SŠPU
STŘEDNÍ ŠKOLA PRŮMYSLOVÁ
A UMĚLECKÁ, OPAVA

ZÁVĚREČNÁ STUDIJNÍ PRÁCE

dokumentace

Robotické rameno ovládáné pomocí gest

Autor: Sofja Klopcová

Obor: 18-20-M/01 INFORMAČNÍ TECHNOLOGIE
se zaměřením na počítačové sítě a programování

Třída: IT4

Školní rok: 2023/24

Poděkování

Chci poděkovat panu učiteli Godovskému za poskytnutí veškerého hardware a konzultace během celé práci.

Prohlášení

Prohlašuji, že jsem závěrečnou práci vypracovala samostatně a uvedla veškeré použité informační zdroje.
Souhlasím, aby tato studijní práce byla použita k výukovým a prezentačním účelům na Střední průmyslové a umělecké škole v Opavě, Praskova 399/8.

V Opavě 1. 1. 2024

.....
Podpis autora

Abstrakt

Výsledkem projektu je funkční pohyblivé rameno, které zvládá chytnout kuličku z konce stavebnice a posune ji na začátek a také za dobře odvedenou práci se uklonit. Ovládané je gesty které jsou snímány kamerou notebooku. Hlavním aspektem projektu je kontrola jednotlivých servomotoru ramene pomocí gest. Kód který umožní zobrazovat gesta je psaný v Pythonu, a holá pohyblivost ramene je obstarána Arduinem které využívá C++.

Klíčová slova

poghyblivé rameno, gesta, Python, C++, Serva, Servomotory, knihovny, ukázky.

Abstract

The result of the project is a functional moving arm that manages to grab the ball from the end of the kit and move it to the beginning, as well as take a bow for a job well done. It is controlled by gestures that are captured by the laptop's camera. The main aspect of the project is the control of the individual servomotors of the arm using gestures. The code that allows the gestures to be displayed is written in Python, and the bare arm's mobility is handled by an Arduino which uses C++.

Keywords

moving arm, gestures, Python, C++, Servos, Servomotors, libraries, demos.

Obsah

Úvod

Tato dokumentace detailně popisuje celkovou funkcionalitu mého projektu. Když jsme byli seznámeni s požadavky na ukončení čtvrtého ročníku ročníkovým projektem, rozhodla jsem se zaměřit svůj projekt především na oblast hardwaru. Po dlouhém a náročném průzkumu a s pomocí pana Godovského jsem nakonec zvolila robotické rameno pro svůj projekt. Bohužel, proces ovládání mě přivedl zejména k softwarovým aspektům.

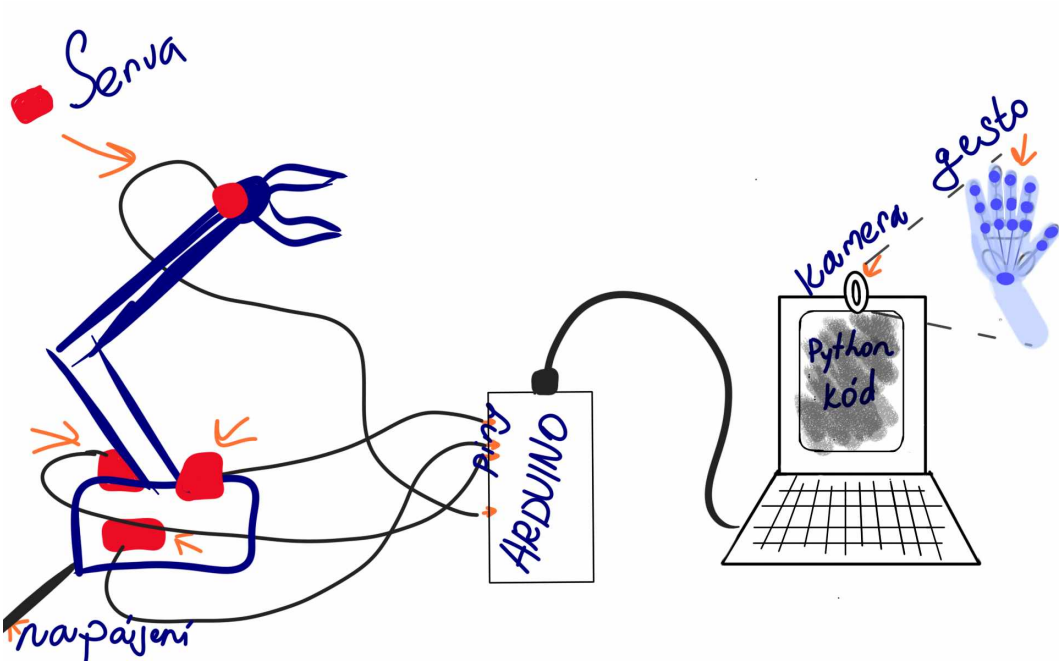
Rameno je ovládáno pomocí předdefinovaných gest zachycených kamerou notebooku. Původně byl také zvažován koncept úplného ovládání ramene tak, aby kopírovalo pohyb lidské ruky. Nicméně k realizaci tohoto nápadu by bylo nutné dokoupit Pohyblivý senzor Kinect pro Xbox, což by bylo spojeno s významnými finančními náklady. Z tohoto důvodu jsem se rozhodla od tohoto přístupu ustoupit.

Cílem projektu bylo zprovoznit pohyb ramene celkem jednoduchým ale také efektivním způsobem. Díky tomu jsem si skvěle procvičila Python a C++. Motivoval mě ten pocit že to rameno rozpohybuji.

V první kapitole naleznete seznam všech potřebných součástek pro sestavení samotného ramene. Samotnou fyzickou montáž jsem však neprováděla, a proto se budu věnovat zejména softwarovým aspektům. Tato část je doplněna krátkou teorií o využitých technologiích. V závěru kapitoly jsou uvedeny odkazy na všechny použité zdroje.

1 Princip fungování projektu

Zde je strukturovaný náčrt fungování celého projektu. Kamera zachytává specifické gesto, a pomocí kódu využívajícího knihoven převádí obraz na nezbytné hodnoty, které jsou následně rozpoznány robotickým ramenem. Tyto hodnoty jsou poté předány do Arduina, který následně přiřazuje odpovídající signály na určité piny, aby serva ramene byla schopna rozpoznat a vykonat požadovanou akci.



Obrázek 1.1: Princip fungování projektu.

2 HARDWARE

2.1 NETIŠTĚNÉ SOUČÁSTKY

Vzhledem k tomu, že jsem rameno nesestavovala, nemohu poskytnout detailní popis postupu montáže. Nicméně zde je výčet potřebných součástek:

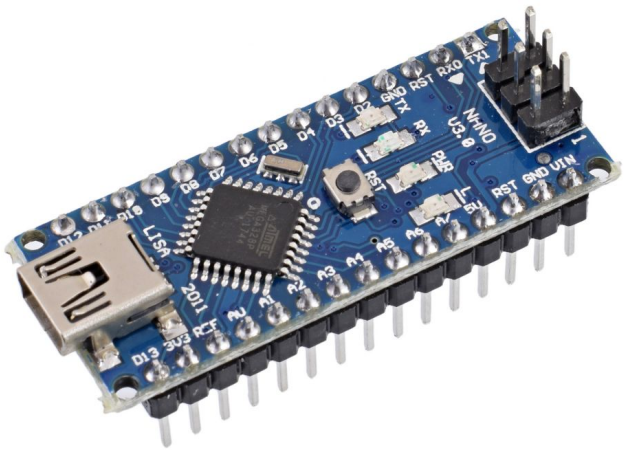
- 3 - 955 nebo 946 servo
- 1 - SG90 SERVO
- 1 - Samojistná matice M6
- 1 - Šroub M6x25
- 2 - Samojistné matice M3
- 2 - Šrouby M3 x 20
- 1 - Šroub M3 x 10 se šestihrannou hlavou
- 9 - Samojistné matice M4
- 1 - Šroub M4 x 40
- 1 - Šroub M4 x 30
- 5 - Šroub M4 x 20
- 1 - Závitová tyč M4 x 60mm
- 1 - Závitová tyč M4 x 32 mm
- 25 - Kulové koule o průměru 6 mm
- Ložisko 1 - 606zz
- Některé podložky M4

Návod na samotné sestavení ramene naleznete na tomhle odkazu

2.1.1 Arduino nano 328

popis

Arduino hraje klíčovou roli při oživení samotného hardwaru. Napsaný kód, který podrobněji popíšu v následující kapitole, je nahrán do mikrokontroléru pomocí USB kabelu. Díky správnému zapojení ramene se dosahuje bezproblémového fungování celého systému.

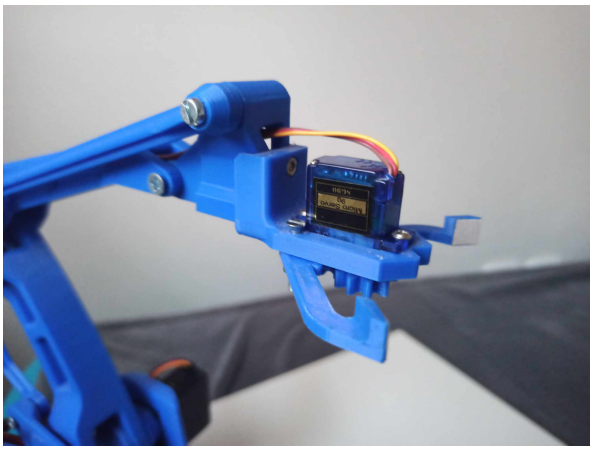


Obrázek 2.1: Arduino nano 328

2.1.2 Servomotory

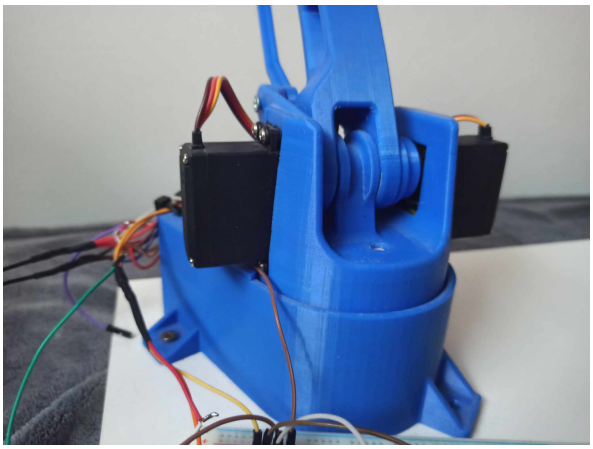
Popis

Toto rameno zahrnuje tři velká serva a jedno menší. Každé servomotor disponuje třemi výstupy: kabelem určeným pro specifický port, záporným a kladným pólem. Nejmenší servomotor je umístěno v kleštičkách.



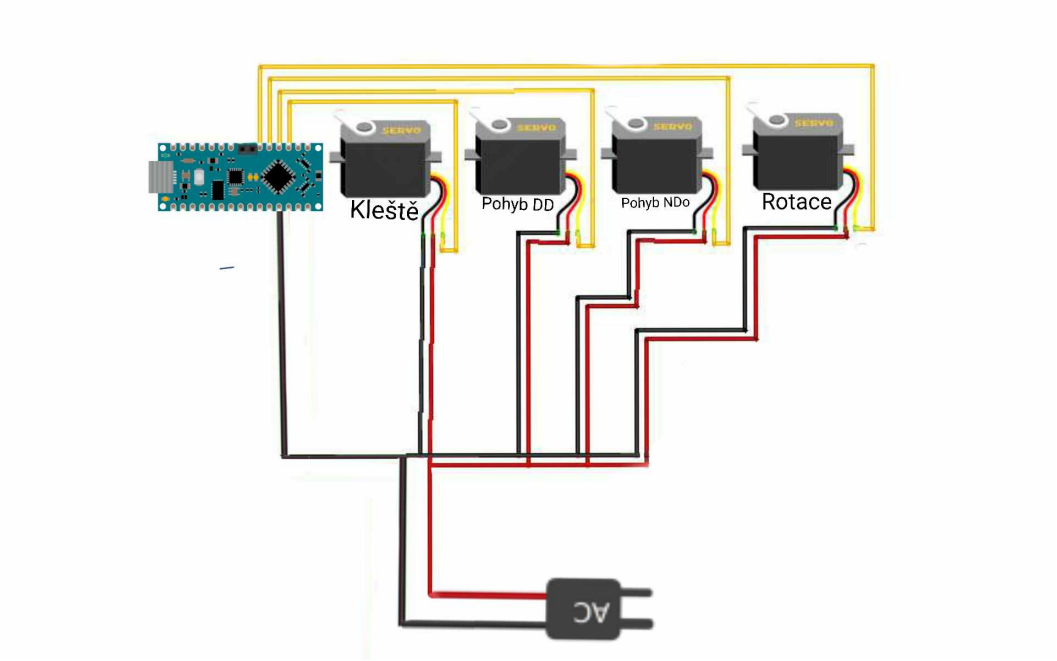
Obrázek 2.2: Servo nacházejíci se v kleštich ramene[**?**].

Zbylé tři se zabývají rotací a pohyby nahoru-dolů, dopředu-dozadu



Obrázek 2.3: Serva v těle ramene.

2.1.3 Obvod



Obrázek 2.4: Zapojení ramene.

3 SOFTWARE

3.1 ZÍSKÁNÍ GEST

Začneme importem knihoven, a to **OpenCV math** a **MediaPipe 0.10.8**

Knihovna OpenCV

OpenCV představuje multiplatformní svobodnou knihovnu určenou pro manipulaci s obrazem. Tato knihovna je významnou měrou implementována v programovacím jazyce C++, nicméně je dodávána s Python obalem, což umožňuje její použití i v prostředí Pythonu. OpenCV nalézá uplatnění při zpracování obrazu z kamer a provádění úloh, jako je rozpoznávání psaného textu či obličejů.

Osobně jsem tuto knihovnu využila pro celkové zpracování obrazu z integrované kamery mého notebooku.

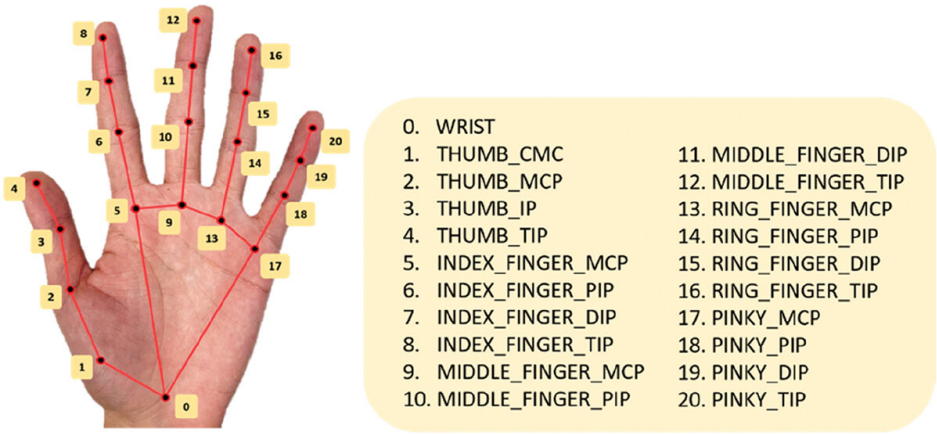
```
cap = cv2.VideoCapture(0)
...
image = cap.read()
imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
results = hands.process(imageRGB)
```

Kód 3.1: Obraz, a jeho převedení do správného formátu

Převede snímaný obraz do formátu RGB, který potřebujeme pro vykreslení pixelů ve správném pořadí.

Knihovna MediaPipe

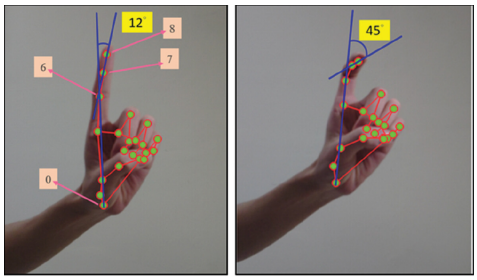
Tato knihovna je specializována na detekci lidských částí těla, přičemž jsem se v rámci svého projektu zaměřila výhradně na detekci dlaní. Pro účely této detekce vyžaduje knihovna orientační body ruky. Model orientačních bodů pro ruku je schopen předpovídat 21 přesných souřadnic pro umístění každého orientačního bodu na ruce.



Obrázek 3.1: Orientacni body dlaně.

Knihovna math

Ve spolupráci s výše zmíněnou knihovnou jsem získala souřadnice jednotlivých kloubů, což umožnilo přesné zachycení postavení ruky. Kromě toho jsem prováděla další výpočty pro určení vzdálenosti mezi prsty, což bylo nezbytné pro správné identifikování gesta potřebného k ovládání kleštiček.



Obrázek 3.2: Použití úhlu v gestech.

Obrázek 3.3: Použití měření vzdálenosti v gestech.

3.2 Ověření funkčnostní ramene

Předtím, než jsem mohla začít s implementací nápadu na pohyblivé rameno, bylo nezbytné ověřit jeho funkčnost a porozumět pohybu jednotlivých částí. K tomuto účelu jsem využila jednoduchý kód. Postupně jsem připojovala jednotlivé drátky k desce s Arduino a zjišťovala, který drát odpovídá za kterou část a jak se tato část pohybuje. Narazila jsem na jednu komplikaci, kdy servomotory stávkovaly, když jsem posílala extrémní hodnoty (0, 180). Po upravení rozsahu v intervalu ve funkci se vše dostalo do pořádku.

```
#include <Servo.h>

Servo myservo;

int pos = 0;
void setup() {
  myservo.attach(9);
}

void loop() {
  for (pos = 2; pos <= 160; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 160; pos >= 2; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

Kód 3.2: Ukázka kódu k ověření funkčnosti

3.3 Práce s Arduinem

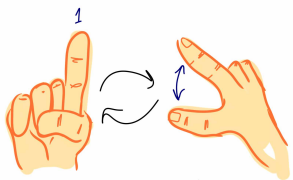
3.3.1 Inicializace hodnot

První věc, kterou musíme vzít v úvahu, je, že nezbytně potřebujeme mít na svém počítači program Python a také PySerial. Většinu kódu v Pythonu se dá sehnat na internetu a odkazy na ty stránky naleznete níže v použité literatuře. Takže se jdeme přesunout k popisu. Pro kód v C++ je na začátek potřeba inicializovat veškeré potřebné hodnoty. např:

```
1  const int min= 2;  // minimalní hodnota rozmezí pro vzdálenost
2  const int max = 160; // maximalní hodnota rozmezí pro vvzdálenost
3
4  bool end;
5  int data = 0;
6  int serialData = 0;
7  bool changeMode;
8  int mode;
9  const int servoPose = 6; //kolik stavu má rameno
10 const int servoCount = 4; // počet serv
11
12 Servo servo[4]; // pole pro serva
13 const byte servoPins [] = {2,3,4,5}; //piny pro serva
```

Kód 3.3: Inicializace potřebných hodnot

- Vzhledem k tomu, že pro řízení pohybu kleští potřebujeme hodnoty v rozmezí od 0 do 180, je nutné přemapovat posílanou vzdálenost. K tomu účelu využívám proměnné "min"a "max", které určují rozsah posílaných hodnot.
- Dále je tu proměnná "changeMode", která slouží k přepínání mezi režimem snímání počtu prstů a režimem měření vzdálenosti mezi prsty.



Obrázek 3.4: Dvá módy pro kontrolu ramene.

- V proměnné "servoPos"je uložen počet všech možných stavů, ve kterých se může nacházet mé rameno.
- Vzhledem k tomu, že má rameno 4 servomotory, vytvořila jsem pro ně pole. Každý servomotor vyžaduje svůj vlastní pin, který je následně specifikován v proměnné "servoPins".

```
if(j == (servoMoves[data][0] - 1)) {
  if(data == 0 || data == 1) {
    changeMode = true;
    mode = data;
  }
```

Kód 3.4: Podmínka pro přepínání módu

- Tato jednoduchá podmínka byla vytvořena pro kontrolu aktuální polohy ramene. Pokud se rameno nachází v horní nebo dolní poloze, dojde k přepnutí režimu pro měření vzdálenosti. To umožňuje manipulaci s ramenem, například pro chycení nebo pustení kuličky.

```
   int servoValues [13] [3] = {
    {20,50,20},  //puvodni stav
    {170,80,45}, //dole sběr kuličky cast1
    {123,113,45}, //dolu sběr kuličky cast2
    ...
    {123,130,133},  //uklona cast2
  };

```

Kód 3.5: Dvourozměrné pole na potřebné hodnoty

- Dále používáme dvourozměrné pole pro uložení hodnot pro servomotory. Hodnoty jsou vždy uspořádány ve třech sloupcích v jednom řádku, jelikož pro ovládání servomotorů v kleštích jsou potřebné odlišné hodnoty.

```
  int servoMoves [servoPose] [3] = {
    {2,1,500}, //dolu ke kulice 1
    {2,3,500}, //nahoru ke kulice 2
    {5,5,700}, //ne  3
  ...
```

```
};

```

Kód 3.6: Další pole pro několik hodnot

- Následující pole obsahuje několik různých hodnot. Na prvním místě je počet opakování pohybů, na druhém místě je pořadí hodnot pro servomotory z předchozího pole. Poslední hodnota v poli určuje časový odstup (delay).

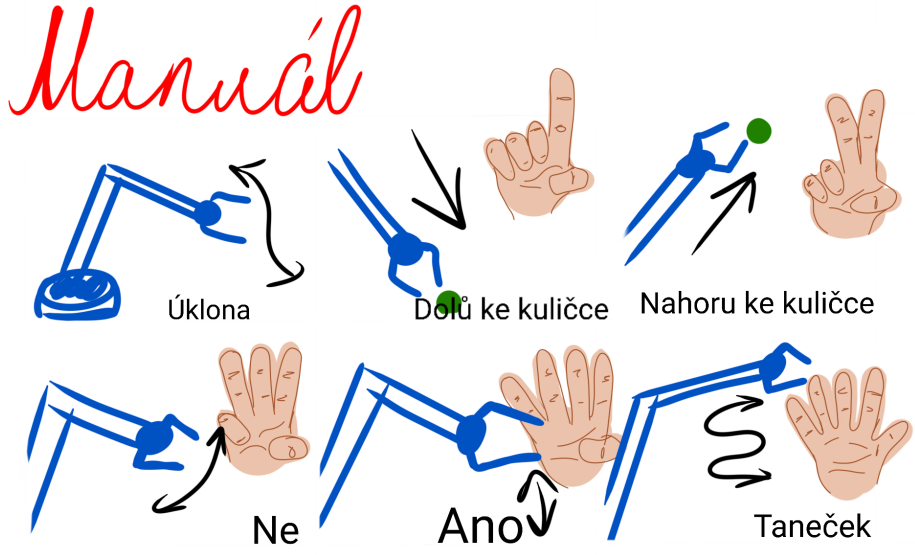
```
void move(int data) {
  int cycle = 0;
  for(int j = 0 ; j < servoMoves[data][0];j++) {
    for(int i = 1; i < servoCount;i++) servo[i].write(servoValues[
                    (servoMoves[data][1])+cycle][i - 1]);
    cycle++;
    if(cycle > 1) cycle=0;
    if(j == (servoMoves[data][0] - 1)) {
      if(data == 0 || data == 1) {
        changeMode = true;
        mode = data;
      }
      end = true;
      Serial.begin(9600);
    }
    delay(servoMoves[data][2]);
  }
}


```

Kód 3.7: Funkce na rozpohybocání ramene

- Jedná se o jednu z klíčových funkcí, která slouží k pohybu ramena. Jakmile je načtena jedna hodnota, funkce postupuje v cyklu o jeden prvek dál v poli "servoValues", dokud nedojde na konec. Poté se zkontroluje, zda jsou data rovna 0 nebo 1, což signalizuje, zda se rameno nachází v pozici u kuličky. V případě potvrzení této pozice se změní hodnota proměnné "changeMode"a přepne se na režim měření vzdálenosti mezi prsty.

3.3.2 Manuál na ovládání



Obrázek 3.5: Dvá módy pro kontrolu ramene.

ZÁVĚR

Cílem celé práce bylo dosáhnout pohybu ramene a získat lepší porozumění fungování příslušných knihoven. Ve finální verzi je rameno schopno úspěšně uchopit kuličku a přesunout ji z konce na začátek. Celý projekt byl implementován pomocí kombinace jazyků Python a C++. Možné využití tohoto projektu může být rozmanité, například v továrnách, skladech, v nanochirurgii nebo jako vzdělávací hračka.

Podařilo se mi dosáhnout původního cíle a rameno nyní pracuje tak, jak jsem si představovala na začátku vývoje. I přestože má můj kód určité nedostatky v optimalizaci, věřím, že by mohl být v budoucnu vylepšen. Rovněž plánuji detailněji studovat knihovnu OpenCV a prozkoumat další její funkce.

odkaz na github: https://github.com/Sofjaklopcova/maturita

SEZNAM POUŽITÝCH INFORMAČNÍCH ZDROJŮ

[1] Základní informace o gestech [Online]. 2023 CEMI MBA Studies s.r.o. [cit. 2020-08-24]. Dostupné z: https://www.cemi.cz/blog/neverbalni-komunikace-co-prozradi-gesta

[2] Typy gest v běžném životě [online]. 1998 [cit. 2020-08-24]. Dostupné z: https://orangeacademy.cz/clanky/rec-tela/

[3] Řeč tela a znakový jazyk, 2023, Slůně - svět jazyků, s.r.o. Dostupné z: https://www.slune.cz/aktualita/znakovy-jazyk/

[4] Knihovna math v pythonu Copyright © 2023 itnetwork.cz. https://www.itnetwork.cz/python/zaklady/python-tutorial-knihovny-math-a-random

[5] Knihovna opencv v pythonu Copyright © 2023 itnetwork.cz. https://www.python-programator.cz/clanky/python-pro-zpracovani-obrazu-opencv

[6] Komunikace s arduinem © 2023 Arduino https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756

[7] Komunikace s arduinem 2 Copyright © 2023 Circuit Digest. All rights reserved. https://circuitdigest.com/microcontroller-projects/arduino-python-tutorial

[8] Zkouška funkčnosti https://docs.arduino.cc/learn/electronics/servo-motors

[9] Zkouška funkčnosti https://problemsolvingwithpython.com/11-Python-and-External-Hardware/11.03-Controlling-an-LED/

Seznam obrázků