

# Project Surveillance Report

Aditya Harikrish  
Anusha Nath Roy  
Gaurav Singh  
Yash Anil Bhatia

## 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, 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 of our teammates—Gaurav and Anusha—due to the constraints of distance learning.

### Gaurav's End

Gaurav has the radar. It has a  $180^\circ$  sweeping angle. It cycles through 7 angles— $0^\circ$ ,  $30^\circ$ ,  $60^\circ$ ,  $90^\circ$ ,  $120^\circ$ ,  $150^\circ$  and  $180^\circ$ —and records the distance measured by the distance sensor for each of these angles.

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

Then, he 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`, one for each axis (see fig. 1).

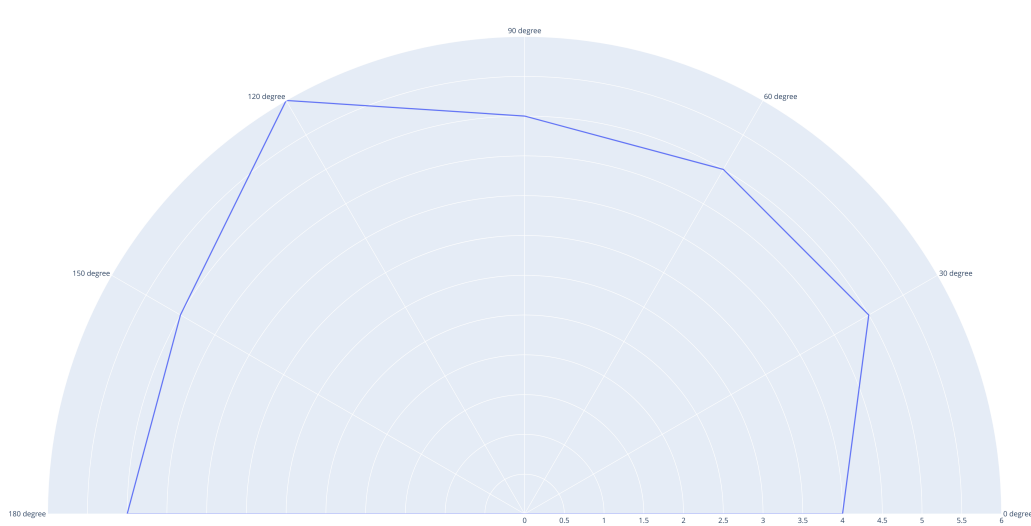


Figure 1: Interactive graph in the browser, plotter by `radar_plotter.py`

## Anusha's End

Anusha has the motion detection system. She 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). Anusha runs `telegram/telegram.ino`, which contains the code for Telegram bot and the end-user's interaction with Anusha'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).

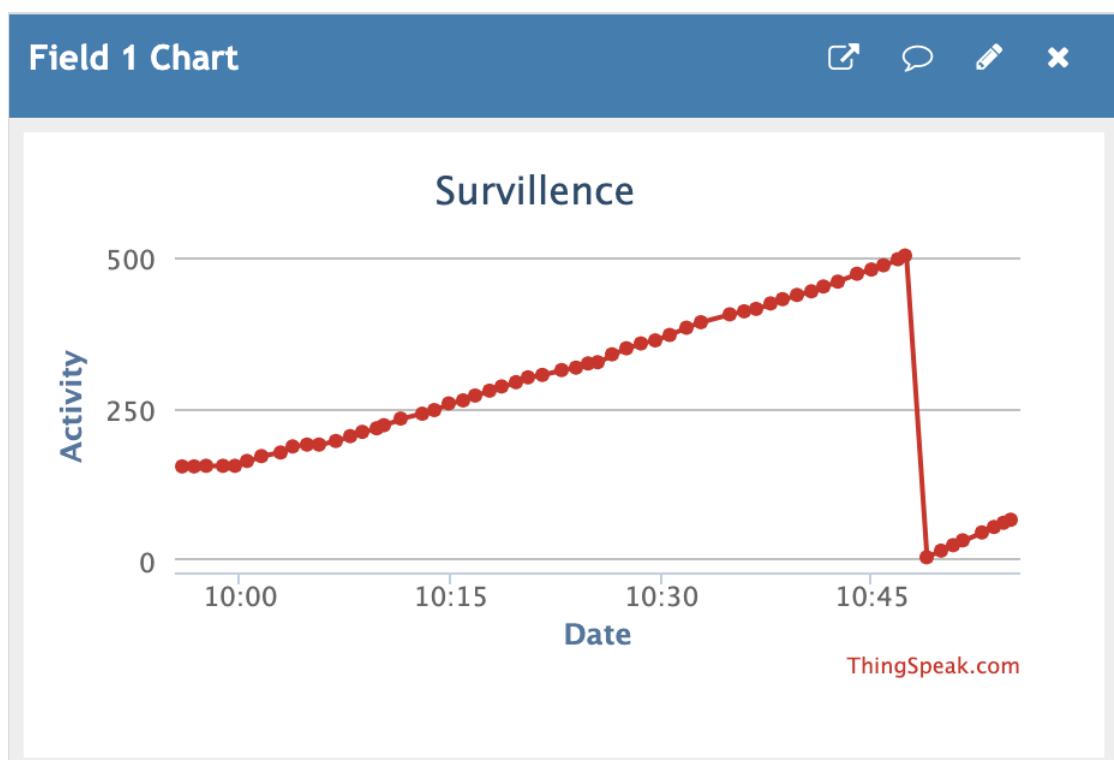


Figure 2: Activity vs time plot on ThingSpeak