Střední odborná škola a Střední odborné učiliště Lanškroun
Kollárova 445

Praktická maturitní práce

Téma číslo 9

Programování LEGO Mindstorms

Jméno a příjmení	**Jakub Pecháček**
Obor	**26-41-L/01 Mechanik elektronik**
Školní rok	**2012/13**
Datum odevzdání	**29. 3. 2012**

Zadání praktické maturitní práce

Žák: **Pecháček Jakub**

Třída, obor: **4. ME –Mechanik elektrotechnik**

Název práce: **Programování LEGO Mindstorms**
Anotace:

> **Vytvoř program pro robot LEGO Mindstorms s využitím senzorů pro identifikace v prostoru - labyrintu a s možností vyhledání hořící svíčky a její sfouknutí**

Vedoucí praktické maturitní práce: **Bc. Lédl Milan**

Datum zadání praktické maturitní práce: **24. 10. 2012**

Termín odevzdání praktické maturitní práce: **29. 3. 2013**

Školní rok: 2012/2013

„Prohlašuji, že tuto praktickou maturitní práci jsem vypracoval samostatně s využitím informací, které jsou uvedeny v seznamu použité literatury“

V Lanškrouně

Obsah

1. Úvod

Zadání mé práce bylo vytvořit robota schopného lokalizovat a zneškodnit oheň v podobě svíčky. Aby to nebylo tak jednoduché, robot se musí pohybovat v poli ohraničeném černou páskou a musí dbát na to, aby se žádné ze 4 čtyř předem neurčených překážek(zdí), umístěných v poli, pokud možno nedotkl. Herní pole je označeno černou páskou, která je pouze informační a robot jí může přejet a vyjet tak z hřiště.

2. Parametry

2.1. Desky

2.1.1. Raspberry Pi

- procesor ARM1176JZF-S z rodiny ARM11 taktovaný na 700 Mhz
- grafický procesor VideoCore IV, podporující OpenGL ES 2.0, 1080p30, MPEG-4
- 256 MB (v novějších 512 MB) RAM sdílených s grafickou kartou
- jeden až dva USB porty
- Obrazový výstup Composite RCA, HDMI, DSI
- Zvukový výstup přes 3,5 mm konektor, HDMI
- slot pro SD nebo MMC kartu
- ethernetový adaptér 10/100 s konektorem RJ45
- 12×GPIO, UART, I²C, sběrnici SPI
- Dva montážní otvory
- podpora JTAG Debug

2.1.2. STM32F4-Discovery

- STM32F407VGT6 microcontroller - 32-bit ARM Cortex-M4F jádro, 1 MB Flash, 192 KB RAM
- On-board ST-LINK/V2
- Napájení: Z USB sběrnice nebo z externího 5 V zdroje
- LIS302DL, ST pohybový senzor, 3-osý digitální akcelerometr
- MP45DT02, ST zvukový senzor
- CS43L22, zvukový D/A převodník
- 8x LED: 4 uživatelské a 4 informační
- Dvě tlačítka (jedno uživatelské a druhé jako reset)

- 3×12-bit, 2.4 MSPS A/D převodníky až s 24 kanály
- 2×12-bit D/A převodníky
- 17 časovačů
- Až 3xI2C rozhraní
- Až 4 USARTy/2 UARTy
- Až 3 SPI
- USB 2.0 high-speed výstup
- CRC výpočetní jednotka
- Generátor náhodných čísel

2.2. Senzory

2.2.1. Ultrazvuk

- Pracovní napětí: 5V
- Pracovní proud: 15mA
- Pracovní frekvence: 40Hz
- Max. vzdálenost: 4m
- Min. Vzdálenost: 2cm
- Měřící úhel: 15˚
- Inicializační impuls: 10uS TTL úrovně
- Délka výstupního impulzu odpovídá naměřené vzdálenosti
- Rozměry: 45*20*15mm

2.2.2. IR detekce čáry

a) IR dioda
   - Pracovní napětí: 1,4V
   - Pracovní proud: 20mA
   - Pracovní vlnová délka: 940nm

b) IR tranzistor
   - V_{ce}: 30V
   - Pracovní vlnová délka: 940nm
   - IR filtr

2.3. Napájení

2.3.1. Baterie

- Použitá technologie: Li-ion
- Kapacita: 2600mAh
- Jmenovité napětí: 3,7V
- Maximální napětí po nabití: 4,2V
- 4ks

2.3.2. DC-DC převodník na 5V (LM2596)

- Vstupní napětí:4.5-35V
- Výstupní napětí: 3-33.5V (nastavitelné)
- Výstupní proud: 2A, špičkově 3A
- Efektivita převodu: až 92% (zvyšuje se s výstupním napětím)
- Spínací frekvence: 150KHz
- Minimální rozdíl vstupního a výstupního napětí: 2V
- Napěťová regulace: ± 0.5%
- 43.6 * 21.5 * 13.5 mm

2.4. Pohon

2.4.1. Lego NXT motory

- Převodový poměr: 1:48
- Napájecí napětí: 9V
- Rychlost otáčení: 160ot/min
- Maximální proud: 800mA

2.4.2. MIG 480 Race

- hmotnost: 105 g
- rozměry: 28x50mm
- max proud: 10 A
- průměr hřídele: 2.3mm

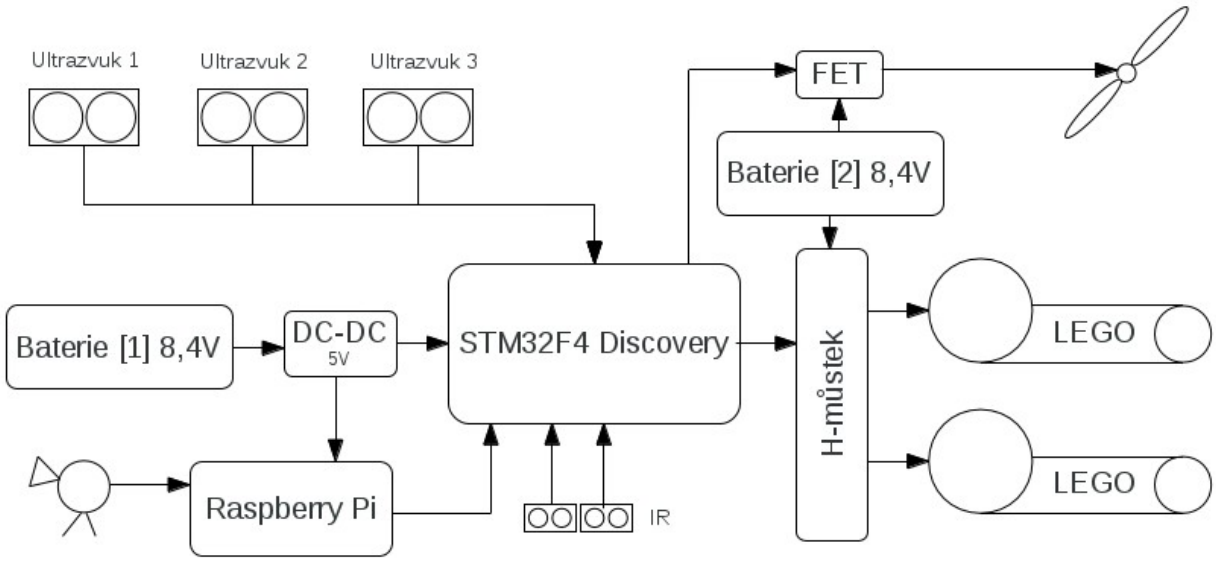
2.4.3. H-můstek

- Napájecí napětí: 4,5 – 36V
- Maximální proud: 600mA na motor, špičkově 1,2A

3. Popis

3.1. Jak to celé funguje

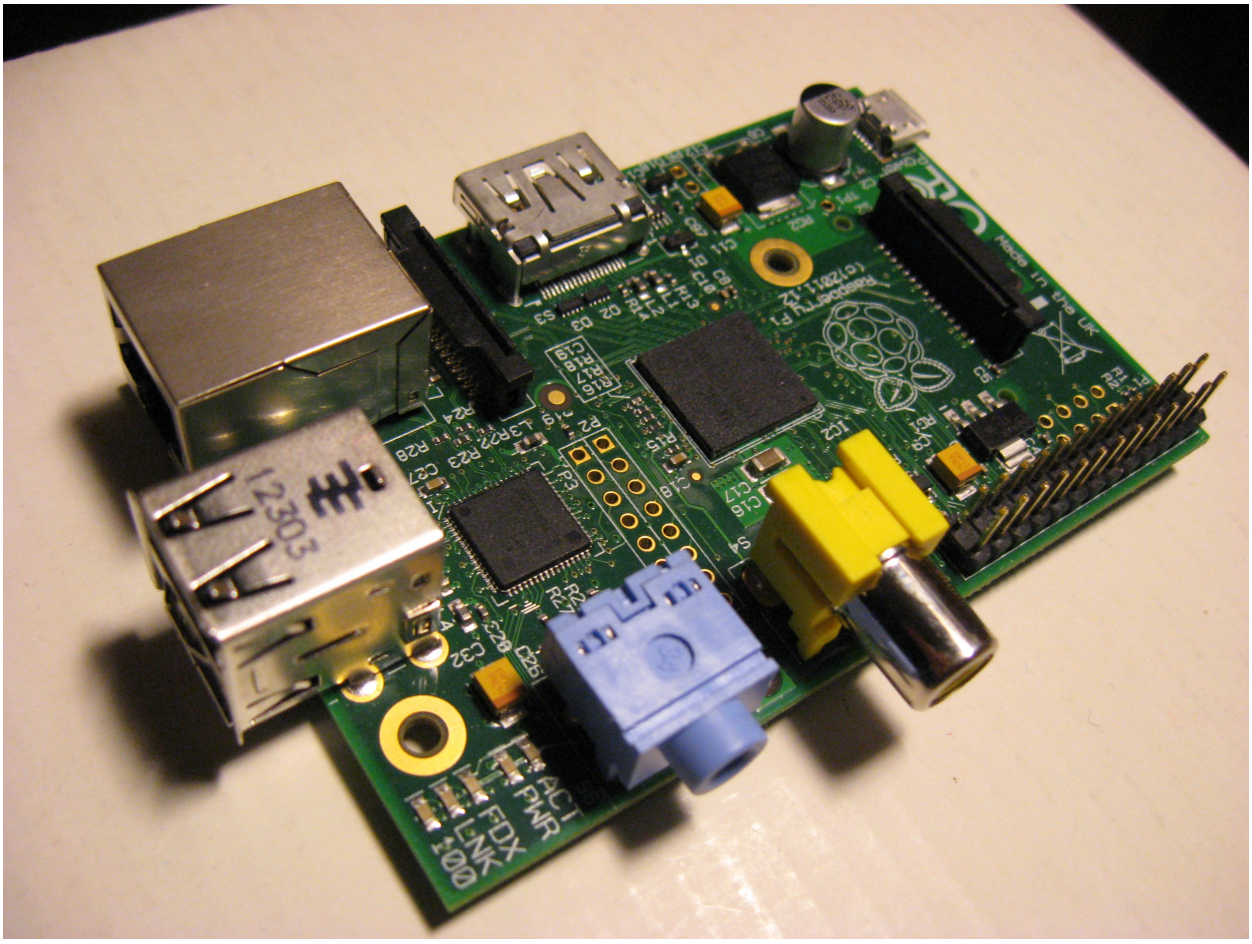
Tento robot je kompletně řízen ARM procesorem STM32F4 který obsluhuje vše od senzorů až po motory. Nejdůležitější vstupní senzor je web-kamera připojená přes druhý ARM procesor a to na desce Raspberry Pi, která obraz z kamery vyhodnotí a pošle souřadnice X a Y, do hlavního procesoru, kde se nachází nejbližší svíčka.



3.2. Raspberry Pi

3.2.1. Úvod

Raspberry Pi je jednodeskový počítač s deskou velikosti zhruba platební karty. Vyvíjí ho britská nadace Raspberry Pi Foundantion s cílem podpořit výuku informatiky ve školách. Jeho základem je SoC BCM2835 firmy Broadcom, který obsahuje procesor ARM1176JZF-S s taktem 700 Mhz, grafický procesor VideoCore IV a 256 megabajtů paměti RAM, u Raspberry Pi model B prodávaných před 15. říjnem 2012, od tohoto data by měl být Raspberry Pi prodáván s 512 MB pamětí RAM. Pro boot systému a trvalé uchování dat je určen slot na SD kartu.



3.2.2. Operační systém

Jelikož Microsoft nenabízí žádný operační systém pro procesory architektury ARM, má Linux na těchto zařízeních téměř výhradní postavení(např.: Android v mobilních telefonech využívá jádro Linuxu). Přímo pro tento typ procesoru je možnost si zkompilovat celý systém nebo stáhnout připravenou speciální distribuci pro toto zařízení a to Raspbian odvozený od linuxové distribuce Debian. Tento OS je připraven s lehkým prostředím LXDE, které se částečně podobá prostředí ve Windows. Systém jako takový obsahuje skoro vše, co je třeba k práci(internetový prohlížeč, PDF čtečka, kalkulačka...).

3.2.3. Program

Jako programovací jazyk jsem zvolil C++. Program jsem vyvíjel na počítači s Linuxem a jako programovací prostředí jsem použil multiplatformní prostředí Code::Blocks.

Navázání komunikace

```
int connect_STM32F4()
{
   // write(fd, "raspi\n", 6);


    counter = 0;
    int ok;
    while((ok != 1) && (counter < 10))
    {
        write(fd, "raspi\n", 6);
        counter++;
```

```
        int n = read(fd, buff, sizeof buff);
        if(n != 0)
        {
            char * pch;
            pch = strtok (buff,"\n");

            while (pch != NULL)
            {
                //printf ("%s\n",pch);
                char *s1 = pch;
                char *s2 = "STM32F4";
                int i = strcmp(pch, s2);

                if(i == 0)
                {
                    ok = 1;
                }

                pch = strtok (NULL, "\n");
            }
            memset(&buff, 0, sizeof buff);


        }
        cout << counter <<  endl;
    }
    if(ok == 1)
    {
        printf("Spojeni navazano\n");
        return 1;
    }
    printf("Spojeni se nezdarilo - Opakuji\n");
    return 0;
}
```

Jako jednoduchou identifikaci zařízení a základní ověření funkčnosti jsem nechal program vyslat slovo „raspi“ a kontrolovat přijímaná data zda neobsahují slovo „STM32F4“ co vyšle kit po přijmutí slova „raspi“.

Inicializace web-kamery

```
int init_opencv()
{
    san_cap.VideoCapture::set(CV_CAP_PROP_FRAME_WIDTH,320);
    san_cap.VideoCapture::set(CV_CAP_PROP_FRAME_HEIGHT,240);
    san_cap.VideoCapture::set(CV_CAP_PROP_BRIGHTNESS,0.5);

    cout << "Width of frame: " <<  san_cap.VideoCapture::get(CV_CAP_PROP_FRAME_WIDTH) << endl;
       // Width of the frames in the video stream
    cout << "Height of frame: " << san_cap.VideoCapture::get(CV_CAP_PROP_FRAME_HEIGHT) << endl;
       // Height of the frames in the video stream
    cout << "Image brightness: " << san_cap.VideoCapture::get(CV_CAP_PROP_BRIGHTNESS) << endl;
       // Brightness of the image (only for cameras)
    return 0;
}
```

Výběr nejsvětlejších míst v obraze

```
Mat theshold_input(Mat img) //v maticove promenne img je snimek z webkamery
{
    cvtColor(img, hsv, CV_RGB2HSV);
    // split image to H,S and V images
    split(hsv,slices);
```

```
    slices[0].copyTo(hue); // get the hue channel
    slices[1].copyTo(sat); // get the sat channel
    slices[2].copyTo(val); // get the V channel

    threshold (val, val1, lowSliderPosition3,255, CV_THRESH_BINARY);

    Mat element(10,10,CV_8U,Scalar(1));
    erode(val1,val1,element);
    return val1;
}
```

Detekce obrysů a jejich středů

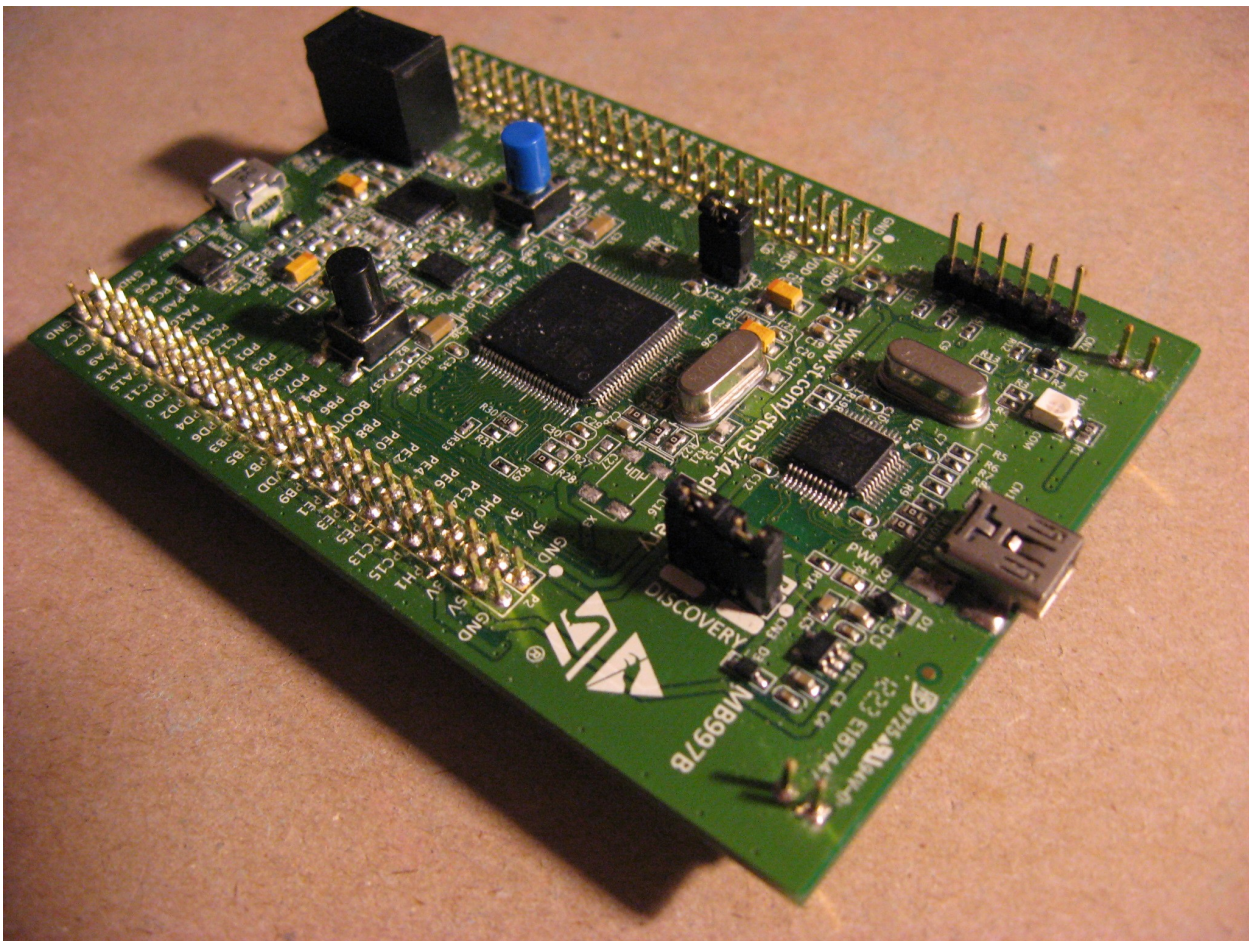
```
            threshed = theshold_input(c_out);
            vector<vector<Point> > contours;
            vector<Vec4i> hierarchy;

            /// Detect edges using canny
            /// Find contours
            findContours( threshed, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE,
Point(0, 0) );

            /// Get the moments
            vector<Moments> mu(contours.size() );
            for( uint32_t i = 0; i < contours.size(); i++ )
            {
                mu[i] = moments( contours[i], false );
            }

            ///  Get the mass centers:
            vector<vector<Point> > contours_poly( contours.size() );
            //vector<Point2f>center( contours.size() );
            vector<Point2f> mc( contours.size() );
            for( uint32_t i = 0; i < contours.size(); i++ )
            {
                mc[i] = Point2f( mu[i].m10/mu[i].m00 , mu[i].m01/mu[i].m00 );
            }
            vector<Rect> boundRect( contours.size() );
            /// Draw contours
            Mat drawing = Mat::zeros( threshed.size(), CV_8UC3 );
            for( uint32_t i = 0; i < contours.size(); i++ )
            {
                approxPolyDP( Mat(contours[i]), contours_poly[i], 3, true );
                boundRect[i] = boundingRect( Mat(contours_poly[i]) );
            }
            /* for( uint32_t i = 0; i< contours.size(); i++ )
             {
                 Scalar color = Scalar( rng.uniform(0, 255), rng.uniform(0,255),
rng.uniform(0,255) );
                 // drawContours( input, contours, i, color, 2, 8, hierarchy, 0, Point() );
                 circle( input, mc[i], 4, color, -1, 8, 0 );
                 rectangle( input, boundRect[i].tl(), boundRect[i].br(), color, 2, 8, 0 );

                 stringstream ss;//create a stringstream
                 ss << "X:"<< (int)mc[i].x << " Y:" << (int)mc[i].y << "   ---Candle:" << i <<
"\n";//add number to the stream

                 putText(input, ss.str(), Point2f(mc[i].x +10 , mc[i].y +30),
CV_FONT_HERSHEY_COMPLEX, 0.5, Scalar::all(255), 1, 8, false );
                 //cout << mc[i].x << ", " << mc[i].y << endl;
             }*/
            if(contours.size() > 0)
            {
                Scalar color = Scalar( rng.uniform(0, 255), rng.uniform(0,255),
rng.uniform(0,255) );
                // drawContours( input, contours, i, color, 2, 8, hierarchy, 0, Point() );
                circle( input, mc[contours.size()-1], 4, color, -1, 8, 0 );
                rectangle( input, boundRect[contours.size()-1].tl(), boundRect[contours.size()-
1].br(), color, 2, 8, 0 );

                stringstream ss;//create a stringstream
                ss << "X:"<< (int)mc[contours.size()-1].x << " Y:" << (int)mc[contours.size()-
1].y << " - Candle:" << contours.size()-1 << "\n";//add number to the stream
```

```
                putText(input, ss.str(), Point2f(mc[contours.size()-1].x +10 ,
mc[contours.size()-1].y +30), CV_FONT_HERSHEY_COMPLEX, 0.5, Scalar::all(255), 1, 8, false );
                //cout << mc[i].x << ", " << mc[i].y << endl;


                        ///send top
                        stringstream sd;
                        sd << "c|" << (int)mc[contours.size()-1].x << "/" <<
(int)mc[contours.size()-1].y << "\n";
                        std::string s = sd.str();
                        const char *Candle_Pos = s.c_str();

                        sd.seekg(0, ios::end);
                        int sizee = sd.tellg();

                        if(sizee < 11)
                        {
                            write(fd, Candle_Pos, sizee);
                        }
            }
```

3.3. STM32F4-Discovery

3.3.1. O co jde?

Je to vývojová deska procesoru STM32F407VGT6 od firmy STMicroelektronic. Cena této desky přibližně odpovídá prodejní ceně jednoho procesoru na této desce umístěné, takže je dotována firmou.



3.3.2. Co nabízí?

Na této desce je umístěno kromě procesoru také:

- mini_USB – programování a debug
- SWD konektor – programování procesoru na desce nebo mimo
- ST-LINK/V2 – celý programovací blok
- 4 uživatelské LEDky
- 1 uživatelské tlačítko
- uživatelské micro-USB
- 3-osý akcelerometr
- mikrofon
- Zvukový výstup – 3,5 Jack
- dva 50ti pinové rozšiřující konektory

3.3.3. Jaké jsou jeho možnosti?

- Takt: 168MHz (externí oscilátor na 8MHz)
- Napájení: 1.8 V - 3.6 V
- 3×12-bit, 2.4 MSPS A/D převodníky až s 24 kanály
- 2×12-bit D/A převodníky
- 17 časovačů
- Až 3xI2C rozhraní
- Až 4 USARTy/2 UARTy
- Až 3 SPI
- USB 2.0 high-speed výstup
- CRC výpočetní jednotka
- Generátor náhodných čísel
- Hash
- DMA
- Watchdog
- Ethernet (bez HW na desce)
- Přerušení (externí/interní)
- a mnoho dalšího

3.3.4. Po zakoupení

K této desce není žádné příslušenství. Prodává se s nahraným Demo programem, který nejenže podle naklonění rozsvěcí a zhasíná LEDky umístěné kolem akcelerometru, ale i po připojení k počítači přes micro-usb konektor se chová jako myš závislá na naklonění.

3.3.5. Propojení s PC

Potřebný ovladač pro operační systém v PC je nutno stáhnout z internetové stránky desky na www.st.com. Dobrá volba je i stáhnout a nainstalovat program „ST-Link Utility“ ze stejné adresy. Tato utilita dokáže pracovat s pamětí čipu(nahrávaní, stahování programů a kompletní mazání paměti).

Po instalaci nezbytných podpůrných programů je třeba zvolit programovací prostředí. Já jsem zvolil zdarma dostupné od firmy Keil a to µVision 4. Podporuje jak přímé nahrávání programu do paměti čipu, tak i jeho ladění.

3.3.6. Keil µVision 4

Pro rychlý začátek vyvíjení programů a učení se ST připravilo sadu příkladů také pro toto prostředí takže po otevření projektu je možné ho okamžitě zkompilovat a nahrát do MCU. Při vytváření nového projektu je nutné nastavit cesty ke knihovnám, bez kterých bychom jen stěží něco naprogramovali.

3.3.7. Programování ARM procesorů

ARM je velice úsporná a šetřivá architektura. Standardně není hodinový signál přiveden skoro nikam a bez něj nic v této architektuře nepracuje. Takže pokud potřebujeme změnit stav pinu tak se nejdříve musí přivést hodinový signál na příslušný port, a až poté je možné nastavit vlastnosti jednotlivých pinů.

```
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE); //přivede hodiny na port A

  GPIO_InitStructure_Ultrasonic.GPIO_Pin = GPIO_Pin_15;
  GPIO_InitStructure_Ultrasonic.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure_Ultrasonic.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure_Ultrasonic.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStructure_Ultrasonic.GPIO_PuPd = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure_Ultrasonic);
```

Inicializace pinu 15 na portu A

```
rs = received_string2;
s1 = "raspi";
i = strcmp(s1, rs);
if(i == 0)
{
       USART_puts(USART2, "STM32F4\n");
       STM_EVAL_LEDToggle(LED6);
}
ps = strtok(rs,"|");
while (ps != NULL)
{
       s1 = "c";
       i = strcmp(s1, ps);
       if(i == 0)
       {
              ps = strtok (NULL, "/");
              while (ps != NULL)
              {
                     sscanf (ps, "%d", &candle[cnt]);
                     ps = strtok (NULL, "/");
```

```
                        cnt++;
                }
                        candle_position(candle [0],candle [1]);
                        cnt = 0;
        }
                ps = strtok (NULL, "|");
}
```

Navázání spojení s Raspberry Pi a případné předání příchozích dat o pozici

3.4. Ultrazvukové senzory

3.4.1. Co je to ultrazvuk?

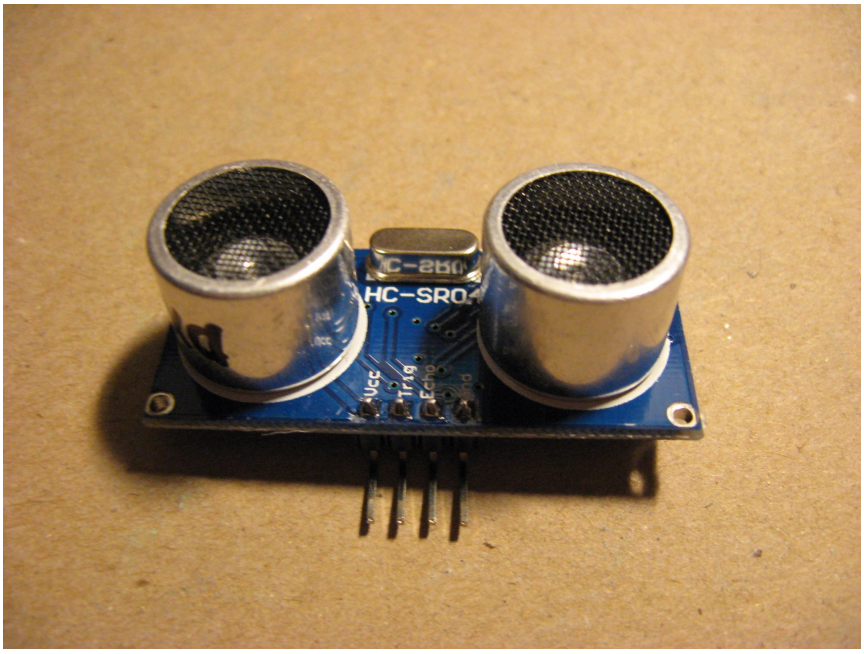
Ultrazvuk je akustické vlnění, jehož frekvence leží nad hranicí slyšitelnosti lidského ucha, tedy nad hranicí zvuku = cca 20 kHz. Tím pádem, byť má stejnou fyzikální podstatu jako zvuk, je pro lidské ucho neslyšitelný, ale řada živočichů může část ultrazvukového spektra vnímat (delfíni, psi, netopýři)

Vlnová délka ultrazvuku je menší než vlnová délka zvukového vlnění, proto je ultrazvuk méně ovlivněn ohybem. Výrazný je jeho odraz od překážek, absorpce ve vzduchu (v plynech) a je méně pohlcován kapalinami a pevnými látkami. Absorpce také narůstá s frekvencí ultrazvuku.

3.4.2. Senzor HC-SR04

Tento senzor byl vyvinut pro spolupráci hlavně s Ingardenem (open-source platforma založená na mikrokontroleru ATMega od firmy Atmel a grafickém vývojovém prostředí, které vychází z prostředí Wiring). Tomu jsem se musel přizpůsobit a vybrat komunikační piny tolerující 5V logiku přesto že sami používají 3V logiku. Senzor toleruje TTL úrovně signálu.

3.4.3. Konstrukce senzoru

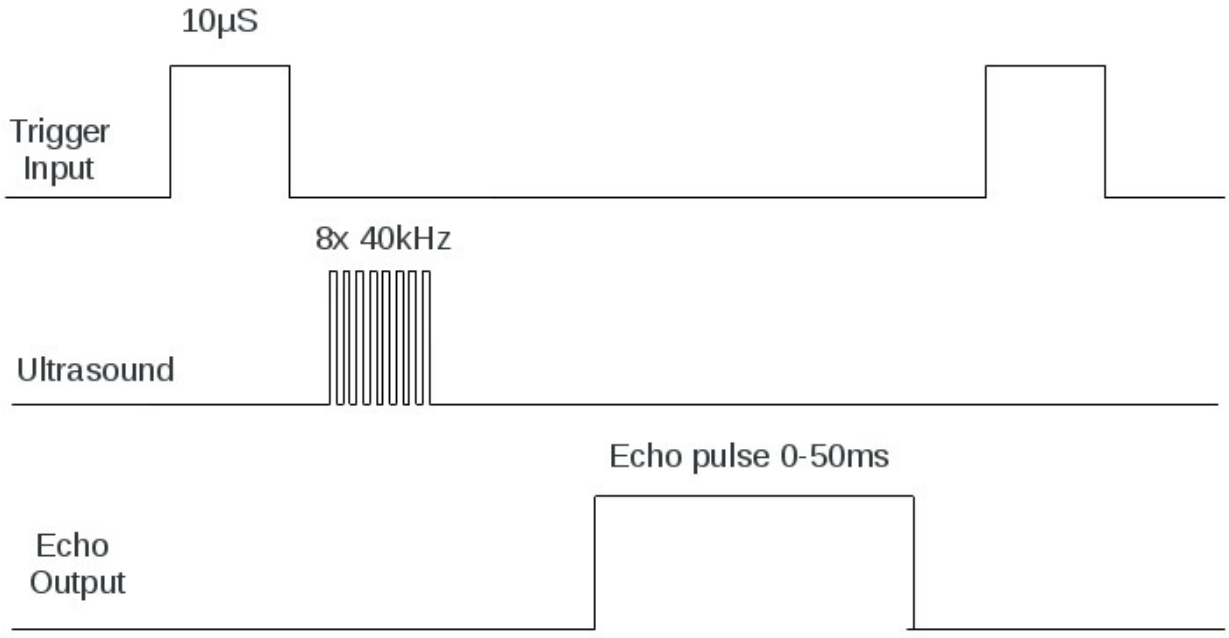


Senzor disponuje čtyřmi piny. Dva pro napájení a dva pro komunikaci. Napájení senzoru je 5V.

3.4.4. Komunikace senzoru

Senzor pro komunikaci využívá dvou pinů. Jeden pro start měření a druhý pro odečtení změřené vzdálenosti.

Start měření se provede odesláním impulsu o délce 10µs na Trigger pin senzoru. Změřená vzdálenost se projeví na Echo pinu jako impuls s proměnnou délkou závislou na změřené vzdálenosti. Při nezjistitelné vzdálenosti vlivem buď úhlem pohledu na překážku nebo její nadměrné vzdáleností se výstupní impuls natáhne až na 100ms.

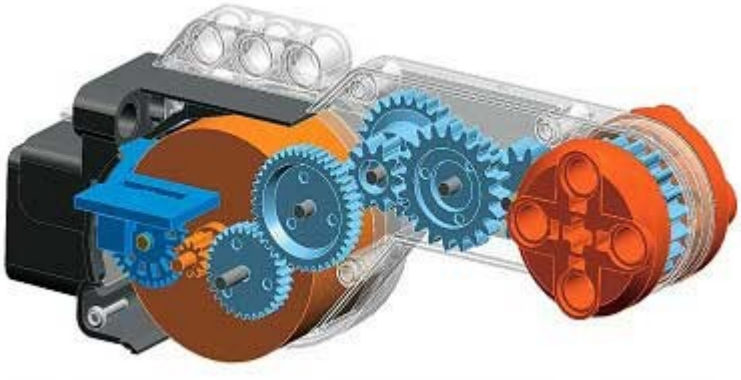


3.5. Lego NXT motory

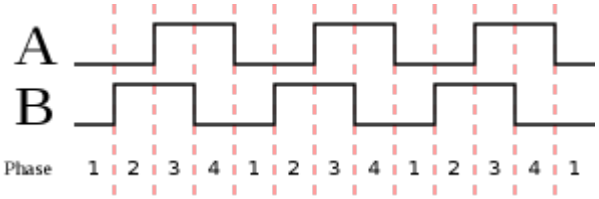
3.5.1. Zapojení konektoru

Pin konektoru	Barva vodiče kabelu	Funkce
1	Bílá	Motor M1.1
2	Černá	Motor M1.2
3	Červená	GND
4	Zelená	5V napájení enkodéru
5	Žlutá	Enkodér 1
6	Modrá	Enkodér 2

3.5.2. Konstrukce



3.5.3. Enkodéry otáčení



Každý motor obsahuje dva enkodéry otáčení fázově posunuté o 90 stupňů. Podle toho, který enkodér předbíhá druhý lze jednoznačně poznat směr otáčení. Jedno otočení hřídele motoru vyvolá 180 impulzů na každém enkodéru. Této vlastnosti využívat k přesnému řízení robota. Díky enkodérům lze robota nebo jen jedno kolo pootočit o určitý úhel. Lehkým výpočtem kdy obvod kola je 16cm se jeden rozlišitelný stupeň otočení bude rovnat 0,08cm. S těmito údaji lze vypočítat i pootočení robota o určitý počet stupňů. Poloměr otáčení kdy jedno kolo stojí je 15,5cm.

$$L = ((2 \cdot \pi \cdot 15.5) \div 360) \cdot \alpha$$

L = vzdálenost nutná k ujetí, aby se robot pootočil o úhel `alfa`

`V programu STM32 to vypadá takto:`

```
void move_robot_degree(int degr, int torque) // degr = ((2*pi*15.5)/360)*uhel = 0.270526034 *
uhel
{
	degr = 2.70526034f * degr;
	if(degr < 0)
	{
		degr = degr * -1;
		stop(left,block);
		move_unit_single(right, forward, torque, length, degr);
	}else if(degr > 0)
	{
		stop(right,block);
		move_unit_single(left, forward, torque, length, degr);
	}
}
```

Funkce k pootočení robota o zadaný počet stupňů

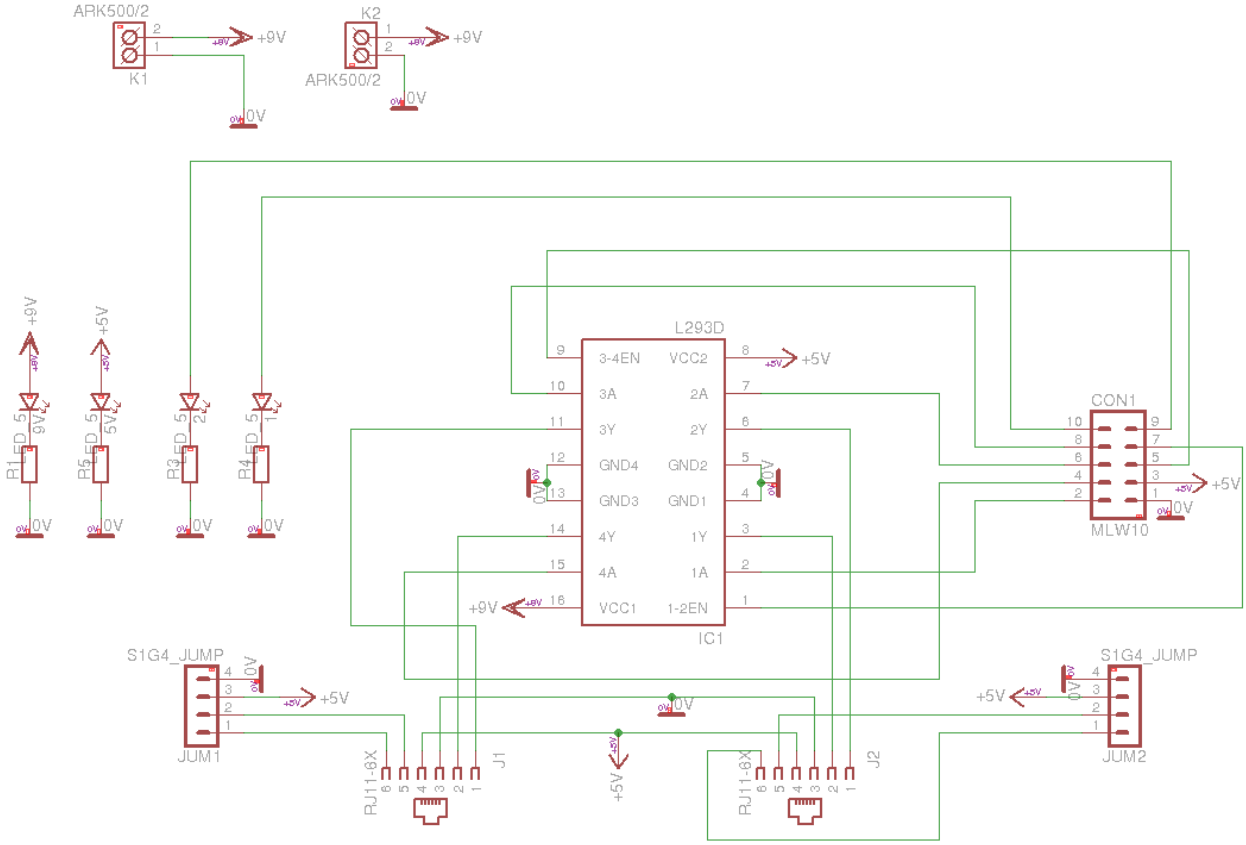
```
void move_unit_single(int motor, int way, int torque, int unit, int num)
{
	if(motor == left)
	{
		start_left_enc = left_encoder;
		if(unit == degree)
		{
			if(way == forward)
```

```
                        {
                                while(start_left_enc + num >= left_encoder)
                                {
                                        move(left, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }else
                        {
                                while(start_left_enc - num <= left_encoder)
                                {
                                        move(left, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }
                }else if(unit == length)
                {
                        num = (180*num)/160;
                        if(way == forward)
                        {
                                while(start_left_enc + num >= left_encoder)
                                {
                                        move(left, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }else
                        {
                                while(start_left_enc - num <= left_encoder)
                                {
                                        move(left, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }
                }
        }else
        {
                start_right_enc = right_encoder;
                if(unit == degree)
                {
                        if(way == forward)
                        {
                                while(start_right_enc + num >= right_encoder)
                                {
                                        move(right, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }else
                        {
                                while(start_right_enc - num <= right_encoder)
                                {
                                        move(right, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }
                }else if(unit == length)
                {
                        num = (180*num)/160;
                        if(way == forward)
                        {
                                while(start_right_enc + num >= right_encoder)
                                {
                                        move(right, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }else
                        {
                                while(start_right_enc - num <= right_encoder)
                                {
                                        move(right, way, torque);
                                        //if(candle_saw == 1)candle_sawed();
                                }
                        }
                }
        }
}
```

Pootočení kola robota o zadané jednotky a směr

3.6. H-můstek

3.6.1. Celkové zapojení



3.6.2. Nastavení

Pro každý motor má dva ovládací piny. Vhodnou kombinací těchto pinů lze dosáhnout změny otáčení motoru nebo jeho zastavení a nereagování na hlavní vstupní pin. Dále je na můstku vstup pro řízení každého motoru.

3.7. MIG 480 Race

3.7.1. Řízení

Pro řízení a spínání tohoto motoru využívám výkonového tranzistoru, který při TTL úrovni signálu je plně sepnut – IRL540. Tento tranzistor je přímo řízen procesorem, kde na jeho výstupu je PWM signál.

3.7.2. PWM

Tato modulace využívá změny střídy k dosažení změny výkonu motoru. Jeho výkon lze regulovat od 0% do 100%.

4. Hlavní program pro STM32F4

4.1. Náhodné ježdění po hřišti

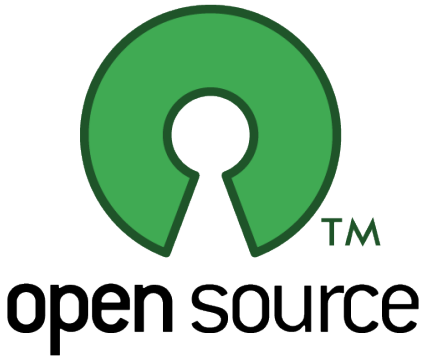
```
void chaos(void)
{
    move_double(forward, 1000); //oba motory naplno dopředu
    while(1)
    {
        if(left_encoder<right_encoder?(left_encoder>(enc_last_left+stop_length)):
(right_encoder>(enc_last_right+stop_length))) //když ujede určitou vzdálenost
        {
           enc_last_left+=stop_length;
           enc_last_right+=stop_length;
           for(t=1;t<9;t++) //8x se pootočit o 45 stupňů
                                                {
                                                        for (h = 1; h < 45; h++) //pootočit robota o
45 stupňů
                                                        {
                                                                if(candle_saw ==
1)candle_sawed();//když vidí svíčku
                                                                move_robot_degree(1, 800);//pootočit o
1 stupeň
                                                                stop_robot(block);//zastavit robota
                                                        }
                                                        stop_robot(block);
                                                        if(t!= 8)Delay(500);
                                                        if(candle_saw == 1)candle_sawed();//když vidí
svíčku
                                                }
            move_double(forward, 1000);//oba motory naplno dopředu
        }
        if(left_white || right_white || (distance_out[left_ultra] < 20) ||
(distance_out[right_ultra] < 20) || (distance_out[center_ultra] < 20)) //pokud je na čáře nebo
vidí stěny
        {
            stop_robot(block); //zastavit robota
            if(left_white || (distance_out[left_ultra] < 20)) //je na čáře nebo u stěny levým
senzorem
            {
                for (h = 1; h < chaoss; h++) //pootočit o 90-120 stupňů
                    {
                        if(candle_saw == 1)//když vidí svíčku
                           candle_sawed();
                        move_robot_degree(1, 800);//pootočit o 1 stupeň
                        stop_robot(block);//zastavit robota
                    }
            }
            else if(right_white || (distance_out[right_ultra] < 20)) //je na čáře nebo u stěny
pravým senzorem
            {
                for (h = 1; h < chaoss; h++) //pootočit o 90-120 stupňů
                    {
                        if(candle_saw == 1)//když vidí svíčku
                           candle_sawed();
                        move_robot_degree(-1, 800);//pootočit o 1 stupeň
                        stop_robot(block);//zastavit robota
                    }
            }
            else if(distance_out[center_ultra] < 20)  //je u stěny středním senzorem
            {
               move_unit_double(backward, 800, length, 150);
               stop_robot(block);
               for (h = 1; h < chaoss; h++) //pootočit o 90-120 stupňů
                    {
                        if(candle_saw == 1)//když vidí svíčku
                           candle_sawed();
                        move_robot_degree(-1, 800); //pootočit o 1 stupeň
                        stop_robot(block);//zastavit robota
                    }
            }
```

```
            stop_robot(block); //zastavit robota
            chaoss += 10;
            if(chaoss > 120)
            {
                chaoss = 90;
            }
            move_double(forward, 1000); //oba motory naplno dopředu

        }
                        if(candle_saw == 1) //když vidí svíčku
                    candle_sawed();
    }
}
```

5. Open Source

5.1. Co to znamená?

Tyto dvě anglická slova znamenají volnost -> volnost zdrojového kódu. Každý ho může upravovat, šířit, sdílet, propagovat (ještě záleží na konkrétní licenci)... prostě je volný!



5.2. Co má společného Open Source a tento robot?

Skoro vše!

Při stavbě a používání robota využívám tyto OpenSource řešení:

- Fedora – Linuxová distribuce – veškeré programování, navrhování, hledání – hlavní OS
- Raspbian – Distribuce linuxu speciálně pro Raspberry Pi
- OpenCV – knihovna speciálně na práci s obrazem a jeho rozpoznávání
- Code::Blocks – multiplatformní multijazyčné programovací prostředí
- LibreOffice – kancelářský balík ve kterém píši tuto dokumentaci
- GCC – kompilátor který kompiloval programy pro Raspberry Pi

5.3. Git repozitář

5.3.1. Co je to?

Git je systém správy verzí vytvořený Linusem Torvaldsem, původně pro vývoj jádra Linuxu.

Slouží hlavně pro Open Source vývojáře. Podporuje verzování, spojování, spolupráci více vývojářů na jednom projektu ...

5.3.2. Github a tento robot

Všechny zdrojové soubory z tohoto robata jsem umístil do vzdáleného repozitáře na serveru http://github.com, konkrétně pod touto adresou: https://github.com/omron93/fire-pi

6. Rozpiska součástek

6.1. Řízení

Název	Počet	Cena
STM32F4 – Discovery	1x	423,00 Kč
Raspberry Pi	1x	1 390,00 Kč
Raspberry Pi – krabička	1x	175,00 Kč
Raspberry Pi – SD karta	1x	355,00 Kč
Raspberry Pi – Flash	1x	1 090,00 Kč
Raspberry Pi – USB HUB	1x	105,00 Kč
	Celkem:	3 538,00 Kč

6.2. Osazení desek plošných spojů

Název	Počet	Cena
LED 5MM IR940 30mW/20°	2x	10,00 Kč
L-53P3BT	2x	6,00 Kč
P-SW-09D	2x	22,20 Kč
Konektor XINYA PFL10	2x	10,00 Kč
Konektor XINYA PFL26	2x	20,00 Kč
Konektor XINYA MLW10G	2x	12,00 Kč
Konektor XINYA MLW26G	1x	11,00 Kč
Svorkovnice PTR AKZ120/2DS-5.08-V-GREY	3x	14,10 Kč
Oboustranný kolík XINYA S1G40 2,54mm	2x	18,00 Kč
Dutinková lišta XINYA BL250G	2x	50,80 Kč
Oboustranný kolík S2G10 2mm	2x	7,00 Kč
WEBP 6-6 CONN	2x	27,00 Kč
RG-MPFK6W-BAL WHITE	1x	6,50 Kč
RJ konektor WS 6-6	4x	10,00 Kč
1N4007	1x	1,00 Kč
SOKL 16	1x	2,00 Kč
L293D	1x	38,40 Kč
LED	7x	14,00 Kč
KON-PC-SPK	10x	15,00 Kč
KON-PC-SPK-PI	40x	40,00 Kč
	Celkem:	335,00 Kč

6.3. Ostatní

Název	Počet	Cena
IRL540N	1x	33,30 Kč
18650 Li-Ion – Držák	2x	44,00 Kč
TrustFire 2500mAh – Li-Ion	4x	340,00 Kč
Pouzdro na 18650	2x	40,00 Kč
Webkamera	1x	300,00 Kč
Motor	1x	250,00 Kč
Vrtule	1x	150,00 Kč
Ultrazvuk	3x	300,00 Kč
DC-DC LM2596	1x	80,00 Kč
DC Digit Voltmetr	1x	150,00 Kč
	1x	
Merkur	1x	100,00 Kč
Čokoláda	1x	20,00 Kč
Páska	1x	30,00 Kč
	Celkem:	1 837,30 Kč

7. Závěr

7.1. Použité vybavení

V této práci jsem se zaměřil na využití nejmodernějších procesorů ARM a nejnovější pokrokový software s důrazem na svobodnost a otevřený zdrojový kód. Tento záměr se povedl a až na složité programovací prostředí pro STM32F4 od firmy ST, které nemá vhodnou svobodnou alternativu, jsem využíval spolu s linuxem převážně projektů s otevřeným zdrojovým kódem.

7.2. Soutěž

Jako další cíl této práce byla účast na soutěži RoboRave, kde se mi s pomocí (soutěž byla pouze pro minimálně dvoučlenné týmy) povedlo obsadit 1. místo. Robot při této soutěži sfoukl nejvýše 3 svíčky a to opakovaně. Občas se stalo, že robot viděl soupeřovu svíčku a snažil se jí sfouknout.

7.3. Možné vylepšení

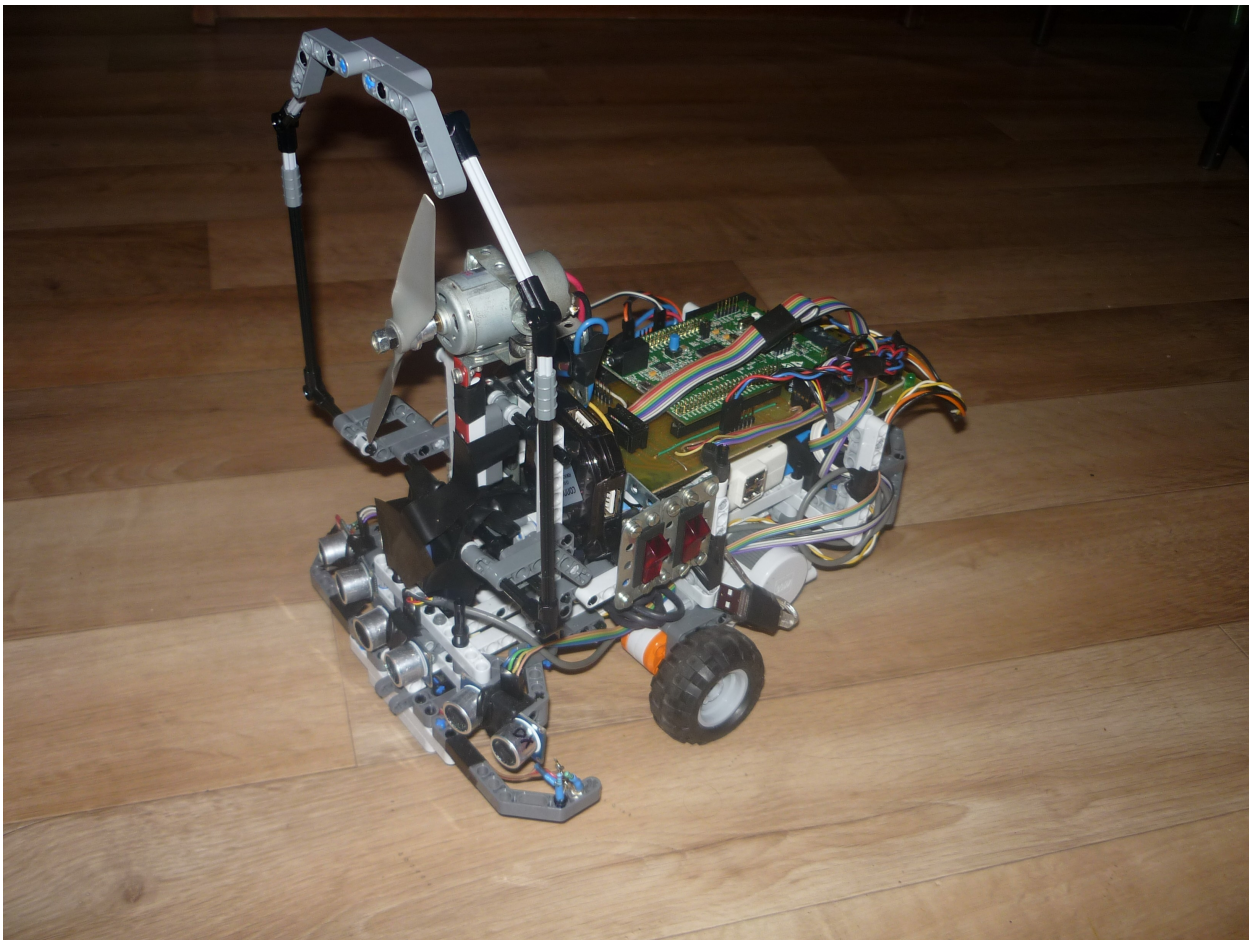
U tohoto robota bylo v plánu laserové snímání okolí a tím i vyhledání všech překážek. Bohužel naše web-kamera dost dobře nerozlišovala červený laserový paprsek. Proto nebylo možné tento způsob orientace využít. Nicméně rozpoznávací software na tento druh snímání jsem již měl vytvořen, ovšem jeho nasazení by ještě potřeboval několik zásadních úprav. Robot by tak získal dokonalý přehled o překážkách před ním, a jeho vyhnutí se jim by mu nečinilo sebemenší problém.

8. Zdroje

- http://pandatron.cz/?2612&opencv_%96_pocitacove_videni_pro_kazdeho
- http://cs.wikipedia.org/wiki/Ultrazvuk
- www.micropik.com/PDF/HCSR04.pdf
- http://www.philohome.com/nxtmotor/nxtmotor.htm
- http://blog.tkjelectronics.dk/2011/06/nxt-motor-shield/
- http://blog.tkjelectronics.dk/2011/10/nxt-shield-ver2/
- http://www.st.com/web/en/catalog/tools/PF252419
- http://www.raspberrypi.org/
- http://elinux.org/RPi_Hub
- http://github.com
- http://cs.wikipedia.org/wiki/Git

9. Přílohy

9.1. Foto robota



9.2. Přílohy na DVD

- Tato dokumentace
- Prezentace k této dokumentaci
- Další fotografie robota
- Zdrojové kódy pro Raspberry Pi
- Raspbian
- Zdrojové kódy pro STM32F4 – Discovery
- Keil µVision4
- Dokumentace plošných spojů – Eagle 6.3
- Kopie repozitáře na github.com