# Smart Agro IoT Solutions

**Project Documentation**

##### Digital Labs

##### Sri Lanka Telecom PLC

###### Oshani Jayawardane and Shaveen Fernando

# Introduction

*The Smart Agro project intends to incorporate cutting-edge technologies into the development of a user-friendly agricultural monitoring and control platform for contemporary farmers. The current development's primary goal is to bring the system to the doorstep of the domestic user and promote urban domestic agriculture. The platform promises to produce high-quality output while making gardening an appealing choice for busy city dwellers.*

## How the project aims to meet the requirement:

The ultimate goal of Smart Agro is to develop a safe, streamlined custom dashboard for tracking and managing fields and crops. For optimal output and crop safety, ambient conditions must be maintained at all times, especially those grown in urban environments and greenhouses.

Smart Agro's custom designed sensor modules, properly positioned within the cultivated regions, could detect environmental and soil conditions and give the user real-time feedback via the dashboard.

Actuators may be configured to operate automatically in accordance with timers or other preset conditions. The user also has the option of manually setting their chosen crop-specific parameters through the dashboard. In essence, the dashboard serves as the user's primary interaction with the crop or field.

# Project Outline





Sensors + sensor modules



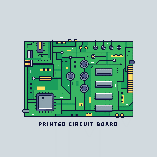
I

Cloud

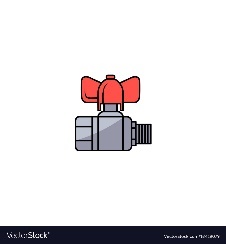
II

Gateway

*Control signals*



Database



End user interface / Dashboard

client

Control circuit

Application layer

Actuators

Middleware / data processing layer

Network layer

Physical Sensing layer

1. Data transmission from sensor module to control circuit (Network Protocols)
   1. Wired
   2. Wireless – RF (Radio Frequency), LoRa, LoRaWAN, Zigbee, Bluetooth, NFC
2. Data transmission from control circuit to end user application / cloud (Data Protocols)
   * MQTT, COAP, AMQP, HTTP

|  |  |
| --- | --- |
| Presentation / Application | MQTT, COAP, AMQP, HTTP, DDS, WebSockets |
| Transport | UDP, TCP |
| Networking | IPv6, IPv4 |
| Data Link / Physical | Wi-fi, LoRa, LoRaWAN, Bluetooth, Zigbee, ZWave, Cellular (GSM, LTE-M, 5G), Ethernet, SigFox, NB-IoT, RF, NFC |

##### Sensors and communication protocols

Commercially accessible sensors could be installed within the farmed fields. IoT development has led to a trend of falling sensor costs. Vendors are also incorporating new technology into sensors to make them more power efficient, dependable, and long-lasting.

The choice of communication protocol used for communication between sensors and the control circuit varies with environmental factors and scale of the farm. Radio Frequency or a wired networks are the preferable solutions for small fields. LoRa or LoRaWAN might be utilized if the field is bigger. Commercial sensor modules could be purchased separately. some vendors offer sensors connected to a LoRaWAN collector/hub itself.

##### Control Circuit and Gateway

It is proposed that the control circuitry be specifically developed. A basic ESP32 microcontroller was adopted for prototyping and testing. The required functionality from the ESP could then be isolated, and a custom PCB could be manufactured using a schematic comparable to the present circuit.

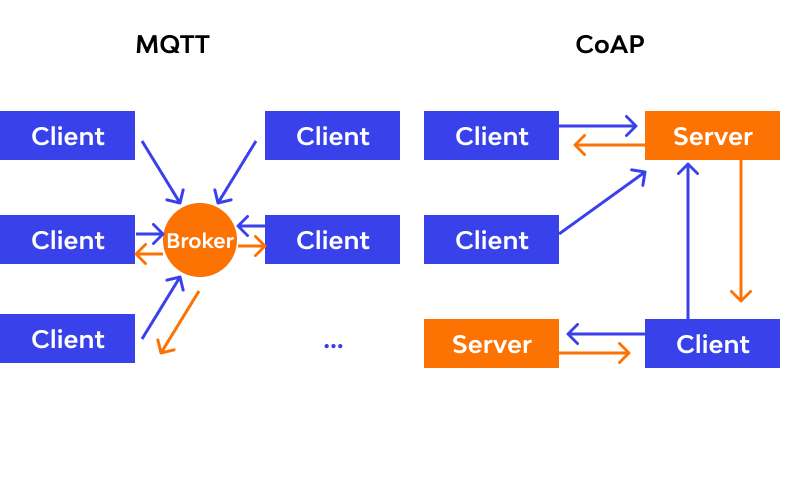
A gateway is a conventional hardware or software application that connects the control circuitry to the Internet (cloud). It uses network protocols such as Ethernet, Wi-Fi / cellular (5G), or satellite to transform data from low-power technologies like Zigbee, Bluetooth, and LoRa into a standard data protocol like MQTT.

##### Control Circuit to Application / Middleware Communication

Out of the data protocols that communicate data from the client to the application, MQTT and COAP seems to be the better option due to multiple reasons. Both protocols were designed specifically to meet the communication challenges presented by low-power, resource-constrained devices operating in loss-prone networks. Since both MQTT and COAP are simpler than HTTP, they have lower latencies and draw less power. HTTP on the other hand is too heavy for IoT. AMQP is most used in server-based analytical environments such as the banking industry and is not suitable for IoT sensor devices with limited memory, due to its heaviness.

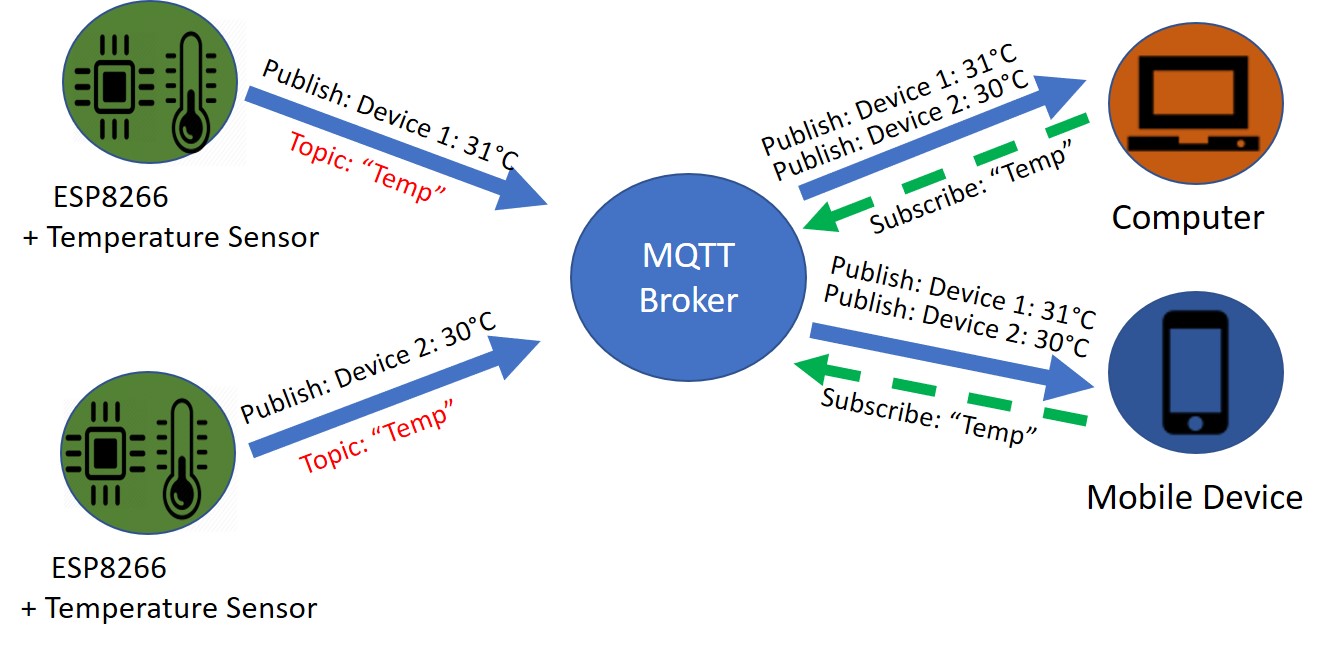
MQTT and COAP also have their own differences.

|  |  |
| --- | --- |
| **MQTT** | **COAP** |
| Publish/subscribe model with central broker, makes MQTT a many-to-many protocol | Request/Response model. Therefore a one-to-one protocol |
| Runs on TCP | Runs on UDP |
| QoS guarantees data delivery | No QoS levels |
| Supports persistent communications | Does not support persistent communications |



Out of these two MQTT appears to be the superior choice for the project since it is lightweight, secure, and easy to implement. It is slowly becoming the standard protocol for IoT, like HTTP is for the web.

###### MQTT Architecture



MQTT is a publish/subscribe model running on TCP/IP. A client can publish to the central broker on a topic and one or many clients can subscribe to that topic to receive data.

There are multiple brokers available commercially, including Mosquitto, HiveMQ, CloudMQTT, EMQX and many more. These brokers can be configured to run on a developer’s local server such as a Raspberry Pi or the service is available on the cloud. Most of these brokers provide their own custom designed dashboards to make it convenient for the user to make connections between clients. Majority of these brokers provide a free, restricted public option for testing. If the developer or customer is satisfied with the service, they may purchase a suitable package the service has to offer.

##### IoT platforms

IoT platforms are third party software that offer services including data visualization and analytics, device management, connectivity management for IoT applications. Amazon Web Services, Azure IoT suite, IBM Watson, Oracle IoT, Google cloud platform are some of the major IoT platforms that can be readily integrated into a project. These platforms can be instantly accessed with minimal integration time and effort. Users have the added benefit of lesser security and consistency issues, alongside a constant support system.

Despite of all the benefits, these cloud solutions cannot be altered or customized. A sound knowledge about the platform and the interface is necessary before using it. However, not everyone has the capability to regularly employ such advance technologies. In contrast, a personalized dashboard frequently caters to a specific type of user and application. They are constructed with built-in functionality for themselves. These dashboards could be designed specifically for daily use of regular non-tech customers. The idea of closing the accessibility and convenience gaps between average farmers and these high-end technologies is one of the driving factors that led to the creation of the Smart Agro Dashboard.

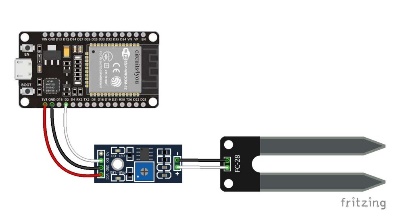
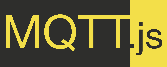
##### End User Application

Smart Agro’s end user application is a custom designed dashboard as mentioned above. A user can register with the system, and their information is saved in the Smart Agro Database. The dashboard then allows users to examine the real-time data from the sensors they have chosen. A dynamic MQTT topic is created in the backend and transmitted as a subscription to the broker when the user chooses the field, plot, and particular sensor. As an added functionality, users can preset conditions to control actuators in their fields manually through the dashboard. MQTT JavaScript client over WebSocket is used to handle these MQTT connections and messages in the browser. MQTT.js and the Paho MQTT client library are the most popular JavaScript client libraries available at the moment.

# Project Timeline

## Expanded Project Timeline

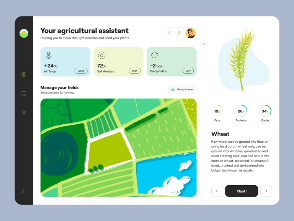
# Stage 01 Prototype

Eclipse Paho | The Eclipse Foundation

**Subscribe to Topic**

**Topic**

**Topic**



**MQTT Broker**

**Subscribing client**

**Publishing client**

* **Publishing Client**
  + **YL69 soil moisture sensor attached to a ESP32 through a wired medium**
  + **IDE used - Arduino**
  + **Language - C**
* **MQTT Broker**
  + **HiveMQ public Broker**
  + **HiveMQ WebSocket cluster**
* **Subscribing Client**
  + **Paho JavaScript Client**
  + **MQTT.js JavaScript Client**
  + **Dashboard design – front end: HTML, CSS, JS using bootstrap framework**
  + **Dashboard design – back end: Node.js with Express.js framework and EJS for templating**
  + **Dashboard design – databases: MongoDB using Mongoose**

## Project Stage Flow

Implemented

1. Displayed sensor data on Arduino serial monitor (Sensor to ESP)

**ADD PHOTO**

1. Connected to HiveMQ MQTT broker and monitored data and connection statuses on HiveMQ cloud public dashboard, Arduino serial monitor (ESP to broker)

Used library – pubsubclient

**ADD PHOTO OF ARDUINO MONIOTR, HIVE DASHBOARD**

1. Connected to HiveMQ MQTT broker and monitored data through NodeRED dashboard (broker to NodeRED)

**ADD PHOTOS YOU SENT ME**

1. Connected to Mosquitto MQTT broker via a Raspberry Pi Local server and monitored data through Raspberry Pi command interface

**I WILL ADD PHOTO**

1. Connected to Mosquitto MQTT broker via a Raspberry Pi Local server and subscribed to published data through Amazon Web Services (AWS) IoT Platform (broker to AWS)

**I WILL ADD PHOTO**

1. Designed Custom dashboard to view data dynamically real time using MQTT JS client

(tried both Paho Js client and Mqtt.js clients separately)

(broker to custom dashboard)

**ADD ONE PHOTO**

Yet to implement

1. Push data into a database and retrieve data back as a history log of sensor information

**I WILL ADD PHOTO**

1. Set controls using dashboard and publish to broker. Then from broker to a different ESP client for controlling part

(mention why controlling this ESP client via dashboard is inconvenient. Mention why direct control is more convenient)

**ADD WHATSAPP DIAGRAM HERE**

# Future Implementations

* + - 1. **MENTION ABOUT CUSTOM PCB**
      2. **SECURITY ENHANCE and battery optimization**

# A guide to Contemporary Farming

**MEKA MAMA LIYANNAM**

Urban contemporary farming produces food inside city limits to bring food production to busy, populated areas. Added benefits of this kind of farming are, community and domestic involvement, optimal utilization of resources, ease of product access and tightening food security.

Even within the broader term “urban contemporary farming” can be categorized into several categories.

1. Urban farming
2. Community gardening
3. Subsistence farming
4. Homesteading

Urban farming

This kind of farming is profit motivated. It’s usually undertaken as a commercial enterprise. However, no big pieces of land are required for this kind of farming. Different modern solutions have been introduced

. This kind of farming is predicted to have high demand in the future due to people becoming more educated about food and agricultural practices. The upward trends in demand for locally-grown organic indicate profitable business opportunities in the sector.