TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Řídící jednotka mobilního robotu s wifi rozhraním

Bakalářská práce

Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy

Autor práce: **Tomáš Hmiro**
Vedoucí práce: Ing. Miroslav Holada, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Řídící jednotka mobilního robotu s wifi rozhraním

Jméno a příjmení: **Tomáš Hmiro**
Osobní číslo: M17000038
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s mobilním robotem „Pásák“ na pracovišti školitele, seznamte se s wifi modulem ESP32 wroom-32d.
2. Prostudujte možnosti realizace nové řídící jednotky se zaměřením na webový server, wifi sítě, PWM.
3. Robot Pásák upravte a vylepšete jeho vlastnosti.
4. Navrhněte a odlaďte řídící program obvodu tak, aby byly názorně demonstrovány možnosti tohoto obvodu při iteraktivním ovládání robotu Pásák přes webové rozhraní a wifi síť.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	cca 30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. 1. vyd. Praha: BEN – technická literatura, 2005. ISBN80-7300-141-1.

[2] www.esp32.com

Vedoucí práce:	Ing. Miroslav Holada, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	9. října 2019
Předpokládaný termín odevzdání:	18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 18. října 2019

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že texty tištěné verze práce a elektronické verze práce vložené do IS STAG se shodují.

2. 9. 2020 Tomáš Hmiro

Řídicí jednotka mobilního robota s wifi rozhraním

Abstrakt

Tato práce se zabývá mobilním pásovým robotem a jeho řídicí jednotkou, obvodem ESP32 a jeho programováním. Také se zabývá programováním webového rozhraní, použitého pro ovládání, a fyzickou realizací těla a obvodů robota.

Tato práce vznikla jako shrnutí činností provedených na mobilním robotu s pracovním názvem Pásák. Účelem práce bylo celkové vylepšení parametrů robota od vylepšení řídicího modulu z PICAXE na ESP32 přes změnu komunikace z Bluetooth na Wi-Fi a úpravu elektroniky robota v podobě výměny H-můstku a řídicí desky až po samotné softwarové vylepšení, díky kterému je možné ovládat robota přes prohlížeč z libovolného chytrého zařízení, což robota přesouvá do skupiny IoT robotů. V textu jsou popsány některé základní znalosti z použitých odvětví programování (psaní webů v HTML a JavaScriptu, programování ESP32) a komunikací (I^2C, http).

Klíčová slova: TUL, ESP32, HTLM, JavaScript, Wi-Fi, IoT, webový server

Control unit of mobile robot with wifi interface

Abstract

This work deals with mobile robot with crawler belt chassis and it's control unit, circuit ESP32 and programing of it. Next it deals with making of web page, movement control and physical realization of body and circuits of robot.

This work was created as a summary of activities made on mobile robot named Pásák. Purpouse of this job was complete improvement of robot's parameters. Starting at changing of control unit from PICAXE to ESP32 continues with change of communication type from Bluetooth to Wi-Fi and modification in electronic parts like change of H-bridge and motherboard. Ending with software improvements which allow to control robot throught web browser on any smart device. That makes this robot one of IoT robots. There is some fundamental knowledge in the text describing different branches of used programming (web creation in HTML and JavaScript, programming of ESP32) and communication (I^2C, http).

Keywords: TUL, ESP32, HTML, JavaScript, Wi-Fi, IoT, WebServer

Poděkování

Zde bych chtěl poděkovat vedoucímu práce, panu Ing. Miroslavu Holadovi, Ph.D. za časté a rozsáhlé konzultace, které vedly ke zlepšování robota. Panu Ing. Leoši Petržílkovi za výrobu prototypových DPS. Také svým rodičům a hlavně sestře za pomoc se zpracováním práce.

Obsah

Seznam obrázků

Seznam zkratek

AHRS	Attitude and heading reference system, poloha a směr v referenčním systému
AP	Access point, přístupový bod
DPS	Desky plošných spojů
FDM	Fused Deposition Modeling, modelování depozicí taveniny
FFF	Fused Filament Fabrication, výroba z taveného vlákna
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
http	HyperText Transfer Protocol, protokol přenosu hypertextu
MIME	Multipurpose Internet Mail Extension, multifunkční rozšíření pro internetovou poštu
SDK	Software development kit, Systémový vývojový nástroj
SLA	Stereolitografie
STL	Standard Triangle Language, jazyk popisující objekt sítí trojúhelníků
TUL	Technická univerzita v Liberci
URL	Uniform Resource Locator, jednotná adresa zdroje

1 Úvod

Mobilní roboty se řadí do oblasti robotiky, která se zabývá roboty schopnými se pohybovat v daném okolním prostředí. Robotika se zabývá jejich studiem, výzkumem a realizací návrhů konstrukcí a mechanismů. Robot se skládá z různých částí, které se dělí do skupin mechanického, elektrotechnického, řídicího a pohonného inženýrství. Mobilní roboty lze dělit podle různých kritérií, ale mezi základní rozdělení patří autonomní a dálkově řízené.

Autonomní roboty by měly být schopny samostatně vykonávat zadané úlohy jako například přesunout se z bodu A do bodu B podle zadaných pravidel, třeba pomocí sledování barevných čar na podlaze. Také by měly disponovat schopností reagovat na různé eventuální změny v jejich okolí, jako jsou překážky, na které mohou reagovat zastavením, vyhnutím se a vrácením se na původní trasu, popřípadě i upozorněním dispečinku. Mohou se také umět pohybovat ve zcela neznámém prostředí, zmapovat ho, poté se v něm orientovat a například dosáhnout požadovaného cíle. [1]

Dálkově ovládané roboty jsou ovládány operátorem, člověkem, který má většinou vizuální informaci o okolí robota, ať už fyzickou, virtuální, generovanou, nebo z kamer. Ale i dálkově ovládané roboty mohou mít částečně zabudované autonomní chování, například při ztrátě signálu od operátora mohou zastavit nebo se přesunout, aby uvolnily prostor. [1]

Dalším parametrem dělení může být prostředí, ve kterém se robot pohybuje, například po souši, uvnitř nebo venku, pod vodou, ve vzduchu, ve vesmíru, nebo jde o hybridní roboty, pokud se pohybují ve více prostředích. Kritérií, podle kterých se dají roboty dělit, je opravdu mnoho: podle typu podvozku, počtu stupňů volnosti pohybu, počtu kol, zdroje energie a mnoha dalších. [1]

Mobilní roboty mají mnohá využití: počínaje zábavou přes pomoc při dopravě a průzkumu až k záchraně lidských životů. Většina robotů je stavěna na míru nějaké aplikaci, ale ne všichny, některé jsou stavěny za účelem testování schopností a možností jednotlivých dílů nebo jen na výuku. A právě za tímto účelem je vytvořen i mobilní pásový robot, o kterém tato práce pojednává. Tento robot je dálkově ovládaný jeho podvozek je tvořen dvěma pásy, každý je hnán jedním elektrickým motorem a je napájen baterií. Jeho účel je převážně výukový. Většina původních majitelů na této stavebnici zkoušela nějaký komunikační prvek, aby zjistala jeho vlastnosti.

2 Mobilní pásový robot

2.1 Historie robota

Původní stavitelé pásového robota Ondřej Smola (programování) a Jakub Štěpánek (konstruktér) vytvořili v roce 2011 pásového robota, aby vyzkoušeli možnosti, které nabízel mikrokontrolér PICAXE a bezdrátový Bluetooth modul OEMSPA310. Kvůli nedostatkům Bluetooth modulu pro komunikaci s mobilním telefonem, což byl původní cíl, byl v roce 2015 přestavěn studentem Michaelem Klemanem na využití platformy LPCXpresso 1769, která měla vylepšit Bluetooth ovládání a navíc dovolit robotu dělat plynulé zatáčky. [2] Ani toto řešení nebylo konečné, a tak se v roce 2017 vrátil původní mikroprocesor PICAXE a změnila se platforma z telefonu na PC. Ondřej Špetlík poté opravil nefunkční části původního kódu a vytvořil program pro PC určený k ovládání robota, ale zase bez spojitého zatáčení.

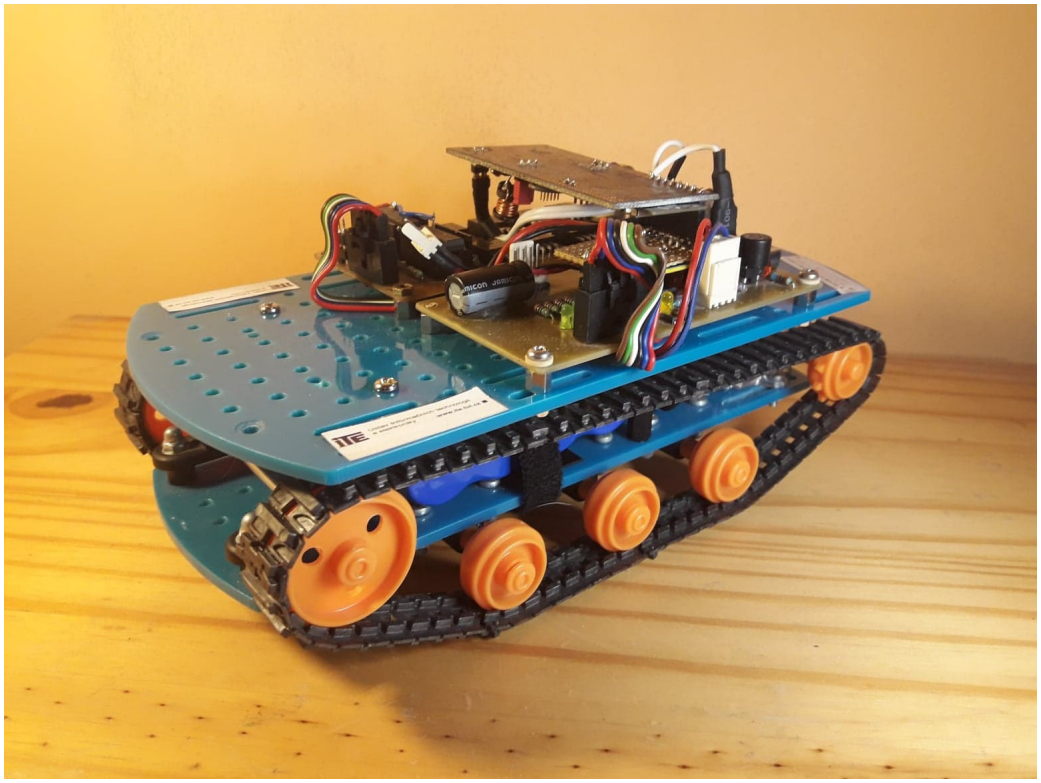
Avšak poté zůstávala limitace ovládání jen pro PC, které má Bluetooth a řídicí program, proto se opět změnila řídicí jednotka, a to na ESP32, které může využívat kromě Bluetooth i Wi-Fi. To by mělo dovolit ovládat robota snáze, protože dnes již skoro každý vlastní nějaké chytré zařízení s přístupem k Wi-Fi a webovému prohlížeči. K prvnímu připojení není potřeba žádné instalování aplikace, stačí se připojit na vnitřní AP robota a je možné ho řídit nebo změnit v nastavení síť, ke které se má připojit, a následně ho ovládat v podstatě stejně, akorát v případě rozsáhlejší sítě i na větší vzdálenosti. Bohužel při větších vzdálenostech začíná být problematické, že operátor nemá vizuální kontakt s robotem. ESP32 navíc disponuje více než jedním kanálem PWM, takže případný spojitý pohyb není v tomto ohledu omezen.

2.2 Úpravy provedené v rámci této bakalářské práce

Bakalářská práce navazuje na projekt, který se zabýval přípravou robota na ovládání pomocí ESP. Bakalářská práce posouvá cíle projektu zase o něco dále.

V části softwarové se jedná o předělání webového rozhraní, jeho rozšíření o Javascript, a změnu způsobu spojení s klientem v prohlížeči. Díky přechodu na asynchronní webserver a použití SPIFFS k ukládání dat by se měla vylepšit zpětná komunikace od robota k operátorovi, který doposud mohl jen ovládat směr, ale vlastně nevěděl, zda se robot hýbe a kam.

V hardwarové části projektu byla upravena kompozice robota, byl v podstatě zcela přestavěn. Byl mu vyměněn starý akumulátor za nový s vyšším napětím. Byl



Obrázek 2.1: Robot na začátku bakalářské práce

přidán spínaný zdroj na řídicí desku a H-můstek byl poupraven, aby se mohl robot lépe pohybovat. Následně v bakalářské práci budou předělány celé obvody řízení, aby byly plně přizpůsobeny 3,3V logice. V dnešní době je velké množství H-můstků již navržených, a tak místo stavění a ladění nového bude starý nahrazen modulem již vyrobeným. Deska s procesorem bude nahrazena novou s ESP32 a spínaným zdrojem.

2.3 Cíle bakalářské práce

Tato práce si dává za cíl vytvořit webové prostředí pro mobilního robota, přes které bude ovládán, a bude možné přes něj měnit některá nastavení. Například měnit práva připojeným zařízením, aby mohl robotem pohybovat jen jeden klient. A také možnost změnit Wi-Fi, ke které se ESP připojí.

V části hardwaru půjde hlavně o další vylepšení jízdních vlastností robota. Přizpůsobení H-můstku a celé logiky na 3,3V by mělo dovolit zvýšení maximálního proudu pro motory pásů, čímž se zvýší rychlost pohybu. Pro vylepšení vzhledu by mohl být vytvořen vnější kryt, jenž by sloužil i ke krytí obvodů, které byly doposud odkryté.

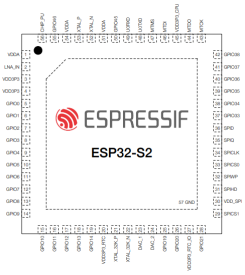
3 ESP32 a výrobce Espressif Systems

Firma Espressif systems je nadnárodní firma založená v dubnu 2008 v Shanghai, Číně. V prosinci 2013, pět let po svém založení, vydala svůj první produkt, a to s čipem ESP8089, 2.4GHz Wi-Fi SoC vyvinutým pro tablety a set-top boxy, zatím jen v provedení integrovaného obvodu. Není tomu ani půl roku, kdy na trh s IoT prorazila s ESP8266EX, nízkospotřebovým, vysoce integrovaným Wi-Fi čipem, který začaly extení firmy osazovat na vývojové moduly. Po roce a půl úspěchů s ESP8266 přišla s novým čipem, a to s ESP32, její vlajkovou lodí, která ji vyzdvihla na první příčky IoT řešení. Opět při výrobě různých vylepšení pro ESP32 nezůstala sama, zapojily se i externí firmy, které vytvářely různé vývojové kity a vylepšení pro modul ESP32. Během roku 2017 otevřela několik nových poboček, jednu dokonce v České Republice. Během následujících let se o své produkty velmi dobře starala a vytvářela podporu pro všechny desky série LyraT, tedy desky podporující Alexa SDK. ESP32 bylo dokonce vybráno německým výzkumným projektem leteckého střediska, které vyvíjejí komerční sondážní rakety, takže se čip od Espressif systems dostal až do vesmíru.[3]

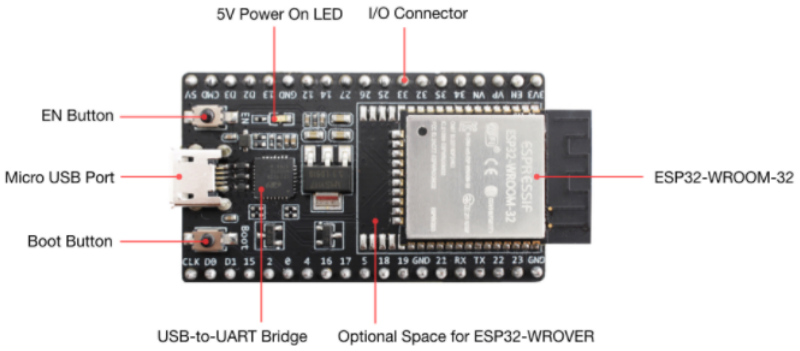
3.1 Varianty a provedení ESP32

Jak bylo řečeno, čipů od firmy Espressif je více a prodávají se v různých formách. Od základních provedení, jako holý čip, přes základní moduly s vyvedenými piny a minimální ochranou, vývojové desky s přípravou k programování přes UART (SparkFun ESP32 Thing, devKit od Olimexu) až k plně vývojovým deskám, které jsou přizpůsobeny určitému účelu. Ne všechna provení jsou přímo od firmy Espressif. Některé firmy se vývoje modulů chopily dříve, například Olimex a SparkFun. Espressif vydal později svůj vlastní vývojový kit pod názvem ESP-DevKit, vlastně základní modul s přizpůsobením na programování přes UART a připojením do nepájivého pole.

Poté však Espressif vydali o trochu lepší model. ESP-Wrover-Kit[3.3] je dražší verze modulu, která má přizpůsobení nejen pro programování přes UART, ale má připraven slot na SD karty, konektor pro připojení kamery a také displej, avšak není již určen k přímému zapojení na nepájivé pole. V podstatě se jedná o jakési napodobení Raspberry Pi. V této práci bylo však potřeba vlastní rozložení a mnohem menší řešení. Tak byl využit modul ESP32 samostatně a DPS byla vytvořena konkrétně pro účely práce. Po krátké úvaze bylo rozhodnuto o použití modulu ESP32-Wroom-32D[3.4].

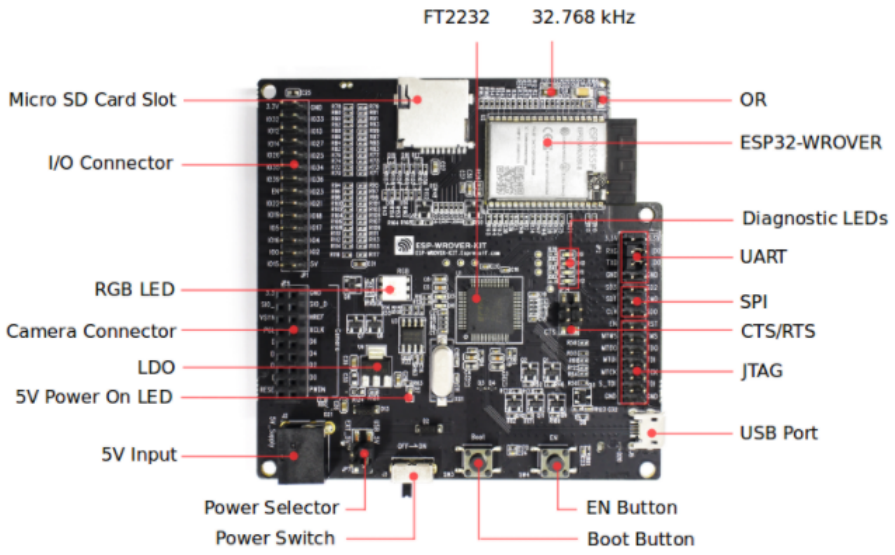


Obrázek 3.1: Znázornění ESP32 jako samostatného SoC [4]



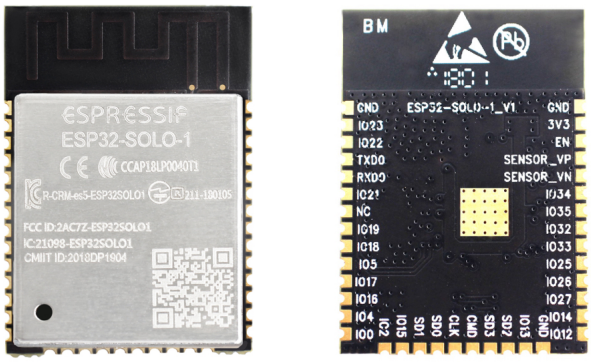
Obrázek 3.2: ESP32-DevKitC v4, konkrétně s čipem ESP32-Wroom-32 [5]

Dále se čipy ESP dělí ještě podle vnitřní stavby, podle toho, jestli mají 1jádrový nebo 2jádrový procesor. Jedná se o mikroprocesory Xtensa LX6, podle značení lze určit, o kterou variantu se jedná. Značení D určuje dvoujádrový nebo dva jednojádrové procesory, označení S vždy jednojádrový procesor. Starší verze Wroom má konkrétně procesory dva po jednom jádře, Wrover má jeden procesor se dvěma jádry a navíc 8MB PSRAM paměti. Další značení doplňují informaci o modulech, například jestli mají integrovanou anténu či nikoli, rychlost a napětí použité paměti, atd.



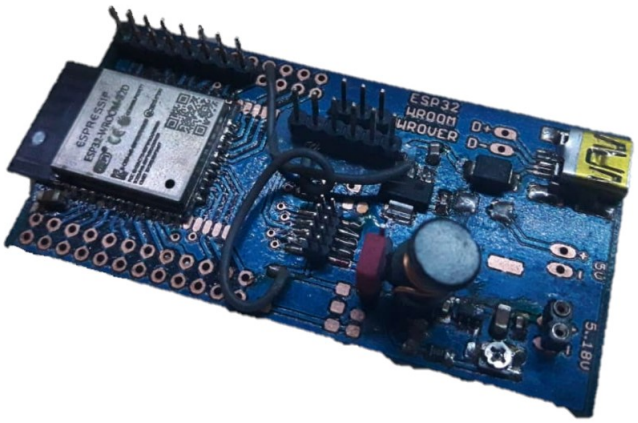
Obrázek 3.3: ESP-Wrover-Kit v4.1 [6]

3.2 Parametry a vlastnosti použitého modulu



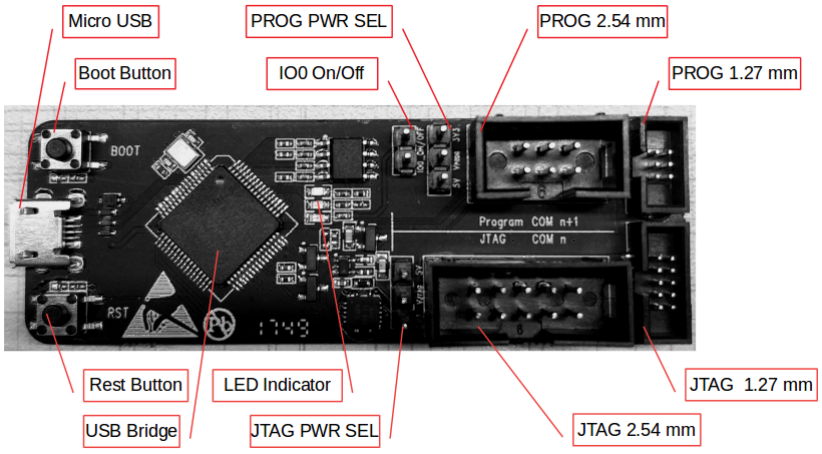
Obrázek 3.5: ESP32, jedna z variant provedení modulu sice jiný typ, ale z vnějšku shodný s použitým[7]

Robot je tedy řízen obvodem ESP32-Wroom-32D, což je modul osazený čipem s dvoujádrovým procesorem Xtensa LX6 s frekvencí 240MHz, Rom pamětí 448KB a SRAM 520KB. Modul má poté 36 GPIO, z toho 22 uživatelsky programovatelných. Analogové vstupy mají 12bitový A/D převodník s nastavením rozlišení. Má



Obrázek 3.4: DPS osazená ESP32-Wroom-32D

vestavěny sběrnice, některé i hardwarově, například SPI, I^2C, I^2S, UART a další. Podporuje komunikaci po Wi-Fi s protokoly 802.11 b/g/n, HT40 s maximální datovou propustností 300 Mbit/s. Také podporuje Bluetooth ve verzi 4.2. Podporuje frekvence v rozsahu 80-240 MHz. Má 512kB SRAM, 4MB flash paměť. Modul běží na 3,3V logice a podle výrobců zvládne celkový proudový odběr až 1100mA. Je tedy velmi robustní k použití v této práci.



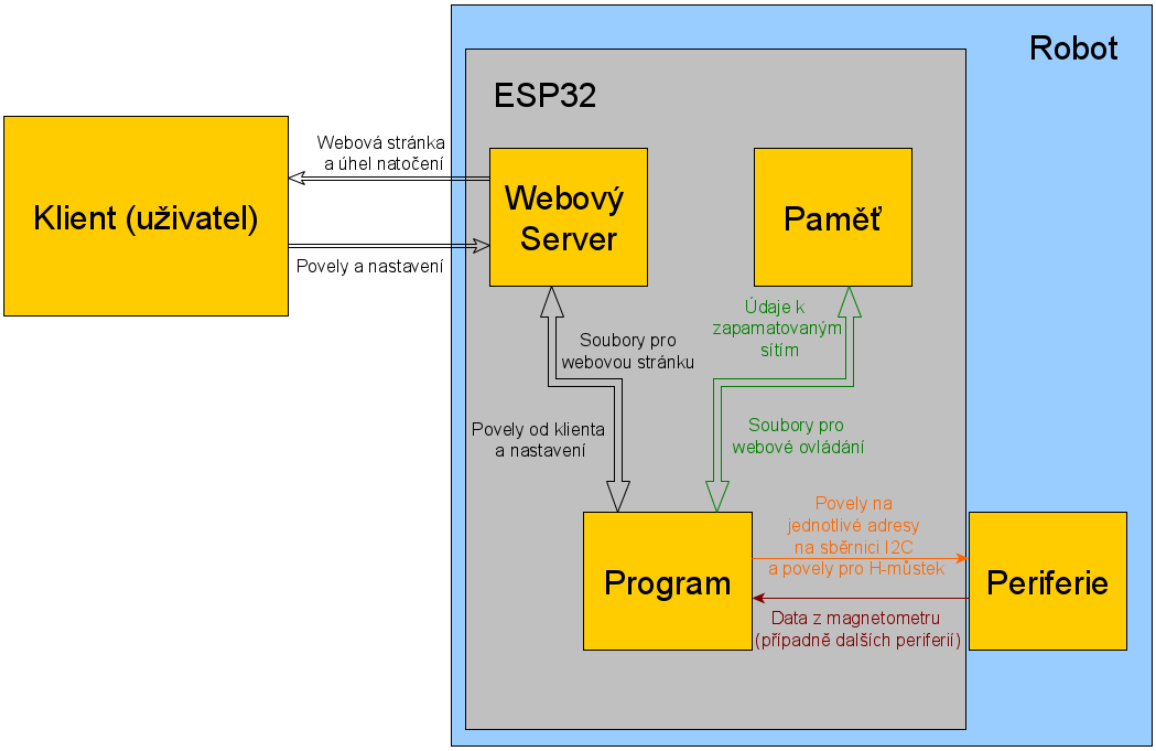
Obrázek 3.6: ESP-Prog, použitý programátor od Espressif Systems [8]

Lze programovat v jakémkoliv prostředí jazyka C/C++, avšak nejlépe podporované jsou ArduinoIDE, Eclipse IDE, Visual C a placené, ale o mnohé funkce rozšířené Visual Micro, rozšíření pro Microsoft Visual Studio. Podporuje samozdřejmě i programování v Assembleru, ale to by bylo pro účely práce zbytečné.

Je tu možnost programování i v jiných vyšších jazycích například, Python nebo Java. V případě Pythonu je možné dokonce využít OTA - Over the Air. Funkce, díky níž můžete programovat zařízení bez nutnosti přímého připojení k PC. Připojení poté probíhá přes síťový port, programátor tím však přijde o možnost kontroly funkčnosti při debugování programu přes sériový port, protože síťový sériový port ještě není plně u této funkce doladěn. V podstatě OTA podporuje jen několik málo zpráv: chybovou zprávu, která může rozlišit o jakou chybu komunikace se jedná, například chyba autentizace hesla ad., zpráva o průběhu nahrávání programu, neboli kolik procent už je nahraných, a zpráva při započetí a ukončení nahrávání programu. Má to však stále své výhody, například je možné programovat více než jedno ESP v čase. Je-li připojeno k jedné síti více modulů, je možné programovat všechny naráz. Důležité je dodat, že program na progrmování přes OTA musí být připraven, není to jen jako u programování přes UART, kde se spojí PC a ESP USB kabelem, a pokud jsou přiraveny ovladače, může se programovat.

Robot v této práci je však jen samostatný prototyp a ladění programu bylo potřebné, proto OTA nebylo využito, ale v budoucnu by mohlo jít o dobrý způsob vylepšování softwarové části robota. Jelikož byl pro práci vybrán „holý“ modul ESP32-Wroom-32D a ne DevKit, není možné programovat jednoduše připojením USB kabelu k modulu, je nutné využít programátor. Výrobce ESP doporučuje v takových případech využít jejich vlastní programátor[3.6], který nabízí dvě možnosti komunikace. První možností je JTAG, který se však z neznámého důvodu nepodařilo uvézt do provozu a to u žádné z prací, které pracovaly s ESP-Prog a modulem ESP32-Wroom-32D. Byla tedy využita možnost druhá, programovací interface s UART převodníkem.

4 Řídicí software robota



Obrázek 4.1: Blokové schéma řízení a komunikace

4.1 Asynchronní webový server

V projektu, který předcházel této práci, byl robot řízen přes „normální“ webový server, jehož obsluha probíhá v nekonečné smyčce, tzv. loop. Obsah odesílaný uživateli do webového prostředí byl psán přímo v kódu velmi nepřehledným způsobem. Výsledek sice nebyl uplně špatný a fungoval, ale vylepšení ve formě samostatných souborů uložených v SPIFFS a asynchoronního webového serveru práci trochu usnadní a hlavně bude výsledný kód a práce s ním mnohem přehlednější a snazší.

Uživatelé ESP, Arduina a dalších jsou velmi aktivní ve vymýšlení nových vylepšení pro mikropočítače, a tak je veliké množství příkladů použití asynchronního webového serveru. Autoři mají velice rozsáhlou dokumentaci na GitHubu této

knihovny [9]. Jeho rozšíření podporuje SPI flash file system, takže je možné vytvořit a následně odesílat celé soubory z paměti klientovi. SPIFFS velmi usnadňuje tvorbu webu, tím že lze vytvořit soubory HTML stránky, jejich CSS soubor, případně JavaScript, a následně ještě pomocí preprocessoru dopňovat proměnné před odesláním konkrétního souboru. V podstatě se dostáváme do bodu, kdy obsluha serveru v ESP skoro odpovídá obsluze serveru kdekoliv jinde. V podstatě se toto rozšíření stará za nás o http protokol, což bylo dříve třeba obstarávat v programu. Nyní jsou jednotlivé soubory potřebné pro vytvoření webového prostředí uloženy v paměti samostatně, lze je lépe upravovat a pomocí nastavení obsluhy je odesílat klientovi.

K základnímu používání Asynchronního webového serveru a SPIFFS je potřeba jen několik příkazů. Je nutné definovat server port, většinou standardně port 80, samozdřejmě nastavit AP nebo připojení k již vytvořené Wi-Fi síti a následně nadefinovat obsluhu příchozích dotazů. Pro připojení ke konkrétní síti je tu funkce z knihovny WiFi.h. V základním zadání *WiFi.begin(SSID, heslo k síti);*. Většinou se ale snažíme ošetřit připojení k síti nějakým rozšířením, proto je většinou rozšířen o kontrolu do sériové linky. U robota však bylo potřeba, aby se v případě, že se nedokáže připojit k síti, kterou chceme, přepojil buď zpět na síť, ke které již připojen byl, nebo aby se minimálně přestal po nějaké době snažit připojit k síti, ke které to nejde. Funkce byla tedy značně rozšířena a následně byla ještě doplněna o kontrolu sítí v okolí a v případě připojení k nějaké síti o její zapamatování. Také je nutné, aby se k robotu dalo přistoupit i v případě, že žádná Wi-Fi v dosahu není, a tak robot vytváří i svůj vlastní přístupový bod (AP), který je odpojen od internetu a slouží pouze k ovládání robota. Není to však primární způsob ovládání robota. Tento přístupový bod se spouští snáze než připojení k Wi-Fi, a to přes funkci *WiFi.softAP(jméno sítě, heslo)*. Posledním nastaveným jménem sítě tohoto AP je ESP32 s heslem PasovyRobot123. Ošetření všech dotazů na webový server je stejné jak pro klienty z AP, tak pro klienty z vnější sítě.

```
AsyncWebServer server(80);

server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
    index(request);
});
```

Obrázek 4.2: Příklad nastavení asychronního webového serveru na dotaz GET index

Pro oblushu webového serveru se používají funkce z knihovny ESPAsyncWebServer.h. K ošetření dotazů je tu funkce *server.on();* viz obrázek [4.2], kde do pole requestu odesíláme odpověď. Request obsahuje větší množství informací, které je možné využít ke čtení informací z prohlížeče, například informace o klientovi a další. V této práci je využito jen několik z nich, k nastavení Wi-Fi je použita metoda getParam, která obsluhuje GET, POST a FILE parametry. Tedy data poslaná klientem zpět webovému serveru. Funkce GET má dvě důležité části, a to jméno (name) prametru a hodnotu (value), která obsahuje informaci, například hodnotu slidebaru nebo text z textového pole. To jsou pro obsluhu robota důležitá data; nejen k tomu, jak se má pohybovat, ale i k jaké Wi-Fi síti se má připojit, nebo které zařízení ho

ovládá. Tato data se dají lehce použít k autentizaci zařízení bez nutnosti zadávat heslo, což by bylo v tomto případě přehnané, protože robot nenese žádná příliš důležitá data. Pokud si nepřidáme obsluhu pro vyvolání dat z paměti přímo do kódu, není žádný rychlý způsob přístupu k paměti, který by nevyžadoval přepsání nebo úpravy kódu. Je tedy dostatečnou autentizací i jen IP adresa, která vlastně slouží hlavně k tomu, aby nedošlo k situaci, že se dva uživatelé na dvou zařízeních hádají o to, kdo robota ovládá. To se však dá snadno obejít tím, že se na jiné zařízení nastaví stejná IP adresa jako původní zařízení uložené jako řidič. To však ničemu nevadí. Bylo by zbytečné zavádět složité autentizace uživatele/řidiče pomocí hesel a účtů, jelikož se jedná jen o prototypového robota a řidičem je zatím vždy jen člověk. Pokud by se ovládání přeneslo na nějakou umělou inteligenci, asi by se musely tyto mezery vyplnit, aby případný narušitel nesabotoval řízení.

Tou asi nejdůležitějších funkcí requestu je funkce send, pomocí níž je odesílána webová stránka. Funkce send má 6 přetížení, ale nejčastěji byla použita dvě z nich. Pomocí prvního je možné odesílat přímo v argumentu funkce napsanou odpověď [4.3], nebo pomocí druhého přetížení jako ve většině případů použití u tohoto robota posílat soubor ze souborového systému.

```
request->send(SPIFFS, "/index.html", String(), false, processor);

request->send(202, "text/html", "<h1>************** Nejsi ridic! ***************</h1>");
```

Obrázek 4.3: Příklad dvou přetížení funkce send

Ve verzi programu pro projekt byl problém využít k obsluze webu JavaScript, tak byl jen velmi povrchně implementován základ php. Avšak díky rozšíření bylo nyní možné používat JavaScript. Pomocí něj se tedy na pozadí webové stránky, asynchronně, odesílají data jako změny směru, při změnách sítě její údaje, změna řidiče a samozřejmě i pohyb, respektive rotace robota. Tato data následně čteme v programu metodou getParam.

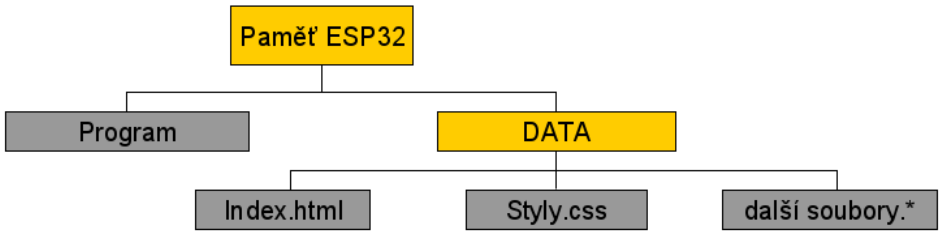
Samotný asynchoronní webový server má ještě spoustu funkcí, které značně přesahují potřebu této práce. Největší nevýhodou je náročnost na procesorový čas, a to i v případě dvoujádrového procesoru, který má EPS32. Jeho funkce se sice snaží zaměstnávat jen jedno jádro, avšak při častých dotazech na obsluhu ze strany klienta způsobují zpomalení reakcí na obsluhu řízení, avšak ne tak značnou, aby se jeho využití nevyplatilo. Pokud by bylo ESP32 použito čistě pro komunikaci s klientem a pro kontrolu gyroskopu a řízení by byl využit nějaký čip navíc, je možné, že by byl program stabilnější a nedocházelo by k přetížení. Problém by se dal řešit například lepším přerozdělením jednotlivých akcí mezi procesory, což ale bohužel v Arduinovském C/C++ příliš nejde.

Místo přidávání dalších procesorů, změny jazyka nebo jiných mikropočítačů postačí snížit počet dotazů, tedy zvýšit čas mezi jednotlivými dotazy. Asynchronnost není řešena jen tím, že máme k dispozici dvě jádra, ale také přes přerušovací systém, který se pak soustředí na vykonávání akcí z přerušení. Je tedy lepší nezatěžovat ESP příliš velkým množstvím dotazů. Například snížení dotazů na úhel natočení kom-

pasu dokonce poslouží jako filtr. Čip gyroskopu je velmi citlivý, stačí malá změna a myšlená střelka kompasu tvořená daty z magnetometru je vychýlena, čímž vzniká veliký šum. Proto i malé ustálení v podobě zpomalení čtení vytvoří vlastně hladší pohyb. Nejvíc je logicky tento šum vidět, když robot nehnutě stojí. Magnetometr je očividně dost ovlivněn i elektromagnetickým rušením, a tak toto „filtrování“ ničemu neuškodí.

4.2 Souborový systém SPIFFS

SPI Flash File System, jedná se vlastně o rozšíření pro uživatelskou/programovou paměť, které dovoluje programátorovi ukládat do paměti soubory. Při nahrávání programu je třeba nastavovat, jak velká část paměti náleží uživateli a jak velká část programu. U ESP8266 se jedná o dodatečné rozšíření, ale u ESP32 je již implementováno a stačí zahrnout potřebné knihovny. Knihovna SPIFFS je knihovna přiložená již v balíčku esp-idf, není tedy nic snažšího, než ji přidat k ostatním používaným knihovnám a začít ji používat. Opět stejně jako většina dalších knihoven obsahuje i příklady použití, ze kterých se dá o knihovně lecos dozvědět, často i bez nutnosti studovat celou dokumentaci. Oproti ArduinoIDE je navíc prostředí Visual Micro připraveno i na tuto možnost, a tak je nahrání souborů otázkou jen několika kliknutí. Data nahrávaná do paměti se nemusí kompilovat, a tak je nahrání dat do SPIFFS mnohonásobně rychlejší, než nahrání programu.



Obrázek 4.4: Grafické znázornění SPIFFS u ESP

V práci je poté SPIFFS využit k uložení webové stránky a jejích podpůrných souborů a také k ukládání dat o sítích, ke kterým se ESP připojuje. Data v této paměti se dají číst, zapisovat, přepisovat, mazat i upravovat. Programátor tomu však musí přizpůsobit program. Mezi přiloženými příklady použití je připraven program na čtení dat v souborovém systému, který poskytuje možnost vypsání dat přes sériovou linku. Stejného principu lze využít i pro vypsání souborů do webové stránky, avšak ne vždy je žádoucí mít všechna data přístupná, například právě citlivá data v podobě SSID a hesel sítí, ke kterým se ESP připojuje. Není vhodné, aby každý, komu se podaří připojit k ovládacímu modulu robota, znal všechna citlivá data. A tak v podstatě stačí, aby dotaz směřovaný do paměti ESP nebyl přidán do obsluhy serveru. V případě, že chce obsluha, řidič, soubor v prohlížeči vyvolat, musí znát jeho název a server musí vědět, co má dělat, pokud se na něj dotazuje. Nejdříve musí být nastavena obsluha serveru, aby při dotazu na nějakou adresu, většinou na adresu

názvu souboru, odeslala soubor z paměti. Je tu také možnost tuto obsluhu přidat k obsluze tzv. Index page, a provádět obsluhu přes metodu getParam. Ta může ESP říct, jaký soubor uživatel hledá, a případně ho může obsloužit. Je tedy prakticky nemožné dostat se do paměti bez předchozí přípravy programu ESP, což je veliká výhoda z pohledu zabezpečení.

Používání SPIFFS je díky knihovnám opravdu snadné. Jako skoro u všech komunikací se musí nejdříve inicializovat, a to pomocí funkce *SPIFFS.begin(true);*, je možné také kontrolovat, jestli se toto spuštění podařilo, pokud SPIFFS.begin(true) vrátí hodnotu false, znamená to, že navázání spojení s pamětí někde selhalo. Pokud k tomu nedojde, je možné začít s pamětí pracovat. Vždy je potřeba soubor nejdříve otevřít. [4.1]

File fname = SPIFFS.open("/nazev souboru", mod); (4.1)

Soubory mohou mít samozdřejmě i přípony, název není vyloženě omezen na anglickou abecedu, ale může dojít k tomu, že se některé znaky špatně přeloží, je tedy lepší psát názvy bez diakritiky. U módu je asi jasné, že jde o volbu mezi čtením, zápisem a rozšířením (FILE_READ, FILE_WRITE, FILE_APPEND). Poté bývá dobré kontrolovat, zda soubor, který má být otevřen, existuje, v případě úspěšného nalezení souboru, je možné číst, zapisovat nebo rozšiřovat již načatý soubor. Zapisovat lze znak po znaku nebo rovnou celé řetězce znaků (string), funkce *fname.print();*, *fname.printf();* nebo *fname.println();*. Číst lze přímo znak po znaku *fname.read();*, po bytech *fname.readBytes();* nebo celé řetězce *fname.readString();*, je také možné nastavit čtecí podmínky přímo, například dokud nenarazíme na nějaké specifické ukončovací znaménko, *fname.readBytesUntil();* a *fname.readStringUntil();*. V programu robota je však využita jen funkce read a print, více zatím není potřeba. V režimu append fungují zapisovací funkce, text se přidává na konec souboru. Na konci každé komunikace, jak čtecí, tak zapisovací, je dobré soubor uzavřít, *fname.close();*. Mazání souborů je snazší, není třeba k souborům přímo přistupovat, stačí jednoduché: *SPIFFS.remove("/název souboru");*, podobným způsobem se dají soubory i přejmenovat, *SPIFFS.rename("/původní název", "/nový název");*

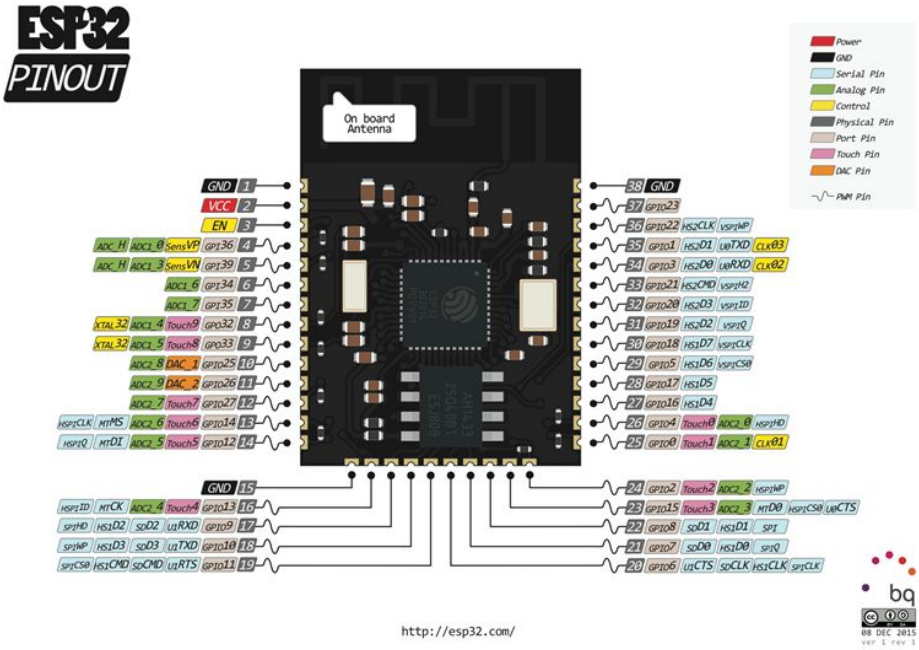
4.3 Řízení pohybu robota

Řízení pohybu je v podstatě jednoduché. Řidič, respektive uživatel, ve webovém ovládání pomocí čtyř směrových tlačítek a stop tlačítka volá na obsluhu jednotlivých směrů a na slidebaru může volit poměr střídy PWM, která ovládá rychlost motorů. JavaScript se stará o asynchronní odesílání povelů k robotovi. Proti tomu program v ESP vyhodnocuje data, která získala obsluha serveru. Tedy ukládá do proměnných změny směru pohybu, nastavení sítě a zároveň čte z I^2C natočení gyroskopu v režimu kompasu, jehož data odesílá na dotaz z webu, který je nastaven časovačem v JavaScriptu. Pravidelně, v nekonečné smyčce, poté mění hodnoty na výstupech, čímž ovládá pohyb.

4.3.1 Regulace rychlosti pomocí PWM

PWM, nebo také pulsně šířková modulace je jedním ze způsobů, jak vytvořit základní D/A převod. Pomocí délky pulsů snížíme celkový přenesený výkon na výstup. U robota je tento princip využit k ovládání rychlosti pohybu pásů.

ESP32 podporuje dva typy PWM. Prvním je LED PWM, softwarové řešení dostupné pomocí funkcí začínajících *ledc...();*, primárně je tato funkce určena, jak je z názvu patrné, k použití na ovládání led diod, ale nic ji neomezuje v použití na ovládání externího H-můstku motorů. Je možné si nadefinovat až 16 kanálů s různým přednastavením PWM a je možné ho spustit na téměř libovolném pinu. Druhým je hardwarové PWM, MCPWM (Motor Control Pulse Width Modulator), které by mělo být určeno k ovládání motorů, ale to lze použít jen na konkrétních pinech, což je velmi omezující, a tak bylo použito softwarové řešení.



Obrázek 4.5: ESP32, rozložení pinů, zde je vidět, které piny lze využít na LED PWM (vlnka před číslem pinu) [10]

Softwarové řešení nabízí 4 časovače, které se mohou přepínat mezi pomalým a rychlým režimem. Procesor tedy nabízí osm vysokorychlostních a osm nízkorychlostních generátorů PWM, kterým lze přidělovat časovače podle potřeby programátora. Vysokorychlostní časovače se skládají z multiplexoru, kterým se volí ze dvou zdrojů hodin. V práci nebylo potřeba rozhodovat, jaký časovač využít, protože základní knihovny obsahují funkce, které tyto potíže řeší automaticky, stačí jen nastavit počáteční parametry.

Před použitím je potřeba všechny tyto parametry nastavit. K tomu slouží následující příkazy. [4.2]

ledcSetup(kanál, frekvence, rozlišení); (4.2)

Kanál je volen v rozmezí od 0 do 15, frekvence je omezena použitým krystalem a rozlišením [4.3], které lze volit v rozmezí 1 až 16 bitů.

$$f_{max} = \frac{f_{osc}}{2^{Res}} \tag{4.3}$$

Poté, co je PWM kanál nastaven, chceme-li ho použít, musíme k němu přiřadit pin [4.4] a střídu v patřičném rozlišení. [4.5]

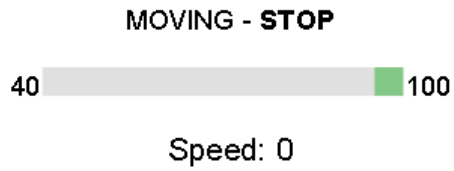
ledcAttachPin(pin , kanál); (4.4)

ledcWrite(kanál, PWM); (4.5)

Pro výslednou funkci je jedno, jestli je nejdříve spojen pin s kanálem, nebo jestli je nastavena střída u kanálu. Obě tyto funkce mohou být používány často ke změnám. Například můžeme plynule měnit střídu nebo přepínat, na kterých pinech je kanál připojen . Pro odpojení pinu od kanálu, ke kterému je připojen, je tu funkce [4.6]:

ledcDetachPin(pin); (4.6)

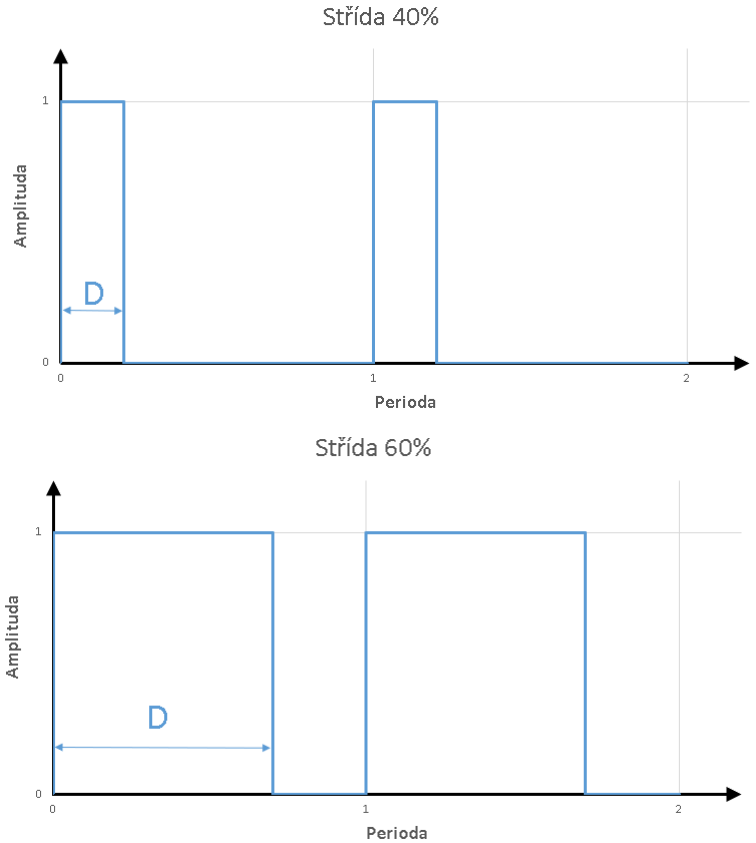
Funkce jsou v práci využity tak, aby tvořily logické kombinace pro H-můstek. Využita je varianta postavená na jednom PWM kanálu, takže v případě, že je potřeba, aby jedna strana stála a druhá se pohybovala, je korespondující pin odpojen od kanálu s PWM a je na něj nastavena logická 0. Postupně do budoucna je možné, že bude program poupraven do formy řízení tanku, tedy dva PWM kanály pro každou stranu, ovládání tak bude více spojité. Tohoto řízení by se opět dalo více využít pro nějaký typ autonomního řízení. Možná bude tedy implementováno pouze pro řídicí jednotku a ne i do uživatelského rozhraní na webové stránce.



Obrázek 4.6: Slidebar k ovládání rychlosti robota

$$
\begin{aligned}
U &= \frac{1}{T}\int_0^T f(t)dt \\
&= \frac{1}{T}\left(\int_0^{DT} U_{max}dt + \int_{DT}^{T} U_{min}dt\right) \\
&= D \cdot U_{max} + (1-D) \cdot U_{min}
\end{aligned} \tag{4.7}
$$

Na stránce webového ovládání se pomocí slidebaru [4.6] volí poměr střídy. Střída lze volit od 40 % do 100 % kvůli tomu, že při střídě menší než 40 % mají motory již příliš malý výkon, než aby se robot pohyboval [4.7]. Spodní hranice 40 % je pouze pro souvislý pohyb, pokud při tomto nastavení robot zastaví, většinou se znovu nerozjede. Pro výpočet napětí na výstupu regulovaném pomocí PWM lze použít vzorec [4.7]. U je ve vzorci napětí, U_{max} a U_{min} jsou maximální a minimální hodnota napětí na výstupu, T je perioda a D je čas z periody, po který je napětí v maximu výchylky. Z tohoto vzorce nakonec vyplývá, že pokud je $U_{min} = 0$, potom je $U = D \cdot U_{max}$. Pro $D = 0,5$ (50% střída) je na výstupu $U = \frac{U_{max}}{2}$. Stejný vzorec platí i pro proud. Pro výsledný výkon stále platí, že $P = U \cdot I$, tedy výkon při 50% střídě je $\frac{1}{4}$ výkonu při 100% střídě. Jsou-li motory na robotu určené na 9 V při méně než 4,5 V, jsou pravděpodobně na hranici své momentové charakteristiky. Otáčky motorů robota jsou přímo úměrné napětí.

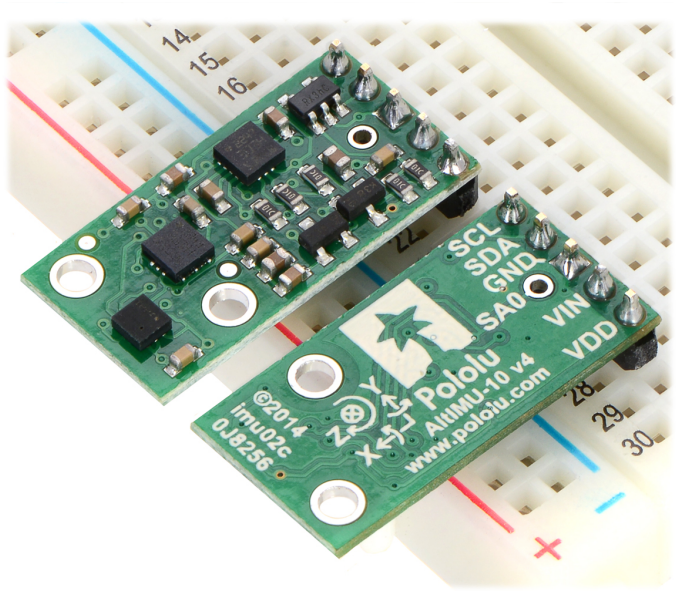


Obrázek 4.7: Grafické znázornění poměru střídy

4.3.2 Obvod AltIMU – kompas

Původně měl robot u pásů připevněny dva snímače otáček, které však byly velmi nepřesné, a to nejen po stránce mechanické, protože odrazky pro senzory se často uvolňovaly, ale i po stránce elektrické, kdy v závislosti na okolním prostředí a rychlosti otáčení někdy prostě nereagovaly.

Proto bylo vhodné tyto senzory nahradit něčím přesnějším, aby se v budoucnu dala plně implementovat odometrie. Jednou z možností byl například nějaký vačkový mechanický systém, který by nebyl závislý na prostředí, nebo magnetický senzor. Také by bylo možné využít nějaký typ lepší světelné závory a rotačního n-kodéru, ale u všech těchto řešení se vždy nakonec ukázal problém s místem, kterého je okolo hřídele převodovky hnaných kol málo.

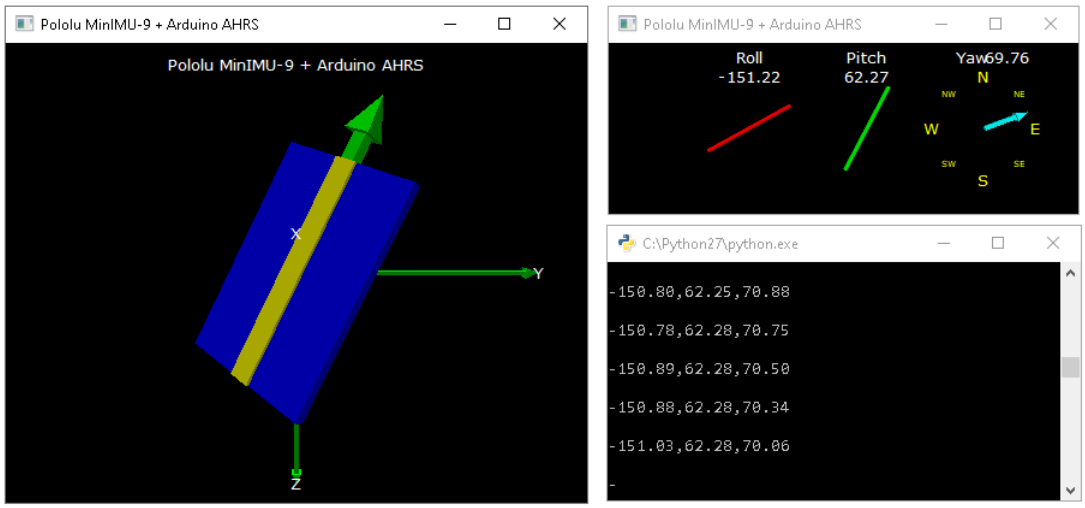


Obrázek 4.8: AltIMU-10 v4 gyroskop, akcelerometr, kompas a tlakoměr od firmy Pololu [11]

Magnetometr byl tedy nakonec asi nejvhodnějším řešením. Není potřeba ho instalovat přímo ke hnaným kolům, navíc je možné ho v budoucnu využít i na přesnější určování polohy. K řešení byl tak použit obvod AltIMU 10 v4 [4.8], který obsahuje gyroskop, akcelerometr, magnetometr a výškoměr. Magnetometr je možné po několika matematických úpravách použít jako kompas. Výrobce, Pololu, přidává na svých webových stránkách kód pro Arduino, který obsahuje většinu matematických úprav potřebných k určování natočení senzorů v prostoru. Obsahuje tedy úpravy pro kompas, gyroskop i akcelerometr, a dokonce přikládává i program v Pythonu pro PC, který vizualizuje polohu celého senzorického tištěného spoje v prostoru [4.9]. Spolu s tlakoměrem je pak možné zjišťovat i výšku, ve které se deska nachází.

Do řešení byl zatím zakomponován jen kompas, a to pouze k zobrazování natočení, do řízení implementován není. Později, nebo v jiné práci by se dal společně s akcelerometrem a gyroskopem využít ke kontrole stavu robota, případně i zapojit

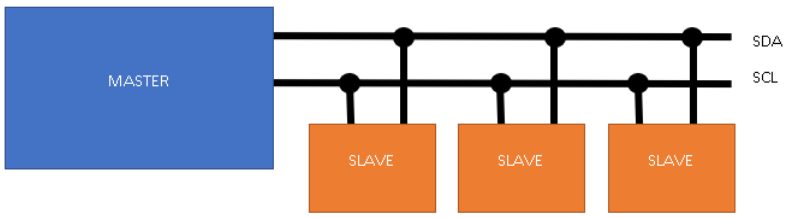
tyto informace přímo do řízení. Zatím se data dají nanejvýše využít k manuálnímu hlídání přímého směru jízdy, ale protože není možné řídit robota bez vizuální kontroly jeho okolí, je to stále jen testovaná nadstavba.



Obrázek 4.9: Vizualizace polohy v prostoru (AHRS)[11]

4.4 Sběrnice I^2C

Ke komunikaci s gyroskopem je využita sběrnice I^2C. Je to sériová dvouvodičová multimasterová sběrnice vyvinutá firmou Philips, původně byla určena ke komunikaci pomalých vnitřních periferií. Je to licencovaná sběrnice, u které se od roku 2006 platí již jen za adresy. Sběrnice je tvořena dvěma vodiči SDA (synchronní data) a SCL (synchronní hodiny). Klidová úroveň sběrnice je v logické „1“, oba vodiče jsou tedy přes odpor připojeny k napájecímu napětí. Každá periferie na sběrnici má svou adresu o délce 7, nebo 10 bitů. Poslední 3 bity přidělené adresy jsou volitelné, aby mohlo být použito více stejných zařízení. Počet připojených zařízení je omezen na 128 a délka vedení je omezena kapacitou vodičů na 400pF kvůli chybám na vedení.



Obrázek 4.10: Ilustrace topologie sběrnice I^2C

Komunikaci zahajuje master startovací podmínkou a ukončuje ji opět master podmínkou ukončovací, tvz. START/STOP podmínka. Logická hodnota na vodi-

či SDA se mění, pouze pokud je SCL v logické „0“, výjimkou jsou jen startovací a ukončovací podmínky. Startovací podmínka je tedy změna SDA z logické „1“ do logické „0“, když je SCL v logické „1“, a ukončovací podmínkou je poté změna SDA z logické „0“ do logické „1“, když je SCL v logické „1“.

Velikou výhodou u programování ESP, nebo jiných modulů na bázi Arduina, je, že pro většinu periferií existují nějaké dodatečné knihovny, díky kterým je komunikace značně zjednodušena. V případě použitého gyroskopu tedy stačí od výrobce, Pololu, stáhnout patřičnou knihovnu, v tomto případě konkrétně knihovnu LSM303.h, přidat ji společně s knihovnou Wire.h k řešení a následná komunikace je již mnohem snazší. Většinu komunikace poté řeší připravená knihovna, stačí jí jen dávat povely. Výrobce s knihovnou dodává i příklady pro Arduino. Je tedy možnost pochopit principy fungování komunikace s modulem AltIMU přímo z funkčního kódu.

Podle dokumentace k ESP jsou pro I^2C připraveny například piny 21 (SDA) a 22 (SCL). Po připojení modulu k SDA a SDC ho stačí již jen připojit k napájení. V programu není potřeba řešit jak komunikovat, stačí již jen použít funkci *Wire.begin();*, která se spojí s připojeným modulem, a následně je možné vyčítat z jednotlivých částí data.

V této práci je využita část z příkladu k matematické úpravě dat, aby bylo možné používat střelku kompasu. U měření úhlu natočení dochází k chybě, a tak otočení modulem o 90° neznamená, že kompas vrátí otočení o 90°. Pokud je však potřeba přesnější měření, vždy se tyto vady dají nějak kompenzovat, popřípadě je možné udělat několik měření a následně kompenzaci chyby vypočíst a přijatá data upravovat. S touto chybou musí zatím řidič počítat, je jen lehce kompenzována v programu. Řešení této kompenzace je sice v principu jednoduché, ale i tak je to rozsáhlé téma a k přesné kompenzaci zatím nedošlo.

O data natočení si pravidelně žádá webový klient. Na webovém ovládání jsou data měření reprezentována jako rotující obrázek tanku chovající se jako kompas, jehož střelkou je kanón. Robot samotný sice kanón nemá, avšak je tanku podobný, takže intuitivním ukazatelem směru je právě kanón.

5 Komunikace mezi robotem a řidičem

5.1 Protokol http

„Protokol trasferu hypertextových informací je aplikační protokol vyvinutý pro distribuované, spolupracující informační systémy používající hypermedia. Jedná se o obecně použitelný, objektově orientovaný protokol. Protokol http definuje soubor pravidel pro přístup k souborům různého charakteru a pro přenos informací. Dovoluje přístup k prostředkům dostupným z různých aplikací, proto zjednodušuje implementaci uživatelského agenta.“[12]

Komunikace pro řízení robota je postavená na http protokolu. Komunikace http je založena na principu požadavků a odpovědí. Http protokol tedy v podstatě spojuje klienta, webový prohlížeč (např.: Mozilla Firefox, Safari, Google Chrome, Opera a další) nebo jiný software, s cílovým http serverem, kterým je v této práci program běžící na ESP32. Požadavky na server se v protokolu http formátují jako text s kódováním ASCII a odpověď od serveru je ve formátu podobném elektronické poště, MIME, což je standard pro formátování textu, který umožňuje e-mailům (ne jen jim) odeslaným přes internet používat znaky nad rámec znaků ASCII.

Základní průběh komunikace vypadá asi tak, že klient, povětšinou webový prohlížeč, získá URL adresu od uživatele, z něj si prohlížeč přes DNS server zjistí cílovou IP adresu, přes protokol TCP naváže spojení a odesílá http žádost. Nejčastěji žádost **GET** („chci“ nebo „dej mi“) a umístění žádaného souboru a nakonec doplní HTTP/1.1, což je verze protokolu, ve které tuto žádost odesílá a očekává v něm odpověď. Server přijme tuto žádost, podívá se do adresáře, o kterém mluví žádost, zkontoluje, v jaké verzi protokolu má odpovědět, podle čehož připojí hlavičku a soubor nebo jinou položku odešle. Hlavička obsahuje stavovou odpověď, hlavičku 1 a 2, může i více, prázdný řádek a požadovaný dokument. Pro HTML dokument by mohla vypadat například [5.1]:

```
HTTP/1.1 200 OK
Content-type: text/html
Date: Fri, 20 March 2020 15:20:15 GMT

<html>
<head>
...
```

(5.1)

Http je bezstavový, informace potřebné ke komunikaci jsou obsaženy v odesílaných zprávách. Server ani klient si data nemusejí ukládat, pokud však chtějí tato data používat, používají k tomu jiné metody, například soubory cookies. Stavy jsou poté většinou hned na prvním řádku hlavičky. První dva řádky jsou skoro vždy v této podobě. Například pokud server nenajde požadovaný cíl, může odpovědět hlavičkou HTTP/1.1 404, což je asi nejznámější chybový kód odeslaný v případě nenalezení požadované položky. Pro položky umístěné jinde, než je uživatel, potažmo klient hledá, jsou určeny chyby 3xx, pro jiné chyby 4xx a 5xx.

Již víme, že všechna data: obrázky, soubory, zvuky, videa a další jsou reprezentována znaky z ASCII tabulky, případně rozšířené ASCII tabulky, a o jaká data se jedná je přidáno do hlavičky odesílaných dat. Viz druhý řádek v příkladu [5.1].

Tato práce používá verzi 1.1, která je zatím asi nejrozšířenější. V provozu je však již i verze 2.0, která je binární a využívá komprimaci a multiplexing ke zrychlení přenosů. Zároveň však pro svou funkci vyžaduje https - tedy http secure, což je zajišťováno certifikáty, které by prý měly být u nových typů ESP také podporovány i hardwarově, tedy ESP by mělo obsahovat paměť navíc určenou jen k tomuto účelu. U stávajících ESP si můžeme tyto certifikáty ukládat na externí médium, protože jinak zabírají místo v paměti. Pro tuto práci však nejsou nijak potřebné, zůstává tedy zatím u starší verze http.

5.2 Jazyk HTML

Http protokol nám pomáhá posílat data, ale čisté poslání dat nestačí k tomu, aby řidič věděl, co to znamená. Je potřeba vytvořit určité rozhraní, pomocí kterého bude operátor robota ovládat a kde může vidět, co robot dělá, zatím pouze zmiňovaný kompas a řádek vypisující typ pohybu. Toho lze dosáhnout pomocí stránky psané v HTML.

HTML (HyperText Markup Language) je programovací jazyk vyšší úrovně používaný k tvorbě webových stránek. Do stejné skupiny se řadí ještě například SGML, XML, WML, VXML. V práci je však z jmenovaných použit hlavně HTML, i když některé součásti ostatních ML již do aktuální verze HTML pronikly. Aktuální verzí jazyka je verze HTML 5.2. Od verze HTML 4 se často označuje jako DHTML (dynamický HTML), protože byl jazyk rozšířen o návrh rámů, kaskádových stylů (CSS) a především o možnost tvorby dynamických webových stránek. Jazyk byl specifikován konsorciem W3C (World Wide Web Consorcium). Jazyk, respektive kód, je tvořen z kódových značek (tagů), klient je poté dekóduje do webového dokumentu. [12]

XML (eXtensible Markup Language) v postatě rozšiřuje HTML o možnost tvořit vlastní značky, čímž rozšiřuje funkčnost webového rozhraní nad rámec HTML. Využití našlo například u webových obchodů.

VRLM (Virtual Reality Modeling Language) rozšiřuje webové rozhraní o prostorové interaktivní simulace, včetně možností scriptování pomocí Javy nebo JavaScriptu.

Java je objektově orientovaný jazyk vyvinutý firmou Sun Microsystems. Pro-

gramy vytvořené v Javě se snaží být nezávislé na systémové platformě. Toho je dosaženo tím, že jazyk je interpretem a programy jsou prováděny systémovými prostředky klienta. Programy psané v Javě se dají dělit do dvou skupin. Java aplikace, programy prováděné interpretem jazyka Java. Java aplety, v podtatě podprogramy, které jsou vloženy do webových stránek. Na rozdíl od aplikací je vykonává webový klient a jsou v mnohém ochuzeny oproti samostatně běžícím aplikacím. Platí tedy, že Java a JavaScript není to samé. I když se to v mnohém podobá, hlavně v názvu.

5.3 Webová stránka

Struktura a zpracování kódu webové stránky se značně liší od kódu v jazyce C/C++, C#, Pythonu a dalších. Výsledný kód není potřeba překládat, lze tedy psát v jakémkoliv textovém editoru. Výhodou použití nějakého speciálního prostředí je například to, že není potřeba psát celé tagy a myslet na to, který je a nebo není párový. V případě této práce byl kód psán opět v Microsoft Visual Studiu s rozšířením Visual Micro. Toto prostředí napomáhá s psaním kódu takzvaným našeptáváním, a to jak u psaní HTML kódu, tak při psaní JavaScriptu a kaskádových stylů.

Psaní webových stránek není nijak složité. Kód se skládá z tagů, kdy některé jsou párové a některé ne. Tagy se píší do ostrých závorek. Každý tag má svůj význam a v některých rozšiřujících jazycích pro psaní webových stránek je i možnost tvorby tagů vlastních (XML). Základ stránky se dělí na dvě části. Uvnitř tagu *<html></html>* se rozdělí na hlavičku *<head></head>* a tělo stránky *<body></body>*. Hlavička obsahuje název stránky, různá nastavení a odkazy na soubory stylů nebo zdrojových kódů skriptů. Tělo potom obsahuje tagy, ze kterých je tvořena zobrazovaná část, obsah webové stránky, případně opět odkazy na zdrojové kódy skriptů nebo i samotné skripty. Každý tag má své rozšiřující vlastnosti, některé jsou společné, například styl, ve kterém se dá nastavit zobrazovaný vzhled. Styly se často upravují pomocí souboru CSS. Pokud chceme, aby byla webová stránka responzivnější, je možné pro změny stylů používat i JavaScript, kterým se dá samozdřejmě vytvářet více než jen změny ve stylech. Pod pojmem styly je myšleno například: velikost písma, barva textu, pozadí stránky, a mnoho dalšího.

Pravděpodobně základními/nejpoužívanějšími tagy (po již jmenovaných) jsou:

- **h1-h6** – nadpisy 1. až 6. úrovně
- **p** – odstavec, opět lze použít více úrovní
- **a** – odkaz
- **div** – oddíl
- **ul** – nečíslovaný seznam

```
<h1>Hello world!</h1>
```

Obrázek 5.1: Ukázka nadpisu první úrovně, kód a jeho zobrazení v prohlížeči Google Chrome

```
<h1 style ="color: blue" >Hello world!</h1>
```

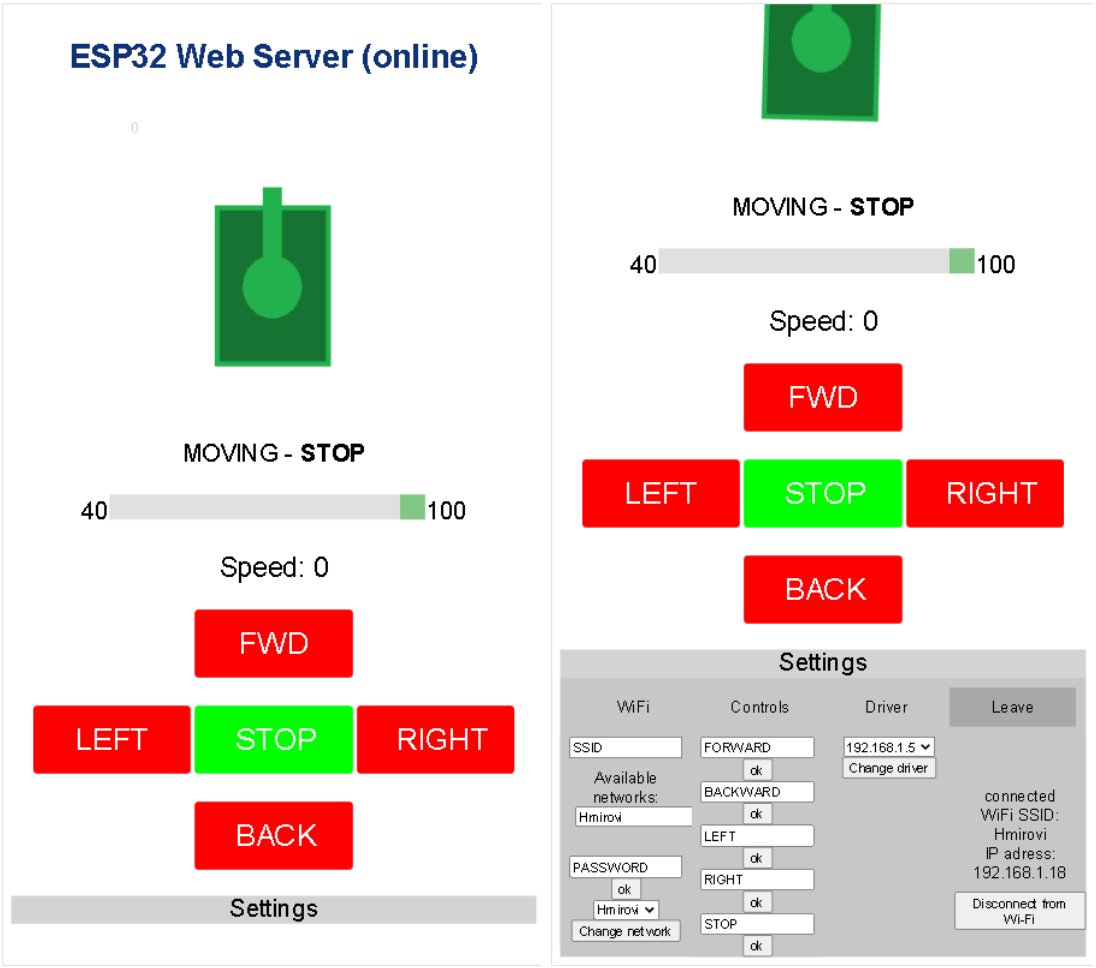
Obrázek 5.2: Ukázka vlastnosti style, kód a jeho zobrazení v prohlížeči Google Chrome

Pomocí do tohoto místa jmenovaných tagů je možné vytvořit základní webovou stránku. Pro stránky s formuláři se vyplatí znát ještě *input, button, submit, select* a *script*. Další tagy jsou většinou spíše pro krásu, ne že by byly všechny méně důležité, ale pravděpodobně nebudou tolik využívané jako ty, jež byly zmíněny. Mezi tagy se potom píše zobrazovaný text, například pro nadpis první úrovně s textem: „Hello world!“ by se zapsal následovně [5.1]. Pro většinu tagů je to velmi podobné. Jednotlivé vlastnosti se poté píší dovnitř uvozovacího tagu [5.2]. Psaní jednoduché textové webové stránky tím poté připomíná psaní dokumentu například v LaTeXu.

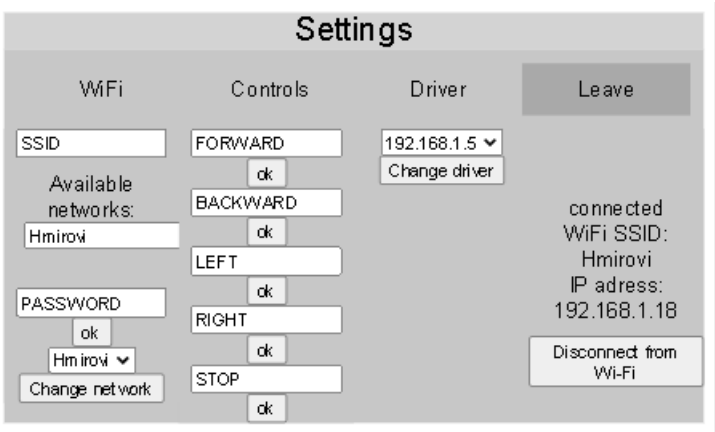
Vlastní stránka vytvořená pro řízení obsahuje několik prvků. Hned pod nadpisem je kompas ukazující natočení vůči výchozí poloze robota. Pod kompasem je „stavový“ řádek, který obsahuje poslední pohybový povel daný robotovi. Pod ním je slidebar k nastavování rychlosti pohybu a ještě níže je vypsána u položky Speed aktuální nastavená rychlost. Po zapnutí robota je u této položky 0, která však znamená neomezenou rychlost. Následují směrová tlačítka, která lze ovládat i pomocí klávesnice. A poslední a nejrozsáhlejší položkou je nastavení [5.4].

V prvním sloupci nastavení se nachází možnost připojit robota k okolní síti, a to přímo zadáním hesla a názvu sítě, nebo volbou ze seznamu zapamatovaných nebo nalezených sítí. V dalším sloupci lze měnit klávesy, kterými je možné ovládat pohyb robota. V položce řidič lze zvolit z připojených zařízení to, kterému chceme

předat práva ovládat robota, nebo stiskem tlačítka „Leave“ (v posledním sloupci) opustit pozici řidiče a přenechat ji dalšímu zařízení, které se připojí nebo připojilo po původním řidiči. V posledním sloupci jsou informace o tom, k jaké síti je robot připojen a na jaké adrese ho v této síti najdeme, tato položka je přidána hlavně kvůli možnosti nastavení robota přes jeho AP. Poslední tlačítko „Disconnect from Wi-Fi odpojí připojenou Wi-Fi a zanechá spuštěné pouze připojení přes AP.



Obrázek 5.3: Ukázka webového rozhraní pro prohlížeč v chytrém telefonu



Obrázek 5.4: Větší náhled na nastavení

5.4 JavaScript

JavaScript byl vyvinut ve společnosti NetScape kolem roku 1996. S původním názvem LiveScript, který byl za nějakou dobu změněn na JavaScript, Java v té době byla poměrně nová a oblíbená, a tak se asi v marketingu společnosti rozhodli trochu toho využít. Java byla využívaná ve webových prohlížečích, ale měla nevýhodu v tom, že se musel nejdříve načíst „plug-in“, což způsobovalo zpomalení v prohlížení, nebyl to jediný problém. V závislosti na problémech Java utrpěla a s ní i JavaScript. Nakonec se však trochu z tohoto spojování vyprostil a začal být opět využívaný. Jeho využití je široké, od prostého rozšíření interaktivy a rozšíření možností vzhledu webových stránek až ke složitým simulacím a grafickému zobrazování. Díky JavaScriptu může například klient kontrolovat data bez nutnosti odesílat je na server, což značně urychluje některé aplikace. Webové stránky navíc mohou být mnohem responzivnější. V roce 1997 se Microsoft a Netscape pustili do spolupráce, z níž vzešla první verze ECMAScriptu. V roce 1999 vyšel standard ECMA-262, což byla poslední verze ECMAScritu. Od té doby všechny webové prohlížeče implementují jednotlivé verze, a jelikož se ECMAScript špatně vyslovuje a nezní to zrovna moc pěkně, uchytilo se jméno JavaScript. Přestože je ECMA-262 standard, tvůrci prohlížečů si ho vykládali a stále vykládají každý trochu jinak, a tak i dnes se může stát, že se setkáme s nekompatibilitou [13].

Nekompatibilita způsobuje, že se stránka v každém prohlížeči chová jinak, může dojít i k plně nefunkčním provedením. Většina webdesignerů poté uvádí, ve kterých prohlížečích je jejich web plně funkční. Ovládání pásového robota neboli webová stránka, přes kterou lze robota řídit, byla primárně testována na prohlížeči Google Chrome v PC, na ostatních prohlížečích testována nebyla, je tedy možné, že nebude kompatibilní.

JavaScript má oproti některým jiným programovacím jazykům dvě zásadní výhody. Za prvé nemusí se kompilovat, takže jde opět psát kdekoliv. A za druhé používá primárně jen dva typy proměnných, *var* jako stálou proměnnou a *let* jako dočasnou proměnnou, která se často využívá například v cyklech. Do obou typů proměnných

lze zapsat vlastně cokoliv a není nutné řešit, jestli ukládáte do proměnné číslo celé nebo s desetinou čárkou, nebo jestli do ní ukládáte řetězec. Kvůli tomu však může dojít i ke špatné interpretaci. Ku příkladu je veliký rozdíl mezi tím, jestli číslem násobíte textový řetězec, nebo jestli násobíte čísla mezi sebou. Je tedy i tak důležité vědět, jak jednotlivé proměnné zapsat.

Ve vytvořené ovládací webové stránce však není implementován JavaScript kvůli počítání. Jeho hlavní funkcí je komunikace se serverem. K té je využit AJAX (Asynchronnous JavaScript and XML). AJAX kombinuje JavaScript s přístupem k serveru. V podstatě dovoluje klientovi nechat uživatele dále pracovat, zatímco klient vyřizuje komunikaci se serverem. AJAX je na ovládacím webu například nastaven, aby se každých x sekund dotazoval na úhel natočení robota. Původně bylo využito i jQuery, později se však ukázalo, že jQuery není dobrá volba v případech, kdy se má robot ovládat přes zařízení připojené na AP (bez přístupu k internetu), protože je tato knihovna moc velká na uložení do paměti, a pokud není v paměti a chceme využít její interentovou variantu, je bez přístupu k síti nedosažitelná. Byla tedy nakonec využita jen část AJAXu obsahující XMLhttpRequest. Jeho součástí jsou právě funkce, díky kterým lze na pozadí webové stránky komunikovat se serverem.

Samozřejmě je JavaScript využit i ke zlepšení dynamiky webové stránky. V JavaScriptu je vytvořena změna barvy tlačítek řízení, otevírání bloku nastavení, potvrzování změn na slidebaru, potvrzování formulářů v nastavení či rotace „střelky“ kompasu. Také je pro počítač přidána funkce eventů klávesnice, je tedy možné ovládat robota pomocí klávesnice.

Hlavním důvodem implementace JavaScriptu ve webovém ovládání je ale právě posílání dat zpět webovému serveru na ESP.

Pro efektivní použití JavaScriptu je dobré do tagu přidat parametr *Id* a tag si tak označit. Následně je možné si v kódu JavaScriptu daný element uložit do proměnné *var proměnná = document.getElementById(”Id”);*, pak je možné přistupovat k jednotlivým atributům elementu a jejich změnami nastavovat jeho vlastnosti, například je možné mu měnit barvy *proměnná.style.color = ”#00ff00”* (změna barvy na zelenou). To je kupříkladu využito u ovládacích tlačítek, která mění svou barvu v závislosti na zvoleném směru. Obdobným způsobem se dá přistupovat k textu v textových polích a dají se tak odesílat uživatelem vyplněná data na server. K odesílání je tedy použit AJAX a jeho součást XMLhttpRequest.

Nejdříve si vytvoříme instanci objektu *var xhttp = new XMLHtmlRequest();* přes tuto instanci poté voláme jednotlivé metody. Hlavní použitou metodou je *xhttp.open(metoda otevření nejčastěji ”GET” , url cíle, asynchronní/synchronní komunikace);* . Poté stačí jen zavolat metodu pro odeslaní dotazu k serveru *xhttp.send();*. Pokud je očekávána odpověď, je nutné ji číst v čase, kdy dorazí, a k tomu jsou připraveny metody a eventy kontrolující stav komunikace. Event *xhttp.onreadystatechange* se vyvolá pokaždé, když se změní stav komunikace uložený ve vlastnosti *xhttp.readyState*. Ten má pět stavů. První stav je nastaven po nastavení metody open a má hodnotu 0. Druhý má hodnotu 1 a odpovídá mu zavolání metody send. Další s hodnotou 2 odpovídá stavu po přijetí hlavičky, ale ještě před přenosem celé vyžádané informace. Stav s hodnotou 3 odpovídá načítání při-

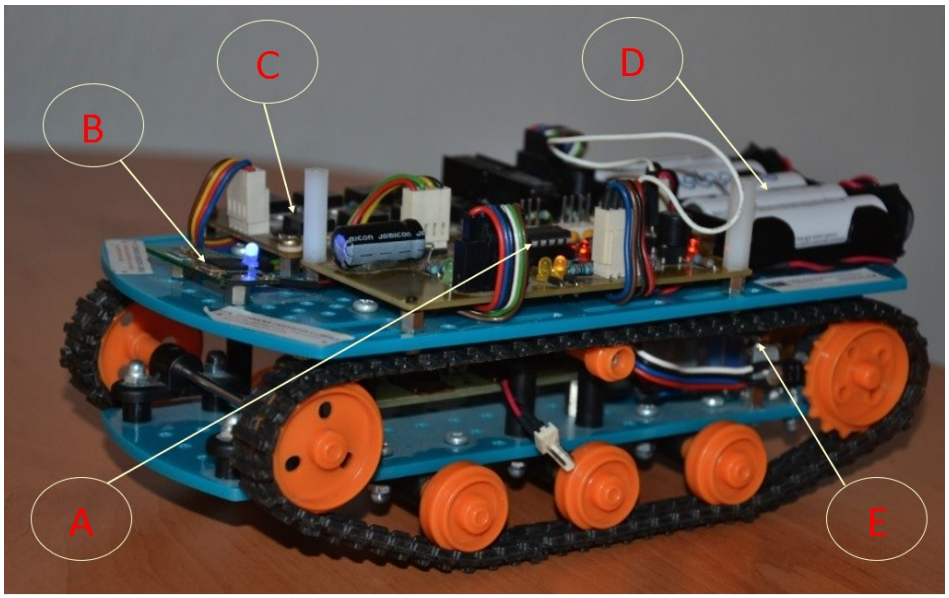
jímaných dat a poslední stav s hodnotou 4 je úspěšně dokončená komunikace. Po odeslání požadavku pomocí metody send ESP reaguje obsluhou webserveru a odpoví, co má nastvano jako reakci na dotaz na zadanou adresu. Pro pohyb reaguje pouze uvnitř ESP a zpět neposílá údaj o úspěchu či neúspěchu, aby se zvýšila rychlost komunikace. Pokud se však klient dotazuje na natočení robota, odesílá mu zpět úhel natočení ve formě textu. Tato přijatá hodnota je ve formě malého čísla vložena ke kompasu, aby měl řidič možnost vidět s větší přesností natočení. Tuto hodnotu natočení následně script přepočte a pootočí střelku kompasu.

Schopnosti JavaScriptu jsou mnohonásobně větší, ale v práci jsou využity jen základní funkce a příkazy, hlavně z důvodů omezené paměti ESP. Pokud ESP pracuje jen v režimu AP, připojený klient nemá přístup k internetu, a tedy i k většině rozšiřujících funkcí JavaScriptu.

Otáčení střelky kompasu je zprostředkováno asi nejsložitější funkcí, která je na stránce použita, ale i tak je poměrně jednoduchá. Nedříve si vytvoříme v HTML stránce canvas, „prostředí“ pro render 2D a 3D obrazů a načteme si soubory obrázků *obrázek = new Image(); obrázek.src = ”soubor-obrázku.png”;* , kterými následně budeme otáčet a po dokončení načtení obrázku spustíme volání funkce pro překreslování *obrázek.onload(){setInterval(překreslení, čas v milisekundách);}*. Překreslování probíhá tak, že se nejdříve vyčistí canvas, *ctx.clearRect(0,0,200,200);*, tím, že překreslí obdélník o rozměrech 200 na 200 pixelů s počátkem v bodě [0,0]. Uložíme si výchozí transformaci *ctx.save();*, abychom mohli vždy otáčet vůči počátku a ne vůči poslední pozici. Posuneme si počátek o 100x100 pixelů (polovina rozlišení obrázku) *ctx.translate(100,100);*, pootočíme souřadnicový systém o přečtený úhel *ctx.rotate(úhel * (Math.PI/ 180)*, vezmeme obrázek a vložíme ho posunutý opačně vůči posunu souřadnic o 100x100 pixelů *ctx.drawImage(obrázek, -100,-100);* , aby se vložil na střed, a nakonec obnovíme původní transformaci *ctx.restore();*. Pokaždé, když se klient zeptá na natočení robota a dostane odpověď, přepíše JavaScript hodnotu natočení. Díky pravidelnému volání překreslení se obrázek točí.

6 Hardwarové úpravy a vylepšení

Během práce se kromě programu ladil a upravoval také samotný robot. Stavebnice, ze které je postaven, umožňuje změny stavby robota, ale tím také dovoluje dělat při jeho stavbě chyby. Chyby, které udělal původní vlastník, bylo potřeba opravit, a to se stalo již během projektu, ze kterého práce vychází. Problémem bylo špatné umístění baterií, které způsobovaly, že těžiště bylo v zadní části robota. Baterie byly umístěny nad zadní nápravou, kde jsou zároveň i motory a převody k hnaným kolům [6.1 D,E]. Na obrázku [6.1] je patrné, že má robot vpředu připevněn Bluetooth modul [6.1 B] a za ním jsou připevněny řídicí deska [6.1 A] a starý H-můstek [6.1 C].



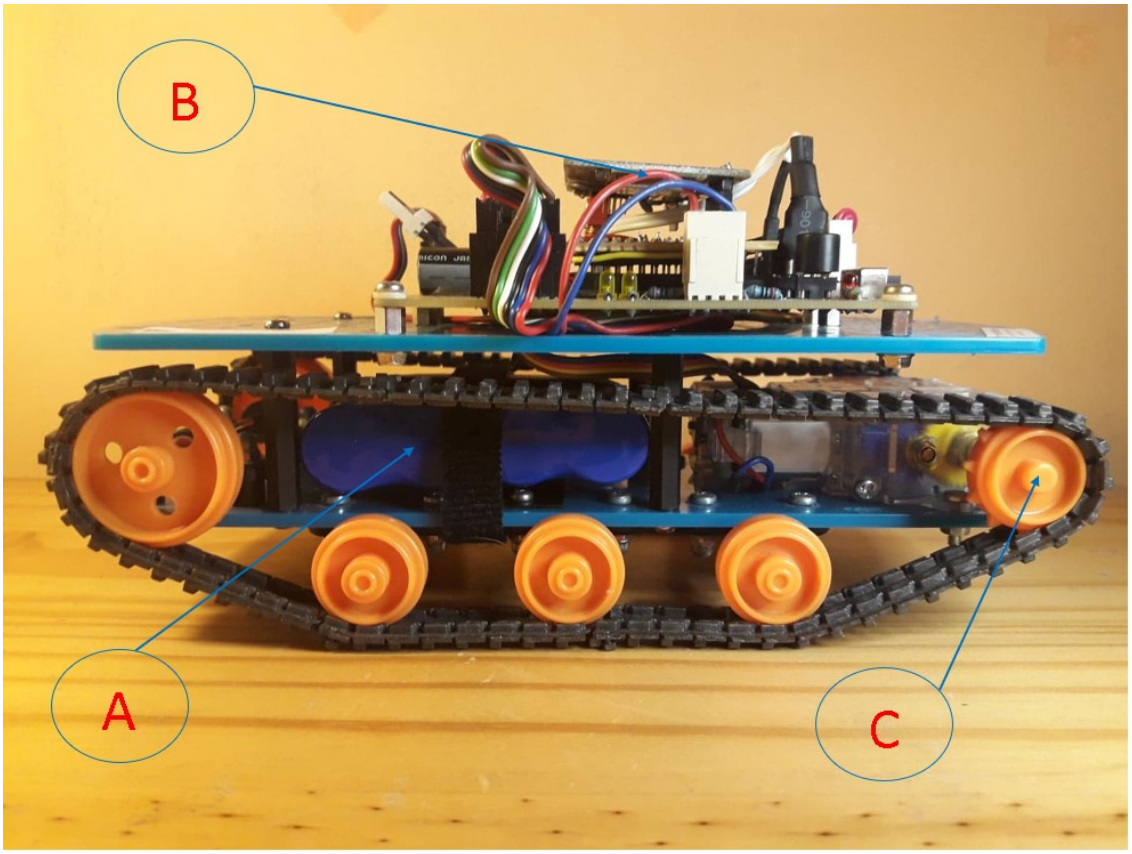
Obrázek 6.1: Původní sestava ještě před projektem

Při přestavbě bylo nutno robota rozebrat, a tehdy se přišlo na to, že se zadní hnaná kola protáčí. Přestavba si tedy vyžádala nahrazení kol, která nyní byla mnohem menší. Menšími koly robot sice získal na síle, ale pohyboval se mnohem pomaleji. Navíc se ukázalo, že rozteč náprav byla u stavebnice fixně nastavena a nebylo možné hýbat nápravami tak, aby byly napnuté pásy. Musel se tedy ještě řešit problém, jak umístit nápravu tak, aby byly pásy stále napnuté [6.2 C]. Po úpravě původní stavebnice to již nebyl problém, ale robot přišel o svou rychlost ztrátou původních velkých kol.

Na obrázku [6.2] je tedy vidět, že nejtěžší součásti (baterie a motory s převodov-

kami [6.2 A]) jsou umístěny ve spodním patře a na vrchu robota se nachází již jen řídicí deska a starý H-můstek[6.2 B].

Práce si tedy mimo jiné dala za úkol rychlost mu navrátit. Kromě již zmíněného nového H-můstku bylo vhodné nahradit malá kola. Tehdy přišel návrh na 3D tisk.



Obrázek 6.2: Sestava na konci projektu s malými hnanými koly a přesunutým těžištěm

6.1 Využití 3D tisku

Jelikož sám robot je sestaven za účelem testování některých technologií, na začátku se jednalo o Bluetooth, teď spíše o testování komunikace Wi-Fi zprostředkované modulem ESP32 a testy modulu jako takového, hodilo se vyzkoušet i některé technologie na tvorbu kostry samotného robota a při této příležitosti vyzkoušet schopnosti domácí 3D tiskárny z vlastních zdrojů. Testovanou tiskárnou se tedy stal Ender 3 [6.3] od čínského výrobce Creality. Tato tiskárna je postavena po vzoru tiskáren Josefa Průši, konkrétně je inspirovaná modelem Prusa i3.

3D tisk je v průmyslu dostupný už od 80. let 20. století, ale mezi širokou veřejnost se propracoval až v roce 2009. Aby mohl být využit širokou veřejností, bylo třeba navrhnout 3D tiskárnu, která bude levná na výrobu a zároveň dostatečně přesná, a to

Obrázek 6.3: 3D tiskárna Ender 3 [14]

zároveň s tím, že její autor si patent nenechá jen pro sebe, ale nechá ho pod volnou licencí (creative common nebo open source). Návrh první takovéto tiskárny začal v roce 2005 v Anglii na Univerzitě v Bath. Doktor Adrianme Bowyer tehdy začal s nápadem na RepRap (Replicating Rapid Prototyper neboli rychle se replikující se prototypér). Na projektu RepRap se poté podílelo mnoho návrhářů a kodérů z celého světa, kteří dali dohromady něco, co je nyní základem většiny veřejnosti dostupných 3D tiskáren. Sám český velikán v 3D tisku Josef Průša se drží stejného nápadu na tiskárnu, která je zčásti sebereplikovatelná. Z původních cen okolo 20 tisíc amerických dolarů, klesla cena veřejně dostupných tiskáren dnes již na cenu někdy i 100krát nižší. Ceny dále klesají a tiskárny jsou stále čatěji používány, a to i ve školních projektech a bakalářských pracích [15].

6.1.1 Typy stavby 3D tiskáren

V rámci RepRap proběhl vývoj takzvané „additivní metody“, tedy výroby výsledného objektu pomocí přidávání materiálu, opakem je obrábění, při kterém je z bloku materiálu obroben výsledný tvar. Konkrétně šlo o metodu označovanou jako FFF/FDM (Fused Filament Fabrication/ Fused Deposition Modeling), není to jediná metoda, ale je jednou z nejvíce rozšířených. Tato metoda je jednou z nejsnazších. Do ohřáté trysky je přiváděno vlákno většinou o průměru 1,75 mm a tryska následně svým pohybem nanaší jednotlivé vrstvy, čímž tvoří výsledný objekt. K jejímu fungování je tedy potřeba minimálně tří motorů pro tříosý prostorový pohyb a pak jeden pro každou trysku na dávkování vlákna. Tedy z technického hlediska asi principielně nejméně složité řešení. Tímto způsobem jde poté tento princip rozšířit na skoro jakýkoliv materiál. Pro příklad jiných materiálů lze uvést beton (v poslední době je to jeden z rychlých způsobů budování malých rychlých domků), kov (svářeným způsobem), různé polymery a plasty. Takovéto tiskárny, jakékoliv velikosti, mohou mít různá uspářádání své konstrukce. U stolních to bývá řešeno většinou

jako kartézský manipulátor, pohyb je v každé ose lineární. U větších tiskáren se začaly využívat i angulární roboty, protože mají lepší rozsah pohybu a zabírají menší plochy. Tisková plocha u FDM tiskáren je omezena jen konstrukcí mechanismu, který vede trysku. Jinak je možné tisknout například na posuvný pás a tím docílit „nekonečného“ prostoru pro tisk. [15]

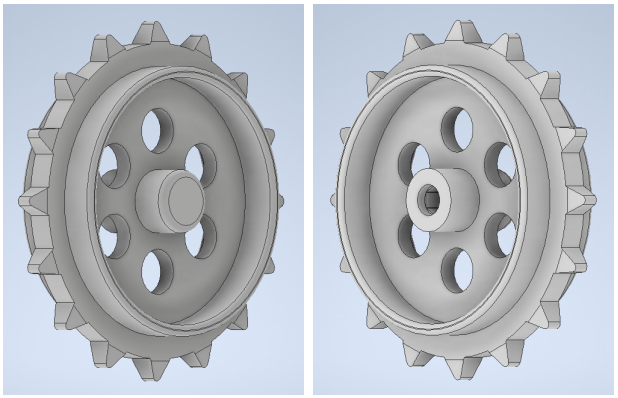
Další, poměrně hodně rozšířenou metodou je metoda SLA (stereolitografie), zde je průběh tisku o něco jinačí. Hlavním rozdílem je, že je tisk prováděn do tekutého materiálu, resinu - tekutého polymeru, a výtisk je postupně vytahován z nádobky. Tento tisk se dá rozdělit na další podkategorie, podle technologie vytvrzování vrstvy. První z nich by mohla být laserová, která postupně směrovaným laserem o určité vlnové délce (většinou UV) vytvrzuje vrstvu za vrstvou, takže se vlastně moc od FDM metody neliší. Nejsložitější na této metodě je přesné směrování laserového paprsku. Druhou metodou je selektivní osvicování projektorem, kdy se vytvrzuje celá vrstva najednou, ale je potřeba dobrá čočka, což může být drahé. Třetí, zatím poslední metodou je osvicování UV světlem skrze LCD panel. U této metody je kvalita nejvíce ovlivněna rozlišením LCD panelu, u kterého roste cena s rozlišením. [16]

Všechny metody mají společné, že výtisk je vytahován z nádržky, vytahovaný materiál se vytvrzuje zespodu a k nádržce se při tvrdnutí přilepí, čímž také vzniká nejvíce potíží. SLA je jinak asi nejpřesnější metoda 3D tisku a pokud se osvicuje celá vrstva najednou, je také jednou z rychlejších metod. K vytahování je potřebný jen jeden motor, ale oproti FDM má mnohonásobně dražší materiál, ze kterého se tiskne, a to asi pětinásobně, zároveň je také omezena velikost tiskové plochy, protože potřebujeme nádrž na resin tak velikou, aby se do ní vešel celý model, u tohoto tisku posuvný pás nepřipadá v úvahu, alespoň zatím. Podobným způsobem jako je SLA se dá tisknout například i z kovového prachu.

6.1.2 Navrhování modelů

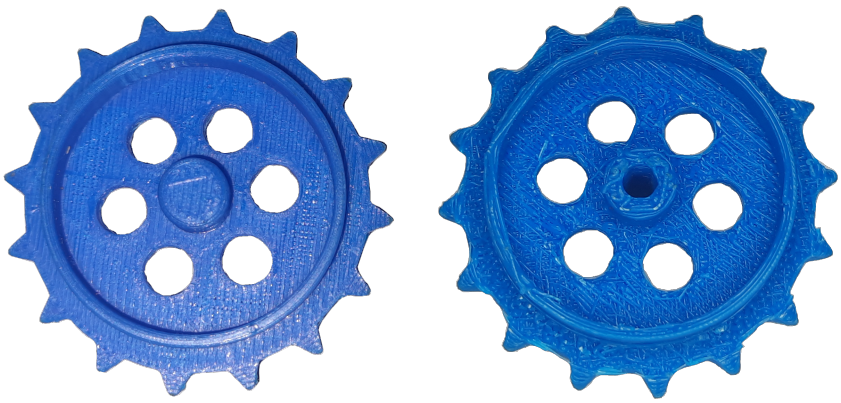
Technická univerzita spolupracuje se společností Autodesk, a tak byl vybírán software z jejich nabídky, který je dostupný pro studenty. Z několika různých návrhových prostředí, která jsou od Autodesku dostupná, byl vybrán Autodesk Inventor Profesional 2020. Návrhová prostředí jednotlivých CAD programů jsou si velmi podobná a většinou se jedná jen o malé rozdíly mezi jednotlivými systémy. Tento program je poměrně připravený na návrh modelů pro 3D tisk a jeho ovládání relativně přehledné, takže nebylo potřeba uchylovat se k hledání softwaru jiného vydavatele. V prostředí Inventoru je možnost přímého tisku nebo uložení vymodelované součásti do souboru stl, model tvořený sítí trojúhelníků. Pro přímý tisk by bylo potřeba nastavit všechny potřebné parametry tiskárny, což u programu dodávaného k tiskárně je již od výrobce vyplněno, a tak byl využit program dodávaný k tiskárnám od firmy Creality. Ve firmě Creality poupravili program Cura a značně ho zredukovali, aby byl více přehledný a přizpůsobený začátečníkům ve 3D tisku. Původní Cura obsahuje veliké množství předvoleb a je k jejímu využití dobré znát již nějaké začátky.

Návrh kol [6.4] byl velmi jednoduchý hlavně díky faktu, že bylo možné předem

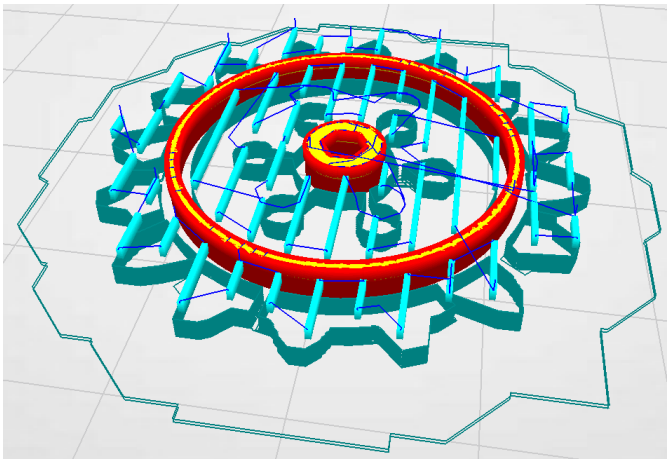


Obrázek 6.4: Návrh nových kol

změřit původní kola. Avšak při tisku i přes dobrou přesnost tiskárny(v osách x a y 0,1 mm a v ose z 0,05 mm) vždy může dojít k nepřesnostem, je třeba s tím při návrhu počítat. U takto malých rozměrů dochází většinou k tomu, že se plast při tisku rozteče o trochu víc, než je velikost trysky, a tak poté bývají takto vytvořené objekty o trochu širší respektive užší v případě děr. K tomuto jevu dochází, protože horký plast potřebuje nějaký čas tuhnout, ale u malých modelů tryska nanáší během chladnutí již další vrstvu a předchozí ještě nemusí být dostatečně vytvrzená. Je možné tento problém řešit omezením času na jednotlivé vrstvy, kdy se nastaví čas tisku na vrstvu, a pokud je vrsva vytištěna rychleji, je tisk pozastaven, dokud neuběhne nastavený čas. Při tomto řešení však vznikají další problémy. Například se může stát, že vrstva vytuhne naopak moc, a tím je poté snížena pevnost splynutí jednotlivých vrstev. Nebo jelikož čekání neprobíhá v dotyku s modelem, ale tryska je nadzvižena, může dojít ke standartní chybě krokových motorů a projeví se opakovatelnost tisku, kdy se tryska netrefí na správné místo, model je tak křivý. Některé metody byly při tisku vyzkoušeny, ale nakonec bylo nejvhodnější počítat s původní nepřesností, která bývá většinou v mezích $\pm 0,05$ milimetru, avšak i tak může dojít k nemožnosti takový model využít. Některé části je proto dobré navrhnout a vyzkoušet ještě před tiskem celého modelu. Proto část kol, do které se zasouvá hřídel, byla vytištěna několikrát předem a vyzkoušena, aby byl model přizpůsoben této chybě.



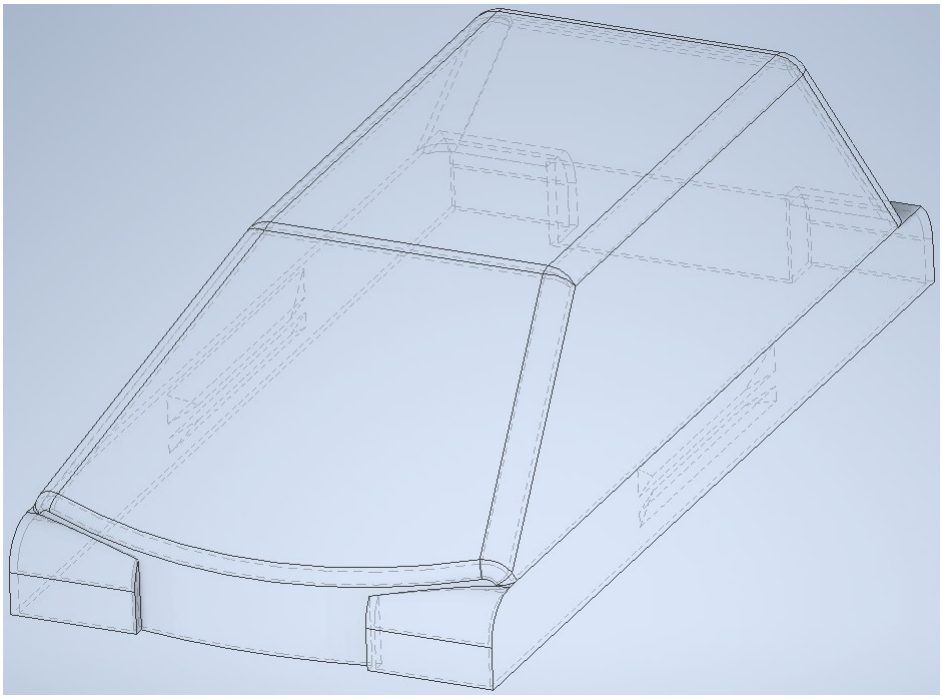
Obrázek 6.5: Vytištěné kolo z přední a zadní strany



Obrázek 6.6: Ukázka z programu Creality Slicer, zobrazení vrstev převodu stl na G-code

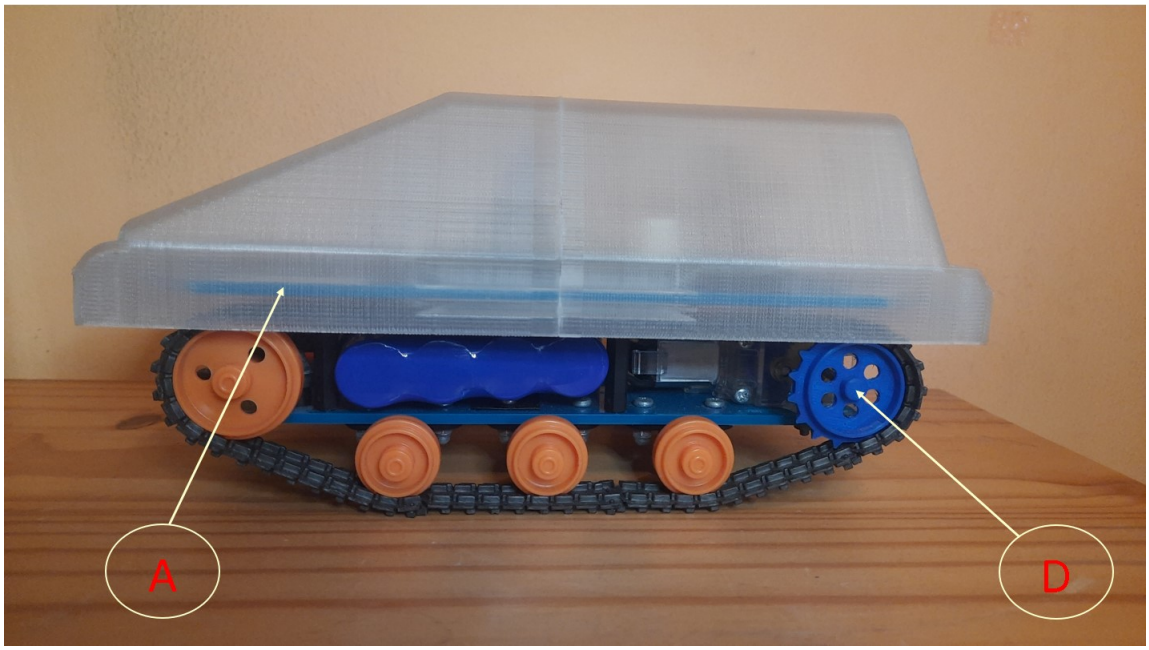
Při tisku také vznikají i další chyby způsobené samotnou technologií. Jak je vidět u vytisknutého kola [6.5], hladká je jen jedna strana. To je způsobeno tím, že nelze tisknout do vzduchu, a tak se tisknou podpory. Podpory jsou jen jako lešení a nejsou zcela vyplněné. Mezi jednotlivými zdmi tohoto lešení se tak vlákno občas prověsí. A při odstraňování tohoto lešení jeho části zůstanou spojeny s modelem. Je tedy potřeba takovou část buď opracovat, zabrousit, nebo jako v případě kol jen schovat na nepohledovou stranu. Na obrázku z programu na tvorbu g-kódu, sliceru, je vidět, jak se součást bude tisknout a jak bude právě ono lešení podpírat jednotlivé části [6.6]. Kolo se tisklo v leže a od zadní po přední stranu, právě aby byla přední strana hladká, aby nebylo třeba vytištěný model nějak opracovávat. Červené a žluté části v programu reprezentují tištěný model a jeho výplň, světle modré a tyrkysové představují opory a tmavě modře jsou vyznačeny přesuny trysky, kdy by neměla klást materiál.

Další navrhovanou součástí je vrchní kryt robota. Všechny obvody robota jsou zatím úplně odkryté, a tak by se mohlo stát, že by se i jen při manipulaci mohlo



Obrázek 6.7: Návrh krytu v prostředí Autodesk Inventor, drátové zobrazení se skrytými hranami

nějak narušit kabelové propojení. Navrhnout novou součást je trochu složitější než návrh kol, která již byla na světě. Design je třeba navrhnout a zároveň se oprostit od představy tanku. Zároveň by měl kryt jednoznačně určovat, kde má robot přední a kde zadní část, protože v ovládání je jasně dáno spuštění pohybu vpřed či vzad. Také je nutné, aby dobře držel, ale zároveň šel snadno demontovat, kdyby bylo potřeba. Zkušený designer by určitě odsoudil výsledný kryt [6.8], ale svůj účel překrytí odkrytých a volně přístupných obvodů by měl plnit stejně dobře jako určení přední a zadní části robota, což je vidět na obrázku [6.8 A], kde je vidět i nová kola [6.8 B]



Obrázek 6.8: Robot na konci práce s nasazeným krytem a novými velkými hnanými koly

6.2 Úpravy na elektronice robota

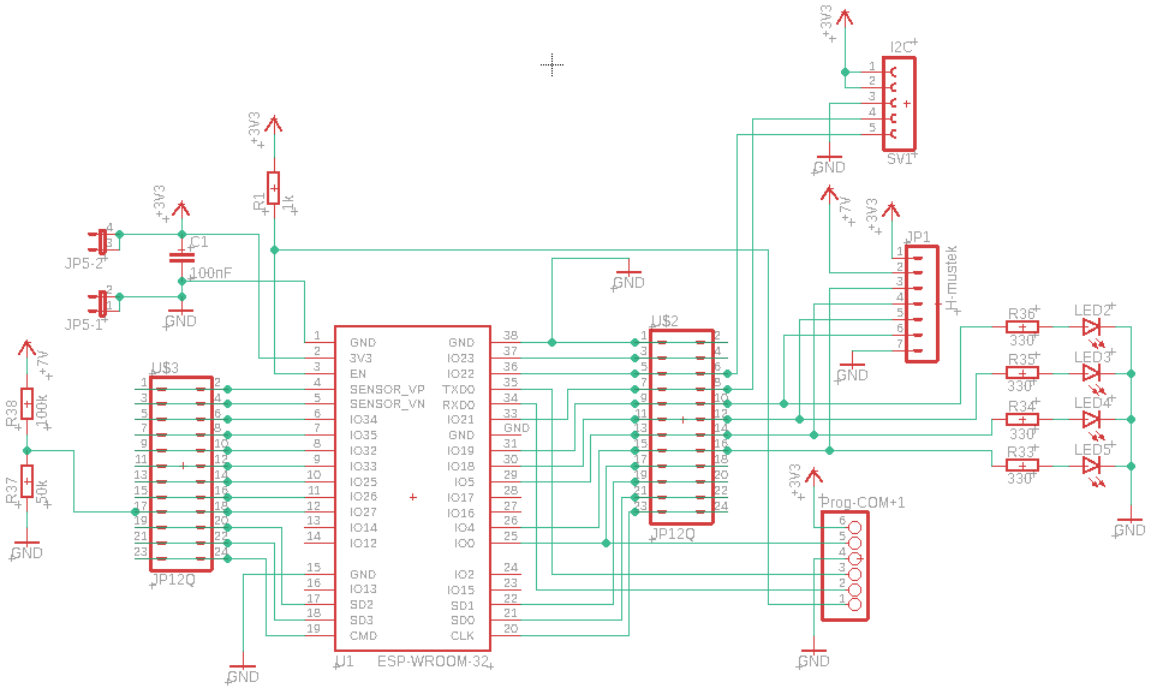
Robot je vždy kombinací mechanické a elektrické části, kde pro elektrickou část se dělí ještě na hardware a software. O softwarové části bylo snad řečeno dost, a tak se práce přesouvá do části hardwarové. V případě řídicího obvodu byl zmíněn modul ESP32, kterého se tato práce týká. Je však třeba doplnit, že byla navržena nová základní deska, která by měla nahradit desku s paticí pro PICAXE. A také byl nahrazen H-můstek.

6.2.1 Řídicí deska

Robot je poměrně malý, a tak bylo vhodné starou velkou desku s paticí pro PICAXE, která je na robotu už od jeho vzniku, nahradit deskou novou. Aby mohla být stará deska doposud používána, byla vytvořena redukce pro připojení vývojové desky s ESP. Tato patice je velmi neelegantní řešení, protože tohle spojení nemá žádnou pevnost a v případě potřeby nahrání programu se musí vždy celá vývojová deska odpojit od robota a na testování opět k robotu připojit, což je neefektivní.

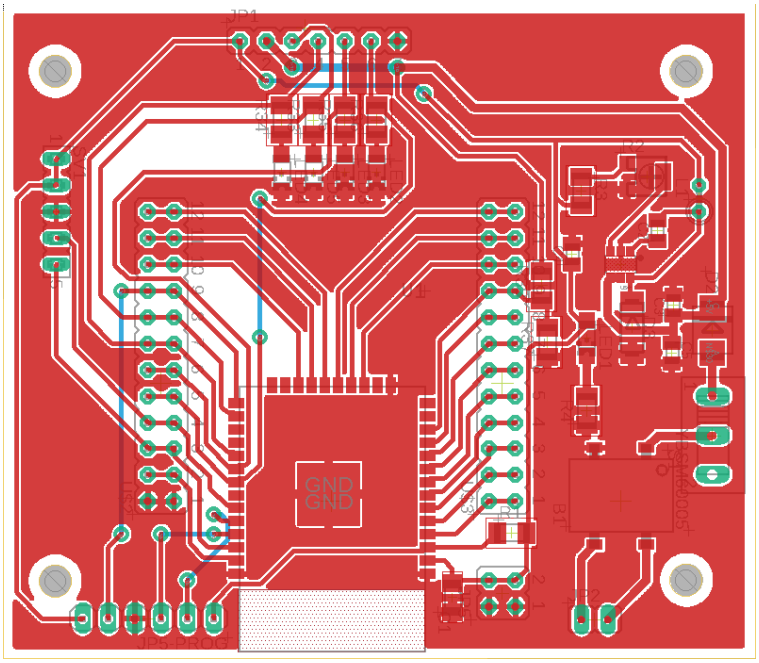
Nová deska by dovolila nahrání programu přímo na robotu, a navíc by bylo možné využít i větší množství pinů ESP.

Nová základní deska bude tedy osazena kromě ESP32 i spínaným zdrojem [6.11] a bude dále rozbočovat na připojení programátoru, gyroskopu/akcelerometru AltIMU, motorů (H-můstek), a navíc bude možné připojit cokoliv dalšího přes připravené

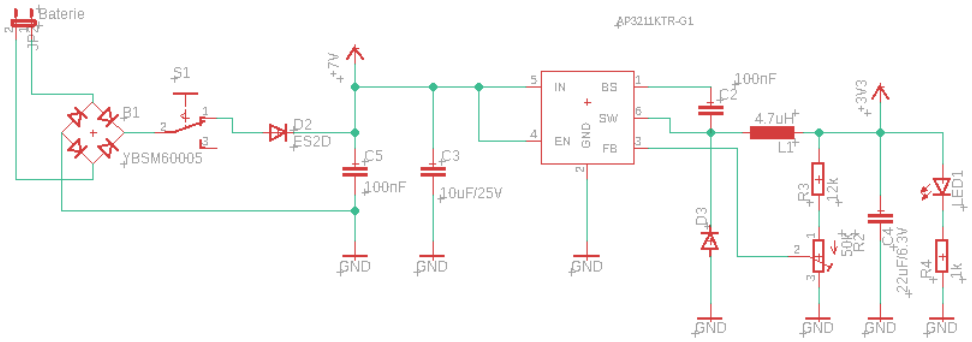


Obrázek 6.9: Schéma hlavní desky

kontakty. Tím se tato deska stane řídicím modulem. Spínaný zdroj byl již otestován v rámci projektu na původní vývojové desce [3.4]. Návrh desky [6.10] sice již proběhl, ale její výroba byla odložena na neurčito.



Obrázek 6.10: Návrh hlavní desky



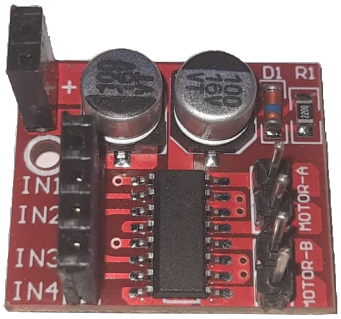
Obrázek 6.11: Schéma spínaného zdroje

6.2.2 Nově instalovaný H-můstek

Ještě během projektu byl využíván H-můstek původní, který byl lehce upraven. Byl doplněn o několik odporů, aby se posunul pracovní bod tranzistorů, což pomohlo k výkonu motorů. Ani toto řešení však nebylo dostatečné a bylo tedy rozhodnuto o výměně starého můstku. V dnešní době již není potřeba H-můstek navrhovat z jednotlivých součástek, lze zakoupit můstky všech možných specifikací. Byl tedy vybrán H-můstek, který může být napájen 2 až 10 V, ovládací napětí je 1,8 až 7 V , stálý proud výstupem může být až 1,5 A, špičkově dokonce 2,5 A. Motory robota odebírají okolo 0,25 A, což je dostatečný přesah, a špičkový rozběhový proud se pohybuje okolo 1 A. H-můstek je tedy sice univerzální, ale podle měření je lepší než původní. Má menší ztráty na napětí a při testování se ani nezahřál. Oproti původnímu H-můstku je navíc velikostně asi více než 10krát menší. Kvůli jeho jednoduchosti a metodě zpracování je však potřeba odpojovat ho od napájení, má totiž i v klidovém stavu stále proudový odběr.

IN1/3	IN2/4	MOTOR A/B
0	0	stojí
0	1	1. směr otáčení
1	0	2. směr otáčení
1	1	stojí

Tabulka 6.1: Logická funkce nového H-můstku



Obrázek 6.12: Nový H-můstek

7 Závěr

Již během projektu, ze kterého vychází tato bakalářská práce, jsem se seznámil s výrobky společnosti Espressif Systems, především používaným ESP32, jeho verzemi a vývojovými kity. Zjistil jsem původní funkce robota a vytvořil mu program pro základní ovládání výstupů. Také jsem vytvořil webovou stránku pro jeho ovládání.

Během této práce jsem následně vylepšil řízení rychlosti jak v ovládání, tak v programu robota. Prozkoumal jsem různé možnosti, jak omezit ovládání robota jen na jedno připojené zařízení a nakonec vybral omezení přes IP adresu klienta.

Předělal jsem celou webovou stránku s ovládáním, přidal JavaScript, který jsem se zapochodu učil, a ke komunikaci jsem zavedl asynchronní webový server. Vše k webové stránce je nyní uloženo v SPIFFS jako jednotlivé soubory upravovatelné samostatně. Přidal jsem rozšířené ošetření připojování k sítím.

Prozkoumal jsem možnosti určování polohy a vybral řešení přes kompas, akcelerometry a gyroskop, ze kterých jsem implementoval do webového rozhraní a programu zatím jen kompas, který jsem do webové stránky přidal jako otáčející se tank, aby mohl řidič lépe kontrolovat směr pohybu.

Našel jsem vhodný H-můstek, který by nahradil původní, který značně omezoval výkon, a vyzkoušel jsem ho na robotu. Přizpůsobil jsem mu program ovládání, aby se hýbal správně podle tabulky [6.1]

V programu Eagle, který teď již spadá pod Autodesk jsem navrhl novou řídicí desku, která zatím nebyla vytvořena, ale je možné, že ji na pásák dodáme později i mimo tuto práci. V Inventoru od stejného vydavatele jsem navrhl nová kola a vytiskl je na své 3D tiskárně, čímž jsem vyzkoušel její přesnost na poměrně malých předmětech. Navrhl jsem vrchní kryt a na stejné tiskárně jej vytiskl.

Celé testování probíhalo většinou na ESP-Wroom-32 osazeném na Devkitu v1. Musel jsem tedy přesto, že se jedná o podobný model, debugovat pro ESP32-Wroom-32D, který je použitý na samotném robotu. Bylo překvapivé, že se starší verze osazená na Devkitu osvědčila jako testovací prvek lépe, než modul osazený na desce přímo pro robota, krom jiného hlavně díky možnosti testovat na nepájivém poli a bez nutnosti používat externí programátor.

Použitá literatura

[1] NOVÁK, Petr. *Mobilní roboty: pohony, senzory, řízení*. 1. vydání. Praha: BEN - technická literatura, 2005. ISBN 80-7300-141-1.

[2] KLEMAN, Michal. *Řídící systém pásového robotu na platformě ARM*. Liberec, 2016. Bakalářská práce. Technická univerzita v Liberci.

[3] *Espressif: Company: Milestones* [online]. Shanghai: Espressif Systems, 2019 [cit. 27. 08. 2020]. Dostupné z: https://www.espressif.com/en/company/about-us/milestones.

[4] *ESP32-S2 Datasheet*. 1. vydání. Espressif systems, 2020. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-s2_datasheet_en.pdf.

[5] *ESP32-DevKitC V4 Getting Started Guide* [online]. Shanghai: Espressif Systems, 2020 [cit. 18. 08. 2020]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html.

[6] *ESP-WROVER-KIT V4.1 Getting Started Guide* [online]. Shanghai: Espressif Systems, 2020 [cit. 18. 08. 2020]. Dostupné z: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-wrover-kit.html.

[7] *Espressif* [online]. Shanghai: Espressif Systems, 2019 [cit. 26. 08. 2020]. Dostupné z: https://www.espressif.com/.

[8] FU, Yuan Ming. *ESP-Prog guide en* [online]. Shanghai: Espressif Systems, 2020 [cit. 19. 08. 2020]. Dostupné z: https://github.com/espressif/esp-iot-solution/blob/master/documents/evaluation_boards/ESP-Prog_guide_en.md.

[9] *ESPAsyncWebServer* [online]. Bulharsko: Espressif Systems, 2020 [cit. 23. 08. 2020]. Dostupné z: https://github.com/me-no-dev/ESPAsyncWebServer.

[10] KOYANAGI, Fernando. *ESP32: Internal Details and Pinout* [online]. Brasil: Autodesk, 2020 [cit. 26. 08. 2020]. Dostupné z: https://www.instructables.com/id/ESP32-Internal-Details-and-Pinout/.

[11] *Pololu - AltIMU-10 v4* [online]. Las Vegas, Nevada: Pololu Corporation, 2001--2020 [cit. 15. 08. 2020]. Dostupné z: https://www.pololu.com/product/2470.

[12] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009. ISBN 978-80-7232-388-3.

[13] SUEHRING, Steve. *JavaScript: krok za krokem*. 1. vydání. Brno: Computer Press, 2008. ISBN 978-802-5122-419.

[14] *Ender-3 3D Printer* [online]. Shenzhen: Creality 3D Technology Co., 2020 [cit. 18. 08. 2020]. Dostupné z: https://www.creality.com/goods-detail/ender-3-3d-printer.

[15] KLOSKI, Liza Wallach; KLOSKI, Nick. *Začínáme s 3D tiskem*. 1. vydání. Brno: Computer Press, 2017. ISBN 978-80-251-4876-1.

[16] PRŮŠA, Josef. *Představujeme Original Prusa SL1 – novou open-source SLA 3D tiskárnu* [online]. Praha: Josef Průša, 2018 [cit. 15. 08. 2020]. Dostupné z: https://blog.prusaprinters.org/cs/original-prusa-sl1-nova-sla-3d-tiskarna_33979/.

[17] SOSINSKY, Barrie A. *Mistrovství - počítačové sítě: [vše, co potřebujete vědět o správě sítí]*. 1. vydání. Brno: Computer Press, 2010. ISBN 978-80-251-3363-7.

Přílohy

Přiložené CD ROM

Které obsahuje:

- Text práce v elektronické podobě ve formátu PDF
- Archiv ve formátu ZIP se zdrojovými kódy pro ESP32 i se soubory webové stránky
- Archiv ve formátu ZIP se všemi obrázky použitými v práci
- Archiv ve formátu ZIP se soubory schématu a návrhu řídicí desky z programu EAGLE
- Archiv ve formátu ZIP se soubory krytu a kol, ve verzi pro tisk (.stl) i úpravu v programu Inventor (.ipt).