ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

REALIZACE ELEKTRONICKÝCH ZAŘÍZENÍ NA BÁZI PLATFORMY ARDUINO

BAKALÁŘSKÁ PRÁCE

Pavel Eschler

Informatika se zaměřením na vzdělávání

Vedoucí práce: Mgr. Denis Mainz

Plzeň, 2016

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 10. dubna 2016

.....

vlastnoruční podpis

Poděkování

Děkuji za odbornou pomoc a řadu cenných podnětů vedoucímu mé bakalářské práce panu Mgr. Denisi Mainzovi.

ZDE SE NACHÁZÍ ORIGINÁL ZADÁNÍ KVALIFIKAČNÍ PRÁCE.

OBSAH

SEZNAM POJMŮ

Software - programové vybavení

Pin - anglický výraz pro svorku

Java - programovací jazyk

Processing - programovací jazyk pro realizaci fyzických projektů

ZIP - souborový formát určený ke kompresi a archivaci dat

schránka - systémové dočasné úložiště informací

RAW - označení syrových (nezpracovaných) dat ze snímače

TTL - označení pro tranzistorově-tranzistorovou logiku

CPU - centrální procesorová jednotka, vykonává strojové instrukce, obsluhuje vstupy a výstupy

ÚVOD

Téma bakalářské práce bylo zvoleno s ohledem na výzvu a možnosti, které platforma Arduino představuje. Před začátkem psaní této práce jsem neměl žádnou zkušenost s touto platformou nebo jinými druhy programovatelných obvodů a ani mé znalosti programovacích jazyků nebyly na dobré úrovni.

Práce spojuje dva hlavní předpoklady pro realizaci a ovládání fyzických projektů, a to část programovou a hardwarovou. Platforma Arduino je v dnešní době velmi populární, především díky open-source technologii a zároveň díky tomu, že celá platforma je uživatelsky přívětivá a za předpokladu použití netechnickým uživatelem i bezpečná.

Tvorba vlastních fyzických projektů v domácím prostředí je velmi pohodlná a má mnohdy významný dopad na vědu a techniku po celém světe. Komunita okolo platformy Arduino je v dnešní době velmi rozšířená na celém světě. Uživatelé si mezi sebou vyměňují poznatky a nápady zcela zdarma.

Bakalářská práce je rozdělena do dvou hlavních kapitol. První kapitola se zabývá hardwarem, rozšířeními a programovým vybavením platformy Arduino a možnými alternativami k této platformě. Druhá kapitola je zaměřena na praktické využití platformy při realizaci vybraných projektů.

Cílem bakalářské práce je shrnout poznatky dosažené v jednotlivých kapitolách a vytvořit díky nim komplexní úlohu simulující hudební nástroj. Práce má zároveň za cíl popularizovat ve školství využití programování za pomoci fyzického výstupu, nejen u technických oborů ale i netechnických jako je například hudební výchova.

1 TEORETICKÁ ČÁST

1.1 PŘEDSTAVENÍ PROJEKTU ARDUINO

Arduino je open-source (otevřený-zdroj - znamená technickou legální dostupnost pro prohlížení a úpravu) projekt vytvořený skupinou autorů, kde hlavními představiteli jsou Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino a David Mellis. [1]

Hlavním impulsem pro vznik projektu Arduino, který vznikl v roce 2005 v italském městě Ivrea, byla snaha o vytvoření jednoduchého a především levného vývojového setu pro studenty, který měl být alternativou stávajících drahých desek BASIC Stamp. V roce 2006 byla cena desky BASIC Stamp téměř čtyřikrát vyšší než deska Arduino.

[2, s. 8][3, s. 2]

Projekt Arduino byl navržen tak, aby byl snadno použitelný pro začátečníky, kteří nemají žádné elektronické nebo programátorské zkušenosti. S Arduinem může uživatel vytvářet objekty, které mohou ovládat nebo reagovat pomocí připojení elementárních elektronických součástek na světlo, zvuk, dotyk, a pohyb, případně jiné fyzikální veličiny za pomocní rozšiřujících modulů. Arduino bylo použito k realizaci mnoha zajímavých myšlenek, zahrnujících například vytvoření hudebních nástrojů, robotů, světelných soch, her, interaktivního nábytku, a dokonce i interaktivního oděvu. [3, s. 1-2]

Arduino se používá v mnoha vzdělávacích programech po celém světě, zejména návrháři a umělci, kteří chtějí snadno vytvářet prototypy, ale nepotřebují hluboké porozumění o technických detailech jejich výtvorů. Jelikož bylo vytvořeno pro použití netechnickým uživatelem, obsahuje software mnoho příkladů kódu, na kterých je demonstrováno jak pracovat s deskou Arduino. [4, s. 1]

Název Arduino je známý díky své hardwarové konstrukci, může se tedy zdát, že software je opomíjen, opak je ale pravdou, jelikož pro fungování hardwaru je softwarové řízení nezbytné. Hardware i software se shodně nazývají "Arduino." Kombinace umožňuje vytvořit projekty, které snímají a ovládají fyzický svět. Řídící software Arduino IDE je zdarma, open-source a multiplatformní (Windows OS, Linux OS a Macintosh OS). Řídící desky (boardy) se dají levně zakoupit, orientační cena za Arduino Uno rev. 3 v roce 2016 je 25$ - přibližně 625kč (http://store-usa.arduino.cc/products/a000066). Jelikož i hardwarový design (sestavení boardu) je open-source, může si desku kdokoliv sestavit

za cenu jednotlivých součástek. Podpora Arduina je tvořena celosvětovou komunitou prostřednictvím fór a wiki. Fóra a wiki nabízí příklady projektů a řešení problémů, se kterými se uživatelé v průběhu vývoje setkávají. [4, s. 1-2]

1.2 ARDUINO IDE

Software Arduino IDE je druh integrovaného vývojového prostředí (IDE - Integrated Development Environment) napsaného v jazyce JAVA. Jedná se o speciální program běžící na počítači, který umožňuje psát, testovat a nahrávat programy do desek Arduino. Kód napsaný v prostředí IDE se označuje jako sketch (návrh). Arduino IDE vzniklo z výukového prostředí Processing mírnou úpravou a přidáním podpory Wiring. Software IDE napsaný kód před nahráním do desky nejprve přeloží do jazyka C, přeložený kód je předán dále avr-gcc kompilátoru, který provede finální překlad do kódu, kterému mikrokontrolér rozumí. Tento poslední krok je velmi důležitý, protože odstraňuje veškeré složitosti programování mikrokontrolérů. Arduino IDE je k dispozici pro platformy Windows OS, Macintosh OS a Linux OS. [2, s. 33][3, s. 17] [4, s. 2]

1.2.1 INSTALACE NA WINDOWS OS

Tento návod popisuje instalaci softwaru a ovladačů pro Arduino Uno na platformě Windows 7, návod je stejný i pro Windows XP a Windows Vista. Návod neřeší možné problémy v systému Windows 8, který v současné době vyžaduje několik různých řešení problému s instalací ovladačů k desce. Tato řešení lze najít v diskuzi na fóru Arduino pod názvem "Chybějící digitální podpis ovladače na Windows 8" (http://forum.arduino.cc/index.php?topic=94651.15). [2, s. 34-36]

1) Na stránkách pro stažení (https://www.arduino.cc/en/Main/Software), je vhodné vybrat ze seznamu stažení neinstalační verzi souborů v archivu ZIP pro Windows. Jakmile je stažení dokončeno, je nutné archiv rozbalit, a obsah umístit do libovolné složky na disku, např. "C:/Program Files/Arduino/".

2) Připojíme desku Arduino k počítači pomocí USB kabelu. Jakmile je deska připojena, rozsvítí se zelená LED dioda s označením "ON", která signalizuje, že je deska napájena. Systém Windows se následně pokusí nalézt ovladač zařízení. Toto hledání ve většině případu skončí neúspěšně. Průvodce ručním nalezením ovladače je vhodné ukončit a pokračovat podle následujících bodů.

3) V nabídce Start zadáme do pole pro hledání programů a souborů příkaz "devmgmt.msc", po nalezení potvrdíme výběr pomocí Enter. Otevře se okno Správce zařízení. Správce zařízení zobrazuje hardware počítače a připojené periferie, v tomto seznamu se zobrazuje Arduino Uno jako neznámé zařízení.

4) Kliknutím pravým tlačítkem myši na Arduino Uno se rozbalí seznam, ze seznamu vybíráme možnost "Aktualizovat software ovladače". V otevřeném okně vybereme možnost "Vyhledat ovladač v počítači".

5) Klikneme na procházet a najdeme složku s rozbalenými soubory, které jsme umístily do složky v prvním kroku tohoto návodu.

6) Otevřeme složku Drivers a z ní vybereme soubor Arduino Uno.

7) Klikneme na pokračovat, Windows dokončí instalaci ovladače.

8) Pro otevření softwaru Arduino IDE spustíme soubor "Arduino.exe", který se nachází ve složce s rozbalenými soubory.

1.2.2 INSTALACE NA MAC OS X LION

Tento návod popisuje instalaci softwaru a ovladačů pro Arduino Uno na platformě Mac OS X Lion, návod je stejný i pro verze Leopard, Snow Leopard a Muntain Lion. Předchozí verze tohoto operačního systému mohou vyžadovat různé postupy, které je nutné dohledat na fórech Arduino. [2, s. 37-38]

1) Otevřeme stránku pro stažení (https://www.arduino.cc/en/Main/Software), a vybereme ze seznamu stažení neinstalační verze souborů v archivu ZIP pro Mac OS X 10.7 Lion nebo novější. Jakmile je stažení dokončeno, přesuneme soubor do složky aplikací.

2) Připojíme desku Arduino k počítači pomocí USB kabelu. Jakmile je deska připojena, rozsvítí se zelená LED dioda s označením "ON", která signalizuje, že je deska napájena. Po připojení se zobrazí dialogové okno, které informuje že bylo připojeno nové síťové rozhraní.

3) Klikneme na tlačítko "Předvolby sítě" a v okně, které se objeví, klikneme na "Aplikovat". V seznamu na levé straně okna se zobrazí informace o tom,

že Arduino není nakonfigurováno, tato informace je chybná, software a deska v tuto chvíli již bude fungovat.

4) Okno Předvoleb sítě zavřeme.

5) Spouštěcí soubor Arduino se nachází ve složce Aplikace.

1.2.3 INSTALACE NA LINUX

Instalace softwaru Arduino na Linux je velmi proměnlivá a záleží vždy na konkrétní distribuci systému. Podrobnosti k instalaci jsou dostupné na adrese: "http://arduino.cc/playground/Learning/Linux". [1][2, s. 39]

1) Otevřeme stránku pro stažení (https://www.arduino.cc/en/Main/Software), a vybereme ze seznamu stažení instalační balíček pro 32bit nebo 64bit verzi Linux.

2) Jakmile je stažení dokončeno, extrahujeme instalační balíček z archivu.

3) Otevřeme extrahovanou složku Arduino a najdeme soubor "install.sh". Následně na tento soubor pravým tlačítkem klikneme a zvolíme možnost spuštění v terminálu z kontextového menu. Proces instalace proběhne rychle a na ploše se zobrazí nová ikona.

4) Připojíme desku Arduino k počítači pomocí USB kabelu. Jakmile je deska připojena, rozsvítí se zelená LED dioda s označením "ON", která signalizuje, že je deska napájena.

5) Spouštěcí soubor aplikace je umístěn na ploše.

1.2.4 GRAFICKÉ ROZHRANÍ

Grafické rozhraní softwaru Arduino IDE (Obrázek 2) obsahuje základní panel nástrojů, s jehož podobou se setkáváme v různých programech pro Windows OS. Panel nástrojů Arduino se skládá z rozbalovacích menu, uspořádaných dle významu funkcí do menu:

<u>Soubor</u>

- Nový - vytvoří novou instanci editoru se základní strukturou sketche.
- Otevřít - umožňuje načíst existující sketch z pevného disku počítače.

- Otevřít Předešlé - rozbalí krátký seznam naposledy otevřených sketchí pro jejich rychlé načtení.
- Projekty - zobrazuje aktuální sketche v rámci souborové struktury sketchí Arduino.
- Příklady - Zobrazí příklady projektů, které daná verze Arduino IDE poskytuje. V příkladech se také zobrazují importované knihovny příkladů. Příklady jsou řazeny formou stromové struktury, která poskytuje rychlý přístup podle zvoleného tématu. Příklady jsou přístupné pouze pro čtení, v případě editace je nutné změněný kód uložit do jiné složky.
- Zavřít - ukončí aktuální instanci programu IDE.
- Uložit - uloží sketch se současným názvem a do současného umístění.
- Uložit jako - umožňuje uložit sketch do nového umístění a s odlišným názvem.
- Nastavení stránky - zobrazí možnosti nastavení pro tisk s náhledem stránky.
- Tisk - odešle sketch k tisku dle předdefinovaných parametrů v Nastavení stránky.
- Vlastnosti - otevře okno předvoleb rozhraní Arduino IDE, kde je možné přizpůsobit některá nastavení IDE, např. jazyk, velikost fontu editoru, síť.
- Ukončit - Ukončí všechny instance IDE. Při dalším spuštění tyto instance automaticky všechny otevře.

Úpravy

- Zpět - vrátí jeden nebo více kroků, které byly učiněny při úpravách sketche.
- Znovu - umožňuje při využití funkce Zpět jít dopředu.
- Vyjmout - odebere vybraný text ze sketche a umístí ho do schránky.
- Kopírovat - duplikuje vybraný text ze sketche do schránky
- Kopírovat pro užití ve fóru - zkopíruje celý kód sketche do schránky ve formě vhodné pro publikování na fóru, včetně zbarvení syntaxí.
- Kopírovat jako HTML - umístí celý kód sketche do schránky ve formě HTML kódu, který je vhodný pro publikování na webových stránkách.
- Vložit - vloží obsah schránky na pozici kurzoru v editoru.

- Vybrat vše - vybere a označí celý obsah editoru.
- Go to line - přejde na začátek zvoleného řádku.
- Zakomentovat / Odkomentovat - vloží nebo odebere označení komentáře(//) na začátku každého vybraného řádku. Funkce umí pouze zakomentovat nebo odkomentovat celý řádek, nikoliv jeho část.
- Zvětšit odsazení / Zmenšit odsazení - přidá nebo odebere odsazení textu na vybraném řádku (funkce tabulátoru), popřípadě odstraní mezery před textem.
- Najít - otevře okno s funkcí najdi a nahraď, které umožňuje najít a nahradit text v aktuální sketchi.
- Najít další - vyhledá další výskyt hledaného textu. Funguje pouze v případě, že byla použita funkce Najít.
- Najít předchozí - vyhledá předchozí výskyt hledaného textu. Funguje pouze v případě, že byla použita funkce Najít.

Projekt

- Kontrola/Kompilace - zkontroluje výskyt chyb ve sketchi, pokud je sketch v pořádku, zkompiluje ji a informuje o využití paměti v oblasti Konzole. V případě výskytu chyb se kompilace neprovede a program v oblasti Konzole informuje o chybách, včetně popisu chyb.
- Nahrát - zkontroluje, zkompiluje a nahraje binární soubor do zvolené desky pomocí přednastaveného portu.
- Nahrát pomocí programátoru - dojde k přepsání zavaděče na desce, před dalším použitím je nutné obnovit zavaděč pomocí funkce Vypálit zavaděč. Tato funkce umožňuje využít plnou kapacitu flash paměti pro sketch. Použitím této funkce nedojde ke zničení desky.
- Export kompilovaného binaru - uloží soubor .hex, který může sloužit jako záloha, nebo může být do paměti desky nahrán pomocí jiných nástrojů.
- Zobraz adresář s projekty - otevře adresář pomocí exploreru systému, ve kterém je aktuální sketch uložena.

- Přidat knihovnu - přidá na začátek sketche knihovnu vložením příkazu #include. Zároveň umožňuje spravovat knihovny a importovat nové ze ZIP souboru.
- Přidat soubor - přidá zdrojový soubor do sketche. Nový soubor se objeví v nové záložce v okně sketche.

Nástroje

- Automatické formátování - upraví formátování kódu, např. odsazení řádků, umístění počáteční a koncové složené závorky. Význam této funkce je výhradně estetický.
- Archivuj projekt - archivuje kopii aktuální sketche do ZIP souboru. Archiv je umístěn ve stejném složce jako sketch.
- Uprav kódování a znovu nahraj - opraví případné rozdíly mezi kódovací tabulkou editoru a kódovací tabulkou operačního systému.
- Sériový monitor - otevře sériový monitor a iniciuje výměnu dat s připojenou deskou prostřednictvím přednastaveného sériového portu. Otevření obvykle resetuje desku, pokud deska podporuje reset pomocí otevření sériového portu. V otevřeném okně se zobrazují data vytištěná na monitor pomocí příkazu print, resp. println.
- Sériový Plotter - tato funkce je dostupná od verze Arduino IDE 1.6.6 . Umožňuje vizualizaci příchozích dat po sériovém portu. Zobrazuje raw hodnoty do grafu s osou X a Y. Osa Y se automaticky mění dle raw hodnoty. Osa X je fixně nastavena na 500bodů, kde jeden bod je reprezentován příkazem println. Výsledný graf je vytvořen spojením jednotlivých bodů, kde souřadnice bodu Y odpovídá raw hodnotě, a souřadnice bodu X odpovídá pořadí příkazu println. Ukázka zobrazení je na Obrázek 1, kde každý vrchol reprezentuje dva příkazy println a to println(0) a println(5).
- Vývojová deska - vyžaduje výběr desky, která bude použita pro nahrání sketche.
- Port - obsahuje všechna sériová (skutečná nebo virtuální) zařízení počítače.
- Programátor - umožňuje výběr programátoru pro programování desky nebo čipu, který nepoužívá integrovaný převodník USB-serial na desce.

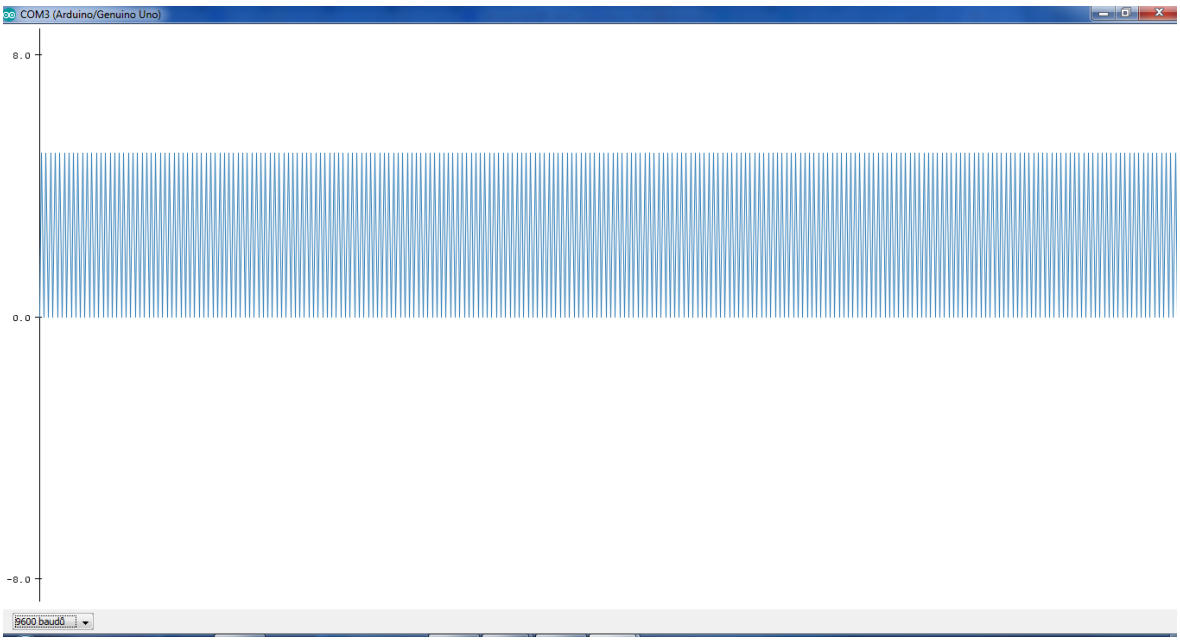
- Vypálit zavaděč - umožňuje vypálit zavaděč mikrokontroléru na desce Arduino. Pro běžné použití není tato funkce zapotřebí. Je využitelná ve chvíli, kdy desku osadíme novým mikrokontrolérem. Zavaděč umožňuje externí programování mikrokontroléru.

Nápověda

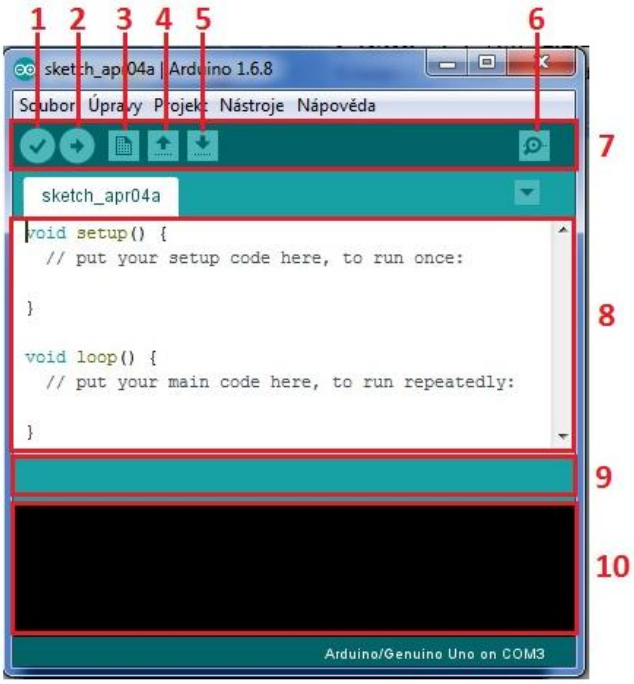
Obsahuje rychlý přístup k nápovědě související se softwarem Arduino IDE a programováním desek. Pro přístup k nápovědě není nutné připojení k internetu, veškerý obsah je umístěn ve složce programu na pevném disku.

Jedinou interaktivní funkcí nápovědy je funkce "Najdi v referenční příručce" , tato automaticky otevře tu část nápovědy, která se přímo týká funkce nebo příkazu označeného kurzorem v editoru kódu.

Arduino IDE (Obrázek 2) dále obsahuje panel nástrojů rychlých funkcí (7), textový editor kódu (8) a pole výstupu informací rozdělených do pole zpráv (9) a konzole (10). Oblast textového editoru (8) zobrazuje kód programu jako text. Je téměř identická s běžným textovým editorem. Text je barevně rozlišován podle toho, zda-li je softwarem rozpoznám. Software umožňuje také automatické formátování.



Obrázek 1: Výstup sériového plotteru(zdroj vlastní)



Obrázek 2: Grafické rozhraní (zdroj vlastní)

Vysvětlivky k číselnému značení na Obrázek 2

1 - Ověřit

2 - Nahrát

3 - Nový

4 - Otevřít

5 - Uložit

6 - Sériový monitor

7 - Panel Nástrojů - obsahuje nejpoužívanější funkce.

8 - Textový editor

9 - Oblast zpráv (informuje při ukládání a zobrazuje případné chyby)

10 - Konzole (zobrazuje textový výstup softwaru včetně kompletních chybových zpráv)

1.3 HARDWARE ARDUINO

1.3.1 DESKY

Desky platformy Arduino jsou založeny mikrokontrolérech ATmega a obsahují paměť pro program a pro dat. Hardwarové sestavení desek je open-source. Desky lze rozdělit do pěti kategorií:

1) Základní použití - Arduino desky s označením UNO, 101, PRO, PRO MINI, MICRO, NANO.

2) Pokročilé použití - Arduino desky s označením MEGA, ZERO, DUE.

3) Internetové použití - Arduino desky s označením YÚN, MKR1000.

4) Módní použití - Arduino desky s označením GEMMA, LILYPAD USB, LILYPAD MAIN BOARD, LILYPAD SIMPLE, LILYPAD SIMPLE SNAP.

5) Užití pro 3D tisk - MATERIA 101

Existují i starší typy desek Arduino s označením MEGA ADK, ETHERNET, ROBOT, LEONARDO, EXPLORA, MINI, FIO. Podrobné informace o deskách jsou dostupné na stránkách výrobce (http://www.arduino.cc/en/Main/Products).

1.3.2 ROZŠÍŘENÍ

Rozšiřující moduly Arduino desek mají za cíl poskytnout při práci s deskou především další typy rozhraní, které nejsou integrovány na vývojové desce (nejčastěji integrovaným rozhraním je USB). Rozhraní lze chápat jako prostředek pro komunikaci desky s okolím.

Oficiální rozšíření, které desky arduino podporují

1) Arduino Motor Shield - umožňuje ovládat relé, elektromagnety, DC a krokové motory.

2) Arduino Proto Shield - obdoba nepájivého kontaktního pole, umožňuje rychlou realizaci projektů bez nutnosti pájení a zároveň realizaci hotových projektů pomocí připájení přímo k desce.

3) Arduino Ethernet Shield - umožňuje desce Arduino připojení k internetu pomocí síťového kabelu s konektorem RJ-45, zároveň umožňuje napájení desky přes tento

konektor. Shield obsahuje navíc slot pro micro-SD kartu, která může být použita pro ukládání souborů posílaných po síti.

4) Arduino GSM Shield - umožňuje desce Arduino připojení k internetu, přijímat a vysílat hlasové hovory a SMS zprávy.

5) Arduino WiFi Shield 101 - umožňuje desce Arduino připojení k internetu prostřednictvím bezdrátové WiFi komunikace.

6) Arduino USB Host Shield - umožňuje k desce Arduino připojit pomocí USB periferní zařízení, například flash disk, externí pevný disk, klávesnici, myš, GPS přijímač atp.

1.4 ALTERNATIVY K ARDUINU

Arduino alternativy můžeme dělit na dvě základní skupiny dle osazeného mikrokontroléru na desce, a to čipy z rodiny AVR - ATmega na deskách Arduino a desky bez čipu AVR. [5]

1.4.1 TI MSP430 LAUNCHPAD

Za cenu přibližně 10$ je Ti MSP430 LaunchPad skvělá nízkonákladová alternativa k deskám Arduino. Deska Ti MSP430 LaunchPad podporuje 3 rozšíření. [5]

1 - Energia IDE

2 - CCS Cloud

3 - Code Composer Studio

Nevýhodou může být ve srovnání s Arduinem nízké množství dostupných projektů. Desku vyrábí a distribuuje společnost Texas Instruments.

1.4.2 NETDUINO

Deska Netduino je založena na mikroprocesorech ARM a je programována pomocí rozhraní .NET. Ve srovnání s Arduino Uno jsou desky Netduino značně výkonnější a to díky procesoru s frekvencí až 168Mhz. Netduino podobně jako Arduino poskytuje velké množství různých desek, které lze najít na stránkách výrobce (https://www.adafruit.com/search?q=netduino&b=1). [5]

1.4.3 TEENSY

Vývojová řada desek Teensy je řadou malých desek na bázi mikroprocesoru ARM Cortex-M4 s frekvencí až 75Mhz. Teensy desky se programují pomocí softwaru Arduino

IDE, díky tomu lze realizovat vybrané projekty Arduino i s pomocí této desky, bez nutnosti velkého zásahu do programového kódu. [5]

1.4.4 PARTICLE PHOTON

Deska je založena na procesoru ARM Cortex M3 s taktovací frekvencí až 120Mhz. Deska disponuje WiFi rozhraním a je programována pomocí Photon IDE založeného na principu cloudu. [5]

1.4.5 ESP8226

Deska ESP8226 není zcela vývojovou deskou, jedná se spíše o mikroprocesor s WiFi připojením. ESP8226 je populární zejména díky ceně začínající na 2$, za tuto cenu je deska dostupná v různých konfiguracích. Deska může být programována pomocí softwaru Arduino IDE.

2 PRAKTICKÁ ČÁST

2.1 KLON OSOYOO UNO R3

Vybrané projekty popsané v následujících kapitolách byly realizovány a ověřeny na klonu desky Arduino Uno R3. Osoyoo Uno R3 je sestaveno dle open-source hardwarového designu, za použití stejných součástek. Deska je díky tomu věrnou kopií originálního sestavení desky Arduino Uno R3, od které se funkčně nijak neliší. Drobné rozdíly jsou pouze v barvě samotné desky a umístění některých popisů na desce.

Vlastnosti desky Osoyoo Uno R3

Konektor pro externí napájení umožňuje napájení desky zdrojem(případně baterií) v rozsahu od 6 - 20V, avšak doporučené napětí externího zdroje by mělo být v rozsahu 7-12V. Vnitřní průměr konektoru je 2,1mm.

Napěťový regulátor reguluje napětí vstupního zdroje na 5V, v případě vstupního napětí vyššího než 20V může dojít k poškození desky.

Mikrokontrolér ATmega16U2 zajišťuje komunikaci přes USB, zobrazuje se jako virtuální port COM v Arduino IDE.

USB konektor typu B umožňuje napájet desku a zajišťuje komunikaci mezi deskou a počítačem. Maximální přípustné proudové zatížení je 500mA. USB konektor počítače poskytuje napětí ve výši 5V. Pro připojení desky k počítači se používá USB a-B kabel.

Tlačítko RESET stisknutím dočasně spojí reset pin (Obrázek 3: 22) se zemí, dojde tak k přerušení běžícího programu a jeho znovu spuštění od začátku. RESET pin nahrazuje funkci resetovacího tlačítka na desce, pokud není tlačítko fyzicky přístupné. Resetu se dosáhne propojením se zemí (například pomocí externího tlačítka) .

AREF pin umožňuje zvýšit citlivost čtení analogových vstupů (Obrázek 3: 17), tedy snížit horní hranici čtení (standardně je horní hranice čtení analogových vstupů 5V).

Piny s označením GND představují virtuální zem.

Deska obsahuje 14 vstupně-výstupní digitálních pinů, označených na desce 0-13 (Obrázek 3: 8,9 a 10), každý pin může fungovat v režimu vstup nebo výstup, toto nastavení se provádí v řídícím programu. Vstup, resp. výstup pracuje ve dvou režimech 0 a 1, tyto režimy jsou reprezentovány napětím 0 a 5V. Proudové zatížení každého pinu

může dosáhnout až 40mA, pokud je deska napájena pomocí USB portu počítače, nesmí součet proudového zatížení přesáhnout 500mA. Některé piny mají speciální funkce:

- pin 0(RX) a 1(TX) - slouží pro příjem (RX - receive) a vysílání (TX - transmit) sériových dat TTL. Tyto piny jsou propojeny s piny ATmega16U2 (Obrázek 3: 3).
- pin 2 a 3 - umožňuje vyvolání externího přerušení, přerušení může být vyvoláno nízkou hodnotou, náběžnou nebo sestupnou hranou signálu, případně při změně hodnoty.
- PWM(piny 3, 5, 6, 9, 10 a 11) - piny označené znakem "~", 8-bit (může nabývat 256 různých hodnot), výstup s funkcí pulzní šířkové modulace - simulování analogového výstupu pomocí dvoustavového signálu.
- SPI: 10(SS), 11(MOSI), 12(MISO), 13(SCK) - umožňuje připojit sériové periferní zařízení, komunikace SPI je podporována při použití knihovny SPI.

Hlavice ICSP ATmega16U2 je vstupem pro přímé programování mikrokontroléru, používá se pro aktualizaci nebo nahrání firmwaru (zavaděče).

Deska dále obsahuje několik informačních LED diod, dioda na Obrázek 3: 12 je propojena s pinem 13. Svítí při hodnotě HIGH, nesvití při hodnotě LOW. Tato dioda je vhodná pro rychlé ověření funkčnosti desky. Dioda s označením na desce "ON" (Obrázek 3: 13) indikuje napájení desky. Diody s označením TX a RX (Obrázek 3: 14) indikující příjem(RX) a odesílání(TX) dat, jsou spojeny s piny TX a RX (Obrázek 3: 9 a 10) a mikrokontrolérem ATmega 16U2 (Obrázek 3: 3).

Hlavice ICSP ATmega328 je vstupem pro přímé programování mikrokontroléru, používá se pro aktualizaci nebo nahrání firmwaru (zavaděče).

Mikrokontrolér ATmega328 je jednočipový 8-bit mikrokontrolér, který obsahuje SRAM, Flash, CPU, 14 digitálních i/O pinů a 6 analogových vstupních pinů. Tento čip zajišťuje vykonání příkazů, tedy fungování celé desky, operační napětí je 5V. Mikrokontrolér ATmega328 je tedy klíčovým prvkem desky. Mikrokontrolér disponuje Flash pamětí, která není závislá na napájení a její obsah je zachován i po odpojení

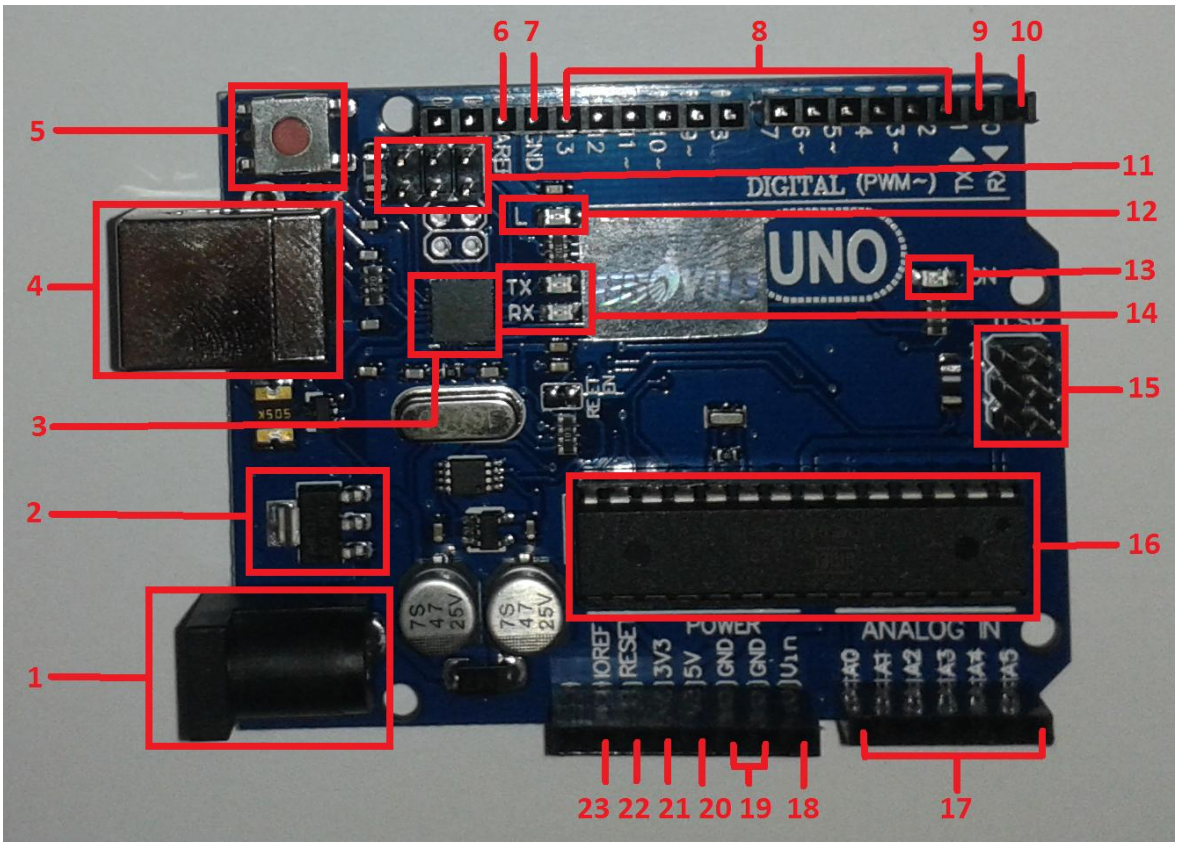
napájení, její velikost je 32KB, kde 0.5KB zabírá zavaděč(bootloader) a zbylých 31.5KB je uživatelsky přístupný pro program. V případě potřeby je možné zavaděč smazat a zpřístupnit tak celou paměť pro program. Dále obsahuje 1KB EEPROM paměť, její obsah je opět zachován i po odpojení napájení, tato paměť je využívána pro uložení hodnot z analogových vstupů. Lze ji chápat jako pevný disk počítače. Poslední paměť je paměť typu SRAM, která pro uchování dat potřebuje napájení, po odpojení se paměť vymaže, tato paměť je využívána pro uchování dočasných informací, které vznikly při běhu programu, který je umístěn ve flash paměti. Výpočetní výkon zajišťuje procesor o taktu 16MHz (je schopný zpracovat až 16 miliónů instrukcí za sekundu).

Vstupní analogové piny s označením na desce A0-A5 poskytují rozlišení 10-bit (každý vstup je schopen rozlišit až 1024 různých hodnot). Pomocí těchto vstupů lze číst signál z analogového čidla a převést ho pomocí a/D převodníku do digitální podoby. Ve výchozím nastavení čte hodnoty v rozsahu 0-5V. Horní hranici lze pomocí AREF pinu snížit a tak dosáhnout zvýšení rozlišovací schopnosti. Analogové vstupy mohou být zároveň použity jako digitální vstupně-výstupní piny, při tomto použití lze označení uvažovat jako i/O piny 14-19, například příkaz digitalWrite(19, HIGH) tedy nastaví na analogovém pinu s označením A5 maximální hodnotu, tedy 5V.

Vstupní napájecí pin označen Vin umožňuje napájení desky stejně jako konektor pro externí napájení, například pomocí baterie kde k Vin pinu je připojen kladný pól baterie a záporný pól je připojen k pinu GND. V případně, že je připojeno napájení pomocí konektoru, lze pomocí tohoto pinu přistupovat k neregulovanému napětí externího zdroje.

Deska umožňuje přistupovat k regulovanému napětí ve výši 5V a 3.3V prostřednictvím pinů označených na Obrázek 3: 20 a 21. Napájecí pin 3.3V poskytuje maximální proudové zatížení 50mA.

IOREF pin je informačního charakteru a poskytuje referenční napětí, se kterým mikrokontrolér pracuje. Tento pin je na desce obsažen pro možné další rozšíření desky.



Obrázek 3: Osoyoo UNO R3 (zdroj vlastní)

Vysvětlivky k číselnému značení na Obrázek 3

1 - Konektor pro externí napájení

2 - Napěťový regulátor

3 - Mikrokontrolér ATmega16U2

4 - USB konektor typu B

5 - Tlačítko RESET

6 - AREF pin

7 - GND pin

8 - Vstupní/výstupní digitální piny

9 - Pin 1(TX)

10 - Pin 0(RX)

11 - ICSP ATmega16U2

12 - LED

13 - Power LED (ON)

14 - RX a TX LED

15 - ICSP ATmega328

16 - Mikrokontrolér ATmega328

17 - Vstupní analogové piny

18 - Vstupní napájecí pin

20 - Výstupní napájecí 5V pin

21 - Výstupní napájecí 3.3V pin

22 - RESET pin

23 - IOREF pin

2.2 PRVNÍ SPUŠTĚNÍ

Při prvním spuštění Arduino IDE je potřeba v sekci nastavení zvolit desku, se kterou budeme pracovat, model desky najdeme na zadní straně této desky viz Obrázek 4. Zároveň je nutné zvolit port, přes který budeme desku programovat, při připojení přes USB se nastaví automaticky virtuální sériový port COM3.

Obrázek 4: Osoyoo Uno R3 zadní strana (zdroj vlastní)

2.3 BLIKÁNÍ LED

2.3.1 POPIS PROJEKTU

Tento projekt je zaměřen na základní pochopení fungování obvodu. Na příkladu si lze ověřit funkčnost desky, přenos dat mezi PC a deskou a správnost nastavení. V uvedeném příkladu se bude dioda střídavě rozsvěcet a zhasínat a to po stanovených intervalech. V zapojení je použit číslicový vstupně-výstupní pin 8, který pracuje ve dvou režimech HIGH a LOW. Hodnota HIGH je ekvivalentem hodnoty 255, kdy v tomto stavu mikrokontrolér desky nastaví na pinu 8 maximální napětí, tj. +5V. Hodnota LOW je ekvivalentem hodnoty 0, mikrokontrolér nastaví na pinu 8 0V, respektive připojí pin k zemi. [6, s. 26-30]

2.3.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

5) LED dioda

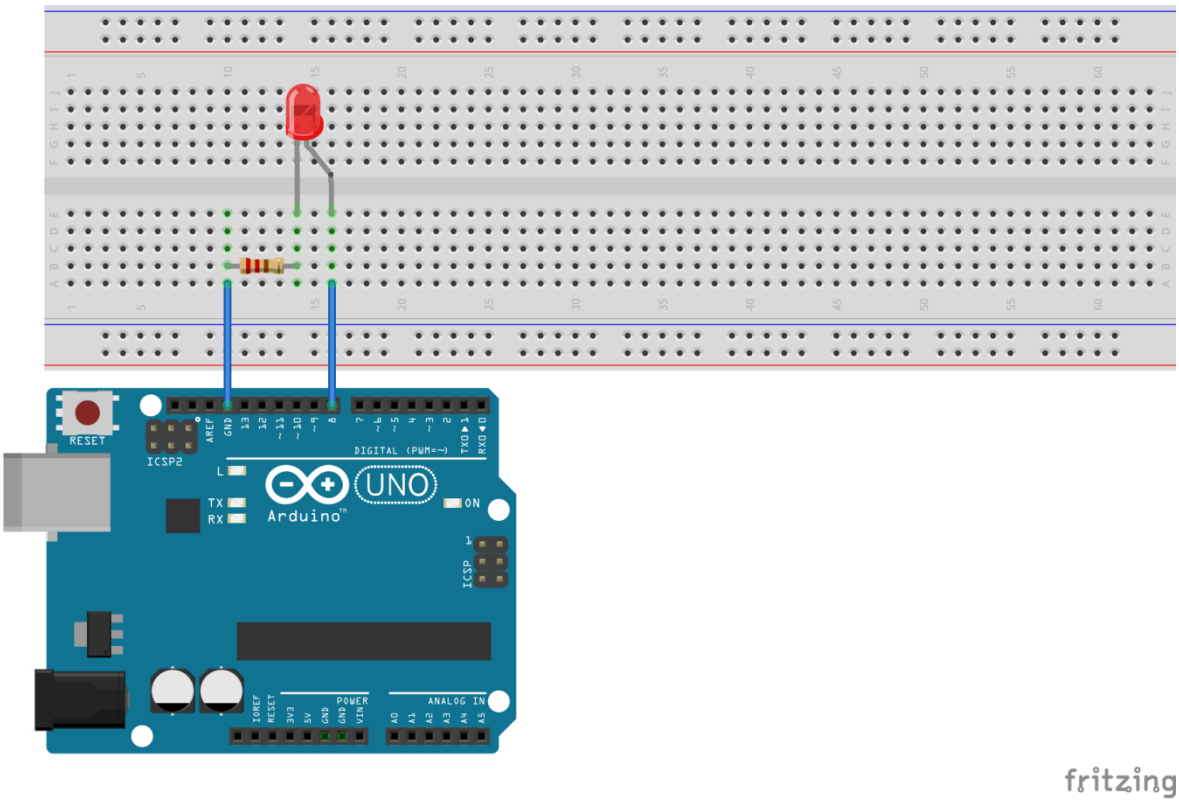
6) Rezistor 220 Ohm

2.3.3 KÓD PROGRAMOVÉHO ŘÍZENÍ

```
void setup() { //Instrukce se po restartu nebo novém nahrání provede
právě jednou
pinMode(8, OUTPUT); //Nastav pin 8 jako výstupní
}
void loop() {  //Instrukce se provádí neustále dokola
digitalWrite(8, HIGH); //Na pin 8 zapiš hodnotu HIGH = 255(dec) = 5V
delay(1000); //Před vykonáním další instrukce počkej 1000 ms = 1 s
digitalWrite(8,LOW); //Na pin 8 zapiš hodnotu LOW = 0(dec) = 0V
delay(1000);
}
```

2.3.4 ZAPOJENÍ

Katoda diody je zapojena přes rezistor k zemi (GND) a anoda je připojena k pinu 8 (Obrázek 5).



Obrázek 5: Blikání LED (zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2)

2.3.5 ROZŠÍŘENÍ

Jednou z možných variant, jak rozšířit tento projekt je kódování znaků za použití morseovy abecedy. Zároveň můžeme otestovat nastavení analogového pinu A5 jako digitální výstup. Lze například odvysílat pozdrav "ahoj", kde:

A=.- H=.... O=--- J=.---

Tečka bude reprezentována rozsvícením diody po dobu 500ms, čárka = 3*tečka, tedy po dobu 1500ms, mezi jednotlivými znaky pro odlišení je pak nutné nechat malou pauzu, mezi jednotlivými písmeny je pro odlišení pauza vyšší než je délka znaku čárka.

Neoptimalizovaný kód programového řízení je popsán v 2.3.6, jeho optimalizovaná a univerzálnější podoba v 2.3.7. Univerzálnost řešení spočívá v užití detekce znaků a optimalizace je zajištěna výstupem funkce volané parametrem.

2.3.6 KÓD PROGRAMOVÉHO ŘÍZENÍ ROZŠÍŘENÍ

```
void setup() {  //Instrukce se po restartu nebo novém nahrání provede
právě jednou
  pinMode(19, OUTPUT); //Nastav pin A5=19 jako výstupní

  delay(3000); //Vyčkej 3s.

  //.- = A
```

```
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(2000); // odlišení dvou písmen A-H

  //.... = H
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(2000); // odlišení dvou písmen H-O

  //--- = O
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(2000); // odlišení dvou písmen O-J

  //.--- = J
  digitalWrite(19, HIGH);
  delay(500); // tečka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
  delay(200); // odlišení dvou znaků vedle sebe morseovy abecedy
  digitalWrite(19, HIGH);
  delay(1500); //čárka morseovy abecedy
  digitalWrite(19, LOW);
}
void loop() {  //Instrukce se provádí neustále dokola
}
```

2.3.7 KÓD PROGRAMOVÉHO ŘÍZENÍ OPTIMALIZACE

```
int led = 19;  //hodnota 19 reprezentuje číslo použitého pinu
int teckaDelay = 500; //doba rozsvícení tečky
int carkaDelay = 1500;  //doba rozsvícení čárky
int morseDelay = 200; //doba pauzy mezi dvěma znaky morseovy abecedy
int znakDelay = 2000; //doba pauzy mezi dvěma písmeny

void setup() {  //Instrukce se po restartu nebo novém nahrání provede
právě jednou
  pinMode(led, OUTPUT); //Nastav pin 19 jako výstupní

  sekvenceMorseovky("...");    // S
  sekvenceMorseovky("---");    // O
  sekvenceMorseovky("...");    // S
  delay(5000);
  sekvenceMorseovky(".-");    // A
  sekvenceMorseovky("....");   // H
  sekvenceMorseovky("---");   // O
  sekvenceMorseovky(".---");  // J

}

void loop() {  //Instrukce se provádí neustále dokola
}
//funkce sekvenceMorseovky
//vstup je pole znaků tečka a čárka
void sekvenceMorseovky(char* sekvence)
{
  int i = 0;
  while (sekvence[i] != NULL) //pole znaků musí obsahovat alespoň jeden
znak
  {
    teckaCarka(sekvence[i]);  //odešle znak na i-té pozici do funkce
teckaCarka
    i++;
  }
  delay(znakDelay);
}
//funkce teckaCarka
//rozhudje o znaku a dle toho nastavuje délku rozsvícení LED, vstupem je
//jeden znak
void teckaCarka(char znak)
{
  digitalWrite(led, HIGH);  //rozsvítí LED
  if (znak == '.')  //pokud je znak . nechá diodu svítit po dobu 500ms
  {
    delay(teckaDelay);
  }
  else  //pokud není znak . musí být -, LED bude svítit 1500ms
  {
    delay(carkaDelay);
  }
  digitalWrite(led, LOW);
  delay(morseDelay);
```

2.4 DIGITÁLNÍ ČTENÍ

2.4.1 POPIS PROJEKTU

Projekt je zaměřen na monitorování stavu stisknutí tlačítka případně přepínače. Pomocí digitálního pinu 8, nastaveného jako vstup, čteme přivedenou hodnotu. Pomocí digitálního výstupu 7 pak dochází v návaznosti na digitální vstup 8 k rozsvěcení LED diody. Použitý tlačítkový spínač má dvojici vývodů na dvou stranách, vývody jsou vnitřně propojeny vertikálně. Pokud dojde ke stisknutí tlačítka, propojí se vývody zároveň horizontálně, v tomto stavu jsou tedy všechny vývody navzájem propojeny. Pin 8 rozlišuje dvě logické hodnoty 0 (vypnuto) při uzemnění a 1 (zapnuto) při 5V. V případě, že pin 8 přes rezistor neuzemníme, bude pin náhodně vracet stavy 0 a 1 a obvod tak nebude fungovat správně. Připojená LED dioda k pinu 7 má informační charakter, při stisknutí tlačítka svítí, při puštění opět zhasne. Stav 0 a 1 zároveň zobrazujeme pomocí sériového monitoru.

2.4.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

5) Tlačítkový spínač

6) Rezistor 10k Ohm

7) LED dioda

8) Rezistor 220 Ohm

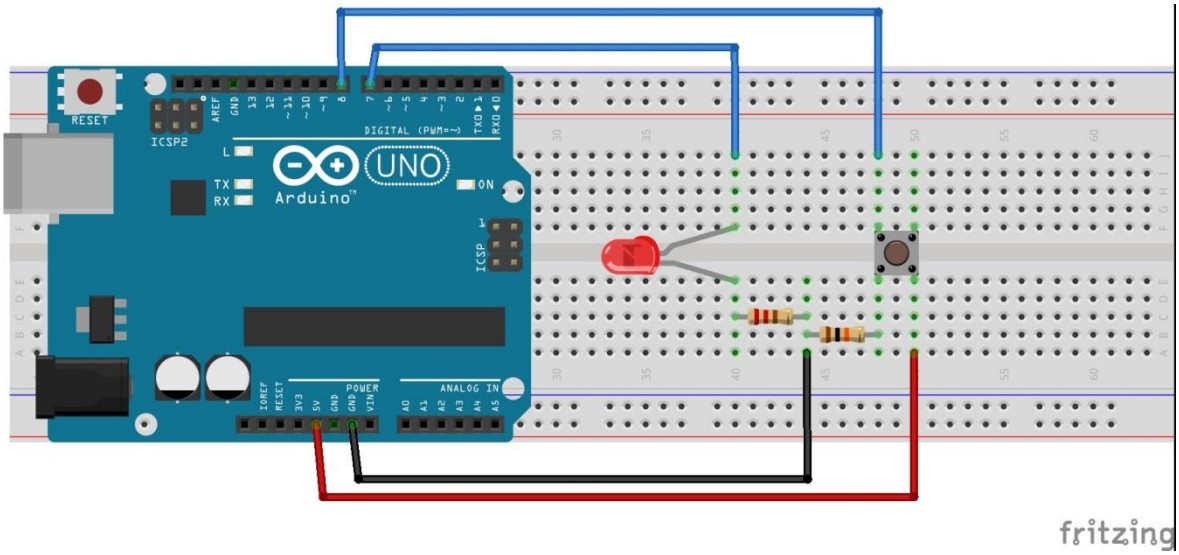
2.4.3 KÓD PROGRAMOVÉHO ŘÍZENÍ

```
int tlacitko = 8; //hodnota 8 reprezentuje číslo pinu, ke kterému je
tlačítko připojeno
int LED = 7; //hodnota 7 reprezentuje číslo pinu, ke kterému je připojena
anoda diody
int stavTlacitka;//deklarace proměnné, do které budeme ukládat stav pinu
8

void setup() {//Instrukce se po restartu nebo novém nahrání provede právě
jednou
  Serial.begin(9600);//inicializace sériové komunikace, parametr určuje
počet bitů za sekundu, v tomto případě 9600
```

```
  pinMode(tlacitko, INPUT);//nastav pin 8 jako vstupní
  pinMode(LED, OUTPUT);//nastav pin 7 jako výstupní
}

void loop() {//Instrukce se provádí neustále dokola
  stavTlacitka = digitalRead(tlacitko);//přečte stav pinu 8 a uloží tento
stav do proměnné
  digitalWrite(LED, stavTlacitka);//zapiš na pin 7 stav tlačítka
  Serial.println(stavTlacitka);//vytiskne na obrazovku sériového monitoru
stav tlačítka 0 nebo 1
  delay(10);//pauza 10ms mezi jednotlivým čtením vstupu
}
```

2.4.4 ZAPOJENÍ

Vývod tlačítkového spínače je na jedné straně připojen přes rezistor 10kohm k zemi. Vertikálně opačný vývod tlačítka je připojen k digitálnímu pinu 8. Vývod na druhé straně tlačítka je připojen k napájecímu pinu 5V. Anoda diody je připojena k digitálnímu pinu 7 a katoda je připojena přes rezistor 220ohm k zemi (Obrázek 6).



Obrázek 6: Digitální čtení (zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2)

2.5 ANALOGOVÉ ČTENÍ

2.5.1 POPIS PROJEKTU

V tomto projektu je ukázáno, jak načítat analogové hodnoty z fyzického světa za použití potenciometru. Potenciometr je jednoduchá mechanická součástka, která poskytuje různé velikosti odporu na základě pootočení hřídele. Při průchodu napětí přes potenciometr na analogový vstup dojde k úbytku napětí, které odpovídá velikosti odporu. Výsledné napětí lze změřit jako analogovou hodnotu převedenou na digitální. Deska analogově rozlišuje hodnoty 0-5V, digitálně 0-1023, deska tedy rozlišuje hodnoty po přibližně 4.888mV. Například 1V bude digitálně odpovídat hodnotě 205. Cílem této

úlohy je monitorování stavu potenciometru pomocí sériové komunikace mezi deskou Osoyoo Uno a IDE softwarem běžícím na počítači. V ukázkovém kódu je zajištěn zpětný přepočet na hodnotu napětí, po otevření sériového monitoru souběžně s běžícím programem v desce, bude program vypisovat aktuální hodnotu napětí na vstupu A0 s přesností na dvě desetinná místa.

2.5.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

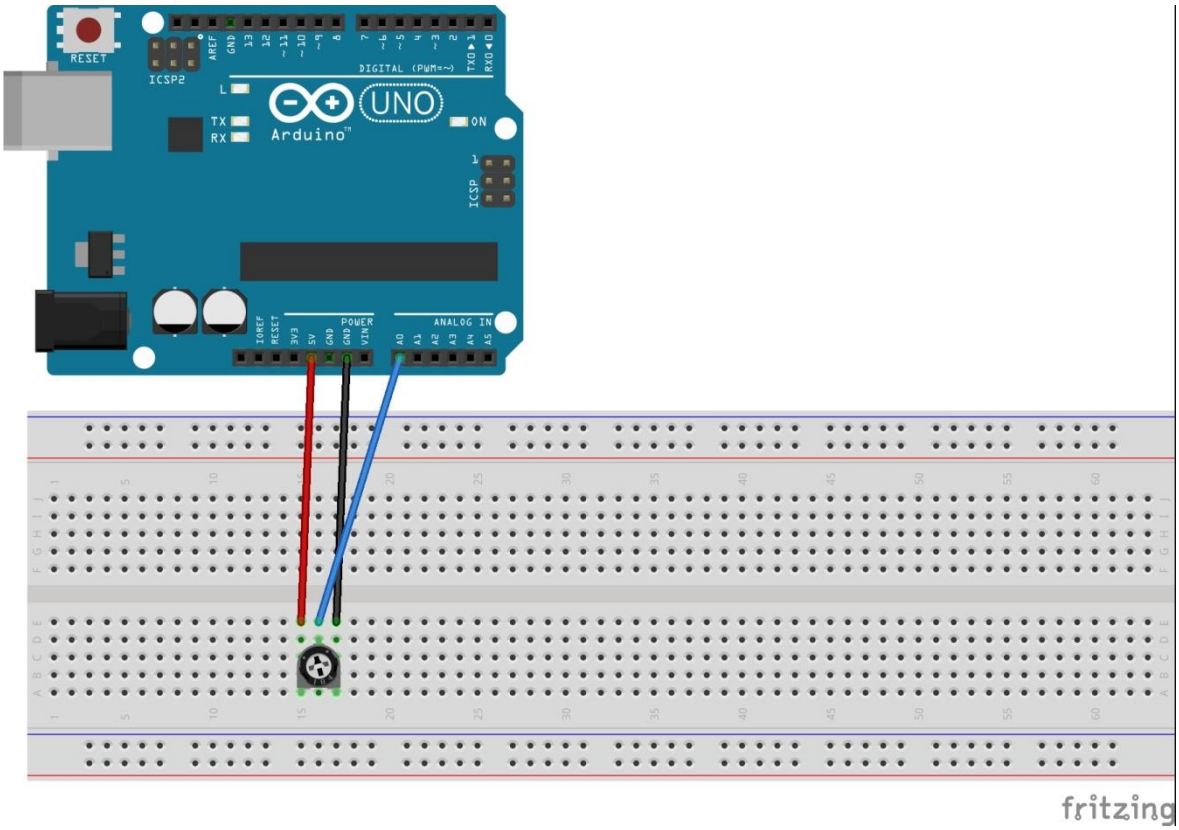
5) Potenciometr 10k Ohm

2.5.3 KÓD PROGRAMOVÉHO ŘÍZENÍ

```
int hodnotaVstupu;//deklarace globální proměnné typu inteeger
float napetiVstup;//deklarace globální proměnné typu float

void setup() {//Instrukce se po restartu nebo novém nahrání provede právě
jednou
  Serial.begin(9600);//inicializace sériové komunikace, parametr určuje
počet bitů za sekundu, v tomto případě 9600
}

void loop() {//Instrukce se provádí neustále dokola
  hodnotaVstupu = analogRead(A0);//do proměnné uloží aktuální digitální
hodnotu na pinu A0, hodnota je v rozsahu 0-1023, kde 1023 odpovídá
rozsahu napájení, tedy 5V
  napetiVstup = ((float)hodnotaVstupu * 5) / 1023; //uloží do proměnné
hodnotu vstupu přepočtenou na napětí, dochází zde k přetypování int to
float
  Serial.println(napetiVstup);//vytiskne na obrazovku sériového monitoru
hodnotu napětí na vstupu A0 zaokrouhlenou na dvě desetinná místa
  delay(10);        //pauza 10ms mezi jednotlivým čtením vstupu, použita
pro stabilizování čtení
}
```

2.5.4 ZAPOJENÍ

Potenciometr připojíme k desce prostřednictvím tří vodičů, jeden krajní vývod potenciometru připojíme k napájecímu pinu 5V, druhý krajní vývod k pinu GND a prostřední vývod připojíme k analogovému pinu A0 (Obrázek 7).



Obrázek 7: Analogové čtení (zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2)

2.6 HRACÍ KOSTKA

2.6.1 POPIS PROJEKTU

Projekt je zaměřen na práci s více vývody desky najednou. V uvedeném příkladu budeme pomocí funkce random(min, max) generovat pseudo-náhodné číslo, které bude simulovat výsledek po hodu hrací kostkou. Samotný hod kostkou je simulován stisknutím tlačítka, pokud je tlačítko stisknuto, vygeneruje random(1, 7) náhodné číslo v rozsahu 0 až 6 (od min do max-1). Pseudo-náhodné generování je zajištěno pomocí funkce randomSeed(analogRead(0)), která při zavolání čte hodnotu analogového vstupu A0, tento vstup je velmi citlivý a není-li připojen, je schopen zachytit elektromagnetický šum z okolí, kde tato hodnota lze považovat za náhodnou. Vygenerované číslo se následně zobrazí pomocí LED diod, které jsou rozmístěny stejně jako černé tečky na hrací kostce. V projektu je využito pole hodnot pro nastavení výstupních pinů a dále vícerozměrné pole (7*7) pro správné zobrazení pomocí LED.

2.6.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

5) Tlačítkový spínač

6) Rezistor 10k Ohm

7) LED dioda bílá 6ks

8) LED dioda červená 1ks

9) Rezistor 220 Ohm 7ks

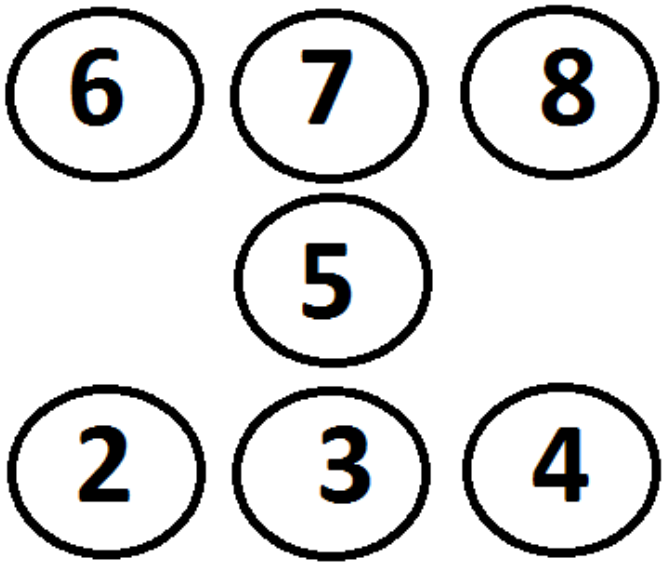
2.6.3 KÓD PROGRAMOVÉHO ŘÍZENÍ

```
byte tlacitko = 9;//hodnota 9 reprezentuje číslo pinu, ke kterému je
tlačítko připojeno
byte LEDpiny[7] = {2, 3, 4, 5, 6, 7, 8};//hodnoty v poli reprezentují
čísla pinů, ke kterým jsou připojeny anody LED diod
byte nahodne;//proměnná pro uložení generovaného pseudo-náhodného čísla
byte cislo[7][7] = {//vytvoří pole 7*7 a uloží do něj hodnoty viz níže
  {0, 0, 0, 0, 0, 0, 0},//0, tento řádek se nikdy nebude prohledávat
jelikož generujeme čísla v rozsahu 1-6
  {0, 0, 0, 1, 0, 0, 0},//1 - aktivní pin 5
  {1, 0, 0, 0, 0, 0, 1},//2 - aktivní pin 2 a 8
  {1, 0, 0, 1, 0, 0, 1},//3 - aktivní pin 2,5 a 8
  {1, 0, 1, 0, 1, 0, 1},//4 - aktivní pin 2,4,6 a 8
  {1, 0, 1, 1, 1, 0, 1},//5 - aktivní pin 2,4,5,6 a 8
  {1, 1, 1, 0, 1, 1, 1}//6  - aktivní pin 2,3,4,6,7 a 8
};
void setup() {
  Serial.begin(9600);//inicializace sériové komunikace, parametr určuje
počet bitů za sekundu, v tomto případě 9600
  randomSeed(analogRead(0));//pomocí analogového vstupu A0 vygeneruje
pseudo-náhodné číslo
  pinMode(tlacitko, INPUT);//nastav pin 8 jako vstupní
  for (int i = 0; i <= 6; i++)//příkazy v cyklu se vykonají 7*, hodnota
i bude v rozsahu od 0-6
  {
    pinMode(LEDpiny[i], OUTPUT);//postupně nastaví piny v rozsahu 0-6
jako výstupní, cyklus pro rozsvícení všech LED diod
    digitalWrite(LEDpiny[i], HIGH);//postupně bude diody rozsvěcet
    delay(200);//pauza mezi jednotlivým rozsvěcením
  }
  delay(1000);//po dobu jedné sekundy bude svítit všech 7 diod zároveň
  for (int i = 0; i <= 6; i++)//cyklus pro zhasnutí všech LED diod
  {
    digitalWrite(LEDpiny[i], LOW);//postupně bude diody zhasínat
    delay(200);//pauza mezi jednotlivým zhasínáním
  }
}
void loop() {
  if (digitalRead(tlacitko) == 1)//pokud je tlačítko stisknuto, proveď
následující:
  {
```

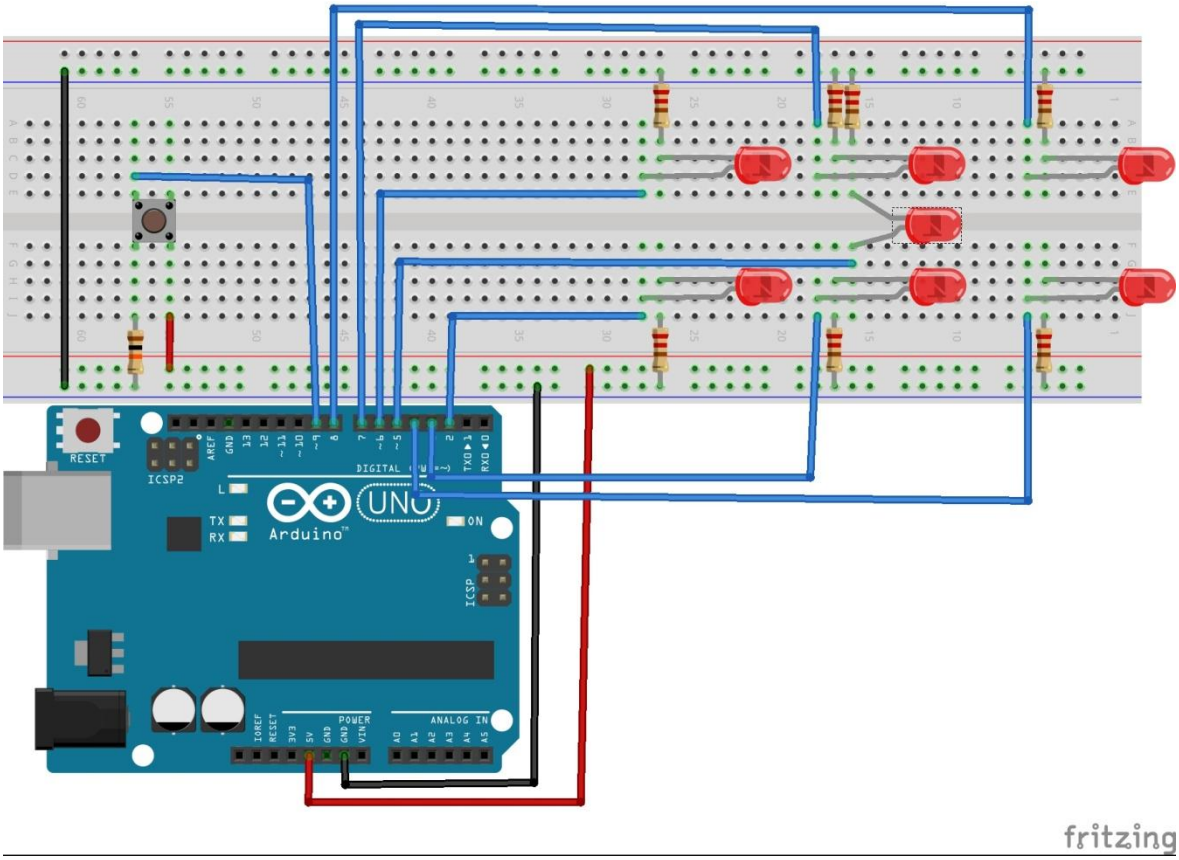
```
    nahodne = random(1, 7);//do proměnné ulož pseudo-náhodné číslo
v rozsahu 1-6
    Serial.println(nahodne);//vytiskne generované pseudo-náhodné číslo na
obrazovku sériového monitoru
    for (int i = 0; i <= 6; i++)//zhasne všechny rozsvícené diody -
předchozí zobrazené číslo
    {
      digitalWrite(LEDpiny[i], LOW);
    }
    for (int i = 0; i <= 6; i++)//cyklus pro rozsvícení diod dle matice
    {
      if (cislo[nahodne][i] == 1)//pokud je v definovaném řádku na pozici
0-6 hodnota 1
      {
        digitalWrite(LEDpiny[i], HIGH);//rozsvítí diodu připojenou
k odpovídajícímu pinu
      }
    }
    delay(500);//pauza mezi dvěma stavy stisknuto
  }
}
```

2.6.4 ZAPOJENÍ

Zapojení tohoto projektu je poměrně náročné, a vyžaduje zvýšenou pozornost. Zapojení tlačítka je vysvětleno v projektu Blikání LED. Zapojení diod je vysvětleno v projektu Digitální čtení. V zapojení je využitá společná zem, přivedená na krajní linii nepájivého kontaktního pole (Obrázek 9). Připojení diod je realizováno zleva doprava a zdola nahoru k pinu 2 až 8 viz Obrázek 8 a Obrázek 9.



Obrázek 8: Schéma rozložení diod s číselným označením pinů



Obrázek 9: Hrací kostka (zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2)

2.7 FOTOREZISTOR

2.7.1 POPIS PROJEKTU

Projekt je zaměřen na pochopení nutnosti kalibrace projektu pro jeho správné fungování. V projektu je zároveň ukázáno použití funkce map() a switch(). Fotorezistor je polovodičová součástka jejíž rezistence se mění v závislosti na intenzitě osvětlení. Při vysokém osvícení se rezistence pohybuje v řádu ohmů, naopak při vysokém zakrytí je rezistence v řádu mega ohmů. [7, s.72]

Kalibrační hodnoty se zjišťují v kapitole 2.7.3, dochází k přepisování hodnot minima a maxima, je-li hodnota vstupu nižší, resp. vyšší. Nové minimum a maximu pro přehlednost vypisuje program na obrazovku sériového monitoru. Jakmile známe aktuální minimum a maximum, kterého za současného osvětlení dosáhneme, musíme tyto hodnoty před použitím programu 2.7.4 zapsat do odpovídajících proměnných. Zapsáním hodnot dojde ke kalibraci programu. Samotný program pak odečítá hodnotu na analogovém pinu A0, která odpovídá intenzitě osvětlení, pomocí pře-mapování hodnot vstupu ve funkci Y=map(x, minStupniceX, maxStupniceX, minY, maxY), kde x je aktuální

hodnota osvětlení, a Y je návratová hodnota v rozsahu 0-3. Funkce map() je nejčastěji používána při závislosti digitálního pinu s analogovým, kde digitální piny pracují v rozsahu 0-255 a analogové v rozsahu 0-1023. Například, chceme-li hodnotu analogového vstupu zapisovat na digitální výstup, vypadalo by pře-mapování takto:

digitalOut=map(analogIN, 0, 1023, 0, 255)

Samotná funkce switch() funguje obdobně jako if, pro rozsáhlejší dělení není if vhodný, proto jej pro přehlednost může zastoupit podmínkové větvení switche, kde vstupní hodnota je porovnávána s nastavenou hodnotou v case, pokud je výsledek pravda, dojde k provedení příkazů v bloku case, pokud není ani jedna z podmínek splněna, je vhodné používat nastavení default, nastavení je bez parametru, a má za úkol ukončit hledání výsledku v bloku switch.

2.7.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

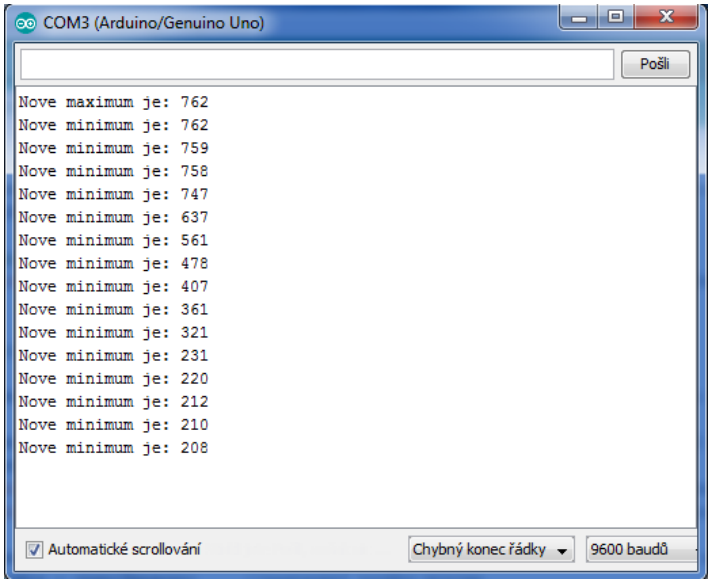
5) Fotorezistor

6) Rezistor 10k Ohm

2.7.3 KÓD PROGRAMOVÉHO ŘÍZENÍ - ZJIŠTĚNÍ MINIMA A MAXIMA

```
/*
   Kalibrační sketch FotorezistorMinMax.ino
*/

int sensorMin = 1023;  //hodnota minima sensoru, nastavená na začátku na
maximum rozsahu čtení, v případě že by byla hodnota nastavena na 0,
minimum by se odečítáním hodnot nikdy nezměnilo.
int sensorMax = 0;     //hodnota maxima sensoru, nastavená na začátku na
minimum rozsahu čtení, v případě že by byla hodnota nastavena na 1023,
maximum by se odečítáním hodnot nikdy nezměnilo.
int cteniSensoru;      //proměnná pro ukládání aktuální hodnoty senzoru

void setup() {
  Serial.begin(9600);  //inicializace sériové komunikace, parametr určuje
počet přenášených bitů za sekundu, v tomto případě 9600
}

void loop() {
```

```
  int cteniSensoru = analogRead(A0);  //do proměnné uloží aktuální
hodnotu senzoru připojeného k pinu A0

  if (cteniSensoru > sensorMax) {  //pokud je přečtená hodnota z pinu A0
vyšší než aktuální maximální hodnota, nastaví nové maximum
    sensorMax = cteniSensoru;      //nastavení nového maxima
    Serial.print("Nove maximum je: ");  //vytištění textu na obrazovku
sériového monitoru bez odřádkování
    Serial.println(sensorMax);          //vytištení hodnoty nového maxima
na obrazovku sériového monitoru
  }
  if (cteniSensoru < sensorMin) { //pokud je přečtená hodnota z pinu A0
menší než aktuální minimální hodnota, nastaví nové maximum
    sensorMin = cteniSensoru;     //nastavení nového minima
    Serial.print("Nove minimum je: ");  //vytištění textu na obrazovku
sériového monitoru bez odřádkování
    Serial.println(sensorMin);          //vytištení hodnoty nového minima
na obrazovku sériového monitoru
  }
  delay(100);       // pauza mezi jednotlivým čtením senzoru
}
```

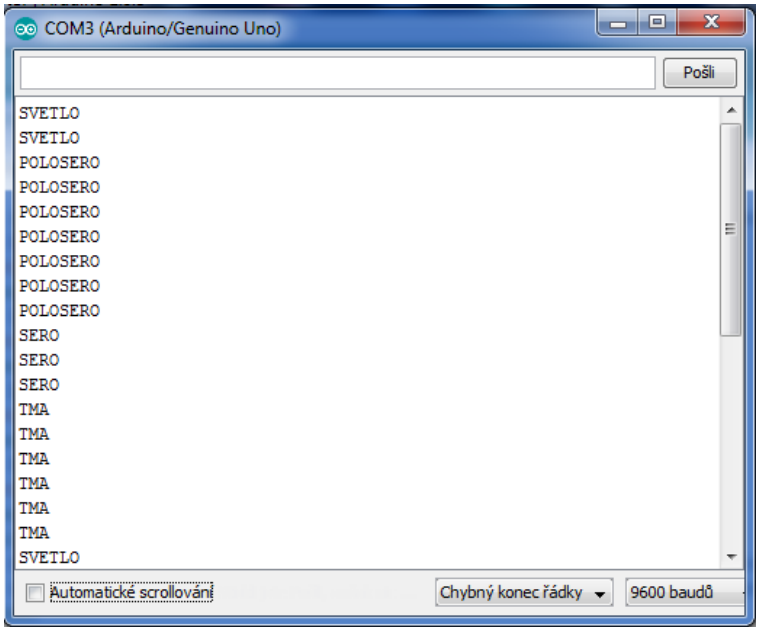


Obrázek 10: Výpis sériového monitoru (zdroj vlastní)

2.7.4 KÓD PROGRAMOVÉHO ŘÍZENÍ - VÝSTUP ÚROVNĚ OSVĚTLENÍ

```
int sensorMin = 150;   //hodnota minima sensoru, tuto hodnotu zjistítme
pomocí sketche FotorezistorMiMax
int sensorMax = 750;   //hodnota maxima sensoru, tuto hodnotu zjistítme
pomocí sketche FotorezistorMiMax
int cteniSensoru;      //proměnná pro ukládání aktuální hodnoty senzoru
int rozsah;            //pomocná proměná pro ukládání přemapované hodnoty
ze senzoru

void setup() {
  Serial.begin(9600); //inicializace sériové komunikace, parametr určuje
počet přenášených bitů za sekundu, v tomto případě 9600
}

void loop() {
  cteniSensoru = analogRead(A0); //do proměnné uloží aktuální hodnotu
senzoru připojeného k pinu A0
```

```
  int range = map(cteniSensoru, sensorMin, sensorMax, 0, 3);
//přemapování hodnoty ze senzoru, (x, minPuvodni, maxPuvodni,
minNove,maxNove)

  switch (range) {  //přepínač, obdoba složených podmínek if, je rychlejší
a přehlednější
    case 0:         //fotorezistor je zakytý - hodnota vstupu 150-300
      Serial.println("TMA");
      break;        //ukončí blok přepínač
    case 1:         //fotorezistor je téměř zakytý - hodnota vstupu 300-
450
      Serial.println("SERO");
      break;
    case 2:         //fotorezistor je téměř odkrytý - hodnota vstupu 450-
600
      Serial.println("POLOSERO");
      break;
    case 3:         //fotorezistor je odkrytý - hodnota vstupu 600-750
      Serial.println("SVETLO");
      break;
    default:
      // díky přemapování vstupu nikdy nebude nutné použít, v těle
přepínače být tedy nemusí
      // obsah v bloku default se vykonná, pokud není žádná podmínka
přepínače platná
      break;
  }
  delay(100); // pauza mezi jednotlivým čtením senzoru
}
```

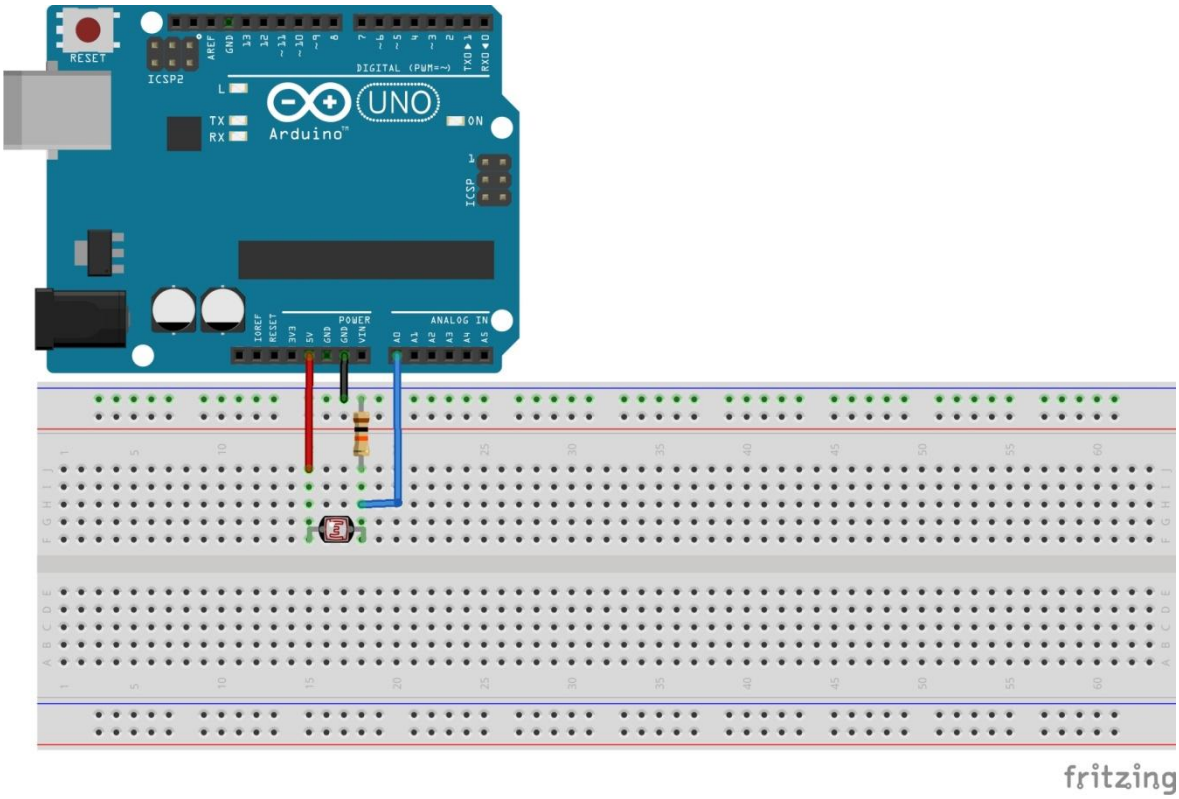


Obrázek 11: Výpis sériového monitoru (zdroj vlastní)

2.7.5 ZAPOJENÍ

Fotorezistor připojíme k desce prostřednictvím tří vodičů, jeden vývod fotorezistoru připojíme k napájecímu pinu 5V, druhý vývod k pinu GND pomocí 10k ohm rezistoru,

analogový vstup A0 napojíme mezi rezistor a vývod fotorezistoru viz Obrázek 12.



Obrázek 12: Fotorezistor (zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2)

2.8 DOTYKOVÉ PIANO

2.8.1 POPIS PROJEKTU

Projekt simuluje funkci piána pomocí dotykových ploch, vytvořených z alobalu nebo jiného vodivého materiálu. Ke snímání dotyku, bez nutnosti dotykového kapacitního modulu, využívá knihovnu CapacitiveSensor, která nastavuje piny jako senzory s režimem kapacitní citlivosti. Instalace jednotlivých senzorů vyžaduje odpory s rezistencí od 50k ohm do 50M ohm. Při zvyšující hodnotě rezistence, je senzor citlivější na přítomnost lidského těla. Senzory jsou schopné díky různým rezistencím rozeznat přímý kontakt s dotykovou plochou (1Mohm) a nebo pouhé přiblížení k dotykové ploše (10Mohm a více dle požadované vzdálenosti). Hlavní předností tohoto projektu je využití všech analogových vstupů v digitálním režimu. Díky těmto pinům je možné simulovat celý rozsah oktávy (7 celých not + 5 půl not). Přečtená hodnota senzoru se programově porovnává s mezí, pokud je mez překročena, program vyhodnotí vstup jako dotknutí se. Dle senzoru, jehož hodnota překročila mez, přehraje po stanovenou dobu při stanovené frekvenci odpovídající tón pomocí připojeného reproduktoru. V zapojení jsou použity dva

druhy rezistorů, základní má rezistenci 1Mohm, zbývající 2Mohm - tyto rezistory byly použity z důvodu nedostatku 1Mohm rezistorů, jsou zapojeny paralelně, výsledná rezistence je tedy dle vzorce R1=(R2+R2)/(R2*R2)=4/4=1[Mohm]. Podrobné informace k fungování knihovny CapacitiveSenzor a významu některých funkcí jako například millis() je možné dohledat na oficiálních stránkách Arduina (http://playground.arduino.cc//Main/CapacitiveSensor?from=Main.CapSense). Velikosti rezistorů důrazně nedoporučuji kombinovat, jelikož se vstupy kalibrují dle hodnot mezi piny 2-3, je nutné stejné hodnoty dodržet v celém zapojení.

2.8.2 POUŽITÉ SOUČÁSTKY

1) Osoyoo Uno R3

2) USB kabel

3) Nepájivé kontaktní pole

4) Propojovací vodiče

5) Rezistor 1M Ohm 5ks

6) Rezistor 2M Ohm 14ks (náhrada 1M Ohm 7ks)

7) Reproduktor

2.8.3 KÓD PROGRAMOVÉHO ŘÍZENÍ

```
/* Autor Pavel Eschler
   Arduino Uno piano s kompletním rozsahem tónu 5té oktávy za pomocí
analogových vstupů A0 až A5 = digitální PWM vstupy 14 až 19.
   Celý projekt zabíá 7 036 bytů úložného místa pro program.
   Glovální proměnné zabírají 583 bytů dynamické paměti.
*/
#include <CapacitiveSensor.h> //Importování knihovny CapacitiveSensor
Library.

#define reproduktor 19  //definuje proměnnou reproduktor na pinu A5=19
jako LED, tedy výstupní, lze použít jeden z neobsazenýh PWM pinu,
například 11

// definuje pin 2 pro odesílání a pin 3 až 18 pro příjem
// piny 2-9 jsou použity jako základní tóny
CapacitiveSensor   cs_2_3 = CapacitiveSensor(2, 3); // mezi piny 2 a 3 je
1Mohm rezistor, pin 3 funguje jako dotykový senzor při připojení kovového
kontaktu
CapacitiveSensor   cs_2_4 = CapacitiveSensor(2, 4); // mezi piny 2 a 4 je
1Mohm rezistor, pin 4 funguje jako dotykový senzor při připojení kovového
kontaktu
CapacitiveSensor   cs_2_5 = CapacitiveSensor(2, 5); // mezi piny 2 a 5 je
1Mohm rezistor, pin 5 funguje jako dotykový senzor při připojení kovového
kontaktu
```

```
CapacitiveSensor   cs_2_6 = CapacitiveSensor(2, 6); // mezi piny 2 a 6 je
1Mohm rezistor, pin 6 funguje jako dotykový senzor při připojení kovového
kontaktu
CapacitiveSensor   cs_2_7 = CapacitiveSensor(2, 7); // mezi piny 2 a 7 je
1Mohm rezistor, pin 7 funguje jako dotykový senzor při připojení kovového
kontaktu
CapacitiveSensor   cs_2_8 = CapacitiveSensor(2, 8); // mezi piny 2 a 8 je
1Mohm rezistor, pin 8 funguje jako dotykový senzor při připojení kovového
kontaktu
CapacitiveSensor   cs_2_9 = CapacitiveSensor(2, 9); // mezi piny 2 a 9 je
1Mohm rezistor, pin 9 funguje jako dotykový senzor při připojení kovového
kontaktu

// piny 14-18 jsou použity jako pultóny
CapacitiveSensor   cs_2_14 = CapacitiveSensor(2, 14); // mezi piny 2 a 14
je 1Mohm rezistor, pin 14 funguje jako dotykový senzor při připojení
kovového kontaktu
CapacitiveSensor   cs_2_15 = CapacitiveSensor(2, 15); // mezi piny 2 a 15
je 1Mohm rezistor, pin 15 funguje jako dotykový senzor při připojení
kovového kontaktu
CapacitiveSensor   cs_2_16 = CapacitiveSensor(2, 16); // mezi piny 2 a 16
je 1Mohm rezistor, pin 16 funguje jako dotykový senzor při připojení
kovového kontaktu
CapacitiveSensor   cs_2_17 = CapacitiveSensor(2, 17); // mezi piny 2 a 17
je 1Mohm rezistor, pin 17 funguje jako dotykový senzor při připojení
kovového kontaktu
CapacitiveSensor   cs_2_18 = CapacitiveSensor(2, 18); // mezi piny 2 a 18
je 1Mohm rezistor, pin 18 funguje jako dotykový senzor při připojení
kovového kontaktu

void setup()
{
  cs_2_3.set_CS_AutocaL_Millis(0xFFFFFFFF); // vypne autokalibraci na
kanálu 1
  Serial.begin(9600); //inicializace sériové komunikace, parametr určuje
počet přenášených bitů za sekundu, v tomto případě 9600
}

void loop()
{
  long start = millis();//nastaví časovač, millis je rychlejší
alternativa k delay

  // přiřazení vstupů k proměnným
  long notaC =  cs_2_3.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 3, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaD =  cs_2_4.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 4, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaE =  cs_2_5.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 5, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaF =  cs_2_6.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 6, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaG =  cs_2_7.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 7, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
```

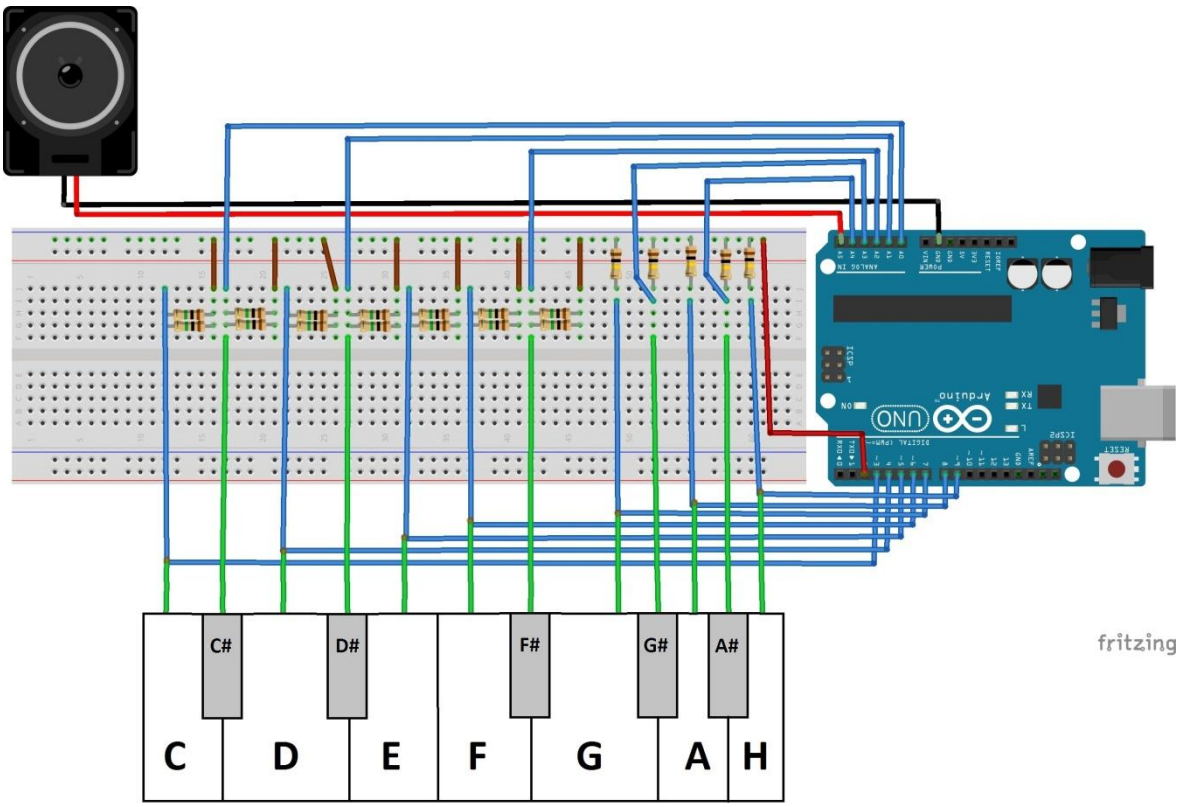
```
  long notaA =  cs_2_8.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 8, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaH =  cs_2_9.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 9, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více

  long notaCs =  cs_2_14.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 14, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaDs =  cs_2_15.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 15, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaFs =  cs_2_16.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 16, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaGs =  cs_2_17.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 17, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více
  long notaAs =  cs_2_18.capacitiveSensor(60);//načte hodnotu senzoru na
pinu 18, parametr 60 označuje citlivost čtení a nastavuje se v závislosti
na použitém rezistoru, pro 10Mohm je doporučeno 80 a více

  Serial.print(millis() - start); // zkontroluje výkon v milisekundách
  // tisk hodnot na monitor
  Serial.print("\t");             // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaC);           // vytiskne hodnotu senzoru připojeného
k pinu 3
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaD);           // vytiskne hodnotu senzoru připojeného
k pinu 4
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaE);           // vytiskne hodnotu senzoru připojeného
k pinu 5
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaF);           // vytiskne hodnotu senzoru připojeného
k pinu 6
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaG);           // vytiskne hodnotu senzoru připojeného
k pinu 7
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaA);           // vytiskne hodnotu senzoru připojeného
k pinu 8
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaH);           // vytiskne hodnotu senzoru připojeného
k pinu 9
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor

  Serial.print(notaCs);          // vytiskne hodnotu senzoru připojeného
k pinu 14
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
```

```
  Serial.print(notaDs);          // vytiskne hodnotu senzoru připojeného
k pinu 15
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaFs);          // vytiskne hodnotu senzoru připojeného
k pinu 16
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.print(notaGs);          // vytiskne hodnotu senzoru připojeného
k pinu 17
  Serial.print("\t");            // vytiskne na obrazovku sériového
monitoru odsazení - tabulátor
  Serial.println(notaAs);        // vytiskne hodnotu senzoru připojeného
k pinu 18, ukončí řádek, další výpis proběhne od začátku nového řádku

  // pokud se dotkneme kovového kontaktu připojeného mezi čtecí pin
a rezistor, hodnota překročí mez a rozezní reproduktor, jedná se o další
nastavení citlivosti
  // frekvence 523Hz odpovídá notě C 5té oktávy, další noty jsou opět
frekvenče nastaveny dle 5-té oktávy, parametry frekvence a doba znění lze
libovolně měnit
  if (notaC > 150) tone(reproduktor, 523, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 523Hz
střídat napětí 0 a 5V
  if (notaD > 150) tone(reproduktor, 587, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 587Hz
střídat napětí 0 a 5V
  if (notaE > 150) tone(reproduktor, 659, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 659Hz
střídat napětí 0 a 5V
  if (notaF > 150) tone(reproduktor, 698, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 698Hz
střídat napětí 0 a 5V
  if (notaG > 150) tone(reproduktor, 784, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 784Hz
střídat napětí 0 a 5V
  if (notaA > 150) tone(reproduktor, 880, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 880Hz
střídat napětí 0 a 5V
  if (notaH > 150) tone(reproduktor, 988, 250); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 250ms bude při frekvenci 988Hz
střídat napětí 0 a 5V
  //frekvence 554Hz odpovídá frekvenčně C# 5té oktávy
  if (notaCs > 150) tone(reproduktor, 554, 150); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 150ms bude při frekvenci 554Hz
střídat napětí 0 a 5V
  if (notaDs > 150) tone(reproduktor, 622, 150); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 150ms bude při frekvenci 622Hz
střídat napětí 0 a 5V
  if (notaFs > 150) tone(reproduktor, 740, 150); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 150ms bude při frekvenci 740Hz
střídat napětí 0 a 5V
  if (notaGs > 150) tone(reproduktor, 831, 150); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 150ms bude při frekvenci 831Hz
střídat napětí 0 a 5V
  if (notaAs > 150) tone(reproduktor, 932, 150); // pokud je překročena
mez, aktivuje pin reproduktoru a po dobu 150ms bude při frekvenci 932Hz
střídat napětí 0 a 5V

  // vypne genereování tónu do reproduktoru, lze jím nastavit délku
generování tónu hromadně pro všechny noty
```

```
  /* začátek blokového komentáře
    if (notaC<=150  &  notaD<=150  &  notaE<=150 & notaF<=150  &
 notaG<=150  &  notaA<=150 &  notaH<=150 &  notaCs<=150 &  notaDs<=150 &
 notaFs<=150 &  notaGs<=150 &  notaAs<=150)
      {
        delay(250);
        noTone(reproduktor);
      }
    konec blokového komentáře */
  delay(10);   // pauza definující rychlost čtení, omezuje tok dat na
monitor sériového portu a zajišťuje stabilitu čtení
}
```

2.8.4 ZAPOJENÍ

Na desku nepájivého kontaktního pole připojíme pin 2 na svislici užívanou pro společnou zem, k této svislici postupně připojíme rezistory 1Mohm, v případě 2Mohm rezistorů tyto rezistory po dvou připojíme paralelně (vznikne tak rezistence mezi konci o požadované velikosti 1Mohm). Druhý vývod rezistorů postupně připojíme k digitálním pinům 3 až 9 (celé noty) a analogovým vstupům nastavených v digitálním režimu A0 až A4, neboli 14 až 19 (půlené noty). Mezi vývody rezistorů a piny připojíme pomocí vodičů jakékoliv vodivé plošky (např. alobal), které zajistí prostor pro dotek prstem.



Obrázek 13: Dotykové piano(zdroj vlastní, vytvořeno ve Fritzing beta 0.9.2 a následně upraveno)

ZÁVĚR

Platforma Arduino umožňuje za velmi přívětivou cenu realizovat širokou škálu projektů, díky kterým je uživatel o krok blíže k pochopení fungování programově řízených obvodů a počítačů. Za hlavní přínos práce považuji ověření Osoyoo klonu desky Arduino, která dokázala, že není nutné kupovat vždy nejdražší součástky a příslušenství. Osoyoo Uno klon dosahuje v projektech stejných výsledků jako originální deska, přičemž tento nebo jiný klon se dá pořídit i za desetinu ceny originální desky.

V praktické části práce jsem se snažil podat modifikované existující projekty co nejjednodušší formu pro čtenáře. Praktická část lze považovat za rychlý náhled do tajů programování desek řízených mikrokontrolérem. Jednotlivé projekty jsou řazeny vzestupně dle obtížnosti. Poslední projekt v kapitole 2.8 Dotykové piano jsem realizoval s ohledem na možné využití ve vzdělávání. Hudba a technika nejsou příliš spojovanými směry, i přesto technika může mít přímou spojitost s výukou hudby a naopak. Tomuto tvrzení nasvědčuje fakt, že před realizací posledního projektu pro mne byly pojmy jako oktáva, její rozsah, význam not a půl not zapovězeny.

Platforma Arduino si díky své jednoduchosti a fyzickému výstupu postupně razí cestu ve vzdělávacích programech po celém světě. Platforma především vzbuzuje zájem o techniku a poukazuje na fakt, že kód který uživatel naprogramuje nemusí nutně zůstat uvnitř osobního počítače a být tak pro většinu uživatelů jen nutnou překážkou. Díky vizualizaci kódu prostřednictvím vývojové desky, dochází u uživatele ke zvýšení zájmu o programování jako takové.

RESUMÉ

Arduino platform let the user to implement wide variety of schemes for very pleasant price. That steps up user's understanding for use of programme controlled circuits and computers. From my view I see the main contribution of this thesis in verificating clone of Arduino board, Osoyoo. That proves it's not allways necessary to buy the most expensive components and accessories. Osoyoo Uno clone has the same results like the original board, whereby this clone or anotheo one can be bought for one tenth price of the original one.

In practical part I tried to demonstrate modified, already existing schemes as easy as it possible. Practical part can be taken as the quick preview of programming boards controlled by microcontroller.

All the projects are sorted gradually from easiest to hardest. I designed the last scheme in the chapter 2.8, Touch piano, for possible use in education. Music and technology might not seem much connected, despite technology can be used in music teaching. This can be proved by me, not having a clue what do octave or notes mean, before creating the last scheme.

Arduino platform is being used more and more in educational programs all over the world thanks to its simplicity and physical output. The platform raises interest in technology and shows that the code programmed by user doesn't have to stay only inside personal computer and therefore be just a necessary obstacle. Thanks to the vizualization of the code by developmental board the user's interest in programming can raise.

SEZNAM LITERATURY

1. *Arduino* [online]. 2016 [cit. 2016-02-29]. Dostupné z: http://www.arduino.cc/
2. NUSSEY, John. *Arduino for dummies*. West Sussex, England: Wiley, c2013. --For dummies. ISBN 978-1-118-44642-3.
3. VODA, Zbyšek. *Průvodce světem Arduina*. Vydání první. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.
4. MARGOLIS, Michael. *Arduino cookbook*. 2nd ed. Sebastopol, Calif.: O'Reilly, c2012. ISBN 14-493-1387-6.
5. UDANIS, ALEX. 5 *Great Arduino Alternatives*. In: All about circuits [online]. Boise: EETech Media, 2015 [cit. 2016-02-29]. Dostupné z: http://www.allaboutcircuits.com/news/5-great-arduino-alternatives/
6. BANZI, Massimo a Michael SHILOH. *Getting Started with Arduino*. USA. Maker Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2014. ISBN 978-1-4493-6333-8.
7. MONK, Simon. *30 Arduino projects for the evil genius*. New York: McGraw-Hill, c2010, xiii, 191 p. ISBN 00-717-4133-X.

SEZNAM OBRÁZKŮ, TABULEK, GRAFŮ A DIAGRAMŮ

PŘÍLOHY

Příloha 1: DVD-R obsahující elektronickou verzi bakalářské práce, programové kódy k jednotlivým projektům a software Arduino IDE.