Bombs Light Clock: Extended with Internet-Based Alarm Detection

Description

The Bombs Light Clock relied on a microphone-based sound detection system to identify alarm signals. However, this approach had its limitations, including sensitivity to background noise and environmental conditions, which led to false positives or missed detections. To enhance its reliability, the system was extended to leverage internet-based real-time alarm data provided by the Israeli Home Front Command, ensuring accurate and timely notifications.

This extension transforms the Bombs Light Clock from a locally dependent device into a more robust and connected system, making it an essential tool for personal safety in challenging environments.

Motivation

Problem with the Microphone-Based System

The original design used a microphone and FFT (Fast Fourier Transform) analysis to detect alarm sounds. While this approach was innovative, it faced several challenges:

- Environmental Noise: Background noise or interference could trigger false positives.
- Volume Dependency: The system struggled to detect alarms in noisy environments or when the alarm was faint.
- Pattern Matching Issues: Minor variations in the alarm frequency pattern sometimes caused the system to miss detections.

Solution with Internet-Based Alarm Data

To overcome these limitations, the project was extended to use real-time alarm data from the Israeli Home Front Command API. By fetching alerts directly from a trusted source, the system:

- Ensures accurate alarm detection, unaffected by environmental noise.
- Eliminates the dependency on the microphone and FFT analysis.
- Provides real-time response to national alerts, synchronized with official systems.

Why This is an Important Product

1. Addressing Real-World Needs:
    - In a country with frequent alarms, timely and accurate notifications are critical for public safety.
    - Individuals with hearing impairments or those in noisy environments often struggle to detect alarms. This device bridges that gap with a reliable visual alert system.
2. Enhanced Reliability:
    - The extension to use internet-based alerts ensures 100% accuracy by synchronizing with national alert systems.
    - By removing the dependency on sound detection, the system becomes robust and immune to external noise.
    - Encourages safety and awareness while serving a functional purpose.

The Circuit

- NeoPixel LED Strip (12 LEDs): Connected to pin 15, used for both the clock display and the explosion effect.
- ESP32 Microcontroller: Handles Wi-Fi connectivity, real-time API requests, and LED control.
- (Original Setup) Microphone: Previously connected to pin 26 for sound detection, now replaced with internet-based detection.

Features

Light Clock

- Displays the current hour with red LEDs.
- Displays the minutes in 5-minute increments with green LEDs.
- Updates every minute and resets the hour after 60 minutes.

Internet-Based Alarm Detection

- Fetches real-time alerts from the Home Front Command API.
- Detects active alarms and triggers the explosion effect.
- Resumes normal clock functionality when no alerts are active.

Explosion Effect

- Simulates an “explosion” by cycling random colors on all LEDs for 40 seconds.
- Provides a clear and immediate visual indication of an alarm.

Development Process

Original Alarm Detection with FFT

1. The microphone (connected to pin 26) captured audio signals, which were analyzed using the arduinoFFT library.
2. The FFT transformed the audio signal into its frequency-domain representation, allowing specific frequencies to be identified.
3. A predefined array of target frequencies (alarmPattern) and thresholds were used to detect the unique pattern of an alarm sound.
4. If the pattern was matched within a time window, the system triggered the “explosion effect.”

Extended Internet-Based Alarm Detection

1. Real-Time Alerts:
   - The system connects to Wi-Fi and fetches data from the Home Front Command API
   - The API provides structured JSON data with live alerts.
2. Improved Detection:
   - The system processes the API response to check for active alerts.
   - Upon detecting an alert, the explosion effect is triggered, and the user is visually notified.
3. Fallback to Clock Mode:
   - If no alerts are detected, the system resumes its normal clock functionality.

Video link:

Attaching three videos:

Light Clock with internet connection detection // this is the new version video !!!!

Light Clock with microphone detection

Use Of Light Clock Video

Created By:

Asaf Ramati 209252154

Rony Rabinovitz 314834425

Code

```
#include <Adafruit_NeoPixel.h>
#include <WiFi.h>
#include <HTTPClient.h>

// LED strip settings
#define PIN 15              // Pin connected to the LED strip
#define NUMPIXELS 12        // Number of LEDs in the strip
Adafruit_NeoPixel strip(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

// Wi-Fi settings
const char* ssid = "YaelMoses";       // Replace with your Wi-Fi SSID
const char* password = "10203040"; // Replace with your Wi-Fi password

const char* apiUrl = "https://www.oref.org.il/WarningMessages/alert/alerts.json";
bool isAlarmPlaying = false;

// Clock settings
int currentHourLed = 0; // The LED representing the current hour
int minutesCounter = 0; // Counter for minutes (0 to 59)

void setup() {
  Serial.begin(115200);
  strip.begin();
  strip.show(); // Ensure all LEDs are off initially
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    Serial.println("\nConnecting");
    while(WiFi.status() != WL_CONNECTED){
        Serial.print(".");
        delay(100);
    }
}

void loop() {
```

```
  if (isAlarmPlaying) {
    // If the alarm is playing, run the explosion effect
    triggerExplosionEffect();
    isAlarmPlaying = false; // After the effect, switch back to clock mode
  } else {
    // Check for the alarm
    checkForAlarm();

    // If no alarm, continue running the clock
    if (!isAlarmPlaying) {
      updateClock();
    }
  }
  delay(50); // Short delay to prevent overloading the processor
}

void updateClock() {
  strip.clear(); // Clear all LEDs

  // Light up the LED representing the current hour
  strip.setPixelColor(currentHourLed, strip.Color(255, 0, 0)); // Red light for the hour

  // Calculate how many green LEDs to light up for the minutes
  int greenLeds = minutesCounter / 5; // Light up one green LED for every 5 minutes

  // Light up green LEDs for the minutes
  for (int i = 1; i <= greenLeds; i++) {
    int greenLedIndex = (currentHourLed + i + NUMPIXELS) % NUMPIXELS; // Calculate position
from the hour LED
    strip.setPixelColor(greenLedIndex, strip.Color(0, 255, 0)); // Green light
  }

  strip.show(); // Display the changes on the LEDs

  // Update the time
  minutesCounter++;

  if (minutesCounter >= 60) { // If an hour has passed
    minutesCounter = 0; // Reset the minute counter
    currentHourLed = (currentHourLed + 1) % NUMPIXELS; // Move to the next LED for the next
hour
  }
}

void checkForAlarm() {

   if (WiFi.status() == WL_CONNECTED){
      HTTPClient http;
      http.begin(apiUrl);

      int httpCode = http.GET();
      if (httpCode  > 0 ) {
          // HTTP request successful
        if (httpCode == HTTP_CODE_OK) {
          String payload = http.getString(); // Get the response payload
          Serial.println("Response:");
          Serial.println(payload);

          // Check if there are active alerts
          if (payload.indexOf("alerts") > -1) { // Basic check for alerts
```

```
            Serial.println("ALERT DETECTED! Triggering explosion effect...");
            isAlarmPlaying = true;
          }else {
            Serial.println("No alerts detected.");
          }
        }else {
         // HTTP request failed
         Serial.printf("HTTP GET failed, error: %s\n", http.errorToString(httpCode).c_str());
        }
      }
    }else {
       Serial.println("Wi-Fi disconnected!");
    }
}

void triggerExplosionEffect() {
  Serial.println("Alarm Detected! Triggering explosion effect...");

  unsigned long startTime = millis(); // Record the time when the effect starts

  while (millis() - startTime < 40000) { // 40 seconds = 40000 milliseconds
    for (int j = 0; j < NUMPIXELS; j++) {
      strip.setPixelColor(j, strip.Color(random(255), random(255), random(255))); // Random colors
    }
    strip.show();
    delay(100);
  }

  strip.clear(); // Turn off all LEDs after 40 seconds
  strip.show();
}
```

alerts.json

```
//This file was created for test purposes
// for open a local server change
// const char* apiUrl = “http://<your-computer-ip>:8000/alerts.json";
// replace your-computer-ip with the actual ip.

{
  "data": {
    "alerts": [
      {
        "region": "Test Region",
        "timestamp": "2025-01-01T12:00:00Z"
      }
    ]
  }
}
```