Project TURF :
Smart Sprinkler System with Real-Time Feedback

Patrick Armstrong, Ryan Williamson, Caleb Edwards

***Abstract*— Watering a lawn is a common activity for almost every house owner. Deciding how long and at what time a lawn should be watered can vary depending on where a lawn is located and what kind of climate is in that location. Project TURF is intended to solve the problem of time and length for watering a lawn through automated systems. By using sensor arrays and a remotely accessible sprinkler controller, Project TURF will automatically suggest when and for what duration a lawn should be watered. A user will have full control via an online user-interface for either creating a custom schedule themselves or using the automated schedule that was derived by the system specifically for them. This report outlines the design and implementation of a Smart Sprinkler system with real-time feedback.**

I. Introduction

A. Motivation

Watering and taking care of a lawn is a task that can be a challenge for home-owners. Yet the commercial market has not completely met the needs of residential owners. There should be a lot of competition in this market with how many people are using sprinkler controllers to water their yards. However, current state-of-the art requires a person with a sprinkler controller to go to their sprinkler box in their garage and manually enter times and duration for each zone in their yard. Few people are confident of how long or what time of day they should even water their yard; it is almost an educated guess for most people with controllers. Based on these times and duration settings that are manually entered, the old controller will turn on and water that zone for the specified time duration. Project TURF's controller is designed to assist the home owner, and does not require educated guesses for programming the controller.

Project TURF is a smart sprinkler controller based on different weather sensors placed throughout a user's yard. There are multiple zones controlled by one valve box and each valve box is controlled by the sprinkler controller. A zone is a subsystem in a sprinkler system that is controlled from a valve box, and there can be many valve boxes depending on the size of the yard and system. Project TURF provides real time feedback to the user to make informed decisions about when and for what duration to water each zone.

This system is based on recommendations sent to the user through a user interface (UI). There will be sensors that collect data throughout the day, and the data will get stored in Project TURF's controller. The controller will have a time series database that records all these values. Based on these results, a predictive algorithm will send a recommendation to the UI stating a duration and time that the user should water that zone. The user then can accept these new recommendations, maintain the default settings, or enter some manual settings based on their preference.

The controller will collect data from a hub station that will represent a weather station, as well as data from other sensors in each zone. This will allow the user to have that extra feedback on their system, viewed through the UI. Each zone of the sprinkler system will have different sensors, including a moisture sensor placed in the dirt of that zone. The data fed to the controller from these results are how the recommendations are determined for unique watering times of each zone. All of the system components and their relationship to one another are shown in detail in Fig. 2.

B. Background

Project TURF is built around open source material that is already available on the web [1]. Project TURF will have a UI that allows any user who has access to view data collected, and make decisions based on the collected data. The difference between Project TURF and other open source projects is that our system provides recommendations. Based on the data that the system collects, a recommendation schedule is created on the UI and put in place. The user will then have the option to change settings in the controller based on this recommended schedule, or leave it as is. Project TURF is about providing more control to the user. This system will not take over and automate everything for the user, unless it is specified that the user wants that.

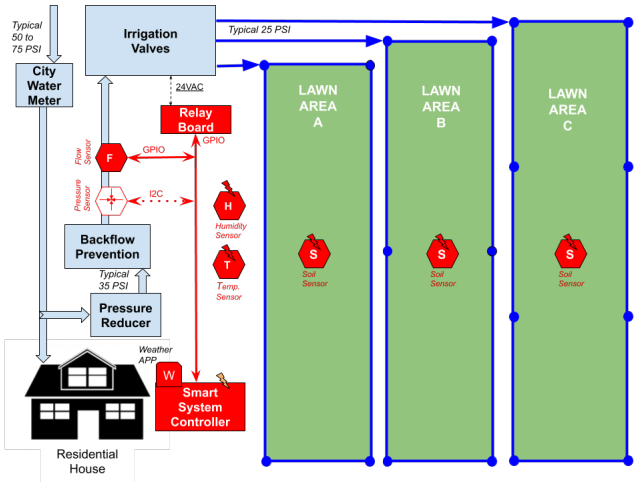

Fig. 1. Layout with added implementations.

As seen in Fig. 1, the design of this project is more complex than standard systems. Sensors will be placed in

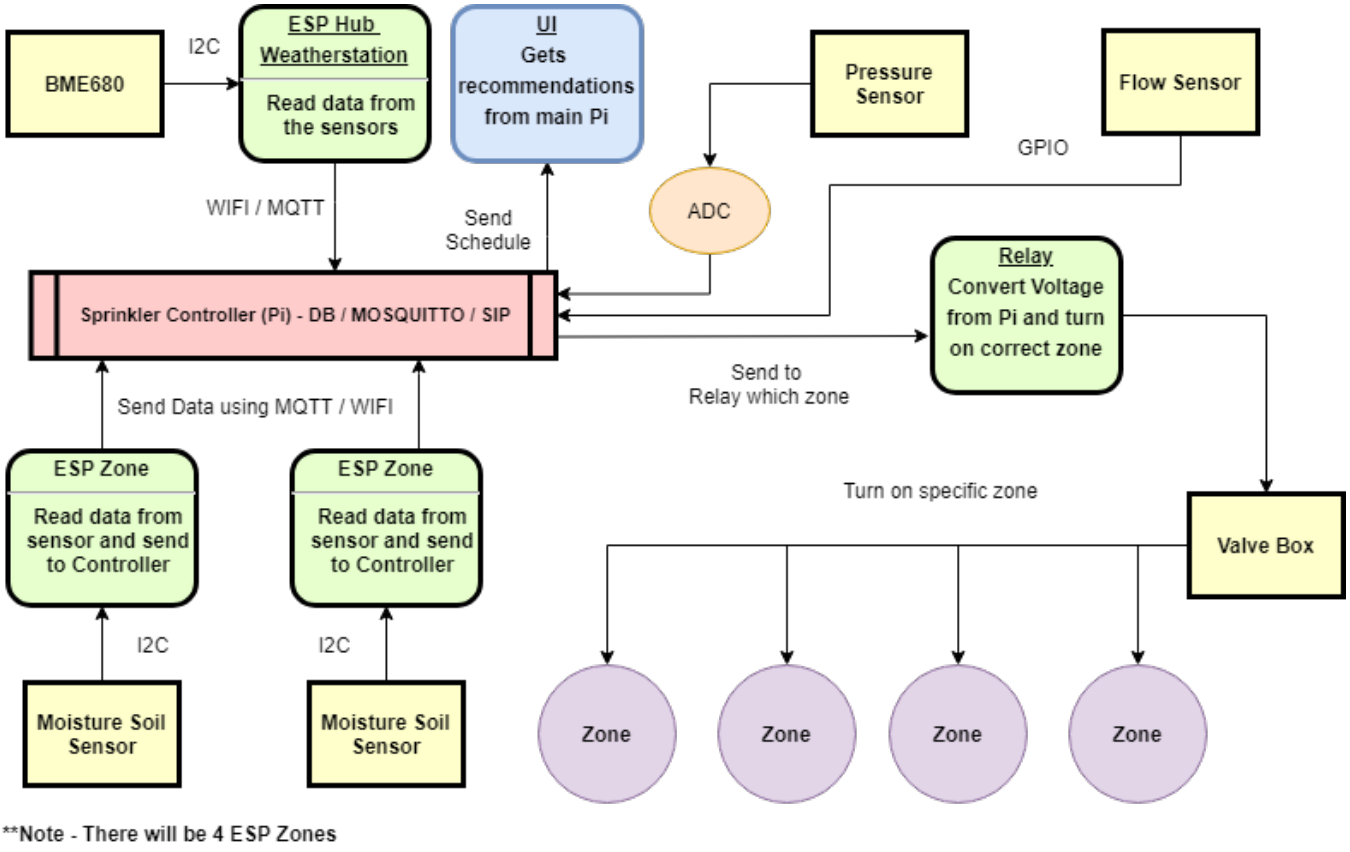

Fig. 2. Block Diagram of Sprinkler Controller

each zone of a user's yard and are expected to send data to the controller, which can be placed anywhere on the premises as long as the WiFi signal extends to each device. Each of the red components in Fig. 1 will be explained in detail in this paper to provide information on why this is a better approach for a sprinkling system.

C. Related Work

OpenSprinkler is an open source sprinkler controller project that uses many of the same elements that Project TURF uses [2]. OpenSprinkler uses sensors to send data to the controller, which then has a user interface to display what is being changed and other data features relevant to the user. OpenSprinkler uses ESP devices to read in data and send that data to the controller. The controller uses WiFi which enables the user to access the UI. The controller developed in Project TURF will have all of these same features, and will use WiFi to allow access to the UI.

The OpenSprinkler controller makes changes automatically for the user; for example if it is raining, it will turn off the sprinklers [2]. In contrast Project TURF will have a forecast prediction that determines the schedule and will create a schedule based on that as well as with other data. If the user decides to use the recommended schedule, they may. Or they may choose to create any custom schedule that is acceptable to them.

Another open source project that Project TURF is incorporating is called the Sustainable Irrigation Platform (SIP) [1]. SIP allows for the creation of custom plugins that can manipulate the base programming and create any schedule within a seven day period. Our controller uses SIP, but creates new custom plugins to add more features like scraping weather data off the web, weather data display, and a relay plugin to turn on our valves.

D. Project Demonstration

Project TURF will be demonstrated with an accelerated timeline. Usually this system makes decisions after collecting data over a 24 hour span. This will not be the case during the demonstration. The poll time of the sensors will increase in frequency to once every minute, instead of once every hour. This data will then be sent to the controller, which will process the data and create a schedule based on the accelerated data timeline.

To simulate the data collection, a box of dirt will be used as one of our zones. We will manually water the dirt to show the change of moisture from the soil sensor.

To show that the relay board works, the relay will be connected to three different valves through which compressed air will flow. The sprinkler heads will pop-up, signaling when a valve has been turned on.

II. Hardware

A. ESP32 Microcontroller

The ESP32 is a low cost microcontroller that is being used to connect to different sensors via I^2C protocol and send data over WiFi. The ESP32s has a 32 bit microprocessor with 520KB of SRAM and 11MB of instruction memory [3]. This is sufficient to do most small computing and is adequate for Project TURF.

Fig. 3. ESP32 with Bluetooth and WiFi capabilities.
[3]

As seen from Fig. 3, the ESP32 is a small controller that can be placed on a PCB and will fit in an irrigation box with ease. It has enough input pins to connect to at least two sensors over I^2C. This will allow the collection of sensor data from any sensor that has I^2C communication. The ESP32 even has the ability to communicate using the SPI protocol if faster or different communication is needed.

The ESP32 can also come equipped with Bluetooth and/or WiFi chips. This allows for wireless communication for the sprinkler controller. The ESP32 can gather data over I^2C, format the data with its processor, and then transmit the data over a wireless protocol. WiFi is the choice of communication for Project TURF.

B. Raspberry Pi Model 3 B+

The Raspberry Pi Model 3 B+ is the main controller of Project TURF. It hosts the database, the Pi Server, and the SIP software. The database stores all the system's weather and sensor data. The Pi Server comprises the Mosquitto MQTT broker an a local weather web scraper. The SIP is the UI upon which Project TURF is built. The Raspberry Pi specific Linux OS–Raspbian–will be installed on the Pi in this project. It is a Linux distribution which was primarily created for the Raspberry Pi.

The main programming language used on the Pi is Python version 3.6, which will be referred to as Python3. Python3 comes installed with the Raspbian OS and will be used throughout the project. To make sure that all future users of this project are up to date with the necessary Python libraries and modules, we are using Python's Package Installer (PIP). PIP provides the ability to capture the current package list used in a *requirements.txt* file. Any device using pip can then install all the required libraries on their Python environment using this text file.

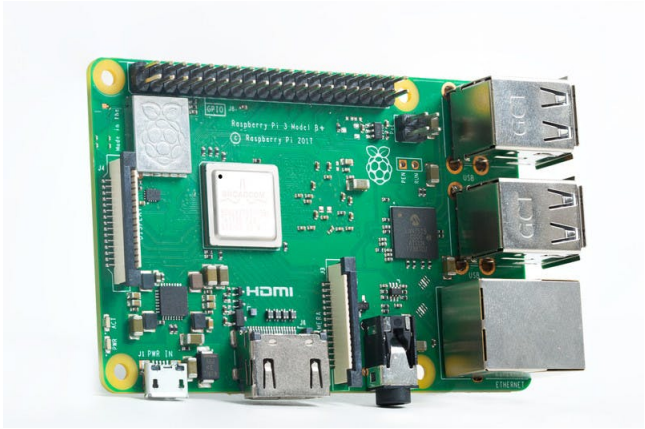

Fig. 4. Raspberry Pi 3 Model B+
[4]

Fig. 4 shows an open and uncased Raspberry Pi Model 3 B+ board. This is to show what the Pi provides as far as specifications for use. When in application use, it will only need to be plugged in through the micro USB port. This will give it power, and upon boot it will initiate all the applications necessary for the controller to run.

The Pi has an HDMI output that can be used for configuring and debugging the device. It has four USB slots to connect a mouse or keyboard. It also has a WiFi chip onboard to allow connection to a network. The memory of the Pi is contained on an external SD card. On this system a 32GB card is used. After the OS and all other installations take place, the memory card has roughly 20GB of memory remaining. This is what the database will use long term. If this proves to be insufficient, a user can use a 64GB SD card or they can switch to storing data to the cloud.

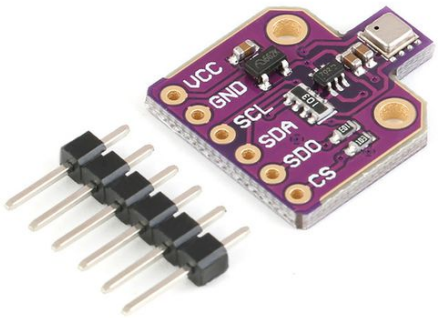

Fig. 5. BME680
[5]

C. BME680 Sensor

The BME680 [5] from Adafruit is a 4-in-1 digital sensor capable of gathering temperature, humidity, pressure, and air quality. The sensor uses the I^2C communication protocol to send/receive data, and it is commonly used for at-home

weather stations and air quality measurements. Project TURF uses the BME680 to detect spikes in humidity and pressure which informs the system that it is currently raining. The system uses local weather data from the BME680 and compares it with scraped weather data from the web to help improve the system's decisions for scheduling stations on rainy days.

Fig. 6. Moisture sensor being tested in an outside environment.

D. I^2C Moisture Soil Sensor

We used a water-proofed Whitebox Labs capacitive 3-in-1 I^2C sensor capable of measuring temperature, light, and moisture [6]. One thing to note is that for the waterproofed version, the light sensor is covered by an adhesive-lined heat shrink and doesn't produce usable data. One moisture sensor will represent an individual zone; see Fig. 1 for an example setup. Both moisture and temperature data will be sent to the system's Raspberry Pi controller via the MQTT protocol. Fig. 6 shows the setup we used for gathering our soil composition (our ideal moisture range for grass). It was placed outside for approximately a month while performing tests such as not watering for 3-4 days and digging up the dirt to see how dry/moist it was. Based on our tests and gathering of data we determined–using online resources–our ideal moisture range to be 400-450, where a value of 300 is extremely dry and 400-450 is wet/damp, with the maximum being 800. Based on this moisture reading our controller will either add/reduce watering time or remove zones from running.

E. Flow Sensor

The flow sensor used in Project TURF is a commercial Digitgen 3/4 inch water flow Hall sensor. The sensor contains a plate with a magnet connected on one side, which rotates when water or air passes across it. When the plate makes a full rotation, the magnet causes a pulse to be sent from the sensor. To determine the volume of water that flowed through the system, Project TURF performs a calculation using three metrics: the counted pulses, the diameter of the sensor, and the amount of time passed. The purpose of the flow sensor is to gauge the accuracy of the amount of water flowing though the TURF system.

F. Pressure Sensor

The main purpose of the FTVOGUE stainless steel pressure transducer sensor is to help monitor the water pressure of the TURF system. Municipal water pressure measures somewhere between 35 to 100 psi (pounds per square inch) and an average sprinkler system requires about 35 psi to operate correctly. The pressure sensor that Project TURF is using is capable of measuring up to 100 psi and outputs a linear analog signal corresponding to the measured pressure values with 5V being 100psi and 0.5V being 0psi. A Raspberry Pi cannot receive analog signals, so the pressure signal has to be converted to an 8-bit digital signal using a PCF8591 analog to digital converter chip that output an I^2C signal. The pressure sensor is located between the municipal water supply and the irrigation valves so that the sensor can monitor the entire system's pressure.

There are two purposes of the pressure sensor in Project TURF. One is to ensure that the pressure of the TURF system maintains a nominal pressure rating during operation. The second purpose is to monitor the system for leaks or breaks. If a leak or break occurs within the system, the pressure reading will be in the range of 0 to 5 psi. The controller stores the data collected by the pressure sensor in the database.

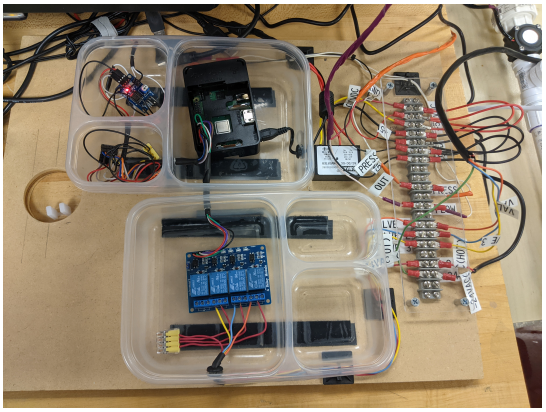

Fig. 7. Raspberry Pi controller connected to relay board.

G. Relay Board and Power

The SainsSmart 4-channel relay board enables the TURF system to turn the sprinkler valves on/off. It is connected to

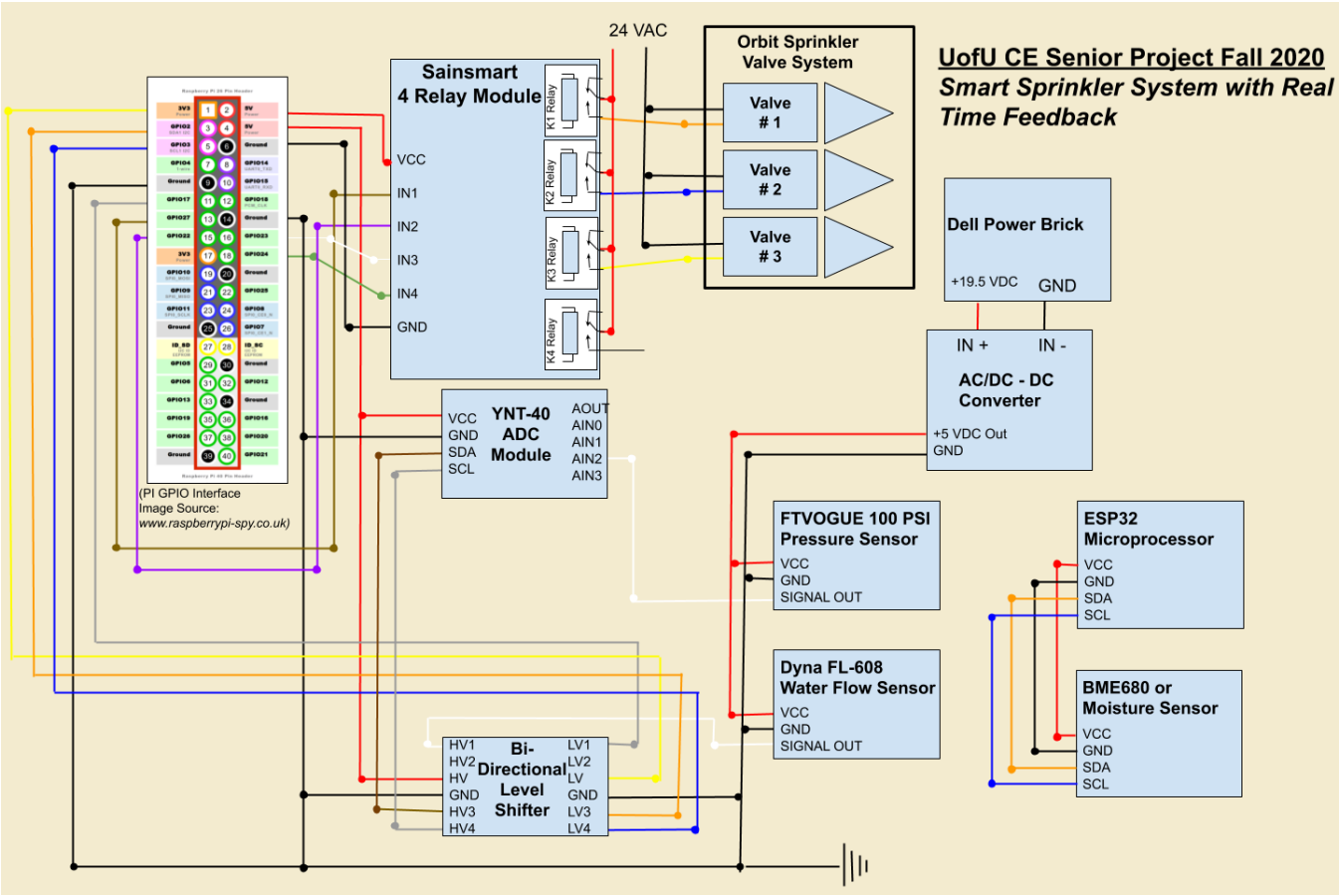

Fig. 8. Wiring Diagram of Sprinkler Controller

the Raspberry Pi using four GPIO pins (Fig. 7). Specifically the pins used by the relays are GPIO pins 27, 22, 23, and 24 as seen in Fig. 8. The pins are in an active low state, meaning the relays will activate if the GPIO signal is low. The relays themselves are connected to a 24V AC power supply on one side, and on the other are connected to the sprinkler valves they will activate.

The wiring of the rest of the TURF system can be seen in Fig. 8. The relay module, AC/DC converter, and the bi-directional level shifter are all driven by the Raspberry Pi. The flow and pressure sensors are driven by a separate power supply due to the current limitations of the Raspberry Pi [7]. All four modules are connected to the Raspberry Pi to their specific input and output GPIO pins. The ESP32 microprocessor is also shown connected to the BME680 or the I^2C Moisture sensor which have identical connections.

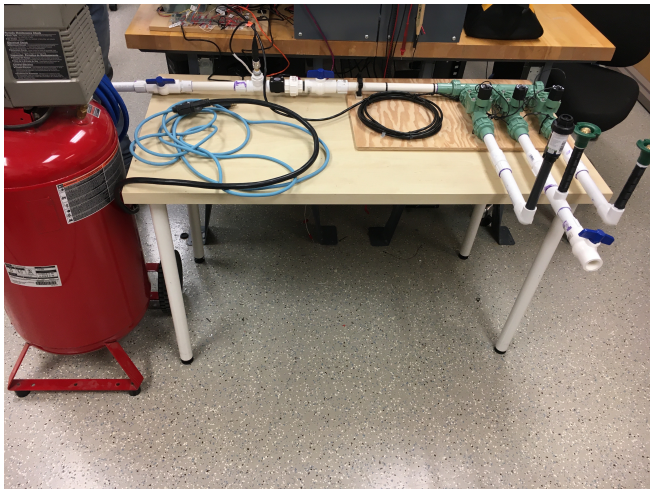

Fig. 9. Demo of sprinkler heads.

H. Sprinkler System

The custom Project TURF sprinkler system that was made for the demonstration (Fig. 9 was built using 3/4 inch PVC pipes and the Orbit 3-valve heavy duty preassembled manifold. The connections were sealed using PVC primer and PVC cement, or Teflon tape (for threaded connections) to ensure a watertight and airtight seal. The sprinkler heads used

in the demo were the Orbit brass nozzle pop-up sprinkler head and the Orbit 2 inch pop-up sprinkler head. The valve manifold is connected to pipe containing the flow and pressure sensors which is then connected to either a spigot for tests with water, or an air compressor for testing with air. From the middle valve, near the end of the run of pipe there is a manual turn valve. This valve, when opened, enables the system to simulate a leak. The system shown in Fig. 9 is also waterproof.

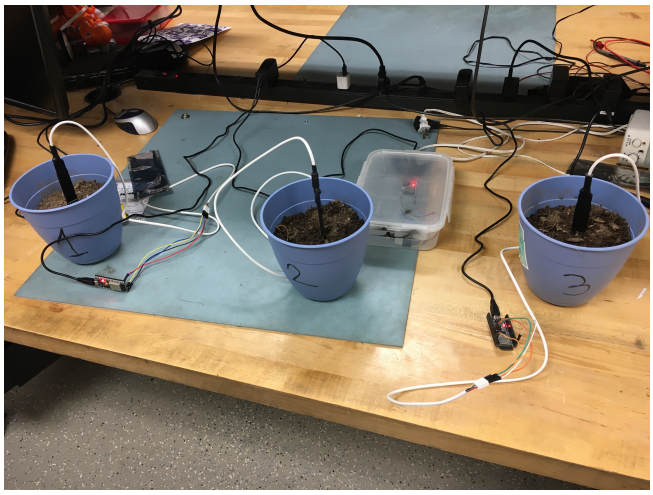

Fig. 10. Demo of 3 zones with moisture soil sensor.

III. Software

A. MicroPython Programming Language

MicroPython is flashed to all ESP32 modules on Project TURF. MicroPython is a lightweight subset of Python, as it contains only a portion of the libraries and packages that Python normally does. As a result, it is much smaller flash size and is ideal for microcontrollers. It only needs 256KB of memory and 16KB of RAM to execute [8].

MicroPython can be accessed on any ESP32 through remote shell (rshell) [9]. This package allows access to the embedded device's Read–eval–print loop (REPL). It also allows access to the file system on the device, as well as read/write access to these files. On the ESP32, the boot.py file and main.py file are executed on boot up in that order. This is how Project TURF was created and debugged.

B. InfluxDB Database

This controller uses InfluxDB, an open-source time series database, to log all the sensor data that we collect for Project TURF [10]. In time series databases data is paired to the time it was entered into the database allowing for easy visualization and access of data using various open source tools. Watering a lawn relies heavily on time, so logging data via this type of database allows for easy analysis of trends based on similar times of day or yearly cycles.

C. Message Queuing Telemetry Transport (MQTT) Protocol

The TURF system employs the MQTT protocol to communicate. This is a wireless communication protocol that is based upon a Topic/Publish protocol developed in 1999 [11]. There is a broker that is installed on the Raspberry Pi and will be hosted and always available on this Pi. The broker selected for Project TURF is an open-source broker called Mosquitto [12].

All Publish/Subscribe messages route through this broker. The ESP32 microcontrollers in this system will publish their data, which has already been formatted, to a certain topic. The Pi that is hosting the broker will also be running another server that is subscribing to these topics that are being published by the ESP32s. This server then takes this data and adds it to the database.

D. SIP

Sustainable Irrigation Platform (SIP), is the software interface on which Project TURF is built [1]. It is an open-source irrigation control software that is written in Python and JavaScript and runs under the Linux OS. This open-source project allows for contribution via plugins. Project TURF will be built by adding custom plugins and manipulating the main source code in a way to allow for database access, creating new schedules, and allowing a user to view data in an easy and informative manner. We chose SIP because we wanted to move away from the traditional sprinkler control boards which are hard to use and offer little to no feedback about a system's watering habits. This allows a user to access the interface from a desktop or phone anywhere in their home. The homepage can be seen in Fig. 11.

Fig. 11. SIP Home Page

1) Data Display Plugin: The data display plugin provides an interface to view local weather, BME680 station reading, and moisture/temperature for each zone setup. The weather is scraped locally from Google and provides five days worth of information (Fig. 12). As for the moisture, a user can view up to 5000 of the most recent readings for each zone. The table is built using a JavaScript function called DataTables which allows a user to filter and search data. By selecting the zone in the drop down, the table is updated using JavaScript and a post method. New data can be accessed every hour. A refresh of the page is needed to see the weather and BME680 reading, but the table for each zone will update for every zone change.

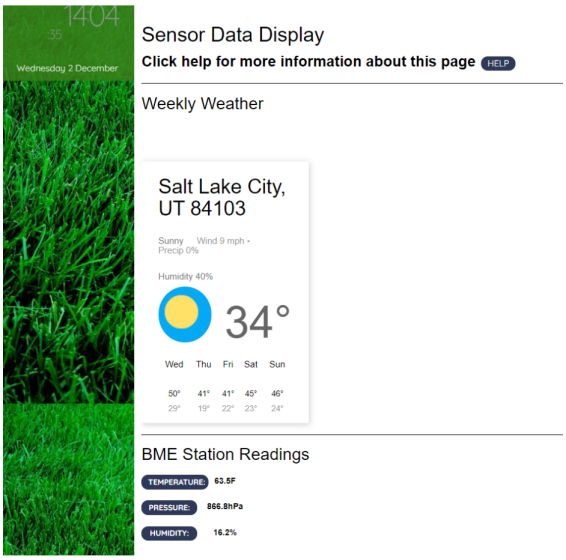

Fig. 12. SIP Data Display Page

2) Controller Plugin: The controller plugin is where all of the times and adjustments for sprinkler programs are calculated. The controller page requires the user to input data for each zone to create a baseline watering time duration. The data required to calculate a baseline per zone is area (ft^2), inches per week of water, the sprinkler head type with corresponding gallons per minute (GPM), and the number of sprinkler heads as seen in Fig. 13. Each day the plugin calculates a precipitation rate (PR) for every zone which can then be used to calculate a base time duration. To find the precipitation rate we used this equation [13]:

$$PR(in/hr) = \frac{96.25 * GPM}{Area(ft^2)}$$

To calculate the baseline time duration for each zone divide the amount of inches per week of water by the precipitation rate. The time is then divided by 3, the number of times that will be watered per week.

$$Time(min/day) = \frac{water(in) * 60}{PR * 3(days/wk)}$$

In Utah you can obtain the amount of water per week you should give a lawn by going to the Division of Water Resources website: https://conservewater.utah.gov/guide.html

Once the baseline time duration is calculated the controller then needs to adjust the time amount by the soil moisture levels in each zone. The controller queries the database for the soil moisture levels in each zone and checks if they are in the target range. As stated previously the target soil moisture level is 400-450. If the moisture level is above 450, it removes a minute of time per 25 points above the target value. Similarly if the moisture level is below 400, it adds a minute of time per 25 points below the target value. The resulting time is what will be used for that day's watering program.

The controller plugin also runs the code to capture both the flow and pressure sensor data. SIP uses a Python library called blinker; specifically SIP uses blinker to signal when a sprinkler program will be run. The controller plugin uses this blinker signal to capture the amount of time the water is flowing through the system. It takes the pulse count from the flow sensor over that time frame and stores the gallons per minute (GPM) of the specific run. The controller also records and stores the water pressure value after a few seconds have passed to get an accurate reading while the system is running.

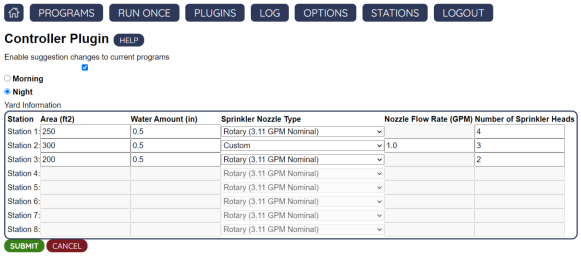

Fig. 13. SIP Controller Page

3) Relay Plugin: The relay plugin enables the SIP to control the sprinkler valves within the TURF system. By using the same blinker signal that the controller plugin uses, the relay plugin enables and disables separate signals to the relay board using the Raspberry Pi's GPIO pins. The pins may be set by the user, Fig. 14, but we recommend leaving them set at the default values due to the Pi's launch behavior. Not using the default values may result in multiple relays turning on simultaneously due to the default behavior of the Raspbian OS.

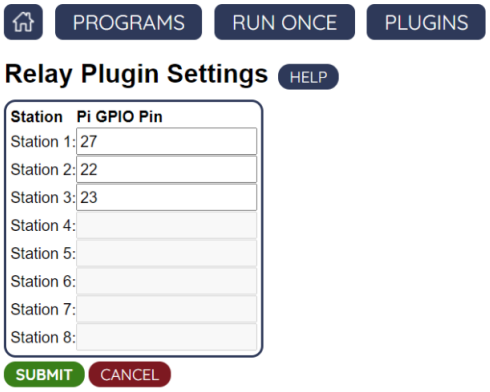

Fig. 14. SIP Relay Page

IV. Cost

The cost of this project is straightforward when dealing with the required components, but more variable when dealing with setup and overall design/protection of the system. The system currently uses a homemade protection case for the Raspberry Pi that would be difficult to predict production cost. The ESP32s are connected by wires to the sensors in use. For distribution/production, they could all be placed on a PCB which would increase cost by approximately ten dollars. There is also a variable cost corresponding to the number of zones in a yard. The base estimate/cost as of this report is 45

dollars per zone. Some of the variable costs when calculating a cost for this project are listed in Table I.

TABLE I
LIST OF ITEMS WITH VENDOR LINKS AND COST

Component	Vendor	Cost	#
BME680	Adafruit	$19.00	1
Raspberry Pi 3 B+	Adafruit	$65.00	1
ESP32	Espressif	$15.00	# of Zones
Relay Board	Sain Smart	$9.00	1
I^2C Moisture Soil Sensor	WhiteBoxLabs	$29.00	# of Zones
Pressure Sensor	Amazon	$25.00	1
Flow Sensor	Amazon	$12.00	1

As seen from Table I, the material cost for the main components of the TURF system would cost approximately 130 dollars. This does not include production, software, taxes, or shipping. Project TURF is slightly more expensive when dealing with available commercial controllers, but most commercial controllers do not come with the sensors as this project does. These sensors allow for data collection and manipulation to provide an accurate watering system customized for the user's yard. Specifically, it is designed to provide an accurate schedule for each of the user's zones in the system. A user no longer needs to worry about the stress of over or under watering.

V. DIFFICULTIES

There were a few difficulties that arose with the development of this project. This included the hardware and the software as well as the ability to effectively test. The trouble with the hardware included the power required to connect and operate the flow and pressure sensors to the Raspberry Pi over long wiring distances. The software problems that occurred were related to the algorithm development for the controller and the version of MicroPython that was flashed to the ESP32. The details of these difficulties are described in detail in the sections below.

A. Power

The maximum recommended current draw for a single Raspberry Pi output pin is 16mA and an aggregate current of 50mA for a bank of pins [7]. The amount of current required to run the relay board, flow sensor, pressure sensor, AC/DC converter, and the bi-directional level shifter could exceed the 50mA maximum of the Raspberry Pi. To solve the current draw issue and potential long wiring distance, we added another separate power supply to drive the flow and pressure sensors. The power supply we used was an old Dell laptop power supply that output 19.5V DC which was then down-converted to 5V using a AC/DC TO DC Buck Power Converter. Project TURF had no issues with adding the new power supply and the sensors operated as expected.

B. Testing

In order to effectively test and verify the correctness of our interface we needed to be able to setup our finished project in a real yard. We were only able to test outside once, as seen in Fig. 15, mostly due to inclement weather conditions. So many unknown factors come into consideration when our Controller is making a decision to add/reduce time or stop scheduled watering times all together. Due to these constraints we were unable to dynamically test our controller's ability to accurately adjust/predict the best amount of watering time for each zone. With any good interface, it is never complete and can always be improved; but the best way of accomplishing such is by testing over a longer time period.

Fig. 15. Custom made Sprinkler Array

C. Algorithm

The original plan for Project TURF was to utilize a machine learning algorithm. This algorithm would have taken in all of our sensor data and reported an output value indicating how long to water a particular zone. This algorithm was a supervised, linear regression machine learning model. This means that the model had to have initial data with correct output values to work. Since we did not have adequate initial testing data, the model could not be completed without giving false training data to the model.

The machine learning model was still created using emulated data and resides in the project's repository. The machine learning algorithm can be used if the user collects data for a season or two and has a correct schedule for that time frame. Once the data is gathered, the user can use that as training data for the model. The model will then be able to take all future data from the sensors and generate a watering time for that zone to which the data is associated. For now, the controller takes in only a portion of the sensor data and generates a schedule based on that value.

D. MicroPython

Micropython is an open-source implementation of Python 3 [8] that includes most of the standard libraries and is optimized to run on micro-controllers. Thus, all of the supporting libraries for sensors that are written in Micropython will not operate on newer versions of Micropython, unless the owners update their repositories with appropriate changes. This was a good learning experience, because we spent a couple weeks

trying to figure out bugs; we even tried switching to Arduino IDE. In the end, all we had to do was use an older version of Micropython that supported the sensors library design.

VI. Future

A. Machine Learning

Machine learning is not currently implemented in Project TURF, due to the insufficient initial data. One possible solution to using machine learning would be to create a supervised linear regression model. This would be a multiple regression model, as we have multiple inputs to one output [14].

Since all of the sensor data is stored in the database, we have training data for our model. A user would collect data for a season or two and gather enough data to create a model. This is also dependent on how well the output fit. This means that the amount of time each zone was watered should be fairly accurate, or how the user wants it to be. This will give the data the correct output for each input to create a valid model.

Once the training data is used to create the model, all future data can be put into this model to get an output of how long to water the zone. This would be the new way in creating the schedule for each zone. All a user needs is some training data, and a linear regression model can be created.

B. Custom PCB and Casings

This project could be made for professional production with a PCB and with a custom case for the controller and the ESPs. The ESP32 could be placed on a PCB and the sensors could connect to the PCB instead of directly to the ESP32. This would give the system a cleaner, more professional look.

A case could be made for the ESP32 and the Raspberry Pi. Standard sprinkler controller casings can be found on Amazon for around $30.00 and would fit the sizing of a Raspberry Pi well [15]. On the other hand, the ESP32 would require a custom casing. In this project, plastic food storage container is used with with holes cut into it using gaskets to cover the holes to seal it. These could be more developed with more research.

C. Mesh Network

A mesh network would be very useful in the case of a sprinkler controller. This would allow nearly infinite zones with nearly infinite distance to the controller. This is possible in a mesh network, because all the nodes in the network could talk to each other. When one zone gets the data, it would then send it to the next closest node, which would then pass it to the next closest node until it reaches the server or database that stores all the data.

Python has libraries for Bluetooth Low Energy mesh networks and WiFi mesh networks from PyPI [16]. These are relatively new libraries and are made for IOT devices. This would be on the scale of newer territory for the ESP32s and any implementation would be extremely valuable.

D. Battery Powered

Each ESP32 in this project can either be powered by battery or by wire. There are pros and cons to each choice, and there seems to be no right solution for this controller. Hard wiring each zone would mean that there would be no maintenance when dealing with the power supply of the ESP32. The problem in this case is that a power source would need to be developed for each zone.

A Battery-powered microcontroller is more dynamic and allows the user to place the zone anywhere or place in any pre-existing setup. If a sprinkler system is already setup, then the user only needs to place the battery-powered ESP32 with the existing valve box. The issue with the battery-power is that regular maintenance is required (batteries are required to be changed). One fix to this would be some self-maintaining source, such as a solar panel.

VII. Conclusion

There are many people with brown, dead yards. Their grass does not get enough water, or it gets too much. Sprinklers get left running and water is wasted. Project TURF updates sprinkler control to modern state of the art technology. Project TURF automates and predicts the best time of day to water your yard, as well as determine and predict the length of time your yard should be watered. It displays to the UI the schedule that has been determined and lets the user decide if these recommendations seem valid. This system assists the user, without taking the control away from the user.

Project TURF will provide users a way to save money on water, as well as do less work in managing their yard. Real-time feedback is implemented, so no waiting on data is necessary. Data will always be available to the user on the UI. The difference between the Project TURF controller and other smart sprinkler controllers is that Project TURF provides schedules to the user that are based on data coming from the user's yard, which the user can implement as they see necessary.

At the end of the day, Project TURF should make lawn care easier. It is straightforward and easy to maintain. The sensors send data to the ESP32, and the ESP32 transfers data to the Raspberry Pi for processing. After data has been processed, a schedule is sent to the UI that the user can choose to accept or deny. The lawn will be more green and healthy by the end of the experience. Lawn care should be made easy.

References

[1] Dan-in-CA. (2020) Sustainable Irrigation Platform. [Online]. Available: https://dan-in-ca.github.io/SIP/

[2] Ray Wang, Samer Albahra. (2020) Open Sprinkler. [Online]. Available: https://opensprinkler.com/

[3] ESPRESSIF SYSTEMS. (2019) ESP32 Overview | Espressif Systems. [Online]. Available: https://www.espressif.com/en/products/hardware/esp32/overview

[4] Raspberry Pi Foundation. (2020) Buy a Raspberry Pi 1 Model B+ – Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-1-model-b-plus/

[5] randomnerdtutorials. (2020) MicroPython: BME680 with ESP32/ESP8266. [Online]. Available: https://randomnerdtutorials.com/micropython-bme680-esp32-esp8266/
[6] Catnip I2c Soil Moisture. (2020) I2C Soil moisture sensor by Catnip electronics on Tindie. [Online]. Available: https://www.tindie.com/products/miceuz/i2c-soil-moisture-sensor/
[7] RaspberryPi. (2020) GPIO - Raspberry Pi Documentation. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md
[8] MicroPython. (2020) Micropython. [Online]. Available: https://micropython.org/download/#esp32
[9] PyPI. (2020) rshell. [Online]. Available: https://pypi.org/project/rshell/
[10] InfluxData. (2020) InfluxDB Open Source Time Series Database | InfluxDB | InfluxData. [Online]. Available: https://www.influxdata.com/products/influxdb-overview/
[11] MQTT. (2020) MQTT. [Online]. Available: https://mqtt.org/
[12] Mosquitto. (2020) Eclipse Mosquitto: An open source MQTT broker. [Online]. Available: https://mosquitto.org/
[13] W. S. University. (2020) Sprinkler Application Rate. [Online]. Available: http://irrigation.wsu.edu/Content/Calculators/Sprinkler/Sprinkler-Application-Rate.php
[14] R. Python. (2020) Linear Regression in Python. [Online]. Available: https://realpython.com/linear-regression-in-python/
[15] Amazon. (2020) Amazon sprinkler casing. [Online]. Available: https://www.amazon.com/Orbit-57095-Weather-Resistant-Outdoor-Mounted-Controller/dp/B000VYGMF2/
[16] PyPI. (2020) BLE Mesh. [Online]. Available: https://pypi.org/project/bluetooth-mesh/