AgriTech: Farm Water Management

Introduction

You'll be helping Petrichor AgriTech, to develop an innovative solution for farm water management.

Maintaining the right balance between water consumption and soil moisture management is crucial to get a good crop yield at efficient costs. We'll develop an automatic sprinkler system based on soil and air parameters, with information coming from embedded sensors.

You are expected to complete at least the basic requirements. You can choose to pick one or more advanced features as well. The exact scope should be defined by your group and then validated by your mentor and the Great Learning team.

Basic Requirements

This is a simple system to manage and actuate sprinklers based on soil temperature and moisture in a large farm. The same setup and technology can be replicated across various farms as needed, so scale is one of the key factors. It should be deployed on AWS Cloud Computing infrastructure, for better data gathering, efficiency, and scale.

Real and simulated soil temperature & moisture sensors need to continuously feed data to the AWS Cloud, via AWS IoT Core. That data then can be streamed and acted upon accordingly.

The system would also fetch air temperature and humidity readings of the farm location, from an open weather API.

The initial soil sensor information can be stored in a cloud database along with unique device ids and their lat/long coordinates. Similarly, information can be stored about the sprinkler locations.

The farm has a predetermined topology when it comes to sensor and sprinkler locations. Each soil sensor comes under the range of a particular sprinkler. This can be directly mapped in the system.

The system needs to continuously monitor the incoming soil and air readings. Based on a reasonable linear difference algorithm that you can define, it should decide whether a particular soil sensor location requires water. If enough soil sensors (based on a predefined percentage) mapped to a sprinkler raise an alarm, the system should send a command to the sprinkler to turn on. Similarly, it should decide when to turn them off.

The features and systems essential for the system to function are:

- Soil sensor and sprinkler simulators
    - There should be at least 20 soil sensors and 5 sprinklers, with each sensor mapped to a particular sprinkler
    - These should be IoT Core Things publishing and receiving information
    - You can run all of them under one process (if you prefer) on a local machine or an EC2 instance
    - (**OPTIONAL**) You are free to add 1, or more, real devices with temperature and moisture sensors if you have them available
- Air temperature and humidity information of a representative lat/long based location of the farm
    - You can use https://openweathermap.org/api or any equivalent api
    - This can be directly sent to your computing solution, via an EC2 instance, local machine, or a Lambda function
- AWS IoT Core to receive all the data, messaging back to the sprinklers, and passing the data further down to streaming and database entities
- Database in cloud to store raw information and decisions
    - You can use DynamoDB (preferably) or managed MongoDB/DocumentDB/MySQL/PostgreSQL (in the cloud)
- Quick turnaround decision making (within 5 minutes)
    - You can look at a streaming and analytics solution like Kinesis along with Lambda
    - Another option can be code in Lambda or EC2 instance, periodically checking the latest readings in the database and acting accordingly
- Ability to display (text or visual) the state of various sensor and sprinkler systems

Advanced Features

- The sensors and actuators could be auto-provisioned using a program which can use AWS IoT Core APIs. You should be able to set up a soil sensor or sprinkler through the command line (or through a UI) to register a thing with a unique name, with appropriate mapping. It should deliver the necessary keys and certificates for the device to connect and transfer data.

- Real-time visual dashboard of the activity. This can show an entity-based or a map-based view of the current sensor states and sprinkler activity
    - You can use an IoT dashboard like Node-red for a simple view.
    - You can also go for a simple but custom frontend app using Python Flask or AWS API Gateway/Lambda to serve the relevant information, in more appealing

formats.

- Long term processing to highlight data based insights of the farm. This can include some of the following, and any other relevant analytics:
    - Areas of low and high water consumption
    - Water consumption trends based on air temperature and humidity
    - Visual representations of farm water usage patterns, with location information (pie-charts, bar-charts, bee-hives)

Evaluation Criteria

Total Project Points: **600**

● Intermediate milestone deliverables	: 50 Points
● Soil sensors and sprinkler simulators - generation and IoT core push	: 100 Points
● Air temperature and humidity data fetch and ingestion	: 50 Points
● Data ingestion and storage	: 100 Points
● Close to real-time computation and sprinkler control	: 150 Points
● Overall architecture and design	: 50 Points
● Advanced feature development	: 50 Points
● Final slide deck presentation and demo	: 50 Points

Intermediate Milestone

An intermediate milestone is to help you keep pace and build things systematically over the course of the capstone project timeline. You are urged to complete and demonstrate the following functionality for the intermediate milestone:

- Identify and set up your main data storage and central processing system
- Build the soil sensor simulators pushing appropriate information to IoT Core
- IoT Core ingesting and passing on the data to your storage/streaming solution
- Data modeling and storage for entities (sensors/sprinklers) and sensor data
- (Optional) Lat-long based weather api data ingestion and storage

Final Deliverables:

- All the code including sensor and actuator (sprinkler) code, data ingestion related code if any, weather api fetch code, the main logic flow of identifying alarm conditions and taking action based on that, configuration settings (textual README and/or screenshots of AWS console screen configurations), etc.
- A simple architecture diagram showing your entire system architecture.
- A small demo screencast video (preferably) or multiple screenshots highlighting interaction with your system and its behaviour.
- A slide deck presentation talking about the process and the output.