ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Příprava laboratorních úloh pro bakalářský předmět Interaktivní aplikace s Arduinem
Student:	Gabriela Hánová
Vedoucí:	Ing. Ivo Háleček
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Připravte a realizujte soubor úloh pro plánovaný bakalářský volitelný předmět Interaktivní aplikace s Arduinem. Použijte mikrokontrolér Arduino Esplora. Součástí řešení nechť je příprava popisů a návodů pro studenty ve formě použitelné pro výukový systém EDUX.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 10. prosince 2015

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra číslicového návrhu

Bakalářská práce

Příprava laboratorních úloh pro bakalářský předmět Interaktivní aplikace s Arduinem

Gabriela Hánová

Vedoucí práce: Ing. Ivo Háleček

16. května 2017

Poděkování

Děkuji svému vedoucímu práce Ing. Ivovi Hálečkovi, za pečlivé a trpělivé vedení práce a časté konzultace. Dále také děkuji Dr.-Ing. Martinu Novotnému za poskytnutí cenných připomínek při zpracování této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 16. května 2017 .....

Odkaz na tuto práci

Hánová, Gabriela. *Příprava laboratorních úloh pro bakalářský předmět Interaktivní aplikace s Arduinem.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Tato práce se zabývá přípravou bakalářského předmětu Interaktivní aplikace s Arduinem (BI-ARD), zejména návrhem a implementací programovacích úloh pro jednotlivá cvičení na přípravcích Arduino Esplora a Arduino Uno. Text obsahuje studijní podklady sloužící primárně studentům prvního ročníku bakalářského studia jako úvod do vestavných systémů a ke zvýšení zájmu o obor Počítačové inženýrství Fakulty informačních technologií ČVUT.

Práce si klade za cíl zhodnotit, které studijní okruhy jsou vhodné pro výuku základů práce s mikrokontroléry a následně k těmto tématům vytvořit praktické příklady vhodných laboratorních úloh s využitím vývojové desky Arduino.

Klíčová slova platforma Arduino, Arduino Esplora, Arduino Uno, vestavné systémy, edukativní materiály, programování mikrokontrolérů, open-source hardware

Abstract

This bachelor thesis describes development of the undergraduate course Interactive Applications with Arduino (BI-ARD). It deals with the design and implementation of course assignments using microcontroller boards Arduino Esplora and Arduino Uno. The thesis contains course materials (lecture notes and course assignments) aimed at the first-year students that can be used as an introduction to Embedded Systems Programming. These course materials should increase an interest in the Computer Engineering Program at Faculty of Information Technology of the Czech Technical University in Prague.

The goal of the thesis is to evaluate which learning objectives are useful when teaching the basics of microcontrollers. The output of the thesis is a set of course assignments using microcontroller board Arduino.

Keywords Arduino platform, Arduino Esplora, Arduino Uno, Embedded Systems Programming, course materials, programming microcontrollers, open-source hardware

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Na trhu práce můžeme obecně pozorovat malý zájem počítačových inženýrů o návrh a programování vestavných systémů, přestože je jich ve firmách velký nedostatek. Tento trend se promítá i do nižšího počtu uchazečů o obor Počítačové inženýrství Fakulty informačních technologií ČVUT.

Jedním z návrhů na zlepšení této situace bylo vytvoření nového bakalářského předmětu Interaktivní aplikace s Arduinem, určeného studentům již od prvního ročníku studia. Cílem je nenásilnou formou nalákat studenty ke studiu hardwaru a vestavných systémů a tím zvýšit atraktivitu oboru Počítačové inženýrství.

Pro výuku bylo zvoleno Arduino, otevřená platforma pro vývoj aplikací na mikrokontrolérech ATmega. Jeho výhodou je velké množství dostupných knihoven, díky kterým studenti nemusí nejprve studovat datasheety. Navíc k vývoji aplikací pro Arduino stačí pouze znalost jazyku C/C++, kterou si studenti osvojili již v prvním semestru. Pro práci na cvičeních byl vybrán přípravek Arduino Esplora (pouze jedno cvičení zahrnuje práci i s přípravkem Arduino Uno). Arduino Esplora vyniká velkým množstvím již zabudovaných periferií a tak minimalizuje šanci na poničení desky studenty.

Práce si klade za cíl zhodnotit, které studijní okruhy jsou vhodné pro výuku základů práce s mikrokontroléry a následně k těmto tématům vytvořit praktické příklady vhodných laboratorních úloh s využitím vývojové desky Arduino.

První kapitola se věnuje technické specifikaci platformy Arduino a přípravků, které budou používány k výuce. Diskutuje vhodnost těchto přípravků, jakožto didaktických pomůcek. Dále kapitola hodnotí, která témata jsou vhodná k výuce a poskytuje úvod do jejich problematiky. Druhá kapitola se zabývá již konkrétními laboratorními úlohami. Jsou zde uvedena zadání v podobě, v jaké jsou na výukovém systému EDUX. U každého zadání je shrnuto, jaké studijní okruhy si studenti procvičili a co se díky laboratorní úloze naučili.

Výsledkem této práce jsou podklady pro celkem šest základních a tři bonusová laboratorní cvičení. Ty zahrnují jak teoretický úvod do konkrétní pro-

blematiky, tak samotné programovací úlohy. Tyto materiály již byly během jednoho semestru při výuce úspěšně použity.

Závěr práce se zaměřuje na zhodnocení vytvořených programovacích úloh a návodů po jednom semestru běhu předmětu.

Kapitola 1

Analýza a návrh řešení

Tato kapitola se věnuje technické specifikaci platformy Arduino a vývojových desek Arduino Esplora a Arduino Uno. Diskutuje vlastnosti přípravků, shrnuje jejich klady a zápory, jakožto didaktických pomůcek pro výuku základů vestavných systémů.

Dále kapitola analyzuje, která témata jsou vhodná k procvičení na laboratorních cvičeních a poskytuje úvod do problematiky těchto vybraných studijních okruhů.

1.1 Arduino

Arduino je otevřená platforma pro vývoj na mikrokontrolérech ATmega od firmy Atmel. V době psaní práce bylo podle oficiální dokumentace Arduina vytvořeno 29 typů vývojových desek [7]. Jednotlivé typy se mezi sebou liší velikostí, funkcemi nebo výkonem. Mezi nejznámější patří například Uno, Mega, Leonardo, Micro nebo Esplora.

Arduino je vyvíjeno pod svobodnou licencí – dokumentace hardwaru pod licencí Creative Commons Attribution ShareAlike 3.0 a softwaru pod licencí GPL (zdrojové kódy), případně LGPL (knihovny) [8]. Díky této otevřenosti je možné zakoupit mnoho různých klonů, které jsou levnější než oficiální Arduino desky. Mezi tyto známější mutace řadíme například Freeduino nebo Seeduino. Některé klony se zase od prototypu odchylují jinými vlastnostmi, například klon Safeduino je odolnější vůči poničení. Dále se na trhu můžeme setkat s pojmem Genuino, což je pouze název pro oficiální Arduino přípravky vyvážené mimo území USA.

Pro Arduino přípravky je typické rozšíření funkcionality pomocí tzv. shieldů, desek které stačí pouze nasadit na piny Arduina, případně připojit pomocí nepájivého pole.

Součástí platformy je vývojové prostředí Arduino napsané v jazyce Java. Prostředí umožňuje vytvoření tzv. sketche, programu pro desku Arduino a jeho následný překlad a nahrání do přípravku. Aplikace se vyvíjí v jazyce

C/C++ s využitím knihovny Wiring (často se lze v literatuře setkat s nepřesným pojmem programování v jazyce Wiring). Prostředí je multiplatformní, obsahuje množství knihoven a navíc také sériový monitor. V současné chvíli je k dispozici verze IDE Arduino 1.8.1. Verze prostředí použitého pro vytvoření laboratorních úloh a jejich referenčních řešení je 1.7.8.

Na laboratorních cvičeních budou studenti pracovat hlavně s přípravkem Arduino Esplora (v jedné úloze i s přípravkem Arduino Uno).

1.1.1 Arduino Esplora

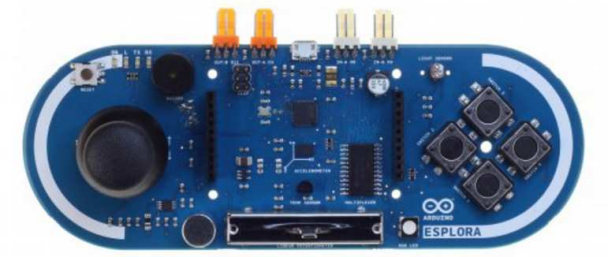
Na vývojové desce Arduino Esplora nalezneme mikroprocesor ATmega32u4 z rodiny AVR. Tento osmibitový mikroprocesor obsahuje 32 kB paměti Flash (z toho 4 kB zabírá bootloader), 2,5 kB paměti SRAM a pracuje na hodinovém kmitočtu 16 MHz [3]. Dále lze využít i 1 kB paměti EEPROM a celkem 5 čítačů/časovačů. Ke připojení k PC slouží USB port 2.0. Čip podporuje i rozhraní SPI, ke kterému můžeme přistupovat pomocí volně dostupné SPI knihovny.

Arduino Esplora se od ostatních Arduino přípravků liší hlavně velkým množstvím již zabudovaných periferií. Kromě 4 notifikačních LED diod obsahuje dále:

- analogový joystick s centrálním tlačítkem
- 4 tlačítka
- posuvný potenciometr
- mikrofon
- světelný senzor
- teplotní senzor
- piezoměnič
- RGB LED diodu
- tříosý akcelerometr
- konektor TFT LCD displeje
- 2 vstupy a 2 výstupy pro sadu modulů TinkerKit

Platforma Arduino obsahuje knihovnu Esplora, která umožňuje snadné ovládání vestavěných periferií na přípravku. Knihovna je součástí vývojového prostředí Arduino. Zahrnuje například funkce pro ovládání jasu RGB LED, čtení teploty z teploměru nebo čtení pozice posuvného potenciometru. V oficiální dokumentaci (viz [9]) nalezneme podrobný popis jednotlivých funkcí spolu s ukázkovými příklady použití.

V době psaní práce je cena oficiální vývojové desky Arduino Esplora cca 1300 Kč bez DPH.



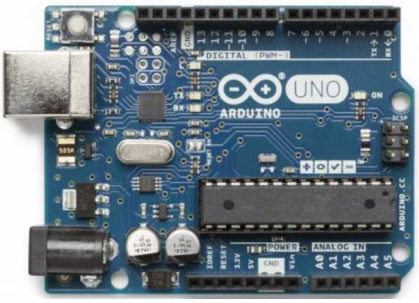
Obrázek 1.1: Arduino Esplora [1]

1.1.2 Arduino Uno

V současné době přípravek Uno řadíme mezi nejběžněji používaný typ Arduina. Vývojová deska Arduino Uno obsahuje mikroprocesor ATmega328P z rodiny AVR. Osmibitový mikroprocesor poskytuje 32 kB paměti Flash, 2 kB paměti SRAM, 1 kB paměti EEPROM a pracuje na hodinovém kmitočtu 16 MHz [10]. Nalezneme zde 14 digitálních vstupně-výstupních pinů a 6 analogových vstupních pinů. Na digitální piny 2 a 3 je možné nastavit externí přerušení.

Na rozdíl od přípravku Arduino Esplora, Arduino Uno nenabízí kromě notifikační LED, žádné zabudované periferie. Funkcionalitu lze ale snadno rozšířit pomocí shieldů.

V době psaní práce je cena oficiální vývojové desky Arduino Uno cca 600 Kč bez DPH.



Obrázek 1.2: Arduino Uno [2]

1.1.3 Shrnutí

Použití platformy Arduino pro výuku základů práce s hardwarem je vhodné z několika důvodů. Jednou z výhod je velké množství knihoven volně k dispo-

zici, díky kterým nemusí studenti studovat složitější datasheety. Pro práci na cvičeních by měly být zcela dostačující knihovny, které již obsahuje Arduino IDE, například knihovna pro nevolatilní paměť EEPROM nebo SD kartu. Informace o jednotlivých knihovních funkcích spolu s ukázkovými příklady lze nalézt na oficiálních stránkách Arduina.

Dále k vývoji aplikací pro Arduino stačí pouze znalost jazyku C/C++, kterou si studenti osvojili již v prvním semestru.

Díky vestavěným periferiím přípravku Arduino Esplora je také minimální šance na poničení a opotřebování vývojové desky studenty (na rozdíl například od Arduino Uno).

Ideální je i nízká cena (Arduino Esplora stojí celkem cca 1700 Kč včetně TFT LCD displeje) a jednoduché programování přípravku přes USB port.

Vývojové prostředí Arduino je multiplatformní, podporuje tedy programování přípravku v různých operačních systémech (především v MS Windows, Linuxu a Mac OS X). Bohužel ale obsahuje jen velmi málo klasických nástrojů běžných u vývojových prostředí. Chybí například možnost formátování a automatického doplňování kódu. Prostředí postrádá i klasický nástroj pro ladění programů, což ale pro účely předmětu nevadí, jelikož pro jednoduché ladění mohou studenti využít vestavěný sériový monitor.

Mezi nevýhody platformy patří omezení velikosti paměti pro Arduino aplikace na 32 kB (z toho 4 kB zabírá bootloader). To by mohl být pro některé studenty problém při vypracování rozsáhlejší semestrální práce.

Dále u laboratorních úloh využívajících EEPROM může příliš častým přístupem do paměti dojít k jejímu rychlému poškození (životnost EEPROM nepřesahuje 100 000 cyklů zápisů/mazání).

V případě nákupu čínských klonů Arduina je potřeba vyhledat a nainstalovat speciální ovladače, jelikož klony nepoužívají stejnou čipovou sadu pro převodník USB jako originální Arduino přípravky (jejich ovladače jsou součástí Arduino IDE).

Díky široké uživatelské komunitě Arduina lze na internetu nalézt i velké množství neoficiálních knihoven (například pro použití časovačů). Při jejich využití je ovšem třeba obezřetnosti, jelikož mnohé obsahují chyby.

1.2 Tématické okruhy cvičení

Tato kapitola hodnotí, která témata jsou vhodná pro výuku studentů, kteří nemají žádné předchozí zkušenosti s programováním přípravků platformy Arduino a mikrokontrolérů obecně. Obsahuje úvod do problematiky jednotlivých studijních okruhů a dále vysvětluje, proč je vhodné zvolená témata vyučovat.

Při výběru vhodných témat k výuce, jsme se nejprve zaměřili na přípravek, který bude na cvičeních využíván nejvíce, Arduino Esplora. Tento přípravek obsahuje mnoho zabudovaných periferií, jejichž použití samo o sobě souvisí hned s několika studijními okruhy.

Například na blikání RGB LED se snadno vysvětlí problematika rozdílu analogového a digitálního signálu. V úloze, která vyžaduje stisknutí několik tlačítek najednou, je zase výhodné vysvětlit a použít konečný automat. Dále SD karta připojená k displeji zase souvisí s problematikou perzistivní paměti. K vysvětlení této pamětí se dá využít i paměť EEPROM mikroprocesoru Arduino Esplora.

Protože studenti budou k ladění svých programů využívat sériový monitor vývojového prostředí Arduino, je vhodné představit i princip sériové komunikace a jeho fyzickou implementaci rozhraní UART. Při práci s displejem, který ke komunikaci používá rozhraní SPI, lze zmínit i tuto implementaci principu sériové komunikace.

Dále jsme se zaměřili na druhý přípravek, který můžeme k výuce využít, Arduino Uno. Laboratorní úloha využívající tento přípravek bude muset obsahovat zapojení vnější periferie (nebo shieldu), jelikož na rozdíl od Arduino Esplora sám o sobě žádné vestavěné periferie neobsahuje. Zde lze představit problematiku přerušení, jelikož je vhodné ovládat připojenou periferii pomocí vnějšího přerušení.

1.2.1 Hlavní smyčka programu

Každý program určený pro přípravky platformy Arduino musí obsahovat dvě základní funkce, `setup()` a `loop()`.

Funkce `setup()` se volá pouze jednou a to po spuštění programu (případně resetování přípravku). Slouží k inicializaci proměnných a knihoven. Pokud pracujeme s přípravkem Arduino Uno, tak v této funkci ještě nastavujeme používané piny (jako vstupní nebo výstupní). U přípravku Arduino Esplora za nás piny (které využívají vestavěné periferie) nastavuje knihovna Esplora.

Oproti tomu `loop()` neboli smyčka, je funkce, jejíž tělo se opakovaně vykonává, dokud není program podmínečně ukončen nebo odpojen od napájení. Ve funkci `loop()` se nachází hlavní výkonný kód programu.

1.2.1.1 Analýza

Úvodní téma cvičení by se mělo věnovat seznámení s vývojovým prostředím Arduino a základům programování přípravků platformy Arduino. Studentům je tedy potřeba představit dvě základní funkce nezbytné pro kompilaci a nahrání programu do přípravku, `setup()` a `loop()`.

Na toto téma navazuje první laboratorní úloha Seznámení s Arduinem (viz Úloha 2.1).

1.2.2 Sériová komunikace

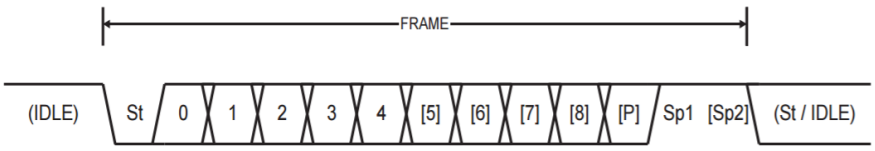
Sériová komunikace se (na rozdíl od komunikace paralelní) vyznačuje sekvenčním přenosem jediného bitu za jednotku času. Jednou z fyzických implementací principu sériové komunikace nazýváme rozhraní USART (Universal

Synchronous and Asynchronous Receiver and Transmitter). V následujících odstavcích se zabýváme pouze používanějším asynchronním módem USARTu mikroprocesoru ATmega32u4, který obsahuje přípravek Arduino Esplora.

Při přenosu dat odešle vysílající rozhraní UART sériově data z pinu Tx a druhé rozhraní je na pinu Rx přijme.

Data přenášená pomocí UART jsou organizována do tzv. paketů. Vysílající UART přidá na začátek datového paketu start bit (logickou nulu definující začátek paketu), 5 až 9 bitů dat (závisí na nastavení UARTu) a na konec jeden až dva stop bity (logické jedničky definující konec paketu). Mezi data a stop bit je možné vložit i paritní bit pro kontrolu integrity dat, viz obr. 1.3.

V klidovém stavu (tj. nevysíláme ani nepřijímáme data) je úroveň signálu logická jedna.



St	Start bit (logická 0)
(n)	Data (0 až 8)
P	Paritní bit
Sp	Stop bit (logická 1)
IDLE	Žádný přenos (logická 1)
FRAME	Datový paket

Obrázek 1.3: Diagram přenosu dat pomocí UART na ATmega32u4 [3]

Vysílající UART přijme data ze sběrnice paralelně a následně je pomocí posuvného registru odesílá sekvenčně. Přijímající UART odstraní metadata (start bit, stop bit(y), volitelný paritní bit) a sekvenčně přijatá data pomocí posuvného registru složí opět do bytů. Data jsou posílána nejméně významným bitem (tzv. LSB - least significant bit) napřed.

Po detekování start bitu začne přijímající UART číst přicházející data s určitou frekvencí, kterou specifikuje tzv. baud rate. Baud rate udává počet změn stavu signálu za sekundu. Měří se v bitech za sekundu (bps). Je nezbytné, aby obě rozhraní UART pracovala na stejné baud rate (rychlost se může lišit maximálně o 10 %). Jelikož se jedná o asynchronní přenos dat, obsahuje vysílající i přijímající UART generátor hodinového signálu (jehož součástí je i generátor baud rate).

Přijímající UART dokáže detekovat několik druhů chyb v přenosu (např. chybu v paritě nebo chybějící start/stop bit).

UART ATmega32u4 dále podporuje tzv. full duplex mód. Full duplex mód umožnuje simultánně odesílat i přijímat data, jelikož oba procesy pracují s rozdílnými registry (na rozdíl od half duplex módu, kde lze v jednom okamžiku pouze vysílat nebo přijímat).

Výhodou použití rozhraní UART je nižší počet pinů potřebných pro přenos, kontrola chyb přenosu pomocí paritního bitu, generování interního hodinového signálu. Jedná se o dobře dokumentovanou a široce používanou implementaci sériové komunikace. Mezi nevýhody řadíme hlavně nutnost předem dohodnout rychlost přenosu, dále velikost dat omezenou maximálně na 9 bitů, celkově pomalý přenos a nepodporování multiple slave nebo multiple master systémů (na rozdíl od rozhraní SPI). Baud rate vysílajícího a přijímajícího UARTu se navíc nesmí lišit o více než 10 %.

Arduino Esplora i Arduino Uno obsahují právě jedno rozhraní UART. Pro přenos používají piny 0 (Rx) a 1 (Tx). Sériová komunikace využívá tzv. TTL (tranzistorově-tranzistorové logiky), tj. logickou jedničku zde reprezentuje napětí 5 V a logickou nulu napětí 0 V.

Součástí platformy Arduino je i knihovna Serial, která umožňuje snadné ovládání rozhraní UART.

Mezi základní funkce této knihovny patří například `Serial.begin()`, která zahájí sériovou komunikaci a inicializaci baud rate přenosu (pro komunikaci s PC se používají standardní hodnoty od 300 do 115200 bps).

```
void Serial.begin(long speed);
```

speed rychlost v bitech za sekundu

Návratová hodnota funkce nic nevrací

Další knihovní funkce je `Serial.write()`, pomocí které lze odeslat data na sériový port (v bytech).

```
uint8_t Serial.write(uint8_t data);
```

data data k odeslání

Návratová hodnota počet zapsaných bytů

Funkce `Serial.read()` přečte příchozí data ze sériového portu.

```
uint8_t Serial.read (void);
```

funkce nemá parametry

Návratová hodnota první přijatý byte

Vývojové prostředí Arduino navíc poskytuje i sériový monitor, který lze využít k ladění laboratorních úloh. Dále obsahuje také knihovnu SoftwareSerial, která umožňuje sériovou komunikaci i na jiných pinech než hardwarem určenými piny 0 a 1. Díky této knihovně lze komunikovat se dvěma přípravky s rozhraním UART nebo komunikovat jen s jedním přípravkem, ale zároveň ponechat jeden sériový port volný pro ladící účely.

Stejné funkce, které byly zmíněny u knihovny Serial nalezneme i u SoftwareSerial, liší se pouze nutností vytvořit před začátkem přenosu instanci

objektu SoftwareSerial, která definuje příslušný Rx a Tx pin. Nevýhodou této knihovny je, že pokud je využíváno více softwarových sériových portů najednou, pouze jeden může přijímat data.

1.2.2.1 Analýza

Přípravky Arduino Esplora a Arduino Uno používají pro přenos dat standardně sériovou komunikaci. Studenti budou od počátku semestru využívat této komunikace mezi přípravky a PC. Další využití je možné například v rámci komplexnější semestrální práce mezi přípravky a zvolenou periferií. Sériový přenos dat studenti využijí i v rámci ladění svých laboratorních úloh při použití sériového monitoru, který je součástí Arduino IDE.

Na toto téma navazuje druhá laboratorní úloha Komunikace s PC a ladění pomocí sériové linky (viz Úloha 2.2).

1.2.3 Konečný automat

Dle [11] definujeme konečný automat M jako sekvenční logický obvod reprezentovaný šesticí $M = (X, Y, Q, Q_0, \delta, \lambda)$, kde

- X je množina možných kombinací hodnot vstupních proměnných M
- Y je množina možných kombinací výstupních hodnot M
- Q je množina možných kombinací hodnot vnitřních proměnných M (množina stavů)
- Q_0 je počáteční stav
- δ je stavově přechodová funkce: $\delta : X \times Q \to Q$
- λ je výstupní funkce:
  - $\lambda : X \times Q \to Y$ (typ Mealy)
  - $\lambda : Q \to Q$ (typ Moore)

Jedná se tedy o výpočetní model jednoduchého počítače, který se skládá z několika stavů, mezi kterými se přechází na základě vstupních symbolů. Konečný automat (Finite State Machine) popisujeme tzv. grafem přechodů. Graf přechodů obsahuje uzly (vnitřní stavy automatu) a hrany (přechody mezi uzly). Hrany jsou ohodnoceny vstupním symbolem a v případě automatu typu Mealy i výstupním symbolem. Automat typu Moore má výstup přiřazen k uzlu. Rozdíl mezi automatem typu Moore a Mealy je tedy v tom, že u automatu typu Moore pozorujeme reakci na vstup až při přechodu do dalšího stavu, zatímco u automatu typu Mealy je reakce viditelná ihned.

Platforma Arduino poskytuje knihovnu Finite State Machine (ke stažení zde [12]), která usnadňuje implementaci konečného automatu.

1.2.3.1 Analýza

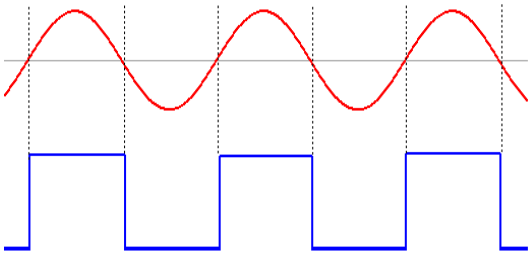
Při řešení úloh, ve kterých je nutné reagovat na více vyvolaných událostí, je vhodné využít k implementaci konečný automat. Deterministický konečný automat jasně formuluje jeden výstup pro každý zadaný vstup. Díky tomu lze přesně definovat průběh a chování programu. Tuto vlastnost konečného automatu studenti ocení například při vytváření programu, který detekuje stisknutí (a puštění) více tlačítek najednou.

Protože se studenti na laboratorním cvičení setkají s problematikou konečného automatu poprvé, je vhodné vytvořit jednoduchý ukázkový příklad návrhu a implementace.

Na toto téma navazuje druhá laboratorní úloha Komunikace s PC a ladění pomocí sériové linky (viz Úloha 2.2).

1.2.4 Analogový a digitální signál

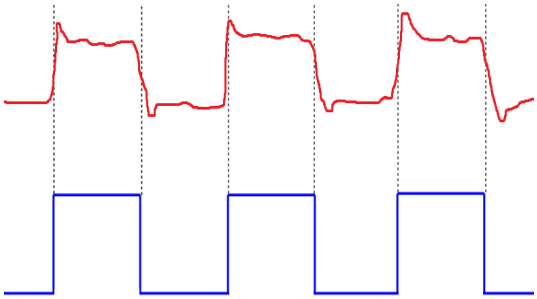
Rozlišujeme dva druhy signálu, jakožto nosiče informace, analogový a digitální. Analogový signál charakterizujeme jako spojitou funkci v čase i napětí. Analogový signál tedy může nabývat libovolné hodnoty. Oproti tomu digitální signál spojitou funkcí není a proto může nabývat pouze omezeného množství hodnot (nejčastěji dvou). Rozdíl mezi analogovým a digitálním signálem vidíme na obrázku 1.4.



Obrázek 1.4: Rozdíl mezi analogovým a digitálním signálem

Ač by měl dle definice digitální signál na obrázku 1.5 nabývat pouze striktně jednu ze dvou hodnot (jak je zobrazeno v dolní části obrázku), v praxi se velikost signálu často drobně mění (jak je zobrazeno v horní části obrázku). Především změna signálu není zcela okamžitá, ale spíše pozvolná. Protože nás ovšem zajímá pouze jestli vstupní veličina je nebo není v daném stavu (např. jestli RGB LED svítí nebo ne), tyto drobné odchylky zanedbáváme. Tyto odchylky jsou navíc tak malé, že ani nebývají postřehnutelné okem.

Přípravky Arduino poskytují dva typy vstupů, analogové a digitální. Konkrétně na přípravku Arduino Uno nalezneme 14 digitálních a 6 analogových vstupních pinů. Liší se rozdílným zpracováním příchozího napětí – digitální vstup pouze registruje, zda napětí na pinu je (5V) nebo není (0V), zatímco



Obrázek 1.5: Odchylky digitálního signálu

analogový měří velikost napětí a to poté převádí do digitální podoby na číslo. Analogové vstupy lze tedy teoreticky využít zcela stejně jako digitální.

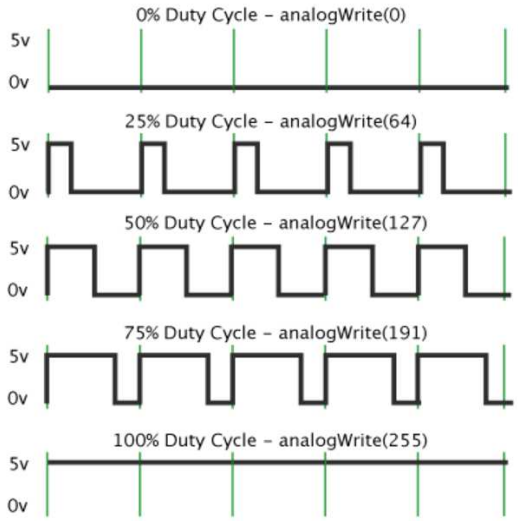
Analogové vstupy jsou měřeny a zpracovávány analogovými obvody (ekvivalentně k tomu digitální vstupy jsou zase zpracovávány digitálními obvody). Analogové obvody jsou více náchylnější k šumu a zkreslení (drobné odchylky v napětí) než obvody digitální. Analogové obvody je také často složitější navrhnout, na druhou stranu digitální obvody bývají dražší [13].

Na přípravku Arduino Esplora nalezneme hned několik analogových vstupů, jelikož obsahuje zabudované senzory, jako například teploměr nebo mikrofon. Analogový signál, které tyto senzory snímají je pro snadnější manipulaci nutné převést na signál digitální pomocí analogově digitálního převodníku (zkráceně A/D převodníku). A/D převodníky převádí vstupní analogovou veličinu na digitální signál pomocí vzorkování (proces odběru vstupního signálu v definovaných okamžicích). Při převodu analogového signálu na digitální vždy dochází ke ztrátě informace.

Ekvivalentně k A/D převodníkům existují i digitálně analogové převodníky (zkráceně D/A převodníky), které převádí vstupní digitální veličinu na analogový signál. D/A převodník lze využít například při generování tónu. Mikroprocesor přípravku Arduino Esplora neobsahuje D/A převodník a proto k využití vestavěného piezoměniče využívá pulsně šířkovou modulaci, neboli PWM (Pulse Width Modulation).

PWM je ve své podstatě jednoduchý D/A převodník. Simuluje analogový signál pomocí změny „šířky“ signálu, tzv. střídy (tj. poměru délky signálu v logické jedničce ku délce signálu v logické nule během jedné periody). Střídu (Duty Cycle) vyjadřujeme poměrem nebo procentuálně. Pokud je hodnota střídy 100 %, nachází se signál neustále v logické jedničce, pokud je hodnota střídy 0 %, pak se signál nachází v logické nule.

Na některé digitální piny přípravku Arduino lze připojit PWM. Na obrázku 1.6 vidíme rozdíl v průběhu signálu při zápisu různé číselné hodnoty na pin pomocí vestavěné funkce `analogWrite()`. Takto lze snadno ovládat například jas LED.



Obrázek 1.6: Pulsně šířková modulace (PWM) [4]

1.2.4.1 Analýza

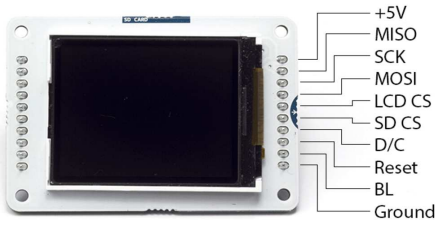
Jelikož při studiu vestavných systémů studenti přicházejí neustále do kontaktu s analogovým a digitálním signálem, je vhodné toto téma vysvětlit v některé z prvních laboratorních úloh. Při používání přípravku Arduino Uno se navíc studenti setkají s graficky rozlišenými analogovými a digitálními piny.

Studenti by si měli uvědomit, že k signálu získaného z analogových vstupů lze přistupovat dvěma způsoby – buď nás zajímá jak se průběh signálu v čase mění (analogový přístup) nebo nás zajímá pouze zda signál je nebo není v logické jedničce (digitální přístup). Na toto téma je vhodné vytvořit úlohu rozdělenou do dvou částí, první se bude zabývat analogovým přístupem k signálu a druhá digitálním přístupem k signálu. Cílem úlohy je zdůraznit, že rozdíl mezi analogovými a digitálními vstupy spočívá pouze v rozdílném zpracování stejného vstupního napětí.

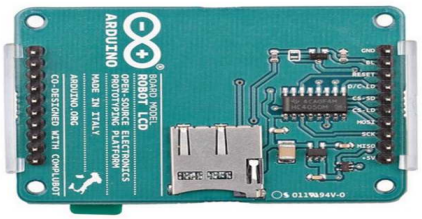
Na toto téma navazuje třetí laboratorní úloha Joystick a RGB dioda (viz Úloha 2.3).

1.2.5 TFT LCD displej a rozhraní SPI

Součástí přípravku Arduino Esplora je konektor pro TFT LCD displej. Na tento displej lze vykreslovat text, obrázky i různé geometrické tvary pomocí knihovny TFT, která je součástí platformy Arduino. Velikost úhlopříčky displeje je 1.77 palců. Displej má rozlišení 160x128 pixelů. Na obrázku 1.7 vidíme přední stranu displeje s popiskem jednotlivých pinů. Na zadní straně displeje se nachází slot na mikro SD kartu (viz obrázek 1.8). Cena displeje je v době psaní práce cca 400 Kč bez DPH.



Obrázek 1.7: Přední strana displeje s popisem pinů [5]



Obrázek 1.8: Zadní strana displeje [5]

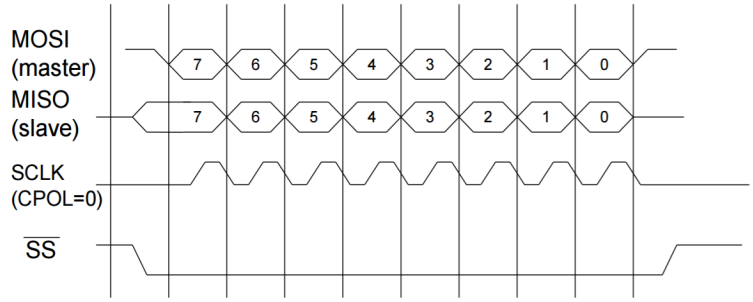
Ke komunikaci mezi mikrokontrolérem a displejem je použito rozhraní SPI (Serial Peripheral Interface). Rozhraní SPI je (stejně jako USART) jednou z fyzických implementací principu sériové komunikace. Používá se ke komunikaci mikrokontroléru s jedním nebo více periferními zařízeními (např. řadiči displejů, A/D a D/A převodníky, pamětí EEPROM). Rozhraní se skládá z jednoho řídícího zařízení (tzv. Master) a jednoho nebo více podřízených zařízení (tzv. Slave). Sběrnici rozhraní tvoří tři vodiče pro přenos dat a volitelně ještě jeden vodič pro výběr podřízené periferie.

K přenosu slouží signály MISO (Master In Slave Out), který umožňuje přenos dat od podřízeného zařízení k řídícímu a MOSI (Master Out Slave In), který naopak umožňuje přenos dat od řídícího zařízení k podřízenému. Synchronizaci přenosu zajišťuje hodinový signál SCLK (Serial Clock) řídícího zařízení. Pokud je v zapojení více podřízených zařízení, je k jejich výběru nutné použít ještě signál SS (Slave Select) (případně se setkáme i s názvem CS (Chip Select)). Řídící zařízení pak nastaví hodnotu signálu SS podřízeného zařízení do logické nuly v případě, že s ním chce komunikovat, jinak je nastaví do logické jedničky. Tak je umožněna komunikace více zařízení pomocí sdílených sběrnic. Časové průběhy signálů rozhraní SPI vidíme na obrázku 1.9.

Platforma Arduino poskytuje knihovnu SPI, kterou je při práci s displejem nutné zahrnout do projektu.

1.2.5.1 Analýza

Prostřednictvím displeje se studenti seznámí s další možnou fyzickou implementací principu sériové komunikace, rozhraním SPI. Pomocí displeje mohou navíc navíc zobrazovat obrázky uložené na SD kartě. Displej je ideální k vý-



Obrázek 1.9: Časové průběhy SPI [6]

pisu výstupu vestavěných senzorů přípravku Arduino Esplora (např. zobrazení křivky zvuku z mikrofonu). Pro výpis více položek na displej je také vhodné vytvořit menu, na kterém si studenti mohou zase procvičit implementaci konečného automatu.

Na toto téma navazuje čtvrtá laboratorní úloha Displej a senzory (viz Úloha 2.4).

1.2.6 Perzistentní paměti

Tato kapitola obsahuje úvod do problematiky perzistentních pamětí (EEPROM, SD karta). Perzistentní paměti umožňují, aby uložená data byla dostupná i po přerušení napájení. Tento typ paměti nazýváme také nevolatilní.

1.2.6.1 Paměť EEPROM

Mikroprocesory ATmega32u4 a ATmega328P přípravků Arduino Esplora a Arduino Uno obsahují 1024 B paměti EEPROM (Electrically Erasable Programmable Read-Only Memory).

Součástí platformy Arduino je knihovna EEPROM, usnadňující práci s touto pamětí. Umožňuje z EEPROM číst data pomocí funkce `EEPROM.read()` a zapisovat data do EEPROM pomocí funkce `EEPROM.write()` (nebo `EEPROM.update()`).

```
uint8_t EEPROM.read(int address);
uint8_t EEPROM.write(int address, uint8_t value);
uint8_t EEPROM.update(int address, uint8_t value);
```

address adresa v paměti
value data k zapsání do paměti

Návratová hodnota data uložená na dané adrese

Zápis na EEPROM trvá cca 3,3 ms. Jelikož životnost EEPROM nepřesahuje 100 000 cyklů zápisů/mazání, je třeba paměť využívat s rozvahou. Například místo funkce `EEPROM.write()` je vhodné použít funkci `EEPROM.update()`, která do paměti zapíše pouze v případě, že se zapisovaná data na dané adrese liší od těch již uložených.

1.2.6.2 SD karta

Na zadní straně TFT LCD displeje se nachází slot na mikro SD kartu. Snadné ovládání SD karty umožňuje knihovna SD, která je součástí platformy Arduino. SD knihovna podporuje souborové systémy FAT16 a FAT32. Ke komunikaci mezi mikrokontrolérem a SD kartou je použito rozhraní SPI. Z tohoto důvodu je k programu, který pracuje s SD kartou, nutné připojit i knihovnu SPI, která je také součástí platformy Arduino. V tabulce 1.1 vidíme, které piny zajišťují komunikaci přípravků Arduino Esplora a Uno s SD kartou.

Tabulka 1.1: Piny SPI rozhraní umožňující komunikaci s SD kartou

Arduino	MOSI	MISO	SCK	CS
Esplora	16	14	15	8
Uno	11	12	13	10

Při práci s SD kartou je třeba v programu nastavit pin CS (Chip Select) SD karty, například takto:

```
#define SD_CS     8
```

Makro SD_CS je poté nutné předat funkci `SD.begin()`, která inicializuje SD kartu i SD knihovnu.

```
bool SD.begin(uint8_t cspin);
```

cspin číslo CS (Chip Select) pinu SD karty

Návratová hodnota true v případě úspěšné inicializace, jinak false

Mezi další užitečné funkce SD knihovny patří například funkce pro vytvoření nové složky, otevření a smazání souboru nebo ověření, zda se daný soubor či složka nachází na SD kartě.

1.2.6.3 Analýza

Problematika pamětí se přednáší již ve 2. semestru v předmětu BI-SAP [11]. Zařazením laboratorní úlohy na práci s EEPROM a SD kartou si tedy studenti probranou látku vyzkouší v praxi. Výhodou je snadné ovládání obou pamětí pomocí knihoven EEPROM a SD.

Při práci s EEPROM je potřeba studenty poučit, že se musí snažit omezit zápisy do paměti, kvůli omezené životnosti. Studenti mohou později využít EEPROM i ve svých semestrálních pracích, pokud budou potřebovat permanentně uložit data, například nejlepší skóre v různých hrách. Paměť EEPROM je relativně malá (pouze 1024 B) a proto je v případě ukládání větších dat vhodné využít SD kartu. Na druhou stranu knihovna SD je poměrně veliká a je třeba studenty upozornit, že v případě většího projektu (např. semestrální práce) snadno může dojít k zaplnění paměti přípravku (velikost paměti je pouze 32 kB včetně 4 kB bootloaderu). V souvislosti s SD kartou je studentům představena i jiná fyzická implementace principu sériové komunikace, SPI. Dále si studenti vyzkouší i práci se souborovým systémem SD karty.

Na toto téma navazuje pátá laboratorní úloha SD karta (viz Úloha 2.5).

1.2.7 Přerušení

Mikroprocesor ATmega32u4 přípravku Arduino Esplora v nekonečné smyčce sekvenčně provádí jednotlivé instrukce instrukčního cyklu. Změnu pořadí těchto instrukcí umožňuje tzv. přerušení (interrupt). Přerušení je signál pro mikroprocesor, že došlo k události, která vyžaduje okamžitou pozornost.

Podle toho, co přerušení vyvolalo je dělíme na vnější (externí) a vnitřní (interní). Zdrojem vnějšího přerušení jsou běžně periferie (např. časovač nebo COM port). Vnitřní přerušení vyvolá samotný mikroprocesor v případě, že dojde k chybě (např. výpadek stránky nebo pokus o dělení nulou). Pokud o vnější přerušení žádá více periferií najednou, rozhoduje o jejich prioritě tzv. řadič přerušení. Vnější přerušení se dále dělí na maskovatelné (přerušení lze zakázat a opětovně povolit speciálními instrukcemi) a nemaskovatelné (vyvolané kritickou událostí, např. chybou v hardwaru).

Přerušení způsobí vyvolání podprogramu, tzv. obsluhy přerušení (Interrupt Service Routine). Obsluha přerušení nemá deklarovány žádné parametry ani návratovou hodnotu. Globální proměnné programu, jejichž hodnota se mění v obsluze přerušení, je vhodné deklarovat jako volatile (zakáže optimalizaci proměnné kompilátorem).

Adresu obsluhy přerušení obsahuje tzv. vektor přerušení. Všechny vektory přerušení nalezneme v tabulce vektorů přerušení (Interrupt Vector Table). Priorita jednotlivých přerušení je závislá na pozici v tabulce vektorů přerušení, čím nižší pozice (tedy nižší adresa vektoru přerušení), tím vyšší priorita.

Mikroprocesor ATmega32u4 umožňuje využít celkem pět pinů pro vnější přerušení. Všechny tyto piny ale používají vestavěné periferie na přípravku Esplora a proto nejsou vyvedené ven k použití.

Na rozdíl od Esplory, mikroprocesor ATmega328P přípravku Arduino Uno obsahuje pouze dva piny, které podporují vnější přerušení. Tyto piny jsou ale volně přístupné k využití.

V tabulce 1.2 vidíme, které piny Arduina umožňují vnější přerušení, jednoznačně definované číslem v záhlaví tabulky (tedy např. INT0 označuje pře-

rušení č. 0).

Tabulka 1.2: Piny Arduina, které podporují vnější přerušení

Arduino	INT0	INT1	INT2	INT3	INT4	INT5	INT6
Esplora	0	1	2	3	-	-	7
Uno	-	-	2	3	-	-	-

Platforma Arduino poskytuje knihovní funkci pro snadnou práci s vnějším přerušením, `attachInterrupt()`. Tato funkce umožňuje definovat jaká procedura (obsluha přerušení) se zavolá po vyvolání určeného přerušení.

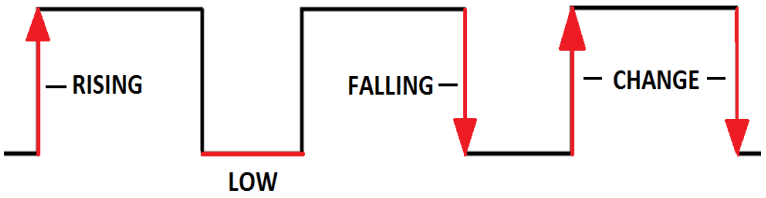
Uvnitř zvolené obsluhy přerušení nelze volat funkci `delay()` a návratová hodnota funkce `millis()` se neinkrementuje.

```
void attachInterrupt(uint8_t interrupt, void (*isr)
(), uint8_t mode);
```

interrupt číslo vyvolaného přerušení
isr obsluha přerušení
mode určuje, ve kterém stavu (nebo změně stavu) pinu je vyvoláno přerušení:

`LOW`	pin je ve stavu LOW (log. nula)
`RISING`	změna stavu pinu z LOW (log. nula) na HIGH (log. jedna)
`FALLING`	změna stavu pinu z HIGH (log. jedna) na LOW (log. nula)
`CHANGE`	změna stavu pinu

Návratová hodnota funkce nic nevrací



Obrázek 1.10: Znázornění konfigurace vyvolání přerušení

1.2.7.1 Analýza

Mechanismus přerušení je dnes běžným prvkem každého procesoru. Problematika přerušení se vyučuje již v druhém semestru v předmětu BI-SAP (Struktura a architektura počítačů) [14]. BI-SAP je ale obsahově velmi rozsáhlý předmět a není zde dostatek času, aby si studenti vyzkoušeli práci s vnějším přerušením. Je proto vhodné zařadit do výuky Arduina laboratorní úlohu na procvičení přerušení, ideálně v druhé půlce semestru, kdy už je teorie odpřednášena v předmětu BI-SAP. Úloha by měla pracovat s vnějším přerušením, protože studentům bude srozumitelnější výstup přerušení na některé z periferií spíše než pouhé hlášení chyby.

Jelikož Arduino Esplora nemá vyvedené piny pro vnější přerušení, bude pro tuto konkrétní úlohu nutné použít jiný přípravek, konkrétně Arduino Uno. Arduino Uno má právě dva piny umožňující vnější přerušení. Studenti si tedy vyzkouší jak práci s jiným typem vývojové desky, tak i připojení periferie k desce. Vhodné je použít některý z Arduino shieldů, aby se zamezilo možnosti, že studenti desku poničí (například špatným zapojením).

Na toto téma navazuje šestá laboratorní úloha Návrhářské přístupy pro složitější aplikace (viz Úloha 2.6).

Kapitola 2

Realizace

Tato kapitola se zabývá jednotlivými laboratorními úlohami. Popisuje a následně analyzuje celkem šest základních a tři bonusová zadání pro studenty.

Na přiloženém CD se nachází úlohy ve formě použitelné pro výukový systém EDUX. Tato zadání navíc obsahují teorii, která již byla zmíněna v kapitole 1.2, Tématické okruhy cvičení. Dále jsou přiložena i vzorová řešení jednotlivých úloh.

2.1 Úloha 1: Seznámení s Arduinem

Účelem úvodní úlohy je seznámit studenty s vývojovým prostředím Arduino a přípravkem Arduino Esplora. Je představena základní struktura projektu, tj. význam funkcí `setup()` a `loop()` (viz kapitola 1.2.1). Studenti se seznámí s knihovnou Esplora, naučí se používat vybrané knihovní funkce a zároveň se orientovat v oficiální dokumentaci platformy Arduino. Vhodné je začít například jednoduchým rozsvícením vestavěné RGB LED.

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX. Toto zadání je rozděleno do tří částí, které na sebe navazují.

Úloha 1.1: Morseova abeceda – rozblikání RGB LED

Vaším úkolem je na přípravku Arduino Esplora pomocí RGB LED zablikat písmena ARD v Morseově abecedě.

Nejprve připojte k programu knihovnu Esplora pomocí direktivy include (tato knihovna je součástí Arduino IDE). Dále vytvořte funkci `flash( int duration)` pro jedno bliknutí RGB LED. K implementaci využijte připravenou šablonu zdrojového kódu (viz příloha A.1.1).

Použijte funkci `Esplora.writeGreen (int value)` pro zapnutí ze-

lené RGB LED. Proměnná value zde určuje jas příslušné barvy – využijte připraveného makra BRIGHT_GREEN, které nastavuje jas zelené barvy na 50 (poté zkuste změnit hodnotu makra a experimentujte s intenzitou jasu).

Dále využijte funkci `delay (int x)` pro pozdržení programu na x milisekund. Tuto funkci využijete například pro určení délky doby rozsvícení LED. Na závěr ve funkci `loop()` pomocí funkce `flash()` a `delay()` zablikejte písmena ARD. Najděte vhodně dlouhé pozdržení programu mezi jednotlivými písmeny tak, aby byly jednotlivé znaky dobře čitelné.

Poznámka Znaky A,R,D v Morseově abecedě jsou „.-“ (A), „.-.“ (R), „-..“ (D).

Poznámka 2 Prostudujte si oficiální dokumentaci knihovny Esplora (viz [9]).

Úloha 1.2: Morseova abeceda – rozblikání RGB LED po stisku tlačítka

Pozměňte program tak, se zablikání ARD spustilo až po stisknutí tlačitka SWITCH 1. Využijte funkce `Esplora.readButton()`. Vyhledejte si parametry a návratovou hodnotu této funkce v oficiální dokumentaci knihovny Esplora.

Úloha 1.3: Morseova abeceda – změna barvy po stisku tlačítka

Tato úloha je bonusová. Pozměňte program tak, aby blikání ARD probíhalo cyklicky, ale stisknutí tlačítka okamžitě (i v průběhu svícení RGB LED) změnilo barvu blikání. Jedno stisknutí změní barvu jen jednou (i při dlouhém stisku). Použijte jen jedno tlačítko pro změnu barvy.

Poznámka 3 Při implementaci úlohy využijete funkce `millis()`, která vrací počet milisekund od spuštění programu.

Poznámka 4 Všimněte si, že po dokončení kompilace projektu se zobrazí informace o tom, kolik projekt zabírá místa v paměti a jaká je maximální velikost projektu. Dejte si proto pozor, aby vaše projekty tento limit nepřesáhly!

V úlohách 1.1 a 1.2 si studenti vyzkouší rozblikat RGB LED a detekovat zmáčknutí tlačítka. V úloze 1.3 je třeba okamžitě reagovat na událost (stisknutí tlačítka). Je zde vidět rozdíl v použití funkce `delay()` a `millis()` v programu. Úloha 1.3 je nejsložitější a vyžaduje nejvíce času. Je to zároveň

příprava na pozdější úlohu na přerušení (kde řešíme stejný problém, ale jinak implementujeme řešení).

2.2 Úloha 2: Komunikace s PC a ladění pomocí sériové linky

Ve druhé úloze se studenti seznámí s principem sériové komunikace mezi přípravkem a PC (viz kapitola 1.2.2). Naučí se pracovat se sériovým monitorem, který je součástí vývojového prostředí Arduino a který využijí při ladění svých úloh.

Dále si studenti vyzkouší naprogramovat konečný automat (viz kapitola 1.2.3). Před začátkem implementace je vhodné ukázat připravený příklad návrhu a implementace jednoduchého automatu na limonádu.

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Úloha 2.1: Výpis na sériový monitor

V první úloze si vyzkoušíte výpis na sériový monitor. Nejdříve na začátku programu zapněte sériovou komunikaci pomocí funkce `Serial.begin( speed)` a její parametr nastavte na 9600 baudů. Baud je jednotka modulační rychlosti, která udává počet změn stavu signálu za sekundu. Zde tedy určujete rychlost přenosu 9600 bitů za sekundu.

Detekujte zmáčknutí tří tlačítek (SWITCH 1 až SWITCH 3) a v případě stisknutí rozsviťte RGB LED (každé tlačítko rozsvítí jinou barvu) a název dané barvy vypište (jednou) na sériový monitor. V případě, že není žádné tlačítko zmáčknuté, ponechte RGB LED vypnuté.

Poznámka Sériový monitor spustíte kliknutím na ikonu lupy v pravém horním rohu Arduino IDE.

Poznámka 2 Prostudujte si oficiální dokumentaci knihovny Serial (viz [15]).

Úloha 2.2: Morseova abeceda

V prvním cvičení jste si vyzkoušeli pomocí RGB LED zablikat písmena ARD v Morseově abecedě. Dnes na tuto úlohu navážeme a vaším cílem bude zablikat písmena, slova a věty poslaná z PC přes sériovou linku.

Nejdříve si stáhněte soubor morse_letters.h [1], ve kterém najdete pole řetězců letters obsahující všechna písmena v Morseově kódu. Použijete

toto pole ve svém programu při překladu písmen v latince zaslaných z PC. Doporučujeme použít funkci `flash(int duration)` pro zablikání jednoho znaku (tečky nebo čárky) z minulého cvičení a vytvořit novou funkci `flashSequence(char * sequence)`, pro zablikání jednoho celého písmene.

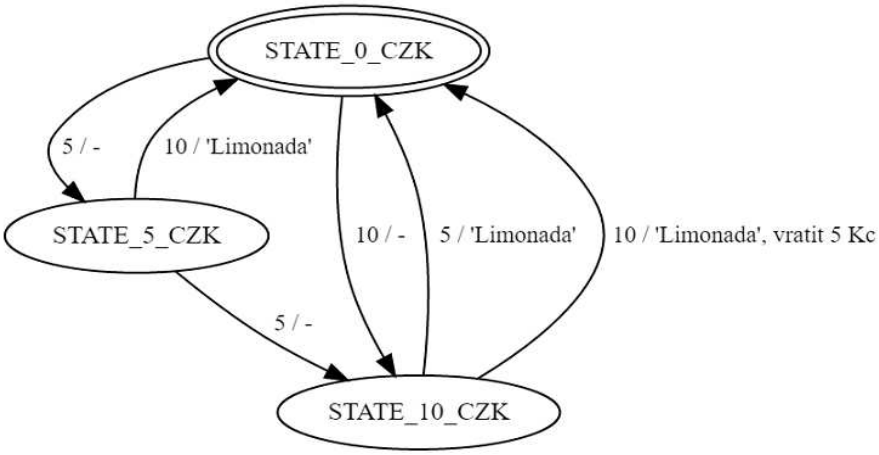
Nezapomeňte odlišovat délku mezer mezi jednotlivými písmeny a slovy.

Poznámka 3 Povšimněte si rozdílu mezi funkcemi `Serial.print()` a `Serial.println()`.

Příklad návrhu a implementace konečného automatu

Zde uvedeme krátký příklad návrhu a implementace jednoduchého automatu na limonádu. Do automatu můžete vhazovat pouze mince o hodnotě 5 a 10 Kč. Limonáda stojí 15 Kč, pokud vhodíte mince navíc, automat je vrátí.

- Nejprve navrhneme a graficky znázorníme řešení pomocí grafu přechodů (viz obrázek 2.1). V tomto případě jednotlivé stavy znázorňují celkový počet korun, který jsme již do automatu vložili.
- Hrany neboli přechody automatu jsou označeny vstupem a výstupem. Vstup jsou zde vhozené peníze (např. 5 Kč) a výstup vydaná limonáda, případně i vrácené peníze navíc.
- Vstup je v grafu označen znakem vedle hrany před lomítkem, výstup naopak za lomítkem.
- Pozn.: Jedná se o automat typu Mealy (podrobněji v předmětu BI-SAP).



Obrázek 2.1: Úloha 2: Automat na limonádu

- Implementace automatu na limonádu:

```
1  enum states {
2    STATE_0_CZK, STATE_5_CZK, STATE_10_CZK
3  };
4  enum states STATE, NEXT_STATE;
5
6  void setup() {
7    // pocatecni inicializace
8    STATE = STATE_0_CZK;
9  }
10
11 void loop() {
12 // nacteni vstupu automatu (5 nebo 10 Kc)
13   switch (STATE)
14   {
15         case STATE_0_CZK:
16                 if (vstup == 5) {
17                         NEXT_STATE =
                              STATE_5_CZK;
18                 } else if (vstup == 10) {
19                        NEXT_STATE =
                             STATE_10_CZK;
20                 }
21                 break;
22
23         case STATE_5_CZK:
24                 if (vstup == 5) {
25                        NEXT_STATE =
                             STATE_10_CZK;
26                 } else if (vstup == 10) {
27                         Vydej_limonadu();
28                         NEXT_STATE =
                              STATE_0_CZK;
29                 }
30                 break;
31
32         case STATE_10_CZK:
33                 if (vstup == 5) {
34                         Vydej_limonadu();
35                         NEXT_STATE =
                              STATE_0_CZK;
36                 } else if (vstup == 10) {
```

```
37                              Vydej_limonadu();
38                              Vrat_penize();
39                              NEXT_STATE =
                                   STATE_0_CZK;
40                      }
41    }
42    STATE = NEXT_STATE;
43 }
```

Úloha 2.3: Výpis tlačítek na sériový monitor

Naprogramujte konečný automat, který bude reagovat na stisknutí (resp. uvolnění) tlačítek SWITCH 1 a 2 výpisem na sériový monitor a bliknutím RGB LED. Doporučujeme si nejprve prostudovat ukázkový příklad zadání a implementace konečného automatu na limonádu.

Pokud došlo ke stisknutí (a uvolnění) tlačítka SWITCH 1, proveďte Akci A:

Akce A

- výpis „AKCE A“ na sériový monitor (pouze jednou)
- 100 ms bliknutí červené LED

Pokud došlo ke stisknutí (a uvolnění) tlačítka SWITCH 2, proveďte Akci B:

Akce B

- výpis „AKCE B“ na sériový monitor (pouze jednou)
- 100 ms bliknutí zelené LED

Pokud došlo ke stisknutí (a uvolnění) obou tlačítek SWITCH 1 i SWITCH 2, proveďte Akci C:

Akce C

- výpis „AKCE C“ na sériový monitor (pouze jednou)
- 100 ms bliknutí modré LED

K implementaci využijte připravenou šablonu zdrojového kódu (viz příloha A.2.1). Doporučujeme si nakreslit graf přechodů.

Poznámka 4 Každou akci proveďte důsledně až po uvolnění tlačítka (ne již během stisknutí).

V úlohách 2.1 a 2.2 si studenti vyzkouší práci se sériovým monitorem a zopakují si ovládání RGB LED a tlačítek z předchozí úlohy. Seznámí se s funkcemi knihovny Serial platformy Arduino umožňující sériovou komunikaci.

V úloze 2.3 je třeba reagovat na více vyvolaných událostí (stisknutí (a puštění) tlačítek). Tento typ úloh je vhodné řešit pomocí konečného automatu. Studenti mají k dispozici příklad návrhu i implementace jednoduchého konečného automatu, kterým se mohou při vypracování úlohy inspirovat.

2.3 Úloha 3: Joystick a RGB dioda

Účelem třetí úlohy je ukázat, že rozdíl mezi analogovými a digitálními vstupy spočívá pouze v rozdílném zpracování stejného vstupního napětí (viz kapitola 1.2.4). K tomu budeme využívat analogovou veličinu získanou z vestavěného joysticku (souřadnice jeho polohy). Nejprve nás bude zajímat, jak se průběh signálu v čase mění – hodnotu analogové veličiny (převedenou na číslo) zobrazíme jako výstup RGB LED (pohybem joysticku tedy budeme měnit jas RGB diody). Poté se zaměříme pouze na to, zda na vstupu je napětí nebo není – analogový vstup tedy budeme vnímat jako by byl digitální. Výstup opět zobrazíme na RGB LED (pohybem joysticku tedy buď rozsvítíme nebo zhasneme RGB LED).

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Úloha 3.1: Joystick - analogový přístup

Detekujte pohyb joysticku a podle jeho polohy (nahoře, dole, vpravo, vlevo) rozsviťte RGB LED (pro každý směr zvolte jinou barvu). Jas RGB LED regulujte podle toho, jak moc je joystick vychýlen od základní pozice ve středu. Pokud tedy bude joystick v krajní poloze, bude RGB LED svítit nejjasněji, zatímco ve středu bude vypnutá.

Nejdříve vyhledejte v dokumentaci knihovny Esplory funkce vhodné ke čtení polohy joysticku. Vypište si na sériový monitor hodnoty souřadnic x a y.

Dále pomocí vestavěné funkce `writeRGB()` zapněte příslušné RGB

[1]Soubor morse_letters.h je součástí přiloženého CD.

LED. Jako jeden z parametrů této funkce zvolte aktuální hodnotu souřadnice joysticku (x pro horizontální pohyb, y pro vertikální pohyb).

Jelikož se rozsah souřadnice joysticku a jasu RGB LED liší, je třeba rozsah souřadnice „přemapovat“ na hodnoty od 0 do 255. Toho docílíte použitím funkce `map()`, viz poznámka.

Vypisujte na sériový monitor jak původní hodnotu souřadnice joysticku, tak i přemapovanou hodnotu, použitou pro ovládání RGB LED.

Poznámka Pro „přemapování“ rozsahu hodnot jedné periferie na druhý (např. slider a RGB LED) můžete využít funkce `map()`, viz [16].

Úloha 3.2: Joystick - digitální přístup

Ve druhé části úlohy opět podle polohy joysticku rozsviťte RGB LED (pro každý směr zvolte jinou barvu). Nyní ovšem regulujte intenzitu jasu RGB LED pomocí slideru. Pokud je tedy joystick nakloněn např. vlevo, rozsviťte příslušnou LED o intenzitě jasu, která odpovídá aktuální poloze slideru (vlevo nejvyšší, vpravo nejnižší).

Pokud se joystick nachází uprostřed ponechte RGB LED vypnuté. Při stisknutí tlačítka joysticku zapněte piezoměnič a výšku tónu ovládejte přes slider.

Vyhledejte v dokumentaci knihovny Esplory funkci vhodnou k detekci stisknutého tlačítka. Dále pomocí funkce `writeRGB()` zapněte příslušné RGB LED. Jako jeden z parametrů této funkce zvolte hodnotu polohy slideru. Najděte tedy v dokumentaci vhodnou funkci pro čtení polohy slideru (povšimněte si v jakém rozsahu se toto číslo pohybuje!). Vypisujte na sériový monitor jak původní hodnotu slideru, tak i přemapovanou hodnotu, použitou pro ovládání LED.

Nakonec implementujte čtení hodnoty tlačítka na joysticku. V případě, že je stisknuté, zapněte piezoměnič pomocí funkce `Esplora.tone()` (výška tónu dle pozice slideru). V opačném případě ponechte piezoměnič vypnutý.

K implementaci využijte připravenou šablonu zdrojového kódu (viz příloha A.3.1).

Poznámka 2 RGB LED využívá stejný timer pro PWM (pulsně šířkovou modulaci – způsob převedení digitálního signálu na analogový) pro červenou složku LED jako pro tón. Červená LED tedy po zahrání tónu přestane do resetu fungovat. Je to chyba v Esploře a nemusíte se tím zabývat.

Úloha 3.3: Hra Bum do krtka

Vytvořte vlastní verzi hry Bum do krtka na přípravku Arduino Esplora. Princip hry je jednoduchý – na RGB LED budou v krátkých intervalech problikávat 4 barvy (každá je přiřazena jednomu tlačítku) a vaším cílem je stihnout stisknout správné tlačítko než se rozsvítí další barva. Vítězí ten, který zvládne správně stisknout více než polovinu tlačítek ze zadaného intervalu.

K implementaci využijte připravenou šablonu zdrojového kódu (viz příloha A.3.2).

V úloze 3.1 studenti sledují, jak se data získaná z joysticku v čase mění (např. jak moc vlevo se joystick posunul). Získaná data tedy vnímají „analogově“ (analogově je v uvozovkách, protože analogová hodnota je převedena s určitou přesností na diskrétní množinu hodnot rozsahu příslušné periferie (např. u RGB LED na 0 až 255)).

V úloze 3.2 naopak pracují s přečtenými hodnotami jako s digitálními, tedy je podstatné pouze jestli je nebo není joystick v určité poloze (tj. vlevo, vpravo, nahoře, dole).

V rámci úlohy si studenti vyzkouší práci s joystickem, sliderem a piezoměničem. V souvislosti s piezoměničem lze studentům vysvětlit problematiku pulsně šířkové modulace (viz kapitola 1.2.5).

Poslední úloha 3.2, hra Bum do krtka, je jednodušší a slouží k zopakování použití již probraných periferií (RGB LED, tlačítka) a výpisu na sériový monitor.

2.4 Úloha 4: Displej a senzory

Účelem této úlohy je podrobněji seznámit studenty s vestavěnými senzory (mikrofonem, tříosým akelerometrem, teploměrem a světelným senzorem) a TFT LCD displejem na přípravku Arduino Esplora (viz kapitola 1.2.5).

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Úloha 4.1: Zobrazení výstupu senzorů na TFT LCD displej

Zobrazte na displeji hodnotu teploměru, mikrofonu a akcelerometru. Vytvořte menu, kde bude možné si zvolit jeden z těchto senzorů a vypsat jeho výstup na displej. Menu se bude ovládat pomocí tlačítek SWITCH 1 až 4:

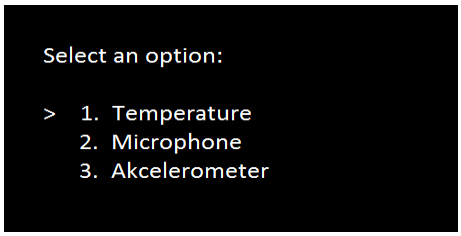
- **SWITCH 1**: pohyb dolů

- **SWITCH 2**: vybrat senzor
- **SWITCH 3**: zpět do menu
- **SWITCH 4**: pohyb nahoru

Poznámka Prostudujte si dokumentaci knihovny pro ovládání TFT LCD displeje (viz [17]).

Domovská obrazovka

Menu neboli domovská obrazovka slouží jako rozcestník pro výpis jednotlivých senzorů. Může vypadat například jako na obrázku 2.2.



Obrázek 2.2: Úloha 4: Domovská obrazovka

Pohyb po menu se zobrazí pomocí znaku „>“ před prvkem seznamu. (Zde se tedy aktuálně nacházíte na položce Temperature.) Dejte pozor, aby displej při pohybu po menu neblikal (tj. nemažte při posuvu znaku „>“ celou obrazovku, ale pouze tento znak). Zobrazení jednotlivých položek menu vypadá následovně:

Teploměr

Zvolením položky teploměr v menu, přejdete na obrazovku, kde se bude průběžně zobrazovat teplota (aktualizovaná každé 2 sekundy) ve stupních Celsia. Výpis může vypadat například jako na obrázku 2.3.

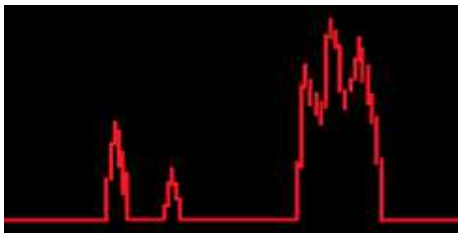
Obrázek 2.3: Úloha 4: Zobrazení výstupu teploměru

Tip: Pro obnovu dat po 2 sekundách, využijte funkce `millis()` z první úlohy.

Poznámka 2 Funkce pro výpis na displej `EsploraTFT.text()` bere jako parametr pole znaků, ale funkce pro přečtení teploty `readTemperature()` vrací číselnou hodnotu. Pro zobrazení teploty na displej tedy nejdříve přetypujte získaná data na string a poté použijte funkci `toCharArray()`.

Mikrofon

Zvolením položky mikrofon v menu, přejdete na obrazovku, kde se bude vykreslovat křivka podle momentálního zvuku z mikrofonu. Posuňte křivku na ose y lehce nahoru tak, aby byla vidět i v klidovém stavu, například jako na obrázku 2.4.



Obrázek 2.4: Úloha 4: Zobrazení výstupu mikrofonu

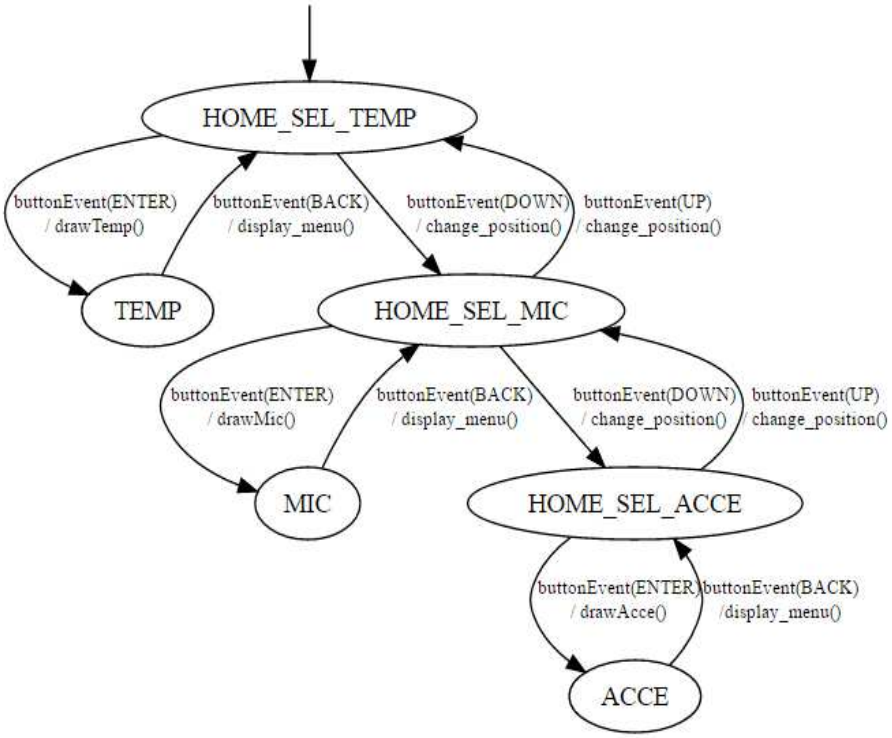
Akcelerometr

Po zvolení položky akcelerometr v menu, zobrazíte na displeji kolečko, které se bude pohybovat v závislosti na aktuálním náklonu přípravku Esplora (viz obrázek 2.5).

Obrázek 2.5: Úloha 4: Zobrazení výstupu akcelerometru

Poznámka 3 Dejte pozor na zapisování mimo rozsah displeje - často způsobuje „zvláštní“ chování displeje (např. zobrazení útvarů na náhodných pozicích).

Úlohu implementujte pomocí konečného automatu. Kostru programu vytvořte podle připraveného grafu přechodů (viz obrázek 2.6). K implementaci využijte připravenou šablonu zdrojového kódu (viz příloha A.4.1).



Obrázek 2.6: Úloha 4: Graf přechodů konečného automatu

Použijte ve svém programu funkci `buttonEvent(int button)` (viz příloha A.4.2). Tuto funkci využijete při implementaci pohybu po domovské obrazovce. Funkce vrací true, jestliže bylo zadané tlačítko (parametr button) stisknuto a puštěno, případně false pokud stisknuto nebylo nebo naopak nebylo ještě puštěno.

Ve čtvrté úloze si studenti vyzkouší zobrazit výstup vestavěných senzorů přípravku Arduino Esplora na displeji. Výběr senzoru se provede pomocí vytvoření domovské obrazovky (menu). Implementací menu si studenti zopakují a procvičí naprogramování konečného automatu (dle předem připraveného grafu přechodů). Dále se také seznámí funkcemi knihovny pro ovládání TFT LCD displeje.

Klade se důraz na to, aby studenti nepřepisovali neustále zbytečně celý displej, ale pouze konkrétní změněnou část a jen pokud dojde k nějakému pohybu po displeji (a omezili tak problikávání displeje).

Implementace funkce `buttonEvent()` pro detekci stisknutí (a puštění) tlačítka je studentům poskytnuta již hotová, aby se mohli zaměřit hlavně na ovládání periferií.

2.5 Úloha 5: SD karta

V páté úloze se studenti naučí pracovat s nevolatilní pamětí EEPROM a SD kartou (viz kapitola 1.2.6). Seznámí se s funkcemi knihoven pro jejich ovládání, které jsou součástí platformy Arduino.

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Úloha 5.1: Zobrazení kořenového adresáře SD karty

Vaším prvním úkolem je vypsat na TFT LCD displej obsah kořenového adresáře SD karty.

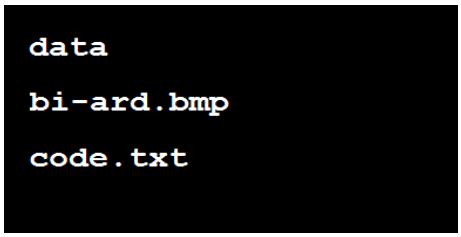
Nejprve inicializujte displej stejně jako v minulém cvičení (nezapomeňte k vašemu programu připojit správné knihovny pro práci s displejem). Přidejte k programu také knihovnu SD karty.

Inicializujte SD kartu pomocí funkce `SD.begin(cspin)`, jejímž parametrem je číslo CS (chip select) pinu. Tento pin je nezbytný pro komunikaci mikrokontroléru s SD kartou (viz kapitola 1.2.5). Přidejte proto do programu řádek:

```
#define SD_CS  8
```

Makro SD_CS poté zadejte jako parametr funkci `SD.begin()`. Pokud funkce `SD.begin()` vrátí false, vypište na displej „No SD card inserted!“ a ukončete program.

Dále nastavte barvu pozadí na černou a barvu textu na bílou. Pomocí funkce `SD.open()` otevřete kořenový adresář a vypište jeho obsah na displej. Pro vypsání souborů v adresáři využijte funkci `SD.openNextFile()`. Vaše úloha by po dokončení měla vypadat na displeji jako obrázek 2.7:

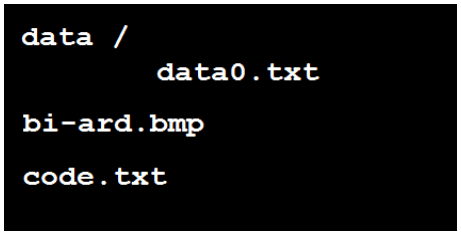


Obrázek 2.7: Úloha 5: Zobrazení kořenového adresáře SD karty

Poznámka Prostudujte si knihovnu SD karty (viz [18]).

Úloha 5.2: Zobrazení adresářové struktury SD karty

Tato úloha je bonusová. Vypište na displej celou adresářovou strukturu SD karty. Pro vypsání obsahu jednotlivých adresářů doporučujeme použít rekurzi. Vaše úloha by po dokončení měla vypadat na displeji jako obrázek 2.8:



Obrázek 2.8: Úloha 5: Zobrazení adresářové struktury SD karty.

Poznámka 2 Neměňte obsah SD karty!

Úloha 5.3: Zobrazení obrázku z SD karty

Nejprve inicializujte TFT LCD displej. Nastavte pozadí displeje na bílou barvu. Poté inicializujte SD kartu pomocí funkce `SD.begin()`. Pokud funkce `SD.begin()` vrátí false, vypište na displej „No SD card inserted!“ a ukončete program.

Zkontrolujte, zda se v kořenovém adresáři SD karty nachází obrázek s názvem **bi-ard.bmp**. Využijte k tomu funkce `SD.exists()`. Pokud funkce `SD.exists()` vrátí false, vypište na displej „bi-ard.bmp missing!“ a ukončete program. Obsah SD karty nijak neměňte! Pokud obrázek na kartě naleznete, nahrajte ho pomocí funkce `SD.loadImage()` a zkontrolujte, zda se jedná o korektní soubor funkcí `SD.isValid()`. Pokud je obrázek validní, zablikejte zelenou RGB LED, pokud ne, pak zablikejte červenou.

Nyní zobrazte obrázek na displej do levého horního rohu.

Úloha 5.4: Posouvání obrázku po displeji

Tato úloha je bonusová. Posouvejte obrázek po displeji v závislosti na pohybu joysticku. Při stisknutí tlačítka SWITCH 1, uložte souřadnice polohy x a y do paměti EEPROM. Po resetování přípravku zobrazte obrázek v poloze určené souřadnicemi uloženými v EEPROM.

Nejprve je třeba nahrát do EEPROM prvotní souřadnice, kde x = 0 a y = 0. Vytvořte proto zvlášť nový projekt, kde pouze ve funkci setup dvakrát zavoláte funkci `EEPROM.update()`, například následovně:

```
1  #include <EEPROM.h>
2
3  void setup() {
4     // uloz hodnotu 0 na adresu 0 v~pameti
         EEPROM
5     EEPROM.update(0,0);
6     // uloz hodnotu 0 na adresu 1 v~pameti
         EEPROM
7     EEPROM.update(1,0);
8  }
9
10 void loop() {
11 }
```

Takto do paměti EEPROM nahrajete hodnotu 0 na adresy 0 a 1 (pouze v případě, že se na těchto adresách stejné hodnoty už nevyskytují, v tom případě nedochází k přepsání).

Nahrajte kód do přípravku, zavřete tento projekt a vraťte se k původnímu programu ze 2. úlohy. Přidejte detekci pohybu joysticku, tj. uložte aktuální hodnoty souřadnic x a y. Tyto hodnoty je potřeba namapovat na velikost displeje. Použijte k tomu funkci `map()`. Velikost displeje zjistíte pomocí funkcí `EsploraTFT.width()` a `EsploraTFT.height()`. Poté zobrazte na displej obrázek na nových souřadnicích. Vykreslujte obrázek v delším časovém intervalu (např. 500 ms).

V případě, že došlo ke stisknutí tlačítka SWITCH 1, použijte funkci `EEPROM.update()` k uložení aktuálních souřadnic. Aby se po vypnutí přípravku zobrazil obrázek na uložených souřadnicích, zavolejte ve funkci `setup()`, dvakrát funkci `EEPROM.read()` a přečtete tak uložené hodnoty souřadnic x a y. Návratová hodnota funkce `EEPROM.read()` je číslo uložené v paměti EEPROM na adrese, kterou zadáte jako parametr. Podle předchozího příkladu bychom tedy pro získání hodnoty souřadnice x volali x = `EEPROM.read(0)`.

Poznámka 3 Životnost paměti EEPROM je omezena na 100 000 zápisů/mazání. Vyhněte se proto příliš častému zápisu do paměti. Pokud nemusíte, použijte funkci `EEPROM.update()` místo `EEPROM.write()`.

Úloha 5.5: Secret Code

Vaší další úlohou bude nasimulovat fungování trezoru. Pomocí tlačítek na Arduino Esplora zadáte posloupnost čísel (neboli bezpečnostní heslo) a pokud je tento vybraný kód správný, trezor se vám otevře, jinak zůstane zamčený.

Na začátku je třeba trezor uzamknout. To provedete tak, že stisknete tlačítko joysticku, rozsvítíte červenou RGB LED a na displej vypíšete:

*** Locked! ***
Enter code:

Nyní následuje zadání hesla. Nejdříve na SD kartě otevřete soubor code.txt, kde najdete uložené bezpečnostní heslo. Toto heslo poté načtěte (nekopírujte!) do svého programu například v tomto formátu:

```
int secret_code[4]
```

Pole `secret_code[]` tedy bude obsahovat správný bezpečnostní kód (definovaný souborem na SD kartě) a pořadí stisknutých tlačítek se musí shodovat s číslicemi, které jsou v tomto poli uložené. Po zadání každé vstupní číslice pomocí tlačítka, vypište na displej hvězdičku (*). Načtěte všechny čtyři číslice a teprve poté, pokud byla nějaká číslice chybná, proveďte výpis ve tvaru:

Wrong secret code!
*** Locked! ***
Enter code:

V případě, že kód byl zadán bez chyby, vypište:

*** Unlocked! ***

a rozsviťte zelenou RGB LED. Takto signalizujete otevření trezoru.

V páté úloze si studenti nejprve vyzkouší vypsat obsah SD karty. Zobrazení dat se provede na TFT LCD displej a tak se zároveň zopakuje používání funkcí pro ovládání displeje ze čtvrté úlohy. V souvislosti s SD kartou se studenti seznámí i s komunikací pomocí sériového periferního rozhraní (SPI), které se využívá pro přenos mezi mikrokontrolérem a SD kartou.

Dále si studenti vyzkouší i práci s pamětí EEPROM (čtení i zápis dat). Je zdůrazněno, že je nutné omezit časté zápisy do paměti, kvůli nižší životnosti paměti EEPROM.

2.6 Úloha 6: Návrhářské přístupy pro složitější aplikace

V šesté úloze se studenti seznámí s problematikou přerušení (viz kapitola 1.2.7). Vyzkouší si naprogramovat vnější přerušení pomocí přípravku Arduino Uno, Funduino Joystick Shieldu a RGB LED. Naučí se používat funkci `attachInterrupt()` ke konfiguraci vnějšího přerušení.

Dále si studenti vyzkouší naprogramovat aplikaci komunikující s přípravkem Arduino Esplora nebo Arduino Uno pomocí sériové linky.

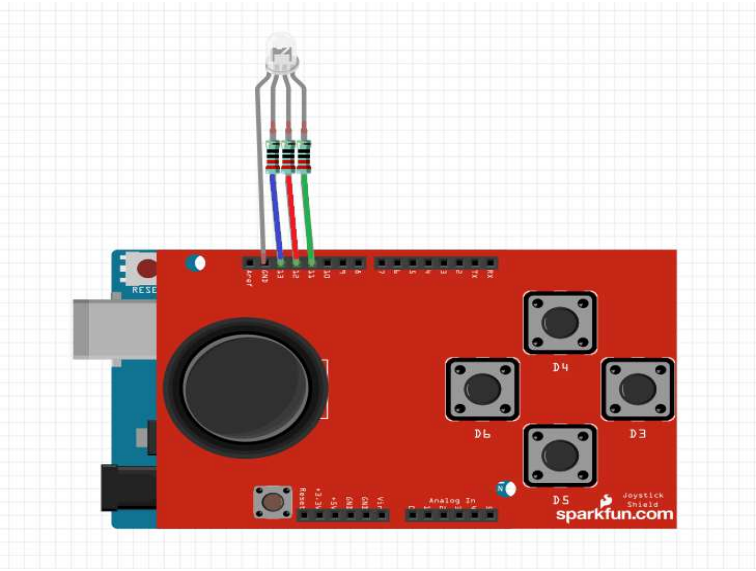
Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Úloha 6.1: Blikání RGB LED pomocí přerušení

V této úloze budete používat přípravek Arduino Uno a Funduino Joystick Shield V1.A.

Rozblikejte RGB LED s frekvencí 1 Hz. Při každém zmáčknutí tlačítka na Joystick Shieldu změňte (jednou) barvu RGB LED. Využijte vnějšího (externího) přerušení.

- Nejprve připojte RGB LED k joysticku podle obrázku 2.9.



Obrázek 2.9: Úloha 6: Zapojení RGB LED na Funduino Joystick Shield V1.A

- Založte nový projekt v Arduino IDE a nezapomeňte změnit v sekci Vývojová deska Arduino Esplora na Arduino Uno.
- Zadefinujte piny připojené RGB LED například následovně:

```
#define BLUE_PIN 11
#define GREEN_PIN 12
#define RED_PIN 13
```

- Ve funkci `setup()` tyto piny nastavte jako výstupní pomocí funkce `PinMode()` následovně:

```
pinMode(GREEN_PIN, OUTPUT);
```

- Nyní je třeba zjistit, na které piny je možné nastavit přerušení. Dle oficiální dokumentace naleznete na Arduino Uno (a dalších Arduino přípravcích založených na čipu ATmega328P) právě dva takové piny a to pin 2 a pin 3.
- Připojení pinů přípravku Arduino na tlačítka shieldu je následující:
  - tlačítko nahoru: pin 2
  - tlačítko dolů: pin 4
  - tlačítko vlevo: pin 5
  - tlačítko vpravo: pin 3
  - tlačítko joysticku: pin 8

  Externí přerušení tedy můžeme vyvolávat pomocí tlačítek nahoru a vpravo.
- Pro nastavení přerušení využijte funkci `attachInterrupt()`. Tato funkce v případě definované události (viz kapitola 1.2.7) vyvolá přerušení.

  Volání funkce tedy vypadá například takto:

```
1 #define INTERRUPT_PIN 2
2
3 void setup() {
4  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
5  attachInterrupt(digitalPinToInterrupt(
      INTERRUPT_PIN), do_something, FALLING);
6 }
7
8 void do_something() {
9 }
```

V této ukázce tedy po stisknutí pravého tlačítka dojde k vyvolání přerušení a zavolá se funkce `do_something()`.

- K rozsvícení RGB LED využijte funkce `digitalWrite()`.

Poznámka Pro práci s joystickem se můžete inspirovat zdařilou dokumentací k SparkFun Joystick Shield (viz [19]). Dejte si ovšem pozor na jiné zapojení pinů!

Poznámka 2 Proměnné, jejichž hodnotu měníte v obsluze přerušení, je vhodné deklarovat jako tzv. volatile, např.: `volatile int i`.

Úloha 6.2: Vývoj aplikace komunikující s přípravkem Arduino Esplora nebo Uno

V této úloze si vyzkoušíte naprogramovat aplikaci, která komunikuje s Arduinem pomocí sériové linky. Arduino bude aplikaci posílat údaje o teplotě získané z teploměru (v případě Arduino Uno o poloze joysticku) a aplikace naopak zadá Arduinu, kterou barvu RGB LED má rozsvítit. Následující návod se týká OS Windows.

Poznámka 3 Následuje návod na vypracování úlohy na přípravku Arduino Esplora. Pracujete-li na Arduino Uno, pozměňte jej dle toho (především co se týče čtení joysticku místo teploty).

- Založte v C/C++ IDE (např. v Netbeans) nový projekt.
- Implementaci samotné sériové komunikace máte předpřipravenou v souborech SerialPort.h a SerialPort.c [2]. Připojte tyto soubory ke svému projektu.

- Pro zahájení komunikace je třeba nejdříve otevřít příslušný COM port pomocí funkce `connect()`, kterou naleznete v souboru SerialPort.c. Prostudujte si, jak tato funkce pracuje.
- Ve funkci `main()` tedy zavolejte funkci `connect()` a předejte jí příslušné parametry, handle a název COM portu. Handle deklarujete například takto:

  ```
  HANDLE hFile;
  ```

- Zavolejte funkci `writeData()`, která pošle Arduinu informaci o barvě RGB LED k rozsvícení. Předejte handle jako parametr.
- Zavolejte funkci `readData()`, která přečte data poslaná z Arduina, konkrétně momentální teplotu ve stupních Celsia. Předejte handle jako parametr.
- Ošetřete návratové hodnoty všech tří volaných funkcí, tj. v případě, že vrátí false, nepokračujte dále v programu. Pozor! Před ukončením `main()` uvolněte handle pomocí funkce `CloseHandle()`:

  ```
  CloseHandle(hFile);
  ```

- Nyní otevřete nový projekt v Arduino IDE. Zahajte sériovou komunikaci zavoláním funkce `Serial.begin()`.
- Ve smyčce čtěte hodnoty ze sériové linky a podle získaných hodnot, rozsviťte příslušnou RGB LED.
- Zároveň čtěte hodnotu teploměru pomocí `Esplora.readTemperature()` a posílejte tento údaj aplikaci například funkcí `Serial.println()`.
- Nakonec nahrajte program do Arduina a spusťte kód v IDE. Zkontrolujte, že svítí správná RGB LED a na standardní výstup se vypisuje teplota.

Poznámka 4 Pro komunikaci s COM portem vyšším než 9, je třeba ve funkci `createFile()` zadat název portu takto: „\\\\.\\COM10“ (narozdíl od portů 1 až 8, u kterých stačí zadat „name“, tedy například: „COM8“).

Poznámka 5 Podrobnější informace o sériové komunikaci pod OS Windows viz [20].

V šesté úloze si studenti vyzkouší práci s přípravkem Arduino Uno. Naučí se tedy pracovat s některými novými vestavěnými funkcemi, jako například `pinMode()` nebo `digitalWrite()`.

V úloze 6.1 se studenti seznámí s mechanismem (vnějšího) přerušení. Pomocí funkce `attachInterrupt()` nastaví vnější přerušení a sami naimplementují obsah obsluhy přerušení. Je zdůrazněno, že obsluha přerušení má být ideálně co nejkratší, není vhodné v ní používat blokovací funkce a že proměnné měněné v obsluze přerušení je vhodné deklarovat jako volatile. Studenti mohou experimentovat se změnou módu přerušení (tj. při které změně na pinu dojde k přerušení).

Cílem úlohy 6.2 je naprogramovat aplikaci, která komunikuje s Arduinem pomocí sériové linky. Studenti si mohou vybrat, jestli budou pracovat s přípravkem Arduino Esplora nebo Arduino Uno. Připravený návod se ale týká pouze Arduina Esplora. Jelikož je implementace sériové komunikace v jazyce C++ v OS Windows značně rozsáhlá a složitá, studenti dostanou funkce pro otevření příslušného COM portu (`connect()`), zápis dat (`writeData()`) i čtení dat (`readData()`) již hotové. Jejich úkolem je si zdrojový kód prostudovat a příslušné funkce vhodně zavolat ve funkci `main()`. Pro srovnání je studentům představeno i (kratší a jednodušší) řešení úlohy v jazyce Python[3].

2.7 Bonusové úlohy

Tato kapitola obsahuje dodatečné úlohy, které lze využít k získání bodů navíc nebo v případě absence na cvičení.

2.7.1 Bonusová úloha 1

První bonusová úloha procvičuje práci s displejem, sliderem a RGB LED. Je složena ze dvou částí, které na sebe navazují.

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

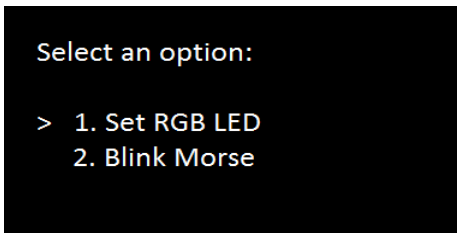
> První bonusová úloha je rozdělena na dvě části. V první části nastavíte hodnotu RGB LED pomocí slideru. Ve druhé části zablikáte v Morseově kódu krátký text zobrazený na displeji.
>
> Vytvořte menu, kde bude možné si zvolit jednu ze dvou částí bonusové úlohy. Menu se bude ovládat pomocí tlačítek SWITCH 1 až 4:
>
> - **SWITCH 1**: pohyb dolů
> - **SWITCH 2**: vybrat položku menu

[2]Soubory SerialPort.c a SerialPort.h jsou součástí přiloženého CD.
[3]Řešení 6. úlohy v jazyce Python je přiloženo na CD.

- **SWITCH 3**: zpět do menu
- **SWITCH 4**: pohyb nahoru

Domovská obrazovka

Menu neboli domovská obrazovka slouží jako rozcestník mezi oběma částmi bonusové úlohy. Může vypadat například jako na obrázku 2.10.



Obrázek 2.10: Bonusová úloha 1: Domovská obrazovka

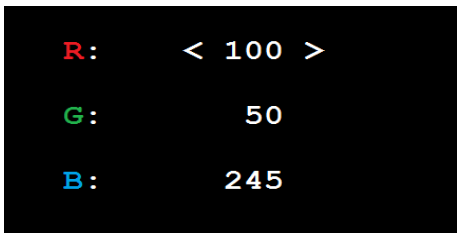
Pohyb po menu se zobrazí pomocí znaku „>“ před prvkem seznamu (na obrázku 2.10 se tedy nacházíte na položce Set RGB LED). Dejte pozor, aby displej při pohybu po menu neblikal (tj. nemažte při posuvu znaku „>“ celou obrazovku, ale pouze tento znak).

Pro implementaci menu využijte stavového automatu podobně jako ve čtvrté úloze. Doporučujeme si nejdříve nakreslit graf přechodů.

Bonusová úloha 1.1: Nastavení RGB LED pomocí slideru

Zobrazte hodnoty jasu jednotlivých barev RGB LED na displeji a měňte je pomocí slideru.

Obrazovka po zvolení položky menu Set RGB LED bude vypadat například jako na obrázku 2.11.



Obrázek 2.11: Bonusová úloha 1: Nastavení RGB LED pomocí slideru

- Pomocí tlačítka SWITCH 2 budete přecházet mezi položkami R, G a B (jednotlivé barvy RGB LED) a nastavovat pro každou barvu zvlášť hodnotu jasu (od 0 do 255) sliderem. Číselná hodnota jasu

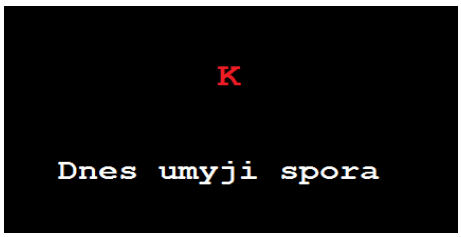
RGB LED se tedy bude s posuvem slideru na displeji měnit. Samotným posunem číslic se ale jas ihned nezmění, k tomu dojde až po stisknutí tlačítka SWITCH 1 (potvrzuje vybranou novou hodnotu jasu).

- Pohyb mezi položkami R, G a B se zobrazí pomocí znaků „<“ a „>“ okolo právě upravované hodnoty jasu příslušné barvy. (Zde tedy právě měníte jas červené barvy RGB LED.)
- Pro otestování aktuálně zobrazené hodnoty jasu využijete tlačítka SWITCH 4. Po jeho zmáčknutí krátce problikne aktuálně zobrazená hodnota na displeji. Uložené ovšem stále zůstává původní nastavení, k trvalé změně dochází pouze po stisknutí tlačítka SWITCH 1.
- Zobrazené hodnoty jasu na obrazovce Set RGB se nebudou měnit po přechodu mezi dalšími obrazovkami (např. menu).
- V úloze tedy využijete tlačítek SWITCH 1 až 4:
  - **SWITCH 1**: potvrzení aktuálně zobrazené hodnoty jasu na displeji
  - **SWITCH 2**: posun mezi jednotlivými položkami R, G a B
  - **SWITCH 3**: přesun zpět do menu
  - **SWITCH 4**: probliknutí RGB LED pro otestování nastavené hodnoty jasu

Bonusová úloha 1.2: Morseova abeceda

Zablikejte v Morseově kódu vámi zvolený text zobrazený na TFT LCD displeji.

Obrazovka po zvolení položky menu Blink Morse bude vypadat například jako na obrázku 2.12.



Obrázek 2.12: Bonusová úloha 1: Morseova abeceda

- Nejprve je třeba po jednotlivých písmenech vybrat text, který poté zablikáte.

- K výběru jednoho písmene abecedy využijte slider, tj. musíte vhodně namapovat číselný rozsah slideru (0 až 1023) na rozsah celé abecedy (0 až 26). Rozsah abecedy je zde 26 znaků písmen a 1 znak mezery. Spolu s posunem slideru se tedy budou na displeji postupně zobrazovat odpovídající písmena abecedy (viz „K“ na obrázku 2.12). Zobrazujte vždy pouze jedno písmeno.
- Mezeru při výběru na displeji znázorněte podtržítkem „_“.
- Pomocí tlačítka SWITCH 1 přidáte aktuálně zobrazené písmeno k textu, který bude později zablikán v Morseově kódu. Tento text také zobrazte na displeji (viz „Dnes umyji spora“ na obrázku 2.12). Nevolte text zbytečně dlouhý, 7 až 10 znaků je zcela postačující.
- Použijte tlačítko SWITCH 2 ke smazání posledního znaku textu k zablikání.
- Dále použijte tlačítko SWITCH 4 ke spuštění blikání vybraného textu v Morseově kódu.
- Nezapomeňte odlišit při blikání kódu délku mezer mezi jednotlivými záblesky, písmeny i slovy. Mezera mezi záblesky je nejkratší, poté následuje mezera mezi písmeny a nejdelší mezera je mezi slovy.
- V úloze tedy využijete tlačítek SWITCH 1 až 4:
  - **SWITCH 1**: přidá aktuálně zobrazené písmeno k textu k zablikání
  - **SWITCH 2**: smaže poslední písmeno zvoleného textu k zablikání
  - **SWITCH 3**: přesun zpět do menu
  - **SWITCH 4**: spustí zablikání zvoleného textu v Morseově kódu

V první bonusové úloze si studenti zopakují vytvoření menu na displeji implementované pomocí konečného automatu, stejně jako ve čtvrté úloze. Graf přechodů si ale tentokrát musí připravit sami. Zároveň si připomenou práci s knihovnou pro ovládání TFT LCD displeje.

2.7.2 Bonusová úloha 2

Ve druhé bonusové úloze si studenti zopakují práci s displejem, piezoměničem a pamětí EEPROM.

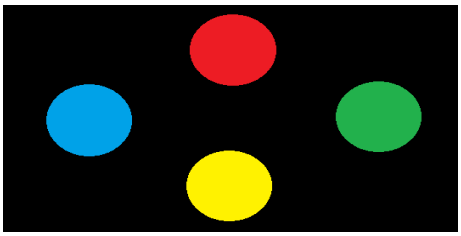
Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Ve druhé bonusové úloze vytvoříte vlastní verzi hry Simon Says na přípravku Arduino Esplora. Úloha je rozdělena na dvě části.

Bonusová úloha 2.1: Hra Simon Says

Princip hry Simon Says je jednoduchý – na displeji se zobrazí 4 barevné segmenty ve stejném sestavení jako tlačítka. Po zahájení hry začnou barevné segmenty blikat v (nejdříve kratších, časem stále delších) sekvencích. Cílem hry je po doblikání každé sekvence postupně stisknout tlačítka, ve stejném pořadí jako zhasínaly jednotlivé barevné segmenty.

- Na začátku hry zobrazte na displeji všechny barevné segmenty, například jako na obrázku 2.13.



Obrázek 2.13: Bonusová úloha 2: Hra Simon Says

- K vytvoření barevných segmentů můžete využít funkcí knihovny TFT LCD displeje pro vytvoření různých geometrických tvarů jako kruh (`EsploraTFT.circle()`) či čtverec (`EsploraTFT.rect()`).
- Stisknutím tlačítka joysticku zahájíte hru, tj. spustíte blikání barevných segmentů v jednotlivých sekvencích. Zvyšujte obtížnost hry tak, že nejprve spustíte kratší sekvence (např. pouze 3 bliknutí) a postupně je prodlužujete.
- Pro náhodné blikání segmentů můžete využít funkce `random()`, například takto:

  ```
  int to_blink = random(MAX_NUM)% 4;
  ```

  Máme k dispozici pouze čtyři barvy/tlačítka, proto modulo čtyřmi.
- Nepřepisujte zbytečně celý displej pokud nemusíte - při blikání jednotlivých barevných segmentů smažte pouze jeden segment a zbytek ponechte.

- Ve chvíli, kdy se uživatel splete a zadá chybnou sekvenci, se na displej vypíše skóre (počet správně stisknutých sekvencí).
- Stisknutím tlačítka joysticku znovu zahájíte hru.

Bonusová úloha 2.2: Hra Simon Says – ukládání skóre

- Využijte piezoměniče a přidejte zvukový doprovod ke každému bliknutí barevného segmentu během hry. Použijte funkce knihovny Esplora `tone()` a `noTone()` (parametrem těchto funkcí je frekvence tónu a délka tónu v ms). Pro každou barvu zvolte jinou frekvenci tónu, aby bylo snadnější rychle rozpoznat, jaký barevný segment právě bliknul.
- Pozměňte původní program tak, aby měl hráč nárok na 2 chyby, tj. prohra následuje až po třetí chybě. Počet chyb zobrazujte na displej (například do levého dolního rohu).
- Uložte nejvyšší skóre do paměti EEPROM a zobrazte ho po každé prohře spolu s aktuálním skóre. Nejprve je potřeba nahrát do EEPROM prvotní nejvyšší skóre (tedy 0), které poté budete pouze aktualizovat. Vytvořte proto zvlášť nový projekt, kde pouze ve funkci `setup()` zavoláte `EEPROM.update()`, například následovně:

```
1  #include <EEPROM.h>
2
3  void setup() {
4    // ukladam hodnotu 0 na adresu 0 v~
       pameti
5    // EEPROM
6     EEPROM.update(0,0);
7  }
8
9  void loop() {
10 }
```

Takto do paměti EEPROM nahrajete hodnotu 0 na adresu 0 (pouze v případě, že se na této adrese stejná hodnota už nevyskytuje, v tom případě nedochází k přepsání). Nahrajte kód do přípravku, zavřete tento projekt a vraťte se k vašemu původnímu programu.

- Nyní už pouze vždy po prohře přečtete nejvyšší skóre funkcí `EEPROM.read()` a v případě, že je aktuální skóre vyšší než skóre uložené v EEPROM, přepíšete toto skóre pomocí `EEPROM.update()`.
- Zvolte si „tajnou“ sekvenci tlačítek, po jejímž stisknutí (při zobrazení skóre po prohře) se nejlepší skóre smaže. Je tedy třeba vynulovat skóre uložené v paměti EEPROM.

Ve druhé bonusové úloze si studenti vytvořením vlastní verze hry Simon Says zopakují použití funkcí knihovny pro ovládání TFT LCD displeje a práci s vestavěným piezoměničem na přípravku Arduino Esplora. Dále si studenti zopakují práci s pamětí EEPROM (zápis a čtení dat).

2.7.3 Bonusová úloha 3

Ve třetí bonusové úloze si studenti vyzkouší naprogramovat přípravek Arduino Esplora, tak aby emuloval počítačovou myš a klávesnici.

Následně uvádíme text zadání laboratorní úlohy v podobě, v jaké je na výukovém systému EDUX.

Bonusová úloha 3.1: Emulace myši a klávesnice

- Ovládejte pohyb kurzoru myši pomocí joysticku. Krátké stisknutí tlačítka joysticku odpovídá stisknutí levého tlačítka myši a dlouhé stisknutí (2 s) tlačítka joysticku odpovídá stisknutí pravého tlačítka myši.
- Naprogramujte stisknutí kláves Alt + Tab (tj. klávesovou zkratku, která způsobí přepnutí do naposledy zobrazeného okna) po zmáčknutí tlačítka SWITCH 1. Dále si vyzkoušejte stisknutí kláves Enter, Esc a šipky vpravo.
- Po připojení Arduina k počítači odešlete stisknutí kláves Win + R (tj. klávesovou zkratku, která otevře okno Spustit program), vypište odkaz na stránky http://www.arduino.org/software#ide a potvrďte odesláním kódu klávesy Enter. Program způsobí, že se sám spustí prohlížeč s adresou pro stažení Arduino IDE.

Při vypracování úlohy postupujte následovně:

- Na začátku programu zavolejte funkci `Mouse.begin()` a `Keyboard.begin()`.

- Ve smyčce opakovaně čtěte hodnoty souřadnic x a y joysticku a získaná čísla namapujte pomocí funkce map na hodnoty kurzoru. Kurzor je možné ovládat funkcí `Mouse.move()`. Tato funkce má tři parametry, první dva slouží k určení souřadnic kurzoru a třetí k ovládání kolečka myši. Souřadnice předané této funkci nejsou absolutní, ale relativní, tj. udávají o kolik se změní vůči aktuální poloze kurzoru. Např. příkaz `Mouse.move (9, -2, 0)` tedy posune kurzor o 9 doprava a o 2 nahoru. Osa x leží na horní hraně obrazovky, osa y na levém okraji a počátek os je v levém horním rohu.
- Funkcemi `Mouse.press()` a `Mouse.release()`, případně `Mouse.click()` ovládejte kliknutí myši. Parametrem těchto příkazů je konkrétní tlačítko (MOUSE LEFT, MOUSE RIGHT, MOUSE MIDDLE). Pokud neuvedete žádný parametr, je defaultně vybráno levé tlačítko.
- Protože hodnoty souřadnic joysticku mohou v klidovém stavu v poloze uprostřed lehce kolísat, je třeba vytvořit tzv. mrtvou zónu („deadzone“) kolem tohoto bodu. Znamená to, že v nějakém malém rozpětí okolo středu joysticku nebudete kurzorem hýbat, i když dojde ke změně souřadnic v rámci tohoto rozsahu.
- Funkce `Keyboard.press()` slouží ke stisknutí klávesy. K uvolnění dané klávesy využijte funkce `Keyboard.release()`, případně `Keyboard.releaseAll()`, která uvolní všechny aktuálně stisknuté klávesy.
- Pro výpis textu můžete použít `Keyboard.print()` a `Keyboard.println()`.

Bonusová úloha 3.2: Zobrazení a skrytí plochy PC

Tato úloha je bonusová. Pomocí světelného senzoru ovládejte zobrazení a skrytí plochy. Pokud senzor detekuje tmu (tj. bude zakryt), přepněte se příslušnou klávesovou zkratkou (Win + D) na plochu. Ve chvíli, kdy bude senzor znovu na světle, skryjte plochu opětovným stisknutím kláves Win + D.

Poznámka Pro stisknutí funkčních a speciálních kláves využijete předdefinované konstanty. Jejich názvy (a číselné hodnoty) naleznete v oficiální dokumentaci platformy Arduino (viz [21]).

Přípravky Arduino založené na mikroprocesoru ATmega32u4 (Arduino Esplora) je možné naprogramovat tak, aby emulovaly myš a klávesnici na PC. Ve třetí bonusové úloze si studenti vyzkouší práci s funkcemi knihoven Mouse a Keyboard, které umožňují například snadné ovládání pohybu kurzoru myši

nebo stisknutí tlačítek na klávesnici.

Závěr

V práci jsem se zabývala vytvořením materiálů pro nový bakalářský předmět Interaktivní aplikace s Arduinem. Cílem práce bylo zvolit vhodné studijní okruhy k výuce studentů, kteří nemají předchozí zkušenosti s programováním vestavných systémů a dále pak dle těchto vybraných témat vytvořit zadání laboratorních úloh.

Při výběru vhodných tématických okruhů cvičení jsem se zaměřila na vlastnosti přípravků použitých k výuce, Arduino Esplora a Arduino Uno.

První tématický okruh se tedy věnuje základům programování pro přípravky Arduino – představuje základní strukturu Arduino projektu. Na toto téma navazuje první laboratorní úloha Seznámení s Arduinem.

Jako druhý studijní okruh byla zvolena sériová komunikace, protože přípravky Arduino ji standardně využívají k přenosu dat. V souvislosti se sériovou komunikací je představena i fyzická implementace principu sériové komunikace, rozhraní UART. Na toto téma navazuje druhá laboratorní úloha Komunikace s PC a ladění pomocí sériové linky.

Třetím tématickým okruhem byl vybrán konečný automat, protože jej studenti využijí při řešení úloh, ve kterých je nutné reagovat na více vyvolaných událostí (např. stisknutí (a puštění) více tlačítek na přípravku Arduino Esplora). Na toto téma navazuje druhá laboratorní úloha Komunikace s PC a ladění pomocí sériové linky.

Jelikož při používání přípravků Arduino budou studenti pracovat s analogovými a digitálními vstupy, čtvrtý tématický okruh se věnuje vysvětlení problematiky analogového a digitálního signálu. Na toto téma navazuje třetí laboratorní úloha Joystick a RGB dioda.

Pátý tématický okruh se věnuje popisu TFT LCD displeje přípravku Arduino Esplora a také rozhraní SPI, které je použito ke komunikaci mezi mikrokontrolérem a displejem. Na toto téma navazuje čtvrtá laboratorní úloha Displej a senzory.

Šestý tématický okruh obsahuje úvod do problematiky perzistentních pamětí EEPROM a SD karty, které jsou součástí přípravku Arduino Esplora

(resp. SD karta je součástí připojeného TFT LCD displeje). Na toto téma navazuje pátá laboratorní úloha SD karta.

Závěrečný sedmý tématický okruh vysvětluje princip přerušení na mikroprocesoru přípravku Arduino Esplora. Na toto téma navazuje šestá laboratorní úloha Návrhářské přístupy pro složitější aplikace.

V rámci této práce vznikly podklady pro celkem šest základních a tři bonusová laboratorní cvičení. Tyto materiály obsahují jak stručný úvod do problematiky zvolených studijních okruhů, tak i jednotlivá zadání programovacích úloh, které vybrané okruhy procvičují. Součástí studijních podkladů jsou i šablony zdrojových kódů prvních čtyř laboratorních úloh.

Na přiloženém CD jsou poté k dispozici podklady pro laboratorní cvičení ve formě použitelné pro výukový systém EDUX a dále pak také vzorová řešení jednotlivých úloh pro potřeby vyučujících.

Vytvořené laboratorní úlohy byly během letního semetru akademického roku 2015/2016 úspěšně použity ve výuce.

Literatura

[1] Arduino Esplora. [online], [cit. 2017-03-10]. Dostupné z: https://store.arduino.cc/arduino-esplora

[2] Arduino Uno. [online], [cit. 2017-03-10]. Dostupné z: https://store.arduino.cc/arduino-uno-rev3

[3] Atmel Corporation: *ATmega16U4-32U4 Datasheet*. [online], 2015, [cit. 2017-03-03]. Dostupné z: http://www.atmel.com/Images/Atmel-7766-8-bit-AVR-ATmega16U4-32U4_Datasheet.pdf

[4] Hirzel, T.: Arduino PWM. [online], [cit. 2017-04-29]. Dostupné z: https://www.arduino.cc/en/Tutorial/PWM

[5] Arduino LCD Screen. [online], [cit. 2017-04-15]. Dostupné z: https://store.arduino.cc/arduino-lcd-screen

[6] Skrbek, M.: Vestavné systémy: Časové průběhy SPI. [online], 2011, [cit. 2017-05-10]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-VES/_media/lectures/bives_pred_05.pdf

[7] Arduino Products. [online], [cit. 2017-03-03]. Dostupné z: https://www.arduino.cc/en/Main/Products

[8] Baichtal, J.: *Arduino for beginners: essential skills every maker needs*. Indianapolis: Que, 2014, ISBN 9780789748836.

[9] Arduino Esplora Library. [online], [cit. 2017-03-10]. Dostupné z: https://www.arduino.cc/en/Reference/EsploraLibrary

[10] Atmel Corporation: *ATmega328P Datasheet*. [online], 2016, [cit. 2017-03-03]. Dostupné z: http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[11] Kubátová, H.: Struktura a architektura počítačů: Paměti. [online], 2017, [cit. 2017-04-21]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-SAP/_media/lectures/10/sap-10-2-pameti.pdf

[12] Arduino Finite State Machine Library. [online], 2015. Dostupné z: https://playground.arduino.cc/Code/FiniteStateMachine

[13] Analog vs. Digital Signal. [online], [cit. 2017-04-29]. Dostupné z: https://learn.sparkfun.com/tutorials/analog-vs-digital

[14] Kubátová, H.: Struktura a architektura počítačů: ISA, podprogramy, přerušení. [online], 2017, [cit. 2017-04-21]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-SAP/_media/lectures/09/sap-9-podprogramy.pdf

[15] Arduino Serial Library. [online], [cit. 2017-04-15]. Dostupné z: https://www.arduino.cc/en/reference/serial

[16] Arduino Map Function. [online], [cit. 2017-03-30]. Dostupné z: https://www.arduino.cc/en/Reference/Map

[17] Arduino TFT Library. [online], [cit. 2017-04-10]. Dostupné z: https://www.arduino.cc/en/Reference/TFTLibrary

[18] Arduino SD Library. [online], [cit. 2017-04-10]. Dostupné z: https://www.arduino.cc/en/Reference/SD

[19] Joystick Shield Quickstart Guide. [online], 2010, [cit. 2017-04-10]. Dostupné z: https://www.sparkfun.com/tutorials/171

[20] Denver, A.: Microsoft Serial Communications. [online], 1995, [cit. 2017-04-10]. Dostupné z: https://msdn.microsoft.com/en-us/library/ff802693.aspx

[21] Arduino Keyboard Modifiers. [online], [cit. 2017-04-21]. Dostupné z: https://www.arduino.cc/en/Reference/KeyboardModifiers

Příloha **A**

Šablony zdrojových kódů laboratorních úloh

V této příloze se nachází šablony zdrojových kódů čtyř laboratorních úloh. Studenti naleznou tyto šablony již zakomponované do zadání na výukovém systému EDUX.

A.1 Úloha 1: Seznámení s Arduinem

A.1.1 Morseova abeceda - rozblikání RGB LED

```
1  #include <Esplora.h>
2
3  // doba bliknuti tecky (v~ms)
4  #define DOT 200
5   // doba bliknuti carky (v~ms)
6  #define DASH 800
7  // doba vypnuti LED mezi jednotlivymi  bliknutimi
8  #define NO_LIGHT 200
9   // intenzita jasu zelene LED
10 #define BRIGHT_GREEN 50
11
12 void flash(int duration){
13   // parametr duration urcuje dobu, kdy je LED
14   // zapnuta
15 }
```

A.2 Úloha 2: Komunikace s PC a ladění pomocí sériové linky

A.2.1 Výpis tlačítek na sériový monitor

```
1  #include <Esplora.h>
2
3  enum states {
4  STATE_SW1, STATE_SW2, STATE_SW1SW2
5  };
6  enum states STATE, NEXT_STATE;
7
8  void setup() {
9    // pocatecni inicializace
10 }
11
12 void loop() {
13   switch (STATE) {
14
15         case STATE_SW1:
16           if(sw1_released == true) {
17                 Akce_A();
18                 NEXT_STATE = STATE_SW1;
19           } else if(sw2_released == true) {
20                 Akce_B();
21                 NEXT_STATE = STATE_SW2;
22           }
23
24         // doplnte dalsi stavy
25   }
26   STATE = NEXT_STATE;
27 }
```

A.3 Úloha 3: Joystick a RGB dioda

A.3.1 Joystick - digitální přístup

```
1  #include <Esplora.h>
2
3   // zde nastavte hranici pro pozici joysticku nahore
4  #define UP_TRESHOLD
5  //  zde nastavte hranici pro pozici joysticku dole
6  #define DOWN_TRESHOLD
7   // zde nastavte hranici pro pozici joysticku vpravo
```

```
8  #define RIGHT_TRESHOLD
9   // zde nastavte hranici pro pozici joysticku vlevo
10 #define LEFT_TRESHOLD
11
12 #define CENTER 0
13 #define LEFT 1
14 #define UP 2
15 #define RIGHT 3
16 #define DOWN 4
17 #define PUSH 5
18
19 // funkce pro cteni pozice joysticku
20 int readJoystickKey() {
21         // joystick je stisknuty
22         if (button == LOW) {
23                 return PUSH;
24         }
25
26         // joystick je nahore
27         if (yPos > UP_TRESHOLD) {
28                 return UP;
29         }
30
31         // joystick je dole
32         if (yPos < DOWN_TRESHOLD) {
33                 return DOWN;
34         }
35
36         // joystick je vpravo
37         if (xPos > RIGHT_TRESHOLD) {
38                 return RIGHT;
39         }
40
41         // joystick je vlevo
42         if (xPos < LEFT_TRESHOLD) {
43                 return LEFT;
44         }
45
46         // joystick je uprostred
47         return CENTER;
48 }
49
50 void loop() {
51
```

```
52          // doplnte vlastni kod
53
54          switch (pos_joystick) {
55                  case LEFT:
56                  case RIGHT:
57                  case UP:
58                  case DOWN:
59                  case CENTER:
60                  case PUSH:
61          }
62  }
```

A.3.2 Hra Bum do krtka

```
1   void loop() {
2     // je treba, aby barvy blikaly nahodne:
3     int num = (random(MAX_NUM) % 4);
4
5     switch(num) {
6           case 0:
7                   // zde rozsvitte cervenou RGB LED
8                   push_button(SWITCH_1, "red");
9                   break;
10
11          case 1:
12                  // zde rozsvitte zelenou RGB LED
13                  push_button(SWITCH_2, "green");
14                  break;
15
16          case 2:
17                  // zde rozsvitte modrou RGB LED
18                  push_button(SWITCH_3, "blue");
19                  break;
20
21          case 3:
22                  // zde rozsvitte zlutou RGB LED
23                  push_button(SWITCH_4, "yellow");
24   }
25
26   if(na RGB LED jiz probliklo 30 barev) {
27          // zde vypiste pocet spravne stisknutych
28          // tlacitek
29
30     if(spravne zmacknuto vice nez polovina tlacitek) {
```

```
31              // zde vypiste na monitor * Winner! *
32     } else {
33              // zde vypiste na monitor * Loser! *
34     }
35  }
36 }
37
38 void push_button( cislo tlacitka, nazev barvy ) {
39     // velikost MAX_NUM urcuje dobu rozsviceni LED
40     for(int i=0; i < MAX_NUM; i++) {
41
42         if( je stisknute spravne tlacitko ) {
43              // zde vypiste na seriovy monitor jmeno
                    rozsvicene barvy
44           }
45     }
46 }
```

A.4 Úloha 4: Displej a senzory

A.4.1 Zobrazení výstupu senzorů na displej

```
1  enum states {
2    HOME_SEL_TEMP, HOME_SEL_MIC, HOME_SEL_ACCE, TEMP,
       MIC, ACCE
3  };
4  enum states STATE, NEXT_STATE;
5
6  void setup() {
7    // zde provedte inicializaci displeje
8
9    // zobrazte menu:
10   display_menu();
11   STATE = HOME_SEL_TEMP;
12 }
13
14 void loop() {
15
16   switch (STATE) {
17
18   // menu - temperature
19   case HOME_SEL_TEMP:
20
21       if (buttonEvent(DOWN)) {
```

```
22          // je stisknuto tlacitko 'pohyb dolu'
23          // zmena menu:
24          change_position();
25          NEXT_STATE = HOME_SEL_MIC;
26        }
27        else if (buttonEvent(ENTER)) {
28          // je stisknuto tlacitko 'vybrat senzor'
29          NEXT_STATE = TEMP;
30         }
31         break;
32
33   // menu - microphone
34   case HOME_SEL_MIC:
35
36        if (buttonEvent(DOWN)) {
37          // je stisknuto tlacitko 'pohyb dolu'
38          // zmena menu
39           change_position();
40           NEXT_STATE = HOME_SEL_ACCE;
41        }
42        else if (buttonEvent(UP))
43        {
44          // je stisknuto tlacitko 'pohyb nahoru'
45          // zmena menu
46           change_position();
47           NEXT_STATE = HOME_SEL_TEMP;
48         }
49         else if (buttonEvent(ENTER))
50         {
51            // je stisknuto tlacitko 'vybrat senzor'
52           NEXT_STATE = MIC;
53          }
54          break;
55
56   // menu - accelerometer
57   case HOME_SEL_ACCE:
58
59        if (buttonEvent(UP)) {
60             // je stisknuto tlacitko 'pohyb nahoru'
61             // zmena menu
62             change_position();
63             NEXT_STATE = HOME_SEL_MIC;
64        }
65        else if (buttonEvent(ENTER)) {
```

```
66              // je stisknuto tlacitko 'vybrat senzor'
67             NEXT_STATE = ACCE;
68           }
69           break;
70
71     // draw temperature
72     case TEMP:
73
74        if (buttonEvent(BACK)) {
75              // je stisknuto tlacitko 'zpet do menu'
76              // zobrazeni menu
77              display_menu();
78              NEXT_STATE = HOME_SEL_TEMP;
79        }
80        else {
81               // funkce draw_temp vykresli aktualni
82               // teplotu kazde 2 sekundy
83               draw_temp();
84        }
85        break;
86
87     // draw microphone
88     case MIC:
89
90          if (buttonEvent(BACK)) {
91                // je stisknuto tlacitko 'zpet do menu'
92                // zobrazeni menu
93               display_menu();
94               NEXT_STATE = HOME_SEL_MIC;
95          }
96          else {
97                // funkce draw_mic vykresli krivku podle
98                // aktualniho zvuku z~mikrofonu
99                draw_mic();
100         }
101         break;
102
103    // draw accelerometer
104    case ACCE:
105
106        if(buttonEvent(BACK)) {
107             // je stisknuto tlacitko 'zpet do menu'
108             // zobrazeni menu
109             display_menu();
```

```
110             NEXT_STATE = HOME_SEL_ACCE;
111         }
112         else {
113             // funkce draw_acce zobrazi kolecko,
114             // ktere se bude pohybovat v~zavislosti na
115             // naklonu Esplory
116             draw_acce();
117         }
118         break;
119  }
120  STATE = NEXT_STATE;
121 }
```

A.4.2 Detekce stisknutí (a puštění) tlačítka

```
1  #define DOWN  SWITCH_1
2  #define UP    SWITCH_4
3  #define ENTER SWITCH_2
4  #define BACK  SWITCH_3
5
6  byte buttonFlag = 0;
7
8  bool buttonEvent(int button) {
9
10   switch(button) {
11
12     case UP:
13       if (Esplora.readButton(UP) == LOW) {
14         buttonFlag |= 1;
15       }
16       else if (buttonFlag & 1) {
17         buttonFlag ^= 1;
18         return true;
19       }
20       break;
21
22     case DOWN:
23      if (Esplora.readButton(DOWN) == LOW) {
24        buttonFlag |= 2;
25      }
26      else if (buttonFlag & 2) {
27        buttonFlag ^= 2;
28        return true;
29      }
```

```
30       break;
31
32      case BACK:
33       if (Esplora.readButton(BACK) == LOW) {
34         buttonFlag |= 4;
35       }
36       else if (buttonFlag & 4) {
37         buttonFlag ^= 4;
38         return true;
39       }
40       break;
41
42      case ENTER:
43       if (Esplora.readButton(ENTER) == LOW) {
44         buttonFlag |= 8;
45       }
46       else if (buttonFlag & 8) {
47         buttonFlag ^= 8;
48         return true;
49       }
50    }
51     return false;
52  }
```

Příloha B

Seznam použitých zkratek

EEPROM Electrically Erasable Programmable Read-Only Memory

IDE Integrated Development Environment

PWM Pulse width modulation

RGB LED Red-Green-Blue Light-Emitting Diode

SPI Serial Peripheral Interface

SRAM Static Random Access Memory

TFT LCD Thin-Film Transistor Liquid Crystal Display

USART Universal Synchronous/Asynchronous Receiver and Transmitter

Příloha C

Obsah přiloženého CD

- `src`
  - `BP_Hánová_Gabriela_2017.tex` ..... zdrojová forma práce ve formátu LaTeX
  - `impl` ..... zdrojové kódy laboratorních úloh
    - `uloha1` ..... 1. úloha
    - `uloha2` ..... 2. úloha
    - `uloha3` ..... 3. úloha
    - `uloha4` ..... 4. úloha
    - `uloha5` ..... 5. úloha
    - `uloha6` ..... 6. úloha
    - `bonusova_uloha1` ..... 1. bonusová úloha
    - `bonusova_uloha2` ..... 2. bonusová úloha
    - `bonusova_uloha3` ..... 3. bonusová úloha
- `text` ..... text práce
  - `BP_Hánová_Gabriela_2017.pdf` ..... text práce ve formátu PDF
  - `zadani` ... zadání úloh ve formě použitelné pro výukový systém EDUX
    - `uloha1.txt` ..... 1. úloha
    - `uloha2.txt` ..... 2. úloha
    - `uloha3.txt` ..... 3. úloha
    - `uloha4.txt` ..... 4. úloha
    - `uloha5.txt` ..... 5. úloha
    - `uloha6.txt` ..... 6. úloha
    - `bonusova_uloha1.txt` ..... 1. bonusová úloha
    - `bonusova_uloha2.txt` ..... 2. bonusová úloha
    - `bonusova_uloha3.txt` ..... 3. bonusová úloha