VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

PROGRAMOVÁNÍ ARDUINO POMOCÍ MATLAB/SIMULINK

ARDUINO PROGRAMING WITH MATLAB/SIMULINK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Jan Bartoněk
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Radomil Matoušek, Ph.D.
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Jan Bartoněk**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Programování Arduino pomocí Matlab/Simulink

Stručná charakteristika problematiky úkolu:

Prostředí Matlab/Simulink má integrovatelnou podporu pro velmi známou vývojovou platformu Arduino. Příprava edukativně zajímavých úloh pro tuto platformu i rešerše stávajících možností, bude předmětem této práce.

Cíle bakalářské práce:

- Rešerše Arduino a možností vývoje pod Matlab/Simulink.
- Návrh několika úloh a jejich popis.
- Fyzická realizace hw.
- Demonstrační video výsledků práce.

Seznam doporučené literatury:

SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer Press, 2016. ISBN 978-80-251-4840-2.

VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Obsahem této bakalářské práce je rešerše možností programování mikrokontrolerů Arduino prostřednictvím programu Matlab/Simulink. Součástí práce je také stručná rešerše platformy Arduino. Dále byly vytvořeny a popsány dvě praktické úlohy, které mohou sloužit k edukativním účelům. První z nich je aplikace demonstrující využití DHT22 senzoru, vytvořená pomocí App Designeru, což je součást Matlabu. Druhá úloha reprezentuje regulaci polohy levitujícího tělesa v trubici s využitím Simulinku. K oběma úlohám byla vytvořena demonstrační videa a popis, který je součástí této práce.

ABSTRACT

The content of this bachelor´s thesis is a research of possibility of programming an Arduino microcontroller using Matlab/Simulink program. Part of the thesis is also a brief research of the Arduino platform. Furthermore, two practical tasks were created and described in this work, which can be used for educational purposes. The first being an application demonstrating the use of the DHT22 sensor, created using App Designer, which is part a of Matlab. The second task deals with the regulation of the position of a levitating body in the tube, using Simulink. Demonstration videos were created for both tasks and are part of this work.

KLÍČOVÁ SLOVA

Arduino, Matlab, Simulink, App Designer, senzor, měření teploty, měření vlhkosti, ultrazvukový senzor, vzduchová levitace, PID regulátor

KEYWORDS

Arduino, Matlab, Simulink, App Designer, sensor, temperature measurement, humidity measurement, ultrasonic sensor, air levitation, PID controller

BIBLIOGRAFICKÁ CITACE

BARTONĚK, Jan. *Programování Arduino pomocí Matlab/Simulink*. Brno, 2020. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/125036. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky. Vedoucí práce Radomil Matoušek.

PODĚKOVÁNÍ

Rád bych poděkoval mé rodině a přátelům za podporu a trpělivost, kterou se mnou při psaní této práce měli. Dále bych chtěl poděkovat doc. Ing. Radomilu Matouškovi, Ph.D. za odborné vedení mé práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Radomila Matouška, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 22. 5. 2020 .....

Jan Bartoněk

OBSAH

1 ÚVOD

Obsahem této bakalářské práce bylo využití platformy Arduino prostřednictvím prostředí Matlab/Simulink a vytvoření dvou edukačně zajímavých úloh. Součástí práce byla i stručná rešerše platformy Arduino. Práce je doplněna demonstračními videi k realizovaným úlohám.

Práce je rozdělena do čtyř kapitol. První dvě se věnují rešeršní části. Kapitola jedna detailně popisuje platformu Arduino. Kapitola dvě je zaměřena na program Matlab/Simulink a na jeho knihovny, jejichž prostřednictvím je možné programovat desky Arduino.

Ve druhé polovině této bakalářské práce se autor věnuje dvěma praktickým úlohám. Každá z nich obsahuje popis, schéma zapojení hardwaru, objasnění a zdůvodnění použitého řešení. Na konci obou kapitol se nachází vyhodnocení konkrétní úlohy.

První praktický úkol využívá Matlab a jeho doplněk App Designer k vytvoření jednoduché aplikace, která slouží k měření teploty a vlhkosti vzduchu v místnosti pomocí senzoru DHT22. Aplikace vykresluje teplotu i vlhkost v reálném čase a při určité teplotě spustí ventilátor připevněný k servomotoru. Motor se začne otáčet a uživatel může regulovat jak rychlost jeho otáčení, tak otáčky ventilátoru. Po ochlazení se ventilátor i servomotor zastaví. Matlab v tomto případě provádí veškeré výpočty a Arduino tedy slouží jako levná měřící karta.

Druhá praktická úloha se zabývá regulací polohy tělesa v trubici. Trubice je spojená s ventilátorem pomocí nástavce, který byl vymodelován v Inventoru a následně vytisknut na 3D tiskárně. V horní části trubice je ultrazvukový senzor HC-SR04, který zjišťuje aktuální pozici tělesa. Soustava je řízena pomocí PID regulátoru. Ten zajišťuje, aby požadovaná výška tělesa, kterou si určí uživatel potenciometrem, byla stejná jako výška reálná. Na rozdíl od první úlohy využívá tato úloha Simulink a jeho knihovny. Díky tomu je možné program nahrát přímo na desku Arduino a není potřeba, aby byla propojena s počítačem.

Spojení Arduino a Matlab je diskutováno mnoha autory, přičemž existuje velké množství prací, které pojednávají o tomto tématu. Uveďme například [1], [2], [3].

Téma bakalářské práce jsem si vybral hlavně z důvodu, že programování Arduina je mým koníčkem, a tudíž mě psaní bavilo. Neměl jsem však větší zkušenosti s využitím Matlabu/Simulinku jako nástroje pro programování Arduina, což jsem považoval za výzvu i nový zdroj poznání.

2 ARDUINO

Arduino je open-source elektronická platforma založená na mikrokontrolerech (MCU) ATmega od firmy Atmel. Je určené hlavně pro výuku studentů na školách, ale používají ho i inženýři, umělci, programátoři, nebo lidé, kteří mají elektrotechniku jako koníček [4]. Jedná se o malý jednodeskový počítač. Arduino je především vhodné pro úplné začátečníky v oblasti programování mikroprocesorů, kvůli velice široké základně uživatelů a s tím spojeným velkým počtem detailních návodů. Ale jak již bylo zmíněno, není jen pro studenty a najde uplatnění i v komplexních projektech. Na obr. 1 je zobrazeno komunitní logo Arduina, které znázorňuje skutečnost, že je platforma Arduino open-source [4]. Místo vpravo dole si každý uživatel může změnit podle jeho potřeby.

Obr. 1: Komunitní logo Arduina [4]

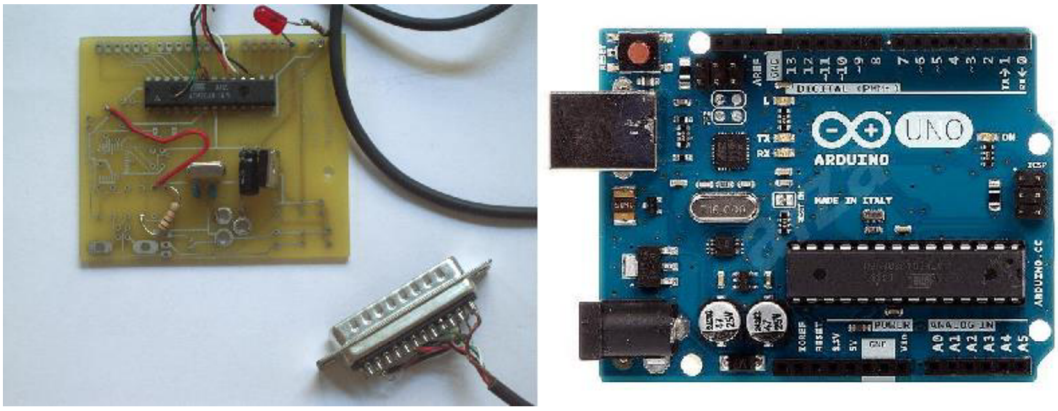
2.1 Historie Arduina

Označení Arduino je převzato z názvu italského baru ve městě Ivrea. Bar di Re Arduino, který je pojmenován na počest italského krále Arduina Ivrejského, který vládl v Itálii v letech 1002-1004. Právě tento bar byl oblíbeným místem Massimoha Banzi, který společně s jeho tehdejším studentem Davidem Cuartiellesem založili v roce 2005 projekt Arduino. [5], [6]

V době, kdy Arduino vzniklo, působil Massimo Banzi jako učitel na IDII (Interaction Design Institute Irvea). Banzi, stejně jako ostatní učitelé spoléhal, při výuce svých studentů, na mikrokontroler BASIC Stamp, od californsnké firmy Parallax [6]. Bohužel měl tento mikrokontroler dva zásadní problémy. Neměl dostatečný výpočetní výkon na projekty, které chtěli Banziho studenti vytvořit a také byl poměrně drahý. Stál přibližně 100 amerických dolarů. Proto Banziho napadlo vytvořit mikrokontroler, který by tyto problémy neměl. Dal si za cíl vyvinout zařízení, které je jednoduché, snadno připojitelné k různým senzorům, motorům nebo relé, snadno programovatelné a finančně dostupné pro jeho studenty [5].

Banzi se svými kolegy vytvořil Arduino na základě Wiringu, open-source frameworku, který vyvinul roce 2003 Hernando Barragán. Jedná se o elektronickou platformu složenou z programovacího jazyka a vlastním IDE (Integrated Development Environment), určenou k programování mikroprocesorů [7]. Samotný Wiring je odvozený od dalšího open-source projektu, Processingu. Což je grafické vývojové prostředí z roku 2001, určené pro lidi, kteří nejsou zdatní programátoři, ale přes to by rádi vytvářeli zajímavé projekty [8].

První prototyp vytvořil Banzi s kolegy v roce 2005. Název Arduino, nedostal hned, ale až později téhož roku. Jelikož měl velký úspěch mezi Banziho studenty, poměrně za krátkou dobu, v roce 2006, vznikla první oficiální Arduino deska – Arduino Seriál. Jak už název napovídá, deska používala ke komunikaci mezi chipem a počítačem sériový port, neměla totiž ještě zabudované USB (Universal Seriál Bus) rozhraní. Její cena byla přibližně 30 amerických dolarů, což byl oproti mikrokontrolerům BASIC Stamp značný rozdíl [5]. Banzi oproti ostatním výrobcům jednodeskových počítačů nešetřil na počtu vstupních a výstupních pinů, konkrétně na Arduinu Serial je 14 digitálních pinů, 6 analogových pinů a 4 napájecí. Dalším odlišným prvkem byla barva – Arduino používá modrou barvu a na zadní straně každého originálního Arduina je malá mapa Itálie. Arduino Serial je založeno na mikroprocesoru Atmega8, který ovšem kvůli své nedostatečné paměti brzy přestal uživatelům stačit. Proto byly další desky osazovány čipy s větší pamětí, jako Atmega168 nebo Atmega328, který se používá dodnes v nejrozšířenější desce – Arduinu UNU [6]. Porovnání mezi prvním prototypem a nejpoužívanější Arduino deskou je zobrazen na obr. 2 níže.



Obr. 2: Porovnání prvního prototypu s Arduinem UNO [4],[5]

Tvůrcům Arduina se povedlo vytvořit jednodeskový počítač, který si našel fanoušky všech věkových kategorií, po celém světě. Splnili cíle, které si dali. Jejich produkt je levný, takže si ho mohou pořídit i studenti, je intuitivní a kompatibilní s nejrůznějšími senzory, motory atd. Skutečnost, že je Arduino kompletně open-source, dala vzniknout komunitě, která mezi sebou sdílí své unikátní projekty a navzájem si pomáhá. Člověk si může dokonce sestavit i své vlastní Arduino, podle návodů a schémat, které jsou volně dostupné na internetu. Tím vznikají takzvané klony, které často originální Arduino i vylepšují.

2.2 Typy Arduino desek

Jak již bylo zmíněno na začátku této kapitoly, platforma Arduino využívá čipy ATmega od firmy Atmel. Součástí desky Arduino obr. 2 jsou například LED (Light-Emitting Diode) diody, tlačítko pro restart programu, vstupní a výstupní piny, z nichž některé se dají používat pro PWM (Pulse Width modulation) řízení, dále napájecí konektor nebo konektor USB pro komunikaci s počítačem. Některé Arduino desky mají za svým názvem například Rev3 nebo R3. Jedná se o označení verze jednotlivých desek. Rozdíly mezi verzemi neovlivňují rozložení pinů. Maximální proud, který mohou desky Arduino odebírat, závisí na typu připojení, které uživatel zvolí. V tab. 1 jsou tyto hodnoty zobrazeny.

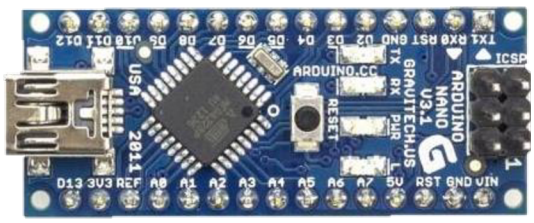
Tab. 1: Informace o napájení Arduino desek [10]

Typ připojení	Rozsah napájecího napětí [V]	Max. proud [mA]
Souosý konektor	6 až 15 (20)	1000
USB port	4,75 až 5,25	500
VIN+GND	5,8 až 14,8 (19,8)	1000

V další části si představíme základní typy Arduino desek. S ohledem na to, že je těchto platforem mnoho, byly vybrány pouze ty nejpoužívanější.

2.2.1 Arduino NANO

Arduino NANO, které můžeme vidět na obr. 3, patří, jak název napovídá, k nejmenším deskám této platformy. Menší už je jen Arduino Mini. Tyto dvě desky jsou si velice podobné, hlavním rozdílem je to, že NANO má zabudovaný USB převodník. NANO je mezi uživateli Arduina velice oblíbené. Díky svým minimalistickým rozměrům se hodí prakticky do každého projektu. Přes svou malou velikost má NANO dokonce o dva analogové a osm digitálních pinů více, jako Arduino UNO. Postrádá jen speciální konektor na napájení [9]. Díky své velikosti je také kompatibilní jak s nepájivým polem, tak s běžně používanými deskami plošných spojů.



Obr. 3: Arduino NANO [4]

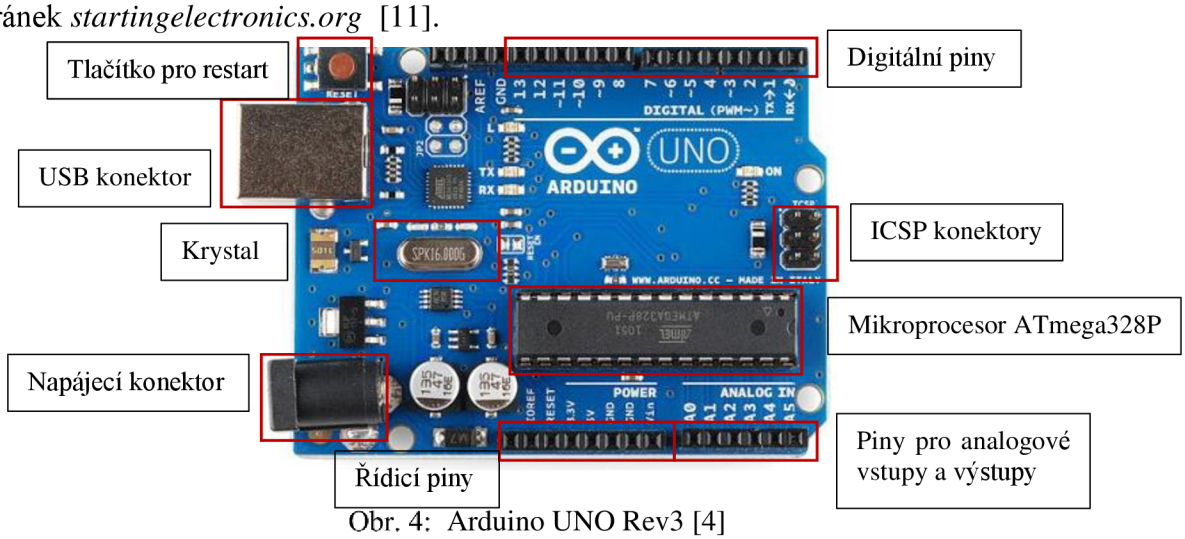
Jeho „srdcem“ je mikrokontroler ATmega328, nabízející dostatečnou paměť k většině projektů. Ostatní technické specifikace jsou uvedeny v tab. 2.

Tab. 2: Technické specifikace Arduino NANO [4]

Mikroprocesor	ATmega328P
Provozní napětí	5 V
Vstupní napětí	7-12 V
Flash paměť	32 KB, z nichž 2 KB používá bootloader
SRAM	2 KB
EEPROM	1 KB
Krystal	16 MHz
Analogové vstupní piny	8
Digitální I/O piny	22 z toho 6 PWM
Stejnosměrný proud na jeden I/O pin	40 mA
Spotřeba energie	19 mA
Velikost	18 x 45 mm
Váha	7 g

2.2.2 Arduino UNO Rev3

V dnešní době nejpoužívanější Arduino deska je Arduina UNO Rev3, zobrazená a popsaná na obr. 4. Jak již bylo zmíněno výše, „Rev3“ značí třetí a zatím nejnovější verzi desky. Rozdílů mezi prvním vydáním a verzí číslo tři je několik. Například pozice tlačítka pro restart, které bylo přesunuto vedle USB konektoru pro lepší kompatibilitu Arduina s jeho takzvanými *Shieldy* (pozn. běžně užívaný pojem pro přídavné rozhraní, které umožňuje připojení přídavných periferií jako je například H-můstek pro motory, akcelerometr a další). O Shieldech je v této práci samostatná podkapitola níže. Další změna byla přidání celkem čtyř pinů. Dva pro komunikaci I2C (Inter-Integrated Circuit), tedy SCL (Seriál Data Line) a SDA (Serial Clock Line). Tyto dva piny jsou propojeny s analogovými piny A4 a A5. Dále byl přidán pin IOREF (Input/Outout Reference), který zajišťuje, aby bylo napětí Shieldu a Arduino stejné. Poslední přidaný pin byl zanechán nevyužitý. Ostatní změny jsou k dohledání v jednom z článků internetových stránek *startingelectronics.org* [11].



Obr. 4: Arduino UNO Rev3 [4]

To, že je Arduino UNO nejpoužívanější deska z „rodiny“ Arduino, je zapříčiněno tím, že je na internetu nejvíce zdokumentované. Je tedy vhodné především pro začátečníky, ale využijí je i zkušenější uživatelé.

UNO znamená v italštině jedna. Toto označení bylo vytvořeno společně s verzí 1.0 IDE pro Arduino. UNO je také první v řadě Arduino desek, které má USB rozhraní [12].

Z této vývojové desky se postupně vyvinuly další dvě speciální. Arduino Ethernet a Arduino Bluetooth. První zmíněná deska je prakticky stejná jako UNO, jen s rozdílem, že místo USB portu je na desce připájen Ethernet port a najdeme na ní slot pro SD kartu [9]. Druhá deska, jak název napovídá, také nepoužívá USB, ale Bluetooth.

Arduino UNO se tedy dá napájet buď přes USB rozhraní nebo přes AC/DC adaptér anebo přes externí baterii. Technické specifikace jsou, stejně jako u výše popsaného NANA, v tab. 3.

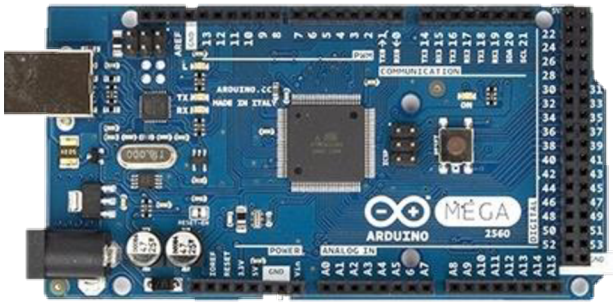
Tab. 3: Technické specifikace Arduino UNO Rev3 [4]

Mikroprocesor	ATmega328P
Provozní napětí	5 V
Vstupní napětí (doporučené)	7-12 V
Vstupní napětí (limitní)	6-20 V
Flash paměť	32 KB, z nichž 0,5 KB používá bootloader
SRAM	2 KB
EEPROM	1 KB
Krystal	16 MHz
Zabudované LED	13
Analogové vstupní piny	6
Digitální I/O piny	16 z toho 6 PWM
Stejnosměrný proud na jeden I/O pin	20 mA
Stejnosměrný proud pro 3,3 V pin	50 mA
Velikost	53,4 x 68,6 mm
Váha	25 g

2.2.3 Arduino MEGA2560

Posledním typem Arduino vývojové desky, která bude v této práci více popsána, je Arduino MEGA2560. Jeho předchůdcem je Arduino MEGA1280, které bylo osazeno čipem ATmega1280. Z důvodu potřeby většího výpočetního výkonu se však přešlo na mikrokontroler ATmega2560 [9].

Design Arduina MEGA vznikl prodloužením Arduina UNO. Díky větším rozměrům, bylo možné na desku umístit výkonnější čip, o němž byla řeč v minulém odstavci a také větší počet pinů – na desce je 54 digitálních a 16 analogových.



Obr. 5: Arduino MEGA2560 [4]

Arduino MEGA2560 je zobrazeno na obr 5 a jeho technické specifikace můžeme opět, jako v předchozí podkapitole, vidět v tab. 4.

Tab. 4: Technické specifikace Arduino MEGA2560 [4]

Mikroprocesor	ATmega2560
Provozní napětí	5 V
Vstupní napětí (doporučené)	7-12 V
Vstupní napětí (limitní)	6-20 V
Flash paměť	256 KB, z nichž 8 KB používá bootloader
SRAM	8 KB
EEPROM	4 KB
Krystal	16 MHz
Zabudované LED	13
Analogové vstupní piny	16
Digitální I/O piny	54 z toho 15 PWM
Stejnosměrný proud na jeden I/O pin	20 mA
Stejnosměrný proud pro 3,3 V pin	50 mA
Rozměry	53,3 x 101,52 mm
Váha	37 g

2.2.4 Ostatní Arduino desky

Za zmínku jistě stojí i Arduino Due, typ vývojové desky, která navazuje na řadu Mega. Hlavním rozdílem je implementace mnohonásobně výkonnějšího čipu – Atmel SAM3X8E. Tento procesor má 32bitové jádro a jeho taktovací frekvence je 84Mhz. Díky těmto parametrům se jedná o nejvýkonnější vývojovou desku z platformy Arduino [9].

Následující dvě Arduino desky, které zde budou stručně popsány, byly vybrány díky svým specifickým účelům. Jsou jimi Arduino Lilypad a Arduino Esplora.

První zmíněná deska je určená k výrobě takzvaných *e-textilií* [12]. Je možné ji přišít k oblečení pomocí vodivé nitě, čímž se dají vytvořit velmi zajímavé projekty jako například svítící plesové šaty, či vesta pro cyklisty, která ukazuje změnu směru jízdy.

Arduino Esplora je stejně jako výše zmíněný Lilypad výrazné svým vzhledem, ale hlavně tím, že je v něm zabudovaný joystick, tlačítka, potenciometr, bzučák, teploměr a tříosý akcelerometr. Nechybí zde ani piny pro připojení LCD (Liquid-Crystal Display) displeje [9]. Jedná se o hybridní vývojovou desku a s její pomocí se dá vytvořit například samostatný herní set.

2.3 Arduino Shieldy

Shieldy jsou desky, který mají své piny, podobně jako Arduino Nano, vyvedené směrem dolů. Dají se tedy zasunout do Arduino desek. Slouží k rozšíření možných způsobů využití Arduina [6]. Existuje celá řada Arduino Shieldů, každý z nich má specifické vlastnosti a slouží k jiným účelům.

Například Motor Shield. Jak již název napovídá, jedná se o Shield, který umožňuje připojení a ovládání motorů. Dá se k němu připojit jeden bipolární krokový motor, nebo dva DC motory [12].Uživatel ním může řídit jejich rychlost i směr otáčení.

Dalším příkladem je Ethernet Shield, díky němuž je Arduino schopno komunikovat prostřednictvím LAN (Local Area Network). Je v něm zabudován i slot na SD (Security digital) kartu, pomocí které je Shield schopný ukládat velké množství naměřených dat. [12]

Jako poslední zde popsaný Adruino Shield byl vybrán Wi-Fi Shield obr. 6. Tento Shield je velmi oblíbený, protože, jak je podle názvu zřejmé, umožní Arduinu bezdrátovou komunikaci přes Wi-Fi. Díky bezdrátové komunikaci poté může uživatel vytvořit například svou vlastní *chytrou* domácnost. V Shieldu je také zabudován slot na SD kartu.

Je důležité, aby si před použitím Shieldu dal uživatel pozor a zkontroloval si, jestli je Shield kompatibilní právě s jeho vývojovou deskou Arduino.

Obr. 6: Arduino Wi-Fi Shield [4]

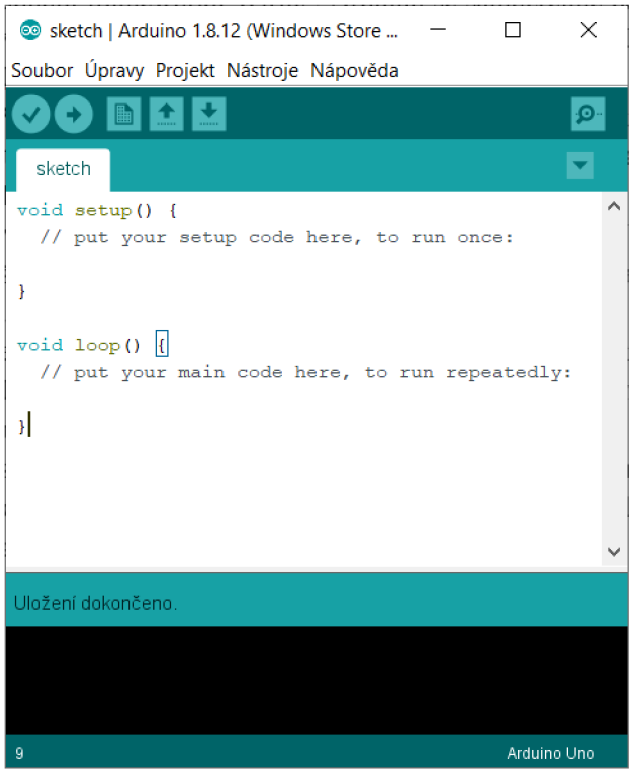
2.4 Arduino IDE

Jedná se o oficiální vývojové prostředí pro programování platformy Arduino. Jak již bylo zmíněno, je napsané v programovacím jazyce Java, vzniklo z Proccesingu a je v něm implementována podpora frameworku Wiring [13]. Arduino se samozřejmě dá programovat i v jiných vývojových prostředích, jako jsou například Stino – A sublime Text Plugin for Arduino, Visual Micro – Arduino IDE for Microsoft Visual Studio and Atmel Studio nebo v Matlabu. Právě Matlabu a jeho knihovně Simulink je věnovaná následující kapitola a oba praktické úkoly této práce jsou řešeny pomocí tohoto programu.

Arduino IDE je díky své jednoduchosti a přehlednosti jednoznačně nejpopulárnější vývojové prostředí pro programování Arduino desek. Jeho nevýhoda je v tom, že není dobrý editor zdrojového kódu, například nezvládá našeptávat nebo nevhodně odsazuje.

Aktuální verze Arduino IDE je 1.8.12 a je dostupná pro operační systémy Windows, Linux i Mac OS X. Je volně ke stažení na oficiálních stránkách Arduina [4], kde je k dispozici i Arduino Web Editor, díky němuž nemusí uživatel stahovat IDE, ale může začít programovat svoje Arduino online skrze webový prohlížeč.

Arduino IDE ve své základní podobě, tak jak si ho uživatel stáhne z internetu, nabízí velké množství příkladů (Soubor – Příklady), na nichž si začátečník může vyzkoušet možnosti, které Arduino nabízí, ale hlavně obsahuje předinstalované knihovny k různým senzorům, servomotorům či LCD displeji. Nejsou zde ovšem veškeré knihovny. Ty, které si uživatel bude chtít přidat, musí umístit do složky *libraries*, kterou najde v adresáři, kam nainstaloval Arduino ID.



Obr. 7: Arduino IDE

Na obr. 7 je zobrazeno, jak vypadá Arduino IDE při prvním spuštění. Vlevo nahoře je umístěno tlačítko *Ověřit*, které po stisknutí zkontroluje kód programu. Pokud je nalezena chyba, je následně zvýrazněna v příslušné části kódu. Vedle tlačítka *Ověřit* je tlačítko s názvem *Nahrát*, které zdrojový kód zkontroluje a pokud je bezchybný, nahraje jej na příslušnou Arduino desku. Další tlačítka pojmenovaná *Nový*, *Otevřít* a *Uložit* není třeba nějak popisovat, dělají přesně to, co mají v názvu. Za zmínku stojí ikona, která je vpravo nahoře. Jedná se o takzvaný *Sériový monitor*, na němž je možné vidět například výstupy z různých senzorů nebo přes něj může uživatel posílat data do Arduina. Je to tedy vstupně výstupní periferie.

Jak je vidět na obr. 7, program je rozdělen do dvou částí. Funkce *void setup()* se vykoná vždy pouze jednou při nahrání programu do Arduina nebo při stisknutí tlačítka restart. Slouží k inicializaci proměnných a nastavení módu pinů. Ve druhé funkci *void loop()* se nachází kód, který se bude opakovat do té doby, než uživatel odpojí Arduino od napájení. Obě tyto funkce jsou nutnou podmínkou pro to, aby program fungoval – musí být v kódu obsaženy, i když jsou prázdné, jinak program skončí chybou.

3 MATLAB

Matlab je programovací jazyk a interaktivní vývojové prostředí. Jeho název vznikl spojením prvních slabik slov *matrix laboratory* (maticová laboratoř) – zprvu byl především určen pro práci s maticemi, avšak dnes již zvládá mnohem více užitečných funkcí. První komerční verze byla představena v roce 1985 společností MathWorks. Do té doby byl Matlab zdarma a využívali ho zejména studenti Cleva Molera, profesora na Univerzitě v Novém Mexiku, autora původní verze Matlabu a spoluzakladatele firmy MathWorks. Matlab je napsaný pomocí programovacích jazyků C/C++, Fortranu a Javy. [14] Jeho aktuální verze je R2020a. V dnešní době je využívaný především inženýry, akademickými pracovníky a studenty. Jak již bylo zmíněno, tento program není zdarma a cena licence pro domácí účely začíná na 119 euro [15]. Ovšem díky Matlab Campus Wide licenci je Matlab pro studenty a zaměstnance VUT bezplatný a praktická část této práce je řešena právě pomocí ní.

3.1 Simulink

Simulink je grafické prostředí pro modelování a simulaci dynamických systémů. Jedná se o nadstavbu Matlabu umožňující uživateli vytvářet bloková schémata, znázorňující reálné systémy. Je možné v něm modelovat například fyzikální soustavy, systémy pro zpracovávání signálů či algoritmy řídicích systémů. [16]

Je možné jej spustit v kartě HOME, kliknutím na jeho ikonu nebo zadáním *simulink* do *Command Window*. Obsahuje předinstalované knihovny rozdělené podle jejich účelu, například matematické operace nebo zdroje signálu. Jednotlivé funkce knihoven jsou znázorněny bloky, které uživatel intuitivně skládá za sebe. Velká výhoda Simulinku je, že pokud si uživatel neví s nějakým blokem rady, stačí na něj kliknout pravým tlačítkem a vybrat *Help*, objeví se podrobná nápověda, kde je vše důkladně vysvětleno.

Matlab i Simulink mohou být využity pro komunikaci s rozličným hardwarem, přičemž hobby platformy Arduino, Raspberry Pi nebo LEGO MINDSTORMS (EV3) jsou zdarma. Tato práce se zabývá programováním Arduina, tudíž ostatní výše zmíněný hardware zde není více popsán. Pro práci s Arduinem je nutné nainstalovat dvě rozšíření distribuovaná společností MathWorks, která jsou popsána v podkapitolách níže. Jedná se o Matlab Support Package for Arduino Hardware a Simulink Support Package for Arduino Hardware.

3.2 Matlab Support Package for Arduino Hardware

Toto rozšíření umožňuje sériovou komunikaci mezi Arduinem a Matlabem. Uživatel může zapisovat nebo číst data ze senzorů, nebo například pohybovat servomotorem přes Arduino a ihned vidí výsledky skrze Matlab, aniž by proběhla kompilace.

Instalace tohoto rozšíření je velice jednoduchá. V Matlabu stačí rozkliknout kartu *HOME*, poté položku *Add-Ons* a z ní vybrat *Get Hardware Support Packages*. Otevře se okno *Add-Ons*, kde si uživatel vybere požadované rozšíření a zvolí *Install*. Je ovšem nutné být přihlášený přes MathWorks účet, který je pro studenty VUT stejný jako jejich osobní účet. Bez přihlášení není možné rozšíření nainstalovat. Uživatel se může přesvědčit, že instalace proběhla v pořádku tak, že v *Add-Ons* vybere *Manage Add-Ons*, v nově otevřeném okně klikne pravým tlačítkem na vybrané rozšíření a zvolí Setup. V něm následně otestuje, že je Arduino správně připojeno k počítači, na jakém je portu a nahraje do něj program, který rozbliká zabudovanou LED diodu v desce.

Po instalaci a propojení Arduina s počítačem pomocí USB je vše připraveno a uživatel se může pustit do programování. Arduino je možné připojit také pomocí Bluetooth nebo Wi-Fi. Tyto dvě možnosti nebyly v praktické části využity, proto se o nich v této práci autor více nezmiňuje. Dokumentace k nim je dostupná na internetových stránkách společnosti MathWorks [17].

Při použití Matlab Support Package for Arduino Hardware je Arduino využito jako vstupně výstupní zařízení, neprobíhá kompilace kódu a všechny výpočty a zpracování probíhají v Matlabu na procesoru počítače. Desky Arduino se v tomto případě dají využít jako levné měřící karty. Díky Matlabu je možné rychle vizualizovat naměřená data pomocí jeho funkcí na vykreslování grafů [18]. Je ovšem nutné, aby bylo Arduino připojeno k počítači.

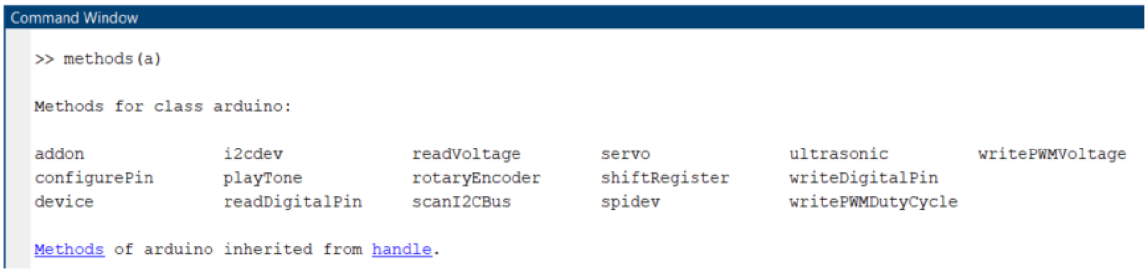
3.2.1 Funkce pro Matlab Support Package for Arduino Hardware

Pro práci s Arduinem je prvně nutné vytvořit objekt, který reprezentuje Arduino. Poté je možné využít celou řadu funkcí pro práci s Arduinem. Níže je popsána zmíněná funkce zajišťující propojení Arduina s Matlabem a také funkce nejpoužívanější.

- *a = arduino()* vytvoří objekt reprezentovaný identifikátorem *a*, která reprezentuje Arduino a spustí komunikaci mezi Arduinem a Matlabem. V *Command Window* se zobrazí parametry daného Arduina jako jsou: port, název desky, dostupné piny a knihovny. Při použití Arduino klonu je nutné k funkci připsat dva parametry – port a název desky. Funkce poté může vypadat například takto: *a = arduino ('COM3', 'Uno')*.
- *readDigitalPin(a, pin)* a *writeDigitalPin(a, pin)*, slouží k přečtení nebo zapsání logické nuly či jedničky na jeden z digitálních pinů.
- *writePWMVoltage(a, pin, voltage)* slouží pro přidělení přesné hodnoty napětí na PWM pin. Maximální hodnota napětí je k dohledání v datasheetu příslušné Arduino desky, ovšem většinou se rozmezí napětí pohybuje mezi 0 až 5 volty.
- *writePWMDutyCycle(a, pin, dutyCycle)* je obdobou předchozí funkce. Slouží k nastavení hodnoty PWM signálu v rozmezí od 0, kdy je *dutyCycle* 0,00 %, až do 1, kdy je *dutyCycle* 100 %.
- *readVoltage(a, pin)* slouží k přečtení hodnoty napětí na určitém analogovém pinu. Využívá se například při načtení polohy potenciometru.

- *ultrasonic(a, triggerPin, echoPin)* umožnuje práci s ultrazvukovým senzorem. Vytvoří se objekt, se kterým může uživatel dále pracovat.
- *readDistance(u)* slouží k určení vzdálenosti mezi ultrazvukovým senzorem a měřeným objektem v metrech. Parametrem této funkce je objekt vytvořený předchozí funkcí.

Pro komunikaci mezi Matlabem a Arduinem existuje mnoho dalších zajímavých funkcí, které zde nejsou popsány. Důvodem je především to, že většina z nich nebyla v praktické části této práce využita a také to, že je jich značné množství a nebylo by možné z rozsahových důvodů popsat všechny. Jejich seznam však může uživatel jednoduše zobrazit pomocí funkce *methods(a)*. Výpis funkcí pro programování Arduino s použitím přednastavených knihoven I2C, SPI (Serial Peripheral Interface) a Servo je zobrazen na obr. 8. Veškerá dokumentace k těmto i ostatním funkcím je dohledatelná na stránkách společnosti MathWorks [17].



Obr. 8: Funkce *methods(a)*

3.3 Simulink Support Package for Arduino Hardware

Jedná se o rozšíření, které umožňuje programování Arduino v Simulinku. Tento podpůrný balíček byl využit v druhém úkolu praktické části této bakalářské práce. Jeho instalace je obdobná jako instalace rozšíření, které je popsáno výše, tudíž zde nebude znovu rozepsána. Je však důležité provést správnou konfiguraci balíčku pro Arduino desku, kterou bude uživatel používat.

Po spuštění Simulinku a připojení Arduina k počítači je nutné v kartě *HARDWARE* kliknout na *Hardware Setings*. Tím se zobrazí nastavení, ve kterém uživatel vybere *Solver*. Zde nastaví *stop time* na *inf*. Díky tomu se spuštěná simulace bude vykonávat teoreticky do nekonečna. Poté uživatel klikne na *Hardware Implementation*, kde zadá název používané Arduino desky, následně rozklikne rozbalovací menu *Target hardware resources*, zde klikne na *Host-board connection* a vybere port, kterým je Arduino připojeno k jeho počítači. Výchozí hodnota je nastavena na *Automatically*, ovšem může se stát, že Simulink port nenajde, proto je vhodné využít přiřazení čísla portu manuálně. Dále je vhodné zkontrolovat, zda je přenosová rychlost (baud rate) Arduino desky shodná s nastavením v Simulinku. To se provede kliknutím na záložku *Serial port properties*, která se nachází ve stejném menu jako nastavení portu a zde uživatel vybere požadovaný baud rate.

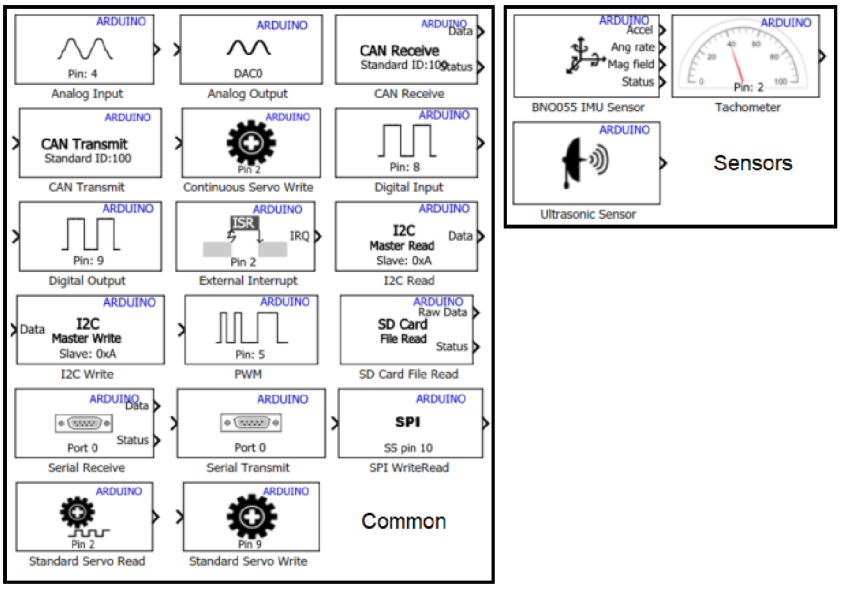
Hlavním rozdílem oproti balíčku pro programování Arduina skrze Matlab je to, že zde program běží přímo na Arduino desce [18]. Simulink Support Package for Arduino Hardware funguje tak, že si uživatel vytvoří algoritmus ze speciálních bloků pro Arduino, které nalezne v seznamu knihoven, o nichž byla řeč výše. Samozřejmě do něj může zakomponovat i funkce z ostatních knihoven. Po zhotovení algoritmu a spuštění simulace se automaticky vygeneruje kód, který se nahraje na desku Arduino, kde běží v nekonečné smyčce.

Simulace se dá spustit pomocí tlačítka *Monitor & Tune* (v dřívějších verzích Matlabu – *external mode*), tímto způsobem může uživatel měnit nastavení svého algoritmu přímo při zapnuté simulaci. Pokud je návrh provedení hotov, je možné jej do Arduino desky nahrát pomocí tlačítka *Build Stand-Alone*. Poté je možné Arduino odpojit od počítače a program bude běžet samostatně, nezávisle na Simulinku. Samozřejmě je nutné, aby v tomto případě bylo Arduino napájeno jiným způsobem.

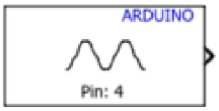
3.3.1 Funkce pro Simulink Support Package for Arduino Hardware

Funkce pro tento rozšiřující balíček jsou velmi podobné těm, které jsou určeny pro balíček Matlab Support Package for Arduino Hardware. Jsou rozřazeny do šesti kategorií – *Common, Ethernet Shield, MKR Motor Center, Sensors, Utilities* a *WiFi*. Praktická část této práce využívá určité funkce z kategorií *Common* a *Sensors*, proto budou více popsány právě funkce z nich. Detailní popis ostatních funkcí je opět dostupný na internetových stránkách společnosti MathWorks [19].

Na obr. 9 jsou zobrazeny všechny funkce z obou zmíněných kategorií. Z nich byly vybrány ty nejdůležitější a ty, které byly využity v této práci. Jejich popis společně s jejich ikonami jsou obsahem následujících odstavců.



Obr. 9: Kategorie *Common* a *Sensors* z knihovny *Simulink Support Package for Arduino Hardware*



Analog Input

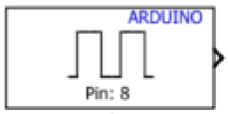
Blok *Analog Input* měří aktuální napětí na daném analogovém pinu, který uživatel nastaví kliknutím na blok v jeho parametrech. Výstup z tohoto bloku je u většiny Arduino desek v rozsahu od 0 do 1023 (10 bit). V případě desky Due nebo Arduino MKR1000 je výstup 12bitový. Jestliže je naměřené napětí rovno napětí na pinu GND (ground), pak je výstup nula. V případě, že je naměřené napětí rovno referenčnímu napětí, které je u většiny desek 5 V, je výstup z bloku Analog Input 1023. Uživatel může dále měnit vzorkovací čas (sample time), ten je při výchozím nastavení jedna sekunda. Jeho minimální hodnota je však 0.000001 s.

Analog Output

Analog Output je blok, který generuje napětí na pinu DAC (Digital to Analog Converter). Ten však není na všech deskách Arduino, nachází se pouze na deskách Due, MKR1000, MKR WIFI 1010 a ZERO. Datový typ vstupu pro tento blok je uint16. U desky Due je vstup12bitový a u všech ostatních výše zmíněných pak 10bitový.

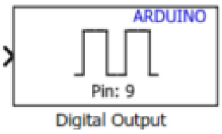
Continuous Servo Write

Dalším zde popsaným blokem je *Continuous Servo Write*. To určuje směr a rychlost otáčení servomotoru. Vstupní hodnoty k tomuto bloku jsou v rozsahu od -90 do 90. Hodnota -90 zapříčiní maximální rychlost otáčení serva na jednu stranu, hodnota 90 poté maximální rychlost otáčení na stranu druhou. Nula servo zastaví. Při zadání hodnoty menší než -90 nebo vyšší než 90 se bude blok chovat stejně, jak by se zadala hodnota maximální.



Digital Input

Blok *Digital Input* zjišťuje, zda je hodnota na zadaném pinu LOW (0 V) nebo HIGH (5 V nebo 3,3 V – záleží na typu desky). Při hodnotě LOW je výstup nula a při hodnotě HIGH jedna. Opět je možné nastavit vzorkovací čas, výchozí hodnota je zde 0,1 s a minimální hodnota je stejná jako u bloku Analog Input. Datový typ výstupu je boolean.



Digital Output

Digital Output funguje na opačném principu než blok předchozí. Jestliže je na vstupu logická nula, nastaví hodnotu na přirazeném digitálním pinu na LOW, v opačném případě, tedy když je na vstupu logická jednička, bude na pinu HIGH.

PWM

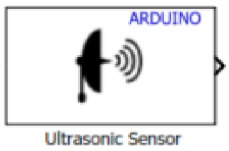
Blok *PWM* generuje, jak je z názvu patrné, PWM signál. Vstupní hodnoty jsou od 0 do 255 což odpovídá 0 až 100 % pracovního cyklu. Stejně jako u *Continuous Servo Write* při zadání hodnoty menší než nula nebo vyšší, než 255 se bude blok chovat, jako by uživatel zadal mezní hodnotu.

Standard Servo Read

Dalším blokem je *Standart Servo Read*, který vrací hodnoty v rozmezí od 0 do 180, které znázorňují pozici natočení servomotoru ve stupních. Datový typ výstupu je uint8 a znovu je zde možné nastavit vzorkovací čas.

Standard Servo Write

Předposledním blokem, který je zde popsaný, je *Standart Servo Write*. Je protikladem k předchozímu bloku. Akceptuje hodnoty od 0 do 180 a následně nastavuje pozici servomotoru.

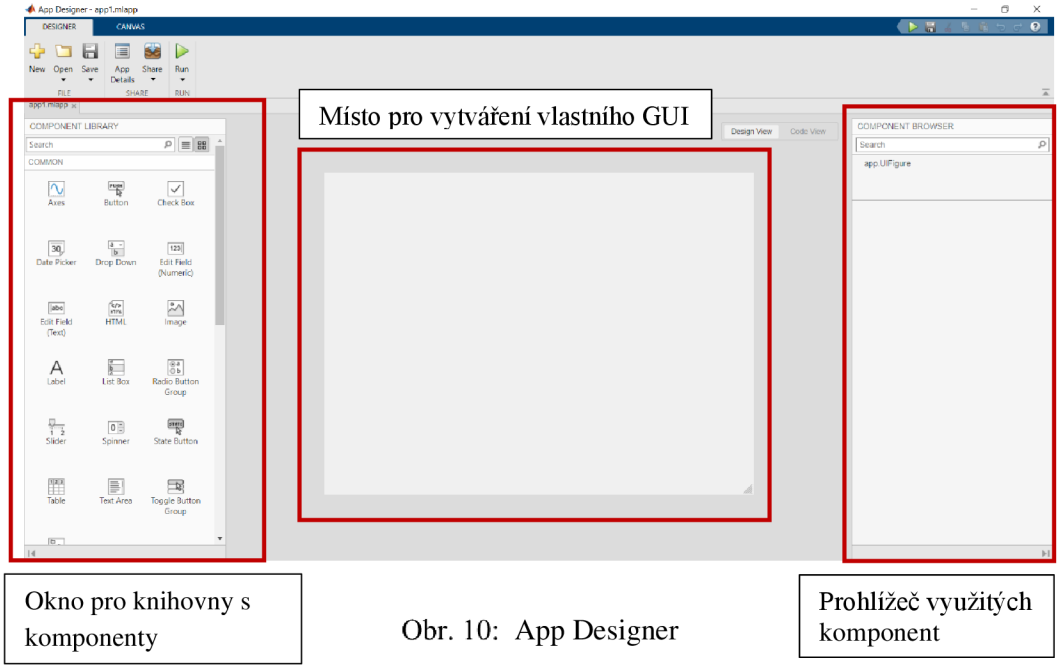


Posledním zde popsaným blokem je *Ultrasonic Sensor*. Ten byl využit v druhém praktickém úkolu k určení aktuální výšky válečku. Slouží k měření vzdálenosti mezi ním a snímaným objektem. Hodnota jeho výstupu je v metrech a její datový typ je double. Je možné jej využít jak k tří-pinovým, tak ke čtyř-pinovým ultrazvukovým senzorům a je možné u něj také nastavit vzorkovací čas.

3.4 App Designer

App Designer je stejně jako Simulink rozšíření Matlabu. Umožňuje vytvářet profesionálně vypadající aplikace i uživatelům, kteří nejsou zkušení softwaroví vývojáři [20]. To je zapříčiněno hlavně jeho jednoduchým provedením.

Spouští se v kartě *HOME* po kliknutí na *New* a poté na *App*. Druhá možnost je napsat *appdesigner* do *Command Window*.



Obr. 10: App Designer

Na obr. 10 je zobrazeno GUI (Graphic User Interface) App Designeru. Je velice intuitivní, v levé části se nachází knihovna s komponenty, díky nimž si uživatel může vytvořit své vlastní GUI. Stačí je myší přetáhnout do prostřední části, zde poté vznikne uživatelské rozhraní, které je však ještě nutné naprogramovat, aby fungovalo, tak jak uživatel zamýšlel. Do části, kam se píše program se lze dostat kliknutím na *Code View* nebo klávesovou zkratkou *Shift* + *F7*. Po zhotovení kódu se aplikace pouští stisknutím *Run* v horní části.

App designer byl použit při tvorbě prvního praktického úkolu a jeho implementace s Arduinem je vysvětlena v následující kapitole.

4 ÚLOHA 1 - MĚŘENÍ TEPLOTY A VLHKOSTI

Cílem tohoto praktického úkolu bylo pomocí App Designeru navrhnout a vytvořit jednoduchou aplikaci, která bude měřit aktuální teplotu a vlhkost v místnosti a bude tyto hodnoty vykreslovat v reálném čase do grafů. Při určité teplotě, kterou si uživatel nastaví podle svých představ, se zapne ventilátor připevněný k servomotoru. Uživatel bude mít možnost regulovat jak rychlost otáčení servomotoru, tak otáčky ventilátoru. Konstrukce bude místnost ochlazovat do doby, než teplota klesne pod předem danou hranici. Zvyšování teploty je zde simulováno pomocí žárovky. Úkol má mít edukativní charakter, aby namotivoval případné zájemce o studium programování mikrokontrolerů Arduino, či pomohl již pokročilejším uživatelům.

4.1 Použité komponenty

Při tvorbě aplikace byly využity tyto hardwarové prvky: Arduino Uno, senzor teploty a vlhkosti DHT22, servomotor Tower Pro MG996R, 4 – pinový ventilátor Arctic F9 PWM, jeden 10 kΩ rezistor, 12 V, 5W žárovka a zdroj 12 V napětí. Arduino Uno bylo zvoleno proto, že jeho výpočetní výkon byl k tomuto úkolu zcela dostačující. Na výběr bylo i Arduino Mega, to je využito v druhé praktické úloze, kde je potřeba větší paměti. Uno je podrobně popsáno v kapitole 2, proto zde již jeho popis není znovu uveden. Popis ostatních součástek a důvod jejich výběru je však níže. Byl využit rozšiřující balíček Matlab Support Package for Arduino Hardware, neboť je kompatibilní s App Designerem.

4.1.1 Senzor teploty a vlhkosti

Existuje celá řada teploměrů, které se dají propojit s deskami Arduino. Při psaní této práce byly v laboratoři dostupné tyto čtyři teplotní senzory, z kterých autor vybíral. A to: DS18B20+, teplotní čidlo DS18B20, teploměry a vlhkoměry DHT11 a DHT22.

První dva jmenované mají totožný rozsah měření teploty od -55 °C do 125 °C, přesnost měření ±0,5 °C i rozlišení, které je 12 bitů. Oba také komunikují pomocí sběrnice One-Wire, která umožnuje paralelní zapojení více senzorů [6]. Hlavní rozdíl spočívá v tom, že čidlo DS18B20 je vodotěsné, tudíž je vhodné do vlhkého či prašného prostředí.

Zbylé dva senzory zvládají měřit jak teplotu, tak vlhkost. Dle jejich označení je zřejmé, že jsou ze stejné řady. Senzor DHT11 je levnější než DHT22, ovšem jeho parametry nejsou nijak výrazné. Rozsah měření teploty je pouze 0–50 °C s přesností ±1 °C a rozsah měření vlhkosti je jen 20-90 % s přesností ±4 %. DHT22 je novější a tomu odpovídají i jeho parametry, které jsou výrazně lepší než u jeho předchůdce. Rozsah měření teploty senzoru DHT22 je od -40 °C do 80 °C s přesností ±0,5 °C a rozsah měření vlhkosti je 0–100 % s přesností ±2 % [6]. Byl vybrán senzor DHT22.

Jeho teplotní rozsah není tak velký jako u předchozích dvou senzorů, ovšem k účelům tohoto praktického úkolu je dostačující, a navíc disponuje funkcí měření vlhkosti.

Vybraný senzor je zobrazen na obr. 11. Jeho zapojení k desce Arduino je následující. První pin zleva slouží k napájení, druhý k odesílání dat, třetí pin zůstává nevyužit a poslední se připojí na zem. Dále je potřeba pro umožnění komunikace s deskou Arduino mezi první dva zmíněné piny zapojit 10 kΩ pull-up rezistor. Názorná ukázka zapojení je k dispozici níže v podkapitole *Schéma zapojení*.

Obr. 11: Senzor teploty a vlhkosti DHT22

4.1.2 Servomotor

Servomotor je typ motoru, u něhož lze, na rozdíl od ostatních motorů, nastavit přesnou polohu natočení jeho osy [6]. V této praktické úloze se rozhodovalo mezi dvěma servomotory: MG996R a MG995. Oba jsou od stejného výrobce, kterým je Tower Pro. Jejich parametry jsou prakticky totožné a lze je dohledat na internetových stránkách výrobce [21]. Hlavním rozdílem je to, že servo MG995 je kontinuální, což znamená, že funguje podobně jako běžný DC (direct current) motor s převodovkou.

Kontinuální serva lze v Matlab Support Package for Arduino Hardware řídit jedinou funkcí, kterou je *writePosition(s, position)*. S její pomocí je možné ovládat směr i rychlost otáčení. Při zadání hodnoty 0 na místo proměnné *position* se bude servomotor otáčet maximální rychlost na jednu stranu, při zvyšování hodnoty se bude otáčet pomaleji, při hodnotě 0,5 se zastaví. Dalším zvyšováním proměnné *position* se bude motor otáčet na druhou stranu, analogicky hodnota 1 zapříčiní rychlost maximální.

Tradiční serva, mezi která se řadí MG996R, se ovládají stejnou funkcí, jako serva kontinuální. Jejich hřídel je však možné otočit pouze od 0° do 180° a platný parametr *position* funkce *writePosition(s, position)* je v rozmezí od 0 do 1. Jestliže tedy bude uživatel potřebovat nastavit specifický úhel, je třeba využít znalosti přímé úměrnosti a proměnou *position* si dopočítat.

Při návrhu řešení byly testovány oba zmíněné servomotory a vybrán byl MG995R. Jeho provozní napětí je výrobcem uváděné v rozsahu od 4,8 do 7,2 V a rychlost otáčení je 0,17s/60° při 4,8 V a 0,13s/60° při napětí 6 V [21]. To z důvodu, že u tohoto servomotoru je možné snadno nastavit jeho počáteční a konečná polohu. U kontinuálního servomotoru je naopak jednoduché nastavit rychlost jeho otáčení, u vybraného modelu to bylo vyřešeno pomocí cyklu *for* popsaného v podkapitole níže.

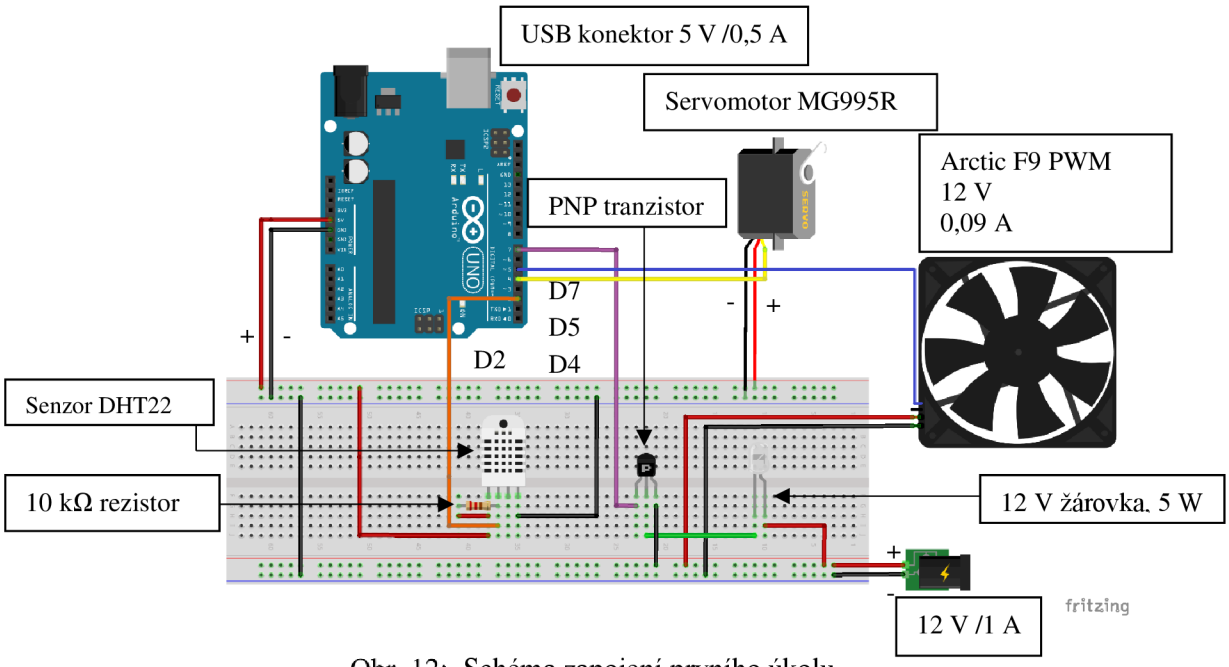
4.1.3 Ventilátor

Ventilátor je DC motor s lopatkovým kolem, který svým otáčením vytváří proud vzduchu, ten v této úloze slouží k ochlazení prostředí. Ventilátory se dělí podle směru průtoku vzduchu na axiální a radiální (centrifugové) [22]. Dále se ventilátory dělí na dvou, tří a čtyř pinové. U dvou pinových jsou jeho vodiče využity pouze na napájení. Většinou se jedná o 12 V stejnosměrných. Tří pinový ventilátor má navíc vodič na měření otáček. Čtyř pinový má ještě vývod na regulaci otáček pomocí PWM signálu, proto byl vybrán k tomuto úkolu. Byl využit 4pinový axiální ventilátor napájený na 12 V s maximální hodnotou proudu 0,09A.

Řízení otáček u dvou a tří pinových ventilátorů je možné pomocí MOSFETu typu PNP, což je unipolární tranzistor a pomocí PWM regulace, kterou umožňuje Arduino. V úkolu byl však využit čtyř pinový ventilátor, který obsahuje vestavěné tranzistory a rychlost otáček jeho lopatek je řízena PWM signálem.

4.2 Schéma zapojení

Na obr.12 je zobrazeno schéma zapojení všech výše popsaných součástek, potřebných pro tento praktický úkol. Jak již bylo řečeno je potřeba propojit napájecí a datový pin senzoru teploty a vlhkosti DHT22 pomocí 10 kΩ pull-up rezistoru. Je nutné připojit PWM vodič ventilátoru na k tomu určený pin Arduina. PWM piny jsou znázorněny vlnovkou. Arduino UNO umožnuje maximální napětí 5 V, proto bylo potřeba využít tranzistoru k ovládání žárovky. Servomotor by měl být z důvodu vyšších nároků na proud napájen samostatně, ovšem tento model není určen k dlouhodobému používání, ale spíše k demonstraci možnosti využití Matlab Support Package for Arduino Hardware a App designeru. Schéma bylo vytvořeno, pomocí programu Fritzing [23].



Obr. 12: Schéma zapojení prvního úkolu

4.3 Popis Aplikace

Výsledná aplikace je zobrazena na obr. 13 níže. Popisky v aplikaci a komentáře kódu jsou napsány v angličtině, a to z jednoho prostého důvodu, může se stát, že student, který bude chtít aplikaci napodobit anebo bude potřebovat využít některou její část, nemusí umět česky. Aplikace je rozdělena do dvou hlavních částí, kterými jsou část pro zadávání vstupu od uživatele a část pro zobrazování výstupu z DHT22 senzoru.

Obr. 13: Výsledná aplikace

První část, skládající se ze sekcí *Setup*, *Temperature setpoint*, *Velocity of the fan*, a *Velocity of the servomotor,* je část řídící. Uživatel si zde pomocí posuvného ukazatele nastaví požadovanou teplotu, při které se má zapnout servomotor s ventilátorem. Teplota je ve °C a pro přehlednost se zobrazí v okénku pod ukazatelem. V pravém horním rohu se nachází vypínač ovládající žárovku. Po stisknutí tlačítka *Start* začne DHT22 senzor

měřit aktuální teplotu a vlhkost. Tyto údaje se budou v reálném čase vykreslovat do dvou grafů, které jsou v dolní části aplikace. Aktuální hodnoty teploty i vlhkosti jsou zobrazeny v okénkách *Current*. Aplikace dále od momentu stisknutí tlačítka *Start* zaznamenává nejvyšší a nejnižší hodnoty obou měřených veličin.

Jestliže teplota prostředí převýší mezní teplotu, kterou si uživatel zvolil, automaticky se zapne servomotor spolu s ventilátorem, který je k němu připevněný. Zeleně se rozsvítí dvě kontrolky v sekci *Setup* značící pohyb těchto dvou komponent a souběžně začne červeně svítit kontrolka v sekci *Temperature* značící převýšení zadané teploty. Nyní má uživatel možnost regulovat rychlost otáčení servomotoru i rychlost otáček ventilátoru pomocí dvou přepínačů umístěných v horní části aplikace.

Při opětovném klesnutí teploty pod mezní hranici se servomotor s ventilátorem vypnou a kontrolkám se vrátí výchozí barva. Zastavit činnost aplikace lze i bez toho, aby teplota klesla pod zadanou hranici, a to stisknutím tlačítka *Stop*. Tlačítkem *Clear* se poté dají vymazat grafy i naměřené hodnoty a aplikaci je možné spustit znovu.

4.4 Popis kódu

Z důvodu rozsáhlosti kódu zde nebude popsán celý, nýbrž jen jeho části. Kompletní zdrojový kód je k dispozici v příloze 1. Kód, který byl automaticky vygenerován Matlabem, zde popsán také nebude a to proto, že vygenerované komentáře, které jsou jeho součástí, jsou dostatečně výstižné.

První část kódu, která zde bude rozebraná je *startupFcn(app)*. Je zobrazena na obr. 14. Tato funkce se vykoná jen jednou, a to při zapnutí aplikace. Jde o obdobu funkce *setup*, která je využívána v Arduino IDE. V případě tohoto úkolu slouží funkce pro inicializaci Arduino desky, serva, DHT22 senzoru a k vypnutí ventilátoru. K vytvoření aplikace byly využity dvě knihovny, které jsou načteny pomocí funkce *app.a = arduino*. Jedná se o knihovnu *Servo*, která je předinstalovaná a knihovnu *Adafruit/DHT22*, kterou je nutné si ji stáhnou skrze tento internetový odkaz [24] a zkopírovat do složky, ke které se uživatel dostane skrze následující cestu ve svém adresáři - *MATLAB\SupportPackages\R2019b\3P.instrset\arduinoide.instrset\libraries*.

Jak je vidět na obr. 14, kód je téměř stejný, jako by byl psán přímo v Matlabu s využitím Matlab Support Package for Arduino Hardware, jen s jediným rozdílem, čímž je to, že před proměnné se píše *app*.

```
function startupFcn(app)
    app.ard = arduino('COM6', 'Uno', 'Libraries', {'Adafruit/DHT22','Servo'});    % Arduino
    app.serv = servo(app.ard,"D4");                                                % Servo
    app.dht = addon(app.ard, 'Adafruit/DHT22', 'D2');                              % DHT22 sensor
    writeDigitalPin(app.ard, 'D5',0);                                              % Fan off
```

Obr. 14: Funkce startupFnc(app)

Další popsaná část kódu je zobrazena na obr. 15. Jedná se o funkci *draw(app)*, která byla vytvořena za účelem měření teploty a vlhkosti a také následného vykreslování těchto hodnot do grafů. Funkce je implementována do výsledného kódu tak, že při stisknutí tlačítka *Start* se spustí cyklus *while*, v němž se tato funkce vykonává do doby, než je stisknuto tlačítko *Stop*.

```
function [temp,hum] = draw(app)
        temp = readTemperature(app.dht);
        app.CurrentEditField_T.Value = temp;
        hum = readHumidity(app.dht);
        app.CurrentEditField_H.Value = hum;
        t = datetime("now") - app.startTime;
        addpoints(app.t_axes,datenum(t),temp)
        addpoints(app.h_axes,datenum(t),hum)
        drawnow
end
```

Obr. 15: Funkce draw(app)

V prvních dvou řádcích se přečte aktuální teplota pomocí funkce *readTemperature()*, uloží se do proměnné *temp*, která je následně vypsána do pole pro aktuální teplotu (*Current*). Následující dva řádky fungují analogicky k těm předchozím s rozdílem, že se pomocí funkce *readHumidity()* změří senzor DHT22 vlhkost prostředí. Jeho hodnota se poté vypíše do příslušného pole.

Zbylé čtyři řádky kódu, slouží k vykreslení naměřených hodnot teploty a vlhkosti do grafů v reálném čase. Funkce *addpoints(an, x, y)* je určená právě k těmto účelům. Parametry *x* a *y* jsou přirozeně souřadnice požadovaného bodu a parametr *an* je zkratka pro *animatedline,* tu je nutné nejprve vytvořit. Obr. 16 ukazuje jak na to. Její parametr je název požadovaného grafu.

```
app.t_axes = animatedline(app.UIAxes_T);
app.h_axes = animatedline(app.UIAxes_H);
```

Obr. 16: Funkce animatedline

Poslední, zde popsaná část kódu, je ta, která umožňuje rotaci servomotoru. Metoda *app.VelocityoftheservoKnob.Value* zjišťuje polohu přepínače rychlost otáčení servomotoru a ukládá ji do proměnné *valueservo*. Podle ní se následně pomocí struktury *if* uloží do proměnné *app.Vs* příslušná hodnota. Tato hodnota poté slouží v cyklu *for* jako proměnná pro krok, který se v každé iteraci cyklu vykoná. Mezní hodnoty cyklu *for* jsou nastaveny na 1/3 a 2/3, což odpovídá 60°, ve kterých se servomotor otáčí. Analogicky k tomuto cyklu funguje i druhý cyklus *for*, díky němuž se servo otáčí na druhou stranu. Kód je zobrazen na obr. 17.

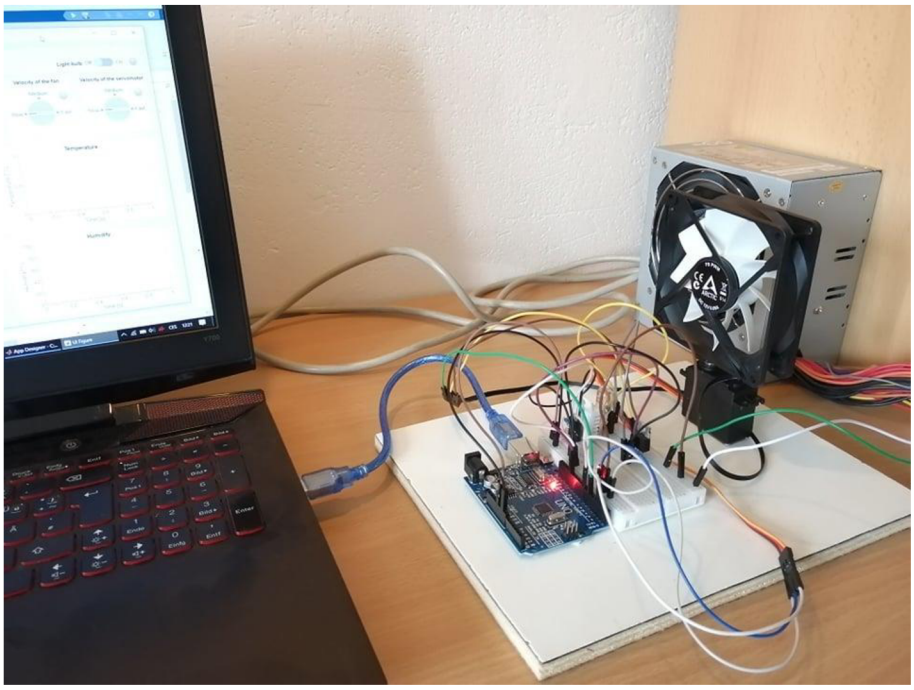
Obdobně funguje i část kódu, která umožňuje regulaci otáček ventilátoru. Ta zde, právě kvůli duplicitě není zobrazena, ale je součástí přílohy 1.

```
% Servo speed
valueservo = app.VelocityoftheservoKnob.Value;

if strcmp (valueservo,'Slow')
    app.vS = 1/180;
elseif strcmp (valueservo,'Medium')
    app.vS = 1/120;
elseif strcmp (valueservo,'Fast')
    app.vS = 1/90;
end

for i = (1/3):app.vS:(2/3)
    writePosition(app.serv,i);
    pause(0.01);
end

[~,~] = draw(app);

for i = (2/3):-app.vS:(1/3)
    writePosition(app.serv,i);
    pause(0.01);
end
```

Obr. 17: Část kódu umožňující pohyb servomotoru

4.5 Zhodnocení úkolu

Za pomocí Matlab Support Package for Arduino Hardware a App Designeru, byla vytvořena demonstrační aplikace, která pomocí DHT22 senzoru měří aktuální teplotu a vlhkost prostředí a následně tyto naměřené hodnoty vykresluje do grafů. Byl využit servomotor a ventilátor k vytvoření jednoduchého větráku, který se zapne při určité teplotě. Výsledný model je zobrazen na obr. 18.

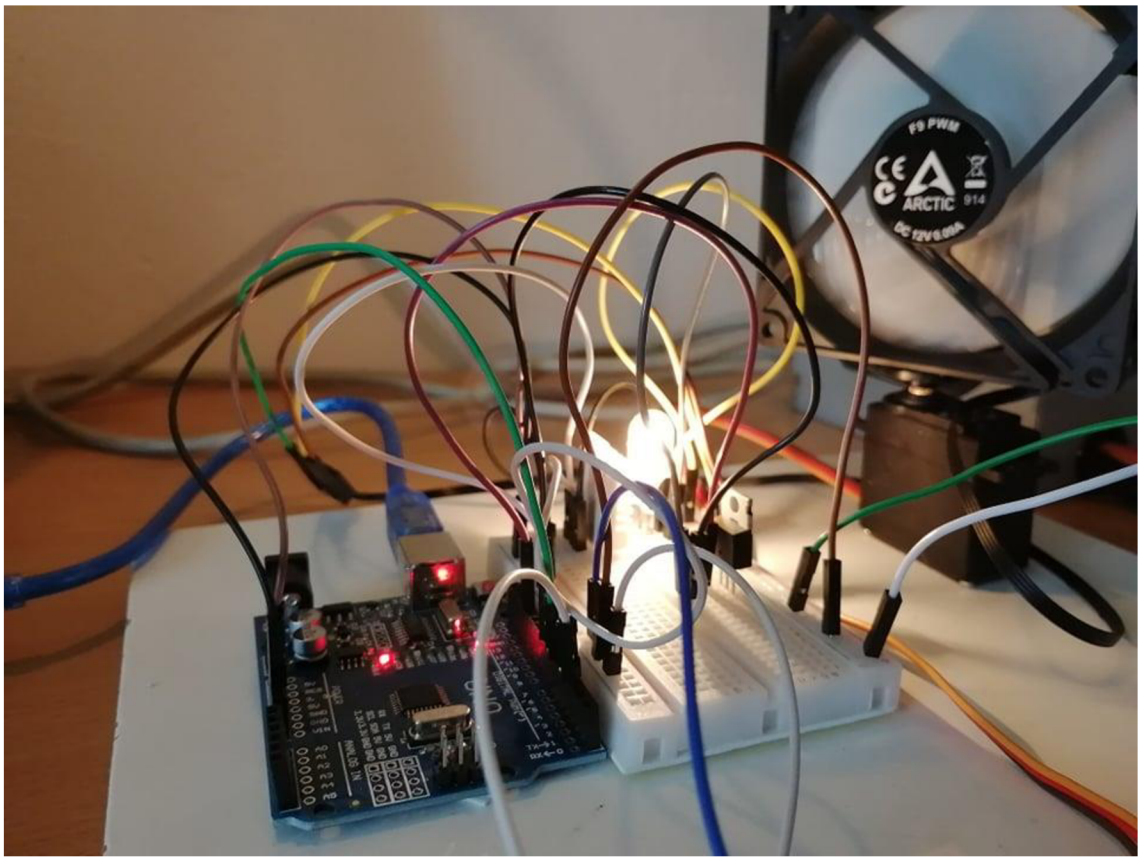


Obr. 18: Výsledný model pro měření teploty a vlhkosti

Jak již bylo zmíněno, při dlouhodobém používání by mělo servo mít vlastní napájení (4,8 – 7,2V) z důvodu vyšších nároků na proud.

Aplikace by se samozřejmě dala rozšířit. Například použitím Arduino desky, která obsahuje slot na SD kartu, na níž by se zapisovaly naměřené hodnoty nebo využitím bluetooth, či wifi modulů, které by byly schopny data posílat dejme tomu na mobilní telefon. To však nebylo cílem tohoto úkolu, jeho záměrem bylo ukázat začátečníků, či mírně pokročilým uživatelům, možnosti programování platformy Arduina pomocí Matlabu, což bylo splněno.

Na obr. 19, je zobrazen detail výsledného modelu v chodu při spuštění aplikace.



Obr. 19: Detail výsledného modelu

5 ÚLOHA 2 - VZDUCHOVÁ LEVITACE

Záměrem tohoto úkolu bylo zkonstruovat model pro vzduchovou levitaci tělesa a pomocí Simulinku vytvořit program, díky němuž bude moci uživatel regulovat polohu zmíněného tělesa.

5.1 Použité komponenty

K tvorbě tohoto úkolu byly využity následující komponenty: Arduino Mega, ventilátor Akasa – DFS922512H, potenciometr, 3 LED diody, 3 220 Ω rezistory, servomotor MG996R, ultrazvukový senzor HC-SR04, externí zdroj 12 V napětí, průhledná trubice a komponenty navrhnuté v programu Inventor a vytisknuté na 3D tiskárně. Ty jsou k dispozici ve formátu *.stl* v příloze číslo 3. Z důvodu využití stejného servomotoru jako v předchozím úkolu zde nebude popsáno. Arduino Mega, stejně jako Arduino UNO v minulé kapitole, zde také nebude znovu popsáno, jeho důkladná charakteristika je v kapitole 2.2.3. Jako těleso určené k levitaci byl využit molitanový váleček. Při testování byly vyzkoušeny i různé polystyrenové kuličky, ty měli však mnohem větší výchylku než váleček, jehož výchylka k požadované hodnotě je minimální.

5.1.1 Ultrazvukový senzor HC-SR04

Autor této práce se při výběru senzoru, který bude zjišťovat aktuální výšku válečku v trubici, rozhodoval mezi infračerveným senzorem vzdálenosti GP2Y0A21Yk0F a ultrazvukovým senzorem HC-SR04, obr. 20. Vybrán byl druhý zmíněný, a to z důvodu, že je přesnější. Jeho odchylka je pouze ± 3 mm. Další důležité vlastnosti tohoto senzoru jsou jeho rozsah měření, který se pohybuje od 2 do 400 cm a jeho zorný úhel, který je menší než 15°. Při měření vzdálenosti, která převyšuje dva metry, se chyba měření mírně zvyšuje [9]. To ovšem není případ tohoto úkolu, zde byla měřena vzdálenost pouze od 2 do 60 cm. HC-SR04 byl tedy ideální volbou pro účely tohoto úkolu.

Obr. 20: Ultrazvukový senzor HC-SR04

Ultrazvukový senzor HC-SR04 funguje tak, že se na 5 mikrosekund pomocí Arduino desky sepne pin Trigger, čímž ultrazvukový vysílač vyšle vysokofrekvenční pulz, který je následně zachycen ultrazvukovým přijímačem. Z pinu Echo se poté získá

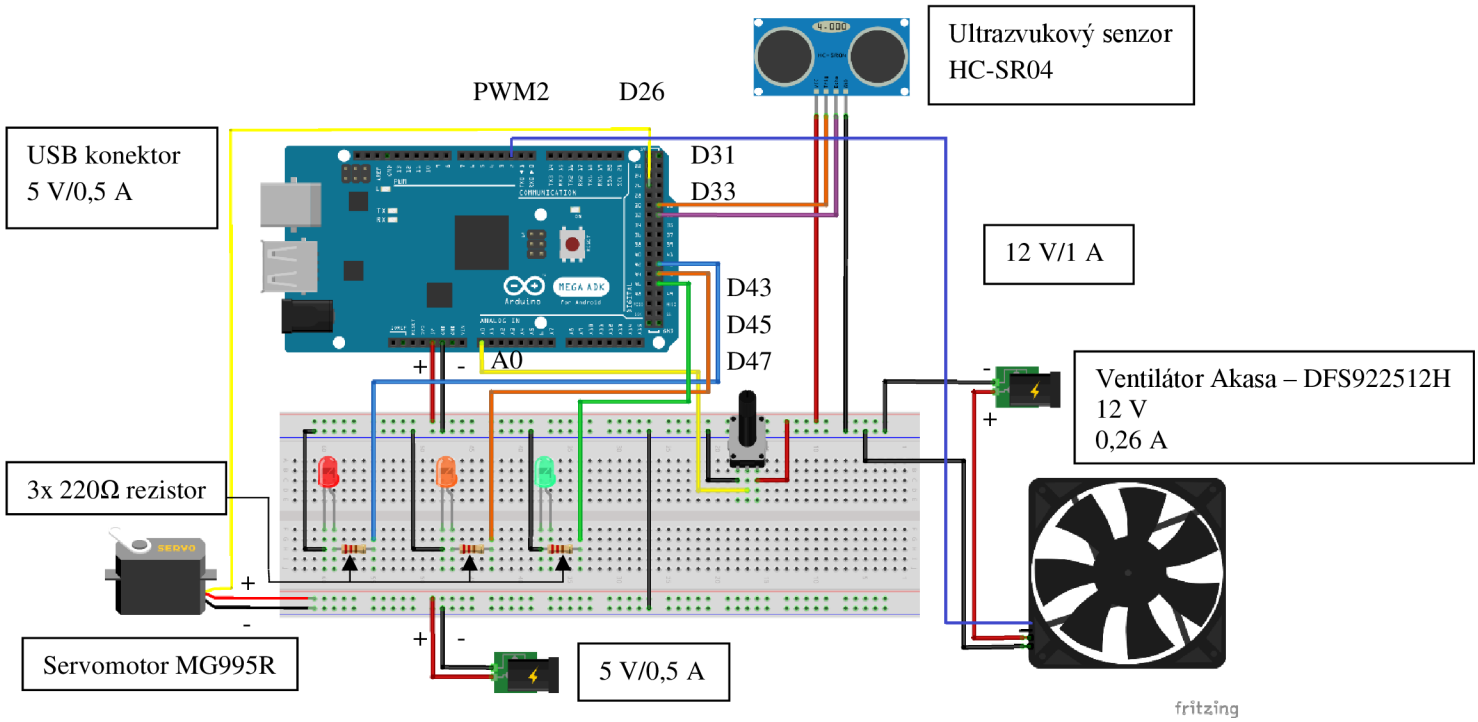
délka impulzu. Ta je potřeba převést pomocí konstanty, která se vypočítá v závislosti na tom, v jaké teplotě či atmosféře je senzor využit. Návod pro vypočítání konstanty je k dispozici v následujícím internetovém odkazu [25]. Ovšem při využití senzoru při pokojové teplotě v Matlabu/Simulinku ani ArduinoIDE není potřeba konstantu znát. Na ultrazvukový senzor HC-SR04 jsou vytvořeny knihovny, které jeho použití výrazně usnadňují.

5.1.2 Ventilátor

Byl vybrán, stejně jako v minulém úkolu, 4 – pinový axiální ventilátor. Ovšem v tomto případě byl při výběru kladen důraz na průtok vzduchu, který je u vybraného modelu udávaný výrobcem 96,52 m^3/h a na rozsah otáček, který je od 600 do 3000 otáček za minutu. Jedná se opět o 12 V ventilátor, nyní je však jeho maximální hodnota proudu 0,26 A. Jeho rozměry jsou 92 x 25 mm, kdy 92 mm je jeho průměr, byl totiž vybrán větrák kruhový. Díky těmto vlastnostem je ventilátor vhodný k této úloze, protože má dostatečný výkon k zvednutí válečku v trubici a udržení jej v požadované poloze.

5.2 Schéma zapojení

Na obr. 21 je zobrazeno schéma zapojení, které bylo využito v této praktické úloze. Součástky jako LED diody, rezistory, potenciometr nebo zdroje napětí nebyly v předchozí kapitole popsány, a to z důvodu, že se očekává, že čtenář této práce je s jejich charakteristikou obeznámen.

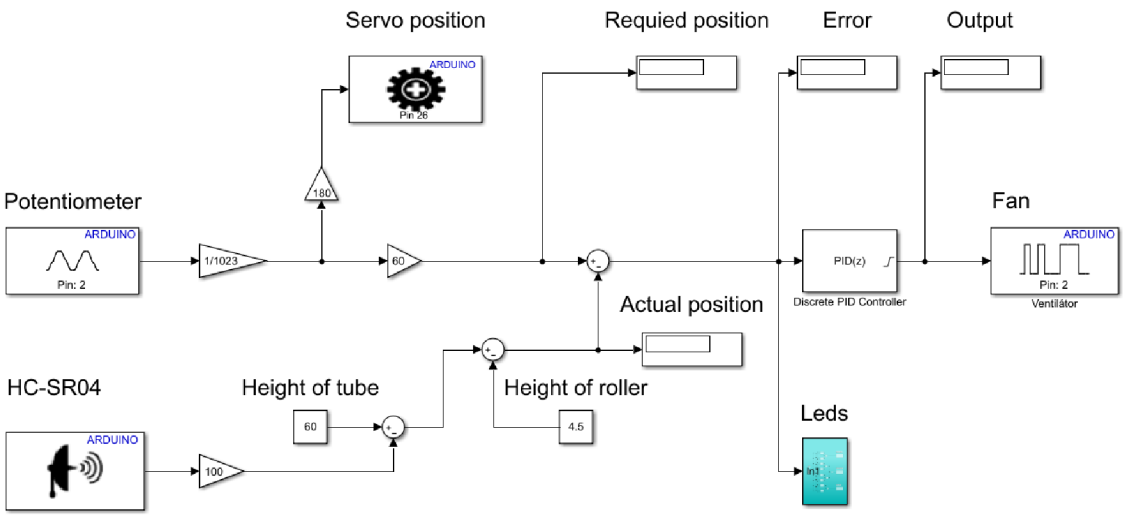


Obr. 21: Schéma zapojení druhého úkolu

Ventilátor je zapojen stejným způsobem, jako u předchozího úkolu. Servomotor má však nyní již vlastní zdroj napájení, a to kvůli větší časové náročnosti této úlohy. Je důležité, aby byly záporné vývody externích zdrojů napětí propojeny s pine GND Arduina. Zapojení ultrazvukového senzoru HC-SR04 je snadné, stačí jeho prostřední dva vývody (Trig a Echo) propojit s libovolnými dvěma digitálními piny. Samozřejmě je také nutné vyřešit napájení senzoru připojením jeho VCC pinu na 5 V Arduina a dokončením obvodu tím, že se propojí piny GND. Při zapojení potenciometru je nutné jeho prostřední vývod propojit s libovolným analogovým pinem. Zbylé dva krajní vývody slouží k napájení a nezáleží na tom, který se použije jako VCC a GND. Zapojení LED diod je zřejmé, rezistory o hodnotě 220 Ω byly využity pro odvod přebytečného napětí, které proudí do diod. Bez nich by mohlo dojít k poškození Arduina. Využití LED diod je vysvětleno v následující podkapitole.

5.3 Popis modelu v Simulinku

Výsledný model vzduchové levitace vytvořený v Simulinku je zobrazen na obr. 22. Část *Leds* je pro přehlednost modelu *zabalena* do subsystému, je však k dohledání v příloze 2. LED diody v tomto úkolu slouží k indikaci velikosti chyby (Erroru), která vzniká rozdílem požadované výšky tělesa a její aktuální hodnotou. Při chybě menší než - 3 a větší než 3 mm se rozsvítí červená LED dioda, v případě, že je chyba v rozmezí od - 3 do - 1 mm nebo od v rozmezí od 1 do 3 mm rozsvítí se dioda žlutá. V případě minimální chyby, tj. ± 1 mm se rozsvítí zelená LED dioda.



Obr. 22: Výsledný model vzduchové levitace v Simulinku

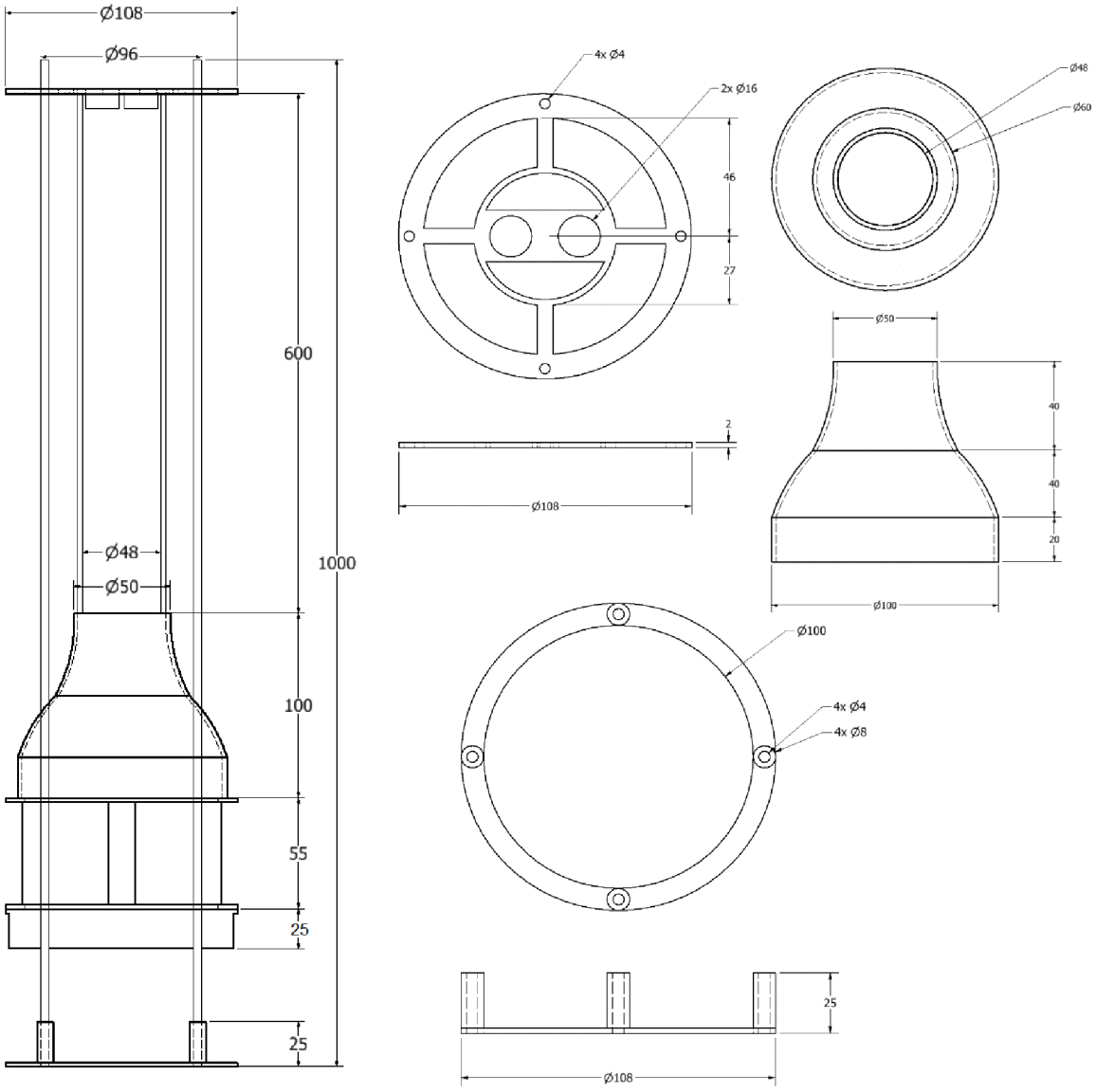
Pomocí potenciometru připojeného k analogovému pinu číslo dva si uživatel nastaví požadovanou výšku válečku. Ta je v rozmezí od 0 do 60 cm. Výstup potenciometru je však od 0 do 1023, takže je nutné využít přímé úměrnosti. Požadovaná výška je pro názornost zobrazena pomocí servomotoru, na němž je přilepena šipka ukazující nastavenou hodnotu na stupnici za ní.

Blok *Analog Input* má nastavený vzorkovací čas na jednu desetinu sekundy, stejně jako blok *Ultrasonic*, který snímá aktuální polohu válečku a pomocí jednoduché matematiky ji převádí na vzdálenost od bodu nula, což je počáteční poloha válečku.

Rozdíl (chyba) mezi požadovanou a aktuální hodnotou je vstup pro blok PID Controller. Ten chybu zpracuje a v závislosti na ní vyšle určitou hodnotu v rozpětí od 0 do 255 do bloku PWM, čímž se řídí otáčky ventilátoru.

5.4 Náčrt konstrukce

Pro přibližnou představu výsledného modelu byl vytvořen jeho náčrt, který je zobrazen na obr. 23. Je zde model celkový a zvlášť jeho jednotlivé části.

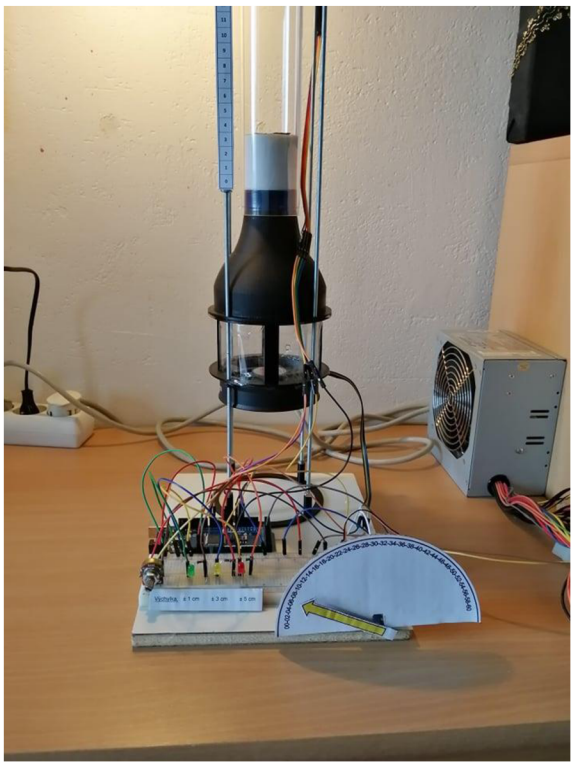


Obr. 23: Náčrt konstrukce a základní rozměrové parametry

5.5 Zhodnocení úkolu

Cílem tohoto úkolu bylo vytvořit projekt pro vzduchovou levitaci tělesa a umožnit uživateli pomocí PID regulátoru regulaci jeho polohy. Byl využit Simulink a jeho rozšíření Simulink Support Package for Arduino Hardware. V něm byl vytvořen model, který byl nahrán do Arduino Mega, to bylo připojeno ke všem zmíněným součástím popsaným výše. Výsledný model je zobrazen na obr. 24 a jeho detail je poté zobrazen na obr. 25.

Obr. 24: Kompletní model



Obr. 25: Detail modelu

Bylo dosaženo uspokojivých výsledků, což je vidět na demonstračním videu k tomuto příkladu. To je k dispozici v příloze 3. Váleček má totiž v každé vyzkoušené požadované poloze po chvíli odchylku ± 1 mm, což rozsvítí, jak bylo řečeno, zelenou LED diodu.

Výsledný program běží samostatně na desce Arduino Mega. To bylo ovšem po celou dobu testování připojeno k PC a tím i napájeno. Jako rozšíření tohoto úkolu by bylo tedy například vhodné využít externího napájení pro samotné Arduino.

6 ZÁVĚR

Matlab/Simulink nabízí zajímavé způsoby jak programovat mikrokontrolery Arduino. Rešerše těchto možností a samotné platformy Arduino byla předmětem teoretické části této bakalářské práce. V praktické části byly vytvořeny dvě úlohy, které mohou sloužit k edukativním účelům na Ústavu automatizace a informatiky.

První praktický příklad využívá Matlabu a jeho rozšíření Matlab Support Package for Arduino Hardware a App Designer. Byla vytvořena aplikace, která díky senzoru DHT22 měří aktuální teplotu a vlhkost. Tyto hodnoty poté vykresluje v reálném čase do grafu. Do aplikace byly přidány funkce, které ukládají maximální a minimální hodnoty obou zmíněných měřených veličin. Dále byla do obvodu přidána žárovka, kterou může uživatel v prostředí programu ovládat a tím simulovat nárůst teploty na senzoru DHT22. Při určité hodnotě přednastavené teploty, kterou si uživatel sám nastaví, se spustí ventilátor, jehož účelem je naopak teplotu snižovat. Rychlost otáčení servomotoru i rychlost otáček ventilátoru může uživatel v prostředí aplikace regulovat.

Druhý praktický úkol využívá naopak Simulink a jeho rozšiřující balíček Simulink Support Package for Arduino Hardware. Byl zkonstruován model pro vzduchovou levitaci molitanového válečku v trubici. Pozici válečku v trubici lze regulovat tak, že uživatel zadá pomocí potenciometru požadovanou polohu válečku, která se zobrazí za využití servomotoru na stupnici. Připojený ultrazvukový senzor HC-SR04 slouží jako zpětná vazba pro aktuální polohu válečku**.** Rozdíl reálné polohy válečku a požadované polohy vstupuje do PID regulátoru, jehož výstupem je hodnota PWM signálu, který ovládá ventilátor umožňující levitaci válečku.

Práce byla pro autora jednoznačně přínosná, protože jej obohatila o znalosti využití Matlabu a Simulinku jako nástroje pro programování Arduina, s čímž měl minimální zkušenosti. Osvojil si programování tohoto mikroprocesoru, s čímž nehodlá skončit, a naopak by si rád rozšířil obzory programování i jiných mikroprocesorů než Arduino.

7 SEZNAM POUŽITÉ LITERATURY

[1] HÁJEK, Jan. *Možnosti implementace řídících algoritmů na mikrokontroléru Arduino s použitím prostředí IDE, Matlab&Simulink a SciLab*. Ostrava: Technická universita Ostrava, Fakulta elektrotechniky a informatiky, 2017. 55 s. Vedoucí bakalářské práce doc. Ing. Štěpán Ožana, Ph.D.

[2] ZIMEK, Tomáš. *Arduino: programování v prostředí Matlab/Simulink.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 77 s. Vedoucí bakalářské práce doc. Ing. Radomil Matoušek, Ph.D.

[3] KLAPÁLEK, Jaroslav. *Vzduchová levitace dvou ping-pongových míčků*. Praha: České vysoké učení technické v Praze, Fakulta elektrotechnická, 2017. 59 s. Vedoucí bakalářské práce Ing. Jiří Zemánek.

[4] *Arduino* [online]. ©2020 [cit. 2020-04-10]. Dostupné z https://www.arduino.cc/.

[5] KUSHNER, David. The Making of Arduino: How five friends engineered a small circuit board that´s taking the DIY world by storm. In: *spectrum.ieee.org* [online]. 2011-10-26 [cit. 2020-04-10]. Dostupné z https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino/.

[6] HUGHES, John M. *Arduino: A Technical Reference.* Sebastopol: O´Reilly Media, Inc, 2016. ISBN 978-14-91921-76-0.

[7] *Wiring* [online]. ©2020 [cit. 2020-04-10]. Dostupné z https://www.wiring.org.co/.

[8] *Processing* [online]. ©2020 [cit. 2020-04-10]. Dostupné z https://www.processing.org/.

[9] VODA, Zbyšek a tým HW Kitchen. *Průvodce světem Arduina.* 2. vydání. Bučovice: Martin Stříž, 2017. ISBN 978-80-87106-93-8.

[10] SVOBODA, Aleš. Vše o napájení Arduina. In *navody.arduino-shop.cz* [online]. 2018-09-03 [cit. 2020-04-10]. Dostupné z: https://navody.arduino-shop.cz/technikuv-blog/napajeni-arduina.html/.

[11] Starting Electronics. Differences Between the Original Arduino Uno, R2 and R3. In: *startingelectronics.org* [online]. 2012-10-15, poslední aktualizace 2013-02-28 [cit. 2020-04-10]. Dostupné z https://startingelectronics.org/articles/arduino/uno-r3-r2-differences/.

[12] SELECKÝ, Matúš. Arduino: uživatelská příručka. Přeložil Martin HERODEK. Brno: Computer press, 2016. ISBN 978-80-251-4840-2.

[13] AQEEL, Adnan. Introduction to Arduino IDE. In: *theengineeringprojects.com* [online]. 2018-10-3 [cit. 2020-04-15]. Dostupné z https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-ide.html/.

[14] Mathworks. Articles. *A Brief History of MATLAB* [online]. Mathworks, ©2018 [cit 2020-04-20] Dostupné z https://www.mathworks.com/company/newsletters/articles/a-brief-history-of-matlab.html

[15] *Mathworks* [online]. ©1994-2020 [cit. 2020-04-20]. Dostupné z https://www.mathworks.com/.

[16] Humusoft. Simulink. *Humusoft* [online]. Humusoft ©1991-2020 [cit. 2020-05-07]. Dostupné z: https://www.humusoft.cz/matlab/simulink/.

[17] MathWorks. MATLAB Support Package for Arduino Hardware. *MathWorks* [online]. MathWorks ©2017 [cit. 2020-05-07]. Dostupné z: https://www.mathworks.com/help/supportpkg/arduinoio/index.html

[18] MathWorks. Build Arduino projects using high-level programming and block diagrams. *MathWorks* [online]. MathWorks ©1994-2020 [cit. 2020-05-07]. Dostupné z: https://www.mathworks.com/discovery/arduino-programming-matlab-simulink.html/.

[19] MathWorks. Simulink Support Package for Arduino Hardware. *MathWorks* [online]. MathWorks ©2017 [cit. 2020-05-10]. Dostupné z: https://www.mathworks.com/help/supportpkg/arduino/model-preparation.html/.

[20] MathWorks. App Designer. *MathWorks* [online]. MathWorks ©2017 [cit. 2020-05-12]. Dostupné z : https://www.mathworks.com/products/matlab/app-designer.html.

[21] *TowerPro* [online]. ©2014 [cit. 2020-05-12]. Dostupné z: https://towerpro.com.tw/.

[22] PELONIS, Sam. Axial Vs. Centrifugal Fans In: *pelonistechnologies.com* [online]. 2015-11-04 [cit. 2020-05-13]. Dostupné z: https://www.pelonistechnologies.com/blog/axial-vs.-centrifugal-fans/.

[23] *Fritzing* [online]. ©2020 [cit. 2020-05-13]. Dostupné z: https://fritzing.org/home/.

[24] MathWorks. DHT22 Add-On Library for Arduino. *MathWorks* [online]. MathWorks ©2019 [cit. 2020-05-13]. Dostupné z : https://www.mathworks.com/matlabcentral/fileexchange/72441-dht22-add-on-library-for-arduino/.

[25] M, Lubomír. Měřič vzdálenosti ultrazvukový. In: *navody.arduino-shop.cz* [online]. 2016-03-11 [cit. 2020-05-15]. Dostupné z https://navody.arduino-shop.cz/navody-k-produktum/meric-vzdalenosti-ultrazvukovy.html/.

8 SEZNAM ZKRATEK

DAC (Digital to Analog Converter) – digitálně analogový převodník

DC (Direct Current) – stejnosměrný elektrický proud

GUI (Graphic User Interface) – grafické uživatelské rozhraní

I^2C (Inter-Integrated Circuit) – interní datová sběrnice

IDE (Integrated development environment) – vývojové prostředí

IDII (Interaction Design Institute Ivrea) – Institut interaktivního designu Ivrea

LAN (Local Area Network) – lokální síť

LCD (Liquid-crystal display) – display z tekutých krystalů

LED (Light-emitting diode) – elektroluminiscenční dioda

PWM (Pulse width modulation) – pulzně šířková modulace

SCD (Synchronous Clock) – hodinový signál

SD (Secure Digital) - paměťová karta

SDA (Synchronous Data) – datový kanál

SPI (Serial Peripheral Interface) – sériové periferní rozhraní

USB (Universal Seriál Bus) – univerzální sériová sběrnice

9 SEZNAM PŘÍLOH

1. Kód k prvnímu praktickému úkolu
2. Model vytvořený v Simulinku
3. Soubor Přílohy.zip obsahující obě demonstrační videa, soubory pro spuštění programů a části modelu vzduchové levitace ve formátu .stl, které byly vytištěny na 3D tiskárně

PŘÍLOHY

1. Zdrojový k ód k prvnímu praktickému úkolu:

```
    properties (Access = private)
        ard;
        serv;
        dht;
        start;
        t_axes;
        h_axes;
        t_max = 0;
        t_min = 40;
        h_max = 0;
        h_min = 100;
        vS;
        startTime;
    end

    % function for messuring
    methods (Access = private)
        function [temp,hum] = draw(app)
                temp = readTemperature(app.dht);
                app.CurrentEditField_T.Value = temp;
                hum = readHumidity(app.dht);
                app.CurrentEditField_H.Value = hum;
                t = datetime("now") - app.startTime;
                addpoints(app.t_axes,datenum(t),temp)
                addpoints(app.h_axes,datenum(t),hum)
                drawnow
        end
    end
```

```
% Code that executes after component creation
function startupFcn(app)
            app.ard = arduino('COM6', 'Uno', 'Libraries', {'Adafruit/DHT22','Servo'});% Arduino
            app.serv = servo(app.ard,"D3");                                          % Servo
            app.dht = addon(app.ard, 'Adafruit/DHT22', 'D8');                        % DHT22
            writeDigitalPin(app.ard, 'D6',0);                                        % Fan off
            writeDigitalPin(app.ard, 'D13',0);                                       % Bulb off

end
```

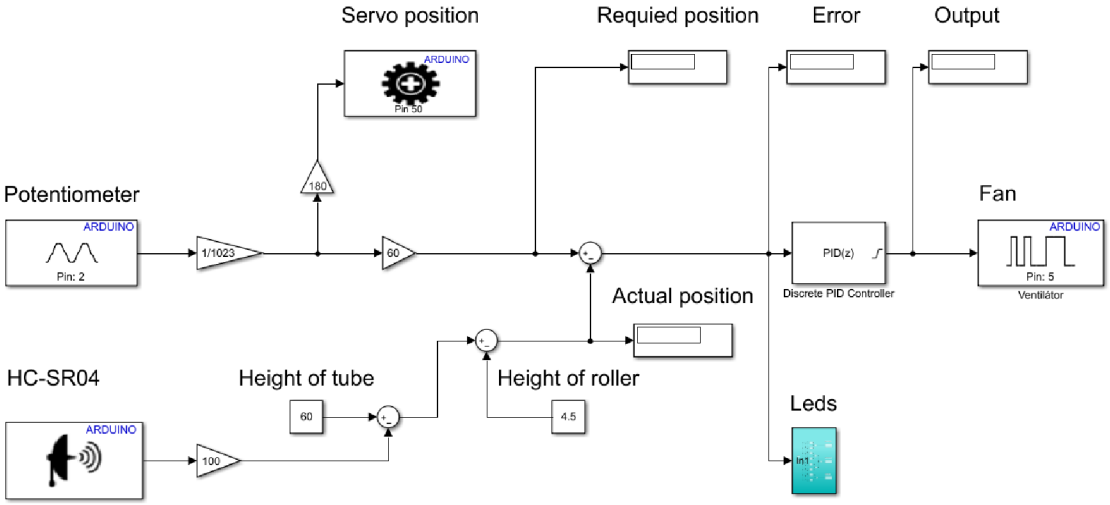
```
% Button pushed function: StartButton
function StartButtonPushed(app, event)
            app.start = 1;
            app.StartButton.BackgroundColor = '0 1 0';
            app.StartButton.FontWeight = "bold";
            app.startTime = datetime("now");
            app.t_axes = animatedline(app.UIAxes_T);
            app.h_axes = animatedline(app.UIAxes_H);
            while true
                [temp,hum] = draw(app);
                %% max temp
                if temp > app.t_max
                    app.t_max = temp;
                    app.MaximumEditField_T.Value = app.t_max;
                end
                %% min temp
                if temp < app.t_min
                    app.t_min = temp;
                    app.MinimumEditField_T.Value = app.t_min;
                end
                %% max hum
                if hum > app.h_max
                    app.h_max = hum;
                    app.MaximumEditField_H.Value = app.h_max;
                end
                %% min hum
                if hum < app.h_min
                    app.h_min = hum;
                    app.MinimumEditField_H.Value = app.h_min;
```

```
            end
            if temp > app.TemperaturesetpointSlider1.Value
                % colors of the lamps
                app.FanLamp.Color ='0 1 0';
                app.ServomotorLamp.Color = '0 1 0';
                app.TempHighLamp.Color ='1 0 0';
                % Fan speed
                valuefan = app.VelocityofFanKnob.Value;

                if strcmp (valuefan,'Slow')
                    writePWMDutyCycle(app.ard,'D6',0.65);
                elseif strcmp (valuefan,'Medium')
                    writePWMDutyCycle(app.ard,'D6',0.75);
                elseif strcmp (valuefan,'Fast')
                    writePWMDutyCycle(app.ard,'D6',1);
                end
                % Servo speed
                valueservo = app.VelocityoftheservoKnob.Value;
                if strcmp (valueservo,'Slow')
                    app.vS = 1/180;
                elseif strcmp (valueservo,'Medium')
                    app.vS = 1/120;
                elseif strcmp (valueservo,'Fast')
                    app.vS = 1/90;
                end

                for i = (1/3):app.vS:(2/3)
                    writePosition(app.serv,i);
                    pause(0.01);
                end

                [~,~] = draw(app);

                for i = (2/3):-app.vS:(1/3)
                    writePosition(app.serv,i);
                    pause(0.01);
                end

            else
                % colors of the lamps
                app.FanLamp.Color ='0.65 0.65 0.65';
                app.ServomotorLamp.Color = '0.65 0.65 0.65';
                app.TempHighLamp.Color ='0.65 0.65 0.65';

                writeDigitalPin(app.ard, 'D6',0);      % FAN OFF
                writePosition(app.serv,0.5);           % servo off
                pause(0.5);
            end

            if app.start == 0
                writePosition(app.serv,0.5);           % servo off
                writeDigitalPin(app.ard, 'D6',0);      % FAN OFF
                app.FanLamp.Color ='0.65 0.65 0.65';
                app.ServomotorLamp.Color = '0.65 0.65 0.65';
                app.TempHighLamp.Color ='0.65 0.65 0.65';
                break
            end
        end
    end

    % Button pushed function: StopButton
    function StopButtonPushed(app, event)
        app.start = 0;
        % colors of lamps
        app.FanLamp.Color ='0.65 0.65 0.65';
        app.ServomotorLamp.Color = '0.65 0.65 0.65';
        app.TempHighLamp.Color ='0.65 0.65 0.65';

        app.StartButton.BackgroundColor = '0.96 0.96 0.96';
        app.StartButton.FontWeight = "normal";
        app.StopButton.BackgroundColor = 'red';
        app.StopButton.FontWeight = "bold";

    end
```

```
% Button pushed function: ClearButton
function ClearButtonPushed(app, event)
    app.UIAxes_T.cla;
    app.UIAxes_H.cla;
    app.MaximumEditField_T.Value = 0;
    app.MinimumEditField_T.Value = 0;
    app.CurrentEditField_T.Value = 0;
    app.MaximumEditField_H.Value = 0;
    app.MinimumEditField_H.Value = 0;
    app.CurrentEditField_H.Value = 0;

    app.t_max = 0;
    app.t_min = 40;
    app.h_max = 0;
    app.h_min = 100;

    app.TemperaturesetpointSlider.Value = 0;
    app.EditField.Value = 0;

    app.StartButton.BackgroundColor = "0.96 0.96 0.96";
    app.StartButton.FontWeight = "normal";
    app.StopButton.BackgroundColor = '0.96 0.96 0.96';
    app.StopButton.FontWeight = "normal";

    app.TempHighLamp.Color ='0.65 0.65 0.65';
end

% TemperaturesetpointSlider
function TemperaturesetpointSliderValueChanged(app, event)
    value = app.TemperaturesetpointSlider.Value;
    app.EditField.Value = value;
end

% Value changed function: LightbulbSwitch
function LightbulbSwitchValueChanged(app, event)
    value = app.LightbulbSwitch.Value;
    if strcmp (value,'Off')
        writeDigitalPin(app.ard, 'D13',0);
        app.bulb_Lamp.Color = '0.65 0.65 0.65';
    else
        writeDigitalPin(app.ard, 'D13',1);
        app.bulb_Lamp.Color = '1 1 0';
    end
end
```

2. Model vytvořený v Simulinku:

- Celkový:



- Subsystém „Leds“