**2 DISTANCE , TEMP, HUMIDITY**

import RPi.GPIO as GPIO

import Adafruit\_DHT

import time

import sys

GPIO.setwarnings(False)

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#GPIO Pins(Front)

GPIO\_TRIGGER\_F = 18

GPIO\_ECHO\_F = 24

GPIO\_TRIGGER\_B = 20

GPIO\_ECHO\_B = 21

#GPIO Pins(Back)

#GPIO Pin (IN / OUT)

GPIO.setup(GPIO\_TRIGGER\_F, GPIO.OUT)

GPIO.setup(GPIO\_ECHO\_F, GPIO.IN)

GPIO.setup(GPIO\_TRIGGER\_B, GPIO.OUT)

GPIO.setup(GPIO\_ECHO\_B, GPIO.IN)

def distance\_front():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER\_F, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER\_F, False)

StartTime\_F = time.time()

StopTime\_F = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO\_F) == 0:

StartTime\_F = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO\_F) == 1:

StopTime\_F = time.time()

# time difference between start and arrival

TimeElapsed\_F = StopTime\_F - StartTime\_F

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance\_f = (TimeElapsed\_F \* 34300) / 2

return distance\_f

def distance\_back():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER\_B, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER\_B, False)

StartTime\_B = time.time()

StopTime\_B = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO\_B) == 0:

StartTime\_B = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO\_B) == 1:

StopTime\_B = time.time()

# time difference between start and arrival

TimeElapsed\_B = StopTime\_B - StartTime\_B

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance\_b = (TimeElapsed\_B \* 34300) / 2

return distance\_b

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

distFront = distance\_front()

print ("Front Distance = %.1f cm" % distFront)

distBack = distance\_back()

print ("Back Distance = %.1f cm" % distBack)

print("------------------------------")

humidity, temperature = Adafruit\_DHT.read\_retry(11, 4)

print('Temp: {0:0.1f} C Humidity: {1:0.1f} %'.format(temperature, humidity))

print("==============================")

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

**DOUBLE ULTRASONIC**

import RPi.GPIO as GPIO

import time

GPIO.setwarnings(False)

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#GPIO Pins(Front)

GPIO\_TRIGGER\_F = 18

GPIO\_ECHO\_F = 24

GPIO\_TRIGGER\_B = 20

GPIO\_ECHO\_B = 21

#GPIO Pins(Back)

#GPIO Pin (IN / OUT)

GPIO.setup(GPIO\_TRIGGER\_F, GPIO.OUT)

GPIO.setup(GPIO\_ECHO\_F, GPIO.IN)

GPIO.setup(GPIO\_TRIGGER\_B, GPIO.OUT)

GPIO.setup(GPIO\_ECHO\_B, GPIO.IN)

def distance\_front():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER\_F, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER\_F, False)

StartTime\_F = time.time()

StopTime\_F = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO\_F) == 0:

StartTime\_F = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO\_F) == 1:

StopTime\_F = time.time()

# time difference between start and arrival

TimeElapsed\_F = StopTime\_F - StartTime\_F

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance\_f = (TimeElapsed\_F \* 34300) / 2

return distance\_f

def distance\_back():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER\_B, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER\_B, False)

StartTime\_B = time.time()

StopTime\_B = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO\_B) == 0:

StartTime\_B = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO\_B) == 1:

StopTime\_B = time.time()

# time difference between start and arrival

TimeElapsed\_B = StopTime\_B - StartTime\_B

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance\_b = (TimeElapsed\_B \* 34300) / 2

return distance\_b

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

distFront = distance\_front()

print ("Front Distance = %.1f cm" % distFront)

distBack = distance\_back()

print ("Back Distance = %.1f cm" % distBack)

print("==============================")

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

**CLOUD IMAGE UPLOAD (Firebase)**

import RPi.GPIO as GPIO

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

# BCM Mode 15

GPIO.setup(10, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)

from datetime import datetime

from picamera import PiCamera

from time import sleep

import os

import pyrebase

firebaseConfig = {

'apiKey': "AIzaSyBmso63O2WQm7wf\_itHaFOCM77LyZ8Z2ag",

'authDomain': "fyp1-86727.firebaseapp.com",

'databaseURL': "<https://fyp1-86727-default-rtdb.firebaseio.com>",

'projectId': "fyp1-86727",

'storageBucket': "fyp1-86727.appspot.com",

'messagingSenderId': "278070492320",

'appId': "1:278070492320:web:c8d6b66f77d09012d85dc5",

}

firebase = pyrebase.initialize\_app(firebaseConfig)

storage = firebase.storage()

camera = PiCamera()

while True:

try:

for i in range (0,5):

# if GPIO.input(10) == GPIO.HIGH:

print("pushed")

now = datetime.now()

dt = now.strftime("%d%m%Y%H:%M:%S")

name = dt+".jpg"

camera.capture(name)

print(name+" saved")

storage.child(name).put(name)

print("Image sent")

os.remove(name)

print("File Removed")

sleep(2)

except:

camera.close()

**LIVE FEED on 192.168.43.71:8000**

# http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming

import io

import picamera

import logging

import socketserver

from threading import Condition

from http import server

PAGE="""\

<html>

<head>

<title>Raspberry Pi - Surveillance Camera</title>

</head>

<body>

<center><h1>Raspberry Pi - Surveillance Camera</h1></center>

<center><img src="stream.mjpg" width="640" height="480"></center>

</body>

</html>

"""

class StreamingOutput(object):

def \_\_init\_\_(self):

self.frame = None

self.buffer = io.BytesIO()

self.condition = Condition()

def write(self, buf):

if buf.startswith(b'\xff\xd8'):

# New frame, copy the existing buffer's content and notify all

# clients it's available

self.buffer.truncate()

with self.condition:

self.frame = self.buffer.getvalue()

self.condition.notify\_all()

self.buffer.seek(0)

return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):

def do\_GET(self):

if self.path == '/':

self.send\_response(301)

self.send\_header('Location', '/index.html')

self.end\_headers()

elif self.path == '/index.html':

content = PAGE.encode('utf-8')

self.send\_response(200)

self.send\_header('Content-Type', 'text/html')

self.send\_header('Content-Length', len(content))

self.end\_headers()

self.wfile.write(content)

elif self.path == '/stream.mjpg':

self.send\_response(200)

self.send\_header('Age', 0)

self.send\_header('Cache-Control', 'no-cache, private')

self.send\_header('Pragma', 'no-cache')

self.send\_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')

self.end\_headers()

try:

while True:

with output.condition:

output.condition.wait()

frame = output.frame

self.wfile.write(b'--FRAME\r\n')

self.send\_header('Content-Type', 'image/jpeg')

self.send\_header('Content-Length', len(frame))

self.end\_headers()

self.wfile.write(frame)

self.wfile.write(b'\r\n')

except Exception as e:

logging.warning(

'Removed streaming client %s: %s',

self.client\_address, str(e))

else:

self.send\_error(404)

self.end\_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):

allow\_reuse\_address = True

daemon\_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:

output = StreamingOutput()

#Uncomment the next line to change your Pi's Camera rotation (in degrees)

#camera.rotation = 90

camera.start\_recording(output, format='mjpeg')

try:

address = ('', 8000)

server = StreamingServer(address, StreamingHandler)

server.serve\_forever()

finally:

camera.stop\_recording()

**ULTRASONIC**

#Libraries

import RPi.GPIO as GPIO

import time

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

**SERVO MOTOR**

String command;

#define blueLed 8

#define whiteLed 9

#define redLed 10

void setup() {

Serial.begin(9600);

pinMode(2,OUTPUT); //right motors forward

pinMode(3,OUTPUT); //right motors reverse

pinMode(4,OUTPUT); //left motors reverse

pinMode(5,OUTPUT); //left motors forward

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

delay(1000);

Serial.println("Type Command (forward, backward, left, right, stop)");

}

void loop() {

if (Serial.available()) {

command = Serial.readStringUntil('\n');

command.trim();

if (command.equals("forward")) { //move forward(all motors rotate in forward direction)

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(2,HIGH);

digitalWrite(5,HIGH);

}

else if (command.equals("backward")) { //move reverse (all motors rotate in reverse direction)

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

}

else if (command.equals("left")) { //turn left (right side motors rotate in forward direction, left side motors doesn't rotate)

digitalWrite(4,LOW);

digitalWrite(3,HIGH);

}

else if (command.equals("right")) { //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

}

else if (command.equals("stop")) { //turn right (left side motors rotate in forward direction, right side motors doesn't rotate)

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

}

Serial.print("Command: ");

Serial.println(command);

}

}

Integration Trial:

from numpy import divide

import RPi.GPIO as GPIO

import time

from firebase import firebase

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(10, GPIO.IN, pull\_up\_down=GPIO.PUD\_DOWN)

from datetime import datetime

from picamera import PiCamera

from time import sleep

import os

import pyrebase

firebaseConfig = {

'apiKey': "AIzaSyBmso63O2WQm7wf\_itHaFOCM77LyZ8Z2ag",

'authDomain': "fyp1-86727.firebaseapp.com",

'databaseURL': "https://fyp1-86727-default-rtdb.firebaseio.com",

'projectId': "fyp1-86727",

'storageBucket': "fyp1-86727.appspot.com",

'messagingSenderId': "278070492320",

'appId': "1:278070492320:web:c8d6b66f77d09012d85dc5",

}

firebase = pyrebase.initialize\_app(firebaseConfig)

storage = firebase.storage()

camera = PiCamera()

fbCon = firebase.FirebaseApplication('https://fyp1-86727-default-rtdb.firebaseio.com/', None)

GPIO.setmode(GPIO.BCM)

GPIO\_TRIG = 11

GPIO\_ECHO = 18

GPIO.setup(GPIO\_TRIG, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

GPIO.output(GPIO\_TRIG, GPIO.LOW)

Time.sleep(2)

GPIO.output(GPIO\_TRIG, GPIO.HIGH)

Time.sleep(0.00001)

GPIO.output(GPIO\_TRIG, GPIO.LOW)

while GPIO.input(GPIO\_ECHO)==0:

start\_time = time.time()

while GPIO.input(GPIO\_ECHO)==1:

Bounce\_back\_time = time.time()

pulse\_duration = Bounce\_back\_time - start\_time

distance = round(pulse\_duration \* 17150, 2)

data = {

'Distance' : distance

}

result = fbCon.post('TestData', data)

print (f"Distance: {distance} cm")

if (distance < 15):

try:

for i in range (0,5):

# if GPIO.input(10) == GPIO.HIGH:

print("pushed")

now = datetime.now()

dt = now.strftime("%d%m%Y%H:%M:%S")

name = dt+".jpg"

camera.capture(name)

print(name+" saved")

storage.child(name).put(name)

print("Image sent")

os.remove(name)

print("File Removed")

sleep(2)

except:

camera.close()

GPIO.cleanup()

—-----------------------

pip3 install git+https://github.com/ozgur/python-firebase

pip3 install requests

pip3 install python-firebase

from firebase.firebase import FirebaseApplication

—----------------------

**INTEGRATION FIREBASE(OVERWRITE)**

import RPi.GPIO as GPIO

import time

import pyrebase

firebaseConfig = {

'apiKey': "AIzaSyBmso63O2WQm7wf\_itHaFOCM77LyZ8Z2ag",

'authDomain': "fyp1-86727.firebaseapp.com",

'databaseURL': "https://fyp1-86727-default-rtdb.firebaseio.com",

'projectId': "fyp1-86727",

'storageBucket': "fyp1-86727.appspot.com",

'messagingSenderId': "278070492320",

'appId': "1:278070492320:web:c8d6b66f77d09012d85dc5",

}

firebase = pyrebase.initialize\_app(firebaseConfig)

storage = firebase.storage()

database = firebase.database()

def senddata(dist):

database.child("Distance")

data = {"Distance": dist}

database.set(dist)

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

dist1 = int(dist)

if dist1 < 15:

print('Less than 15cm...')

senddata(str(dist))

print('Sent Data')

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

**INTEGRATION FIREBASE(IMAGE+ULTRASONIC)**

import RPi.GPIO as GPIO

import time

import pyrebase

from datetime import datetime

from picamera import PiCamera

from time import sleep

import os

firebaseConfig = {

'apiKey': "AIzaSyBmso63O2WQm7wf\_itHaFOCM77LyZ8Z2ag",

'authDomain': "fyp1-86727.firebaseapp.com",

'databaseURL': "https://fyp1-86727-default-rtdb.firebaseio.com",

'projectId': "fyp1-86727",

'storageBucket': "fyp1-86727.appspot.com",

'messagingSenderId': "278070492320",

'appId': "1:278070492320:web:c8d6b66f77d09012d85dc5",

}

firebase = pyrebase.initialize\_app(firebaseConfig)

storage = firebase.storage()

database = firebase.database()

camera = PiCamera()

def senddata(dist):

# try:

now = datetime.now()

dt = now.strftime("%d%m%Y%H:%M:%S")

img = dt + ".jpg"

camera.capture(img)

print("Captured")

database.child(dt)

data = {dt: dist}

database.set(data)

storage.child(img).put(img)

print("Image sent")

os.remove(img)

print("File Removed")

#except:

# camera.close()

#GPIO Mode (BOARD / BCM)

GPIO.setmode(GPIO.BCM)

#set GPIO Pins

GPIO\_TRIGGER = 18

GPIO\_ECHO = 24

#set GPIO direction (IN / OUT)

GPIO.setup(GPIO\_TRIGGER, GPIO.OUT)

GPIO.setup(GPIO\_ECHO, GPIO.IN)

def distance():

# set Trigger to HIGH

GPIO.output(GPIO\_TRIGGER, True)

# set Trigger after 0.01ms to LOW

time.sleep(0.00001)

GPIO.output(GPIO\_TRIGGER, False)

StartTime = time.time()

StopTime = time.time()

# save StartTime

while GPIO.input(GPIO\_ECHO) == 0:

StartTime = time.time()

# save time of arrival

while GPIO.input(GPIO\_ECHO) == 1:

StopTime = time.time()

# time difference between start and arrival

TimeElapsed = StopTime - StartTime

# multiply with the sonic speed (34300 cm/s)

# and divide by 2, because there and back

distance = (TimeElapsed \* 34300) / 2

return distance

if \_\_name\_\_ == '\_\_main\_\_':

try:

while True:

dist = distance()

print ("Measured Distance = %.1f cm" % dist)

dist1 = int(dist)

if dist1 < 15:

print('Less than 15cm...')

senddata(str(dist))

print('Sent Data')

time.sleep(1)

# Reset by pressing CTRL + C

except KeyboardInterrupt:

print("Measurement stopped by User")

GPIO.cleanup()

**YOUTUBE LIVE STREAM**

* raspivid -o - -t 0 -vf -hf -fps 30 -b 6000000 | avconv -re -ar 44100 -ac 2 -acodec pcm\_s16le -f s16le -ac 2 -i /dev/zero -f h264 -i - -vcodec copy -acodec aac -ab 128k -g 50 -strict experimental -f flv rtmp://a.rtmp.youtube.com/live2/[your-secret-key-here]
* raspivid -o - -t 0 -vf -hf -fps 30 -b 6000000 | ffmpeg -re -ar 44100 -ac 2 -acodec pcm\_s16le -f s16le -ac 2 -i /dev/zero-f h264 -i - -vcodec copy -acodec aac -ab 128k -g 50 -strict experimental -f flv [SERVER URL]/[STREAM NAME]

**LIVE FEED on the WEB**

cd Downloads

./ngrok http http://192.168.43.71:8000