Interactive Digital Systems – Autumn 2021 – Hand-in part 2

Introduction

The aim of this hand in is to connect the library of p5 with the Serial Control to activate the USB port and give us access to the Arduino board. The p5 gives us access to use the camera so we can program it to collaborate with our hand and fingers. The way we use it is to interact with the light on the Arduino board depending on what hand signs you make. The tools we use is: p5.js, p5 Serial Control, Arduino and ESP32 – and we use it all to make the light on the board start and stop along with making hand signs.

ESP32

An ESP32 is a piece of hardware, constructed of a series of low-cost microcontrollers with integrated WI-FI and Bluetooth. We use the ESP32 to give visual feedback, in this case LED lights, to the actions carried out in the P5.js editor. The ESP32 cannot work on its own, and is programmed through the Arduino, however when this is done, the ESP32 is able to remember the given commands in Arduino, when Arduino is not active.

Arduino

Arduino is an open-source platform for hardware and software which among other things are used for programming the code for a connected board, which in our case is the ESP32. The code could for example be instructions to the ESP32 microcontroller on how to read a sensor for light or a button. These instructions are written in the Arduino Software IDE, which is based on the programming language called Processing, which is then send to the ESP32.

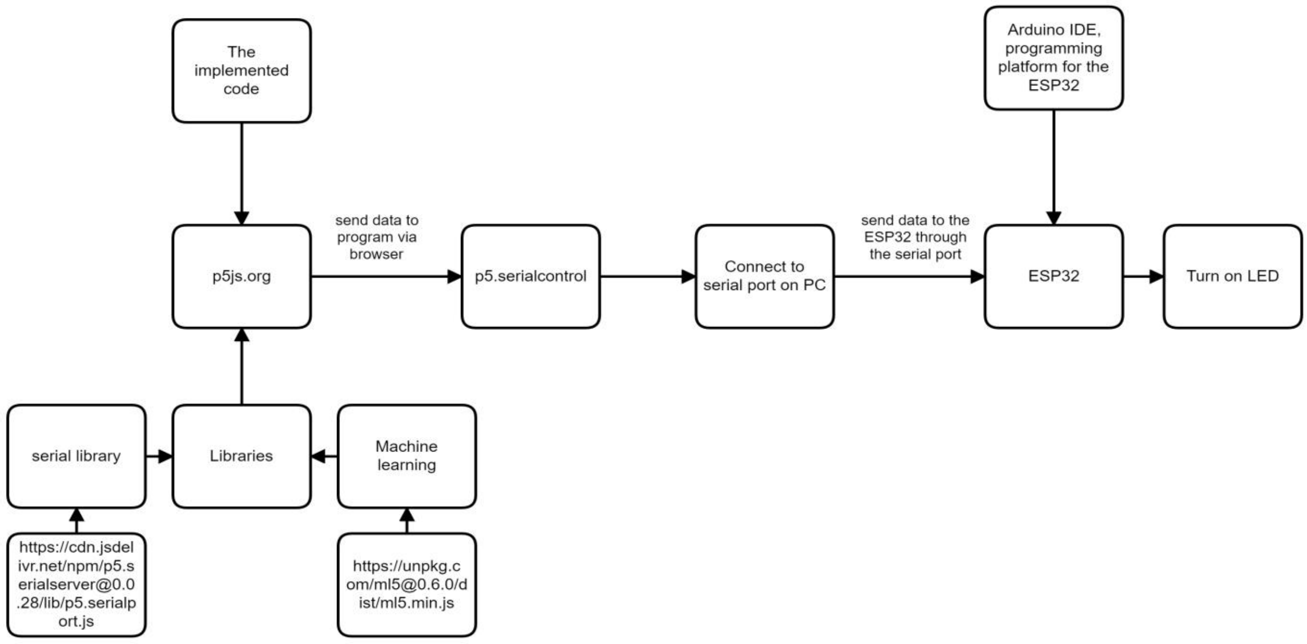
P5js

P5.js is a JavaScript library, which contains a variety of drawing functions. As described on the organizations website, "p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, beginners, and anyone else" (Ye & Masso, 2021). The P5.js editor allows the user to create visual code and enables both graphic visualization as well as on screen video interaction. It has therefore been a great tool in the hand-in, as it allowed us to implement the machine learning feature, by extending the code given in the IDS class. Here we updated the code to recognize a certain hand gesture, which will later be transmitted to the ESP32. We decided to extend the given p5 code instead of creating it from scratch as we did not find our solution fulfilling, within the given frame of experience and time.

P5 Serial Control application

It is not possible, in general, to access a computers serial port through a web browser. As the p5.js sketch is found on a web browser we must find a way to connect to the computers serial port (USB port) that is connected to the ESP32. In order to solve this issue, we use the P5 serial control application, as it can both serve HTML/JavaScript pages, as well as communicate with a serial port. The p5 serial controller is therefore used to connect the Arduino microcontroller to the P5.js sketch. This is done by using a websocket to serial.server, p5.serialserver and the p5.serialport library. Located in the index.html. As a result, the serial control application holds great value to the entirety of the system, as it works as the link between online and local. Allowing the ESP32 to give visual feedback from the on-screen video interaction in the p5.js sketch.

Model of system



The implemented code
Arduino IDE, programming platform for the ESP32
send data to program via browser
p5js.org
p5.serialcontrol
Connect to serial port on PC
send data to the ESP32 through the serial port
ESP32
Turn on LED
serial library
Libraries
Machine learning
https://cdn.jsdelivr.net/npm/p5.serialserver@0.0.28/lib/p5.serialport.js
https://unpkg.com/ml5@0.6.0/dist/ml5.min.js

As shown in the image and discussed in the text above, the system is created through 4 different units. The Arduino IDE, the ESP32, P5.js online editor and P5 serial control application. They are connected in the fashion where the P5.js editor gathers information from the implemented library and code. This is passed through the P5 serial control application to the ESP32. The ESP32 is programmed by the code in the Arduino IDE, allowing it to interact with the serial control application.

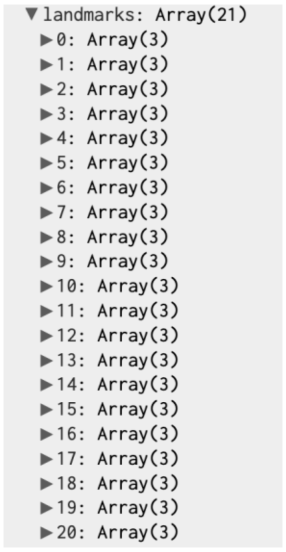
Machine learning:

The machine learning aspect of the hand in is dedicated to the p5.js sketch. We have used the handpose library that we were given in class and used it with our p5js sketch. The implementation of the library teaches the system to recognize the displayed fingers and points along them. This enables the system to recognize when the user is displaying a “finger gun” handpose. From that we can command the ESP32 to make the LED lights blink whenever this action is carried out. The library is located in index.html.

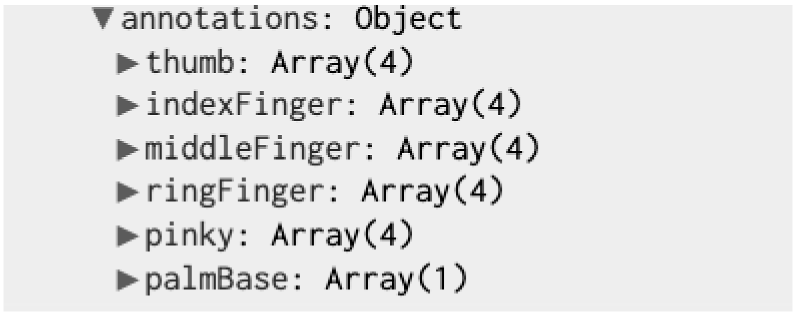
Data structures:

The ML program is coded and taught in a way, that the machine tries to find different point on a hand.

These points are defined by an x, y and z coordinate. Measuring width, height and depth. These points are saved in an array. The machine locates, in total, 21 sets of points on the hand and identifies these as “landmarks”. Following this, the points are sorted and divided into the categories, “palm”, “thumb”, “indexFinger”, “middleFinger”, “ringFinger” and “pinky”. Each of these categories contains 4 points and are saved as a new array, containing only these points. Excluding the “palm” category, since it only contains one point.



```
▼landmarks: Array(21)
  ▶0: Array(3)
  ▶1: Array(3)
  ▶2: Array(3)
  ▶3: Array(3)
  ▶4: Array(3)
  ▶5: Array(3)
  ▶6: Array(3)
  ▶7: Array(3)
  ▶8: Array(3)
  ▶9: Array(3)
  ▶10: Array(3)
  ▶11: Array(3)
  ▶12: Array(3)
  ▶13: Array(3)
  ▶14: Array(3)
  ▶15: Array(3)
  ▶16: Array(3)
  ▶17: Array(3)
  ▶18: Array(3)
  ▶19: Array(3)
  ▶20: Array(3)
```



```
▼annotations: Object
  ▶thumb: Array(4)
  ▶indexFinger: Array(4)
  ▶middleFinger: Array(4)
  ▶ringFinger: Array(4)
  ▶pinky: Array(4)
  ▶palmBase: Array(1)
```

Description of the code

This part of the hand in will focus on explaining the written code and simplifying the many lines. In the first function we define the window, using the createCanvas, where the video output will be inserted. Afterwards, we define several variables, such as the video, which contains the createCapture functions, which will be called later on in the draw function, but also the handpose variable, containing the trained machine program for detecting hand motions. Furthermore, the final 4 lines of the code beneath, is where we initialize the P5 serial control, and set up the link between the web browser editor and the serial control application.

```
function setup() {
  createCanvas(640, 480);
  video = createCapture(VIDEO);
  video.hide();
  handpose = ml5.handpose(video, modelReady);
  serial = new p5.SerialPort();
  serial.open("COM3");
  serial = new p5.SerialPort();
  serial.open(serialPortName);
}
```

The function *drawHand* is called if hands (and the length of the hands) are detected in an if statement down in ourdraw function. Moreover, the landmarks, meaning the different points along each finger, are displayed through an ellipse, to give a clear visual understanding of the fingers position.

```
function drawHand(hand) {
  let landmarks = hand.landmarks;
  colorMode(RGB);
  fill(255, 100);
  stroke(255);
  for (let i = 0; i < landmarks.length; i++) {
    let [x, y, z] = landmarks[i];
    ellipse(x, y, 24);
```

The part with the hand and fingers continues with “averageDepth”, which is the z value of the thumb and the index finger divided with 2. This is used to activate the if statement regardless of the hands position as long it is in front of the camera and contained inside the edges of the camera. Thereafter sends data to the serial control if the hand is inside the positions.

```
  var averageDepth =
    (hand.annotations.thumb[3][2] + hand.annotations.indexFinger[3][2]) / 2;
  if (
    hand.annotations.thumb[3][1] < hand.annotations.indexFinger[3][1] - 2.0 &&
    hand.annotations.thumb[3][0] >
      hand.annotations.indexFinger[3][0] + averageDepth
  ) {
    data = 100;
    serial.write(data);
    console.log(data);
  }
```

The next part shows the fingers and how they are connected. It draws the hand and marks the lines along the finger in 5 different colors.

```
  let annotations = hand.annotations;
  let parts = Object.keys(annotations);
  let count = 0;
  for (let part of parts) {
    for (let position of annotations[part]) {
      colorMode(HSB);
      stroke(count * 20, 255, 255);
      count++;
      strokeWeight(8);
      noFill();
      beginShape();
      for (let position of annotations[part]) {
        let [x, y, z] = position;
        vertex(x, y);
      }
      endShape();
    }
```

The draw function is called every time the framerate is refreshed. It contains several functions, such as translate(), which defines the webcam camera angle, scale() which defines how much the videofeed is zoomed. The image() which is used to display the video feed. Lastly, we have the hands if statement, drawing the hand if hands and hands.length are larger than 0, meaning that they contain a value larger than 0. It is also here that we use the serial functions to send data to the ESP32 through the serial controller application.

```
function draw() {
  translate(width, 0);
  scale(-1, 1);
  background(150);
  data = 0;
```

```
 serial.write(data);
 console.log(data);
 if (video) {
   image(video, 0, 0);
 }
 if (hands && hands.length > 0) {
   let hand = hands[0];
   drawHand(hand);
 }
```

The following piece of code is what we use from the serial library, which we use, in order to establish connection between our web browser and the serial port. In our case, the serial port name is “COM3”, hence the reason why we use it. If others want to use the code, they will have to import the library and change the serialPortName variable to the name of their serial port.

```
var serialPortName = "COM3";
serial.open(serialPortName);
serial.write(data);
```