Project Surveillance
Report

Aditya Harikrish - 2020111009
Anusha Nath Roy - 2020101124
Gaurav Singh - 2020111014
Yash Anil Bhatia - 2020101007

Initial Objectives

Our initial objective was to create a security device that clicked several pictures using an ESP32-Cam module whenever motion was detected by a PIR sensor. The motivation was to save disc space by only clicking pictures when necessary.

Challenges and Consequent Change in Objectives

We were not able to upload any code to the ESP32-Cam module. We tried several options, but were unable to get it to work. Therefore, we changed our objective while retaining the same theme—security. We designed

1. a radar with an ultrasonic distance sensor, and
2. a motion detection warning system integrated into a Telegram (a popular messaging app) bot for the end user to interact with.

Implementation

Our circuit is spread across two nodes—Node 1 and Node 2—due to the constraints of distance learning.

Node 1's End

Node 1 has the radar. It has a 180° sweeping angle. It cycles through 7 angles—0°, 30°, 60°, 90°, 120°, 150° and 180°—and records the distance measured by the distance sensor for each of these angles.

Node 1 first runs `src.py`, which sets up oneM2M by creating a container. On oneM2M, we have the data for the 7 graphs that plot distance measured versus time, one for each of the axes along which distance is measured.

Then, it runs `radar/radar.ino` and `radar_plotter.py` simultaneously. `radar/radar.ino` is what runs the radar itself while `radar_plotter.py` plots an interactive graph (a spider plot) in `localhost:8000` using the data stored in the oneM2M container, one for each axis (see fig. 1).

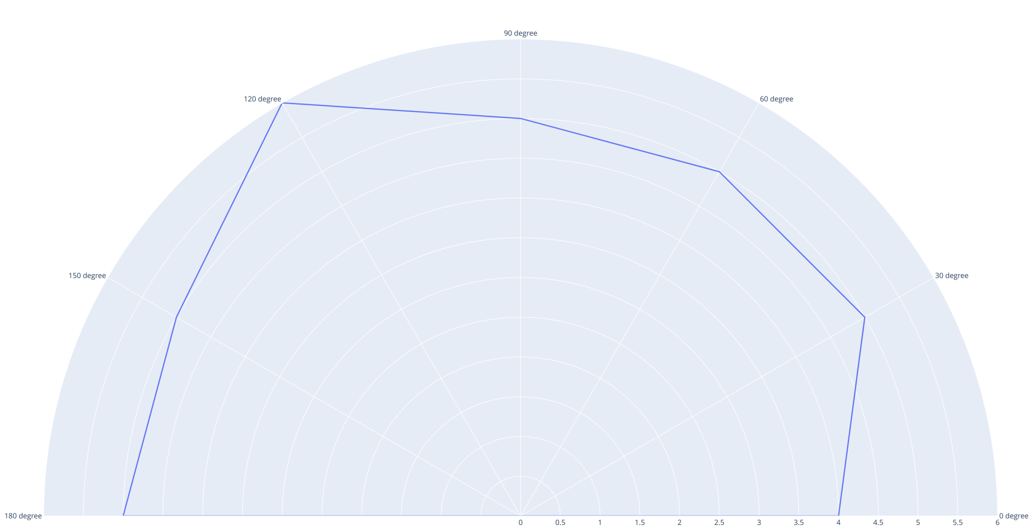


Figure 1: Interactive graph in the browser, plotter by `radar_plotter.py`

Node 2's End

Node 2 has the motion detection system. It has a PIR motion sensor along with a buzzer and an LED. Originally, the plan was to integrate both, the radar and the motion sensor into a single system, but that proved infeasible due to online classes. A graph of the number of times motion is detected is plotted on ThingSpeak as a function of time (see fig. 2). Node 2 runs `telegram/telegram.ino`, which contains the code for Telegram bot and the end-user's interaction with Node 2's circuit. The bot can be given the following commands:

1. `/start` or `/help` for a list of commands that can be executed,
2. `/led_warn` to turn the LED on and `/led_off` to turn the LED off,
3. `/buzzer_warn` to turn the buzzer on and `/buzzer_off` to turn the buzzer off, and
4. `/led_state` and `/buzzer_state` to learn about the status of the LED and the buzzer respectively (i.e. whether they are on or off).

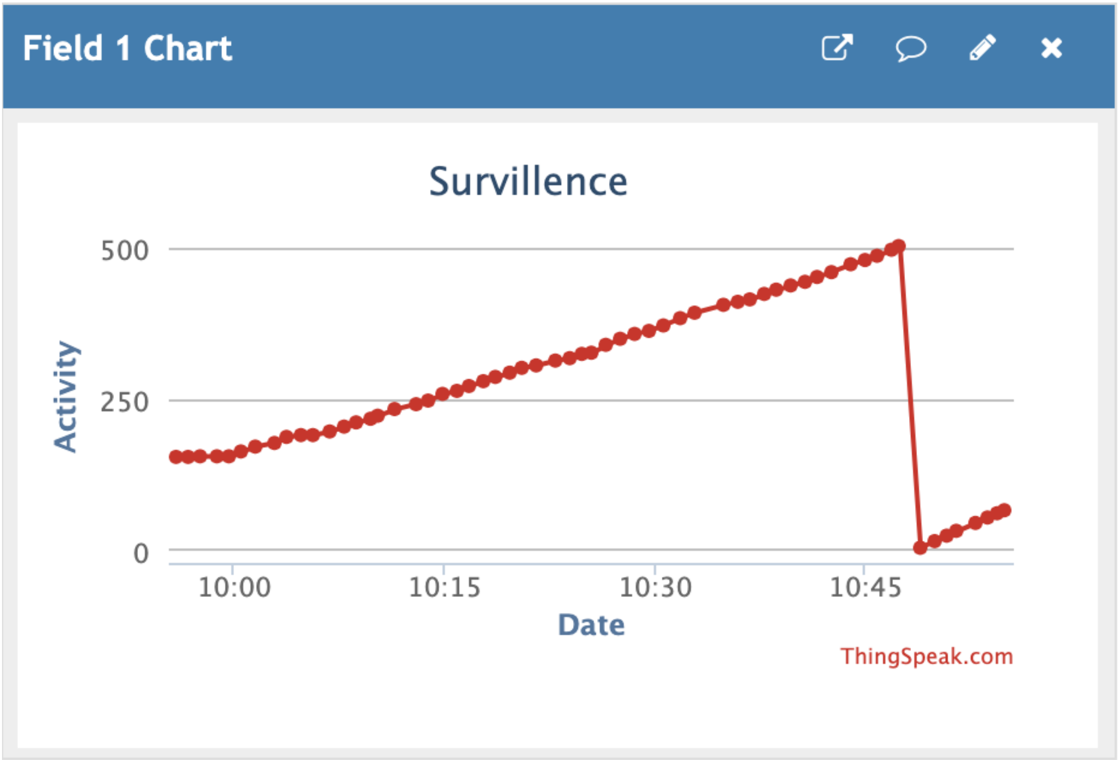
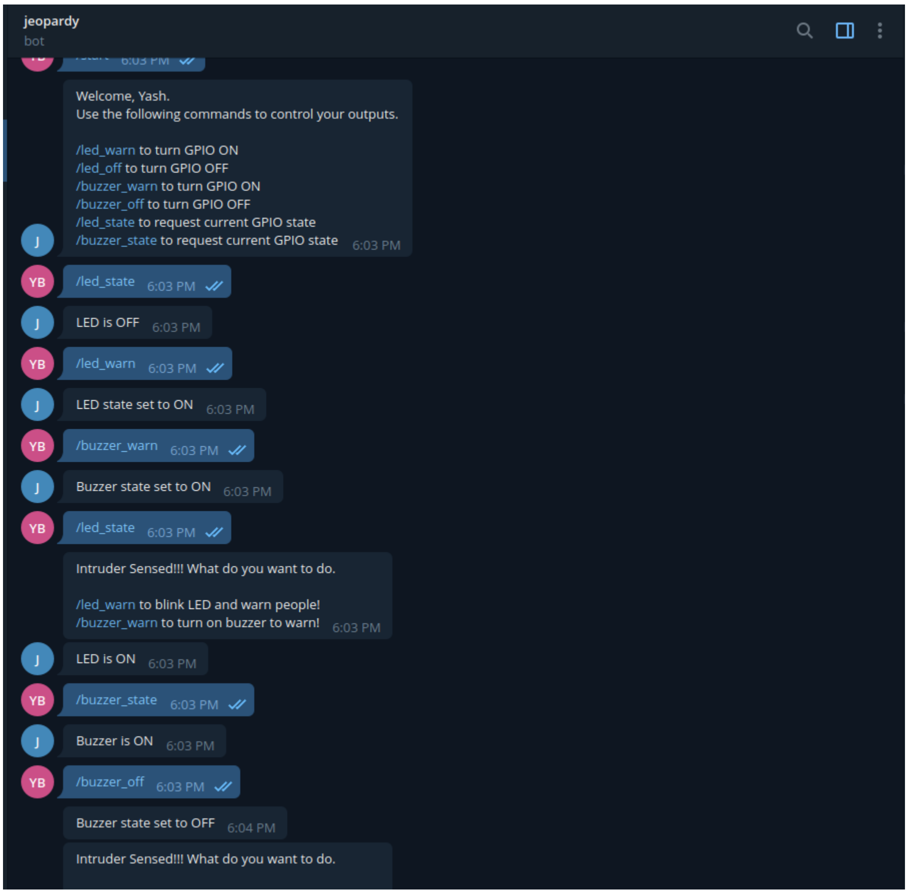


Figure 2: Activity vs time plot on ThingSpeak



YB: /start 6:03 PM

J: Welcome, Yash.
Use the following commands to control your outputs.

/led_warn to turn GPIO ON
/led_off to turn GPIO OFF
/buzzer_warn to turn GPIO ON
/buzzer_off to turn GPIO OFF
/led_state to request current GPIO state
/buzzer_state to request current GPIO state 6:03 PM

YB: /led_state 6:03 PM

J: LED is OFF 6:03 PM

YB: /led_warn 6:03 PM

J: LED state set to ON 6:03 PM

YB: /buzzer_warn 6:03 PM

J: Buzzer state set to ON 6:03 PM

YB: /led_state 6:03 PM

Intruder Sensed!!! What do you want to do.

/led_warn to blink LED and warn people!
/buzzer_warn to turn on buzzer to warn! 6:03 PM

J: LED is ON 6:03 PM

YB: /buzzer_state 6:03 PM

J: Buzzer is ON 6:03 PM

YB: /buzzer_off 6:03 PM

Buzzer state set to OFF 6:04 PM

Intruder Sensed!!! What do you want to do.

Figure 3: The Telegram bot