Project HawkEye

29.10.2025

Indian Institute of Technology, Kharagpur
Department of Physics
M.Sc. Physics

Innovation Lab

Presented by

Tanmoy Bhakat- 24PH40053

Tushar Majhi-24PH40055

Himanshi-24PH40021

Rohit Bauri-24PH40037

Sayantan Samanta-24PH40041

Overview

This project, **"Laser Alert System using ESP32,"** is a simple yet effective security system designed to detect any interruption in a laser beam. When the laser light on the LDR is blocked, the system immediately activates a **buzzer alarm** and sends a **real-time alert message** to the user's phone via **Telegram Bot**.

It includes:

- **Hardware setup** with ESP32, LDR sensor, laser module, and buzzer.
- **Software implementation** using Arduino IDE and Telegram Bot API.
- **Live alert mechanism** for intrusion detection.
- **Practical demonstration** and results included in the PPT.

This repository contains:

- 🧾 **Source Code** for ESP32.
- 🧩 **Circuit Diagram** and component details.
- 🎥 **Project Demonstration Slides (PPT)**.

Code

```
/*
 * PROJECT: HAWKEYE LASER DEFENDER (Telegram Alert System)
 * PLATFORM: ESP32 Dev Module
 *
 * FUNCTION: Detects when a laser beam (LDR) is broken, triggers local alarms,
 * and sends an instant notification via Telegram Bot (HTTPS).
 *
 * NOTE: This project requires manual configuration of sensitive credentials
 * and may require SSL certificate setup depending on the board used.
 *
 */

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

// ---------- USER CONFIGURATION: CREDENTIALS (MUST BE REPLACED) ----------
// To set up, follow the instructions in the README file.

// 1. WiFi Network (Must be replaced by the user)
const char* ssid = "YOUR_WIFI_SSID";
const char* password = "YOUR_WIFI_PASSWORD";
```

```
// 2. Telegram Bot Credentials
String botToken = "YOUR_BOT_TOKEN"; // Get this from BotFather
// Add multiple chat IDs to the array to send alerts to multiple users
String chatIDs[] = { "YOUR_CHAT_ID_1", "YOUR_CHAT_ID_2" };

// ---------- PIN & DEVICE SETUP ----------
const int LDR_PIN     = 34;  // LDR Analog Input (GPIO34 is input-only)
const int BUZZER_PIN  = 25;  // Active Buzzer (D25)
const int RELAY_PIN   = 26;  // 5V Relay (D26)

// ---------- ALERT SETTINGS ----------
// Adjust this value after initial testing. LDR value is 0-4095 on ESP32.
const int LDR_THRESHOLD_DEFAULT = 400;
const unsigned long ALERT_COOLDOWN = 30000; // 30 seconds cooldown between
messages
const unsigned long LOOP_DELAY = 200; // ms

int ldrThreshold = LDR_THRESHOLD_DEFAULT;
unsigned long lastAlertMillis = 0;

void setup() {
  Serial.begin(115200);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);

  Serial.println("\n=== HAWKEYE Laser Alert System ===");

  WiFi.begin(ssid, password);
```

```
  Serial.print("Connecting to WiFi");
  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED) {
    delay(300);
    Serial.print(".");
    if (millis() - start > 20000) {
      Serial.println("\nWiFi connect timeout.");
      start = millis();
    }
  }
  Serial.println("\nWiFi connected. IP: " + WiFi.localIP().toString());
  Serial.println("System armed. LDR Threshold: " + String(ldrThreshold));
}

void loop() {
  int ldrValue = analogRead(LDR_PIN); // 0..4095 on ESP32 ADC
  Serial.print("LDR: "); Serial.println(ldrValue);

  unsigned long now = millis();
  bool beamInterrupted = (ldrValue < ldrThreshold); // True if light drops below threshold

  if (beamInterrupted && (now - lastAlertMillis > ALERT_COOLDOWN)) {
    Serial.println(">>> BEAM INTERRUPTED! Sending alert...");
    // --- Local Alert Activation ---
    digitalWrite(BUZZER_PIN, HIGH);
    digitalWrite(RELAY_PIN, HIGH);
    delay(5000); // Buzzer/Relay active for 5 seconds
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(RELAY_PIN, LOW);
```

```
    // --- Send Telegram Message ---
    String msg = "🚨 HAWKEYE Alert! Laser tripwire broken. LDR Value=" + String(ldrValue);
    sendToAll(msg);

    lastAlertMillis = now;
  }

  delay(LOOP_DELAY);
}

void sendToAll(const String &text) {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi not connected - cannot send message");
    return;
  }

  // Simple URL-encoding for spaces and other special characters
  String t = text;
  t.replace(" ", "%20");
  t.replace("#", "%23");

  for (unsigned int i = 0; i < sizeof(chatIDs)/sizeof(chatIDs[0]); ++i) {
    String url = "https://api.telegram.org/bot" + botToken + "/sendMessage?chat_id=" +
chatIDs[i] + "&text=" + t;
    Serial.println("Sending to: " + chatIDs[i]);
    HTTPClient http;
    http.begin(url); // Connects securely to the Telegram API endpoint
    int code = http.GET();
    Serial.printf("HTTP code: %d\n", code);
```

```
    http.end();
    delay(300); // Pause between recipients
  }
}
```

📜 Guide: How to Get the Necessary Credentials

1. 🔑 Obtain Your Telegram Bot Token

This token is the password that allows your ESP32 to send messages *as* your bot..

- **Action:** Open Telegram and search for the official account: **`@BotFather`**.
- **Command:** Send the command `/token` (if you already created the bot) or `/newbot` (if starting fresh).
- **Result:** BotFather will provide a token that looks like: `1234567890:ABC-DEF1234567890_GHIJ-KLMNO`.
- **Usage in Code:** Replace the placeholder `YOUR_BOT_TOKEN` with this entire string.

2. 💬 Find Your Telegram Chat ID

The Chat ID tells the bot *which user* to send the message to. Since your bot is public, we'll use a simple method:

- **Action A (Start Chat):** Search for your bot, **`@Your_bot`**, and send it any message (e.g., "Start").

  Action B (Get Update URL): Open your web browser and navigate to the following URL, replacing the placeholder with your actual bot token: https://api.telegram.org/bot<YOUR_BOT_TOKEN>/getUpdates

- **Result:** The browser will display a large block of code (JSON). Look inside this code for the first instance of `"chat":{"id":...`
  - The long number next to `"id"` (e.g., `1175576925`) is your **Chat ID**.
- **Usage in Code:** Replace the placeholder `YOUR_CHAT_ID_1` with this number. If you want to send alerts to multiple users, repeat this process for each user and add their IDs to the `chatIDs[]` array.

3. 🌐 Finalizing WiFi Credentials

- **Action:** Replace `YOUR_WIFI_SSID` and `YOUR_WIFI_PASSWORD` with the exact name and password of the Wi-Fi network you are using (e.g., your mobile hotspot).