CS684 Project

Tentative Deadline:

Project is divided into two tasks:

Tasks	Release Date	Doubt clearing session	Deadline
Task 1	22/03/2021	25/03/2021	01/04/2021
Task 2	29/03/2021	05/04/2021	08/04/2021

Final Presentation and Demonstration: 12/04/2021 and 15/04/2021

Task 1:

Project has three parts -

1. Hardware side (Firebird V and components)
2. Algorithm side (based on Simulation)
3. IoT side (Web based and GUI development)

Both tasks will cover each part of the project.

Hence Task 1 is divided into two subtasks - Task 1A and Task1B

Task 1A:

Algorithm Side (on Simulator)

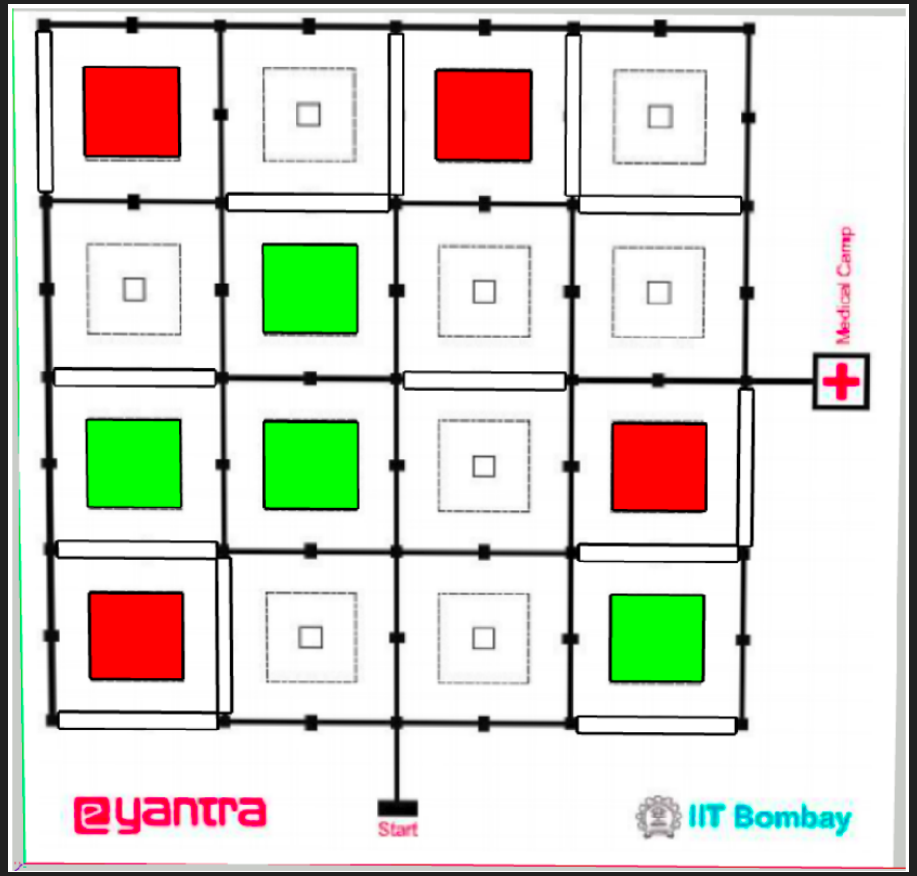
Problem Statement - Scan the entire medical camp and find out the plot locations of injured people.

Output - <Plot No> - <Type of Injury>

E.g. 1 - MajorInjury

Challenges:

1. Line following
2. Color sensor interfacing
3. Debris detection
4. Path planning



Task 1B:

Hardware and IoT side

Problem Statement: Get one request from the server, satisfy the request through robot and acknowledge back to server.

Note: Robot can start from the node of requested plot. It just have to traverse till mid-point marker.

Challenges:

1. Color sensor interfacing
2. End to end communication
3. Interactive GUI development
4. Time keeping on Robot.

Hardware Side (Hints):

1. Color Sensor Interfacing:

https://cs684-iitb.github.io/CS684-Spring-2021/Resources.html

2. Communication between Firebird V and ESP32:

https://drive.google.com/drive/folders/1vac3uNvSwOX17k1qqO0ARTgmKYEigYFd

3. Timer/Counter on Firebird V: (Alternative RTC on ESP32)

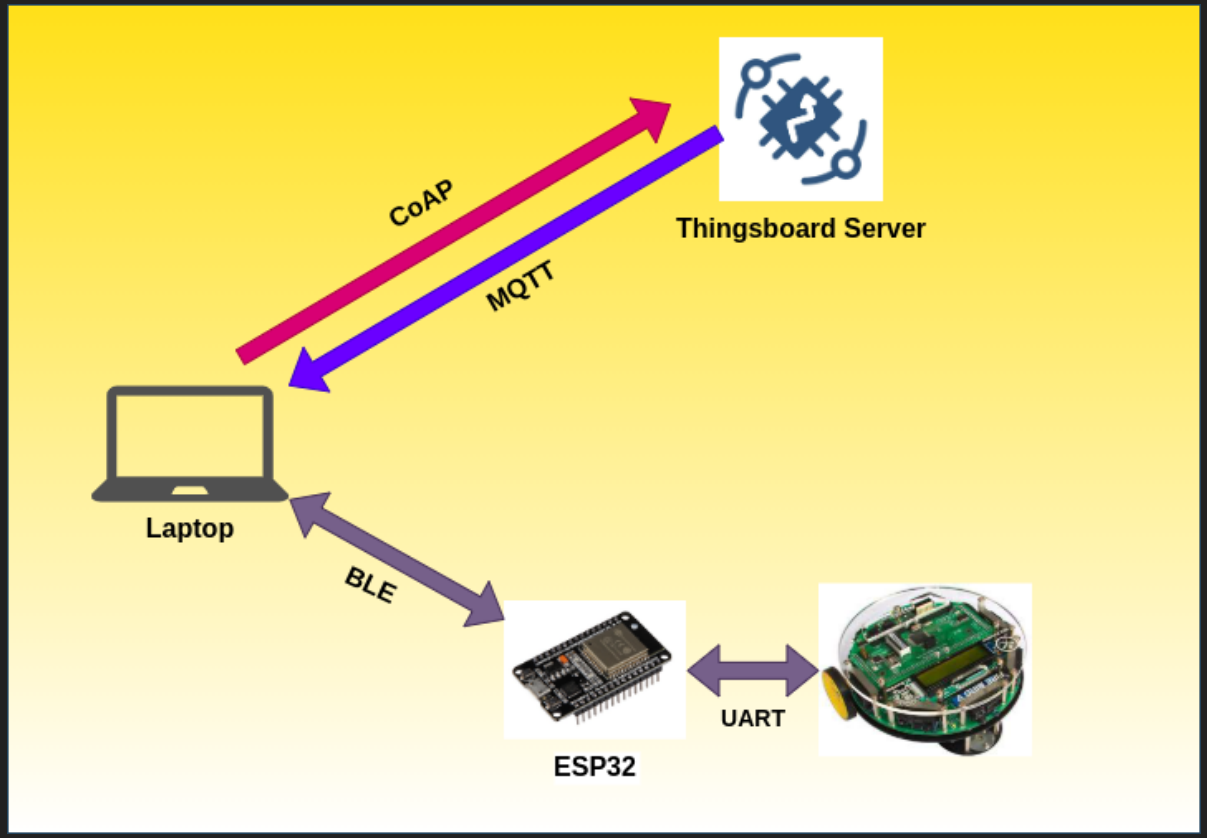
https://drive.google.com/drive/folders/1AMpRDXjxTLKOb-KZuS81j4n-uo7Nea3e

CS684 Final Project

IoT Part

Role of IoT

1. The arena is an abstraction of a disaster-affected area.
2. The robot is tasked with scanning the area (or arena), and satisfying the on-demand requests.
3. These requests (rpc) are sent from the cloud server at an interval of approximately 45 seconds.
4. The requests are intercepted by a laptop and forwarded to the robot.
5. The robot can choose to perform the operations as demanded by the requests or ignore them.
6. If the robot chose to perform the operations, the result should be returned to the laptop and laptop forwards the result to the server.



Overview of Communication

Communication

1. **Thingsboard** to **Laptop** - **MQTT**
2. **Laptop** to **Thingsboard** - **CoAP**
3. **Laptop** to **ESP32** - **BLE**
4. **ESP32** to **Laptop** - **BLE**
5. **ESP32** to **Firebird 5** - **UART**
6. **Firebird 5** to **ESP32** - **UART**

Data Formats

1. **Thingsboard** to **Laptop** - **JSON (as specified in the next slides)**
2. **Laptop** to **Thingsboard** - **JSON (as specified in the next slides)**
3. **Laptop** to **ESP32** - **Any**
4. **ESP32** to **Laptop** - **Any**
5. **ESP32** to **Firebird 5** - **Any**
6. **Firebird 5** to **ESP32** - **Any**

Server Requests

1. Requests are of two types: **Fetch Nearest** and **Scan**.
2. **Fetch Nearest** request demands reaching the nearest specified injury location and beep the buzzer. Convey the result to the thingsboard after **reaching and beeping at the location**.
3. **Scan** request demands scanning a particular **plot**. Convey the result to the thingsboard after **reaching, identifying and beeping at the location**.
4. **The request should be completed in stipulated time also provided in the request (completeIn).**
5. **The result should also contain the time taken by the robot to complete the request.**

Fetch Nearest - Request


```
{
    "method": "fetchNearest",
    "params": {
        "type": "majorInjury",
        "id": 2415498415,
        "serverTime": 1616352000,
        "completeIn": 10
    }
}
```

Fetch Nearest - Result


```
{
    "id": 2415498415,
    "plot": 3,
    "timeTaken": 7
}
```

Scan - Request


```
{
    "method": "scan",
    "params": {
        "plot": 7,
        "id": 65318,
        "serverTime": 1616352045,
        "completeIn": 40
    }
}
```

Scan - Result


```
{
    "id": 65318,
    "type": "minorInjury",
    "timeTaken": 35
}
```

Presentation of Theme Run

1. Develop a nice and self-explanatory UI for the theme run.
2. It can be a web app, a desktop app, or a mobile app (or cross platform).
3. It should show what is happening on the arena and what the robot is doing, for example, the plot it is visiting next, the type of injury scanned, the request it is performing, etc.
4. Use your good judgement to determine how much information to display.

Thank you!