Automated Plant Watering System with Arduino

Nicholas Lau
000377939

Automated Plant Watering System with Arduino

Abstract – The goal of this project was to understand the principles of electrical and computer engineering by working with automation to design an efficient system. Automation is a helping hand in our society that ensures things run smoothly and efficiently without human intervention, letting people do things that require their presence. Using an Arduino, we created a smart plant watering system by hooking up the sensory components (moisture sensor) and mechanical components (pump) via an NPN transistor configuration to the core of the system (Arduino). The pump will operate only when the sensor detects that the soil is has reached a programmable threshold, ensuring that the plant stays alive while also saving water. When compared to other solutions on the market, our solution is lower-costing than other smart systems while also being more intelligent than traditional timer systems. While the Arduino solution suffers from calibration issues, it has the potential to save significant amounts of water. This system can be taken further by implementing it on a larger scale, integrating it with other smart systems such as the Internet of Things and expanding its functionality.

Keywords - Home automation, Automatic control, Microprocessors, Water conservation

I. Introduction

In the fast paced world today, time and resources are a valuable commodity that automation can save. Automation, the use of control systems that automatically operates other systems, is an innovation that allows for little to no human intervention. From this, systems can operate more efficiently because less guesswork and human error can occur since most automations are open, connected systems. Another benefit of automation is that it allows for people to focus their efforts elsewhere on more critical tasks that require their presence, or simply give them extra downtime to relax. Today, electronics and data are the biggest forms of automation used widely across many industries, from data processing to manufacturing. The Internet of Things (IoT), systems that are connected to the Internet and each other to enable enhanced interactions, is projected be “50 to 100 billion devices by 2020” [1]. With more connected devices and systems, automation will be smarter on utilizing resources and making smarter decisions based on a given situation. Through automation, the systems of tomorrow will be more efficient and intelligent, reducing the burden of global issues such as climate change.

Our goal of the project was to understand what goes into electrical and computer engineering through automation and through this project, how to design an efficient system. As we are going into Simon Fraser University, having a better view of what these fields will achieve allows us to make better decisions as we go further in our studies in becoming engineers. In regards with the project, the creation is an automated plant watering system using an Arduino, and the criteria were affordability, water-saving and automation; with little human intervention, it would be able to operate efficiently on its own. The moisture sensor and Arduino are the “smart” aspects of this project as the system will only dispense water to the soil as needed, balancing the plant’s survival with the conservation of water.

In this report, the project’s components and concepts (Section II), the overall design (Section III), the progress of the project (Section IV), the implementation of the project and findings (Section V) will be discussed.

II. Background

A. *General Automated Watering Systems*

Today, automated watering systems on the market use irrigation with three types of operation: timer-, sensor- and/or smart-based. All of these operations are set-and-forget systems; once programmed, they can operate on their own. These types are more effective than hand watering as they don’t require human-labour and can water at the source: the roots. However, most solutions on the market come with significant trade-offs in terms of effectiveness and price.

Timer-based Systems

The most basic type of watering systems is timer based. Using a built-in timer, the system runs water through the system at specified intervals. While it is serviceable, the timed nature means that it can often waste water by delivering too much water due to external environmental changes, such as rain and humidity. As well, these systems are often made cheaply to entice buyers, meaning that the quality of the parts and pump also take a hit.

Timer- and Sensor-based Systems

The next type of water systems is based on using sensors in conjunction with timers to determine the best time to water the soil. When it rains, the programmed intervals are delayed based on the connected sensors to save water and the plants. These systems are built well for outdoor and indoor environments, but are considerably more expensive than timer exclusive ones as the controller and moisture sensors are sold separately.

Sensor- and Smart-based Systems

With the Internet of Things, these smart watering systems are based on leveraging sensor readings from the soil and weather forecasts from the Internet to only water when necessary. Due to their IoT nature, users can wirelessly communicate it to program and monitor the system. Moreover, it can also communicate with other IoT connected devices to enable further interactions and savings. However, these systems cost more than any other type of system, not to mention that the parts are sold separately. In addition, there are security and privacy concerns regarding IoT devices [1].

B. Controller

With an automated watering system, the controller is designed to handle the operation of the system. This component is critical in telling how the system should operate under specific circumstances; water should be dispensed only when the soil is dry. The component will typically have a microcontroller that will be programmed to dictate how things should communicate. To make it "smarter," more sensory components must be connected to give the controller more knowledge on the current. It may also be equipped with wireless antennas to communicate with external sources such as the user, the Internet and other devices.

C. Sensor Systems

To add some level of intelligence to the controller, sensors are used to give it more "observability" on the environment. In doing so, it can better determine the next course of actions and improve overall performance. In automated plant watering systems, moisture sensors are most commonly used among them in tandem with interval timers. Smarter systems typically use the Internet to "sense" the current and upcoming weather conditions through forecasts.

D. Actuator

To allow water to run from the source to the soil, an actuator is needed to change the state of the soil. A pump, which is mechanical, takes the signal from the controller and runs itself so that the soil remains moist.

E. Programming Language

The basis of the operation of a microcontroller comes from the programming language it is built upon. The language and its code are what determines how the microcontroller, will operate the system. There are two types of languages for executing instructions: interpreted and compiled.

Interpreted Language

Interpreted languages have the advantage of having shorter development times and being more flexible with changes immediately taking place. However, it requires more "machine resources" because interpreted languages are less efficient since they must be translated into machine code [2]. Due to their non-static nature, they are also less reliable. Languages using interpreted execution are: Python, Ruby and MATLAB.

Compiled Language

On the other hand, compiled languages are more reliable and efficient because their instructions are built beforehand with static data checks. When compared to interpreted languages, they consume considerably less CPU time and memory [2]. However, this type is less flexibile and platform-agnostic, and often requires longer development times to achieve a similar end-result. Common complied languages include the following: C/C++, Fortran and C#.

F. Water Usage

Water is a precious commodity in many countries, and with only a small percentage of it being potable, the usage of this water is a hot topic with regards to the future. According to [3], "outdoor water use in the United States remains the largest end use of water among single-family residential customers." As such, keeping usage and conservation in balance is critical for this project.

G. *Circuitry Components and Concepts*

Arduino

Arduino as a whole encompasses the open-source hardware, software and community aspects. It is used to create DIY projects, proof of concepts and prototyping for a commercial product.

In hardware, various vendors create different boards based on designs created by Arduino for different form factors and purposes. This can vary in different microcontrollers, IO configurations and on-board components. For instance, the Arduino Lilypad can be used for wearables and fitness applications, while the larger Arduino Mega is for more complex and larger-scaled systems.

For the software, Arduino has an integrated development environment (IDE), the software that allows programming, that allows developers to use the C and C++ languages.

In the community, developers, hobbyists and manufacturers can share hardware and software designs to learn and improve.

Ohm's Law

Voltage is the "potential difference," or the difference in potential energy per Coulomb charge across a specified distance [4]. The SI Unit for voltage V is its name, voltage or V. This is often specified as:

$\Delta V = V_f - V_i$

$$K_f + qV_f = K_i + qV_i$$

Where Kinetic Energies K and charge q are grouped by initial i and final f.

In a circuit, voltage decreases when in a series circuit from the first to last component, while they will retain the same voltage in parallel.

Current is the "motion of charges" from a negative to positive pole (electron flow) [4]. The SI unit for current I is amperes or A. There are currently two types of current: direct (DC) and alternate (AC). The former sends a direct and constant flow of electrons, which is ideal for electronics as it simplifies the need for additional components and assures the preservation of data. The latter oscillates in a sinusoidal pattern [4]; it switches polarity and thus the direction of flow. One advantage of this is that the amount of energy lost due to heat and other factors is reduced considerably, meaning that it can still retain energy over longer distances better than DC.

For current, the same current is maintained across a series, but will split based on the resistance in the subsequent paths.

Resistance is the impedance of electron flow, which is affected by a variety of factors. The SI unit for resistance R is Ohms, or Ω. Resistance can be defined as [4]:

$$R = \frac{\rho L}{A}$$

Where length L, area A and resistivity, the intrinsic property of impedance of the material, ρ all affect resistance. Resistance should not be confused with resistivity as the former is an extrinsic factor based on the shape and size of the object.

In a series circuit, multiple resistors are equivalent to one the sum of all of them. However, resistors in parallel can be related with the following.

$$R_{total} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \cdots$$

Combining these three factors together, Ohm's Law tells the relationship between the three concepts. This is:

$$I = \frac{\Delta V}{R}$$

Resistors

Resistors are small electrical components that reduce current flow or voltage levels in a circuit. This is particularly useful when sensitive electronics with lower rated currents are utilized in a circuit. With Ohm's Law, the appropriate resistor can be found. Resistors have four coloured bands representing the resistance of the resistor and the accuracy of its resistance.

Pull-down resistors pull down the voltage to 0V when no other devices are active, cancelling out interference and electrical noise [5]. Pull-up resistors pull the voltage level up to the source when other devices aren't active [5].

Transistors

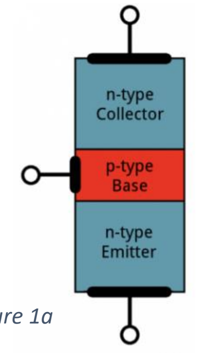


Figure 1a

Transistors are a critical component that allow for control as a switch or an amplifier. According to [6], there are two types: NPN, or PNP transistor. A transistor can be broken down into three different parts: collector (C), base (B) and emitter (E) as shown in Figure 1a.

For an NPN, positive voltage flows from C to E, with current flowing to B as shown in Figure 1b. As this current increases, it will conduct more and vise-versa [7]. For a PNP one, it is the exact opposite as seen in Figure 1c; voltage flows from E to C, with conductivity increasing as current pulled from B decreases.

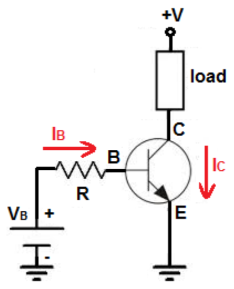


Figure 2b

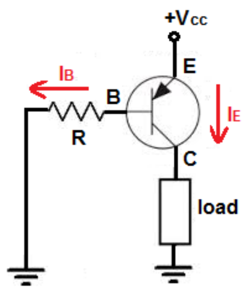


Figure 3c

Capacitors

This component is useful in that it can collect charge "relatively slowly" in which the energy can be stored and released very quickly [4]. As a result, this rapid release of energy is useful in keeping power during drops in voltage, defibrillators and flash photography [4]. Due to a capacitor's nature to resist change, they are critical in reducing voltage noise, or variations in voltage supplied in the circuit [4]. These parts are measured in Farads or F.

Ground

Grounding is a critical safety measure in protecting users and the electronic components themselves. In a circuit, fuses and breakers are used to switch off or disconnect the circuit, preventing further damage. The ground wire on the other hand works in tandem with these two measures by providing a way for the circuit breaker to trip the circuit and also as a way to pull the voltage down to zero, preventing electrical shock.

Microcontroller

This component is the core of the majority of electronic systems that contains the "peripherals, memory and a processor," or a system on a chip (SoC) [8]. Microcontrollers are programmable and can be used for a wide variety of purposes.

Digital, Analog and Pulse Width Modulation (PWM)

Analog signals are "infinite number[ed]" in values between a range of numbers [9]. While this allows for smooth transitions, it is also susceptible to noise. Most components in circuits are analog, but these kinds of circuits are harder to design versus digital ones.

Digital signals are finitely valued between a range that must step to reach a different level [9]. Unlike analog, digital uses a "binary scheme" [9].

PWM is a type of digital signal that switches between on and off at a specific interval to achieve a specific effect such as dimming or direction.

III. Project Design

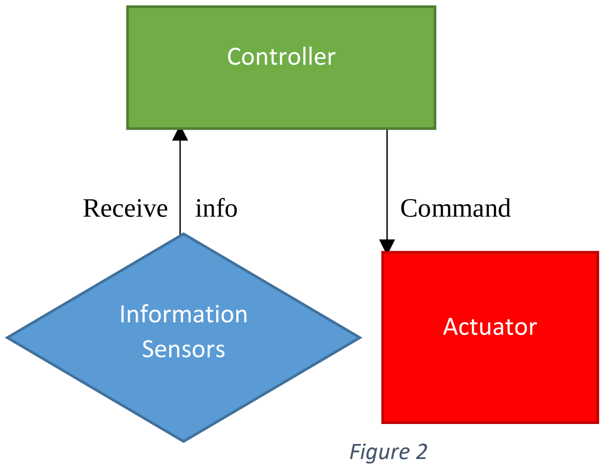


Figure 2

A. *Controller*

In the project, the controller will be the Arduino Uno R3, a pre-assembled board equipped with an Atmel ATmega328P microcontroller [10]. This microcontroller will be in charge of taking information from the sensors and then determining whether action needs to be taken via the actuator. This means that after a certain threshold of moisture, it will send a command to the pump to run until it is no longer needed. Figure 2 shows the relationship. On the board, it is capable of receiving and sending analog signals via conversion, digital signals including PWM, ground and voltages up to 5V as shown. If connected via USB, the user can see what readings the sensors are getting. See Appendix A for a diagram labelling the different parts of the Arduino.

B. *Information Sensors*

The information sensors are what will make the system "smart" by giving it insight on the environment's situation. In this project, a moisture sensor operating at 3.3-24V will be the component giving the controller information about the soil. Moisture readings are analog (0-255), but are converted to digital (0-1023) signals for the Arduino to understand in ascending order; the bigger the number, the dryer it is. A point to keep in mind is that the moisture sensor is not waterproof and as such, metal contacts should not be exposed to water.

C. *Actuator*

In the circuit, the actuator, or the pump, will be the mechanical component that changes the state of the soil. As this is a component that requires 6-12V, it cannot be hooked directly to the Arduino. This will be detailed in the next part.

D. *Connections*

To facilitate the communication between each component, they are connected with wires. This can be seen in Appendix B as a schematic and C in photos.

The information sensors are connected to the Arduino's ground and 5V rail as well as through analog pin A0 since the moisture sensor operates within the Arduino's limits.

Since the actuator operates at a higher voltage than the 5V the Arduino can deliver, it must be hooked up to a separate look with its own power rail connection. To allow the controller to manipulate the actuator, a NPN transistor connected via PWM and a resistor is used as a switch to control the amount of current passing through. In conjunction with this, a capacitor and resistor are implemented in parallel to reduce noise.

Should the user desire to know what the Arduino is doing, a Serial connection polling at a rate of 9600 bits/sec. to a computer via USB will give moisture sensor readings. This connection can also deliver power to the Arduino, but not the circuit that the pump exists in.

The project will be using Direct current (DC) as the project has electronic components that require a steady and constant current.

IV. PROJECT EXECUTION

A. *Components and Cost*

In the project, price is a factor that the Arduino platform excels in. By creating solutions through DIY (do it yourself) projects, a better understanding of circuitry and engineering can be realized. In Table 1, the core components of the project are shown and Table 2 shows the cost of increasing the scale per unit. Finally, Table 3 compares our project against commercial products on the market.

Table 1: Core Components

Component	Cost (CAD)
Arduino Uno R3	$33.00
Breadboard	$5.00
Total	$38.00

Table 2: Expandable Components

Component (per piece)	Cost (CAD)
Pump	$12.00
Transistor (NPN TIP131)	$1.50
Capacitor (105pF)	$0.08
Jumper Wires	$0.50
Diode (1N4740A) (per pc)	$0.32
Moisture Sensor (per pc)	$11.00
Piping (per metre)	$1.02
Total	$26.42

Table 3: Comparison of Solutions

Per pc	T [11]	T+S[12]	S+I[13]	S - Proj
Core	$99.99	$72.99	$279.99	$38.00
Expand	-12 pc-	~$50.00	~$50.00	$26.42
Total	$99.99	$129.99	$329.99	$64.42

Where T = Timer, S = Sensor, I = Smart

While our solution costs more than the timer-only product, the Arduino project is more scalable than the former while cheaper than the multi-operational based solutions.

B. *Arduino Programming*

The Arduino Uno was programmed with the Arduino IDE via USB with the C++ language.

During operation, the Arduino will be running in a constant loop; there are no ending states. When first powering on, it will first read the signal from the moisture sensor, and if the soil is dry, it will turn on the pump. If the soil is moist or the actuator is turned on or off, it will always wait 5 seconds to re-evaluate the state of the soil. A program flowchart is shown in Figure 3 and the full code can be found in Appendix D.

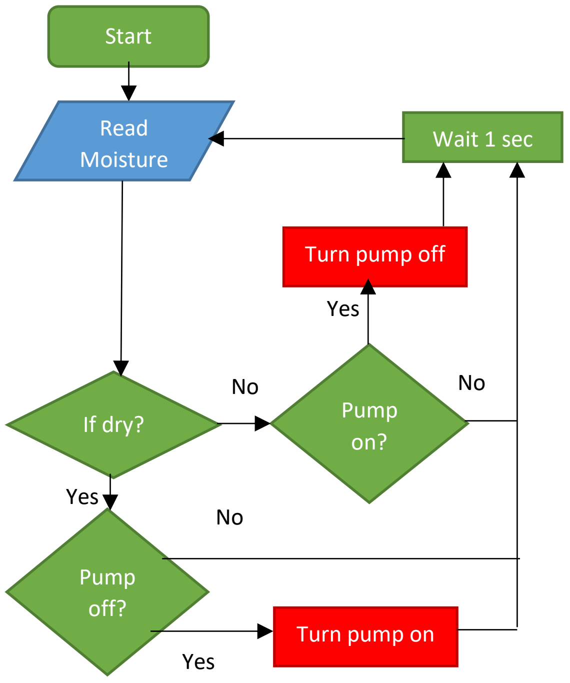


Figure 3

C. Watering System Operation

For the whole system to function, the actuator is connected to the water source on one end and the soil on the other via tubing. In the soil, the tubing has small holes to allow water to evenly distribute. For the source, we went with a water reservoir or cup for testing purposes. Appendix E shows the set-up.

V. RESULTS AND FINDINGS

A. *Results*

We found that with our system, the water is able to permeate into the soil better than hand-watering it since the tubing goes into the soil versus only being on the surface. As such, the Arduino project is more efficient with water since it can effectively deliver water at the source: the roots of a plant. It is also more cost-effective than solutions on the market because it is easily programmable and scalable with lower costing parts.

For the moisture sensor, it can be read properly by the Arduino Uno and can deliver the proper action based on that, although it requires calibration. During operation, we found the pump was still operating despite overflow; the water didn’t permeate the soil quickly enough for the sensor to pick up.

A point to note is that the system is not waterproof, meaning that extra care needs to be taken with the water reservoir and the exposed components. This is especially true with the moisture sensor and pump which have exposed prongs. The water exposure limitation means that our solution cannot be used in wet environments due to a risk of water damage.

B. Areas of Improvement

To adapt our project for more versatility, creating a solution to protect the hardware from water damage is critical; creating a case or enclosure would help to solve this issue. Soldering the components together instead of loosely connecting them onto a breadboard would also improve connection reliability.

To prevent overwatering, re-calibrating the sensor and pump would fix the issue. To do this, reprogramming the Arduino is necessary by slowing the speed of the pump through the transistor and re-adapting the readings from the moisture sensor.

Adding a connection aspect would keep the user in the know. This could be implemented with Wi-Fi or Bluetooth to utilize the Internet of Things. Using this, the operator could view historical information on the soil state and the activity of the actuator. Moreover, they could also manually run the pump or re-adjust the parameters.

To expand on the project, adding additional gardening functions would improve the utility of this project. Adding sunlamps and programming them at specific times of the day would aid in speeding up plant growth.

This project has many applications in terms of where an automated watering system would be most beneficial. The project could be scaled up for larger fields or greenhouses where human labour is time consuming and costly. Moreover, the upfront and maintenance costs with the Arduino are more affordable for small businesses and individuals versus commercial products. For dry areas where water is particularly lacking, this system would be able to efficiently dispense it only when needed.

VI. Summary

We were able to meet the goals of this project: to learn more about engineering, automation and circuitry while also designing an efficient system. While there are flaws with our solution, the creation of this Arduino-powered plant watering system has allowed us to better understand how hardware components and software interact with each other.

VII. Acknowledgements

We would like to thank Pooya Taheri Gharagozloo for the continued support and help in the progress of the project. Without his guidance, our progress would have been much slower than the timeline we set ourselves.

References

[1] K. Britton, “Handling privacy and security in the internet of things,” *Journal of Internet Law*, vol. 19, no. 10, p. 3 – 7, Apr. 2016. [Online]. Available: EBSCOHost, http://cclsw2.vcc.ca:2048/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=114521106&site=ehost-live . Accessed on: June 28, 2016.

[2] A. Ellero and P. Pellegrini, “Computer Language Efficiency via Data Envelopment Analysis,” *Advances in Operations Research*, vol. 2011, p. 1-14. Available: EBSCOHost, https://cclsw2.vcc.ca/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aci&AN=70865834&site=ehost-live . Accessed on : Aug. 20, 2016

[3] M. Shimabuku, “Impact evaluation of residential irrigation audits on water conservation in Colorado,” *Journal: American Water Works Association*, vol. 108, no. 5. Available: EBSCOHost, http://cclsw2.vcc.ca:2085/ehost/detail/detail?sid=5dd1a041-78d4-4e12-96d9-727ad7cf3f17%40sessionmgr102&vid=0&hid=118&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#AN=115135495&db=aci

[4] R. D. Knight, *Physics for scientists and engineers: a strategic approach 3rd edition.* Boston, Massachusetts, USA: Pearson Education, 2013, pp. 880-1034.

[5] J. Blum, Producer, *Tutorial 03 for Arduino: Electrical engineering basics*. [Online]. Jeremy Blum, 2011. Available: YouTube, https://www.youtube.com/watch?v=abWCy_aOSwY&list=PLA567CE235D39FA84&index=3 . Accessed on: July 11, 2016.

[6] HyperPhysics, “Transistors” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/trans.html

[7] J. Blum, Producer, *Tutorial 05 for Arduino : Motors and transistors.* [Online]. Jeremy Blum, 2011. Available: YouTube, https://www.youtube.com/watch?v=5bHPKU4ybHY . Accessed on: July 13, 2016

[8] Future Electronics, “What is a microcontroller?” n.d. [Online]. Available: http://www.futureelectronics.com/en/Microcontrollers/microcontrollers.aspx . Accessed on : Aug. 20, 2016.

[9] Jimbo, “Analog vs. digital” n.d. [Online]. Available: https://learn.sparkfun.com/tutorials/analog-vs-digital . Accessed on : Aug. 20, 2016

[10] Arduino, “Arduino - ArduinoBoardUno” n.d. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno . Accessed on : Aug. 20, 2016

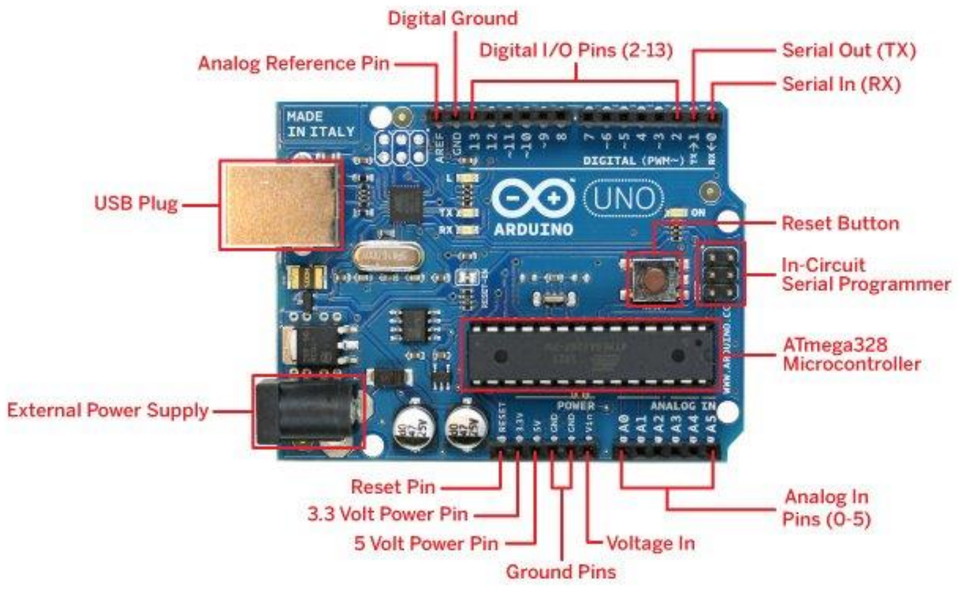
[11] Home Depot, “Flowerhouse Solar RainMaker Automatic Watering System,” 2016. [Online]. Available: https://www.homedepot.ca/en/home/p.solar-rainmaker-automatic-watering-system.1000748748.html. Accessed on : Aug. 21, 2016

[12] Home Depot, « Orbit Watermaster 6-stn Indoor Profes Controller, » 2016. [Online]. Available: https://www.homedepot.ca/en/home/p.6-stn-indoor-profes-controller.1000117594.html . Accessed on: Aug. 23, 2016

[13] Home Depot, « Rachio Iro 8 zone SMART Irrigation controller, » 2016. [Online]. Available : https://www.homedepot.ca/en/home/p.8-zone-smart-irrigation-controller.1000835544.html . Accessed on : Aug. 21, 2016

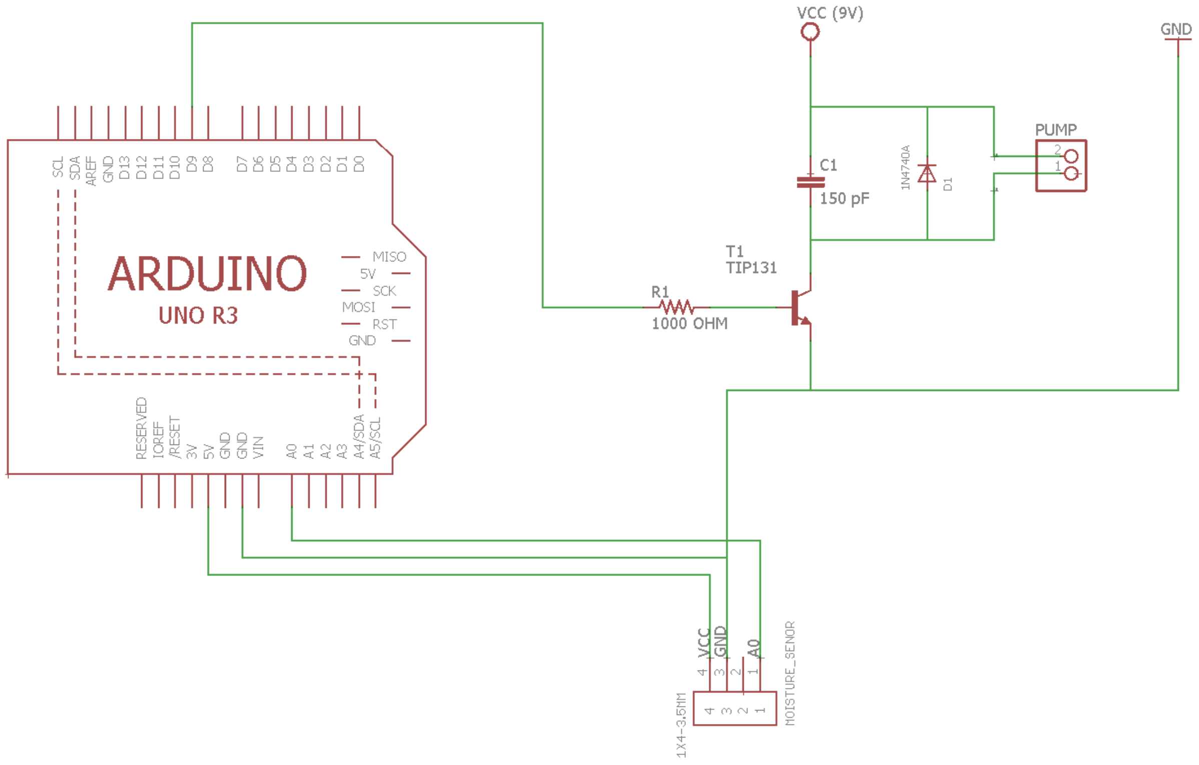
[14] RoboMart, Arduino Uno R3 Board, Image, 2015. [Online]. Available: https://www.robomart.com/arduino-uno-online-india . Accessed on: Aug. 18, 2016

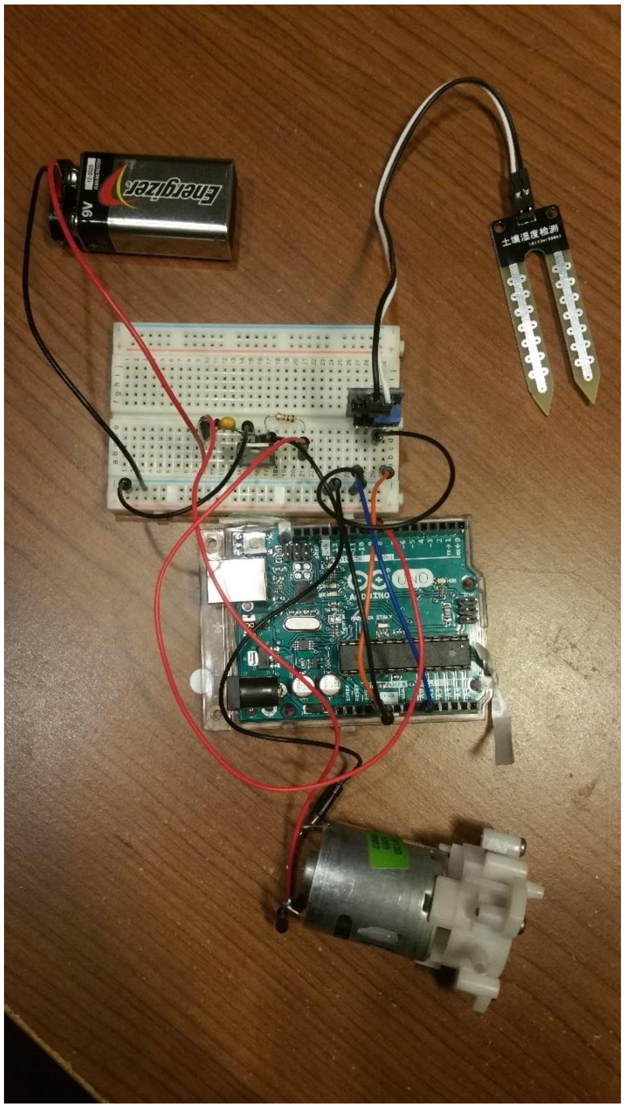
VIII. Appendix



Appendix A – Labelled parts of Arduino Uno [14]

Appendix B – Watering System with Arduino Schematic

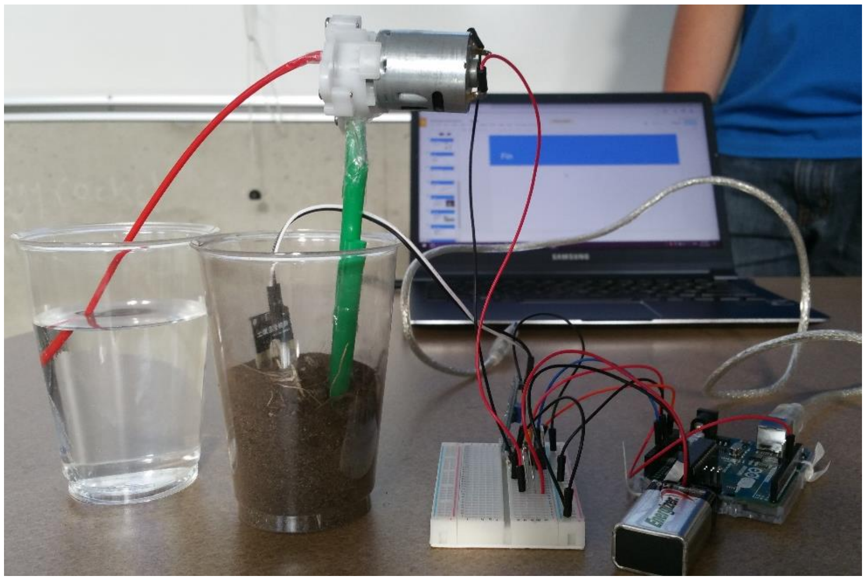




```
/* sensor reading in room = 676
   LOWER NUMBER = MORE MOIST
   NICHOLAS LAU 000377939
   LAST EDITED: AUG 5, 2016

*/

const int moisture_sen = A0;
const int transistor = 9;

void setup()
{
   pinMode(moisture_sen,INPUT);
   pinMode(transistor,OUTPUT);
   Serial.begin(9600);
}

void loop()
{
   int val = analogRead(moisture_sen);
   Serial.print("sensor = ");
   Serial.println(val);
   delay(1000);

   if (val < 675)
   {
      analogWrite(transistor, 200);
   }
   else
   {
      analogWrite(transistor,80);
   }
}
```

Appendix D – Arduino Uno code in C++

Appendix C – Wiring of the watering system



Appendix E – Full set-up of watering system