LAMPIRAN

1. *Source Code* Halaman Pendeteksi

```
window.onload = async () => {
const video = document.getElementById('video');
const maskImageCount = 5;
const noMaskImageCount = 8;
const trainImagesContainer = document.querySelector('.train-images');
for (let i = 1; i <= maskImageCount; i++) {
const newImage = document.createElement('IMG');
newImage.crossOrigin = "anonymous";
newImage.setAttribute('src', `/public/img/mask/${i}.jpg`);
newImage.classList.add('mask-img');
trainImagesContainer.appendChild(newImage);
}
for (let i = 1; i <= noMaskImageCount; i++) {
const newImage = document.createElement('IMG');
newImage.crossOrigin = "anonymous";
newImage.setAttribute('src', `/public/img/no_mask/${i}.jpg`);
newImage.classList.add('no-mask-img');
trainImagesContainer.appendChild(newImage);
}
const mobilenetModule = await mobilenet.load({version: 2, alpha: 1});
const classifier = await trainClassifier(mobilenetModule);
Promise.all([
faceapi.nets.tinyFaceDetector.loadFromUri('/public/models'),
faceapi.nets.faceLandmark68Net.loadFromUri('/public/models'),
faceapi.nets.faceRecognitionNet.loadFromUri('/public/models'),
faceapi.nets.faceExpressionNet.loadFromUri('/public/models')
]).then(startVideo)
function startVideo() {
navigator.getUserMedia(
{
video: {}
},
stream => video.srcObject = stream,
err => console.error(err)
)
};
```

```
video.addEventListener('play', () => {
const canvas = document.getElementById('mycanvas');
const displaySize = { width: video.width, height: video.height }
faceapi.matchDimensions(canvas, displaySize)
setInterval(async () => {
const detections = await faceapi.detectAllFaces(video, new
faceapi.TinyFaceDetectorOptions()).withFaceLandmarks().withFaceExpressio
ns()
const resizedDetections = faceapi.resizeResults(detections, displaySize)
canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height)
faceapi.draw.drawDetections(canvas, resizedDetections)
const tfTestImage = tf.browser.fromPixels(video);
const logits = mobilenetModule.infer(tfTestImage, 'conv_preds');
const prediction = await classifier.predictClass(logits);
if (prediction.label == 1) {
document.getElementById('hasil').innerHTML = "Tidak menggunkan
masker";
} else {
document.getElementById('hasil').innerHTML = "Menggunakan masker";
}
}, 100)
});
};
async function trainClassifier(mobilenetModule) {
const classifier = knnClassifier.create();
const maskImages = document.querySelectorAll('.mask-img');
maskImages.forEach(img => {
const tfImg = tf.browser.fromPixels(img);
const logits = mobilenetModule.infer(tfImg, 'conv_preds');
classifier.addExample(logits, 0);
});
const noMaskImages = document.querySelectorAll('.no-mask-img');
noMaskImages.forEach(img => {
const tfImg = tf.browser.fromPixels(img);
const logits = mobilenetModule.infer(tfImg, 'conv_preds');
classifier.addExample(logits, 1);
});
return classifier;
}
```

2. *Source Code* Palang Pintu

```
#include <Arduino.h>
#include <Servo.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#define USE_SERIAL Serial
ESP8266WiFiMulti WiFiMulti;
Servo servo;
// Pin Input & Output
int trigPin = D5;
int echoPin = D6;
// Variable Sensor
long duration;
int distance;
void setup() {
USE_SERIAL.begin(9600);
USE_SERIAL.println();
USE_SERIAL.println();
USE_SERIAL.println();
for(uint8_t t = 4; t > 0; t--) {
USE_SERIAL.flush();
delay(1000);
}
WiFi.mode(WIFI_STA);
WiFiMulti.addAP("KADETECH", "D164NT1!");
servo.attach(D3);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
}
void loop() {
if((WiFiMulti.run() == WL_CONNECTED)) {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
```

```
HTTPClient http;
USE_SERIAL.println("Sending Get Request to Server.....");
http.begin("http://192.168.100.64:5000/");
int httpCode = http.GET();
if(httpCode > 0) {
if(httpCode == HTTP_CODE_OK) {
// HTTP_CODE_OK == 200
String payload = http.getString();
USE_SERIAL.println(payload);
// Mulai
if(distance<20){
if(servo.read() == 150){
if(payload == "true"){
servo.write(90);
}
}else{
if(payload == "false"){
servo.write(150);
}
}
}else{
servo.write(150);
HTTPClient http;
http.begin("http://192.168.100.64:5000/dari-arduino/false/");
int httpCode = http.GET();
USE_SERIAL.println(httpCode);
}
}
}else{
// Jika Httpcode error
USE_SERIAL.printf("[HTTP] GET... failed, error: %s\n",
http.errorToString(httpCode).c_str());
}
http.end();
}
delay(1000);
}
```