THE UNIVERSITY OF WESTERN AUSTRALIA

CITS 5506
The Internet of Things
Lecture 04

Part 1 -----Project (Finalization, Presentation Guidelines, Tools)
Part 2 ------Components of IoT (continuation)

Dr Atif Mansoor
atif.mansoor@uwa.edu.au

THE UNIVERSITY OF WESTERN AUSTRALIA

Project and Project Presentation

Project

- Think about an IoT product or service
- Is it too basic (not very interesting)
- Is it worthwhile ?
  - Is it solving some problem
  - Is it improving some existing process (time & effort saving)
  - Is it improving some existing solution
  - Is it reducing the cost
  - Is it providing some new service
- If offered, will you yourself will be interested to purchase it or at least use it

Project

- What is the do-ability of the Project
  - Technical Expertise required
  - Infrastructure required / Money Required
  - Availability of Components etc
  - Time required for the project
  - Use a bottom-Up Approach (Start from simple functionalities & add the complexity iteratively)

Project

- Prepare a Framework
  - Conceptual design
  - Various sub-systems
  - How can you check the functionality of the sub-systems individually?

Project

- Prepare a Timeline
  - Understand inter-dependent or independent modules.
  - Independent modules can be developed independently, but keep in mind their interfacing.
  - Interdependent modules can be developed with some assumed/simulated input, keep in mind interfacing.
  - Keep in mind the correct functionalities of the modules.
  - Consider (*strongly)* that the module will not work the very first time, so plan for extra time cushion.

Project

- In case of problem among groups member, discuss among yourselves. Try to resolve, otherwise discuss with Unit Coordinator at the earliest.

- Prepare an approximate Timeline at the start.

- Try to write the project report with the project progressing. A group member can be assigned to work on it from the start. Use various collaboration tools e.g slack etc. Github for software development.

Project

- Expect problems and malfunctions.
  - Be accommodating and supportive.
  - Be mentally prepared.
  - Keep extra time for unforeseen circumstances.
  - Order the items/sensors keeping mind their availability and delivery time (claimed delivery time on websites is not realistic in many cases)
  - Order from local Jaycar and Altronics stores
  - The list of items need to be stated in project proposal document and require prior approval from Unit Coordinator.
  - Order extra critical items.

Term Projects should be decided by 6 pm, Tuesday, 22 August, 2023. Document for writing project titles already shared on MS Teams on First Come First Serve.

Titles will be approved by UC by Thursday 24 August 2023. Duplicate or similar projects will be removed.

Project Proposal Document to be submitted by Thursday, 31 August 2023.

Feedback on Project Proposal by Unit Coordinator by Thursday, 07 September 2023.

Teams at MS Teams

Every group will have private channel to communicate and collaborate on the term project at MS Teams.

Team accountability Document is uploaded in Group Project link at MS Teams

Team accountability Document needs to be uploaded at MS Teams weekly by midnight on Sunday starting week 7 (First one due on midnight of 17 September 2023).

Any dispute or disagreement within Group needs to be reported at the earliest to Unit Coordinator, and not at the end of the semester.

Project Presentation : Guidelines

Organization of Presentation

Presentation should have a clear organization focusing on :

1. What
2. Why
3. How

Do not forget to insert slide number in your presentation

What

Title of Project:

Title ----- attractive and exciting.

At the same time, the title must convey the meaning, the area of work and the goals of the project.

What

Comment on Following Title :

Temperature monitoring for productivity in CSSE building

Suggest few better Title for this:

How : Design

Broad Block Diagram

Flow Diagram

Steps

- What subsystems
- How (Hardware, Software)
- Sequence

How : Presenting the implementation

THE UNIVERSITY OF WESTERN AUSTRALIA

Two ways: First

- Start from Initial design
- Normally the initial design changes during the process, so discuss these changes and reasoning in your final presentation i.e., You are depicting your ability to encounter problems and then handling them.

How : Presenting the implementation

THE UNIVERSITY OF WESTERN AUSTRALIA

Two ways: Second

- Present your final design first, and present how you reach to it through the intermediaries steps.
- You are depicting your ability to encounter problems and then handling them.

Experimentation

- Details of your checking of functionality of the Project
  - Time (How long you evaluated your product)
  - Data (How much data is gathered and analysed for product evaluation)
  - Functionality
    - Does project perform its all aimed functions?
    - Or performs some of it (if so why)
    - Or some functions work but not reliably (if so why)
    - Can you simulate some of the functions if you lack hardware/data
    - Accuracy (Have you compared with Ground Truth)
- Cost of your product and its comparison with market product.

Conclusion : Strengths and Limitation

- **No project can be 100% in the first go. So, what have you found from the project/ experimentation ?**
  - Strengths of your project (may not be wow but something)
  - Limitation of the project (may be many, but it is an achievement if you know them because only then you can improve)
  - Can you compare yours with some existing product?

Aim : You have to demonstrate that you are knowledgeable about your project

Future Work

Optimization:

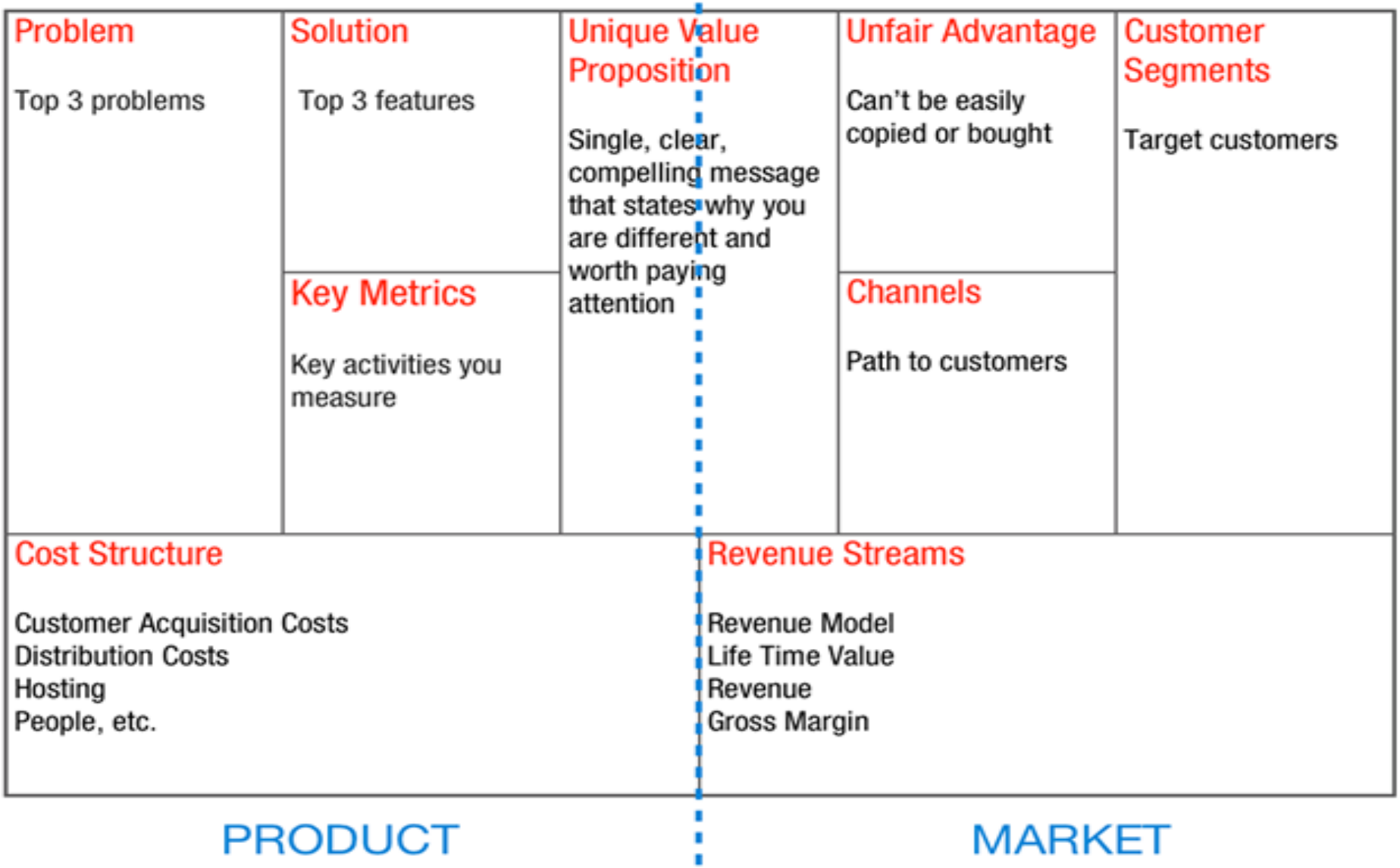
Some could be as following:

- Weakness removal/ Strengthening
- Cost Reduction
- Added Functionalities

Entrepreneurial Journey

The Lean Canvas

- Lean Canvas is a 1-page business plan template created by Ash Maurya that helps you deconstruct your idea into its key assumptions.
- It is adapted from Alex Osterwalder's Business Model Canvas and optimized for Lean Startups.
- It replaces elaborate business plans with a single page business model.



Lean Canvas is adapted from The Business Model Canvas (http://www.businessmodelgeneration.com) and is licensed under the Creative Commons Attribution-Share Alike 3.0 Un-ported License.

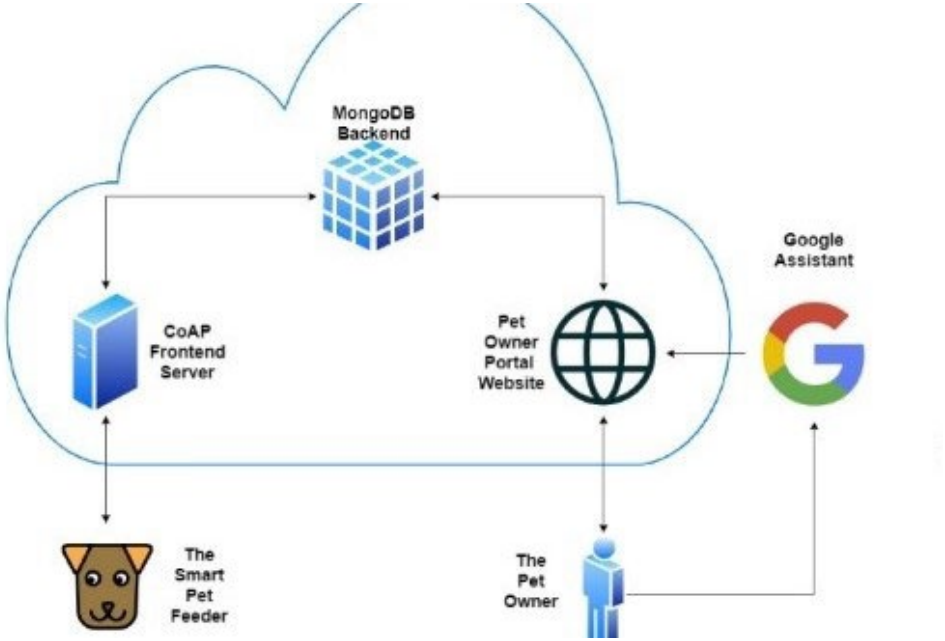
Previous IoT Projects : Tools

Smart Pet Feeder

The system utilises a front-end web interface connected to a **NoSQL database**, namely **MongoDB** for our back-end storage and a back-end server utilising **Constrained Application Protocol (CoAP)** to communicate with a series of hardware pet feeders connected to the internet via Wi-Fi. A single python interface was created using the **pymongo** library to interface with the database by both the CoAP Server and the Web Server.

The system can be hosted on a variety of platforms using virtual machines. It sufficiently automates the pet-feeding process however compromises were made in terms of power consumption and **Google Assistant integration** that would be the basis for future work. **Google Cloud** was selected as the cloud service provider for the project since Google currently offers $300 worth of free credit

Smart Pet Feeder



Smart Pet Feeder

The **HTML5 Up website** template called **Massively** is used. HTML 5 Up templates are fully responsive to the accessing device and are free to use under creative commons license.

The web application framework that was chosen to develop the logic of the website is **Python Flask**. Flask is an easy framework to understand for Python developers and uses Jinja to render html templates for users of the website.

To display the food consumption of people's pets, we used the **Chart.js JavaScript library.** Chart.js is a useful JavaScript library for creating **interactive graphs and plots on websites**. The development website is currently deployed on a Google Cloud Debian virtual machine, and only supports the HTTP protocol.

Smart Pet Feeder

- The Google assistant implementation is done using two main components of the **Google Dialogflow console**, **Intent** and **Fulfillment**.

- Google Dialogflow provides much flexibility for accessing Google Assistant by training it with training phrases to catch any variable names mentioned and provide a customized response. This process is done by the Intent component of Dialogflow.

- The scope of Intent is not just limited to voice-based commands, but also has non-verbal signals that trigger the event. One example of this is, feeding the pet (dropping of food) at a specified time, specified quantity.

- Google Dialogflow intents can be connected to other devices at home/work over Google assistant enabled devices/applications.

How Does CoAP Function?

- CoAP functions as a sort of HTTP for restricted devices, enabling equipment such as sensors or actuators to communicate on the IoT.
- These sensors and actuators are controlled and contribute by passing along their data as part of a system.
- The protocol is designed for reliability in low bandwidth and high congestion through its low power consumption and low network overhead.
- In a network with a lot of congestion or limited connectivity, CoAP can continue to work where TCP-based protocols such as MQTT fail to exchange information and communicate effectively.

Smart Pet Feeder

Another important component of Google Dialogflow is Fulfillment which receives a “POST” webhook call from Google Intent including the phrases spoken into Google assistant.

In the Fulfillment component we design Node.js code and the json packages it uses. The json package file we create bridges the dependencies between the firebase functions and the libraries that we are using for this project. The external Node.js library that we are using is Axios. Axios enables faster, easier and promised based “handshake” between asynchronous HTTP requests to REST endpoints and perform CRUD operations. Here we are using the “GET” request on the webserver to drop food in the bowl. Here we maintain each entry in the log and the logged entries are integrated with google cloud console.

The main challenges of integrating Google Assistant with our application were based mainly around the authentication and security requirements that Google require of external applications.

Smart Pet Feeder

THE UNIVERSITY OF WESTERN AUSTRALIA

Blynk

Blynk is a Platform with iOS and Android apps to control Arduino, Raspberry Pi and the likes over the Internet.

It's a digital dashboard where you can build a graphic interface for your project by simply dragging and dropping widgets.

Eclipse Paho Android Service: The Paho Android Service is an MQTT client library written in Java for developing applications on Android.

MQTT

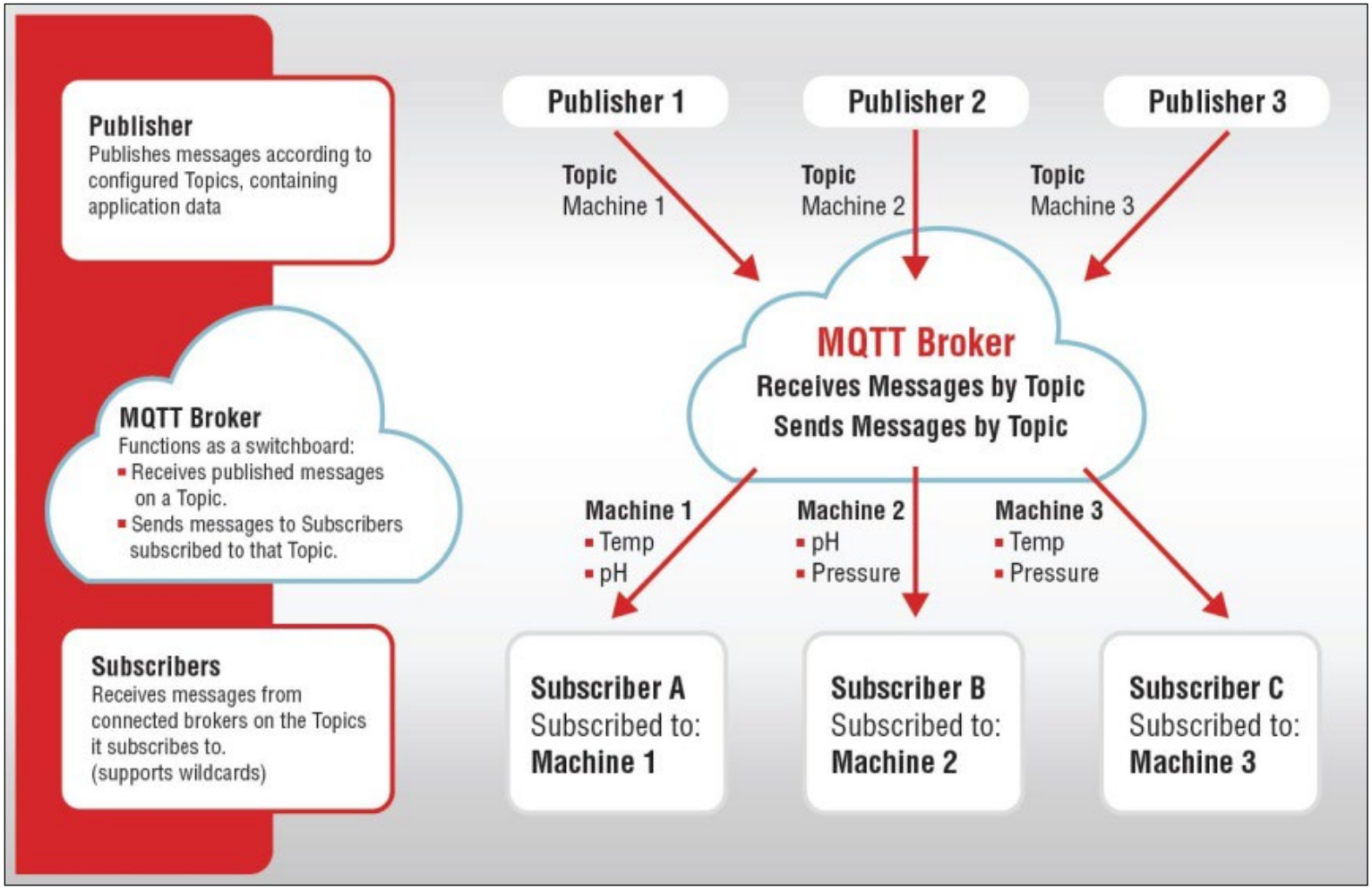
MQTT (Message Queuing Telemetry Transport) is a machine-to-machine connectivity protocol that runs over TCP/IP.

MQTT

Lightweight, simple, MQTT is based on a publish-subscribe structure:

- A Publisher sends messages according to Topics, to specified Brokers.
- A Broker acts as a switchboard, accepting messages from publishers on specified topics, and sending them to subscribers to those Topics.
- A Subscriber receives messages from connected Brokers and specified Topics.

MQTT


Publisher
Publishes messages according to configured Topics, containing application data
MQTT Broker
Functions as a switchboard:
Receives published messages on a Topic.
Sends messages to Subscribers subscribed to that Topic.
Subscribers
Receives messages from connected brokers on the Topics it subscribes to.
(supports wildcards)
Publisher 1
Publisher 2
Publisher 3
Topic Machine 1
Topic Machine 2
Topic Machine 3
MQTT Broker
Receives Messages by Topic
Sends Messages by Topic
Machine 1
Temp
pH
Machine 2
pH
Pressure
Machine 3
Temp
Pressure
Subscriber A
Subscribed to:
Machine 1
Subscriber B
Subscribed to:
Machine 2
Subscriber C
Subscribed to:
Machine 3

Smart Street Light

- The first tier of the operator dashboard communicates with TTN and relies on the TTN (**The Things Network**) Application SDK. Firstly, this tier connects to the TTN application given the application ID and an access key.

- Secondly, a handler is registered to listen for uplink messages sent to TTN by the Smart Street Light System. When this handler hears an uplink message it inserts the payload with a timestamp into a database hosted on the local machine. Presently, this is an **SQLite database** created by the first tier of the operator server using the node-sqlite3 library

Smart Street Light

- The second tier of the dashboard is a web application that displays the information collected from TTN to the system operator.
- This tier was developed using the **Express framework** and its project generator to provide a simple application skeleton

*Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
*Node.js is asynchronous event-driven JavaScript runtime, designed to build scalable network applications

Smart Weather Station

- A website was developed encompassing **Microsoft Azure**, **Django** and **React** to display the real-time data in a graphical representation which is accessible to a user with internet access anytime.
- Django is a Python-based free and open-source web framework. Its primary goal is to ease the creation of complex, database-driven websites.
- The server is a Microsoft Azure Virtual Machine operating with Ubuntu 18.04 LTS and a static IP. The application had two main parts: receiving data sent by a pre-configured device; and displaying data to a frontend webserver.

Smart Weather Station

- Once the server receives information, it is validated to see which device it belongs to and stored in an **SQLite database connected to Django.** This database is queried for relevant results whenever the webserver is loaded for viewing. The web server lists all the devices that it contains in its database. Upon clicking a specific device, a large volume of the most recent entries is returned, along with a graph displaying the data over time.

- It should also be noted that Microsoft Azure was used merely as a convenience, but any webserver (including local) would accomplish a similar feat (e.g. AWS, VMWare and so on).

SMART WATERING SYSTEM

Soil sensors connected to a ESP8266 which handles wireless connectivity over 802.11n Wi-Fi. It runs custom firmware developed in MicroPython which allows ease of development for those familiar with the Python syntax.

ESP8266 is a low-cost Wi-Fi microchip, with built-in TCP/IP networking software, and microcontroller capability, produced by Espressif Systems.

SMART WATERING SYSTEM

- The software component is composed of a database, backend and frontend. The **PostGresSQL** database works in conjunction with the **Phoenix framework** backend to create, read, update and delete real-time data transmitted by the hardware. This data is transmitted over HTTP and interacts with an application programming interface (API).

- Phoenix is a framework for building HTML5 apps, API backends and distributed systems.

- JWT (JSON Web Token) mechanism to verify the owner of some JSON data.

SMART WATERING SYSTEM

- To build the user interface the Vue.js framework was leveraged. The main purpose of the Web UI is to provide a clear and simple-to-use interface to present the live and historic data to the user. The developers ran into issues when it came to organising a suitable file structure for the Flask application.

- In the end, amongst other concerns, it was decided to migrate the application to Phoenix/Elixir due to a simpler, easier language/framework to work with. The dashboard that was developed is a full stack web application composed of three core components; a database, backend and frontend. This is written in PostGres/Pheonix/Vue.js

SMART WATERING SYSTEM

Initially, it was expected that the Python framework Flask would be leveraged for the development of the backend language. However, due to a number of factors including unfamiliarity, unnecessary complexity and limited time it was decided to leverage the powerful language called Elixir.

This is a functional language focused on concurrent programming with a clean, efficient and precise syntax. The Elixir programming language is wrapped in the Phoenix framework for web development and by default uses the PostGresSQL database.

Smart Desk Occupancy

- Sensors are connected to an ESP8266 Node MCU that periodically sends HTTPS Post request to Azure IoT Hub.
- The message is then forwarded automatically to Azure Service Bus which acts as a message queue system.
- The back-end python file checks this message and processes it, then updates the dashboard that displays real-time UI data for users.

Smart Desk Occupancy

The NodeMCU has an integrated Wi-Fi module which connects to a local access point to communicate this information to Azure IoT Hub via HTTPS.

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module.

A **device handler** is created in the backend which is responsible for forwarding messages from the Azure IoT platform using the Azure Service Bus package. It also acts as a gateway to process the information received from multiple devices and allows this information to be used on front-end dashboard for display.

Following this, a real-time dashboard is created through Pusher library on a website to provide visual statistics of the data.

Smart Desk Occupancy

- The dashboard otherwise known as the front-end is run on a Python Flask server. It contains a mixture of HTML, CSS and JavaScript. The front-end communicates with the back-end server using Pusher. Pusher is a library which allows simple, real-time communication. Every time the device handler receives a message, it triggers a Pusher update which is loaded in real-time on the front end.

- The dashboard contains a chart that shows red rectangle for occupied and green for availability, it also contains the number of available vs occupied desks.

Tools used

- In another project, Node Red MQTT at Things Network is used.
- Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.
- In another project, Twilio provides an API for sending messages over Facebook messenger, WhatsApp, SMS or other mediums
- In another Dashboard is created with the help of Tago IO.
    - https://tago.io/

Tools used

- **Flutter UI** software development kit was used as the framework for the mobile application as it is and open-source framework and supports both Android and IOS.

- Also D3.js is used to visualize the data. D3.js is a JavaScript library for producing dynamic, interactive data visualizations in web browsers.

- The method of generating random tokens comes from Cryptojs, and its hash function is used to encrypt the session.

Soil Moisture Project

The system is composed of an Arduino microcontroller, Lora shield, moisture sensor and solenoid. Together, these components enable transmission of soil moisture data to a SQLite3 database on an Amazon AWS EC2 Instance.

Soil Moisture Project

- The Lora shield connects to a local LoRa gateway which provides a built-in **MQTT** broker. The Things Networks automatically publishes the data to the broker through the connection. We used the **SQLite database.** SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.

- The front-end of the website has been written with Flask The moisture data has been visualised with packages Plotly and Dash, the Two python visualisation packages that act as our front end for plotting our soil moisture on the internet.

Smart monitor for solar power

- IoT-based monitor will be constantly monitoring the energy output from the solar panels.
- IoT technology is integrated with various sensors to effectively measure the voltage, light intensity, temperature and humidity.
- The components consist of Arduino board, LoRaWAN shield, built-in voltage sensor, temperature, humidity sensor and light dependent resistor (LDR).

Smart monitor for solar power

- A cloud platform called **Adafruit IO** for the deployment of IoT solutions is used in this project.
- Adafruit cloud is configured to store and visualise the data, allowing it to remain in the system for up to 30 days before needing to be manually offloaded to a physical or cloud-based storage options.
- The reason why Adafruit IO was chosen specifically for this project, was because of its ability to display the visualised data in real-time, as well as having an Application Programming Interface (API) which allows for external tools, including custom scripts, to interact with the data stored on the service.

https://learn.adafruit.com/