ECE 1000 Final Report: Automatic Plant Watering System

Summer Morris, Isaiah Short
Electrical Engineering Department
Tennessee Tech
Cookeville, Tennessee
sdmorris42@tntech.edu & ilshort42@tntech.edu

***Abstract*—This project report covers the Automatic Plant Watering System. This project was developed based on the environmental challenge set by IEEE. This the goal of this project is to develop an automatic watering system that can keep the plant at an optimum moisture level, while using less energy and water. With the Automatic Plant Watering System, the soils moisture levels will stay within the ranges of 29 to 41 percent.**

Keywords—Water Pump, OLED, Raspberry Pi Pico, Moisture Sensor, Automatic, MicroPython

I. Introduction

It can be difficult to give plants the right amount of water necessary to grow. With an automatic plant watering system, plants will not only be able to receive the perfect amount of water, but will also decrease the amount of wasted water. This project is significant because it is motivated behind the environmental challenge in IEEE. With a more precise watering system, plants will be able to flourish while not wasting water. It is crucial to implement systems similar to this project, especially in locations where water is more difficult to obtain. This project was developed by Summer Morris and Isaiah Short, both of which are electrical engineering majors. To begin the project, research is necessary in order to fully understand the project.

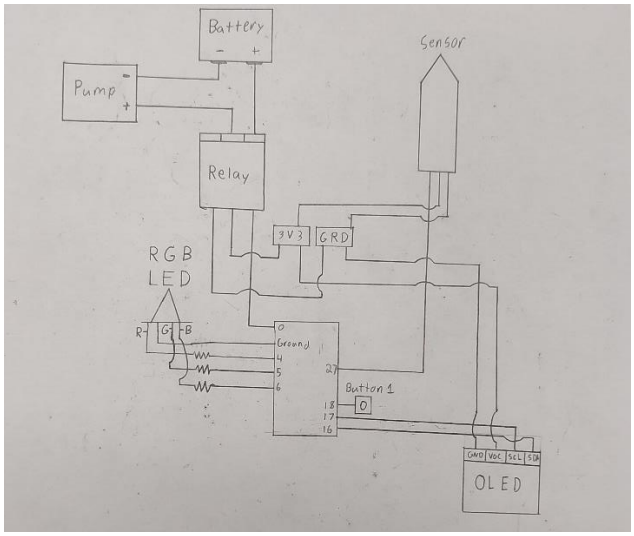
II. Background

The team used the article written by Les Pounder to figure out how to show and erase text on the OLED screen [5]. The team used the article written by Abhilekh Das that demonstrated how to code the temperature onto a OLED screen in order to display the current moisture level on the OLED [1]. The group used the article written by Admin [4] on how to control a RGB LED with a Raspberry Pi Pico and Chat GPT [6] to figure out how increase the brightness of the RGB LED. The team used the code written by JCWilliams1003 to code the moisture sensor [7]. The group used the code written by Rahul Khanna to code the pump to turn on and off [3]. The group used the circuit diagram drawn by leonthorn2001 to connect the battery to the pump and relay [2].

III. Project Description and Formulation

The materials used in this project include a Raspberry Pi Pico, moisture sensor, OLED screen, RGB LED, water pump, relay, and a 9 volt battery. The AOUT on the moisture sensor is connected to pin 27 on the Raspberry Pi Pico. The voltage pin and ground pin on the moisture sensor is connected to the 3V3 pin and ground pin on the Pico respectively. For the OLED, the clock is connected to pin 17, the data pin is connected to pin 16, VCC is connected to 3V3, and ground is connected to ground on the Pico. The RGB LED connects the red, ground, green, and blue pins to pins 2, ground, 3, and 4 respectively. For the relay, pin s is attached to pin 0 on the Pico, the negative pin is attached to ground, and the positive pin is connected to the VBUS pin on the Pico. On the other side of the relay, the middle is connected to the positive wire on the water pump, the left terminal is connected to the positive side of the 9 volt battery, and the right terminal is not connected to anything. The positive wire on the water pump is connected to the relay, as mentioned above, and the negative wire on the pump is connected to the negative side of the 9 volt battery.

Fig. 1. Schematic of the project.



The code first imports all necessary items for the components in the project. The code then labels and defines the red, green, and blue pins on the RGB LED. This will allow the user to change the RGB LED to specific colors. The code also sets the duty cycle for the red, green, and blue pins, which will allow the user to control the brightness of the RGB LED. The code then goes on to label the pins for the push button, water pump, and moisture sensor. The code then takes the measured values on the moisture sensor and defines it as a percentage

between 0-100. By doing so, the user is able to easily understand the outputs of the sensor.

Fig. 2. The welcome dialog on the OLED.

The code then tells the OLED to display the message "Automatic Plant Watering System" for 2 seconds. This message will appear automatically when the Raspberry Pi Pico is turned on and the code is run. This text is used to start the automatic plant watering system because the text gives the system a soft start and looks better overall. After two seconds, the OLED clears and the regular code begins. The OLED will display the percent moisture levels and will update every second. The percentage will also print on the computer as a backup in case the OLED stops functioning.

Fig. 3. While the pump is off.

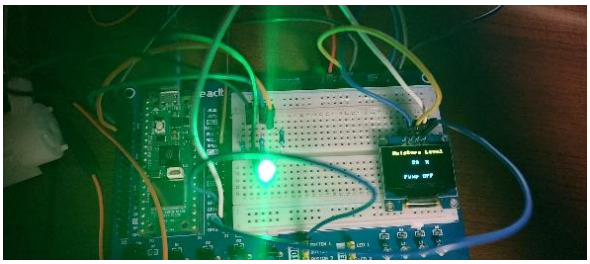


Fig. 4. While the pump is on.

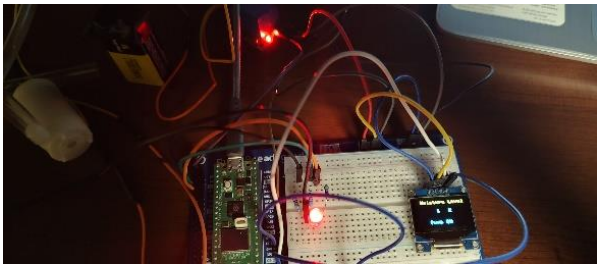
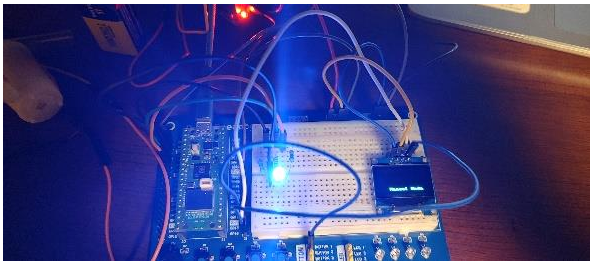


Fig. 5. While the system is in manual mode.



While the percentage is greater than or equal to 40 percent, the pump is turned off, the RGB LED is green, and the OLED displays "Pump OFF" below the percentage. If the moisture level is less than or equal to 30 percent, the pump will turn on, the RGB LED will turn red, and the OLED will display "Pump ON" below the moisture level. If the sensor values are between 30 and 40 percent, the system will either remain on or off until it reaches 30 or 40 percent. The percentage needed to turn on and off the pump has a 10 percent difference due to the fact that most plants have a wide range of moisture levels acceptable for the plant to thrive. By setting the upper and lower bound apart, it will decrease the time the pump is running, therefore saving water and energy. If button 1 is pressed on the Raspberry Pi Pico, the pump will turn on, the RGB LED with turn blue, the OLED will display "Manual Mode", and the code will print "Manual Mode" on the computer. The manual mode will turn off after 5 seconds.

IV. Discussion and Results

The automatic plant watering system continually monitors the moisture level and updates the percentage every second. The moisture level, as long as the water pump has sufficient water and power is supplied, will continually stay within the range of approximately 29 percent to 41 percent, when given a minimum moisture level of 30 percent and maximum level of 40 percent. The RGB LED turns green when the moisture level is within that range, red when below that range, and blue when the user overrides the sensor and puts the system in "manual mode."

The team enjoyed building and testing the circuit the best, especially when it came to testing the OLED and RGB LED's. Summer built the circuit and programmed the circuit to complete the tasks necessary for this project. Summer also wrote the project report and found the sources necessary to complete the project. Isaiah brought in the soil and container for the water. The GitHub was made by both Summer and Isaiah. Isaiah took the video demonstrating this project. If more time was permitted for this project, the team would continue to test the moisture sensor in order to provide a more accurate moisture percentage. The team would provide a power supply of some kind that can allow the Raspberry Pi Pico to function without a computer. The team would also set up a buzzer and sensor to monitor and alert the user once the water supplied to the pump is low. The team could also design a waterproof container to put the Raspberry Pi Pico and other sensitive components in for safety.

V. Conclusion

The purpose of the Automatic Plant Watering System was motivated behind the environmental challenge set by IEEE. This project is necessary to give plants sufficient water to thrive while conserving energy and water. After the conclusion of this project, the team has gained skills in programming with MicroPython, working with a Raspberry Pi Pico, and using components such as the OLED and moisture sensor. The team also learned how to properly conduct research and cite the sources used in IEEE format. These skills are important because they will all be used in an professional electrical engineering career. The Automatic Plant Watering System will continually

monitor and water the plant in order to keep the moisture levels between approximately 29 to 41 percent.

REFERENCES

[1] A. Das, "Read Raspberry Pi Pico's Onboard Temperature Sensor (MicroPython Code)," *Electrocredible*, May 31, 2022. https://electrocredible.com/raspberry-pi-pico-temperature-sensor-tutorial/ (accessed Apr. 21, 2024).

[2] leonthorn2001, "PlantyPi - Raspberry Pi Pico Plant Watering Device.," *Instructables*, 2024. https://www.instructables.com/PlantyPi-a-Raspberry-Pi-Pico-Plant-Watering-Device/ (accessed Apr. 21, 2024).

[3] R. Khanna, "Automatic-Plant-Watering-System-using-Raspberry-Pi-Pico-," *GitHub*, Jun. 15, 2021. https://github.com/Rahul24-06/Automatic-Plant-Watering-System-using-Raspberry-Pi-Pico-/blob/main/final.py (accessed Apr. 20, 2024).

[4] Admin, "Controlling RGB LED using Raspberry Pi Pico," *How To Electronics*, Oct. 02, 2022. https://how2electronics.com/controlling-rgb-led-using-raspberry-pi-pico/#google_vignette (accessed Apr. 21, 2024).

[5] L. Pounder, "How to Use an OLED Display With Raspberry Pi Pico (Updated)," *Tom's Hardware*, Apr. 23, 2021. https://www.tomshardware.com/how-to/oled-display-raspberry-pi-pico (accessed Apr. 20, 2024).

[6] OpenAI, "ChatGPT response to 'how to convert the code to 16-bit'," unpublished.

[7] JCWilliams1003, "ECE_1000_Soil_Moisture_Sensor_Example.py," *GitHub*, Apr. 15, 2024. https://github.com/JCWilliams1003/ECE-1000-Spring-2024-Final-Project-Insert-Project-Name/commit/8e8547829173db8dfa8677a816fea5ad9e60c26d#diff-eba3eff776c3232725d6370afc73d853b2316c58a1f0c294994ed762906e2f5e (accessed Apr. 23, 2024).