Středoškolská technika 2018

Setkání a prezentace prací středoškolských studentů na ČVUT

Mobilní robot ARDUINO II

Jakub Vobořil

VOŠ, SPŠ a JŠ Kutná Hora
Masarykova 197

VYŠŠÍ ODBORNÁ ŠKOLA, STŘEDNÍ PRŮMYSLOVÁ ŠKOLA
A JAZYKOVÁ ŠKOLA S PRÁVEM STÁTNÍ JAZYKOVÉ ZKOUŠKY

PRAKTICKÁ MATURITA

Mobilní robot ARDUINO II

2018 **Jakub Vobořil**

Zadání práce

Čestné prohlášení:

Prohlašuji, že jsem praktickou maturitní práci vypracoval samostatně a použil jsem pouze literatury uvedené v soupisu.

Souhlasím s trvalým umístěním mé práce ve školní knihovně, kde bude k dispozici pro potřeby školy.

V Kutné Hoře dne ....

podpis

Anotace

Úkolem maturitní práce je vytvoření řídící jednotky modelu RC auta. Práce se zaměřuje na ovládání pomocí mikroprocesoru a vývoj aplikace. Přenos informací z mobilní aplikace do Arduina je řešen pomocí Bluetooth modulu, ten dále řídí H můstek, který ovládá motor. Součástí práce je měření rychlosti pomocí IR senzoru a náklonu do boku pomocí akcelerometru.

Klíčová slova

RC auto, mikroprocesorové řízení, Arduino, ATmega328, Wiring, Bluetooth modul HC-05, měření rychlosti, IR senzor, měření náklonu, akcelerometr MPU6050

Abstract

The task of the maturity project is create microprocessor-control unit for RC car model. Project focuses on controlling using microprocessor and mobile application. Transfer of informations to the Arduino is solved by Bluetooth module and it controlls h-bridge, which is driving the motor. Part of the project is speed measurement by IR sensor and tilt measurement by accelerometer.,

Keywords

RC car, microprocessor control, Arduino, ATmega328, Wiring, Bluetooth module HC-05, speed mesasurement, IR sensor, tilt measurement, accelerometer MPU6050

Obsah

Čestné prohlášení:....3

Anotace....4

Klíčová slova....4

Keywords....4

Obsah....5

1.Teoretický rozbor prvků práce a jejich aplikace v práci....6

1.1.Struktura....6

1.2.I2C sběrnice....6

1.3.Bluetooth....7

1.4.Řízení....7

1.5.Měření rychlosti....9

1.6.Zapojení LED světel....10

1.7.Ovládání servomotoru....10

1.8.Ovládání H-můstku....11

1.9.Baterie....11

2.2.Komunikace s aplikací....12

2.3.Ovládání modelu....12

2.4.Vytvoření mobilní aplikace pro OS Android....12

3.2.Realizace konstrukce....14

....15

....15

Obrázek 11 – osazovací plán 1:1....15

4.Seznam použité literatury a on-line zdrojů....15

Čestné prohlášení:....3

Anotace....4

Klíčová slova....4
Keywords....4
Obsah....5
1.Teoretický rozbor prvků práce a jejich aplikace v práci....7
1.1.Struktura....7
1.2.I2C sběrnice....7
1.3.Bluetooth....8
1.4.Řízení....8
1.5.Měření rychlosti....10
1.6.Zapojení LED světel....11
1.7.Ovládání servomotoru....11
1.8.Ovládání H-můstku....12
1.9.Baterie....12
2.2.Komunikace s aplikací....13
2.3.Ovládání modelu....13
2.4.Vytvoření mobilní aplikace pro OS Android....13
3.2.Realizace konstrukce....15
....16
....16
Obrázek 11 – osazovací plán 1:1....16
4.Seznam použité literatury a on-line zdrojů....16

1. Teoretický rozbor prvků práce a jejich aplikace v práci

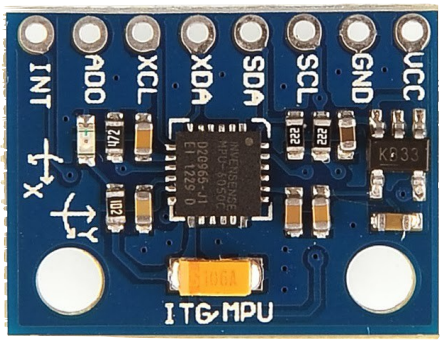
1.1. Struktura

Práce se skládá z osmi základních bloků. Řídící blok získává příkazy z mobilní aplikace pomocí Bluetooth modulu a pomocí H-můstku ovládá motor a servo řízení. Dále komunikuje pomocí I2C s akcelerometrem a získává data o náklonu a přijímá logické úrovně generované IR senzorem pro výpočet rychlosti.

Obrázek 1 – Blokové schéma

1.2. I²C sběrnice

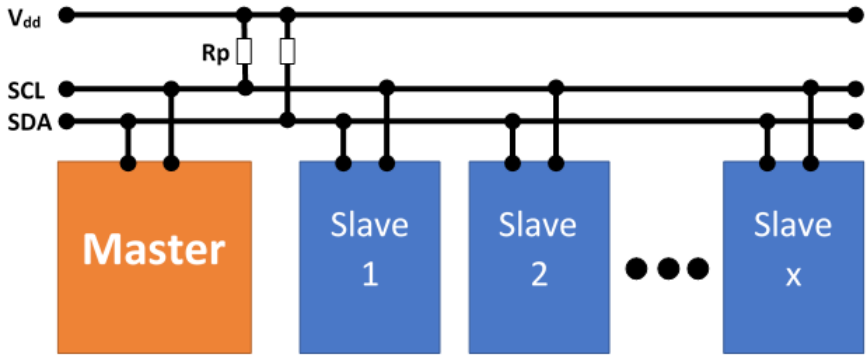
nebo také IIC je sériová sběrnice. Tato sběrnice má čtyři vodiče – dva pro napájení, jeden pro hodinový signál a druhý pro data. Na tuto sběrnici se může připojit až 128 zařízení. Arduino pak může pracovat v master módu a bude řídit komunikaci ve sběrnici a generovat hodinový signál. Ostatní zařízení, která pracují ve slave módu, pracují jen na příkaz mastera. V tomto případě je tedy Arduino master a akcelerometr MPU6050



Obrázek 1-MPU6050

slave. Proto spuštění této sběrnice, vysílání příkazů i přijímání dat po sběrnici je realizováno v programu Arduina a MPU6050 pouze odpovídá na příkazy.

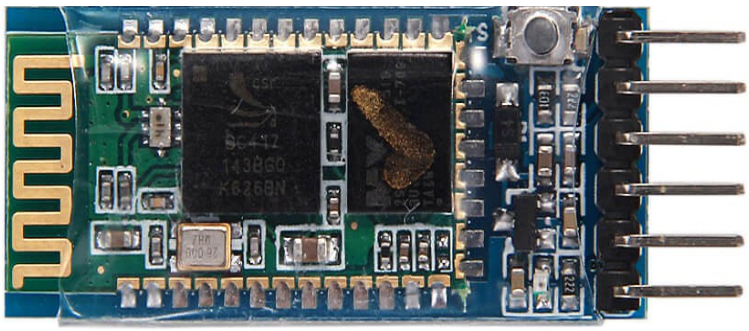
MPU6050 je akcelerometr a gyroskop v jednom pouzdře. Také obsahuje DMP (Digital Motion Processor), který nám velice zjednoduší práci s daty, protože v reálném čase sbírá data ze zmíněných dvou zařízení a poskytuje nám přímo tři důležité hodnoty, které udávají rotace kolem jednotlivých os.



Obrázek 2 – I²C komunikace

1.3. Bluetooth

Bluetooth je otevřený standard pro bezdrátovou komunikaci mezi zařízeními. Modul Bluetooth HC-05 komunikuje bezdrátově s mobilem a s Arduinem pomocí rozhraní UART (RX,TX). Tímto způsobem přenáší informace z aplikace do řídící jednotky a vysílá zpět do aplikace informace o náklonu a rychlosti. Modul HC-05 jsem zvolil z důvodu další využitelnosti, protože obsahuje oba módy - master i slave.



Obrázek 3 – HC-05

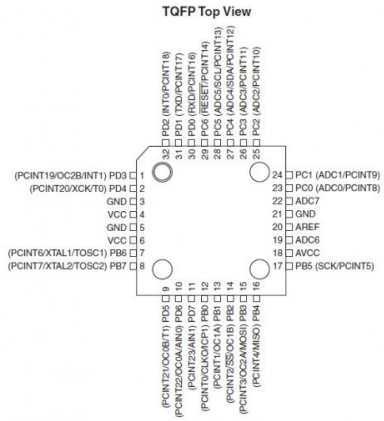
1.4. Řízení

Řídící část ovládá celý systém a zřizuje komunikaci mezi jednotlivými prvky systému. Obvykle k řízení se využívají programovatelná zařízení. Tím může být pro průmyslové projekty programovatelný automat nebo pro menší projekty, na které nejsou kladeny vysoké nároky, postačí i mikrokontrolér. Mikrokontroléry neboli programovatelné mikropočítače vychází z principu programovatelných logických obvodů. Základním nutným prvkem je procesor, vstupní a výstupní rozhraní, operační paměť, paměť programu a oscilátor. Funkce, které jsou potřeba pro řízení této práce, mikrokontrolér splňuje a navíc oproti programovatelnému automatu je lepší z hlediska ceny a rozměru. Pro řízení modelu

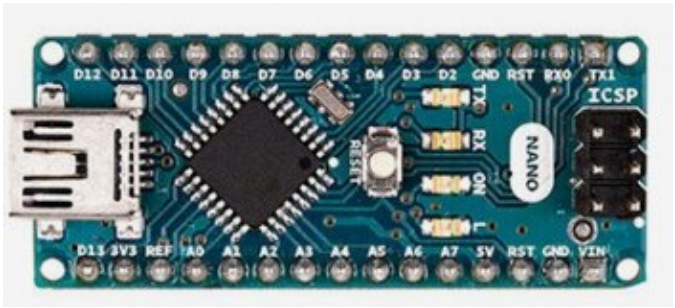
RC auta jsem zvolil mikrokontrolér ATmega328, s kterým jsme již byli seznámeni v hodinách Mikroprocesorové techniky.

Zvolil jsem ATmega328 v provedení SMD z důvodu malých rozměrů a přítomnosti programátoru přímo na desce Arduino Nano. To jsem použil z důvodu jeho malých rozměrů, a tím jeho lepší zastavění do řídící jednotky. Projekt lze rozšířit o řadu dalších senzorů například senzor kouře, nebo doplněním ultrazvukového senzoru jej učinit samořiditelným.

Napájení řídící části je na pinu a GND. Zde je připojena externí baterie. Na tomto vstupu může být napětí v rozmezí od 7 do 12 voltů, limit je až 20 voltů. Toto napětí se pak upravuje stabilizátorem na 5V. Tento stabilizátor se stará také o napájení logiky H-můstku, bluetooth modulu, IR senzoru a akcelerometru. Na pinech D0(RX), D1(TX) a D3 je připojen bluetooth modul, na pinech D4 a D5 je připojen H-můstek, na pinu 6 je připojeno ovládání servomotoru a na pinu D7 je připojen IR senzor. Na analogových pinech A0 a A1 jsou připojeny báze tranzistorů, které spínají LED světla. Na analogové piny Arduina A4 a A5 je připojen datový a hodinový vodič sběrnice pro komunikaci s modulem MPU6050 , na které jsou přivedeny pull-up rezistory.



Obrázek 3 – Schéma ATmega328P v provedení SMD



Obrázek 4 – Arduino Nano

Pro tento projekt by stačil i samotný mikrokontrolér s krystalem. Od této možnosti jsem upustil, protože nemám dostatečné zkušenosti s pájením SMD součástek. Arduino Nano jsem použil, protože jsem ho dostal při odborné stáži v Německu a veškeré testování a programování jsem dělal na něm. Arduino Nano se vyrábí nejen s ATmega328, ale také s ATmega168. Použití ATmega168 by stačilo pouze na řízení RC auta, jelikož velikost programu přesahuje jeho velikost paměti programu 16KB, využil jsem ATmega328P, která

má paměť 32KB a je zde dostatek místa i pro další možná rozšíření. Arduino Nano jsem umístil do patice, protože je škoda nenajít mu případné další využití.

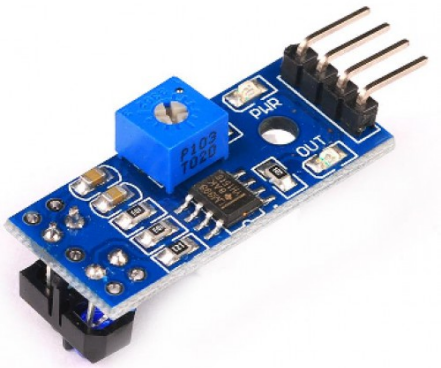
Parametry ATmega328P:

Parametr	**Hodnota**
Typ paměti programu	Flash
Velikost paměti programu	32 KB
Rychlost CPU	20 (MIPS)
Byty RAM	2,048
Data EEPROM	1024 (B)
Digitální komunikační rozhraní	1x UART,2x SPI,1x I2C
Časovače	2x 8-bit, 1x 16-bit
Komparátory	1
Teplotní rozsah	-40 do 85 (°C)
Rozsah napětí	1,8 do 5,5 (V)
Počet vývodů	32

Tabulka 1 – Parametry ATmega328P

1.5. Měření rychlosti

Měření rychlosti se provádí pomocí IR senzoru, který snímá odrazy světla od hliníkové folie připevněné na jedno zadní kolo. Počet otáček vynásobíme obvodem kola a získáme ujetou dráhu. V programu se načítá čas, za který kolo udělá jednu obrátku. S pomocí toho vypočítáme rychlost [m/s], kterou poté odesíláme do aplikace telefonu.



Obrázek 5 – IR senzor

1.6. Zapojení LED světel

Protože výstup mikrokontroléru je logický, tak může nabývat napětí přibližně 0V nebo 5V. Výstupy jsou připojeny na báze tranzistorů přes rezistory, které zde musí být, aby omezovaly proud do báze, a tím zabránily zničení tranzistorů. LED diody také nemůžou být připojené přímo na 5 voltů, protože jsou konstruované na menší napětí, proto vzniká přebytek energie a je nutno do zapojení přidat předřadný rezistor. Na předřadném rezistoru bude zbývající napětí, taktéž nám sníží protékající proud, který by měl maximálně dosahovat hodnoty maximálního proudu napsaného v datovém listu u daného typu LED diody. Elektrický odpor předřadného rezistoru pro každý druh diody v práci se vypočítá přes následující vztah

1.7. Ovládání servomotoru

Servo motory slouží pro nastavení určité polohy ovládaného mechanismu a následné držení v této poloze. Stejnosměrné servo motory se využívají například pro ovládání robotické paže nebo pro nastavení kormidla u leteckých modelů. Jejich hlavní výhodou je malý rozměr a malá hmotnost s relativně velkou silou.

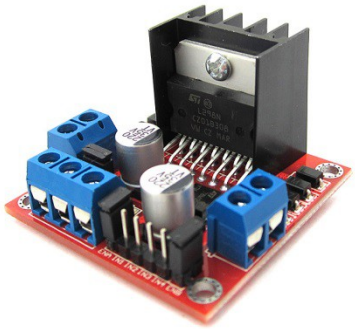
Obrázek 6 – Servo MG996R

Tyto motory obvykle neumožňují otáčení neustále dokola, ale udržují nastavený úhel natočení. Úhel se pohybuje nejčastěji v rozsahu 0° až 180°. Nastavení tohoto úhlu se provádí zasláním PWM impulsu o určité délce. Neutrální poloha (90°) odpovídá obvykle délce impulsu 1,5ms. Délka 0,5ms odpovídá úhlu 0° a impuls délky 2,5ms nastavuje úhel 180°. Impulsy se posílají motoru pravidelně každých 20ms.

1.8. Ovládání H-můstku

H-můstek je elektrický obvod sestavený z tranzistorů určený pro řízení stejnosměrného motoru. Obvykle se prodávají integrované obvody či moduly, které obsahují dva H můstky. Tyto obvody je možné použít pro řízení jednoho krokového motoru, nebo dvou stejnosměrných motorů (lze řídit rychlost i směr otáčení). H-můstek obsahuje TTL logiku, která při určité kombinaci (viz Pravdivostní tabulka) rozhoduje, jestli se má motor otáčet dopředu či dozadu. Chtěl jsem použit H-můstek postavený z MOSFET tranzistorů, bohužel se mi ho nepovedlo oživit a tak jsem vybral modul L298N, jehož maximální proud je 2A.

ENA	INT1	INT2	popis
0	x	x	Motor A stojí
1	0	0	Motor se vypne a zastaví
1	0	1	Motor se otáčí dopředu
1	1	0	Motor se otáčí dozadu
1	1	1	Motor se vypne a zastaví



Obrázek 7 – L298N

Tabulka 2 – Pravdivostní tabulka L298N

1.9. Baterie

Na napájení a ovládání motoru a servomotoru jsem použil čtyři baterie AA, každá o napětí 1,5 voltu a kapacitě 2400mAh. Pro napájení Arduina jsem použil baterie 1604 9V.

2. Programová konstrukce práce

2.1. Struktura

- Načtení knihovny

- Konfigurace adresy MPU6050
- Konfigurace pinů
- Deklarace proměnných
- Nastavení pinu servomotoru
- Spuštění sériové komunikace na rychlosti 9600 baud/s
- Spuštění komunikace
- Konfigurace vstupů a výstupů
- Vykonávání příkazů pro řízení modelu
- Funkce pro komunikaci s MPU6050
- Přepočítání hodnot na úhel
- Výpočet rychlosti
- Odesílání hodnot do aplikace

2.2. Komunikace s aplikací

Modul HC-05 komunikuje pomocí sériové komunikace s Arduinem a to pomocí funkce Serial.read() uloží danou hodnotu do proměnné, která se poté testuje. Odesílání hodnot ze senzorů do aplikace se provádí pomocí funkce Serial.print a Serial.println pro zakončení řádku. Poté se vypíší hodnoty v aplikaci.

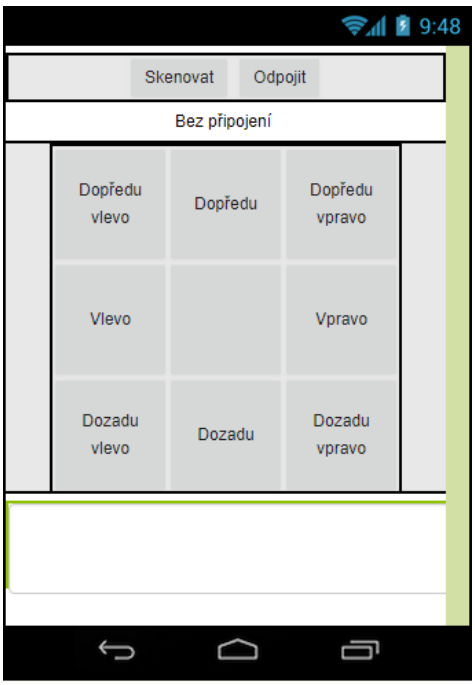
2.3. Ovládání modelu

Ovládání se provádí testováním hodnoty proměnné poslané z aplikace pomocí funkce if a else if. Pokud se hodnota proměnné rovná podmínce vykonají se dané příkazy pro směr jízdy a natočení serva.

2.4. Vytvoření mobilní aplikace pro OS Android

Pro vytvoření mobilní aplikace jsem se rozhodl, protože mi žádná z dostupných veřejně dostupných aplikací nevyhovovala z důvodu absence okna pro výpis dat ze senzorů. Aplikaci jsem vytvářel v programu MIT App Inventor. Ač jsem mobilní aplikaci nikdy nevytvářel a nikdy neprogramoval v jazyce Java, s pomocí několika tutoriálů jsem se velice rychle naučil, jak vývojářský program ovládat.

Nejprve se založil nový projekt a pojmenoval jej. Poté se otevře nové okno, kde jsem vybíral a rozmisťoval objekty, ty se dělí na viditelné a neviditelné. Poté jsem vyzkoušel rozmístění na našem telefonu pomocí aplikace MIT App Companion, která po spárování s programem vykreslila můj návrh v mobilu. Poté co jsem jej uspořádal objekty podle mých představ, přepnul jsem prostředí do módu programování. Zde jsem vybíral z několika skupin bloků - kontrolních, logických, matematických, textových a dalších. Také zde jsou funkce objektů, které jsem si vybral v grafické části. Program se sestavuje pomocí bloků dohromady. Po sestavení jsem aplikaci otestoval ji v MIT App Companion a poté vyexportoval do formátu .apk. Pomocí vygenerovaného QR kódu otevřeme odkaz, ze kterého se nám stáhne a nainstaluje naše vytvořená aplikace do mobilu.



Obrázek 8 – Grafická část aplikace

3. Mechanická konstrukce práce

3.1. Návrh konstrukce

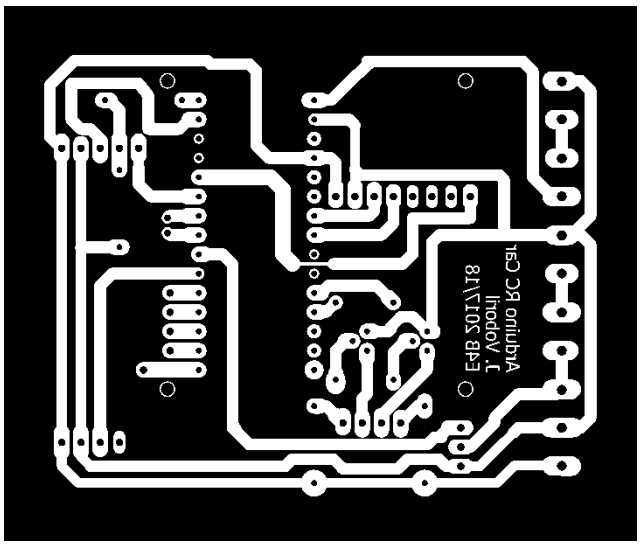
Plošný spoj je navržen tak, aby byl především vešel do místa původní řídící jednotky. Obsahuje všechny moduly kromě infračerveného senzoru, který je umístěný na karoserii auta. Dané moduly jsou rozmístěny kolem mikrokontroléru, tak aby byly co nejblíže propojeny. Z nedostatku místa jsem musel umístit modul H-můstku nad plošný

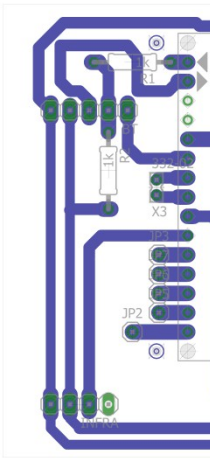
spoj. Plošný spoj také obsahuje spínací tranzistory pro rozsvěcení světel. V návrhu jsem přidal k některým pinům plošku, do které není nic připojeno.

Obrázek 9 – Schéma řídící jednotky

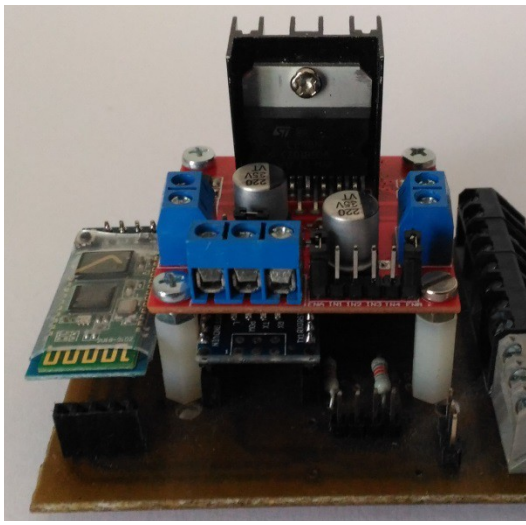
3.2. Realizace konstrukce

Tvorba plošného spoje začíná vytvořením nebo výběrem schématu. Schéma i plošný spoj se dnes navrhuje převážně v CAD programech na počítači. Po sestavení schématu se vytvoří návrh plošného spoje, na který se rozmístí součástky a pospojováním vývodů se vytvoří cesty. Plošný spoj se poté vytiskne na pauzovací papír v negativu jako je například obrázek 9. Cuprextitová deska se očistí speciální gumou a tím se zbaví případných nečistot. Když je deska vyčištěna nanese se na ni fólie citlivá na UV složku světla a přilepí se s ní na ohebnou podložku, která se nemůže roztavit nažehlením. Poté se deska odřízne z podložky a vloží se do UV osvitové jednotky, kde se překryje sklem a přiloží se pauzovací papír, přes který se poté plošný spoj osvítí. Plošný spoj vložíme do roztoku $NaHCO_3$ kde se nechá po dobu pěti minut. Následně desku ponoříme do chloridu železitého a části, které byli osvíceny zůstávají a neosvícené části se postupně odleptávají. Poté plošný spoj vyvrtáme a necháme rozpustit zbytek UV folie v hydroxidu sodném a opláchneme vodou. Nyní stačí nastříkat stranu plošného spoje pájitelným lakem, to je z důvodu, aby nám spoj nezoxidoval a lépe se nám na něj chytala pájka. Po zaschnutí laku osazujeme součástky. Nejprve se osazují nízké součástky, poté postupujeme podle velikosti. Vývody součástek se poté zaletují pájecím perem ohřátým přibližně na 350 stupňů Celsia a pájkou-cínem. Pro všechny moduly jsem použil konektory, pro jejich možné vyjmutí a použití v jiných projektech.





Obrázek 10 – Návrh plošného spoje 1:1 v negativu



Obrázek 12 – osazená řídící jednotka s H-

4. Seznam použité literatury a on-line zdrojů

I2C sběrnice

https://arduino.cz/propojujeme-arduino-s-jinym-i-zarizenimi/

Mikrokontrolér

https://cs.wikipedia.org/wiki/Jedno%C4%8Dipov%C3%BD_po%C4%8D%C3%ADta%C4%8D

Obrázek 2 – HC-05

https://www.geekbuying.com/item/UART-HC-05-RS232-TTL-Wireless-Bluetooth-Transceiver-Module-For-Arduino-Raspberry-Pi-344574.html

Obrázek 3 – Schéma ATmega328 SMD

https://roboindia.com/store/atmega-328-smd

Obrázek 4 - Arduino Nano

https://store.arduino.cc/usa/arduino-nano

Tabulka 1 - Parametry ATmega328

https://www.microchip.com/wwwproducts/en/ATmega328

Obrázek 5 – IR senzor

http://www.bitstoc.com/index.php?route=product/product&product_id=231

Tabulka 2 – L298N pravdivostní tabulka

https://www.raspberrypi.org/forums/viewtopic.php?t=177188

Obrázek 6 – L298N

https://www.geekbuying.com/item/L298N-H-Bridge-Motor-Driver-Module-5-25V-2A-For-Arduino-AVR-STM32-344410.html

5. Přílohy

5.1. Program Arduino RC Car

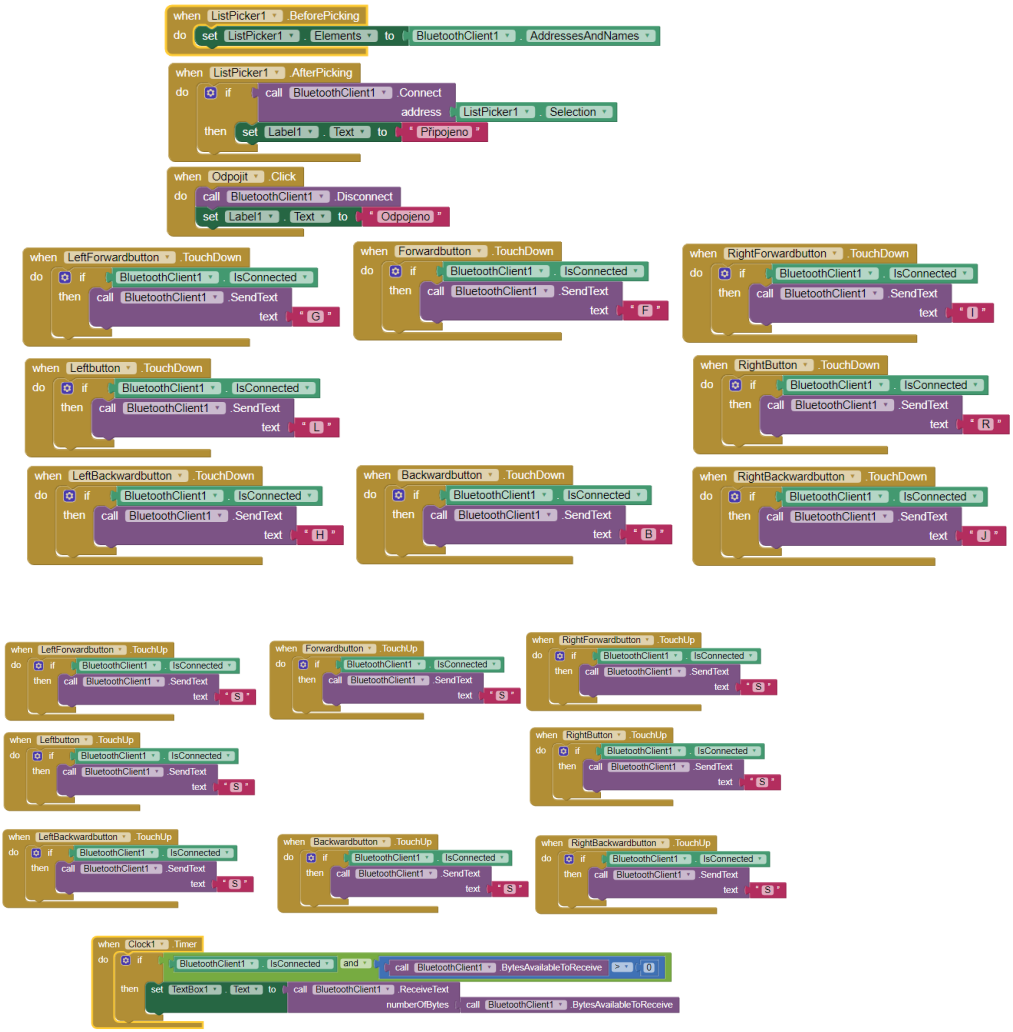
```
#include <Servo.h>
#include <Arduino.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
MPU6050 mpu;
#define LED_PIN 13 // číslo pinu s integrovanou LED diodou

bool dmpReady = false;  // když je DPM připraveno, obsahuje true
uint8_t mpuIntStatus;   // stav externího přerušení z DPM
uint8_t devStatus;      // stav poslední operace
uint16_t packetSize;    // velikost paketu z DPM
uint16_t fifoCount;     // počet bytů ve FIFO zásobníku
uint8_t fifoBuffer[64]; // FIFO zásobník

// proměnné důležité pro výpočet
Quaternion q;           // [w, x, y, z]         kvaternion
VectorFloat gravity;    // [x, y, z]            vektor setrvačnosti
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll úhly

// ================================================================
// ===               INTERRUPT DETECTION ROUTINE                ===
// ================================================================

volatile bool mpuInterrupt = false;     // true pokud DMP vyvolalo přerušení
void dmpDataReady() {
mpuInterrupt = true;
}

 Servo rizeni;
  const int motorA1  = 4;
  const int motorA2  = 5;
  const int BTState = 7;
//Useful Variables
  int i=0;
  int j=0;
  int state;

void setup() {
  rizeni.attach(6);
    // Set pins as outputs:
    pinMode(motorA1, OUTPUT);
    pinMode(motorA2, OUTPUT);
    pinMode(BTState, INPUT);
```

```
    pinMode(LED_PIN, OUTPUT);

// připojíme se na I2C sběrnici
Wire.begin();

// inicializujeme UART
Serial.begin(9600);
while (!Serial); // wait for Leonardo enumeration, others continue immediately

// inicializujeme MPU-6050
Serial.println(F("Initializing I2C devices..."));
mpu.initialize();

// ověříme připojení k MPU-6050
Serial.println(F("Testing device connections..."));
Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));

// incializujeme DMP
Serial.println(F("Initializing DMP..."));
devStatus = mpu.dmpInitialize();

// ujistíme se, že funguje
if (devStatus == 0) {
    // zapneme DMP
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // externí přerušení Arduina nabindujeme na funkci dmpDataReady
    Serial.println(F("Enabling interrupt detection (Arduino external interrupt 0)..."));
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    Serial.println(F("DMP ready! Waiting for first interrupt..."));
    dmpReady = true;

    // zjistíme si, jak velké pakety DMP vrací
    packetSize = mpu.dmpGetFIFOPacketSize();
} else {
    // Když dojde k chybě, tak:
    // 1 = selhala úvodní komunikace s DMP
    // 2 = selhala aktualizace nastavení DMP
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));



}

digitalWrite(LED_PIN, LOW);
}


void loop() {
  //Stop car when connection lost or bluetooth disconnected
     if(digitalRead(BTState)==LOW) { state='S'; }
     if(Serial.available() > 0){
      state = Serial.read();
    }

  /*********************Forward***************************/
  //If state is equal with letter 'F', car will go forward!
    if(state== 'F'){
     digitalWrite(A0, 1);digitalWrite(A1, 0);
      digitalWrite(motorA1, 1); digitalWrite(motorA2, 0);
      rizeni.write(90);
    }
  /*********************Forward Left**********************/
  //If state is equal with letter 'G', car will go forward left
    else if (state == 'G') {
     digitalWrite(A0, 1);digitalWrite(A1, 0);
      digitalWrite(motorA1, 1); digitalWrite(motorA2, 0);
        rizeni.write(105);
    }
  /*********************Forward Right*********************/
  //If state is equal with letter 'I', car will go forward right
    else if (state == 'I') {
    digitalWrite(A0, 1);digitalWrite(A1, 0);
        digitalWrite(motorA1, 1); digitalWrite(motorA2, 0);
       rizeni.write(75);
    }
  /*********************Backward**************************/
  //If state is equal with letter 'B', car will go backward
    else if (state == 'B') {
      digitalWrite(A0, 0);digitalWrite(A1, 1);
      digitalWrite(motorA1, 0);   digitalWrite(motorA2, 1);
        rizeni.write(90);
    }
  /*********************Backward Left*********************/
  //If state is equal with letter 'H', car will go backward left
    else if (state == 'H') {
       digitalWrite(A0, 0);digitalWrite(A1, 1);
```

```
      digitalWrite(motorA1, 0);   digitalWrite(motorA2, 1);
        rizeni.write(105);
    }
  /**********************Backward Right**********************/
  //If state is equal with letter 'J', car will go backward right
    else if (state == 'J') {
       digitalWrite(A0, 0);digitalWrite(A1, 1);
      digitalWrite(motorA1, 0);   digitalWrite(motorA2, 1);
        rizeni.write(75);
    }
  /**************************Left**************************/
  //If state is equal with letter 'L', wheels will turn left
    else if (state == 'L') {
      digitalWrite(motorA1, 0);   digitalWrite(motorA2, 0);
        rizeni.write(105);
    }
  /**************************Right**************************/
  //If state is equal with letter 'R', wheels will turn right
    else if (state == 'R') {
      digitalWrite(motorA1, 0);   digitalWrite(motorA2, 0);
        rizeni.write(75);
    }

  //If state is equal with letter 'S', stop the car
    else if (state == 'S'){
        digitalWrite(motorA1, 0);  digitalWrite(motorA2, 0);
       }
// pokud není DMP připravené, nebudeme dělat nic
if (!dmpReady) return;

// zde provádíme náš kód, cyklus ověřuje, zda nemá DMP připravena nějaká data
while (!mpuInterrupt && fifoCount < packetSize) {
    //
}

// resetujeme proměnnou informující o přerušení vyvolané z DMP a získáme INT_STATUS byte
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();

// získáme velikost FIFO zásobníku
fifoCount = mpu.getFIFOCount();

// zjistíme, zda nedošlo k přetečené zásobníku
// pokud k němu dojde, je třeba optimalizovat kód v cyklu výše,
// případně přerušit provádění mezi delšími výpočty, je-li to třeba
```

```
// případně přerušit provádění mezi delšími výpočty, je-li to třeba
if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
    // vyčistíme zásobník
    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));

// pokud je vše v pořádku, zpracujeme data z DMP
} else if (mpuIntStatus & 0x02) {
    // čekání na správnou délku dat
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();

    // přečteme paket ze zásobníku
    mpu.getFIFOBytes(fifoBuffer, packetSize);

    // pokud je v zásobníku víc než jeden paket, tímto zajistíme, že bude přečten v dalším cyklu
    fifoCount -= packetSize;

    // naplnění proměnných s vypočitanými hodnotami
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    //Výpis YAW/PITCH/ROLL
   // Serial.print("ypr\t");
  //  Serial.print(ypr[0] * 180/M_PI);
   // Serial.print("\t");
  //  Serial.print(ypr[1] * 180/M_PI);
  //  Serial.print("\t");
    Serial.println(ypr[2] * 180/M_PI);
    if ((ypr[2] * 180 / M_PI) > 30 or (ypr[2] * 180 / M_PI) < -30)Serial.println("Pozor:Velky naklon!");
}



    }
```

5.2. **Program Aplikace Arduino RC car – Android**


when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
when ListPicker1 .AfterPicking
do if call BluetoothClient1 .Connect
address ListPicker1 . Selection
then set Label1 . Text to " Připojeno "
when Odpojit .Click
do call BluetoothClient1 .Disconnect
set Label1 . Text to " Odpojeno "
when LeftForwardbutton .TouchDown
when Forwardbutton .TouchDown
when RightForwardbutton .TouchDown
when Leftbutton .TouchDown
when RightButton .TouchDown
when LeftBackwardbutton .TouchDown
when Backwardbutton .TouchDown
when RightBackwardbutton .TouchDown
when LeftForwardbutton .TouchUp
when Forwardbutton .TouchUp
when RightForwardbutton .TouchUp
when Leftbutton .TouchUp
when RightButton .TouchUp
when LeftBackwardbutton .TouchUp
when Backwardbutton .TouchUp
when RightBackwardbutton .TouchUp
when Clock1 .Timer
BluetoothClient1 . IsConnected and call BluetoothClient1 .BytesAvailableToReceive > 0
set TextBox1 . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive