****

**Princess Sumaya University for Technology**

**Department of Data Science**

**Robotics Programming Course**

Surveillance system using raspberry pi

Done by:

Sara Abbas 20190716

Nada Fayyad 20180711

Jana Salti 20190230

**1. Abstract**

Surveillance systems are one of the most significant security measures that can be implemented in any setting, whether it's a house, a workplace, or a public place. This project makes use of a Raspberry pi microprocessor and its camera module to implement a small scale surveillance system. This research paper will walk you through the design and implementation of such a system.

**2. Introduction**

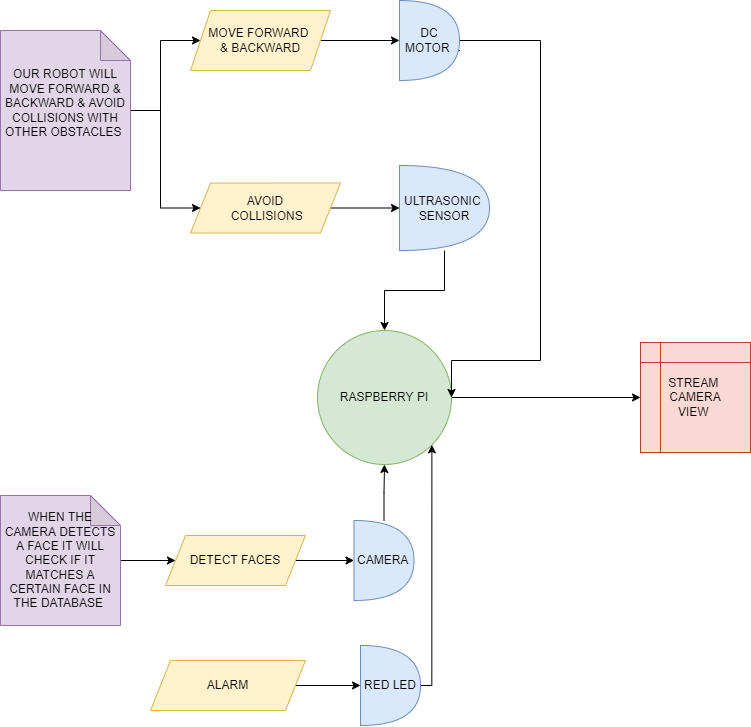
This project makes use of raspberry pi and its camera module to capture wanted faces and match them to a certain predefined database. We will be using an ssh connection to connect between the raspberry pi and the desktop we’ll be monitoring the camera from. Through the ssh connection, we’ll also be sending python files to the raspberry pi to perform certain tasks and functionalities like object avoidance using an ultrasonic sensor, face detection and recognition using the camera module.

**3. Project design and implementation**

**3.1 Design**

Table 3.1 GPIO Pin requirements

| Mission | Components | Pin Requirements [GPIO] |
| --- | --- | --- |
| Moves forwards and backwards | DC motor | 4 pins |
| Avoids collisions | Ultrasonic sensor | 2 pins [1 on trig l 1 on echo] |
| Alarm system | Red LED | 2 pins |
| Detect faces | Camera | 0 pins |



*Figure 3.1 Flowchart of surveillance system using raspberry pi*

**3.2 Implementation**

Our robot will be able to move in four different directions; forward, backward, left and right. While it’s moving, the ultrasonic sensor will be calculating the distance between it and any obstacle. If the obstacle is almost in 10cm range, the robot will move backwards and steer away from the object. The camera module on our robot will be able to detect any face, if it recognizes the face that matches any of the faces within a database, an alarm will be issued. A red LED will start blinking on and off and a stand still image will be taken of the detected person. The alarm message will be sent via the python SMTP email service and the image of the detected person will be attached to it. This email will be sent to whomever this surveillance system may concern.

**3.2.1 Object Avoidance**

For our robot to move forward and backward, 2 dc motors were used which were connected to a motor driver (L298N was used) powered up by 4 cells of 1.5v batteries. This motor driver was connected to the raspberry pi GPIO pins. For the object avoidance functionality, we decided to use an ultrasonic sensor since it was low budget and readily available. We found the ultrasonic sensor more suitable for object collision than an IR sensor because we could specify a certain distance range using an ultrasonic sensor, as for an IR sensor, it is more suitable for line following and different functionalities that don’t require certain distance range flexibility.

The object avoidance functionality was that when the ultrasonic sensor on the front of our robot detects an object that is within 10cm range, it moves backwards then rotates to the right.

Our main challenge was with how fragile the yellow dc motors are and how easily the wires connecting them to the h bridge could get detached. We tried soldering them but still had some difficulty with their circuitry. We have worked the 2 dc motors before in previous projects and showed video proof that we know how to work them. As for this project, the hardware has failed us to implement object avoidance to its full potential.

**3.2.1a Code**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BCM)

in1=16

in2=12

in3=21

in4=20

TRIG = 17

ECHO = 27

GPIO.setup(TRIG,GPIO.OUT)

GPIO.setup(ECHO,GPIO.IN)

GPIO.setup(in1,GPIO.OUT)

GPIO.setup(in2,GPIO.OUT)

GPIO.setup(in3,GPIO.OUT)

GPIO.setup(in4,GPIO.OUT)

time.sleep(5)

def stop():

GPIO.output(in1, 0)

GPIO.output(in2, 0)

GPIO.output(in3, 0)

GPIO.output(in4, 0)

def forward():

GPIO.output(in1, 1)

GPIO.output(in2, 0)

GPIO.output(in3, 1)

GPIO.output(in4, 0)

def back():

GPIO.output(in1, 0)

GPIO.output(in2, 1)

GPIO.output(in3, 0)

GPIO.output(in4, 1)

def left():

GPIO.output(in1, 0)

GPIO.output(in2, 0)

GPIO.output(in3, 1)

GPIO.output(in4, 0)

def right():

GPIO.output(in1, 1)

GPIO.output(in2, 0)

GPIO.output(in3, 0)

GPIO.output(in4, 0)

while True:

i=0

avgDistance=0

for i in range(5):

GPIO.output(TRIG, False)

time.sleep(0.1)

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

while GPIO.input(echo)==0:

pulse\_start=time.time()

while GPIO.input(echo)==1:

pulse\_end=time.time()

pulse\_duration = pulse\_end - pulse\_start

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

avgDistance=avgDistance+distance

avgDistance=avgDistance/5

if avgDistance < 10:

stop()

time.sleep(1)

back()

time.sleep(1.5)

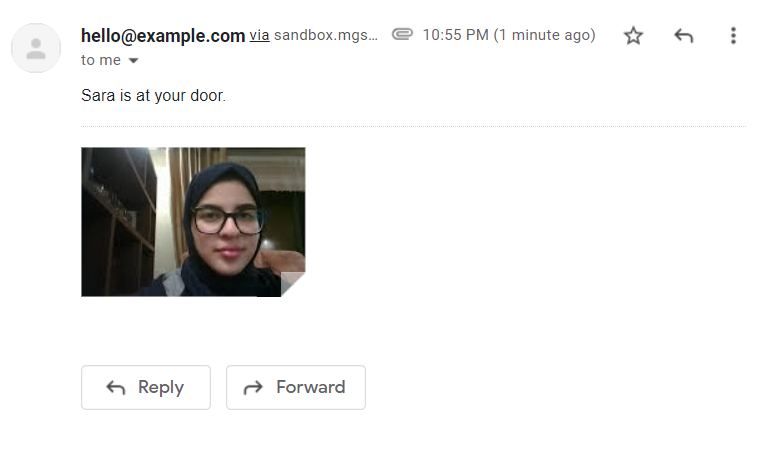
right()

else:

forward()

**3.2.2 Face detection and recognition**

We used mobaxterm instead of putty and filezilla to see the camera view when we run the code. We found mobaxterm more suitable as it included a terminal as well as gui and file exchange. We ran the python file headshots.py which took 20 images for the same person. We then ran train\_model.py to train our model on these 20 images, then we opened a live view of the camera using face\_req.py to see if the face was being detected and recognized. Running the python file face\_req\_email.py, If the face matches the images in the database, the raspberry pi sends an email using mailgun API with the recognized face attached. The files we ran for this part of the project were all included in the following github repository. <https://github.com/carolinedunn/facial_recognition>



*Screenshot of email sent using raspberry pi*

Attached to our submission is a video that shows how the face is detected and recognized while any other face is considered and labeled as ‘Unknown’.

**4. Conclusion**

While working on our robot we faced many problems regarding the programming, the hardware and the software, but we finally reached our goal where our robot was able to achieve our main task that is detecting and recognizing faces whether they are in the database or not and sending an email when the face is detected and recognized. However, our robot was approximately 70% - 80% efficient as sometimes the robot matches unknown faces to faces that are saved in the database due to lighting conditions and similarity. We think that this problem can be fixed by training the robot on more images of the same face with different light settings and angles.