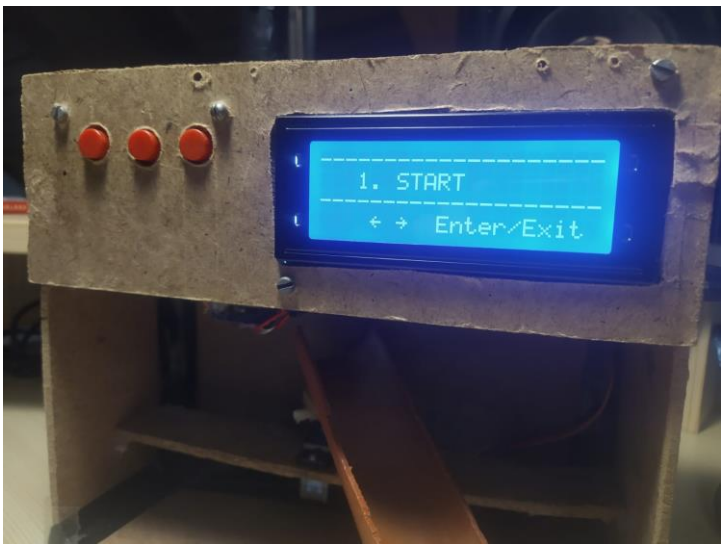
SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

# ZÁVĚREČNÁ STUDIJNÍ PRÁCE

**dokumentace**

## Arduino - třídička

Jan Wolf




| | |
|---|---|
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE<br>se zaměřením na počítačové sítě a programování |
| **Třída:** | IT4 |
| **Školní rok:** | 2021/2022 |

***Poděkování:***

*Chtěl bych tímto poděkovat panu učiteli Grussmannovi za užitečné rady a také panu učiteli Lučnému za pomoc při psaní dokumentace.*

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2021

*podpis autora práce*

# ANOTACE

Projekt se zabývá vývojem třídičky barev s využitím RGB senzoru. Tento senzor funguje na principu zachycení a filtrace základních barev, kterými jsou červená, zelená a modrá. Toto je realizováno za pomocí barevných filtrů, fotodiod a převodníku, které jsou součástí RGB senzoru TCS3200. Práce se zabývá testováním měřených hodnot senzoru, algoritmizací zápisu a měření barev, a také realizací jednoduchého uživatelského menu s výpisem na LCD displej.

Na projekt byl využit mikrokontrolér Arduino UNO s procesorem ATmega328P. Programová část je řešena v jazyce Wiring, který je odnož jazyka C++ a s využitím externích knihoven pro zjednodušení a přehlednost. Třídička dokáže zapsat barvu do dvojrozměrného pole a následně ji rozlišovat. Toto je realizováno uživatelským menu ovládaným tlačítky.

## Klíčová slova

Arduino UNO, mikrokontroler, RGB, TCS3200, dvojrozměrné pole

# OBSAH

# ÚVOD

K výběru mé závěrečné práce mě vedl můj zájem o elektrotechniku, hardware a programování mikrokontrolerů. V rámci své práce se věnuji převážně testování a programování barevného senzoru TCS3200 s využitím vývojové desky Arduino UNO. S Arduinem jsem měl zkušenosti již z hodin pana učitele Godovského.

Cílem práce bylo vytvořit program sloužící k uživatelskému přidávání barev a následnému třídění. Toto je řízeno pomocí tlačítek a menu na displeji. Mimo jiné jsem se také věnoval mechanické realizaci pro automatický posun a roztřídění předmětů.

V průmyslu jsou barevné senzory hojně využívány například pro detekci a třídění podle barevné značky, kontrolu etiket, popisů nebo správného obsahu. Nalezneme je nejčastěji ve strojním průmyslu na montážních linkách a kontrolách kvality.

V mé dokumentaci se věnuji popisu vývoje mého projektu, využitým technologiím a mojí principiální realizaci. Součástí je také obrázková dokumentace průběhu vývoje třídičky, co se týče vizuálního provedení, využitých komponentů a uživatelského menu.

# 1 VÝROBA A VÝVOJ TŘÍDIČKY

## 1.1 Počátky a testování

Začátky mojí práce na ročníkovém projektu spočívaly především ve výběru vhodného hardware. Dále jsem si navrhl zapojení jednotlivých součástek a testoval jejich funkčnost.

Jako mikrokontrolér jsem zvolil vývojovou desku Arduino UNO, se kterou jsem měl zkušenosti z hodin pana učitele Godovského. Při výběru senzoru pro zjišťování barev, mě přesvědčil senzor TCS3200, kvůli jeho dobré dostupnosti a využití v podobných aplikacích.

Moje začátky testování probíhaly v programu Visual Studio Code v kombinaci s pluginem PlatformIO, ve kterém jsem programoval celou aplikaci. Postupoval jsem krok po kroku a přidával další součástky. Prvním krokem byl test již zmíněného barevného senzoru. Funkčnost jsem ověřoval jednoduchým zápisem jednotlivých frekvencí RGB do pole a následným výpisem na sériovou linku. Tento proces jsem zkoušel na barevných papírech a následně ručně zapisoval hodnoty do dvojrozměrného pole, které jsem dále pomocí podmínky rozlišoval.

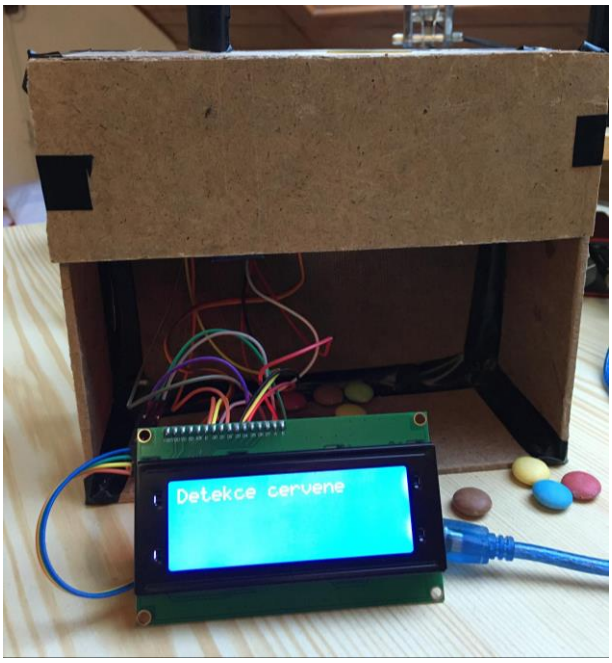
Zde jsem ale narazil na první velký problém, kterým byl zápis složek barev vůči okolnímu světlu, který se mi ovšem později podařilo vyřešit. Jednoduše řečeno, zapsaná barva měla jiné RGB složky při silném denním světle než například v šeru, jelikož na snímač zároveň dopadalo i okolní světlo. Senzor sice disponuje čtyřmi LE diodami, které mají za úkol daný předmět nasvítit. Toto nasvícení ovšem „nepřebije“ okolní světlo. Z tohoto důvodu mě napadla myšlenka dynamického přidávání barev.

## 1.2 Výroba krabičky

Jako další krok mě čekala výroba samotné krabičky, kde budou umístěny součástky. Třídičku jsem od počátku chtěl vyrobit pro účel třídění lentilek či jiných malých předmětů. Na projektu jsem začal pracovat již o velkých prázdninách, neměl jsem ale možnost 3D tisku krabičky. Abych mohl na projektu pokračovat, rozhodl jsem se krabičku vyrobit po svém.

Věděl jsem, že je velmi důležité, aby lentilky padaly na senzor pokaždé ve stejné poloze z důvodu správného snímání barvy. Kvůli ručně vyráběným mechanickým prvkům, lentil-

ky nepadaly přesně na místo snímání senzoru, nebo se při manipulaci zasekávaly. Snímání barev se proto stalo méně spolehlivé.

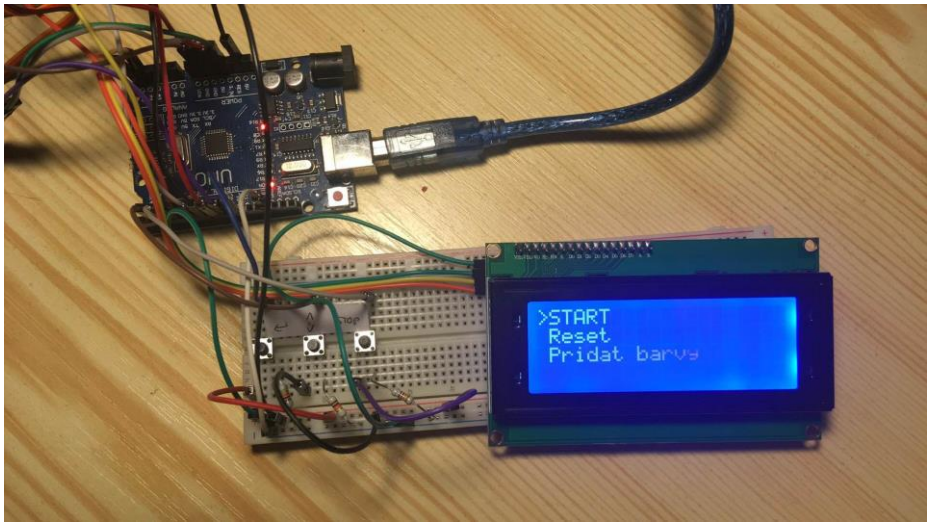



*Obrázek 1: Detekce barvy*

Z těchto důvodů, a také na doporučení pana učitele Grussmanna, jsem přidal další funkci, a to *Přidat barvu*, která umožnovala uživatelsky přidat jakoukoliv barvu. Tím jsem částečně vyřešil problém s okolním světlem, protože barva se nasnímala a zapsala vůči okolnímu svitu.

## 1.3 Konečný vývoj

Po vytvoření krabičky jsem se vrhnul na programování tlačítek, displeje a menu. Menu jsem poprvé vytvořil bez využití jakýchkoli knihoven, tudíž docházelo k opakování a nepřehlednosti kódu. Toto prozatímní menu jsem vypisoval na displej a ovládal pomocí tlačítek, připojených na nepájivém kontaktním poli.




*Obrázek 2: Jednoduché menu*

Následně jsem napájel tlačítka a rezistory na plošný spoj a zabudoval společně s displejem do předního krytu krabičky. Tím jsem dokončil výrobu krabičky.

Na doporučení pana učitele Grussmanna, jsem přeprogramoval menu pomocí knihovny *CMBMenu*, která umožnuje více úrovňové menu a přehlednější kód. Pro větší spolehlivost tlačítek a dalších funkcí jsem upravil tlačítka přes knihovnu *OneButton*, která obsahuje funkce, jako například *doublepress* nebo *longpress*.

Jednou z posledních funkcí byl čítač jednotlivých barev, zobrazený na displeji při třídění a funkce *Reset* pro resetování čítače.

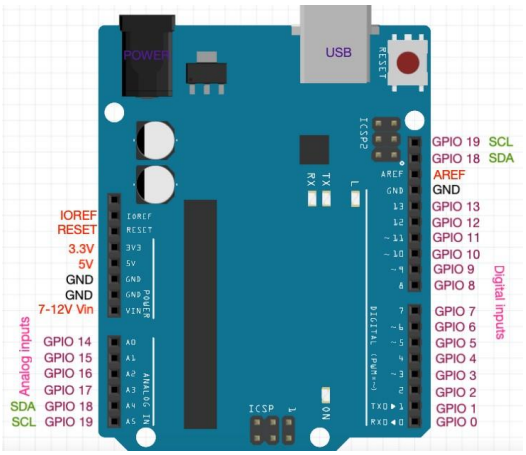
Projekt byl tímto téměř hotov. Bohužel, v průběhu vývoje jsem narazil na další problém a tím byla malá citlivost senzoru. Barvy, které jsou si svými složkami RGB podobné, nedokáže senzor dostatečně citlivě nasnímat a následně rozlišit. S tímto jsou spojeny i výše zmíněné problémy s mechanickými prvky. Například, pokud lentilky spadly na senzor obráceně či kousek vedle, senzor je špatně nasnímal. Kvůli těmto problémům jsem musel omezit počet přidávaných barev. Přesto ale třídička funguje často nespolehlivě.

# 2 VYUŽITÉ TECHNOLOGIE

## 2.1 Hardware

### 2.1.1 Arduino UNO

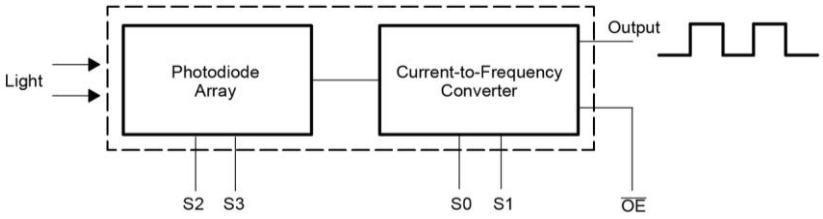
Srdce projektu tvoří vývojová deska Arduino UNO, stojící na procesoru ATmega328P, pracující na frekvenci 16 MHz a disponuje flash pamětí 32 KB. Tato deska nabízí 14 digitálních vstupně výstupních pinů a 6 vstupních analogových pinů. Arduino disponuje USB konektorem pro propojení s počítačem, *vin* konektor pro připojení AC/DC zdroje a konektorem pro 9 V baterii. Jednočip pracuje s napětím 5 V, ale umožnuje jako výstupní napětí také 3,3 V.



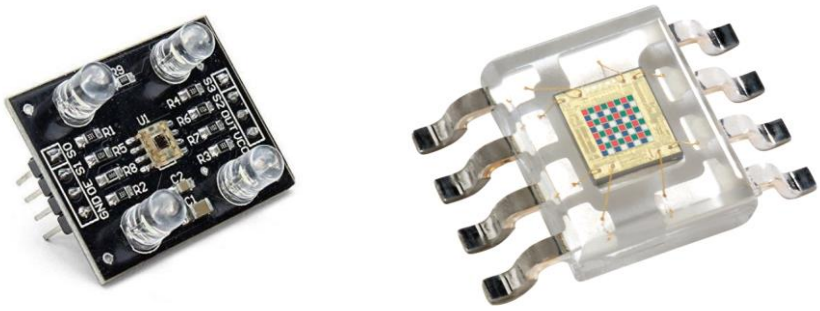

*Obrázek 3: Pinout Arduino UNO*

### 2.1.2 Detektor barev TCS3200

Tento senzor funguje na principu přeměny intenzity světla, odraženého od barevného předmětu, na elektrický proud a následně na frekvenci jednotlivých složek RGB. Provozní napětí je 3 V–5 V. Doporučuji využít 5 V, protože LE diody poté vyzařují více světla a detekce se stává spolehlivější. TCS3200 disponuje čipem, který obsahuje pole fotodiod s filtrem pro červenou, zelenou, modrou a žádnou barvu. Tyto diody spolu s převodníkem zajišťují přeměnu intenzity světla na elektrický proud.




*Obrázek 4: Princip fungování senzoru*



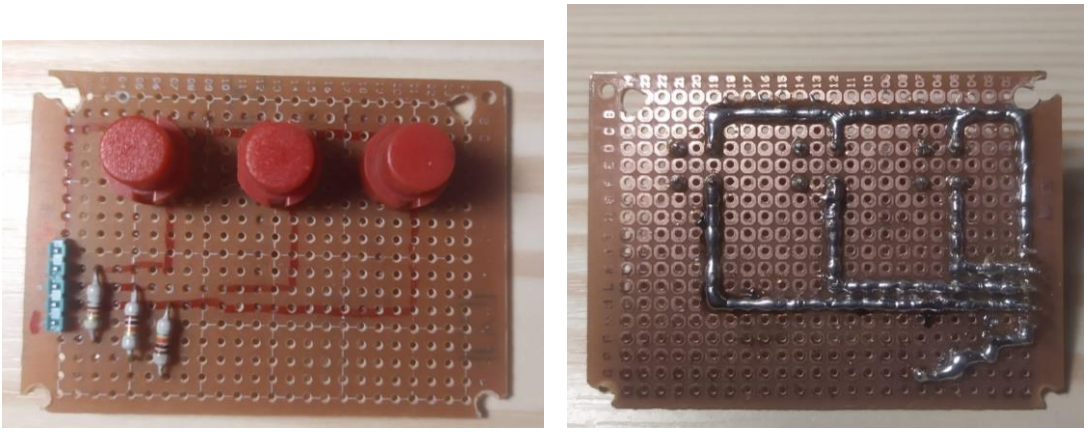
*Obrázek 5: Detail snímače*

### 2.1.3 LCD display s I2C převodníkem

Tento modře podsvícený displej umožňuje vypisování znaků ve 20 sloupcích a 4 řadách. Kvůli jednoduchosti zapojení jsem se rozhodl využit I2C převodník čili k zapojení jsem využil pouze 2 vodiče SDA a SCL, a samozřejmě také vodiče pro napájení. Intenzita podsvícení se reguluje trimerem na I2C převodníku. Ke komunikaci jsem využil knihovnu pro jednoduché ovládání *LiquidCrystal_I2C*.

### 2.1.4 Tlačítkový modul

Pro ovládání menu na displeji jsem využil trojice tlačítek a rezistorů. Tyto součástky jsem kvůli efektivnějšímu provedení následně napájel na univerzální plošný spoj a zabudoval do krabičky třídičky. Tlačítka jsou uzemněné 10 KΩ rezistorem, mají 3 výstupní konektory pro čtení hodnot z tlačítek a společnou anodu a katodu.



*Obrázek 6: Tlačítkový modul*

### 2.1.5 Mikro servo SG90

Pro posuv předmětů ze zásobníku a pro otáčení jsem využil dvou servo motorů. Provozní napětí je 5 V–6 V.

## 2.2 Software

### 2.2.1 Programovací jazyk Wiring

Celý program je napsaný v jazyce Wiring, který byl vyvinut pro programování mikrokontroléru a nejčastěji je spojen s vývojovým prostředím Arduino IDE. Wiring je odnož jazyků C a C++ a využívá knihovnu *Arduino.h* pro funkce, které jsou nutné k práci s mikrokontrolerem.

### 2.2.2 Visual Studio Code a plugin PlatformIO

Jako integrované vývojové prostředí jsem využil program Visual Studio Code. Tento program umožňuje nainstalování mnoha pluginů pro rychlejší a efektivnější vývoj, kterým je například tzv. *našeptávač*, nebo profesionální plugin pro programování aplikací pro mikrokontrolery PlatformIO, který jsem využil.

### 2.2.3 KiCad 5.1.10

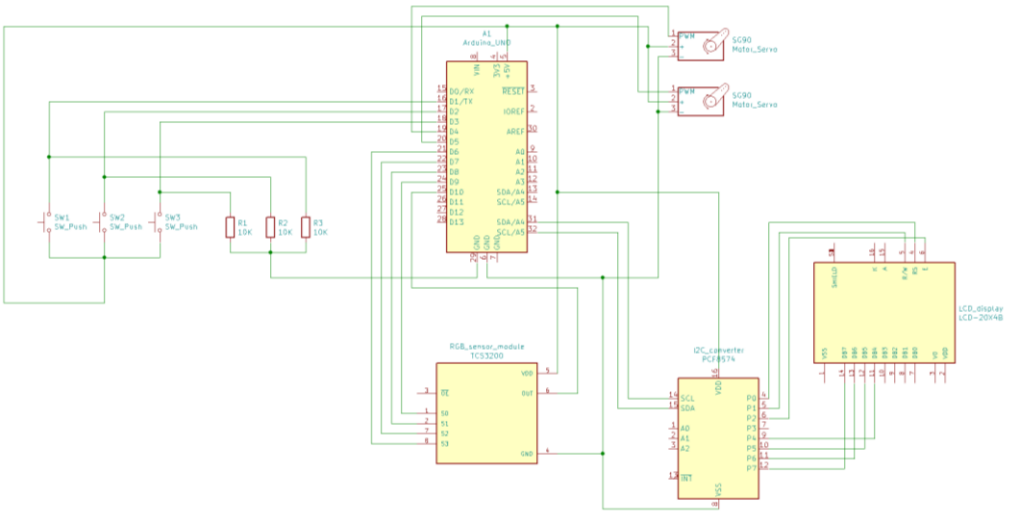
Pro nákres schématu zapojení, jsem si vybral oper-source software KiCad, který umožňuje projektování plošných spojů a disponuje obrovskou databází nejrůznějších elektrotechnických součástek.

# 3 ZPŮSOBY ŘEŠENÍ A POUŽITÉ POSTUPY

## 3.1 Využité součástky

- Mikrokontroler Arduino UNO
- RGB senzor TCS3200
- LCD display 20x4 znaků
- I2C převodník PCF8574
- 2x mikro servo SG90
- 3x tlačítko
- 3x rezistor 10 KΩ
- Univerzální plošný spoj
- Malé nepájivé kontaktní pole

## 3.2 Schématické zapojení



*Obrázek 7: Elektrotechnické schéma*

## 3.3 Zápis barev

Hlavní částí projektu byla práce se senzorem barev. RGB složky byly ukládány do statického dvojrozměrného pole. V první části vývoje třídičky jsem do dvojrozměrného pole

ručně zapisoval vypsané složky, které jsem rozlišoval a poté vypisoval na sériovou linku. Toto se brzy projevilo jako nepraktické. Později jsem zapisování naprogramoval jako automatické, pomocí stisknutí tlačítka.

*Ukázka kódu zápisu barvy*:

```
do{  //Čtení červené barvy, pomocí nastavení pinů S2 a S3 na LOW
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  //Zápis frekvence do proměnné redFrequency
  redFrequency = pulseIn(sensorOut, LOW);

  //Čtení zelené barvy, pomocí nastavení pinů S2 a S3 na HIGH
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  //Zápis frekvence do proměnné greenFrequency
  greenFrequency = pulseIn(sensorOut, LOW);

  //Čtení modré barvy, pomocí nastavení pinů S2 na LOW a S3 na HIGH
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  //Zápis frekvence do proměnné blueFrequency
  blueFrequency = pulseIn(sensorOut, LOW);

  int RGB[] = {redFrequency, greenFrequency, blueFrequency};

  for (int i = 0; i <= pocetBarev; i++)
    colors[aktualniBarva - 1][i] = RGB[i];
} while (millis() - cas < 1500);
delay(1000);
servo1.write(50);
```

V této části programu proběhne načtení frekvencí ze senzoru, pomocí funkce *pulseIn* a uložení těchto proměnných do pole *RGB*. Funkce *pulseIn* vrací čas v mikrosekundách a říká nám, po jakou dobu byl puls, který čteme z předdefinovaného pinu, HIGH nebo LOW. Tato doba závisí na frekvenci a mění se s intenzitou nasnímané barvy. Následně v cyklu *for* se tato trojice hodnot uloží do globálního dvojrozměrného pole *colors*. Řádky dvojrozměrného pole se mění podle globální proměnné *akutalniBarva*, která se inkrementuje při opětovném volání funkce.

Pole *colors* jsem kontroloval pomocí pomocné funkce *vypisPole()* na sériové lince.

## 3.4 Rozlišování barev

Jakmile byly barvy zapsané do pole *colors*, následovalo jejich rozlišování. V začátcích projektu jsem si jednoduše změřil a zapsal složky RGB, například pro červenou barvu. Dále v programu se tyto složky kontrolovaly a vypisovaly na displej „Detekce červené barvy“. Toto statické zapisování se ukázalo jako velmi nespolehlivé, hlavně kvůli problémům s okolním světlem a samozřejmě také nedynamičností. Toto se dělo díky jinému nasvícení, například ve tmě či šeru. Barvy se tím pádem rozlišovaly špatně (viz výše).

*Ukázka kódu rozlišování barev*:

```
int pom = 0; //pomocná proměnná pro kontrolu jednotlivých složek RGB
//Kontrola právě měřené barvy (pole RGB) se změřenými barvy v poli colors
    for (int r = 0; r < pocetBarev; r++){
      for (int c = 0; c < 3; c++){
        if (RGB[c] + (odchylka) >= colors[r][c] &&
        RGB[c] - (odchylka) <= colors[r][c]){
          pom++;
          if (pom == 3)
            return r;
        }
      }
    }
    return -1;
```

Nyní program za pomocí dvou cyklů a jedné podmínky kontroluje, zda se aktuální RGB, změřené ze senzoru, rovná některému ze zapsaných v poli. Zde jsem musel brát v potaz odchylku při měření. Ovšem kvůli malé citlivosti senzoru tato odchylka „zasahovala“ do jiných barev. Pokud podmínka platí, funkce vrátí číslo řádku, což odpovídá změřené barvě. Pokud ne, inkrementuje se řádek a cyklus kontroluje, zda se nejedná o další barvu.

Aby došlo ke zvýšení spolehlivosti, prováděl jsem toto měření po dobu dvou sekund. Návratová hodnota funkce *mereniZapisRGB()* (viz ukázky kódu výše) představuje index pole právě změřené barvy. Tato barva se ale může lišit od té, která se zde skutečně vyskytuje. Proto se tato hodnota zapisuje do pole *help*. Prvek pole *help* s nevyšší hodnotou se zde vyskytuje nejčastěji. Jestliže například první index pole *help* obsahuje číslo kolem tisíce a hodnoty na jiných indexech jsou menší, jako výsledná barva se vybere ta s nejvyšší hodnotou.

Tento princip měření je přesnější, než kdyby se RGB frekvence ze senzoru kontrolovaly pouze pomocí jedné podmínky.

*Ukázka kódu měření:*

```
do{
  int color;
  color = mereniZapisRGB(aktualniBarva);
  if (color == 0)
    help[0]++;
  if (color == 1)
    help[1]++;
  if (color == 2)
    help[2]++;
} while (millis() - time < 2500); //měření po dobu 2500 milisekund
```

## 3.5 Knihovna CMBmenu.h

Na doporučení pana učitele Grussmanna jsem naprogramoval jednoduché víceúrovňové uživatelské menu, ovládaného pomocí tlačítek. Využil jsem knihovnu *CMBmenu.h*, čímž se zvýšila přehlednost a jednoduchost kódu. Prvním krokem při tvorbě menu je přidání tzv. **node** neboli uzlu. K tomuto uzlu je přiřazeno číslo úrovně menu, text, který chceme vypisovat a jedinečné ID.

*Ukázka kódu přidávání node*:

```
const char g_Start_pc[] PROGMEM = {"1. START"};
const char g_Reset_pc[] PROGMEM = {"2. Reset"};
const char g_Pridat_pc[] PROGMEM = {"3. Pridat barvu"};
const char g_PridatBarvu_pc[] PROGMEM = {"Vhodte predmet"};
```

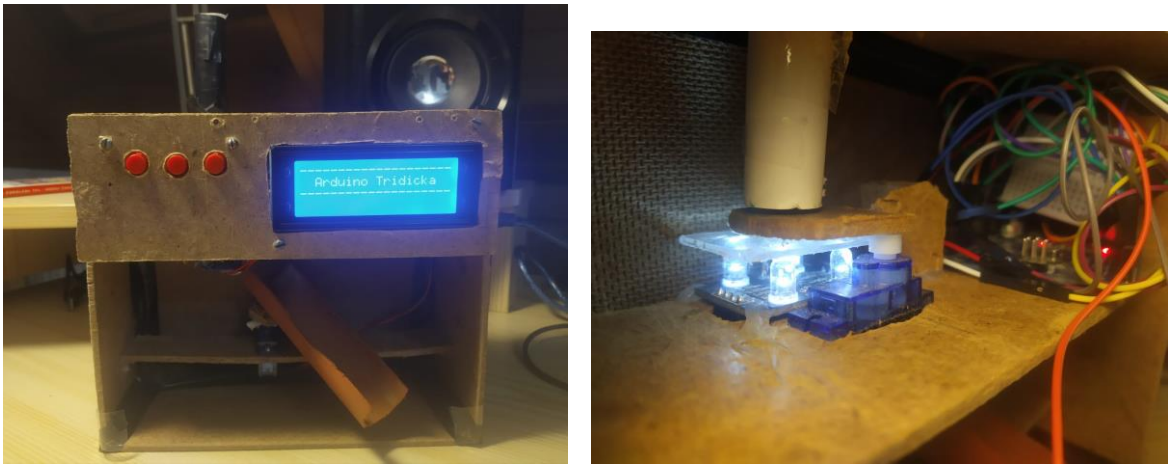
Dále pomocí tlačítek jednoduše vyvoláváme funkce *left(), right(), enter(), exit()*, pro pohybování v menu. Předtím ještě musíme sestavit a vypsat menu pomocí funkcí *buildMenu()* a *printMenu()*.

Při stisku tlačítka, reprezentující funkci *enter()*, se kontroluje ID, podle kterého je vyvolána určitá funkce.

# 4 VÝSLEDKY ŘEŠENÍ

## 4.1 Hardwarové a mechanické provedení

Výsledné zařízení, co se týče vzhledu a provedení, odpovídá mým představám, které jsem měl na začátku své práce. Třídička má vestavěný LCD displej, tlačítkový modul, zásobník na lentilky a oba servo motory tak, jak jsem si představoval.




*Obrázek 8: Výsledné řešení*

Na druhou stranu je zde pár záležitostí, které bych udělal jinak nebo lépe, kdybych je býval věděl na začátku mé práce na projektu. Jednou z nich je, že bych celou třídičku vytvořil na třídění větších předmětů, než jsou lentilky. Na snímač by se totiž nasnímalo více barevného světla, které se odráží od předmětu, z důvodu jeho vetší plochy. Tím pádem by vznikl větší kontrast mezi barevnými předměty.

Jako další věc bych upravil mechanické prvky, vymodeloval a vytisknul je na 3D tiskárně tak, aby se předmět dostal na snímač vždy do stejného místa, a tak pokaždé správně přečetl barvu. Vhodné by bylo také napájení komponentů na plošný spoj a změna vývojové desky na Arduino Nano, aby došlo k redukci vodičů.

Vzhledem k výše uvedeným problémům, které se týkají mechanických částí a velikosti předmětů, se třídička chová často nespolehlivě a nepřesně.

## 4.2 Softwarové provedení

Co se týče funkčnosti samotného programu, za pomocí ošetřených tlačítek, včetně funkcí *doublepress* a *longpress*, je možné se pohybovat v menu. Program dokáže zapsat barvu do pole a následně, po programátorem určený čas, měřit a rozlišovat dané barvy. Čítač barev, zobrazovaný na displeji a pole zapsaných barev, se dá vynulovat funkcí *Reset*.


*Obrázek 9: Ukázka menu*

# Závěr

Úkolem bylo vytvořit třídičku s využitím RGB senzoru TCS3200 s možností dynamického přidávání vlastních barev. Zadaný cíl byl splněn včetně výroby krabičky, osazení komponentů, napájení tlačítek na plošný spoj a automatického posuvu lentilek.

Je zde bezpochyby několik částí projektu, které by vyžadovaly vylepšení či jinou realizaci, jak jsem již zmínil v předchozích kapitolách. Jednou z nich je realizace třídičky na větší předměty, než jsou lentilky a vytisknutí potřebných mechanických prvků na 3D tiskárně. Dále by bylo vhodné například vytvoření dopravníkového pásu, po kterém by se předměty posouvaly a třídily se tak, jak je realizováno v průmyslu. Dalším vylepšením by mohlo být navrhnutí PCB desky plošného spoje, kvůli redukci vodičů. Kvůli malé citlivosti barevného senzoru by bylo dobré uvažovat o jeho výměně za přesnější. Z těchto důvodů se jedná spíše o amatérský projekt a jeho realizaci, tudíž by pravděpodobně nenašel své místo v praxi.

# SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Funkce pulseIn(). Www.arduino.cc [online]. Itálie: Arduino, 2020 [cit. 2021-12-13]. Dostupné z: https://www.arduino.cc/reference/en/language/functions/advanced-io/pulsein/

[2] Color Detection Using TCS3200/230. Create.arduino.cc [online]. SurtrTech, 2019 [cit. 2021-12-13]. Dostupné z: https://create.arduino.cc/projecthub/SurtrTech/color-detection-using-tcs3200-230-84a663

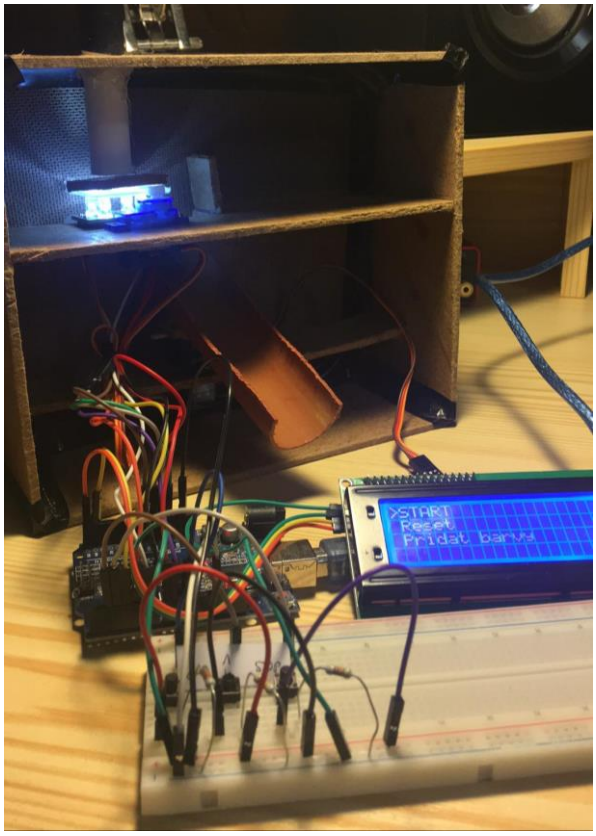
[3] MbLib. Github.com [online]. Michael Bernhard, 2021 [cit. 2021-12-13]. Dostupné z: https://github.com/mchlbrnhrd/mbLib

[4] OneButton. Github.com [online]. Matthias Hertel, 2013 [cit. 2021-12-13]. Dostupné z: https://github.com/mathertel/OneButton

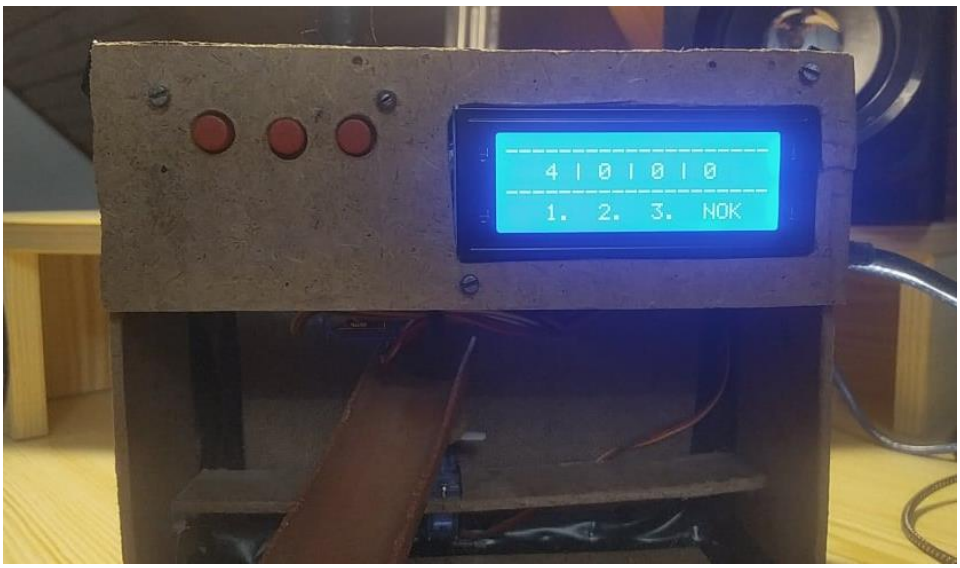
[5] Práce s LCD. Hwkitchen.cz [online]. Česká republika: HWKITCHEN, 2020 [cit. 2021-12-13]. Dostupné z: https://www.hwkitchen.cz/navody-hwkitchen/prace-s-lcd-je-jednoducha-arduino-navody/

# SEZNAM PŘÍLOH

# Příloha č. 1: Fotodokumentace




*Obrázek 10: Třídička ve vývoji*




*Obrázek 11: Čítač barev*