EC 544 Project Proposal

Title and Team Information

Title: Face Recognition Security Alert System

Aya Kassem: ayak@bu.edu

Kaede Kawata: kkawata@bu.edu

Introduction

Internet of Things (IoT) networks are networks of physical devices that are connected and that can communicate data over the internet. Those networks can be very useful when implemented in emergency alert systems: such implementation would allow a network of IoT devices to not only detect an emergency, such as a security breach, for example, but to also report this emergency and send out alerts to other devices on the same network. A useful way of ensuring security and confidentiality using the Internet of Things could be to detect a security breach (using facial recognition, for example), and send out an alert to the closest device/person to the breach by tracking the locations of all IoT devices on the network.

One rising problem with IoT when used with WiFi, Bluetooth, or cellular hotspot, however, is that ensuring the connectivity of devices on such networks can be expensive in terms of both money and power. Additionally, the connectivity of the devices could be very range-limited, which could be difficult to deal with for applications where longer-range communication between different devices is needed. Our goal is to develop a security system that will grant us

the benefits of relying on an IoT network all the while avoiding the IoT connectivity problem just stated.

A good way to achieve this goal would be to use mesh networking to build our own decentralized IoT network. The mesh network itself will allow a lower power and wider range connectivity on the IoT network compared to other previously stated options such as WiFi, Bluetooth or cellular networks. Additionally, moving from a centralized to decentralized network would add more security to the system by shifting its control to multiple nodes across the network instead of one central control node. We will increase the security of this network even more by using a custom-built router for it using the “OpenWrt” open-source Linux operating system. The plan is to use this mesh IoT network to track and communicate the location of all devices (which will be ESP32 devices connected to GPS modules) on the network. Another ESP32 will play the role of the network’s gateway. In addition to being the gateway, it will be connected to both a GPS and a camera module and will allow us to use Computer Vision to perform facial recognition on humans attempting to unlock a door. An unauthorized human unlocking the door will be considered a “security breach”, and an alert signal will be sent to an ESP32 node on the mesh network. The tracked and communicated locations would then be used to alarm the closest active device on the network by making an LED blink and a buzzer go off. This project would be connected to Boston University’s EC 544 course by relying on concepts such as Decentralized Computing, Computer Vision, Computer Networks, and Authorization. We do not have any prior artwork related to this project, and we would be getting started on it for the scope of this class.

Resources

To build our project, we will need a combination of hardware and software resources which can be grouped into the different sub-goals that our project has to accomplish:

- **Facial Recognition**
    - ESP32 CAM
    - Micro-USB cable
    - FTDI Programmer Board
    - Jumper Wires/Breadboarding Equipment
    - ESP-IDF

- **Door Lock Simulation Mechanism**
    - Contact Switch
    - Jumper Wires/Breadboarding Equipment
    - ESP-IDF

- **Mesh Network Router**
    - Raspberry Pi (model 3 or 4)
    - Raspberry Pi Imager
    - MicroSD Card
    - OpenWrt Image
    - Monitor
    - Keyboard & Mouse

    - Wifi Router
    - Ethernet LAN Cable
    - USB-C Cable
    - Micro-HDMI cable

- **Mesh Network Trackers**
    - 3 HiLetgo ESP32-WROOM-32E Modules
    - 3 Micro-USB cables
    - 3 Neo-6M-0 GPS Modules
    - Jumper Wires/Breadboarding Equipment
    - ESP-IDF

- **Alert System**
    - 2 LEDs
    - 2 Buzzers
    - Jumper Wires/Breadboarding Equipment
    - ESP-IDF

Technical Risk Areas & Risk Management

As our project goes on, we acknowledge that we might face some issues that can impact our project directly or indirectly. The main reason behind the existence of these issues is the fact that there are many tools that, despite us being unfamiliar with, are the best solutions to achieve the different goals we want our project to accomplish.

We therefore decided to use those solutions, but take into account the technical risks that our unfamiliarity with them creates. Here are some examples:

1. **Building Our Own Decentralized Network**

Due to our lack of familiarity with the concept, building a decentralized network across different microcontrollers such as ESP32 modules could be challenging.

We plan on building our own mesh network and have the gateway of the mesh network be an ESP32 connected to an access point, and then have it communicate wirelessly to other ESP32 devices via mesh networking. However, having never built such a system before and not having gone over mesh networking in class yet, our own experience with mesh networking comes from online resources. We therefore acknowledge that we might run into technical complexities and challenges that might require time and debugging, and we will make sure to closely follow the documentation for the different modules of the mesh network to avoid any failures caused by those complexities.

2. **Building our own router**

In order to build a decentralized mesh network, one node on the network, the gateway (which will be an ESP32 CAM) has to be connected to WiFi. Instead of using an ISP WiFi router, we decided to increase the security of our network by using OpenWrt to build our own WiFi access point.

Despite the Linux Operating System not being new to us, we had not heard of OpenWrt before having to brainstorm ideas for this project. A risk here is that according to online documentation, using OpenWrt incorrectly can easily corrupt the firmware of a device. We will therefore have to

be very careful and follow instructions very closely (while making sure instructions are correct) when flashing the OpenWrt image onto the Raspberry Pi.

3. Building the Interfaces Between the Different Parts of the Project

We have a pretty good idea and understanding of how each individual module within our project could be built to achieve the "sub-goals" of our project stated above (in the resources), and we made sure to do as much research as possible to ensure that those different sub-modules are as compatible as possible.

Based on previous experience, however, getting different modules to work together always ends up being a challenging task. We therefore estimate that getting the project to work as a whole after getting individual parts of it working (aka linking the facial recognition, mesh network trackers, and alert system parts together) will also be a technical risk to seriously take into account. Compatibility can even be a problem we will have to take care of within each module itself because of the different devices we want to interface and communicate with: some devices will be communicating on the mesh Network (the ESP32-WROOM-32E modules), some will run over WiFi (ESP-32 CAM), some will communicate over a serial connection, etc.

To solve this problem we will spend some time drawing a diagram involving all the modules communicating in this project. This also means taking into account the input and output of each module when establishing the connection between modules and using online resources to make sure that it is possible for each module to take the corresponding input and output we want it to have.

4. Using the ESP-IDF Environment

For this project, we will be using ESP-IDF to program the ESP32 modules instead of Arduino IDE. We decided that ESP-IDF would be the better option because it supports FreeRTOS and has been developed by Espressif and is native to ESP32. This means that using it would allow us to use the ESP32 board to its full potential if needed.

The ESP-IDF environment, however, is not as intuitive to use as the Arduino IDE, and since neither group members have real experience with ESP-IDF, one of the challenges of this project would be to set up this new environment and familiarize ourselves with it. We will therefore take into consideration the learning curve that will be required to become comfortable with using ESP-IDF, and although we will try to avoid this option, we might also consider using PlatformIO, a more developer-friendly option consisting of a cross-platform embedded development environment that supports ESP-IDF.

5. Using FreeRTOS

We decided to run a Real-Time Operating System on our ESP-32 modules to ensure that the timing of our resulting device is as consistent, predictable, and exact as possible. FreeRTOS, being open-source and supported by ESP-IDF, seemed like an ideal solution for our implementation.

Despite the existence of FreeRTOS documentation out there to help us and us having an understanding of Real-Time Operating Systems from lectures, we estimate that getting the hang of FreeRTOS-based development might be a challenge we will have to tackle since we currently only have little practical experience with it.

Technical Approach

In order to develop our project, we decided to modularize it into three modules such as the router module, the mesh networking module and the computer vision module:

- **Router**

This module includes setting up Raspberry pi as a router. The Raspberry Pi requires the installation of OpenWrt which is an open-source OS that will help us build our own router.

1. Download OpenWrt OS from their official website. Make sure to download the correct version that corresponds to your correct Raspberry Pi type.
2. From the Raspberry Pi official website, download the Raspberry Pi Imager. Using the Imager, transfer the OpenWrt image to the Micro SD card
3. Connect the Raspberry Pi to the monitor using an HDMI cable and to the internet using an Ethernet cable. Access the OpenWrt website to finish the configuration to turn the Raspberry Pi into an OpenWrt gateway for Mesh Networking.
    a. Determine the default gateway of your ISP router
    b. Disconnect your WiFi
    c. Insert the Micro SD card with the flashed OpenWrt image into the Raspberry Pi and turn the Raspberry Pi on
    d. Connect the Raspberry Pi to your laptop using an Ethernet LAN cable
    e. Get the IP address of your Raspberry Pi
    f. SSH into your Raspberry Pi using the IP address found in step 3.e
    g. Reconfigure required Raspberry Pi files
    h. Restart the network and unplug the Ethernet LAN cable from your laptop

- **Mesh Networking**

This module includes building a mesh network among ESP32-WROOM-32E modules, ESP32 CAM, and Raspberry Pi, and wiring around them.

1. Program ESP32 and ESP32 CAM so that they can find others and send the necessary information to each other using ESP-WiFi-Mesh protocol and its API. The root node of the mesh network will be automatically selected.
    a. Configure the devices by editing the configuration file
    b. Write code that connects one node to another using TCP server communication
    c. Write code that implements the leader election algorithm to automatically select root node
2. As each ESP32 gets added to the network, check to see if the communication between one another is working properly.
    a. Using testing code provided by ESP-IDF official, test each node to make sure that all nodes are working properly
3. Once the mesh network between ESP32s and ESP32 CAM is established, build circuits that allow ESP32 to get its location and send it as a message to other nodes.
    a. Create a circuit with a buzzer and LEDs attached to each ESP32
    b. Write code to control a buzzer and LEDs depending on data it receives

- **Computer Vision**

This module includes setting up ESP32 CAM, its facial recognition tool, and the wiring around it.

1. Set up the ESP32 CAM

a. Write code to define your ESP-32 CAM model

b. Write code to connect the ESP-32 CAM to the Raspberry Pi's OpenWrt network

c. Program the ESP32 CAM using an FTDI programmer board

d. Get the ESP32 CAM's IP address

e. Start the stream

f. Use the IP address found in step d to access the stream and save faces

2. Add the contact switch to the circuit and update the code: code now needs to check for a face at the switch's signal

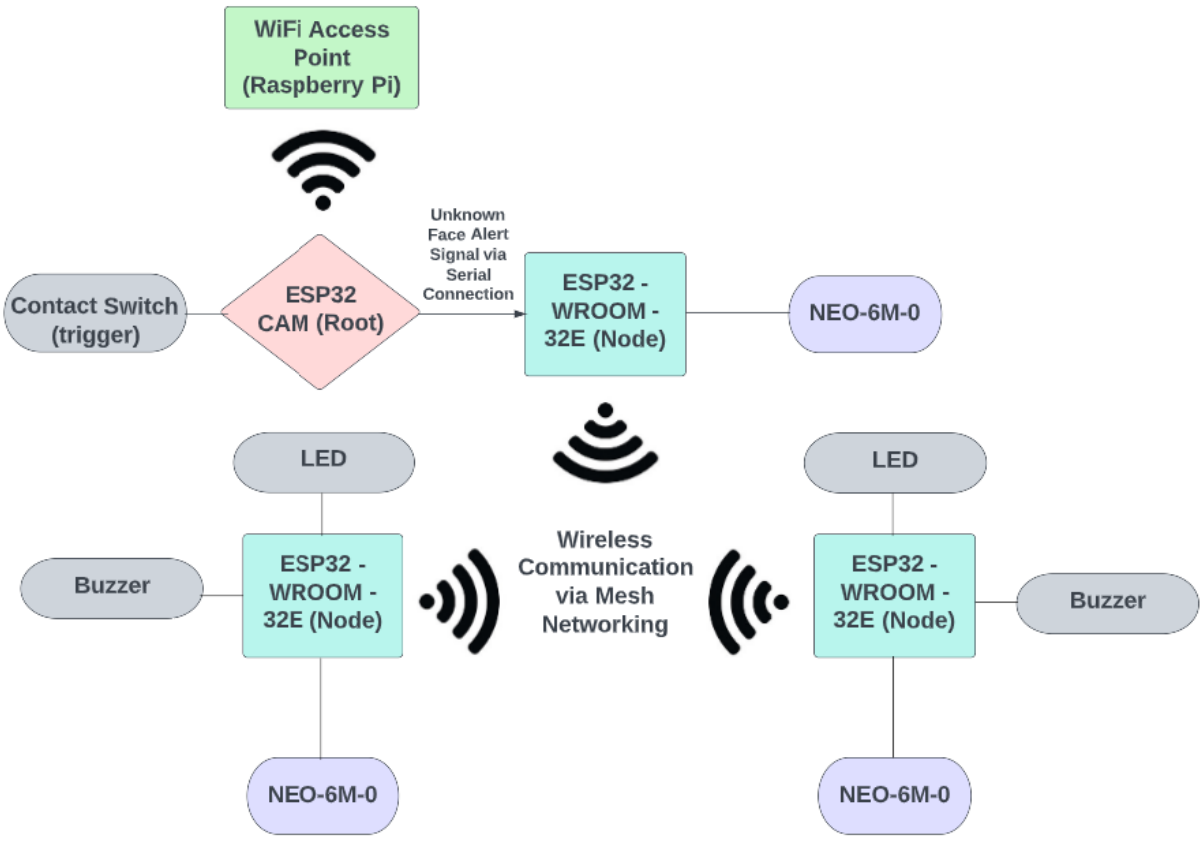

Diagram Showing the Functionality of our Face Recognition Security Alert System

Team Contributions

Each member are in charge of the following modules:

Aya: Router module

Kaede: Computer Vision module

Both: Mesh Network module, Integration of all parts

Milestones

Week 1: Order/find all materials needed for the project

Things that need to be ordered/found are Raspberry Pi, FTDI Programmer Board, Contact Switch, Neo-6M-0 GPS Modules.

Week 2: Set up the Raspberry Pi as a router

Tasks for this week include:

- ☐ Install OpenWrt OS on the Raspberry Pi
- ☐ Set up the configuration of the network on the OpenWrt website

Week 3: Work on mesh network between ESP32s and Raspberry Pi

Tasks for this week include:

- ☐ Write code to turn ESP32 to a node in the mesh network and debug
- ☐ Write code to turn ESP32 CAM to a node in the mesh network and debug

Week 4: Work on mesh network between ESP32s and Raspberry Pi

Tasks for this week include:

- ☐ Write code to turn ESP32 to a node in the mesh network and debug
- ☐ Write code to turn the Raspberry Pi to a node in the mesh network and debug

Week 5: Work on face recognition code for ESP32 CAM

Tasks for this week include:

- ☐ Build a circuit with the ESP32 CAM, FTDI Programmer Board, and Contact Switch for the face recognition module
- ☐ Write code for face recognition and debug

Week 6: Work on face recognition code for ESP32 CAM and integrate all the parts together

Tasks for this week include:

- ☐ Write code for face recognition and debug
- ☐ Integrate face recognition module to mesh network

Week 7: Integrate all the parts together and prepare for the final presentation

Tasks for this week include:

- ☐ Integrate face recognition module to mesh network
- ☐ Prepare final presentation slides and practice