Mask detector using Raspberry Pi Pico and MQTT

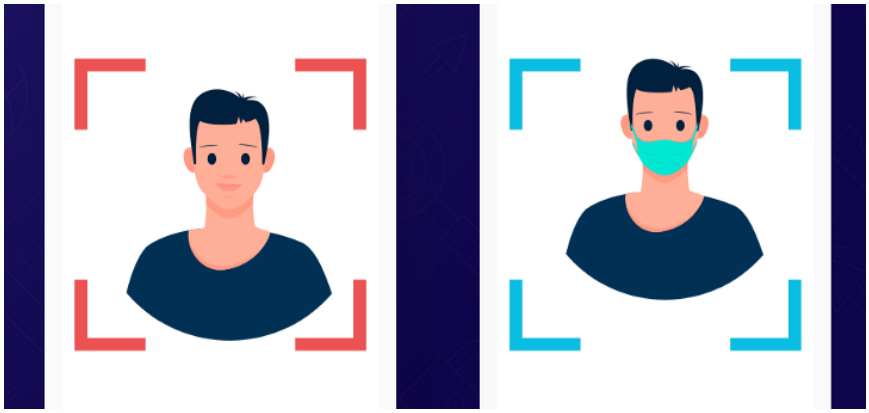
Rajesh S, Harshitha JP, Nikhil M

01 ABSTRACT

"Prevention is better than cure" is one of the effective measures to prevent the spreading of COVID-19 and to protect mankind. Many researchers and doctors are working on medication and vaccination for corona. COVID-19 spreads mostly by droplet infection when people cough or if we touch someone who is ill and then to our face (i.e rubbing eyes or nose). Ongoing pandemic shows that it is much more contagious and spreads fast. Thus wearing a mask is helpful in preventing the infection and spreading of the virus. But the monitoring of mask wearing is a tiresome process and could even lead to infection of the person monitoring. Thus a camera device attached to a door or gate or screening center can detect whether the person is wearing a mask or not and could initiate a response automatically. This could prove to be very helpful in reduction of the spread of virus

02 INTRODUCTION

Iot -Internet of Things is used almost everywhere now. It gives us the capability to monitor certain actions remotely and at times even without human interference. This Project aims at providing a solution to the mask monitoring problem that can be seen at various institutions and organizations where a large number of people usually study or work. The entry of the people can be monitored by the use of IOT devices and Face recognition through AI that can then give instructions to the IOT device to perform certain actions. In this project we have implemented an LED blinking when the person is not wearing a mask is detected during the screening.



Not wearing mask

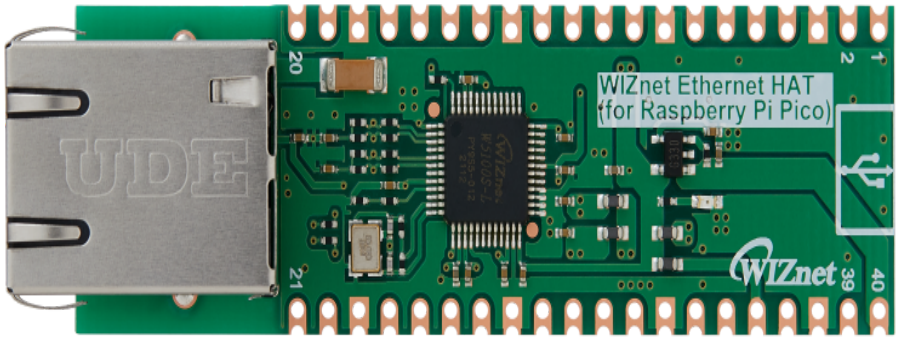
Wearing Mask

03 TOOLS USED

1. Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

2. Raspberry pi Pico board



RP2040 Designed by Raspberry Pi, RP2040 features a dual-core Arm Cortex-M0+ processor with 264KB internal RAM and support for up to 16MB of off-chip Flash. A wide range of flexible I/O options includes I2C, SPI, and — uniquely — Programmable I/O (PIO). These support endless possible applications for this small and affordable package.

3. MQTT

MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT). It is designed as an extremely lightweight publish/subscribe messaging transport that is ideal for connecting remote devices with a small code footprint and minimal network bandwidth. MQTT today is used in a wide variety of industries, such as automotive, manufacturing, telecommunications, oil and gas, etc.

4. LED bulb

Led bulb to indicate whether the person is wearing a mask or not.

5. Jumper wires and Breadboard

To connect the node mcu board and pico board.

04 WORKING OF PROJECT

1. Facemask detection using open-cv

There are two main steps.
-Identify human Face and Mouth in each frame of input video
-Identify Person is using Mask or not
The video is analyzed for facemask and the data after detecting whether the mask is on or off is generated.

2. Ethernet connection to MQTT

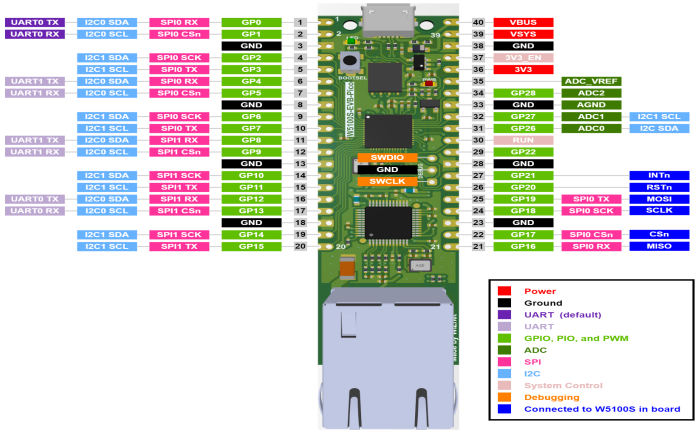
The ethernet connection to the pico board is established using ethernet libraries. Establish an MQTT connection to publish the data obtained from the face detection file i.e *facedetection.py* using ethernet. After connection establishment, send the data as a message.

3. LED blinking using pico board

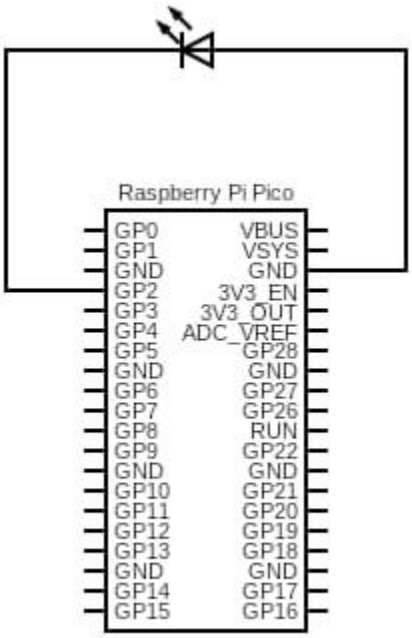
The data subscribed from the MQTT server is fetched to the pico board and the code in the pico board turns on/off the LED bulb based on whether the person is wearing the mask or not.

05 DIAGRAMS

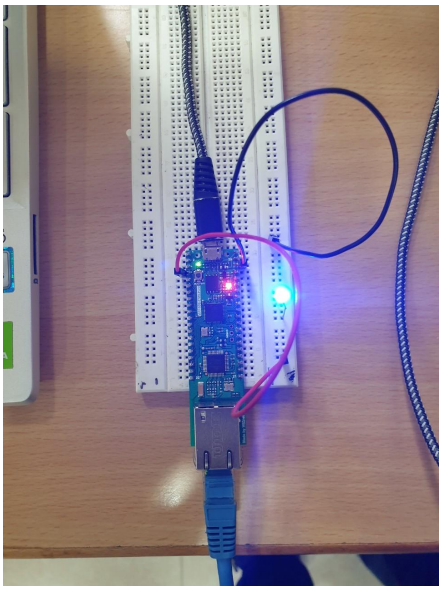
Pin diagram of raspberry pi pico board

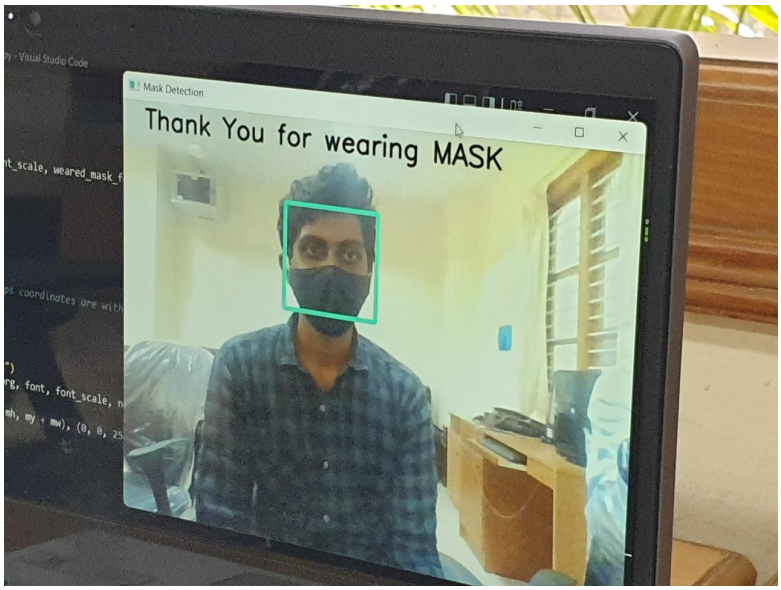


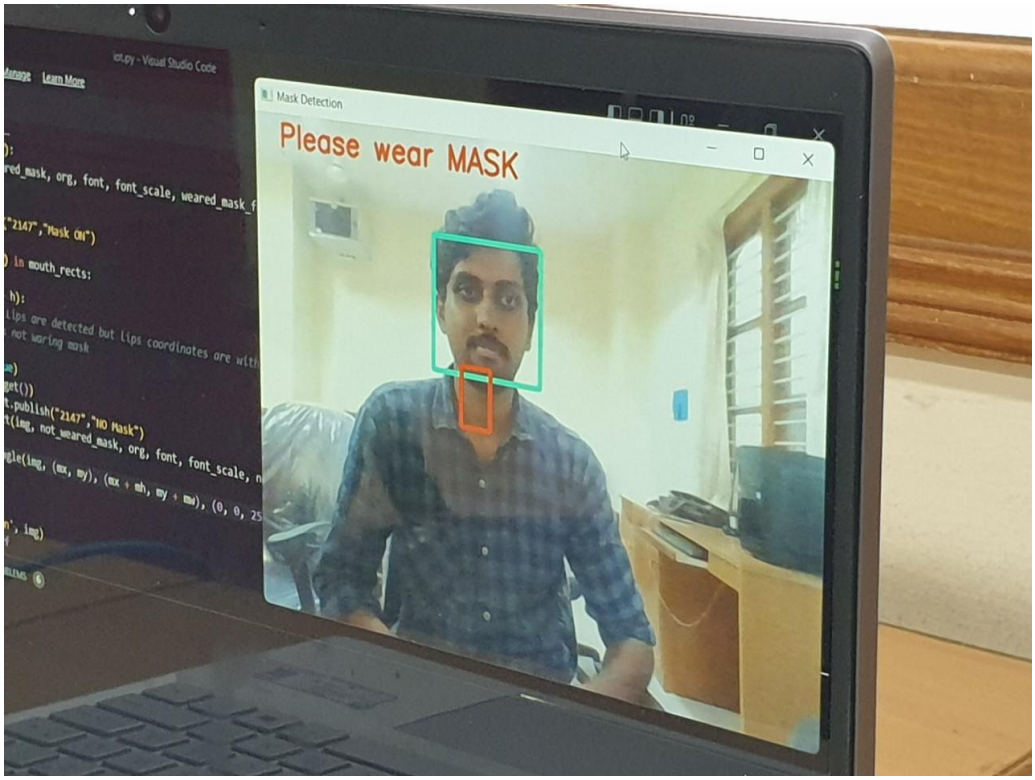
CIRCUIT DIAGRAM



06 IMAGES







07 CODE EXPLANATION

Facemask detection using open-cv in python

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted Cascade of Simple Features .Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

Positive images – These images contain the images which we want our classifier to identify.
Negative Images – Images of everything else, which do not contain the object we want to detect.

```
# import the necessary packages
import numpy as np
import cv2
import random
```

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
import paho.mqtt.client as mqtt

MQTT_HOST = '10.5.15.103'
MQTT_PORT = 1883
MQTT_CLIENT_ID = 'Python MQTT client'
MQTT_USER = 'YOUR MQTT USER'
MQTT_PASSWORD = 'YOUR MQTT USER PASSWORD'
TOPIC = 'home/#'

def on_connect(mqtt_client, user_data, flags, conn_result):
        mqtt_client.subscribe(TOPIC)

def on_message(mqtt_client, user_data, message):
            payload = message.payload.decode('utf-8')

mqtt_client = mqtt.Client(MQTT_CLIENT_ID)
mqtt_client.username_pw_set(MQTT_USER, MQTT_PASSWORD)
mqtt_client.on_connect = on_connect
mqtt_client.on_message = on_message
mqtt_client.connect(MQTT_HOST, MQTT_PORT)


# Fetch the service account key JSON file contents
cred =
credentials.Certificate('face-detection-project-73a1c-firebase-adminsdk-15iqc-77995e76e5.json')
# Initialize the app with a service account, granting admin privileges
firebase_admin.initialize_app(cred, {
            'databaseURL': "https://face-detection-project-73a1c-default-rtdb.firebaseio.com/"
})

ref = db.reference('/facemask')

# initialize a dictionary that maps the name of the haar cascades to
 their filenames
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
mouth_cascade = cv2.CascadeClassifier('haarcascade_mcs_mouth.xml')
upper_body = cv2.CascadeClassifier('haarcascade_upperbody.xml')


# Adjust threshold value in range 80 to 105 based on your light.
```

```
bw_threshold = 80

# User message
font = cv2.FONT_HERSHEY_SIMPLEX
org = (30, 30)
weared_mask_font_color = (0, 0, 0)
not_weared_mask_font_color = (0, 0, 255)
thickness = 2
font_scale = 1
weared_mask = "Thank You for wearing MASK"
not_weared_mask = "Please wear MASK"

# Read video
cap = cv2.VideoCapture(0)

while 1:
  # Get individual frame
ret, img = cap.read()
img = cv2.flip(img,1)

# Convert Image into gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Convert image in black and white
 (thresh, black_and_white) = cv2.threshold(gray, bw_threshold, 255, cv2.THRESH_BINARY)
    #cv2.imshow('black_and_white', black_and_white)

# detect face
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# Face prediction for black and white
faces_bw = face_cascade.detectMultiScale(black_and_white, 1.1, 4)
    #print(ref.get())
    if(len(faces) == 0 and len(faces_bw) == 0):
        cv2.putText(img, "No face found...", org, font, font_scale, weared_mask_font_color,
thickness, cv2.LINE_AA)
    elif(len(faces) == 0 and len(faces_bw) == 1):

 # It has been observed that for white mask covering mouth, with gray image face
prediction is not happening
        cv2.putText(img, weared_mask, org, font, font_scale, weared_mask_font_color, thickness,
cv2.LINE_AA)
        ref.set(False)
        print(ref.get())
```

```
        mqtt_client.publish("2147","Mask ON")
    else:
 # Draw rectangle on face
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (207, 252, 3), 3)
            roi_gray = gray[y:y + h, x:x + w]
            roi_color = img[y:y + h, x:x + w]

 # Detect lips counters
            mouth_rects = mouth_cascade.detectMultiScale(gray, 1.5, 5)
# Face detected but Lips not detected which means person is wearing mask
        if(len(mouth_rects) == 0):
            cv2.putText(img, weared_mask, org, font, font_scale, weared_mask_font_color,
thickness, cv2.LINE_AA)
            ref.set(False)
            print(ref.get())
            mqtt_client.publish("2147","Mask ON")
        else:
            for (mx, my, mw, mh) in mouth_rects:

                if(y < my < y + h):
 # Face and Lips are detected but lips coordinates are within face coordinates which
`means lips prediction is true and # person is not wearing mask

                    ref.set(True)
                    print(ref.get())
                    mqtt_client.publish("2147","NO Mask")
                    cv2.putText(img, not_weared_mask, org, font, font_scale,
not_weared_mask_font_color, thickness, cv2.LINE_AA)

                    cv2.rectangle(img, (mx, my), (mx + mh, my + mw), (0, 0, 255), 3)
                    break

 # Show frame with results
    cv2.imshow('Mask Detection', img)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
# Release video
cap.release()
cv2.destroyAllWindows()
```

Data passing using MQTT through ethernet and LED blinking code - Arduino C

To explain the working of MQTT protocol we will divide the MQTT session into 4 stages such as connection, authentication, communication, and termination.

First of all, a TCP / IP connection is initiated from client to broker by using a standard or custom port which is defined by a broker's operation.
While establishing a connection, it's important to recognize whether the server has continued an old session or a new session.
The old session continues if reused client identity is provided to the broker.

```
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#define LED 2

Update these with values suitable for your network.
byte mac[]    = {  0xDE, 0xED, 0xEA, 0xFE, 0xFE, 0xAA };
IPAddress ip(172, 16, 0, 100);
IPAddress myDns(192, 168, 0, 1);
IPAddress server(10, 5, 15, 103);
EthernetClient ethClient;
PubSubClient client(ethClient);
```

The MQTT client calls a callback method on a separate thread to the main application thread. The client application does not create a thread for the callback, it is created by the MQTT client. The MQTT client synchronizes callback methods. Only one instance of the callback method runs at a time

```
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  char messageBuffer[30];
  memcpy(messageBuffer, payload, length);
  messageBuffer[length] = '\0';
  Serial.println(messageBuffer);
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
  if(strcmp(messageBuffer, "Mask ON") == 0){
```

```
   digitalWrite(LED_BUILTIN, HIGH);
   digitalWrite(LED, HIGH);
    turn the LED on (HIGH is the voltage level)
   delay(500);                    wait for a second
   client.publish("tTopic","Light ON");
   }
   else{
      digitalWrite(LED_BUILTIN, LOW);
      digitalWrite(LED, LOW);
      client.publish("tTopic","Light OFF");turn the LED off by making the voltage LOW
      delay(500);                    wait for a second
      }
}Method used for automatically reconnect.
void reconnect() {
  Loop until we're reconnected
 while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");
  Attempt to connect
  if (client.connect("arduinoClient")) {
    Serial.println("connected");
    Once connected, publish an announcement...
      client.subscribe("2147");
  } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    Wait 5 seconds before retrying
    delay(5000);
  }
 }
}

void setup()
{
 Ethernet.init(17);  WIZnet W5100S-EVB-Pico
 Serial.begin(9600);
 pinMode(LED, OUTPUT);

 while (!Serial) {
  ; wait for the serial port to connect. Needed for native USB port only
 }

  Serial.println("Initialize Ethernet with DHCP:");
 if (Ethernet.begin(mac) == 0) {
```

```
    Serial.println("Failed to configure Ethernet using DHCP");
     Check for Ethernet hardware present
    if (Ethernet.hardwareStatus() == EthernetNoHardware) {
      Serial.println("Ethernet shield was not found.  Sorry,  cannot run without hardware. :(");
      while (true) {
        delay(1); do nothing, no point running without Ethernet hardware
      }
    }
    if (Ethernet.linkStatus() == LinkOFF) {
      Serial.println("Ethernet cable is not connected.");
    }
    try to configure using IP address instead of DHCP:
    Ethernet.begin(mac, ip, myDns);
  } else {
    Serial.print("  DHCP assigned IP ");
    Serial.println(Ethernet.localIP());
  }
  give the Ethernet shield a second to initialize:
  delay(1000);
  Serial.print("connecting to ");
  Serial.print(server);
  Serial.println("...");

  client.setClient(ethClient);
  client.setServer(server, 1883);
  client.setCallback(callback);
  Allow the hardware to sort itself out
  delay(1500);
  pinMode(LED_BUILTIN, OUTPUT);

}
void loop()
{
    delay(1000);
  if (!client.connected()) {
    reconnect();
  }
 client.loop();
```

Firebase json file

JSON is used for transmitting data in web applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

{
 "type": "service_account",

 "project_id": "face-detection-project-73a1c",

  "client_email":
"firebase-adminsdk-15iqc@face-detection-project-73a1c.iam.gserviceaccount.com",
 "client_id": "104620413262830124784",

 "auth_uri": "https://accounts.google.com/o/oauth2/auth",

  "token_uri": "https://oauth2.googleapis.com/token",

 "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-15iqc%40face-detection-project-73a1c.iam.gserviceaccount.com"

}

08 REFERENCES

https://create.arduino.cc/projecthub/karem_benchikha/facemask-detection-88fa1