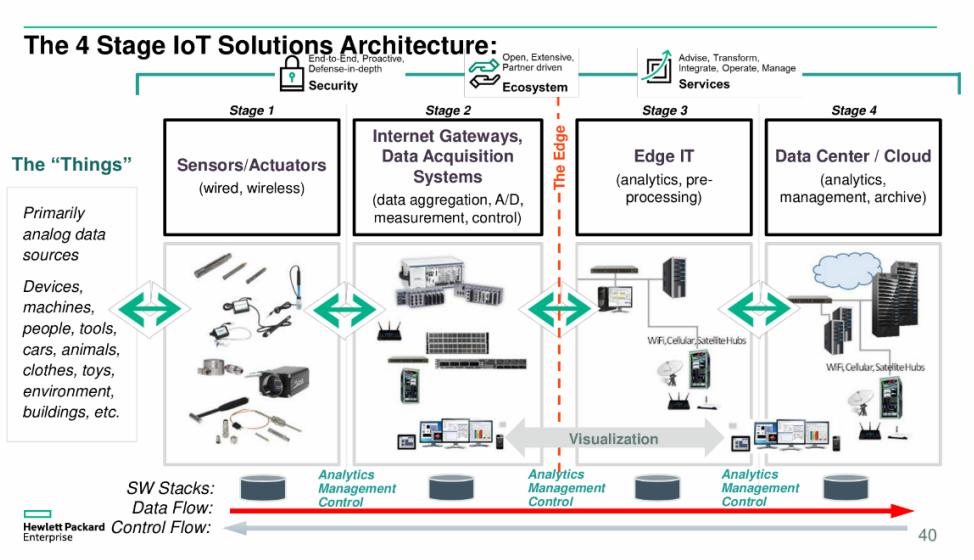
Internet vecí (IoT) označuje fyzické objekty (alebo skupiny takýchto objektov) so senzormi, spracovateľskými schopnosťami, softvérom a ďalšími technológiami, ktoré sa spájajú a vymieňajú si údaje s inými zariadeniami a systémami prostredníctvom internetu alebo iných komunikačných sietí.[1]

Môžem povedať, že táto definícia celkom dobre vystihuje základnú myšlienku tohto pojmu, ale myslím si, že jej chýba konkrétnosť. Čo sa týka akademickej definície internetu vecí, v tejto oblasti neexistuje vedecký konsenzus. Vo všeobecnosti si rôzni akademici definíciu mierne upravujú po svojom, ale ak to zhrnieme, najlepšia definícia by bola takáto:

An open and comprehensive network of intelligent objects that have the capacity to auto-organize, share information, data and resources, reacting and acting in face of situations and changes in the environment.[2]

Ďalším dôležitým aspektom porozumenia IoT je to, že je to sieť inteligentných zariadení, ktoré si vymieňajú dátu a táto sieť má svoju hierarchiu a logiku. To znamená, že jedno chytre zariadenie, aj keď má nejaký softvér a vykonáva konkrétnu úlohu (napríklad smartphone), nie je IoT, kým ho nepripojíte k sieti a nedáte mu konkrétnu úlohu v tejto sieti. Ako som už písal, sieť má svoju hierarchiu, ale pre štandardizáciu a koordináciu rôznych vývojov musí byť premyslená na základe nejakej architekturnej konceptuálnej modely. Napríklad v sieti Internet sú také konceptuálne modely, ako OSI a TCP/IP. V rámci IoT je tiež podobná architekturna model. Pokial' ide o vzhľad tejto modely, vedecká spoločnosť opäť nemá konsenzus. Na internete nájdete veľa rôznych architektúr s rôznymi počtami vrsťiev a názvami, ale všetky majú približne rovnaký význam. V ďalšej práci budem

používať architektonický model, ktorý sa skladá zo štyroch vrstiev.



Obr. 1.1: Vrstvy architektúry IoT riešení [3]

Zoznam jednotlivých vrstiev tohto modelu (obr. 1.1) je nasledový:

- **Prvá vrstva:** Táto vrstva obsahuje snímače údajov a rôzne elektrické pohony na interakciu s prostredím. Môžu komunikovať s vyššou vrstvou aj medzi sebou. Objekty na tejto vrstve nemusia byť *inteligentné* (napríklad DHT 11, motorček, mikrokontrolér).
- **Druhá vrstva:** Na tejto vrstve sa nachádzajú brány prepojenia systému. Teda zariadenia, ktoré prijímajú údaje z predchádzajúcej vrstvy, poskytujú príkazy pohonov, vykonávajú primárne spracovanie a filtrovanie údajov a majú softvér. Táto úroveň sa zvyčajne fyzicky nachádza veľmi blízko k prvej a je jednou z najdôležitejších úrovní v celej sústave. Na tejto vrstve môže byť aj vizualizácia zozbieraných údajov.
- **Tretia vrstva:** Táto vrstva slúži na zber a spracovanie údajov z predchádzajúcej vrstvy a ich pripravenie pre archiváciu alebo pre konečného spotrebiteľa. Zvyčajne táto úroveň slúži na zmiernenie zaťaženia hlavných serverov a na rýchlejšiu odpoveď používateľovi, ale môžu byť výnimky. Tretia vrstva nie je povinná pre použitie v IoT architektúre.
- **Štvrtá vrstva:** Je konečnou vrstvou, na ktorej sa informácie analyzujú, archivujú alebo sa dostanú ku konečnému používateľovi používanie aplikácií.

Vychádzajúc z celého vyššie uvedeného, možno povedať, že koncept IoT a jeho architektúra riešia otázky zberu, spracovania a transportovania dát. Preto pri na-

vrhovaní IoT riešení treba sústrediť pozornosť práve na správne manipulovanie s dátami.

V svojej prace sa budem snažiť dôsledne predstaviť študentom podstatu teoretického konceptu IoT a detailne ukázať všetky úrovne architektúry IoT riešení na príkladoch z praxe.

1.2 Moje skúsenosti s vyučovaním IoT a prácou s elektronikou

V tejto práci sa budem často rozhodovať na základe svojich skúseností, preto si myslím, že pred začatím samotnej analýzy a plánovania je potrebné sa podrobnejšie pozrieť na moje skúsenosti so vzdelávaním v oblasti IoT. Moje zoznámenie s IoT sa stalo až nedávno, v treťom ročníku môjho štúdia na univerzite v rámci predmetu IoT[4]. Materiál tohto predmetu umožňuje rozšíriť znalosti v oblasti IoT, zoznámiť sa s jej rôznymi aspektmi, aby sme na konci semestra mohli navrhnuť a vytvoriť produkt na základe tejto koncepcie. Cieľom tohto kurzu je poskytnúť základné znalosti na úrovni junior developera v tejto oblasti. Celkovo som s týmto kurzom spokojný, ale chcem zdôrazniť jeho hlavnú nevýhodu, a to nevyváženosť praktických cvičení. Podľa môjho názoru boli praktické cvičenia viac zamerané na interakciu so systémom *Docker* a menej na prácu s hardvérom. Okrem toho bolo učenie sa práce s hardvérom umiestnené až na konci a to tiež nie je podľa mňa najlepšia myšlienka.

Považujem, že základom IoT je interakcia s mikrokontrolérmi a mikroprocesormi, preto sa budem snažiť venovať väčšinu praktických úloh práve tejto interakcii. Je možné poznámenať, že IoT je koncepcia, ktorá sa stavia na komunikácii, nie na hardvéri.

Chcel by som spomenúť zaujímavú skúsenosť z účasti na workshopu zameranom na zoznámenie sa s *ESP32*. Táto skúsenosť bola veľmi zaujímavá, pretože štruktúra prezentácie na tomto podujatí je tou, ktorú chcem zrealizovať aj vo svojom projekte. Teda, samotný webinár bol postavený takým spôsobom, že po krátkej teoretickej časti nasledovala praktická časť, čo umožňovalo udržiavať pozornosť poslucháčov neustále v strehu. V takomto vzdelávacom prístupe môžem identifikovať iba jednu nevýhodu - problém zaostávajúceho účastníka. Ak študent nestíha doháňať ostatných aspoň na jednej úrovni, bude pre neho veľmi ťažké udržiavať krok s ostatným materiálom.

1.3 Analýza podobných riešení a cieľovej skupiny projektu.

1.3.1 Analýza riešení na Slovensku

Pri hľadaní podobných projektov na Slovensku som našiel len veľmi málo referencií. Preto som dospel k záveru, že internet vecí sa na Slovensku veľmi nerozvíja. Jediný skutočne podobný kurz sa organizoval na *Strednej odbornej škole elektrotechnický v Žiline*[5]. Materiály celého kurzu nie sú bohužiaľ *open source*, preto som analyzoval dostupné informácie z ich webovej stránky a reportáže. Po ich pre-skúmaní som si všimol, že to je presne to, čo sme potrebovali. Ak som pochopil z dostupných útržkov informácií, tento týždňový kurz bol rozdelený do dvoch skupín: *Robotika a . IoT*

Počas kurzu študenti pracovali s hardvérom, inštalovali potrebný softvér pre prácu a, samozrejme, programovali, a to je presne to, čo chcem implementovať do svojej práce.

Moje tvrdenie vychádza z faktu, že všetky komponenty sa dajú kúpiť s prisájkovanými pinmi a takéto komponenty sa cenovo minimálne líšia od bežných riešení.

Veľkou nevýhodou tohto kurzu je, že všetky jeho materiály nie sú zdokumentované a zverejnené na internete. To neumožní opäťovné vytvorenie tohto kurzu alebo jeho samostatné zvládnutie. Preto si myslím, že by bolo správne sprístupniť môj projekt čo najväčšiemu počtu ľudí, čiže umiestniť všetky študijné materiály na internet.

1.3.2 Analýza medzinárodných riešení

Naozaj prekvapujúco som na internete našiel mnoho rôznych kurzov IoT v angličtine. Existujú rôzne typy kurzov, platené a bezplatné, ale väčšina z nich je zamieraná na dospelú populáciu. Môžem uviesť niekoľko zaujímavých riešení pre školy.

Prvým je výučbový kurz od spoločnosti *Cisco*¹ s názvom *Introduction to IoT* [6]. Je zaujímavý tým, že prebieha úplne online a môže ho absolvovať hodikto bez učiteľa. Avšak, tento kurz má dosť problémov, napríklad, že sa v ňom nachádza prevažne len teoretická časť. Z praktických cvičení sú tam len testy na overenie znalostí a ich odpovede možno ľahko nájsť na internete. Preto sa mi zdá, že kurz

¹<https://www.cisco.com/>

nie je kompletný.

Spoločnosť *Cisco* sa snažila tento problém riešiť spoluprácou s univerzitami. Príkladom takej spolupráce je kurz s názvom *IoT Step by Step 2021*[7]. V tomto študijnom programe sa kurz od *Cisco* používa iba ako teoretický základ, po jeho dokončení deti prechádzajú na praktickú časť. To je príklad dosť dobreho riešenia tohto problému.

Ďalšou prácou, na ktorú by som chcel upozorniť, je kurz *Internet of Things Education Package*[8] od spoločnosti *Software AG*. V tomto kurze je jedno originálne riešenie, o ktorom by som chcel napísť, a to o možnosti použitia svojho smartfónu ako stanice zberu dát pre svoj IoT projekt. Toto je jednoduché a geniálne riešenie zároveň. Študentom nie je potrebné poskytovať ďalšie zariadenie na zber dát a nemusia navrhovať a testovať svoje riešenie, pretože v súčasnosti každý má smartfón, ktorý má všetky potrebné základné senzory. Preto takéto riešenie veľmi šetrí čas, vzdelávanie a peniaze. Pri vývoji svojho kurzu zväžim možnosť implementácie takéhoto riešenia.

Zhrnutím môžem povedať, že väčšina bezplatných, medzinárodných produktov, ktoré som našiel, sú dosť zaujímavé na štúdium. Žiaľ, nemohol som vidieť, ako tieto kurzy prebiehali, a nenašiel som ani recenzie používateľov, ale myslím, že som bol schopný pochopiť architektúru školenia a informácie, ktoré tieto kurzy poskytujú. Hoci sa tieto výučbové kurzy líšia od toho, čo chcem urobiť, zohľadním informácie, ktoré som získal v rámci prípravy tejto analízy.

1.3.3 Analýza znalostí študentov stredných škôl na Slovensku

Pred začatím vývoja kurzu je potrebné uvedomiť si úroveň znalostí, ktoré majú študenti. V našom prípade sú to študenti stredných škôl na Slovensku. Keďže som nedosiahol stredné vzdelanie na slovenských školách, priamo nebudem schopný posúdiť znalosti, ktoré tieto školy poskytujú. Preto som sa opýtal niekoľko svojich známych a priateľov, ktorí sa učili na slovenských školách.

Zaujímali ma tieto otázky:

- Rozsah tém, ktoré študenti stredných škôl preberajú v predmete informatika.
- Kvalita znalostí, ktoré získavajú študenti.
- Možné špecifikácie vzdelávania.

Na základe tohto prieskumu môžem povedať, že cieľová skupina môjho kurzu by už mala vedieť:

- Základy programovania.
- Základy práce s doskou Arduino Uno.

S ohľadom na vyššie uvedené parametre možno dospieť k záveru, že táto založená báza umožní neplýtvanie časom na vysvetľovanie elementárnych vecí z oblasti programovania/elektroniky a umožní sa sústrediť na zložitejšie veci. Chcem poznamenať, že nemôžem presne vedieť úroveň znalostí každého potenciálneho študenta, preto budem vychádzať práve z vyššie uvedených zaverov.

1.4 Podrobné plánovanie projektu a jeho jednotlivých častí

1.4.1 Základné princípy

Pri vývoji tohto projektu sa chcem držať týchto princípov:

1. *Dostupnosť* - hlavný princíp. Pretože práve ona umožní dostať informácie k väčšiemu množstvu ľudí, keďže popularita je hlavnou hodnotou toho, čo robíme v ére internetu. Dostupnosť zahŕňa nasledujúce aspekty:
 - *Open source* - Všetka vykonaná práca a všetky údaje musia byť voľne dostupné. To umožní akémukoľvek záujemcovi reprodukovať alebo doplniť moju prácu. Projekt musí mať dobrú dokumentáciu.
 - *Cenovo dostupné* - Všetky zariadenia musia byť maximálne dostupné a lacné. V ideálnom prípade, ak by všetky potrebné zariadenia pre kurz boli súčasťou štandardného zariadenia *Arduino Kit*, pretože už je k dispozícii takmer vo všetkých stredných školách na Slovensku. Ale je dôležité pochopiť, že kvalita kurzu je prioritnejšia ako nízka cena, takže ak je to potrebné, budem vyberať drahšie zariadenia.
2. *Kvalita informácií* - Kurz musí obsahovať len to, čo je skutočne potrebné a musí byť maximálne užitočný pre svoju cieľovú skupinu.
3. *Atraktivita* - Študent nesmie byť nudný počas výučby. Tento princíp je pomerne subjektívny, takže neviem, či ho dokážem uplnie splniť.

Verim vieru, že dodržaním týchto zásad môžeme vytvoriť dobrý produkt, ktorý sa môže ďalej zdokonaľovať. Preto sa môj ďalší postup môže odkazovať práve na tieto body.

1.4.2 Moja vízia projektu a jeho plánovania

Z kontextu úlohy a počas prvých konzultácií s veducim tejto bakalárskej práce, som pochopil, že konečným cieľom tohto projektu je vytvorenie IoT produktu študentami a počas tejto vývojovej práce sa študenti naučia základy IoT.

Mojimi hlavnými úlohami ako vývojára tohto projektu sú:

- Vymyslieť a vyrobiť konkrétny IoT produkt.
- Rozdeliť vývoj tohto produktu na niekoľko častí, ktoré budú základom pre lekcie.

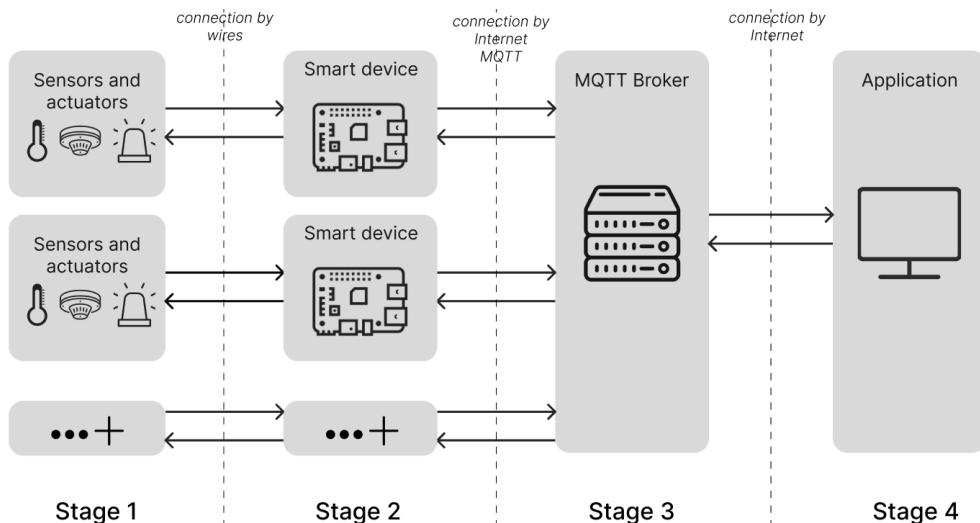
V ďalšej časti podrobnejšie opíšem oba kroky tejto práce.

Počas dlhých rozhovorov o tom, ako by mal projekt vyzerať, som dospel k záveru, že produkt, ktorý budú deti využívať, by mal byť zaujímavý a názorný v používaní. To znamená, že študenti by mali byť schopní ľahko overiť fungovanie svojho výrobku v triede alebo doma.

Avšak je potrebné ešte zvážiť, že projekt by mal byť celkom dostupný, pretože školy si zvyčajne nemôžu dovoliť veľké náklady na jeden projekt. Ideálne by bolo, ak by všetky potrebné súčiastky boli súčasťou základného *Arduino Kitu*, pretože sú už implementované vo väčšine slovenských škôl.

Pokiaľ ide o samotnú štruktúru projektu, myslím si, že by nemala byť príliš zložitá, aby ju študenti mohli bez problémov realizovať, ale nemala by byť ani príliš jednoduchá. Po dlhšom premýšľaní som vypracoval architektúru projektu a rozdelil som ju do 4 fáz.

- Prvá fáza: Tu sú všetky snímače a akčné členy, ktoré sú súčasťou projektu.
- Druhá fáza: Tu sa nachádza mikrokontrolér alebo mikroprocesor, ktorý číta údaje zo snímačov na prvej úrovni a vykonáva primárne spracovanie údajov a prípadne posielá príkazy aktuátorom na prvej úrovni. Objekty na prvej a druhej úrovni v rámci tohto istého zariadenia sú fyzicky v tesnej blízkosti a prenášajú údaje prostredníctvom vodičov.
- Tretia fáza: Toto je server/MQTT broker, ktorý sa nachádza v rámci triedy (napríklad počítač učiteľa). Prijíma údaje zo všetkých zariadení na prvej úrovni a spracováva ich. Komunikácia medzi druhou a treťou úrovňou prebieha prostredníctvom internetu pomocou protokolu MQTT.
- Štvrtý stupeň: Tu sa nachádza webová aplikácia, v ktorej môžu študenti zoobraziť údaje zozbierané z prvej úrovne.



Obr. 1.2: Koncepcia architektonického modelu projektu

Na obrázku 1.2 si môžete podrobnejšie pozrieť môj koncept. Myslím si, že takéto pomerne štandardné a jednoduché architektonické riešenie bude pre študentov najvhodnejšie na učenie.

1.4.3 Výber témy pre projekt

Po vytvorení konceptu som začal premýšľať o konkrétnej téme projektu, ktorá by zodpovedala schéme na obrázku 2. Po dlhom premýšľaní a konzultácii s učiteľom som prišiel s nápadom vytvoriť systém na riadenie mikroklímy skleníkov.

Podstatou tohto systému je zber údajov o teplote a vlhkosti v skleníkoch, analýza zozbieraných údajov a v prípade, že teplota alebo vlhkosť je mimo normy, potom sa mikroklíma skleníka riadi pomocou aktuátorov, ktoré uvedú ukazovatele do normálu.

Systém sa skladá z nasledujúcich komponentov:

- Inteligentné zariadenie, ktoré je umiestnené v skleníku. Jeho úlohou je zber údajov o mikroklíme pomocou senzorov a ich analýza. Ak sú zozbierané údaje mimo normy, zariadenie aktivuje akčne členy, ktoré vyrovňávajú klímu. Všetky zozbierané údaje zo snímačov sa odosielajú na server.
- MQTT broker, ktorý prijíma údaje, ich prenáša do webovej aplikácie.
- Webová aplikácia, ktorá umožňuje regulovať mikroklímu v každom skleníku a zobrazovať aktuálnu teplotu a vlhkosť.

Na prvý pohľad je táto myšlienka dobrá a dobre zapadá do koncepcie architektonického modelu (obr. 1.2), ale zdá sa to tak len na prvý pohľad. Problémy sa

začínajú vo fáze rozšírenia implementácie projektu v školách. Teoreticky tento projekt obsahuje nielen senzory, ale aj akčné členy na zmenu teploty a vlhkosti.

So senzormi by nemali byť žiadne problémy, potrebujeme len senzory svetla a vlhkosti (LDR a DHT11) a tie sú už dostupné v štandardných súpravách *Arduino Kit*. Ale s akčními členami je to úplne iná záležitosť. Pri analýze existujúcich riešení podobných nášmu projektu [9] [10] [11] som si uvedomil, že na realizáciu tohto projektu by som potreboval nasledujúce pohony:

1. Elektrický motorček (minimálne 12V) na zdvíhanie a spúšťanie okienka.
2. Počítačový ventilátor na vetranie skleníka.
3. Elektrické vodné čerpadlo na zvlhčovanie pôdy skleníka.

Tieto komponenty už nie sú súčasťou súpravy *Arduino Kit*, takže školy si budú musieť tieto nástroje zakúpiť dodatočne, čo pre nás nie je veľmi výhodné. Koniec koncov, jednou zo zásad, ktoré chcem pri tvorbe projektu dodržiavať, je minimalizovať dodatočné finančné náklady na kurz.

Ďalším problémom pri vývoji tohto systému je náročnosť jeho implementácie v školách. Predstavme si, že 20 študentov absolvuje kurz a implementovalo systém podľa návodu. A teraz, aby ste tento systém otestovali, musíte sa veľmi snažiť, pretože školy zvyčajne nemajú niekoľko skleníkov. Samozrejme, môžete sa pokúsiť otestovať systém v rámci jednej alebo viacerých tried, ale podľa môjho názoru je to dosť náročná úloha.

Vzhľadom na všetky uvedené nevýhody som si uvedomil, že myšlienka vyvinúť systém riadenia klímy v skleníku IoT nie je relevantná. Myslel som si však, že základ koncepcie je celkom zaujímavý, a tak som na základe predchádzajúcej myšlienky vytvoril novú, ktorá vyriešila všetky predchádzajúce problémy.

Po niekoľkých ďalších konzultáciách a premýšľaní o možných riešeniach som prišiel s vhodnou myšlienkovou bez zjavných nevýhod. Podstatou nového konceptu je vyvinúť systém zberu dát o počasí.

Systém pozostáva z niekoľkých meteorologických staníc navrhnutých študentmi, servera, ktorý zbiera údaje z týchto meteorologických staníc, a webovej aplikácie, z ktorej si môžete pozrieť všetky zozbierané údaje v grafoch, tabuľkách atď. Meteorologická stanica meria teplotu, vlhkosť a svetlo.

Ako vidíte, tento systém je zjednodušenou verziou predchádzajúceho systému. Ako som už napísal vyššie, od predchádzajúcej myšlienky som sa nevzdialil a jednoducho som odstránil komponenty, ktoré s ňou kolidovali, podľa logiky žiadne komponenty, žiadne problémy. Pri analýze predchádzajúceho riešenia som si uvedomil, že hlavným problémom boli pohony, pretože tie projekt

úmerne komplikovali. Preto nové riešenie už nemá žiadne pohony, má len najjednoduchšie snímače, ktoré nie je potrebné dokupovať do škôl.

Napriek tomu, že nové riešenie je trochu zjednodušené, nemyslím si, že by to mohlo nejako ovplyvniť kvalitu vedomostí poskytovaných žiakom. Práve naopak, zjednodušenie hardvéru projektu nám umožní venovať viac pozornosti programovaniu a spracovaniu zozbieraných údajov. Podľa môjho názoru práve toto bude pre mladých programátorov v oblasti internetu vecí užitočnejšie. Kvôli prehľadnosti na záver zhrňme tému projektu.

Téma projektu: Systém na zber údajov o počasí. Architektúra projektu podľa mojej koncepcie (obr. 1.2):

- Prvá fáza: Senzory tepla, vlhkosti a svetla (DHT11/22, LDR)
- Druhá fáza: Inteligentné zariadenie (ESP32/Raspberry pi)
- Tretia fáza: Server, ktorý slúži ako MQTT broker
- Štvrtá fáza: Webová aplikácia, ktorá zobrazuje všetky zozbierané údaje.

Chcem poznamenať, že toto je len teoretický koncept, konečný výsledok sa môže počas vývoja dopĺňať a meniť.

1.4.4 Výber softvéru a hardvéru na školenie

Pri čítaní predchádzajúcej časti ste si možno všimli, že použitie takých dosiek ako *ESP32* alebo *Raspberry Pi 3/Pico* v projekte môže byť nevhodné a lepším riešením by bolo nahradiť ich *Arduino Uno*. Koniec koncov, *Arduino Uno* je už v školách dostupné, takže nemusíte kupovať nič iné a nebudete musieť študentom vysvetľovať základy používania tohto zariadenia, pretože s ním už pracovali.

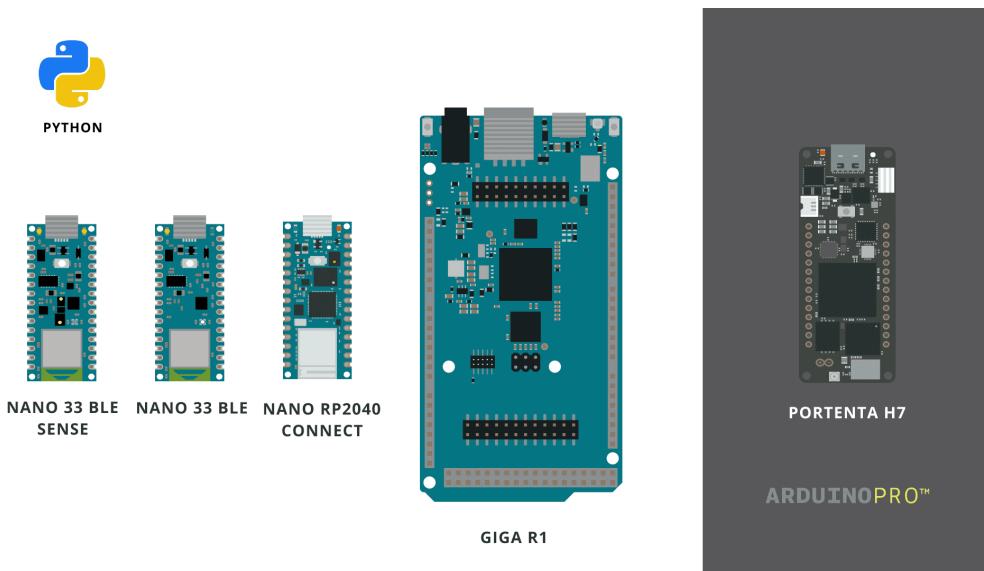
A ak vezmeme do úvahy len vyššie uvedené faktory, *Arduino* vyzerá naozaj výhodnejšie v porovnaní s mikrokontrolérmi ako *ESP32* alebo *Raspberry Pi 3/Pico*. Aby som bol úprimný, na začiatku tejto práce som bol tiež fanúšikom používania *Arduina*. V tejto časti však dokážem, prečo bude výber dosky *ESP32* albo *Raspberry Pi 3/Pico* v tejto práci vhodnejší, ako rodina dosiek *Arduino*.

Prvá vec, ktorú je potrebné si uvedomiť, je, že vo väčšine štandardných mikrokontrolérov (napr. *ESP32*, *Raspberry Pi 4*, *Raspberry Pi Pico*) už má zabudovaný WiFi modul, zatiaľ čo bežné *Arduina* ho nemajú a preto ich bude treba kúpiť samostatne. Na internete som našiel len niekoľko modelov *Arduina* s integrovaným WiFi modulom, ako napríklad *ARDUINO UNO WiFi REV2*² a *Arduino Nano*

²<https://store.arduino.cc/products/arduino-uno-wifi-rev2>

*RP2040*³. Avšak v súčasnosti sú tieto zariadenia buď rovnako drahé ako napríklad *ESP32*, alebo niekedy aj drahšie.

Ďalším faktom v prospech doskam tipu *ESP32* je základný programovací jazyk. V prípade rodiny *Arduino* ide zvyčajne o jazyk *C++*, zatiaľ čo u mikroprocesorov je to *Micropython* (takže ten istý *Python*, len s menšou funkcionálitou). Podľa mňa je to dosť dôležitý parameter pre ľahkosť vývoja. Každý, kto už niekedy programoval v jazyku *Micropython*, môže potvrdiť, že programovanie v porovnaní s *C++* je oveľa jednoduchšie. Väčšina stredoškolákov už dobre ovláda *Python*, takže programovanie v *Micropython* by nemalo byť komplikované. Bohužiaľ, v súčasnosti nie sú všetky dosky rodiny *Arduino* podporované *Micropythonom*⁴, existujú len niektoré samostatné platformy, ktoré to môžu. Tieto zariadenia môžete vidieť na obrázku 1.3. Ako vidíte, v tomto zozname nie je najrozšírenejší *Arduino Uno*.



Obr. 1.3: Zoznam platforiem skupiny *Arduino*, ktoré podporujú *Micropython* [12]

Chcel by som ešte dodať, že dosky tipu *ESP32* môžu vykonávať širší rozsah úloh než mikrokontroléry, preto výber mikroprocesora ako hlavného zariadenia pre programovanie umožní vyniesť zručnosti študentov na novú úroveň a v budúcnosti budú schopní vykonávať zložitejšie úlohy.

1.4.5 Výber modelu mikrokontroléra

Vzhľadom na špecifická projektových podmienok som hodnotil model mikrokontroléra podľa nasledujúcich kritérií:

³<https://store.arduino.cc/products/arduino-nano-rp2040-connect>

⁴<https://docs.arduino.cc/learn/programming/arduino-and-python>

- Jednoduchosť použitia
- Nízka cena
- Dostupnosť

Ďalším dôležitým parametrom, ktorý by mal byť jednoznačne zahrnutý v doske, je zabudovaný Wi-Fi modul. Zariadenia bez tohto parametru nebudú zvažované. Neberiem do úvahy samostatné čipy, ktoré nie sú pripojené k doske, pretože nie sú absolútne praktické pre prácu s žiakmi stredných škôl.

Ďalej podrobnejšie zanalyzujem jednotlivé kritériá a vyberiem najlepší motel dosky s mikroprocesorom. Začneme s dostupnosťou. Dostupným mikroprocesorom považujem ten, ktorý je ľahko dostupný na globálnom trhu. To znamená model, ktorý nie je deficitný alebo vzácny a je možné ho bez prekážok kúpiť na populárnych internetových obchodoch, ako sú *alza.sk* alebo *aliexpress.com*. Podľa tohto parametra môžem vyzdvihnúť nasledujúce dosky: *ESP32*, *Raspberry Pi 3/4*, *Raspberry Pi Pico W*, *ESP8266*. Na základe svojich skúseností môžem povedať, že najdostupnejšou a najpopulárnejšou voľbou je *ESP32*.

Nasledujúcim kritériom je jednoduchosť použitia. Považujem za logické, ak dáme študentam zariadenie, ktoré bude najviac podobné tomu, čo už poznajú (v tomto prípade *Arduino Uno*). Najvhodnejšie na tomto poli vyzerá *Raspberry Pi Pico W*, pretože z môjej skúsenosti môžem povedať, že toto zariadenie má najjednoduchšiu prvotnú konfiguráciu zo všetkých modelov, ktoré poznám. Okrem toho, komunikácia s týmto zariadením prebieha cez sériový port, rovnako ako u *Arduino Uno*. Podobné charakteristiky v tejto oblasti má aj *ESP32*, ale treba si uvedomiť, že pre tejto model existuje veľa rôznych konfigurácií, a môj projekt potrebuje štandardizované dosky.

Nakoniec sme sa pozreli na otázku ceny. V súčasnosti najlacnejšími modelmi, ktoré som skúmal vysšie, sú *ESP32* (10,81 €)⁵, *ESP8266* (7,59 €)⁶, *Raspberry Pi Pico W* (7,59 €)⁷. Všetky ceny som bral z online obchodu "*rpihop.cz*"⁸, pretože tam sú, podľa môjho subjektívneho názoru, dostatočne primerané ceny.

Po zhodnotení všetkých týchto parametrov som dospel k záveru, že *Raspberry Pi Pico W* je najoptimálnejšou voľbou pre túto prácu.

⁵<https://rpihop.cz/esp32-a-esp8266/3884-dfrobot-firebeetle-2-esp32-e-iot-mikrokontroler-s-podporou-wi-fi-bluetooth.html>

⁶<https://rpihop.cz/esp32-a-esp8266/1944-nodemcu-esp8266-wifi-vyvojova-deska.html>

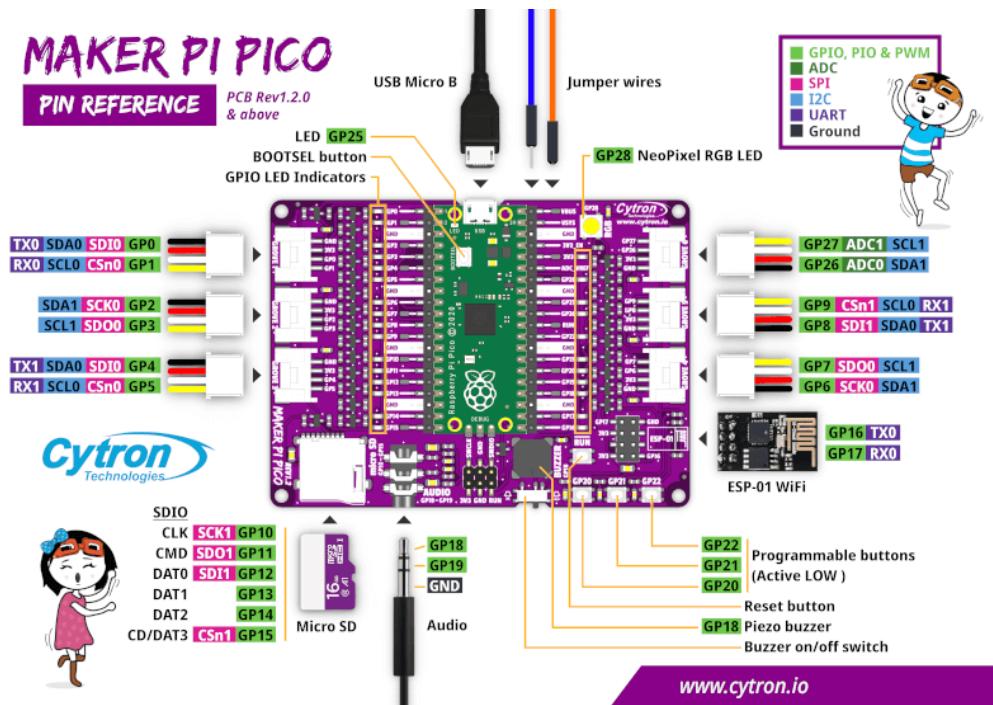
⁷<https://rpihop.cz/raspberry-pi-pico/5073-raspberry-pi-pico-w-5056561803173.html>

⁸<https://rpihop.cz/>

1.4.6 Dodatočné periférie pre Raspberry Pi Pico W

Počas jednej z konzultácií mi bolo odporučane pre potreby prace použiť sa na pomocný prvok pre *Raspberry Pi Pico W* s názvom *Cytron Maker Pi Pico Base*⁹ (obr. 1.4). Ide o pomocnú dosku, ktorej účelom je zjednodušiť pripojenie externých modulov. Skúšal som túto dosku pri vývoji meteorologickej stanice a môžem potvrdiť, že táto pomocná doska skutočne šetrí čas a nervy.

Hlavnou nevýhodou však je, že táto doska je dosť draha, konkrétnie 11,02 € za kus. Okrem toho je potrebné dokúpiť špeciálne káble pre pripojenie modulov. Napriek zvýšeniu nákladov projektu, čo je v rozpore s mojím princípom ekonomického prístupu, implementujem túto dosku do kurzu, ale pridám aj možnosť nepoužívať ju na zlacnenie konečného výsledku.



Obr. 1.4: Vihľad *Cytron Maker Pi Pico Base* [13]

1.4.7 Výber MQTT brokera

Zo svojich skúseností s vývojom riešení IoT môžem povedať, že technológie HiveMQ¹⁰ a Mosquito¹¹ sú medzi vývojármi najobľúbenejšie. Preto pokladám za logické používať tieto MQTT broukery. Ak tieto dve služby zoberieme do úvahy z

⁹<https://rpishop.cz/pico-karty/3854-cytron-maker-pi-pico-base-deska-pro-pi-pico-pro-zacatecniky-38515758.html>

¹⁰<https://www.hivemq.com/>

¹¹<https://mosquitto.org/>

hľadiska vývoja nášho produktu IoT, nie je medzi nimi veľký rozdiel. Existuje len malá výhoda Hivemq v podobe príťažlivejšieho webového rozhrania.

Pri príprave tejto kapitoly som sa zoznámil s študentom Šimonom Pavlišinom, ktorého bakalársky projekt sa týkal vývoja IoT Gateweja a služby MQTT brokera.

Jeho projekt sa volá *Otvorený IoT Lab pre stredné školy*[14] a vyvíja v ňom *Gateway* (ďalej GW) založenú na technológii Mosquito.

Tento projekt má niekoľko zaujímavých vlastností:

- Samotná aplikácia brokera je nainštalovaná na *Raspberry Pi* tretej alebo štvrtnej verzie a beží autonómne. Čiže pre lepšie pochopenie si tento projekt môžete predstaviť ako wifi router, ktorý funguje v rámci jedného laboratória alebo učebne, ale namiesto distribúcie wifi slúži ako MQTT broker.
- Ďalšou vlastnosťou je, že na tomto GW možno vytvárať rôzne aplikácie, ktoré budú bežať paralelne. To sa dosiahne pomocou Docker¹² kontajnerov.

Po oboznámení sa s koncepciou projektu Šimona Pavlišina a otestovaní jeho raného prototypu som si uvedomil, že je to presne to, čo potrebujem pre svoju prácu.

Pre väčšiu istotu uvediem nasledujúce argumenty:

1. Možnosť paralelného behu mnohých aplikácií na jednom GW je veľmi silným argumentom pri školení mnohých ľudí.
2. Už pri práci s raným prototypom tohto projektu som bol presvedčený o jeho použiteľnosti.
3. Keďže tento GW podporuje Docker, môžem na programovanie aplikácie svojho projektu použiť nástroj NodeRed. Výhody tohto nastroja oproti iným možnostiam opíšem v nasledujúcej časti.

1.4.8 Prečo NodeRed?

Na základe mojich skúseností môžem povedať, že programovanie pomocou nástroja NodeRed je jedno z najjednoduchších vo svojej oblasti.

Myslím si, že pri vývoji aplikácie, ktorá je na štvrtej úrovni, je potrebné, aby táto fáza bola čo najjednoduchšia a najľahšia. Podľa môjho názoru je to totiž jedna z najťažších fáz vývoja a myslím si, že ak túto časť príliš skomplikujete, bude materiálu na 5 prednášok príliš veľa. Preto si myslím, že NodeRed bude najjednoduchší a najnázornejší spôsob, ako to urobiť.

¹²<https://www.docker.com/>

2 Syntetická časť

V tejto časti chcem nielen opísať konečný produkt, ktorý som vytvoril, ale aj celú cestu jeho vývoja. Týmto spôsobom pomôžem čitateľom tejto práce, aby sa pri vývoji podobných vzdelávacích projektov nedopustili podobných chýb. Vývoj celého projektu som rozdelil do nasledujúcich etáp:

1. Vývoj meteorologickej stanice (1-2 vrstvy architektúry).
2. Vývoj aplikácie (4. vrstva architektúry).

Samostatnou etapou je testovanie celeho riešenia, ale to bude trochu neskôr.

2.1 Počiatočný vývoj meteostanice

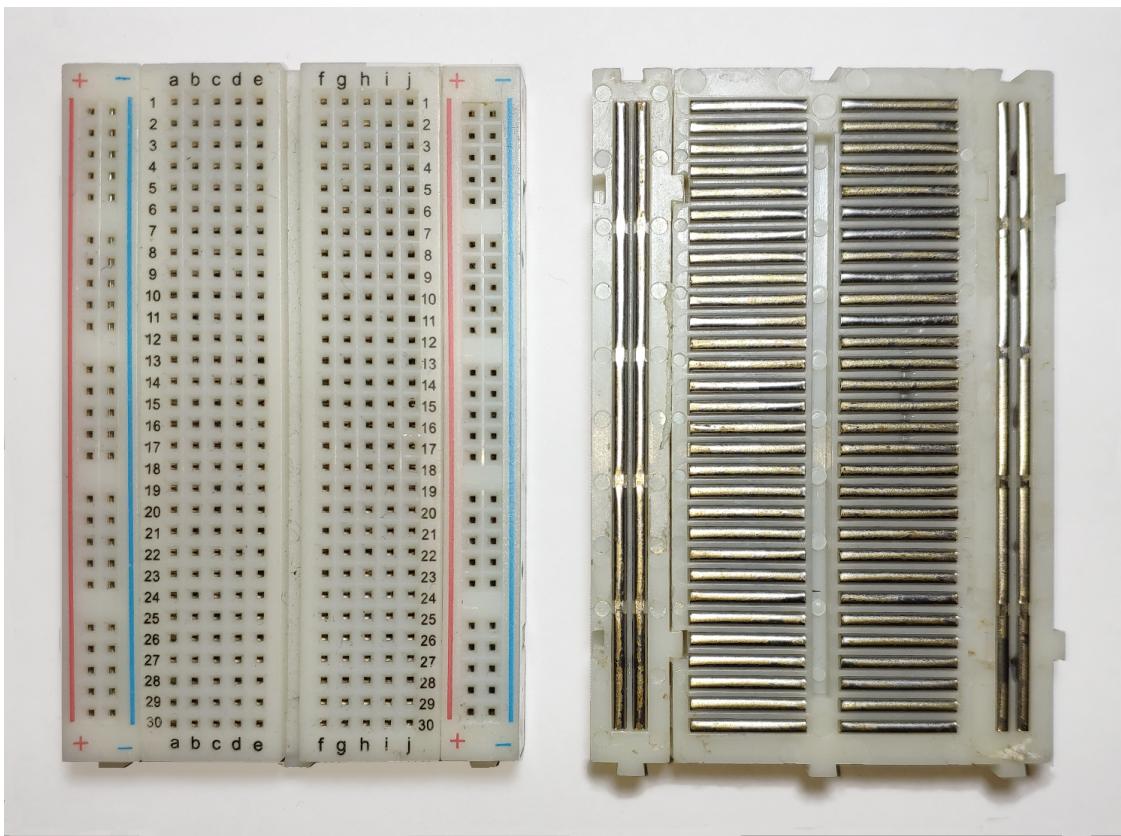
Svoj vývoj som začal od prvých vrstiev podľa konceptu architektúry tohto projektu 1.1, pretože je to pre mňa najlogickejšie riešenie. Preto som ako prvú vec začal vyvíjať meteorologickú stanicu.

2.1.1 Prvé pokusy

Ked' som začal pripravovať projekt, nemal som jasný plán postupu ani predstavu o tom, ako by mala meteorologická stanica vyzeráť vo svojej konečnej podobe. Kvôli nedostatku skúseností a praktických zručností som mal pri vypracovávaní projektu problémy. Prvé pokusy vývoja boli skôr experimentmi a oboznamovaním sa s výrobným procesom než serióznym vývojom. Po niekoľkých týždňoch takejto neefektívnej práce som však presne pochopil, ako tento projekt vyvíjať a aké základné funkcie musí tento projekt obsahovať.

Veľmi dôležitú úlohu zohrala skutočnosť, že som pomerne intenzívne komunikoval so svojim veducim praci, ktorý mi v rámci možností radil vo všetkých otázkach, ktoré ma v súvislosti s procesom vivoja zaujímali.

Na začiatku vývoja projektu som používal *Raspberry Pi Pico WH* a všetky pripojenia modulov som robil na *breadboarde* (obr. 2.1). Ale po čase som na uľahčenie vývoja začal používať *Cytron Maker Pi Pico Base*.



Obr. 2.1: Vzhľad breadboardu z oboch strán[15]

2.1.2 Zásady vývoja

Pre správne splnenie úloh, musel som si vytvoriť podrobný, postupný plán činnosti pre vývoj meteostanice.

Po premyslení tohto problému som prišiel s nasledujúcim akčným plánom:

1. Vytvoriť hardvérovú časť meteostanice, konkrétnie:
 - Model toho, čo by mala táto meteorologická stanica obsahovať.
 - Nakresliť elektricky obvod na pripojenie modulov.
 - Vykonanie schémy na reálnom príklade.
2. Vytvoriť softvérovú časť meteorologickej stanice a implementovať nasledujúce funkcie:
 - Zber údajov zo senzorov.
 - Odosielanie údajov na MQTT broker.
 - Úspora energie (zabezpečiť možnosť autonómie).
 - Tolerancia porúch.
 - Dodatočné fičury.

2.2 Hardvérová časť meteostanice

Meteorologická stanica by mala odčítať nasledujúce údaje:

- Teplota
- Vlhkosť vzduchu
- Intenzita svetla

Na tieto úlohy som použil senzor LDR (intenzita svetla) a DHT11 (teplota a vlhkosť). Namiesto DHT11 by bolo možné použiť samostatné snímače teploty a vlhkosti, ale to je zbytočná komplikácia procesu vývoja. Namiesto DHT11 je môžne použiť novšiu verziu DHT22. V ich zapojení nie je osobitný rozdiel, ale na softvérovej úrovni novšia verzia používa inú knižnicu.

Počas priameho vývoja som sa stretol s problémom, že počas autonómnej pre-vádzky zariadenia nie je jasné, či funguje alebo má problém. Preto som do zaria-denia pridal LED diódu, ktorá informuje vývojára o stave meteostanice.

Na pripojenie k *Raspberry Pi Pico* budete potrebovať dva rezistory, a to $10\text{k}\Omega$ a $220\text{k}\Omega$.

Konečný zoznam použitých komponentov je tu:

- DHT11
- LDR
- LED dióda
- $10\text{k}\Omega$ rezistor
- $220\text{k}\Omega$ rezistor

Na obrázku vidíme schému pripojenia všetkých modulov k doske *Raspberry Pi Pico*.

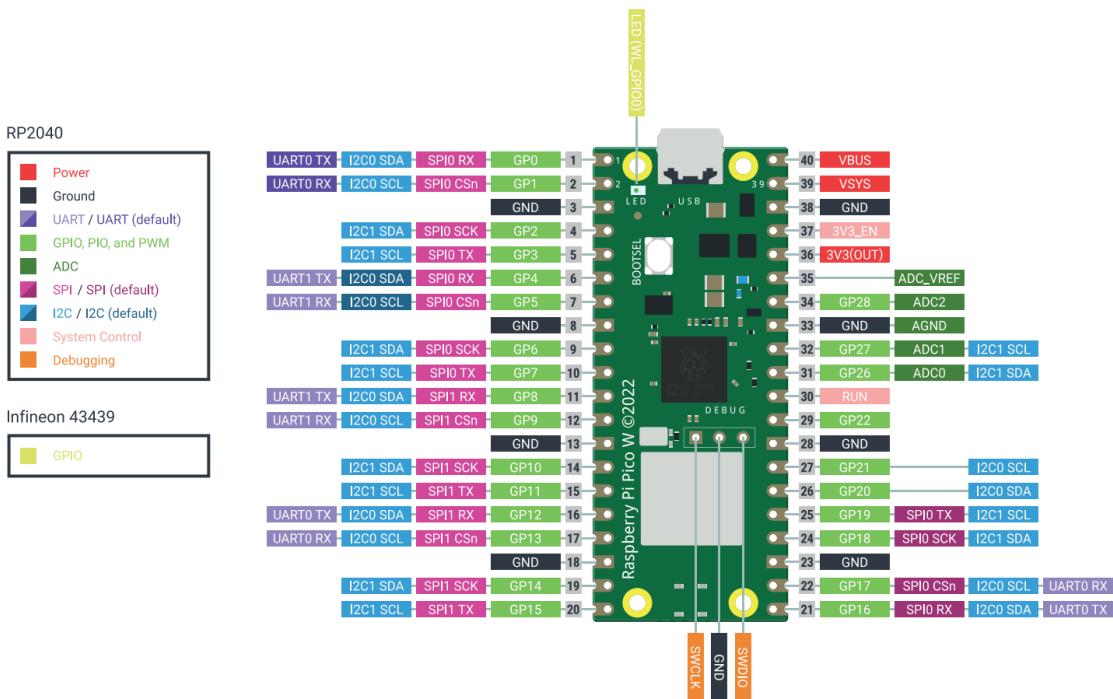
2.2.1 Schéma zapojenia

Ako už bolo opísané v teoretickej časti, projekt bude vyvinutý na základe doski *Raspberry Pi Pico W*. Preto bola všetka práca vykonaná na tejto doske. Ale ja som otestoval fungovanie svojho projektu aj na *ESP32* so zabudovaným WiFi modu-lom a môžem povedať, že som žiadne problémy nezistil.

Preto sa v prípade potreby môže projekt realizovať aj pomocou doski na báze čipu *ESP32*, a nie *Raspberry Pi Pico W*. Chcem však upozorniť, že pred vedením

kurzu musí učiteľ dôkladne otestovať verziu *ESP32*, ktorú majú študenti k dispozícii, pretože existuje veľa konfigurácií dosiek s čipom *ESP32* a ja nemôžem zabezpečiť fungovanie projektu na každej z nich. To znamená, že *Raspberry Pi Pico W* je zárukou, že všetko bude fungovať tak, ako som ako vývojár zamýšľal.

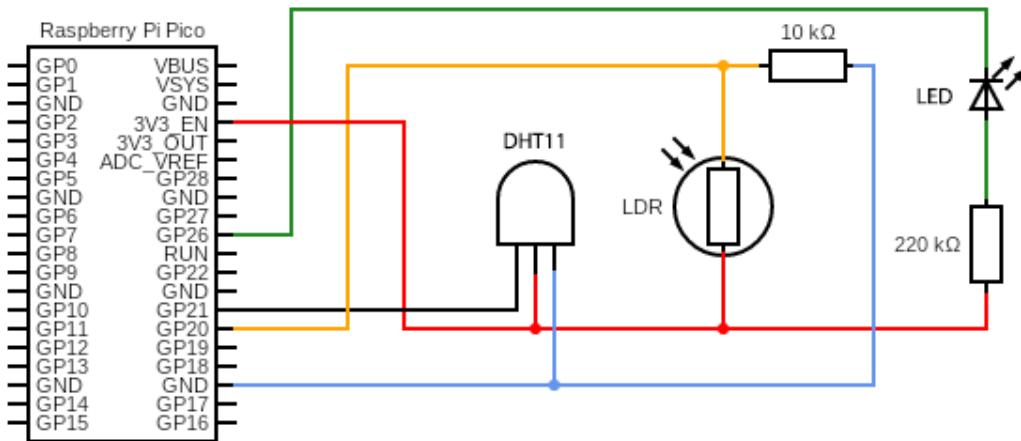
Ak sa teda vrátimo k téme pripojenia jednotlivých modulov k doske mikroprocesora, potrebujeme dva digitálne piny na pripojenie diódy a DHT11 a jeden analógový pin na pripojenie LDR senzora. Je potrebné poznamenať, že LDR senzory sú dvoch typov, s digitálnymi a analógovými výstupmi. V projekte možno použiť obe verzie, ale je potrebné zohľadniť rozdiel v ich pripojení na softvérovej úrovni. Ja použijem analógovú verziu LDR, pretože je súčasťou *Arduino Kit*.



Obr. 2.2: Rozloženie pinov na doske *Raspberry pi pico W* [16]

Ako vidne na obrázku 2.2, *Raspberry Pi Pico W* má iba tri analógové piny, a to: GP26, GP27, GP28. Môžete použiť ktorýkoľvek z nich, ja použijem GP26. Na pripojenie DHT11 a LED diódy môžete použiť ktorýkoľvek z dostupných digitálnych pinov, ja použijem GP21, a GP20. Na obrázku 2.3 môžete vidieť celú schému pripojenia k mikroprocesoru *Raspberry Pi Pico W*.

Dodatočne by som chcel upozorniť, že schéma zapojenia znázornená na obrázku 2.3 môže byť v niektorých prípadoch nefunkčná, pretože počet pinov na rôznych verziách modulov (napr. DHT11) sa môže lísiť od toho, čo som uviedol. Preto by mal učiteľ pred vedením kurzu skontrolovať existujúcu konfiguráciu modulov a v prípade potreby zmeniť schému zapojenia.



Obr. 2.3: Chéma pripojenia k doske Raspberry pi pico W

2.2.2 Príklady realizácie schémy zapojenia

Celkovo som tento schému (obrázok 2.3) realizoval v troch formách:

1. Pomocou *Raspberry Pi Pico W* (obrázok 2.4).
2. Pomocou *Raspberry Pi Pico W* s použitím *Cytron Maker Pi Pico Base W* (obrázok 2.5).
3. Pomocou *ESP32* (obrázok 2.6).

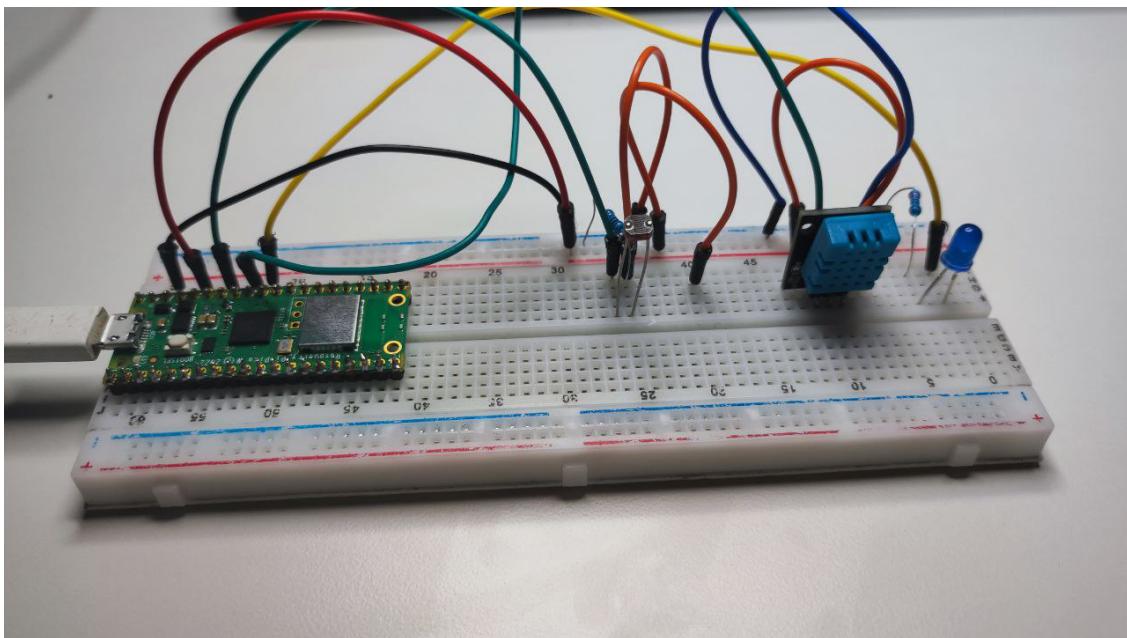
Ako je môžne vidieť na obrázkoch 2.4 a 2.6, medzi vzhľadom pripojenia *ESP32* a *Raspberry Pi Pico W* nie je veľký rozdiel. Najjednoduchšie sa pripájal *Raspberry Pi Pico W* s pomocnou doskou (obr. 2.5), pretože nebolo potrebné použiť dodatočné rezistory ani veľké množstvo káblov.

Môžem tiež poznamenať, že schéma rozpojenia na obrázku 2.3 je len príkladom a ak spojenie urobíte trochu inak pri dodržiavaní logiky spojenia, nemali by vzniknúť žiadne problémy.

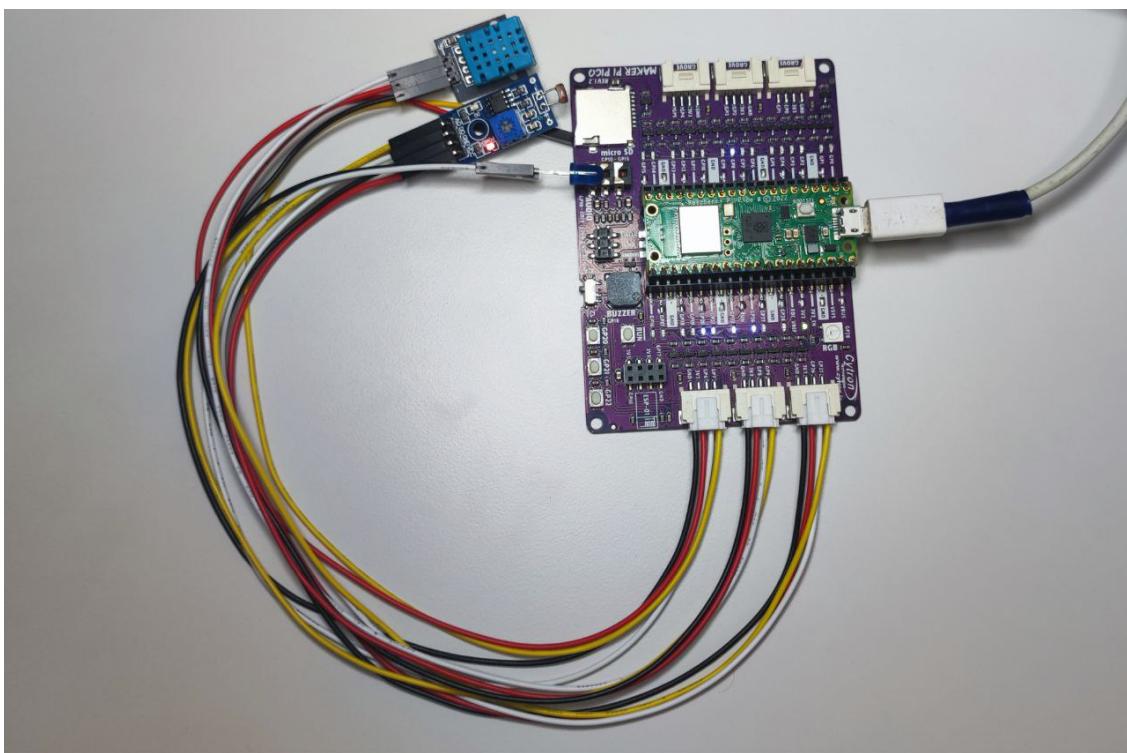
Pri práci s mikroprocesormi je dôležité dodržiavať bezpečnostné pravidlá, konkrétnie zabrániť skratom pri zostavovaní fyzickej kópie projektu. Pri práci s *ESP32* som kvôli nedostatku skúseností spálil mikroprocesor skratom. Viac takýchto incidentov som už nepripustil.

2.3 Vývoj softvérovej časti meteostanice

Softvér bol napísaný v programovacom jazyku *MicroPython*, ako bolo napísané v predchádzajúcich častiach bakalárskej práce.



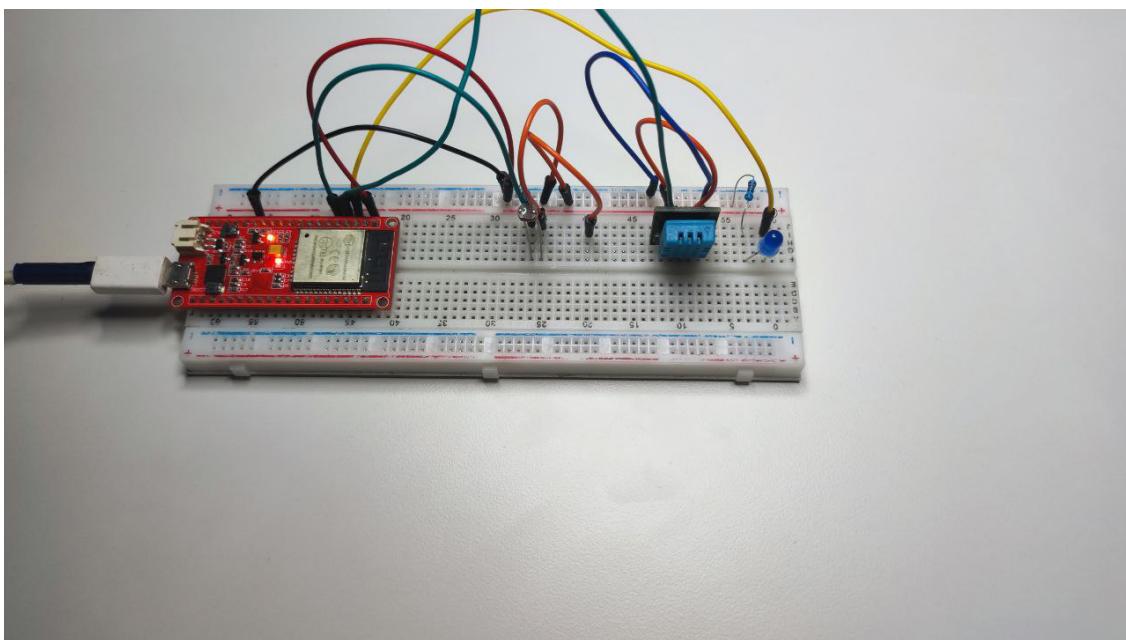
Obr. 2.4: Fyzický príklad schémy zapojenia 2.3 pomocou *Raspberry Pi Pico W*



Obr. 2.5: Fyzický príklad schémy zapojenia 2.3 pomocou *Raspberry Pi Pico W* s doplnkovou doskou *Cytron Maker Pi Pico Base*

Softvér pre meteorologickú stanicu bol vyvinutý v troch iteráciách:

1. Počiatočné vytvorenie kódu so všetkými potrebnými funkciami.
2. Refactoring.



Obr. 2.6: Fyzický príklad schémy zapojenia 2.3 pomocou *ESP32*

3. Testovanie.

V nasledujúcich podkapitolách podrobnejšie opíšem každú z týchto fáz.

2.4 Hlavná časť vývoja (Prvá iterácia)

2.4.1 Postup vývoja

V prvej iterácii vývoja kódu som sa nestaral o čistotu kódu a nerobil som žiadny refaktoring. To mi umožnilo sústrediť sa na samotný vývoj.

Počas vývoja kódu som postupne pridával nové funkcie do meteorologickej stanice a snažil som sa dodržať vyššie uvedený plán vývoja.

Pozrime sa na celý vývojový cyklus prvej iterácie a podrobnejšie rozoberme jednotlivé fázy pridávania nových funkcií pri zachovaní časovej hierarchie pridávania do projektu:

1. Čítanie senzorových dát - implementoval som možnosť čítať metriky vlhkosti, teploty a svetla zo senzora LDR a DHT11.

Aby som zabránil načítaniu nesprávnych informácií, odčítam údaje 3-krát a vypočítam z nich medián. Ak sa teda z nejakého neznámeho dôvodu jedenkrát odčítajú nesprávne údaje, v nasledujúcich krokoch sa nezohľadnia.

2. Pripojenie k internetu - táto fáza bola najjednoduchšia na realizáciu.

3. *Pripojenie k MQTT* - na implementáciu tejto a ďalšej etapy som použil knižnicu tretej strany, pretože základný balík *MicroPythonu* neobsahoval knižnicu, ktorú som potreboval.
4. *Odoslanie údajov* - zozbierané a spracované údaje z prveho kroku sa odošlú do sprostredkovateľa MQTT.

V počiatočných fázach som údaje posielal vo forme troch čísel, ale časom som si uvedomil, že tento typ údajov sa dosť ľažko číta, preto som začal údaje posielat vo forme JSON správy.

5. *Šetrenie energie* - keďže meteorologická stanica má byť autonómna, šetrenie energie je dôležitou súčasťou tohto produktu.

Šetrenie energie som realizoval znížením taktu procesora, alebo takzvaným *uspávaním* mikroprocesora. Dosahuje sa to pomocou špeciálnych príkazov a v prvých fázach vývoja som používal funkciu `sleep()`, ktorá pozastaví aktuálny proces, čím sa zníži zaťaženie procesora. V budúcnosti som ju však nahradil energeticky úspornejšou funkciou `deepSleep()`, ktorá zastaví všetky aktuálne procesy a prejde do úsporného režimu.

Túto fázu som vyvinul na základe týchto uvedených experimentov [17][18][19].

6. *Problémová signalizácia* - ak meteorologická stanica nie je pripojená k počítaču, spotrebiteľ alebo vývojár nemá možnosť zistiť aktuálny stav systému bez prístupu k sériovému portu. Preto som sa v tomto bode rozhodol pridať LED diódu, ktorá bude signalizovať aktuálny stav zariadenia.

Teraz počas prevádzky zariadenia dióda signalizuje úspešnosť takých akcií, ako napr.:

- Pripojenie k wifi.
- Pripojenie k sprostredkovateľovi MQTT.
- Odosielanie údajov.

Ak dióda svieti 5 sekúnd, akcia bola úspešná, a ak bliká, akcia sa nepodarila. Na lepšie označenie problémov možno čas blikania a ich počet pre jednotlivé akcie meniť.

7. *Tolerancia porúch* - počas prevádzky meteostanice existuje možnosť, že pripojenie k internetu alebo k serveru MQTT sa môže z nejakých príčin prerušiť.

V čase takýchto problémov s pripojením môže dôjsť k strate zozbieraných údajov, pretože sa nebudú môcť dostať na server. Preto som sa rozhodol tento problém vyriešiť nasledujúcim spôsobom.

Pred odoslaním údajov sprostredkovateľovi MQTT program skontroluje sieťové pripojenie a ak nie je dostupné, uloží zozbierané údaje do buffer súboru. Ak sa internetové pripojenie počas nasledujúcich iterácií opäť objaví, všetky údaje, ktoré sa nahromadili v súbore, sa odošlú na server. Toto je pomerne jednoduchý, ale účinný algoritmus.

8. *Aktualizácia času* - pri vývoji predchádzajúceho kroku som narazil na problém, že odoslané údaje sa nedajú rozlíšiť podľa času ich načítania. To znamená, že strana servera nerozumie presnému času merania a potenciálny spotrebiteľ môže analyzovať údaje len podľa času ich odoslania na server.

Preto som do JSON správy, ktorá sa posiela na server, pridal parameter `time`, ktorý znamená presný čas tohto merania v UTC. Narazil som však na ďalší problém. Konkrétnie na problém kontextualizácie času.

Ak je aplikácia spustená cez IDE(napr. *Thonny*), tak by nemal nastať žiadny problém, pretože IDE automaticky aktualizuje čas na mikroprocesore, ale ak je aplikácia spustená bez mikroprocesora, tak sa čas neaktualizuje a nastaví sa na začiatok éry¹(1970-01-01 00:00:00 UTC). Preto som do programu implementoval automatickú aktualizáciu času, ale na jej vykonanie je potrebné mať prístup na internet.

2.4.2 Pomocná knižnica

Pri vývoji softvéru pre meteorologickú stanicu ma napadlo vytvoriť samostatnú knižnicu s funkciami, o ktorých predpokladám, že by mohli byť pre študentov s malými skúsenosťami s programovaním dosť náročné. Mohla by tiež obsahovať niektoré funkcie, ktoré štandardizujú údaje.

Vytvoril som teda takúto knižnicu a pomenoval som ju `helper.py`.

Táto knižnica obsahuje nasledujúce funkcie:

- `def do_connect_wifi(ssid, password)` - Pripája sa k zadanej sieti WiFi.
Vráti: `True, False`.(v závislosti od úspešnosti pripojenia)
- `def do_connect_broker(name, server, port, login, password)` - Pripája k MQTT brokeru.
Vráti: Objekt MQTT brokera, `None`.(v závislosti od úspešnosti pripojenia)

¹<https://docs.micropython.org/en/latest/library/time.html>

- `def send_mqtt_message(mqtt, topic, message)` - Odošle správu na zadaný MQTT broker.
Vráti: True, False.(v závislosti od úspešnosti pripojenia)
- `def create_json(time, temperature, humidity, light)` - Konvertuje zadané údaje do formatu JSON.
Vráti: JSON v textovej forme.
Tento príkaz slúži na štandardizáciu správ odosielaných na MQTT server.
- `def read_light(adc_pin)` - Meria osvetlenie z daného pinu.
Vráti: Percent osvetlenia od 0 do 100.

2.4.3 Kritické problémy, s ktorými som sa stretol počas vývoja.

Ako som už napísal, pri práci s mikroprocesormi je potrebné dodržiavať bezpečnostné opatrenia, aby ste predišli rôznym nepríjemným situáciám. Pri vývoji softvéru sa však vývojár môže dopustiť niektorých nezrejmých chýb, ktoré môžu viesť k úplnej strate kódu, alebo dokonca firmvéru. Počas svojej práce som sa s jedným takýmto problémom stretol a v tejto časti ho opíšem, aby prípadný čitateľ tejto práce neurobil podobnú chybu.

Dovoľte mi teda uviesť stručný prehľad tohto problému. Pri opise vývoja som v časti o šetrení energie písal o vhodnejšom použití funkcie `deepsleep()` namiesto funkcie `sleep()`. Počas aktívneho vývoja, keď má váš kód nejaké problémy, sa však pri použití funkcie `deepsleep()` môže váš kód dostať do tzv. "dead loop". Zvyčajne sa problémy s "dead loop" riešia reštartovaním systému, ale nie v prípade funkcie `deepsleep()`. Aby fungovala správne, hlavný súbor vašej aplikácie sa musí volať `start.py`, to znamená, že je to súbor, ktorý sa spustí pri každom zapnutí mikroprocesora.

Riešením tohto problému je úplné prepísanie systému mikroprocesora. Možno existujú humánnejšie spôsoby riešenia tohto problému, ale v mojom prípade je to jediný spôsob, ktorý som mohol urobiť.

Aby ste tomuto problému predišli, odporúčam vám používať funkciu `deepsleep()` až v záverečných fázach, keď ste si istí, že program pracuje správne. V počiatočných fázach vývoja vám odporúčam používať funkciu `sleep()`.

2.4.4 Výsledok vývoja a potreba refactoringu.

Takže keď som implementoval všetky potrebné funkcie a vyriešil som všetky problémy súvisiace s vývojom meteostanice, získal som produkt, ktorý cyklicky

vykonáva tieto kroky:

- Prebudí sa a pripojí sa k sieti.
- Sníma a odosiela údaje na server.
- Zaspí na 5 minút.

Je to veľmi jednoduchý algoritmus, ale pre meteostanicu nepotrebuje nič viac. V tejto etape už teda projekt vyzerá viac-menej dobre, takže je čas premýšlať o tom, ako budú môcť študenti realizovať projekt.

Po zamyslení sa nad touto otázkou som dospel k záveru, že na to, aby študenti mohli pracovať efektívne, je potrebné vytvoriť kostru projektu, na základe ktorej budú študenti vykonávať ďalší vývoj. To znamená, že namiesto toho, aby študenti robili všetko od začiatku, budú mať základ projektu a zoznam funkcií, ktoré musia urobiť. Pri tejto metóde študenti strácajú pocit neurčitosti a nepochopenia, a tak sa kód študentov stáva štandardizovanejším, čo zlepšuje možnosť hodnotenia.

Koreň projektu je logické vytvoriť na základe už hotového projektu, teda toho, čo som už urobil. Je tu však jeden problém, momentálne je môj kód pre meteorologickú stanicu veľmi hrubý a neprehľadný na čítanie. Preto som dospel k záveru, že softvér, ktorý som vytvoril, je potrebné refaktorovať.

2.5 Refactoring (Druhá iterácia)

Takže v čase, keď som začal refaktorovanie, vyzeral môj projekt takto:

- `main.py` - tu sa nachádza hlavná časť kódu.
- `helper.py` - pomocná knižnica, o fungovaní ktorej som už písal.

Kód v súbore `main.py` bol pred refaktorovaním veľmi neprehľadný a bolo veľmi ťažké identifikovať funkcie, ktoré by mohli byť súčasťou kostry projektu. Preto bolo rozhodnuté rozdeliť kód v tomto súbore na niekoľko častí.

Počas jednej z konzultácií mi učiteľ Miroslav Biňas vnukol myšlienku, že proces fungovania meteostanice možno reprezentovať ako konečný stavový automat a kód projektu možno rozdeliť do jednotlivých stavov projektu. Táto myšlienka sa mi zdala naozaj veľmi chytrá, preto som kód projektu zmapoval na tento model. V nasledujúcich kapitolách budem podrobnejšie hovoriť o tom, ako tento koncept vyzerá a aký bol konečný výsledok.

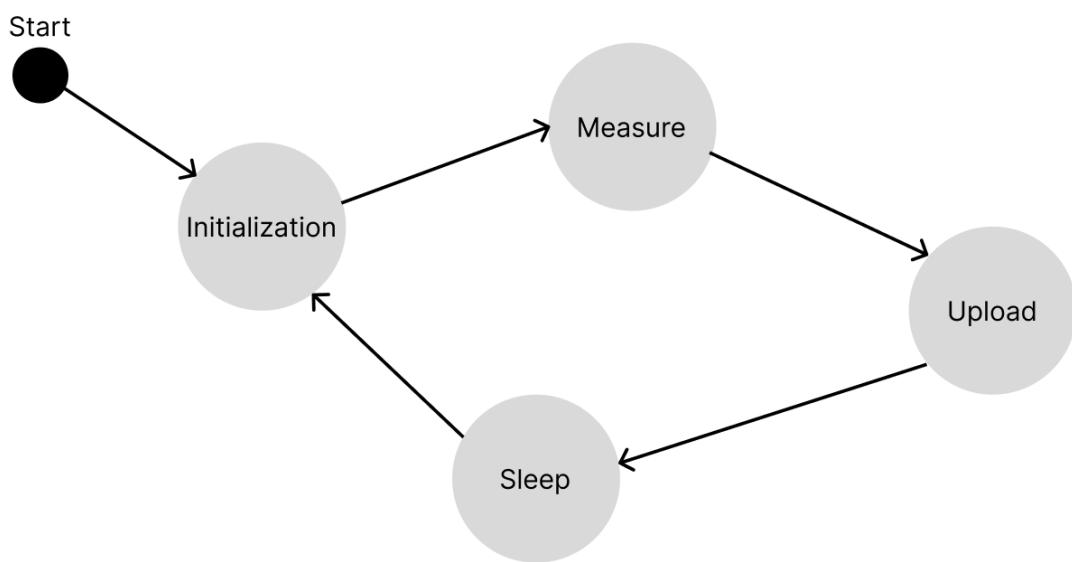
2.5.1 Konečný stavový automat v projekte

Konečný stavový automat je koncept, ktorý opisuje veci v zmysle jednotlivých stavov a počet týchto stavov je konečný[20]. V našom prípade použijem tento pojem na opis cyklickej logiky meteostanice v rámci jednotlivých stavov.

Analyzoval som logiku meteostanice a identifikoval som nasledujúce stavy:

- Inicializácia - zariadenie sa pripojí k wifi a MQTT brokeru.
- Meranie - zariadenie meria parametre teploty, vlhkosti a svetla.
- Odosielanie - zozbierané údaje sa spracujú a odošlú na server MQTT.
- Uspanie - zariadenie prejde do energeticky efektívneho režimu na konštantný čas.

Všetky tieto stavy sa cyklicky striedajú jeden za druhým donekonečna. Na obrázku 2.7 môžete vidieť, ako vyzerajú vo forme diagramu.



Obr. 2.7: Práca meteostanice vo forme konečno stavovo automatu.

2.5.2 Výsledok refaktorovania kódu.

Takže na základe konečného stavového automatu zobrazeného na obrázku 1 som prepracoval štruktúru celého projektu. Nezmenený zostal len súbor `helper.py`

Pre lepšie pochopenie novej štruktúry projektu najprv uvediem všetky súbory, ktoré sa v projekte nachádzajú:

- `start.py` - štartovací súbor, ktorý obsahuje logiku prechodu z jedného stavu do druhého.
- `states.py` - súbor, ktorý obsahuje celú logiku jednotlivých stavov.
- `helper.py` - súbor s pomocnými funkciami.
- `settings.py` - tu sa nachádzajú konštantné parametre, ktoré sa v projekte používajú na uľahčenie prístupu k nim.

Teraz sa pozrime na štruktúru projektu a jednotlivé súbory podrobnejšie.

Ako som povedal, poradie prechodu z jedného stavu do druhého sa nachádza v súbore `start.py`, každý stav je obsiahnutý vo funkciách:

- `def init(context)` - inicializácia
- `def measure(context)` - meranie
- `def upload(context)` - nahrávanie
- `def sleep_pico(context)` - uspávanie

Tieto funkcie sa nachádzajú v súbore `states.py` a práve tieto funkcie budú musieť študenti naprogramovať. Okrem týchto funkcií môžu študenti vytvoriť aj iné funkcie, ale hlavné je, aby sa názov uvedených funkcií nemenil. Súbor `start.py` obsahuje aj triedu `Context`, ktorá sa používa ako globálna štruktúra s premenými. Tieto premenné sú pomocné a používajú sa v rôznych funkciách. Funkcie v súbore `helper.py` nebudú študenti programovať a používajú sa skôr ako knižnicu. Súbor `settings.py` obsahuje konfiguračné premenné, ktoré sa zvyčajne používajú len raz. Tieto premenné som oddelil do samostatného súboru, aby som zlepšil čistotu kódu a uľahčil zmenu týchto údajov.

2.6 Testovanie (Tretia iterácia)

2.6.1 Ciele

Účelom testovania je skontrolovať, či projekt neobsahuje výrazné problémy alebo nejasnosti v návrhu projektu. Testovanie by sa malo vykonávať na cielovej skupine projektu a malo by sa prednostne vzťahovať na celý projekt.

Ďalším účelom testovania je overenie množstva práce, ktorú študenti dokážu vykonať v stanovenom čase. Robí sa to preto, aby sa projekt dal ľahšie rozdeliť na niekoľko častí (prednášok). Je to preto, aby sa projekt ľahšie implementoval do kurzu.

2.6.2 Príprava

Test sa mal uskutočniť v Gymnáziume svätého Tomáša Akvinského². Na test bola určená 1 hodina a 30 minút. Vo vymedzenom čase som chcel študentom odovzdať čo najviac informácií, preto som si pripravil podrobný plán postupu, aby sa minimalizovali časové straty.

Test som rozdelil na teoretickú a praktickú časť. V teoretickej časti som si dal za úlohu povedať o týchto veciach:

1. Čo je to IoT.
2. Porozprávať o vrstvách architektúry IoT riešení.
3. Vysvetliť architektúru môjho projektu.
4. Vyjadriť študentom, čo budú robiť v praktickej časti.

Pri plánovaní praktickej časti som nemohol predpovedať, kolko práce budú študenti schopní urobiť v čase určenom na test. Preto som naplánoval pomerne veľa položiek, a to:

1. Poskladanie hardvérovej časti projektu.
2. Testovanie blikaním LED diódy.
3. Meranie hodnôt zo senzorov.
4. Pripojenie k internetu a MQTT brokeru.
5. Odosielanie údajov vo formáte JSON.
6. Implementácia zníženia spotreby energie.
7. Signalizácia problémov pomocou LED diódy.
8. Zápis a čítanie údajov zo súboru.

Praktická časť bola implementovaná na doskách *ESP32*, pretože škola mala k dispozícii len ich. S ohľadom na to som upravil schému pripojenia k doske.

Na programovanie som sa rozhodol použiť *IDE Thoony*, pretože už bolo nainštalované na školských počítačoch.

Pri príprave a realizácii testu mi pomáhal Šimon Pavlišin a ja som bol vedúcim testu.

²<https://www.gta.sk/>

2.6.3 Úroveň znalostí študentov

Test bol realizovaný na študentoch tretieho ročníka Gymnázia svätého Tomáša Akvinského. V čase testovania študenti už mali za sebou praktické úlohy s *Arduino Uno* a *ESP32*. Mali teda základné zručnosti v programovaní mikroelektroniky. Takéto schopnosti mohli zrýchliť niektoré fázy praktickej časti (napr. pripojenie modulov k doske *ESP32*).

2.6.4 Výsledky

Na teste sa zúčastnilo 12 študentov s rôznou úrovňou motivácie a znalostí. Počas testu sa mi podarilo spracovať celú teoretickú časť a prvé dva body praktickej časti. Študenti mali problémy s poskladaním hardvéru a mali veľké problémy s pochopením *Thonny*, takže veľa času sme venovali riešeniu problémov súvisiacich s týmto IDE.

Podľa študentov bola teoretická časť pomerne ľahko pochopiteľná, ale praktická časť bola dosť nejasná.

2.6.5 Záver

Tento test umožnil získať veľa užitočných informácií a pomohol vyhnúť sa niektorým problémom pri písaní scenára pre lekcie.

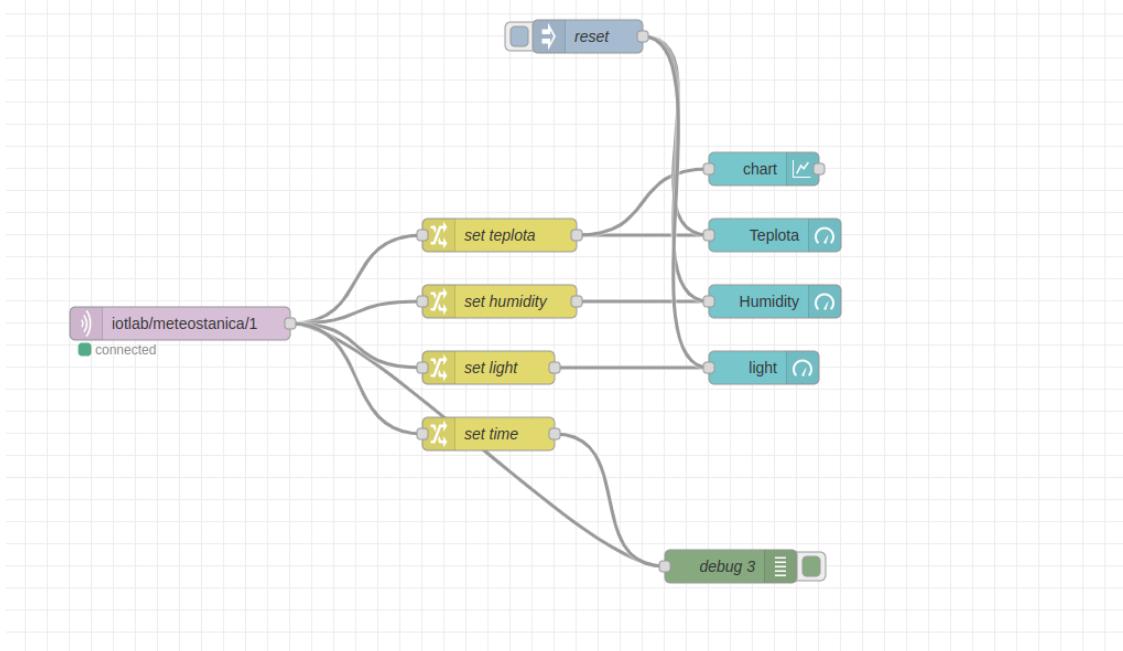
Fakt toho, že títo študenti už mali hodiny s *ESP32*, im nedal výraznú výhodu, pretože moduly k *EP32* pripojovali dosť dlho. Prekvapilo ma množstvo času stráveného riešením kritických problémov s *IDE Thonny*. Študenti nechápali, ako ho používať, a robili veľmi jednoduché chyby, ale ich riešenie mi zabralo dosť času. Myslím si, že tento problém vznikol v dôsledku môjho zlého plánovania, pretože som predpokladal, že táto fáza nezaberie veľa času, a tak som jej nevenoval dostatočnú pozornosť. Túto skúsenosť zohľadním pri príprave svojich hodín.

Vo všeobecnosti mi tento test poskytol predstavu o približnom množstve informácií, ktoré musím poskytnúť na jednej vyučovacej hodine môjho kurzu. Poskytol mi tiež neoceniteľné skúsenosti pri práci so študentmi.

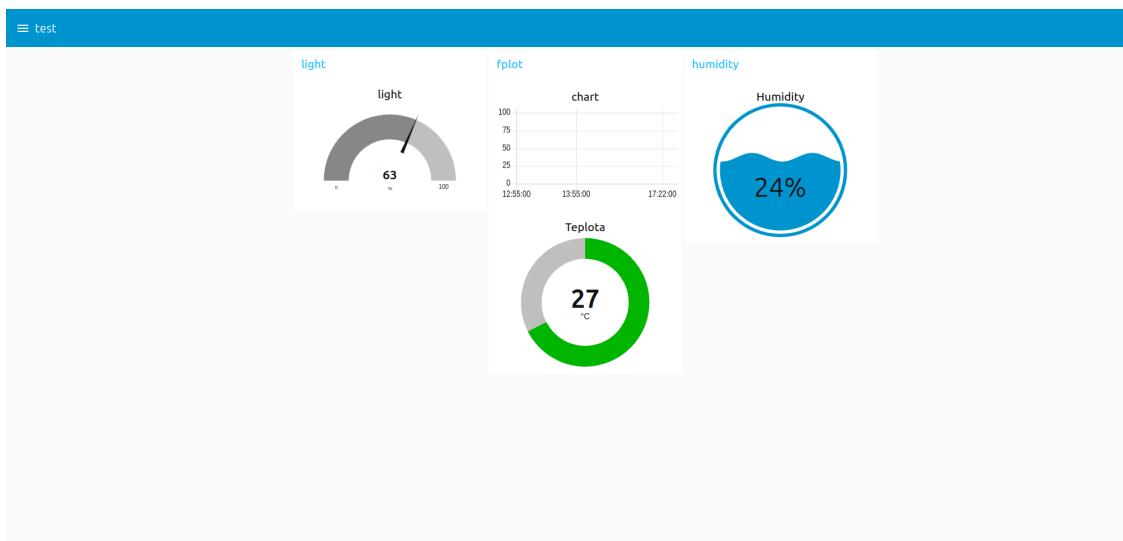
2.7 Zhromažďovanie údajov na bráne a ich spracovanie

Po ukončení vývoja meteorologickej stanice považujem za účelné ukázať študentom, ako pracovať so získanými údajmi, spracovať ich a vizualizovať.

Ako už bolo popísané v teoretickej časti, najlepšie sa to robí pomocou programovacieho jazyka *Node Red* na pripravenej IoT bráne. Vytvoril som teda pomerne jednoduchý program (obr. 2.8) na vizualizáciu údajov získaných z meteorologickej stanice. Jeho vizualizáciu si môžete pozrieť na obrázku 2.9. Napriek tomu,



Obr. 2.8: Logika aplikácie na vizualizáciu dát vytvorennej v *Node Red*



Obr. 2.9: Vizualizácia aplikácie vytvorennej na *Node Red*.

že logika programu je veľmi jednoduchá, na základe mojich skúseností môžem povedať, že na bežné zoznámenie s jazykom *Node Red* tohto stačí.

Program na obrázku 2.8 je len príkladom. Učitelia si pri implementácii môjho projektu do svojho kurzu môžu vytvoriť vlastný scenár na základe vedomostí svo-

jich študentov.

2.8 Analiza pridanej hodnoty projektu

2.8.1 Hlavná časť analýzy

Táto časť opisuje cenu niekoľkých projektových konfigurácií, a konkrétnie:

1. Pomocou *Raspberry Pi Pico WH* (obrázok 2.4).
2. Pomocou *Raspberry Pi Pico WH* s použitím *Cytron Maker Pi Pico Base W* (obrázok 2.5).
3. Pomocou *ESP32* (obrázok 2.6).

Ceny som prebral z webovej stránky *rpihop.cz*, pretože si myslím, že tam uvedené ceny plne zodpovedajú cenám na trhu. Okrem toho som analyzoval ceny na stránke *aliexpress.com*, aby som zobrazil, potenciálne, najnižšie možné ceny.

Treba poznamenať, že v čase písania tejto práce ešte neboli odstránené škodlivé následky krízy mikročipov³, takže cenová politika sa môže za nejaký čas výrazne zmeniť.

Ako už bolo spomenuté v iných kapitolách, stredné školy na Slovensku už zvyčajne majú k dispozícii stavebnice Arduino Kit, ktoré obsahujú moduly potrebné na kurz: DHT11, LDR, LED diódy. Preto som sa rozhodol ich cenu do kalkulácie nezahŕňať. Pre GW som sa rozhodol použiť *Raspberry Pi 3 Model A+*, pretože tento model je dostatočne výkonný na svoje úlohy.

Všetky výpočty sú k dispozícii v tabuľkách 2.1, 2.2 a 2.3. Výpočty boli vykonané v mene Euro(€).

	<i>ESP32</i>	<i>Raspberry Pi 3 Model A+</i>	Celková cena	Cena projektu pre 30 študentov
<i>rpihop.cz</i>	10,81 €	33,88 €	44,69 €	358,18 €
<i>aliexpress.com</i>	6,67 €	85,43 €	92,10 €	285,53 €

Tabuľka 2.1: Analýza ceny projektu na základe dosky *ESP32*.

³<https://www.telefonica.com/en/communication-room/why-is-the-microchip-crisis-affecting-us-this-much/>

	<i>Raspberry Pi Pico WH</i>	<i>Raspberry Pi 3 Model A+</i>	Celková cena	Cena projektu pre 30 študentov
rpishop.cz	8,86 €	33,88 €	42,74 €	308,54 €
aliexpress.com	5,83 €	85,43 €	91,26 €	260,33 €

Tabuľka 2.2: Analýza ceny projektu na základe dosky *Raspberry Pi Pico WH*.

	<i>Raspberry Pi Pico WH</i>	<i>Cytron Maker Pi Pico Base W</i>	<i>Raspberry Pi 3 Model A+</i>	Celková cena	Cena projektu pre 30 študentov
rpishop.cz	8,86 €	11,02	33,88 €	42,74 €	308,54 €
aliexpress.com	5,83 €	-	85,43 €	-	-

Tabuľka 2.3: Analýza ceny projektu na základe dosky *Raspberry Pi Pico WH* a *Cytron Maker Pi Pico Base W*.

2.8.2 Záver analýzy

Vzhľadom na vyššie uvedené výpočty môžeme povedať, že pri nákupe komponentov z *aliexpress.com* nebudú úplne lacnejšie ako v iných internetových obchodoch mikroelektroniky. Ak sa projekt bude vyvíjať na základe dosky *ESP32*, potom bude lacnejšie kúpiť ich na *aliexpress.com*, ale musíte brať do úvahy kvalitu tohto výrobku. Pri analýze ceny som na stránke *aliexpress.com* nenašiel dosku *Cytron Maker Pi Pico Base W*, takže ak sa rozhodnete pre túto konfiguráciu, musíte si toto zariadenie kúpiť na iných stránkach.

3 Vyhodnotenie

V tejto časti podrobne opíšem projekt, ktorý sa podarilo zrealizovať. Zanalyzujem jeho plusy a minusy. Uvediem aj možné alternatívne riešenia alebo doplnenia existujúceho projektu.

3.1 Téma a architektúra projektu

Vypracoval som systém zberu meteorologických dát (obr. 1.2), ktorý zahŕňa 3 úrovne architektúry IoT (obr. 1.1).

Počas plánovania a vývoja projektu bola otázka výberu témy pomerne diskutabilná, pretože výber témy priamo súvisí s architektúrou projektu a jeho obsahom. Obával som sa, že projekt bude drahší a príliš komplikovaný, preto som do projektu zapojil len senzory. Chápal som tiež, že je dôležité ukázať študentom niekoľko úrovní architektúry internetu vecí, preto som sa rozhodol pridať do projektu IoT bránu.

Výber hlavného inteligentného zariadenia pre meteorologickú stanicu bol tiež dosť diskutabilný. Veď najlacnejšou možnosťou by bolo *Arduino Uno*, ale ja som vybral *Raspberry Pi Pico W* z viacerých dôvodov opísaných v osobitnej časti.

Ak projekt posúdite z hľadiska externého, môžete dospieť k záveru, že jeho komponenty možno nahradí lacnejšími alebo energeticky úspornejšími. Pri vývoji tohto projektu som však chcel vytvoriť platformu na učenie, nie priemyselný výrobok. Preto som väčšinu svojich rozhodnutí robil z hľadiska práce so študentmi.

Verím, že architektúra môjho riešenia je racionálne vyvážená a dobre premyslená.

3.2 Softvér meteostanice

Architektúra softvéru meteostanice je založená na 4 fyzikálnych stavoch jeho životného cyklu (obr. 2.7). Jednotlivé funkcie softvéru sú rozdelené medzi týmito

stavmi. Softvér obsahuje základné funkcie IoT riešenia. Na programovanie meteostanice sa použil programovací jazyk *Micropython*.

Pre prípadných študentov, ktorí by tento projekt vyvíjali, som vytvoril kostru programu. Preto plne naprogramovaný projekt slúži len ako príklad pre učiteľa.

Myslím si, že architektúra kódu je dokonalá, ale napĺňanie funkcií jednotlivých stavov sa ukázalo ako dosť chaotické a neprehľadné. Preto by som neodporúčal, aby študenti presne reprodukovali môj kód.

3.3 Program na bráne

Projekt má bránu MQTT, ktorá zhromažďuje všetky údaje odoslané z meteorologických staníc a zobrazuje ich na webovom rozhraní. Návrh brány bol prevzatý z bakalárskej práce Šimona Pavlišina[14]. Programovací jazyk použitý v tejto fáze je *Node Red*.

Môžem povedať, že program, ktorý som napísal, je pomerne jednoduchý. Je to spôsobené tým, že som nechcel túto etapu komplikovať, ale prvýkrát sa zoznámiť s jazykom *Node Red* a z mojich skúseností môžem povedať, že takéto ľahké programy sú na túto úlohu ako stvorené.

V prípade potreby môže učiteľ na základe znalostí svojich študentov a dostupných údajov vymyslieť iný scenár.

3.4 Testovanie

Projekt bol testovaný na reálnych študentoch stredných škôl. Testovanie sa uskutočnilo na študentoch Gymnázia svätého Tomáša Akvinského na 12 člennej skupine a trvalo 1 hodinu a 30 minút. Cieľom testovania bolo overiť možnosť implementácie môjho projektu do vzdelávacieho kurzu v praxi a overiť, na koľko približných častí by sa mal projekt rozdeliť, aby sa dal preniesť do kurzu.

Výsledkom testovania bolo, že v projekte neboli zistené žiadne kritické problémy. Po testovaní sa ukázalo, že tento projekt by sa mohol rozdeliť na 4 - 5 častí pre implementáciu do kurzu.

Žiaľ, vzhľadom na mnohé okolnosti som nemohol projekt otestovať v plnom rozsahu a myslím si, že to je dosť veľké pochybenie tejto práce. Veď na objektívne posúdenie tejto práce je potrebné mať dobre otestovaný produkt.

3.5 Zhrnutie

Myslím si, že táto práca je celkom dobrá, ale ešte sa dá teoreticky rozvinúť. Podľa môjho názoru je téma vzdelávacích projektov veľmi široká, takže tento projekt môže byť začiatkom niečoho nového.

4 Záver

V tejto bakalárskej práci som analyzoval existujúce edukačné projekty pre študentov stredných škôl zamerané na propagáciu IoT a spoznávanie jeho základných funkcií.

Na základe tejto analýzy som naplánoval a vypracoval projekt s architektúrou IoT zameraný na jeho implementáciu do učebného kurzu stredných škôl na Slovensku.

Projekt obsahuje základné funkcie typických riešení IoT (pripojenie na internet, sledovanie času, odosielanie údajov na server atď.) Pre jednoduchšiu implementáciu do vzdelávacieho kurzu som pre študentov vytvoril kostru projektu.

Projekt bol otestovaný na vzorke 12 študentov. Napriek tomu, že táto skupina je pomerne malá a projekt neboli úplne rozpracovaný, umožnilo to pochopiť niektoré jeho hlavné nedostatky a pomôže vyhnúť sa ďalším ťažkostiam pri implementácii projektu v kurze.

K projektu bola vytvorená podrobná dokumentácia.

Literatúra

1. *Educational initiative IoT Step by Step 2021*. [online]. [cit. 2023]. Dostupné z : https://en.wikipedia.org/wiki/Internet%5C_of%5C_things.
2. SOMAYYA MADAKAM R. Ramaswamy, Siddharth Tripathi. *Internet of Things (IoT): A Literature Review*. IT Applications Group, National Institute of Industrial Engineering (NITIE), Vihar Lake, Mumbai, India, 2015. Dostupné tiež z: https://www.scirp.org/html/56616%5C_56616.htm.
3. *Základy internetu vecí: lecture 1* [online]. [cit. 2023]. Dostupné z : <https://kurzy.kpi.fei.tuke.sk/iot1/2023/lecture.01.html>.
4. BIŇAS, Miroslav. *Základy internetu vecí* [online]. [cit. 2022]. Dostupné z : <https://kurzy.kpi.fei.tuke.sk/iot1/>.
5. *školenie Internet vecí (IoT) a robotika*. [online]. [cit. 2023]. Dostupné z : <http://www.soseza.sk/milan/akcie/2017IoT.php>.
6. CISCO. *Cisco Networking academy: Introduction to IoT* [online]. [cit. 2023]. Dostupné z : <https://www.netacad.com/courses/iot/introduction-iot>.
7. *Educational initiative IoT Step by Step 2021*. [online]. [cit. 2023]. Dostupné z : <https://csn.chnu.edu.ua/news/iot-step-by-step-2021/>.
8. *Internet of Things Education Package*. [online]. [cit. 2023]. Dostupné z : <https://education.softwareag.com/internet-of-things>.
9. INGEIMAKS, Ingeimaks. *DIY home automation greenhouse* [online]. 2021. [cit. 2023]. Dostupné z : <https://www.hackster.io/Ingeimaks/diy-home-automation-greenhouse-fd82f1>.
10. LANG, Benjamin. *Automated Greenhouse (Arduino)* [online]. 2022. [cit. 2023]. Dostupné z : https://www.youtube.com/watch?v=APs9M0DYuLM&ab_channel=BenjaminLang.
11. *Automated Greenhouse* [online]. [cit. 2023]. Dostupné z : <https://www.instructables.com/Automated-Greenhouse/>.

12. *MicroPython with Arduino Boards* [online]. [cit. 2023]. Dostupné z : <https://docs.arduino.cc/learn/programming/arduino-and-python>.
13. *Cytron Maker Pi Pico Base* [online]. [cit. 2023]. Dostupné z : <https://rpishop.cz/pico-karty/3854-cytron-maker-pi-pico-base-deska-pro-pi-pico-pro-zacatecniky-38515758.html>.
14. PAVLIŠIN, Šimon. *Otvorený IoT Lab pre stredné školy*. 2023.
15. *Breadboard* [online]. [cit. 2023]. Dostupné z : <https://en.wikipedia.org/wiki/Breadboard>.
16. *Raspberry Pi Pico and Pico W* [online]. [cit. 2023]. Dostupné z : <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>.
17. *Raspberry PI Pico W Power Consumption (mA) and How to Reduce It*. [online]. [cit. 2023]. Dostupné z : <https://peppe8o.com/raspberry-pi-pico-w-power-consumption/>.
18. PRESENTS, element14. *Measuring the Raspberry Pi Pico W's Power Consumption - Workbench Wednesdays*. [online]. 2022. [cit. 2023]. Dostupné z : https://www.youtube.com/watch?v=GqmnV_T4yAU&ab_channel=element14presenters.
19. OLUJINMI, TOMISIN. *Raspberry Pi Pico vs. ESP32: Which Microcontroller Is Right for You?* [online]. 2022. [cit. 2023]. Dostupné z : <https://www.makeuseof.com/raspberry-pi-pico-vs-esp32-microcontroller/>.
20. ŠESTÁKOVÁ, Eliška. *Konečný automat* [online]. [cit. 2023]. Dostupné z : https://www.youtube.com/watch?v=C5KerRqrC8c&ab_channel=Eli%C5%A1ka%C5%A0est%C3%A1kov%C3%A1.

Slovník

DHT11 Digital Humidity and Temperature sensor.

GW gateway.

IDE Integrated Development Environment.

IoT Internet of Things.

JSON JavaScript Object Notation.

LDR light dependent resistors.

LED light-emitting diode.

MQTT Message Queuing Telemetry Transport.

Zoznam príloh

Príloha A Štruktúra projektu

A Štruktúra projektu

Z hardvérového hľadiska sa projekt skladá z dvoch častí:

- Meteostanice - zbierajú údaje z prostredia a prenášajú ich do GW. Túto časť vyvíjajú študenti. Každý jednotlivý študent si vyvíja vlastnú meteorologickú stanicu, preto je potrebné pri plánovaní kurzu zabezpečiť, aby boli všetky potrebné moduly pre meteorologické stanice k dispozícii v dostatočnom množstve.
- IoT GW - zariadenie, ktoré sa používa na zhromažďovanie informácií zo všetkých meteorologických staníc jednej študijnej skupiny. Prevádzku tohto zariadenia je potrebné nakonfigurovať pred začatím štúdia študentov.

A.1 Súčasti meteostanice

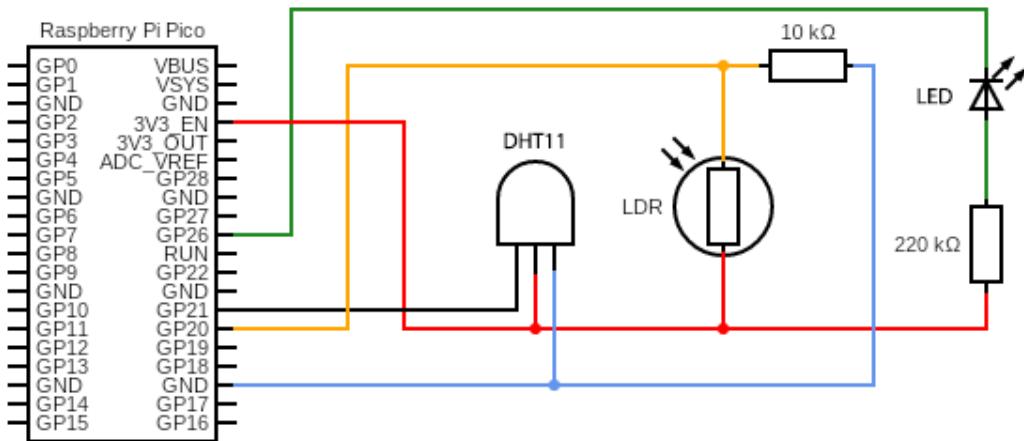
A.1.1 Hardvér meteostanice

Hardvér meteostanice pozostáva z:

- *Raspberry Pi Pico W* - hlavné inteligentné zariadenie, ktorého logiku musia naprogramovať študenti.
- LDR - senzor na meranie osvetlenia.
- DHT11 je senzor na meranie vlhkosti a teploty.
- LED dióda - slúži na signalizáciu problémov.

Všetky komponenty meteorologickej stanice sú pripojené k doske *Raspberry Pi Pico W* pomocou káblov, schéma zapojenia je znázornená na obrázku A.1. Na zjednodušenie pripojenia môžete použiť pomocnú dosku *Cytron Maker Pi Pico Base W*.

V prípade potreby možno dosku *Raspberry Pi Pico W* zameniť na dosku *ESP32*. Teoreticky by s použitím tejto dosky nemali byť žiadne problémy, ale táto verzia nebola úplne otestovaná.



Obr. A.1: Chéma pripojenia k doske Raspberry pi pico W

A.1.2 Softvér meteostanice

Celá logika meteostanice je napísaná v programovacom jazyku *MicroPython* verzie v1.20.0.

Program projektu sa skladá z niekoľkých súborov:

- **start.py** - štartovací súbor celého programu, ktorý obsahuje logiku prechodu z jedného stavu do druhého.
- **states.py** - súbor, ktorý obsahuje logiku jednotlivých stavov.
- **helper.py** - pomocné funkcie.
- **settings.py** - súbor pre systémové premenné.
- **data.txt** - súbor vyrovnávacej pamäte na ukladanie neodoslaných údajov.

Projekt používa aj knižnicu tretej strany s názvom **umqtt.simple** na pripojenie k MQTT brokeru.

Životný cyklus meteostanice pozostáva zo 4 stavov a je popísaný funkciami v kóde aplikácie.

Konkrétnie funkcie:

- **init** - v tejto funkcií sa zariadenie pripojí k WiFi a MQTT a synchronizuje čas. Úspešnosť pripojenia je signalizovaná prostredníctvom LED diódy.
- **measure** - zariadenie meria vlhkosť, teplotu, úroveň svetla a čas merania. Potom sa údaje prevedú do formátu JSON na ďalší export.

- **upload** - zozbierané údaje sa odošlú do MQTT brokera. Ak sa zariadenie nemohlo pripojiť k sieti WiFi alebo k brokeru MQTT, údaje sa archivujú do súboru data.txt a odošlú sa pri ďalších pokusoch na pripojenie k internetu.
- **sleep_pico** - zariadenie sa prepne do energeticky efektívneho režimu na pridelený čas.

Počas školenia študenti dostanú tieto funkcie prázdne a ich úlohou je naprogramovať vyššie uvedenú logiku v rámci týchto funkcií. Učiteľ počas programovania vystupuje ako supervízor a mentor.

A.2 Súčasti IoT brány

A.2.1 Hardvér

GW sa skladá len z jednej časti, konkrétnie z mikropočítača *Raspberry Pi 3*. Môžete použiť aj novšiu verziu, konkrétnie *Raspberry Pi 4*.

Zariadenie musí byť pripojené k internetu prostredníctvom kálového pripojenia alebo cez adaptér WiFi.

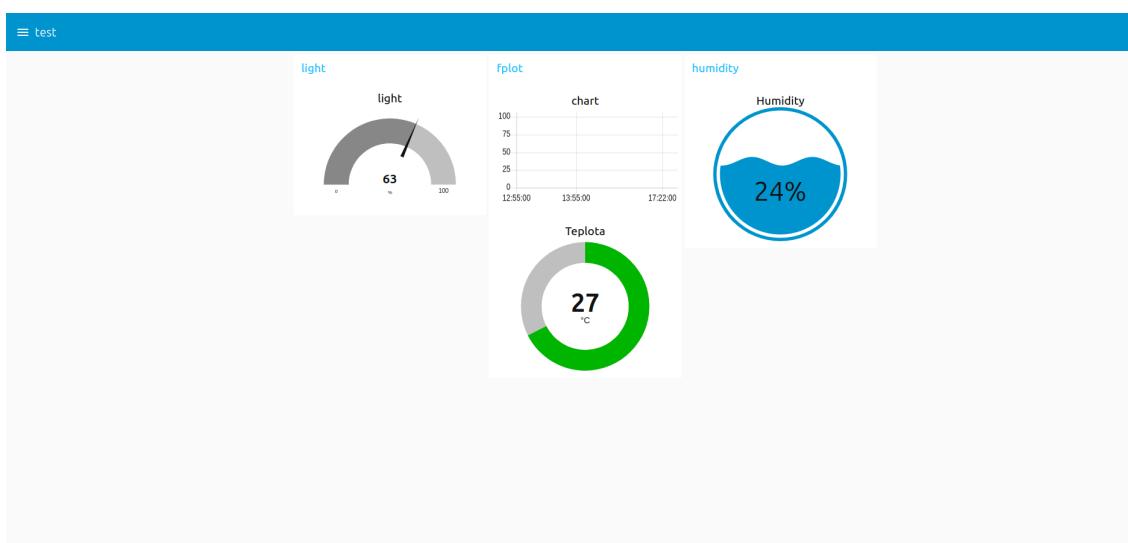
A.2.2 Softvér

GW je založený na technológii Mosquito a má nainštalovaný Docker a *Node Red*. Podrobnejšiu konfiguráciu systému nájdete v bakalárskej práci *Otvorený IoT Lab pre stredné školy*[14].

V tejto fáze vývoja projektu študenti používajú programovací jazyk *Node Red* na analýzu a vizualizáciu údajov odosielaných z meteorologickej stanice. Na obrázku A.2 je znázornený príklad takejto vizualizácie.

A.2.3 Alternatívny postup

Vo všeobecnosti môže ako brána slúžiť akýkoľvek iný počítač (napr. počítač učiteľa), pokiaľ je na ňom nakonfigurovaný MQTT broker (napr. *Hivemq* alebo *Mosquito*) a nainštalovaný Docker a *Node Red*.



Obr. A.2: Vizualizácia aplikácie vytvorennej na *Node Red*.