

**eGROW Plant Monitoring System**

**Adam Holler** and **Clay McKinley**, with **Dr. Malinowski**

Senior Project 2018-2019 for Electrical Engineering

*Proposal*

Outline

[**Problem Statement**](#_m7afyo3yea5z) **2**

[**Requirements**](#_smgrvxay14uq) **3**

[**Review of Literature/Prior Work**](#_rv8nrqh0je9f) **4**

[**Parts Ordered**](#_v2ievld45x7) **5**

[**Specifications**](#_fpuy7q5u0n7q) **6**

[**Block Diagram**](#_hxdy3l21xk11) **7**

[**Operational Modes**](#_k39771vooxhf) **8**

[**Engineering Efforts**](#_as01s5iwrlt) **9**

[**Timeline/Division of Labor**](#_bgczotr5vw3g) **10**

# Problem Statement

 With technology increasing day by day, home automation is becoming more available and affordable. One area that is growing substantially is the home monitoring system market. Whether it be for burglaries, thermostats, or lighting, there are solutions that are affordable for the average consumer. We plan to add to an area of the market where only industrial versions are readily available, which is plant/greenhouse monitoring. It will be a low-cost solution for cloud-based plant monitoring.

# Requirements

 We will use affordable microcontrollers (such as TI Sensor Tags) to mount at each plant/pot. It will record ambient temperature, board temperature, humidity, sunlight, moisture levels, pressure and also possibly an alert for if the plant tipped over using an accelerometer. If time permits, we may also add the ability to water the plant with a water bottle mount in case the owner is out of town and needs to do a one-time dousing of the plant. The data will be accessible in different formats to the user and we hope to add alarm notifications as well. We plan to use a long-lasting battery, so it should be a one-time setup and hopefully the user does little maintenance to the boards. We also are planning to have the ability to use it both indoors and outdoors. Figure 1 shows how we propose the network will function.

# Review of Literature/Prior Work

 We researched into communication protocols such as Bluetooth Low Energy & ZigBee vs Wi-Fi, 6LoWPAN. In our research, it was clear that ZigBee and Bluetooth Low Energy were going to be our best options due to their low energy consumption. Below is a figure found during our research that shows average power consumption for transmitting and receiving of different communication protocols. We compared and contrasted Bluetooth Low Energy and ZigBee, finding that the former was more energy efficient for smaller wireless networks but ZigBee was more efficient for larger networks due to its mesh network capabilities. Ultimately, we decided to go with ZigBee because its performance was comparable to Bluetooth Low Energy and it allowed us to research and use a protocol that is not widely used in our program.



# Parts Ordered

4x BLUETOOTH SENSOR TAG 296-38831-ND
3x DEBUGGER FOR SENSORTAG 296-42039-ND

1x Networking Modules XBee Gateway ZigBee to Ethernet Intl 888-X2E-Z3C-E1-W

1x Capacitive soil moisture sensor

# Specifications

* *TI Sensor Tag* (CC2650 wireless MCU)
	+ 10 low-power sensors, including ambient light, digital microphone, magnetic sensor, humidity, pressure, accelerometer, gyroscope, magnetometer, object temperature and ambient temperature.
	+ Ultra-low power, coin cell battery, ARM Cortex-M3.
	+ Uses Zigbee or 6LoWPAN.
* *Capacitive Soil Moisture Sensor v1.2*
	+ 5V, Analog readings 2.4-4.4V
* *Digi XBee® Gateway*
	+ Protocols: UDP/TCP, DHCP. Security: SSL tunnels, WEP-40, WEP-104, WPA/WPA2, Authentication with PSK and EAP
	+ OS: Digi Embedded Linux

# Block Diagram



# Operational Modes

Sensors:

The TI Sensor Tags awake to gather data, then they send that data to the ZigBee/Ethernet gateway which then gets ported to the cloud server. Otherwise they are asleep. The Sensors wait a specified amount of time before waking up to transmit again.

Digi Gateway:
 Receive data from sensors via ZigBee, upload to cloud server via Ethernet.

App:
 Read data from cloud to display for user. Pretty straightforward.

# Engineering Efforts

The first thing we tested was the capacitive moisture sensor. It does not have a data sheet so we have hooked up the sensor to the power supply and oscilloscope. It needs a supply of around 5V and the analog output is between 2.4-4.2V.

We initially booted up the TI Sensor Tags we ordered and used the TI Smart Tag App to rename the devices and see how they operate. After reading through a lot of documentation, we have arrived on using Code Compiler Studio. We had some trouble installing the application. We also figured out how to use the TI DevPack and hook up to the computer via Micro USB cable. Using the BLE Device Manager, we got marginal success with connecting to the device. We still are researching how to program the device, with the ultimate goal of configuring GPIO pin for the moisture sensor.

We also have read through documentation for Digi Gateway, it did come with an ethernet cable which is nice.

# Timeline/Division of Labor

* *November/December/January* - Start programming TI Tags and enable firmware for Zigbee
* *February* - Work on Zigbee Networking (and continue programming)
* *March* - Work on App, Alarms, and Cloud
* *April* - Continue development and wrap up work
* *May* - Finish final paper and presentation

*Division of Labor*
Right now, we are still discovering, so we are working together. Once we get a grip on some things (and now we have multiple debuggers) we can start working separately.