VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATIZACE ŘÍZENÍ RC MODELŮ

AUTOMATION OF RC MODELS CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JAN VÁVRA**
AUTHOR

VEDOUCÍ PRÁCE **prof. Dr. Ing. PAVEL ZEMČÍK**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Vávra Jan**
Program: Informační technologie
Název: **Automatizace řízení RC modelů**
Automation of RC Models Control
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte problematiku radiového řízení (RC) modelů, zejména se zaměřením na časování signálů mezi RC přijímačem a servy.
2. Navrhněte takový postup, aby RC model získal nové lepší vlastnosti, například automatické udržování rychlosti a směru u modelů aut apod.
3. Navrhněte technické vybavení pro úpravu signálů s využitím vhodného existujícího embedded zařízení.
4. Vyberte vhodný model a algoritmus pro řízení a implementujte jej. Výsledky implementace demonstrujte.
5. Diskutujte dosažené výsledky a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího práce

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Zemčík Pavel, prof. Dr. Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato práce se zabývá řízením RC modelů pomocí mikrokontroléru ESP-32 ovládané přes mobilní zařízení s OS Andriod pomocí WiFi. Cílem této práce je obohatit ovládání o automatizované chování, které by zjednodušilo nebo případně obohatilo požitek z řízení. Práce se zabývá používání pulzní šířkové modulace pro ovládání serv, tvorbou uživatelského rozhraní a protokolem pro komunikaci mezi mobilem a vývojovou deskou. Výsledkem práce je aplikace na OS Android, ze které lze posílat pokyny na mikropočítač, který ovládá servo řízení a elektromotor modelu. Systém má předdefinované triky přímo na mikropočítači a možnost definovat vlastní triky, které jsou po aktivaci postupně odesílány na mikropočítač v podobě jednotlivých instrukcí.

Abstract

This thesis is about controlling RC model with microcontroller ESP-32 through WiFi enabled mobile device with OS Android. Goal of this thesis is to make controlling RC model easier and more fun. Parts of this thesis are dealing with using pulse width modulation to control servos, creating a user interface and making a protocol for comunication between RC model and mobile device. Result of this work is application on OS Android, which sends instructions on microcomputer, that controls steering servo and elecromotor. System has predefined stunts on microcomputer and operator has a choice of defining custom stunt in application, which is send to microcomputer in series of individual instructions

Klíčová slova

Automatizace, RC, ESP-32, C, Android, Android SDK, Java, PWM, WiFi, servo, ovládání

Keywords

Automation, RC, ESP-32, C, Android, Android SDK, Java, PWM, WiFi, servo, control

Citace

VÁVRA, Jan. *Automatizace řízení RC modelů*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Automatizace řízení RC modelů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Vávra
15. května 2019

Poděkování

Chtěl bych tímto poděkovat mému vedoucímu práce prof. Dr. Ing. Pavlu Zemčíkovi za odborné rady, připomínky a materiální pomoci při řešení této bakalářské práce.

Obsah

Kapitola 1

Úvod

Existuje mnoho nadšenců, kteří své RC modely různě obohacují pomocí moderních technologií, například přidáním kamery pro pohled z auta, nebo diod, kterými chtějí RC modelu dát vlastnosti podobné běžnému automobilu (blinkry, brzdové světla, osvícení pro jízdu ve tmě, apod.) nebo ho zlepšit například senzory pro zabránění nárazu apod. Bohužel ne vždy nám k tomu stačí pouze části specifické pro RC modelování. V posledních dobách se na trhu ukazují jednoduše programovatelné mikropočítače, kterými můžeme jak rozblikat ledku, umět ovládat serva a tím udávat směr a rychlost modelu, tak komunikovat přes WiFi nebo Bluetooth a tím pádem nahradit původní přijímač a mít možnost třeba získávat obraz z již zmiňované kamery nebo senzoru.

RC modely jsou rádiem řízené (radio controlled) zmenšené podoby aut, letadel, lodí, apod. Pohyb modelům zajišťují malé elektromotory a pro udání směru se používají serva. Pokyny, kam mají zahnout nebo jak zrychlit, jim předává přijímač, který poslouchá příkazy od vysílače umístěného na ovladači, kde pomocí páček nebo volantu se určuje směr a rychlost.

Práci jsem si vybral, protože mám kladný vztah k elektronice a automobilům. Dále jsem se chtěl naučit naprogramovat aplikaci pro Android a vyzkoušet si ovládání modelu přes mikropočítač.

Práce je rozdělena do jednotlivých kapitol. V kapitole 2 je přehled signálů, mikropočítačů a bezdrátových technologií, které jsou aktuálně používané. V kapitole 3 jsou vypsány existující implementace. Kapitola 4 pojednává o mém pochopení zadání a sepsání kroků řešení. V kapitole 5 nalezneme návrh jednotlivých částí práce.

Kapitola 2

Shrnutí současného stavu v RC technice a mikropočítačích

Tato kapitola je souhrn znalostí pro pochopení této práce. Lze zde najít informace o aktuálně nejpoužívanějších mikropočítačích, o bezdrátových přenosových technologiích a o signálech používaných v RC technice, které lze použít pro komunikaci mezi zařízeními. Tato kapitola není encyklopedickým výčtem všech informací, nýbrž zde čtenář najde informace, které byly nezbytně nutné nastudovat pro vypracování práce.

2.1 Přehled používaných mikropočítačů

První část této sekce se zabývá současnými třemi nejpopulárnějšími značkami mikropočítačů dostupné na trhu, parametry aktuálních modelů a jejich srovnáním. Běžné využití těchto MCU je pro takzvané "Do It Yourself"projekty, které mají rozsah od jednoduchých, jako je například budík, až po složité, jako je automatizovaný systém zavlažování skleníku[1].

S těmito rozsáhlými projekty je spojen pojem Internet of Things, což je paradigma, které hovoří, že dřív nebo později budou všechna zařízení (identifikována pomocí RFID) propojena internetovou sítí, kde mezi sebou budou komunikovat a plnit cíle bez zásahu lidí [7]. Jelikož pro komunikaci se často používají bezdrátová připojení, druhá část se zabývá technologiemi pro bezdrátovou komunikaci.

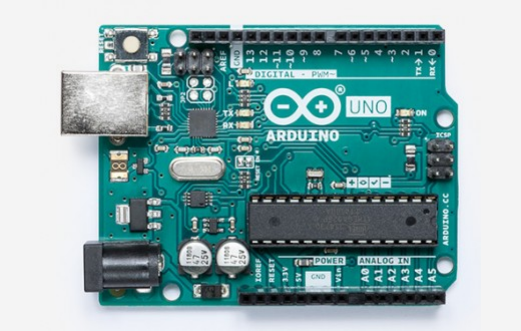
Vývojové desky Arduino a její částí

První deska Arduina se objevila v roce 2005, kdy se v italském městě Ivrea rozhodla skupina lidí z Interaction Design Institute vyrobit levný vývojový set pro studenty, kteří neměli dostatek financí pro koupi tehdejší alternativy jmenné BASIC Stamp. Po uchycení mezi zdejšími studenty se Arduino rozšířilo do celého světa, čemuž pomohl fakt, že se jedná o Open Source projekt [32].

K Arduinu bylo také vyvinuto vlastní vývojové prostředí Arduino IDE, které je napsané v jazyce Java. Programová část vznikla z výukového prostředí Processing, kde v dnešní době lze použít i jazyk Wiring [32]. Jednotlivým programům pro Arduino se říká "sketches", které mají souborovou příponu .ino [15].

[1] Příklad: https://www.instructables.com/id/Automated-Greenhouse/

[2] Převzato z https://store.arduino.cc/arduino-uno-rev3



Obrázek 2.1: Arduino Uno[2]

Většina desek od Arduina běží s procesorem od firmy Atmel [32]. Velmi populární deskou pro univerzální použití je Arduino UNO, která je vybavena 8-bitovým procesorem ATmega328, který má 32 KB vnitřní paměti, 1 KB EEPROM a 2 KB SRAM. Má periferii pro RTC, generování PWM na 6 pinů a další. [22]. Mezi zajímavé desky patří Arduino Lilypad, které jsou použity pro wearable zařízení, kde příklad takového využití může být cyklistická mikina s přišitými blinkry. Dalším takovým příkladem je Arduino Yún, které obsahuje jak klasický procesor ATmega32u4, tak ještě čip Atheros, který je schopen běhu Linuxu.

Pro připojení k součástkám jsou na Arduinu, a mnohých dalších mikropočítačích, výstupy zvané piny. U Arduina lze tyto piny dělit na napájecí, analogové a digitální. Napájecí piny slouží k dodávání napětí pro funkčnost periferií. Piny pro napájení se liší podle typu desky, můžou být 9 V, 5 V nebo 3,3 V a zemnící pin (GND). Digitální spadají pod piny "general purpose input and output"(GPIO) neboli piny pro obecné použití. Můžeme je nakonfigurovat na výstupy (3,3 V nebo 0 V) nebo očekávat na nich vstup. Některé piny mají speciální funkčnost, jako například generování PWM modulace pro rozblikání ledek nebo pohon servomotorů, podpora sériové komunikace pomocí SPI, získávání přerušení z externích modulů nebo UART komunikaci. Analogové piny slouží pro konverzi analogového signálu na digitální pomocí ADC převodníků nebo je lze použít pro komunikaci I2C [4].

Pro Arduina existují tzv. "Arduino shieldy", které slouží jako hardwarové rozšíření pro vývojové desky, například WiFi shield, Ethernet shield, Bluetooth shield, apod. Je třeba ovšem dát pozor na kompatibilitu.

Vzhledem k popularitě Arduina se objevily také levnější alternativy, nazývané klony. Lze je poznat podle pojmenování, které obsahuje nějakou část z názvu "Arduino". Některé tyto alternativy nejsou jen levnější kopií, ale jsou modifikované pro konkrétní činnost, například AndruPilot, což je klon pro ovládání létajících zařízení, nebo Rainbowduino, které je modifikované pro práci s maticovými RGB LED displeji [32].

Počítače Raspberry Pi a jeho části

Raspberry Pi není mikorpočítač jako Arduino, ale spíše mikro počítač, jinými slovy levný počítač o velikosti kreditní karty. Jsou vyvíjeny britskou nadací Raspberry Pi Foundation, která vznikla v roce 2008. První produkty se objevily až v roce 2012. Hlavní motivací vzniku bylo zlepšení znalostí žáků v oblasti počítačové techniky [23].

[3]Převzato z https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Obrázek 2.2: Raspberry Pi 3 Model B+[3]

Hlavní vlastností platformy je, že všechna potřebná elektronika (procesor, grafické jádro, operační paměť) je na jednom čipu [23]. Mezi první a stále vyvíjené desky patří modely A a B, kde rozdíl mezi modely je v počtu vstupů a velikosti paměti SDRAM. Procesory jsou od firmy Broadcom běžící na ARM architektuře, kde nejnovější modely mají 64-bitovou instrukční sadu s taktem 1,4GHz [4].

Pro funkčnost je potřeba operační systém. Existují pro to speciální systémy optimalizované pro Raspberry. Mezi zástupce patří Raspbian, který je optimalizovanou verzí linuxové distribuce Debian, který lze ovládat jak přes příkazový řádek, tak přes grafické rozhraní. Mezi zajímavé patří RISC OS, který je první operační systém pro ARM procesory [23].

Programovat lze přímo na Raspberry v mnoha jazycích. Výchozí podpora je pro jazyky Python, C, C++, Java, Ruby a Wolfram Language. Pro použití ve školách k učení základů programování je zde jazyk Scratch, kde se programy tvoří pomocí kombinací grafických bloků [13].

Raspberry Pi na nových modelech obsahuje pinové pole o 40 pinech. Na rozdíl od Arduina je zde napájení řešeno pouze přes USB. V poli se nacházejí jak piny pro GPIO, tak nekonfigurovatelné piny, na které je buď přiváděno napětí 3,3 V, 5 V nebo 0 V (GND). Stejně jako u Arduina, GPIO piny lze nakonfigurovat pro generování PWM (softwarové nebo hardwarové), SPI komunikace, I2C nebo UART [3].

Stejně jako Arduino i Raspberry má oficiální přídavné moduly, kterými lze rozšířit funkce počítače. Mezi příklady lze zahrnout dotykový displej, kameru, Power over Ethernet nebo Sense HAT, což je rozšíření vybaveno s 8x8 displejem a různými senzory jako například: gyroskop, akcelerometr, měřič vlhkosti a další [13].

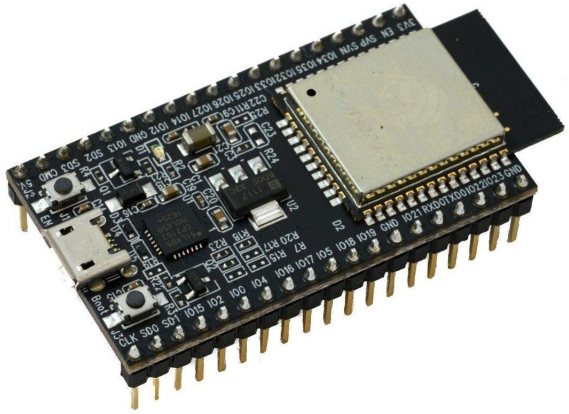
Seznámení s Espressif Systems a jejich mikropočítači

Espressif Systems je šanghajský výrobce aktuálně nejlevnějších mikrokontrolerů na trhu. Většina ESP je vybavena WiFi moduly, čímž se perfektně hodí pro jakékoli IoT projekty [27]. Dalším plusem je jejich nízká spotřeba. Běží na harvardské architektuře, což znamená, že mají separátní instrukční a datovou paměť [9][5].

Mezi hlavní zástupce zde patří ESP8266 a ESP32. ESP8266 je starší jedno-jádrový model, který poskytuje periferie, jako například UART, I2C, generování PWM modulace,

[4]Informace k modelům jsou převzaty z https://www.raspberrypi.org/products/raspberry-pi-3-model-a-plus/ https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

A/D převodník nebo rozhraní pro příjímání/vysílání infračerveného záření. Jeho následník ESP32 rozšířil počet periferií a GPIO výstupů (z 17 na 36) na čipu, přidal podporu Bluetooth 4.2 a Bluetooth Low Energy a zvedl takt jádra. Mezi novou funkcionalitu patří například rozhraní pro hardwarovou akceleraci hashování/šifrování (AES/SHA/RSA accelerator) nebo periferii pro ovládání serv a elektromotorů v RC technice (MCPWM) [27] [5]. Oba lze napájet buď přes microUSB port, který je používán pro naflashování paměti, nebo přes napájecí piny 3,3 V a zem[5].



Obrázek 2.3: ESP32-DevKitC[6]

Oficiálním jazykem je C spolu s SDK pro daný mikropočítač (ESP8266 SDK, ESP-IDF pro ESP32). Pro oba je také vyvinuto komunitní open-source "Arduino core", takže lze ESP programovat stejně jako jakékoli Arduino. Oficiální frameworky má Espressif na své Github stránce[7]. Pro ESP8266 vznikla řada dalších frameworků s podporou různých jazyků. Příkladem může být NodeMCU (s vlastní odnoží ESP8266) založený na jazyku Lua[8] nebo MicroPython, což je Python pro embedded zařízení [12].

2.2 Přehled bezdrátových technologií

Tato sekce se zabývá vybranými bezdrátovými technologiemi pro propojení a komunikaci mezi zařízeními.

Technologie Bluetooth

Bluetooth [11], definovaný standardem IEEE 802.15.1, je určen pro bezdrátovou komunikaci na krátkou vzdálenost. Byl vyvinut organizací Bluetooth Special Interest Group za účelem redukovat množství kabelů připojených k PC. Existuje několik verzí, kde aktuální je verze 5 z roku 2016, která se snaží zaměřit svou technologii na IoT zařízení.

Bluetooth pro komunikaci používá rádiové pásmo 2,402 GHz – 2,480 GHz rozdělené do 79 kanálů. Maximální dosah je 100m s přímou viditelností se zařízením, ale ve valné většině případů je nižší. Výhodou Bluetooth je nízký vysílací výkon [11]. Používá typ komunikace master-slave, kde master může komunikovat až se 7 zařízeními slave. Tato vzniklá sít se

[5]Viz https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[6]Převzato z https://cdn.sos.sk/productdata/90/d5/9dcaac3b/esp32-devkitc.jpg
[7]https://github.com/espressif
[8]https://www.nodemcu.com/index_en.html

nazývá *Piconet*. Dále existuje ještě propojení *Scatternet*, což je spojení dvou a více Piconetů tak, že jedno slave zařízení z nadřazené sítě může být master pro jinou síť [31]. Zařízení Bluetooth se dělí do tříd podle výkonu vysílání. Ovšem platí přímá úměra závisející na spotřebované energii a výkonu. Pokud se střetnou zařízení s různou třídou, musí se dostat do vzájemného dosahu, například třída 1 a 2 musí být ve vzdálenosti 10 m, aby mohli komunikovat [11].

Třída	Výkon (mW)	Výkon (dbM)	Dosah (m)	Zařízení
1	100	20	~100	Bluetooth přístupový bod
2	2.5	4	~10	Klávesnice, Myš
3	1	0	~1	Sluchátka pro mobil

Tabulka 2.1: Třídy Bluetooth (převzato z [28])

Zaplá a viditelná zařízení před připojením poskytují o sobě informace jako název a třídu, seznam služeb a technické specifikace. Při navázání spojení jsou tyto hodnoty předány a uloženy pro případné pozdější spárování, aby nemusela probíhat výměna informací. Zařízení mají 48 bitovou fyzickou adresu (BD_ADDR), ale lze i adresovat pomocí jednoduchého názvu definovaného výrobcem, které lze změnit [11].

Základní hardwarové komponenty pro Bluetooth jsou *Bluetooth radio*, což je vysílač a příjmač v pásmu 2,4GHz, *Bluetooth Link Manager*, který připravuje data pro komunikaci, a *Bluetooth Link Controller*, který se stará o navázání spojení, identifikaci, přístup a komunikaci.

Technologie WiFi

Technologie WiFi (Wireless Fidelity) [8], spadá pod několik standardů IEEE 802.11x, kde x je označení verze. Tento standard popisuje bezdrátovou komunikaci používající elektromagnetické rádiové vlny v sítích WLAN [11]. Použité frekvenční pásmo se liší dle verze, převažuje ale 2,4GHz, což vedlo k zahlcení této frekvence.

Prvotní WiFi byla vyvinuta roku 1997 (802.11), kde její rychlost byla 2 Mb/s a využívala pásmo v rozsahu od 2,400 GHz do 2,485 GHz. Z důvodu přehlcení se používal tzv. hopping, což umožňovalo změnu frekvence v určitém datovém rámci. Následně v roce 1999 přišly dva standardy: 802.11a, který pracuje s pásmem od 5,745 GHz do 5,805 GHz a tím pádem dosahuje nižších vzdáleností, a 802,11b, který má klasické 2,4 GHz pásmo. Tyto verze navzájem nejsou kompatibilní. V roce 2003 vyšel standard 802.11g, který je zpětně kompatibilní se standardem "b"a je nejvíce používán. Dalším pokrokem byl standard 802.11n, který je rychlejší a má větší vzdálenost pro přenos než jsou jeho předchůdci [11]. Srovnání je dostupné v tabulce 2.2.

Standard	Pásmo	Šířka pásma	Max. rychlost	Dosah v budově	Dosah venku
802.11	2,4 GHz	20 MHz	2 Mb/s	~20 m	~100 m
802.11a	5 GHz	20 MHz	54 Mb/s	~35 m	~120 m
802.11b	2,4 GHz	20 MHz	11 Mb/s	~35 m	~140 m
802.11g	2,4 GHz	20 MHz	54 Mb/s	~38 m	~140 m
802.11n	2,4/5 GHz	20/40 MHz	72/150 Mb/s	~70 m	~250 m

Tabulka 2.2: Vybrané standardy 802.11 (převzato z [8])

Jakékoli zařízení se standardem 802.11a/b/g/n může operovat ve 4 možných módech. *Master* mód (neboli Přístupový bod) vytváří službu s možností připojení se k síti. Toto rozhraní tvoří síť se specifickým jménem (SSID), kanálem a příslušnými službami. Master spravuje také přihlašování, zesilování paketů, atd. Zařízení připojení k master rozhraní jsou v módu *managed*, neboli také v módu klient. Komunikace mezi klienty nelze na přímo, pouze přes master zařízení, na které jsou připojeni. *Ad-hoc* mód vytváří síť "multipoint-to-multipoint", kde není žádný přímý přístupový bod a všechny zařízení mohou komunikovat se svými sousedy přímo. *Monitorovací* mód pasivně poslouchá provoz na daném kanálu a zároveň do sítě neposílá žádná data [8].

Technologie RFID

RFID [21] je bezdrátový bezkontaktní systém, který využívá elektromagnetické pole pro přenos dat za účelem identifikace zboží, osob nebo k bezhotovostní platbě. RFID je rychlejší a spolehlivější než magnetické proužky a čárové kódy, které se snaží nahradit. Zároveň může sloužit k identifikaci a sledování objektů. Základními komponentami jsou RFID čtečka a transpondér, který ukládá a uchovává informaci a na dotaz jí pošle čtecímu zařízení.

Komunikace probíhá tím způsobem, že anténa čtečky vysílá periodicky signál ve svém frekvenčním pásmu. Pokud transpondér je v dosahu čtečky a je naladěn na stejné pásmo, může čtečka přijímat informace z transpondéru. Vzhledem k tomu, že je běžný pasivní transpondér (jako například bezkontaktní platební karta), napájení je řešeno pomocí elektromagnetických vln zasílaných čtečkou. Po dosažení minimální potřebné energie na kondenzátoru, transpondér začne vysílat své informace.

Podle použité frekvence lze dělit pásma transponderů na nízké, vysoké, ultra vysoké a mikrovlnné pásmo. Nízké používají frekvenci kolem 125 – 134 kHz, což umožňuje komunikaci na velmi krátkou vzdálenost do 20 cm. Vysoké frekvence se pohybují v 13,56 MHz a umožňují čtecí vzdálenost běžně do 1 m. Tato frekvence umožňuje projít i přes stíněné nebo rušené prostředí. Velmi vysoké frekvence jsou v rozmezí od 860 – 960 MHz a dovolují rychlejší přenos na vzdálenosti do jednotek metrů. Mikrovlnné pásmo má podobnou frekvenci jako WiFi nebo Bluetooth (2,45 GHz), což dovoluje čtení až na desítky metrů.

Technologie NFC

Near field communication [21] je komunikace na velmi krátkou vzdálenost kolem jednotek cm. Rychlost přenosu není nijak rychlá, závisí na použitém standardu. Ovšem lze použít jinou technologii, např. Bluetooth, pro zrychlení přenosu, kde NFC se použije pro spárování zařízení. NFC je založeno na standardech RFID, zahrnujíc ISO/IEC 14443 a FeliCa. Toto spadá do normy ISO/IEC 18092 definované neziskovou organizací NFC Fórum. Tato technologie je často v mobilních zařízeních, kde může být využívána například pro spárování zařízení nebo jako elektronická peněženka.

Vzdálenost a rychlost přenosu závisí na standardu. Již dříve zmíněná skupina standardů spadající pod RFID, používající frekvenci 13,56 MHz a dosahují rychlosti od 106 kb/s do 424 kb/s, umí komunikovat se zařízeními vzdálena několik centimetrů. Standard ISO/IEC 15693 dosahuje mnohem větších vzdáleností (až do 1,5 m) na úkor rychlosti, která je maximálně do 26 kb/s.

Na rozdíl od RFID, kde šlo posílat pouze informace uložené na transpondéru, lze NFC použít pro přenos jakýchkoliv dat. Také umožňuje různé režimy přenosů jako například Čtení/zápis, kde je umožněno číst nebo nahrávat data na transpondér ze NFC čtečky, emulace karty, kde aktivní zařízení (například mobil s podporou NFC) simuluje pasivní (kartu), nebo peer-to-peer, kde obě zařízení si mohou mezi sebou vyměňovat informace.

Modelářské technologie

Modelářské technologie [30] zahrnují komunikaci na rádiových frekvencích 13 MHz 27 MHz, 35 MHz, 40 Mhz. Tyto frekvence jsou nejdéle na trhu, běžně se s nimi lze setkat buď u levnějších RC hraček nebo pro řízení modelů pomocí FPV (first person view) kamery. Je třeba ovšem dát pozor na určité typy vysílačů, jelikož mohou zarušit celé pásmo a tím znemožnit ovládání jakéhokoliv jiného modelu. V České Republice lze pásmo 35 MHz pouze použít jen pro létající modely, ostatní lze aplikovat na libovolný model[9]. Tyto pásma jsou tolerantnější na překážky, takže spojení za malou překážkou (např strom) netrpí velkým rušením.

V dnešní době je spíše více používané pásmo 2,4 GHz. Na rozdíl od předchozích pásem, zde vysílače nemusí nepracovat pouze na jedné frekvenci a můžou se postupně přeladťovat, což zajišťuje odolnost proti rušení. Toto umožňuje koexistenci cca 40 vysílačů na jednom místě. Nevýhodou je přímočaré šíření signálu, tzn. pokud model zajede za pevnou překážku, hrozí riziko výpadku mezi přijímačem a vysílačem.

2.3 Přehled signálů pro řízení RC modelů

Tato sekce se zabývá signály, které se používají nebo používaly pro ovládání RC modelů, jak na sebe jednotlivé signály závisí a čím nebo co ovlivňují. Mezi nejdůležitější patří PPM, PCM a PWM, které se používají pro komunikaci mezi přijímačem, vysílačem a servy.

S těmito signály je také spjatý pojem *modulace*, což je proces změny charakteristiky nosného signálu modulačním signálem, jako například amplituda, frekvence, apod. Tímto procesem je možno přidat nosnému signálu informaci, která je následně zasílána zařízením [35] (v našem případě to jsou serva a přijímač).

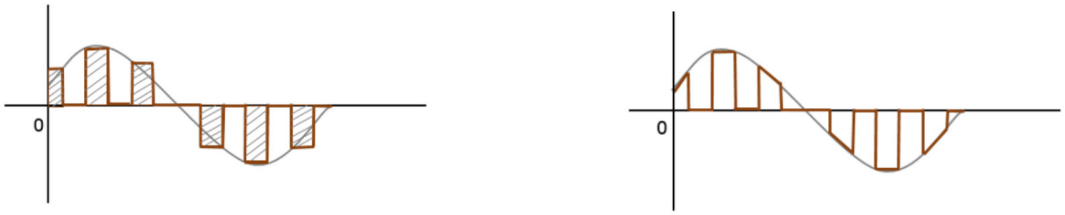
Modulace lze rozdělit na analogovou a pulzní. U analogové modulace se používá jako nosný signál vysokofrekvenční harmonická vlna, která je modulována analogovým modulačním signálem [35]. Impulzní modulace má jako nosný signál periodickou posloupnost obdélníkových impulzů. V těchto kapitolách jsou vypsány modulace, které jsou spjaté s RC technikou.

Pulzní amplitudová modulace (PAM)

Pulzní amplitudová modulace patří mezi základní zástupce impulzní modulace. Jde o typ vzorkování, kde v pravidelných intervalech je snímán vzorek úměrně k amplitudě modulačního signálu. Informace je posílána jako posloupnost vzorků měnících se amplitud.

Podle typu vzorkování se dělí modulace na "Flat Top"neboli rovnoměrný a přirozený PAM [35]. Rovnoměrná modulace po celou dobu trvání impulzu uchovává jednu hodnotu vzorku amplitudy. Tím pádem vršek výsledného signálu je plochý. Obvod, který se pro toto

[9] https://www.ctu.cz/1/download/Opatreni%20obecne%20povahy/VO_R_15_08_2005_27.pdf
[10] Převzato z https://www.elprocus.com/pulse-amplitude-modulation/



Obrázek 2.4: Vlevo je příklad rovnoměrné PAM a vpravo přirozeného PAM[10]

používá, je *Sample and Hold*. Přirozená modulace naopak následuje amplitudu po celou dobu trvání impulzu, tudíž více připomíná originální signál.

Tento signál je historicky prvním způsobem kódování. Bohužel je velmi náchylný na rušení a zkreslení, tudíž se v RC technice nepoužívá. Používá se při vzorkování pulzní kódové modulace [29][35]. Existují i nové verze jako PAM5, která je využívána v ethernetových sítích pro zasílání dat rychlostí 1 Gb/s, nebo PAM16, která je používána pro rychlosti až 10 Gb/s, kde číslo značí počet úrovní amplitudy pro zakódování dat. [14].

Pulzní polohová modulace (PPM) a použití při vysílání

Jeden z používaných modulací je pulzně polohová modulace. Zde výstupní hodnota modulačního signálu nezávisí na amplitudě nebo šířce (jsou konstantní), ale je předána jako poloha impulzu v daném rámci, kde délka rámce běžně odpovídá vzorkovací periodě [35].

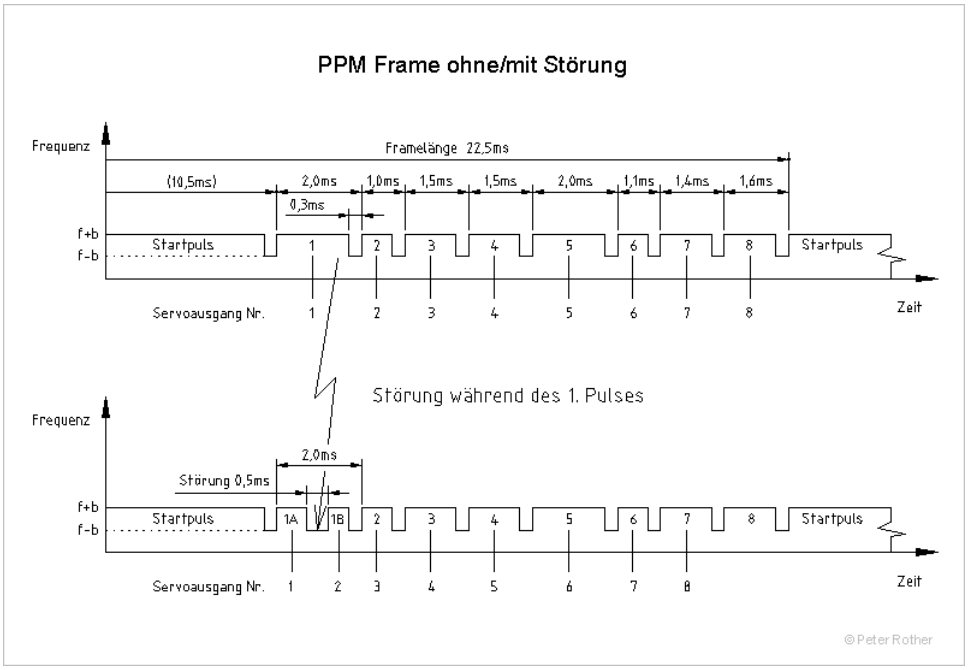
V RC technice se pomocí PPM zakódovávají hodnoty jednotlivých kanálů. Opakovací perioda kompletního běžného signálu PPM (rámce) je zpravidla 22,5 ms , což umožňuje vložení až 8 kanálů. Jednotlivé kanály se přenáší postupně za sebou a jsou odděleny dvěma synchronizačními impulzy jednotné šířky. Je třeba, aby vysílač zasílal synchronizační pulzy z důvodu udržování správné pozice pulzů u obou zařízení [29].

Příklad takového rámce je ukázán na obrázku 2.5 vrchní, kde lze vidět 8 signálů PWM pro jednotlivé kanály za sebou. Interval rámce začíná synchronizačním impulzem opačné polarity. Pak následují PWM signály, které v PPM rámci můžou trvat 0.7–1.7 ms v kladné polaritě a 0.3 v záporné (ukončovací impulz), což dohromady dává možnost mít 1–2 ms impulzu PWM. Jednotlivé kanály jsou odděleny ukončovacím impulzem. Za posledním kanálem následuje ukončovací impulz a následný interval bez jakýchkoli impulzů, trvající minimálně 3,8 ms, který vyplňuje zbytek rámce [29]. Tato výplň může trvat od 6.5 ms až po 16.5 ms v závislosti na šířce jednotlivých PWM signálů [24].

Výhody použití PPM v RC technice je výroba levných a malých součástek nepotřebujících procesor pro zpracování velkého množství dat, jelikož vysílaný řetězec je velmi jednoduchý. Nevýhodou ovšem je nemožnost zjistit chybné impulzy ve vysílaném rámci (zkreslení viz Obrázek 2.5 spodní), což způsobí nevyzpytatelné chování modelu. Toto lze zpozorovat, když se model dostává na konec dosahu vysílače [26].

Mezi RC vybavení, které používají PPM patří [29]:

- RC vysílače/přijímače
- Autopiloty/Stabilizační systémy
- Moduly pro připojení k počítači



Obrázek 2.5: Příklad [26] PPM rámce generovaného vysílačem. Vrchní graf je bez zkreslení a spodní se zkreslením rámce

Pulzně kódová modulace (PCM) a použití při vysílání

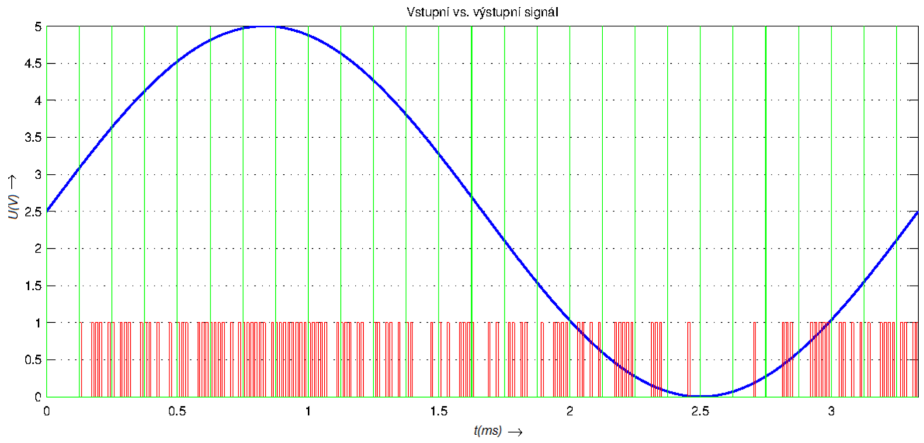
Pulzní kódová modulace je určena pro převod analogového signálu na číslicový, kde informace v okamžitých vzorcích analogového signálu je reprezentována digitálními slovy v sériovém bitovém toku [19]. Nejčastěji je bitový tok reprezentován jako unipolární RZ (Return to 0) signál [35], což znamená že napěťové hodnoty jsou pouze kladné, kde polovina vyšších kladných hodnot je reprezentována jako logická "1"a nižší kladné hodnoty až 0 V je reprezentováno jako logická "0"[19]. PCM modulace se tvoří pomocí tří operací, mezi které patří vzorkování, kvantování a kódování.

Při *vzorkování* je vstupní analogový signál vzorkován podle zadané vzorkovací frekvence. Výsledek vzorkování je flat-top PAM, kde délka pulzu je rovna vzorkovací periodě [35].

Následujícím krokem je *kvantování*, kde jednotlivé vzorky PAM signálu se převedou na hodnotu kvantizační hladiny. Počet hladin je dán počtem bitů výstupního signálu a lze je zjistit ze vzorce $N_{KV} = 2^{nb}$, kde N_{KV} je počet kvantizačních hladin a nb (number of bits) je počet bitů výstupního signálu. Například pro 8 bitový signál PCM je počet hladin 256 [35].

Posledním krokem je *kódování* PCM modulace. Vstupním signálem je zde kvantovaný signál s jednotlivými kvantovacími úrovněmi a výstupním je kódovaný digitalizovaný signál s informacemi o původním analogovém signálu. Kvantovaný signál je tedy převeden na posloupnost bitových slov, kde jednotlivé slovo odpovídá konkrétní kvantifikační úrovni [19]. Pokud se vezme 8 bitové kódování a je přijata úroveň 128 výstupní signál bude posloupnost bitů 10000000. Kódování se může dělit podle napěťového rozsahu na *unipolární*, kde hodnoty kódovaného signálu jsou pouze kladné, a *bipolární*, kde hodnoty kódovaného signálu mohou být kladné i záporné [19].

[12] Převzato z *Impulzové modulace* [35]



Obrázek 2.6: Porovnání vstupního signálu (analogový signál, modrá) a výstupního signálu (PCM, červená)[12]

V počítačové technice je PCM využíváno pro kódování audio streamů. Konkrétně je používáno ve standardu G.711 pro přenos hlasu přes Internet (VoIP) [20], jako nekomprimované audio soubory formátu WAV, AIFF nebo způsob zakódování audia na DVD či Blu-Ray disk pomocí lineární PCM modulace [1] [2].

V RC technice se PCM používá pro komunikaci mezi přijímačem a vysílačem stejně jako PPM. Rozdíl spočívá v možnosti zakódování například jednoznačného kódu přijímače, aby se dva vysílače zároveň nerušili, nastavení fail safe neboli reakce serv modelu při výpadku signálu a kontrolní součet pro zjištění, že model obdržel poškozený rámec [29].

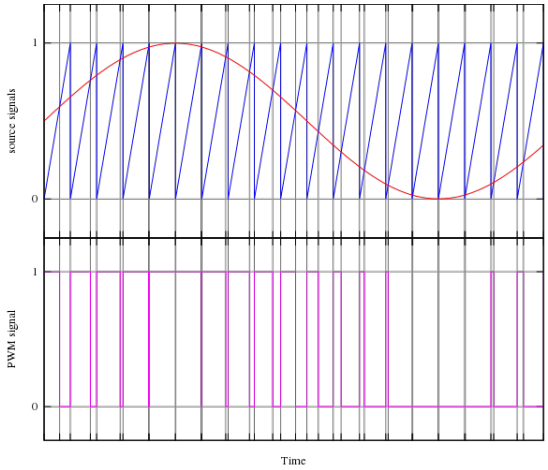
Výhodou PCM je tedy anulování nevyzpytatelného chování, jako je například chvění serv, pomocí kontrolních součtů a fail safe mechanismu, což umožňuje pilotování i na větší vzdálenosti než s PPM. Dále díky kontrolním součtům také nemůže dojít k poškození serv z důvodu chyby v zasílaném signálu. Nevýhodou je cena, vzhledem k tomu, že jak přijímač tak vysílač mají mikroprocesory pro de/kódování modulace. Problém je také při zjišťování přenosové kvality mezi přijímačem a vysílačem. PCM se umí zotavit z krátkodobých rušení, tím pádem pilot nezjistí až do poslední chvíle, jestli je něco špatně, což může vést k havárii modelu [29][25].

Pulzně šířková modulace (PWM) a ovládání serv

Poslední modulace, která je spjata s RC technikou je Pulzně šířková modulace. U tohoto typu je na základě vstupního analogového signálu ovlivňována šířka výstupního obdélníkového impulzu. Vstupní signál je vzorkován stejně jako u PAM, akorát zde se pro vzorkování nepoužívá obdélníkový signál, ale pilový [35].

Z obrázku lze vidět jak je modulace vytvořena pomocí vzorkování pilovým signálem. Pokud je hodnota vstupního analogového signálu vyšší nebo rovna než pilový, výstupní PWM se bude nacházet v logické "1". V momentě, kdy bude hodnota pilového signálu vyšší než analogového, výstupní PWM se překlopí do logické "0"[35]. Poměr mezi dobou, kdy je výstup v logické "1"nebo "0"se říká střída.

[14]Převzato od CyrilB, https://commons.wikimedia.org/wiki/File:Pwm.svg (CC BY-SA 3.0)



Obrázek 2.7: Nahoře je ukázán vstupní (červený) signál a vzorkovací (modrý). Spodní reprezentuje výstupní PWM.[14]

PWM lze použít například pro kontrolování jasu na LED diodě, případně pro generování zvukových efektů například pomocí vývojových desek[15] . V RC technice se PWM používá zejména ke komunikaci se servy a regulátory elektromotorů. Rozsahy šířky impulzů jsou zvoleny od 1ms do 2ms, kde tyto hodnoty představují krajní polohy serv, a opakovací perioda PWM modulace bývá 20ms. Mezi další komponenty, které využívají PWM patří osvětlení, Failsafe moduly, stabilizační systémy a testery serv [29].

[15]Příklad: https://www.hackster.io/106958/pwm-sound-synthesis-9596f0

Kapitola 3

Přehled existujících řešení ovládání RC modelů

V této sekci jsou vypsány již existující zajímavá řešení, které byly použity pro ovládání a automatizaci RC modelů. Všechny tyto řešení se snaží ovládat model pomocí různých typů ovladačů a vývojových desek.

3.1 Předdefinované pohyby quadrokoptér a automatizace modelů

Někteří výrobci quadrokoptér dávají do svých produktů předdefinované chování, které zjednodušují ovládání, jako například udání trajektorie, kterou má quadrokoptéra vykonávat, stabilizace vůči větru nebo možnost dělat otočky ve vzduchu stisknutím tlačítka[1], a tím dělat triky jednodušeji bez možných havárek a poškození quadrokoptéry.

Pro provedení otočky (barrel-roll) je třeba na ovladači držet speciální tlačítko, kterým zajistí provedení otočky. Následně je třeba říct, na jakou stranu má provést otočku, což je provedeno udáním směru náklonu pravým joystickem. Jak je uvedeno ve videu, tohoto základního automatizovaného triku lze využít pro složitější trik, jako například několik otoček za sebou (multiflip) [2].

Quadrokoptéry mohou taky využívat pro své ovládání mobilní zařízení a tím používat jejich vlastnosti pro své schopnosti, třeba GPS lokace pro automatické vrácení quadrokoptéry k ovladači nebo jej používat pro výstup kamery [3].

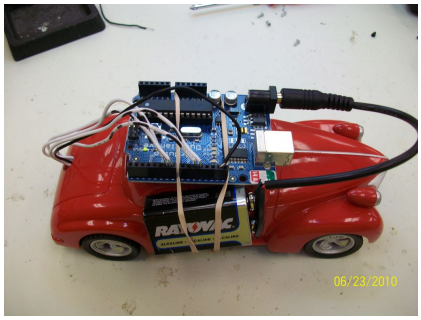
Kromě tohoto existují na internetu řešení, jak automatizovat různé modely. Jedno z nich[10] pojednává o modifikaci modelu tak, že pro ovládání bylo použito Arduino, kde do něj bylo předdefinováno, že model vytvářel svou dráhou pomyslnou osmičku. Jiné řešení [4], zaměřené na letadlo, ukazuje pomocí UAV Dev Board automatické vzlétání a přistávání, které řešitel aktivuje z ovladače.

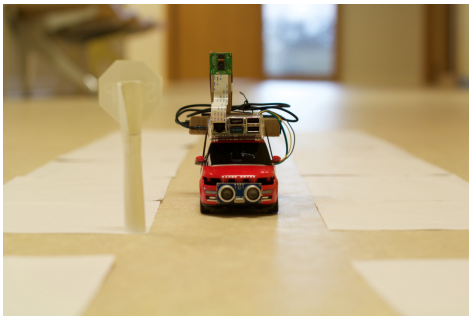
[1] Viz např. https://www.amazon.com/Beebeerun-Quadcopter-Headset-Compatible-Induction-Resistance/dp/B071X4JR7W

[2] Vycházím z videa: https://www.youtube.com/watch?v=rkzlA_3n_04

[3] https://www.youtube.com/watch?v=sL33yPYBdaE

[4] https://www.youtube.com/watch?v=7KyuS1w1t94



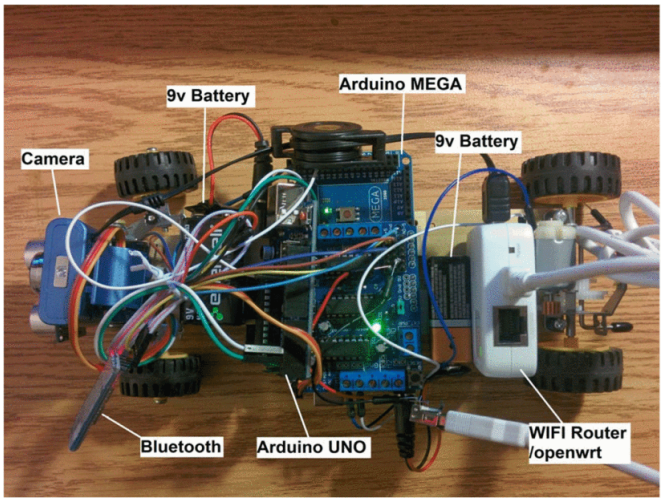


Obrázek 3.1: Na levo lze vidět model s jedním předdefinovaným pohybem, napravo samořídící model[6]

Pokud se bavíme o kompletní automatizaci, lze vzít v potaz řešení Self Driving RC Car [34], kde úkolem bylo, aby model uměl udržet se na dráze, detekovat dopravní značky a umět zastavit před překážkou. Pro ovládání modelu používá Arduino, pro sběr dat a rozhodování je použit Raspberry Pi. Je zde použita kamera pro detekci dráhy a značek, a ultrazvukový senzor HC-SR04 pro prevenci střetu s překážkou. Pro klasifikaci je použito neuronových sítí knihovny OpenCV.

3.2 Ovládání s pomocí mobilního zařízení

Toto řešení je nejpopulárnější vzhledem k ceně a dostupnosti komponent [33][6][16]. Používají se pro to mikropočítače kvůli jejich flexibilitě a jednoduchosti modifikace. Společnými vlastnostmi těchto konkrétních řešení je použití vývojových desek Arduino pro ovládání servo motorů. Implementační jazyky jsou v těchto řešeních stejné. Jazyk pro Arduino je C s použitím knihoven pro přístup k periferiím vývojové desky. Pro Android používají jazyk Java, jelikož to je primární jazyk pro vývoj na Android zařízení.



Obrázek 3.2: Příklad zapojení řešení, kde je Bluetooth použit pro příjem povelů a směrovač pro zasílání výstupu kamery[8]

[6]Převzato z *Autonomous Control of RC Car Using Arduino* [10] a z *Self Driving RC Car* [34]
[8]Převzato z *AndroRC* [16, Obrázek 2.]

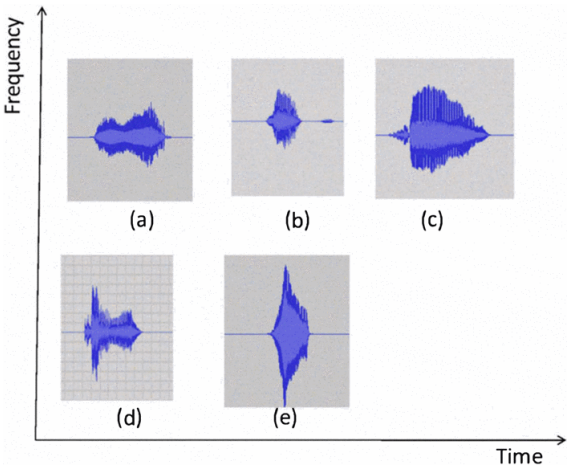
Vysílací a přejímací zařízení se liší v implementacích. Jedno z možností je ovládání přes mobilní telefon s OS Android a kde přijímač je Bluetooth modul, který je zapojený k Arduinu [33][16] a je používán pouze pro zasílání povelů k pohybu. Tento modul je upřednostněn v těchto konkrétních řešeních z důvodu menší spotřeby energie než WiFi. Další variantou je implementace [6], kde mobil je právě přijímačem a je propojený s vývojovou deskou přes USB, což umožňuje využití vlastností telefonu, například kamera, blesk apod. Ovladačem je počítač, který posílá povely přes WiFi, kde model i ovladač jsou připojeny k lokálnímu síťovému zařízení (například směrovač) pomocí kterého si vyměňovali informace. Alternativou může být řešení [16], kde používáme Bluetooth pro pohyb a zároveň směrovač jako vysílač obrazu z kamery na mobilní zařízení. Pro toto řešení ovšem byly použity dvě vývojové desky Arduino, kde jedna přijímala a řídila pohyb a druhá zpracovávala výstup kamery a pomocí dříve zmíněného směrovače posílala obraz na mobil.

Způsob ovládání je také různorodý. Některá řešení mají implementaci pomocí 4 tlačítek (dopředu, dozadu, doleva, doprava) a posuvníkem pro zvolení rychlosti [6]. Dalším způsobem je ovládání RC modelu pomocí senzoru orientace mobilního zařízení [16], kde Y osa ovládá servomotor a X osa servo řízení, a tím pádem na obrazovku mobilu lze zobrazovat jiné informace, v případě tohoto článku je to výstup z kamery na modelu.

3.3 Ovládání pomocí hlasových povelů

Řešení vychází ze článku "Operation of a radio-controlled car by voice commands"[18], kde studenti univerzity Mahidol v Thajsku zkoumali možnosti ovládání auta pomocí hlasových povelů

Návrh spočíval v předávání povelů přes počítač, kde pro rozpoznávání řeči použili techniku Skrytého Markovova modelu. RC model reagoval na thajské povely Dopředu, Zpátky, Doleva, Doprava a Stop, které předávali počítači pro zpracování. Pokud povel nebyl rozeznán, byl použit nejpodobnější. Hlasový signál byl převeden na digitální pro poslání na paralelní port, kde digitalizace spočívala v převedení pokynů na hexadecimální číslo. Toto bylo následně posláno na konvertor digitálního signálu na rádiový přes ovladač RC modelu.



Obrázek 3.3: Signály pro jednotlivé povely: (a) Dopředu, (b) Dozadu, (c) Doleva, (d) Doprava, (e) Zastav[10]

[10]Převzato z *Operation of a radio-controlled car by voice commands* [18, Obrázek 3.]

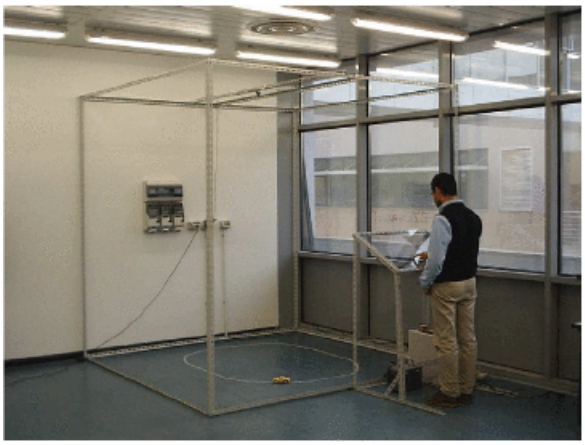
Softwarová implementace spočívala v použití souboru nástrojů pro Skrytý Markovův model a open-source slovníku Julius pro rozpoznávání řeči. Uživatelské rozhraní bylo vytvořeno přes Visual Basic 6, kde byl seznam posledních použitých pokynů, poslední povel, apod.

Výsledek byl testován v různých prostředích, kterými byly tichá místnost, kancelář a hlučná místnost. Úspěšnost rozpoznání pokynů měla nejlepší výsledky pro tichou místnost a kancelář. Dále některé pokyny, které byly podobné, avšak měli jiný význam, byly interpretovány stejně (jako příklad byl uveden "Turn Right"a "Turn Light"[18, str. 17])

3.4 Ovládání pomocí gest rukou

Toto řešení[17] se zabývá možností, jak nahradit běžné zařízení pro ovládání (např. klávesnice, myš, atd.) rozhraním, které bude snímat prsty rukou a na základě jejich polohy provádět dané akce. Experiment s rozhraním byl testován na RC modelu, který se mohl pohybovat vně omezené dráhy.

Tento systém se skládal ze dvou kamer, kde jedna snímá dráhu a polohu RC modelu a druhá čeká na uživatelský vstup a je umístěna pod uživatelskou konzolu. Ta se skládá dále z plexiskla a projektoru, který zobrazuje na plexisklo vstup kamery nad dráhou. Model se tedy pohyboval podle toho, kam uživatel ukazoval prstem na plexisklu.



Obrázek 3.4: Systém kamer a konzole pro ovládání[11]

Pro zpracování obrazu z konzole byl využit stolní počítač, který následně ovládal model. Při snímání ruky bylo třeba zjistit ze vstupu co je prst a co ne. Proto byl použit algoritmus, který pro bod na ruce zjistil její směrnici a tečnu na ní, z jejíž délky mezi dalším bodem ruky zjišťoval, zda je tečna krátká (a tudíž se jedná o prst) nebo dlouhá (jedná se o dlaň). Následně se rozhodne o jaké gesto (neboli kam ukazuje prst) se jedná. Pro ošetření nepřesnosti snímání je před samotným ovládáním třeba podstoupit trénovací fázi, kde algoritmus získá odchylku souřadnic, aby lépe mohl reagovat na uživatelské vstupy.

Z výsledků se zjistilo, že tento způsob ovládání je pro lidi více přirozený a pohodlnější než konvenční rozhraní, jako je například joystick nebo klávesnice a myš. Řešitelé navrhují implementovat tento způsob ovládání do embedded systémů pro lepší efektivitu a snížení ceny.

[11] Převzato z *A vision-based user interface for real-time controlling toy cars* [17, Obrázek 1.]

Kapitola 4

Zhodnocení současného stavu a specifikace zadání

4.1 Zhodnocení existujících řešení

V této sekci bych rád postupně zhodnotil řešení výše vypsané.

Řešení s předdefinovanými schopnostmi bylo pouze nalezeno u quadrokoptér, aut, lodí i letadel. Quadrokoptéry měly tyto schopnosti definovány při výrobě, tudíž fungují přesně, ale nelze přidávat další. Někteří uživatelé by rádi specifikovali vlastní pohyb pro více precizní nebo extravagantní trik. Toto řešení bylo nalezeno pro RC auta, lodě a letadla, kde řešení byla zpracována jako hobby projekty na různých vývojových deskách. Zde je možné si vlastní chování naprogramovat, nicméně v případě, že by tato řešení byla komerčně rozšířena, průměrný uživatel stále postrádá vzdělání, aby si uměl doprogramovat chování sám.

Typy RC modelů	Způsoby ovládání	Způsoby automatizace
Auto	Ovladač, PC, mobil	Speciální chování, Plně autonomní
Loď	Ovladač, PC	Autopilot[1]
Letadlo	Ovladač	Autopilot
Quadrokoptéra	Ovladač, mobil	Autopilot, Speciální chování

Tabulka 4.1: Porovnání nalezených řešení ovládání a automatizace pro jednotlivé typy modelů

Ovládání pomocí programovatelného počítače je, dle mého názoru, nejvíce flexibilní řešení, které v dnešní době existuje. Umožňuje, kromě možnosti ovládání serva a elektromotoru pomocí PWM, také možnost rozšířit původní vlastnosti vozidla, například blinkry, senzory, kamery apod. Ovládání by mohlo být přes zařízení, na které by byla vyvinuta aplikace pro řízení a které by podporovalo daný přenosový protokol pro spojení s mikropočítačem. Ovšem neomezoval bych se pouze na použití Arduina, jelikož v dnešní době lze najít levnější a výkonnější mikropočítače, jako například ESP32, který na rozdíl od většiny vývojových desek Arduino, má v sobě zabudovanou anténu buď pro Bluetooth nebo WiFi, kde u Arduina je třeba anténu dokoupit ve formě samostatného modulu. Také ve výše zmíněném textu je ve dvou řešeních použit Bluetooth, což je ve většině případů méně efektivní než WiFi, která umožňuje komunikaci na delší vzdálenost.

[1] https://www.instructables.com/id/Boat-Autopilot/

	Pásmo	Max. rychlost	Maximální možný dosah
WiFi	2,4/5 GHz	150 Mb/s	~250 m
Bluetooth	2,4 GHz	2 Mb/s	~100 m
NFC	13,56 MHz	424 kb/s	1,5 m

Tabulka 4.2: Porovnání přenosových technologií

Řešení pomocí hlasových příkazů je zajímavé v tom, že model nemusíme ovládat pomocí klasického ovladače jakékoli podoby, ale stačí říkat jak jet. Problém tohoto řešení je podle mě v případě, pokud chceme přesně zřetězit triky za sebou a rozpoznávání řeči nám v jistý okamžik selže, což může vést k bouračce modelu. Propojení ovladače s počítačem za účelem udávání příkazů mi přijde z hardwarového hlediska zajímavé, nicméně toto zařízení není mobilní jako například telefon, kde mikrofon by mohl být použit ze sluchátek.

Poslední řešení se týkalo ovládání modelu pomocí ukazování na projekci dráhy. Zde bych viděl možnost variací povelů pro různé triky specifickými gesty ruky. Ale pro funkčnost je třeba složitý systém, který je těžko přenositelný.

4.2 Cíle a požadavky na výsledné řešení

Cílem této práce je využití mikropočítače k ovládání RC modelu auta přes mobilní zařízení a umožnit mu provádění triků stejně, jako to je možné na quadrokoptérách. Žádné základní ovladače RC aut toto neumožňují, proto jsem se rozhodl úkol implementovat. Tudíž je potřeba vymyslet takové chování, které by bylo pro RC auto zajímavé a pokud možno zapojit do toho i ovládání pro udání směru akce. Kromě tohoto úkolu je třeba zachovat původní funkcionalitu ovladače, co se týče nastavování limitů serv a jeho chování a případně rozšířit o vlastnost aretace ovladače pro udržování rychlosti. Model auta jsem zvolil kvůli menšímu možnému počtu oprav, které by bylo vyšší například u modelů letadel, a dostupnosti povrchu pro pohyb, což vyřazuje modely lodí.

4.3 Technické zadání

Na základě schůzek s vedoucím práce byl upřesněno pracovní zadání, které bude splňovat tyto body:

- RC model bude ovládán pomocí mikropočítače připojeného namísto původního přijímače.
- Mikropočítač bude umět generování PWM modulace, pro možnost ovládání serv, a bude mít modul pro bezdrátové připojení.
- Implementačním jazykem na mikropočítači bude jazyk C, kvůli nízkoúrovňovému přístupu k periferiím.
- Ovládání bude přes aplikaci běžící na mobilním zařízení s OS Android napsaná v jazyku Java.
- Aplikace bude umožňovat ovládání modelu, spouštění předdefinovaného chování a možnost nastavit limity pro servo řízení a elektromotor.

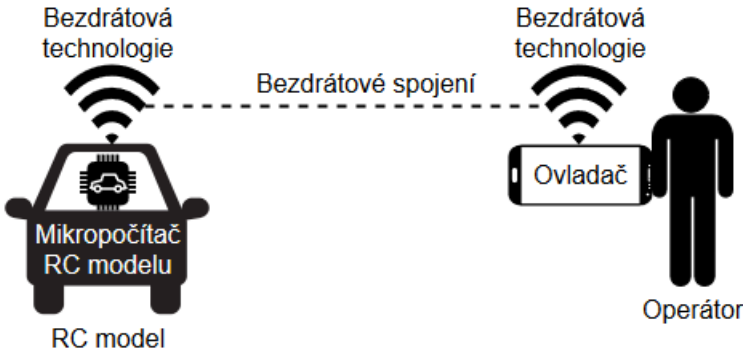
- Komunikace mezi mobilním zařízením a modelem bude probíhat bezdrátově a pokud možno na velkou vzdálenost.
- RC model bude umět vykonávat triky, jako to například umí quadrokoptéry

V následující kapitole jsou body ohledně mikropočítače, aplikace, přenosovém protokolu a předdefinovaných pohybů probráno více do hloubky

Kapitola 5

Návrh systému a jeho dílčích částí

Navrhovaný systém je složen ze čtyř částí, RC auto, ovladač pro řízení, operátor a bezdrátová technologie, která umožňuje komunikaci mezi ovladačem a modelem. Ovladač bude v rukou uživatele, který po zapnutí napájení na RC autě a následném otevřením aplikace se bude moci připojit na mikropočítač pomocí bezdrátové technologie. Ten bude připevněn na modelu a bude umožňovat komunikaci se servy přes PWM. Pro napájení auta bude třeba pouze jeden akumulátor, který bude napájet jak servomotory, tak mikropočítač.



Obrázek 5.1: Model systému

Triky, které vyžadují specifický pohyb modelu budou implementovány přímo na mikropočítači a budou vyžadovat specifický vstup od uživatele, jako například výběr triku a případné udání směru. Vylepšení týkající se ovládání budou na ovladači, například aretace ovladače, a bude je možnost aktivovat či deaktivovat.

5.1 Výběr mikropočítače a jeho připojení

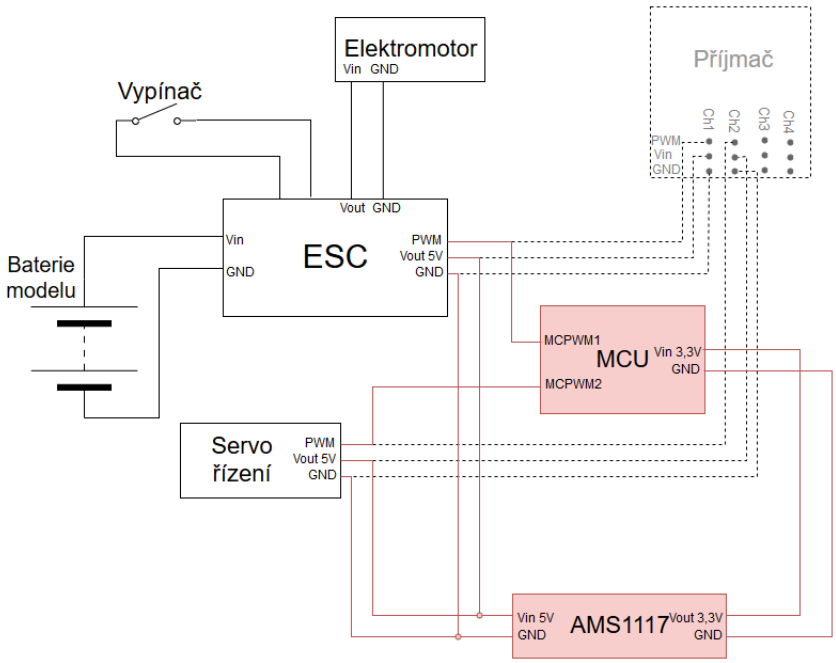
Vybíralo se mezi Arduino UNO a ESP32 na základě velikosti, schopnosti generovat PWM modulaci a možnosti přejímat data bezdrátově. Arduino nemá nativně na vývojové desce bezdrátový modul a pro jeho potřebu je třeba jej dokoupit separátně. ESP32 má vestavěnou anténu, která slouží pro připojení přes Bluetooth nebo WiFi. Tudíž byl vybrán ESP32, konkrétně vývojová deska DevKitC.

Výběr mikropočítače se taky vztahoval na omezení implementačního jazyka. Bylo ovšem potřeba vybrat framework, který umí přistupovat k periferiím MCU. Na oficiální Github

stránce Espressif Systems[1] jsou na výběr 2 frameworky vyhovující úkolu: ESP-IDF (Espressif IoT Development Framework), oficiální vývojový framework pro ESP-32, a Arduino-ESP32, framework pro práci s ESP-32 pomocí Arduino sketches a jeho syntaxe. Rozhodl jsem se použít ESP-IDF vzhledem k lepší práci s periferiemi časovačů pro generování PWM modulace a kvůli možnosti jednoduššího vytváření paralelních procesů.

Obvod připojení mikropočítače k RC modelu

Pokročilé RC modely se skládají ze separátních součástek pro případnou výměnu a zlepšení modelu. Na obrázku 5.2 znázorněno šedými přerušovanými čarami jsou zobrazeny původní zapojení. Baterie napájí elektrický regulátor otáček (zkráceně ESC), ze kterého se dále rozvádí napětí pro ostatní komponenty, kterými jsou elektromotor a přijímač. Z přijímače se dále napájí všechny zapojené serva paralelně. Signály jsou posílány z přijímače jednotlivým servům a ESC na každém kanálu.



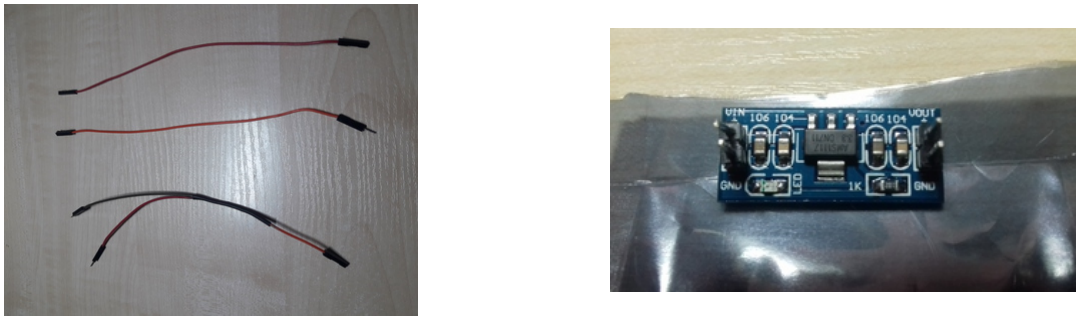
Obrázek 5.2: Obvod zapojení RC modelu. Šedou a přerušovanou čarou jsou zobrazeny původní hardware/zapojení a červenou je nový hardware/zapojení.

Pro možnost ovládání modelu přes mikropočítač je tedy třeba nahradit přijímač tímto zařízením. Je třeba zohlednit fakt, že přijímač je napájen výstupem z regulátoru otáček (ESC), ze kterého vystupuje napětí 5 V. Pro napájení našeho mikropočítače ESP32 je potřeba 3,3 V +-0,3 V[5]. Dále je třeba brát v potaz, že serva potřebují vyšší napětí a proud (tj. 5 V/2 A).

Řešení je na obrázku 5.2 červenou barvou. Pro snížení napětí byl použit stabilizátor AMS1117-3,3, který zajišťuje regulaci z 5 V na 3,3 V +- 0,1 V a výstupní proud v minimální hodnotě 11 mA a maximální 800 mA[2]. Maximální proud, který je potřeba pro funkcionalitu ESP32, je 350 mA, kde 50 mA si vezme jádro MCU při maximálním vytížení,

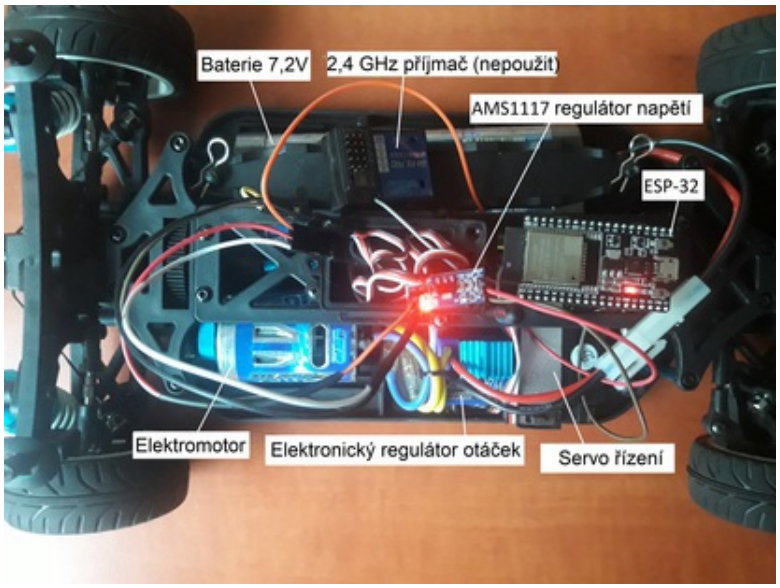
[1] https://github.com/espressif
[2] http://www.advanced-monolithic.com/pdf/ds1117.pdf

100 mA spotřebuje vysílač a 200 mA přijímač modulu WiFi. Tudíž se vleze do intervalu $< 11,800 >$ mA.



Obrázek 5.3: Na levém obrázku lze vidět typy propojů, na pravém je stabilizátor napětí AMS1117.

Pro připojení byly použity propojovací spoje zobrazené na obrázku 5.3 vlevo. Pro vytvoření paralelního propojení mezi servy a stabilizátorem napětí byl vytvořen speciální propoj.



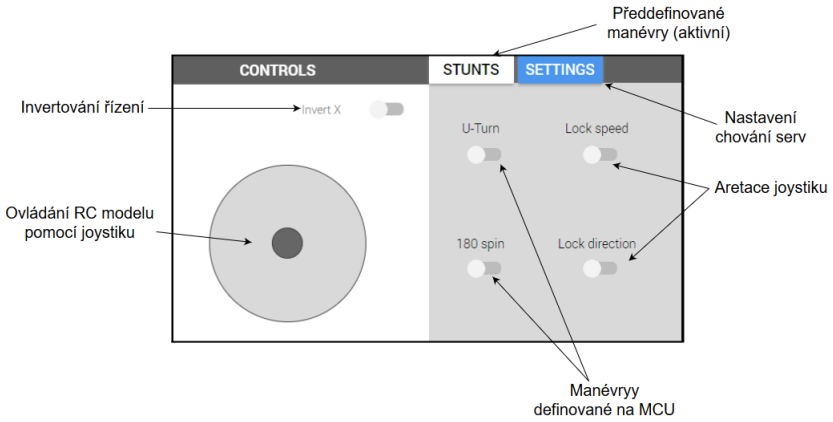
Obrázek 5.4: Modifikovaný RC model

Na obrázku 5.4 lze vidět modifikovaný RC model. Stabilizátor napětí se nachází na původním místě, kde se nacházel přijímač. ESP32 je ukotven pomocí propojů, které jsou propleteny s kostrou modelu.

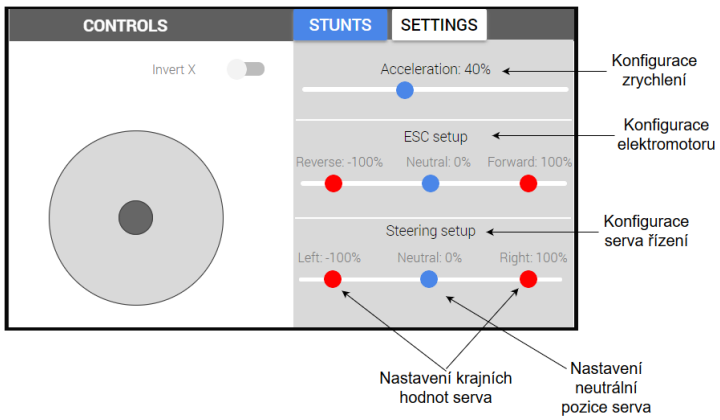
5.2 UI aplikace pro ovládání

Z designového hlediska by se měla aplikace skládat ze dvou sekcí na displeji, kde jednou sekcí se bude ovládat a druhá bude pro předdefinované chování modelu. Na obrázku 5.5 je design, který byl použit. Na levé straně se nachází joystick, pro schopnost ovládání jednou rukou. Zrychlení/zpomalení se ovládá pohybem kuličky joysticku nahoru nebo dolů

a ovládání směru pohyby doleva nebo doprava. Je možné změnit řízení přepínačem "Invert X"nad joystickem. Joystick se po puštění v implicitním nastavení vždy vrátí do středu, čímž se servo a elektromotor dostanou do své neutrální polohy (neboli klidové polohy), kdy se model nehne a kola jsou otočena vpřed. V pravé části se nachází přepínače pro jednotlivé triky. Přepínač byl zvolen proto, aby událost byla spjata s pohybem joysticku.



Obrázek 5.5: UI aplikace s předdefinovanými vlastnostmi



Obrázek 5.6: UI aplikace pro nastavení serv

Dále bylo třeba zachovat původní nastavení serv jako na ovladači. Od toho slouží záložka "Settings". Po kliknutí na záložku se zobrazí posuvníky pro nastavování vlastností serv (viz obrázek 5.6). Mezi ně patří nastavení neutrální polohy serva, dále hraniční polohy serv, které by měly být kvůli zabránění poškození serva. Je zde posuvník pro nastavení limitů elektromotoru ("ESC setup") a pro serva řízení ("Steering setup"). Je zde i posuvník pro výběr zrychlení ("Acceleration") modelu z aktuální na požadovanou. Pro návrat na předdefinované manévry je v horní liště taky záložka "Stunts".

5.3 Návrh použité přenosové technologie a protokolu

Jak již bylo zmíněno, bezdrátové technologie, které ESP32 podporuje, jsou Bluetooth a WiFi. Jelikož již bylo v zhodnocení řešení stanoveno, že WiFi je lepší jak rychlostně tak v dosahu než Bluetooth, bude použita pro přenos instrukcí WiFi. Mikropočítač se bude

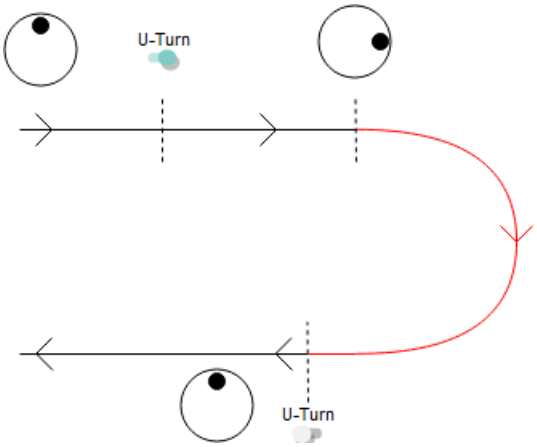
tvářit jako přístupový bod, na který se bude možno mobilním zařízením připojit a následně z něj posílat na model pohybové instrukce.

Transportní protokol by měl být co nejrychlejší, aby nevznikalo zbytečné zpoždění mezi instrukcemi. Tomuto omezení nejvíce vyhovuje transportní protokol UDP. Ovšem je třeba počítat z možnou ztrátovostí paketů na cestě z ovladače na model, tudíž je třeba implementovat fail-safe mechanismus, který pokud neobdrží zprávu za určitý čas, tak model vrátí do neutrálního stavu.

Aplikační protokol musí předávat informace o pohybu a aktivním triku. Tudíž ovladač by měl posílat zprávy, kde bude zapsáno aktuální udávaná rychlost (neboli souřadnice Y-ové osy joysticku), směr zatočení (X-ová osa joysticku), trik, který se má provést, a sériové číslo, které udává jestli byl trik proveden nebo ne. Sériové číslo si mikropočítač bude pamatovat a tím zamezí provedení již dokončeného triku.

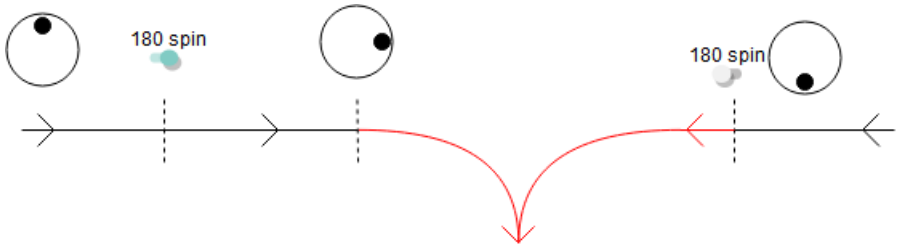
5.4 Speciální dovednosti RC modelu

Jeden z triků, které můžete vidět na obrázku 5.5, je *U-Turn*. Tento trik provede otočku o 180°. Nejprve se aktivuje přepínač a model bude čekat na udání směru. Po překročení jisté hranice na joysticku RC auto provede manévr buď doleva nebo doprava. Po provedení se přepínač vypne.



Obrázek 5.7: Diagram manévru U-Turn, červená barva značí automatizovanou část, šipka udává směr auta

Další manévr je pojmenován *180 spin*. Tento trik provede to, že auto se otočí o 180°, ale pojede opačně. To znamená že pokud je auto otočené přední částí dopředu a jede rovně, tak po vykonání pojede přední částí auta pozpátku stále rovně. Stejně jako u předchozího, směr kudy provede manévr je udán joystickem po zapnutí přepínače.



Obrázek 5.8: Diagram manévru 180 spin, červená je automatizovaná část, šipka značí směr auta

Další volby ovlivňují chování joysticku. *Lock speed* způsobí, po aktivaci přepínače, aretaci ovládání rychlosti (nahoru/dolů). Tím pádem se joystick nevrátí do své neutrální pozice po puštění. *Lock direction* aretuje udávání směru (vlevo/vpravo) stejně jako u rychlosti. Po vypnutí přepínače se joystick, pokud je puštěn, vrátí do své původní pozice.

Jelikož existující řešení mívají právě jenom tyto předdefinované chování, měla by tato páce také zahrnovat možnost, vytvořit si vlastní chování. Jelikož budeme využívat mobilního zařízení, bylo by možné vytvořit menu pro definování vlastních instrukcí, kde jedna instrukce by se měla skládat ze směru, rychlosti a délky trvání. Spouštění by probíhalo stejně jako u předdefinovaných pomocí přepínačů, kde po dokončení by se přepínač vypnul.

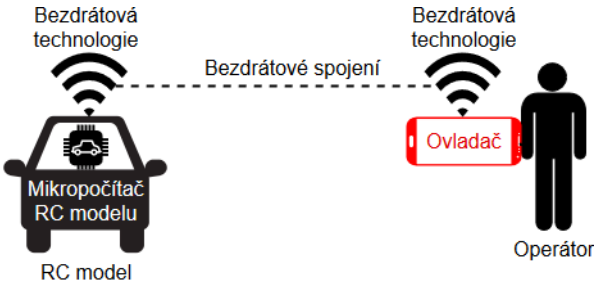
Kapitola 6

Implementace částí systému

Tato kapitola pojednává o konkrétní implementaci řešení této práce. Skládá se z kapitol seřazených dle modelu systému od ovladače operátora, přes přenosový protokol až po implementaci na mikropočítači ESP32.

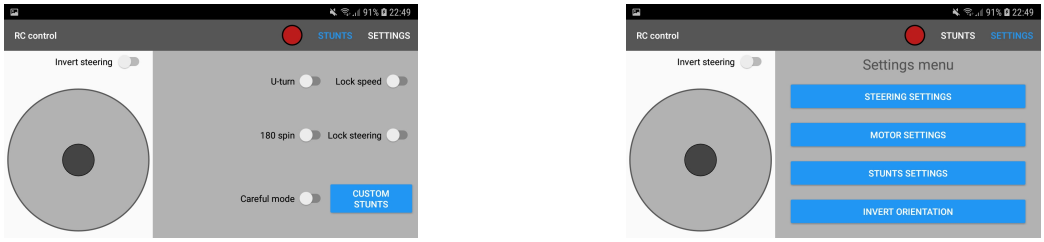
6.1 Implementace vysílače/ovladače (aplikace na Android)

Tato část se zabývá zvýrazněnou částí modelu, tj. implementací aplikace na OS Android

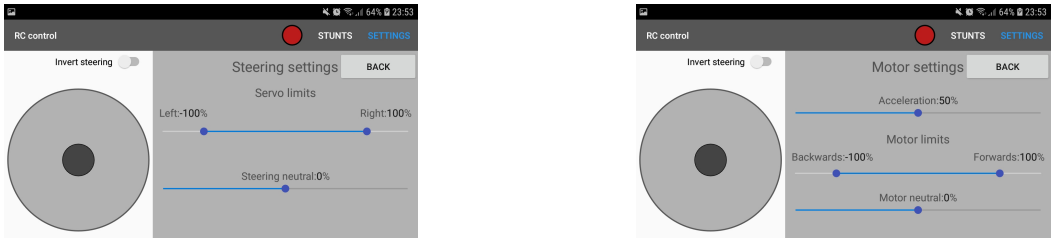


Obrázek 6.1: Model systému

Výsledná aplikace lze vidět na obrázcích 6.2 a 6.3. Rozložení se skládá ze tří částí: Horní lišta s položkami menu a kontrolkou připojení, levé okno, kde se nachází joystick, a pravé okno, kde se zobrazuje aktivní položka s menu. Po spuštění se ukáže rozložení, které lze vidět vlevo nahoře. Dále jsou zde vyobrazeny jednotlivé stavy pravého okna při procházení menu, které může být buď okno s triky, rozcestník nastavení, nastavování serva řízení nebo rychlosti. Design se lehce liší od návrhu tím, že místo jednoho menu pro nastavení serva řízení a rychlosti se vytvořilo menu, kde si operátor vybere, co chce nastavovat.



Obrázek 6.2: Aplikace v jednotlivých možných stavech (triky, nastavení menu)



Obrázek 6.3: Aplikace v jednotlivých možných stavech (nastavení serva řízení, nastavení elektromotoru)

Aplikace byla vyvíjena v jazyce Java pro OS Android pomocí vývojového prostředí Android Studio[1] používající Android SDK. Implementace je rozdělena do jednotlivých tříd, které jsou:

- MainActivity.java – zde je provedena inicializace aplikace a menu prvků, volání konstruktorů tříd, definice přechodů menu
- Joystick.java – inicializace prvků levého okna a jeho chování
- WifiStation.java – pro připojení zařízení k RC modelu a metody pro předávání informací pomocí aplikačního protokolu
- WifiThread.java – vytvoření vlákna pro běh WifiStation a předávání informací z vlákna GUI
- Special.java – definice chování přepínačů pro vykonání triku
- SettingsRangebar.java, SettingsSeekbar.java – chování posuvníků v menu nastavení
- GlobalVariables.java – definované statické proměnné pro jednoduchou modifikaci či rozšíření
- SpecialSeekbar.java – posuvník pro nastavení drsnosti povrchu
- AbstractInteractElement.java – obsahuje abstraktní třídu, ze které dědí SettingsRangebar, SettingsSeekbar a Special
- ExternalRangebarOnEditorActionListener.java – posluchač událostí pro textové položky SettingsRangebar
- CustomStunt.java – obsahuje uložené instrukce vlastního triku
- StuntEditor – metody pro editování třídy CustomStunt
- XmlSerializer – pro uložení a načtení vlastních triků pomocí XML souborů

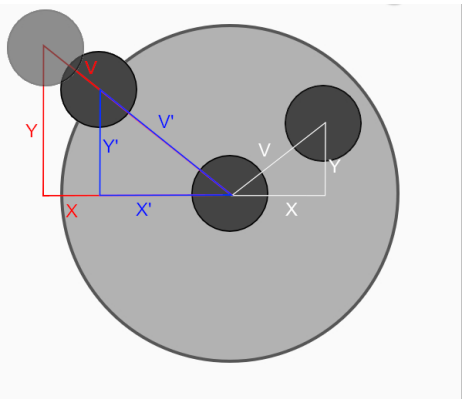
Rozvržení prvků UI je napsáno v XML souborech, kde activity_main.xml je hlavní soubor uživatelského rozhraní, který spojuje všechny soubory XML do jednoho. Kromě základních prvků uživatelského rozhraní, kterými Android SDK disponuje, bylo třeba doinstalovat RangeBar[2], který umožňuje nastavovat hodnoty z obou stran posuvníku na rozdíl od SeekBar, který umožňuje nastavovat pouze jednu hodnotu v posuvníku. Android Studio také obsahuje zvlášť soubory pro nastavení stylů (styles.xml), neměnných řetězcových literálů (strings.xml) a barev (colors.xml), které jsou používány v souborech rozložení UI.

[1] https://developer.android.com/studio
[2] https://github.com/oli107/material-range-bar

Joystick

Joystick se skládá ze 2 objektů, *ImageView* pro úchyt joysticku a *RelativeLayout* sloužící jako ohraničení úchytu. Chování těchto prvků je definováno v Joystick třídě, která je vytvořena aplikací při spuštění v metodě *OnCreate()* voláním konstruktoru, kde se definuje chování jednotlivých prvků spjatých s joystickem. Mezi tyto, kromě již zmíněného úchytu a ohraničení, patří přepínače pro invertování řízení, pro aretaci joysticku ve směru rychlosti a ve směru řízení.

Chování joysticku je simulováno pohybem úchytu, kde při doteku uvnitř ohraničení je volána metoda *onTouch*. Poté co se operátor dotkne, vyvolá se událost *ACTION_DOWN* na místo doteku se přesune úchyt. To je implementováno voláním *setTransformX* a *setTransformY* nad úchytem, kde parametry k těmto metodám se získají ze vzorce $XY = eventXY - radius$, kde XY je výsledná hodnota X,Y-ové souřadnice, $eventXY$ jsou hodnoty souřadnic získané vyvolanou událostí a $radius$ je hodnota poloměru ohraničení. Jelikož nastavujeme $transformX$,$transformY$, což jsou souřadnicové hodnoty od počátku místa vytvoření objektu, proto musíme odečíst poloměr pro zajištění, aby úchyt byl pod operátorovým palcem. Ještě před nastavením těchto získaných hodnot se provede kontrola, jestli náhodou by nebyl úchyt mimo ohraničení pomocí Pythagorovy věty $c^2 = a^2 + b^2$, kde a je souřadnice X, b je Y a c je vzdálenost od počátku. Pokud je uvnitř ohraničení, přesune se na místo dotyku. Pokud ovšem ne, uplatní se podobnost trojúhelníků, kde se vypočítá poměr ze vzorce $ratio = radius/distance$, kde $distance$ je vzdálenost mezi počátkem a místem dotyku. Následně se tímto poměrem vynásobí souřadnice, aby úchyt byl uvnitř ohraničení a provede se nastavení hodnot. Pokud operátor pohne prstem vyvolá akci *ACTION_MOVE* a celý algoritmus se opakuje. Po puštění úchytu se vyvolá událost *ACTION_UP* a za normálního chování nastaví $transformX$ a $transformY$ na 0.



Obrázek 6.4: Příklad posunů úchytu, jeden pro uvnitř ohraničení, druhý je mimo s příkladem použití podobnosti trojúhelníků.

Pro získání hodnot z joysticku se použije metod *getCurrentX* a *getCurrentY*, které vrátí uložené hodnoty souřadnic po nastavení polohy úchytu. Z věcí, které lze ještě v levém okně najít je přepínač pro obrácení hodnoty X, což se může hodit při ovládání modelu směřujícího k operátorovi, čímž nedojde k zmatení stran.

Nastavení ovladače

Pod menu záložkou *Settings* lze najít nastavení limitu serva a elektromotoru. Zde se ukáže další menu, kde operátor vybere z nabídky možností nastavení řízení nebo rychlosti. Pod záložkou s řízením se objeví dva posuvníky, jeden pro neutrální polohu, což slouží pro případ, kdyby servo na modelu bylo vychýlené a byla by potřeba ho nastavit směrem rovně (neutrální poloha), a pro nastavení limitů, což může být jak pro servo, které se umí více nebo méně vychýlit než dovoluje náprava modelu, nebo pro redukování ostrosti zatáčení.

Podobně má toto nastavení i záložka s rychlostí, kde lze nastavit neutrální polohu elektromotoru a limity maximální možné rychlosti v jistém směru. Kromě těchto je zde i posuvník pro nastavení zrychlení, kde hodnota značí, jak rychle auto bude zrychlovat.

Tyto posuvníky jsou buď z třídy *SettingsRangebar* pro složitý posuvník, nebo *SettingsSeekbar* pro jednoduchý, kde při zavolání konstruktoru v metodě *onCreate* třídy MainActivity se nadefinuje výše zmíněné chování, kde hodnoty při změně se nastaví v třídě *WifiStation* a jsou použity pro výpočty. Všechny posuvníky mají nad sebou TextBox, který ukazuje aktuálně nastavenou hodnotu. Nastavení hodnoty jde buď přes interakcí s posuvníkem, nebo přes příslušný TextBox, který po kliknutí umožňuje nastavit hodnotu přesně pomocí klávesnice, kde pro potvrzení je třeba zmáčknout *Hotovo*. Pokud operátor chce nastavit výchozí hodnotu, po kliknutí na TextBox smaže nastavené údaje a potvrdí.

Předdefinované triky

Pod nabídce záložky *Stunts*, která je po spuštění zobrazena, se nachází triky a schopnosti, které upraví chování joysticku. Mezi triky patří zmíněný U-Turn a 180 spin, mezi upravení chování joysticku patří aretace v daném směru (v ose X nebo Y). Chování triků je ovlivněno nastavením posuvníku *Terrain setup*, který je v menu nastavení.

Triky se spouštějí pomocí přepínačů, jejichž chování je definované ve třídě *Specials*. Interakce s přepínači pouze vybírá jaký trik chce operátor nechat provést. Tudíž každý Special má své vlastní ID, které se předává třídě *WifiStation* pro poslání na model. Jeden Special může být aktivní v určitý okamžik, v případě přepnutí se všechny ostatní deaktivují a nastaví se nové ID. Každý Speciál má také svou podmínku, za které se aktivuje, která závisí na poloze úchytu joysticku. Na ovladači se splnění podmínky projeví přepnutím přepínače do polohy vypnuto. Při vytváření třídy Special je předána třída *SpecialSeekbar*, která podle polohy posuvníku ovlivňuje ID triku podle vzorce: $stuntID = stuntDefaultID + seekbarValue * numberOfStunts$, kde stuntID je výsledné ID, stuntDefaultID je výchozí ID triku, seekbarValue je aktuální hodnota polohy posuvníku a numberOfStunts je celkový počet triků. Pokud je trik aktivovaný a operátor se rozhodne nastavit povrch, zruší tím vybraný trik.

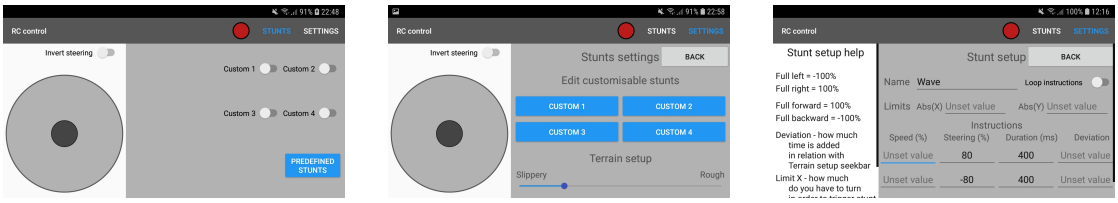
Aretace joysticku jsou definovány v třídě Joystick po zavolání jeho konstruktoru. Pokud je aktivován přepínač *Lock Speed* nebo *Lock Steering*, tak po puštění joysticku se provede kontrola, které přepínače jsou zapnuty a které ne. Pokud je přepínač v daném směru zapnut, zůstane po puštění úchytu zachována hodnota v dané ose. Pokud je přepínač deaktivován po puštění, vrátí se úchyt do nulového bodu.

Vlastní triky

Kromě předdefinovaných triků, má operátor možnost si definovat vlastní. Pro jejich spuštění je třeba přepnout pomocí tlačítka pod záložkou Stunts na *Custom stunts*. Jejich editace probíhá v nastavení triků. Zde má operátor možnost si svůj trik pojmenovat, stanovit jak

dlouho pojede model daným směrem a rychlostí, limity pro aktivaci stejně jako například u U-turn nebo 180 spin, a možnost dát trik do smyčky. Instrukce jsou v procentech pro řízení a rychlost, v milisekundách pro čas provádění.

Vlastní triky jsou nadefinované ve třídě *CustomStunt*, kde jejich instance jsou uchovávány v globální třídě. Obsahuje informace o jméně, limitech pro aktivaci, instrukcích a zda se má pouštět ve smyčce. Editace probíhá ve třídě *StuntEditor*, která obsahuje informace o aktuálně editovaném triku a políčka pro editaci. Při zahájení editace se předá trik funkci *LoadSetup*, která načte informace o triku na obrazovku. Pokud má trik více instrukcí, jsou dynamicky přidány řádky tak, aby byl poslední prázdný (konfigurace 0,0,0). Při každé změně jsou data uloženy do editovaného triku pomocí funkce *Save*. Pokud uživatel změní pohled (například pomocí tlačítka Back nebo přechodem do záložky s triky), jsou informace při změně uloženy do XML souborů, ze kterých jsou triky načítány při startu aplikace. O serializaci se stará třída *XmlSerializer*.



Obrázek 6.5: Stavy aplikace související s vlastními triky

Pohyby vlastních triků, na rozdíl od předdefinovaných, kde chování je generováno mikropočítačem, jsou tyto generovány mobilním zařízením, stejně jako kdyby byl pohyb udáván pomocí joysticku. Ovšem má to své výjimky, například se na něj nevztahují omezení řízení a rychlosti motoru nastavitelné pomocí limitů, nesimuluje zrychlení, ani zatáčení jako při ovládání joystickem a hodnoty jsou předávány v procentech místo pozice vůči počátku. Při nastavení doby vykonávání instrukce takové, že není dělitelná 20, tak je zbytková část uřezána.

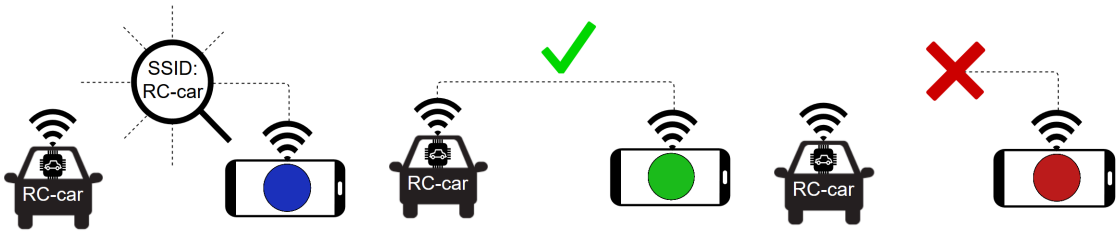
Připojení a zasílání pokynů

Aby ovladač mohl komunikovat s mikropočítačem na RC modelu, je třeba, aby se mobilní zařízení připojilo na přístupový bod vytvořený mikropočítačem. Aby se mohlo pracovat s prvky pro připojení a posílání dat přes WiFi, je třeba vytvořit nové vlákno. K tomu slouží třída *WifiThread*. Objekt této třídy je vytvořen po načtení všech prvků aplikace, kde se osvědčilo, že horní menu se načítá jako poslední, kde aplikace volá metodu *onCreateOptionsMenu*. Třída dědí z třídy *AsyncTask*, která zajišťuje jednoduché vytváření a používání vláken[3]. Z této třídy nás zajímají metody *doInBackground*, která je prováděna na jiném vlákně po zavolání metody *execute()*, a *onPostExecute*, která se vyvolá po dokončení práce v předchozí metodě.

Po vytvoření a spuštění tohoto vlákna pomocí execute() se zavolá konstruktor třídy WifiStation, ve kterém se pokusí připojit na model, kde SSID přístupového bodu mikropočítače je *RC-car*. Připojení je testováno po dobu 20 sekund v 10 ms intervalech, kde se kontroluje SSID k aktuálně připojenému přístupovému bodu. Pokud se nepodaří připojit, ukončí se práce na vlákně. Poté co se úspěšně připojí, nastaví všem interaktivním prvkům (přepínače, posuvníky), které volají WifiStation, instanci vytvořeného objektu, provede se

[3]https://developer.android.com/reference/android/os/AsyncTask

inicializace klienta a nastaví se opakovaná příprava dat a odesílání co 20 ms, což je velikost jednoho rámce PWM modulace pro servo nebo elektromotor. Následně v *onPostExecute* se nastaví vzhled kontrolky. Kontrolka může být ve třech stavech, Připojuji, Připojeno, Nepřipojeno, viz obrázek 6.6. Pokud je ve stavu Nepřipojeno, lze na kontrolku kliknout a tím proces vytvoření vlákna a připojení zopakovat.



Obrázek 6.6: Jednotlivé stavy kontrolky připojení: Připojuji, Připojeno, Nepřipojeno

Příprava dat spočívá v získání aktuálních hodnot úchytu joysticku a následné úpravě podle toho, jestli se má jednat o změnu směru či rychlosti, kde v první řadě jsou tyto hodnoty převedeny do procent. Zatočení vozidla je ovlivněno změnou v ose X, pro kterou platí:

$$Turn(x) = \begin{cases} Left\lceil leftBase^{-x} - neutral \rceil, & \text{pro } x < 0 \\ offset - neutral, & \text{pro } x = 0 \\ Right\lceil rightBase^{x} + neutral \rceil, & \text{pro } x > 0 \end{cases}$$

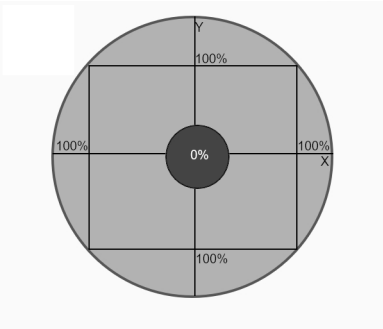
$$Left(s) = \begin{cases} neutral - offset & \text{pro } s < -neutral \\ -s + offset & \text{pro } -neutral < s < maxLeft - neutral \\ -(maxLeft - neutral) + offset & \text{pro } s > maxLeft - neutral \end{cases}$$

$$Right(t) = \begin{cases} neutral + offset & \text{pro } t < neutral \\ t + offset & \text{pro } neutral < t < maxRight + neutral \\ maxRight + neutral + offset & \text{pro } t > maxRight + neutral \end{cases}$$

kde:

- Turn(x) – funkce pro výpočet výsledné hodnoty v procentech pro poslání, kde x je poloha úchytu joysticku na osy X
- Left(s) – funkce pro výpočet levého zatočení
- Right(t) – funkce pro výpočet pravého zatočení
- leftBase, rightBase – základ pro výpočet exponenciální funkce na danou stranu ze vzorce: $base = max^{\frac{1}{joystickMax}}$
- maxLeft,maxRight – nezáporné číslo vyjadřující maximální dosažitelnou hodnotu v procentech na danou stranu (vlevo, vpravo)
- neutral – neutrální hodnota serva řízení v procentech
- offset je číslo, které posouvá hodnoty pro poslání tak, aby byly nezáporné

- joystickMax – hodnota udávající polohu na osách, která značí maximální vychýlení serva viz obrázek 6.7



Obrázek 6.7: Znázornění hodnoty joystickMax pro maxima nastavená na 100%. Díky tomuto pomyslnému čtverci lze zatáčet a mít rychlost na 100%.

Zrychlení vozidla probíhá ve 2 krocích. Nejdříve je převedena poloha úchytu Y osy na procento, kde platí:

$$Percent(y) = \begin{cases} -\frac{y * maxForward}{joystickMax} & \text{pro } y < 0 \\ offset + neutral & \text{pro } y = 0 \\ \frac{y * maxBackwards}{joystickMax} & \text{pro } y > 0 \end{cases}$$

kde:

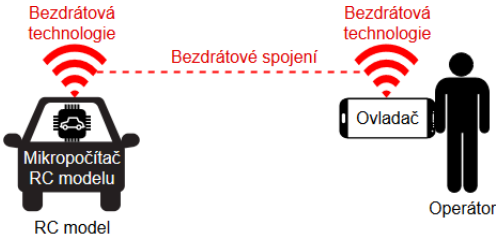
- Percent(y) – funkce pro výpočet mezivýsledku
- maxForward, maxBackwards – nezáporné číslo vyjadřující maximální dosažitelnou hodnotu na danou stranu (dopředu, dozadu)
- neutral – neutrální hodnota serva motoru v procentech

Následně pokud je prováděno zrychlení vpřed, vypočte se aktuální rychlost ze vzorce $accelerate = accelerate + \frac{100}{accelerateRatio}$, kde $accelerateRatio$ je hodnota v procentech, která určuje míru zrychlení, která se přičítá dokud $accelerate$ není větší než výsledek Percent(y). Pro zpomalení je použit podobný vzorec $accelerate = accelerate - \frac{100}{accelerateRatio}$, ale míra se odečítá dokud není menší než výsledek Percent(y). Poté je proveden posun o offset a neutrál vyjádřené vzorcem $finalSpeed = accelerate + offset + neutral$. Offset se vypočítá ze vzorce $offset = 200 - Min(neutralSpeed, neutralSteering)$ kde neutralSpeed je neutrál pro rychlost a neutralSteering je neutrál pro řízení a konstanta 200 je výchozí offset zvolen tak, aby posílané hodnoty nebyly záporné.

Po získání procentuálních hodnot rychlosti a řízení jsou tyto hodnoty zakódovány pro poslání spolu s offsetem, aktivním trikem a sériovým číslem do protokolu a odeslány. Pokud selže spojení mezi ovladačem a modelem, je tento opakovaný proces zrušen a kontrolka nastavena na nepřipojeno.

6.2 Protokol pro komunikaci

Tato část se zabývá zvýrazněnou částí modelu 6.8, tj. protokolem pro komunikaci mezi mikropočítačem a mobilním zařízením



Obrázek 6.8: Model systému

Protokol se skládá z pěti hodnot: offset, rychlost a zatočení v procentech, číslo triku a sériové číslo. Aby paket zabral co nejmenší místo, tak offset, rychlost a zatočení mají každý vyhrazený prostor velikosti datového typu short (16 bitů), id triku a sériové číslo mají velikost datového typu char (8 bitů).

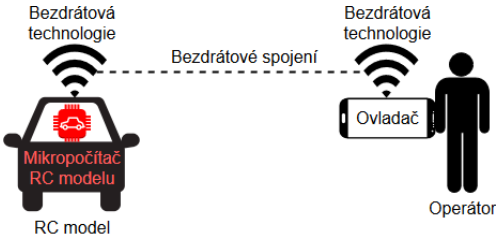
offsetLo	offsetHi	turnLo	turnHi	speedLo	speedHi	stuntID	serial

Tabulka 6.1: Obsah paketu od nejnižšího indexu po nejvyšší při odchodu z ovladače. Jedna buňka je naplněna číselnou hodnotou o velikosti char (8 bitů)

Pro docílení nejvyšší rychlosti kódování a dekódování, jsou hodnoty o velikosti short rozděleny na části high, kde jsou vyšší bity, a low, kde jsou nižší, o velikosti char pomocí bitových posuvů. Sériové číslo se inkrementuje pokaždé, co se změní trik, kde pokud dosáhne maximální hodnoty (tj. 255), tak se sériové číslo resetuje a pokračuje se od nuly.

6.3 Implementace na mikropočítači ESP32

Tato část se zabývá implementací mikropočítače ESP32 na RC modelu.



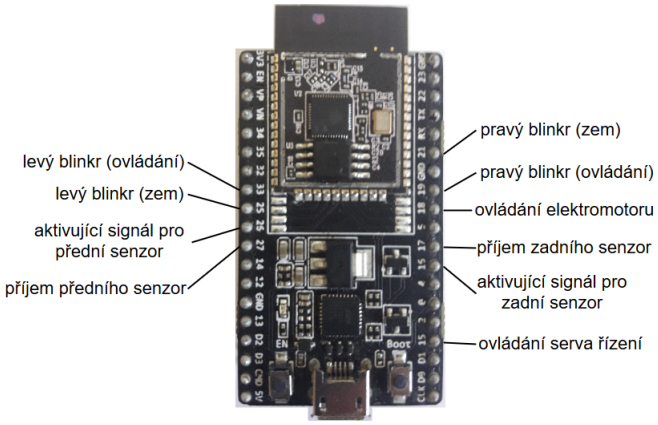
Obrázek 6.9: Model systému

Mikropočítač je používán pro příjem a zpracování dat z paketu a následnou interpretaci chování podle přijatých hodnot. Jelikož ESP32 má dvě jádra, jsou jednotlivé akce rozděleny do úkolů (Tasks), které jsou navzájem synchronizovány pomocí semaforů. Mezi tyto úkoly patří: příjem a dekódování paketu, nastavení aktuální střídy PWM pro ovládání serv a získávání dat ze senzorů.

Program pro ESP se skládá ze šesti zdrojových souborů, mezi které patří:

- esprc_main.c – hlavní smyčky jednotlivých úkolů a jejich inicializace
- servo.c – funkce pracující s periferií MCPWM pro nastavení chování serv
- wifi.c – funkce pro inicializaci přístupového bodu, serveru a následný příjem dat z ovladače
- sensor.c – funkce pro získávání dat ze senzorů
- blinker.c – funkce pro blikání ledek

Pro komunikaci s externími prvky, jako jsou senzory nebo serva, používá GPIO piny, kde jejich rozvržení je:



Obrázek 6.10: Popis vstupů a výstupů pinů ESP32

Inicializace

V prvé řadě inicializace se nastaví chování MCPWM, kde tato periferie je přímo určena pro práci s různými servy a elektromotory. Její nastavení je první, protože když nastane na jádru fatální chyba, například z nedostatku napětí, model přestane zrychlovat kvůli nastavení hodnot do neutrálu. Jsou zde nastaveny dva výstupy, jeden pro servo řízení a druhý pro elektromotor, kde PWM pro oba výstupy má opakovací dobu 20 ms (50 Hz) a střída je nastavena na velikost 1,5 ms v log. 1, jinak řečeno je v neutrálu.

Po nastavení MCPWM se nastaví WiFi přístupový bod. Je zde nastavena obsluha, která kontroluje zda se nějaké zařízení připojilo nebo odpojilo. Dále jsou nakonfigurovány údaje přístupového bodu, jako je SSID, heslo, maximální počet připojitelných zařízení a zabezpečení. Po zapsání a povolení přístupového bodu proces přejde do spánku a čeká, až ho obsluha probudí po připojení zařízení. Jakmile je probuzen, pokračuje se ve vykonávání kódu.

Nakonec se vytvoří semafory pro synchronizaci úloh procesoru. Jsou zde dva pro hlavní úlohy - příjem pokynů a pohybu, dva pro povolení měření senzoru a jeden pro přístup ke sdílené proměnné. Jelikož paměť ESP je symetrická, není třeba vytvářet speciálně sdílené proměnné pro mezi-úlohovou komunikaci. Zbývá vytvořit zmiňované úlohy, mezi které patří:

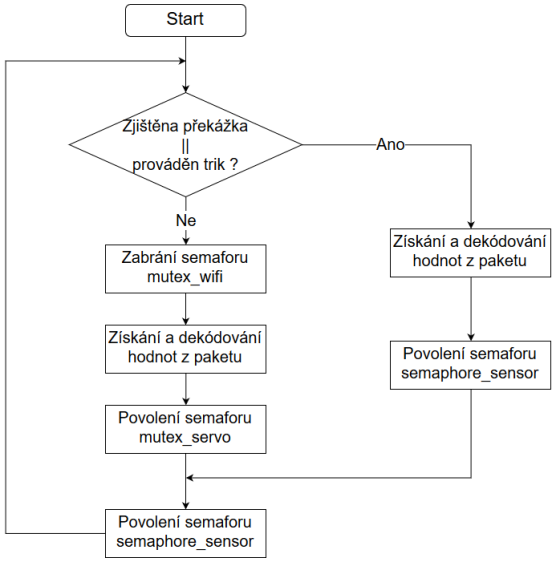
úloha pro nastavování střídy PWM (*servo_main*), úloha pro příjem a dekódování zpráv z paketů (*wifi_main*) a úloha pro zjišťování vzdálenosti na senzorech (*sensor_main*).

Příjem pokynů

Příjem pokynů probíhá v úloze *wifi_main*. Zde je volána funkce *reciveve_instructions*, díky které se získávají nové instrukce. Je použit blokující socket, kde při vypršení časového limitu se instrukce nastaví do neutrální polohy. Zápis instrukcí z paketu se provádí do struktury *TInstructions*, která obsahuje hodnotu řízení v procentech (-100% plné zatočení doleva, 100% plně doprava), udávanou rychlost v procentech (-100% couvání, 100% jízda vpřed), trik pro provedení a sériové číslo triku.

Před zápisem se dekóduje offset, který se získá pomocí vzorce $offset = packet[0] + packet[1] << 8$, kde indexace paketu je odvozena z tabulky 6.1 a posuv je o 8, jelikož velikost datového typu char je 8 bitů. Podobně se dekóduje procento zatočení a rychlost: $percent = packet[x] + packet[x+1] << 8 - offset$, kde x nabývá hodnoty pro zatočení dvě a pro rychlost čtyři. Nakonec jsou přiřazeny ID triku a sériové číslo z packet[6] a packet[7].

Před každým cyklem je semafor *mutex_wifi*, který určuje, jestli už hodnota byla čtena nebo ne, povolována úlohou *servo_main*. Po naplnění struktury je povolen semafor *mutex_servo*, který naznačuje, že je možné vzít hodnotu ze struktury a následně je povolen *semaphore_sensor* pro běh úlohy získání hodnoty ze senzoru. Pokud je vykonáván trik nebo model narazí na překážku, úloha začne vykonávat separátní smyčku, kde aktualizuje vstupy z ovladače, nečeká na *mutex_wifi* a povolí *semaphore_sensor* tak, aby nahradil zaneprázdněnou úlohu *servo_main*.



Obrázek 6.11: Vývojový diagram pro úlohu *wifi_main*.

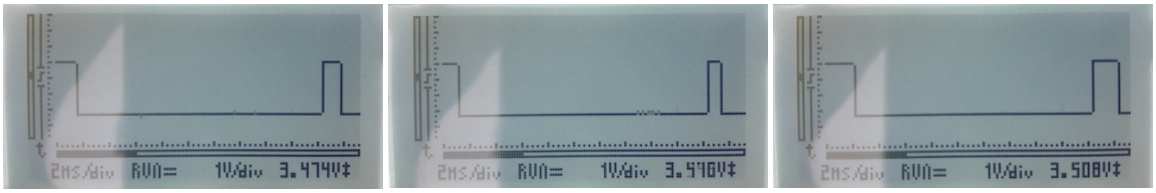
Pohyb

Instrukce získané z paketu a zapsané do struktury TInstructions si následně bere úloha *servo_main*, která má na starost modifikaci PWM pro pohyb modelu. Ještě před vstupem do smyčky je třeba naučit elektrický regulátor, kde je předek vozidla, resp. jaká střída zna-

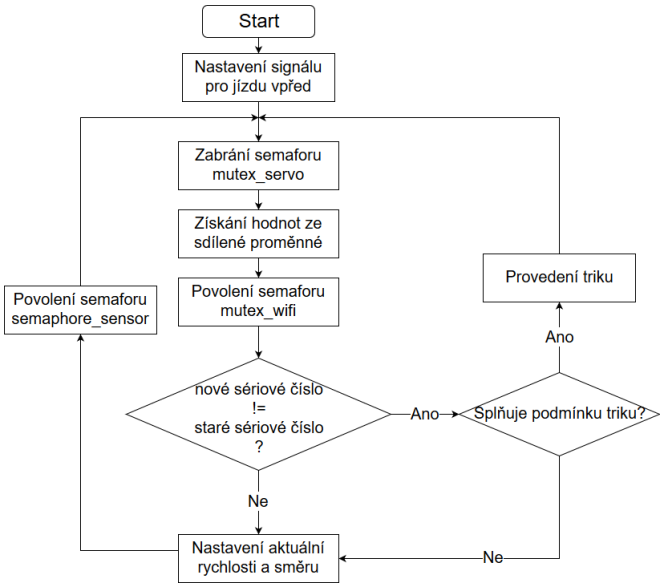
mená pohyb vpřed. Proto je na 1 sekundu nastaven směr dopředu a poté zpět do neutrálu. Následně úloha vstoupí do smyčky.

Na začátku každého cyklu se pokusí úloha zabrat semafor *mutex_servo*, který značí naplnění struktury s instrukcemi. Pokud se podaří zabrat semafor,zapíše si do lokálně vytvořené struktury informace ze sdílené a povolí semafor *mutex_wifi*.

Pro změnu PWM jsou volány funkce *set_steering* nebo *set_speed*, která volá knihovní funkci *mcpwm_set_duty_in_us*, která modifikuje střídu na výstup serva řízení nebo elektromotoru. Jelikož tato funkce bere velikost střídy v mikrosekundách, je třeba převést procenta na tuto hodnotu. Proto byla vytvořena funkce *percent_to_duty*, která proměnnou přepočítá pomocí vzorce: $duty = \frac{500 * percent}{100} + 1500$, kde duty je výsledná střída v mikrosekundách, 1500 je hodnota pro výchozí neutrální pozici a konstanty 500 a 100 slouží pro přepočet procent na mikrosekundy.



Obrázek 6.12: Střídy pro stavy: neutrálu (1,5 ms), plně doleva (1 ms), plně doprava (2 ms)



Obrázek 6.13: Vývojový diagram pro úlohu *servo_main*.

Triky

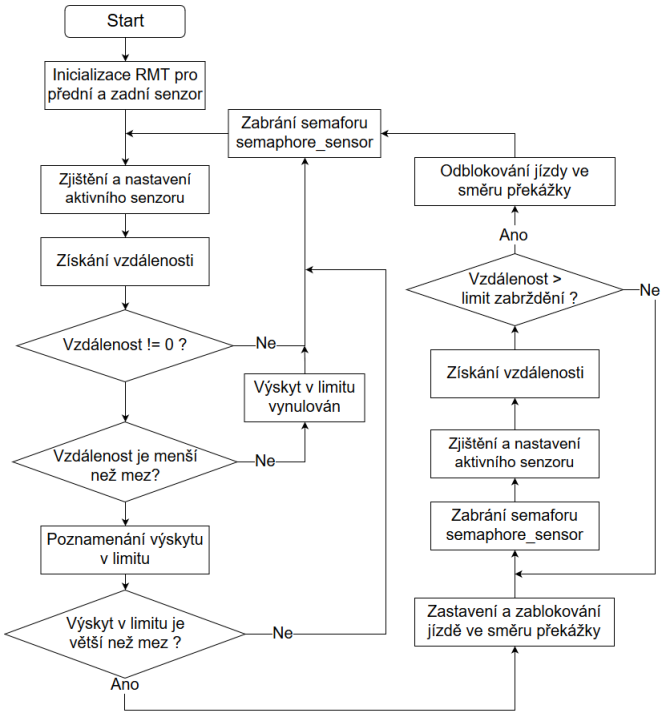
Úloha *sensor-main* si pamatuje poslední sériové číslo, které uchovává v proměnné *previous_serial*. Pokud příjme paket s jiným číslem než uloženým, zavolá funkci *perform_stunt*, která čeká na splnění podmínky pro daný trik, například na určité procento zatočení, což platí jak pro 180 spin, tak pro U-turn. Jakmile operátor dostatečně zatočí, provede se trik v daném směru.

Například pokud se aktivuje U-turn, operátor stále může jet rovně a mírně zatáčet. Jakmile ale procento zatočení překročí 40%, třeba doleva, tak auto provede levou otočku. Po vykonání triku se přepíše předchozí sériové číslo a model lze řídit normálně. Pokud je trik zrušen, tak se vykoná tzv. nultý trik, který přepíše předchozí sériové číslo. Podobně se toto děje pokud je trik změněn, nejdříve se vykoná nultý trik a následně lze provést zvolený.

Pro 180 spin je speciální, že funguje jak při jízdě vpřed, tak i vzad. Při provedení otočky ze zpátečky není problém hned přejít do jízdy vpřed, jelikož regulátor otáček toto povoluje. Při otočce opačné vzniká problém, protože normální chování regulátoru je začít brzdit. Toto lze obejít při zaslání posloupnosti pokynů zpátečka-neutrál-zpátečka v 60 ms intervalech mezi nimi, což donutí regulátor přejít okamžitě z jízdy vpřed do zpátečky.

Senzory

Pro automatické detekování a zabrzdění před překážkou jsou použity ultrazvukové senzory HC-SR04, které jsou umístěné na přední a zadní části modelu. Obsluha těchto senzorů je v úloze *sensor_main*. Pro zasílání využívá "Remote Control"modulu, díky kterému lze zasílat a přijímat různé typy signálů. Před hlavní smyčkou se inicializují dva přijímače a dva vysílače.



Obrázek 6.14: Vývojový diagram pro úlohu *sensor_main*.

Následně je přechod do hlavní smyčky. Zde se získává vzdálenost z aktivního senzoru, který se určí podle směru jízdy. Pokud se model pohybuje dopředu, aktivní senzor je přední, pokud jede dozadu, aktivní senzor je zadní. Vzdálenost senzoru od objektu je získávána z funkce *get_distance*. Ta zašle signál na aktivní senzor, který drží log. 1 10 us a log. 0 dalších 10 us. Následně je z ringbuffer paměti získána struktura *rmt_item32_t*, ze které můžeme získat dobu mezi posláním a přijetím signálu zpět. Poté je proveden výpočet vzdálenosti ze vzorce $distance = (time * speedOfSound)/2$, kde time je čas mezi posláním a přijetím,

speedOfSound je rychlost zvuku, kde bereme přibližnou hodnotu 340 ms^{-1} a konstanta 2 je kvůli tomu, že zvuk před přijetím vykoná dvojnásobnou cestu (k objektu a zpět). Aby se odlišilo mezi senzory, je výstup zadního senzoru vynásoben -1.

Protože HC-SR04 není přesný senzor, rozhodl jsem se počítat výskyty určitého prahu vzdálenosti, kde je jako výchozí hodnota nastavena 60 cm. Pokud jsou získané hodnoty nižší nebo rovny než tento práh, poznačíme si výskyt. Pokud těchto výskytů nastane 4 po sobě, mikropočítač prohlásí, že narazil na překážku a nastaví do proměnné *return_sensor* 1 nebo 2 podle toho, jestli je objekt před nebo za modelem. Při další iteraci úlohy pohybu je tato proměnná zaregistrována, model nouzově zastaví a zakáže operátorovi řídit. Následně se nastaví do téže proměnné příznak, že model byl zastaven v daném směru, a zablokuje možnost jet ve směru překážky. Poté úloha senzoru kontroluje, jestli model odjel dostatečnou vzdálenost od překážky. Pokud ano povolí jízdu v blokovaném směru.

Protože se stávalo, že tato úloha vyhladověla ostatní úlohy a nedovolila ovládat model, na konci cyklu se zabírá semafor, který povoluje jak wifi, tak servo úloha. Limit povolení tohoto semaforu je nastaven na dvě, tudíž mezi přijetím nového pokynu úlohou wifi, může dvakrát získat vzdálenost.

6.4 Testování

Při testování tohoto systému pro komerční použití je třeba postupovat následovně. U hardwarových částí se ověřuje správné zapojení jednotlivých komponent modelu mezi sebou a také dostačující proud a vstupní napětí pro napájení součástek, kde při nízkém napětí nám komponenty nebudou fungovat a při dodání moc vysokého zase může nenávratně součástku poškodit. Jelikož v tomto systému napájíme jak servo a elektromotor, tak mikropočítač a senzory z jedné baterie, je třeba zajistit vhodně stabilizované napětí pro nízkonapěťové součástky, aby jsme se vyvarovali zbytečnému poškození a případné potřeby tyto součástky nahradit novými.

U softwarové části mikropočítače je třeba testovat správnou modifikaci výstupní modulace, aby servo řízení se při změně nechvělo, správný příjem a dekódování paketu a funkční synchronizace mezi úlohami na procesoru. U aplikační části je třeba testovat rozložení prvků jako jsou nápisy, posuvníky, tlačítka apod., aby bylo možné nasadit aplikaci na zařízení z různými typy rozlišení displayů. Dále je třeba otestovat jejich funkcionalitu, tzn. že posuvníky správně ovlivňují výstupy, při interakci s prvky se dostanu k hodnotám, které očekávám, úspěšně se spojím s mikropočítačem a umím správně zakódovat a poslat paket. Dále je vhodné otestovat kompatibilitu na různých typech operačních systémů, kde je snaha zaměřit se na nejrozšířenější systémy v moment vývoje.

Po otestování výše zmíněného je třeba ověřit správnost chování předdefinovaných vlastností za různých podmínek a validní výstupy ze senzorů, aby model uměl včas zareagovat.

Testování zapojení

Jelikož při vyvíjení tohoto systému byl zakoupen model hotový, bylo třeba ověřit vstupní napětí pro senzory a mikropočítač, následně výstupy modulace PWM generované mikropočítačem. Kontrola napětí byla provedena pomocí multimetru, kde bylo testováno správný výstup ze stabilizátoru a později, při dokoupení nového regulátoru, také kontrola napětí dodávaná na senzory, kde bylo třeba dokoupit rezistory pro snížení napětí ze 6 V na alespoň 5,5 V.

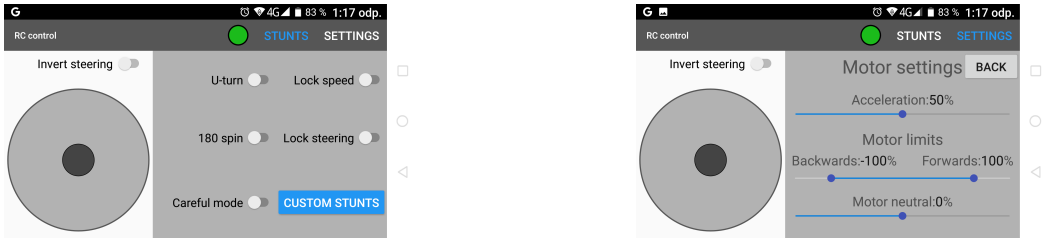
Při testování vstupů serva a elektromotoru byl využíván osciloskop, k porovnání PWM modulace z původního přijímače vůči generované mikropočítačem, kde proběhla následné úprava frekvence, aby servo reagovalo vhodným způsobem.

Testování aplikace

U rozložení prvků aplikace se dbalo zejména na čitelnost nadpisů, správné reakce při stisku tlačítek a přepínačů. Z funkčních záležitostí se ověřovalo úspěšně připojení přes WiFi pokud: je vypnuta WiFi, je připojena na jiný přístupový bod, při vymazání zapamatovaného přístupového bodu z paměti. Dále správné reakce přepínačů a posuvníků v závislosti s joystickem a testování limitních nastavení.

Rozložení prvků bylo testováno jak za pomocí zobrazení náhledu v Android studiu, tak na skutečných zařízeních, kde se také ověřovala zároveň funkcionalita. Kromě referenčního zařízení Samsung Galaxy J7 (2016), na kterém probíhal vývoj, tak byly pro testování použity: Xiaomi Redmi 7 (Android 9.0.4), Doogee S60 (Android 7.0), One Plus 6 (Android 9.0.5), Huawei P20 (Android 9.0.0) a Huawei LYO-L21 (Android 5.1). Byly vybrány mobily jak z různými systémy, tak odlišným rozlišením obrazovky.

Obrázek 6.15: Test rozložení na Xiaomi Redmi 7

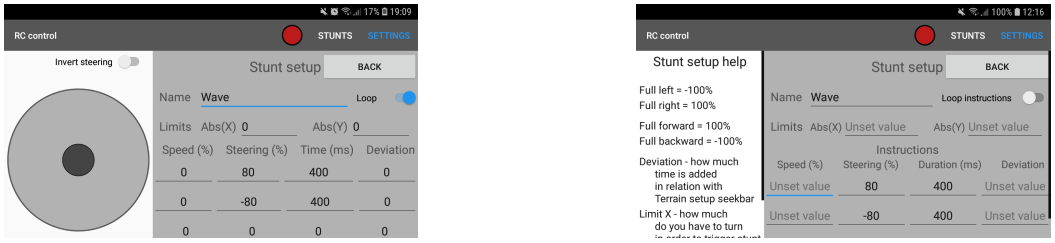


Obrázek 6.16: Test rozložení na Doogee S60 s maximální velikostí písma

Zde bylo objeveno a následně opraveno překrývání nadpisů posuvníků při nastavení písma na "velmi velké"a přetečení přepínačů za obrazovku. Dále bylo opraveno selhávání při pokusu připojit se na model s vypnutou WiFi nebo se zapnutými mobilními daty, díky nimž aplikace nedetekovala přístupový bod modelu. Dále bylo zjištěno, že na verzích OS Android 9.0.0 a vyšší jsou striktnější zabezpečení než na verzích nižších. To se projevuje nemožnosti získat SSID přístupového bodu, kvůli hrozby sledování uživatelů. Toto je vyřešeno výzvou pro povolení (a případné zapnutí) zjišťování Polohy a díky tomu lze přistoupit k hodnotě SSID. Na OS Android 9.0.5 je problém s automatickým nastavením přístupového bodu, tím pádem po selhání připojení se ukáže uživateli postup pro řešení této chyby. Pro ověření opravy chyb proběhl test dvakrát na stejných zařízeních před a po opravě.

Do testování aplikace spadají i vlastní triky jelikož jsou posílány po jednotlivých instrukcích z mobilního zařízení. Zde bylo zejména testováno intuitivnost nastavení. Testování proběhlo na lidech, kteří ovládají anglický jazyk, kde toto kritérium je zejména kvůli

anglických pojmenováních v aplikaci. Při testování se ukázalo, že pokud uživatel si neprošel všechny položky v nastavení, nevěděl jaké hodnoty má zadat do instrukcí. Taktéž některé názvy nebyly plně vysvětlující. Výsledkem bylo přidání vysvětlivek na místo joysticku.



Obrázek 6.17: Nastavení vlastních triků před a po úpravách

Testování chování modelu

Při verifikaci předdefinovaných triků bylo zaměření na přesnost provedeného manévru. Důvod tohoto testování je, že na různých površích se model bude chovat jinak. Proto byly zvoleny 2 testovací povrchy: hladký reprezentovaný linoleem a hrubý reprezentován asfaltem. Bylo sledováno podobnost výsledného pohybu s navrženým. Výsledkem bylo že na asfaltu model nedotáčel triky a tudíž se neshodoval s návrhem. Proto byl zaveden posuvník pro změnu kluzkosti, který upravuje časování instrukcí na mikropočítači.

Při testování senzorů se dbalo na včasnou reakci a následné zastavení před překážkou. Důvodem je fakt, že HC-SR04 je levný a nekvalitní, tudíž je třeba získat více hodnot než lze usoudit jistou vzdálenost od překážky. Testování probíhalo způsobem "crashtestů", kde model byl nasměrován na polystyrenový kvádr a pokusil se do něj vrazit. Výsledkem bylo, že při vyšších otáčkách senzor začínal zaznamenávat nesprávnou vzdálenost, z důvodu snížení jeho efektivní vzdálenosti. Výsledkem bylo vytvoření bezpečného módu, který lze aktivovat z ovladače, jehož zapnutí zpomalí model tak, že rozdíly od neutrálu jsou $\pm 70\ ms$. Po přidání izolace mezi kostru modelu a senzor v podobě molitanu, lze model zrychlit až na 50% (1750 ms) maximálního výkonu elektromotoru bez zaznamenání chybné vzdálenosti mezi překážkou. Ovšem doba, než můžeme bezpečně prohlásit, že jsme narazili na překážku je moc velká pro zastavení ve vysoké rychlosti. Tudíž byl zachován bezpečný mód.

Kapitola 7

Závěr

Cílem této práce bylo automatizovat pohyb RC modelu předdefinovanými pohyby serva řízení a elektromotoru pomocí mikropočítače ovládaného mobilním zařízením. Tento cíl byl splněn.

Poznatky z prvního bodu zadání, který se týká časování signálů, jsou uvedeny v druhé kapitole. Body dva a tři, které se týkají návrhu vylepšení a technického vybavení jsou obsaženy v kapitole pět. Čtvrtý bod, který má popisovat použitý algoritmus, je rozebírán v šesté kapitole.

Výsledkem je model ovládaný mobilním zařízením přes vlastní aplikaci a naprogramování mikropočítače ESP32 pro pohyb, příjem instrukcí a provádění triků, kde mikropočítač a mobilní zařízení komunikují pomocí technologie WiFi. Model má tři předdefinovaná chování, zpětnou otočku (U-turn), otočení zepředu dozadu (180 spin) a bezpečný mód, kde pomocí senzorů dokáže předejít čelnímu a zadnímu nárazu. Dále operátorovi je dovoleno pomocí editoru nadefinovat vlastní chování modelu, která jsou ukládána na mobilní zařízení. Ovládání je provedeno pomocí joysticku, kde limity serv nebo elektromotoru a neutrální polohy jsou nastavitelné.

Při tvorbě této práce jsem si vyzkoušel programování na mikropočítači ESP32 a dále jsem se naučil tvorbu aplikací na OS Android pomocí jazyka Java.

V této práci by se dalo pokračovat implementováním tohoto systému na jiný model, například letadla nebo lodi. Také by bylo zajímavé, ukládat vlastní triky přímo na mikropočítači v paměti flash, aby po opětovném zapnutí je nebylo třeba znovu definovat z mobilního zařízení.

Literatura

[1] Whitepaper Blu-ray Disc Format. Blu-ray Disc Association, CA: Los Angeles, 2005, [Online; navštíveno 7.2.2019].
URL http://www.blu-raydisc.com/Assets/Downloadablefile/2b_bdrom_audiovisualapplication_0305-12955-15269.pdf

[2] Pulse-code modulation. In: Wikipedia: the free encyclopedia. Wikimedia foundation, CA: San Francisco, 2019, [Online; navštíveno 7.2.2019].
URL https://en.wikipedia.org/wiki/Pulse-code_modulation

[3] Raspbery Pi GPIO. Raspberry Pi Fondation, Cambridge, 2018, [Online; navštíveno 6.2.2019].
URL https://www.raspberrypi.org/documentation/usage/gpio/

[4] Introduction to the Arduino Board. Arduino LLC, Monza 2019, [Online; navštíveno 6.2.2019].
URL https://www.arduino.cc/en/reference/board

[5] ESP32 Technical Reference Manual. Espressif Systems, Šhanghaj 2018, [Online; navštíveno 22.1.2019].
URL https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[6] Alepis, E.; Sakelliou, A.: Augmented car: A low-cost augmented reality RC car using the capabilities of a smartphone. In *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*, July 2016, s. 1–7, doi:10.1109/IISA.2016.7785381.

[7] Atzori, L.; Iera, A.; Morabito, G.: The Internet of Things. *Computer Networks*, ročník 54, č. 15, 2010: s. 2787–2805, ISSN 13891286, doi:10.1016/j.comnet.2010.05.010.
URL https://linkinghub.elsevier.com/retrieve/pii/S1389128610001568

[8] Authors, T. W.: *Wireless Networking in the Developing World, Third Edition.* Lulu.com, Copenhagen 2013, ISBN 978-1-4840-3935-9.
URL http://wndw.net/pdf/wndw3-en/wndw3-ebook.pdf

[9] Casner, D.: Guidelines for writing code for the ESP8266. danielcasner.org, Brea 2015, [Online; navštíveno 22.1.2019].
URL http://www.danielcasner.org/guidelines-for-writing-code-for-the-esp8266/

[10] Fisher, C.: Autonomous Control of RC Car Using Arduino. Instructables, CA: San Francisco 2010, [Online; navštíveno 23.3.2019].
URL https://www.instructables.com/id/Autonomous-Control-of-RC-Car-Using-Arduino/

[11] Fuchs, O.: *Internet of Things s podporou Bluetooth a CoAP*. bakalářská práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno 2016, vedoucí práce Musil Petr.

[12] George, D. P.; Sokolovsky, P.: Getting started with MicroPython on the ESP8266. MicroPython.org, Leeds 2019, [Online; navštíveno 22.1.2019].
URL https://docs.micropython.org/en/latest/esp8266/tutorial/intro.html

[13] Halfacree, G.: *The official Raspberry Pi Beginer's Guide How to use your new computer*. Raspberry Pi Trading Ltd., Cambridge 2018, ISBN 9781912047680.

[14] Jain, N.: PAM & Ethernet: A perfect match. EDN network, TX: San Antonio, 2016, [Online; navštíveno 9.2.2019].
URL https://www.edn.com/electronics-blogs/absolute-eda/4441982/PAM---Ethernet--A-perfect-match

[15] Ježková, K.: Seznamte se s Arduino vývojovým prostředím. Arduino.cz, Praha 2014, [Online; navštíveno 21.1.2019].
URL https://arduino.cz/seznamte-s-arduino-vyvojovym-prostredim/

[16] Jing, Y.; Zhang, L.; Arce, I.; aj.: AndroRC: An Android remote control car unit for search missions. In *IEEE Long Island Systems, Applications and Technology (LISAT) Conference 2014*, May 2014, s. 1–5, doi:10.1109/LISAT.2014.6845227.

[17] lannizzotto, G.; Costanzo, C.; Lanzafame, P.; aj.: A vision-based user interface for real-time controlling toy cars. In *2005 IEEE Conference on Emerging Technologies and Factory Automation*, ročník 1, Sept 2005, ISSN 1946-0740, s. 8 pp.–1016, doi:10.1109/ETFA.2005.1612634.

[18] Leechor, P.; Pornpanomchai, C.; Sukklay, P.: Operation of a radio-controlled car by voice commands. In *2010 2nd International Conference on Mechanical and Electronics Engineering*, ročník 1, Aug 2010, s. V1–14–V1–17, doi:10.1109/ICMEE.2010.5558606.

[19] Macháček, Z.; Nevřiva, P.: *Modulované signály*. Vysoká škola báňská - Technická univerzita, Ostrava 2012, ISBN 978-80-248-2600-4.

[20] Matoušek, P.: *Síťové aplikace a jejich architektura*. Vutium, Brno 2014, ISBN 978-80-214-3766-1.

[21] Mertlík, T.: *Popis technologie NFC a jejího zabezpečení*. diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, Brno 2013. 116 s, vedoucí práce Ing. Martin Rosenberg.

[22] Nasir, S. Z.: Introduction to ATmega328. The Engineering Project, Scottsdale 2017, [Online; navštíveno 21.1.2019].
URL https://www.theengineeringprojects.com/2017/08/introduction-to-atmega328.html

[23] Pilch, T.: *Možnosti použití Raspberry Pi pro domácí automatizaci*. bakalářská práce, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Brno 2014. 45 s, vedoucí práce Ing. Ondřej Andrž, Ph.D.

[24] Rother, P.: The million dollar question - PCM or PPM? Possibilities, performance? aerodesign.de, Zorneding 2001, [Online; navštíveno 18.1.2019].
URL http://www.aerodesign.de/peter/2000/PCM/PCM_PPM_eng.html#Anker144123

[25] Rother, P.: PCM Advantages and Disadvantages. aerodesign.de, Zorneding 2001, [Online; navštíveno 19.1.2019].
URL http://www.aerodesign.de/peter/2000/PCM/pcm-vor-nachteile_eng.html

[26] Rother, P.: PPM Advantages and Disadvantages. aerodesign.de, Zorneding 2001, [Online; navštíveno 18.1.2019].
URL http://www.aerodesign.de/peter/2000/PCM/ppm-vor-nachteile_eng.html

[27] Santos, S.: ESP32 vs ESP8266 – Pros and Cons. Maker Advisor, Woodinville 2019, [Online; navštíveno 22.1.2019].
URL https://makeradvisor.com/esp32-vs-esp8266/

[28] Stirparo, P.; Loeschner, J.: Secure Bluetooth for Trusted m-Commerce. *Int'l J. of Communications, Network and System Sciences*, ročník 6, 01 2013: str. 277, doi:10.4236/ijcns.2013.66030.

[29] Trojánek, Z.; Birkus, G.; Hlavinka, M.; aj.: Základní pojmy: PPM, CPPM, PWM, PCM a S.BUS. OpenTx, 9 České Budějovice 2018, [Online; navštíveno 18.1.2019].
URL https://www.opentx.cz/index.php/Z%C3%A1kladn%C3%AD_pojmy:_PPM,_CPPM,_PWM,_PCM_a_S.BUS

[30] Trojánek, Z.; Birkus, G.; Hlavinka, M.; aj.: Základní pojmy: Pásmo vysílače. OpenTx, 9 České Budějovice 2018, [Online; navštíveno 18.1.2019].
URL https://www.opentx.cz/index.php/Z%C3%A1kladn%C3%AD_pojmy:_P%C3%A1smo_vys%C3%ADla%C4%8De

[31] Tsevelnyam, A.: *Lokalizace pomocí BLE rámců*. diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, Brno 2017. 57 s, vedoucí práce Ing. Petr Sysel Ph.D.

[32] Voda, Z.; tým HW Kitchen: *Průvodce světem Arduina*. Nakladatelství Marnit Stříž, Bučovice 2018, ISBN 978-80-87106-90-7.

[33] Vunderl, B.; Žagar, M.; Basch, D.: Remote control of model vehicles using Android mobile devices. In *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2013, s. 901–906.

[34] Wang, Z.: Self Driving RC Car. zhengludwig.wordpress.com, MA: Massachusetts 2010, [Online; navštíveno 23.4.2019].
URL https://zhengludwig.wordpress.com/projects/self-driving-rc-car/

[35] Šiška, M.: *Impulzové modulace*. diplomová práce, Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Brno 2013, vedoucí diplomové práce Ing. Radim Číž, Ph.D.

Příloha A

Výpis hardwarových komponent a návod na zapojení

Komponenty

Mezi použité hardwarové součástky patří: vývojová deska ESP32-DvekitC s mikropočítačem ESP32-WROOM-32D.

Obrázek A.1: ESP32-DevKitC

Stabilizátor napětí AMS1117-3,3V

Obrázek A.2: AMS1117

Dupont propojovací vodiče vidlice-zásuvka a zásuvka-zásuvka

Obrázek A.3: Dunont propojovací vodiče

Ultrazvukový senzor HC-SR04 x2

Obrázek A.4: HC-SR04[1]

[1]Převzato z https://www.gme.cz/data/product/480_480/pctdetail.772-144.1.jpg?ts=1503664334

RC model LRP S10 Blast TC 2

Obrázek A.5: RC model[2]

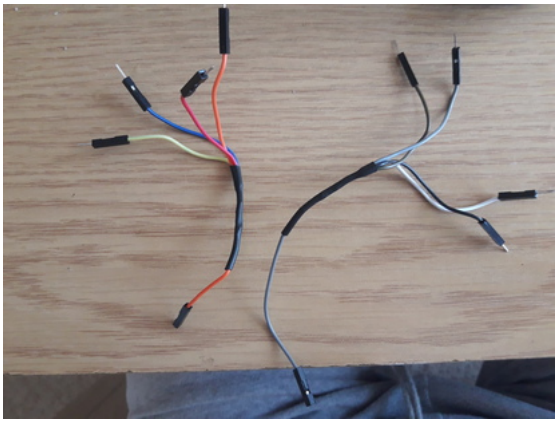
Regulátor otáček Dynamite DYNS2210

Obrázek A.6: Regulátor otáček

[2]Převzato z https://www.pelikandaniel.com/products/L120105/b_3.jpg

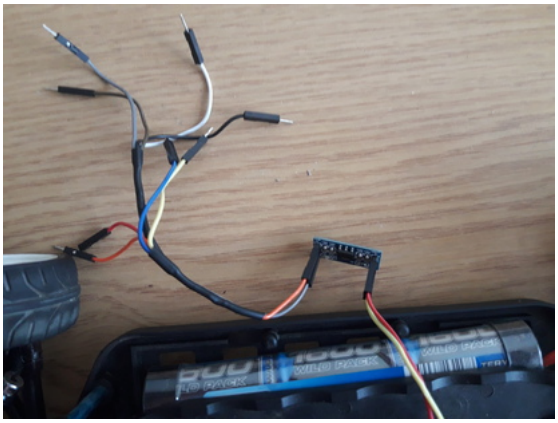
Propojení

Výstup napájení je z regulátoru o velikosti 6 V. Jelikož je potřeba napájet 4 součástky (servo řízení, 2 senzory a stabilizátor), jsou vytvořeny paralelní propoje s 1 vstupem a 4 výstupy. Tím je zajištěno, že napětí zůstane v obvodu stejné.



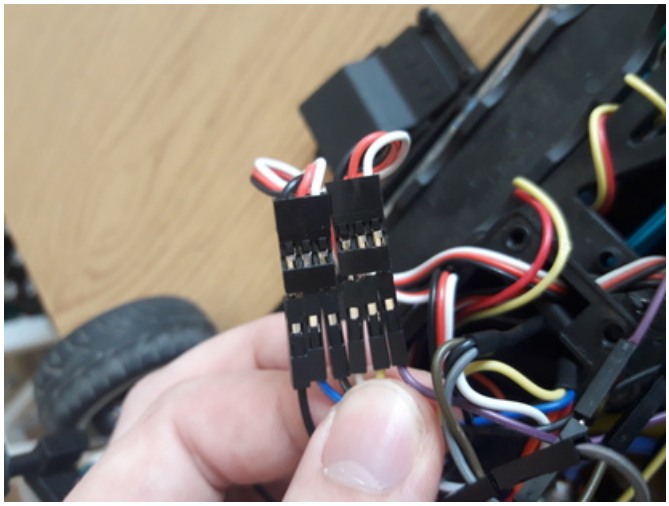
Obrázek A.7: Spájené propoje zásuvka-vidlice-vidlice-vidlice-vidlice

Tyto propoje jsou napojeny na stabilizátor napětí a na výstup stabilizátoru se připojí propoje zásuvka-zásuvka pro napájení mikropočítače ESP32.



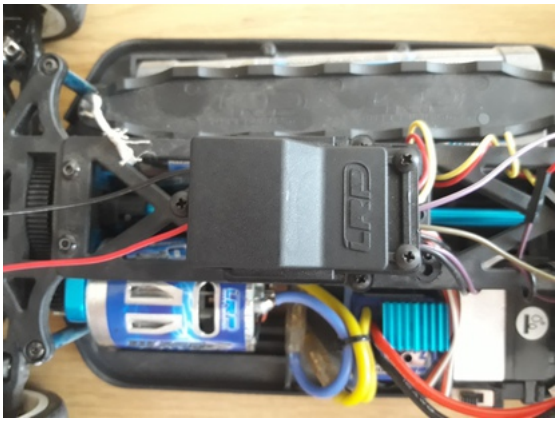
Obrázek A.8: Připojení na AMS1117

Následně se připojí vlastní propoj k regulátoru napětí a servu řízení. Obě součástky mají zásuvkový propoj s barvami bílá/červená/černá, kde bílá je pro PWM, červená je kladné napětí (v případě regulátoru výstup, u serva vstup) a černá slouží jako zem. K bílé zásuvce je připojen propoj vidlice-zásuvka pro vstup PWM modulace ze ESP32.



Obrázek A.9: Připojení k servu a ESC modelu

Aby se vodiče nenamotaly na kola, je využito původního obalu pro přijímač, kde jsou uloženy propoje spolu se stabilizátorem napětí do něj tak, aby ven zůstaly vyčnívat propoje pro napájení součástek (ESP, 2x SR04) a vstupy pro PWM (servo řízení a elektromotor).

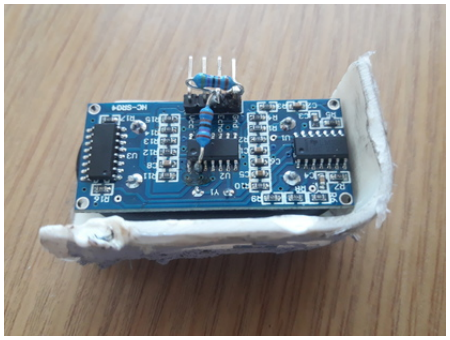


Obrázek A.10: Obal pro původní přijímač RC modelu

Dále je třeba modifikovat senzory, protože jejich výstup, jakožto většiny senzorů na trhu, je 5 V, což neunese ESP32, jelikož maximální napětí, které může přijmout, je 3,6 V. Jednoduché a efektivní řešení v tomto případě je použití odporového děliče. Je třeba upozornit, že tato modifikace může nenávratně poškodit senzor a vykonání tohoto postupu je na vlastní nebezpečí.

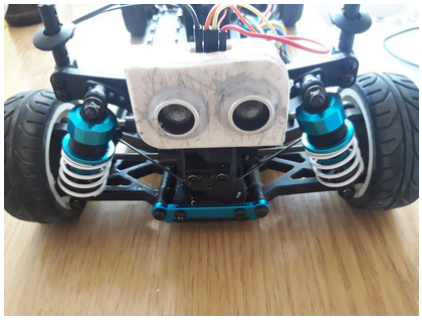
V prvé řadě je třeba přeříznout původní výstupní vodič (Echo). Ten lze najít pomocí multimetru, kde odpor mezi výstupním pinem a U2 přijímačem na senzoru je nulový. Pokud je mezi daným výstupem a pinem nekonečný odpor, je úspěšně přerušeno propojení a lze připájet vlastní rezistory. Mezi přeřezaným výstupem a Echo pinem se připájí 2,7k rezistor a mezi Echo pinem a zemí 4,7k rezistor. Tím je zajištěno na výstupu cca 3,1 V, což stačí pro rozlišení logické 1 od 0.[3]

[3]Návod převzat z: https://www.instructables.com/id/Modify-Ultrasonic-Sensors-for-3-Volts-Logic-prepar/



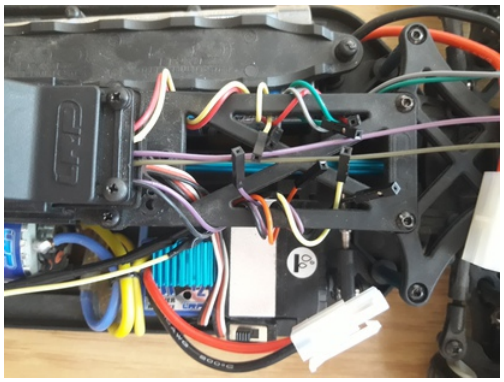
Obrázek A.11: Výsledná modifikace senzoru

Poté je třeba ukotvit senzory k modelu. Zde jsem si vyrobil krabičky, kde jsem vyřezal díru pro piny a senzory tak, abych schoval desku senzoru. Pro přilepení dílů krabiček jsem použil lepící pistole a pro připevnění k modelu byl použit chemoprén, kde mezi krabičku a kostru auta jsem umístil molitan kvůli izolace vibrací. Následně se senzory propojily s mikropočítačem a připojil se propoj pro napájení, kde je důležité, aby napětí přivedené na senzor bylo v rozmezí 4,5 – 5,5 V.



Obrázek A.12: Připevněné senzory vepředu a vzadu na modelu

Nakonec jsou všechny piny vyvedeny před obalem na přijímač, kde je umístěn mikropočítač ESP32.



Obrázek A.13: Vstupy a výstupy pro ESP32

Příloha B

Obsah paměťového média

Na přiloženém paměťovém médiu lze najít tento obsah:

- text práce spolu s latexovými zdrojovými soubory (složka doc)
- zdrojové soubory projektu ovladače pro Android Studio (složka RCcontrol)
- zdrojové soubory projektu mikropočítače pro framework ESP-IDF (složka ESPRC)
- videa, ukazující testování triků modelu (složka videoExamples)
- obrázky s rozložením na jednotlivých testovaných zařízeních (složka androidTest)
- instalační soubor apk pro instalaci aplikace (složka androidApk)

Složka ESPRC

Zde lze najít doporučené členění souborů a složek podle Espressif Systems při použití frameworku ESP-IDF. Pro kompilaci je třeba stáhnout Toolchain z oficiálních dokumentačních stránek a následně z repositáře stáhnout master verzi ESP-IDF[1]. Následně byl stažen example projekt hello_world z ESP-IDF repositáře[2] nad kterou se postavily všechny soubory. Ve složce *main/* lze najít zdrojové kódy, které byly popsány v kapitole 6.3.

Složka RCcontrol

Zde je vytvořený projekt ovladače Android Studiem. Zdrojové kódy lze najít ve složce RCcontrol/app/src/main, kde pod java/com/example/rccontrol lze najít Java třídy a pod res/ jsou grafické zdroje aplikace.

[1] Vše potřebné pro zprovoznění lze najít na: https://docs.espressif.com/projects/esp-idf/en/latest/get-started/

[2] https://github.com/espressif/esp-idf/tree/master/examples/get-started/hello_world