An IOT Solution for Plant Irrigation

A. Zajac
University of Portsmouth
Portsmouth, UK
UP933019@port.ac.uk

***Abstract*—The quantity and timings of water a plant receives is critical to it's survival. This paper will explore an IOT solution for irrigating flora from anywhere in the world and why one would be useful.**

***Index Terms*—IOT, smart, agriculture, irrigation, internet**

I. Introduction

This report will cover the development of an internet-of-things enabled plant irrigation system. This system would provide the ability to control a water dispensing unit both automatically, and manually without the user needing to be present.

This concept is widely used in the commercial agricultural industry where controlled amounts of water need to be dispensed over a large amount of flora (Munoz-Carpena et al., 2003) and (Rajpal et al., 2011). What this report will explore is how practical and appropriate would a system like this be low-scale such as a home or office setting. In order to evaluate this, the following sections of this paper will cover the design process and implementation of a working prototype.

A. Relevant Scenarios

A home scenario would benefit from this concept if the user is away for an extended period of time such as for a vacation as they would be able to water their plants from anywhere in the world. There exist current solutions for this but each have significant drawbacks. The most popular product is a glass bulb that is planted into the soil and slowly releases water at a steady rate. The drawbacks of this solution is that the watering rate is both variable and uncontrollable, therefore little to no precision is possible (Courtney, 2022).

An office or large-home scenario would benefit from this concept if the office were to have too many real plants for anyone to frequently manage. For this scenario, the product would bridge the gap between small-scale home solutions and large-scale industrial solutions. For it do be viable in this scenario, the user would need to have the ability to operate and automate multiple units at once.

B. Prototype Objectives

Therefore. in order for this concept to be considered viable, it would have to meet a set of objectives:

- The watering rate and frequency should be able to be fully controlled.
- There should be a sensor measuring the room's temperature and humidity.
- There should be a sensor measuring how much water is available.
- The user should have the ability to control multiple units.
- The device should be able to automatically release water.

C. Deliverables

This project consists of three artifacts; this report, a video demonstration of an implemented prototype (see appendix A), and source code for any implemented software (see appendix B).

II. Hardware Design

A. Form Factor

The form factor of the device is not a focus of this paper, although there should be a general aim to make it as compact as possible. A compact design is is a bit of a juxtaposition to the concept of a prototype but size of hardware components still should be taken into consideration.

B. Computation

The device will require some sort of a computer for it to function. This computer should be as compact as possible, whilst still having the ability to utilize a range of sensors and be fully programmable.

Hardware that would fit these requirements would be a micro-controller such as a model of Arduino or Raspberry Pi. Some of the most suitable micro-controllers would be the Raspberry Pi Pico, Raspberry Pi Zero, or Arduino Nano. This is due to their small size while still supporting a suitable amount of I/O and enough processing power.

The reason why this project will not use any of the aforementioned micro-controllers is due to availability; there is already an Arduino Uno immediately available for use and is still a perfectly suitable choice, only being slightly larger. The Uno has plenty of input and output pins, and is able to supply 5V to any wired components (see section II-G).

C. Water Source

There are two approaches for sourcing water for the device, one is to have an on-board water storage system, the other is to have the device attached to a plumbing network. The latter would be appropriate for larger-scale scenarios but in a home or office setting may often be impractical. An option would also be to have the water source be interchangeable between the two (see section V-C) - this would require a lot

more engineering and is most-likely not worth the hassle over choosing the more suited water storage solution.

Settling for the water storage option, it would be important to define a suitable capacity. The reason why this won't be taken into consideration is due to the focus of this project being on the pure functionality of the device.

D. Water Dispensing

The most reliable method of releasing water from the proposed system would be to use a solenoid valve (Mercer, 1969).

A cheaper and more accessible solution for this could be to use air pressure. As discussed in section II-C, a container is going to be used to store the water. Therefore, if the water-filled container contains air with a lower pressure than the air outside the container, the water should remain in the container. This pressure is created within a container if it's only opening is below the water line, then the water trying to flow out will create a lower air pressure above it and that pressure would keep it from flowing out. This pressure can reach equilibrium by creating another opening above the water line, allowing the water to flow out (see figure 1).

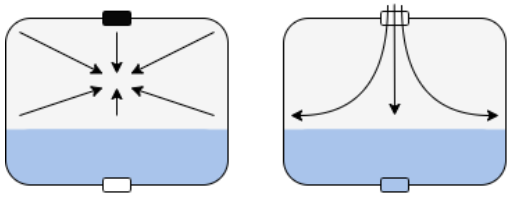

Fig. 1. Container Dispensing, left: off, right: on

This opening could be open and shut using a simple servo motor which would be ideal due to its small size, low cost, and simple programming.

It should also be noted that the opening for the water has to be small enough for the water-tension to not break, as this would result in pockets of air entering through the water outlet hole.

E. Heat-Index Sensors

The reason measuring room heat and humidity was planned for the prototype was to give the use an idea of the conditions the plant is currently in (if for example they were on vacation). This would allow them to judge if the plant requires more or less water than usual, or even if the plant will survive.

The sensor does not have to be precise for this use-case, and the options for cheap and basic sensors boils down to the DHT11 and its successor the DHT22. Table I presents data from Dejan, 2016 of how the two sensors compare with each other.

From table I it can be seen that the DHT22 sensor has superior temperature and humidity ranges, accuracy, and a faster sampling rate. As enticing as these may seem, it would not provide the product with much benefit as the operating conditions would not be this extreme. Therefore, the lower-priced DHT11 would be a better option in this case.

	DHT11	DHT22
Temperature Range	0–50°C/±2°C	-40–125°C/±0.5°C
Humidity Range	20–80%/±5%	0–100%/±2-5%
Sampling Rate	1Hz	0.5Hz
Operating Voltage	3-5V	3-5V
Price	£4	£8

TABLE I
COMPARISON BETWEEN THE DHT11 AND DHT22

F. Water-Level Sensors

There does not exist a large variety of water-level sensors that would be suitable for this project. One that is frequently used for other similar projects involving programmable micro-controllers is the sensor known as "Water Sensor". This sensor consists of parallel power and sense traces and the sensor acts as a resistor – with the resistance inversely proportional to the height of the water ("Water-Sensor-Interface", 2022). It is a cheap and basic sensor which is ideas for its use within this project

The reason this sensor is required for the prototype is to measure the water level of the container (see II-C). This would make it possible for the user to see how much water the device has left stored inside. This was originally the plan although section V-B1 discusses on why this was altered during the implementation process.

G. Power Supply

It is important to select a power supply that will provide enough power to all the components needed for the prototype. Table II presents the operating voltage of all required electrical components for the planned design. It also should be noted that all components mentioned operate under DC current, including the Arduino Uno itself.

Component	Operating Voltage	Source
Arduino Uno	6-20V	"Arduino-Store", n.d.
SG90 Micro Servo	4.8-6V	"SG90-Datasheet", n.d.
DHT11 Sensor	3-5.5V	"DHT11-Datasheet", n.d.
Water Sensor	5V	"Water-Sensor-Datasheet", n.d.

TABLE II
COMPONENT POWER REQUIREMENTS

From table II, it can be seen that all of the components can have their power wired from the Arduino Uno because of its 5v output pin ("Arduino-Store", n.d.). The Arduino itself can be supplied with power from any DC power supply that can output 6-20v through a 5.5mm diameter cylindrical plug with 2.1mm pin hole ("Arduino-Power-Input", 2022). Therefore, the power supply selected for the project is an AC-DC wall adapter with a 9V 1A output - which will also provide enough amperage.

H. Cost Estimation

In order to estimate the total cost of the prototype, table III breaks down the cost of every component that is planned for implementation. All prices shown are averages sourced from a variety of online stores, rounded to the nearest £1, and in GBP. The pricing for non-electronic hardware is not discussed as their impact would not be significant compared to the electronics.

Component	Cost Estimation
Arduino Uno Rev3	£20
SG90 Micro Servo	£3
DHT11 Sensor	£4
Water Sensor	£3
Total	£30

TABLE III
COMPONENT COST ESTIMATIONS

From these estimations, the full prototype manufacturing price should not exceed £35. This price could be drastically reduced by replacing the Arduino Uni with a much cheaper board that would still be viable, section II-B discusses why this did not happen.

Some components are possible to be purchased at lower price points when purchased in bulk; this will not be taken into consideration for this project as only a one-off prototype is being implemented.

I. Wiring

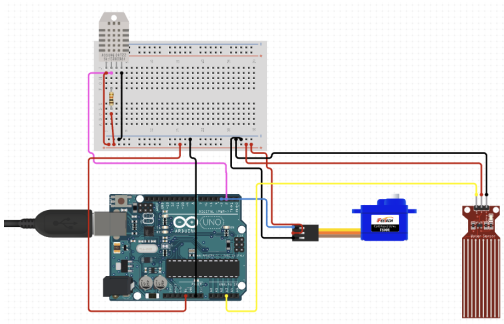

Fig. 2. Wiring Schematic

III. Software Design

A. Firmware

Due to the use of an Arduino Uno micro-controller (as discussed in section II-B), the firmware can be written in either C or C++ so that the AT-chip on the board can assemble it into byte-code (“Arduino”, 2022). It is possible to write code in other languages that is able to be compiles by the chip (such as Python or Java), although the C and C++ libraries are the most widely used and have no drawbacks for this project.

In order to develop the firmware code, Arduino provides an IDE that is suggested for Arduino-related development (“Arduino-Software”, 2022). It is a viable option for an IDE, although there does exist a plugin for the VS Code IDE which may be a better option since it would benefit from all the VS Code features.

B. Communication Protocols

There is a large selection of communication protocols that are viable for use in IOT devices with the most widely used being WiFi and Zigbee (Eshghi, 2021). WiFi would be the most ideal candidate as it is affordable, well protected, and has universal compatibility. The downside to using WiFi would be its high power consumption compared to a protocol such as Bluetooth LE or Radio-Frequency. This would not be a downside in the case of this prototype due to the use of a wall adapter as a main power source (see section II-G).

As will be seen in chapter IV, WiFi is not used in the developed prototype. The reason for this is the micro-controller discussed in section II-B does not have a built-in WiFi chip, and a stand-alone module was not available for the duration of this project. Therefore, this implemented prototype will use the serial communication protocol. This will be used via a universal serial bus for the micro-controller to communicate with a testing computer running the application (see section III-C).

C. Control Software

An outstanding user interface and user experience design is not required for this project to meet its objectives. Therefore an application designed to just get the job done is sufficient. The requirements for this application are:

- Display the temperature and humidity of the room the device is in.
- Display the read-out from the water sensor.
- Allow the user to manually trigger the irrigation process.
- Allow the user to enable and disable automatic irrigation.
- Allow the user to change the frequency of automatic irrigation.
- Communicate with the micro-controller via serial.
- Run on MacOS (the OS of the testing computer).

The Tkinter GUI library for Python would be an excellent choice as it is able to meet all of the mentioned requirements; as well as being simple to learn and implement. The Serial library for Python would also be required for the application to send and receive data from the Arduino Uno.

IV. Implementation

A. Hardware

The first step with implementing the prototype was to assemble the hardware. Figure 3 shows the initial construction, with all components attached to the Arduino Uno via a mini breadboard.

From this, a more final assembly was able to be constructed. The hardware was mounted to a water storage container which was a washed out olive oil bottle (extra virgin) - with both

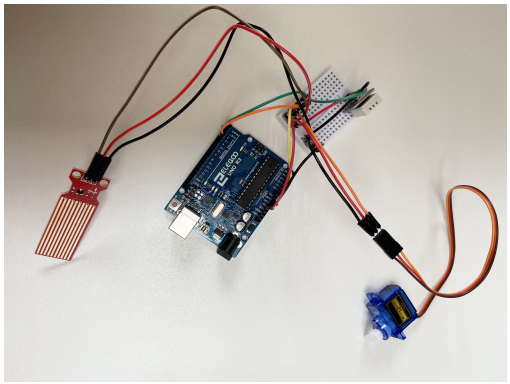

Fig. 3. Hardware Assembly

openings created using surgical-precision kitchen knife skills (see figure 4). At this point there was an issue with mounting the water level sensor, more information on this is discussed in section V-B1.

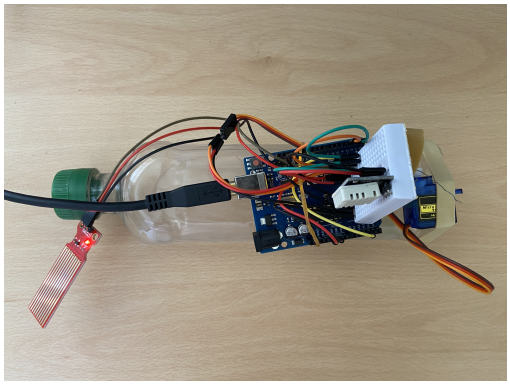

Fig. 4. Final Assembly

After final assembly, a set of brief test were conducted. The only notable issue was with the air intake mechanism - this is discussed in full in section V-B2.

Fig. 5. Assembly Presented in Intended Scenario

B. Firmware

The goal of the firmware is to output data from the sensors, and to take an input for when the water should be released. Therefore the code is simple and does not require a lot of computational power. In the source code (see B), all blocks of code are annotated with their functions.

The Servo library was used in order to provide better control for the servo motor, and the SimpleDHT library was used to interface with the DHT11 sensor. Variables are used to store the newest set of data retrieved from the sensors - this is in case a sensor were to not give a new piece of data during a refresh cycle, the program will still have some data to send to the app. The refresh cycle time is determined by the slowest component, in this case it is the DHT11 sensor which has a 1Hz sampling rate and therefore a 1500ms delay is used (1.5x higher that the DHT11 refresh rate).

The way that the firmware sends the sensor data through serial is by sending a character followed by the data. For example the temperature data would be sent as “t24” or humidity as “h40”. The reason for this is so that when the application reads the incoming data, it can tell what type of data it is. The same is done the other way, when the application calls to trigger the watering procedure, it prints a ‘a’ character to the serial bus that the firmware picks up.

C. User-Facing Software

As discussed in section III-A, the application is developed using Python, alongside the Tkinter library for a GUI and the Serial library for serial communication. The user interface of the application is constructed using Tkinter widgets and can be seen in figure 6.

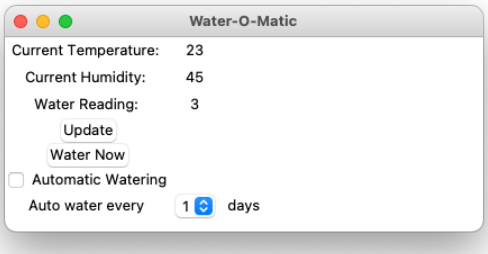

Fig. 6. App GUI

The data that the application receives are decoded (see firmware implementation for more details) and are shown to the user next to their corresponding descriptors. During the implementation there was an issue encountered with this and is discussed in section V-B3.

V. Conclusion

This report was written to explore the viability and practicality of an IOT plant irrigation system. Even with the issues encountered during the design and implementation process,

the project resulted in a somewhat working prototype that met the initially set objectives.

Part A of this chapter will discuss and evaluate the implemented prototype. Part B will discuss what issues were encountered and what ways they have been or could of been solved. And part C will discuss any improvements that could be make to the prototype as well as the future work that can be performed from this report.

A. Prototype Evaluation

Section I-B created a set of requirements that the prototype should meet in order for it to be deemed successful.

- The watering rate and frequency are able to be fully controlled.
- There is be a sensor measuring the room's temperature and humidity.
- There is be a sensor measuring how much water is available but is used in a different way (see section V-B1).
- The user does have the ability to control multiple units (using various baud rates for the serial connection).
- The device does be able to automatically release water.

B. Issues and Solutions

Throughout the implementation of the proposed solution, unexpected issues were encountered. This section discusses these issues and any solutions identified.

1) Water Sensor: An issue that became present with the water sensor was that there was no suitable way to mount it within the water storage container. This meant that the sensor was not able to be used to measure how much water was left in the container as discussed in section II-F. As a solution to this problem, it was decided to pivot the use for the sensor from measuring the water level to measuring the water in the soil. There exist sensors that have been designed to measure soil water contents, but none were available at this stage.

2) Water Dispensing Solution: Although the water dispensing solution discussed in section II-D would work well on paper, when implemented in IV-A it faced some issues. It has problems with reliability, in some cases it does not properly block the air intake, letting the water flow out when not wanted. In some cases – when properly blocked – it would still let in small amounts of air. Both these problems boil down to poor engineering skills and a lack of materials to create a proper mechanism.

3) Data Refreshing: When implementing the user-facing application, there was an issue with automatically updating the data values received from the firmware. This issue arose from the constant looping procedure interrupting the Tkinter builder loop that was running on the same thread. The best solution to this was to run each piece of code on a separate thread but for simplicity, an "Update" button was implemented to manually refresh the values.

C. Future Work and Improvements

1) Form Factor: The form factor and general construction of the hardware has a lot of opportunities for improval. A smaller or even custom PCB along with properly sized wires would drastically reduce the overall size of the electronics.

2) Water Management: The design could be improved by allowing the user to attach it to a constant water supply such as plumbing. More importantly, it could be improved by designing a better method of dispensing the water onto the soil.

3) Hardware Status Indication: The device could have some sort of status indication. This could be either something simple such as an LED light that turns on when the device is set up and ready to go, or an LCD display that shows more information.

4) Software Improvements: Although the focus of this paper was not on the user-facing software, it leaves much to be desired. The application presented is very minimal with pure functionality placed above everything else. Future work could work on improvements to this, especially UI and UX improvements with automatic data refreshing. Another option could be setting up integration with a 3rd party smart home service such as Google Home or Apple Home-kit.

5) Power Supply: A future iteration of the developed prototype could use a battery as a portable power supply. For this to be viable, components that use less power would have to be used such as the micro-controller discussed in section II-B and a communication protocol such Bluetooth LE (see section III-B).

6) Communication Protocol: Section III-B discusses the reasoning behind the use of a universal serial bus for this prototype. This would never be viable for a real product and a wireless protocol would have to be used. Further iterations of this idea would need to have a Wifi, Bluetooth, or Zigbee module as a replacement for the USB.

REFERENCES

"Arduino". (2022). https://en.wikipedia.org/wiki/Arduino

"Arduino-Power-Input". (2022). What power supply can i use with my arduino board? https://support.arduino.cc/hc/en-us/articles/360018922259-What-power-supply-can-I-use-with-my-Arduino-board-

"Arduino-Software". (2022). https://www.arduino.cc/en/software

"Arduino-Store". (n.d.). Arduino uno rev3. https://store.arduino.cc/products/arduino-uno-rev3?selectedStore=eu

"DHT11-Datasheet". (n.d.). Dht11 humidity & temperature sensor. https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

"SG90-Datasheet". (n.d.). Servo motor sg90 datasheet. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

"Water-Sensor-Datasheet". (n.d.). Water sensor module user's manual. https://curtocircuito.com.br/datasheet/sensor/nivel_de_agua_analogico.pdf

"Water-Sensor-Interface". (2022). How water level sensor works and interface it with arduino. https : / / lastminuteengineers.com/water-level-sensor-arduino-tutorial/

Courtney, A. (2022). How long do watering globes last? - smart garden guide. https://smartgardenguide.com/how-long-do-watering-globes-last/

Dejan. (2016). Dht11 & dht22 sensors temperature and humidity tutorial using arduino. https://howtomechatronics.com / tutorials / arduino / dht11 - dht22 - sensors - temperature-and-humidity-tutorial-using-arduino/

Eshghi, B. (2021). Iot top 10 iot communication protocols in 2022. https : / / research . aimultiple . com / iot - communication-protocol/

Mercer, J. (1969). Paper 12: Reliability of solenoid valves. *Proceedings of the Institution of Mechanical Engineers, Conference Proceedings*, *184*(2).

Munoz-Carpena, R., Bryan, H., & Klassen, W. (2003). Automatic soil moisture-based drip irrigation for improving tomato production. *Proceedings of the Florida State Horticultural Society*, *116*.

Rajpal, A., Jain, S., Khare, N., & Shukla, A. K. (2011). Microcontroller-based automatic irrigation system with moisture sensors. *Proceedings of the International Conference on Science and Engineering*.

APPENDIX A
VIDEO DEMONSTRATION

This project includes a video demonstration of the developed prototype as one of its deliverables. This video can be viewed in the ./Video folder of this submission.

APPENDIX B
SOURCE CODE

This project includes the source code of all developed software as one if its deliverables. This source code can be viewed in the ./Source folder of this submission.