Středoškolská technika 2024

Setkání a prezentace prací středoškolských studentů na ČVUT

AUTONOMNĚ ŘÍZENÁ AUTODRÁHA

Michal Pomajba

Purkyňovo gymnázium, Strážnice, Masarykova 379, příspěvková organizace

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 10: elektrotechnika, elektronika a telekomunikace

Autonomně řízená autodráha

Autonomously controlled car track

Autoři: Michal Pomajba
Škola: Purkyňovo gymnázium, Strážnice, Masarykova 379, příspěvková organizace, Masarykova 379, 69662 Strážnice
Kraj: Jihomoravský kraj
Konzultant: doc. Ing. Tomáš Frýza, Ph.D.
Sudoměřice 2023

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Sudoměřicích dne 26.2.2023……

Michal Pomajba

Poděkování

Chtěl bych poděkovat vedoucímu práce doc. Ing. Tomášovi Frýzovi, Ph.D. za cenné rady, věcné připomínky a velkou trpělivost při zpracování této práce.

Tato práce byla vypracována za finanční podpory JMK.

Anotace

Tato práce Autonomně řízená autodráha pojednává o chování modelu auta na autodráze se zaměřením na řízení rychlosti. Cílem práce je vytvoření systému regulace rychlosti modelu auta v závislosti na profilu trasy. Teoretická část práce se zabývá popisem hardwarových podmínek a použitého softwaru. Praktická část pak měřením chování modelu auta, principem struktury programu a vývojem modelu. Práce vychází z předpokladu, že při změně směru musí auto snížit svoji rychlost, aby bylo schopno dráhu projet. Tento předpoklad se snaží dokázat praktická část. Tohoto má být dosaženo na základě regulace příkonu motoru. Výsledkem práce by měla být aplikace, která umožnuje modelu projet dráhu v nejkratším možném čase.

Klíčová slova

model auta, PWM, autodráha

Annotation

This project, Autonomously controlled autodrome, discusses the behaviour of a model car on a car track with a focus on speed control. The aim of the work is to develop a system for controlling the speed of the model car depending on the profile of the track. The theoretical part of the thesis deals with the description of the hardware conditions and the software used. The practical part then measures the behaviour of the car model, the principle of the program structure and the development of the model. The work is based on the assumption that when changing direction, the car must reduce its speed to be able to traverse the path. The practical part tries to prove this assumption. This is to be achieved by controlling the power input of the motor. The result of the work should be an application that allows the model to traverse the track in the shortest possible time.

Keywords

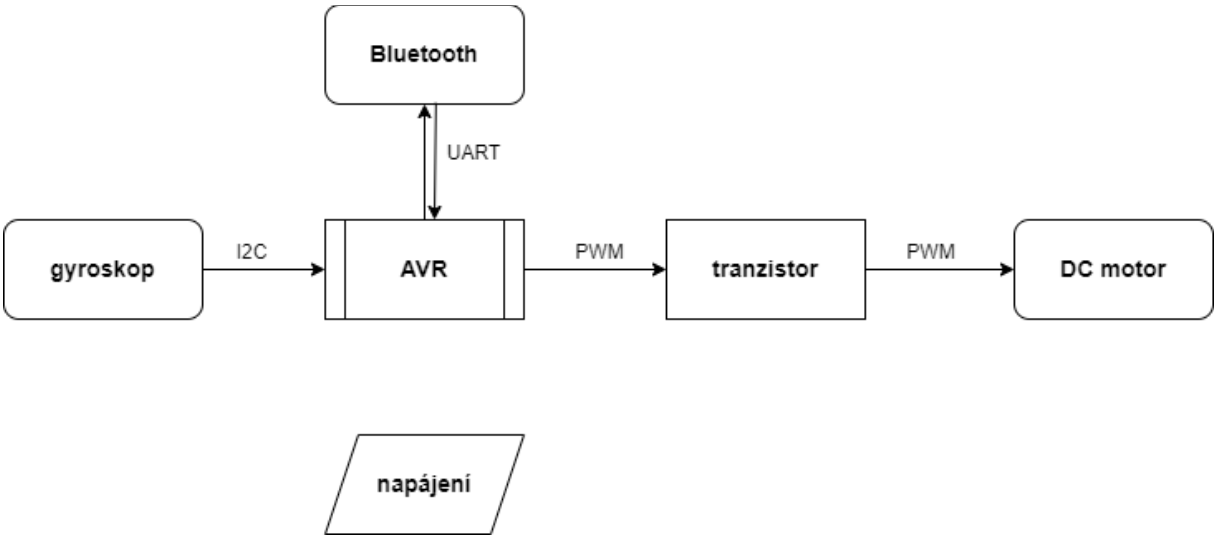
car model, PWM, autodrome

Obsah

1 ÚVOD

Autonomní řízení aut je v dnešní době velmi diskutované téma. V běžném životě nás obklopují různé typy autonomních systému, a právě na tvorbu jednoto specifického autonomního systému se v této práci zaměřím. Cílem této práce je vytvoření modelu auta ovládaném mikrokontrolerem AVR. Model auta bude jezdit po běžně dostupné autodráze. V kolejnicích autodráhy bude stálé elektrické napětí. První projede model auta několik kol pomalou rychlostí. Na základě naměřených dat z gyroskopického senzoru určí čas, který potřebuje na projetí jednotlivých rovinek a zatáček při určité rychlosti a následně si data uloží do paměti ve formě proměnných. Právě tato data budou sloužit pro přiřazovat provozní rychlosti různým částem autodráhy. Tyto rychlosti budou zjištěny na základě měření. Maximální rychlost je nejvyšší možná rychlost, kterou může model auta mít, aby se udržel v autodráze. Těmito rychlostmi bude model auta následně v daných úsecích jezdit. Návrh systému ovládajícího model auta můžete vidět na blokovém schématu na **Obrázek 1**.

Tímto teoreticky dosáhnu nejnižšího možného času, za který může model auta autodráhu projet. Tento projekt by bylo možné použít jako jednoduchý příklad autonomního řízení, nebo i jako autonomního protihráče při závodění s modely aut na autodráze.



Obrázek 1: Návrh blokového schématu celého systému. Vytvořeno v programu diagrams.net.

2 HARDWARE

2.1 Autodráha

Domnívám se, že skoro každý se již někdy setkal s nějakou autodráhou. Základem autodráhy jsou různé bloky, ze kterých se skládá dráha, po které jezdí modely aut. Existují dva základní druhy autodráh. V jednom typu se model auta řídí analogovým signálem a v druhém typu se model auta řídí digitálním signálem. V tomto projektu jsem pracoval s prvním typem, neboli analogovou autodráhou. V každém bloku dráhy jsou většinou dvě kolejnice, které drží model auta na dráze. Díky tomuto stačí regulovat rychlost a model auta bude jezdit pořád dokola. Většina sériově vyráběných autodráh je určena pro 2 hráče, kteří mezi sebou soutěží v tom, čí model auta rychleji projede autodráhu. Úkolem hráče je snižovat nebo zvyšovat rychlost modelu auta na základě toho, jestli model auta projíždí zatáčkou, nebo jede po rovince. Kdyby jel model auta v zatáčce maximální rychlostí, tak by následkem velkého přetížení a těžiště umístěného nad kolejnicí z kolejnice vypadl. Mnou použitou autodráhu můžete vidět na **Obrázek 2**.

Obrázek 2: Ukázka použité autodráhy. Dostupné z: https://www.toy.cz/autodraha-carrera-evo-25220-dtm-fast-lap/

Na stranách kolejnic v autodráze je umístěno elektrické vedení, ze kterého model auta snímá stejnosměrný elektrický proud, pohánějící stejnosměrný elektrický motor. Ten uvádí model auta do pohybu. Elektrické napětí v kolejnicích ovládá hráč přiděleným ovladačem, v němž je zabudovaný regulátor elektrického napětí. Mnou použitá autodráha je napájena napětím 14,8 V a zdroj může poskytovat proud až 2 A. Autodráha se skládá z rovných dílů, kterých je ve stavebnici 16 kusů a délka každého dílu je 340 mm. Dále jsou součástí autodráhy zatočené díly a zatočením 60° a jejich celkový počet je 12 kusů. Dalším nezbytným dílem pro provoz je napájecí díl. Tento díl napájí celou autodráhu a při běžném použití se na něj připojují ovladače rychlosti aut. Součástí různých autodráh jsou také nestandardní díly, jako jsou například naklopené zatáčky a díly pro přejetí z jedné kolejnice na druhou. Tyto díly jsem v tomto projektu nepoužil.

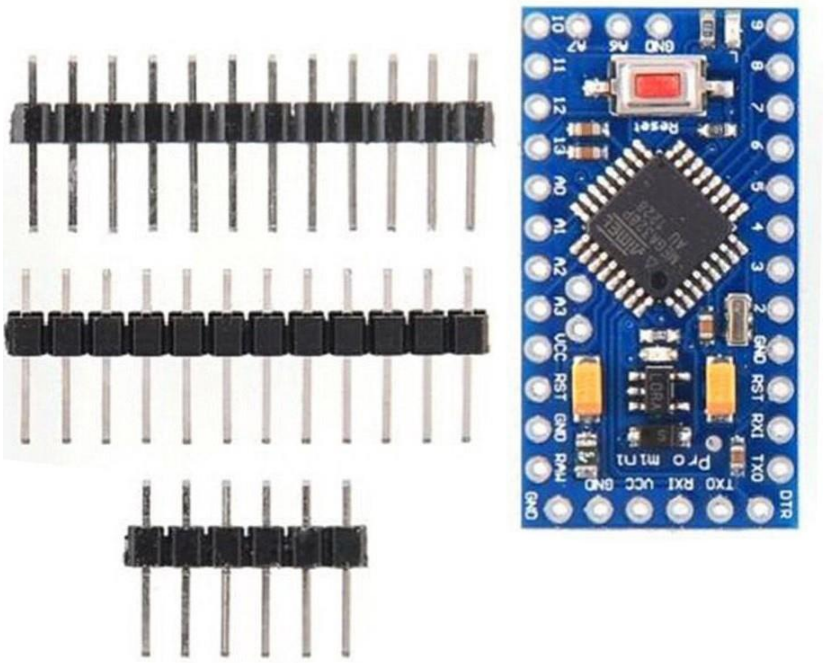
2.2 Arduino

Srdcem tohoto systému je 8bitový mikrokontroler ATmega328P 2.3 zapojený na vývojové desce Arduino Pro Mini [1]. Italská firma Arduino [2] začala vyrábět tyto vývojové desky již v roce 2005 [3]. Od té doby prošly tyto desky patřičným vývojem a jejich výpočetní výkon byl několikanásobně navýšen. Tyto vývojové desky jsou nejvíce využívané pro výukové účely, ale i profesionální uživatele. Například vývojáři je používají pro otestování svých kódů. Vývojové desky od značky Arduino pracují podle nahraného programu napsaného např. v jazyce C++. Arduino je vhodné nejen pro první začátky s programováním, ale i pro vývoj komplexních robotických systémů.

Mezi hlavní výhody Arduina patří široká dostupnost vývojových desek [3], senzorů a příslušenství, ale i nízká pořizovací cena. Arduino je populární po celém světě. Díky tomu je poměrně jednoduché najít nějakou komunitu nebo fórum, které vám může poradit, nebo pomoct vyřešit nějaký problém. Pro Arduino je vyráběno mnoho příslušenství, od gyroskopických, ultrazvukových a optických senzorů, přes krokové motory až po wifi a Bluetooth moduly. Na internetu je zdarma dostupné velké množství knihoven, které zjednodušují psaní kódu pro jednotlivé senzory a moduly. Díky široké škále modelů vývojových desek si můžete vybrat ideální model pro váš projekt. Největší rozdíly mezi jednotlivými modely jsou v rozměrech, výpočetním výkonu, počtu pinů a paměti. V tomto projektu jsem použil vývojovou desku Arduino Pro Mini 2.2.1.

2.2.1 Arduino Pro Mini

Deska Arduino Pro Mini viz. **Obrázek 3** vyniká díky svým malým rozměrům, hmotnosti a nízkému odběru energie [4]. Těchto parametrů bylo dosaženo přítomností pouze nezbytných komponentů pro funkci Arduina. Z tohoto důvodu se tato deska používá nejčastěji ve vestavěných systémech. Tento model je osazen mikrokontrolerem ATmega328P [1]. Tyto desky vyrábí společnost SparkFun ve dvou dostupných variantách 3,3 V model a 5 V model.



Obrázek 3: Arduino Pro Mini. Dostupné z: https://dratek.cz/arduino/880-arduino-mini-atmega328p-5v-16m-klon.html

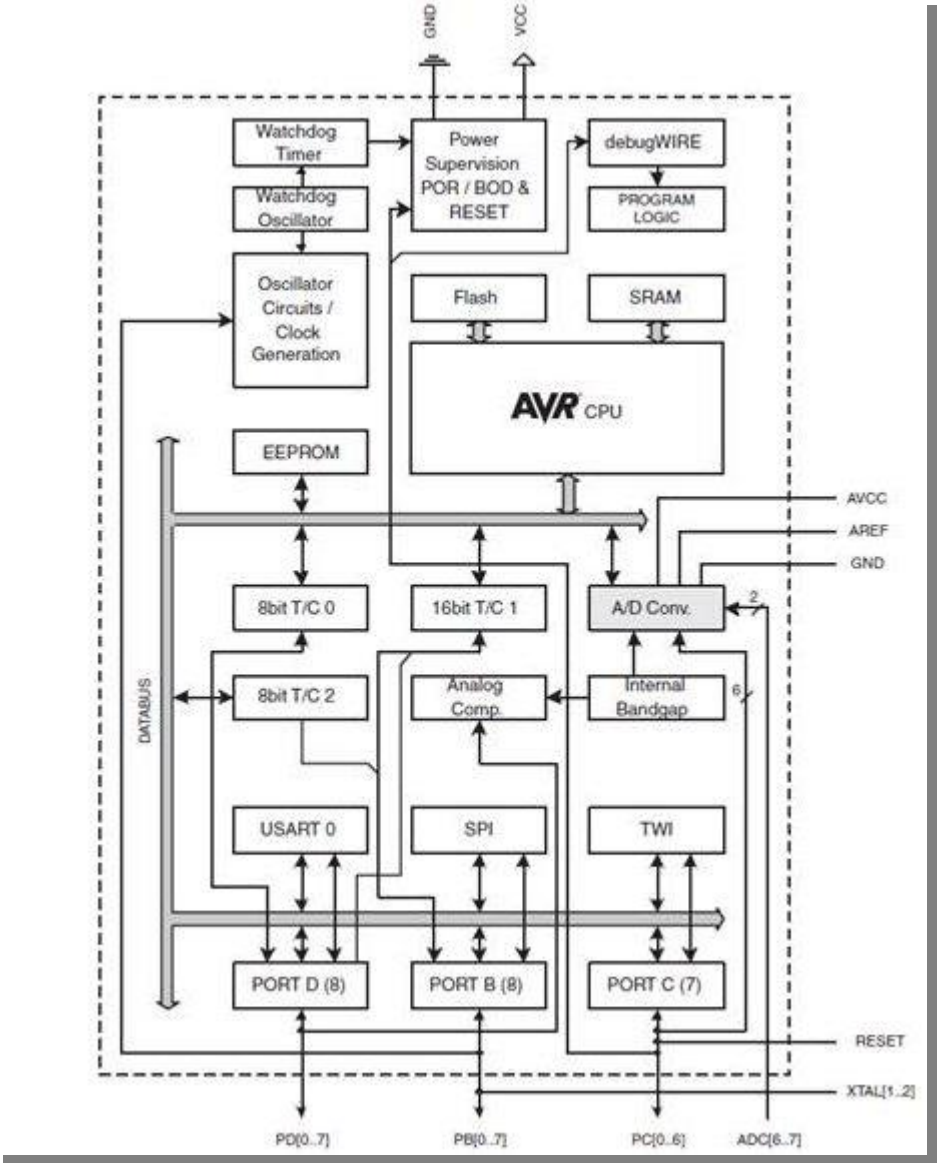
Hlavní rozdíl mezi těmito deskami je v integrovaném oscilátoru. Model s provozním napětím 3,3 V je osazen oscilátorem o frekvenci 8 MHz, zatímco model s provozním napětím 5 V má 16 MHz oscilátor [4]. Já jsem si vybral 5V model, protože Bluetooth modul vyžaduje vyšší napájecí napětí a rovněž 5 V výstup má potenciálně širší využití. V 5V modelu je integrovaný regulátor napětí, který je schopný přijmout napětí 6-12 V a převést ho na 5 V výstup [1]. Tento regulátor využijeme, pokud vstupní napětí připojíme na tzv. Vraw pin. Druhou možností je připojení 5 V zdroje na pin Vcc.

Arduino Mini Pro má 14 digitálních pinů [1], které mají vstupní i výstupní funkci a 6 analogových vstupních pinů. Kvůli malým rozměrů Deska neobsahuje integrovaný programátor. Ten je potřeba dokoupit a připojit přes 4 piny (VCC, GND, TXO, RX1). Tato deska podporuje více druhů komunikací, včetně SPI nebo I2C, kterou budu v tomto projektu používat. Komunikaci I2C (inter-integrated circuit) podporují piny A4 a A5 a komunikaci SPI (serial peripheral interface) podporují piny 10, 11, 12, 13.

2.3 ATmega328P

Srdcem celého systému je AVR mikrokontroler ATmaga328P. Tento osmibitový mikrokontroler je vyrobený architekturou RISC (reduced instruction set computer) [5]. Mikrokontroler obsahuje dva 8-bitové časovače a jeden 16-bitový časovač s oddělenými

předěličkami. Tyto časovače budu později potřebovat ve funkcích interrupts 3.5. Na **Obrázek 4** můžete vidět Blokový diagram mikrokontroleru ATmega328P.



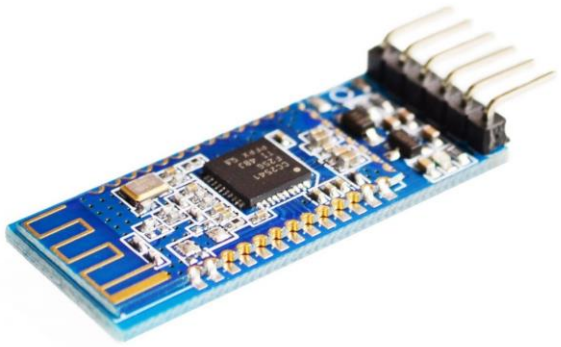
Obrázek 4: Blokový diagram mikrokontroleru ATmega328P. Dostupné z: https://www.arxterra.com/9-atmega328p-timers/

Základní specifikace:

- 16 MIPS (Microprocessor without Interlocked Pipelined Stages)
- 32 kB Flash paměť
- 1 kB EEPROM (Electrically Erasable Programmable Read-Only Memory)
- 2 Kb SRAM (Static Random Access Memory)
- 6 PWM (Pulse-width modulation)
- 32 pinů
- 6 A/D převodník (analog-to-digital)

2.4 Bluetooth

V systému modelu auta je implementovaný Bluetooth modul pro bezdrátovou komunikaci. Komunikace slouží pro jednoduché získávání dat při měření v reálném čase a pro případnou zpětnou vazbu, nebo ovládání modelu auta. V tomto projektu jsem zvolil modul HM-10 s čipem CC2541 [6] viz Obrázek 5. Modul s mikrokontrolerem komunikuje pomocí komunikačního rozhraní UART (universal asynchronous receiver-transmitter) a s jiným Bluetooth modulem komunikuje prostřednictvím Bluetooth 4.0. Maximální rychlost komunikace s mikrokontrolerem činí 115200 Bd, ale já jsem běžně používal továrně nastavenou rychlost 9600 Bd. Modul podporuje obousměrnou komunikaci prostřednictvím Bluetooth i UART. Mezi jeho přednosti patří nízká spotřeba energie, integrovaná anténa a dosah až 60 m. Nastavení modulu probíhá pomocí tzv. „AT“ příkazů, které vysílá mikrokontroler příkazem „Bluetooth.write()“. Jako příklad můžu uvést příkaz „AT+RESET“, který resetuje Bluetooth modul. Více příkazů si můžete najít v příslušné literatuře.

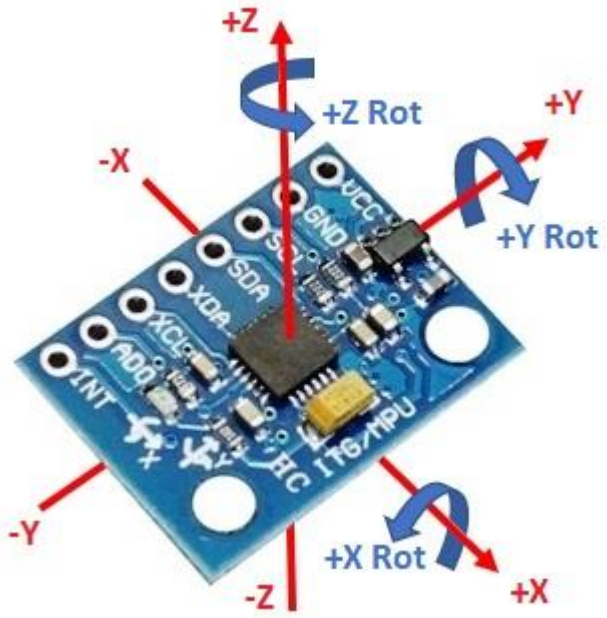


Obrázek 5: Ukázka Bluetooth modulu. Dostupné z: https://dratek.cz/arduino/1312-android-ios-hm-10-klon-bluetooth-4.0-ble-cc2540-cc2541-seriovy-bezdratovy-modul.html

Tento modul jsem využíval především jako vysílač pro měření a zpětnou vazbu systému. Bluetooth signál jsem přijímal zařízením s operačním systémem Android. Pro přijmutí dat je potřeba stažení aplikace. Na internetu je volně dostupných mnoho aplikací s různými funkcemi. Já jsem si vybral aplikaci „Serial Bluetooth Terminal“ od Kai Morich. Součástí aplikace je terminál, který vypisuje přijatá data a také příkazový řádek pro odesílání dat do Bluetooth modulu.

2.5 Gyroskop

Klíčovou část tohoto projektu tvořil kombinovaný senzor GY-521 s čipem MPU-6050 [7] viz **Obrázek 6**. Senzor měří akceleraci a rotaci v osách X, Y a Z. U senzoru je možné nastavit rozsah akcelerace mezi ±2, ±4, ±8, ±16 g. Rozsah rotace je možné nastavit mezi ±250, ±500, ±1000, ±2000 °/s. Senzor umí také měřit teplotu, ale to jsem v tomto projektu nepotřeboval, takže jsem hodnoty teploty ze senzoru ani nenačítal. Senzor pracuje při provozním napětí 3,3-5 V. Senzor komunikuje s mikrokontrolerem pomocí komunikačního rozhraní I2C.



Obrázek 6: Ukázka Gyroskopu. Dostupné z: https://robosynckits.in/product/mpu-6050-3-axis-accelerometer-and-gyroscope-sensor/

2.6 Regulátor napětí

Regulátor napětí obsahuje většina elektrických zařízení, se kterými se v běžném životě setkáváme [8]. Všechna elektrická zařízení mají stanovené ideální napětí, nebo rozsah napětí, při kterém budou správně fungovat. Zdroje elektrického proudu mají často jiné nebo nestabilní napětí. Z toho důvodu se používají regulátor napětí, který zajistí proud o správné hodnotě napětí. Existují různé typy regulátorů napětí.

Mezi ty nejběžnější typy aktivních regulátorů napětí patří lineární a spínací [8]. V tomto projektu jsem použil lineární regulátor L78S09CV [9] s nenastavitelným výstupním napětím o hodnotě 9 V. Toto napětí je vhodné pro napájení vývojové desky Arduino Pro Mini a zároveň pro napájení motoru PWM výstupem z tohoto napětí. Tento regulátor napětí má tři piny. Jeden pro vstupní proud, jeden pro uzemnění a jeden pro výstupní proud.

2.7 Spínací tranzistor

Tranzistor je vlastně elektrický spínač, který však umí proud v obvodu i částečně regulovat [10]. Díky těmto vlastnostem je pro něj skvělým využitím tvorba zesilovačů. V tomto projektu jsem použil tranzistor TIP31A [11], pro ovládání stejnosměrného motoru modelu auta. Je to bipolární NPN tranzistor, který má 3 piny: báze, kolektor a emitor. Pin kolektoru je připojený k motoru, emitor je připojený ke zdroji a báze je připojená k PWM (pulse width modulation) výstupu Arduina. Tranzistor vlastně zesiluje PWM výstup, který následně pohání motor. Mezi pinem báze a emitorem má tranzistor úbytek napětí 0,7 V.

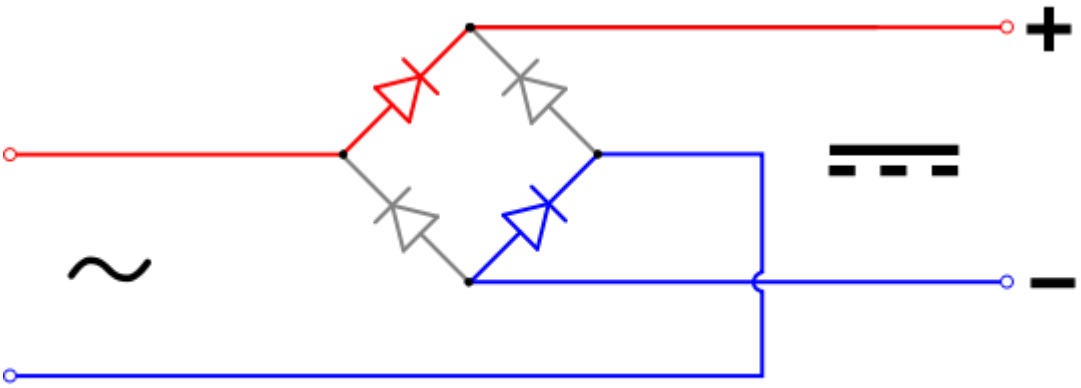
2.8 Stejnosměrný motor

Model auta je poháněn kartáčovým stejnosměrným motorem. Tento motor byl v modelu nainstalován již od výroby a výrobce o něm neposkytuje mnoho informací. Maximální rychlost bez zátěže je zhruba 26 000 RPM [12].

2.9 Diodový můstek

Dioda je polovodičová součástka používaná k usměrnění elektrického proudu [13]. Díky vlastnostem polovodiče uvnitř diody má dioda při vedení elektrického proudu jedním směrem velmi vysoký odpor a při vedení proudu opačným směrem má dioda velmi nízký odpor. V tomto projektu jsem použil diodu SB2100 [14]. V elektrickém obvodu plní dioda ochrannou funkci proti zpětnému přepětí tranzistoru způsobeného motorem. Při zpětném přepětí by se mohl tranzistor nenávratně poškodit.

Diodový můstek se běžně používá k usměrnění střídavého proudu na stejnosměrný [15]. Uvnitř diodového můstku jsou čtyři usměrňovací diody, které jsou zapojeny tak, aby při správném zapojení mezi výstupními piny mohl téct proud pouze jedním směrem, viz **Obrázek 7**. V tomto projektu jsem použil model 2W10 [16], který umožnuje usměrňovat napětí o maximální hodnotě 1000 V a proud o maximální hodnotě 2A. Vstupní piny diodového můstku, označené symbolem střídavého proudu, jsem připojil ke sběrnicím elektrického proudu z autodráhy a výstupní piny jsem zapojil ke stabilizátoru napětí. Kolejnice autodráhy vedou stejnosměrný proud. Pokud bych model auta položil na autodráhu obráceným směrem, změnil bych polaritu vstupního proudu, čímž bych mohl poškodit stabilizátor napětí. Při správném zapojení diodový můstek chrání stabilizátor napětí před přepólováním.



Obrázek 7: Schématická značka a zapojení diodového můstku. Dostupné z: https://en.wikipedia.org/wiki/Diode_bridge

2.10 Kondenzátor

Pro správné fungování regulátoru napětí musím mezi piny Vin - GND a Vout – GND zapojit kondenzátory [17]. Kondenzátory se skládají z vodivých desek a dielektrika mezi nimi. Dielektrikum může být jakýkoli nevodivý materiál. Jednotlivé typy kondenzátorů se od sebe odlišují hlavně typem použitého dielektrika. Mnou použitý kondenzátor byl keramického typu, tedy dielektrikum se skládalo z keramického materiálu [18]. Dále můžeme tyto kondenzátory dělit podle počtu vodivých desek a dielektrik na keramické vícevrstvé a keramické disky. V tomto projektu jsem použil kondenzátory druhého typu keramický disk o hodnotě 0,1 µF.

3 SOFTWARE

3.1 PlatformIO

Při vývoji softwaru pro tento projekt jsem používal vývojové prostředí PlatformIO [19]. Toto vývojové prostředí je možné nainstalovat do mnoho různých aplikací a podporuje operační systémy Windows, MacOS i Linux [20]. PlatformIO jsem používal v aplikaci Visual Studio Code vytvořené firmou Microsoft. Na vývoj softwaru pro platformu Arduino se často používá vývojové prostředí Arduino IDE, avšak PlatformIO má oproti němu řadu výhod. Vývojáře může potěšit možnost kompilace programu pro více druhů desek. Výhodou jsou také návrhy dokončování příkazu a případné návrhy na opravu špatně napsaných příkazů. Není nutná přítomnost knihovny "Arduino.h" při programování v jazyce C++ a neposlední výhodou je jednodušší přidávání knihoven.

3.2 Programovací jazyk C++

Program modelu auta je založený na programovacím jazyce C++. Tento programovací jazyk byl založen již v roce 1980 Bjarnem Stroustrupem [21]. Jazyk C++ vychází z jazyka C, který byl vytvořený o 10 let dříve. Tyto dva jazyky jsou si velmi podobné. Zatím co C byl původně vyvinut pro programování operačních systému, C++ byl vyvinut jako univerzální programovací jazyk. Mnoho populárních systémů, jako například Google, Amazon nebo Facebook jsou alespoň z části napsané tímto jazykem.

Mezi základní koncepty jazyka C++ patří proměnné, řídící struktury, datové struktury, syntaxe a nástroje [21]. Proměnné jsou základem programovacího jazyka, je to způsob, jak uložit informace po delší dobu. Kompilátor čte kód shora dolů a většinou zleva doprava, toto je tzv. „tok kódu". V určitých situacích se může stát, že kompilátor musí kus kódu přeskočit, nebo se musí ve čtení kódu zastavit. Pod slovem nástroje si můžeme představit nějaký software, který nám zjednodušuje programování. Existuje mnoho různých nástrojů, avšak jedním z těch nejdůležitějších nástrojů je IDE (integrated development environment). IDE zajišťuje organizování složek, poskytují nám přehledný způsob, jak je prohlížet a mnoho dalšího. Tento programovací jazyk je momentálně nejvíce používaný v operačních systémech, hrách, prohlížečích a v neposlední řadě bankovních aplikacích.

3.3 Typy použité komunikace

Komunikace jsou způsoby jak mezi senzorem a mikrokontrolerem posílat nějaké informace, neboli data. Existuje mnoho různých druhů komunikací s mnoha různými parametry, jako jsou přenosové rychlosti, energetická úspora nebo třeba hardwarové podmínky. Arduino podporuje pouze některé z těchto komunikací, jako jsou například I2C, SPI a UART. Někdy se setkáme s tím, že senzory podporují více komunikačních standardů, ale většinou podporují pouze jeden. Na komunikace použité v tomto projektu se teď více zaměřím.

3.3.1 I2C

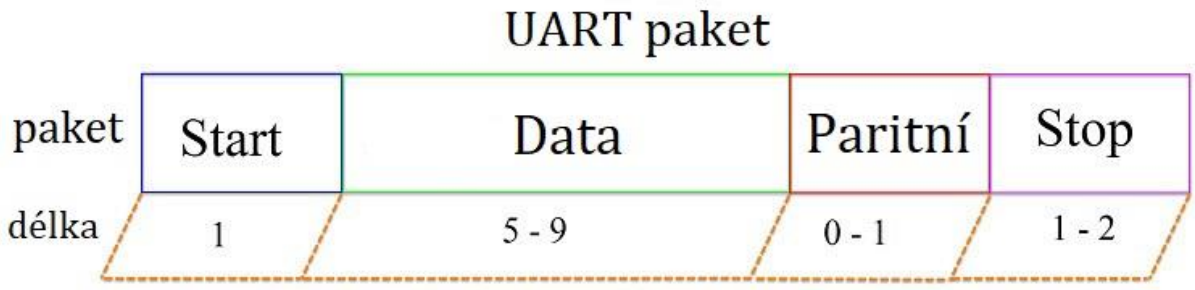
Komunikaci I2C jsem používal pro přenos dat mezi mikrokontrolerem a gyroskopickým senzorem. Výhodou této komunikace je potřeba pouze dvou oddělených vodičů pro přenos dat [22]. Při použití této komunikace rovněž získáváme zpětnou vazbu o správném přijetí dat. V porovnání s komunikaci SPI má však nižší přenosovou rychlost a v jeden okamžik je možné data přenášet pouze jedním směrem. Jak už jsem jednou zmínil, tato komunikace používá dvě přenosové linky, které jsou nazvané SCL (Synchronous Clock) a SDA (Synchronous Data). Vývojové desky Arduino mají vymezené jednotlivé piny, které tuto komunikaci podporují. Arduino Pro Mini má pro datovou linku SCL vymezený pin A5 a pro SDA pin A4 [1].

3.3.2 UART

UART jedna z nejjednodušších a nejpoužívanějších sériových komunikací [23]. Stejně jako SPI a I2C může být komunikace integrována přímo do obvodu mikrokontroleru. Dříve mnoho zařízení, jako například myš, komunikovalo pomocí komunikace UART. Později většina z těchto zařízení přešla na komunikaci pomocí USB. V dnešní době se však s komunikací UART můžeme stále setkat, a to například u GPS a Bluetooth modulů, GSM modemů nebo u bezdrátových komunikačních systémů. V tomto projektu mi sériová komunikace UART zajišťuje přenos dat mezi mikrokontrolerem a Bluetooth modulem.

Hlavní předností této komunikace je přítomnost pouze dvou pinů, které zajišťují komunikaci. Tyto piny jsou pojmenované TX (transmitter) a RX (receiver)[24]. Při zapojení dvou zařízení připojujeme pin RX prvního zařízení na pin TX druhého zařízení a pin TX na pin RX. Na rozdíl od I2C komunikace mohou v jeden okamžik zařízení komunikovat obousměrně. Písmeno 'A' ve zkratce UART znamená, že se jedná o tzv. asynchronní komunikaci, která se vyznačuje tím, že u ní neexistuje žádný hodinový signál pro synchronizaci nebo ověření dat vysílaných z vysílače a přijímaných přijímačem [23].

Sériová komunikace UART, mezi vysílačem a přijímačem, probíhá následujícím způsobem. Vysílač přijme paralelní data, která následně převede na sériová data, která vysílač odešle přijímači [24]. Přijímač data zpátky převede na paralelní data, která může např. poslat do CPU. Paralelní data jsou data, která přijímač přijme v jeden okamžik. Toto je umožněno díky stejnému počtu připojených vodičů jako počtu přenesených bitů signálu. Sériová data jsou data, která vysílač vyšle postupně, a to bit po bitu, v krátkých časových intervalech za sebou. Data v sériové komunikaci jsou přenášená v tzv. paketech. Strukturu typického datového paketu můžeme vidět na Obrázek 8. Start bit je synchronizační bit, který označuje začátek paketu. Datové bity jsou přenášená data k přijímači. Paritní byt je kontrolní bit, který určuje, zda jsou data správně přijata. Datový paket nemusí tento bit vždy obsahovat. Stop bit určuje konec paketu.

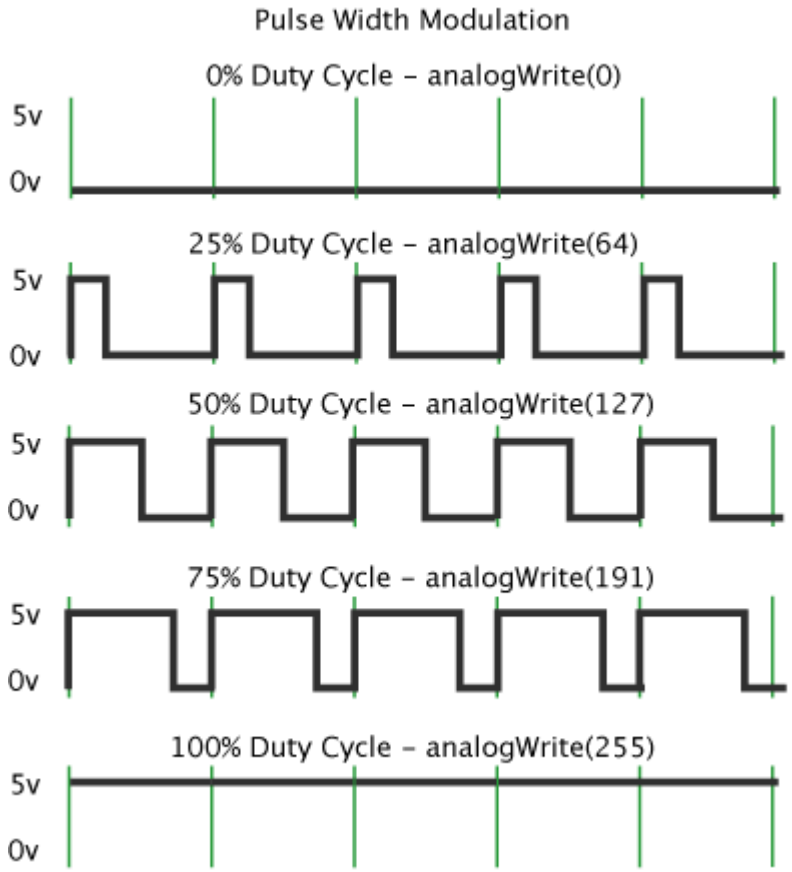


Obrázek 8: Ukázka paketu v UART komunikaci. Dostupné z: https://www.electronicshub.org/basics-uart-communication/

3.4 PWM

PWM výstup je metoda pro získávání analogového výstupu na digitálním pinu. Tuto metodu jsem využil pro ovládání rychlosti stejnosměrného motoru, který pohání model auta. PWM výstup funguje na principu vysokofrekvenčního střídání vysoké a nízké hodnoty na pinu [25]. Pokud bychom přepínali vysokou a nízkou hodnotu s nízkou frekvencí, motor by střídavě zpomaloval a zrychloval. Díky vysoké frekvenci a setrvačnosti motoru se bude motor točit zhruba stále stejnou rychlostí [26]. Rychlost otáčení motoru můžeme regulovat změnou poměru délek vysoké a nízké hodnoty v jednom periodickém cyklu, jak je znázorněno na Obrázek 9. Pokud bude doba vysoké a nízké hodnoty stejná, bude se motor točit stejnou rychlostí, jako kdyby byl napájený polovičním napětím. Pokud budu délku vysoké hodnoty zvyšovat, tak se bude rychlost otáčení motoru zvyšovat a pokud budu délku vysoké hodnoty snižovat, bude rychlost otáčení motoru snižovat.

Příkaz pro vyvolání tohoto výstupu se nazývá “analogWrite(x)” [27]. Za x se dá dosadit pouze celé číslo a hodnota x se pohybuje na škále od 0 do 255, 0 je ekvivalentem pro 0 % a 255 je ekvivalentem pro 100 % viz **Obrázek 9**. Pouze na některých pinech vývojové desky Arduino je možné vytvořit PWM výstup. U desky Arduino Pro Mini jsou to piny 3, 5, 6, 9, 10 a 11. Frekvence PWM výstupů vychází z časovačů mikrokontroleru Arduino [1]. Frekvence výstupu na pinech 3, 9, 10 a 11 je 490 Hz a na pinech 5 a 6 je frekvence 980 Hz [28]. Tyto frekvence se dají případně přenastavit změnou frekvence časovačů mikrokontroleru.



Obrázek 9: Ukázka PWM signálu různých hodnot. Dostupné z: https://docs.arduino.cc/learn/microcontrollers/analog-output

3.5 Interrupty

Interrupty jsou zjednodušeně přerušení hlavního programu, za účelem vykonání vedlejšího programu. Tato přerušení se v počítačích a mikrokontrolerech používají již desítky let [29]. Existují různé typy přerušení s různými funkcemi. Pokud dojde k přerušení programu, mikrokontroler začne vykonávat oddělený program zvaný „ISR“. Do tohoto programu vložíme příkazy, které chceme aby mikrokontroler vykonal a po vykonání těchto příkazu se mikrokontroler vrátí do hlavního programu.

Arduino Pro Mini podporuje 3 typy přerušení, Hardwarové přerušení, přerušení změny pinů a přerušení časovačem [29]. V tomto projektu jsem používal přerušení časovačem, na které se teď více zaměřím. Přerušení časovačem jsou periodická a určujeme je softwarově. Jednou z hardwarových součástek Arduina Pro Mini je 16 MHz oscilátor, na kterém je funkce časovače založena. Mikrokontroler v tomto Arduinu má 3 časovače, timer0, timer1, timer2. Timer1 je 16 bitový časovač a zbylé dva jsou 8 bitové časovače. Počet bitů znamená maximální číslo, do kterého je časovač schopný počítat. 8 bitový časovač může počítat od 0 až do čísla 255 a

16bitový může počítat do čísla 65 535. Každý periodický cyklus oscilátoru se přičte k časovači hodnota 1.

Přerušení může nastat buď při přetečení časovače, nebo když časovač dosáhne určité nastavené hodnoty. Protože 16 MHz je vysoká frekvence a při této frekvenci by časovač přetekl velmi rychle, používají se takzvané předděličky [29]. Předděličky dělí frekvenci oscilátoru, aby hodnota časovače narůstala pomaleji. Předděličky můžou mít hodnoty 1, 8, 64, 256, 1024. Vydělením frekvence oscilátoru těmito čísly nám vzniknou frekvence znázorněné v tabulce. Hodnotu předděliček nastavujeme vždy třemi bity. Označení těchto bitů záleží na použitém časovači. Pokud používáme přerušení při dosažení určité hodnoty časovačem, je třeba tuto hodnotu ještě nastavit. Konečnou frekvenci, podle které bude docházet k přerušení, vypočítáme touto rovnicí.

Frekvence přetečení = Frekvence oscilátoru / (Předdělička x (Hodnota časovače pro přerušení + 1))

Při použití přerušení v tomto projektu, byl použit timer1, docházelo k přerušení při frekvenci 100 Hz podle následující rovnice.

Frekvence přetečení = 16 000 000 / (64 x (2499 + 1)) = 100

4 PRAKTICKÁ ČÁST

4.1 Příprava měřícího modelu

Pro správné určování maximální rychlosti modelu auta na autodráze je nejprve potřeba měřením a následným zpracováním dat z gyroskopu zjistit maximální možnou rychlost v různých zatáčkách a určit, podle kterých dat bude model určovat svoji budoucí rychlost. Abych toto mohl provést, musím nejprve vytvořit měřící model, který mi tato data poskytne.

Základnou tohoto modelu je podvozek ze sériově vyráběného modelu auta pro tuto autodráhu. Tento model je poháněn stejnosměrným motorem. Motor byl ovládán pomocí desky se zabudovaným mikrokontrolerem, která mi byla poskytnuta na Fakultě elektrotechniky a komunikačních technologií v Brně. Deska byla napájena přes kartáčky z kolejnic autodráhy. Tuto desku bylo možné přeprogramovat, pro vytvoření vyššího nebo nižšího příkonu pro napájení motoru. Příkon byl určen softwarovou hodnotou PWM výstupu, který měl frekvenci 246 Hz.

Z této desky byla napájena vývojová deska Arduino Pro Mini 2.2.1, která sloužila pro přijímání, zpracování a odesílání dat z připojených periferií. K vývojové desce byl připojen gyroskop 2.5 a Bluetooth modul 2.4. Bližší informace o těchto periferiích můžete najít v teoretické části této práce. Gyroskop byl na modelu auta orientován osou Y směrem dopředu. Komponenty byly navzájem připojeny pomocí nepájivého pole a kabely s 4.1 piny. Oproti pozdější verzi modelu auta, byla hmotnost auta navýšena o hmotnost nepájivého pole a desky pro ovládání motoru, s čímž je třeba počítat a případně provést ještě měření s finální verzí modelu auta. Hmotnost auta byla zhruba 137 g. Kvůli vyššímu položení komponentů bylo těžiště modelu auta vyvýšeno. S tím je také potřeba počítat při dokončování projektu.

4.2 1. Měření

První měření probíhalo na Fakultě elektrotechniky a komunikačních technologií na VUT v Brně. Zde jsem byl poprvé seznámen s autodráhou a modelem auta. Data v tomto měření jsem získával z tříosého gyroskopu. Měření probíhalo na jednoduchém principu. Mikrokontroler přijímal data, která okamžitě posílal do Bluetooth modulu 2.4. Bluetooth modul bezdrátově přenášel data do telefonu, který je v reálném čase vypisoval na terminál. Tato data jsem si následně v textovém souboru stáhnul. Senzor umí měřit akceleraci, gyroskopická data a teplotu. Protože jsem zatím nevěděl, která data budu při ovládání rychlosti modelu auta používat, nechal jsem do terminálu vypisovat všechna data, kromě teploty. Data o teplotě v tomto projektu potřebovat nebudu.

Při měření jezdil model auta po dráze s oválným tvarem viz Obrázek 10. Dráhu jsem složil z osmnácti bloků, z nichž dvanáct bylo rovných a šest bylo zatočených. Autodráha má vždy dvě kolejnice, ve kterých může jet model auta. Měření jsem prováděl v obou směrech a v obou kolejnicích. Všechna měření jsou zapsána v Tabulka 1. Délka vnitřní dráhy měřila 5,65 m a délka vnější dráhy měřila 6,28 m. Kritická rychlost byla při PWM 95. Model auta při této

rychlosti na vnější dráze v zatáčce vlivem setrvačné síly vypadl z kolejnice autodráhy. Při vypadnutí modelu auta z autodráhy se přeruší napájení celého systému a následkem toho přestane motor pohánět model auta a dojde k přerušení bezdrátové komunikace mezi telefonem a aplikací.

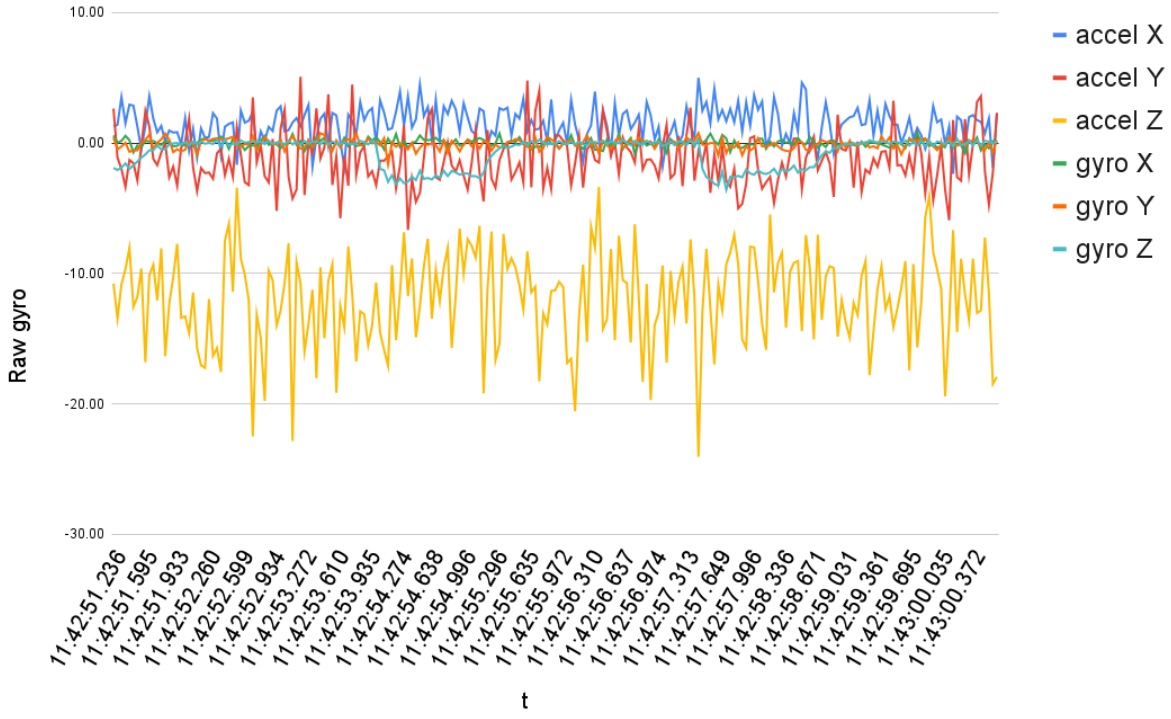
Obrázek 10: Ukázka stavby autodráhy při prvním měření. Foto autor.

Tabulka 1: Naměřené časy a průměrné rychlosti v 1. měření.

č. měření	zatáčka	kolejnice	čas kola t [s]	čas t [s]			průměrný čas t [s]	rychlost PWM	průměrná rychlost v [m/s]	poznámka
				1. kolo	2. kolo	3. kolo				
1	levá	vnější	6,28	11,74	11,8	11,95	11,83	60	0,53	-
2	levá	vnitřní	5,65	11,38	11,41	11,92	11,57	60	0,49	-
3	pravá	vnější	6,28	13,25	13,27	13,38	13,30	60	0,47	-
4	pravá	vnitřní	5,65	11,57	11,77	11,92	11,75	60	0,48	-
5	levá	vnější	6,28	9,92	9,97	10,09	9,99	65	0,63	-
6	levá	vnitřní	5,65	9,05	9,15	9,47	9,22	65	0,61	-
7	pravá	vnější	6,28	11,37	11,36	11,48	11,40	65	0,55	-
8	pravá	vnitřní	5,65	10,44	10,53	10,58	10,52	65	0,54	-
9	levá	vnější	6,28	9,48	9,54	9,57	9,53	70	0,66	-
10	levá	vnitřní	5,65	8,87	8,93	9	8,93	70	0,63	-
11	pravá	vnější	6,28	10,04	10,07	10,1	10,07	70	0,62	-
12	pravá	vnitřní	5,65	8,38	8,77	8,77	8,64	70	0,65	-
13	levá	vnější	6,28	7,42	7,62	7,7	7,58	75	0,83	-
14	levá	vnitřní	5,65	7,09	7,14	7,21	7,15	75	0,79	-
15	pravá	vnější	6,28	8,18	8,21	8,23	8,21	75	0,77	-
16	pravá	vnitřní	5,65	7,35	7,44	7,44	7,41	75	0,76	-
17	levá	vnější	6,28	6,83	7,01	7,02	6,95	80	0,90	-
18	levá	vnitřní	5,65	6,56	6,61	6,61	6,59	80	0,86	-
19	pravá	vnější	6,28	7,5	7,46	7,43	7,46	80	0,84	-
20	pravá	vnitřní	5,65	7,01	7,07	7,06	7,05	80	0,80	-
21	levá	vnější	6,28	5,97	6,05	6,08	6,03	85	1,04	-
22	levá	vnitřní	5,65	5,66	5,65	5,71	5,67	85	1,00	-
23	pravá	vnější	6,28	6,31	6,35	6,37	6,34	85	0,99	-
24	pravá	vnitřní	5,65	5,97	5,96	6,01	5,98	85	0,94	-
25	levá	vnější	6,28	5,41	5,45	5,39	5,42	90	1,16	-
26	levá	vnitřní	5,65	5,21	5,19	5,22	5,21	90	1,09	-
27	pravá	vnější	6,28	5,61	5,67	5,69	5,66	90	1,11	-
28	pravá	vnitřní	5,65	5,33	5,26	5,29	5,29	90	1,07	-
29	pravá	vnitřní	6,28	5	5,04	5,02	5,02	95	1,25	smyk
30	pravá	vnitřní	5,65	4,84	4,89	4,86	4,86	95	1,16	smyk
31	levá	vnější	6,28	-	-	-	-	95	-	výpadek auta
32	levá	vnější	5,65	-	-	-	-	95	-	výpadek auta

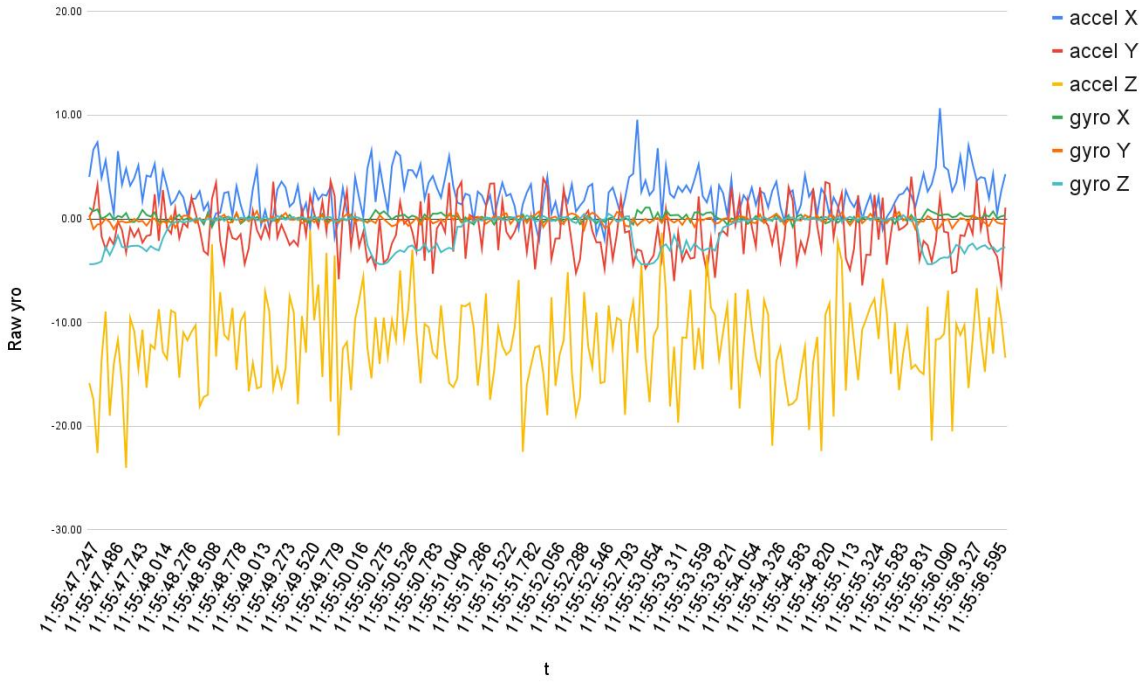
4.3 Zpracování dat

Stažená data v telefonu jsem následně přenesl do počítače. Data byla uložena v textovém souboru formátu TXT. Textové soubory jsem musel přeformátovat do formátu CSV (comma-separated values) pro další použití v programech, jako je například Excel. Jeden řádek textového souboru obsahoval informace o čase, ve kterém byla data načtena, data o akceleraci v ose X, Y, Z a gyroskopická data v ose X, Y a Z. Jednotlivá data byla oddělena středníkem, pro rozpoznání programem při zpracování. Data o čase byla v jednotkách hodin, minut, sekund a milisekund. Jednotlivá měření byla gyroskopem prováděna v krátkých nepravidelných časových intervalech po sobě. Průměrná časová vzdálenost mezi jednotlivými měřeními byla zhruba 43 milisekund. Kvůli připojování Bluetooth modulu a inicializaci programu bylo několik prvních a posledních řádků textového souboru chybných, a proto je bylo potřeba smazat. Upravené textové soubory jsem vkládal do programu Microsoft Excel. Pomocí funkce jsem vytvořil graf dat, naměřených senzorem v závislosti na čase. Některé z těchto grafů znázorňují následující obrázky **Obrázek 11**, **Obrázek 12**, **Obrázek 13** a **Obrázek 14**. Bližší informace o měřeních znázorněných jednotlivými grafy můžete nalézt v Tabulka 1. Informace o čase jsou znázorněny na ose x a data z gyroskopu jsou na ose y.

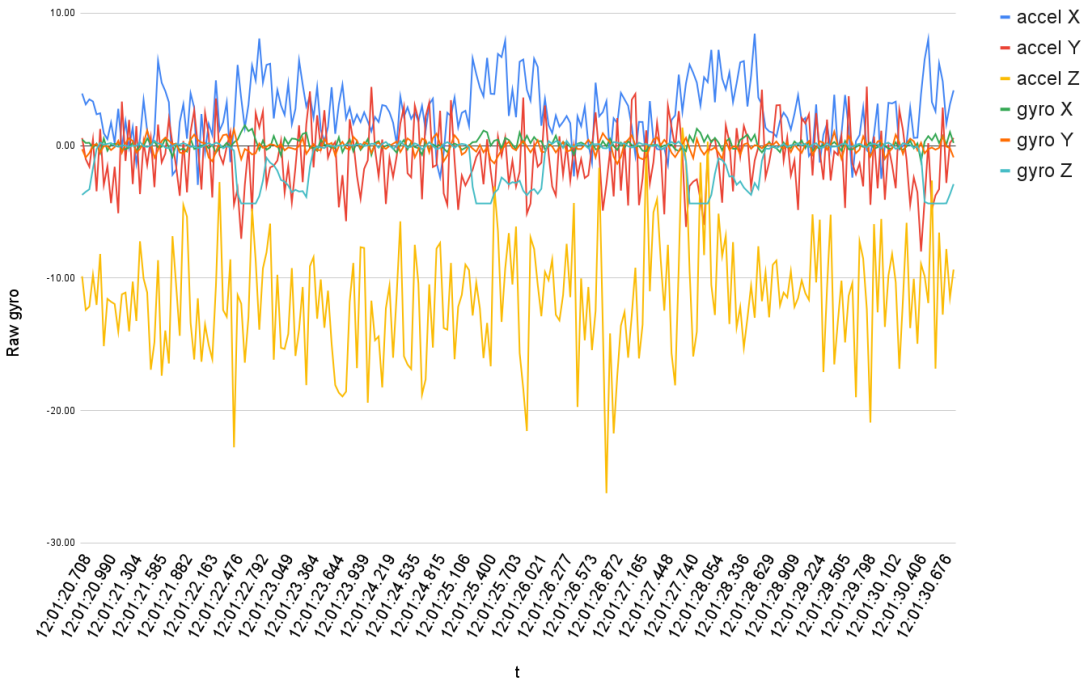


Obrázek 11: Graf měření č. 14. Toto měření probíhalo za konstantního signálu PWM 75 na vnitřní koleji autodráhy (Obrázek 10). Model auta jezdil proti směru hodinových ručiček.

Obrázek 12: Graf měření č. 14. Toto měření probíhalo za konstantního signálu PWM 80 na vnitřní koleji autodráhy (Obrázek 10). Model auta jezdil proti směru hodinových ručiček. Pro lepší orientaci v grafu je křivka akcelerační hodnoty osy X posunuta o +10 bodů a křivka akcelerační hodnoty osy Y o +20 bodů.



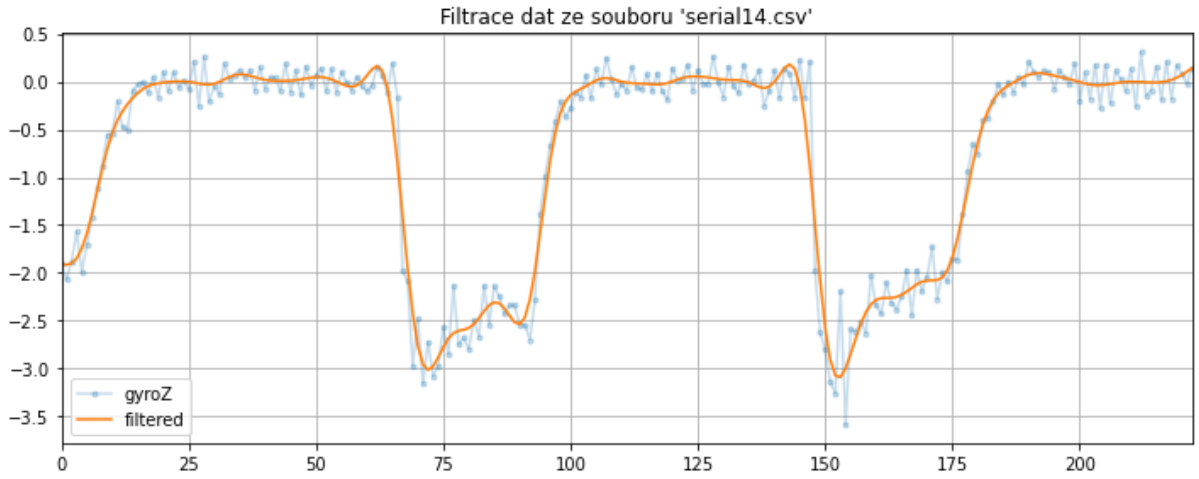
Obrázek 13: Graf měření č. 14. Toto měření probíhalo za konstantního signálu PWM 85 na vnitřní koleji autodráhy (Obrázek 10). Model auta jezdil proti směru hodinových ručiček.



Obrázek 14: Graf měření č. 14. Toto měření probíhalo za konstantního signálu PWM 90 na vnitřní koleji autodráhy (Obrázek 10). Model auta jezdil proti směru hodinových ručiček.

Gyroskop 2.5 byl při měření orientován osou Y dopředu. Již při přípravě měření jsem předpokládal, že gyroskopické údaje o ose X a Y a akceleraci v ose Z nebudu v tomto projektu potřebovat. Tuto myšlenku jsem si později na základě grafu potvrdil. Na zmíněných gyroskopických údajích nejsou znatelné žádné větší odchylky. Při projíždění autodráhou dochází u modelu auta k vibracím a díky vysoké citlivosti gyroskopu jsou výsledná data o akceleraci těmito vibracemi významně ovlivněna. Tento jev můžeme nejlépe pozorovat na datech o akceleraci v ose Z. Předpokládal jsem, že z akceleračních dat osy Y budu moci určovat zrychlování a zpomalování auta, ale vlivem již zmíněných vibrací by to bez přidané filtrace dat nebylo možné. Dále jsem předpokládal, že z akceleračních dat v ose X budu moci určovat hodnotu přetížení v zatáčkách a jejich směr, ale kvůli vibracím to bez filtrace dat nebylo možné. Nakonec jsem se rozhodl pro ovládání modelu auta využívat pouze gyroskopická data z osy Z. U těchto dat je nejlépe znatelný začátek a konec zatáček a také o kolik stupňů za sekundu se model auta v zatáčce otočil, neboli jak moc prudce je zatáčka stočená. V některých částech tato data sice vykazují nějaké odchylky, ale je to nejlepší výstup z gyroskopu, který specifikuje zatáčky. V grafu na obrázku č. 13 si můžete všimnout nárustu gyroskopických hodnot osy Z. Tento jev je způsoben smykem modelu auta v zatáčce.

Pro další ověření dat jsem použil program Colaboratory. V tomto programu jsem dříve zmíněná data přefiltroval. K filtraci byl použit Butterworthův filtr typu dolní propust 3. řádu s mezním kmitočtem na frekvenci 0,2-násobek Nyquistova kmitočtu [30]. Tímto bylo dosaženo přehledného grafu s plynulými křivkami. Na **Obrázek 15** můžete vidět zmíněný graf, kde jsou použita gyroskopická data osy Z, která budu v tomto projektu používat. Tento graf ukatuje znatelný nájezd do zatáčky a také výjezd ze zatáčky prudkým snížením nebo zvýšením hodnot. Dále můžeme na tomto grafu vidět, že rychlost otáčení gyroskopu se v průběhu zatáčky snižovala, což bylo způsobeno třením mezi modelem auta a autodráhou.

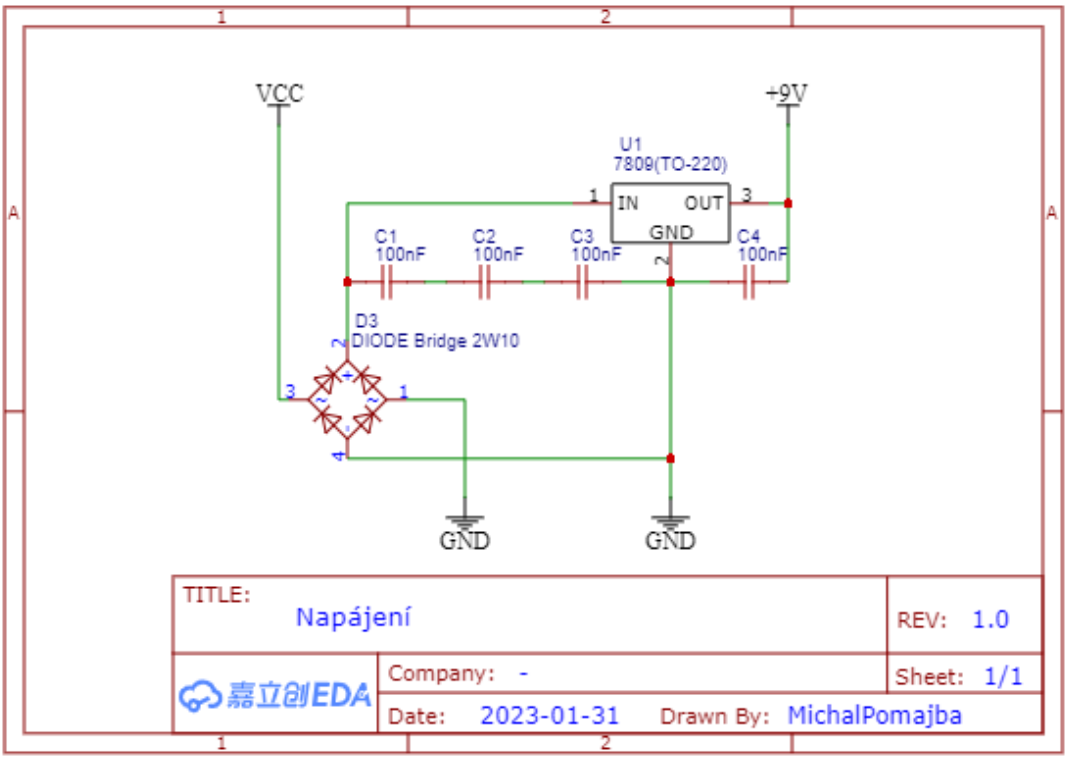


Obrázek 15: Graf přefiltrovaných gyroskopických dat osy Z, filtr typu dolní propust. Mezní kmitočet na frekvenci 0,2-násobeku Nyquistova kmitočtu.

4.4 Napájení

Napájecí zdroj autodráhy má výstupní napětí 14,8V a proud 1 A. Toto napětí je pro regulátor vývojové desky Arduino Pro Mini moc vysoké, a proto jsem musel do obvodu připojit další regulátor napětí 2.6. Tento regulátor napětí vytváří stabilní napětí o hodnotě 9 V, což bude vhodné pro napájení vývojové desky i pro vytváření PWM výstupu, pro pohánění motoru. Oficiální dokumentace k stabilizátoru napětí doporučuje připojit kondenzátor o hodnotě 0,33 µF mezi pin pro vstupní proud a pin pro uzemnění regulátoru a kondenzátor o hodnotě 0,1 µF mezi pin pro výstupní proud a uzemnění stabilizátoru napětí. K výstupnímu pinu a uzemnění jsem připojil keramický diskový kondenzátor s hodnotou 0,1 µF. Protože jsem neměl k dispozici 0,33 µF kondenzátor, zapojil jsem mezi pinem vstupu a uzemněním tři paralelně zapojené keramické diskové kondenzátory s hodnotou 0,1 µF.

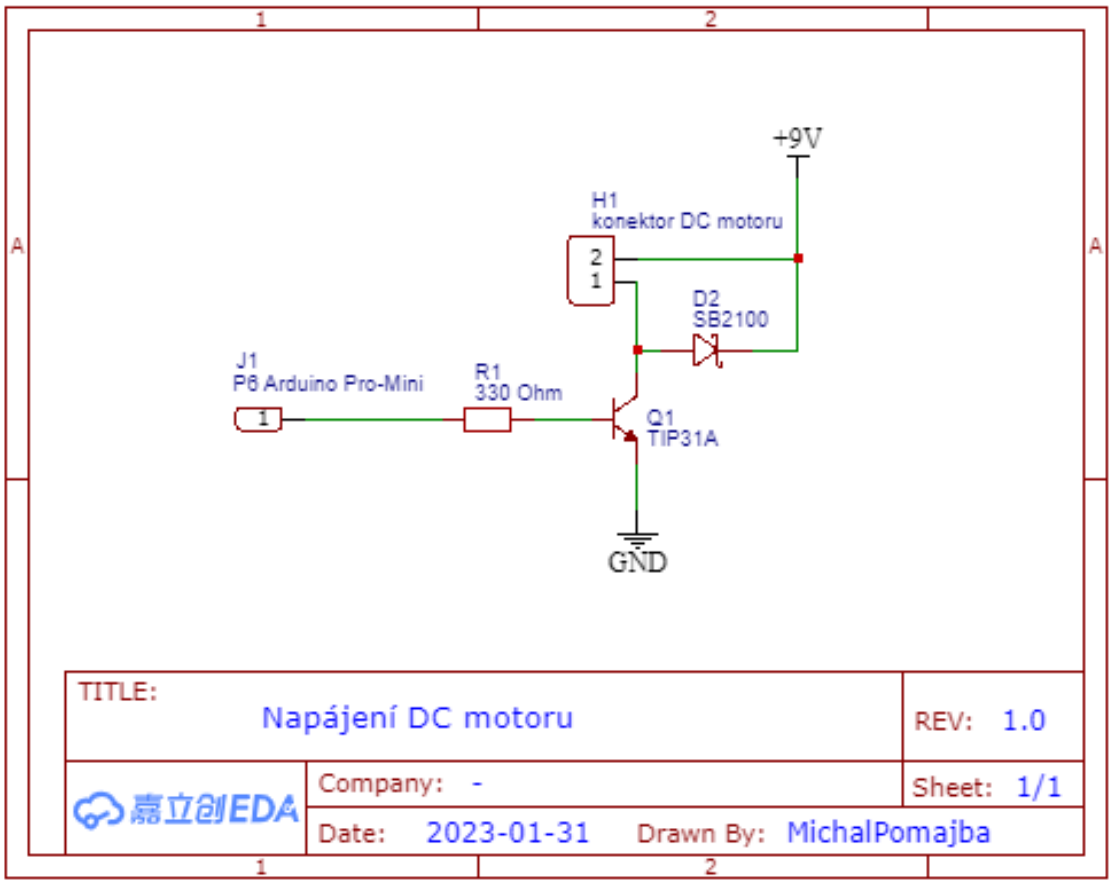
Na jedné straně kolejnice vede vodič kladný pól napájení a na druhé straně kolejnice vede vodič záporný pól napájení. Model auta se na tyto vodiče připojuje vždy dvěma kovovými kartáčky na každé straně. Pokud bych model auta na autodráhu položil obráceně došlo by k prohození pólů, připojených ke stabilizátoru napětí a tím bych ho mohl nenávratně poškodit. Abych tomuto předešel, připojil jsem diodový můstek mezi vodiče napojené na kartáčky a regulátor napětí. Diodový můstek se běžně používá pro přeměnu střídavého proudu na stejnosměrný, to znamená, že při obrácení polarity na vstupních pinech, bude polarita na výstupních pinech stále stejná. Všechna tato zapojení jsou znázorněna na **Obrázek 16**.



Obrázek 16: Schéma zapojení napájecí části systému. Vytvořeno v programu EasyEDA.

4.5 Ovládání motoru

Je více způsobů, jak můžeme ovládat stejnosměrný motor. Základem je k motoru připojit stejnosměrný proud, a potom můžeme ovládat rychlost motoru změnou napájecího proudu, nebo změnou vstupního napětí. Protože mikrokontroler ATmega328P neumí vytvářet analogový výstup, použil jsem pro ovládání motoru PWM výstup. Více o tomto výstupu se dočtete v kapitole 3.4. PWM výstup mikrokontroleru nemá dostatečný proud ani napětí, kvůli tomu jsem musel použít tranzistor 2.7. Tranzistor podle tohoto výstupu spíná a rozepíná silnější proud. Díky zapojení silnějšího zdroje vytváří stejný signál s vyšším napětím a větším procházejícím proudem. Tyto veličiny jsou závislé na připojeném zdroji a parametrech tranzistoru. Tranzistor byl zapojen následovně. Pin kolektor je připojen k motoru, emitor je připojen ke zdroji a báze je připojena k PWM výstupu Arduina. Druhý pin motoru byl připojen ke kladnému pólu napájení. Toto zapojení je znázorněno na **Obrázek 17**.



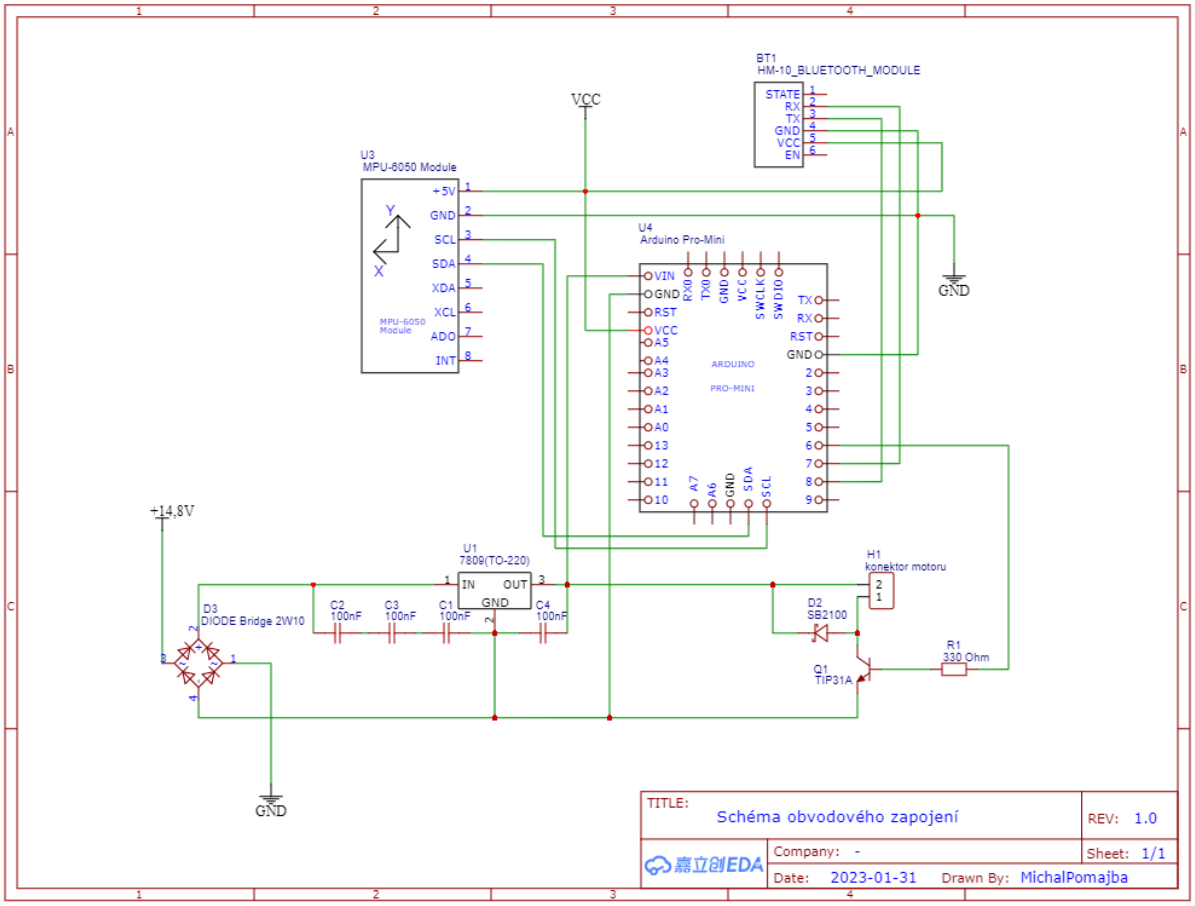
Obrázek 17: Schéma zapojení napájecího systému pro DC motor. Vytvořeno v programu EasyEDA.

Pro PWM výstup jsem používal pin D6. PWM výstupy z pinů D5 a D6 jsou závislé na časovači Timer0. Z dat z měření, která mi byla poskytnuta na VUT v Brně, jsem zjistil, že vhodná frekvence PWM výstupu je 246 Hz [31]. Výchozí nastavení pro časovač Timer0 je 980 Hz [28]. Kvůli správnému ovládání motoru jsem musel změnit frekvenci časovače Timer0 na 246 Hz.

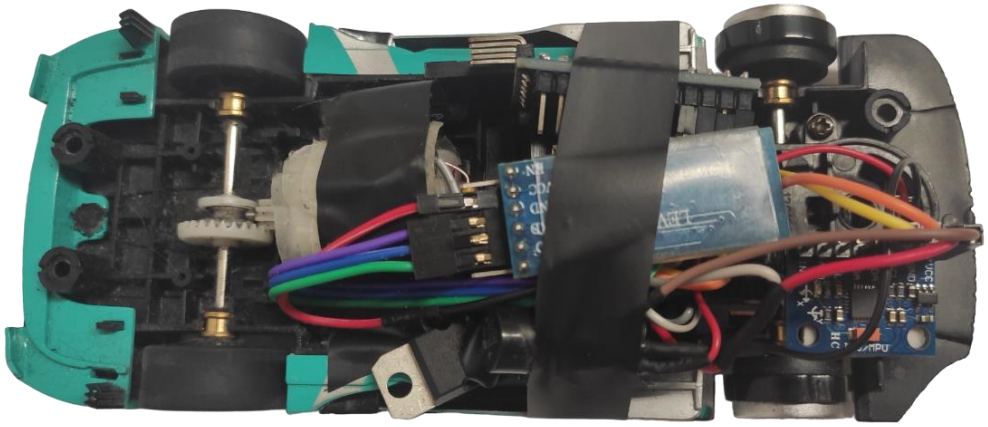
4.6 Měřící model č. 2

V prvním měřícím modelu byly jednotlivé komponenty zapojeny pomocí nepájivého pole. Toto nepájivé pole zvyšovalo hmotnost modelu auta a vyvyšovalo těžiště modelu auta. Tím negativně ovlivňovalo výsledky měření. Abych mohl budoucí naměřené hodnoty použít při určování rychlosti modelu auta, musím sestavit zapojení, se kterým bude mít auto co nejvíce podobné parametry finálnímu modelu. Tohoto dosáhnu absencí nepájivého pole a přímým napojením komponentů pomocí kabelů.

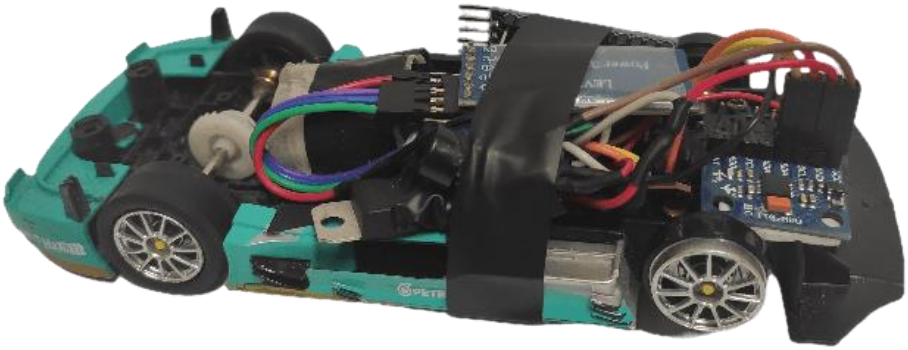
Kabely byly napojeny pomocí standardních konektorů. Na některé komponenty, jako např. Tranzistor, nemohly být připojeny pomocí standardního konektoru, proto jsem je musel připájet a zaizolovat. Komponenty potřebovaly více vstupů pro napájení, než má Arduino Pro Mini napájecích výstupů, proto jsem na některých kabelech vytvořil uzly, díky kterým jsem mohl propojit více pinů z jednoho výstupu. Všechny komponenty se mi podařilo naskládat na podvozek modelu auta a provizorně přichytit pomocí lepící pásky viz **Obrázek 19** a **Obrázek 20**. V tomto případně je nejvhodnější toto provizorní přichycení, kvůli potenciálně možným budoucím změnám. Celé schéma obvodového zapojení můžete vidět na **Obrázek 18**.



Obrázek 18: Celkové schéma obvodového zapojení. Vytvořeno v programu EasyEDA.



Obrázek 19: Model auta autonomně řízené autodráhy. Model obsahuje všechny komponenty potřebné pro autonomní provoz. Foto autor.



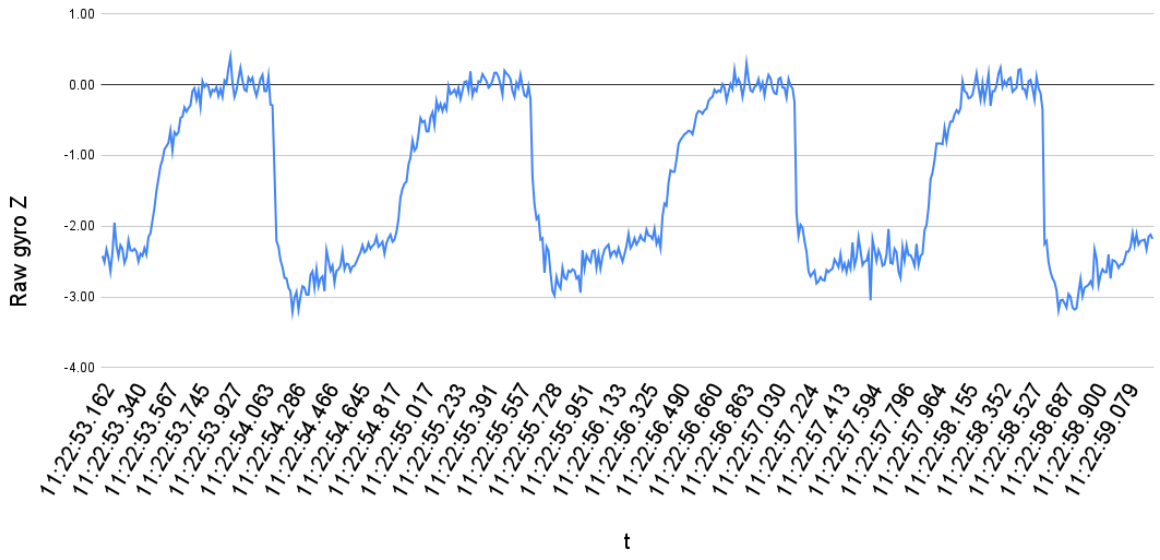
Obrázek 20: Model auta autonomně řízené autodráhy. Model obsahuje všechny komponenty potřebné pro autonomní provoz. Foto autor.

4.7 Měření č. 2

V druhém měření jsem se zaměřil na maximální bezpečnou rychlost na projetí různě dlouhých zatáček, hodnotu gyroskopických dat osy Z v těchto zatáčkách, zjištění vhodné rychlosti pro měření, čas za který auto projede zatáčku při měřící a maximální bezpečné rychlosti a čas za který projede auto rovinku při měřící a maximální bezpečné rychlosti.

4.7.1 Rychlost měřící sekvence

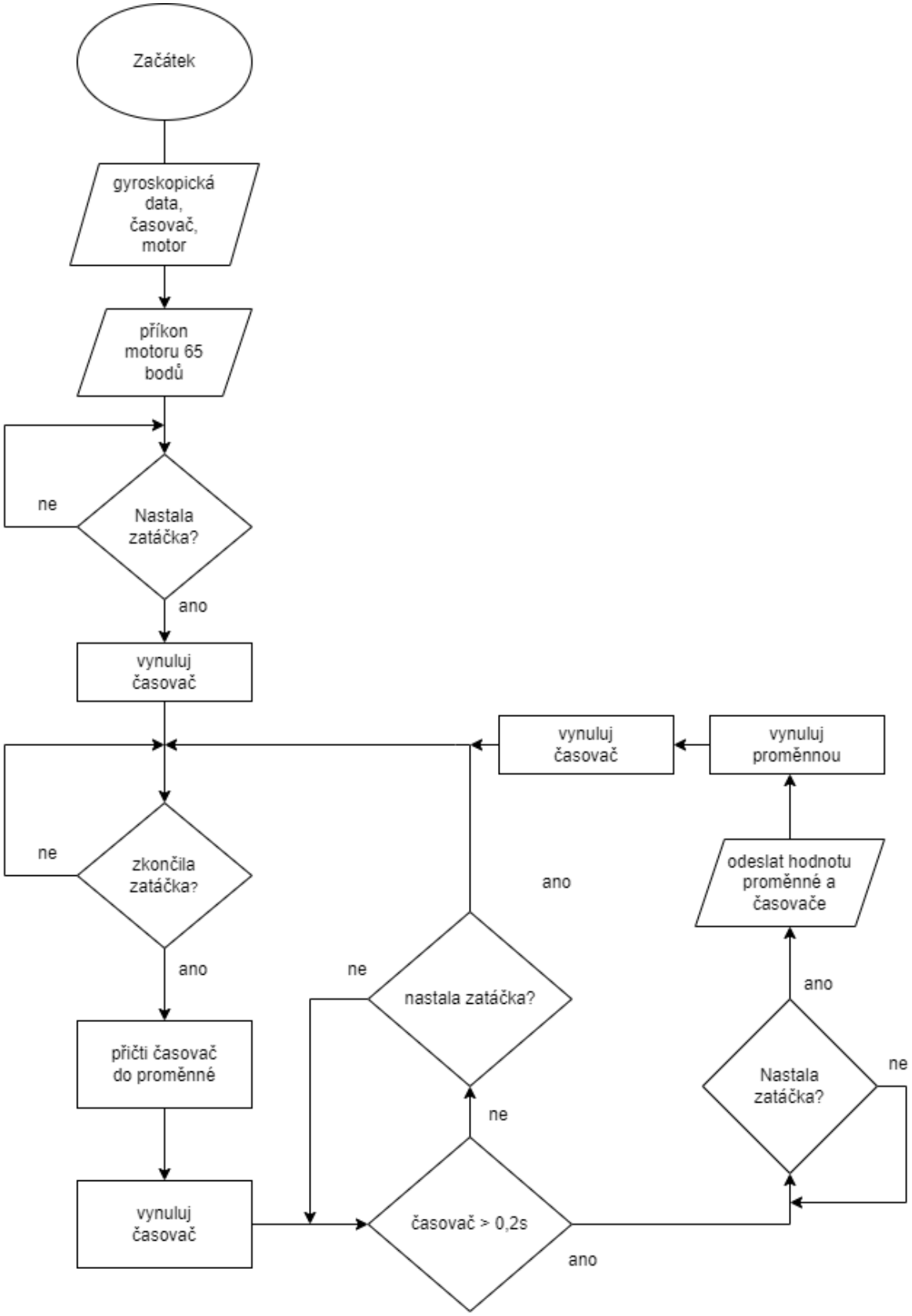
V první části měření jsem se zaměřil na nalezení vhodné rychlosti pro měřící sekvenci modelu. Sestavil jsem autodráhu, která obsahovala různé délky zatáček. Nechal jsem jezdit model auta po dráze a odesílat gyroskopická data prostřednictvím bezdrátové komunikace Bluetooth. Tato data jsem následně zpracoval a vytvořil grafy závislosti hodnoty gyroskopu na čase. Při příliš nízké rychlosti byla hodnota gyroskopu v zatáčkách moc nízká a při průjezdu dlouhou zatáčkou se hodnota rapidně snižovala. Příliš vysoká rychlost způsobovala v zatáčkách smyk modelu auta a ten negativně ovlivňoval gyroskopická data a dále hrozilo vypadnutí modelu auta z autodráhy. Na základě dat z gyroskopu a pozorování jsem jako ideální rychlost pro měřící sekvenci zvolil rychlost při hodnotě PWM 65 viz **Obrázek 21**. Při této rychlosti byl nárust a pokles gyroskopických hodnot na začátku a na konci zatáčky nejlépe rozpoznatelný a nehrozilo vypadnutí modelu auta z autodráhy.



Obrázek 21: Graf gyroskopických dat v závislosti na čase za konstantního signálu PWM 65.

4.7.2 Způsob měření

Nejdříve je třeba si uvědomit, jak model auta může rozpoznat zatáčku. V prvním měření jsem zjistil, že počátek a konec zatáčky je možné bezpečně rozpoznat z gyroskopických dat osy Z. Pokud hodnota těchto dat přesáhne hranici 1,5, model auta projíždí zatáčkou. Jestliže hodnota tuto hranici nepřesáhne, model auta projíždí po rovince. Časovačem jsem měřil délku úseku s vyšší a nižší hodnotou. Kvůli zvýšenému tření a vibracím v zatáčkách, docházelo k výkyvům hodnot. Tento jev bylo potřeba ošetřit. Ošetření probíhalo následujícím způsobem. Pokud nastalo snížení hodnoty pod danou hranici, mikrokontroler si do proměnné napsal hodnotu časovače a časovač vynuloval. V případě, že byla hodnota během 0,2 sekund znova navýšena, mikrokontroler rozpoznal, že model auta projíždí stále stejnou zatáčkou a při dalším snížení hodnoty gyroskopu přičte novou hodnotu časovače k původní hodnotě časovače. Čas 0,2 není dostatečně dlouhý, aby se během něj model dostal přes jeden díl rovinky do další zatáčky. Pokud tedy není čas vyšší než 0,2 s je jisté, že model v zatáčce setrvává. Tento děj se může zopakovat i několikrát za sebou, dokud hodnota gyroskopu nezůstane nízká po dobu alespoň 0,2 sekund. Při posledním snížení hodnoty se časovač naposledy přičte a součet všech hodnot časovačů během jedné zatáčky si mikrokontroler uloží do proměnné a bude tuto hodnotu považovat za čas zatáčky. Zjednodušeně, tento systém funguje tak, že pokud na základě výkyvu gyroskopické hodnoty zaznamená mikrokontroler dvě a více zatáček v průběhu jedné zatáčky, sečte časy těchto zatáček, včetně krátkých rovinek mezi nimi. Celý princip měřícího programu je znázorněn na blokovém schématu na **Obrázek 22**.



Obrázek 22: Blokové schéma programu pro měření časů rovinek a zatáček při měřící rychlosti 65 bodů. Vytvořeno v programu diagrams.net.

4.7.3 Měřící sekvence

Touto rychlostí jsem provedl měření času zatáček a rovinek. Vždy jsem nechal model auta projet 10 zatáček, kvůli rozjetí motoru a ustálení rychlosti. Časy následujících 20 zatáček arovinek jsem zprůměroval a zapsal do **Tabulka 2** a **Tabulka 3**. Do tabulek jsou již zapsána data z dalších měření z článků 4.7.4 a 4.7.5. Měření jsem prováděl u čtyř typů zatáček a čtyř typů rovinek. Obě tabulky jsou rozděleny na měřící a provozní sekvenci. V první tabulce můžete vidět časy a rychlosti zatáček. Tyto zatáčky jsou rozděleny podle délky neboli počtu dílů, ze kterých se zatáčka skládala. Dále byli zatáčky rozděleny podle typu kolejnice na vnitřní a vnější. Při tomto měření byli před zatáčkou vždy dvě rovinky. Druhá tabulka znázorňuje časy a rychlosti rovinek, které byli rozdělené podle délky a typu kolejnice předcházející zatáčky. Každé rovince vždy předcházela trojdílná zatáčka. Podle naměřených časů a hodnot jsem následně určoval počet zatáček autodráhy a časy vysokých rychlostí na rovinkách.

Tabulka 2: Časy jednotlivých typů zatáček při měřící nebo provozní rychlosti.

Typ zatáčky	kolejnice	měřící sekvence		provozní sekvence	
		rychlost PWM	průměrný čas t [s]	rychlost v zatáčce PWM	průměrný čas t [s]
jednodílná	vnější	65	0,294	71	0,251
	vnitřní	65	0,248	71	0,197
dvojdílná	vnější	65	0,624	71	0,515
	vnitřní	65	0,485	71	0,439
trojdílná	vnější	65	0,981	71	0,816
	vnitřní	65	0,773	71	0,698
čtyřdílná	vnější	65	1,379	71	1,15
	vnitřní	65	1,253	71	1,102

Tabulka 3: Časy jednotlivých typů rovinek při měřící nebo provozní sekvenci.

typ rovinky	předešlá zatáčka	měřící sekvence		provozní sekvence			
		rychlost PWM	průměrný čas t [s]	max. rychlost PWM	čas max. rychlosti t [s]	brzdící rychlost PWM	průměrný čas rovinky t [s]
jednodílná	vnější	65	0,355	255	0,08	15	0,308
	vnitřní	65	0,313	255	0,09	15	0,283
dvojdílná	vnější	65	0,621	255	0,15	15	0,487
	vnitřní	65	0,712	255	0,16	15	0,364
trojdílná	vnější	65	0,878	255	0,21	15	0,71
	vnitřní	65	0,896	255	0,23	15	0,608
čtyřdílná	vnější	65	1,098	255	0,28	15	0,848
	vnitřní	65	1,145	255	0,31	15	0,788

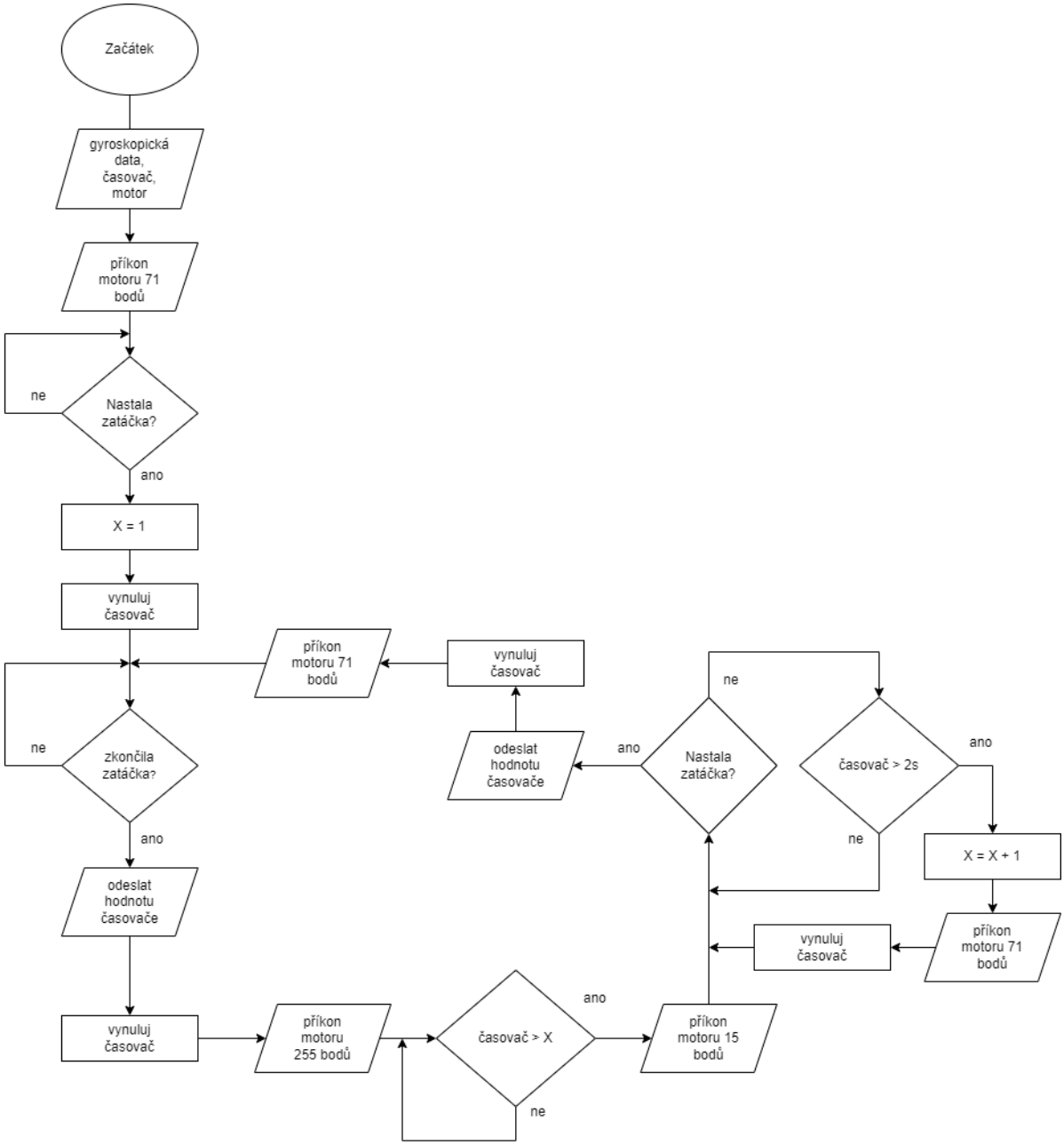
4.7.4 Rychlost zatáček

V dalším měření jsem zjišťoval maximální rychlost, kterou může model auta bezpečně projet zatáčku. Vždy jsem sestavil osově souměrnou autodráhu, ve které byl vždy pouze jeden typ zatáček a rovinek. Postupně jsem přidával rychlost, dokud se model auta v zatáčce nedostával do smyku. Zjistil jsem, že maximální příkon motoru, při kterém může model auta projet autodráhu bez smyku je 71. Na základě dalších testů jsem zjistil, že tato rychlost je stejná pro všechny typy zatáček.

4.7.5 Měření provozní sekvence

V poslední části jsem se zaměřil na zrychlování a zpomalování na rovinkách. Aby čas, za který projede model auta rovinku byl co nejnižší, musel jsem na začátku rovinky navýšit příkon

motoru na maximální hodnotu a pár okamžiků před koncem snížit příkon motoru na minimální hodnotu. Po několika měřeních jsem zjistil, že model auta se při této regulaci postupně rozjíždí a teprve po několika kolech dojde k ustálení rychlosti. Pokud jsem nastavil kratší dobu vysoké rychlosti, model auta nedojel do zatáčky, ale zastavil se před ní. Naopak pokud jsem nastavil vyšší čas vysoké rychlosti, model auta postupně dosáhl tak vysoké rychlosti, že nebyl schopný se udržet v autodráze. Tento problém jsem vyřešil navýšením brzdícího příkonu na hodnotu PWM 15, tedy zhruba 6 % výkonu motoru. Díky tomuto brzdnému příkonu zde byla nižší šance na úplné zastavení modelu před zatáčkou. Tímto jsem prodloužil dobu, při které bude mít autíčko vhodnou rychlost pro nájezd do zatáčky. Měření probíhalo následujícím způsobem. Když mikrokontroler zaznamenal konec zatáčky, nastavil motoru maximální příkon a vynuloval časovač. Po velmi krátkou dobu jel maximální rychlostí a následně nastavil příkon motoru na brzdící hodnotu PWM 15. Pokud časovač přesáhl hodnotu 2 s, navýšil dobu maximální rychlosti o 0,01 s a nastavil příkon motoru na hodnotu PWM 71. Tímto zjistil, že se mu nepodařilo přijet až do zatáčky a pro příští kolo navýšil délku maximální rychlosti. Pokud mikrokontroler zaznamenal zatáčku, zvýšil příkon motoru na hodnotu PWM 71, aby mohl bezpečně projet zatáčkou. Celý princip měřícího programu je znázorněn na blokovém schématu na **Obrázek 23**. Při takovémto provozu modelu auta jsem měřil časy rovinek a zatáček. Vždy jsem naměřil 20 časů rovinek nebo zatáček. Z těchto časů jsem vypočítal průměr, který jsem následně zapsal do **Tabulka 2** a **Tabulka 3**.



Obrázek 23: Blokové schéma programu pro měření vhodného času maximální rychlosti pro různé typy rovinek. Vytvořeno v programu diagrams.net.

4.8 Skenování dráhy

Prvním způsobem pro ověření naměřených hodnot a použitých postupů bylo vytvoření první verze. Provoz první verze nebyl zcela autonomní a bylo při ní potřeba vnějšího zásahu člověka. Tato verze spočívala ve dvou odlišných programech, podle kterých mikrokontroler pracoval. Úkolem prvního programu bylo naměření časů jednotlivých rovinek při měřící rychlosti vytvořené motorem s hodnotou PWM 65. Na základě těchto časů a **Tabulka 3** jsem každé rovince přiřadil příslušnou délku maximální rychlosti. Na autodráze jsem si určil oblast, kterou bylo možné z naměřených dat rozpoznat, (např. nejdelší rovinka). Tato oblast sloužila pro start modelu auta. Druhý program spočíval v akceleraci a brzdění modelu auta na rovinkách stejně

jako v měření 4.7.5. Tento program začínal nadcházející rovinkou za oblastí pro start modelu auta.

4.9 Měření počtu zatáček

Pro správnou funkci autonomně řízené verze systému je potřeba zjistit počet zatáček v jednom kole autodráhy. Jednou z možností je vytvoření orientačního bodu, který by systém pro ovládání modelu auta rozpoznal. Počet zatáček mezi projetím tímto bodem by se rovnal počtu zatáček jednoho kola autodráhy. Toto řešení by však vyžadovalo další periferii mikrokontroleru. Mým úkolem bylo vyřešení tohoto problému bez přítomnosti další periferie, tedy pouze z dat poskytovaných gyroskopem.

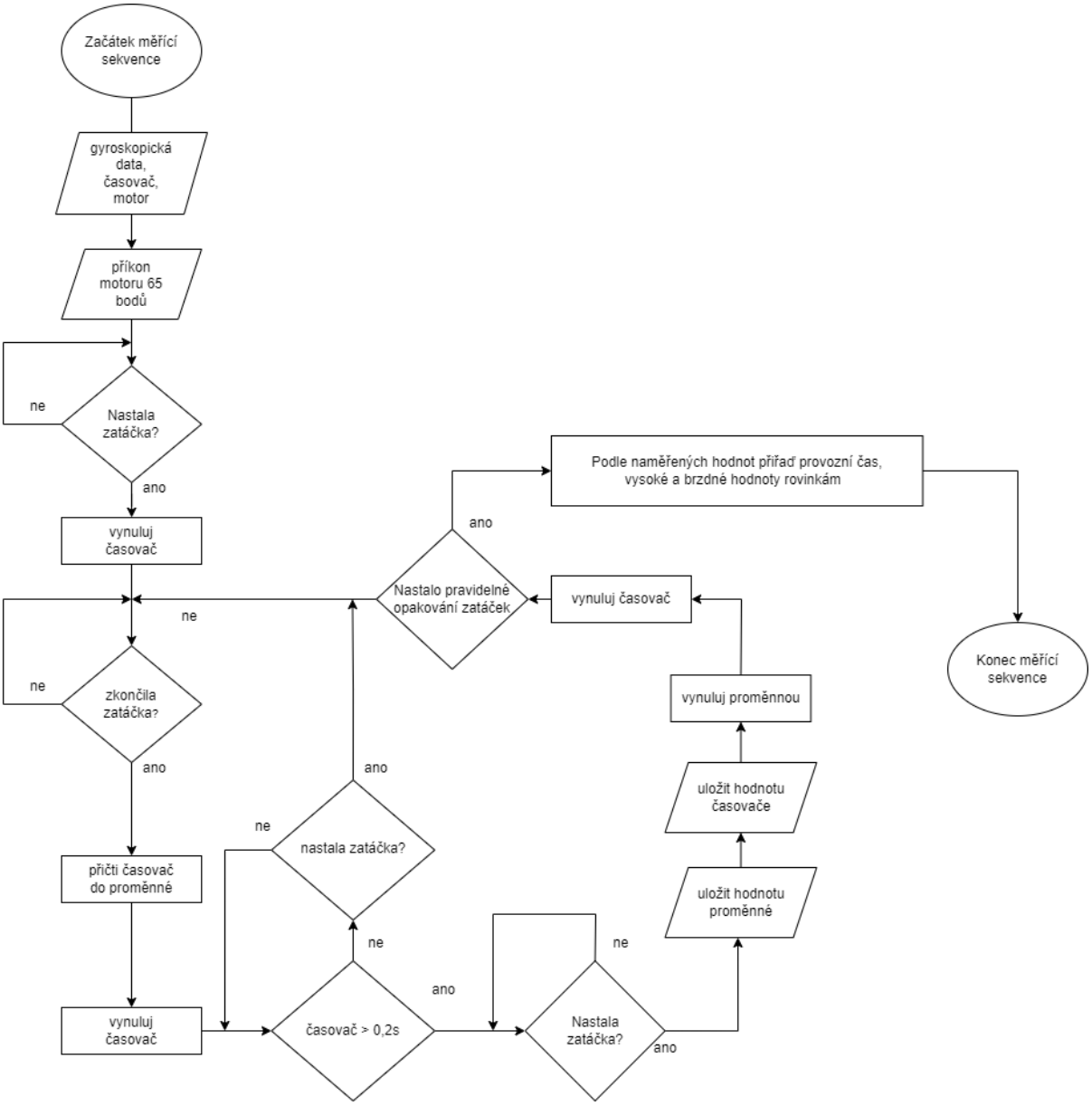
Můj jednoduchý algoritmus měření počtu zatáček spočíval v porovnávání naměřených délek zatáček sekvencí podmínek viz **Obrázek 24**. Vlivem vnějších činitelů nebyly časy stejných zatáček vždy stejné, a proto při porovnávání počítal mikrokontroler s maximální odchylkou 0,17s. Tento čas jsem určil na základě průměrného rozdílu mezi odlišnými zatáčkami. Celý program pro měření zatáček se skládal z několika částí. Každá část programu byla určena pro jeden počet zatáček na autodráze. Pokud mám daný počet zatáček na autodráze, mohu předpokládat, že se zatáčky začnou opakovat právě po tomto počtu zatáček. Pokud se budou tyto zatáčky shodovat, autodráha má právě daný počet zatáček. Pokud se zatáčky v tomto programu nebudou shodovat, budou se shodovat v jiném programu, který předpokládá jiný počet zatáček v autodráze. Počet jednotlivých sekvencí podmínek závisí na maximálním možném počtu zatáček v autodráze. V mém případě je maximální počet zatáček v autodráze roven sedmi, proto vždy v jednotlivých programech dochází k sedmi porovnáním zatáček. Toto omezení počtu zatáček je z důvodu nižšího výpočetního výkonu mikrokontroleru a nižšímu počtu použitých proměnných.

```
kladna_odchylka = 0.17;
zaporna_odchylka = -0.17;
sum = (z1 - z4);
if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
  sum = (z2 - z5);
  if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
    sum = (z3 - z6);
    if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
      sum = (z4 - z7);
      if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
        sum = (z5 - z8);
        if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
          sum = (z6 - z9);
          if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
            sum = (z7 - z10);
            if ((sum > zaporna_odchylka) && (sum < kladna_odchylka)){
              bluetooth.println("3 zatáčky");
            }
          }
        }
      }
    }
  }
}
```

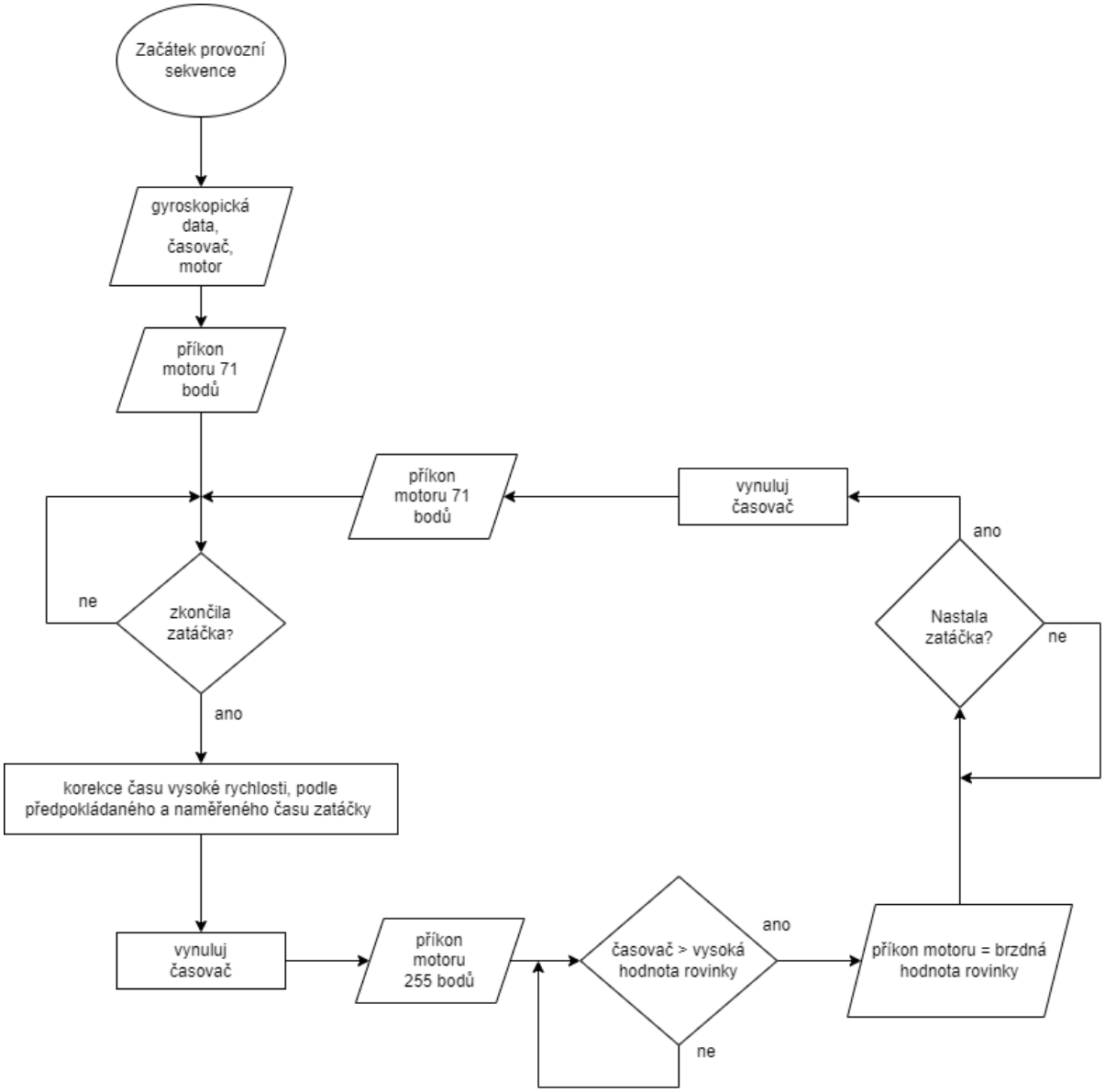
Obrázek 24: Ukázka části programu pro rozpoznání třech zatáček v jednom kol autodráhy. Vytvořeno v programu PlatformIO.

5 AUTONOMNĚ ŘÍZENÁ VERZE

Druhá finální verze tohoto projektu již obsahovala plně autonomní řízení. Po vložení modelu auta na autodráhu projede 7 zatáček kvůli rozjíždění a ustálení měřící rychlosti. Následně projede 7-17 zatáček (v závislosti na počtu zatáček jednoho kola autodráhy) měřící rychlostí. Podle naměřených dat přiřadí jednotlivým rovinkám data pro správnou akceleraci a brzdění. Po měřící sekvenci se program, ovládající model auta, přepne do provozní sekvence, ve které podle přiřazených dat navyšuje a snižuje příkon motoru. Nově naměřené časy rovinek porovnává s dříve určenými časy rovinek a na základě toho upravuje délky vysokých rychlostí rovinek. Princip programu měřící sekvence je znázorněn na blokovém schématu na **Obrázek 25**. Princip programu provozní sekvence je znázorněn na blokovém schématu na **Obrázek 26**. Dosažený čas jednoho kola by se měl blížit minimálnímu možnému času pro tento model auta při stejně sestavené autodráze.



Obrázek 25: Blokové schéma měřící sekvence autonomního programu pro model auta. Vytvořeno v programu diagrams.net.



Obrázek 26: Blokové schéma provozní sekvence autonomního programu pro model auta. Vytvořeno v programu diagrams.net.

6 ZÁVĚR

V rámci práce SOČ jsem dosáhl autonomně řízeného modelu auta na autodráze. Po vložení modelu na autodráhu projede několik kol kvůli naměření dat z autodráhy a naměření počtu zatáček v jednom kole. Na základě těchto dat určí vhodný příkon motoru pro každý úsek autodráhy, tak aby maximálně snížil čas projetí jednoho kola autodráhou. Následuje přepnutí programu do provozní sekvence, v níž bude ovládat motor na základě přiřazených příkonů pro jednotlivé úseky. Tímto dosáhne nejnižšího možného času jednoho kola při dané přilnavosti, výkonu a hmotnosti modelu auta. Program je schopný úpravy vysoké rychlosti při provozu. Pokud nastane změna podmínek provozu autodráhy (např. změna přilnavosti povrchu), program je schopný na danou změnu zareagovat a upravit čas maximální rychlosti.

Tento projekt by bylo možné použít jako jednoduchý příklad autonomního řízení, nebo i jako autonomního protihráče při závodění s modely aut na autodráze. Dále bych se chtěl věnovat porovnání algoritmu a klasického manuálního řízení modelu auta.

7 POUŽITÁ LITERATURA

[1] Arduino Pro Mini, 2022. In: *Arduino* [online]. 15. 12. 2022 [cit. 2023-02-10]. Dostupné z: https://docs.arduino.cc/retired/boards/arduino-pro-mini

[2] Arduino. *Arduino* [online]. [cit. 2023-02-26]. Dostupné z: https://www.arduino.cc/

[3] Arduino, 2001-. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2023-02-10]. Dostupné z: https://cs.wikipedia.org/wiki/Arduino

[4] Arduino Pro Mini Board Schematics 100% Explained, 2021. In: *Adduino* [online]. January 3, 2021 [cit. 2023-02-10]. Dostupné z: https://adduino.com/arduino-pro-mini-board-schematics-100-explained/

[5] ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash, 2015. *ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash* [online]. 01/15 2015, s. 294 [cit. 2023-02-10]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[6] Bluetooth modul HC-10 4.0 BLE klon. *Dratek.cz* [online]. 2016 [cit. 2023-02-10]. Dostupné z: https://dratek.cz/docs/produkty/0/567/1464644818.pdf

[7] Arduino gyroskop + akcelerometr. *Dratek.cz* [online]. [cit. 2023-02-10]. Dostupné z: https://dratek.cz/docs/produkty/0/134/1500635992.pdf

[8] What is a Voltage Regulator?. *Digi-Key* [online]. 7/1/2020 [cit. 2023-02-10]. Dostupné z: https://www.digikey.com/en/maker/blogs/2020/what-is-a-voltage-regulator

[9] *2 A positive voltage regulators* [online]. [cit. 2023-02-10]. Dostupné z: https://www.vpcentrum.eu/index.php?route=product/file&product_id=88515&file_id=2306032

[10] How Transistors Work – A Simple Explanation. *Buildelectroniccircuits* [online]. March 17, 2021 [cit. 2023-02-10]. Dostupné z: https://www.build-electronic-circuits.com/how-transistors-work/

[11] Power transistors: TIP31A. In: *STMicroelectronics* [online]. April 2006 [cit. 2023-02-26]. Dostupné z: https://www.st.com/resource/en/datasheet/tip31a.pdf

[12] Carrera 89200 EVO/DIG 132 Motor E200 Standard. *Modellbau-härtle* [online]. [cit. 2023-02-26]. Dostupné z: https://www.haertle.de/Autorennbahn/Slot+Ersatzteile/Carrera+89200+EVO+DIG+132+Motor+E200+Standard.html

[13] Diode: Definition, Symbol, and Types of Diodes. *Electrical4u* [online]. April 11, 2021 [cit. 2023-02-10]. Dostupné z: https://www.electrical4u.com/diode-working-principle-and-types-of-diode/

[14] SB220 ... SB2100. In: *VP centrum* [online]. 2018-02-01 [cit. 2023-02-26]. Dostupné z: https://www.vpcentrum.eu/index.php?route=product/file&product_id=291937&file_id=2540120

[15] Diode bridge, 2001-. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2023-02-10]. Dostupné z: https://en.wikipedia.org/wiki/Diode_bridge

[16] 2W005 – 2W10. In: *WTE power semiconductors* [online]. [cit. 2023-02-26]. Dostupné z: https://www.mikroshop.ch/pdf/2W10.pdf

[17] How Capacitors Work. *Howstuffworks* [online]. Aug 9, 2021 [cit. 2023-02-10]. Dostupné z: https://electronics.howstuffworks.com/capacitor.htm

[18] Capacitor Ceramics. *Sciencedirect* [online]. 2021 [cit. 2023-02-10]. Dostupné z: https://www.sciencedirect.com/topics/materials-science/capacitor-ceramics

[19] Vývojové PROSTŘEDÍ PLATFORMIO IDE. *HWKITCHEN* [online]. 25.8.2016 [cit. 2023-02-10]. Dostupné z: https://bastlirna.hwkitchen.cz/vyvojove-prostredi-platformio-ide/

[20] PlatformIO, 2001-. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2023-02-10]. Dostupné z: https://de.wikipedia.org/wiki/PlatformIO

[21] What is C++? Basic Concepts of C++ Programming Language. *Guru99* [online]. January 14, 2023 [cit. 2023-02-10]. Dostupné z: https://www.guru99.com/cpp-tutorial.html#2¨

[22] MARGOLIS, Michael, Brian JEPSON a Nicholas Robert WELDIN. *Arduino cookbook: recipes to begin, expand, and enhance your projects*. Third edition. Sebastopol: O'Reilly Media, [2020]. ISBN 978-1-4919-0352-0.

[23] Basics of UART Communication. *Electronics Hub* [online]. June 17, 2017 [cit. 2023-02-10]. Dostupné z: https://www.electronicshub.org/basics-uart-communication/

[24] UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. *AnalogDialogue* [online]. DEC 2020 [cit. 2023-02-10]. Dostupné z: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

[25] Basics of PWM (Pulse Width Modulation). *Arduino Documentation* [online]. [cit. 2023-02-10]. Dostupné z: https://docs.arduino.cc/learn/microcontrollers/analog-output

[26] 9. Pulzně šířková modulace, 2011. *Robot klub Rychnov* [online]. 2014-02-15 [cit. 2023-02-10]. Dostupné z: https://robotika.vosrk.cz/guide/arduino/lesson09/cs

[27] AnalogWrite(). *Arduino* [online]. [cit. 2023-02-10]. Dostupné z: https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

[28] BROWN, Robert. HOW TO CHANGE THE PWM FREQUENCY OF ARDUINO: EPIC GUIDE. In: *Nerdytechy* [online]. JUNE 11, 2021 [cit. 2023-02-26]. Dostupné z: https://nerdytechy.com/how-to-change-the-pwm-frequency-of-arduino/

[29] Using Arduino Interrupts – Hardware, Pin Change and Timer. *DroneBot Workshop* [online]. [cit. 2023-02-10]. Dostupné z: https://dronebotworkshop.com/interrupts/

[30] Signal Filtering in Python. *SWHarden.com* [online]. September 23rd, 2020 [cit. 2023-02-20]. Dostupné z: https://swharden.com/blog/2020-09-23-signal-filtering-in-python/

[31] Arduino-slotcar. In: *GitHub* [online]. Dec 22, 2022 [cit. 2023-02-21]. Dostupné z: https://github.com/tomas-fryza/arduino-slotcar/blob/master/data/speed_70.png