Thomas Sciano
Dennon Meiklejohn
Alexander Bailey
CS-370 Fall 2018 Term Project Report-Demeter

Demeter

Introduction

Demeter is an embedded system meant for use in gardens and small-scale agriculture. It uses two microprocessors, several sensors and a dynamic, interactive website, as well as the state-of-the-art MQTT messaging protocol (the same used by Facebook Messenger and Amazon Web Services). Demeter conveniently monitors various states of the garden, acting as an automated notification system, data logger and garden manager.

Demeter can also be applied as an economically efficient and effective strategy for large scale data measurement and control. It is a cheap and reliable platform to monitor your environment and remotely control aspects of your business, life, and the field.

In the mythology of the ancient Greeks, the goddess Demeter was worshiped for her dominion over agriculture, the harvest, and the cycle of life and death. Project Demeter hopes to emulate her power for the modern gardener, using embedded system technology to make the management of plants easier than ever before.

Problem Characterization

The climate, particularly in locations like Colorado, can be unpredictable. Plants may require attention as the weather changes, and this can prove to be a difficult problem for those with busy schedules. Would it not be great to have an extra helping hand to notify you in case your plants need attention, or even automatically take action if you are unable to yourself?

Enter Demeter.

Solution and Implementation Strategy

Demeter went through several iterations of development. We fluctuated between having too many and too little features, and it was difficult to decide how exactly we wanted to implement everything. What follows is our general train of thought that culminated with our final product.

Methodology

What should Demeter do?

The first decision made was over what sort of measurements we wanted Demeter to take. Based on common-sense and research into the available sensors, we decided on four quantities: soil moisture, humidity, light and temperature. All four are important conditions that need to be monitored in order to sustain the health of plants in a garden, and all four sensors could be acquired at a low cost.

Then we thought, instead of only measuring these four quantities, why not do something with the measurements? Implementing a website that displays the information gathered by the sensors makes the system user-friendly. Easy control mechanisms allow the user to turn a water pump on and off as well. It is also very easy to program automatic scripts that turn on the water pump, which would act as an automated watering system.

Why did we use what we did?

Part of the fun of this project was to experiment and use more parts than were specified in the semester project parameters. Instead of just using a Raspberry Pi, we also used the Arduino framework and an ESP32 microprocessor, as well as several sensors and a dynamic website. We had discussed simplifying the design to only use the Raspberry Pi, however certain libraries we used would have to be adapted from Arduino to Raspberry Pi and we deemed this to be too much work for too little benefit (i.e. re-inventing the wheel). Additionally, we also discussed not using MQTT, but ultimately decided to keep using it as it worked well and is actually an up-and-coming protocol used by big-name corporations.

We also wanted this system to be open source, so other people can learn and collaborate to make things better for everyone.

Functionality walk-through

Demeter is comprised of multiple parts, which have been broken down and explained individually below:

Arduino Framework

Arduino is an open-source platform used for building electronic projects. It consists of a programmable circuit board (i.e. a microprocessor) and a piece of software, such as the Arduino IDE (integrated development environment), to upload program code to the circuit board (the ESP32, see below). The code in this environment is written in C.

ESP32

The ESP32 is a system-on-a-chip microprocessor that runs the program code made with the Arduino framework. It handles the sensor data for soil moisture and light, and also controls the submersible water pump (which can act as a kind of sprinkler). Using certain libraries, it also handles the temperature and humidity data and prepares it for transmission to the MQTT broker. It also handles the WiFi needed for the MQTT protocol transmissions (subscription and publishing), sending the MQTT data and logs to the broker over on the Raspberry Pi.

Raspberry Pi

The Raspberry Pi handles the transmissions sent to it by the ESP32, acting as a MQTT "mosquitto" broker. It stores the sensor data, and displays it in one of the tabs of the website. The website is also run off of the Raspberry Pi, which uses a combination of HTML, CSS and Javascript.

MQTT

Short for **M**essage **Q**ueuing **T**elemetry **T**ransport, MQTT is a messaging protocol configured to use "subscriptions" (analogous to GET) and "publishings" (analogous to POST). It requires a special MQTT *message broker* - this broker receives any incoming publishings and distributes and updates any clients that are subscribed to the broker. The Raspberry Pi, as said before, hosts this critical module. Effectively, the Raspberry Pi is the nexus of the system; it is where all the data is received from the ESP32 peripheral, as well as stored and configured for user interaction on the website.

Sensor Peripherals

For the sensor modules we had a plethora of options in today's market.

- Temperature
  - DHT22 User manual and Wiki
- Soil Moisture Sensor
  - DFrobot Manual/Wiki
- Light Sensor
  - APDS User Manual
  - Broadcom Datasheet
  - Sparkfun Avago Datasheet

The reason these sensor modules were used were for the low price and ease of availability in the currently online market. The low materials cost of this project to add more nodes to the system allows for an economically efficient system in measuring light, temperature, humidity, and soil moisture levels.

The DHT22 is a tried and true standard in measuring temperature and humidity, so it is understandable why this sensor, and sensors like it would be relatively cheap.

The soil moisture sensor by DFrobot, is a little more expensive. But other alternatives can be used in regards the hardware used for measuring the soil. In fact, soil moisture can be read reliably from a graphite rod the size of a pencil.

The APDS-9960 developed by Avago, now Broadcom, is a light and gesture sensor. It is an extremely versatile device for being so readily available and cheap.

"The APDS-9960 device features advanced Gesture detection, Proximity detection, Digital Ambient Light Sense (ALS) and Color Sense (RGBC). The slim modular package, L 3.94 x W 2.36 x H 1.35 mm, incorporates an IR LED and factory calibrated LED driver for drop-in compatibility with existing footprints."

- Avago Datasheet on the APDS-9960 Description

All of these sensors are reliable pieces of computer hardware engineering. The integration of the sensors on the ESP32 with the Raspberry Pi allows for a remote sensing capability which promotes even greater node expansion.

This project is a prime example of how to integrate common readily available electronics into an automated computer system, that is IoT-Scaleable. Future prospects with such low cost and powerful devices leads to even greater possibilities only needing just a small task force team to complete.

Website

The website we built on the Raspberry Pi can control our pumps and LEDs through the *VueJS* framework, which is a derivative of *NodeJS*. Using the MQTT messaging protocol, it can send the commands directly to every part of the system, and this gives users a simple interface to interact with the project without needing to understand the back end functionality. In this way, the website acts as an illusionist, similar to an operating system, by masking the more technical aspects of the system with a user-friendly interface. This interface also serves as a good starting point for future development and has potential to scale to many more pump nodes in the near future.

With the use of MQTT we can also develop quickly with modern Ready built iOS and Android apps that act as MQTT Dashboards on your phone or tablet.

We used a simple IoT MQTT app on the android to even display local data in just 5 minutes.

Libraries

Sparkfun APDS-9960 Library:

We used the APDS-9960 sensor, it has libraries for Arduino, but we wanted to use specific pins for the SDA and SCL pins. So we found a modified version made for the ESP8266 that worked out of box.

This solved an issue, for if users wanted to use other pins than the built in SDA and SCL pins on the ESP32, leaving those pins available.

SparkFun APDS-9960 RGB and Gesture Sensor Arduino Library -- Modified for esp8266
https://github.com/sparkfun/APDS-9960_RGB_and_Gesture_Sensor
Using a modified version to allow switching the sda and scl pins to another digital gpio pin
https://github.com/jonn26/SparkFun_APDS-9960_Sensor_Arduino_Library

Adafruit Neopixels Library:

For light control, we originally intended to use the FASTLED Library. But there were some errors with WAIT, in the definitions of one of the libraries we used, so we decided to test and learn the Neopixels library for controlling the WS2812B LED's instead. Which is a good platform for integrating microcontrollers with the ESP32 and other arduino microcontrollers, but it not as powerful as the FastLED library.

#include <Adafruit_NeoPixel.h>
https://github.com/adafruit/Adafruit_NeoPixel
For light control

Arduino PubSubClient Library:

The Pub Sub Library for Arduino- Including ESP32 Support
This library adds integration of MQTT into the ESP32.

ESP32-Wifi library:

We used the Standard Espressif ESP32 Wifi Library:
#include <WiFi.h>
Esp32 Arduino standard wifi library

https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h
The Esp32's library for interfacing with wifi.

PahoMqtt:

A python library for MQTT protocol, to run on the Pi for logging.
https://pypi.org/project/paho-mqtt/
The Eclipse MQTT Protocol

Mosquitto:

An MQTT broker.
Mosquitto:
Acting as the MQTT Broker on the Pi.

DHT22:

For the DHT22- we used Adafruit's DHT Sensor library.
https://github.com/adafruit/DHT-sensor-library/archive/master.zip
And the Unified Sensor library to make it work with the ESP32 better.

Future Functionality

Pumps

Using a meshed network of Raspberry Pi's we will be able to connect any number of pumps concurrently and have them share information amongst one another, which will allow this to scale infinitely from a software perspective. We will only be limited based on how much hardware is available. This capability could give our project much more real-world viability and could potentially be used for actual commercial or personal purposes.

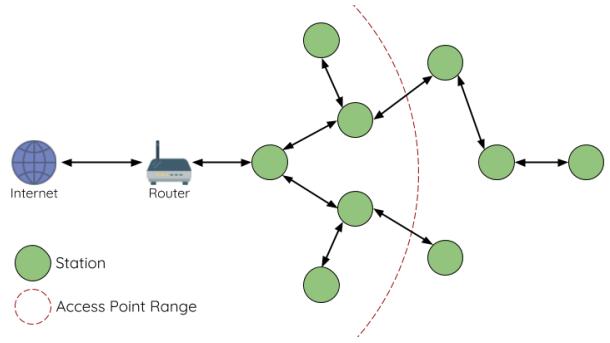
Website

In the near future our website will be able to control and graph the data of multiple pumps at the same time, and will allow for infinite scalability in the near future for as many nodes as we wish to create. Currently the website is very basic and works as intended for personal use in a small amount of nodes, and with a slight refactor we will have the capability to host multiple separate pages through routing and to implement an auth service to allow multiple different users to use this program concurrently with a different subset of pumps and LEDs.

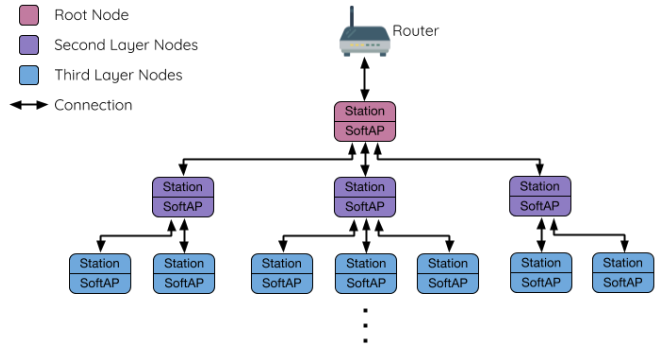
Espressif ESP32- Frameworks

The creators of the ESP32 chip, Espressif, have also designed new frameworks for their microcontrollers.

The ESP-MESH Development Framework:



ESP-MESH Network Architecture



ESP32 Mesh Tree Topology

Espressifs new ESP-MESH framework has allowed many esp32's to create a self healing mesh network. When the parent is lost, a new parent node can be set that has the closest RSSI, to the main access point.

This can also be logged, using the raspberry pi, or straight to Amazon AWS IoT, or Google Cloud IoT.

Recently as of Oct 31,2018, Espressif Announced they have released a Beta version of their ESP-BLE Mesh framework.

The ESP-BLE MESH which works similarly to the ESP-WIFI mesh, but uses bluetooth low energy instead.

In future version, LoraWAN could be used for long range communication.

LoRaWAN

"**LoRaWAN** is a new, private and spread-spectrum modulation technique which allows sending data at extremely low data-rates to extremely long ranges"

- LoRaWAN Technology for Arduino, Waspmote and Raspberry Pi

LoraWAN with this platform could allow for long range, remote sensing of natural resources, parks, and cities. There is also possibilities for this device to work on monitoring the ocean as well.

Such a low cost system, for long range and long lasting monitoring, allows for better data aggregation at a price that is economical for scientific research in the field or at home.

FreeRTOS

The ESP32 can also be Completely freeRTOS for AWS

"Amazon FreeRTOS consists of the following components:

- A microcontroller operating system based on the FreeRTOS kernel
- Amazon FreeRTOS libraries for connectivity, security, and over-the-air (OTA) updates.
- A console that allows you to download a zip file that contains everything you need to get started with Amazon FreeRTOS.
- Over-the-air (OTA) Updates.
- The Amazon FreeRTOS Qualification Program."
    - What Is Amazon FreeRTOS? Source

GPIO infinity

With the effectiveness of this project, and the ease of availability, it would be relatively simple to demonstrate other aspects of Demeter's possibilities. As an MQTT broker, sensor, and output capabilities:

making other projects like temperature controlled environments, power management, and automation on practically anything that needs to be sensed. A PID algorithm could be implemented to create a brewing environment, for an automated brewing system using the MQTT protocol. Being able to control GPIO Pins through MQTT allows systems to be remotely controlled and automated, making any application that requires 120v AC or 5v DC possible as well. Making Demeter an All in One Monitor and Control System.

ESP32 or Raspberry Pi based Soft AP:

The ESP32 can also function as its own Access Point, the same for the raspberry pi depending on if the pi has built in wifi or not.

In the offline version of Demeter, we have tested turning an ESP32 into a SoftAP, and having the raspberry pi auto connect to the ESP32-SoftAP. The Application works flawlessly, and allows for even cheaper deployment.

Github

Demeter

Includes instructions on how to setup your own version of Demeter.

Demeter User Interface

The current rendition of the Demeter User Interface for the web.

Demeter Offline

The offline version which shows the same as the original, but changed IP and Wifi

ESP32-SoftAP

Soft Access Point based on Espressif's Wifi.h esp32.
Allows for a cheaper router based solution if you don't have access to a router.

Conclusions

Demeter has show the options of scalability while being economically sound. A standard user after a few revisions, could make their own systems on the fly. Demeter being Open Source, also allows anyone else to learn and contribute to a greater system.

Making this Project Open Source was a big factor in the decision making process. We wanted anyone to be able to learn from what we had made, as we had learned from others. Collaboration is key to a better world, and making knowledge easily accessible is a big factor in that.

With the ease of availability of all the hardware in this project, it helps everyone who comes after us looking for solutions to their problems.