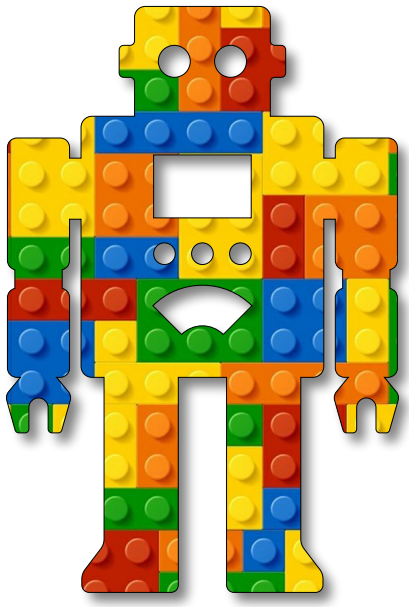
Intelligent Machines Module E Unit #3 finger servo



Content

Module E Unit #3 finger servo

Introduction to finger servo

In this unit we will control a servo using your index finger. This uses the pretrained model handPose to identify the index finger and map it across the canvas.

The position will be sent to the Arduino Nano 33 BLE which is connected to the servo. The servo rotates as your finger moves across the canvas.

! The port.js and the Arduino sketch remain the same.

Sketch E3.1 basic servo sketch

The same sketch as unit #2.

```
#include <Servo.h>
Servo myServo;
int val = 0;

void setup()
{
  myServo.attach(2);
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0)
  {
    val = Serial.read();
    myServo.write(val);
  }
}
```

Notes

All things the same.

Sketch E3.2 index.html

We have covered the use of pretrained models in previous modules at length. So, we won't jumping into this deeply but will still build up gradually. To use a pretrained model we will need to add the library to our index.html file shown below.

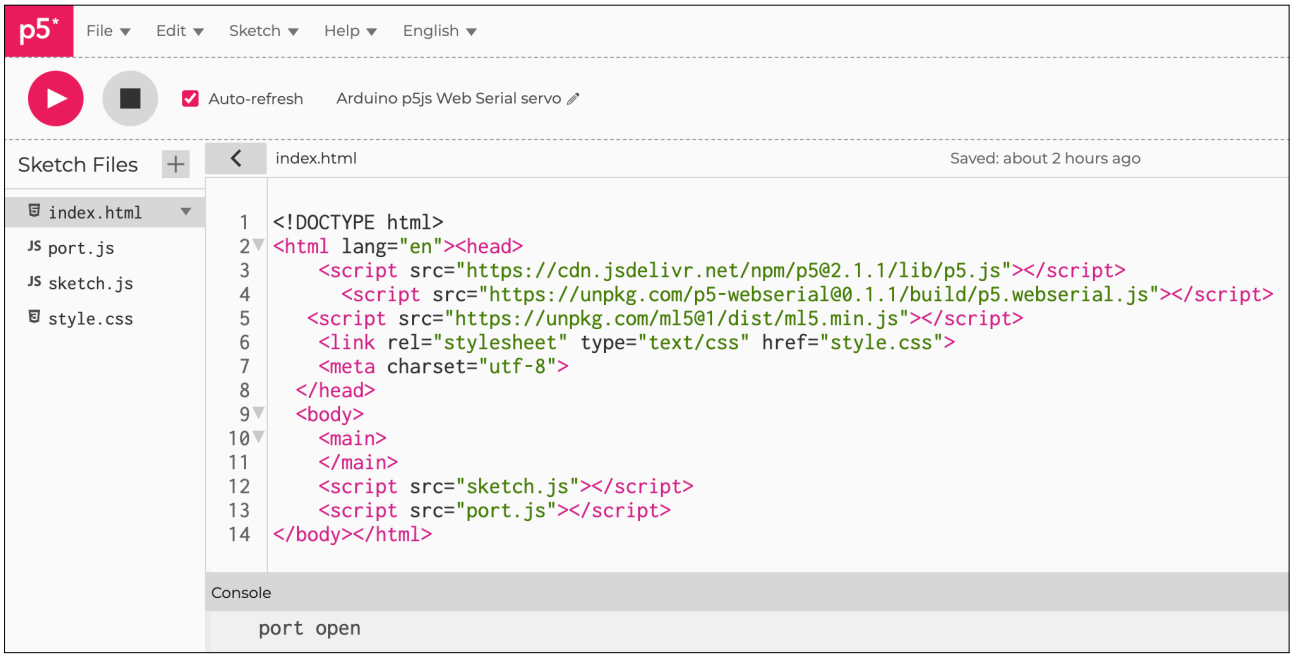
index.html

```
<!DOCTYPE html>
<html lang="en"><head>
    <script src="https://cdn.jsdelivr.net/npm/p5@2.1.1/lib/p5.js"></script>
      <script src="https://unpkg.com/p5-webserial@0.1.1/build/
p5.webserial.js"></script>
   <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">
  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
    <script src="port.js"></script>
</body></html>
```

Notes

We have the three libraries: p5.js, ml5.js and p5.webserial.js.

index.html


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Arduino p5js Web Serial servo
Sketch Files
index.html
port.js
sketch.js
style.css
index.html
Saved: about 2 hours ago
1 <!DOCTYPE html>
2 <html lang="en"><head>
3 <script src="https://cdn.jsdelivr.net/npm/p5@2.1.1/lib/p5.js"></script>
4 <script src="https://unpkg.com/p5-webserial@0.1.1/build/p5.webserial.js"></script>
5 <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
6 <link rel="stylesheet" type="text/css" href="style.css">
7 <meta charset="utf-8">
8 </head>
9 <body>
10 <main>
11 </main>
12 <script src="sketch.js"></script>
13 <script src="port.js"></script>
14 </body></html>
Console
port open

Sketch E3.3 in the beginning

Our starting sketch.

sketch.js

```
let val = 0

function setup()
{
  createCanvas(400, 400)
  navigation()
}

function draw()
{
  background(220)
  serial.write(val)
}
```

Sketch E3.4 getting the video

As I have said, I covered this bit in greater depth in the AI tutorial series. At this point we are using the webcam, taking the video, putting it in the canvas. The flipped part is to make it like a mirror rather than reversed.

sketch.js

```
let val = 0
let video

function setup()
{
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  navigation()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  serial.write(val)
}
```

Notes

You should have mirrored image of yourself on the canvas with a size of 640 x 480 to accommodate the dimensions of the webcam image.

Sketch E3.5 the handPose model

In version 2 of p5.js which I encourage you to start using (but not essential) we need to load pretrained model before we doing anything with it, so we add in async and await into the setup function. This means it is synced and waits for the model to load before doing anything else. This is crucial. The model we will be using is the handPose model.

sketch.js

```
let val = 0
let video
let handPose
let hands = []

async function setup()
{
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  navigation()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  serial.write(val)
}

function gotHands(results)
{
  hands = results
}
```

🗒️ Notes

The `handPose` model is part of `ml5.js`. We wait for it to be uploaded. We then use the video feed to detect `startDetect()` your hands. If it finds any then sends the information (`results`) to another function `gotHands()` as a callback. The results are stored in an array called `hands`. You won't see anything yet as we are only setting it up, we haven't used the information yet.

Sketch E3.6 now to see

Here we will now draw a circle where the tip of your index finger is. As you move your index finger the circle will follow it. We need the information from handPose (in the hands array) which identifies the tip of the index finger. In short this is keypoint 8 in the array. We get the x and y co-ordinates for that index value. Then map them as the co-ordinates of the circle. The x component is then sent to the Arduino.

sketch.js

```
let val = 0
let video
let handPose
let hands = []

async function setup()
{
  handPose = await ml5.handPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.hide()
  handPose.detectStart(video, gotHands)
  navigation()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  if (hands.length > 0)
  {
    fill('yellow')
    let index = hands[0].keypoints[8]
    circle(index.x, index.y, 20)
    val = map(index.x, 0, 640, 0, 180)
    serial.write(val)
  }
}

function gotHands(results)
```

```
{
  hands = results
}
```

🗒️ Notes

You should see the circle follow your index finger. Make sure it can see your full hand (only one hand visible). Move it slowly from left to right and back again so that the circle goes from one edge of the canvas to the other.

🌻 Challenge

Use two servos and track the y value with the other servo.

🛠️ Code Explanation

`if (hands.length > 0)`	Checks to see if you have any hands
`let index = hands[0].keypoints[8]`	If it finds a hand it then finds the index finger tip [8] and uses its co-ordinates [0]
`circle(index.x, index.y, 20)`	Draws the circle to the x, y co-ordinates
`val = map(index.x, 0, 640, 0, 180)`	Maps the x co-ordinate to the horizontal size of the canvas. 0 is 0, 640 is now 180
`serial.write(val)`	Sends the 0-180 value (val) to the Arduino