VYŠŠÍ ODBORNÁ ŠKOLA OBALOVÉ TECHNIKY
A STŘEDNÍ ŠKOLA, ŠTĚTÍ

Lucy

The one who wanders

Průzkumné vozítko sbírající data z okolí

Denisa Krebsová

MATURITNÍ PRÁCE

VYŠŠÍ ODBORNÁ ŠKOLA OBALOVÉ TECHNIKY
A STŘEDNÍ ŠKOLA, ŠTĚTÍ

Zadání maturitní práce s obhajobou

Číslo a název: 3. Konfigurace sítě s využitím Rpi a Arduino

Jméno žáka: Denisa Krebsová

Konzultant: Bc. Ladislav Šourek

Oponent: Ing. Jaroslav Moravec

Datum zadání: 10. ledna 2021

Datum odevzdání: 30. dubna 2021

Doba obhajoby: 15 minut

Zadání: Vytvořte model autonomního vozítka ovládaného přes internet . Vozítko bude určeno pro sběr dat z čidel, které budou součástí projektu. Pro přenos dat z čidel a data ovládání využijte platformy Arduino a Raspberry Pi. Autonomní jízdu bude zajišťovat primární okruh, který bude běžet buď na RPi nebo bude na microcontrolleru připojeném k RPi. Vytvořte aplikací pro ovládání vozítka a zobrazování naměřených hodnot z čidel, které se budou ukládat do database, která bude vytvořena na školním serveru.

Způsob zpracování:

- **Tištěná forma**: rozsah dokumentace 15 – 20 stran textu; v obálce s chlopněmi nebo pevná vazba; součástí práce bude úvodní obálka, zadání práce, harmonogram a prohlášení o souhlasu se zadáním práce, samostatnosti zpracování práce a použitím legálního software
- **Digitální forma**: kopie práce a pracovní soubory, dokumentace, prezentace na přiloženém CD nebo DVD v papírové obálce s jednoduchým HTML rozcestníkem; soubory v alternativních formátech
- **Model projektu**: vytvořte funkční model, který bude simulovat zadání

Počet vyhotovení: 1

Formální úprava práce:

Písmo: velikost 12
Font: Calibri, Arial nebo Times New Roman (zvolený font dodržte v celé práci)
Řádkování: 1,5
Vzdálenost mezi odstavci: 6 b.
Okraje: horní a dolní 25 mm, levý (vnitřní) 40 mm, pravý (vnější) 20 mm
Zarovnání odstavce: do bloku
Číslování stránek: vpravo dolu

Začátek hlavní kapitoly: vždy na nové straně
Dodržení typografických pravidel hladké sazby

Hodnocení:

1. Splnění zadání
2. Plnění plánu práce a účast na konzultacích
3. Aktuálnost a přínosnost tématu
4. Odborná úroveň práce, kvalita zpracování práce, použité prostředky
5. Zpracování dokumentace – typografie, zdroje, struktura, rozsah…
6. Dodržení ČSN ISO 690 a ČSN ISO 690-2 – bibliografické citace dokumentů
7. Hodnocení modelu a prezentace

Podpis žáka …….

Podpis konzultanta …….

Podpis ředitele školy …….

Harmonogram práce MZ

Třída: 4. I	
Studijní obor: 18-20-M/01 Informační technologie	
Jméno studenta:	**Denisa Krebsová**
Konzultant:	**Bc. Ladislav Šourek**
Číslo a název úlohy:	**3. Konfigurace sítě s využitím Rpi a Arduino**

Plán práce

Týden	**Práce**
10. 1. – 24. 1. 2021	Zadání tématu
25. 1. – 7. 2. 2021	Rozvaha o projektu
8. 2. – 28. 2. 2021	Schéma zapojení projektu
1. 3. – 14. 3. 2021	Hardwarová výbava
15. 3. – 28. 3. 2021	Vytváření software
29. 3. – 11. 4. 2021	Zkouška software
12. 4. – 18. 4. 2021	Výroba modelu
19. 4. – 30. 4. 2021	Odevzdání práce

Ve Štětí dne ……

Podpis ……

Kontrola plnění plánu práce

Datum	**Poznámky**	**Podpis**
24.1	S tématem souhlasí	
7.2	Konzultace splněna	
28.2	Schéma předáno	
14.3	splněno	
28.3	splněno	
11.4	splněno	
18.4	Model vyroben	
30.4	Práce odevzdána	

Prohlášení

Třída: 4. I

Studijní obor: 18-20-M/01 Informační technologie

Jméno žáka: ..... **Denisa Krebsová**

Konzultant: ..... **Bc. Ladislav Šourek**

Číslo a název úlohy: ... 3. Konfigurace sítě s využitím Rpi a Arduino

Čestné prohlášení o souhlasu se zadáním maturitní práce

Prohlašuji, že jsem se seznámil s obsahem zadání maturitní práce. Souhlasím se zadaným tématem.[1]

Ve Štětí dne …… Podpis: ……

Čestné prohlášení o samostatnosti zpracování maturitní práce

Prohlašuji, že jsem odevzdanou maturitní práci vypracoval samostatně a uvedl jsem všechny použité zdroje. Uvědomuji si, že prokáže-li se opak, může být má práce hodnocena jako nedostatečná.

Ve Štětí dne …… Podpis: ……

Čestné prohlášení o použití legálního softwarového vybavení

Prohlašuji, že veškeré programové vybavení, které bylo použito při řešení této maturitní práce, bylo užito v souladu s jeho licencí.

Ve Štětí dne …… Podpis: ……

[1] V případě nesouhlasu se zadáním maturitní práce se v den zadávání žák písemně obrátí na ředitele VOŠ, SPŠ, SOŠS a CR s odůvodněním svého nesouhlasu.

Obsah

Obsah 2
Anotace 3
Koncept 3
Postup 5
- Primární okruh 5
  - Příkazy 6
  - Kódy aktuální činnosti 7
  - Stavové kódy 7
- Webová aplikace pro ovládání 9
  - Ovládání pohybu 9
  - Stream videa 10
- Moduly a sběr dat 11
- Síť 13
- Napájení 13
- Dokončení 15

Závěr 16
Zdroje 17

Anotace

Smyslem projektu je vytvořit průzkumné vozítko, které bude možné ovládat na dálku přes internet. V ideálním případě nebude potřebovat ke svému provozu přítomnost člověka. Díky tomu bude moci být umístěno na místa, pro člověka nepříjemná či dokonce nebezpečná, a monitorovat okolní prostor a hodnoty různých proměnných okolí pomocí senzorů. Jeho předností bude jeho modularita a možnost výměny senzorů podle aktuálních potřeb měření. Projekt bude založen na původním modelu Lucy vytvořeném pro soutěž Dobrá škola Moderní škola 4.0 a nadále by měl sloužit pro účely prezentace školy.

Koncept

Hardwarová část projektu bude založena na hardwaru, který jsem převzala z projektu pro soutěž Dobrá škola Moderní škola 4.0. Hlavní prvky, ze kterých se vozítko bude skládat, budou podvozek s elektromotory, **Raspberry Pi (dále jen RPi)**, baterie pro napájení vozítka, ovladač elektromotorů a různé senzory.
Původní baterie bude nahrazena výkonnější a lehčí baterií, která vozítko odlehčí a zároveň poskytne dostatečnou kapacitu a výkon.
Projekt bude rozdělen na 4 okruhy. Těmi budou:

1. Autonomní jízda

Vozítko by mělo být schopné částečné autonomní jízdy. Ta bude využita například v případě ztráty signálu, chyby komunikace nebo zpoždění příkazu. Základním úkolem autonomní jízdy bude zejména ochrana vozidla před nárazem do překážky a zamezení nekontrolovatelné jízdy v případě ztráty signálu.

Autonomní jízdu bude zajišťovat primární okruh, který bude běžet buď na RPi nebo ho bude na mikrokontroleru připojeném k RPi.

2. Vzdálené ovládání

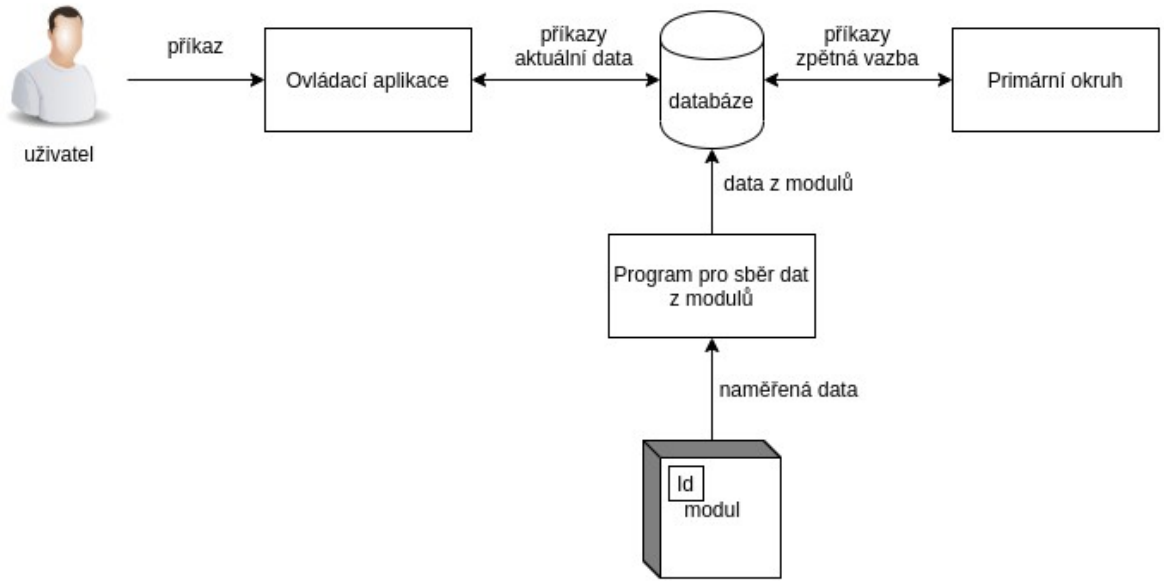
Vozítko bude možné ovládat na dálku a to přes ovládací aplikaci, pravděpodobně půjde o webovou aplikaci ve frameworku Django nebo Flask, která bude běžet na RPi. Přes uživatelské rozhraní aplikace bude možné vozítko ovládat, dále bude zobrazovat aktuální data, mezi ně patří například data z modulů. Data bude aplikace brát z databáze na Rpi, do které bude rovněž zapisovat příkazy provedené uživatelem.

3. Sběr dat

Vozítko bude během svého provozu sbírat data z připojených modulů. Povaha a četnost dat bude závislá na aktuálně připojených modulech. Přijatá data se budou ukládat do databáze, odkud budou dále zpracována a zobrazována v uživatelském rozhraní. Databáze poběží na RPi, software bude pravděpodobně použit PostgreSQL. Nabízejí se i jiné alternativy jako MySQL nebo MariaDB. PostgreSQL je strukturovaná databáze, jednoduchá na konfiguraci, která zároveň nabízí velké množství funkcí a velmi dobrou rychlost, což je ideální pro potřeby tohoto projektu.

4. Modularita

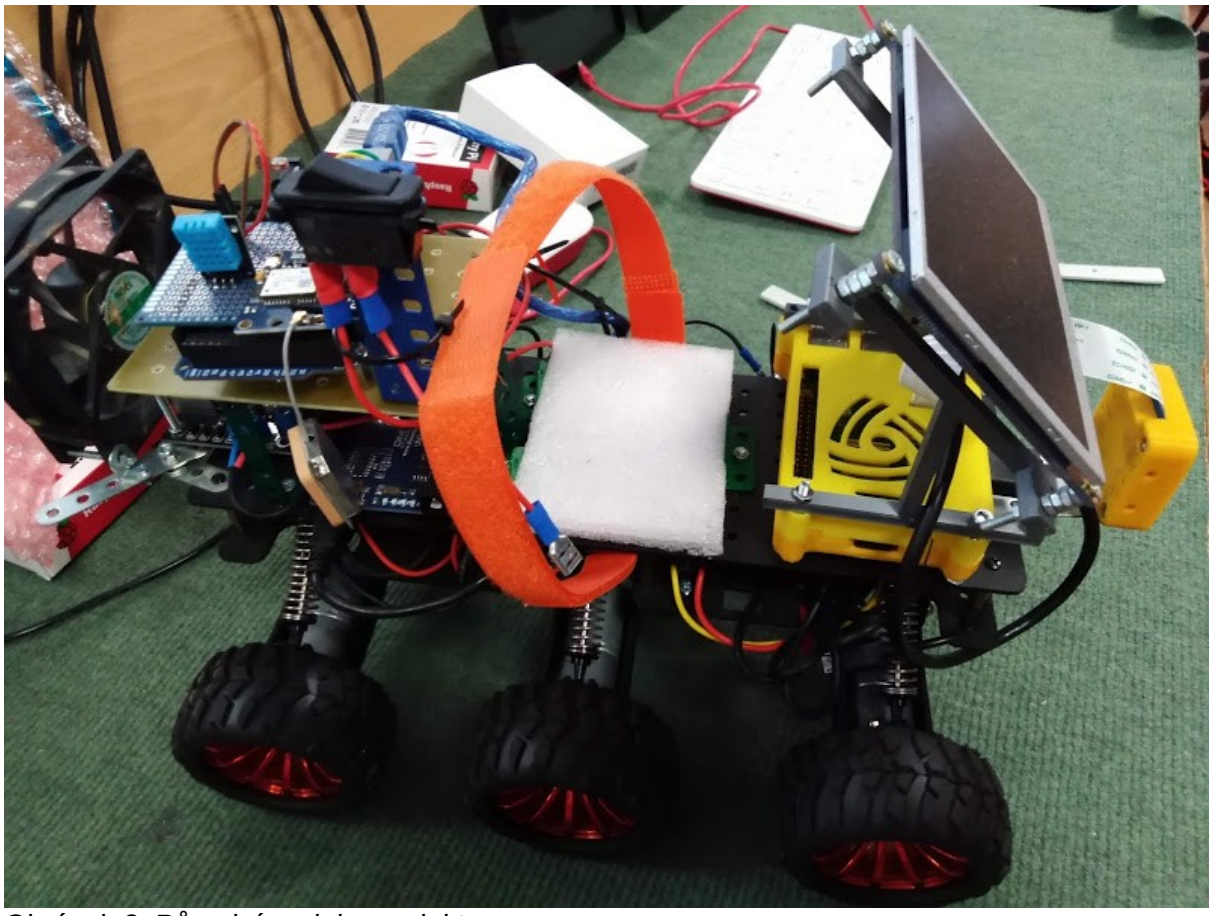
K vozítku bude možné připojit různé vstupní moduly, bude tak možné připojit či vytvořit modul podle aktuálních potřeb měření. Připojit je bude možné na sběrnici USB. Pro komunikaci mezi moduly a RPi bude stanoven přesně daný formát a struktura, mezi možnostmi jsou formáty json, csv či xml. Přijatá data se budou ukládat do databáze odkud budou dále zpracována.



Obrázek 1: Schéma komunikace mezi jednotkami

Postup

Vzhledem k tomu, že jsem předělávala již existují projekt, začala jsem rozebráním stávajícího projektu a identifikací jednotlivých součástek. Na jejich základě jsem se rozhodovala, jaké součástky použiji.



Obrázek 2: Původní podoba projektu

Primární okruh

Jako první jsem začala pracovat na primárním okruhu. RPi jsem přes převodník logických úrovní 3.3V na 5V připojila na motor driver, ovládající elektromotory vozítka. Poté jsem přes GPIO začala zkoušet, jaké hodnoty je potřeba nastavit pro rozjetí, či zabočení vozítka. Napsala jsem si script v programovacím jazyce Python, který mi pomocí modulu pynput umožňoval z příkazové řádky vozítko ovládat šipkami na klávesnici, díky tomu jsem tak mohla v reálném čase doladit hodnoty.

Takovéto řešení však nebylo možné v projektu uplatnit, a to proto že pokud na Raspberry Pi neběží operační systém, GPIO nelze ovládat, na některých pinech se objevilo napětí a vozítko se tak plnou rychlostí rozjelo. Musela jsem proto přistoupit na využití nějakého prostředníka mezi RPi a motor driverem. V tomto případě se nabízela deska **Arduino Pro Micro (dále jen Micro)**, a to zejména díky své velikosti. Deska Micro

je osazeno chipem ATmega32U4, který disponuje více než dostatečným výkonem a pamětí pro tuto implementaci.

Pro připojení desky Micro k RPi jsem využila sběrnici I2C, kde masterem je Rpi. Mezi Rpi a deskou Micro jsem navrhla jednoduchou komunikaci, pomocí kódů o velikosti jednoho bytu, tedy 8 bitů. RPi posílá příkazy desce Micro a poté se desky Micro vyžádá odpověď. Příkazy jsou rozděleny do skupin, skupinu určují první 4 bity. Pomocí příkazů se dá ovládat pohyb vozítka, dá se z něj načíst zpětná vazba nebo se pomocí nich dají nastavit hodnoty proměnných, například výkon.

Na desce Micro běží ve smyčce program, který, pokud je vozítko v pohybu, neustále kontroluje, zda je před nebo za ním dost místa pro pohyb. Využívá k tomu ultrazvukové senzory vzdálenosti HC-SR04. Pokud prostor není dostatečný, vozítko vypne pohon, nezastaví však na místě, k tomu by byla potřeba implementovat brzda. Funkcionalitu brzdy bude možné v budoucnu snadno přidat.

V této smyčce se zároveň při každé iteraci snižuje číslo nastavené po přijetí posledního příkazu. Jedná se o bezpečnostní prvek, kdy pokud během cca 2s nepřijde žádný příkaz, číslo dosáhne nuly a vozítko zastaví. Pokud tedy dojde v systému k chybě, lagu či ztrátě konektivity, vozítko nebude pokračovat v jízdě.

Micro zapisuje na sériové rozhraní informace o změně aktuální činnosti, stavové kódy přijatých příkazů a další data. Tyto data mohou pomoci při případné diagnostice chyb.

Funkčnost komunikace mezi RPi a deskou Micro jsem během vývoje testovala pomocí scriptů napsaných v programovacím jazyce Python.

Příkazy

Skupina příkazů pro jízdu: 0

SYSTÉMOVÝ NÁZEV PŘÍKAZU	BYTE	POPIS	POZNÁMKA
COMMAND_CONTINUE	0x00	Vozítko pokračuje v tom co právě dělá	Obnova příkazu
COMMAND_STOP	0x01	Vozítko zastaví	
COMMAND_RIDE_FORWARD	0x02	Vozítko pojede dopředu	
COMMAND_FORWARD_STEER_LEFT	0x03	Vozítko pojede mírně doleva	zabírá jen levá strana
COMMAND_FORWARD_STEER_RIGHT	0x04	Vozítko pojede mírně doprava	zabírá jen pravá strana
COMMAND_RIDE_BACKWARD	0x05	Vozítko pojedu dozadu	
COMMAND_BACKWARD_STEER_LEFT	0x06	Vozítko pojede dozadu mírně doleva	zabírá jen levá strana
COMMAND_BACKWARD_STEER_RIGHT	0x07	Vozítko pojede dozadu mírně doprava	zabírá jen pravá strana
COMMAND_TURN_LEFT	0x08	Vozítko se otáčí doleva	protichod
COMMAND_TURN_RIGHT	0x09	Vozítko se otáčí doprava	protichod

Skupina příkazů zpětné vazby: 1

SYSTÉMOVÝ NÁZEV PŘÍKAZU	BYTE	POPIS	POZNÁMKA
COMMAND_SEND_STATE	0x11	Micro vrátí kód aktuální činnosti	

Skupina příkazů nastavení hodnot: 2

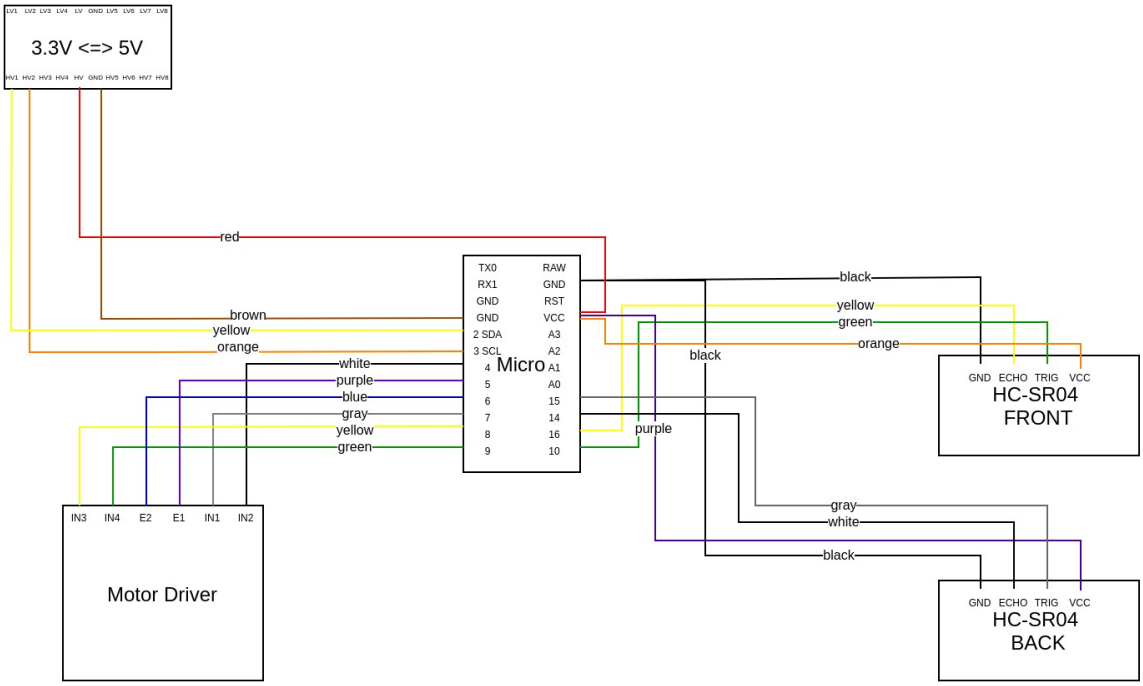
SYSTÉMOVÝ NÁZEV PŘÍKAZU	BYTE	POPIS	POZNÁMKA
COMMAND_SET_PWM_BYTE	0x21	Micro vrátí kód aktuální činnosti	
COMMAND_SET_SPEED_LEVEL	0x22		ZATÍM NEVYUŽITO
COMMAND_SET_FREE_SPACE_MULTIPLIE	0x23		ZATÍM NEVYUŽITO

Kódy aktuální činnosti

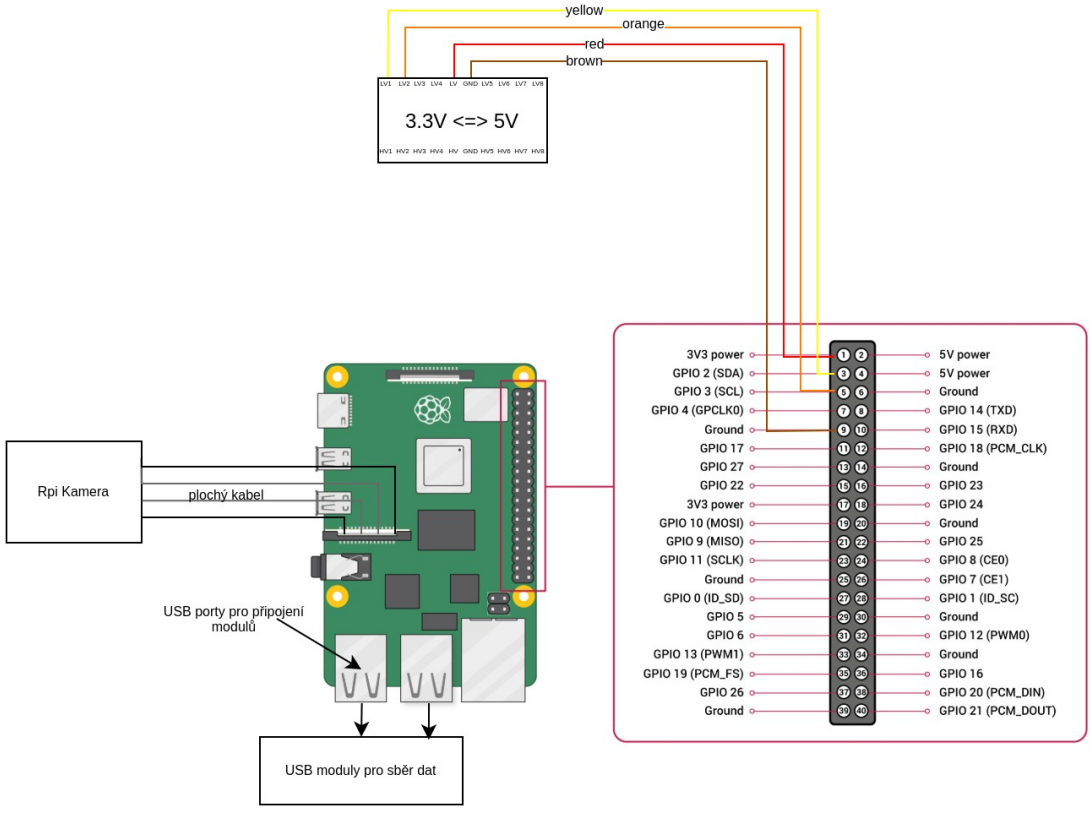
SYSTÉMOVÝ NÁZEV ČINNOSTI	BYTE	POPIS	POZNÁMKA
STOP	0x01	Vozítko stojí	
RIDE_FORWARD	0x02	Vozítko jede dopředu	
FORWARD_STEER_LEFT	0x03	Vozítko jede mírně doleva	
FORWARD_STEER_RIGHT	0x04	Vozítko jede mírně doprava	
RIDE_BACKWARD	0x05	Vozítko jede dozadu	
BACKWARD_STEER_LEFT	0x06	Vozítko jede dozadu mírně doleva	
BACKWARD_STEER_RIGHT	0x07	Vozítko jede dozadu mírně doprava	
TURN_LEFT	0x08	Vozítko se otáčí vlevo	
TURN_RIGHT	0x09	Vozítko se otáčí vpravo	
BARRIER_IN_FRONT_STOP	0x0A	Vozítko stojí, protože se před ním nachází překážka	
BARRIER_IN_BACK_STOP	0x0B	Vozítko stojí, protože se za ním nachází překážka	
NO_COMMUNICATION_STOP	0x0C	Vozítko stojí, protože micro nedostalo žádný příkaz nebo vypršela platnost posledního příkazu	

Stavové kódy

SYSTÉMOVÝ NÁZEV KÓDU	BYTE	POPIS	POZNÁMKA
OK	0x01	Příkaz proveden v pořádku	
ERROR_BARRIER_IN_FRONT	0x02	Příkaz nemohl být proveden kvůli překážce před vozítkem	
ERROR_BARRIER_IN_BACK	0x03	Příkaz nemohl být proveden kvůli překážce za vozítkem	
ERROR_UKNOWN	0x11	Neznámá chyba	
ERROR_NO_VALUE	0x05	K příkazu nebyla dodána hodnota	



Obrázek 3: Schéma zapojení desky Micro s periferiemi



Obrázek 4: Schéma zapojení Raspberry Pi

Webová aplikace pro ovládání

Pokračovala jsem tvorbou webové aplikace ve frameworku Django. Nainstalovala jsem databázový systém PostgreSQL a vytvořil databázi pro projekt. V Djangu jsem vytvořila model pro příkaz (Command) a model pro provedení příkazu (CommandExecution).

Command

SYSTEM NAME	NAME	DESCRIPTION	FLAG
CharField max_length=120	CharField max_length=120	CharField max_length=250	Boolean ZATÍM NEVYUŽITO

CommandExecution

COMMAND	VALUE	TIMESTAMP	STATE	RESULT	NOTE
ForeignKey Command	IntegerField default=0	BigIntegerField Původně: DateTimeField auto_now_add= True	CharField max_length =250	CharField max_length= 250	CharField max_length =1000

Ovládání pohybu

Pokračovala jsem prací na frontendu webové aplikace. V Javascriptu jsem vytvořila jednoduchý script pro ovládání pohybu. Příkazy se posílají v závislosti na tom, jaké šipky na klávesnici jsou v dané chvíli stisknuté, nebo na které tlačítko se směrovkou uživatel kliknul. Společně s příkazem se posílá i časové razítko. K odesílaní dat jsem použila knihovnu JQuery.

Backend aplikace přijme příkaz jaký má primární okruh vykonat a uloží ho společně s časovým razítkem, do databáze jako CommandExecution. Čas jsem původně ukládala do pole DateTimeField s nastavení auto_now_add=True, všimla jsem si ale, že během ukládání často dochází k prodlevám v ukládání a aplikace často vůbec neodpovídala. Rozhodla jsem se proto čas ukládat do pole BigIntegerField ve formátu unixového času, tj. počtu sekund uplynulých od 1.1.1970. Po této změně se počet záseků rapidně snížil. Raději jsem ještě navýšila parametr databáze “shared buffers” na 500MB.

Pro předání příkazů desce Micro jsem napsala script opět v Pythonu, který pravidelně načítá záznamy CommandExecution z databáze, vyhodnocuje jejich platnost a v případě, že jsou platné, je pošle přes sběrnici I2C desce Micro. Záznam CommandExecution v databázi označí jako provedený a odpověď od desky Micro uloží do databáze. Pokud během tohoto procesu vznikne chyba, také se zapíše do databáze.

Pro tento script jsem vytvořila v systému službu, která script spouští při každém zapnutí systému.

Stream videa

Po otestování ovládání pohybu pomocí webové aplikace jsem se věnovala streamu z webkamery. Jako první jsem zkusila použít software Motion. Kvalita přenosu však byla na velmi nízké úrovni a zpoždění bylo několik sekund a to dokonce na stejné síti. Zkusila

jsem stream pomocí protokolu RTSP, ale jeho zpoždění bylo ještě horší. Hledala jsem tedy jiné řešení.

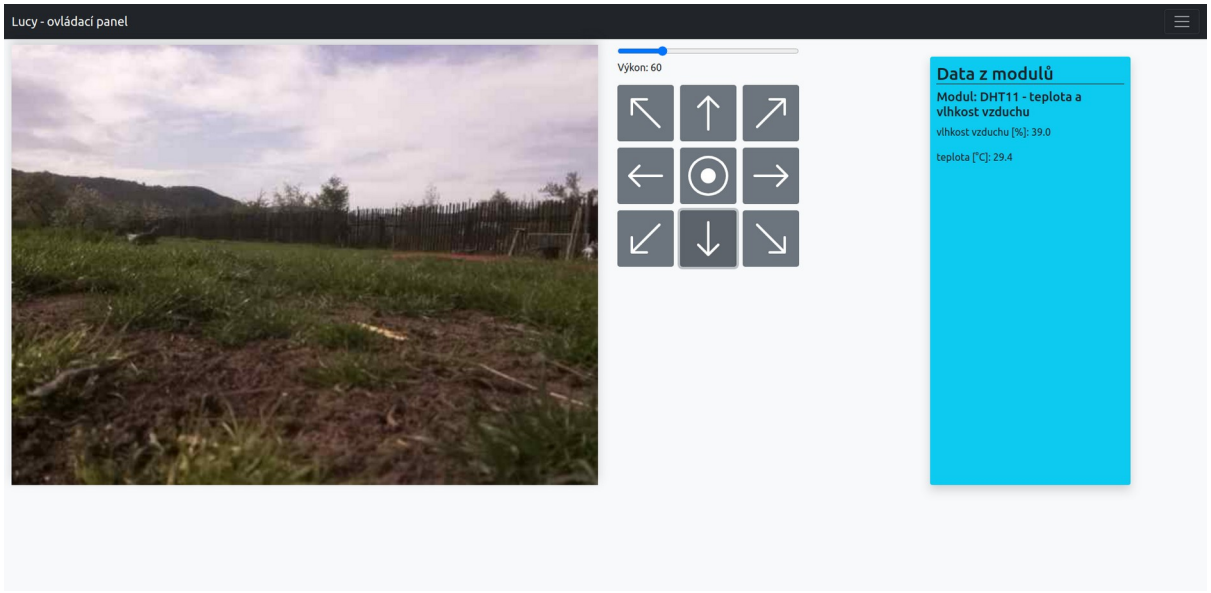
Na GitHubu jsem našla repozitář RPi_Cam_Web_Interface, jedná se o aplikaci v php, která zobrazuje aktuální snímky z kamery, a ze které se dá měnit nastavení kamery. Kód pro načítání obrazu z kamery jsem použila na frontendu aplikace pro ovládání. Vzhledem k tomu, že aplikace zobrazuje aktuální snímky, které se ukládají na SD kartu, nedochází u obrazu k velkému zpoždění.

Toto řešení ale bohužel příliš zatěžuje SD kartu, proto jsem se rozhodla hledat další řešení v podobě streamu. Zkusila jsem použít ukázkovou aplikaci pro NPM balíček raspberrypi_node_camera_web_streamer, NPM je správce balíčků pro JavaScript, výchozí správce balíčků pro prostředí Node.js. Tato aplikace ale bohužel nefungovala, musela jsem tedy její kód trochu upravit. Toto řešení je jednoduché, jeho zpoždění není velké a nezatěžuje SD kartu, je ale náchylnější ke zpoždění. Pro případ, že by toto řešení v budoucnu nevyhovovalo, je možné stream zpětně nastavit na řešení pomocí RPi_Cam_Web_Interface.

Pro stylování frontendu jsem použila Bootstrap, zbytek jsem nastylovala ručně pomocí CSS. Vzhled je jednoduchý a layout snadno se přizpůsobí i mobilnímu telefonu.

Pomocí webové aplikace pro ovládání je také možné vypnout operační systém RPi. V menu jsem umístila tlačítko, které pošle příkaz k vypnutí na server, po jeho stisknutí se objeví odpočet, po jehož uplynutí je možné vozítko vypnout vypínačem.

Pro ovládací webovou aplikaci jsem také vytvořila službu, která aplikaci spouští při každém zapnutí systému.



Obrázek 5: Frontend webové aplikace pro ovládání

Moduly a sběr dat

K vozítku se dají přes USB připojit moduly, za kterých RPi načítá data. Takový modul si můžu vyrobit každý, kdo dodrží daný formát. Komunikace mezi RPi a modulem probíhá přes sériové rozhraní, kdy RPi pošle modulu požadavek na zaslání dat a modul pošle odpověď s naměřenými daty. Pro přenos jsem použila datový formát JSON.

Formát požadavku

```
{
	' operation ' : ' REQUEST_DATA ',
}
```

Formát odpovědi

```
{
	'success': '[true nebo false]',
	'name': '[Název (identifikátor) modulu]',
	'display_name': '[Název, jež bude zobrazen v GUI]',
	'description' : [ popis modulu -> volitelné],
	'error': '[chyba pokud nastala -> volitelné]',
	'variables': [
		{
			'name': '[název proměnné]',
			'type': '[datový typ hodnoty]',
			'value': [hodnota,
			'unit' : [měrná jednotka -> volitelné],
		},
		{
			'name': '[název proměnné]',
			'type': '[datový typ hodnoty]',
			'value': [hodnota,
			'unit' : [měrná jednotka -> volitelné],
		},
		... volitelný počet proměnných
	]
}
```

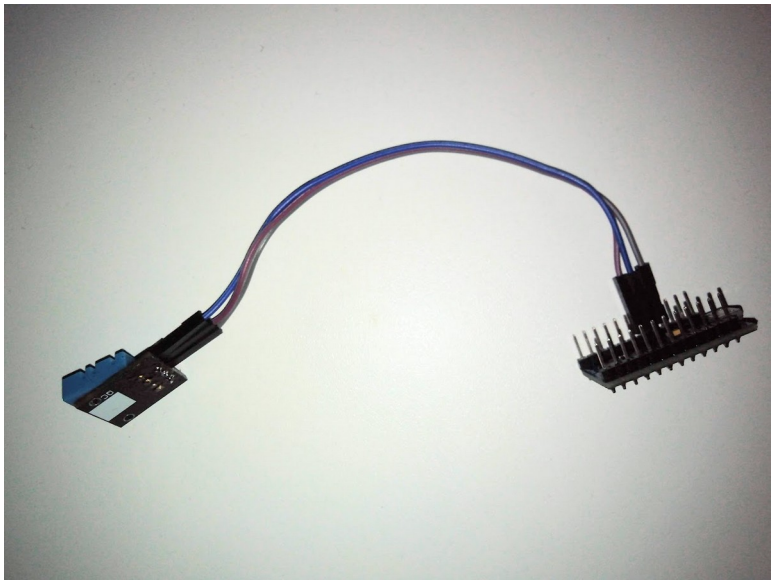
Na RPi běží script v Pythonu, který hledá nařízení připojená na USB a zkouší s nimi navázat sériovou komunikaci. Pokud uspěje, pošle jim požadavek na zaslání dat. Pokud modul pošle odpověď obsahují data v daném formátu, uloží je do databáze. Z databáze si data načítá webová aplikace pro ovládání a nejaktuálnější data se zobrazí frontendu webové aplikace pro ovládání (GUI).

Původně jsem chtěla komunikaci mezi RPi a moduly udělat sofistikovanější. Na začátku komunikace by se RPi s modulem domluvilo na četnosti načítaných dat, prodlevě

mezi načtením dat a dalších případných možnostech. O probíhající komunikaci se měly v paměti udržovat záznamy, v databázi mělo být uloženo, zda je modul právě připojen, či odpojen, a případná další data. Po nespěšných pokusech o sofistikovanější komunikaci s čínskými klony desek Arduino jsem ale od této představy upustila a naprogramovala jsem pouze jednoduché načítání dat. S některými deskami se mi bohužel nepovedlo navázat komunikaci vůbec.

Script pro načítaní dat z modulů a jejich ukládání do databáze se taktéž spouští automaticky po zapnutí systému.

Pro účely testování a prezentace jsem vytvořila jednoduchý modul z další desky Arduino Pro Micro se senzorem DHT-11, který měří teplotu a vlhkost vzduchu.

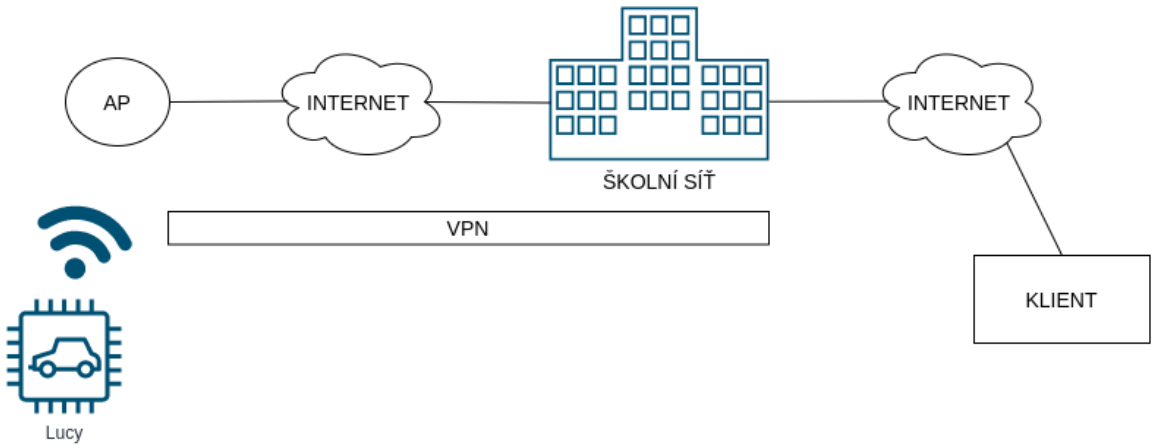


Obrázek 6: Ukázkový modul se senzorem DHT-11

Síť

Vozítko je připojené přes VPN do školní sítě, kde má přidělenou perzistentní adresu. Pro připojení do VPN jsem použila OpenVPN, vozítko se automaticky připojí hned po spuštění systému, pokud má přístup k internetu.

Pro účely projektu byly vytvořeny domény lucy.odbornaskola.cz, která odkazuje na ovládací webovou aplikaci, a lucy-cam.odbornaskola.cz, která odkazuje na stream z kamery.



Obrázek 7: Zjednodušené schéma sítě

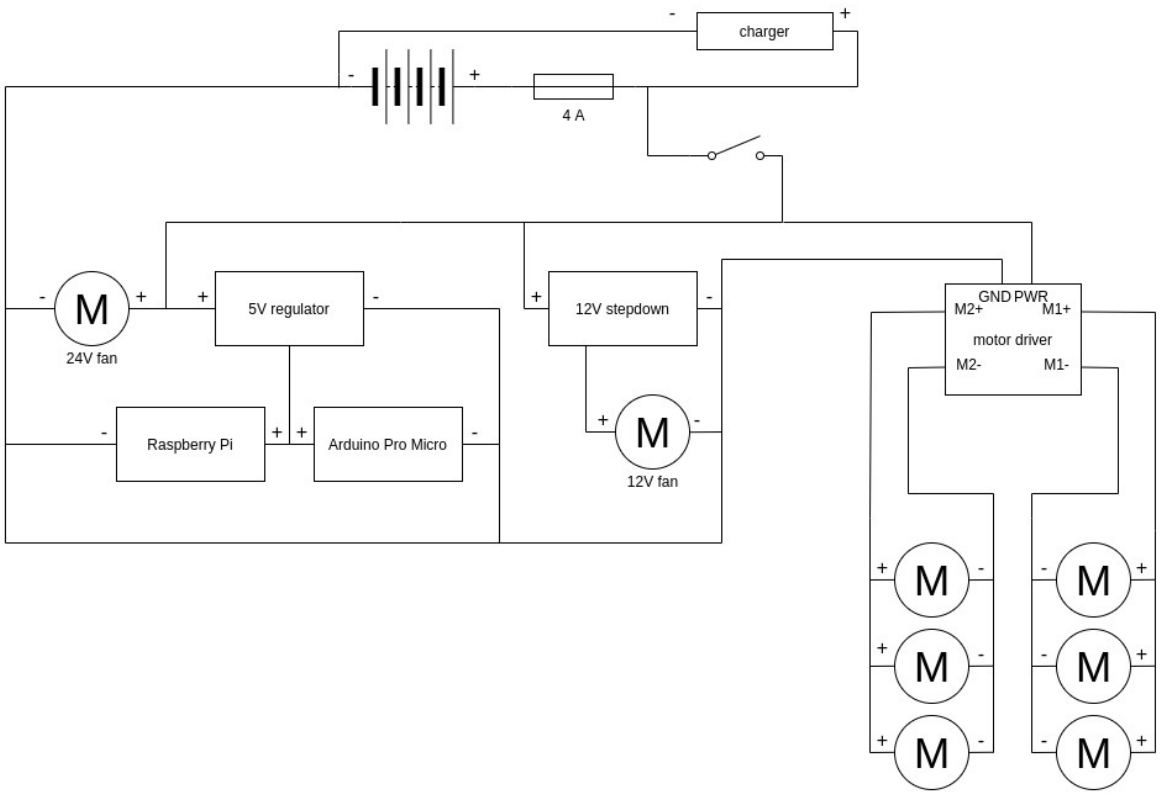
Napájení

Vozítko napájí čtyř článková LiPol baterie s kapacitou 5500mAh a vybíjecími proudy až 120C, která je pro potřeby projektu více než dostačující. Okruh napájení je chráněn tavnou pojistkou a samozřejmě opatřen vypínačem. Hned za vypínačem je hlavní svorkovnice, z níž jsou vyvedeny napájecí obvody všech součástek vozítka. Vozítko je opatřeno 12V stepdown modulem pro napájení ventilátoru a 5V regulátorem pro napájení RPi a desky Micro, jež je součástí primárního okruhu. Druhý ventilátor je napájen přímo z baterie.

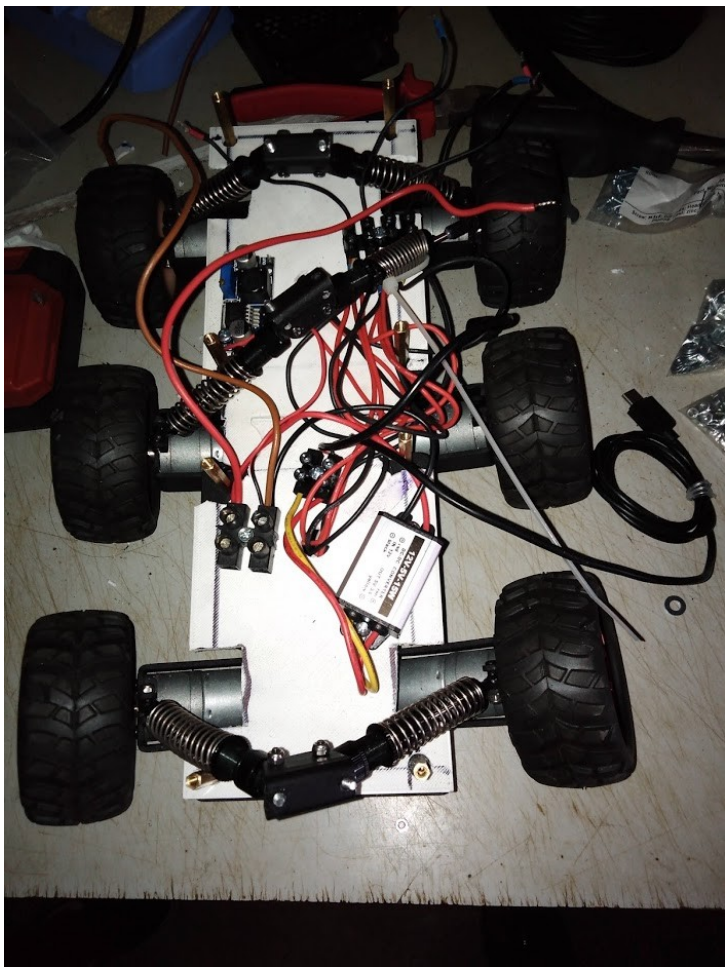
Na servisním konektoru baterie je připojen modul, který v případě poklesu napětí na některém z článků baterie pod nastavenou hodnotu spustí alarm.

Pro nabíjení baterie byl zakoupen speciální balance charger.

Většina elektroinstalace je uložena v podvozku. Uvnitř podvozku je dále uložena svorkovnice pro napájení elektromotorů z motor driveru. Na ploše podvozku jsou uloženy plastové podložky, které slouží jako izolace.



Obrázek 8: Schéma napájení



Obrázek 9: Podvozek s elektroinstalací

Dokončení

Jakmile jsem měla všechny části projektu připravené, stačilo je jen poskládat. K vytvoření těla pro uložení komponent bylo využito 3D tisku. U komponent, které by se časem mohly rozbít, jsem volila připojení konektory, což usnadňuje jejich výměnu.

Celé vozítko je chlazeno dvěma ventilátory umístěnými nad motor driverem a nad RPi, jedná se o komponenty produkující nejvíce tepla.


Obrázek 10: Finální podoba projektu

Závěr

Celkově usuzuji, že zpracování projektu se mi povedlo. Největší problém dělá připojení k síti, s čímž se vzhledem k využitému hardwaru, parametrech školní sítě a faktu, že komunikace probíhá přes internet, počítalo. Zpracování projektu je v rámci možností použitého hardwaru. Na hotovém modelu bohužel nefungují ultrazvukové senzory vzdálenosti, z nějakého důvodu na nich není není napětí, příčinu jsem vzhledem k dostupnosti desky Micro nezjistila. Napoprvé nefungovalo téměř nic, což je běžné, zvláště u čínských kopií hardwaru.

V budoucnu je projekt možné rozšířit o další moduly, funkcionality či vylepšit ty již stávající. Již při práci na projektu jsem přemýšlela, jaké funkce by mohly být v budoucnu přidány, či co by mohlo být vylepšeno a někde jsem kód na tato vylepšení již připravila, aby byla jejich implementace jednodušší.

Během práce na tomto projektu jsem se mnohé naučila, hlavně při hledání různých řešení daného problému. Zdrojové kódy k projektu jsou umístěny také na https://git.asgard.odbornaskola.cz/denisa.krebsova/Lucy.

Zdroje

https://app.diagrams.net/
https://neonaut.neocities.org/blog/2020/exploring-the-hiletgo-pro-micro-clone.html
https://www.raspberrypi.org/documentation/usage/gpio/
https://www.djangoproject.com/
https://djangocentral.com/using-postgresql-with-django/
https://www.postgresql.org/docs/current/
https://distributedsystemsauthority.com/optimizing-postgresql-shared-buffers/
https://pimylifeup.com/raspberry-pi-webcam-server/
Raspberry Pi: Stream video to VLC player, using rtsp protocol.
https://www.youtube.com/watch?v=PtoXKq8u6UA
https://raspberrypi.stackexchange.com/questions/23182/how-to-stream-video-from-raspberry-pi-camera-and-watch-it-live
https://github.com/silvanmelchior/RPi_Cam_Web_Interface
https://www.npmjs.com/package/raspberrypi-node-camera-web-streamer
https://github.com/caseymcj/raspberrypi_node_camera_web_streamer
https://getbootstrap.com/docs/5.0/
https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/
https://www.instructables.com/Raspberry-Pi-Arduino-Serial-Communication/
https://arduinojson.org/