StepSnitch

A Motion-Triggered Security Device with Cloud Connectivity

By Group #54:
Carson Gusaas | gusaa008@umn.edu
Jonas McClung | mcclu360@umn.edu

EE1301 Final Project Report
Professor: Kia Bazargan
May 02, 2025

Project Description

No matter who you are or where you live, security is an important priority, and it isn't always easy to guarantee. Without security measures, you often have no way of knowing whether people or animals are present in your home, business, or any other important location. To solve this problem, we created a cloud-connected motion-detecting security device powered by a Particle Photon 2 microcontroller. This device's features include live web monitoring, email alerts, an alarm, and motion status LEDs.

The most important component of this security device is the HC-SR501 PIR Motion Sensor, which is designed to detect changes in infrared radiation caused by the movement of living organisms. This sensor detects movement at a range of up to 21 feet away, within a 120-degree angle from the sensor, allowing quite a large area to be secured. The device also has several actuating components, the first of which is a small Piezoelectric Speaker. This speaker continuously sounds an alarm during motion detections. Similarly, three addressable iLEDs are used as motion status indicators, glowing green when no movement is detected, and switching to red for the duration of motion detections. Finally, two pairs of green and red diffused LEDs were used to display the current ON/OFF status of the device and its alarm.

For this device to serve its purpose as a security measure, it required some sort of internet connectivity, a web interface for live monitoring, and live alerts. To accomplish this, the Particle Photon 2's wifi connectivity is used to constantly send data to the cloud as it's recorded by the sensor. From there, the interactive website can access and display motion detection data, including time stamps, the current motion status, and a timeline of motion detections over the past hour. This cloud connectivity and webpage also allow the device and alarm to be toggled on or off through a website control panel. Finally, the device uses the third-party service IFTTT to send live motion detection email alerts.

The previously discussed device functionality combines for an efficient, user-friendly security solution with numerous applications. One use case example for this device is home security, whether this means securing your front porch from unauthorized people or animals, or making sure nobody enters your room full of valuable items. Additionally, the device could be used to monitor motion activity at a place of business during closing hours. When an unauthorized person triggers the motion detector, sees the flashing LEDs and alarm, they'll know their presence is no secret.

Coding & Wiring

To understand the code, it may be beneficial to first view the wiring diagram, see *Appendix I*. The code starts with a series of if/else statements, serving as the motion detection logic. The idea behind the logic was for each detection to be continuous, lasting any amount of time, up until the sensor's signal finishes. This was only possible because of the motion sensor's retriggering mode. This would allow the alarm, LED colors, and motion status displayed on the website to all work in sync, continuously for the duration of each motion detection. This was accomplished by using state variables to keep track of when a detection starts and stops, and only triggering LED changes when the motion has just started or ended.

```
//Motion detection, motion logging, and LED logic:

  //Device toggled ON/OFF through website
  if(deviceModeIs == DEVICE_ON) {
    digitalWrite(device_ON_PIN, HIGH);
    digitalWrite(device_OFF_PIN, LOW);

    //IF: motion was just detected:
    if(motionReading == HIGH && previousSensorState == false) {
      currentMotion = true;
      previousSensorState = true; //to prevent retrigger
      motionLogged = false; //to prevent extra logs
      RedLEDs(); //LEDS --> red

      //if statement to make sure the motion is only logged once per detection
      if(motionLogged == false) {
        logDetection();
        motionLogged = true;
      }

    //ELSE IF: motion just stopped being detected:
    } else if(motionReading == LOW && previousSensorState == true) {
      previousSensorState = false;
      currentMotion = false; //no motion
      GreenLEDs(); //LEDS --> green
    }
```

The first if statement allows for the device to be toggled on and off.

The currentMotion variable acts as a live motion status indicator, only being "true" while motion is being detected.

The logic shown in the image allows for motion to only be logged once each time the sensor is triggered. This was key for preventing hundreds of website detection log entries with each motion detection.

The alarm logic follows the same rules, only sounding when the currentMotion variable is true, causing a continuous output while motion is detected. Additionally, the alarm code had to be non-blocking. This was accomplished by using time-tracking variables to create a non-blocking delay.

The first if statement allows the alarm to be toggled on and off.

Subtracting time state variables and checking if greater than 250ms has the same effect as delay(250).

```
//Alarm logic (non-blocking, continious over motion duration)
  if(alarmModeIs == ALARM_ON) {
    digitalWrite(alarm_ON_PIN, HIGH); //alarm green LED ON
    digitalWrite(alarm_OFF_PIN, LOW); //alarm red LED OFF

    if(currentMotion) {
      timeNow = millis();

      //IF: 250ms have elapsed since last alarm tone
      if(timeNow-timeLastBeep >= 250) {
        timeLastBeep = timeNow;
        noTone(speakerPIN);

        //to make sure alarm starts on high tone
        if(beepCount == 0) {
          highTone = true;
        }
```

The speaker alternates between a high-pitched tone and a lower tone every 250ms using if/else statements and flipping a bool as shown.

```
    if(highTone) {
      tone(speakerPIN, 1800, 0); //1800hz (high pitched) sound
      beepCount++;
      highTone = false;
    }else if(!highTone) {
      tone(speakerPIN, 1200, 0); //1200hz (lower pitched) sound
      beepCount++;
      highTone = true;
    }
  }
```

Following the previously shown code, else statements are used to turn the alarm off when motion ends, and turn the iLEDs off when the device is toggled off.

Cloud functions were used to control the device from the website. When the device or speaker toggle button is pressed on the website, a cloud function calls a function within the code, taking a string as an "instruction" parameter. Based on this string, the function flips a variable, toggling the speaker or device on/off.

```
//Called by cloud function (from website) for toggling device on/off
int deviceMode(String inputString) {
  if(inputString == "Device OFF") {
    deviceModeIs = DEVICE_OFF;
    return 0;
  } else if(inputString == "Device ON") {
    deviceModeIs = DEVICE_ON;
    GreenLEDs();
    return 1;
  } else {
    return -1; //something went wrong
  }
}

//Called by cloud function (from website) for toggling speaker on/off
int alarmMode(String inputString){
  if(inputString == "Alarm OFF") {
    alarmModeIs = ALARM_OFF;
    return 0;
  } else if(inputString == "Alarm ON") {
    alarmModeIs = ALARM_ON;
    return 1;
  } else {
    return -1; //something went wrong
  }
}
```

The if-else statements allow the device or speaker to be toggled depending on the string that is sent.

The functions return a different integer based on the if-statement used, which is sent back to the website. A negative integer indicates an error to the user.

Cloud variables are used to transmit detection data to the website. These variables include a time stamp (string) for the most recent detection, a bool stating the current motion status, and 12 integers corresponding to a timeline array with five-minute intervals over the past hour.

```
void logDetection() {
  timeLine[0]++; //increments the detection count for time line. Location 0 is always the most recent 5-minute interval.

  timeStamp = Time.format(Time.now(), "%I:%M:%S %p %m/%d/%Y "); //Records formatted timestamp as hours:mins:secs am/pm month/day/year
  detectionMessage(detectionCount, timeStamp); //output motion detected message (for debugging and testing)

  sendEmail(); //call function to send email notification
}
```

This function is called with each motion detection.

The previously mentioned timeline array must be shifted every five minutes in order to stay up to date. For this purpose, the following function is called with each iteration of the main loop. The function uses time state variables like the speaker logic, only shifting the timeline array every 5 minutes.

```
void timeLineShift() {
    //if 5 mins have passed, shift timeline array to right -- for dynamic time line on website
    T1 = millis();
    if (T1 - T2 >= 300000) { //5 mins in ms
      for (int i = 11; i > 0; i--) {
        timeLine[i] = timeLine[i - 1];
      }
      timeLine[0] = 0;
      T2 = millis();
    }
}
```

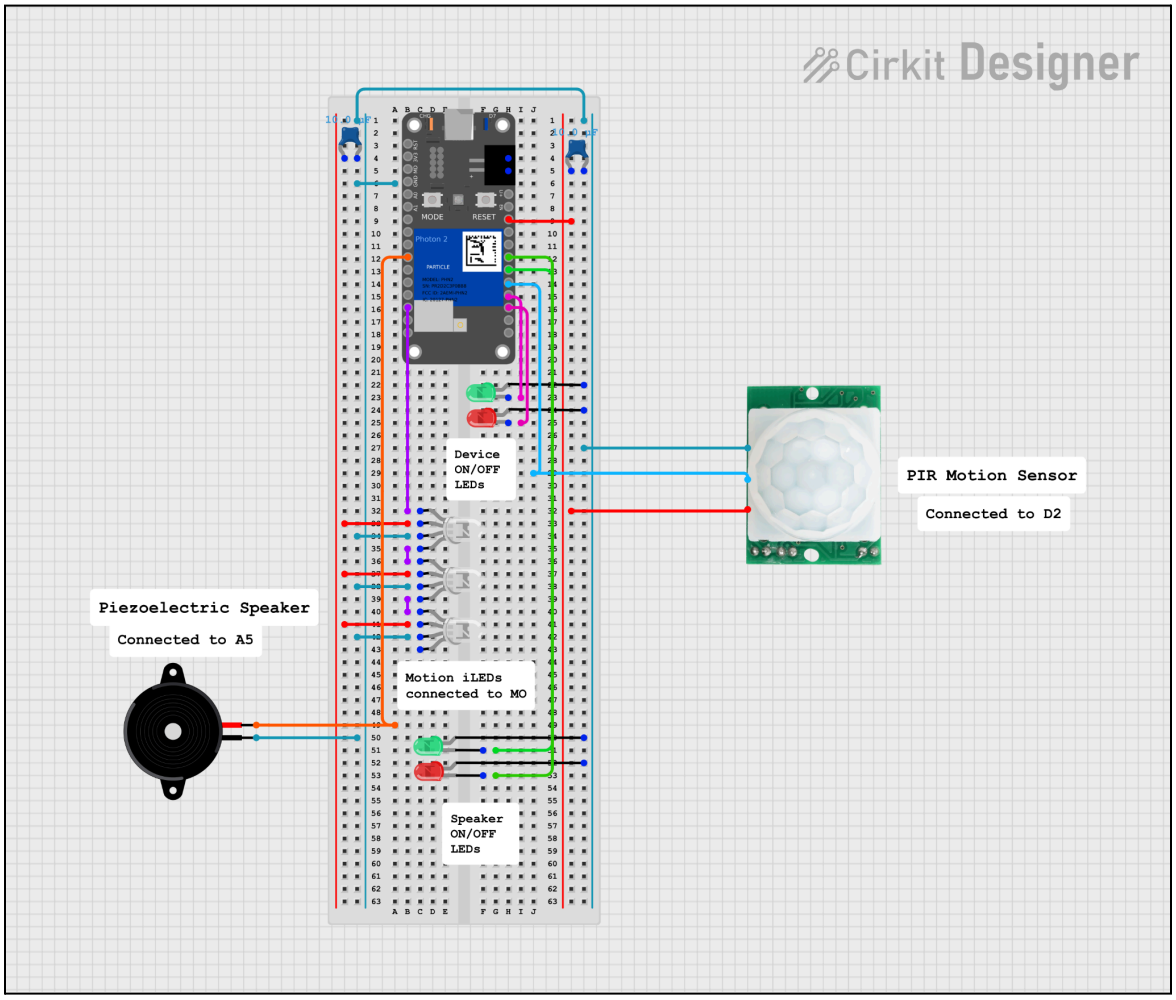
A common array shifting loop is used.

The loop does very little "blocking", as it runs quickly and infrequently.

Additional code includes LED updating functions, a function that sends email alerts by triggering a cloud event, which sends a webhook to IFTTT, the setup code, and global variable declarations. To view these, see *Appendix II*.

Finally, the HTML code for the interactive website, which is mostly generated by AI, is shown in *Appendix III*, and a visual representation of the website is shown in *Appendix IV*.

Appendix I: Wiring Diagram



Appendix II: Full Photon Code [PDF Download]

```
#include "Particle.h"
#include "neopixel.h"
#include <string>

using namespace std;

SYSTEM_MODE(AUTOMATIC);
SYSTEM_THREAD(ENABLED);
```

```
SerialLogHandler logHandler(LOG_LEVEL_INFO);

//learned how to use webhooks to send an email whenever motion is detected from:
https://docs.particle.io/integrations/webhooks/
// and https://docs.particle.io/integrations/community-integrations/ifttt/

                                                                ////DECLARATIONS:////
//for device/alarm toggling:
#define DEVICE_ON 1
#define DEVICE_OFF 0
#define ALARM_ON 1
#define ALARM_OFF 0
int deviceModeIs = DEVICE_ON;
int alarmModeIs = ALARM_OFF;

//for speaker/device LED indictors:
int device_ON_PIN = D3;
int device_OFF_PIN = D4;
int alarm_ON_PIN = D5;
int alarm_OFF_PIN = D6;

//pins for motion sensor and speaker:
int motionPIN = D2;
int speakerPIN = A5;

//variables for motion detection logic:
bool previousSensorState = false;
bool motionLogged = false; //used to ensure only 1 log (or output message) per
detection
bool currentMotion = false;
int detectionCount = 0;
int motionReading;

String timeStamp; //Learned time functions and formatting from:
https://docs.particle.io/reference/device-os/api/time/zone/

//speaker variables
unsigned long int timeNow;
unsigned long int timeLastBeep = 0;
bool highTone;
int beepCount = 0;
```

```
//Timeline variables
unsigned long int T1 = millis();
unsigned long int T2 = 0;
int timeLine[12] = {0}; //each of the 12 elements of this array represent a 5 minute
interval over the past hour
                      //the integer associated with them represent the # of
detections during that interval

//LED configuration:
int PIXEL_COUNT = 3;
#define PIXEL_PIN SPI // S0 Pin
int PIXEL_TYPE = WS2812;
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, PIXEL_TYPE);

//color presets for LEDs
int PixelColorRed  = strip.Color(   0,  255,   0);
int PixelColorGreen  = strip.Color(   255,  0,   0);
int PixelOFF  = strip.Color(   0,  0,   0);

//iLED functions
void RedLEDs();
void GreenLEDs();
void NoLEDs();
void LEDWipeForward(int color, int wait);
void LEDWipeBackwards(int color, int wait);

//serial port detection message for testing
void detectionMessage(int detectionCount, String timeStamp);

//log data (to be sent to website) for each detection
void logDetection();
void timeLineShift();

void sendEmail(); //for IFTTT email triggering

//Functions called by cloud functions
int deviceMode(String inputString);
int alarmMode(String inputString);
```

```
                                                        ////SETUP:////

void setup() {
 //motion sensor pin setup:
 pinMode(motionPIN, INPUT);
 motionReading = digitalRead(motionPIN);

 //for triggered red/green LED indicators for device/speaker toggling
 pinMode(device_ON_PIN, OUTPUT);
 pinMode(device_OFF_PIN, OUTPUT);
 pinMode(alarm_ON_PIN, OUTPUT);
 pinMode(alarm_OFF_PIN, OUTPUT);

 Serial.begin(9600);
 strip.begin();

 Time.zone(-5); // Converts particle time zone to Central Time (from UTC)

 //coud variables
 Particle.variable("DetectionCount", detectionCount);
 Particle.variable("DetectionTime", timeStamp);
 Particle.variable("CurrentMotion", currentMotion);

 //1 hour timeline: 5 min intervals, each variabe holds the # of detections during
that interval. Website accesses these cloud variables
 Particle.variable("Interval1", timeLine[0]);
 Particle.variable("Interval2", timeLine[1]);
 Particle.variable("Interval3", timeLine[2]);
 Particle.variable("Interval4", timeLine[3]);
 Particle.variable("Interval5", timeLine[4]);
 Particle.variable("Interval6", timeLine[5]);
 Particle.variable("Interval7", timeLine[6]);
 Particle.variable("Interval8", timeLine[7]);
 Particle.variable("Interval9", timeLine[8]);
 Particle.variable("Interval10", timeLine[9]);
 Particle.variable("Interval11", timeLine[10]);
 Particle.variable("Interval12", timeLine[11]);

 //cloud functions for toggling device/alarm
 Particle.function("DeviceToggle", deviceMode);
 Particle.function("AlarmToggle", alarmMode);
```

```
}

                                                          ////LOOP:////

void loop() {

 //Motion detection, motion logging, and LED logic:

   //Device toggled ON/OFF through website
   if(deviceModeIs == DEVICE_ON) {
     digitalWrite(device_ON_PIN, HIGH);
     digitalWrite(device_OFF_PIN, LOW);

     //IF: motion was just detected:
     if(motionReading == HIGH && previousSensorState == false) {
       currentMotion = true;
       previousSensorState = true; //to prevent retrigger
       motionLogged = false; //to prevent extra logs
       RedLEDs(); //LEDS --> red

       //if statement to make sure the motion is only logged once per detection
       if(motionLogged == false) {
         logDetection();
         motionLogged = true;
       }

     //ELSE IF: motion just stopped being detected:
     } else if(motionReading == LOW && previousSensorState == true) {
       previousSensorState = false;
       currentMotion = false; //no motion
       GreenLEDs(); //LEDS --> green
     }


 //Alarm logic (non-blocking, continuous over motion duration) ...
   if(alarmModeIs == ALARM_ON) {
     digitalWrite(alarm_ON_PIN, HIGH); //alarm green LED ON
     digitalWrite(alarm_OFF_PIN, LOW); //alarm red LED OFF

     if(currentMotion) {
       timeNow = millis();
```

```
      //IF: 250ms have elapsed since the last alarm tone
      if(timeNow-timeLastBeep >= 250) {
        timeLastBeep = timeNow;
        noTone(speakerPIN);

        //to make sure the alarm starts on high tone
        if(beepCount == 0) {
          highTone = true;
        }

        if(highTone) {
          tone(speakerPIN, 1800, 0); //1800hz (high pitched) sound
          beepCount++;
          highTone = false;
        }else if(!highTone) {
          tone(speakerPIN, 1200, 0); //1200hz (lower pitched) sound
          beepCount++;
          highTone = true;
        }
      }

    } else { //motion stopped
      noTone(speakerPIN); //speaker stops sounding
      if(!highTone) {
        highTone = true; //to make sure speaker starts with high tone next time
      }
    }

  } else { //alarm is toggled off
    digitalWrite(alarm_ON_PIN, LOW); //alarm green LED OFF
    digitalWrite(alarm_OFF_PIN, HIGH); //alarm red LED ON
  }

} else { //device is toggled off, all LEDs are set accordingly
  NoLEDs();
  digitalWrite(device_OFF_PIN, HIGH);
  digitalWrite(device_ON_PIN, LOW);
  digitalWrite(alarm_OFF_PIN, HIGH);
  digitalWrite(alarm_ON_PIN, LOW);
}
```

```
 timeLineShift(); //shift timeline array every 5 mins. (The function checks if 5
mins have passed and acts accordingly)
}



                                                          ////FUNCTIONS:////

//Called by cloud function (from website) for toggling device on/off
int deviceMode(String inputString) {
 if(inputString == "Device OFF") {
   deviceModeIs = DEVICE_OFF;
   return 0;
 } else if(inputString == "Device ON") {
   deviceModeIs = DEVICE_ON;
   GreenLEDs();
   return 1;
 } else {
   return -1; //something went wrong
 }
}

//Called by cloud function (from website) for toggling speaker on/off
int alarmMode(String inputString){
 if(inputString == "Alarm OFF") {
   alarmModeIs = ALARM_OFF;
   return 0;
 } else if(inputString == "Alarm ON") {
   alarmModeIs = ALARM_ON;
   return 1;
 } else {
   return -1; //something went wrong
 }
}

void logDetection() {
 timeLine[0]++; //increments the detection count for the timeline. Location 0 is
always the most recent 5-minute interval.

 timeStamp = Time.format(Time.now(), "%I:%M:%S %p %m/%d/%Y "); //Records formatted
timestamp as hours:mins:secs am/pm month/day/year
```

```
 detectionMessage(detectionCount, timeStamp); //output motion detected message (for debugging and testing)

 sendEmail(); //call function to send email notification
}

void timeLineShift() {
   //if 5 mins have passed, shift timeline array to right -- for dynamic time line on website
   T1 = millis();
   if (T1 - T2 >= 300000) { //5 mins in ms
     for (int i = 11; i > 0; i--) {
       timeLine[i] = timeLine[i - 1];
     }
     timeLine[0] = 0;
     T2 = millis();
   }
}

void RedLEDs() {
   LEDWipeForward(strip.Color(0, 255, 0), 30); //cool animation from neopixel library
   strip.setPixelColor(0, PixelColorRed);
   strip.setPixelColor(1, PixelColorRed);
   strip.setPixelColor(2, PixelColorRed);
   strip.show();
}

void GreenLEDs() {
   LEDWipeBackwards(strip.Color(255, 0, 0), 30);
   strip.setPixelColor(0, PixelColorGreen);
   strip.setPixelColor(1, PixelColorGreen);
   strip.setPixelColor(2, PixelColorGreen);
   strip.show();
}

void NoLEDs() {
 strip.setPixelColor(0, PixelOFF);
 strip.setPixelColor(1, PixelOFF);
 strip.setPixelColor(2, PixelOFF);
 strip.show();
```

```
}

//LED Wipe functions; inspired by functions included in neopixel library example
folder
void LEDWipeForward(int color, int wait) {
 for(int i=0; i<strip.numPixels(); i++) {
   strip.setPixelColor(i, color);
   strip.show();
   delay(wait);
 }
}

//slightly modified to wipe in reverse direction
void LEDWipeBackwards(int color, int wait) {
 for(int i=strip.numPixels(); i > 0; i--) {
   strip.setPixelColor(i, color);
   strip.show();
   delay(wait);
 }
}

//for serial port debugging
void detectionMessage(int detectionCount, String timeStamp) {
     Serial.print("MOTION DETECTION ");
     Serial.print(detectionCount);
     Serial.print(" | ");
     Serial.println(timeStamp); //serial print only works with c strings
}

void sendEmail() {
 Particle.publish("sendEmail", timeStamp, PRIVATE); //publishes particle event,
which triggers webhook to IFTTT, sending email
}
```

Appendix III: Full Webpage Code (AI) PDF Download

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
 <meta charset="UTF-8">
 <title>StepSnitch | Dashboard</title>
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
 <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

   <script>
     const deviceID = "0a10aced202194944a065074";
     const accessToken = "2fd4eaf9ffe647ab84c8c109e4326f72761ae722";
     const baseURL = `https://api.particle.io/v1/devices/${deviceID}`;
   </script>

 <style>
   :root {
     --bg: #0a0a0a;
     --bg-alt: #1a1a1a;
     --text: #eaeaea;
     --card-bg: rgba(255, 255, 255, 0.08);
     --card-shadow: rgba(255, 255, 255, 0.15);
     --text-light: #bbb;
     --highlight-good: #00ff88;
     --highlight-alert: #ff4444;
     --button-bg: #ffffff;
     --button-text: #000000;
     --scrollbar: #ffffff;
   }

   .light-theme {
     --bg: #f4f4f4;
     --bg-alt: #eaeaea;
     --text: #111;
     --card-bg: rgba(0, 0, 0, 0.05);
     --card-shadow: rgba(0, 0, 0, 0.1);
     --text-light: #333;
     --highlight-good: #008800;
     --highlight-alert: #cc0000;
     --button-bg: #000000;
     --button-text: #ffffff;
     --scrollbar: #000000;
   }
```

```css
* {
  box-sizing: border-box;
}

body {
  margin: 0;
  padding: 2rem;
  font-family: 'Segoe UI', sans-serif;
  background: linear-gradient(120deg, var(--bg), var(--bg-alt), var(--bg));
  background-size: 400% 400%;
  animation: gradientShift 30s ease infinite;
  color: var(--text);
  overflow-x: hidden;
  min-height: 100vh;
  transition: background 0.4s, color 0.4s;
}

@keyframes gradientShift {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}

h1 {
  font-size: 3rem;
  margin: 0;
  color: var(--text);
  text-shadow: 0 0 8px var(--text-light);
  animation: fadeSlideIn 1s ease;
}

.subtitle {
  color: var(--text-light);
  margin-bottom: 2.5rem;
  font-style: italic;
  animation: fadeSlideIn 1.5s ease;
}

.card {
  background: var(--card-bg);
```

```
  border-radius: 20px;
  padding: 2rem;
  margin-bottom: 2rem;
  backdrop-filter: blur(12px);
  box-shadow: 0 0 20px var(--card-shadow);
  animation: fadeSlideIn 2s ease;
}

.status span {
  font-weight: bold;
  font-size: 1.4rem;
  display: inline-flex;
  align-items: center;
}

.status .no-motion {
  color: var(--highlight-good);
}

.status .motion {
  color: var(--highlight-alert);
}

.controls button, .theme-toggle {
  margin: 0.5rem;
  padding: 0.8rem 1.6rem;
  background: var(--button-bg);
  border: none;
  border-radius: 12px;
  color: var(--button-text);
  font-size: 1rem;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 0 10px var(--card-shadow);
}

.controls button:hover, .theme-toggle:hover {
  transform: scale(1.05);
  box-shadow: 0 0 20px var(--card-shadow);
}
```

```
   .control-row {
 display: flex;
 align-items: center;
 justify-content: space-between;
 margin-bottom: 1.5rem;
}

.status-indicator {
 display: flex;
 align-items: center;
 gap: 0.6rem;
}

.dot {
 width: 12px;
 height: 12px;
 border-radius: 50%;
 background-color: var(--highlight-good); /* Default green */
 box-shadow: 0 0 8px var(--highlight-good);
}

   .log ul {
     list-style-type: none;
     padding: 0;
     max-height: 250px;
     overflow-y: auto;
   }

   .log li {
     padding: 0.7rem 0;
     border-bottom: 1px solid #333;
     color: var(--text);
   }

   .log button {
     margin-top: 1rem;
     padding: 0.4rem 0.8rem;
     font-size: 0.75rem;
     background: var(--button-bg);
     border: none;
     border-radius: 8px;
```

```
  color: var(--button-text);
  cursor: pointer;
  box-shadow: 0 0 8px var(--card-shadow);
}

.timeline {
  margin-top: 1.5rem;
  font-size: 0.9rem;
  color: var(--text-light);
}

.timeline h3 {
  margin-bottom: 0.5rem;
  font-size: 1.2rem;
  color: var(--text);
}

#timelineChart {
  max-height: 200px;
}

.theme-toggle {
  position: absolute;
  top: 1.5rem;
  right: 2rem;
}

@keyframes fadeSlideIn {
  from {
    opacity: 0;
    transform: translateY(30px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}

::-webkit-scrollbar {
  width: 8px;
}
```

```
  ::-webkit-scrollbar-thumb {
    background: var(--scrollbar);
    border-radius: 4px;
  }

  .log li:hover {
  background: rgba(255, 255, 255, 0.05);
  border-radius: 8px;
  padding-left: 0.5rem;
  transition: all 0.3s ease;
}

.timeline canvas {
  transition: all 0.3s ease;
}

</style>
</head>
<body>

<button class="theme-toggle" onclick="toggleTheme()">Switch to Light Mode</button>
<h1>StepSnitch</h1>
<div class="subtitle">An infrared motion-triggered security system.</div>

<div class="card status">
  <h2>Current Status:</h2>
  <p id="motionStatus"><span class="no-motion">✔ No Motion Detected</span></p>

  <p><strong>Last Detection Time:</strong> <span id="detectionTime">--</span></p>
</div>

<div class="card controls">
  <h2>Controls</h2>
   <div class="control-row">
    <div class="status-indicator">
      <span id="deviceDot" class="dot"></span>
      <span id="deviceStatus">Device is ON</span>
    </div>
    <button onclick="toggleDevice()">Toggle Device</button>
  </div>
```

```
    <div class="control-row">
     <div class="status-indicator">
       <span id="alarmDot" class="dot"></span>
       <span id="alarmStatus">Alarm is ON</span>
     </div>
     <button onclick="toggleAlarm()">Toggle Alarm</button>
   </div>
</div>


<div class="card activity">
   <h2>Recent Activity</h2>
    <div class="timeline" style="margin-bottom: 2rem;">
     <h3>Timeline (Past Hour)</h3>
     <canvas id="timelineChart"></canvas>
   </div>
    <div class="log">
     <h3>Detection Log</h3>
     <ul id="logList">
       <li>04/09/25 14:36 - No motion</li>
       <li>04/09/25 14:21 - Motion detected!</li>
     </ul>
     <button onclick="clearLog()">Clear Log</button>
   </div>
</div>

<script>
   // Track toggle state
   let deviceOn = true;
   let alarmOn = true;
   let previousMotionDetected = false; // NEW: track previous motion state
    // Cloud function buttons
   function toggleDevice() {
     const newState = deviceOn ? "Device OFF" : "Device ON";
     $.ajax({
       url: `${baseURL}/DeviceToggle`,
       method: "POST",
       data: {
         access_token: accessToken,
         args: newState
       },
```

```
      success(data) {
        alert(`DeviceToggle Result: ${data.return_value}`);
        deviceOn = !deviceOn;
        updateDeviceStatus(); // <- NEW
      },
      error(err) {
        console.error("Device toggle failed", err);
      }
    });
  }
   function toggleAlarm() {
    const newState = alarmOn ? "Alarm OFF" : "Alarm ON";
    $.ajax({
      url: `${baseURL}/AlarmToggle`,
      method: "POST",
      data: {
        access_token: accessToken,
        args: newState
      },
      success(data) {
        alert(`AlarmToggle Result: ${data.return_value}`);
        alarmOn = !alarmOn;
        updateAlarmStatus(); // <- NEW
      },
      error(err) {
        console.error("Alarm toggle failed", err);
      }
    });
  }

  function updateDeviceStatus() {
    const statusSpan = document.getElementById('deviceStatus');
    const dot = document.getElementById('deviceDot');

    if (deviceOn) {
      statusSpan.textContent = "Device Status: ON";
      dot.style.backgroundColor = "var(--highlight-good)";
      dot.style.boxShadow = "0 0 8px var(--highlight-good)";
    } else {
      statusSpan.textContent = "Device Status: OFF";
      dot.style.backgroundColor = "var(--highlight-alert)";
```

```
      dot.style.boxShadow = "0 0 8px var(--highlight-alert)";
    }
  }

  function updateAlarmStatus() {
    const statusSpan = document.getElementById('alarmStatus');
    const dot = document.getElementById('alarmDot');

    if (alarmOn) {
      statusSpan.textContent = "Alarm Status: ON";
      dot.style.backgroundColor = "var(--highlight-good)";
      dot.style.boxShadow = "0 0 8px var(--highlight-good)";
    } else {
      statusSpan.textContent = "Alarm Status: OFF";
      dot.style.backgroundColor = "var(--highlight-alert)";
      dot.style.boxShadow = "0 0 8px var(--highlight-alert)";
    }
  }

   // Status & log helpers
  function clearLog() {
    if (confirm("Clear all log entries?")) {
      document.getElementById('logList').innerHTML = '';
    }
  }
   function updateMotionStatus(isDetected) {
    const motionStatus = document.getElementById("motionStatus");
    if (isDetected == true) {
      motionStatus.innerHTML = '<span class="motion">🚨 Motion Detected!</span>';
    } else {
      motionStatus.innerHTML = '<span class="no-motion">✔ No Motion
Detected</span>';
    }
  }
   function toggleTheme() {
    const body = document.body;
    const btn = document.querySelector('.theme-toggle');
    const light = body.classList.toggle('light-theme');
    btn.textContent = light ? 'Switch to Dark Mode' : 'Switch to Light Mode';
  }
   // Fetch all cloud variables
```

```
function fetchParticleVariables() {
  // Fetch CurrentMotion first
  $.ajax({
    url: `${baseURL}/CurrentMotion`,
    method: "GET",
    headers: {
      Authorization: `Bearer ${accessToken}`
    },
    success: function(data) {
      console.log("CurrentMotion:", data.result);
      const isDetected = data.result == true;
      updateMotionStatus(isDetected);
       // If motion just started, log detection time
      if (isDetected && !previousMotionDetected) {
        fetchAndLogDetectionTime();
      }
      previousMotionDetected = isDetected;
    },
    error: function(err) {
      console.error("Failed to fetch CurrentMotion", err);
    }
  });
   // Fetch DetectionTime (for updating status, not log)
  $.ajax({
    url: `${baseURL}/DetectionTime`,
    method: "GET",
    headers: {
      Authorization: `Bearer ${accessToken}`
    },
    success: function(data) {
      console.log("DetectionTime:", data);
      $("#detectionTime").text(data.result);
    },
    error: function(err) {
      console.error("Failed to fetch DetectionTime", err);
    }
  });
}
 // NEW: fetch and insert DetectionTime into the log
function fetchAndLogDetectionTime() {
  $.ajax({
```

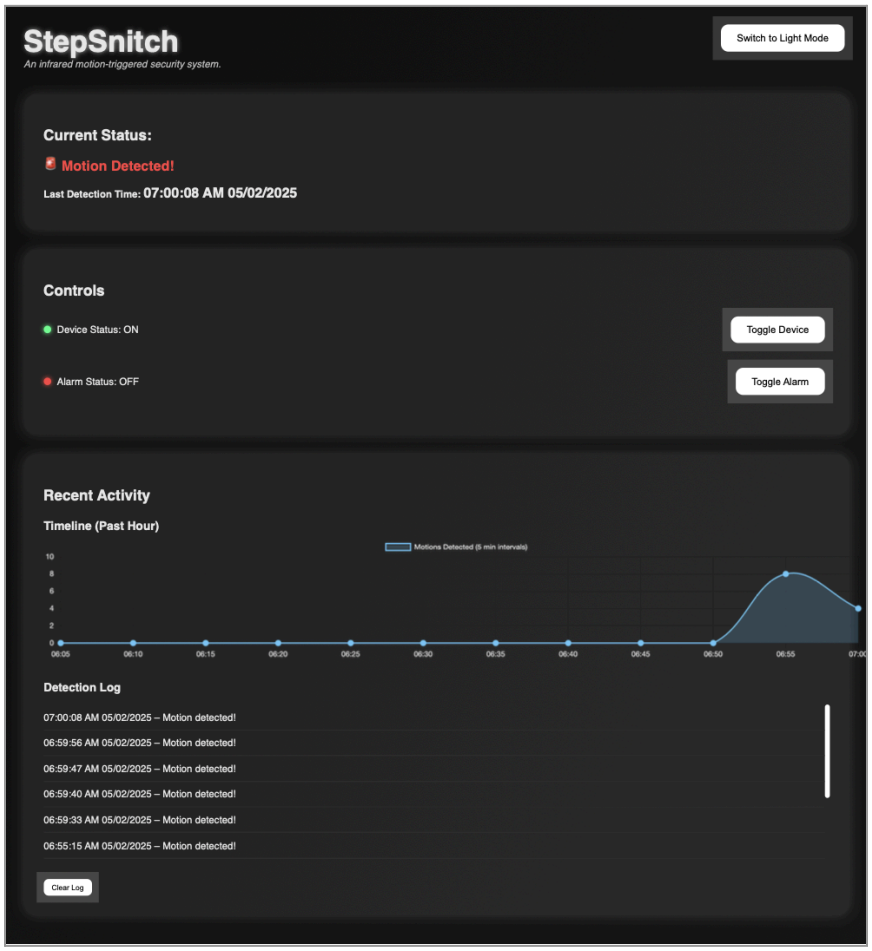
```
      url: `${baseURL}/DetectionTime`,
      method: "GET",
      headers: {
        Authorization: `Bearer ${accessToken}`
      },
      success: function(data) {
        const logList = document.getElementById('logList');
        const newEntry = document.createElement('li');
        newEntry.textContent = `${data.result} - Motion detected!`;
        logList.prepend(newEntry);
      },
      error: function(err) {
        console.error("Failed to fetch DetectionTime for log", err);
      }
    });
  }
   // Start polling
  setInterval(fetchParticleVariables, 1500);

  fetchParticleVariables();
  updateDeviceStatus();
  updateAlarmStatus();

</script>
  <script>
const intervalVars = [
 "Interval1", "Interval2", "Interval3", "Interval4",
 "Interval5", "Interval6", "Interval7", "Interval8",
 "Interval9", "Interval10", "Interval11", "Interval12"
];

// Prebuild labels for last 60 mins in 5 min slices
function generateLabels() {
 const labels = [];
 const now = new Date();
  for (let i = 0; i < 12; i++) {
   const past = new Date(now.getTime() - (i * 5 * 60000)); // 5 minutes * i
   const hours = String(past.getHours()).padStart(2, '0');
   const minutes = String(past.getMinutes()).padStart(2, '0');
   labels.push(`${hours}:${minutes}`);
 }
```

```
  return labels.reverse(); // So left -> right is oldest -> newest
}

// Setup Chart
const ctx = document.getElementById('timelineChart').getContext('2d');
const timelineChart = new Chart(ctx, {
 type: 'line',
 data: {
   labels: generateLabels(),
   datasets: [{
     label: 'Motions Detected (5 min intervals)',
     data: Array(12).fill(0),
     borderColor: 'rgba(100, 200, 255, 1)',
     backgroundColor: 'rgba(100, 200, 255, 0.2)',
     tension: 0.4,
     fill: true,
     pointRadius: 4,
     pointHoverRadius: 8, /* 👈 NEW */
     pointBackgroundColor: 'rgba(100, 200, 255, 1)',
     borderWidth: 2
   }]
 },
 options: {
   responsive: true,
   scales: {
     y: {
       beginAtZero: true,
       ticks: {
         color: getComputedStyle(document.body).getPropertyValue('--text')
       }
     },
     x: {
       ticks: {
         color: getComputedStyle(document.body).getPropertyValue('--text')
       }
     }
   },
   plugins: {
     legend: {
       labels: {
         color: getComputedStyle(document.body).getPropertyValue('--text')
```

```
      }
    }
  }
 }
});


// Function to fetch all intervals
function fetchTimelineIntervals() {
 const promises = intervalVars.map(interval =>
   $.ajax({
     url: `${baseURL}/${interval}`,
     method: "GET",
     headers: {
       Authorization: `Bearer ${accessToken}`
     }
   })
 );

 Promise.all(promises)
   .then(results => {
     const detections = results.map(r => r.result);

     timelineChart.data.datasets[0].data = detections.reverse(); // Reverse to
oldest -> newest
     timelineChart.data.labels = generateLabels(); // Update time labels
     const maxVal = Math.max(...detections);

     // Adjust Y-axis dynamically
     timelineChart.options.scales.y.max = maxVal < 5 ? 5 : Math.ceil(maxVal * 1.2);

     timelineChart.update();
   })
   .catch(error => {
     console.error("Failed to fetch timeline intervals", error);
   });
}

// Update every 10 seconds
setInterval(fetchTimelineIntervals, 10000);
fetchTimelineIntervals(); // initial load
```

```
</script>


</body>
</html>
```

Appendix IV: Website Visual Representation



Note: The timeline does not go off the page on the actual website; this is a visual glitch. The boxes around the buttons also aren't on the actual page.