## Project Summary

For our science fair project, we created a smart automatic sprinkler system that improves existing frameworks with multiple components that help reduce water usage and electrical usage, while also taking into account multiple variables, most notably soil moisture levels and weather forecast data taken from an online database. Using algorithms that we created and implemented, the system decides when to water your plants based on a scoring system, as well as the quantity of rain. The numbers used can all be customized depending on the usage.

## Origin

We got this idea when looking at the sprinkler systems in our gardens. They would always be active, even on rainy days, which wastes a lot of water, and we realized how much water could be saved by considering the weather forecast before irrigating plants.

## Introduction

Water and electricity are two valuable resources, yet traditional watering systems waste significant amounts of both due to inefficient scheduling and overwatering. In order to solve this issue, we've created a smart sprinkler which optimizes plant watering using a soil moisture detection algorithm and real-world weather forecast data. When making this project, we set a clear objective: to improve upon existing smart  sprinkler systems with multiple additions that help reduce water usage and electrical usage, while also taking into account variables such as the weather forecast.

**Method and Results**

In order to conduct research for our project we consulted tables and charts found on an online gardening website in order to collect data about the ideal soil moisture percentage level. We also used the Old Farmer's Almanac, which is an encyclopedia about plants and gardening in order to gather brief descriptions about each of the plants we included. With the help of online sources, we researched the different kinds of transistors as well as their advantages and disadvantages.

In order to make our smart sprinkler system, we first began by writing the piece of code for the Arduino that allows us to read soil moisture data. Once we finished that step, we put it to the test by connecting the soil moisture sensor to the Arduino in order to ensure that both the code, and the sensors worked. While testing, we noticed that on average, water had a soil moisture level of 100%, wet soil had a soil moisture level of 59%, dry soil had a soil moisture level of 34%, and air had a soil moisture level of 5%. Once we were satisfied with the accuracy of the readings the sensor gave us, we wrote the code for the pump, and made a circuit to control a DC motor based on the soil moisture level using a metal oxide field-effect transistor (MOSFET) to test the water pump code. We used a DC motor due to its similarity to a water pump in terms of electricity usage and wiring. We used the same code for both the DC motor and the water pump. This helped us save time during the testing of the code. Additionally, we didn't have to waste water in order to test the software. Once we had the DC motor working, we replaced it with the water pump without altering the code or circuit and made sure it worked, thus completing the base of our design. Once that was completed, we added a weather detection algorithm that determines whether a plant should be watered based on the rain chance percentage as well as the quantity of rain that will fall in the coming 7 days. It would use a reward system

(weather score) and tally points based on the multiple parameters, including the two mentioned above. We've also included a limit in our algorithm for the maximum number of days a plant can be left without water. This made sure that if the weather forecast is wrong, the plants won't suffer. We wrote code that would collect weather data from OpenWeatherMap.com using the ESP32, and then would be transferred onto the Arduino to be used in the algorithm. Simultaneously, we also made a website that acts as a database of various garden and houseplants. This database contains the ideal soil moisture percentage levels for various plants, in order to set the minimum soil moisture level requirement to the right level for a particular plant, as well as a short description of each. We will also be testing for the exact amount of electricity saved with our system by using a voltage meter in order to test the amperage when our Arduino is sleeping.

The framework that we've created is based on 13 customizable variables that can be separated into two categories. The first category our system considers when deciding whether to water your plants is the dryness of the soil, which is detected by a sensor. If the moisture is below a certain level, the water pump will water the plants unless it interferes with the second component. The other, much more sophisticated component considers real-world forecast data before choosing whether to water your plants. We designed this system by creating an algorithm to determine how long before precipitation the plants should block watering. Much like an algorithm that uses machine learning, ours uses rewards to decide when to allow soil moisture sampling. For example, if the mechanism knows that there is an 80% chance of rain in two days, it will treat it differently than a 60%  rain chance in five days. Even more emphasis is put on the

quantity of rain; two variables can be used to differentiate between light, moderate, and heavy rain, which is measured in millimeters.

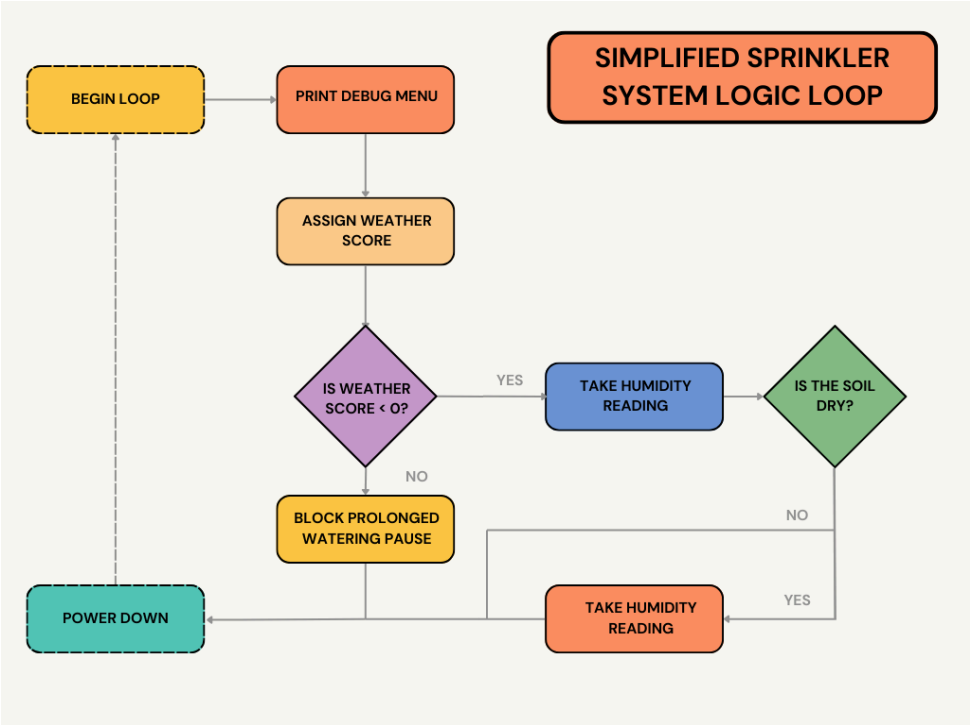
**Appendix A: Average Soil Moisture Reading by Substance**

| **Average Soil Moisture Readings** | | | | |
|---|---|---|---|---|
| **Substance** | **Water** | **Wet Soil** | **Dry Soil** | **Air** |
| **Reversed Voltage (1-1023)**[1] | 1023 | 600 | 350 | 55 |
| **Percentage (1-100)** | 100% | ~59% | ~34% | ~5% |

As mentioned previously, saving electricity was an important factor that shaped our decisions throughout the project. In order to save electricity, we wrote code that turns off the electricity-heavy components of the Arduino when they aren't needed. To do this, we made use of the sleep_n0m1 Arduino library. The amount of water saved varies based on the amount of time the plants are watered for, as well as the frequency of humidity readings. However, a plant using our system that needs to be watered for 15 seconds twice a day can save hundreds of times the electricity compared to one that does not use this system.

[1] Voltage is reversed for clarity since high moisture is represented by a low analog value

**Appendix B: Simplified Sprinkler Logic Loop**




Our design has many practical applications. Although it can be used for indoor plants and still save water due to the soil moisture detection algorithm, it is principally designed for watering outdoors. One advantage of our system is that it can be scaled up or down very easily and could theoretically be used on a plot of any size. The only changes that would have to be made to our prototype would be the number of soil moisture sensors and water pumps, as well as replacing the sprinklers or water pumps with a more powerful counterpart. On a larger scale, tubes connected to the water pumps could be installed in a rain collection tank. This water could then be used to automatically water the plants through a network of pipes. Our sprinkler system would be set up in a very similar way as a traditional arrangement since most of our updates are software-oriented. Production costs would also be similar to that of a regular sprinkler system due to the fact that most of the components we use are the same or similar to those used in a traditional sprinkler. In fact, our system would be more financially friendly in many cases due to

the saved water and electricity costs from bills. The same is true for the saved environmental cost. One of the key features of our design is that it's automatic. All of the variables, while being customizable, have default values that can be used in many scenarios. This means that once the software is installed, it waters automatically, without having to manually be provided every day with upcoming weather data. Our system has the advantage of collecting that information on its own.

**Conclusion**

We obtained our objective which was to create a smart sprinkler system that saves both water and electricity while ensuring its use on a daily basis. While completing this project, we enhanced our knowledge of Arduino and its software, APIs, and electrical components such as transistors. We believe that the design we have created has the potential to change the way we think about gardening and automated agriculture, especially if more accurate and precise hardware is used.

**Bibliography**

Team, AcuRite. Guide: Soil Moisture Recommendations for Flowers, Plants, and Vegetables [Online]. AcuRite; 2018 [Updated 2022 Mar 18; Cited 2025 Feb 23]. Available from: www.acurite.com/blogs/acurite-in-your-home/soil-moisture-guide-for-plants-and-vegetables#:~:text=Recommended%20Soil%20Moisture%20Levels&text=It%20is%20important%20to%20note,between%2041%25%20%2D%2080%25.

Thomas, Robert B: The Old Farmer's Almanac. [Online]. Yankee Publisher; 1996 [Updated 2025 Feb 23; Cited 2025 Feb 23]. Available from: https://www.almanac.com/gardening/growing-guides.

Tutorials Point:Tutorialspoint [Online]. Tutorials Point; [cited 2025 Feb 23]. Available from: https://www.tutorialspoint.com/basic_electronics/basic_electronics_types_of_transistors.htm.

Vishay: irf520 power MOSFET. Vishay [Online] [Cited 2025 Feb 23]. Available from: https://www.vishay.com/docs/91017/irf520.pdf

Shibley, Noah, Grant, Michael. Sleep_n0m1 (v1.1.1, 2017). Github [Cited 2025 Feb 23]. Available at: https://github.com/n0m1/Sleep_n0m1

Arduino. Arduino Forums [internet]. Monza, Italy: Arduino; 2025. Available from: https://forum.arduino.cc/

Arduino LCC. Wifi (v1.2.7, 2017). Github [Cited 2025 Feb 23]. Available at: https://github.com/arduino-libraries/WiFi

Blanchon, Benoît. ArduinoJSON (v7.3.0, 2025). Github [Cited 2025 Feb 23]. Available at:https://github.com/bblanchon/ArduinoJson

McEwen, Adrian. HTTPClient (v2.2.0, 2015). Github [Cited 2025 Feb 23]. Available at: https://github.com/amcewen/HttpClient

Science Buddies. Control a Pump with Arduino (Lesson #15) [Online Video]. [Viewed 2025 Feb 23]. 11 min 4 sec. Available from: https://www.youtube.com/watch?v=To3DKP99-1U&t=586s