Smart Lawn Irrigation System Using Arduino Nano IoT 33

Shivakumar Nyamagoud
Team 06, GMU ECE508 IoT, Summer 2023
Department of Electrical and Computer Engineering
George Mason University, Fairfax, VA
snyamago@gmu.edu

Abstract:

In the US, every household has its own lawn or garden, and townships, parks, and commercial buildings also maintain their own green spaces. However, maintaining the landscaping can be quite expensive due to the need for manual labor in watering the plants and lawns can see in figure 1. If we rely solely on non-smart automatic watering systems, there will be significant water wastage and potential over-hydration of plants, even when rainfall occurs, as the automatic machines continue to water. Additionally, when owners travel out of town, they must hire contractors to maintain and water the lawn, incurring further costs. To address the issues of water wastage and plant/lawn health concerns, a single smart device that fulfills all these requirements at a low cost and with high efficiency is needed. The demand for efficient water management and plant/lawn health has led to the development of smart irrigation systems that utilize Internet of Things (IoT) technologies.

This paper introduces a smart lawn irrigation system utilizing Arduino, the MQTT broker protocol, a weather forecast system, and a soil moisture sensor and store all data using telegraf in a log file with the json format . The objective is to determine the optimal timing for watering the lawn or plants while minimizing water wastage and maintaining their health condition.

Figure 1. Manually watering to lawn

I. Introduction :

In the USA, one of the common challenges faced by households during the summer is the watering of lawns and other gardening plants. As a use case description there are several factors while addressing this issue. Firstly, if the watering system is manually then we need always one individual needs to water the law or grading plant where that scenario is not possible every time due to the other works or going to out of town. Secondly, if we are using just an automated watering system where watering sprinkler starts every day on timely basses then there will be huge of water wastage. And more over lawn may get to over hydrated and gets unhealthy. And lawn needs different amount water and other gardening plant needs different amount of water so automated motor will not take care of this. Thirdly, using a smart irrigation system that incorporates a microcontroller and soil moisture sensor, but without Internet of Things (IoT) connectivity, then it will read the moisture level and waters based on that but if currently soil moisture level extremely low and our system watering the lawn next hour there is extreme rain so that cause us lots of water and power wastage.

To overcome these challenges, in this paper I am going to present the smart lawn irrigation system using Arduino nano IoT 33 where the architecture design and implementation uses WiFi, weather forecast server to fetch the next two days rain forecast, capacitator soil moisture sensor and for communication between smart system and client using MQTT broker protocol after reading and computing all the data from the Arduino the published json file storing the data in telegraf log file for future reference and debugging and enhancement of the same system. the enhancement ideas are discussed in the same paper at after results section. this design is an efficient and cost-effective smart device that addresses all these solution to the above-mentioned issue. Such a device should ensure proper water management, cater to the specific water needs of different plants, and utilize IoT technologies to optimize watering based on real-time data.

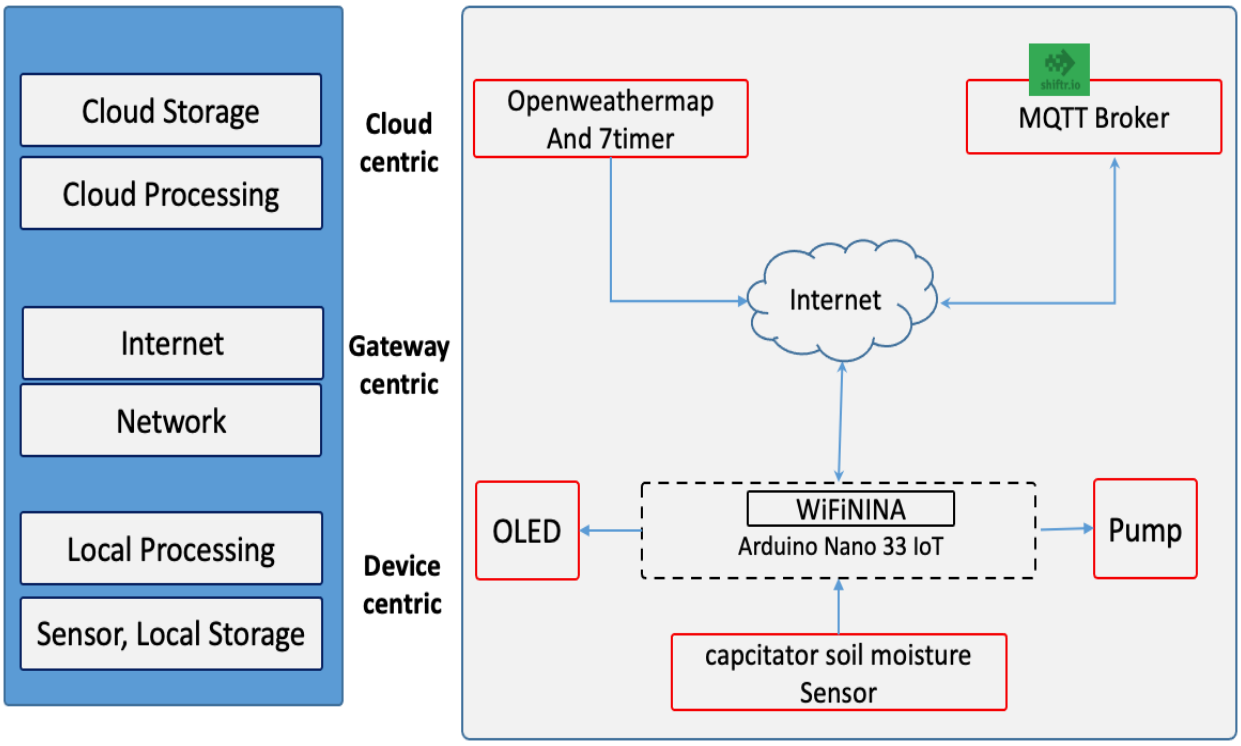

Figure 2. Architecture diagram of smart lawn irrigation system using Arduino nano IoT 33

II. Architecture.

In the architecture of smart lawn irrigation system Arduino nano IoT 33 is designed as per the standard IoT architecture. And it is being divided as three parts Device Centric, Gateway Centric and Cloud Centric. In the device centric Using microcontroller as local processing and local storage. For reading and collecting the data using sensor and as result to act as action using the DC motor as pump for watering the lawn. In gateway centric using internet so to connect internet using WIFININA library and microcontroller should support WiFi and accessing the data from servers. And in last as cloud centric first connecting and getting the data from the weather forecast website and reading the data and publishing the using the MQTT broker server and publish all topics on it and the associated values. And also, the same in telegraf.conf using to collect all the subscribed data in the json string for future reference.

From the Architecture right left side diagram I have shown the required components for smart lawn irrigation system as in the name I used microcontroller as Arduino nano IoT 33 because it supports the WI-FI and connecting to the sensor is bit simple. To display the current data, I used the OLED display screen, and for watering action, I used the DC motor as Pump. And to pull the weather forecast data from server I am using the openweathermap and 7timer servers. And to publish this data using the MQTT broker and shifter,io. Architecture and components are shown in the above figure2.

Data Flow:

You might have question, how the data flows from local processing microcontroller to data gets store in the telegraf log file and when that pump waters the lawn. It all starts with setting up both soil moisture sensor pin, relay pin, initializing the OLED and connecting the WiFi using WIFININA library. Once all initialization is done then start reading the rain and temperature weather forecast and the check the soil moisture level and based on the soil moisture level decide whether pump needs to turn on or off, after that read the motor status and collect all the data like current temperature, soil moisture level and today and tomorrow rain forecast and motor status, then convert in it the json format and initialize and connect to the MQTT broker and once mqtt broker gets connect publish the topic which we just now collected the information. After that in parallel run the telegraf conf file collect all the data in log file. The flow you can see the figure 3 flow chart.

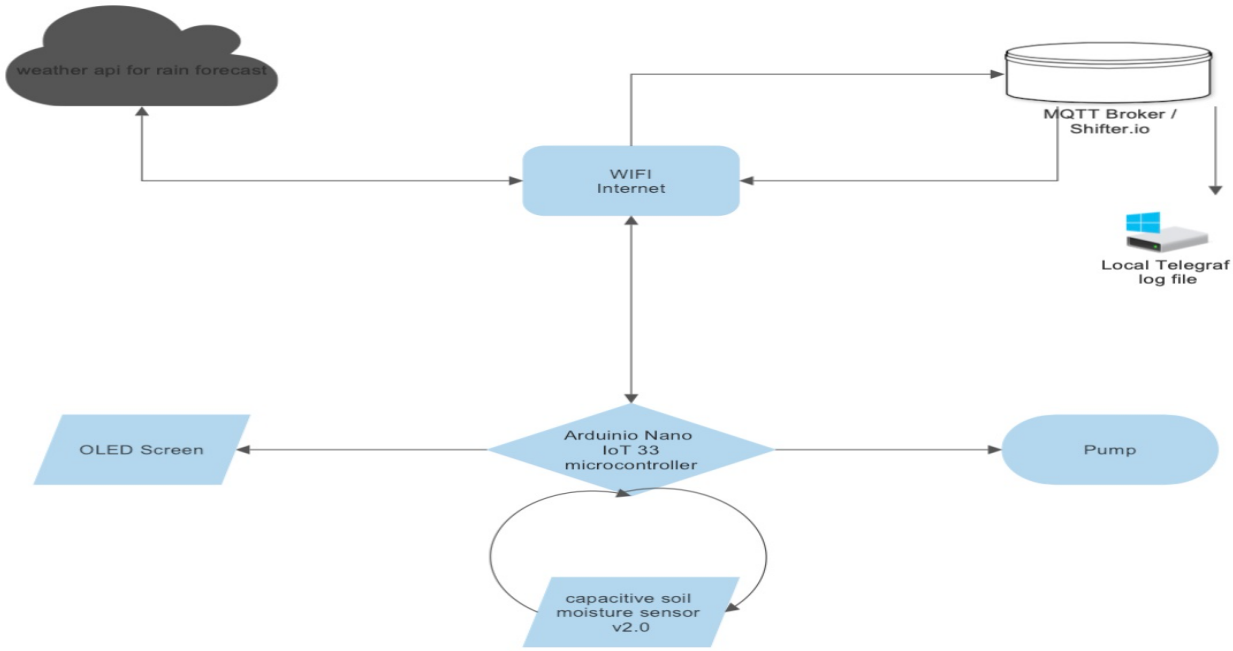

Figure 3 Flow Diagram of the IoT system

III. Components.

Hardware components:

1. Arduino Nano IoT 33:

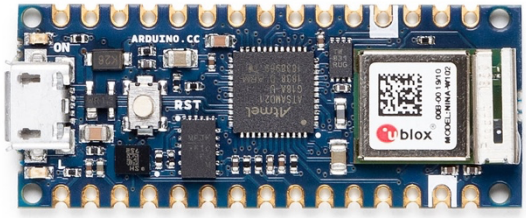

Figure 4 Arduino Nano IoT 33

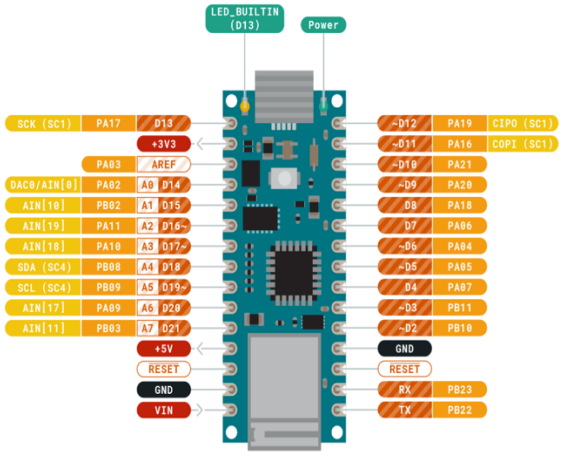

Figure 5 Arduino Nano IoT 33 Pinout

[1] The Arduino Nano 33 IoT combines the Arduino Nano form factor with an easy point of entry to basic IoT and pico-network applications. It is Arduino's smallest board to get started with Internet of Things (IoT). Using the popular Arm® Cortex®-M0 32-bit SAMD21 processor, it also features the powerful u-blox NINA-W102 Wi-Fi module and the ECC608A crypto-chip for security. Wi-Fi is supporting connectivity and cloud compatibility. And also, for Wi-Fi it uses WIFININA library packages. In this project it is the main advantage to get the server data and publish the topics. Apart from that It also supports the Bluetooth and BLE allowing you to control peripheral devices via Bluetooth. But we can use either Wi-Fi or Bluetooth at the same time. And Arduino comes with the inbuilt IMU LSM6DS3 sensor which is combination of accelerometer and gyroscope for it own motion tracking system. Device supports the UART, I2C, and SPI protocol and the clock speed of the device is 48MHz and has total memory of 256 KB SRAM, 1MB flash memory and the Nina W102 uBlox module has 448 KB ROM, 520KB SRAM, 2MB Flash memory. the Arduino Nano 33 IoT runs at 3.3V, this device no more supports the 5V because it is now an option for many modules and 3.3V is becoming the standard voltage for electronic ICs. Refer the figure 4 and 5 for Arduino board and pinout specification.

2. capacitive soil moisture sensor:

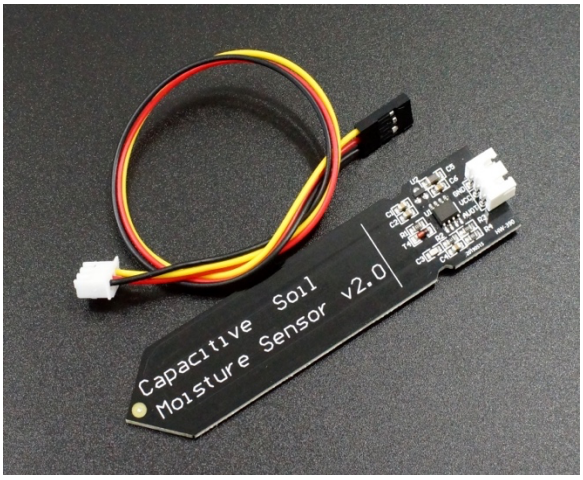

Figure 6 Capacitive soil moisture sensor

[2] Capacitive soil moisture sensor measures soil moisture levels by capacitive sensing rather than resistive sensing like other sensors on the market shown in figure 6. It is made of corrosion resistant material which gives it an excellent service life. Insert it into the soil around your lawn or gardening plants and read the real-time soil moisture data. This module includes an on-board voltage regulator which gives it an operating voltage range of 3.3 ~ 5.5V. but in this project we are providing 3.3 volt as power supply, and it is perfect for low-voltage MCUs. For compatibility with an Arduino, it will need an ADC converter. This soil moisture sensor is compatible with our 3-pin "Gravity" interface, which can be directly connected to the Gravity I/O expansion shield. In the Arduino IDE while coding we use the **analogread()** function to read the moisture value and the value depends on the voltage level it gets, based on the highest value and lowest value means both Wet value and dry value we need to convert it percent to scale of 100 for better reading and understanding.

3. 0.96" OLED Display:

[3] In the project I have used the 0.96” OLED display which has the highest resolution of 28x64 (have the same resolution as 12864, high PPI). The viewing angle of the oled screen is greater than 160 °. This display is the Ultra-low power consumption of only 0.6w in normal condition and it requires 3V-5V voltage supply and no need of shifter IC. Also, it works between -4°F to 158°F temperature range in industrial area which is perfectly fine. This oled is military graded process standard and the one which used in this project it supports yellow color in first two column and rest of the columns are bule color. And this project using the SSD1306 library[7] to support the OLED displaying. It has 4 pins two are VCC and GND and once is SCL is clock signed I2C line and SDA is Data signal I2C line which shown in figure 7.

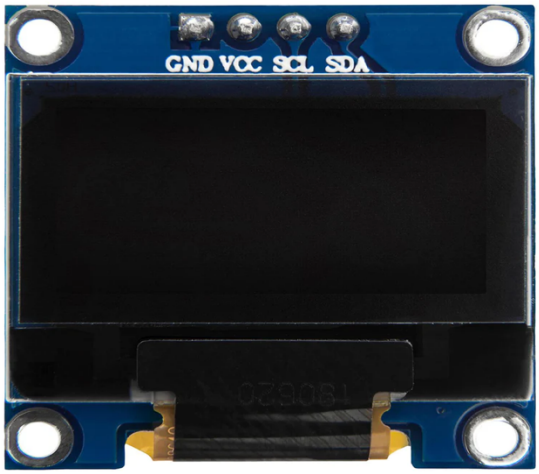

Figure 7 0.96" OLED Display

4. DC Motor(Watering pump)

Figure 8 DC Motor or Watering pump

5V DC motor is the Mini Submersible Water Pump for this motor using the external power supply unit because Arduino will not support the 5V power supply and using any poly pipe which connect to the bot end of the DC motor, and it has big flow rate

that can be 1.2-1.6L/minute. This water pumps provide low noise usage experience.

5. Relay Board.

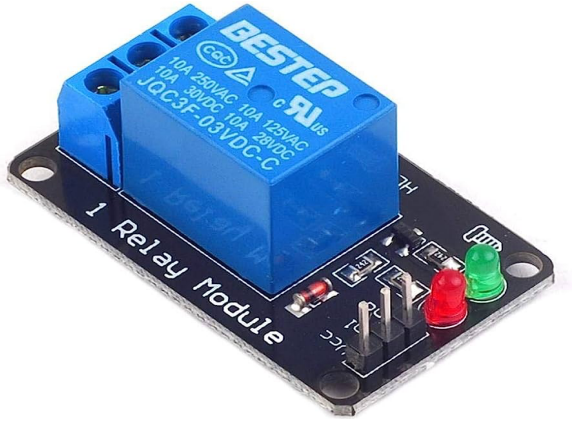

Figure 9 3V Relay Device

Relays are most used switching devices used in electronics. In this project I have used this to Turn on and off the Watering pump. It needs VCC and GND pins for power supply and one pin in controller input for relay which is connected to pin D2 of the Arduino device in the project. And other 3 pins are Normally open terminal and common terminal and Normally closed terminal, I have connected the DC motor one wire to Normally open terminal and other wire from 5V power supply unit to common terminal.

Software Components:

1. Arduino IDE:

[4] Software component Arduino Integrated Development Environment (IDE) which contains a text editor for writing code, message area, and text console a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. It supports many features which shown in figure 10.

Figure 10 Arduino IDE

2. Shifter IO:

[5] Shifter IO is the IoT Platform for Interconnected Projects which Rapidly connect hardware and software using shifter io cloud service and also provides the desktop app for users. It gives the real-time graph, supports Multiple protocols like MQTT broker and HTTP to publish, subscribe and retrieve messages. We can introspect and publish and subscribe topics and Webhooks Engine can only Integrate with existing HTTP services by using the webhooks engine to forward messages. And also supports the error reporting mechanism. Shifter it desktop UI shown in figure 11.

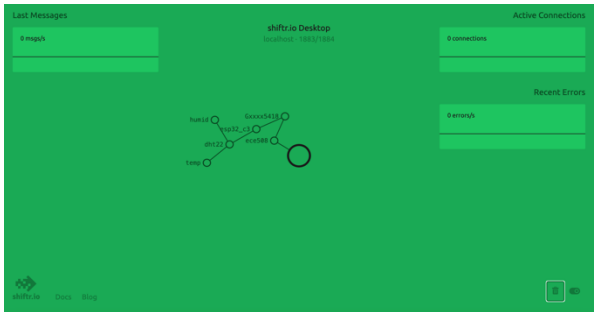

Figure 11 Shifter io desktop UI

3. MQTT Broker :

[6] MQTT is a messaging protocol. It is used to design for transferring messages and uses a publish and subscribe model where a publisher publishes messages on a topic and a subscriber must subscribe to that topic to view the message. In the MQTT it possible to send messages to 1 or multiple clients and at the same time we can subscribe the multiple message topics from the MQTT clients. All clients can publish (broadcast) and subscribe (receive). Shown in figure 12. MQTT brokers do not normally store messages so that in this project to store the published message I am using the telegraf by using the config file which has all the topics and IP and local storage file location to save all published message in json string.

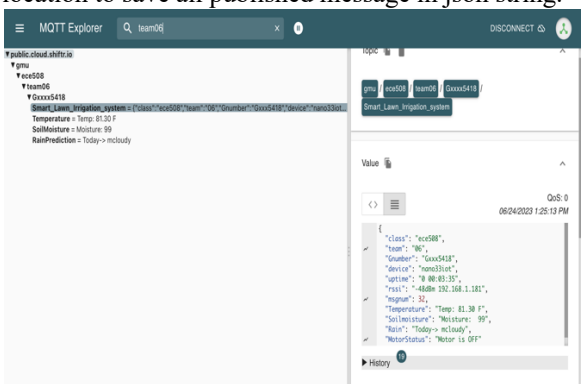

Figure 12 MQTT Desktop View

4. Weather forecast server:

{ "product" : "civillight" , "init" : "2023062706" , "dataseries" : [{ "date" : 20230627, "weather" : "pcloudy", "temp2m" : { "max" : 28, "min" : 16 }, "wind10m_max" : 3 }, { "date" : 20230628, "weather" : "lightrain", "temp2m" : { "max" : 25, "min" : 15 }, "wind10m_max" : 3 }, { "date" : 20230629, "weather" : "clear", "temp2m" : { "max" : 31, "min" : 14 }, "wind10m_max" : 2 }, { "date" : 20230630, "weather" : "cloudy", "temp2m" : { "max" : 35, "min" : 16 }, "wind10m_max" : 3 }, { "date" : 20230701, "weather" : "lightrain", "temp2m" : { "max" : 26, "min" : 20 }, "wind10m_max" : 2 }, { "date" : 20230702, "weather" : "lightrain", "temp2m" : { "max" : 29, "min" : 20 }, "wind10m_max" : 2 }, { "date" : 20230703, "weather" : "lightrain", "temp2m" : { "max"

Figure 13Weather forecast from 7timer server.

{"coord":{"lon":-77.4875,"lat":39.0437},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":301.99,"feels_like":304.37,"temp_min":300.2,"temp_max":303.6,"pressure":1005,"humidity":63},"visibility":10000,"wind":{"speed":3.13,"deg":180,"gust":0},"clouds":{"all":0},"dt":1687793219,"sys":{"type":2,"id":2004257,"country":"US","sunrise":1687772738,"sunset":1687826382},"timezone":-14400,"id":4744870,"name":"Ashburn","cod":200}

Figure 14 Weather forecast from openweathermap server.

To get the rain forecast and current temperature data I am fetching the data from 2 forecast servers. First one 7timer server where it gives the once week rain forecast in days. And second is from openweathermap server where it gives the current rain status and current temperature on hourly basis. So, both are opensource server we can send get request n number of times and give forecast of both hourly and daily up to once week so it help us decide whether we can water our lawn and garden. In the current project I am refereeing only today and tomorrow rain forecast.

IV. Hardware Connection :

As in figure 13 shown need to connect all components and (this diagram drawn using the Fritzing tool) [note : some components look different in this diagram I choose available nearest image).

V. Results from IoT system:

Main aim is to save the water from being waste and save power and main health lawn by avoiding over hydration. In result I can see the above-mentioned requirements are achieved. I used the moisture level as 40% which is moisture level is below 40% and next two days there is no rain then I am not

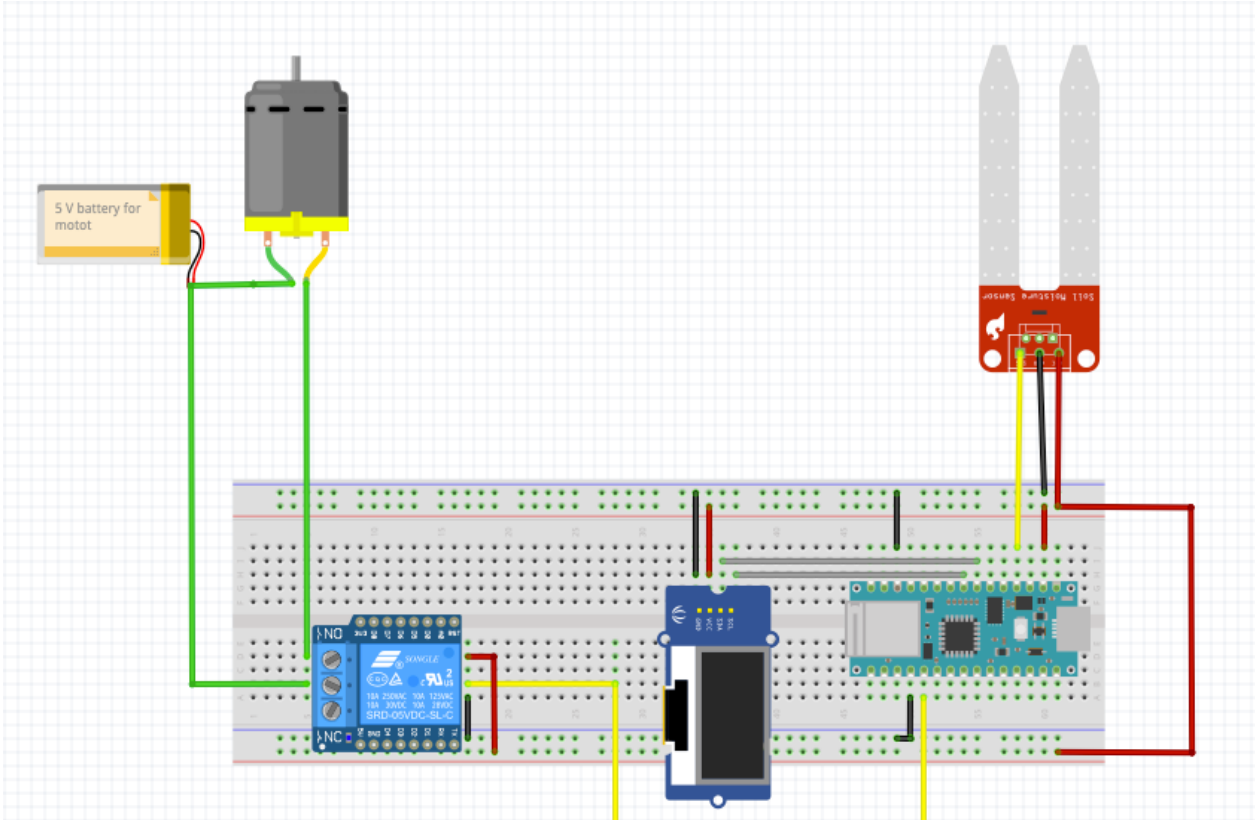

Figure 15 Hardware component connection circuit diagram

watering the lawn and my garden. The motor is remain turned off. If the moisture level is above 40% again motor remains off. If next two days, there is no rain and moisture level is below 40% then I am turning on the motor. Until moisture level gets 100%, I am not turning of the motor. And also in the display I am printing the current temperature in Fahrenheit for the reference. Along wing in the last line printing the todays Date and time EST for reference the history on the saved logs.[Note: all tested results are not from the same day, I have tested in different days to get the different rain forecast]

In the following figures you can see the result of the IoT system.

Figure 16 Moisture level is 100 and Motor is OFF

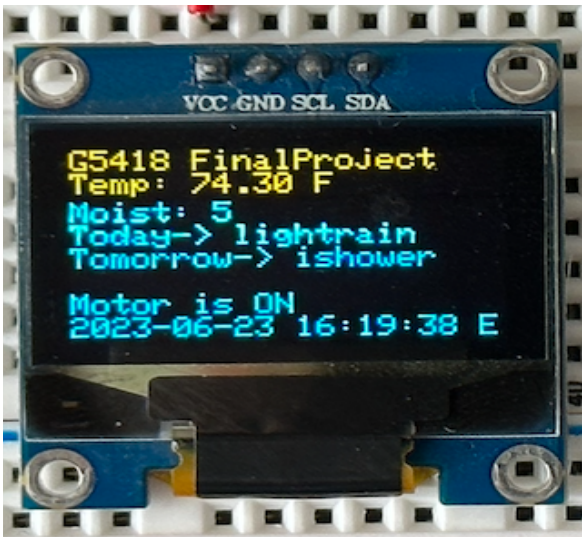

Figure 17 Moisture Level is 5.

In this image Moisture level is 5% and the next two days rain forecast is lightrain so that motor is turned ON.

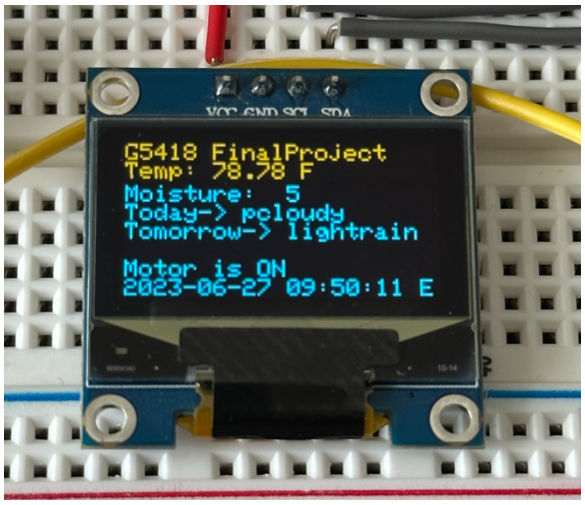

Figure 18 Rain forecast is probably cloudy

In the figure 18 you can see the rain forecast for today is probably cloudy and tomorrow is light rain so Arduino turned on the motoring and started watering to the lawn.

In the MQTT broker also you see the data in the json format which is being published from the Arduino, using shifter.io or any cloud ip you can access this data from any remote location. In figure 19 you can see the data.


```
{
  "class": "ece508",
  "team": "06",
  "Gnumber": "Gxxx5418",
  "device": "nano33iot",
  "uptime": "0 00:05:12",
  "rssi": "-56dBm 192.168.1.181",
  "msgnum": 46,
  "Temperature": "Temp: 79.72 F",
  "Soilmoisture": "Moisture:  100",
  "Rain": "Today-> pcloudy",
  "MotorStatus": "Motor is OFF"
}
```


Figure 19 MQTT Broker Publisher data

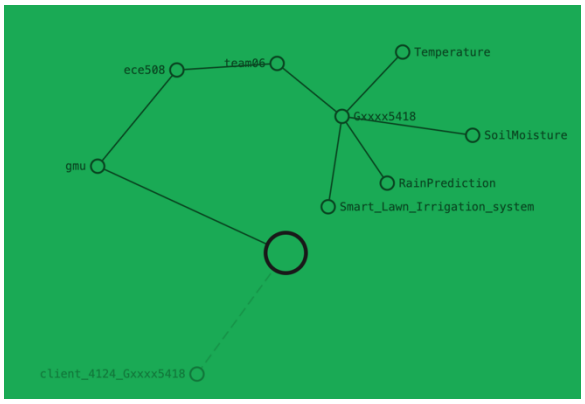

Figure 20 Shifter io desktop graph

In the figure 20 and 21 data is being published using local ip and desktop shifter io.

Figure 21 Desktop shifter io published messages data.

VI. Conclusion:

This paper/project provides Using the Arduino Nano IoT 33 provides the end solution to the problems which I have addressed in introduction of this paper b incorporating advanced technologies such as WiFi connectivity, weather forecast integration, soil moisture sensors, and MQTT communication protocol, this system provides efficient water management and plant care with a low cost and highly efficient smart lawn irrigation system. And also using this kind on IoT system can make the lawn owner to access each data and from remote location also they can observe the watering of the lawn and what is the current moisture level of the soil. So as conclusion this is the low cost. Overall, this smart irrigation system using Arduino nano IoT 33 microcontroller provides the convenient and automated solution for efficient water management by optimizing watering based on real-time data and advanced IoT technologies.

VII. Future work:

As future work or development of this system we can add multiple soil moisture sensor with multiple motor where I can water all types of plants like indoor and outdoor plants as well as bigger lawn area. And to make current system highly precise and more intelligent we need add multiple if and else state with data comparing where it will behave as AI system.

References :

[1] Arduino, “Arduino Nano 33 IoT.” https://docs.arduino.cc/hardware/nano-33-iot.

[2] Soil moisture sensor, “Capacitive Soil Moisture”https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193.

[3] 0.96” OLED Display http://www.lcdwiki.com/0.96inch_OLED_Module_MC096VX.

[4] Arduino IDE “https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics"

[5] shifter io “https://www.shiftr.io”.

[6] MQTT Protocol “https://mqtt.org/getting-started/”.

[7] “Adafruit_SSD1306 Library.” https://github.com/adafruit/Adafruit_SSD1306