Alat Pendeteksi Masker Berbasis Mikrokontroler Arduino UNO

Salma Haya Amalia[1], Yazid Aqsa Raisnaldi[2] & Yuda Dian Febriansyah[3]

[1,2,3]Universitas Singaperbangsa Karawang, Indonesia
Email: [1]2010631170117@student.unsika.ac.id, [2]2010631170128@student.unsika.ac.id, [3] 2010631170159@student.unsika.ac.id

Abstrak. Seluruh dunia sedang digemparkan dengan suatu jenis virus yang telah menyebar hingga ke penjuru negeri. Virus ini pertama kali berasal dari kota Wuhan, China dan bernama Corona Virus Disease 2019 (Covid-19). Kemudian pemerintah menganjurkan untuk menerapkan protokol Kesehatan, salah satunya dengan menggunakan alat pelingdung (APD) pada saat keluar rumah. Alat Pelindung Diri (APD) adalah suatu alat yang mempunyai kemampuan untuk melindungi seseorang yang fungsinya mengisolasi sebagian atau seluruh tubuh dari potensi bahaya di tempat kerja. Salah satu jenis APD yang sering digunakan dan dianjurkan selama masa pandemic corona ialah masker. Penggunaan masker memang harus dilakukan untuk mengantisipasi penyebaran virus apalagi semenjak diterapkannya new normal. Maka dari itu kami bermaksud untuk membuat suatu inovasi untuk mendeteksi orang menggunakan masker atau tidak dengan menggunakan kamera sebagai sensor yaitu alat pendeteksi masker berbasis Arduino UNO. Tujuan diciptakannya alat ini adalah untuk membedakan orang yang memakai masker dengan yang tidak memakai masker, dan berguna sebagai peringatan untuk orang yang tidak memakai masker untuk menggunakan maskernya agar meminimalisir penularan penyakit virus covid-19 di masa pandemic seperti saat ini.

Kata kunci: covid-19, masker, sensor, Arduino UNO

Abstract. *The whole world is being shaken by a type of virus that has spread to all corners of the country. This virus first came from the city of Wuhan, China and was named Corona Virus Disease 2019 (Covid-19). Then the government recommends implementing Health protocols, one of which is by using protective equipment (PPE) when leaving the house. Personal Protective Equipment (PPE) is a tool that has the ability to protect a person whose function is to isolate part or all of the body from potential hazards in the workplace. One type of PPE that is often used and recommended during the corona pandemic is a mask. The use of masks must indeed be done to anticipate the spread of the virus, especially since the implementation of the new normal. Therefore, we intend to make an innovation to detect people using masks or not by using a camera as a sensor, namely an Arduino UNO-based mask detection tool. The purpose of the creation of this tool is to distinguish people who wear masks from those who do not wear masks, and serve as a warning for people who do not wear masks to use their masks in order to minimize the transmission of the Covid-19 virus disease during a pandemic like today.*

Keywords: *covid-19, mask, sensor, Arduino UNO*

1 Pendahuluan

Indonesia sekarang ini sedang mengalami masa kritis karena virus corona. Juru bicara pemerintah Indonesia mengatakan pada tanggal 17 Juli 2020 jumlah kasus positif corona di Indonesia mencapai 83.130 kasus. World Health Organization (WHO) menjelaskan penularan corona melalui kontak langsung dengan penderita, tetesan, udara, fomite, fecal-ral, darah, ibu ke anak, dan penularan dari hewan ke manusia. Oleh sebab itu untuk mengurangi jumlah kasus positif corona maka dianjurkan untuk menerapkan protokol Kesehatan, salah satunya dengan menggunakan alat pelingdung (APD) pada saat keluar rumah. Alat Pelindung Diri (APD) adalah suatu alat yang mempunyai kemampuan untuk melindungi seseorang yang fungsinya mengisolasi sebagian atau seluruh tubuh dari potensi bahaya di tempat kerja. Menteri Tenaga Kerja dan Transmigrasi nomor PER.08/MEN/VII/2010. PP 88 tahun 2019 tentang Kesehatan Kerja juga ditujukan untuk melindungi setiap orang yang berada di tempat kerja agar hidup sehat dan terbebas dari gangguan Kesehatan serta pengaruh buruk yang diakibatkan dari pekerjaan.

Salah satu jenis APD yang sering digunakan dan dianjurkan selama masa pandemic corona ialah masker. Masker adalah kain penutup mulut dan hidung yang sering digunakan oleh dokter atau perawat di rumah

sakit. Masker digunakan untuk melindungi pernafasan dari resiko paparan gas uap, debu, atau udara terkontaminasi atau beracun, korosi atau yang bersifat rangsangan terhadap saluran pernafasan.

Penggunaan masker memang harus dilakukan untuk mengantisipasi penyebaran virus apalagi semenjak diterapkannya new normal. New normal adalah langkah percepatan penanganan COVID-19 dalam bidang Kesehatan, sosial, dan ekonomi. Skenario new normal dijalankan dengan mempertimbangkan kesiapan daerah dan hasil riset epidemiologis di wilayah terkait. Semenjak new normal diterapkan aktivitas masyarakat sudah seperti sedia kala hanya saja perlu tambahan menggunakan masker. Maka dari itu dengan masalah yang ada kami bermaksud untuk membuat suatu inovasi untuk mendeteksi orang menggunakan masker atau tidak dengan menggunakan kamera sebagai sensor yaitu alat pendeteksi masker berbasis Arduino UNO. Tujuan diciptakannya alat ini adalah untuk membedakan orang yang memakai masker dengan yang tidak memakai masker, dan berguna sebagai peringatan untuk orang yang tidak memakai masker untuk menggunakan maskernya agar meminimalisir penularan penyakit virus covid-19 di masa pandemic seperti saat ini.

2 Teori Dasar

Berikut teori dasar yang kami gunakan dalam pembuatan projek ini diantaranya adalah:

1) Arduino UNO
   Ma'arif (2016) menjelaskan bahwa Arduino adalah sebuah papan mikrokontroler dengan sebuah aplikasi untuk pemrogramannya. Arduino menggunakan processor atmel AVR yang didukung dengan modul sebagai proses input dan output dengan bantuan alat sebagai hasilnya modul I/O. Untuk melakukan pemrograman sebuah Arduino, sudah tersedia perangkat lunak Arduino yang dilengkapi dengan kumpulan *library* sehingga dapat mempermudah untuk melakukan pemrograman.
   Arduino Uno adalah papan sirkuit berbasis mikrokontroler ATmega 328. IC (*integrated circuit*) ini memiliki 14 input/output digital (6 output untuk PWM), 6 analog input, resonator kristal keramik 16 MHz, koneksi USB, soket adaptor, pin header ICSP, dan tombol reset.
   Arduino UNO ini digunakan sebagai pengendali utama untuk menguji fungsi pembacaan masukan analog dengan melibatkan aktivitas perancangan antarmuka perangkat keras dan pemrograman perangkat lunak.

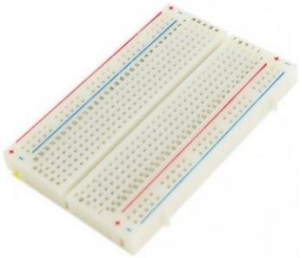
Gambar 2.1 Arduino UNO

2) Buzzer
   Buzzer adalah sebuah komponen elektronika yang berfungsi untuk mengubah getaran listrik menjadi getaran suara. Pada dasarnya prinsip kerja buzzer hampir sama dengan loud speaker, jadi buzzer juga terdiri dari kumparan yang terpasang pada diagfragma dan kemudian kumparan tadi akan tertarik ke dalam atau keluar, tergantung dari arah arus polaritas magnetnya, karena kumparan dipasang pada diagfragma maka setiap garakan kumparan akan menggerakkan diagfragma secara bolak-balik sehingga membuat udara bergetar yang akan menghasilkan suara. Buzzer bisa digunakan sebagai indikator bahwa proses telah selesai atau terjadi suatu kesalahan pada sebuah alat (alarm).

Gambar 2.2 Buzzer

3) Papan Bread Board

Sarmidi & Sidik Ibnu Rahmat (2019) mengatakan bahwa papan ini sendiri merupakan sebuah papan yang dijadikan papan uji coba rangkaian elektronika atau menjadi salah satu penghubung atau menghubungkan antara perangkat-perangkat elektronika. Papan ini memiliki konstruksi berlubang yang dapat digunakan sebagai tempat untuk menancapkan kabel atau alat-alat elektronika lain yang akan dipakai tanpa harus terhubung permanen.



Gambar 2.3 Papan Bread Board

4) Kabel Jumper

Mulyani (2018) menjelaskan bahwa kabel jumper menjadi alat penguhubung atau sebagai penghubung antara perangkat yang digunakan. Kabel jumper ini sendiri memiliki 3 jenis yaitu female-female, female-male, dan male-male.

Gambar 2.4 Kabel Jumper

5) Lampu LED

Light Emitting Diode atau sering disingkat dengan LED adalah komponen elektronika yang dapat memancarkan cahaya monokromatik ketika diberikan tegangan maju. LED merupakan keluarga dioda yang terbuat dari bahan semikonduktor. Warna-warna cahaya yang dipancarkan oleh LED tergantung pada jenis bahan semikonduktor yang dipergunakannya. LED juga dapat memancarkan sinar infrared yang tidak tampak oleh mata seperti yang sering kita jumpai pada remote control tv ataupun remote control perangkat elektronik lainnya.

Gambar 2.5 Lampu LED

6) Kabel Mini USB
Kabel Data Mini USB ini biasa digunakan sebagai kabel untuk transfer data antar dua perangkat dan sebagai kabel untuk pemrograman Arduino yang memiliki soket Mini USB seperti Arduino Nano standar. Selain itu kabel mini usb ini juga bisa dipakai untuk kabel supply Arduino tersebut melalui charger HP.

Gambar 2.6 Kabel Mini USB

7) Webcam
Webcam adalah kependekan dari web camera yang merupakan perangkat kamera digital untuk dihubungkan ke komputer atau laptop. Dengan webcam, maka gambar kita bisa tertangkap secara live kepada siapa pun di berbagai penjuru dunia. Tentunya dengan jaringan internet serta aplikasi yang juga kita gunakan.

Gambar 2.7 Webcam

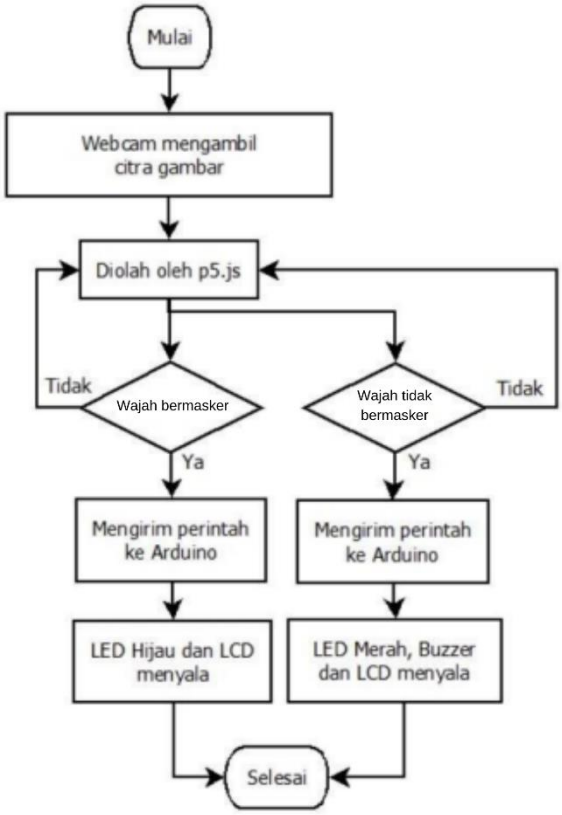
3 Metode

Beberapa metode penelitian yang kami gunakan dalam penelitian ini adalah:

1. Identifikasi masalah
   Tahap ini merupakan tahapan awal sebelum memulai penelitian, dimana dilakukan identfiikasi terhadap masalah yang ada.
2. Pengumpulan data
   Tahapan selanjutnya yaitu mengumpulkan data-data yang dibutuhkan untuk menyelesaikan masalah yang telah diidentifikasi, seluruh data tersebut akan digunakan untuk mendapatkan solusi dari permasalahan yang ada.
3. Menyusun skema alat
   Kemudian tahap berikutnya yaitu Menyusun skema alat dari pendeteksi masker, tujuan dari peracangan ini yaitu untuk memperoleh skema prototype dalam bentuk 3D dan dalam bentuk skema rangkaian.
4. Perancangan alat
   Setelah rangkaian selesai dibuat, langkah selanjutnya yaitu melakukan implementasi terhadap alat pendeteksi masker.
5. Implementasi
   Setelah rangkaian selesai dibuat, langkah berikutnya yaitu melakukan implementasi terhadap alat pendeteksi masker.
6. Hasil penelitian

Tahapan terakhir yaitu hasil dari pengujian tersebut dianalisa untuk mendapatkan hasil yang diharapkan.

3.1 Flowchart



Gambar 3.1.1 Flowchart

Alur kerja sistem pengingat penggunaan masker otomatis yaitu Webcam mendeteksi gambar wajah seseorang dan mengambil data tersebut. Kemudian data tersebut diproses pada script p5.js dan tentunya sudah disambungkan dengan website Teachable Machine. Data gambar diklasifikasikan ke 2 data, yaitu data 1 dan 2. Data 1 berisi gambar bermasker, sedangkan data 2 berisi gambar tidak bermasker. Script p5.js akan mengolah apakah data tersebut termasuk data 1 atau data 2. Setelah hasil didapat, script p5.js mengirim perintah pada Arduino IDE melalui p5.js serial control, kemudian diteruskan pada hardware. Jika gambar diklasifikasikan ke data 1, maka LED hijau dan LCD akan menyala, sedangkan jika data gambar diklasifikasikan ke data 2, maka buzzer, LED merah, dan LCD akan menyala.

3.2 Algoritma

Algoritma Dynamic Programming adalah suatu metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (step) atau tahapan (stage) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Dynamic programming dapat didefiniskan sebagai suatu pendekatan matematik yang memiliki prosedur sistematis yang dirancang sedemikian rupa dengan tujuan untuk mengoptimalkan penyelesaian suatu masalah tertentu yang diuraikan menjadi sub-sub masalah yang lebih kecil yang terkait satu sama lain dengan tetap memperhatikan kondisi dan batasan permasalahan tersebut.

Struktur dynamic programming untuk dapat dimengerti secara lebih jelas dan lebih spesifik, umumnya dideskripsikan dengan suatu sistem notasi. Struktur dynamic programming disebut juga dengan model dynamic programming.

3.2.1 Kelebihan Dynamic programming

Terdapat beberapa kelebihan pada Dynamic Programming diantaranya:

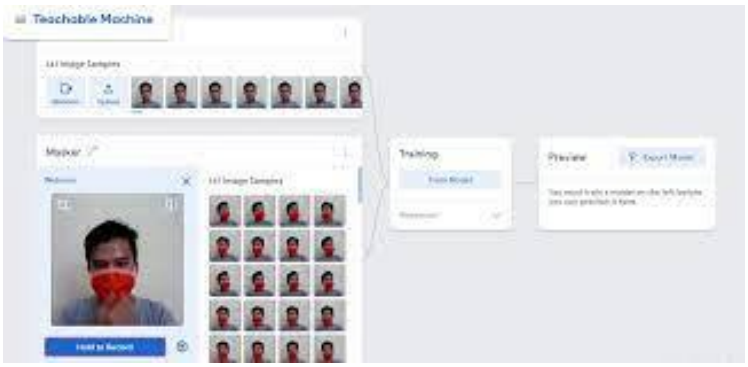
- Proses pemecahan suatu masalah yang kompleks menjadi sub-sub masalah yang lebih kecil membuat sumber permasalahan dalam rangkaian proses masalah tersebut menjadi lebih jelas untuk diketahui.
- Pendekatan Dynamic Programming dapat diaplikasikan untuk berbagai macam masalah pemrograman matematik, karena Dynamic Programming cenderung lebih fleksibel dari pada teknik optimasi lain.
- Prosedur perhitungan Dynamic Programming juga memperkenankan bentuk analisis sensitivitasi terdapat pada setiap variabel status (state) maupun pada variabel yang ada di masing-masing tahap keputusan (stage).
- Dynamic Programming dapat menyesuaikan sistematik perhitungannya menurut ukuran masalah yang tidak selalu tetap dengan melakukan perhitungan satu per satu secara lengkap dan menyeluruh.

3.2.2 Kekurangan Dynamic Programming

Di samping memiliki kelebihan, Dynamic Programming juga memiliki beberapa kekurangan, diantaranya:

- Penggunaan Dynamic Programming jika tidak dilakukan secara tepat, akan mengakibatkan ketidakefisienan biaya maupun waktu. Karena dalam menggunakan Dynamic Programming diperlukan keahlian, pengetahuan, dan seni untuk merumuskan suatu masalah yang kompleks, terutama yang berkaitan dengan penetapan fungsi transformasi dari permasalahan tersebut.
- Dynamic Programming tidak memiliki suatu bentuk formulasi matematik yang baku untuk digunakan secara konsekuen, sehingga perhitungan untuk menghasilkan keputusan optimal yang dilakukan terbatas pada kondisi tertentu.
- Hambatan terbesar pada Dynamic Programming adalah masalah dimensionalitas, yaitu masalah dimana peningkatan variabel keadaan yang digunakan dalam perhitungan pemrograman dinamis akan menambah beban memori komputer serta menambah lama waktu perhitungan.

Pada *project* kali ini, penerapan algoritma Dynamic Programming diterapkan pada aplikasi website yaitu Teachable Machine.



Gambar 3.2.2.1 Teachable Machine

Teachable Machine merupakan alat berbasis web yang membuat pembuatan model pembelajaran mesin menjadi cepat, mudah, dan dapat diakses oleh semua orang. Teachable Machine adalah alat yang dibuat oleh Google, stoj dan yang menggunakan semua fungsi, dengan bantuan TensorFlowJS, untuk membuat pembelajaran mesin dan kecerdasan buatan tersedia untuk semua orang.

Biasanya algoritma pembelajaran mesin dikembangkan menggunakan kerangka kerja berbeda yang dikhususkan untuk pembelajaran mesin, seperti Pytorch atau TensorFlow.

3.3 Devices

Beberapa devices yang digunakan dalam pembuatan projek ini adalah:

- Arduino UNO
  Arduino ini digunakan sebagai komponen utama untuk menjalankan sensor deteksi masker.
- Buzzer
  Buzzer digunakan sebagai alarm apabila sensor mendeteksi orang yang tidak menggunakan masker.
- Papan Bread Board
  Papan ini digunakan sebagai alas untuk nanti digunakan sebagai penghubung dari setiap bahan dalam perangkaian projek.
- Kabel Jumper
  Kabel ini digunakan untuk menghubungkan komponen sat uke komponen lainnya agar rangkaian berhasil dibuat dan dijalankan.
- Lampu LED
  Disini digunakan LED berwarna merah dan hijau. LED akan menyala berwarna merah apabila orang tidak memakai masker dan akan menyala berwarna hijau apabila orang tersebut memakai masker.
- Kabel Mini USB
  Kabel ini digunakan untuk menghubungkan antara Arduino dengan laptop.
- Webcam
  Webcam ini digunakan sebagai sensor pendeteksi masker.

4 Proses *Wiring*

Cara mudah agar format makalah Anda sesuai dengan format makalah yang kami perlukan, gunakan dokumen ini sebagai template dan ketik teks Anda di dalamnya.

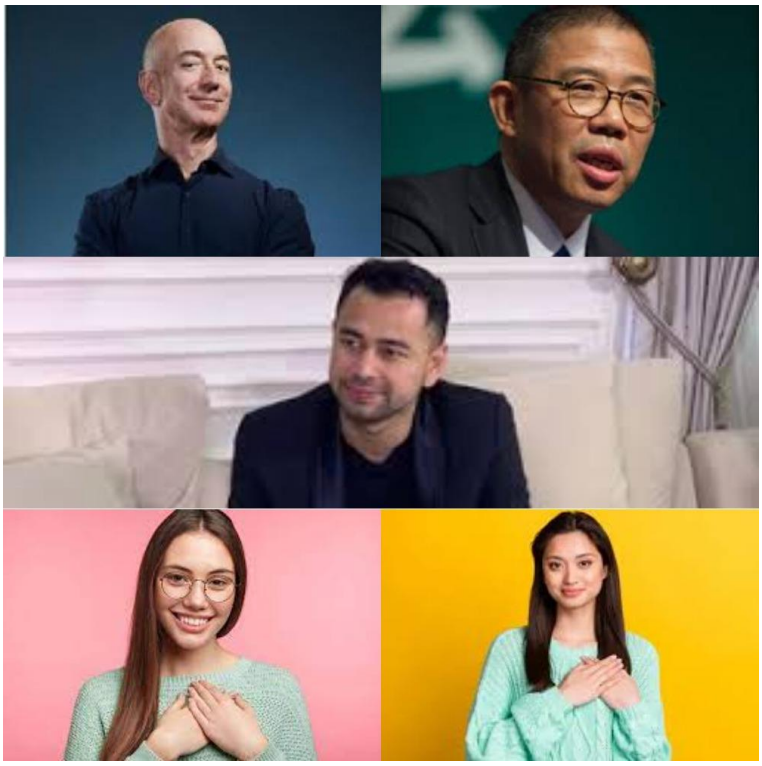
Tabel 1. Device yang digunakan dalam pembuatan projek

NO	Komponen	Jumlah
1	Arduino UNO	1
2	Mini Beardboard	1
3	Lampu LED	1
4	Buzzer	1
5	Kabel Jumper	4
6	Web cam	1
7	Kabel mini USB	1

Setelah disebutkan beberapa devices yang digunakan dalam pembuatan projek, selanjutnya adalah tahapan dalam pembuatan projek. Berikut langkah-langkahnya.

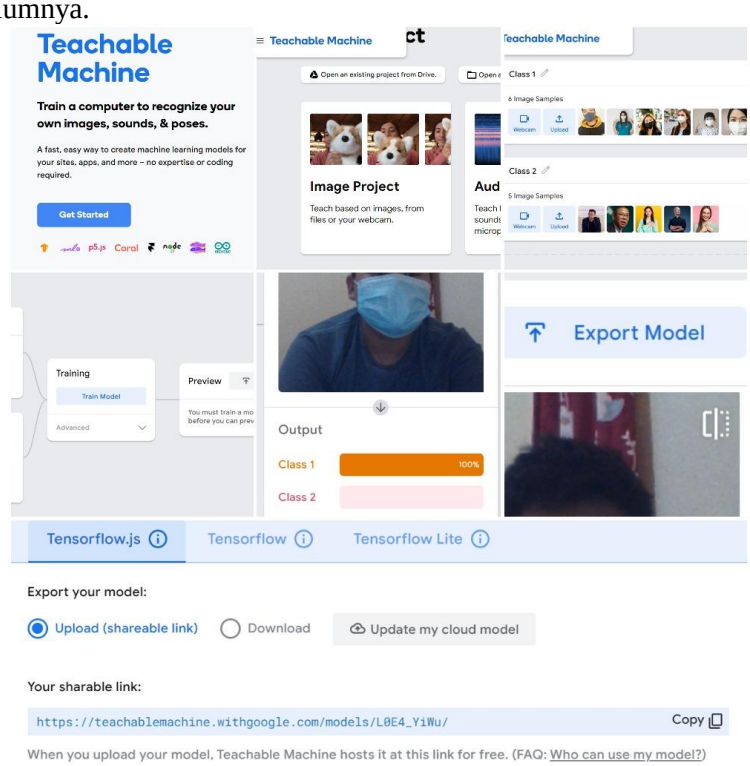
1. Kumpulkan data gambar / foto orang yang menggunakan masker atau tidak.

Gambar 4.1 Orang menggunakan masker



Gambar 4.2 Orang tidak menggunakan masker

2. Buat dan *upload* AI model di https://teachablemachine.withgoogle.com/. Caranya setelah masuk ke dalam website tersebut, klik Get Started lalu pilih Image Project, selanjutnya pilih Standard image model. Kemudian setelah itu kita hanya perlu upload foto / gambar yang sudah kita kumpulkan sebelumnya.



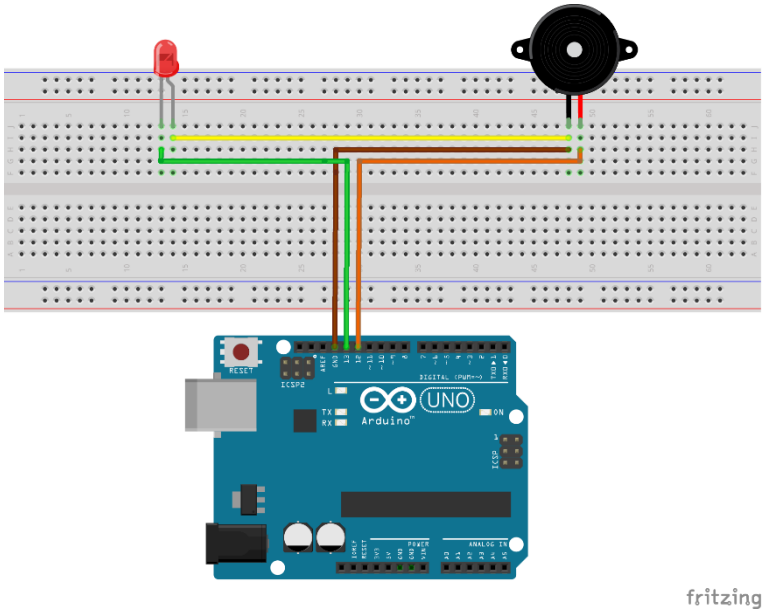
Gambar 4.3 proses upload foto yang sudah dikumpukan

3. Siapkan p5.js script untuk Image recognition di https://editorp5js.org/krantas/sketches/IKUf43rB. Kita buka link p5.js scriptnya lalu ganti model URL dengan link yang kita dapat pada model tensorflow.js sebelumnya dan juga ubah serialPort sesuai dengan com yang kita gunakan.

```
4  const modelURL =
   'https://teachablemachine.withgoogle.com/models/L0E4_YiWu/';
5  const serialPort = 'COM5';
```

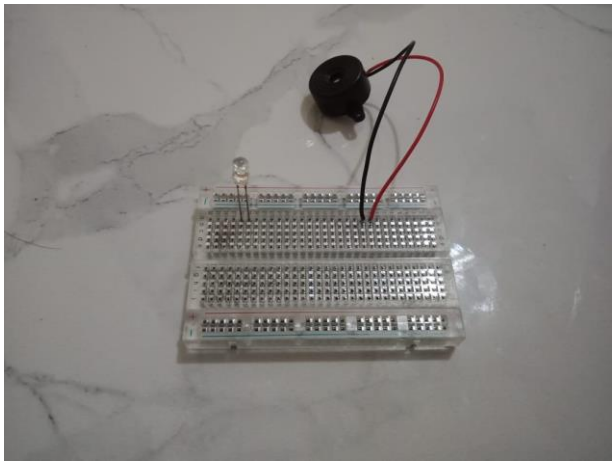
Gambar 4.4 p5.js script untuk image recognition

4. Langkah selanjutnya adalah Menyusun hardware. Namun sebelum mulai menyusun rangkaian, kita harus mengetahui terlebih dahulu model atau konsep dari projek yang akan dibuat. Dan inilah schematic yang akan dibuat.



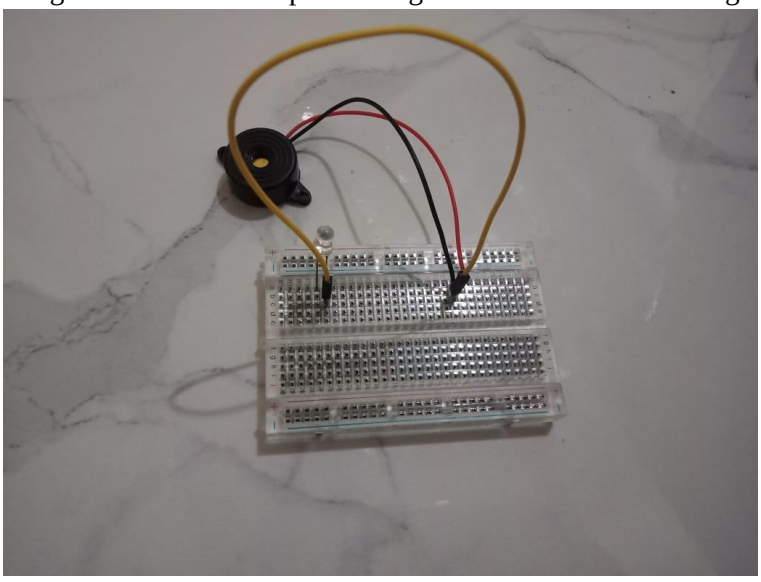
Gambar 4.5 schematic project

5. Setelah mengetahui schematic, kita akan mulai merangkai projek.
   - Pertama siapkan alat. Untuk alat yang digunakan kita memerlukan lampu LED, buzzer, kabel jumper, dan tidak lupa Arduino uno.
   - Kemudian gabungkan lampu LED dan buzzer ke project board.
   -



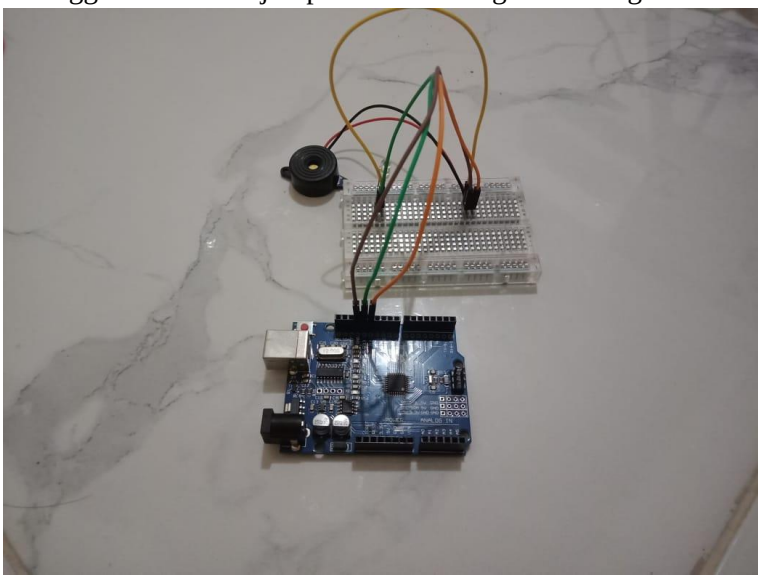
Gambar 4.6 lampu led dan buzzer yang sudah terhubung dengan papan board

- Setelah itu sambungkan kaki kanan lampu led dengan kabel hitam buzzer dengan kabel jumper.



Gambar 4.7 kaki kanan led yang sudah tersambung dengan buzzer dan kabel jumper

- Selanjutnya sambungkan kaki kiri led ke pin 13 arduino dan kabel merah buzzer ke pin 12 arduino dengan menggunakan kabel jumper dan sambungan led dengan buzzer ke GND.



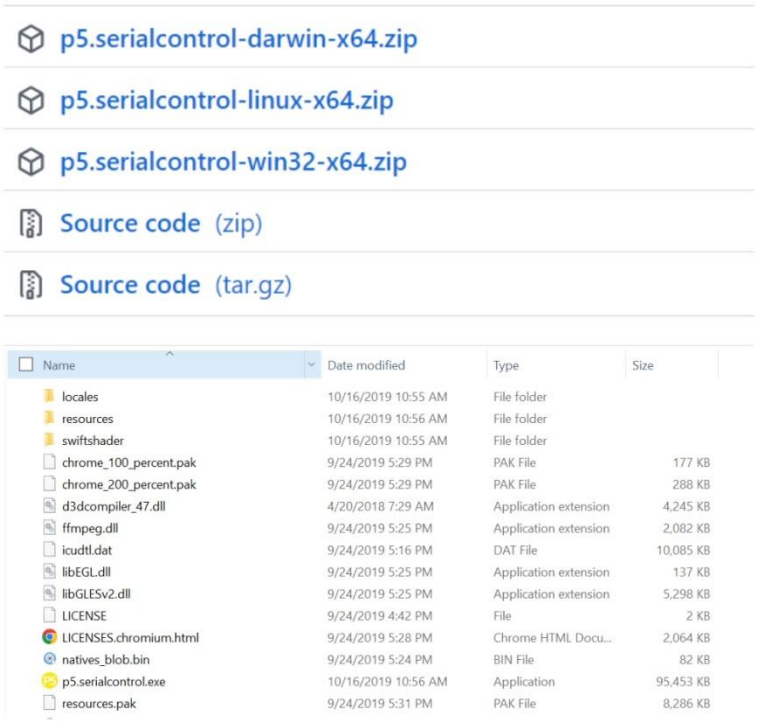
Gamber 4.8 Susunan hardware

- Setelah hardware tersusun dengan rapih, selanjutnya kita akan membuat kode program untuk menjalankan projek tersebut lalu menguploadnya kedalam arduino uno. Berikut program Arduino yang telah dibuat.

```
int led = 13;
int buzzer = 12;

char result;

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);

}

void loop() {
  while (Serial.available() > 0){
    result = Serial.read();
    switch (result) {
      case '1':
          digitalWrite(led,LOW);
          digitalWrite(buzzer,LOW);
          break;
      case '2':
          digitalWrite(led,HIGH);
          digitalWrite(buzzer,HIGH);
          break;
      }
    }
    delay(1000);
}
```

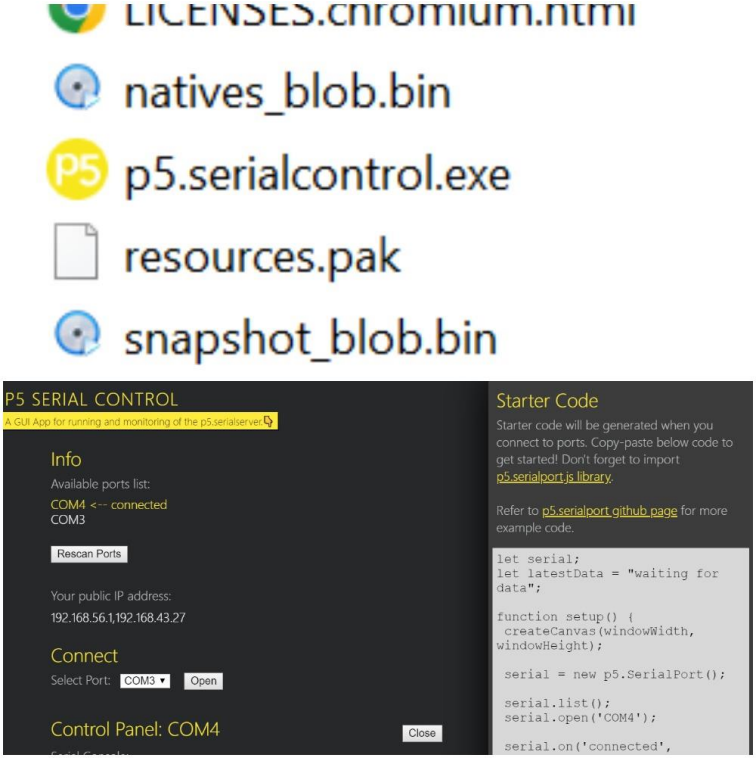
Gambar 4.9

- Download dan unzip p5.js serial control program (https://github.com/p5-serial/p5.serialcontrol/releases).



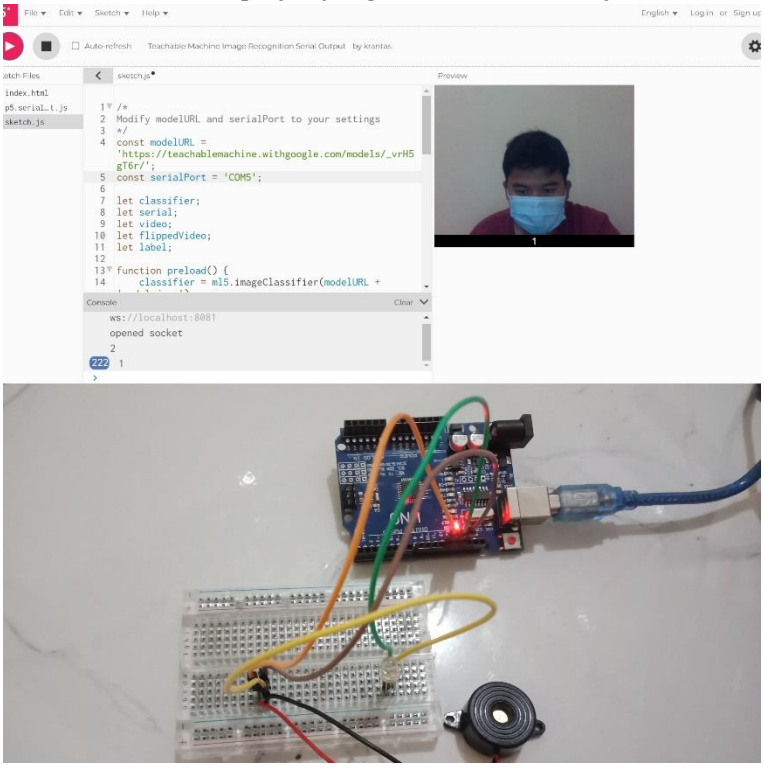
Gambar 4.10

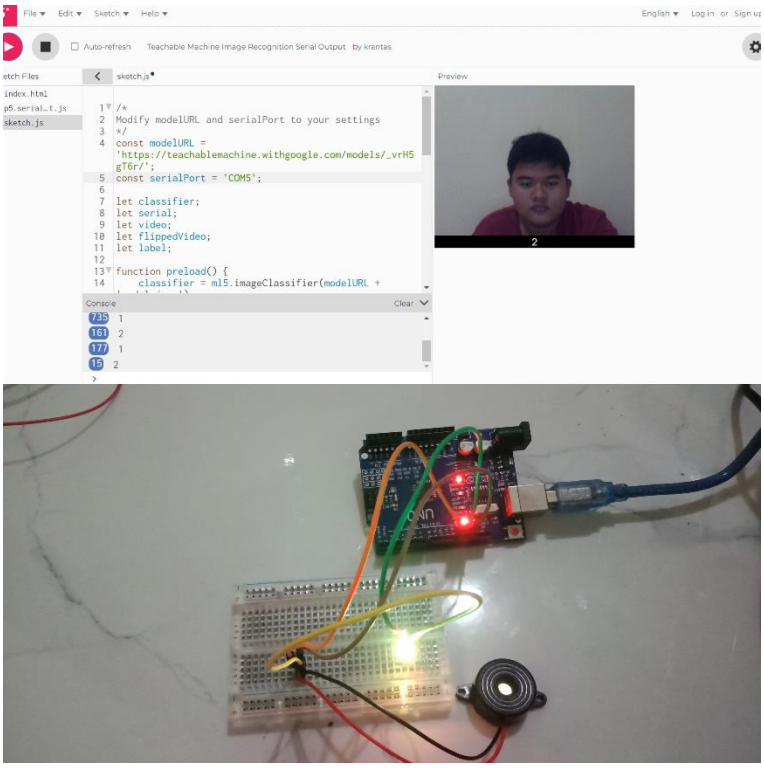
6. Jalankan p5.serialcontrol.exe saat dijalankan pada menu Connect pilih port com sesuai dengan port com pada p5.js script lalu klik open.



Gambar 4.11

7. Langkah terakhir adalah simulasikan projek yang telah dibuat. Caranya klik tombol play pada p5.js.





Gambar 4.12

5 Kode Program

p5.js merupakan library JavaScript yang digunakan dalam koding kreatif. Menggunakan <canvas> pada html sebagai sarana untuk menggambar. library ini bisa digunakan dengan mudah oleh berbagai kalangan mulai dari seniman, desainer, pengajar, pemula, dan banyak lagi. Kode program yang menggunakan p5.js ini berfungsi bukan hanya untuk menggambar tapi jugadapat menghubungkan Techable Machine dengan kamera seperti kode program dibawah ini.

P5.js script

index.html

```
<!DOCTYPE html>
<html lang="">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tachable Machine Image Recognition p5.js Serial Port</title>
    <style>
        html,
        body {
            margin: 0;
            padding: 0;
        }

        canvas {
            display: block;
        }
    </style>
```

```
    <script                                                    type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.4.0/p5.min.js"></script>
    <script                                                    type="text/javascript"
src="https://cdn.jsdelivr.net/npm/p5.serialserver@0.0.28/lib/p5.serialport.js"></script>
    <script                                                    type="text/javascript"
src="https://unpkg.com/ml5@0.6.1/dist/ml5.min.js"></script>
    <script type="text/javascript" src="/sketch.js"></script>
</head>

<body>
    <main>
    </main>
</body>

</html>
```

Pada kode program diatas berfungsi untuk menampilkan gambar saat menjalankan program.

p5.serialport.js

```
/*! p5.serialport.js v0.0.1 2015-07-23 */
/**
 * @module p5.serialport
 * @submodule p5.serialport
 * @for p5.serialport
 * @main
 */
/**
 *  p5.serialport
 *  Shawn Van Every (Shawn.Van.Every@nyu.edu)
 *  ITP/NYU
 *  LGPL
 *
 *  https://github.com/vanevery/p5.serialport
 *
 */
(function(root, factory) {
  if (typeof define === 'function' && define.amd)
    define('p5.serialport', ['p5'], function(p5) {
      (factory(p5));
    });
  else if (typeof exports === 'object')
    factory(require('../p5'));
  else
    factory(root['p5']);
}(this, function(p5) {

  // =============================================================================
  //                         p5.SerialPort
  // =============================================================================


  /**
   * Base class for a serial port. Creates an instance of the serial library and prints
"hostname":"serverPort" in the console.
   *
   * @class p5.SerialPort
   * @constructor
   * @param {String} [hostname] Name of the host. Defaults to 'localhost'.
   * @param {Number} [serverPort] Port number. Defaults to 8081.
   * @example
   *   var portName = '/dev/cu.usbmodem1411'; //enter your portName
   *
   *   function setup() {
   *           createCanvas(400, 300);
   *           serial = new p5.SerialPort()
   *           serial.open(portName);
   *   }
```

```
48.    */
49.   p5.SerialPort = function(_hostname, _serverport) {
50.     var self = this;
51.     this.bufferSize = 1; // How much to buffer before sending data event
52.     this.serialBuffer = [];
53.     //this.maxBufferSize = 1024;
54.     this.serialConnected = false;
55.     this.serialport = null;
56.     this.serialoptions = null;
57.     this.emitQueue = [];
58.     this.clientData = {};
59.     this.serialportList = [];
60.
61.     if (typeof _hostname === 'string') {
62.       this.hostname = _hostname;
63.     } else {
64.       this.hostname = "localhost";
65.     }
66.
67.     if (typeof _serverport === 'number') {
68.       this.serverport = _serverport;
69.     } else {
70.       this.serverport = 8081;
71.     }
72.
73.     try {
74.       this.socket = new WebSocket("ws://" + this.hostname + ":" + this.serverport);
75.       console.log(("ws://" + this.hostname + ":" + this.serverport));
76.     } catch (err) {
77.       if (typeof self.errorCallback !== "undefined") {
78.         self.errorCallback("Couldn't connect to the server, is it running?");
79.       }
80.     }
81.
82.     this.socket.onopen = function(event) {
83.       console.log('opened socket');
84.       serialConnected = true;
85.
86.       if (typeof self.connectedCallback !== "undefined") {
87.         self.connectedCallback();
88.       }
89.
90.       if (self.emitQueue.length > 0) {
91.         for (var i = 0; i < self.emitQueue.length; i ++){
92.           self.emit(self.emitQueue[i]);
93.         }
94.         self.emitQueue = [];
95.       }
96.     };
97.
98.     this.socket.onmessage = function(event) {
99.       var messageObject = JSON.parse(event.data);
100.             // MESSAGE ROUTING
101.             if (typeof messageObject.method !== "undefined") {
102.               if (messageObject.method == 'echo') {
103.               } else if (messageObject.method === "openserial") {
104.                 if (typeof self.openCallback !== "undefined") {
105.                   self.openCallback();
106.                 }
107.               } else if (messageObject.method === "data") {
108.                 // Add to buffer, assuming this comes in byte by byte
109.                 self.serialBuffer.push(messageObject.data);
110.                 if (typeof self.dataCallback !== "undefined") {
111.                   // Hand it to sketch
112.                   if (self.serialBuffer.length >= self.bufferSize) {
113.                     self.dataCallback();
114.                   }
115.                 }
116.                 if (typeof self.rawDataCallback !== "undefined") {
117.                   self.rawDataCallback(messageObject.data);
118.                 }
```

```
119.              } else if (messageObject.method === 'list') {
120.                self.serialportList = messageObject.data;
121.                if (typeof self.listCallback !== "undefined") {
122.                  self.listCallback(messageObject.data);
123.                }
124.              } else if (messageObject.method === "close") {
125.                if (typeof self.closeCallback !== "undefined") {
126.                  self.closeCallback();
127.                }
128.              } else if (messageObject.method === "write") {
129.                // Success Callback?
130.              } else if (messageObject.method === "error") {
131.                //console.log(messageObject.data);
132.                if (typeof self.errorCallback !== "undefined") {
133.                  self.errorCallback(messageObject.data);
134.                }
135.              } else {
136.                // Got message from server without known method
137.                console.log("Unknown Method: " + messageObject);
138.              }
139.            } else {
140.              console.log("Method Undefined: " + messageObject);
141.            }
142.          };
143.
144.          this.socket.onclose = function(event) {
145.            if (typeof self.closeCallback !== "undefined") {
146.              self.closeCallback();
147.            }
148.          };
149.
150.          this.socket.onerror = function(event) {
151.            if (typeof self.errorCallback !== "undefined") {
152.              self.errorCallback();
153.            }
154.          };
155.        };
156.
157.       /**
158.        *
159.        * @method emit
160.        * @private
161.        * @return
162.        * @example
163.        *
164.        */
165.        p5.SerialPort.prototype.emit = function(data) {
166.          if (this.socket.readyState == WebSocket.OPEN) {
167.            this.socket.send(JSON.stringify(data));
168.          } else {
169.            this.emitQueue.push(data);
170.          }
171.        };
172.
173.       /**
174.        * Tells you whether p5 is connected to the serial port.
175.        *
176.        * @method isConnected
177.        * @return {Boolean} true or false
178.        * @example
179.        *             var serial; // variable to hold an instance of the serialport library
180.        *             var portName = '/dev/cu.usbmodem1411';
181.        *
182.        *             function setup() {
183.        *                     createCanvas(400, 300);
184.        *                     serial = new p5.SerialPort();
185.        *                     serial.open(portName);
186.        *                     println(serial.isConnected());
187.        *             }
188.        */
189.        p5.SerialPort.prototype.isConnected = function() {
```

```
190.           if (self.serialConnected) { return true; }
191.           else { return false; }
192.         };
193.
194.       /**
195.        * Lists serial ports available to the server.
196.        * Synchronously returns cached list, asynchronously returns updated list via callback.
197.        * Must be called within the p5 setup() function.
198.        * Doesn't work with the p5 editor's "Run in Browser" mode.
199.        *
200.        * @method list
201.        * @return {Array} array of available serial ports
202.        * @example
203.        *             function setup() {
204.        *             createCanvas(windowWidth, windowHeight);
205.        *             serial = new p5.SerialPort();
206.        *             serial.list();
207.        *             serial.open("/dev/cu.usbmodem1411");
208.        *             }
209.        *
210.        *  For full example: <a href="https://itp.nyu.edu/physcomp/labs/labs-serial-communication/two-way-duplex-serial-communication-using-p5js/">Link</a>
211.        * @example
212.        *             function printList(portList) {
213.        *               // portList is an array of serial port names
214.        *               for (var i = 0; i < portList.length; i++) {
215.        *                 // Display the list the console:
216.        *                 println(i + " " + portList[i]);
217.        *               }
218.        *             }
219.        */
220.         p5.SerialPort.prototype.list = function(cb) {
221.           if (typeof cb === 'function') {
222.             this.listCallback = cb;
223.           }
224.           this.emit({
225.             method: 'list',
226.             data: {}
227.           });
228.
229.           return this.serialportList;
230.         };
231.
232.       /**
233.        * Opens the serial port to enable data flow.
234.        * Use the {[serialOptions]} parameter to set the baudrate if it's different from the p5 default, 9600.
235.        *
236.        * @method open
237.        *  @param    {String}  serialPort  Name  of  the  serial  port,  something  like '/dev/cu.usbmodem1411'
238.        * @param  {Object} [serialOptions] Object with optional options as {key: value} pairs.
239.        *                                  Options include 'baudrate'.
240.        * @param  {Function} [serialCallback] Callback function when open completes
241.        * @example
242.        *             // Change this to the name of your arduino's serial port
243.        *             serial.open("/dev/cu.usbmodem1411");
244.        *
245.        * @example
246.        *             // All of the following are valid:
247.        *             serial.open(portName);
248.        *             serial.open(portName, {}, onOpen);
249.        *             serial.open(portName, {baudrate: 9600}, onOpen)
250.        *
251.        *             function onOpen() {
252.        *               print('opened the serial port!');
253.        *             }
254.        */
255.         p5.SerialPort.prototype.open = function(_serialport, _serialoptions, cb) {
```

```
256.
257.           if (typeof cb === 'function') {
258.             this.openCallback = cb;
259.           }
260.
261.           this.serialport = _serialport;
262.
263.           if (typeof _serialoptions === 'object') {
264.             this.serialoptions = _serialoptions;
265.           } else {
266.             //console.log("typeof _serialoptions " + typeof _serialoptions + " setting to {}");
267.             this.serialoptions = {};
268.           }
269.           // If our socket is connected, we'll do this now,
270.           // otherwise it will happen in the socket.onopen callback
271.           this.emit({
272.             method: 'openserial',
273.             data: {
274.               serialport: this.serialport,
275.               serialoptions: this.serialoptions
276.             }
277.           });
278.         };
279.
280.       /**
281.        * Sends a byte to a webSocket server which sends the same byte out through a serial port.
282.        * @method write
283.        * @param  {String, Number, Array} Data Writes bytes, chars, ints, bytes[], and strings to the serial port.
284.        * @example
285.        * You can use this with the included Arduino example called PhysicalPixel.
286.        * Works with P5 editor as the socket/serial server, version 0.5.5 or later.
287.        *   Written   2   Oct   2015   by   Tom   Igoe.   For   full   example:   <a href="https://github.com/vanevery/p5.serialport/tree/master/examples/writeExample">Link</a>
288.        *
289.        *             function mouseReleased() {
290.        *                       serial.write(outMessage);
291.        *                       if (outMessage === 'H') {
292.        *                         outMessage = 'L';
293.        *                       } else {
294.        *                         outMessage = 'H';
295.        *                       }
296.        *             }
297.        *
298.        *  For  full  example:  <a  href="https://itp.nyu.edu/physcomp/labs/labs-serial-communication/lab-serial-output-from-p5-js/">Link</a>
299.        * @example
300.        *             function mouseDragged() {
301.        *               // map the mouseY to a range from 0 to 255:
302.        *                       outByte = int(map(mouseY, 0, height, 0, 255));
303.        *                       // send it out the serial port:
304.        *                       serial.write(outByte);
305.        *             }
306.        */
307.         p5.SerialPort.prototype.write = function(data) {
308.           //Writes bytes, chars, ints, bytes[], Strings to the serial port
309.           var toWrite = null;
310.           if (typeof data == "number") {
311.             // This is the only one I am treating differently, the rest of the clauses are meaningless
312.             toWrite = [data];
313.           } else if (typeof data == "string") {
314.             toWrite = data;
315.           } else if (Array.isArray(data)) {
316.             toWrite = data;
317.           } else {
318.             toWrite = data;
319.           }
```

```
320.
321.           this.emit({
322.             method: 'write',
323.             data: toWrite
324.           });
325.         };
326.
327.       /**
328.        * Returns a number between 0 and 255 for the next byte that's waiting in the buffer.
329.        * Returns -1 if there is no byte, although this should be avoided by first checking
    available() to see if data is available.
330.        *
331.        * @method read
332.        * @return {Number} Value of the byte waiting in the buffer. Returns -1 if there is
    no byte.
333.        * @example
334.        *             function serialEvent() {
335.        *             inByte = int(serial.read());
336.        *                     byteCount++;
337.        *             }
338.        *
339.        * @example
340.        *             function serialEvent() {
341.        *                     // read a byte from the serial port:
342.        *                     var inByte = serial.read();
343.        *                     // store it in a global variable:
344.        *                     inData = inByte;
345.        *             }
346.        */
347.        p5.SerialPort.prototype.read = function() {
348.          if (this.serialBuffer.length > 0) {
349.            return this.serialBuffer.shift();
350.          } else {
351.            return -1;
352.          }
353.        };
354.
355.       /**
356.        * Returns the next byte in the buffer as a char.
357.        *
358.        * @method readChar
359.        * @return {String} Value of the Unicode-code unit character byte waiting in the
    buffer, converted from bytes. Returns -1 or 0xffff if there is no byte.
360.        * @example
361.        *             var inData;
362.        *
363.        *             function setup() {
364.        *               // callback for when new data arrives
365.        *               serial.on('data', serialEvent);
366.        *
367.        *             function serialEvent() {
368.        *               // read a char from the serial port:
369.        *               inData = serial.readChar();
370.        *             }
371.        */
372.        p5.SerialPort.prototype.readChar = function() {
373.          if (this.serialBuffer.length > 0) {
374.            /*var currentByte = this.serialBuffer.shift();
375.            console.log("p5.serialport.js: " + currentByte);
376.            var currentChar = String.fromCharCode(currentByte);
377.            console.log("p5.serialport.js: " + currentChar);
378.            return currentChar;
379.            */
380.            return String.fromCharCode(this.serialBuffer.shift());
381.          } else {
382.            return -1;
383.          }
384.        };
385.
386.       /**
```

```
387.         * Returns a number between 0 and 255 for the next byte that's waiting in the buffer,
     and then clears the buffer of data. Returns -1 if there is no byte, although this should be
     avoided by first checking available() to see if data is available.
388.         * @method readBytes
389.         * @return {Number} Value of the byte waiting in the buffer. Returns -1 if there is
     no byte.
390.         * @example
391.         *             var inData;
392.         *
393.         *             function setup() {
394.         *               // callback for when new data arrives
395.         *               serial.on('data', serialEvent);
396.         *
397.         *             function serialEvent() {
398.         *               // read bytes from the serial port:
399.         *               inData = serial.readBytes();
400.         *             }
401.         */
402.          p5.SerialPort.prototype.readBytes = function() {
403.            if (this.serialBuffer.length > 0) {
404.              var returnBuffer = this.serialBuffer.slice();
405.
406.              // Clear the array
407.              this.serialBuffer.length = 0;
408.
409.              return returnBuffer;
410.            } else {
411.              return -1;
412.            }
413.          };
414.
415.        /**
416.         * Returns all of the data available, up to and including a particular character.
417.         * If the character isn't in the buffer, 'null' is returned.
418.         * The version without the byteBuffer parameter returns a byte array of all data up
     to and including the interesting byte.
419.         * This is not efficient, but is easy to use.
420.         *
421.         * The version with the byteBuffer parameter is more efficient in terms of time and
     memory.
422.         * It grabs the data in the buffer and puts it into the byte array passed in and
     returns an integer value for the number of bytes read.
423.         * If the byte buffer is not large enough, -1 is returned and an error is printed
     to the message area.
424.         * If nothing is in the buffer, 0 is returned.
425.         *
426.         * @method readBytesUntil
427.         * @param {[byteBuffer]}
428.         * @return {[Number]} [Number of bytes read]
429.         * @example
430.         *             // All of the following are valid:
431.         *             charToFind.charCodeAt();
432.         *             charToFind.charCodeAt(0);
433.         *             charToFind.charCodeAt(0, );
434.         */
435.          p5.SerialPort.prototype.readBytesUntil = function(charToFind) {
436.            console.log("Looking for: " + charToFind.charCodeAt(0));
437.            var index = this.serialBuffer.indexOf(charToFind.charCodeAt(0));
438.            if (index !== -1) {
439.              // What to return
440.              var returnBuffer = this.serialBuffer.slice(0, index + 1);
441.              // Clear out what was returned
442.              this.serialBuffer = this.serialBuffer.slice(index, this.serialBuffer.length +
     index);
443.              return returnBuffer;
444.            } else {
445.              return -1;
446.            }
447.          };
448.
449.        /**
```

```
450.         * Returns all the data from the buffer as a String.
451.         * This method assumes the incoming characters are ASCII.
452.         * If you want to transfer Unicode data: first, convert the String to a byte stream
    in the representation of your choice (i.e. UTF8 or two-byte Unicode data).
453.         * Then, send it as a byte array.
454.         *
455.         * @method readString
456.         * @return
457.         * @example
458.         *
459.         *
460.         *
461.         *
462.         */
463.         p5.SerialPort.prototype.readString = function() {
464.           //var returnBuffer = this.serialBuffer;
465.           var stringBuffer = [];
466.           //console.log("serialBuffer Length: " + this.serialBuffer.length);
467.           for (var i = 0; i < this.serialBuffer.length; i++) {
468.             //console.log("push: " + String.fromCharCode(this.serialBuffer[i]));
469.             stringBuffer.push(String.fromCharCode(this.serialBuffer[i]));
470.           }
471.           // Clear the buffer
472.           this.serialBuffer.length = 0;
473.           return stringBuffer.join("");
474.         };
475.
476.       /**
477.         * Returns all of the data available as an ASCII-encoded string.
478.         *
479.         * @method readStringUntil
480.         * @param {String} stringToFind String to read until.
481.         * @return {String} ASCII-encoded string until and not including the stringToFind.
482.         * @example
483.         *
484.         *                For               full               example:               <a
    href="https://github.com/tigoe/p5.serialport/blob/master/examples/twoPortRead/sketch.js">
    Link</a>
485.         *
486.         *              var serial1 = new p5.SerialPort();
487.         *              var serial2 = new p5.SerialPort();
488.         *              var input1 = '';
489.         *              var input2 = '';
490.         *
491.         *              function serialEvent(){
492.         *                            data = serial1.readStringUntil('\r\n');
493.         *                            if (data.length > 0){
494.         *                            input1 = data;
495.         *                            }
496.         *              }
497.         *
498.         *              function serial2Event() {
499.         *                            var data = serial2.readStringUntil('\r\n');
500.         *                            if (data.length > 0){
501.         *                            input2 = data;
502.         *                            }
503.         *              }
504.         */
505.         p5.SerialPort.prototype.readStringUntil = function(stringToFind) {
506.
507.           var stringBuffer = [];
508.           //console.log("serialBuffer Length: " + this.serialBuffer.length);
509.           for (var i = 0; i < this.serialBuffer.length; i++) {
510.             //console.log("push: " + String.fromCharCode(this.serialBuffer[i]));
511.             stringBuffer.push(String.fromCharCode(this.serialBuffer[i]));
512.           }
513.           stringBuffer = stringBuffer.join("");
514.           //console.log("stringBuffer: " + stringBuffer);
515.
516.           var returnString = "";
517.           var foundIndex = stringBuffer.indexOf(stringToFind);
```

```
518.           //console.log("found index: " + foundIndex);
519.           if (foundIndex > -1) {
520.             returnString = stringBuffer.substr(0, foundIndex);
521.             this.serialBuffer        =        this.serialBuffer.slice(foundIndex        +
    stringToFind.length);
522.           }
523.           //console.log("Sending: " + returnString);
524.           return returnString;
525.         };
526.
527.
528.       /**
529.        * Returns all of the data available as an ASCII-encoded string until a line break
    is encountered.
530.        *
531.        * @method readLine
532.        * @return {String} ASCII-encoded string
533.        * @example
534.        *
535.        * You can use this with the included Arduino example called AnalogReadSerial.
536.        * Works with P5 editor as the socket/serial server, version 0.5.5 or later.
537.        *   Written   2   Oct   2015   by   Tom   Igoe.   For   full   example:   <a
    href="https://github.com/vanevery/p5.serialport/tree/master/examples/readAndAnimate">Link
    </a>
538.        *
539.        *             function gotData() {
540.        *               var currentString = serial.readLine();  // read the incoming data
541.        *                       trim(currentString);                    // trim off trailing
    whitespace
542.        *
543.        *                       if (!currentString) return; {           // if the incoming
    string is empty, do no more
544.        *                         console.log(currentString);
545.        *                         }
546.        *
547.        *                       if (!isNaN(currentString)) {  // make sure the string is a
    number (i.e. NOT Not a Number (NaN))
548.        *                         textXpos = currentString;   // save the currentString to
    use for the text position in draw()
549.        *                         }
550.        *                     }
551.        */
552.         p5.SerialPort.prototype.readLine = function() {
553.           return this.readStringUntil("\r\n");
554.         };
555.
556.       /**
557.        * Returns the number of bytes available.
558.        *
559.        * @method available
560.        * @return {Number} The length of the serial buffer array, in terms of number of
    bytes in the buffer.
561.        * @example
562.        *             function draw() {
563.        *                     // black background, white text:
564.        *                     background(0);
565.        *                     fill(255);
566.        *                     // display the incoming serial data as a string:
567.        *                     var displayString = "inByte: " + inByte + "\t Byte count: "
    + byteCount;
568.        *                     displayString += "  available: " + serial.available();
569.        *                     text(displayString, 30, 60);
570.        *                     }
571.        * */
572.         p5.SerialPort.prototype.available = function() {
573.           return this.serialBuffer.length;
574.         };
575.
576.       /**
577.        * Returns the last byte of data from the buffer.
578.        *
```

```
579.         * @method last
580.         * @return {Number}
581.         * @example
582.         *
583.         * */
584.          p5.SerialPort.prototype.last = function() {
585.            //Returns last byte received
586.            var last = this.serialBuffer.pop();
587.            this.serialBuffer.length = 0;
588.            return last;
589.          };
590.
591.        /**
592.         * Returns the last byte of data from the buffer as a char.
593.         *
594.         * @method lastChar
595.         * @example
596.         *
597.         * */
598.          p5.SerialPort.prototype.lastChar = function() {
599.            return String.fromCharCode(this.last());
600.          };
601.
602.        /**
603.         * Clears the underlying serial buffer.
604.         *
605.         * @method clear
606.         * @example
607.         */
608.          p5.SerialPort.prototype.clear = function() {
609.            //Empty the buffer, removes all the data stored there.
610.            this.serialBuffer.length = 0;
611.          };
612.
613.        /**
614.         * Stops data communication on this port.
615.         * Use to shut the connection when you're finished with the Serial.
616.         *
617.         * @method stop
618.         * @example
619.         *
620.         */
621.          p5.SerialPort.prototype.stop = function() {
622.          };
623.
624.        /**
625.         * Tell server to close the serial port. This functions the same way as
    serial.on('close', portClose).
626.         *
627.         * @method close
628.         * @param {String} name of callback
629.         * @example
630.         *
631.         *             var inData;
632.         *
633.         *             function setup() {
634.         *               serial.open(portOpen);
635.         *               serial.close(portClose);
636.         *             }
637.         *
638.         *      function portOpen() {
639.         *               println('The serial port is open.');
640.         *             }
641.         *
642.         *             function portClose() {
643.         *               println('The serial port closed.');
644.         *             }
645.         */
646.          p5.SerialPort.prototype.close = function(cb) {
647.            //
648.            if (typeof cb === 'function') {
```

```
649.             this.closeCallback = cb;
650.           }
651.           this.emit({
652.             method: 'close',
653.             data: {}
654.           });
655.         };
656.
657.         /**
658.          * Register clients that connect to the serial server.
659.          *
660.          * This is for use with the p5 Serial Control application so the application
661.          * can access and render the names of clients who have connected. Note that
662.          * calling this method does not log the list of registered clients. To do that,
663.          * you'd use:
664.          * serial.on('registerClient', logClientData)
665.          *
666.          * The example demonstates the registerClient method, as well as how you'd log
667.          * the list of clients.
668.          *
669.          * @method registerClient
670.          * @example
671.          *
672.          * function setup() {
673.          *   // Create a new p5 Serial Port object
674.          *   serial = new p5.SerialPort();
675.          *   // List the available ports
676.          *   serial.list();
677.          *   // On port open, call the gotOpen callback
678.          *   serial.on('open', gotOpen);
679.          *   // Register the clients that have connected to the server
680.          *   serial.registerClient();
681.          *   // After registerClient method is done, call the logClientData callback
682.          *   serial.on('registerClient', logClientData)
683.          * }
684.          *
685.          * // Callback to log the client data
686.          * function logClientData(data) {
687.          *   console.log("Client data: ", data)
688.          * }
689.          *
690.          * // Callback to log a message when the port is opened
691.          * function gotOpen() {
692.          *   console.log("Serial port is open.")
693.          * }
694.          */
695.         // p5.SerialPort.prototype.registerClient = function(cb) {
696.         //   if (typeof cb === 'function') {
697.         //     this.registerCallback = cb;
698.         //   }
699.         //   this.emit({
700.         //     method: 'registerClient',
701.         //     data: {}
702.         //   });
703.         //   return this.clientData;
704.         // };
705.
706.         /**
707.          * // Register callback methods from sketch
708.          *
709.          */
710.         p5.SerialPort.prototype.onData = function(_callback) {
711.           this.on('data',_callback);
712.         };
713.
714.         p5.SerialPort.prototype.onOpen = function(_callback) {
715.           this.on('open',_callback);
716.         };
717.
718.         p5.SerialPort.prototype.onClose = function(_callback) {
719.           this.on('close',_callback);
```

```
        };

        p5.SerialPort.prototype.onError = function(_callback) {
          this.on('error',_callback);
        };

        p5.SerialPort.prototype.onList = function(_callback) {
          this.on('list',_callback);
        };

        p5.SerialPort.prototype.onConnected = function(_callback) {
          this.on('connected',_callback);
        };

        p5.SerialPort.prototype.onRawData = function(_callback) {
          this.on('rawdata',_callback);
        };

        // Version 2
        p5.SerialPort.prototype.on = function(_event, _callback) {
          if (_event == 'open') {
            this.openCallback = _callback;
          } else if (_event == 'data') {
            this.dataCallback = _callback;
          } else if (_event == 'close') {
            this.closeCallback = _callback;
          } else if (_event == 'error') {
            this.errorCallback = _callback;
          } else if (_event == 'list') {
            this.listCallback = _callback;
          } else if (_event == 'connected') {
            this.connectedCallback = _callback;
          } else if (_event == 'rawdata') {
            this.rawDataCallback = _callback;
          }
        };
      }));

```

Program diatas berfungsi sebagai Processing Serial Library API. Karena JavaScript di browser tidak dapat berinteraksi langsung dengan port serial pada arduino, perpustakaan ini menyelesaikannya atau bisa dibilang sebagai penghubung antara browser dengan arduino.

Sketch.js

```
/*
Modify modelURL and serialPort to your settings
*/
const modelURL = 'https://teachablemachine.withgoogle.com/models/_______/';
const serialPort = 'COM_';

let classifier;
let serial;
let video;
let flippedVideo;
let label;

function preload() {
    classifier = ml5.imageClassifier(modelURL + 'model.json');
    serial = new p5.SerialPort();
}

function setup() {
    serial.open(serialPort);
    createCanvas(320, 260);
    video = createCapture(VIDEO);
    video.size(320, 240);
    video.hide();
```

```
24.     flippedVideo = ml5.flipImage(video);
25.     classifyVideo();
```

Sama seperti program sebelumnya program ini juga berfungsi menghubungkan kamera yang nanti akan diubah menjadi data yang akan dikirim ke Techable Machine lalu ke arduino.

Arduino program

Program yang digunakan adalah Arduino IDE 1.8.8 untuk memberikan perintah kedalam arduino uno. Pada bagian ini merupakan penjelasan tentang kode program yang terdapat pada arduino.

```
int led = 13;
int buzzer = 12;

char result;

void setup() {
  Serial.begin(9600);
  pinMode(led, OUTPUT);
  pinMode(buzzer, OUTPUT);

}

void loop() {
  while (Serial.available() > 0){
    result = Serial.read();
    switch (result) {
      case '1':
          digitalWrite(led,LOW);
          digitalWrite(buzzer,LOW);
          break;
      case '2':
          digitalWrite(led,HIGH);
          digitalWrite(buzzer,HIGH);
          break;
      }
    }
    delay(1000);
}
```

Pada kode programdiatas menjelaskan led berada pada pin 13 dan buzzer berada pada pin 12. Dimana saat p5.js memberikan data 1 maka led dan buzzer akan berada pada keadaan low atau mati, sedangkan jika data yang diberikana dalah 2 maka led and buzzer akan berada pada keadaan high atau menyala.

6 Hasil

Pendeteksi masker mampu melakukan pendeteksian masker satu persatu dengan cara mendekatkan area wajah agar terbaca secara keseluruhan yaitu di jarak ±1 meter dari kamera, dengan target keberhasilannya:

20 percobaan		True class		
		Bermasker	Tidak Bermasker	Tidak terdektesi
Predicted class	Bermasker	15	3	2
	Tidak Bermasker	3	15	2
	Tidak Terdeteksi	2	2	16
20 percobaan		True class		
		Bermasker	Tidak Bermasker	Tidak terdektesi
Predicted class	Bermasker	A	B	C
	Tidak Bermasker	D	E	F
	Tidak Terdeteksi	G	H	I

Keterangan

- True Positive (TP) = A =15
- True Negative (TN) = E+F+H+I= 15+2+2+16 = 35
- False Positive (FP) = B+C = 3+2 = 5
- False Negative (FN) = D+G = 3+2 = 5
- Precision = TP/(TP+FP) = 15/(15+5) = 0,75
- Recall = TP/(TP+FN) = 15/(15+5) = 0,75
- Accuracy= (TP+TN)/(TP+TN+FP+FN)=(15+35)/(15+35 +5+5)=0,83

Berdasarkan hasil percobaan terdapat beberapa kendala seperti pada bagian jaga jarak, percobaan ke 12 gagal dikarenakan lepasnya kabel. Pada percobaan pendeteksian masker ke 11 pengunjung tidak menggunakan masker namun menggunakan faceshield terbaca bermasker , hal ini disebabkan karena pada bagian awal dataset kami belum mencantumkan dataset orang dengan menggunakan faceshield sehingga sistem tidak dapat mengenali kondisi tersebut. Dengan menggunakan alat ini nantinya mampu bermanfaat membantu penerapan protokol kesehatan khususnya di pintu masuk tempat-tempat umum yang memungkinkan terjadinya antrean atau kerumunan sehingga dapat sehingga dapat bermanfaat membantu mencegah penyebaran COVID-19 berdasarkan anjuran pemerintah mengenai tata cara penerapan protokol kesehatan dengan benar, serta meminimalisir adanya kontak erat antara petugas atau pengawas penerapan protokol kesehatan dengan pengunjung.

7 Referensi

[1] Olipmimi, Dianita. 2021. *Teachable Machine*: Membuat Model *Machine Learning* dengan Mudah. https://medium.com/statistics-uii/teachable-machine-membuat-model-machine-learning-dengan-mudah-27de270a6107. Diakses pada 13 Mei 2022.

[2] Gunawan, Ibnu. 2021. Ap aitu *Teachable Machine*. https://ibnuu.com/apa-itu-teachable-machine. Diakses pada 13 Mei 2021.

[3] Sekolah Robot Indonesia. 2020. Deteksi Masker dengan Arduino (persiapan *new normal*). https://youtu.be/oU1lJ1G5BdI. Diakses pada 28 April 2022.

[4] TIF Exhibition. 2021. Deteksi Masker Otomatis Menggunakan Arduino. https://wsjti.id/2021/product/eyJpdiI6Ilg4cjZId1RNUEpLV0JBRGhnRWhMSHc9PSIsInZhbHVlIjoidUdIZDNcL2srM0M3S2VpSVh1bmYrZFE9PSIsIm1hYyI6IjRhZDIzOTQ5OTJlMzJkOTU3OTA5OGMzNzVkYTYxYzk3ZTQ0OWU1NzcyYzFiNDMwMmY0MWQ5MGM4YmU2NzhlMjEifQ==/deteksi-masker-otomatis-menggunakan-arduino. Diakses pada 11 Mei 2022.

[5] Hermawan, Rudi, Adhy, Dewanto, dan Anwar, Nizirwan. 2020. Sistem Pendeteksi Penggunaan Masker Sesuai Protokol Kesehatan Covid-19 Menggunakan Metode Deep Learning. https://www.researchgate.net/publication/349303365_Sistem_Pendeteksi_Penggunaan_Masker_Sesuai_Protokol_Kesehatan_Covid_19_Menggunakan_Metode_Deep_Learning. Diakses pada 11 Mei 2022.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/