Perimeter Intrusion detection system.

Michael Awoyemi

CMP408: IoT and Cloud Secure Development

2024/25

.

Contents

1 Introduction 1

1.1 Importance or relevance of this topic to IoT & Cloud Secure Development 1

1.2 Objectives 2

2 Procedure 3

2.1 Raspberry Pi 3

2.2 AWS Cloud 5

3 Discussion 7

3.1 General Discussion 7

3.2 Future Work 7

References 8

Appendices 9

Appendix A 9

.

1 INTRODUCTION

Raspberry Pi Zero W	Used to run the software and to connect the different modules with its General-Purpose Input/Output (GPIO) pins.
HC-SR04	Ultrasonic sensor that emits and receives waves to detect distance
Button	Used to stop the program and initialise the lkm
Buzzer	Sound alarm that buzzes different ways depending on the distance detected
LED (Red, yellow, green)	A visual alarm that changes colour depending on distance
AWS Simple Storage Service (S3)	AWS storage for the distance logs created
AWS Simple Notification Service (SNS)	Sends emails alerting of any changes on the s3 bucket.

Table 1

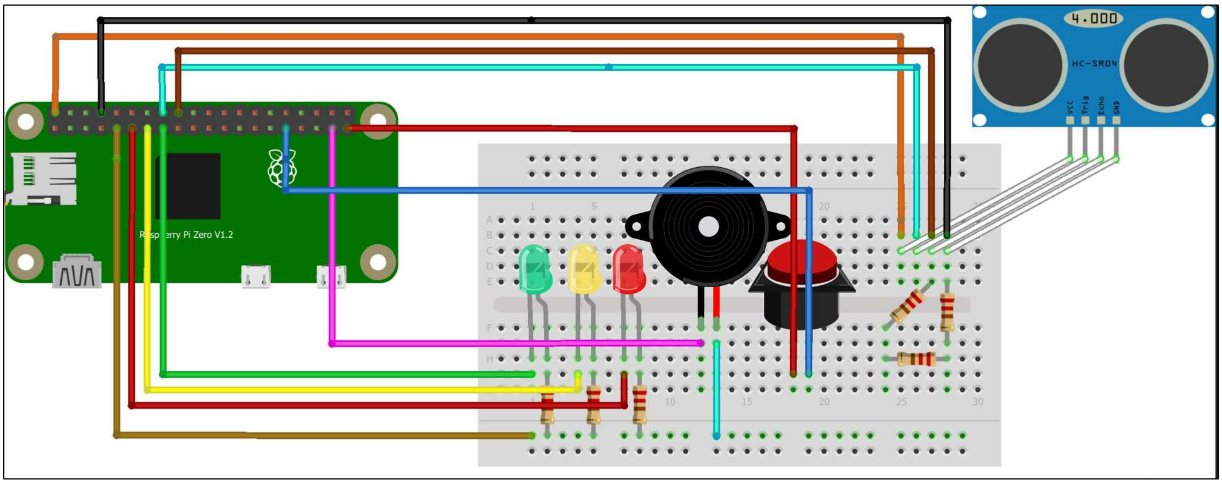

Figure 1: Visualisation of wired Raspberry Pi, breadboard and its components.

1.1 IMPORTANCE OR RELEVANCE OF THIS TOPIC TO IOT & CLOUD SECURE DEVELOPMENT

This project is relevant to IoT because it utilises Raspberry Pi Zero GPIO pins to connect several components and also features cross-compile module kernels. A Loadable Kernel Module (LKM), is dynamically loaded at the kernel level, sending a signal to the user-space python program when the button is pressed. Upon receiving this signal, the program logs alerts to a file and exits.

IoT devices generate vast amounts of data that must be stored securely for future analysis, monitoring, and troubleshooting. This script demonstrates a mechanism for incrementally uploading logs (from an IoT system) to cloud storage, ensuring that only new or modified files are uploaded, thus saving bandwidth. This is crucial for IoT systems, where continuous and efficient data management is vital. Cloud platforms like AWS S3 provide remote access, scalability, high availability, durable, and cost-effective storage solutions to handle the massive volumes of data generated by IoT devices.

1.2 Objectives

The objectives of the project are the following:

1. Measure with the sensor to determine the distance from an object.
2. Alert when appropriate (<20cm)
3. Save output in the log folder
4. Utilise loadable kernel module to call python script
5. Incremental upload log files to Amazon Web Services using S3 bucket
6. Receive a notification when the AWS S3 bucket has been modified for any reason.

2 PROCEDURE

Here is a flow structure diagram of the project. It starts by calling ids.cpp which executes nested while loop and if statement. When the user breaks the loop, it calls the kernel module. Finally, the kernel calls the Python script that will upload the log files onto Amazon Web Services (AWS). The code of all the files can be found in Appendix A.

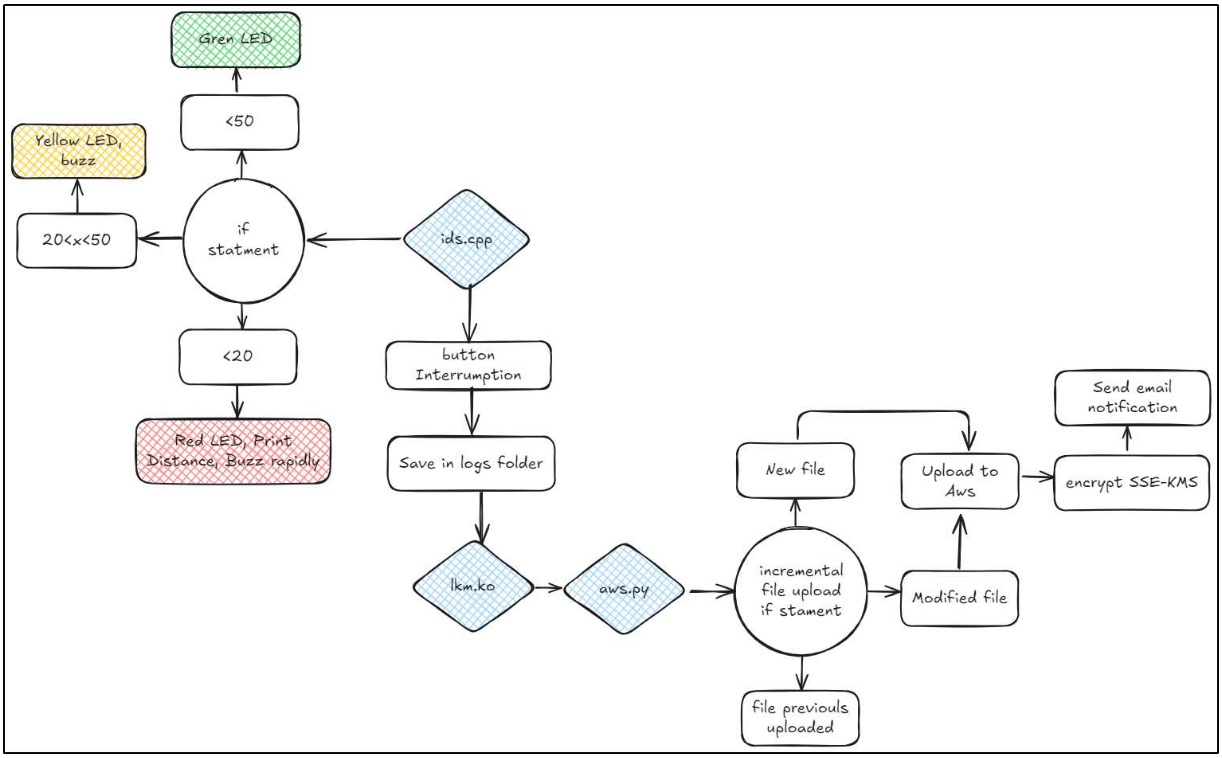

Figure 2: Flow structure diagram

2.1 RASPBERRY PI

This project utilises a Raspberry Pi and an ultrasonic sensor to measure distance and trigger different responses based on proximity. The primary components include an ultrasonic sensor, LEDs, and a buzzer, which interact to provide feedback on object distance.

The program begins by setting up GPIO pins for the ultrasonic sensor, LEDs, and buzzer using the `wiringPi` library. The `setupPins()` function configures these pins as either input or output as necessary. The ultrasonic sensor works by emitting waves via the trigger pin and measuring the waves duration received by the echo pin. This duration is then converted to a distance measurement in centimetres by the `getDistance()` function.

Based on the distance reading, the program controls three LEDs (green, yellow, and red) and a buzzer. If the object is far (>50 cm), the green LED is activated. For moderate proximity (20-50 cm), the yellow LED lights up, and a buzzer sounds with a 300 ms interval. For close proximity (<20 cm), the red LED lights up, and a faster buzzer interval (50 ms) is triggered. In this case, an alert is logged with the current distance and timestamp.

```
pi@raspberrypi:~/Desktop/ultrasonic_alerts $ sudo ./ids_2103939
!Intrusion detected!. Proximity distance of: 7.8 cm at 2024-12-28 21:41:02
!Intrusion detected!. Proximity distance of: 5.7 cm at 2024-12-28 21:41:03
!Intrusion detected!. Proximity distance of: 6.6 cm at 2024-12-28 21:41:03
!Intrusion detected!. Proximity distance of: 10.2 cm at 2024-12-28 21:41:04
!Intrusion detected!. Proximity distance of: 15.2 cm at 2024-12-28 21:41:04
Button pressed. Exiting...
Alerts saved to file: /home/pi/Desktop/ultrasonic_alerts/logs/ids_alert_log_2024-12-28 21:41:08.txt
Kernel module loaded successfully!
```

The alerts are stored in a vector and saved to a file with the saveAlertsToFile() function when the program is interrupted by pressing the push button or via Ctrl+C (a signal handler is used to ensure the alerts are saved if the program is terminated unexpectedly). The file is named with the current timestamp, and after saving, a loadable kernel module is executed to call the aws.py script. This script backs up the saved logs to AWS incrementally (only new or modified files are uploaded).

```
cmp408@localhost:~/btn
[cmp408@localhost btn]$ ls
btn1.c  btn3.c  btn5.c  btn7.c  lkm.c   lkm.mod    lkm.mod.o  Makefile       Module.symvers
btn2.c  btn4.c  btn6.c  btn.c   lkm.ko  lkm.mod.c  lkm.o      modules.order  qqq
[cmp408@localhost btn]$ scp lkm.ko pi@192.168.137.7:/home/pi/Desktop/ultrasonic_alerts/
pi@192.168.137.7's password:
lkm.ko                                            100% 4644   378.2KB/s   00:00
[cmp408@localhost btn]$
```

```
pi@raspberrypi:~/Desktop/ultrasonic_alerts $ dmesg | tail -n 20
[   34.928397] NET: Registered protocol family 31
[   34.928409] Bluetooth: HCI device and connection manager initialized
[   34.928441] Bluetooth: HCI socket layer initialized
[   34.928459] Bluetooth: L2CAP socket layer initialized
[   34.928501] Bluetooth: SCO socket layer initialized
[   34.986640] Bluetooth: HCI UART driver ver 2.3
[   34.986665] Bluetooth: HCI UART protocol H4 registered
[   34.986782] Bluetooth: HCI UART protocol Three-wire (H5) registered
[   34.987070] Bluetooth: HCI UART protocol Broadcom registered
[   36.435952] IPv6: ADDRCONF(NETDEV_CHANGE): wlan0: link becomes ready
[   37.271751] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[   37.271769] Bluetooth: BNEP filters: protocol multicast
[   37.271799] Bluetooth: BNEP socket layer initialized
[   37.887907] Bluetooth: RFCOMM TTY layer initialized
[   37.887948] Bluetooth: RFCOMM socket layer initialized
[   37.887999] Bluetooth: RFCOMM ver 1.11
[   46.128497] fuse: init (API version 7.31)
[  209.285463] lkm: loading out-of-tree module taints kernel.
[  209.296691] Python Script Caller: Initializing module.
[  209.304197] Python Script Caller: Script executed successfully.
pi@raspberrypi:~/Desktop/ultrasonic_alerts $
```

Figure 3 & 4: Kernel module make and loading

2.2 AWS CLOUD

The Python script is designed to incrementally upload files from a local folder to an Amazon S3 bucket, ensuring that only new or modified files are uploaded. It uses the `boto3` library for interacting with AWS S3 and handles the logic of checking whether a file already exists in the S3 bucket and if it needs updating. The script uses the `get_s3_file_metadata()` function to retrieve metadata for a specific file in the S3 bucket using the `head_object API`. It checks if the file exists and returns its size and last modification timestamp. If the file does not exist, it returns `None`. The following function `udpload_folder_to_s3_incremental()` uploads a folder's contents to S3. For each file, it constructs an S3 key (path), checks the file's metadata in S3, and compares the file's size and last modified time with the local file. If the file does not exist in S3 or if it has been modified locally, it uploads the file to S3. If the file is already up to date, it skips the upload. This ensures that only changed or new files are uploaded, optimising data transfer and storage usage.

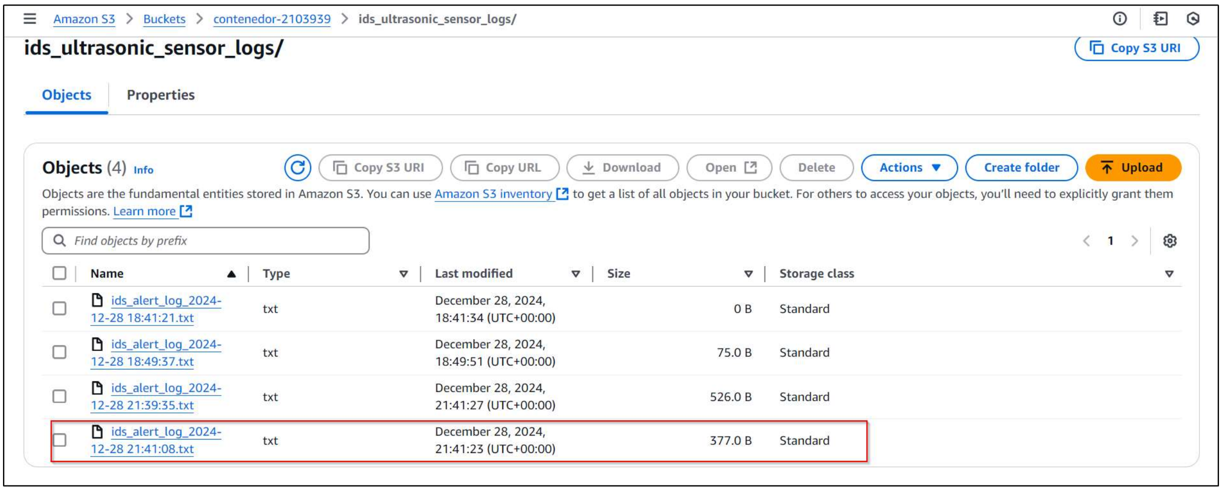

Figure 5: Files incrementally uploaded to the S3 bucket

The cloud is also configured to encrypt the files using SSE-KMS. SSE-KMS also provides an audit trail that shows when your KMS key was used and by whom. Finally, the cloud also sends an email notification to the programmer email, anytime anything is performed on the bucket, whether it is added files, modified files, or removed files. Everything is logged and notified.

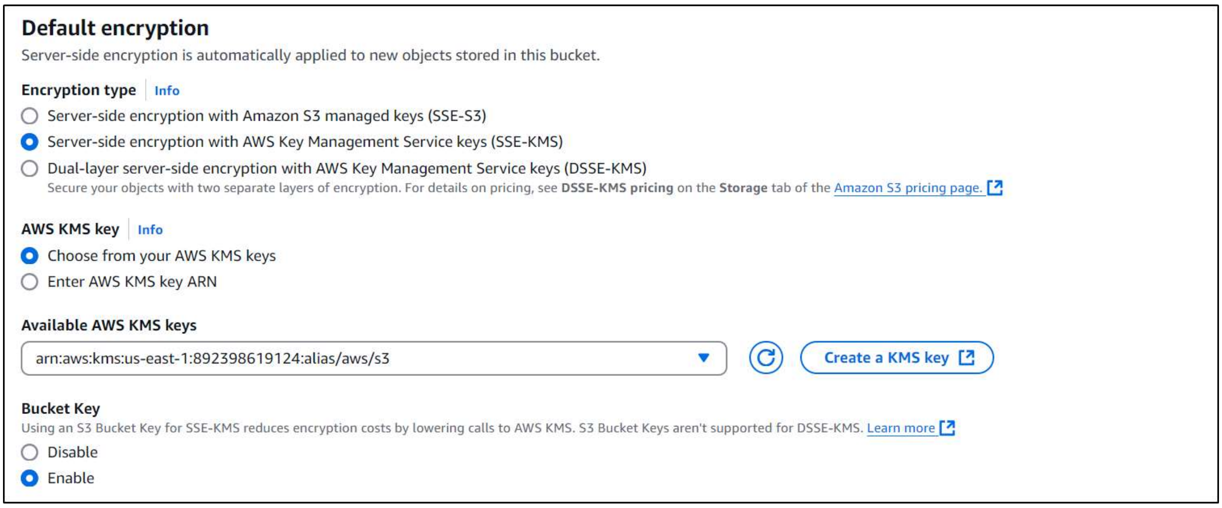

Figure 6: SSE-KMS encryption on AWS management console

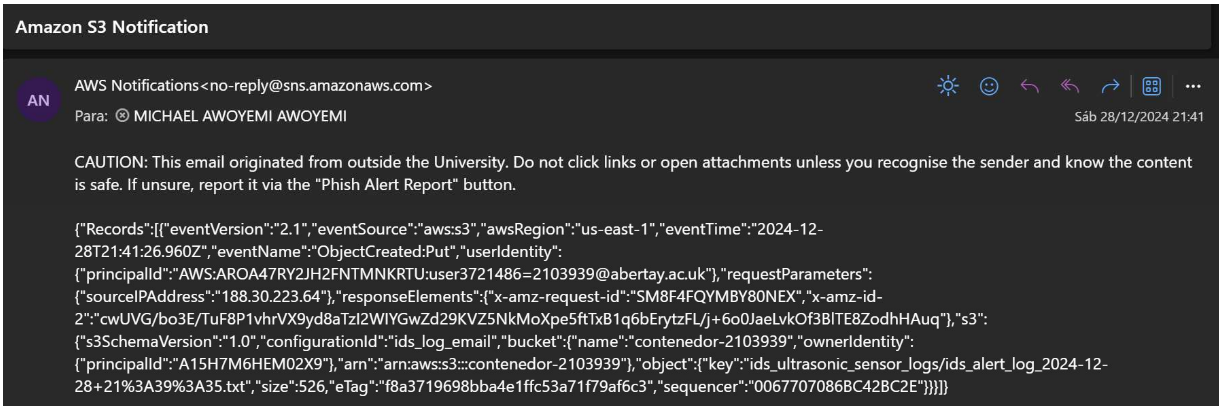

Figure 7: Email notification from AWS

3 DISCUSSION

3.1 GENERAL DISCUSSION

This IoT project is a fully functional perimeter intrusion detection system. The Raspberry Pi is connected to several components such as LEDs, a buzzer, a push button, and an ultrasonic sensor. The program is an if statement nested into a while loop until this is stopped by the button or via ctrl c. The sensor sends waves via its trig pin and receives them via its echo pin. Judging the distance the wave travels until it encounters an object and bounces back, it acts in different ways. If the distance is above 50cm only a green light is turned on, if the distance is between 50cm and 20 a yellow light with a buzz every 300 ms will be triggered. Lastly, if the distance is under 20cm a red light will be on and a rapid buzz every 50ms. Once the program is stopped either by the button or ctrl c, all the red-light output (below 20cm distance) will be logged into a file containing the distance and the timestamp.

The following action is to call the loadable kernel module which operates at a kernel level and its function is to run the last script called aws.py. This is the programme in charge of logging into the S3 bucket and backing up incrementally (only new or modified files). Upon a successful upload of the files to AWS, the files will be encrypted, and the AWS notification manager will send an email to notify of the new object added to the bucket as seen in Figure 7.

3.2 FUTURE WORK

If the programmer had been allocated more time, the following features would have been added to the project:

More sensors would have been incorporated into the project to detect any potentially dangerous distances from all four directions, making it more effective in securing and detecting object proximity.

A checksum verification could have been implemented before uploading, after uploading, and during the downloading of files to verify their authenticity. This feature would prevent accidental or deliberate tampering with the log files. Additionally, file permissions could have been set to read-only or protected with a password upon opening.

Furthermore, additional services on AWS could have been explored to enhance security best practices. This would have included a thorough examination of AWS Identity and Access Management (IAM), with an emphasis on its capabilities for defining user roles and implementing the principle of least privilege. This principle ensures users have only the access necessary to perform their tasks. Finally, the creation of custom policies, the use of managed policies, and the implementation of permissions boundaries would have been addressed.

References

GeeksforGeeks. (n.d.). *Arduino Ultrasonic Sensor*. Available at: https://www.geeksforgeeks.org/arduino-ultrasonic-sensor/ [Accessed 13 Dec. 2024].

Mouser Electronics. (2018). *How to Use a Buzzer with Raspberry Pi*. [video] Available at: https://www.youtube.com/watch?v=0jedQM7qS90 [Accessed 13 Dec. 2024].

Circuit Basics. (n.d.). *How to Use Buzzers with Raspberry Pi*. Available at: https://www.circuitbasics.com/how-to-use-buzzers-with-raspberry-pi/ [Accessed 13 Dec. 2024].

Raspberry Pi Foundation. (n.d.). *Button and Switch Control with Scratch and Raspberry Pi*. Available at: https://projects.raspberrypi.org/en/projects/button-switch-scratch-pi/1 [Accessed 20 Dec. 2024].

Amazon S3 User Guide. (n.d.). *Welcome to Amazon S3*. Available at: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html [Accessed 23 Dec. 2024].

Amazon S3 User Guide. (n.d.). *Event Notifications*. Available at: https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventNotifications.html [Accessed 23 Dec. 2024].

Stack Overflow. (2021). *How to Cross-Compile ARM Version of LKM?* Available at: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjYy6Xn0dKKAxVFVkEAHeyGHJYQFnoECBcQAQ&url=https%3A%2F%2Fstackoverflow.com%2Fquestions%2F70114335%2Fhow-to-cross-compile-arm-version-of-lkm&usg=AOvVaw1Geye1GQFBskJJPLPrCDYd&opi=89978449 [Accessed 27 Dec. 2024].

APPENDICES

APPENDIX A

ids.cpp

```
#include <wiringPi.h>
#include <iostream>
#include <iomanip>
#include <cmath>
#include <ctime>
#include <fstream>
#include <vector>
#include <csignal>
#include <unistd.h>
#include <cstdlib>
using namespace std;

#define TRIG_PIN 23
#define ECHO_PIN 24
#define GREEN_LED_PIN 22
#define YELLOW_LED_PIN 27
#define RED_LED_PIN 17
#define BUZZER_PIN 26
#define BUTTON_PIN 6

vector<string> alertLogs;

// Function to initialize GPIO pins
void setupPins() {
    pinMode(TRIG_PIN, OUTPUT);
    pinMode(ECHO_PIN, INPUT);
    pinMode(GREEN_LED_PIN, OUTPUT);
    pinMode(YELLOW_LED_PIN, OUTPUT);
    pinMode(RED_LED_PIN, OUTPUT);
    pinMode(BUZZER_PIN, OUTPUT);
    pinMode(BUTTON_PIN, INPUT);
    pullUpDnControl(BUTTON_PIN, PUD_UP);
}

// Function to read distance from ultrasonic sensor
double getDistance() {
    // Send pulse to trigger
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    unsigned long pulse_start = micros(); // Start time
    while (digitalRead(ECHO_PIN) == LOW) {
        pulse_start = micros(); // Reset start time when echo is low
    }

    unsigned long pulse_end = micros(); // End time when echo goes high
    while (digitalRead(ECHO_PIN) == HIGH) {
        pulse_end = micros(); // End time when echo goes high
    }
```

```
    // Calculate duration in microseconds
    unsigned long pulse_duration = pulse_end - pulse_start;

    // Calculate distance (in cm)
    double distance = (pulse_duration * 0.0343) / 2;
    return distance;
}

// Function to handle buzzer sound
void buzz(int interval) {
    digitalWrite(BUZZER_PIN, HIGH);
    delay(interval);
    digitalWrite(BUZZER_PIN, LOW);
    delay(interval);
}

// Function to get current date and time as a string
string getCurrentDateTime() {
    time_t now = time(0);
    tm *ltm = localtime(&now);

    stringstream ss;
    ss << 1900 + ltm->tm_year << "-"
       << setw(2) << setfill('0') << 1 + ltm->tm_mon << "-"
       << setw(2) << setfill('0') << ltm->tm_mday << " "
       << setw(2) << setfill('0') << ltm->tm_hour << ":"
       << setw(2) << setfill('0') << ltm->tm_min << ":"
       << setw(2) << setfill('0') << ltm->tm_sec;

    return ss.str();
}

void saveAlertsToFile() {
    const string folderName = "/home/pi/Desktop/ultrasonic_alerts/logs";

    string fileName = folderName + "/ids_alert_log_" + getCurrentDateTime() + ".txt";
    ofstream outFile(fileName);

    // Check if file was created successfully
    if (outFile.is_open()) {
        for (const string &alert : alertLogs) {
            outFile << alert << endl;
        }
        outFile.close();
        cout << "Alerts saved to file: " << fileName << endl;

        // Call the kernel module after saving the file
        int ret = system("sudo insmod /home/pi/Desktop/ultrasonic_alerts/lkm.ko");
        if (ret != 0) {
            cerr << "Error: Unable to load the kernel module!" << endl;
        } else {
            cout << "Kernel module loaded successfully!" << endl;
        }
    } else {
        cerr << "Error: Unable to create file in folder: " << folderName << endl;
    }
}

// Function to check if the button is pressed
bool isButtonPressed() {
    return digitalRead(BUTTON_PIN) == LOW;
}
```

```
int main() {

    // Initialize wiringPi and setup GPIO pins
    if (wiringPiSetupGpio() == -1) {
        cerr << "Failed to initialize wiringPi!" << endl;
        return 1;
    }

    setupPins();

    while (true) {
        // Check if button is pressed, if so, exit the program
        if (isButtonPressed()) {
            cout << "Button pressed. Exiting..." << endl;
            saveAlertsToFile();
            break;
        }

        double distance = getDistance(); // Get the distance from the ultrasonic
sensor
        distance = round(distance * 10) / 10.0;

        if (distance > 50) { // Green LED, no buzzer
            digitalWrite(GREEN_LED_PIN, HIGH);
            digitalWrite(YELLOW_LED_PIN, LOW);
            digitalWrite(RED_LED_PIN, LOW);
            digitalWrite(BUZZER_PIN, LOW); // Ensure buzzer is off
        }
        else if (distance > 20) { // Yellow LED, moderate buzzer
            digitalWrite(GREEN_LED_PIN, LOW);
            digitalWrite(YELLOW_LED_PIN, HIGH);
            digitalWrite(RED_LED_PIN, LOW);
            buzz(300); // Moderate buzzer interval (300ms)
        }
        else { // Red LED, faster buzzer
            digitalWrite(GREEN_LED_PIN, LOW);
            digitalWrite(YELLOW_LED_PIN, LOW);
            digitalWrite(RED_LED_PIN, HIGH);
            buzz(50); // Faster buzzer interval (50ms)

            // Log red LED alert
            stringstream alertStream;
            alertStream << "!Intrusion detected!. Proximity distance of: " << fixed
<< setprecision(1) << distance
                        << " cm at " << getCurrentDateTime();
            string alert = alertStream.str();
            alertLogs.push_back(alert);

            // Print to console
            cout << alert << endl;
        }

        delay(500);
    }

    return 0;
}
```

Lkm.c

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/kmod.h>

static int __init call_python_script_init(void) {
    char *argv[] = { "/usr/bin/python3", "/home/pi/Desktop/ultrasonic_alerts/aws.py",
NULL };
    char *envp[] = { "HOME=/", "PATH=/sbin:/bin:/usr/sbin:/usr/bin", NULL };
    int ret;

    printk(KERN_INFO "Python Script Caller: Initializing module.\n");

    // Call the Python script
    ret = call_usermodehelper(argv[0], argv, envp, UMH_WAIT_EXEC);
    if (!ret) {
        printk(KERN_INFO "Python Script Caller: Script executed successfully.\n");
    }

    return 0;
}

static void __exit call_python_script_exit(void) {
    printk(KERN_INFO "Python Script Caller: Exiting module.\n");
}

module_init(call_python_script_init);
module_exit(call_python_script_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("2103939-Michael");
MODULE_DESCRIPTION("Kernel module calling aws.py.");
```

Makefile

```
ifeq ($(KERNELRELEASE),)
KDIR := /home/cmp408/rpisrc/linux
PWD := $(shell pwd)
default:
	$(MAKE) -C $(KDIR) M=$(PWD) modules

clean:
	$(MAKE) -C $(KDIR) M=$(PWD) clean
else
obj-m := lkm.o
endif
[cmp408@localhost btn]$
```

Aws.py

```
import boto3
import os
from botocore.exceptions import ClientError

def get_s3_file_metadata(bucket_name, s3_key, aws_access_key_id,
aws_secret_access_key, aws_session_token):

    s3_client = boto3.client(
        's3',
        aws_access_key_id=aws_access_key_id,
        aws_secret_access_key=aws_secret_access_key,
        aws_session_token=aws_session_token
    )
    try:
        response = s3_client.head_object(Bucket=bucket_name, Key=s3_key)
        return {
            "size": response['ContentLength'],
            "last_modified": response['LastModified']
        }
    except ClientError as e:
        if e.response['Error']['Code'] == '404':
            return None
        else:
            raise e

def upload_folder_to_s3_incremental(folder_path, bucket_name, s3_folder_name,
aws_access_key_id, aws_secret_access_key, aws_session_token):

    s3_client = boto3.client(
        's3',
        aws_access_key_id=aws_access_key_id,
        aws_secret_access_key=aws_secret_access_key,
        aws_session_token=aws_session_token
    )

    # Walk through the directory and upload each file
    for root, dirs, files in os.walk(folder_path):
        for file_name in files:
            # Construct the full local file path
            local_file_path = os.path.join(root, file_name)

            # Determine the S3 key (path) for the file
            relative_path = os.path.relpath(local_file_path, folder_path)
            s3_key = os.path.join(s3_folder_name, relative_path) if s3_folder_name
else relative_path

            # Replace backslashes with forward slashes for S3 (Windows compatibility)
            s3_key = s3_key.replace("\\", "/")

            # Get metadata of the file in S3
            s3_metadata = get_s3_file_metadata(bucket_name, s3_key,
aws_access_key_id, aws_secret_access_key, aws_session_token)

            # Check if file needs to be uploaded
            upload_required = False
            if s3_metadata is None:
                print(f"File {s3_key} does not exist in S3. Uploading...")
                upload_required = True
            else:
                local_file_size = os.path.getsize(local_file_path)
                local_last_modified = os.path.getmtime(local_file_path)
```

```
                if (local_file_size != s3_metadata['size'] or
                    local_last_modified > s3_metadata['last_modified'].timestamp()):
                    print(f"File {s3_key} has been modified. Uploading...")
                    upload_required = True

            if upload_required:
                try:
                    s3_client.upload_file(local_file_path, bucket_name, s3_key)
                    print(f"Uploaded {local_file_path} to
s3://{bucket_name}/{s3_key}.")
                except Exception as e:
                    print(f"Failed to upload {local_file_path} to
s3://{bucket_name}/{s3_key}: {e}")
            else:
                print(f"File {s3_key} is up-to-date. Skipping...")

    print("Incremental upload completed.")

if __name__ == "__main__":

    aws_access_key_id = "ASIA47RY2JH2OUCOGL36"
    aws_secret_access_key = "B9pcymY2jnmuz7yXsIBgSnS8fOr1Tnj0YTpJFQhp"
    aws_session_token = ""


    folder_path = "/home/pi/Desktop/ultrasonic_alerts/logs"
    bucket_name = "contenedor-2103939"
    s3_folder_name = "ids_ultrasonic_sensor_logs"

    upload_folder_to_s3_incremental(folder_path, bucket_name, s3_folder_name,
aws_access_key_id, aws_secret_access_key, aws_session_token)
```