**PROJECT REPORT**

**OBJECTIVE:**

Our project aim is to create a **Door Monitor System** using a Raspberry Pi 4which would inform the user of when the door is left open using email. We would be using an ultrasonic sensor to detect an object (stopper) in front of the door and use shell scripting to program the sensor. We believe this project would help achieve a higher level of security, especially in places like restricted areas.

**INTRODUCTION:**

What is a Raspberry Pi 4? A Raspberry Pi 4 is a dual-display, single-board computer and is widely used in automation and robotics. Its wide use is mostly based on its portability, speed, and performance.

We would be placing an ultrasonic sensor alongside the door. The ultrasonic sensor will measure the distance between the object (stopper) and the sensor by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected from the target. Ultrasonic sensors measure the distance to the target by measuring the time between the emission and reception. This sensor will monitor if the distance increases between the sensor and the stopper by a certain length, if there is, the Raspberry Pi would notify the user by email, so that they may close the door accordingly or know if someone is entering the room. The Raspberry Pi will also alert the user if the door is left open for more than a certain length of time (e.g more than 10 secs).

**EQUIPMENT:**

* Raspberry Pi 4 Model B
* USB 16GB
* Ultrasonic Sensor (JSN-SR04T-2.0) with cable
* Jumper wires
* Power Cable (Type C)
* MicroUSB to VGA Convertor
* VGA to VGA cable (for extension)
* Peripheral devices (keyboard, mouse, and monitor)
* Resistors
* Breadboard

**BACKGROUND:**

The background of this Raspberry Pi project utilizing an ultrasonic sensor to monitor door activity revolves around the need for enhanced security measures. The project aims to leverage the capabilities of a Raspberry Pi, combined with an ultrasonic sensor, to create a reliable and cost-effective solution. By continuously monitoring the distance between the sensor and the door, the system can detect any movement or changes, alerting users to potential unauthorized entries. This project seeks to provide an efficient and affordable security solution that can be easily implemented in various settings, ranging from residential homes to small businesses, ensuring increased peace of mind and improved overall security.

**PLATFORM & LANGUAGES:**

For this project, the platform we are using is Raspberry Pi Operating System (32-bit) which is a Linux based Operating System. The language used is Bourne Script (.sh).

**METHODOLOGY:**

The Door Monitor System works by getting input from the ultrasonic sensor about the distance between the stopper and the sensor, and at any instance the distance increases above the set criteria the user is informed about the door activity through mail, along with a history of past activities.

We started our project by first setting up the **Raspberry Pi 4** with **Raspberry Pi Operating System (32-bit)** through Raspberry Pi Imager installer.

A screenshot of a computer

Description automatically generated with medium confidence

Before coding we installed the following packages:

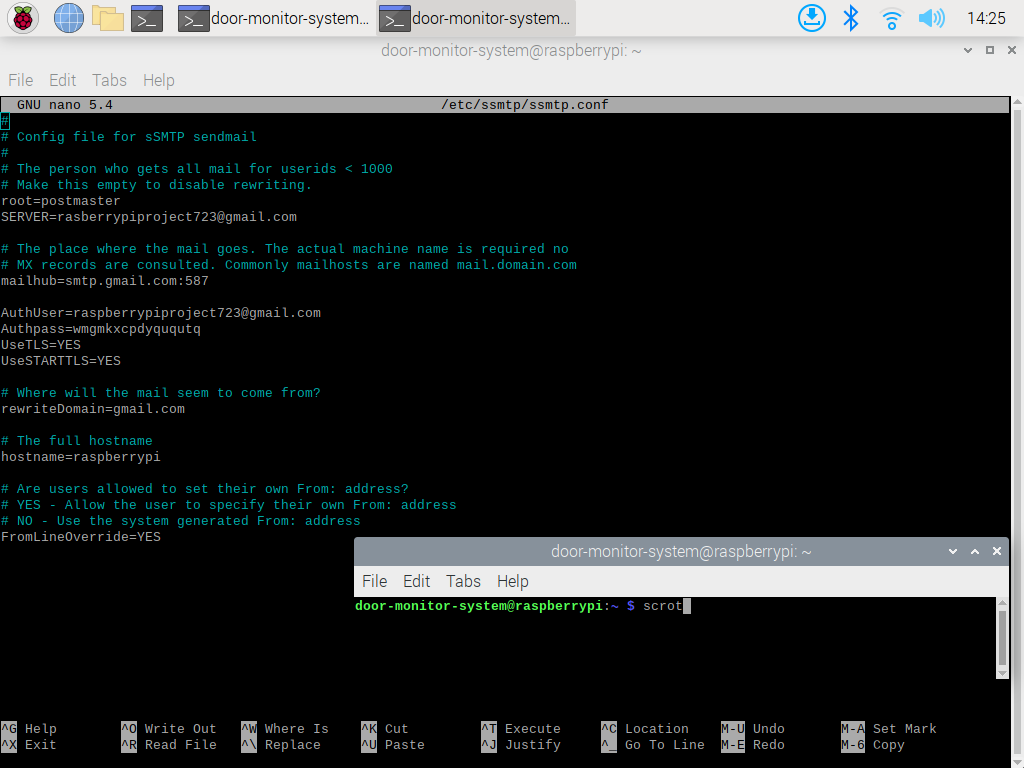
sudo apt-get install wiringpi

sudo apt-get install mailutils

sudo apt-get install ssmtp

* **wiringpi** package is used to install an interface between the GPIO pins of the raspberry pi and the code.
* **mailutils** package is used to cater mailing services.
* **ssmpt** package is used to cater mailing services.

After installation of these packages, we configured the ssmpt config file and Gmail account settings to suit our needs for emailing purposes. We did this as such:



After the configuration was complete, we coded a shell script to send an email to user through the Gmail account [**raspberrypiproject723@gmail.com**](mailto:raspberrypiproject723@gmail.com)**.**

echo "$BODY” | mail -s "$SUBJECT" -A history.txt "$TO"

The above shell script uses **mail** command to send the string variable $**BODY** as the body of the mail, **$SUBJECT** as subject of mail and uses **$TO** string variableas the receiver’s mail. The **-A** mode in mail sends any attachments along with the mail which are specified after it.

Next a shell script was required to read the input from the sensor. Following is the code for reading data from the sensor,

gpio -g read $ECHO

The **gpio -g** command sets the General Input Output Pins(GPIO) pins of the **Raspberry Pi** to read and write mode.

TRIG=26     #pin number

ECHO=18     #pin number

#setting sensor

gpio -g mode $TRIG out

gpio -g mode $ECHO in

gpio -g write $TRIG 0

sleep 0.1

gpio -g write $TRIG 1

sleep 0.00001

gpio -g write $TRIG 0

while [ $(gpio -g read $ECHO) -eq 0 ];

do

START=$(date +%N)

done

while [ $(gpio -g read $ECHO) -eq 0 ];

do

       END=$(date +%N)

done

We set pin number **26** and **18** to **$TRIG(write mode)** and **$ECHO(read mode)** respectively. A small duration pulse is sent to the **$TRIG** pin (which is in write mode) to activate the sensor. A while loop starts by reading the input from the **$ECHO** pin and continues until the input is not equal to 0 (i.e., input is 1), at which point it has noted the starting time of the pulse in nanoseconds in **$START** variable. A second while loop starts immediately after the reading is 1 and continues until it stays 1, meaning the pulse from the sensor has not returned and becomes immediately 0 after it is received at which point the ending time of the pulse is recorded in the **$END** variable.

Next, we use **awk -v** commands, which are part of any Linux based operating system, to perform arithmetic operations on complex and long integers. The distance between the stopper and sensor is calculated via the operation,

DURATION=$(awk -v st="$START" -v et="$END" 'BEGIN { printf "%.9f", et - st }')  #calculating duration wave

    dur=$(awk -v d1="$DURATION" -v conv="1000000000" 'BEGIN { printf "%.9f", d1 / conv }')  #converting nanoseconds to seconds

    DISTANCE=$(awk -v d="$dur" -v sp="17150" 'BEGIN { printf "%.9f", d \* sp }') #calculating distance

    DISTANCE=$(printf "%.0f" "$DISTANCE") #converting to integer

Since the pulse travelled once to the stopper and then back, we divided the speed of sound in air by 2.

The following diagram better describes how the cycles of **$TRIG**  and **$ECHO** work**,**

**A red and blue line on a black background

Description automatically generated with medium confidence**

A small duration pulse triggers the sensor which sends a pulse out making the **$ECHO** pin to send 1 as reading. As the pulse is returned back to the sensor the **$ECHO** pin becomes 0 again, and we calculate the distance by observing the time taken for pulse to travel.

#!/bin/bash

#variables

COUNT=0

DISTANCE=0  #distance b/w sensor and stopper

DURATION=0  #duration of wave transmitted

TRIG=26     #pin number

ECHO=18     #pin number

START=0

END=0

DATE=$(date "+%d-%b-%y")

DAY=$(date "+%A")

TO="k213352@nu.edu.pk"      #user email

SUBJECT="Door Monitor System"       #subject of email

#setting sensor

gpio -g mode $TRIG out

gpio -g mode $ECHO in

sleep 1

#code

while true;

do

    gpio -g write $TRIG 0

    sleep 0.1

    gpio -g write $TRIG 1

    sleep 0.00001

    gpio -g write $TRIG 0

    while [ $(gpio -g read $ECHO) -eq 0 ];

    do

        START=$(date +%N)

    done

    while [ $(gpio -g read $ECHO) -eq 0 ];

    do

        END=$(date +%N)

    done

    DURATION=$(awk -v st="$START" -v et="$END" 'BEGIN { printf "%.9f", et - st }')  #calculating duration wave

    dur=$(awk -v d1="$DURATION" -v conv="1000000000" 'BEGIN { printf "%.9f", d1 / conv }')  #converting nanoseconds to seconds

    DISTANCE=$(awk -v d="$dur" -v sp="17150" 'BEGIN { printf "%.9f", d \* sp }') #calculating distance

    DISTANCE=$(printf "%.0f" "$DISTANCE") #converting to integer

    if [[ $DISTANCE -gt 5 ]];   #checks distance criteria

    then

        if [[ $COUNT -eq 0 ]];      #check to send mail only once

        then

            echo 1 > door\_status.txt    #sets door to open status, validates countdown

            ./countdown.sh &    #& operator runs ./countdown.sh command parallely in background to account for if the door is opened for too long

            TIME=$(date "+%T")

            echo -e "$DATE \t $DAY \t $TIME \t\t Opened\n" >> history.txt     #stores status of door in history

            echo "Door is opened." | mail -s "$SUBJECT" -A history.txt "$TO"    #this commands send email

            COUNT=1

        fi

    else

        if [[ $COUNT -eq 1 ]];      #check to send mail only once

        then

            echo 0 > door\_status.txt    #informs countedown to stop counting since door is closed

            TIME=$(date "+%T")

            echo -e "$DATE \t $DAY \t $TIME \t\t Closed\n" >> history.txt     #stores status of door in history

            echo "Door is closed." | mail -s "$SUBJECT" -A history.txt "$TO"    #this commands send email

            COUNT=0

        fi

    fi

    sleep 1

done

We then combined both of our shell scripts to work together so whenever the distance is greater than **5cm** the activity is stored in a text file named **history.txt**, which tabs the time, date, day, and status of the door and sends and email to the user with the text file as an attachment. Another shell script named **countdown.sh** checks if the door remains open for more than **10 seconds**, by checking the status from **door\_status.txt** (it contains either 1, for open, or 0, for close), if so, it sends another email to the user, alerting it to check for any suspicious activity. The **& operator** along with **./countdown.sh** command helps us to run the shell script in the background, achieving multitasking. The entire process continues after a 1 second interval.

* **countdown.sh**

for i in {10..1}

do

    read status < door\_status.txt

    sleep 1

done

if [[ status -eq 1 ]];

then

    echo "Door is open for more than 10 seconds. Kindly check for any suspicious activity." | mail -s "ALERT! Door Monitor System" "k213352@nu.edu.pk"

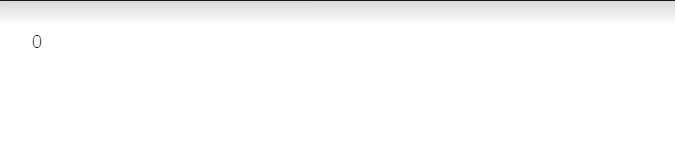
fi

* **history.txt**

A screenshot of a computer

Description automatically generated with low confidence

* **door\_status.txt**



* **Emails**

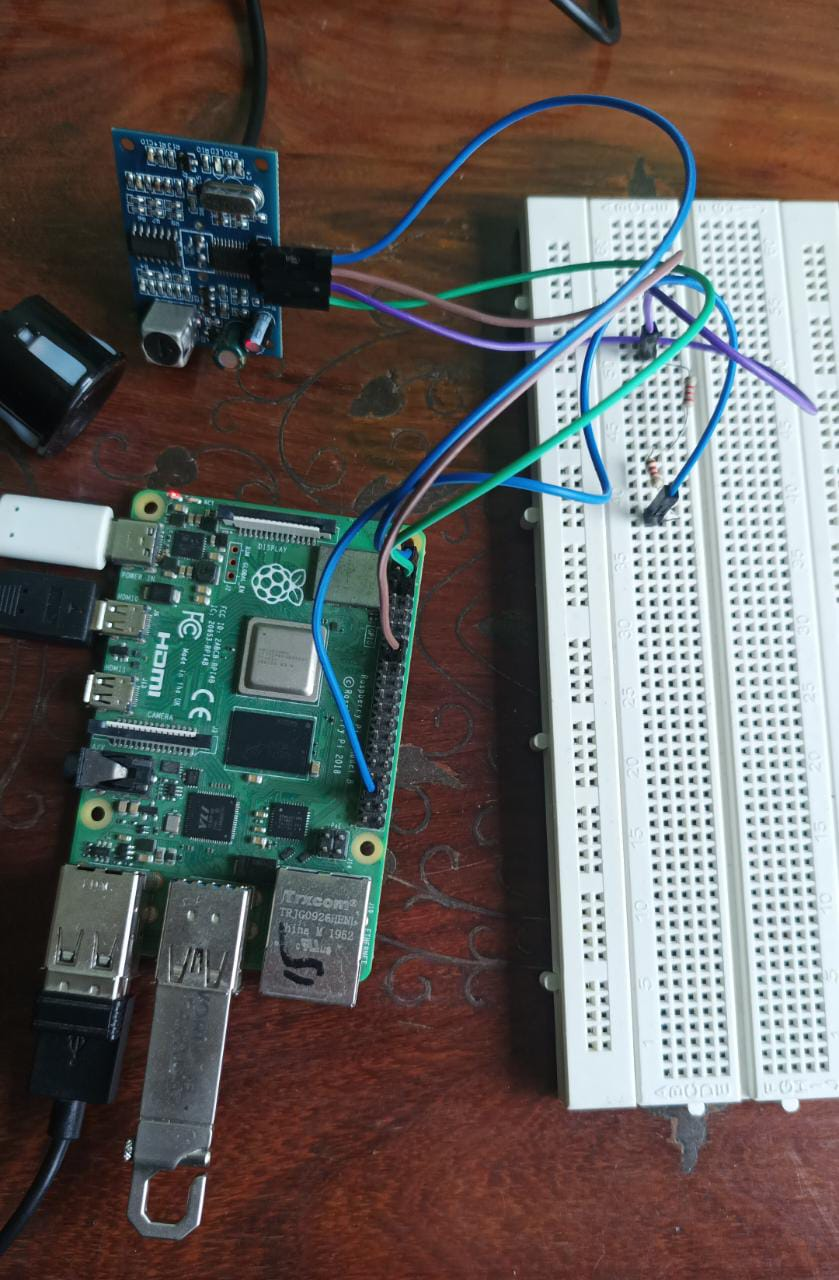
A screenshot of a computer

Description automatically generated with low confidence

A picture containing text, screenshot, font, businesscard

Description automatically generated

Following is the setup consisting of **Raspberry Pi**, **JSN Ultrasonic sensor**, **resistors**, **jumper wires**, **breadboard.** Resistors are used to reduce the **5-voltage** output of the sensor from the **echo** pin to **3.3 voltage**, which is the maximum voltage accepted the **GPIO** pins of the **Raspberry Pi.**

****