Laboratorní úlohy pro předmět Robotika

Jaroslav Malík

Bakalářská práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Jaroslav Malík**
Osobní číslo:	**A13119**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Laboratorní úlohy pro předmět Robotika**
Téma anglicky:	**Laboratory Exercises for the Robotics Course**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma.**
2. **Uveďte popis Arduino robota a možnosti využití ve výuce robotiky.**
3. **Vyberte a popište vhodné senzory a prostředky pro komunikaci s okolím a jejich připojení k robotu.**
4. **Navrhněte a vytvořte vhodné laboratorní úlohy.**
5. **Pro vybrané laboratorní úlohy vytvořte vzorová řešení.**
6. **Vytvořte elektronický manuál, jehož součástí bude popis práce s robotem, popis a připojení vybraných periférií a jednotlivé realizované laboratorní úlohy.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **CORKE, Peter I. Robotics, vision and control: fundamental algorithms in Matlab. Berlin: Springer, 2011, xxiv, 570 s. Springer tracts in advanced robotics. ISBN 978-3-642-20143-1.**
2. **DUDEK, Gregory a Michael JENKIN. Computational principles of mobile robotics. Second edition. New York: Cambridge University Press, 2010, xiii, 391. ISBN 978-0-521-87157-0. Dostupné také z: http://www.loc.gov/catdir/enhancements/fy1010/2010020795-t.html**
3. **WARREN, John-David, Josh S ADAMS a Harald MOLLE. Arduino robotics. New York: Apress, [2011], xxiv, 601. Technology in action. ISBN 978-1-4302-3183-7.**
4. **MARGOLIS, Michael. Arduino cookbook. Second edition. Sebastopol: O'Reilly, 2011, xx, 699. ISBN 978-1-4493-1387-6.**
5. **MARGOLIS, Michael. Make an Arduino-controlled robot. Sebastopol, Calif.: O'Reilly, c2013. Make. ISBN 978-1-449-344375.**
6. **MCROBERTS, Michael. Beginning Arduino. Second ed. Berkeley, CA: Apress, [2013]. Technology in action series. ISBN 978-1-4302-5016-6.**

Vedoucí bakalářské práce: **Ing. Petr Navrátil, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **21. prosince 2018**

Termín odevzdání bakalářské práce: **15. května 2019**

Ve Zlíně dne 21. prosince 2018

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 10. 5. 2019

Jaroslav Malík v. r.
podpis diplomanta

ABSTRAKT

Tato práce se zabývá úlohami pro platformu Arduino Robot, možnostmi této platformy, taktéž i možnostmi připojení vybraných periferií (senzorů, komunikačních prostředků) k této platformě. Klade si za cíl vytvoření praktických a zajímavých laboratorních úloh pro předmět robotika, a tím inicializovat zájem studenta o tento obor. Součástí práce je i vzorové zpracování vybraných úloh tak, že studenta provádí od základní práce s programováním mikropočítače až po složitější úlohy specifické pro mobilní robotiku, jako je například sledování čáry. Arduino Robot je koncipován jako výuková platforma, která má při výuce co nabídnout.

Klíčová slova: Arduino Robot, Arduino IDE, platforma Arduino, laboratorní výukové úlohy, programování mikropočítačů, mobilní robotika

ABSTRACT

This thesis is concerned with exercises for Arduino Robot platform, its possible uses, as well as with possibilities to connect selected peripheries (sensors, communication resources) to the platform. The goal of the thesis is to create practical and interesting laboratory exercises for the robotics course in order to cause an interest in the subject on the part of a student. Part of the thesis is also the exemplar treatment of selected tasks in a way that the student is walked through from the basic work with microcomputer programming up to the more complex tasks specific for mobile robotics such as line following. Arduino Robot is designed as a learning platform which has something to offer in the process of learning.

Keywords: Arduino Robot, Arduino IDE, Arduino platform, laboratory teaching exercises, microcomputer programming, mobile robotics

Děkuji Ing. Petru Navrátilovi, Ph.D., za odborné vedení a cenné rady při psaní bakalářské práce. Také bych rád poděkoval své manželce za podporu při studiu a trpělivost.

OBSAH

ÚVOD

Robotika je relativně mladé, ale zato dynamicky se rozvíjející odvětví. Robotizace a automatizace je v podstatě nevyhnutelná a stává se součástí našich každodenních životů. V mnoha domácnostech již jezdí automatické vysavače a čističe oken. Ve světě se experimentuje s autonomními vozidly. A mnoho dalších robotických systémů pracuje v továrnách, ve skladech a firmách všeho druhu. Tento trend je většinou nazýván jako průmyslová revoluce čtvrté generace (Industrie 4.0). Pokrok v této oblasti je dán a poháněn stále efektivnějšími způsoby zpracování dat. Toho je dosaženo dramatickým nárůstem výpočetní síly (dle empirického pravidla o exponenciálním růstu výkonu zvaného Moorův zákon), spojováním jednotlivých systémů do větších sítí i trend decentralizace jednotlivých celků za účelem prospěšného synergetického spojení.

Robotika je tedy velmi zajímavým, perspektivním a žádaným odvětvím. Proto je vhodné studenty zaujmout praktickými ukázkami (úlohami), které pomohou překonat vstupní bariéru do tohoto oboru. Již na některých základních školách existují různé robotické kroužky, vysoké školy (a nejen ty) organizují různé robotické soutěže, nebo dokonce ligy. Je tedy škoda, že i při výuce není tomuto věnováno více prostoru.

Cílem této práce je mimo jiné i „přívětivou formou“ nalákat studenty ke studiu programování mikropočítačů, a tudíž i různých vestavěných systémů, které jsou základními kameny robotiky.

Pro výuku byla zvolena platforma Arduino Robot. Její výhodou je, že je po hardwarové stránce kompletní a odpadá tím nutnost vývoje, návrhu a výroby této části. Taktéž se můžeme obejít zcela bez pájení, jelikož robot má již některé periferie zabudované či můžeme využít dostupné volné piny a malé kontaktní nepájivé pole (které ale není součástí robota). Další nespornou výhodou je otevřenost celé platformy Arduino s řadou volně dostupných knihoven. Zkušenosti s Arduino Robotem poté může student využít při návrhu vlastního řešení, ale to již není součástí této práce.

První část práce se věnuje robotice a klasifikaci robotů, platformě Arduino a vývojovému prostředí. Dále pak samotné platformě Arduino Robot, periferiím a standardům komunikace po teoretické stránce. Taktéž i dostupné dokumentaci, knihovnám, příkladům a návodům vztahujícím se k této platformě.

V druhé, praktické části, jsou vytvořeny konkrétní laboratorní úlohy a popsán postup řešení. Příklady řešení vybraných úloh jsou uvedeny v příloze. Tyto úlohy jsou zpracovány také do podoby webové stránky v redakčním systému Wordpress, kde jsou podrobněji rozebrány.

Závěr práce poté zhodnocuje vytvořené úlohy a případné komplikace a úskalí objevené v rámci implementace jednotlivých úloh.

I. TEORETICKÁ ČÁST

1 ROBOTIKA

Robotika je disciplína, která se zabývá vytvářením, řízením a programováním inteligentních strojů, robotů. Zahrnuje použití robotů pro řešení různorodých úloh. Zkoumá řídící procesy, akční členy, senzory a algoritmy. Zakládá se na mnoha vědeckých a inženýrských oborech. Je významným podoborem mechatroniky, která v sobě synergeticky spojuje řízení, mechaniku, elektroniku a informatiku. Jedná se o relativně mladý obor, který je stále v etapě rychlého vývoje. Zjednodušeně řečeno, robotika je nauka o robotech. [1] [2] [3]

1.1 Robot

Co je robot? Jednoznačná a ustálená definice bohužel neexistuje, což je dáno i tím, že každá z vědeckých disciplín, která tento technický systém zkoumá, preferuje svůj vlastní pohled na tuto problematiku. V roce 1986 McKerrow definuje robot jako stroj, který může být naprogramován k vykonávání různých činností. Australian Robotics and Automatio Association definuje tři charakteristiky robota:

- umožňuje alespoň nějakou formu mobility
- může být naprogramován k různorodým úkolům
- po naprogramování pracuje v samostatném režimu

V literatuře se často setkáme s rozdělením robotů do dvou kategorií. Rozlišujeme průmyslové roboty, tj. roboty pracující ve výrobě a servisní roboty, čili roboty, které vykonávají určitou službu, tedy činnost, která není výrobní. Roboty můžeme dále dělit do kategorií podle různých hledisek. Podle míry autonomie na: řízený, ovládaný, regulovaný, autonomní či inteligentní stroj. Podle prostředí na: suchozemské, vodní, vzdušné, vesmírné, hybridní… Dle možnosti přemísťování na: stacionární či mobilní. U mobilních dále podle typu pohybu, atd. [4] [2]

Speciální kategorii tvoří humanoidní roboty/roboti. Jsou podobné člověku vzhledem, ale také projevy inteligence. Je u nich kladen důraz na samostatnost a interakci s prostředím. Využívají algoritmy strojového učení, neuronových sítí, umělé inteligence, počítačové rozpoznávání, vnímáni, porozumění a rozhodování. U této kategorie je možné používat slovo robot v životném tvaru podle vzoru pán, neboť již v mnohém připomínají člověka, byť do živé a myslící bytosti mají ještě hodně daleko. Humanoidními roboty se po filozofické stránce ve svých sci-fi knihách hodně věnoval spisovatel Isaac Asimov, který pro ně definoval slavné tři zákony robotiky. [5]

Slovo robot má české kořeny. Byl to totiž Karel Čapek, kdo poprvé roku 1920 (na radu svého bratra Josefa) použil slovo robot ve své knize R.U.R. jako označení pro inteligentní pracovní stroje. Předobraz by se dal hledat i v postavě Golema, postavě z hlíny oživené tajemným šémem, který byl podle pověsti stvořen pražským rabínem Jehudou Löwem. [5]

1.2 Praktická výuka robotiky

Robotika je vyučována na mnoha univerzitách i středních školách. Robotické kroužky se objevují i na základních školách. Není se čemu divit, robotika je perspektivní obor. Odborníci na robotiku odhadují velký nárůst v tomto odvětví. Odhaduje se, že do pár desítek let stroje zvládnou většinu pracovních úkonů lépe než lidé. Proto je ve firmách velká poptávka po kvalifikovaných odbornících v tomto oboru. Nicméně mnoho studentů se tohoto oboru obává právě kvůli jeho komplexitě. Jak tedy snížit vstupní bariéru? Důležitým předpokladem pro výuku je, aby studenti měli přístup ke konkrétnímu zařízení, na kterém by si mohli prakticky odzkoušet svůj algoritmus pro řešení zadané úlohy, a to bez nutnosti znát úplné principy této robotické platformy. Názorné a praktické odzkoušení přinese jistě více zkušeností a větší pocit zadostiučinění než nějaké simulované prostředí. Platforma by měla být dostatečně zdokumentovaná, flexibilní a rozšiřitelná. [6]

Laboratoř robotiky na Fakultě aplikované informatiky Univerzity Tomáše Bati ve Zlíně vlastní pár exemplářů Arduino Robota, který je pro tuto úlohu přímo předurčen. Z alternativních komerčních řešení by šlo jistě také využít například LEGO Mindstorm, ArcBotics Sparki Robot nebo robot mBot.

1.3 Robotické soutěže

Fakulta aplikované informatiky na Univerzitě Tomáše Bati ve Zlíně od roku 2017 pořádá soutěž Robogames. Soutěž je otevřená prakticky pro všechny věkové kategorie, od žáků základních škol, po vysokoškolské studenty a dospělé. Zpravidla se soutěží v pěti disciplínách: Robosumo, Sledování čáry, Bludiště, Robot uklízeč, Převozník nákladu. [7]

Proto je žádoucí, aby student po absolvování laboratorních cvičení získal takové praktické a teoretické znalosti, a aby byl schopen se této soutěže, nebo jí podobné, zúčastnit.

1.4 Sledování čáry

Sledování čáry je jeden z klasických úkolů pro mobilního robota. Robot pomocí senzorů zjišťuje svou polohu vůči čáře a snaží se udržet svou pozici nad čárou. Čára musí být

dostatečně kontrastní. Nejčastěji se používá černá čára na bílém podkladu nebo naopak. V průběhu jízdy tak musí robot upravovat směr svého pohybu. Pokud je čára pod robotem vlevo od středu, je to signál pro zatočení vlevo. Pokud je čára vpravo od středu, robot musí zatočit vpravo. Je-li čára pod středem robota, může robot pokračovat rovně. Aby robot tohoto byl schopen, musí mít vhodné a vhodně rozmístěné senzory (roli hraje i šíře sledované čáry). Pro detekci se zpravidla využívají reflexivní optické senzory nebo kamery [8]. Jestliže cílem je i plynulá a rychlá jízda, pak je nutné robot vhodně řídit. Tady lze s úspěchem použít například PID (Proporčně Integračně Derivační) regulátor.

2 PLATFORMA ARDUINO

Arduino je otevřená (open-source) platforma, která v sobě spojuje snadno použitelný prototypovací hardware a software. Díky své jednoduchosti, uživatelské dostupnosti a přijatelné ceně hardware je Arduino jednou z nejpoužívanějších a nejrozšířenějších platforem tohoto typu. Navzdory své jednoduchosti je tato platforma dostatečně flexibilní i pro pokročilé uživatele. To i díky tomu, že Arduino hardware není striktně vázán na Arduino software.

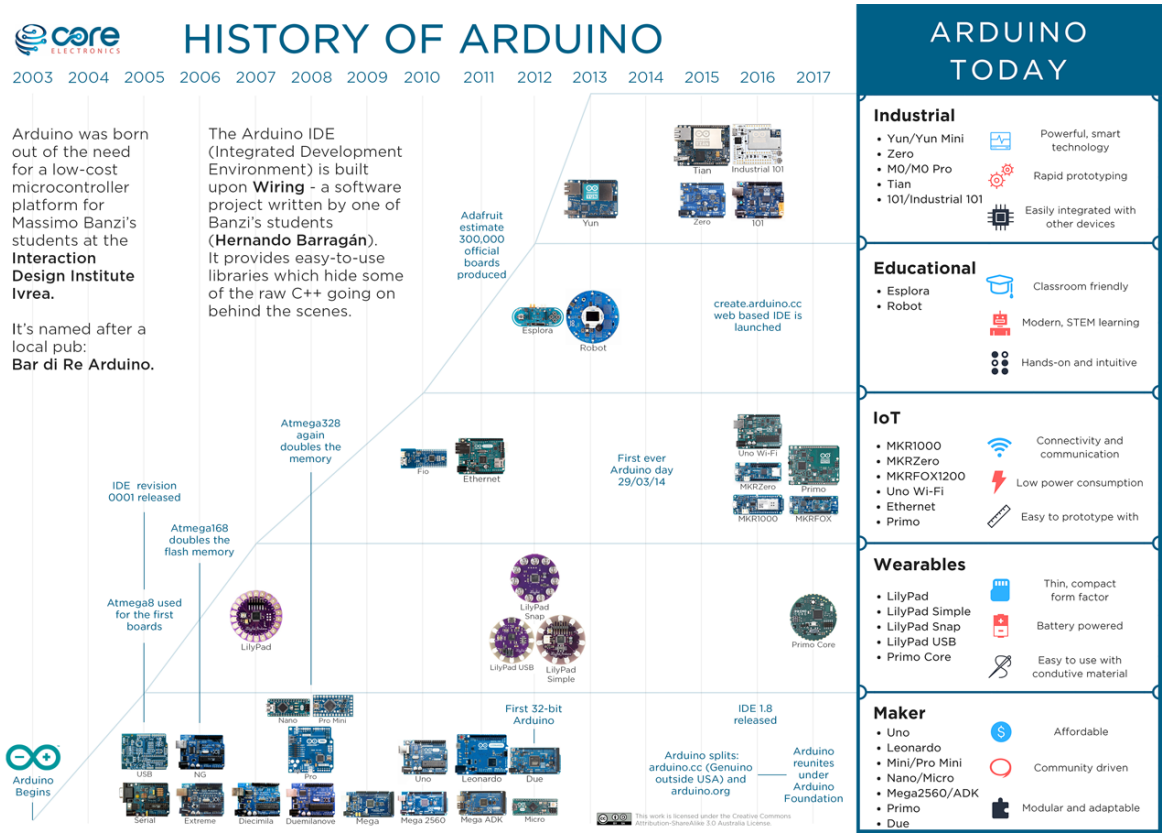
Samotné Arduino je vytvořeno ze spousty jiných open-source komponent, které spojuje v jednoduše použitelný a funkční celek. [9]

Obrázek 1: Open-source projekty, které platforma Arduino využívá. [9]

2.1 Stručná historie

Za počátek projektu Arduino bychom mohli označit rok 2005, kdy tým kolem Massima Banzi a Davida Cuartiellese, v italském Interaction Design Institute ve městě Ivrea, představil levný vývojový set pro studenty, jako náhradu drahé desky Basic Stamp. Díky tomu, že tato vývojová deska měla mezi studenty velký úspěch, rozhodli se tvůrci poskytnout tuto platformu celému světu [10] [11]. Od té doby se komunita kolem této platformy rozrůstala a rozrůstala. Vznikaly a stále vznikají nové vývojové desky. Jejich stručný výčet nalezneme na oficiálních stránkách [12].

Díky otevřenému hardware kromě oficiálních vývojových desek vznikla celá řada desek neoficiálních.



Obrázek 2: Historie vývoje Arduino desek do r. 2017. [13]

3 ARDUINO IDE

Arduino svému úspěchu nevděčí jenom zajímavému hardwaru, ale především jednoduchému vývojovému prostředí. Arduino IDE (Integrated Development Environment) je velice intuitivní, obsahuje primitivní textový editor pro psaní kódu a tlačítka pro kompilaci a nahrání programu na Arduino desku. Neklade tedy nijak zvlášť velké vstupní nároky na pochopení tohoto prostředí, takže i naprostý začátečník si velmi rychle tento nástroj osvojí a je schopen psát vlastní programy.

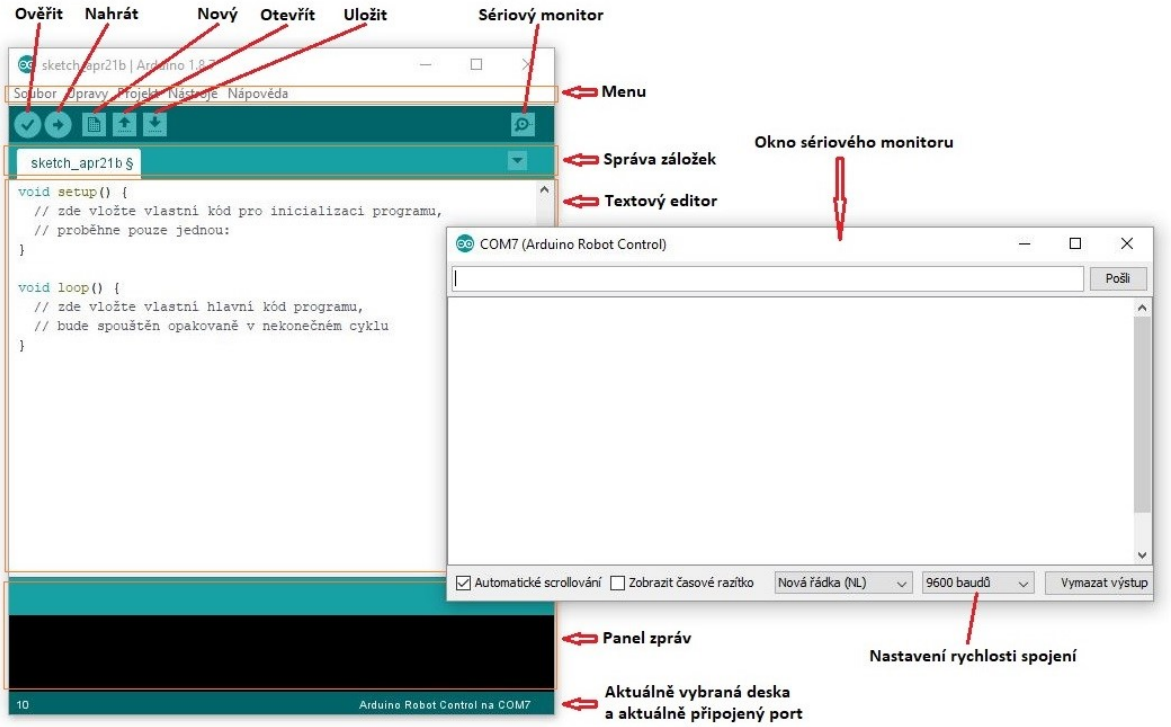
3.1 Vývojové prostředí

Arduino IDE je open-source software, je napsán v jazyce Java, a je tudíž multiplatformní. Běží na operačních systémech Windows, Mac OS X i Linux. Momentálně je Arduino IDE dostupné ve verzi 1.8.9 a dokonce je dostupné i v online verzi tzv. Arduino Web Editor.

Integrovaný textový editor obsahuje jen nejzákladnější funkce, jako je vyhledávání, nahrazování, kopírování, vkládání textu. Aby byl program co nejvíce čitelný, je důležitou vlastností editoru zvýraznění syntaxe. Co ovšem editor postrádá, je jakákoli vyšší funkce, na které jsou programátoři zvyklí z jiných vývojových prostředí, jako je například tzv. intellisense, tj. nápověda v podobě doplňování kódu nebo použitých názvů proměnných a funkcí. Kromě textového editoru se v hlavním okně IDE nachází i panel pro zprávy, které dávají zpětnou vazbu o procesu kompilace, procesu nasazení, případných chybách a upozorněních.

Ve vývojovém prostředí dále nalezneme sériový monitor, pomocí kterého se přes rozhraní UART dají zasílat a přijímat zprávy mezi počítačem a připojenou deskou s mikropočítačem.

Detailnější popis prostředí viz [14].



Obrázek 3: Popis prostředí Arduino IDE.

3.2 Wiring

Wiring je open-source framework pro programování mikropočítačů, který Adruino IDE využívá. Základy tohoto frameworku vytvořil Hernando Barragán v roce 2003 jako svou diplomovou práci na Interaction Design Institute Ivrea v Itálii. V současné době je vyvíjen na Universidad de Los Andes v Bogotě v Kolumbii. Wiring technicky navazuje na projekt Processing, za kterým stojí Casey Reas a Ben Fry z Aesthetics and Computation Group z MIT Media Lab. [15]

Programovacím jazykem v tomto frameworku je C++, který je rozšířen o řadu užitečných příkazů a funkcí (viz: [16]). Proto je programovací jazyk, v takto vzniklém frameworku, v mnoha publikacích kvůli své komplexnosti právem označován jako programovací jazyk Wiring [11].

Framework Wiring byl navržen tak, aby co nejvíce usnadnil vytváření sofistikovaných řešení. Asi nejdůležitějším aspektem je abstrahování jednotlivých pinů mikropočítače jako čísel, čímž došlo k výraznému zobecnění. Syntaxe tak nemusí být vázána na konkrétní hardwarovou platformu. Toho využívají funkce `pinMode()`, `digitalRead()`, `digitalWrite()`, `analogRead()`, `analogWrite()`, etc.

Pojmenování a syntaxe jednotlivých funkcí je výsledkem dlouhého procesu návrhu, který zahrnoval i uživatelské testování se studenty, pro které byl v počátku framework určen. [17]

3.3 Sketch

Projekty jsou organizovány v jednoduché struktuře zvané Sketch, což bychom do češtiny přeložili jako náčrtník. Cílem takto organizovaného projektu je co největší zjednodušení, aby proces kódování byl jednoduchý jako skicování. Aby programátor nemusel projekt složitě nastavovat, složitě inicializovat hardware apod.

Sketch má strukturu složky, která obsahuje stejně pojmenovaný soubor s koncovkou ino. Tento soubor by měl začínat komentářem, který popisuje daný projekt. Komentář ale není povinný, slouží převážně pro programátora, aby se v daném projektu lépe zorientoval a jednoduše zjistil, co dělá. Povinná součást tohoto souboru je funkce `setup()` a `loop()`. Bez těchto funkcí nepůjde projekt zkompilovat a Arduino IDE vypíše chyby: *undefined reference to `setup`* nebo *undefined reference to `loop`*. [18]

Na rozdíl od C++ byste tu marně hledali funkci `main()`. Ta je skryta v samotném frameworku a vypadá cca následovně:

```
Int main(void)
{
  init();
  initVariant();
  setup();
  for (;;) {
    loop();
    if (serialEventRun) serialEventRun();
  }
  return 0;
}
```

Funkce `init()` je také součástí frameworku a slouží k nastavení všech důležitých věcí, které je nutné nastavit. Mimo jiné nastavuje potřebné časovače a jejich frekvence. [19]

3.4 Základní struktura programu

Základní struktura programu je tedy poměrně jednoduchá. Má dvě části, a to funkci `setup()` a funkci `loop()`. Tyto funkce se automaticky vytvoří při založení nového projektu.

```
void setup() {
  // zde vložte vlastní kód pro inicializaci programu,
  // proběhne pouze jednou.
}

void loop() {
  // zde vložte vlastní hlavní kód programu,
  // bude spouštěn opakovaně v nekonečném cyklu.
}
```

Z ukázky implementace funkce `main()` můžeme odvodit jejich význam.

3.4.1 Funkce setup()

Funkce `setup()` je inicializační a provádí se jen jednou na začátku programu při spuštění nebo po restartu. Používá se k nastavení proměnných, nastavení režimu pinů, spuštění knihoven, nastavení sériové komunikace... Jak bylo řečeno výše, tato funkce musí být v programu obsažena vždy, i když žádný kód neobsahuje.

3.4.2 Funkce loop()

Funkce `loop()` je výkonná a provádí se stále dokola v nekonečné smyčce, dokud není deska odpojena od energie, nebo nedojde k restartu. Tato funkce obsahuje hlavní tělo programu. Ve většině implementací tak v jednotlivých iteracích zpracovává vstupy z periferií a adekvátně na ně reaguje.

3.5 První program

První program, který každý při prvním setkání s Arduino napíše, bude pravděpodobně rozblikání LED diody, která je na většině vývojových desek Arduino již vestavěná.

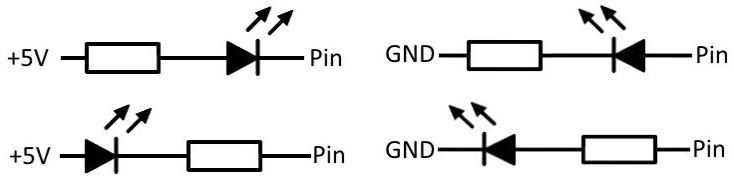
```
/**
 * Blikání
 *
 * Na jednu sekundu rozsviť LED diodu,
 * poté ji zhasni a počkej další sekundu,
 * toto stále opakuj.
 */

// funkce setup se provede právě jednou po restartu
// nebo zapnutí desky
```

```
void setup() {
  // inicializace výstupního digitálního pinu LED_PIN_NUMBER
  pinMode(LED_PIN_NUMBER, OUTPUT);
}

// funkce loop se provádí stále dokola
void loop() {
  // zapnutí LED (HIGH je vysoká úroveň napětí)
  digitalWrite(LED_PIN_NUMBER, HIGH);

  // počkej sekundu
  delay(1000);

  // vypnutí LED nastavením nízké úrovně napětí LOW
  digitalWrite(LED_PIN_NUMBER, LOW);

  // počkej sekundu
  delay(1000);
}
```

Tento program najdeme i v oficiální dokumentaci [20]. Tímto základním programem by měl začít každý student. Pomocí tohoto programu se seznámí hlavně s Arduino IDE, kompilací a nahráním programu na vývojovou desku. V případě úspěchu je efekt ihned vidět v podobě blikající LED diody. Tímto se mimo jiné otestuje, že vše je správně nastaveno a odladěno. Student se poté může pustit do složitějších úkolů bez obav, že někde něco základního přehlédl. [8] [10] [21] [22]

3.5.1 Připojení LED diody

Při připojení externí LED diody je studenta dobré poučit o vhodnosti připojení rezistoru o odpovídajícím odporu.



Obrázek 4: Varianty zapojení LED diody.

Pokud by došlo k zapojení LED diody bez rezistoru, prochází obvodem elektrický proud, který vypočteme podle Ohmova zákona $I = \frac{U}{R}$ a s pomocí 2. Kirchhoffova zákona. Výši napětí na diodě zjistíme ze specifikace (v případě zvolené diody je to $1{,}8\ V$) vnitřní odpor diody i vedení je zanedbatelný, přepokládejme proto vnitřní odpor čipu, ke kterému je dioda

připojena cca, 30 Ω. Při zapnuté (vypnuté) LED diodě je výstupní pin přes mikročip připojen na $5\ V$. Na čipu tedy naměříme napětí $5\ V - \ 1{,}8\ V$, tj. $3{,}2\ V$. Z toho nám plyne, že elektrický proud tekoucí přes diodu a čip má velikost něco kolem $0{,}11\ A$. Dle specifikace čipu (např. ATmega32u4) proud na jeden výstupní pin může být maximálně $0{,}04\ A$ a maximální proud tekoucí přes diodu dokonce jen $0{,}02\ A$, což toto zapojení bez rezistoru značně převyšuje. Při dlouhodobém zapojení tedy riskujeme zničení nejen připojené LED diody, ale i samotného mikropočítače.

Velikost odporu rezistoru, který musíme použít, zjistíme obdobným jednoduchým výpočtem. Elektrické napětí na výstupním pinu je $5\ V$, napětí na zvolené červené diodě $1{,}8\ V$. Z toho nám pomocí 2. Kirchhoffova zákona plyne, že napětí na rezistoru bude $3{,}2\ V$ (vnitřní odpor čipu v tomto výpočtu zanedbáme). Jestliže tedy chceme, aby maximální proud procházející touto větví byl $0{,}02\ A$, musí být minimální odpor rezistoru dle Ohmova zákona $R = \frac{U}{I} = \frac{3{,}2\ V}{0{,}02\ A} = 160\ \Omega$. V praxi se ale častěji použije nejbližší vyšší běžně dostupný odpor $220\ \Omega$, jelikož odpor o velikosti $150\ \Omega$ by byl dle výpočtu nedostatečný.

Pokud je student již v počátku upozorněn na tyto úskalí připojování externích součástek k vývojové desce, hrozí menší riziko, že dojde ke zničení součástky či desky. [22]

4 ARDUINO ROBOT

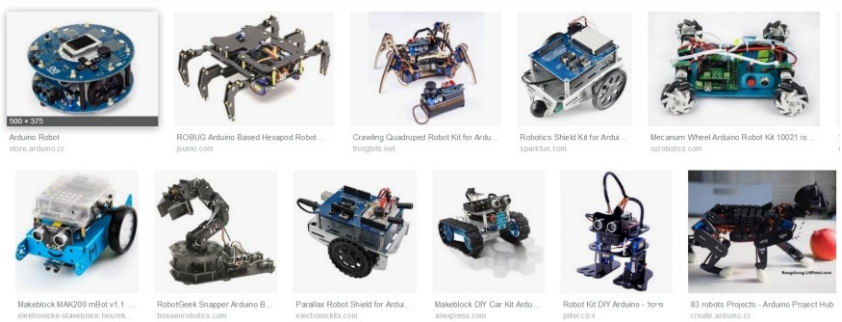
Arduino Robot je platforma na kolech, s mikroprocesorem (ATmega32u4) jak na motorové desce, tak na řídící desce. Je tedy složen ze dvou samostatných desek, které se navzájem dorozumívají pomocí sériového rozhraní.

Obrázek 5: Arduino Robot. [12]

Obě desky jsou plně programovatelné např. prostřednictvím Arduino IDE.

Robot můžeme zařadit do kategorie servisních suchozemských mobilních dvoukolových robotů.

Všechny prvky této robotické platformy (hardware, software i dokumentace) jsou volně dostupné jako open-source, jak je u Arduina dobrým zvykem. Tím pádem má i relativně dobře zpracovanou oficiální dokumentaci. Bohužel ostatní zdroje popisující práci a zkušenosti s tímto robotem lze velmi těžko dohledat. A to hlavně díky faktu, že slovní spojení „Arduino Robot“ je chápáno i obecně, pro jakéhokoli robota postaveného na kterékoli platformě Arduino.



Obrázek 6: Vyhledávání slovního spojení „Arduino Robot“. (Roboty založené na Arduino platformě.)

V oficiální programové dokumentaci nalezneme i základní příklady použití robota, které lze využít jako základ pro vytvoření laboratorních úloh. Bohužel v některých případech je nutné upravit oficiální knihovnu, jelikož obsahuje drobné chyby.

Mimo tuto dokumentaci lze nalézt několik video tutoriálů z roku 2013, které vytvořili sami tvůrci tohoto robota, ve spolupráci s RS Components. [23]

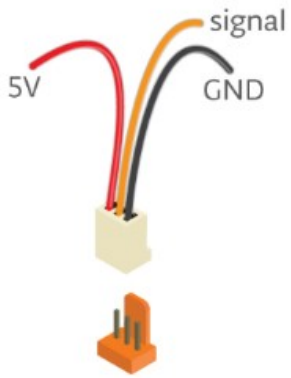
Odkaz na Arduino Robot lze nalézt i v různých publikacích týkajících se Arduino platformy jako celku. V těchto publikacích bývá Arduino Robotu věnováno poměrně málo prostoru. Krásnou výjimkou je kniha Jamese A. Langbridge [24], kde tomuto tématu věnuje celou kapitolu i s příkladem ovládání robota přes platformu Arduino Esplora.

I přes to, že platforma Arduino Robot je již označena jako „Retired“ (tzn. nebude již oficiálně rozvíjena a podporována), má ve výuce robotiky co nabídnout.

Robot byl vyvinut společně s asociací pro robotiku Complubot ze Španělska, za kterou stojí Eduardo Gallego, Ivan Gallego a Nerea de la Riva. Při vývoji tak využili dlouholeté zkušenosti v oblasti konstrukce robotů, získané na mezinárodní soutěži RoboCupJunior. Arduino Robot je tedy přímo určen jako vstupní platforma pro seznámení se s robotikou a jejími základy. [9] [24] [25]

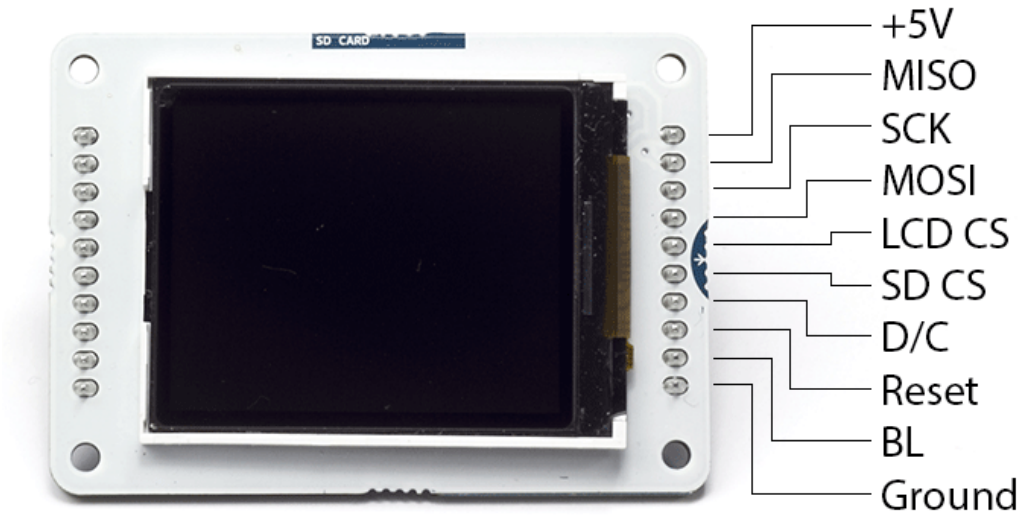
K robotu lze připojit řadu externích modulů. K tomuto účelu slouží dostupné analogové i digitální piny.

Na horní řídící desce je rozmístěno 8 vstupních konektorů „Tinkerkit“, které jsou pomocí multiplexoru připojeny k analogovému pinu mikropočítače. Další 4 tyto konektory najdeme i na motorové desce.



Obrázek 7: Příklad zapojení Tinkerkit konektoru. [26]

K robotu lze připojit také barevný TFT LCD displej s integrovanou čtečkou SD karet.



Obrázek 8: Arduino LCD displej. [12]

Další moduly lze připojit pomocí I2C rozhraní, SPI rozhraní (při nezapojení LCD modulu) nebo pomocí virtualizovaného sériového rozhraní (to hardwarové je určeno pro komunikaci s motorovou deskou).

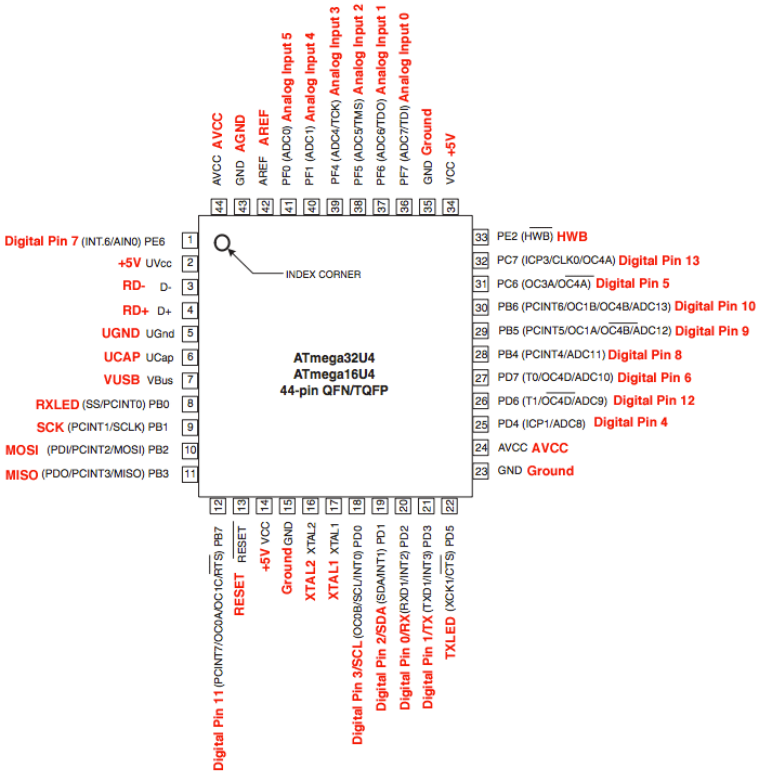
Robot mimo jiné obsahuje vestavěné IR senzory umožňující sledování čáry, digitální kompas, externí EEPROM paměť, pětitlačítkovou klávesnici, otočný potenciometr, reproduktor a další komponenty.

4.1 Mikropočítač ATmega32u4

ATmega32u4 je nízkoenergetický 8bitový mikropočítač založený na architektuře AVR, která je rozšířením RISC architektury (má tedy redukovanou instrukční sadu) s harvardskou architekturou (čili má oddělenou paměť programu od paměti dat).

Mikropočítač je označení pro jednočipový počítač, anglicky microcontroller, někdy překládáno do češtiny jako mikrokontrolér, zkratka MCU nebo µC. Oproti mikroprocesoru, který obsahuje pouze CPU, obsahuje mikropočítač další periferie, jako například paměti RAM a ROM, vstupně výstupní porty etc., integrované do jednoho čipu.

Popis jednotlivých pinů mikropočítače viz obrázek (Obrázek 9).



Obrázek 9: Rozložení pinů na čipu mikropočítače ATmega32u4.
(Červeně jsou zvýrazněny funkce Arduina.) [27]

Mikropočítač ATmega32u4 má již vestavěný USB převodník, takže při připojení k počítači přes USB rozhraní se může tvářit například jako myš nebo klávesnice.

4.2 Podobné desky

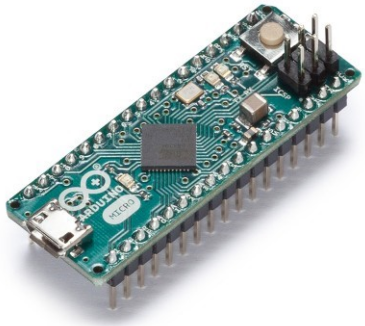
Na stejném mikroprocesoru (ATmega32u4) jsou založeny i jiné vývojové desky Arduino:

4.2.1 Arduino Leonardo

Obrázek 10: Arduino Leonardo. [12]

Arduino Leonardo designově navazuje na populární Arduino Uno, liší se použitým čipem ATmega32u4, a tím pádem i možnostmi použití. Obsahuje 16 MHz krystalový oscilátor, 20 vstupně výstupních pinů, z toho 12 lze použít jako analogové vstupy (s 10-bit A/D převodníkem) a 7 jako výstupy s možností PWM. Pracuje s napětím 5V. Každý pin může poskytovat nebo přijímat proud maximálně o výši 40mA. Piny mají vestavěné pull-up rezistory o velikosti 20-50 k Ω, které jsou ve výchozím stavu odpojené. [12]

4.2.2 Arduino Micro



Obrázek 11: Arduino Micro. [12]

Arduino Micro je jedna z nejmenších vývojových desek rodiny Arduino. Její rozměry jsou 4,8 cm na 1,77 cm. Tato vlastnost ji přímo předurčuje k integrování do různých přístrojů a předmětů. Oproti desce Arduino Leonardo má nižší spotřebu energie (kolem 29 mA, samozřejmě v závislosti na připojených periferiích) a každý pin dle specifikace může poskytovat nebo přijímat elektrický proud pouze o maximální výši 20 mA. Ostatní technické parametry této desky jsou díky osazenému mikropočítači ATmega32u4 shodné. [12]

4.2.3 Arduino Esplora

Obrázek 12: Arduino Esplora. [12]

Arduino Esplora je další z vývojových desek založených na mikropočítači ATmega32u4.

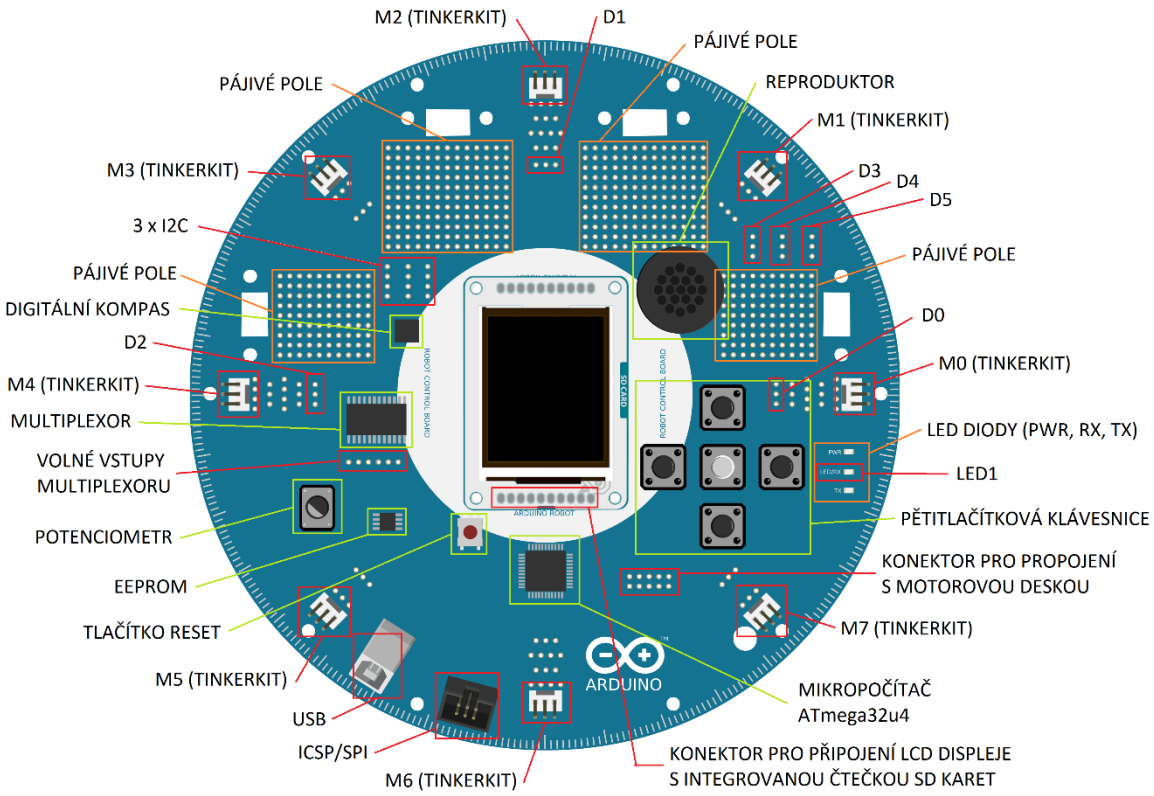
Oproti předchozím deskám se liší tím, že v sobě integruje řadu senzorů a akčních členů. Na této desce tak nalezneme joystick, čtyřtlačítkovou klávesnici, posuvný potenciometr, senzor intenzity světla, akcelerometr, mikrofon, reproduktor...

Deska umožňuje osazení barevného TFT LCD displeje s integrovanou čtečkou SD karet. Jedná se o stejný displej jako u Arduino Robot, jen je připojen na jiné piny mikropočítače.

Na desce jsou osazeny také dva vstupní a dva výstupní konektory „Tinkerkit“ pro případné připojení dalších senzorů či akčních členů. [12] [24]

4.3 Řídící deska

Horní, neboli řídící, deska Arduino Robota je bohatě osázena nejrůznějšími moduly a konektory viz Obrázek 13.

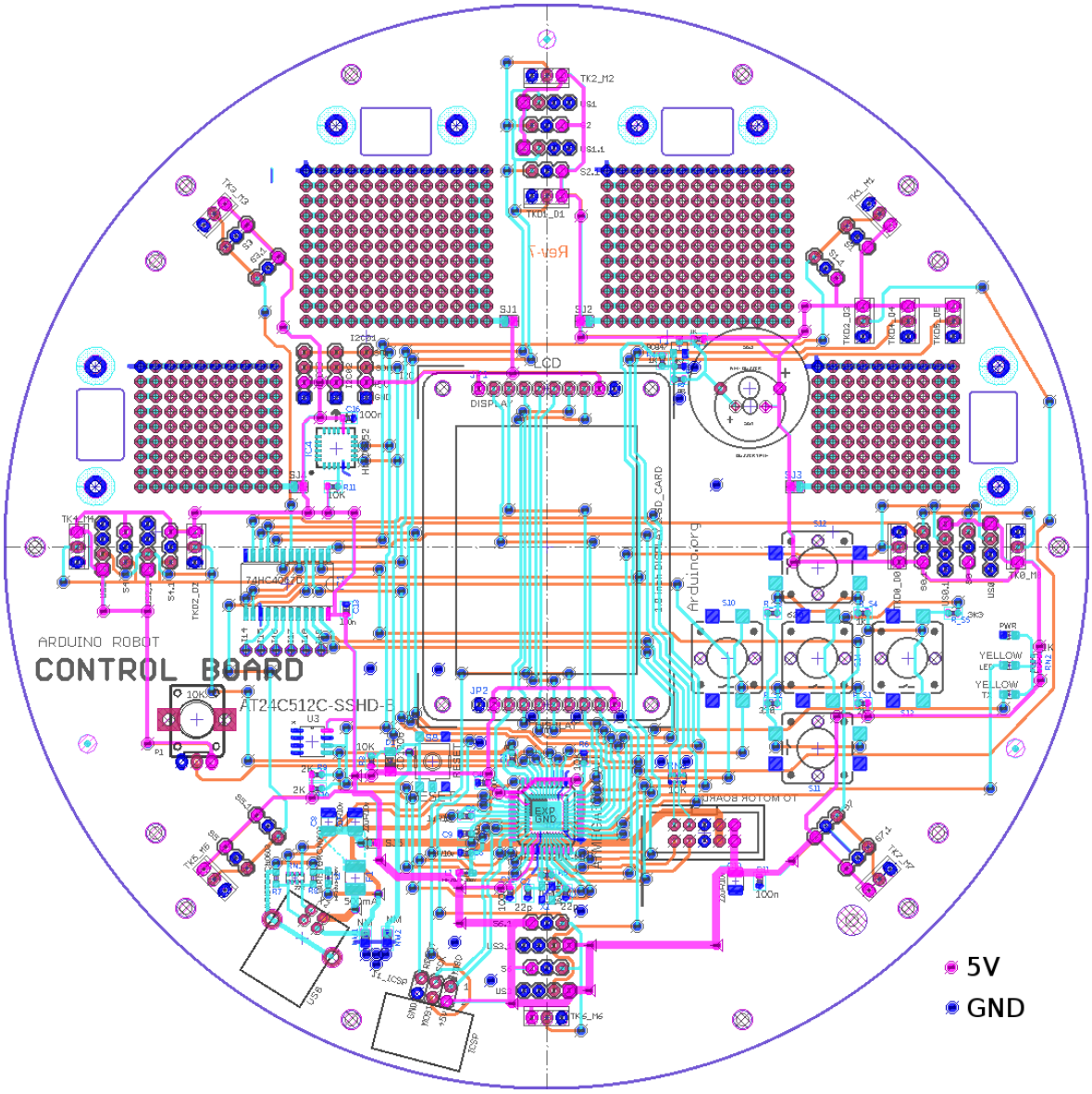


Obrázek 13: Osazení řídící desky.

TinkerKit konektory s piny M0-M7 bývají na starších deskách označeny jako TK0-TK7. Taktéž vývody D0-D5 bývají na starších deskách označeny jako TKD0-TKD5 (nebo jako TDK0-TDK5).

Deska obsahuje ještě v oficiální dokumentaci neuvedené volné vstupy multiplexoru (10-16), které lze v případě potřeby také využít. [12]

Jak již bylo řečeno, jedná se o open-source hardware, to mimo jiné znamená, že je na oficiální stránce volně dostupné i schéma desky plošných spojů s rozvržením zapojení jednotlivých součástek. Toto schéma lze prohlížet pomoci aplikace Eagle, která je v omezené základní verzi pro nadšence zcela zdarma. Tato aplikace může studentovi pomoci se na řídící desce Arduino Robota zorientovat.



Obrázek 14: Zobrazení horní desky plošných spojů v aplikaci Eagle. (Obrázek je pro přehlednost invertovaný a se zvýrazněnými 5V a GND spoji/piny.)

4.4 Knihovna ArduinoRobot.h

Knihovna ArduinoRobot.h je určena pro horní řídící desku Arduino Robota. Obsahuje řadu funkcí pro práci s vestavěnými komponentami, i funkce pro komunikaci s dolní motorovou deskou. Je součástí Arduino IDE od verze 1.0.5. Knihovna také umožňuje ovládat prvky na motorové desce, se kterou komunikuje přes sériové rozhraní UART. V knihovně je již vytvořen objekt Robot třídy RobotControl, tudíž jej již není nutné vytvářet. Ve funkci setup tedy stačí objekt Robot pouze inicializovat zavoláním metody `Robot.begin();`

popřípadě inicializovat práci s konkrétním rozhraním funkcemi `Robot.beginSpeaker();`, `Robot.beginTFT();` či `Robot.beginSD();`

Tabulka 1: Metody třídy RobotControl z knihovny ArduinoRobot.h popsané v oficiální dokumentaci.

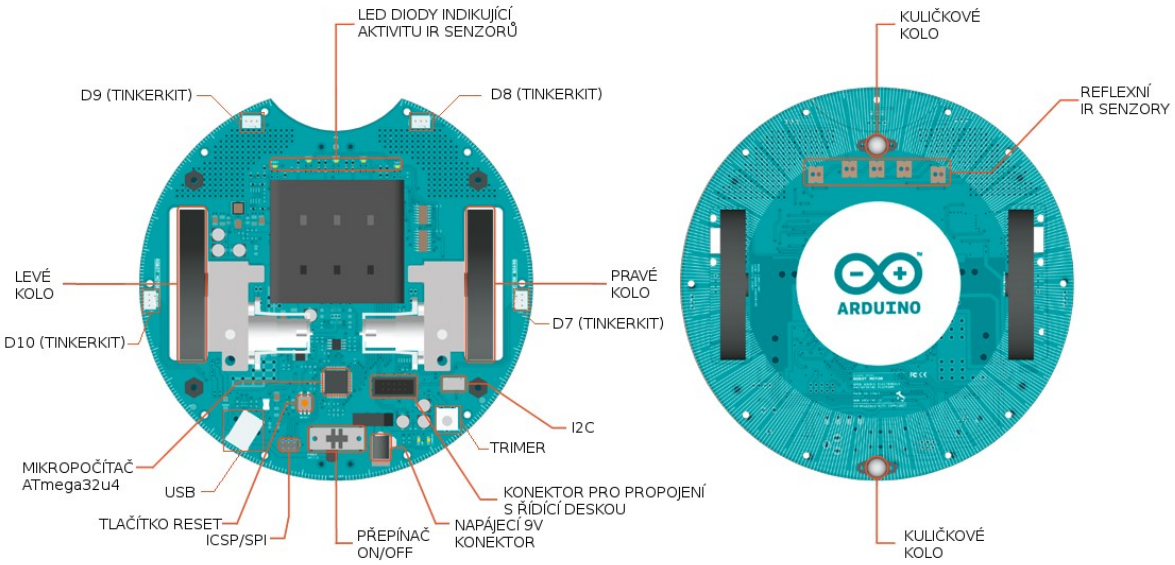
Název funkce	Popis činnosti
`RobotControl`	Konstruktor
`begin()`	Inicializace robota
`setMode()`	Změna pracovního režimu robota (simple, line follow)
`pauseMode()`	Pozastavení nebo obnovení režimově specifické činnosti
`isActionDone()`	Kontrola, zdali je ukončena aktuální činnost motorové desky v daném režimu
`lineFollowConfig()`	Nastavení parametrů pro režim sledování čáry
`digitalRead()`	Čtení digitální hodnoty definovaného portu robota: M0-M6, D0-D5, D7-D10
`digitalWrite()`	Zápis digitální hodnoty na definovaný port robota D7-D10, D0-D5, LED1
`analogRead()`	Čtení analogové hodnoty jako 10-bitové hodnoty definovaného portu robota: M0-M6, D0-D5, D7-D10
`analogWrite()`	Zápis analogové hodnoty jako PWM (8-bit) specifikovaného portu robota: D4 (použití naruší funkci multiplexoru)
`updateIR()`	Čtení hodnoty 5 IR senzorů na spodní straně robota a uložení hodnot do pole
`knobRead()`	Získání analogové hodnoty jako 10-bitové hodnoty ze zabudovaného potenciometru
`compassRead()`	Získání aktuální hodnoty směru ze zabudovaného kompasu
`keyboardRead()`	Detekce stisknutí tlačítek na řídicí desce
`waitContinue()`	Pozastaví aktuální program a čeká na stisknutí jednoho z pěti tlačítek
`motorsWrite()`	Řízení rychlosti a směru motorů
`motorsStop()`	Zastavení obou motorů robota
`turn()`	Natočení robota o daný úhel vzhledem k aktuální orientaci
`pointTo()`	Natočení robota v daném směru (absolutně)
`beginSpeaker()`	Inicializace reproduktoru a zvukové knihovny.
`playMelody()`	Přehrání melodie zadané ve formě řetězce not
`beep()`	Zvuková signalizace
`playFile()`	Přehrání souboru .sqm s hudbu uloženou na SD kartě
`tuneWrite()`	Změna výšky zvuku (při přehrávání hudebního souboru)
`tempoWrite()`	Změna tempa nebo rychlosti přehrávání zvukového souboru
`beginTFT()`	Inicializace TFT modulu
`text()`	Zápis textu na připojený TFT displej
`drawBMP()`	Zobrazení bmp souboru na připojeném displeji
`debugPrint()`	Tisk hodnoty na displeji
`clearScreen()`	Vyplnění obrazovky barvou pozadí (smazání displeje)

`displayLogos()`	Pomocná funkce k zobrazení loga na připojeném displeji
`drawCompass()`	Grafické zobrazení kompasu na displeji
`beginSD()`	Inicializace SD karty
`userNameRead()`	Načtení uživatelského jména z paměti EEPROM
`userNameWrite()`	Zápis uživatelského jména z paměti EEPROM
`robotNameRead()`	Načtení jména robota z paměti EEPROM
`robotNameWrite()`	Zápis jména robota do paměti EEPROM
`cityNameRead()`	Načtení názvu města z paměti EEPROM
`cityNameWrite()`	Zápis názvu města do paměti EEPROM
`countryNameRead()`	Načtení názvu státu z paměti EEPROM
`countryNameWrite()`	Zápis názvu státu do paměti EEPROM

4.5 Motorová deska

Jak sám název napovídá, motorová deska Arduino Robota je primárně určena k řízení vestavěných motorů, a tím pádem k řízení pohybu robota.

Při zapojeném USB jsou motory z bezpečnostních důvodů odpojeny. Pro funkčnost je nutné Arduino Robot odpojit a napájet z baterií. Patice baterií pro 4 baterie typu AA je osazena přímo na desce.



Obrázek 15: Osazení motorové desky.

4.5.1 Diferenciální řízení

Pohyb je založen na tzv. diferenciálním řízení, tj. směr a rychlost pohybu je dána rychlostí otáčení jednotlivých kol. Pokud rychlost otáčení jednotlivých kol je různá, jedno kolo ujede

větší vzdálenost než druhé, a to způsobí zatočení robota ve směru pomaleji se otáčejícího kola. Např. pokud pravé kolo se otáčí rychleji než levé, robot zatočí vlevo. Robot se tak pohybuje po kružnici o poloměru, který je dán vztahem $r = \frac{1}{2}b(v_L + v_R)/(v_L - v_R)$, kde b je rozchod kol a v_L je rychlost levého a v_R pravého kola. Pokud je rychlost otáčení kol shodná a ve stejném směru robot jede přímým směrem. Pokud je rychlost otáčení kol shodná, ale každé kolo se otáčí jiným směrem, robot se točí kolem svého středu. Atd. [1] [28]

4.6 Knihovna ArduinoRobotMotorBoard.h

Knihovna ArduinoRobotMotorBoard.h je určena pro spodní motorovou desku Arduino Robota. Je koncipována jako „stavový automat“, což znamená, že přijímá a plní příkazy zaslané z horní řídící desky přes sériové rozhraní UART. Samotný program pro motorovou desku je následující:

```
/**
 * Jádro motorové desky
 *
 * Tento kód je určen pro motorovou desku Arduino Robota.
 * Jedná se o základní firmware motorové desky.
 * Nahrajte tento program, vždy když potřebujete vrátit
 * naprogramování motorové desky do výchozího stavu.
 */

#include <ArduinoRobotMotorBoard.h>

void setup() {
  RobotMotor.begin();
}

void loop() {
  // obsluha přijatých příkazů
  RobotMotor.parseCommand();
  RobotMotor.process();
}
```

V případě, že je nutné knihovnu rozšířit o zpracování dalších příkazů, lze postupovat podle návodu, který nalezneme na stránce [29].

Tabulka 2: Metody třídy RobotMotor knihovny ArduinoRobotMotorBoard.h popsané v oficiální dokumentaci.

Název funkce	Popis činnosti
`RobotMotor`	Konstruktor
`begin()`	Inicializace motorové desky
`process()`	Provedení různých činností v závislosti na režimu robota
`parseCommand()`	Příjem příkazů od řídicí desky a jejich provedení
`motorsWrite()`	Ovládání směru a rychlosti kol robota z motorové desky
`IRread()`	Čtení hodnoty specifikovaného IR senzoru umístěného na spodní straně motorové desky

5 ROZHRANÍ

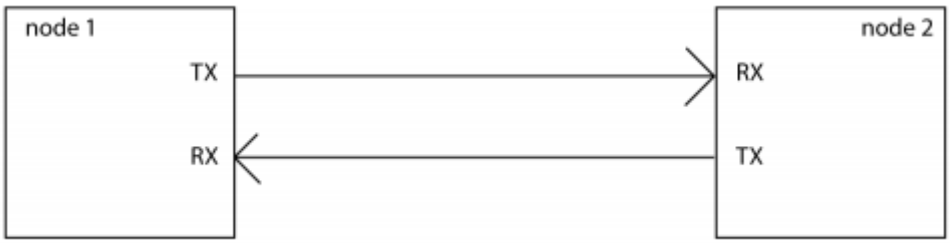
Arduino Robot může komunikovat s okolím pomocí sériového rozhraní UART, rozhraní I2C a rozhraní SPI. Ke všem těmto rozhraním existují v Arduino IDE kvalitní knihovny. Student se tedy nemusí do detailu zaobírat danou specifikací a přímo použít vhodnou knihovnu. Nicméně je dobré mít alespoň povědomí, jak které rozhraní funguje.

5.1 UART

UART (Universal Asynchronous Receiver/Transmiter) neboli univerzální asynchronní přijímač/vysílač je nízkoúrovňový komunikační protokol typu point to point (komunikují spolu právě dvě zařízení).

Jedná se o sériovou komunikaci, přenos dat po komunikačním kanálu probíhá sekvenčně po jednotlivých bitech.

UART je jednosměrný protokol který pro přenos využívá pouze jeden vodič. Pro obousměrnou komunikaci je nutné využít dvou kanálů.



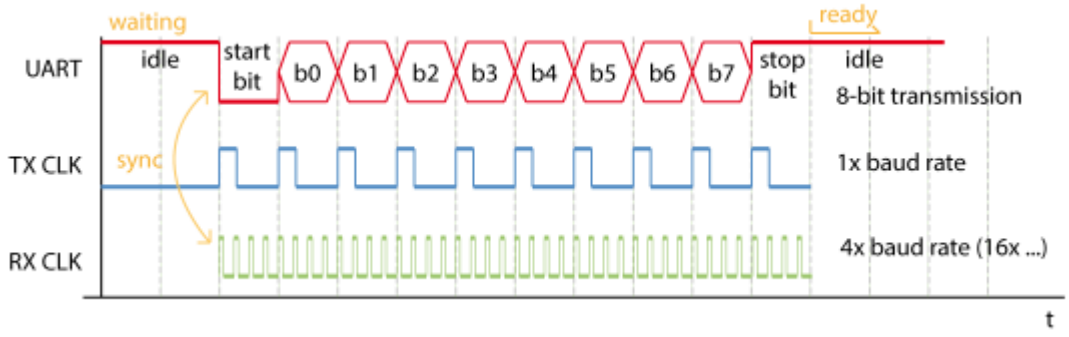
Obrázek 16: Obousměrná komunikace – schéma propojení pinů. [30]

Komunikace je asynchronní, to znamená, že není synchronizovaná hodinovým signálem a nevyžaduje tak spojení pomocí nějakého dalšího vodiče. Zařízení se tedy musí dopředu nastavit na stejnou přenosovou rychlost (baud-rate).

Jednotka přenosové rychlosti je baud (Bd), udává počet přenesených impulsů za sekundu (rozsah přenosové rychlosti je 1 200 až 250 000 impulsů za sekundu).

Schéma komunikace je následující: zařízení, které vysílá zprávu, se pomocí TX pinu připojí na RX pin zařízení, které zprávu přijímá. TX je označení pro vysílač (Transmitter) a RX je označení pro přijímač (Receiver). Vysílač nastaví napěťovou úroveň na logickou 1, tím si přijímač ověří, že je spojení živé. Pokud vysílač chce poslat data po tomto rozhraní, musí napřed odvysílat tzv. start bit (logickou nulu), tím dá druhé straně najevo, že se chystá vysílat data. Poté odvysílá přesně stanovený počet bitů (podle společného dohodnutého nastavení,

většinou 4 nebo 8 bitů) od nejméně významného bitu po nejvýznamnější bit. Poté odvysílá paritní bit (také podle nastavení), komunikaci ukončí stop bitem (log 1) po dobu 1 – 2 dob (délka je dána také společným nastavením). Přijímač většinou pracuje s hodinovým signálem, který je oproti přenosové rychlosti 4 až 16násobný a je synchronizován se sestupnou hranou vysílaného start bitu. Hodnota načteného bitu se poté určí průměrem načtených hodnot. [10] [30] [31]



Obrázek 17: Příklad časování přenosu pomocí protokolu UART. [30]

U Arduino Robota je tento typ spojení použit pro komunikaci mezi spodní a horní deskou, využívá k tomu hardwarovou podporu (ATmega32U4). Pokud bychom přes tento protokol chtěli připojit další zařízení, museli bychom využít nějakou softwarovou knihovnu, např. SoftwareSerial, NeoSWSerial, AltSoftSerial. (Z uvedených knihoven se mi podařilo využít a zprovoznit pouze knihovnu AltSoftSerial, za cenu odstavení některých vestavěných modulů řídící desky.)

Příklad použití:

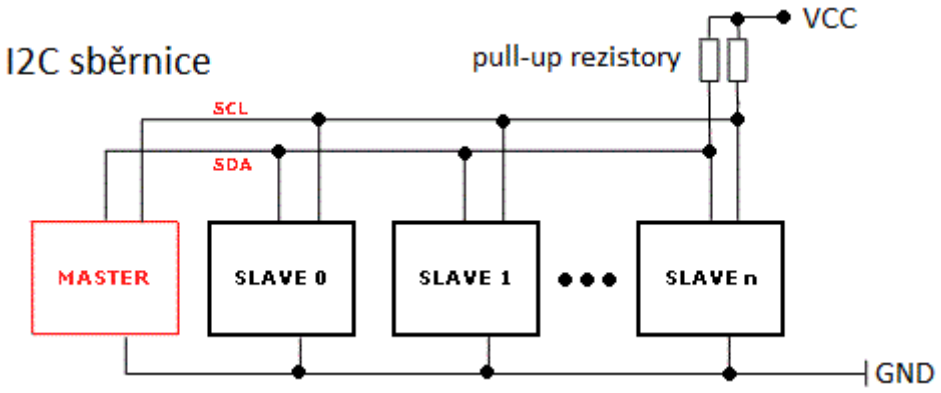
```
#include <AltSoftSerial.h>

AltSoftSerial altSerial;

void setup() {
  // otevře sériový port a nastaví přenosovou rychlost na 9600 bps
  Serial.begin(9600);
  Serial.println("AltSoftSerial Test");
  // otevře virtualizovaný sériový
  altSerial.begin(9600);
  altSerial.println("Hello World");
}
```

```
void loop() {
  char c;

  if (Serial.available()) {
    c = Serial.read();
    altSerial.print(c);
  }
  if (altSerial.available()) {
    c = altSerial.read();
    Serial.print(c);
  }
}
```

5.2 I2C / TWI

Sběrnice I2C (Internal Integrated Circuit) je dvouvodičové synchronní propojení mezi jedním zařízením (Master) a jedním, nebo několika zařízeními (Slaves). V režimu multimaster může být na sběrnici připojeno více zařízení master, ale v jednu chvíli může jako master pracovat jen jedno. Každé ze zařízení master tak musí používat nějakou metodu pro detekce kolizí a v případě kolize okamžitě ukončit vysílání.

První z vodičů slouží k obousměrnému přenosu dat (značí se SDA – serial data), druhý slouží k přenášení hodinového signálu (značí se SCL – serial clock), který vysílá zařízení typu master. Oba tyto vodiče jsou přes odpory připojené na napájecí napětí. Tím pádem i když odpojíme všechna zařízení, je na vodičích logická 1. V praxi se používá ještě třetí, zemnící kabel tak, aby všechna zařízení využívala společnou zem.



Obrázek 18: Zapojení zařízení na I2C sběrnici. [31]

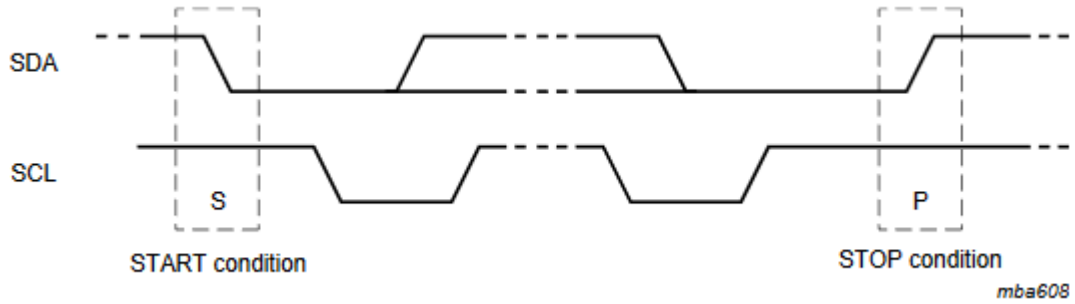
Řízení sběrnice má na starosti zařízení typu master. Zařízení typu slave nemá tedy žádnou možnost přihlásit se o slovo. Toto se v praxi řeší periodickým doptáváním. (Nebo výše zmíněným multimaster módem.)

Každé zařízení má svou specifickou adresu, ta je v základní verzi 7bitová (v rozšířené verzi 10bitová) adresa obsahuje i osmý bit, který udává, zda k zařízení přistupujeme v módu čtení či zápisu (logická 0 pro zápis, logická 1 pro čtení). Ke sběrnici lze tedy připojit až 128 zařízení (nebo 1024 v rozšířené verzi). Nicméně mnohé moduly mají pevně stanovenou adresu nebo část adresy.

Komunikace pak probíhá následovně:

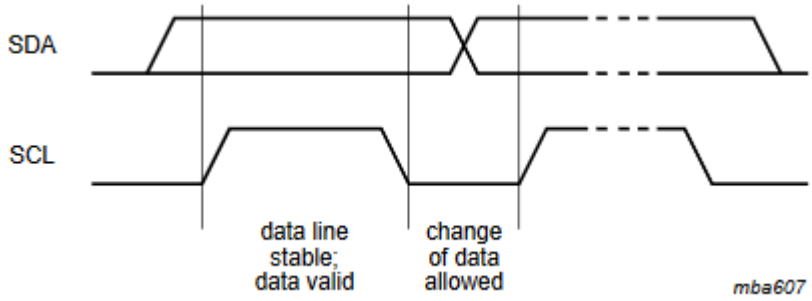
Klidový stav sběrnice je, když na obou vodičích SDA i SCL je logická 1.

Master inicializuje komunikaci zapsáním logické nuly na SDA, tím dá všem připojeným zařízením najevo, že začíná vysílat adresu.



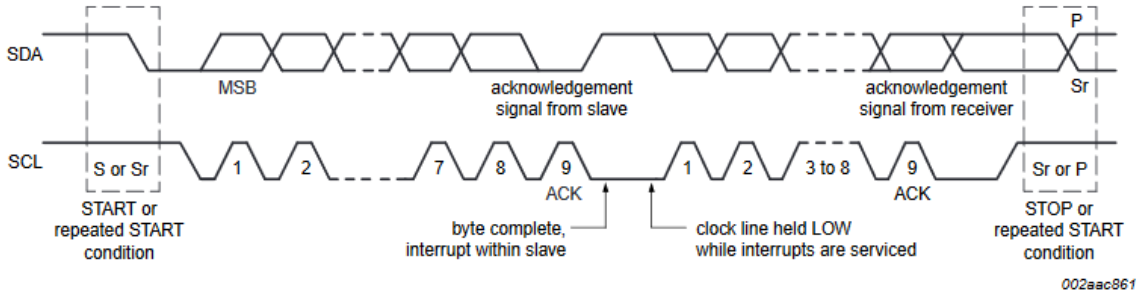
Obrázek 19: Začátek a konec komunikace přes I2C sběrnici. [32]

Data jsou na SDA platná, jen pokud na SCL je logická 1, při logické 0 na SCL může dojít ke změně hodnoty na SDA.



Obrázek 20: Detail změny úrovně na vodiči SDA. [32]

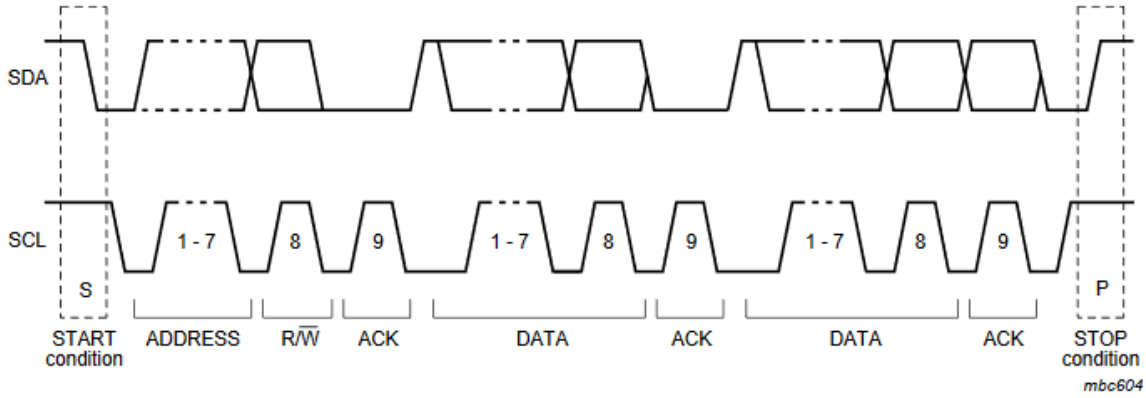
Po odvysílání 8bitové adresy, kde poslední bit určuje druh přístupu (zápis/čtení), odpoví zařízení s danou adresou zapsáním logické 0 na SDA, takzvaný potvrzovací bit (ACK - acknowledge). Poté může komunikaci pozastavit zapsáním logické 0 na SCL vodiči.



Obrázek 21: Pozastavení komunikace po dobu obsloužení požadavku. [32]

Pokud se jednalo o čtení, vyšle svých 8 bitů a master potvrdí potvrzovacím bitem, nebo naopak při zápisu master vyšle svých 8 bitů a zařízení slave potvrdí přijetí.

Komunikaci ukončí master zapsáním logické jedničky na oba vodiče.



Obrázek 22: Příklad komunikace přes rozhraní I2C. [32]

Díky ochranné známce, kterou na I2C vlastní Philips, se tato sběrnice mnohdy označuje jako TWI (Two Wire Interface). Dále se můžeme setkat i s označením IIC. [31] [32]

Pro komunikaci přes toto rozhraní lze využít knihovnu Wire, která je součástí Arduino IDE.

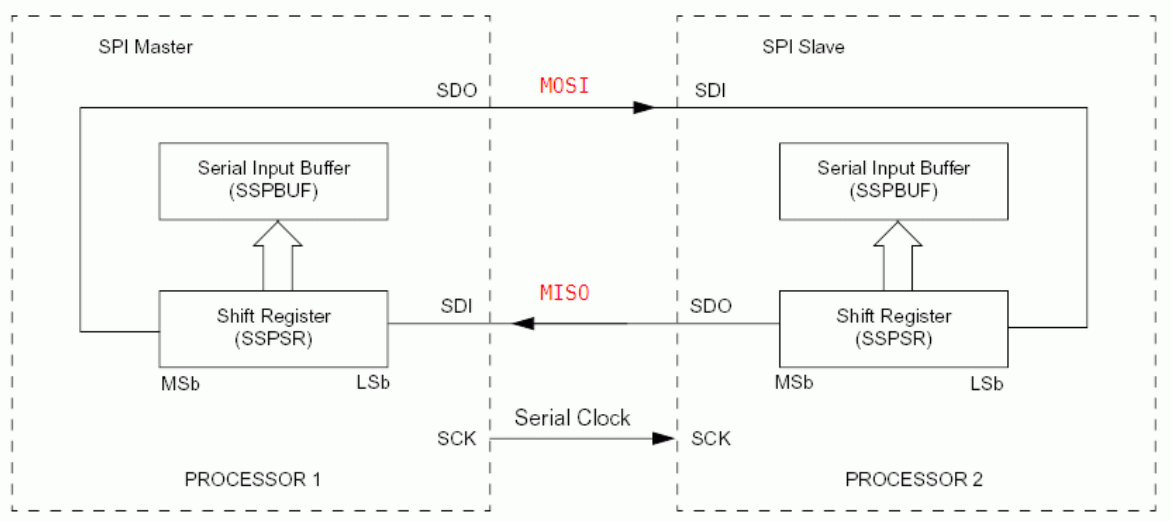
Příklad použití:

```
#include <Wire.h>

void setup()
{
  // připojit i2c rozhraní
  Wire.begin();
  Serial.begin(9600);
}

void loop()
{
  // požádat o 6 bytů od zařízení slave číslo #2
  Wire.requestFrom(2, 6);

  // dokud slave posílá data (může i méně)
  while(Wire.available())
  {
    // načti jeden byte jako znak
    char c = Wire.read();
    // kontrolní tisk
    Serial.print(c);
  }

  delay(500);
}
```

5.3 SPI (a ICSP konektor)

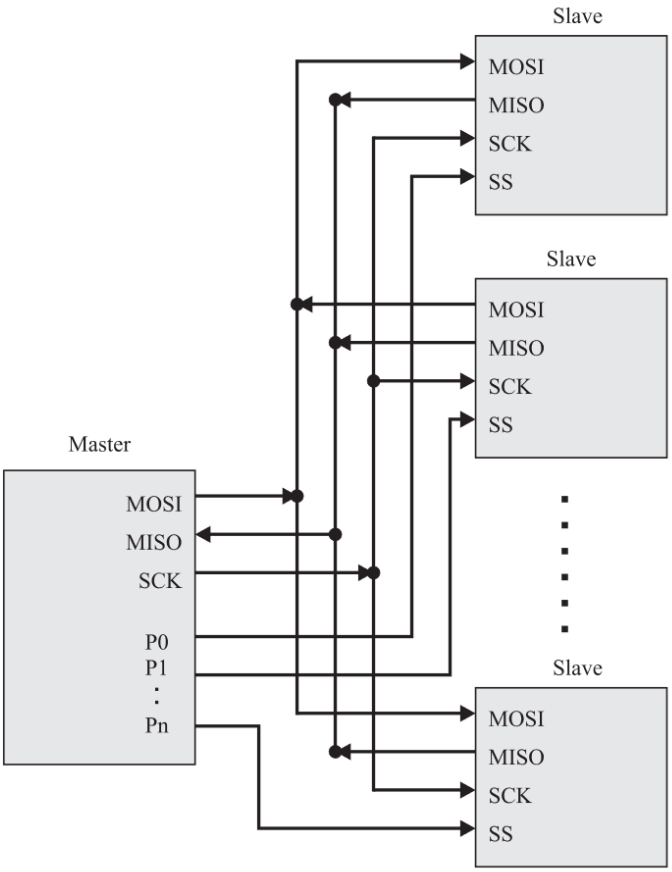
SPI (Serial Peripheral Interface) je synchronní datový protokol pro komunikaci mikropočítače (Master) s jedním nebo více zařízeními (Slave).

K propojení slouží 3 vodiče. MISO (Master In Slave Out) pro zasílání dat od připojeného zařízení na mikropočítač. MOSI (Master Out Slave In) pro zasílání dat z mikropočítače na připojené zařízení. A SCK (Serial Clock) pro přenášení hodinového signálu.



Obrázek 23: Schéma propojení pomocí rozhraní SPI. [31]

V případě připojení více zařízení lze připojit ještě SS (Slave Select) pin pro každé připojené zařízení. Pokud je logická hodnota na SS nastavena na 0, zařízení data přímá, při logické 1 komunikaci ignoruje.



Obrázek 24: Příklad zapojení více modulů typu slave na sběrnici SPI. [31]

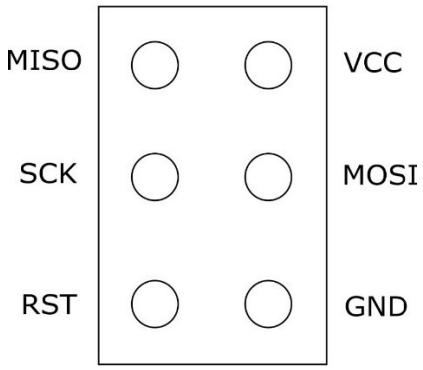
Přenosy probíhají mezi modulem master a některým ze slave modulů v obou směrech (Full Duplex), každý směr využívá jeden z vodičů MISO, MOSI. Oba obvody využívají posuvné registry, které jsou řízeny hodinovým signálem z obvodu master po vodiči SCK.

Protokol komunikace je dosti variabilní a záleží, co dané připojené zařízení podporuje. Jaká je maximální rychlost pro komunikaci. Jestli data jsou řazena od nejméně důležitého bitu nebo naopak. Jestli hodinový signál je nečinný při logické 1 nebo 0, a zda rozhraní pracuje se sestupnou či vzestupnou hranou hodinového signálu.

Rozhraní SPI je principiálně rychlejší (běžně 10 MHz až 70 Mhz) než rozhraní I2C (až 3,5 MHz), na druhou stranu ke své funkci využívá větší počet vodičů, který roste úměrně s počtem připojených zařízení. [10] [31]

Pro jednotlivá zařízení připojená přes toto rozhraní lze většinou nalézt odpovídající knihovnu. V opačném případě se nevyhneme nastudování konkrétní specifikace.

Některé desky Arduino jsou osazeny ICSP (In-Circuit Serial Programing) konektorem, který je propojen s piny SPI sběrnice. [10]



Obrázek 25: ICSP konektor sběrnice SPI. [10]

6 SENZORY

Každý robot potřebuje pro svou činnost i informace o svém okolí. Tyto informace získává právě prostřednictvím připojených senzorů. Arduino Robot disponuje dostatečným počtem vstupů i výstupů, což umožňuje připojení celé řady senzorů.

6.1 Vestavěné senzory

6.1.1 Digitální kompas HMC6352

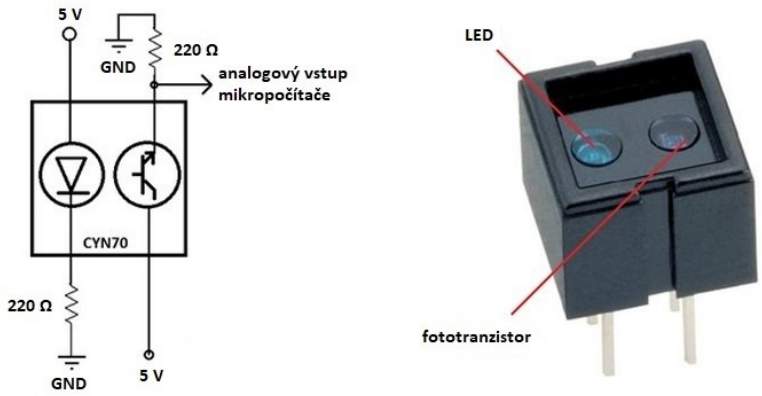
Arduino Robot má osazen digitální kompas Honeywell HMC6352 na sběrnici I2C. (Verze a typ kompasu se může lišit dle verze Arduino Robota). Různé magnetické zdroje, včetně motorů na podvozku, mohou ovlivňovat výsledky měření. Kompas lze proto i kalibrovat pomocí speciálního příkazu. [33]

Obrázek 26: Digitální kompas. [34]

Hodnotu kompasu lze zjistit pomocí funkce `Robot.compassRead()` z knihovny ArduinoRobot.h. Metoda vrací hodnotu v rozsahu 0 až 359 a udává počet stupňů ve směru hodinových ručiček, o které je robot otočen vůči severu. Hodnota 0 znamená sever, 90 východ, 180 jih a 270 západ. [35]

6.1.2 5x Reflexní optický senzor CNY 70

Pětice reflexních senzorů je umístěna na motorové desce robota a je tak předurčena k aplikacím typu sledování čáry. Senzor pracuje na principu reflexe, tzn., že měří se množství odraženého světla od podložky zpět do senzoru. Skládá se z přisvětlovací LED a fototranzistoru.



Obrázek 27: Optický senzor vhodný pro sledování čáry. [34]

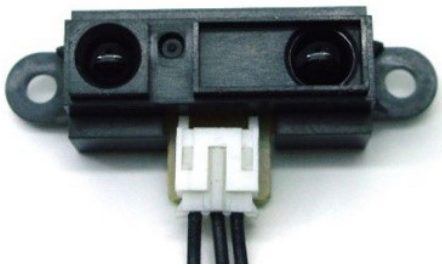
Knihovna ArduinoRobot.h obsahuje metodu `Robot.updateIR();`, která zajistí načtení aktuálních hodnot jednotlivých senzorů z motorové desky do pole `Robot.IRarray[i]`. Pole je indexováno od 0 (pravá strana robota) do 4 (levá strana robota). Na motorové desce lze k hodnotám IR senzorů přistupovat napřímo pomocí funkce `RobotMotor.IRread(i);` z knihovny ArduinoRobotMotorBoard.h, kde i určuje číslo senzoru + 1, je tedy v rozsahu 1 (pravá strana robota) až 5 (levá strana robota). Číslování tedy není konzistentní s číslováním pole `IRread` z knihovny pro řídící desku. Rozsah hodnot je dána rozsahem A/D převodníku, tedy 0–1023. [35]

6.2 Senzory pro měření vzdálenosti

Vybrané senzory jsou pouze příklady vybrané podle aktuální dostupnosti. Existuje celá řada senzorů na stejném principu, lišící se dílčími parametry.

6.2.1 Sharp GP2Y0A21YK infračervený senzor přiblížení

IR senzor vzdálenosti. Pro měření vzdálenosti reflexního objektu pomocí IR. Tento senzor je schopen měřit vzdálenost v rozsahu 10-80 cm s periodou cca 50 ms. Výstup je analogový (0-3.3 V) a nelineární v závislosti na vzdálenosti. [34]



Obrázek 28: Infračervený senzor přiblížení Sharp GP2Y0A21YK. [34]

Specifikace:

- Rozsah snímání: 10-80 cm
- Výstup: Analog (0-3,3 V)
- Vstupní napětí: 4,5-5,5 V
- Proud: 30 mA
- Cena: do 250 Kč [36]

Pro získání naměřené vzdálenosti lze využít knihovnu SharpIR.h.

6.2.2 HC-SR04 ultrazvukový měřič vzdálenosti

Tento ultrazvukový senzor vzdálenosti umožňuje měřit vzdálenost pomocí ultrazvuku s rozsahem 2-400 cm. Měření probíhá pomocí vyslání ultrazvukového impulzu, který se od překážky odrazí a vrátí zpět. Ze změřeného času se poté, dle znalosti rychlosti šíření zvuku prostředím, dopočítá konkrétní vzdálenost. [34]

Obrázek 29: ultrazvukový měřič vzdálenosti HC-SR04. [34]

Specifikace:

- Rozsah snímání: 2-400 cm
- Rozlišení: 0,3 cm
- Měřící úhel: 30 °
- Vstupní napětí: 4,5-5,5 V
- Proud: 15 mA

- Cena: do 100 Kč [36]

Pro získání naměřené vzdálenosti lze využít knihovnu NewPing.h.

6.2.3 Maxbotix LV-MaxSonar-EZ1 High Performance Sonar Module

Modul sonaru LV-MaxSonar-EZ1 měří vzdálenosti také pomocí ultrazvuku. Má rozsah snímání 15-645 cm (v závislosti na velikosti objektu, člověka detekuje do vzdálenosti cca 2 m, malé objekty do 50 cm) s periodou snímání 50 ms. Rozlišení senzoru je 1 palec, což odpovídá 2,54 cm. Výstup je buď analogový (zdrojové napětí děleno 512 na palec), nebo pomocí sériové linky (rozhraní UART). Data jsou posílána ve formátu ASCII v pořadí: velké písmeno R, tři číslice udávající vzdálenost v palcích (max 512), následováno znakem nového řádku (ASCII znak 13). Baud rate je 9600. [34]

Obrázek 30: Sonar LV-MaxSonar-EZ1. [34]

Specifikace:

- Rozsah snímání: 15-645 cm (v závislosti na velikosti objektu)
- Rozlišení: 2,5 cm
- Vstupní napětí: 2,5-5,5 V
- Proud: 3 mA
- Cena: do 600 Kč [36]

Pro získání naměřené vzdálenosti stačí načíst analogovou hodnotu a provést adekvátní výpočet. `Robot.analogRead(SELECTED_PIN)*5*254/1000;`

6.3 Další zajímavé senzory

6.3.1 3osý akcelerometr ADXL335

Akcelerometr je pohybový senzor, který měří zrychlení (i to gravitační). Lze tedy využít i jako polohové čidlo. [34]

Obrázek 31: 3osý akcelerometr ADXL335. [34]

Specifikace:

- Rozsah snímání: -3 g až 3 g
- Vstupní napětí: 3-5 V
- Výstupní napětí: < 3,3 V (0 V = -3 g; 3 V = +3 g)
- Proud: 350 μA
- Cena: do 120 Kč [36]

Hodnoty pro jednotlivé směry získáme přes analogový vstup například pomocí funkce: `Robot.analogRead(SELECTED_PIN);`

6.3.2 Kamera Pixy2

Kamera pro rozpoznávání objektů a barev v obraze. S Arduino deskou lze propojit např. pomocí ICSP konektoru a zasílat předzpracovaná data přes rozhraní SPI. Rozhraní UART a I2C jsou také podporována. Kamera Pixy2 se hodí i pro algoritmy na sledování čáry, pro které má speciální režim nastavení. [34] [37]

Obrázek 32: Pixy2 kamera. [37]

Specifikace:

- Procesor: NXP LPC4330, 204 MHz, duální jádro
- Obrazový snímač: Aptina MT9M114
- Rozlišení: 1296 × 976
- Zorné pole objektivu: 60 ° vodorovně, 40 ° svisle
- Obnovovací frekvence: 60 snímků za sekundu
- Vstupní napětí: 5 V až 10 V
- Typická spotřeba: 140 mA
- RAM: 264 kB
- Flash: 2 MB
- Cena: do 1500 Kč [34]

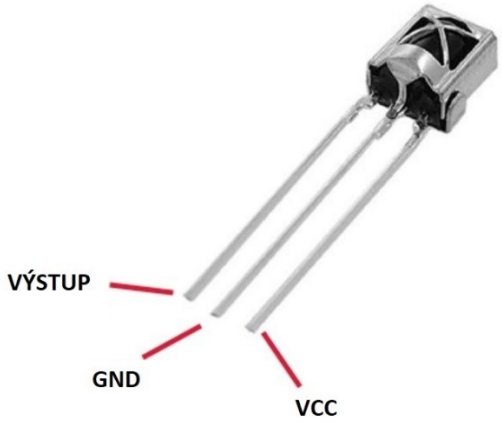
S kamerou je distribuována i knihovna Pixi2.h, díky které je práce s tímto senzorem zábavou.

7 PROSTŘEDKY PRO KOMUNIKACI S OKOLÍM

Komunikace s okolím je pro mnoho robotických implementací klíčová. Robot například komunikuje s nadřazeným systémem, kooperujícím robotem nebo operátorem. K zajištění komunikace lze využít celou řadu vhodných prostředků. Arduino Robot obsahuje vestavěné moduly pro komunikaci s operátorem, kterými jsou LCD displej, pětitlačítková klávesnice a otočný potenciometr. Vzhledem k tomu, že Arduino Robot je mobilní zařízení, jsou spíše vhodné bezdrátové prostředky komunikace (Wi-Fi, Bluetooth, radiové či infračervené spojení)

7.1 Infračervený přijímač VS1838B

Univerzální infračervený přijímač s pracovním kmitočtem 38 kHz. Jako vysílač lze využít hotový dálkový ovladač od některého domácího spotřebiče (např. TV). [36]



Obrázek 33: IR přijímač VS1838B. [36]

Specifikace:

- Napětí: 2.7 až 5.5 V,
- Proud: 0.35 mA (max 0.6 mA)
- Kmitočet: 38 kHz
- Teplotní rozsah: -20 až +65 °C.
- Příjem až na vzdálenost: 18 m
- Cena: do 10 Kč [36]

Pro práci s tímto přijímačem lze využít knihovnu IRemote.h.

7.2 433 MHz vysílač (XY-FST) + přijímač (XY-MK-5V)

Dvojice vysílače a přijímače pro radiovou komunikaci s dosahem při dobrých podmínkách až 200m. Výhodou je jednoduché zapojení a nízká cena. [36]

Obrázek 34: 433 MHz vysílač a přijímač. [36]

Specifikace:

- Přijímač XY-MK-5V:
  - Napájení: DC 5 V
  - Proud: < 6 mA
  - Frekvence: 433.92 MHZ
  - Citlivost: -100 dB
- Vysílač XY-FST:
  - Dosah: 20-200 m (záleží na napětí)
  - Napájení: 3.5-12 V
  - Pracovní mód: AM
  - Rychlost : 9,6 kB/s (teoretická hodnota)
  - Vysílací výkon: 25 mW
  - Frekvence: 433.92 MHZ
- Cena: do 50 Kč [36]

Pro komunikaci mezi těmito moduly lze využít např. knihovnu VirtualWire.h.

7.3 Bluetooth 2.0 Modul V3

Bluetooth modul může zajistit přenos sériové datové komunikace, lze spárovat např. s chytrým telefonem a pomocí vhodné aplikace robot ovládat. [38]

Obrázek 35: Bluetooth modul. [38]

Specifikace:

- Čip Bluetooth: CSR BC417143
- Bluetooth protokol: Specifikace Bluetooth v2.0 + EDR
- Provozní frekvence: 2,4 ~ 2,48 GHz nelicencované pásmo ISM
- Vysílací výkon: ≤ 4 dBm, třída 2
- Přenosová vzdálenost: 20 až 30 m ve volném prostoru
- Podporované profily: sériový port Bluetooth
- Přenosová rychlost sériového portu: 4800 ~ 1382400 / N / 8 / 1 výchozí: 9600
- Vstupní napětí: +3,5 V až + 8 V DC a 3,3 V DC / 50 mA
- Pracovní teplota: -20 až +55 °C.
- Cena: do 500 Kč [38]

Pro komunikaci s tímto modulem lze použít klasické sériové rozhraní UART např. pomocí knihovny AltSoftSerial.h.

7.4 ESP8266 WiFi modul

Tento modul vyžaduje speciální naprogramování, navíc využívá 3,3 V logiku a vyžaduje napájení 3,3 V. Připojení přímo na 5 V Arduino by modul zničilo. Je tedy nutné použít převodník napěťové úrovně. [36]

Obrázek 36: WiFi modul ESP8266. [36]

Tento modul je nutné nejdříve vhodně naprogramovat, k tomu lze využít knihovnu ESP8266WiFi.h.

II. PRAKTICKÁ ČÁST

8 LABORATORNÍ ÚLOHY

Praktická část je věnována samotným laboratorním úlohám.

Před každým laboratorním cvičením je dobré, aby vyučující na obě desky Arduino Robota nahrál základní kód viz přílohy (P I) a (P II) tak, aby studenti měli shodné výchozí podmínky.

8.1 Úloha 1: Hello World – seznámení se s Arduino IDE a robotem

Cílem první úlohy je seznámit studenta s vývojovým prostředím Arduino IDE, kompilací a nahrání programu na řídící desku Arduino Robota. V podúkolech student prozkoumá jednoduché příkladové programy, tím si na příkladech osvojí základní strukturu programu a základní funkce. Dále si vyzkouší i práci se sériovým monitorem.

Úloha 1 - Hello World – seznámení se s Arduino IDE a robotem

První a nejjednodušší program, který se při výuce programování mikropočítačů používá, je rozblikání LED diody. Je to v podstatě ekvivalent tzv. programu „Hello World“ (česky: Ahoj světe) používaného při výuce programovacích jazyků.

Zadání

Seznamte se s Arduino IDE a Arduino Robotem. Rozblikejte vestavěnou LED diodu (LED1) na kontrolní desce Arduino Robota a prozkoumejte základní jednoduché příklady v Arduino IDE. Vyzkoušejte si práci se sériovým monitorem.

Instrukce

- Podle návodu v příloze (P III) nahrajte na horní kontrolní desku Arduino Robota první program.

- Prozkoumejte a zprovozněte obdobný program (*Soubor → Příklady → 01.Basic → Blink*) z vestavěných příkladů v IDE bez pomoci knihovny ArduinoRobot.h. Opět využijte vestavěnou LED diodu (LED1).

V předchozích příkladech si všimněte funkce `delay()`. Tato funkce pozastaví program na daný počet milisekund. Funkce `delay()` je tzv. blokující funkce, čili zabrání programu dělat cokoli jiného dokud daná funkce neskončí. To ale není většinou žádoucí, jelikož program potřebuje zpracovávat i jiné vstupy a výstupy, které by jinak mohly být přehlédnuty. Obecně by se funkce `delay()` neměla používat pro zastavení delší než cca 10 ms nebo

v jednoduchých příkladech. Alternativou je např. časování pomocí funkce `millis()` či funkce `micros()`.

- Prozkoumejte a zprovozněte program (*Soubor → Příklady → 02.Digital → BlinkWithoutDelay*). Všimněte si především provedení podmínky `if (currentMillis - previousMillis >= interval)`, která je bezpečná vůči přetečení.

Arduino IDE obsahuje také i příklady určené přímo pro desku Arduino Robot Control (*Soubor → Příklady → Robot Control*).

- Prozkoumejte a zprovozněte postupně programy:

  - *Soubor → Příklady → Robot Control → learn → Beep*

  - *Soubor → Příklady → Robot Control → learn → LCDWriteText*

  - *Soubor → Příklady → Robot Control → learn → Compass*

  - *Soubor → Příklady → Robot Control → learn → keyboardTest*

Před nahráním každého programu si jej nejprve prostudujte.

U programu *Compas* a *keyboardTest* postupujte dle návodu v příloze (P IV) a vyzkoušejte si tak použití sériového monitoru.

- Prozkoumejte detailněji možnosti sériového monitoru pomocí příkladu (*Soubor → Příklady → 04.Communication → SerialEvent*). Příklad pravděpodobně nebude fungovat, to je dáno tím, že v případě Arduino Robota není definována funkce `serialEventRun()`. Proto přejmenujte funkci `serialEvent()` na `serialEventRun()` nebo funkci `serialEvent()` zavolejte napřímo na konci funkce `loop()`.

Po absolvování této úvodní úlohy si student osvojil všechny důležité základy pro práci s Arduino prostředím a hardwarem. Zná základní strukturu programu. Vyzkoušel si funkce `millis()` a `delay()`. Umí detekovat stisk tlačítka, vypsat jednoduchý text na LCD displej. Umí použít sériový monitor (přijímat a zasílat data přes připojené USB), čehož lze využít při ladění programů.

8.1.1 Bonusový úkol – LED dioda ovládaná tlačítky

Navazující bonusový úkol si klade za cíl získané znalosti zafixovat a rozšířit je o zkušenost se psaním neblokujícího kódu.

Úloha 1 – Bonusový úkol 1.1 – LED dioda ovládaná tlačítky

Zadání

Ovládejte vestavěnou LED diodu pomocí pětitlačítkové klávesnice tak, aby dioda reagovala okamžitě na poslední stisknuté tlačítko podle následujících pravidel:

- při stisku horního tlačítka (BUTTON_UP) se dioda rozsvítí

- při stisku dolního tlačítka (BUTTON_DOWN) dioda zhasne

- při stisku levého tlačítka (BUTTON_LEFT) se dioda rozsvítí a po uvolnění tlačítka zhasne

- při stisku prostředního tlačítka (BUTTON_MIDDLE) se dioda rozsvítí a zhasne po dvou sekundách po uvolnění tlačítka

- při stisku pravého tlačítka (BUTTON_RIGHT) se dioda rozsvítí a zhasne po pěti sekundách po uvolnění tlačítka

8.1.2 Bonusový úkol – Stmívání externí LED diody

Druhý bonusový úkol by měl studenta zasvětit do hardwarové části, zorientovat se v dostupných (vstupních a výstupních) pinech a bezpečně k robotu připojit externí součástku.

Úloha 1 – Bonusový úkol 1.2 – Stmívání externí LED diody

Zadání

K vhodnému pinu na řídící desce Arduino Robota připojte externí LED diodu a její jas ovládejte pomocí vestavěného potenciometru.

Potřebný hardware

LED dioda, vhodný rezistor, nepájivé kontaktní pole, propojovací vodiče.

Instrukce

- Při realizaci vyjděte z příkladu (*Soubor → Příklady → 03.Analog → Fading*).

- Příklad upravte tak, aby byla využita knihovna RobotControl.h.

- Nastudujte si specifikaci robota a vyberte vhodný pin, ke kterému připojíte LED diodu.

    - Nezapomeňte použít i vhodný odpor viz kapitola 3.5.1. Připojení LED diody.

    - Příklady zapojení viz obrázek (Obrázek 4).

8.2 Úloha 2: Karaoke – práce s LCD displejem a reproduktorem

Druhá úloha plynule navazuje na první v prozkoumávání možností robota. Má za cíl nejen seznámit studenta s vestavěným reproduktorem a připojeným LCD displejem, ale také poukázat na omezený prostor pro program i pro data.

Úloha 2 - Karaoke – práce s LCD displejem a reproduktorem

Na řídící desce Arduino Robota jsou integrované různé zajímavé moduly. V tomto cvičení se seznámíme hlavně s připojeným LCD displejem a reproduktorem. Stejný LCD displej lze připojit i na platformu Arduino Esplora.

Zadání

Vytvořte program, který na obrazovku LCD displeje vypíše text písně (např. Skákal pes, Ovčáci čtveráci, Kočka leze dírou, Pec nám spadla, Holka modrooká, Když jsem já sloužil), a který danou píseň přehraje. Během přehrávání se text písně bude červeně podbarvovat jako při karaoke.

Instrukce

Před řešením vlastní úlohy si projděte příklad (*Soubor → Příklady → Robot Control → learn → Melody*) a příklad (*Soubor → Příklady → 02.Digital → toneMelody*).

8.2.1 Bonusový úkol – Hra Pong

Další bonusový úkol procvičuje vykreslování jednoduchých objektů na displej a jejich animaci.

Úloha 2 – Bonusový úkol 2.1 – Hra Pong

Zadání

Vytvořte vlastní verzi hry Pong. Inspiraci můžete čerpat v příkladu *EsploraTFTPong* pro platformu Arduino Elspora.

8.3 Úloha 3: Na dvou kolech – základy pohybu mobilního robota

Cílem tohoto cvičení je prozkoumání možností ovládání pohybu robota z kontrolní desky.

Úloha 3 - Na dvou kolech – základy pohybu robota

Arduino robot je dvoukolová robotická platforma. Pohyb robota je založen na tzv. diferenciálním řízení, tj. směr a rychlost pohybu je dána rychlostí otáčení jednotlivých kol.

Zadání

Vytvořte program pro otestování pohybů robota (vpřed, vzad, doleva, doprava, otočení na místě, pohyb po odpovídající křivce).

Instrukce

- Nastudujte si následující funkce z knihovny ArduinoRobot.h.

```
void motorsWrite(int speedLeft,int speedRight);
void motorsStop();
void pointTo(int degrees);
void turn(int degress);
uint16_t compassRead();
```

- Otevřete nový projekt v Arduino IDE.

- Vložte šablonu projektu, viz příloha (P V).

- Vytvořte program pro otestování pohybu robota.

V tomto cvičení si studenti vyzkoušeli rozpohybovat robot a jeho pohyb ovládat. Při řešení si všímavý student uvědomí, jak nepřesné je využívání vestavěného digitálního kompasu díky rušení z okolí.

8.3.1 Bonusový úkol – Bez kompasu, tedy oficiálně neoficiální cestou

Pro zdárné vyřešení tohoto bonusového úkolu musí student upravit oficiální knihovnu ArduinoRobot.h. Při řešení tak tuto knihovnu detailněji prozkoumá. (Před zadáním úkolu je nutné zkontrolovat, zda je knihovna v původní oficiální podobě.)

Úloha 3 – Bonusový úkol 3.1 – Bez kompasu, tedy oficiálně neoficiální cestou

Zadání

V knihovně ArduinoRobot.h se nachází i funkce pro pohyb neobsažené v oficiální dokumentaci. Jsou to:

```
void moveForward(int speed);
void moveBackward(int speed);
void turnLeft(int speed);
void turnRight(int speed);
void motorsWritePct(int speedLeftPct, int speedRightPct);
```

Některé z těchto funkcí ale v implementaci mohou obsahovat chyby nebo se nechovají dle potřeb tohoto cvičení. Opravte je do podoby použitelné pro toto cvičení a implementujte řešení s jejich pomocí, tj. bez pomoci digitálního kompasu. (Bod s požadavkem otočení robota na severovýchod přeskočte.)

8.3.2 Bonusový úkol – Provedení zapamatované sekvence pohybů

Tento úkol vychází z příkladu (*Soubor → Příklady → Robot Control → explore → R01_Logo*), zadaná sekvence se navíc musí ukládat do permanentní paměti.

Úloha 3 – Bonusový úkol 3.2 – Provedení zapamatované sekvence pohybů

Zadání

Vytvořte program, který zaznamená sekvenci pohybů (vpřed, vzad, vlevo, vpravo) zadaných uživatelem z klávesnice. Tuto sekvenci robot následně provede. V případě, že není zadána žádná sekvence, provede se poslední zadaná (sekvenci je proto nutné vhodně uložit).

8.4 Úloha 4: Pozor na překážku – připojení senzoru pro měření vzdálenosti

Ve čtvrté úloze se student seznámí se senzory pro měření vzdálenosti. Vybraný senzor připojí k robotu a vytvoří program, který zabrání kolizi s překážkou.

Úloha 4 - Pozor na překážku – připojení senzoru pro měření vzdálenosti

Aby byl robot schopen pohybovat se prostorem bez kolizí, je nutné jej vybavit odpovídajícím senzorem. Pro tento účel se hodí některý ze senzorů pro měření vzdálenosti.

Zadání

Vytvořte program, který umožní robotu volný (náhodný) pohyb, aniž by narazil do zdi či jiné překážky. Princip spočívá v periodickém měření vzdálenosti robota od případného objektu před ním. V případě, že robot v dostatečné vzdálenosti zaznamená překážku sníží svoji rychlost, dokud se k překážce nepřiblíží na stanovenou vzdálenost. Poté se robot otočí o náhodný úhel a pokračuje v cestě.

Potřebný hardware

Senzor vzdálenosti (ultrazvukový/IR), nepájivé kontaktní pole, propojovací vodiče.

Instrukce

- Seznamte se s jednotlivými senzory pro měření vzdálenosti.
- Připojte zvolený senzor k odpovídajícímu vstupu.
- Vytvořte testovací program pro čtení hodnoty ze senzoru.
- Vytvořte finální program, který umožní robotu vyhýbat se překážkám.

8.4.1 Bonusový úkol – Bludiště

Arduino Robot svou velikostí není uzpůsoben tomu, aby se účastnil soutěže projetí bludištěm. Nicméně student si alespoň vyzkouší jedno z primitivnějších řešení této úlohy.

Úloha 4 – Bonusový úkol 4.1 – Bludiště

Zadání

Vytvořte program, který umožní robotu jet podél stěny ve stanovené vzdálenosti, a tak ji kopírovat. Implementujte tedy program, který by byl schopen projít bludištěm pomocí pravidla pravé ruky.

8.5 Úloha 5: Sleduj čáru – primitivní sledování čáry

Pátá úloha si klade za cíl seznámit studenta s vestavěnými IR senzory a jednoduchým algoritmem pro sledování čáry.

Úloha 5 - Sleduj čáru – primitivní sledování čáry

Sledování čáry je jednou z běžných disciplín robotických soutěží. Soutěž vyhrává robot, který v nejkratším čase projede zadanou dráhu. Dráha je většinou vymezena černou čárou, o určité šířce, na bílém podkladu. Robot zpravidla musí dráhu co nejpřesněji kopírovat. Případné zkrácení si dráhy by vyústilo v neuznání daného pokusu.

Zadání

Vytvořte program, který umožní robotu, aby svou drahou pohybu kopíroval černou čáru na bílém podkladu. K tomuto účelu využijte vestavěné IR senzory, které má robot umístěné na spodní straně motorové desky.

Potřebné pomůcky

Dráha vytvořená z černé elektrikářské pásky nalepené nejlépe na bílém podkladu (sololitová nebo laminátová deska).

Instrukce

- Seznamte se s metodami pro získání údajů z IR senzorů pro sledování čáry.
- Vytvořte algoritmus pro sledování čáry:
    - v případě, že levý krajní senzor detekuje čáru, robot zahne ostře doleva
    - v případě, že pravý krajní senzor detekuje čáru, robot zahne ostře doprava
    - v případě, že levý střední senzor detekuje čáru, robot zahne mírně doleva
    - v případě, že pravý střední senzor detekuje čáru, robot zahne mírně doprava
    - v případě, že prostřední senzor detekuje čáru, robot jede rovně

Pokud žádný ze senzorů nedetekuje čáru, robot se ostře točí ve směru poslední zjištěné pozice čáry, dokud ji opět nezachytí některý ze senzorů.

Algoritmus lze zjednodušit i do podoby, že v případě zjištění čáry pod jakýmkoli ze tří prostředních senzorů, robot pojede rovně.

Poznámka: Pro začátek se doporučuje dráhu projíždět nižší rychlostí, po odladění algoritmu můžete zkusit rychlost postupně navýšit.

8.5.1 Bonusový úkol – Sumo

Arduino Robot svou konstrukcí není uzpůsoben k tomu, aby se účastnil soutěže Robosumo, kde proti sobě zápasí dva roboty, kteří se snaží svého protivníka vytlačit z vyznačeného prostoru. Nicméně základy řešení této úlohy lze otestovat i s tímto robotem.

Úloha 5 – Bonusový úkol 5.1 – Sumo

Zadání

Vytvořte program, který umožní pohyb robota v ohraničeném kruhovém prostoru. Hranice bude tvořena černou čárou na podlaze, vytvořenou např. pomocí elektrikářské pásky. Jedná se tedy o úlohu detekce hrany. Navíc pokud robot pomocí některého ze senzorů pro měření vzdálenosti lokalizuje v tomto ohraničeném prostoru předmět, vytlačí jej za hranice tohoto prostoru.

Potřebný hardware

Senzor vzdálenosti (ultrazvukový/IR), nepájivé kontaktní pole, propojovací vodiče.

Instrukce

- Vytvořte program pro detekci hrany.

- Připojte vybraný senzor pro měření vzdálenosti.

- Program rozšiřte o vyhledávání a odstraňování objektů v relativní blízkosti robota.

Příklad algoritmu:

Robot se otáčí dokola a pomocí senzoru hledá objekt nacházející se do definované maximální vzdálenosti. Pokud takový nenajde, pokračuje v jízdě a po nějaké době opět zkusí vyhledávání. Pokud narazí na hranici tvořenou černou čárou, otočí se a pokračuje jiným směrem. Pokud robot zaznamená před sebou objekt, tak se rozjede jeho směrem a tlačí jej před sebou, dokud nenarazí na hranici. Poté se robot otočí a pokračuje v cestě.

8.6 Úloha 6: Autíčko na dálkové ovládání – komunikace s okolím

V této úloze si student vyzkouší dálkové ovládání robota pomocí libovolného infračerveného dálkového ovladače.

Úloha 6 - Autíčko na dálkové ovládání – komunikace s okolím

Robot lze na dálku ovládat mnoha způsoby, pomocí Bluetooth, WiFi, radiového či infračerveného spojení. V této úloze pro ovládání robota využijeme dálkový ovladač, např. ten od televize.

Zadání

Vytvořte program pro dálkové ovládání pohybu robota pomocí infračerveného dálkového ovladače.

Potřebný hardware

IR přijímač, dálkový ovladač, nepájivé kontaktní pole, propojovací vodiče.

Instrukce

- Seznamte se s knihovnu IRRemote.h.

- Připojte IR přijímač.

- Vytvořte testovací program pro čtení hodnoty z IR přijímače.

- Vytvořte finální program, který umožní robotu reagovat na přijaté příkazy z dálkového ovladače.

8.6.1 Bonusový úkol – Na drátě

V této úloze student vyzkouší principy ovládání robota pomocí lineárních potenciometrů připojených pomocí delšího vedení. Toto cvičení je tedy předstupeň pro plně bezdrátové ovládání robota například pomocí radiového či Bluetooth ovladače…

Úloha 6 – Bonusový úkol 6.1 – Na drátě

Zadání

Pomocí delšího vedení připojte k robotu dva potenciometry a otestujte přímé řízení robota.

Potřebný hardware

2x 10 kΩ lineární potenciometr (nebo Arduino Joystick PS2), dlouhé propojovací vodiče.

Instrukce

- Připojte potenciometry pomocí dostatečně dlouhého vodiče.

- Vytvořte program, kde každý z potenciometrů ovládá rychlost jednoho kola.

- Vytvořte program, kde jeden z potenciometrů ovládá rychlost a druhým se ovládá směr jízdy (Joystick PS2).

- Prakticky otestujte a vylaďte tyto algoritmy tak, aby ovládání bylo uživatelsky přívětivé.

8.6.2 Bonusový úkol – Bluetooth

Cílem úlohy je propojení robota přes Bluetooth s vybranou mobilní aplikací.

Úloha 6 – Bonusový úkol 6.2 – Bluetooth

Zadání

Vytvořte program, který pomocí Bluetooth modulu propojí robot s vybranou mobilní aplikací. Pro propojení využijte například knihovnu AltSoftSerial.h.

Instrukce

- Připojte k řídící desce modul Bluetooth.

- Nainstalujte si na mobil některou z aplikací, pomocí které můžete mobil s tímto modulem propojit, a tak robot ovládat.

- Napište program pro otestování spojení.

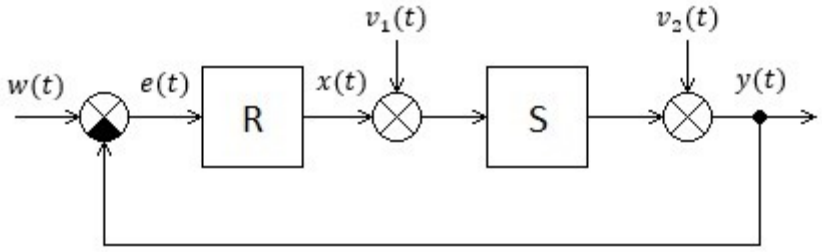
- Vytvořte program pro dálkové ovládání pohybu robota.

8.7 Úloha 7: Sleduj čáru lépe – sledování čáry pomocí regulátoru

Na příkladě sledování čáry s řízením pomocí vhodného regulátoru si student uvědomí, jak diametrálně kvalitnější výsledek dosáhne oproti primitivnímu řešení z šesté úlohy, což by ho mělo motivovat k tomu, aby se této problematice více věnoval.

Úloha 7 - Sleduj čáru lépe – sledování čáry pomocí regulátoru

Jednoduchý regulační obvod je tvořen regulátorem, regulovanou soustavou a zpětnou vazbou.



Obrázek 37: Jednoduchý regulační obvod.

Kde:

- $w(t)$ - žádaná hodnota: žádaná poloha robota vůči čáře

- $y(t)$ – regulovaná veličina: poloha robota vůči čáře

- $e(t)$ – regulační odchylka: rozdíl žádané a aktuální polohy robota $e(t) = w(t) - y(t)$

- $x(t)$ – akční zásah: rozdíl rychlosti motorů (diferenciální řízení)

- $v_1(t)$ a $v_2(t)$ – poruchové veličiny

Polohu robota vůči čáře zjistíme pomocí pětice IR senzorů. To, jak přesně tuto polohu určíme, ovlivní i následnou regulaci.

Pro řízení lze zvolit PID (Proporčně Integračně Derivační) regulátor, který byste již měli znát z předmětu Teorie Systémů. (V případě regulátoru implementovaného mikropočítačem se spíše jedná o PSD regulátor, tedy proporčně sumačně derivační regulátor.) PID regulátor se skládá z Proporcionální, Integrační a Derivační složky. Pro začátek můžete derivační a integrační složku nastavit jako nulovou. (P, PD, PI etc. jsou jen speciální případy PID regulátoru.)

Arduino robot již obsahuje algoritmus pro sledování čáry řízený PD regulátorem (integrační složku má tedy nulovou). Před vlastní implementací si tento algoritmus prozkoumejte a

otestujte. Při testování pravděpodobně narazíte na chybu v přenosu při nastavování parametrů regulátoru způsobenou tím, že jazyk C++ nemá definované pořadí vyhodnocování parametrů funkce.

V knihovně ArduinoRobotMotorBoard.h proto upravte soubor ArduinoRobotMotorBoard.cpp. Řádek 121 až 128 s kódem:

```
case COMMAND_LINE_FOLLOW_CONFIG:
  LineFollow::config(
    messageIn.readByte(),  //KP
    messageIn.readByte(),  //KD
    messageIn.readByte(),  //robotSpeed
    messageIn.readByte()   //integrationTime
  );
  break;
```

nahraďte kódem:

```
case COMMAND_LINE_FOLLOW_CONFIG:
  // vyřešení problému s pořadím vyhodnocování parametrů fce.
  int kp=messageIn.readByte();
  int kd=messageIn.readByte();
  int robotSpeed=messageIn.readByte();
  int integrationTime=messageIn.readByte();
  LineFollow::config(kp, kd, robotSpeed, integrationTime);
  break;
```

Tato chyba se nemusí projevit na všech zařízeních, záleží na implementaci konkrétního kompilátoru.

Pro Arduino lze také najít řadu knihoven, které PID regulátor implementují. Tím si lze ulehčit práci, nicméně doporučuji si zkusit implementovat vlastní řešení.

Zadání

Vytvořte algoritmus pro sledování čáry založený na PID regulátoru. Algoritmus by měl zvládnout i čáru o tloušťce cca 1,5 cm.

Instrukce

- Vyzkoušejte si sledování čáry pomocí vestavěného algoritmu (příklad *Soubor → Příklady → Robot Control → explore → R02_Line_Follow*) a experimentálně nastavte vhodné koeficienty pro tento regulátor.

- Nastudujte si principy PID regulátoru, prostudujte řešení z knihovny ArduinoRobotMotorBoard.h.

- Vytvořte algoritmus pro detekci pozice robota vůči čáře a otestujte jej na různých šířkách čáry.

Při zpracování pravděpodobně objevíte chybu v implementaci komunikace s motorovou deskou, kdy ve funkci `updateIR()` se na odpověď čeká pomocí funkce `delay()` pouze 10 ms, ale odpověď je dána cca po 23 ms. Tato chyba se také nemusí plně projevit, jelikož zpravidla se tato funkce volá periodicky, tudíž pravděpodobně budou dostupná data z předchozího cyklu. Tímto nedostatkem trpí i ostatní funkce, které očekávají odpověď z motorové desky. Řešením tak může být jednoduché prodloužení tohoto intervalu na 23 ms, nebo aktivní čekání stylu: `while (!messageIn.receiveData()) ;`.

- Implementujte vámi zvolený regulátor a jeho pomocí vytvořte program pro sledování čáry.

- Ošetřete krajní hodnoty pro případ, kdy robot ztratí kontakt s čárou.

- Experimentálně nastavte jednotlivé parametry na optimální hodnotu, tj. robot při jízdě uspokojivě kopíruje čáru.

Jako bonus může student realizovat sledování čáry pomocí některé z dostupných knihoven pro PID či jiný regulátor.

8.8 Úloha 8: Sleduj čáru po třetí – vlastní algoritmus na motorové desce

Poslední úloha spočívá v rozšíření knihovny pro motorovou desku Arduino Robota. Oficiální návod, jak při tomto postupovat, lze nalézt na webové stránce [29]. V příkladu ale zvolíme méně invazivní způsob a třídu RobotMotorBoard rozšíříme pomocí dceřiné třídy a přetížení jednotlivých metod. Tento způsob nám umožní oddělení nové funkcionality od oficiální a tím zpřehlední celý kód.

Implementací sledování čáry přímo na motorové desce lze zvýšit přesnost algoritmu díky možnosti zkrácení periody mezi jednotlivými akčními zásahy. Nedochází tak k časovým prodlevám, které by si vyžádala komunikace mezi deskami.

Úloha 8 - Sleduj čáru po třetí – vlastní algoritmus na motorové desce

Program pro motorovou desku je koncipován jako jednoduchý stavový automat, který přijímá a vykonává příkazy zaslané z řídící desky. Pokud chceme algoritmus rozšířit o zpracování nových příkazů nebo o nové stavy (módy), je nutné stávající třídu RobotMotorBoard z knihovny ArduinoRobotMotorBoard.h upravit, nebo lépe rozšířit pomocí dceřiné třídy s využitím dědičnosti.

Zadání

Rozšiřte stávající program pro motorovou desku o vlastní algoritmus sledování čáry. Využijte kód z předchozí úlohy. Původní metody pro sledování čáry musí zůstat stále dostupné, nestačí tedy pouze nahradit jejich vnitřní implementaci.

Instrukce

- Upravte deklaraci viditelnosti metod v třídě RobotMotorBoard z `private` na `protected`, díky tomu bude možné tyto metody využívat i v dceřiné třídě.

- Vytvořte nový program a zkopírujte do něj kód z přílohy (P VI).

- Definujte a implementujte příkaz pro nastavení parametrů regulátoru.

- Definujte a implementujte stav, v němž bude robot čáru sledovat.

- Pro oznámení ukončení sledování čáry můžete využít stejný signál jako oficiální algoritmus.

- Nahrajte program na motorovou desku, nezapomeňte změnit v nastavení vývojovou desku na *„Arduino Robot Control“* a vybrat odpovídající port.

- Vytvořte program pro řídící desku, kde nejdříve zavoláte příkaz pro nastavení parametrů nutných pro sledování čáry a následně přepněte stav motorové desky do vámi nově naprogramovaného stavu sledování čáry (program upravte nejlépe do podoby, kdy umožňuje uživatelsky příjemně zadávat jednotlivé parametry, což usnadní následné vyladění těchto parametrů).

- Opět nezapomeňte při nahrávání zvolit správnou desku a port.

V případě zájmu si student může naprogramovat i speciální stav (mód) pro úlohu detekce hrany (Edge Avoidance). Řídící deska tak bude mít možnost zasílat klasické příkazy pro pohyb robota, ale motorová deska před jejich vykonáním, i v průběhu jízdy, bude kontrolovat stav senzorů a v případě detekce hrany (hranice tvořené černou páskou) pohyb zastaví a dá o tom zprávu zpět řídící desce.

8.9 Bonusové úkoly

Bonusové úkoly jsou uvedeny, pouze pro inspiraci. Jejich hlavním cílem je ukázat studentovi možnosti některých dalších modulů, které lze k Arduino Robotu připojit.

8.9.1 Pixy2 kamera a sledování objektu

Bonusová úloha A - Připojení Pixy2 kamery a sledování objektu

K Arduino Robotu lze připojit nepřeberné množství senzorů a modulů. Za zmínku stojí například kamera Pixy2 pro detekci objektů, čárových kódů, čar a barev.

Zadání

Vytvořte aplikaci, která umožní robotu sledovat vybraný objekt. Tuto úlohu realizujte pomocí modulu Pixy2.

Instrukce

- K robotu připojte servo motor a na něj namontujte modul Pixy2 kamery tak, aby se tento modul mohl pohybovat i ve vertikálním směru (horizontální směr bude zajišťovat samotné otáčení celého robota).

- Kameru připojte pomocí SPI rozhraní (nezapomeňte napřed odpojit LCD displej).

- Naučte kameru rozpoznávat vybraný objekt dle návodu od výrobce.

- Nainstalujte knihovnu Pixy2.h.

- Vytvořte program pro sledování objektu (tj. řiďte otočení servo motoru, na kterém je kamera připevněna, i natočení robota tak, aby střed objektu byl co nejblíže středu zorného pole kamery).

8.9.2 Akcelerometr

Bonusová úloha B - Vodováha

Dalším zajímavým modulem je tříosý akcelerometr, který lze využít i jako gravitační senzor.

Zadání

Vytvořte aplikaci, která robot přemění na vodováhu.

Instrukce

- Připojte senzor k vhodným vstupům.

- Vytvořte program, který na LCD displeji vykreslí bublinku vodováhy, která se bude pohybovat v závislosti na náklonu senzoru.

- Rozšiřte program, tak aby šla vodováha kalibrovat (vynulovat), například pomocí prostředního tlačítka klávesnice.

8.10 Elektronický manuál

Součástí praktické části je i vytvořený elektronický manuál s úlohami, zpracovaný do podoby webových stránek nad platformou Wordpress. Pro realizované úlohy je vytvořena videodokumentace demonstrující výsledné chování jednotlivých algoritmů. Připojení vybraných externích komponent je vizualizováno pomocí nástroje Fritzing, pro který byla vytvořena komponenta řídící desky Arduino Robota. Viz příloha (P VII).

ZÁVĚR

V této práci jsem se věnoval vytvoření laboratorních úloh pro předmět robotika, s využitím platformy Arduino, s cílem studenta zaujmout pro tento obor a pomoci mu tak překonat jistou vstupní bariéru. Jako výukovou robotickou platformu jsem využil Arduino Robot a to nejen z důvodů, že jej Laboratoř robotiky na FAI UTB již vlastní v několika exemplářích a odpadá tak nutnost další investice, ale také z důvodu, že pro tento účel byl Arduino Robot předurčen již svými tvůrci. Navíc se jedná o open-source platformu s relativně dobrou oficiální dokumentací a vestavěnými příklady použití.

Jednotlivé úlohy jsem koncipoval tak, aby studenta provedly od úplných základů práce s programováním mikropočítače až po složitější úlohy pro mobilní roboty, které jsou součástí robotických soutěží. Tímto procesem učení jsem si sám prošel. Na začátku jsem měl jen teoretické povědomí o programování mikropočítačů (z předmětu Programování mikropočítačů). A v rámci vypracovávání práce jsem se postupně dostal až k úspěšnému vyzkoušení si disciplíny sledování čáry na soutěži Robogames, kterou Fakulta aplikované informatiky UTB ve Zlíně pořádá.

V práci jsem se mimo jiné zabýval i možnostmi rozšíření Arduino Robota o různé senzory a komunikační moduly, na které se zaměřují úlohy číslo 4 a 7. Některým zajímavým senzorům jsou věnovány také bonusové úkoly.

Při realizaci vzorových řešení jsem narazil i na pár chyb, které obsahují oficiální knihovny pro platformu Arduino Robot, na které jsem se snažil při tvorbě zadání upozornit. Celkově hodnotím volbu této platformy jako velice výhodnou, a to i díky těmto obsaženým chybám. Student se tedy díky nim poučí a vyvaruje se tak obdobným vlastním chybám, které by si jinak nemusel uvědomit. Navíc je studentovi dána možnost implementovat vlastní, lepší řešení, což by v případě plně odladěného a dokonalého kódu nebylo možné.

Tím, že úlohy jsou koncipovány primárně jako úvod do robotiky, nemohly, a ani nebylo jejich cílem, obsáhnout vše, co v mobilní robotice s tímto robotem lze realizovat. Taktéž úlohy nebyly zaměřeny na složitější algoritmy vyžadující hlubší znalosti z oblasti mobilní robotiky a matematiky, nicméně na příkladu sledování čáry si student mohl uvědomit, že i aplikací jednoduššího regulátoru lze dosáhnout daleko kvalitnějšího výsledku než v případě primitivního přímočarého řešení. I díky tomuto by student měl získat dostatečnou motivaci se robotice věnovat podrobněji.

SEZNAM POUŽITÉ LITERATURY A ZDROJŮ

[1] DUDEK, Gregory a Michael JENKIN. *Computational principles of mobile robotics.* Second edition. New York: Cambridge University Press, 2010, xiii, 391. ISBN 978-0-521-87157-0. Dostupné také z: http://www.loc.gov/catdir/enhancements/fy1010/2010020795-t.html

[2] SKAŘUPA J. *Průmyslové roboty a manipulátory*. 2007. [Online]. [cit. 2019-01-25]. Dostupné z: http://www.elearn.vsb.cz/archivcd/FS/PRM/Text/Skripta_PRaM.pdf.

[3] ŠVEJDA M. *Úvod do robotiky a mechatroniky*. 2014. [Online]. [cit. 2019-01-25]. Dostupné z: https://docplayer.cz/19285012-Mechatronika-a-robotika-jako-vedni-disciplina.html.

[4] TOCHÁČEK D. a J. LAPEŠ. *Edukační robotika*. 2012. [Online]. [cit. 2019-01-25]. Dostupné z: https://kraken.pedf.cuni.cz/~lapej2ap/robo/skripta_edurobo.pdf

[5] CORKE, Peter I. *Robotics, vision and control: fundamental algorithms in Matlab*. Berlin: Springer, 2011, xxiv, 570 s. Springer tracts in advanced robotics. ISBN 978-3-642-20143-1.

[6] NOVÁK M. *Arduino – základ pro levnou robotickou platformu*. 2016. [Online]. [cit. 2019-01-25]. Dostupné z: http://mfi.upol.cz/files/25/2502/mfi_2502_147_153.pdf

[7] DULÍK Tomáš. *RoboGames*. Zlín: Fakulta aplikované informatiky, UTB, 2017. [Online]. [cit. 2019-02-02]. Dostupné z: https://robogames.utb.cz

[8] MARGOLIS, Michael. *Make an Arduino-controlled robot*. Sebastopol, Calif.: O'Reilly, c2013. Make. ISBN 978-1-449-344375.

[9] BANZI M. *How Arduino is open-sourcing imagination*. 2012. [Online]. [cit. 2019-02-20]. Dostupné z: https://www.youtube.com/watch?v=UoBUXOOdLXY

[10] MARGOLIS, Michael. *Arduino cookbook*. Second edition. Sebastopol: O'Reilly, 2011, xx, 699 s. ISBN 978-1-4493-1387-6.

[11] VODA Z., tým HW Kitchen. *Průvodce světem Arduina*. Bučovice: Nakladatelství Martin Stříž, 2017, 240 s. ISBN: 978-80-87106-93-8.

[12] Arduino. *Arduino Products*. [Online]. [cit. 2019-03-23]. Dostupné z: https://www.arduino.cc/en/Main/Products

[13] SAM. *History of Arduino*. 2017. [Online]. [cit. 2019-03-23]. Dostupné z: https://core-electronics.com.au/tutorials/history-of-arduino.html

[14] HW Kitchen. *Arduino IDE*. [Online]. [cit. 2018-12-20]. Dostupné z: https://arduino.cz/arduino-ide

[15] BARRAGÁN H., B. HAGMAN a A. BREVIG. *Wiring*. [Online]. [cit. 2019-03-18]. Dostupné z: http://wiring.org.co/about.html

[16] BARRAGÁN H., B. HAGMAN a A. BREVIG. *Wiring – reference*. [Online]. [cit. 2019-03-18]. Dostupné z: http://wiring.org.co/reference

[17] BARRAGÁN H. *The Untold History of Arduino*. [Online]. [cit. 2019-03-18]. Dostupné z: https://arduinohistory.github.io

[18] Arduino. *Sketch*. [Online]. [cit. 2019-01-10]. Dostupné z: https://www.arduino.cc/en/tutorial/sketch

[19] SLINTÁK V. *Vývojové prostředí a programování Arduina*. 2011. [Online]. [cit. 2019-01-10]. Dostupné z: https://uart.cz/90/ide-a-programovani-arduina

[20] Arduino. *Blink*. [Online]. [cit. 2019-01-10]. Dostupné z: https://www.arduino.cc/en/tutorial/blink

[21] MCROBERTS, Michael. *Beginning Arduino*. Second ed. Berkeley, CA: Apress, [2013]. Technology in action series. ISBN 978-1-4302-5016-6.

[22] WARREN, John-David, Josh S ADAMS a Harald MOLLE. *Arduino robotics*. New York: Apress, [2011], xxiv, 601 s. Technology in action. ISBN 978-1-4302-3183-7.

[23] RS Components. *Arduino Robot Video Tutorial*. [Online]. [cit. 2019-02-20]. Dostupné z: https://www.youtube.com/watch?v=vlxH0DAv5ds&list=PLqnIaXrARxXGF6afLoH7-2Gj9Chu9rfj9

[24] LANGBIDGE J. A. Arduino Sketches: *Tools and Techniques for Programming Wizardry*. John Wiley & Sons, Inc. [2015], 480 s. ISBN: 978-1-119-18371-6

[25] Redakce HW serveru. *Robot Arduino: platforma na kolech*. 2013. [Online]. [cit. 2019-02-20]. Dostupné z: https://vyvoj.hw.cz/konstrukce/robot-arduino-platforma-na-kolech.html

[26] Arduino Verkstad AB. *Basic education shield*. [Online]. [cit. 2019-04-27]. Dostupné z: http://ctc-dev.verkstad.cc/en/course-literature/basic-education-shield

[27] Arduino. *ATmega 32U4-Arduino Pin Mapping*. [Online]. [cit. 2019-04-27]. Dostupné z: https://www.arduino.cc/en/Hacking/PinMapping32u4

[28] LUCAS G. *A Tutorial and Elementary Trajectory Model*. 2001. [Online]. [cit. 2019-03-23]. Dostupné z: http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html

[29] Arduino. *Write Your Own Firmware*. [Online]. [cit. 2019-02-10]. Dostupné z: https://www.arduino.cc/en/Reference/RobotMotorWriteYourOwnFirmwar

[30] MATĚJ Zdeněk. *MCU: protokol UART*. 2015. [Online]. [cit. 2019-02-20]. Dostupné z: http://src.athaj.cz/teaching/rev/arch_uart

[31] DUDÁČEK K. *Sériová rozhraní SPI, Microwire*, I2C a CAN. 2002. [Online]. [cit. 2019-02-20]. Dostupné z: http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf

[32] NXP Semiconductors. *I2C-bus specification and user manual*. 2014. [Online]. [cit. 2019-02-21]. Dostupné z: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[33] Honeywell International. *Digital Compass Solution HMC6352*. 2006. [Online]. [cit. 2019-03-15]. Dostupné z: https://www.sparkfun.com/datasheets/Components/HMC6352.pdf

[34] SparkFun Electronics. *Sparkfun*. [Online]. [cit. 2018-12-20]. Dostupné z: https://www.sparkfun.com

[35] Arduino. *Robot library*. [Online]. [cit. 2018-12-09]. Dostupné z: https://www.arduino.cc/en/Reference/RobotLibrary

[36] ECLIPSERA s.r.o. *ARDUINO-SHOP*. [Online]. [cit. 2018-12-20]. Dostupné z: https://arduino-shop.cz

[37] PixyCam. *Pixy2*. [Online]. [cit. 2018-12-20]. Dostupné z: https://pixycam.com

[38] DFRobot. *DFRobot.* [Online]. [cit. 2019-01-08]. Dostupné z: https://www.dfrobot.com/product-360.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

A/D	Analog/Digital
AVR	Alf and Vegard's RISC processor
CPU	Central Processing Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FAI	Fakulta aplikované informatiky
GND	Ground
I2C	Internal Integrated Circuit
ICSP	In-Circuit Serial Programing
IDE	Integrated Development Environment
IR	InfraRed
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MCU	Microcontroller Unit
MISO	Master In Slave Out
MIT	Massachusetts Institute of Technology
MOSI	Master Out Slave In
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RX	Receiver
SCK	Serial Clock
SCL	Serial Clock
SD	Secure Digital

SDA	Serial Data
SPI	Serial Peripheral Interface
SS	Slave Select
TFT	Thin Film Transistor
TWI	Two Wire Interface
TX	Transmitter
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UTB	Univerzita Tomáše Bati ve Zlíně
VCC	Voltage Common Collector

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: JEDNODUCHÝ UVÍTACÍ PROGRAM

```
/**
 * Jednoduchý uvítací program
 *
 * Tento kód je určen pro řídící desku Arduino Robota
 *
 * Program vypíše na připojený LCD displej:
 * "Ahoj, já jsem Arduino Robot.
 * Připoj mě pomocí USB kabelu.
 * A něco zajímavého mě nauč..."
 */

#include <ArduinoRobot.h>

void setup() {
  // Inicializace řídící desky, displeje a SD čtečky
  Robot.begin();
  Robot.beginTFT();
  Robot.beginSD();

  // Nastavení barvy pozadí na bílou
  Robot.background(255,255,255);
  // Nastavení barvy textu
  Robot.stroke(0,0,0);
  // Vymazání obrazovky
  Robot.clearScreen();

  // Zobrazení loga Arduino Robota
  Robot.drawBMP("lg0.bmp",0,0);
  delay(2000);
  // Zobrazení obrázku robota s připojeným USB kabelem
  Robot.drawBMP("init4.bmp", 0, 0);
  // Vypsání výzvy
  Robot.text("Ahoj, ja jsem", 5, 85);
  Robot.setTextSize(2); // nastavení fontu na 20 px
  Robot.text("Arduino", 10, 95);
  Robot.text("Robot", 23, 112);
  Robot.setTextSize(1); // nastavení fontu na 10 px
  Robot.text("Pripoj me pomoci USB", 5, 128);
  Robot.text("kabelu. A neco zaji-", 5, 138);
  Robot.text("maveho me nauc...", 5, 148);
}

void loop() {
  // Funkce loop() zůstává prázdná
}
```

PŘÍLOHA P II: PROGRAMOVÉ JÁDRO MOTOROVÉ DESKY

```
/**
 * Programové jádro motorové desky Arduino Robota
 *
 * Tento kód je určen pro motorovou desku Arduino Robota.
 * Jedná se o základní firmware motorové desky
 * viz https://www.arduino.cc/en/Reference/RobotMotorBegin.
 *
 * Nahrajte tento program na spodní desku Arduino Robota,
 * vždy když potřebujete vrátit naprogramování motorové
 * desky do výchozího stavu.
 * Doporučení: předem se ujistěte, že i použitá knihovna
 * ArduinoRobotMotorBoard.h je v původním, nezměněném stavu.
 */

#include <ArduinoRobotMotorBoard.h>

void setup() {
  // Inicializace motorové desky
  RobotMotor.begin();
}
void loop() {
  // Příjem příkazů od řídicí desky a jejich obsloužení
  RobotMotor.parseCommand();
  // Provedení různých činností v závislosti na nastveném
  // stavu (módu)
  RobotMotor.process();
}
```

PŘÍLOHA P III: PRVNÍ PROGRAM, NÁVOD

První a nejjednodušší program, který se při výuce programování mikropočítačů používá je rozblikání LED diody. Je to v podstatě ekvivalent tzv. programu „Hello World“ (česky: Ahoj Světe) používaného při výuce programovacích jazyků.

Zadání

Seznamte se s Arduino IDE a Arduino Robotem. Rozblikejte vestavěnou LED diodu (LED1) na kontrolní desce Arduino Robota.

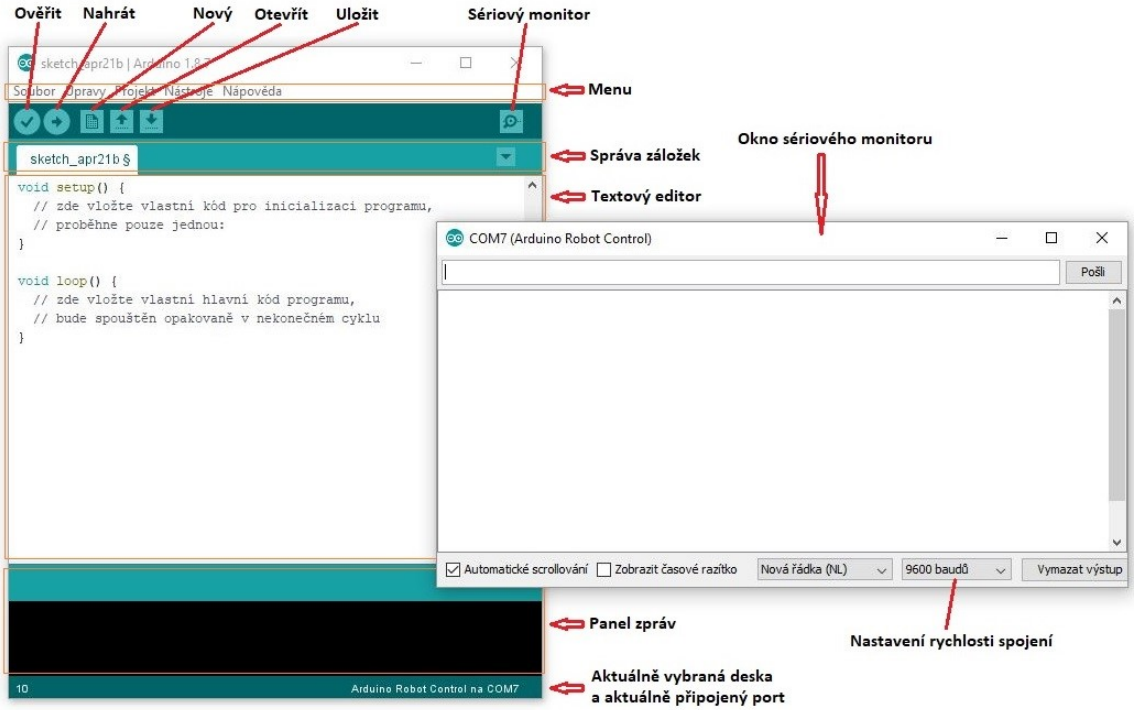
Postup

Nainstalujte Arduino IDE.

Odkaz na stažení instalačního balíčku**: https://www.arduino.cc/en/Main/Software**.

Spusťte program Arduino IDE.

Ikona programu Arduino IDE.

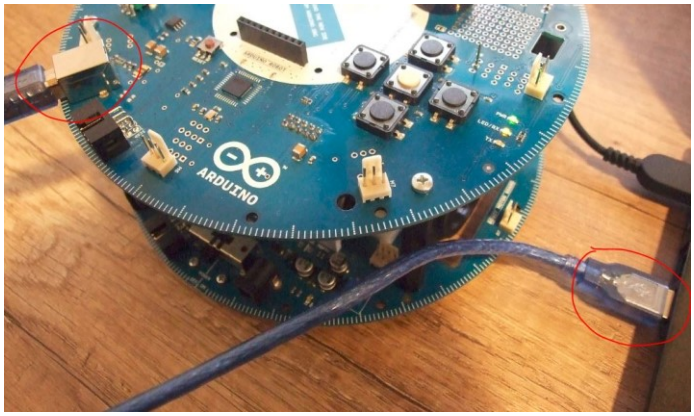


Popis prostředí Arduino IDE.

Vytvořte nový projekt (***Soubor*** → ***Nový***, nebo **<Ctrl + N>**) a zkopírujte následující kód.

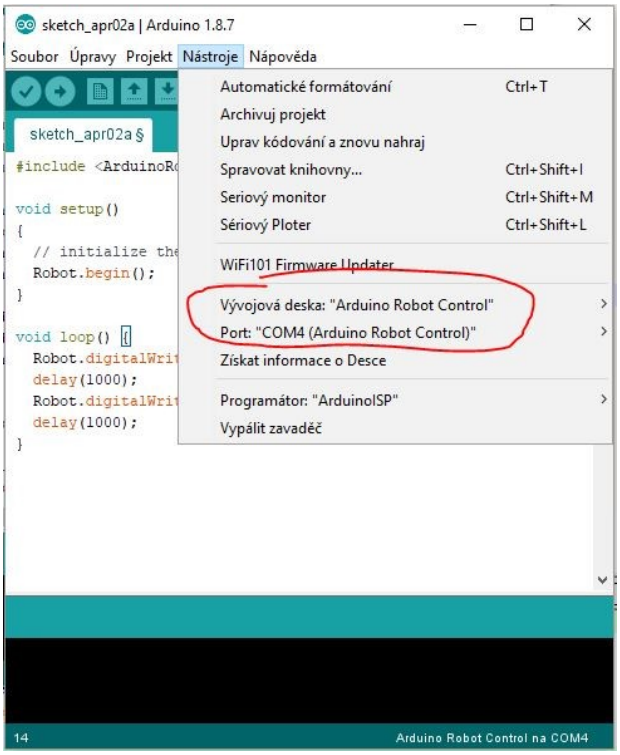
```
/**
 * Blikání vestavěnou diodou LED1
 *
 * Na jednu sekundu rozsviť LED diodu,
 * poté ji zhasni a počkej další sekundu,
 * toto stále opakuj.
 */

// připojení knihovny pro řídící desku Arduino Robota
#include <ArduinoRobot.h>

// funkce setup se provede právě jednou po restartu
// nebo zapnutí desky
void setup() {
  // Inicializace řídící desky,
  // mimo jiné se tímto inicializuje i pin
  // ke kterému je připojena LED dioda
  Robot.begin();
}

// funkce loop se provádí stále dokola
void loop() {
  // zapnutí LED nastavením nízké úrovně napětí LOW
  Robot.digitalWrite(LED1, LOW);
  // počkej sekundu
  delay(1000);
  // vypnutí LED (HIGH je vysoká úroveň napětí)
  Robot.digitalWrite(LED1, HIGH);
  // počkej sekundu
  delay(1000);
}
```

Připojte Arduino robot pomocí USB kabelu.



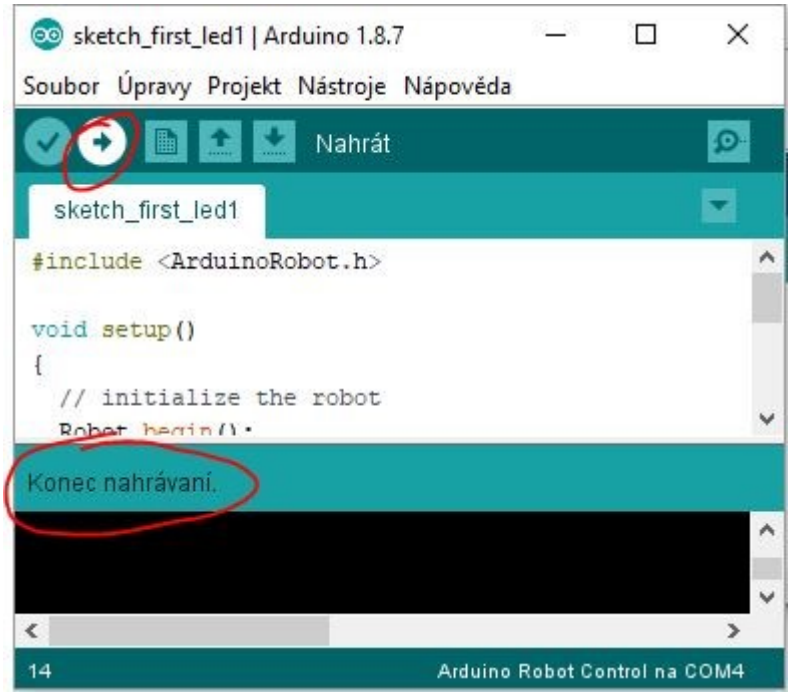
Propojení Arduino Robota pomocí USB s PC.

V menu ***Nástroje → Vývojová deska*** zvolte hodnotu „**Arduino Robot Control**“ a v menu ***Nástroje → Port*** zvolte port ke kterému je tato deska připojena, např. „**COM1 (Arduino robot Control)**„



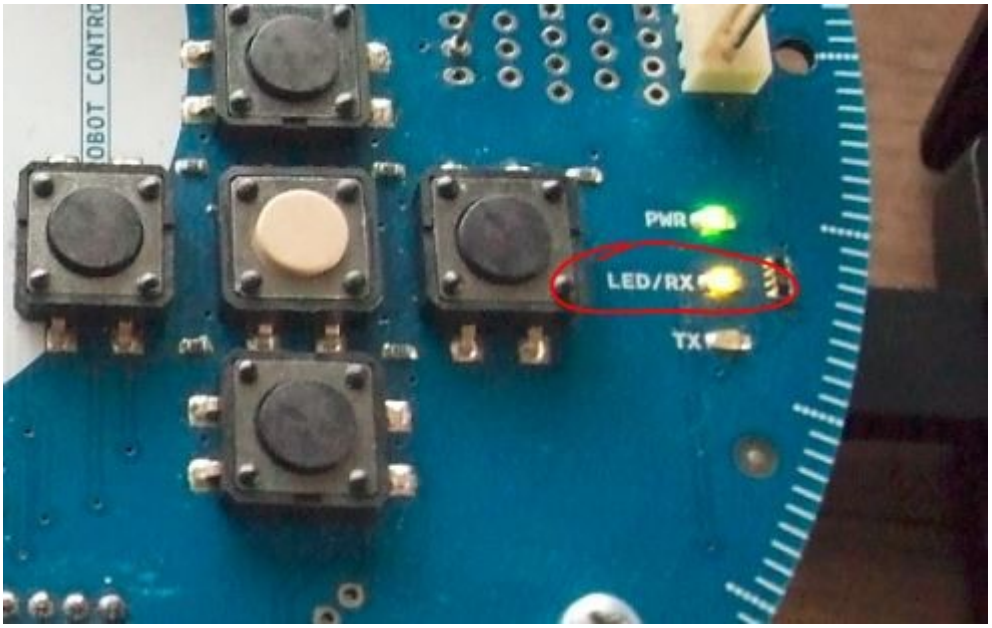
Nastavení vývojové desky a portu.

Poté nahrajte program **<CTRL + U>** nebo v menu ***Projekt → Nahrát.*** V průběhu nahrávání se na řídící desce Arduino Robota rozblikají LED diody **RX** a **TX**. Což indikuje, že mezi deskou a počítačem probíhá komunikace po sériové lince.



Nahrání programu na řídící desku.

Po zkompilování a nahrání programu by se měla dioda označená **LED/RX** rozblikat.

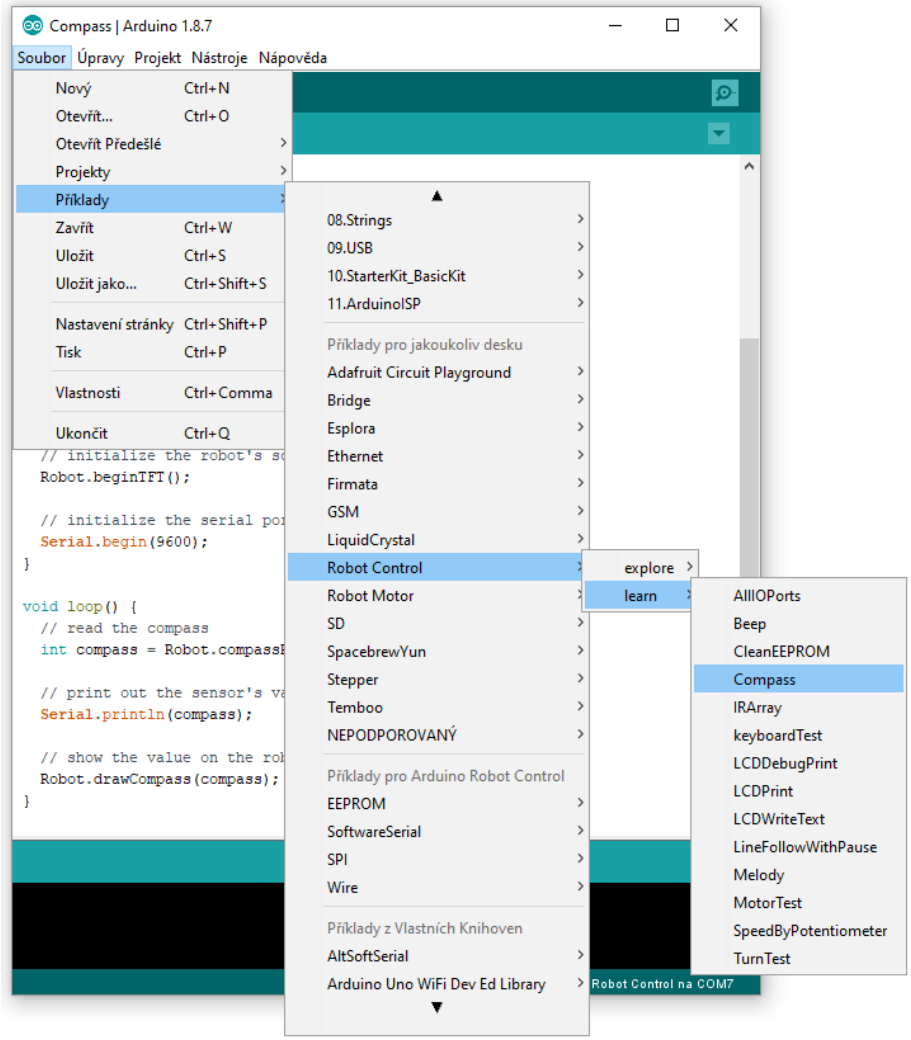


Blikající integrovaná LED dioda.

Všimněte si, že kód: `Robot.digitalWrite(LED1, HIGH);` diodu zhasíná, kdežto `Robot.digitalWrite(LED1, LOW);` diodu rozsvítí.

PŘÍLOHA P IV: SÉRIOVÝ MONITOR

Otevřete si příklad (*Soubor* → *Příklady* → *Robot Control* → *learn* → *Compass*).

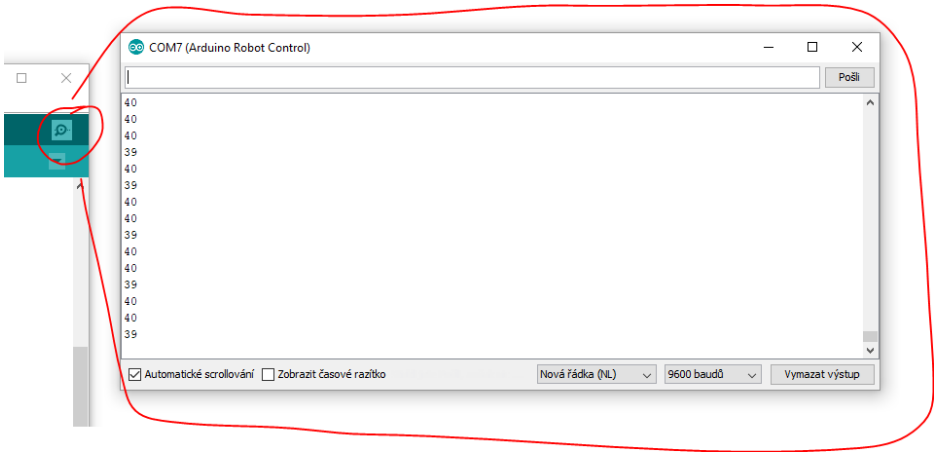


Otevření příkladu „Compass“.

Nahrajte program na řídící desku Arduino Robota.

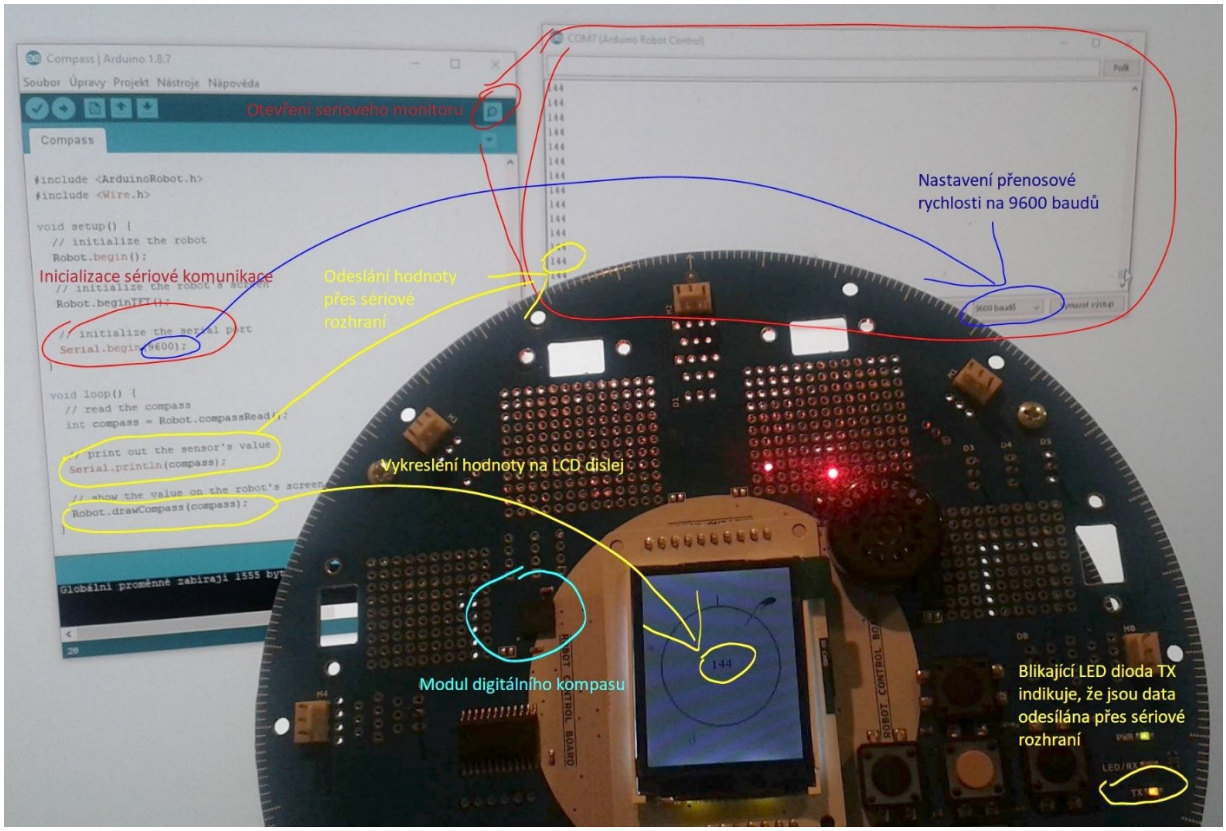
V kódu si všimněte funkce `Serial.begin(9600);`, která slouží k inicializaci sériového spojení a funkce `Serial.println(compass);`, která odešle přes toto spojení předanou hodnotu a znak nového řádku.

V Arduino IDE si otevřete seriový monitor (*Nástroje* → *Seriový monitor*) nebo pomocí klávesové zkratky **<Ctrl + Shift + M>** či ikony lupy v pravé horní části okna.



Otevření sériového monitoru pomocí ikony.

Přenosovou rychlost spojení nastavte na 9600 baudů, tudíž shodnou hodnotu, jaká je nastavena při inicializaci v programu pomocí funkce `Serial.begin();`.



Program Compass a použití Sériového monitoru.

Vyzkoušejte obdobný příklad (*Soubor → Příklady → Robot Control → learn → keyboardTest*), který přes sériové rozhraní odesílá identifikátor stisknutého tlačítka.

Výpis na sériový monitor lze využít k výpisu obsahu jednotlivých proměnných při ladění programu.

PŘÍLOHA P V: ZÁKLADNÍ POHYB ROBOTA

```
/**
 * Základní pohyb robota
 * směr a rychlost pohybu je dána
 * směrem a rychlostí otáčení jednotlivých kol
 */

#include <ArduinoRobot.h>

void setup() {
  // inicializace robota
  Robot.begin();
  delay(5000);
}

void loop() {
  // pohyb vpřed
  /** vložte kód **/
  // rychlé zastavení
  /** vložte kód **/
  delay(1000);
  // pohyb vzad
  /** vložte kód **/
  // pomalé zastavení
  /** vložte kód **/
  delay(1000);
  // otočení doleva o 90 stupňů
  /** vložte kód **/
  delay(1000);
  // otáčení doprava o 270 stupňů
  /** vložte kód **/
  delay(1000);
  // otáčení doprava po dobu cca dvě sekundy
  /** vložte kód **/
  delay(1000);
```

```
  // pohyb po kružnici levotočivě
  /** vložte kód **/
  delay(1000);
  // otočení na severovýchod
  /** vložte kód **/
  // pohyb po čtverci v poloviční rychlosti
  /** vložte kód **/
  delay(1000);
  // pohyb po zubaté křivce
  /** vložte kód **/
  delay(1000);
  // pohyb po zubaté křivce pozpátku
  /** vložte kód **/
  delay(2000);
}
```

PŘÍLOHA P VI: ROZŠÍŘENÍ KNIHOVNY PRO MOTOROVOU DESKU

My_Robot_Motor_Core.ino

```
/**
 * Rozšíření knihovny ArduinoRobotMotorBoard.h
 *
 * Tento kód je určen pro motorovou desku Arduino Robota
 *
 * Poznámka:
 *  Aby tento kód fungoval je nutné upravit v oficiální
 *  knihovně ArduinoRobotMotorBoard.h viditelnost metod
 *  z "private" na "protected".
 *  Při vlastním rozšíření postupujte dle popisů
 *  v přiložených souborech MyArduinoMotorBoard.h
 *  a MyArduinoRobotMotorBoad.cpp
 */

#include "MyArduinoRobotMotorBoard.h"

MyRobotMotorBoard MyRobotMotor;

void setup() {
  MyRobotMotor.begin();
}

void loop() {
  MyRobotMotor.parseCommand();
  MyRobotMotor.process();
}
```

MyArduinoRobotMotorBoard.h

```
#ifndef MyArduinoRobotMotorBoard_h
#define MyArduinoRobotMotorBoard_h

#include "ArduinoRobotMotorBoard.h"

// Číselné kódy stavů
// @todo - definuj kódy pro vlastní nové stavy
#define MODE_MY_MODE 11 // je vhodné zvolit číslo > 10
//#define MODE_MY_SECOND_MODE 12 // zvolte číslo > 10

// Číselné kódy příkazů
// @todo - definuj kódy nových příkazů
#define COMMAND_MY_COMMAND 101 // je vhodné zvolit číslo > 100
//#define COMMAND_MY_SECOND_COMMAND 102 // zvolte číslo > 100
```

```
class MyRobotMotorBoard : public RobotMotorBoard {
  public:
    MyRobotMotorBoard();

    void begin();
    void setMode(uint8_t mode);
    void process();
    void processCommand(uint8_t command);

    void parseCommand();
  protected:
    void processParentCommand(uint8_t command);
    // @todo - zde definuj vlastní proměnné a funkce
  private:
};

#endif
```

MyArduinoRobotMotorBoard.cpp

```
#include "MyArduinoRobotMotorBoard.h"
#include "ArduinoRobotMotorBoard.h"

MyRobotMotorBoard::MyRobotMotorBoard() {
  // RobotMotorBoard::RobotMotorBoard();
}

void MyRobotMotorBoard::begin(){
  RobotMotorBoard::begin();
  // @todo - uprav pro inicializaci svého kódu
  // ... zde vložte svůj kód ...
}

void MyRobotMotorBoard::setMode(uint8_t mode) {
  // nastavení stavů definovaných v původní knihovně
  RobotMotorBoard::setMode(mode);

  // @todo - uprav pro nastavení svých nových stavů
  switch(mode) {
    case MODE_MY_MODE:
      // ... zde vložte svůj kód ...
      break;
    /*
    case MODE_MY_SECOND_MODE:
      // ... zde vložte svůj kód ...
      break;
    */
  }
```

```
  this->mode = mode;
}

void MyRobotMotorBoard::process() {
  // přeskočit zpacování stavu, pokud jeho provádění pozasta-
veno
  if (isPaused) return;
  // zpracování stavů definovaných v původní knihovně
  RobotMotorBoard::process();

  // @todo - uprav pro provedení svých nových stavových akcí
  switch(mode) {
    case MODE_MY_MODE:
      // ... zde vložte svůj kód ...
      break;
    /*
    case MODE_MY_SECOND_MODE:
      // ... zde vložte svůj kód ...
      break;
    */
  }
}

void MyRobotMotorBoard::processCommand(uint8_t command) {
  // zpracování příkazů definovaných v původní knihovně
  processParentCommand(command);

  // @todo - uprav pro zpracování svých příkazů
  switch(command) {
    case COMMAND_MY_COMMAND:
      // ... zde vložte svůj kód ...
      break;
    /*
    case MODE_MY_SECOND_COMMAND:
      // ... zde vložte svůj kód ...
      break;
    */
  }
}

void MyRobotMotorBoard::parseCommand() {
  if(this->messageIn.receiveData()){
    //Serial.println("data received");
    uint8_t command=messageIn.readByte();
    processCommand(command);
  }
}

// obsah této metody je shodný s metodou parseCommand()
// z původní knihovny až na zakomentování části zodpovědné
```

```
// za načtení příkazu zaslaného z řídící desky
void MyRobotMotorBoard::processParentCommand(uint8_t command)
{
  uint8_t modeName;
  uint8_t codename;
  int value;
  int speedL;
  int speedR;
//if(this->messageIn.receiveData()){
//  //Serial.println("data received");
//  uint8_t command=messageIn.readByte();
//  //Serial.println(command);
    switch(command){
      case COMMAND_SWITCH_MODE:
        modeName=messageIn.readByte();
        setMode(modeName);
        break;
      case COMMAND_RUN:
        if(mode==MODE_LINE_FOLLOW)break;
        speedL=messageIn.readInt();
        speedR=messageIn.readInt();
        motorsWrite(speedL,speedR);
        break;
      case COMMAND_MOTORS_STOP:
        motorsStop();
        break;
      case COMMAND_ANALOG_WRITE:
        codename=messageIn.readByte();
        value=messageIn.readInt();
        _analogWrite(codename,value);
        break;
      case COMMAND_DIGITAL_WRITE:
        codename=messageIn.readByte();
        value=messageIn.readByte();
        _digitalWrite(codename,value);
        break;
      case COMMAND_ANALOG_READ:
        codename=messageIn.readByte();
        _analogRead(codename);
        break;
      case COMMAND_DIGITAL_READ:
        codename=messageIn.readByte();
        _digitalRead(codename);
        break;
      case COMMAND_READ_IR:
        _readIR();
        break;
      case COMMAND_READ_TRIM:
        _readTrim();
        break;
      case COMMAND_PAUSE_MODE:
```

```
        pauseMode(messageIn.readByte());
        break;
      case COMMAND_LINE_FOLLOW_CONFIG:
        int kp=messageIn.readByte();
        int kd=messageIn.readByte();
        int robotSpeed=messageIn.readByte();
        int iTime=messageIn.readByte();
        LineFollow::config(kp, kd, robotSpeed, iTime);
        break;
    }
//}
//delay(5);
}
```

PŘÍLOHA P VII: OBSAH PŘILOŽENÉHO CD

```
Obsah CD
|   ArduinoRobot.docx
|   ArduinoRobot.pdf
|   fulltext.docx
|   fulltext.pdf
|   obsah.txt
|   VzorovaReseni.zip
|   youtube_kanal.txt
|   Zadani_uloh.docx
|   Zadani_uloh.pdf
|
+---1_lab_uloha
|   |   LedOvladanaKlavesnici_video.mp4
|   |   LED_fritzing.png
|   |   zpusoby-zapojeni-led-arduino.jpg
|   |   ztmivani_fritzing.png
|   |   ztmivani_realizace.jpg
|   |   ztmivani_schema.png
|   |   ztmivani_video.mp4
|   |
|   +---1_Bonus_LED1_Keys
|   |       1_Bonus_LED1_Keys.ino
|   |
|   +---2_Bonus_LED_Fading
|   |       2_Bonus_LED_Fading.ino
|   |
|   +---2_Bonus_LED_Fading_by_POT
|   |       2_Bonus_LED_Fading_by_POT.ino
|   |
|   +---LED1
|   |       LED1.ino
|   |
|   +---LED1_Blink
|   |       LED1_Blink.ino
```

```
|   |
|   +---LED1_BlinkWithoutDelay
|   |       LED1_BlinkWithoutDelay.ino
|   |
|   +---SerialEvent
|   |       SerialEvent.ino
|   |
|   \---SerialEvent_LCD
|           SerialEvent_LCD.ino
|
+---2_lab_uloha
|   |   karaoke_video.mp4
|   |   pong_video.mp4
|   |
|   +---1_Bonus_Pong
|   |       1_Bonus_Pong.ino
|   |
|   +---OvcaciCtveraci
|   |       OvcaciCtveraci.ino
|   |
|   +---OvcaciCtveraci2
|   |       OvcaciCtveraci2.ino
|   |
|   +---PlayMelody
|   |       PlayMelody.ino
|   |
|   \---ToneMelody
|           pitches.h
|           ToneMelody.ino
|
+---3_lab_uloha
|   |   test_motoru_video.mp4
|   |
|   +---1_Bonus_ZakladyPohybu
|   |       1_Bonus_ZakladyPohybu.ino
```

```
|   |
|   +---2_Bonus_Logo_EEPROM
|   |       2_Bonus__Logo_EEPROM.ino
|   |
|   +---MotorTest
|   |       MotorTest.ino
|   |
|   +---ZakladyPohybu_reseni
|   |       ZakladyPohybu_reseni.ino
|   |
|   \---ZakladyPohybu_zadani
|           ZakladyPohybu_zadani.ino
|
+---4_lab_uloha
|   |   prekazky_realizace.JPG
|   |   prekazky_test_video.AVI
|   |   senzory_fritzing.png
|   |   senzory_realizace.jpg
|   |   senzory_schema.png
|   |   senzory_video.mp4
|   |
|   +---1_Bonus_Bludiste
|   |       1_Bonus_Bludiste.ino
|   |
|   +---Prekazky
|   |       Prekazky.ino
|   |
|   \---Senzory
|           Senzory.ino
|
+---5_lab_uloha
|   |   detekce_hrany_video.mp4
|   |   sledovani_cary_bila.mp4
|   |   sumo_video.mp4
|   |
```

```
|   +---1_Bonus_DetekceHrany
|   |       1_Bonus_DetekceHrany.ino
|   |
|   +---1_Bonus_Sumo
|   |       1_Bonus_Sumo.ino
|   |
|   \---Primitivni_Sledovani_Cary
|           Primitivni_Sledovani_Cary.ino
|
+---6_lab_uloha
|   |   bluetooth_fritzing.png
|   |   bluetooth_sketch.png
|   |   bluetooth_video.mp4
|   |   dratove_ovladani_video.AVI
|   |   drat_fritzing.png
|   |
|   +---1_Bonus_Dratove_Ovladani
|   |       1_Bonus_Dratove_Ovladani.ino
|   |
|   +---2_Bonus_Bluetooth
|   |       2_Bonus_Bluetooth.ino
|   |
|   +---2_Bonus_Bluetooth_Test
|   |       2_Bonus_Bluetooth_Test.ino
|   |
|   +---DalkoveOvladani
|   |       DalkoveOvladani.ino
|   |
|   \---IRemote_test
|           IRemote_test.ino
|
+---7_lab_uloha
|   |   SledovaniCaryRegulator.AVI
|   |
|   \---SledovaniCaryRegulator
```

```
|           SledovaniCaryRegulator.ino
|
+---8_lab_uloha
|   |   Draha1.png
|   |   Draha2.png
|   |   PIDregulator_na_motorove_desce.AVI
|   |
|   +---My_Line_Follow_Robot_Control
|   |       My_Line_Follow_Robot_Control.ino
|   |
|   +---My_Line_Follow_Robot_Motor_Core
|   |       MyArduinoRobotMotorBoard.cpp
|   |       MyArduinoRobotMotorBoard.h
|   |       My_Line_Follow_Robot_Motor_Core.ino
|   |
|   \---My_Robot_Motor_Core
|           MyArduinoRobotMotorBoard.cpp
|           MyArduinoRobotMotorBoard.h
|           My_Robot_Motor_Core.ino
|
+---bonusove_ulohy
|   |   Pixy2_video.AVI
|   |   vodovaha.jpg
|   |   vodovaha_video.3gp
|   |
|   +---bonus_pixy2
|   |       bonus_pixy2.ino
|   |
|   \---bonus_vodovaha
|           bonus_vodovaha.ino
|
+---Eagle
|       5VaGND_cb.png
|       5V_cb.png
|       Arduino-Robot-Control-board-Rev7-SCH.brd
```

```
|       Arduino-Robot-Control-board-Rev7-SCH.pdf
|       Arduino-Robot-Control-board-Rev7-SCH.sch
|       Arduino-Robot-Motor-Board-Rev7-SCH.pdf
|       Arduino-Robot-Motor-Board-Rev7.brd
|       Arduino-Robot-Motor-Board-Rev7.sch
|       base.png
|       eeprom.png
|       GND_cb.png
|       horni_deska.png
|       I2C_cb.png
|       kompas.png
|
+---Fritzing
|   |   Arduino Robot Control Board ( Rev7 ).fzpz
|   |
|   \---parts
|
+---Obsah_SD_karty
|       chase.sqm
|       inicio.bmp
|       init.bmp
|       init2.bmp
|       init3.bmp
|       init4.bmp
|       intro.bmp
|       kt1.bmp
|       kt2.bmp
|       lf.bmp
|       lg0.bmp
|       lg1.bmp
|       melody.sqm
|       menu.sqm
|       pb.bmp
|       pic1.bmp
|       pic2.bmp
```

```
|       pic3.bmp
|       pic4.bmp
|       rescue.bmp
|
+---Robot_Control_Base
|       Robot_Control_Base.ino
|
+---Robot_Motor_Core
|       Robot_Motor_Core.ino
|
\---Wordpress
        ArduinoRobotDB.sql
        ArduinoRobotWP.zip
        navod_k_nasazeni.txt
```