IoT Project: Fulcrum; A Technical Report

Rishap Kumar, Conor King, Megan Lee
rikumar@ucdavis.edu, cfking@ucdavis.edu, mrmlee@ucdavis.edu

I. Why is this an IoT Project ?

Project Fulcrum was an idea that sparked off by the single idea of how we could make an office space more open and flexible to fit in with the needs of employees or users in the system. One example of such a situation is tracking equipment in EEC labs at UC Davis. Suppose a student needs a verification board/hardware boards and there's a limited supply; a Fulcrum tag can be placed on the required asset to track on the administrative side in real time and provide that information to the student/customer via a mobile app. Not only does this work for tracking but can be used to do inventory management and team production freely. To satisfy the sensing needs we are using RFID/NFC readers to track any employees in our system that have entered rooms or work-spaces. We are also using ESP32 and Ultra Wide Band to actively send and receive signals from 2 Anchors and 1 tag to track the desired asset in question. For Computation and automated process of streaming data while maintaining security most of our Computation is done on the AWS server by using technologies like IoT core, AWS Lambda, AWS Amplify, Google Firebase and DynamoDB. We also do Fog Computation on a Raspberry Pi 4; this is used to train an AI to identify the room location of a tag based on its numerical position, which makes finding items more user-friendly. For Communications we primarily rely on WiFi and Mqtt protocols as most of our IoT projects are Wireless and on the cloud. In terms of mobile computing we try to respect the privacy of the user and in this regard most of our computation is done on AWS Amplify back-end and then the data is sent to the Users phone to track the location of items/equipment they need. Our app only takes 30 mb of space to install and is used to purely on receiving information on the cloud by user preference.

II. What are the engineering challenges faced ?

Some of the engineering challenges that we faced was in terms of setting up a cost effective solution for presenting Live tracking data to the app. As you all know the True limitation of cloud projects or IoT projects is data storage. During the entail implementation the plan was to actively store the tracked data and update the data on the mobile app every instance there was new data streamed into the cloud. This method was very ineffective as oftentimes during the testing phase we realized that we did not need to actively track tag sensors in our IoT system but needed their last known location. By using the Engineering design cycle and constant testing/debugging, the decision was made to design a custom API that would be attached to the a button on the mobile app. The buttons function when pressed by the customer in this case is to refresh the current location on the mobile app via a search query of what item they are looking for. This method works far better as the reliability of the tag location depends on when the customer needs it rather than constantly and actively streaming data. Not only did the cost for data search go down tremendously but it also meant that the tag location in this case can be treated like a node. The major advantage to updating the last known location of a node is that we can make efficient data structures like trees or even hashmaps. Which can be used to organize/scale and many tags as the user needs. Another Engineering Limitation that we faced was with Ultra Wide Band. UWB is the technology used to triangulate a tracking tag and two or more anchors. While UWB is great for tracking indoors with accuracy, the limitation we faced is the range. With 2 anchors in set positions the radius of signal reception is between 12 to 15 meters, which is enough to cover 2 large rooms but to cover an entire floor plan or building for tracking there need to be more anchors added too. In our case all of this was made scalable to better suit the tracking needs.

Some of the other challenges faced was making a full stack app to integrate into AWS cloud. To create the app backend rather than using RestAPI, Using AWS Amplify was not only easier but integration was done by Automated process. Then to connect to the app all that was needed was to install the Amplify Package inside the Android studio project.

Our original intention was to host the entire tracking procedure on the AWS server using a DynamoDB database to hold the data and Lambda functions to perform the classifier training and the prediction; however, the machine learning library turned out to be too large to upload to a Lambda function. Therefore, we instead opted for a fog-computing solution, in which the tracking was performed on a Raspberry Pi, which sends the results to the AWS database. Though effective, this solution is somewhat less reliable than a cloud-based system; to improve the system, the AWS's built-in machine learning environment, SageMaker, could be used.

The first machine-learning based tutorial we found used WiFi-based tracking using Arduino Nanos. This would have been a simple and elegant solution, not to mention cheaper; however, that system relied on an environment with many unique ssids, such as residential areas, whereas the intended environments for Fulcrum (schools, offices, large warehouses) might only have one or two detectable WiFi sources. For this reason, the ESP32 board was selected. ESP32 boards produce their own signals (UWB) and do not rely on WiFi.

However, the ESP32 boards carried their own challenges. MQTT is used to communicate the range data to the Raspberry Pi. However, ESP32 boards have a known hardware flaw: once

they disconnect from an MQTT server, they cannot reconnect without a total reset of a board. The code therefore had to be adjusted to prevent the ESP32 boards from disconnecting from the server.