Technical report - Automated plant watering system

Shravya Appannagari ,
Dino Kontschieder and Carlos Franco Verde Arteaga

August 2023

Contents

Chapter 1

The project's idea

We would like to solve a common problem which causes many health issues to plants. When we have a stressful week, or when we are on holiday etc, we often forget to hydrate our plants and they dry out. Since this happens a lot in most of our households, we have decided to take better care of our plants by creating a system which should automatically water the plants whenever its needed.

To make sure our plant stays healthy the owner must have access to all essential information which affect the plant's health. These information should be provided as easy as possible. Furthermore, the owner/user should be informed if there are unhealthy environmental conditions for the plant. Finally, the project must lower expenses to take care of the plant by automatically watering the plant when necessary.

Chapter 2

Results

Our project is realized by using different kinds of sensors like a temperature and a humidity sensor, a water level detection sensor, a light sensor and a soil moisture sensor. Furthermore it includes a motor water-pump, a battery and a micro-controller as part of the hardware. All the delivered information of the sensors are processed in our program by an Arduino Nano 33 IoT.
With this setup we automatized the watering process of the plant when necessary. Furthermore, the owner of the plant gets real time information of the plant details which affect their health like room temperature, room humidity and light. Information like not enough light or an empty water container are shown on a messenger of the Arduino cloud user interface and can be viewed via a desktop PC or with the phone via the Arduino cloud app. The Arduino cloud user interface is only accessible for registered users.

2.1 Used Hardware

Figure 2.1 shows the hardware we used in our project. A battery box, a transistor and a diode are also necessary for the water pump, but are not shown in the figure.

2.1.1 Micro-controller

As we want to use cloud to show the status information of the plant, we need a micro-controller which is compatible with a Cloud, Wi-Fi and Bluetooth. Unfortunately when creating the project proposal we didn't check if the Arduino Uno R3 has Wi-Fi and Bluetooth modules. Therefore, we had to find a new micro-controller where the Wi-Fi and Bluetooth compatible implementation is possible. Therefore we decided

to use the Arduino Nano 33 IoT instead of the Arduino Uno R3. Using this specific micro-controller we extend the options for solving our problem.

2.1.2 Water pump

Before the presentation we used a relay to turn the pump on and off. Now we changed it by using the PWM and a transistor. The motor of the pump will spin faster when the PWM value is higer and will spin slower when the value is lower. In our circuit we implemented a NPN transistor (TIP4C) to switch up the power supply for the motor. Since motors generate current when spinning, we implemented a diode (1N4007) parallel with the motor leads to prevent our circuit from being damaged.

2.1.3 Sensors

- Temperature and humidity sensor (DHT22): To make the DHT22 usable with the Arduino we included the adafruit library for this sensor.

- Water level sensor: Since we use a water container, we need to check the water level to know if there is enough water in it. Therefore we added a water level detection sensor. By immersing this sensor into the water, the output voltage changes in the signal pin of the Arduino. With this sensor we have to be very careful because it is not designed to be fully submersed. Only the exposed traces on the sensors circuit board should be in contact with water. To detect if our water container is empty we need to define a threshold value. By a sensor's value of 600 or less we defined our water container as empty. If the sensor delivers less than 600, the user gets a notification to refill the water container, the motor will be merged in the water, using this value we ensure that the motor will take water from the recipient. The system will turn off the motor slowly and the PID can not be activated if the container has not enough water.

- Photoresistor: Plants may be located in surroundings with less light. To be sure that the plant has enough light we implemented a light sensor. This light sensor is also called photo resistor which changes it's resistor value by changing the light. The boundary for the transition to not enough light is made by defining a threshold value of 400 or less in our program. The program knows when the plant has not enough light by defining a range for a healthy amount of light. For example if the plant is located in a dark room for 3 days the

system will inform the user. As mentioned at the beginning we defined the waiting time of 5 seconds for testing purposes.

- Capacitive moisture sensor: It is stuck into the soil and measures its moisture. Since it has an analog output it delivers values between 0 and 1023.

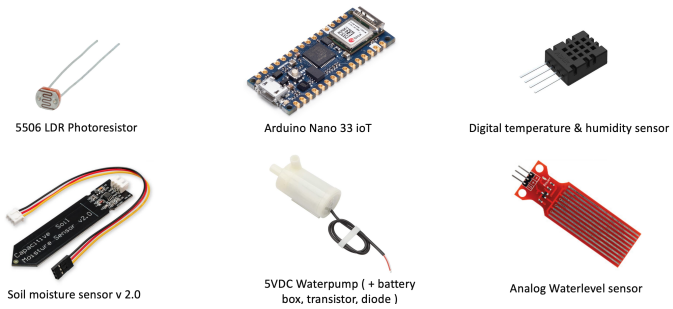

Figure 2.1: Used hardware

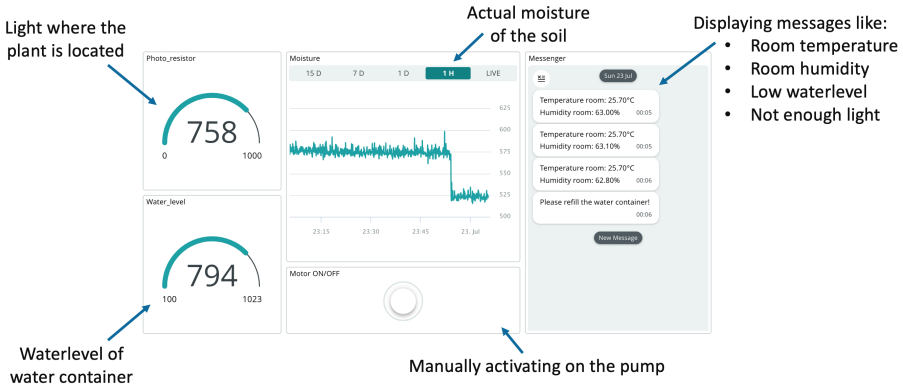

Figure 2.2: UI Arduino Cloud showing the sensor values

2.2 Summary of the project

- The setup automatically waters the plant, based on the measures of the soil moisture sensor and water level detection sensor.

- The user can manually water the plant by pressing the button from the cloud user interface, but the button can only be activated 5 times a day as a maximum. In the program, this is a global variable that can be changed.

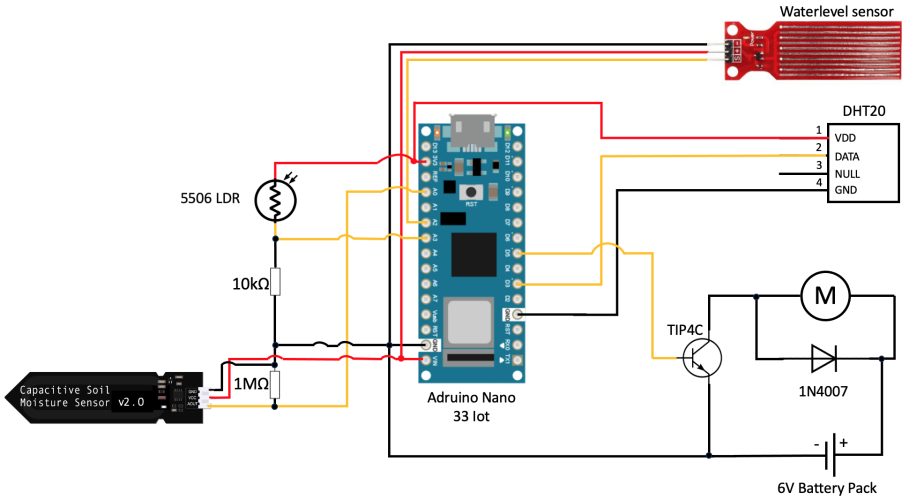

Figure 2.3: Electric circuit

- Using the automated way, based on soil reading, the plant is watered (or the motor is run) either usual or longer time, the control of this is based on a PID. Using the manual way, the motor runs and for this way the difference is that the user can only water the plant 5 times per day.
- The light sensor warns the user if the plant has not enough light continuously for three days. For testing purposes, we have included the waiting time of 5 seconds.

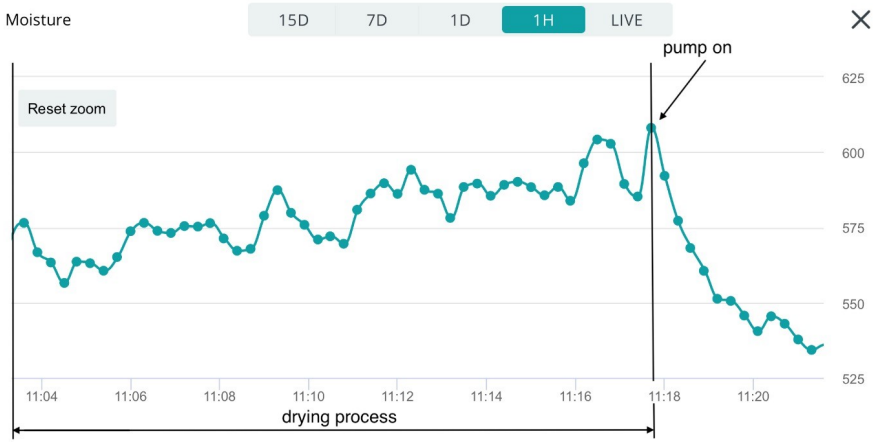

Figure 2.4: Soil measure cloud

The Figure 2.4 shows what happens when the motor is activated.

Chapter 3

Main challenges and solutions

3.1 Capacitive Moisture Sensor

The capacitive moisture sensor seems to deliver unrealistic measurements sporadically. This was the case with all of the capacitive moisture sensors we bought. We tried our system with a resistive soil moisture sensor and the measurements looked fine. Since a resistive soil moisture sensor starts to corrode after a short time we searched for a solution to use a capacitive moisture sensors. After some research we found out that the capacitive moisture sensors we bought had some severe quality issues. In the Figure 3.1, you can see the capacitive moisture sensor v2.0 with a tiny hole close to the resistors. Normally this tiny hole should be exactly between the resistors and not under them. Because of that there was no connection between the 1MOhm resistor and the ground. This makes the response of the sensor by changing the moisture of the soil very slow. We fixed the problem by soldering a 1MOhm resistor between the ground pin and the analog output as in the Figure 3.2.

Furthermore, the sensor needs time to detect the changes of humidity (when we water the plant). To solve this, we calculated the average value of 5 samples of the soil sensor data after waiting 1 second in our program(without stopping the void loop). Based on this average value, we programmed to automatically turn on the water pump, when the sensor is measuring under the set point (the limit for our PID). In the system we also established a range where the PID will not take action (set point 109 and range 5 under our set point).

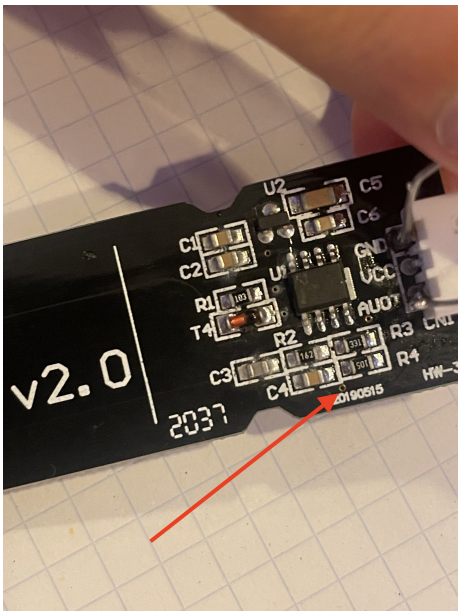

Figure 3.1: hole under resistors

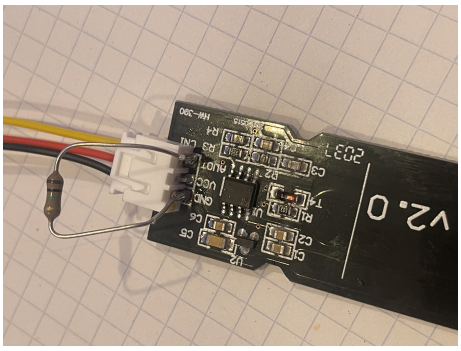

Figure 3.2: Resistors between GND and AOut

3.2 Arduino Cloud

We wanted to display all the sensor data and also warning messages on the Arduino cloud user interface. But in the free Arduino Cloud account there is a limitation of 5 variables which could be displayed. We solved this problem by deleting the Gauge for the temperature sensor and implemented it directly into the messenger. This means the user gets a message with the current value of the temperature and humidity in the room where the plant is located. In the actual code, the message for the temperature and humidity are showed each hour and not each second as before.

3.3 PID Controller

It took a lot of time to find the right configuration for the PID controller.

Table 3.1: Pid in Arduino

Kp	Ki	Kd
10	2	0

The term Kp defines how much the system will power the water pump. The term Ki defines how slow the controller reacts to changes based on the moisture. The term Kd means that the controller controls dampens oscillations better.

The soil moisture sensor needs time for detecting a value and knowing that the pump should not be activated too strong. The changes were based on it. Knowing the influence of these factors, we tried to provide a solution trying different configurations.

Table 3.2: Input and Output mapped for the PID case

Dry	Wet	Water
750	570	550
85	109	120

After many tries we defined kp=10, ki=2 and kd=0 and setpoint=109. The challenge with pid library we used and our system was the analog soil moisture sensor. The soil moisture sensor delivers values between 0 and 1023, but the output value from the library can only work between the values 0 and 255.

The output of the PID controller will be used as PWM value that will be used for the water pump, this value is defined for the value that the soil moisture sensor read(Input of the PID controller). To solve our problem, we mapped the analog

values between 0 and 1023 with the values between 0 and 255, to knowing what will be the output. The Table 3.2 has the mesaured values.

Chapter 4

Changes to the original proposal

We reworked the project with an PID controller which consistently measures the moisture of the soil and when needed activates the water pump. Additionally we implemented a limitation for the manually activation of the water pump which is triggered on the Arduino cloud interface.

4.1 PID

The PID controller continuously calculates the error as the difference between a setpoint and the measure of the process. In our project the setpoint defines the perfect moisture of the soil that we want for our plant. The input of the controller are the measurements of the soil moisture sensor. The output represents the signal for the water pump. It consist of a proportional- (P), an integral - (I) and a derivative part (D).

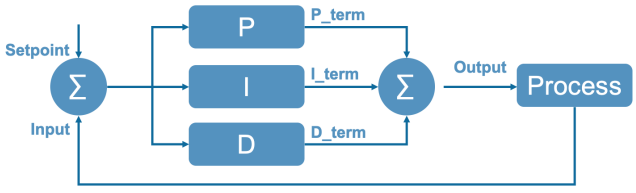

Figure 4.1: PID Controller

4.1.1 PID in Arduino

In our programm we implemented the PID controller with the PIDv2 library by Brett Beauregard. At the beginning a PID instance is created and the variables and

parameters like setpoint, input, output, kp, ki and kd must be defined. The value of the parameters kp, ki and kd were defined by running the finished program until it shows the best responses.

4.2 Manually activation of the water pump

By pushing the Motor ON/OFF button on the cloud interface the water pump starts running for a short period of time. The user should not be able to activate the pump permanently. Therefore we implemented a limitation by comparing the actual time from the internet with our defined reset time for the limitation. We defined the reset time with 00:15:00. With each activation of the Motor ON/OFF button the number of maximal pushes decrease by one. After 5 times the button won't activate the water pump any more. After the requested online time hits the reset time the limits for the button pushed go back to 5.

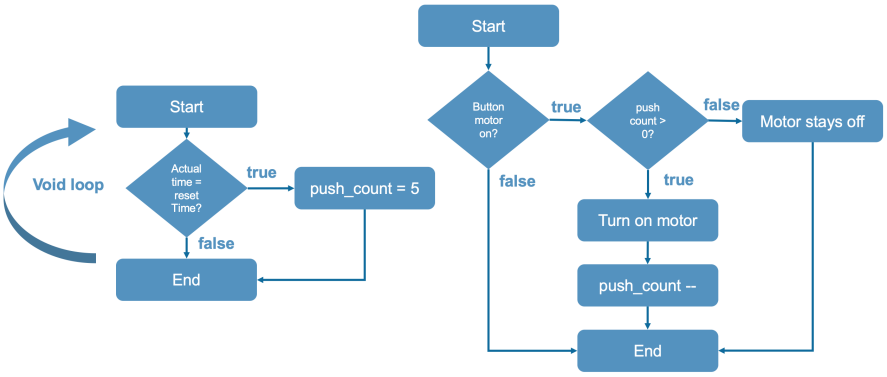

Figure 4.2: Flow chart limitation manually motor on

4.3 Temperature- and humidity sensor (DHT22):

In our original project we implemented the possibility to adjust the watering sequence when the plant is located in an environment with temperatures above 25 degrees. We removed this feature since the PID controller is continuously measuring the soil and then drive the water pump. The unique functionality of this sensor is showing the current and specific temperature and humidity to the user each hour(XX:10:00).

4.4 Timeline

In the timeline we defined the tasks we had in our project. Since we had a grade discussion on 5th of July we had to implement new features like the PID controller, the limitation of manually activating the water pump, showing the message of the temperature and humidity each hour and that the motor will turn off slowly if the container has not enough water. At the end we created a new presentation and video of the project and updated the technical report.

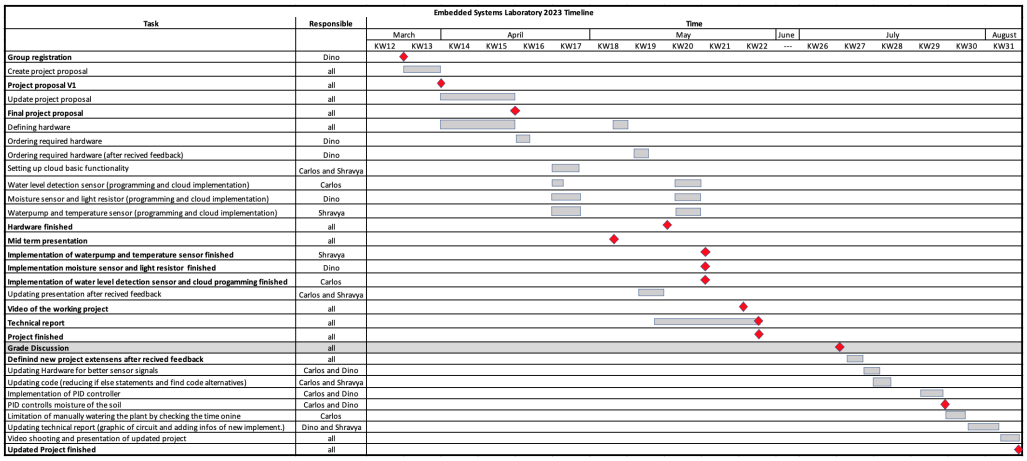

Figure 4.3: Timeline