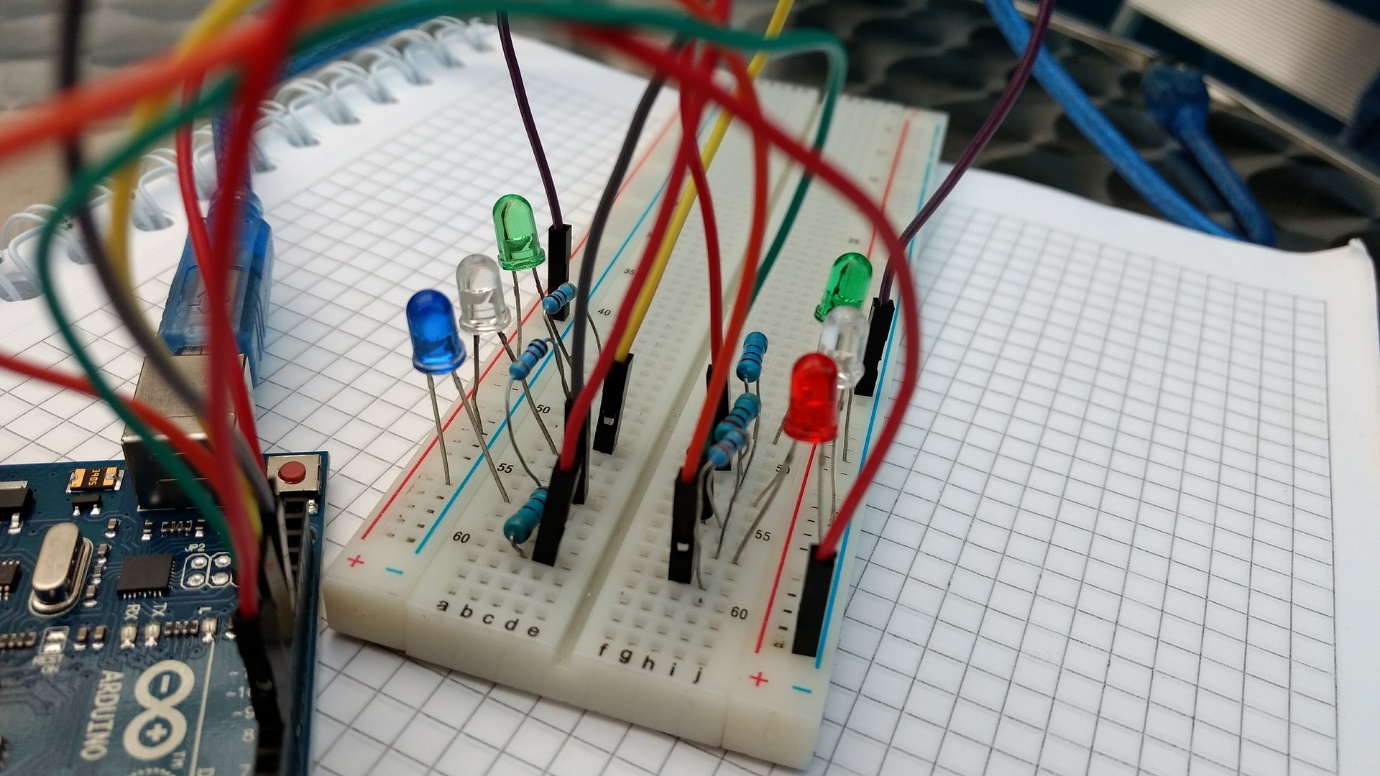
**TECHNICAL REPORT**

***Auto-Watering System***

Embedded Systems (ES1)

**

**LECTURER:** Gerald Hilderink

**DATE:** 29-03-2019

**AUTHORS:** Sam Jarvis - 3734811

Yasser Alhazmi - 405156

*Abstract*

The week 6 assignment proved more flexible than previous weeks which presented different problems. The goal of the final assignment was to use as many of the components from previous weeks as possible. An auto-watering system was constructed using the following parts: one button, two LEDs, one RGB LED, one potentiometer, one servo and one ultrasonic sensor. Some of the techniques learned were LEDs, buttons, button debounce, potentiometer, RGB, functions, correct scoping, state machines, serial communication, Servo, and ultrasonic sensor.

Contents

[Introduction 1](#_Toc4747301)

[Procedure 1](#_Toc4747302)

[Apparatus 1](#_Toc4747303)

[Design and Construction 3](#_Toc4747304)

[Body 3](#_Toc4747305)

[Circuit 5](#_Toc4747306)

[Software 6](#_Toc4747307)

[References 7](#_Toc4747308)

# Introduction

The auto watering system is the product of the week 6 final assignment. The aim of the assignment was to use as many of the components that featured in previous assignments, as possible. However, the practical purpose of the auto-watering system is to solve a real-world problem. The auto-watering system tends to pot plants with minimal user interaction. This means the user can configure the system as they desire and have little to no interaction with the system thereafter. The problem this system solves is neglection of plants due to more important day-to-day activities.

The system works as follows. When the power button is pressed, the system turns on. A green LED illuminates to visually represent this. If the system is on and the power button is held down, the system and LED turn off. Once on, the arm rotates from 0 degrees to 180 degrees at intervals of 20 degrees. At each iteration, it tries to detect a plant. If a plant is detected, the arm stops rotating, and the blue LED illuminates to simulate the system watering a plant. Once it has repeated this and watered all plants in a semicircle, the system waits for a user selected time. The use can determine this time by changing the value of a potentiometer. If the time is two hours, the system will wait at 180 degrees for two hours before repeating the cycle in reverse. The user selected time is illustrated by an RGB LED. The LED will cycle the RGB spectrum for the user selected time. For example, if the user selected time is two hours, the RGB LED will take two hours to cycle from green to red.

## Procedure

### Apparatus

* 1x breadboard
* 19x male-to-male jumper wires
* 1x Arduino Uno
* 2x LEDs
* 1x RGB LED
* 5x 470-ohm resistor
* 1x button
* 1 x 10kΩ resistor
* 1x potentiometer 1kΩ
* 1x ping sensor
* 1x servo
* 1x 3D printed design

Before starting the assignment, it was necessary to test all the apparatus separately to make sure all hardware was working as expected. To do this, a multimeter was set to the correct mode and connected across each component so that if there was a current present, it would flow through the component and the multimeter. If the multimeter made a beep noise, the component was declared working and was used in the system.

The procedure of this plant watering device is complex. The simplest approach to describe it is to follow the program beginning-to-end. The program begins by including the servo library[[1]](#endnote-1) and by following the conventions of the C programming language by defining functions above main. It then continues to declare and initialize the constant and global variables of the program. Pins are declared using the “const int” approach rather than the pre-processor directive “#define” because the former is the advised method according to the Arduino programming reference [[[2]](#endnote-2)]. The servo object is also declared as a global variable. It will be accessed by multiple methods in the program. In the “setup()”[[3]](#endnote-3) method, all pins were assigned an appropriate mode using the “pinMode()”[[4]](#endnote-4) method. In addition to this, the servo was attached to the appropriate pin using “Servo.attach()”[[5]](#endnote-5) method, and serial communication was initiated using “Serial.begin()”[[6]](#endnote-6)with the default baud rate of 9600.

The machine then enters the “loop()”[[7]](#endnote-7) method while still in turned off state. In the loop, the program checks whether the button is clicked. This check includes a button debounce function of 70 milliseconds to eliminate any possible bouncing effect. If the button is clicked, the method “turnMachine()” is called to turn the machine on. The method “turnMachine()” will check the “machineIsOn” variable status. If machine is off, the following steps are taken to turn the machine on:

1. Turn the green LED on.
2. Read the potentiometer value. This is to ensure persistence, that is, the user is not required to input the time interval each time the system is started. The value is mapped to hours and saved as a time interval that is later used as a delay.
3. Print that the machine is turned on via serial communication.
4. Change the machineIsOn variable to true.

After the machine is turned on, the program continues to the loop() method. A “while” loop is initiated if machineIsOn is equal to true. In the while loop, the method rotateServo() is called to rotate the servo. This method will start with angle of 0 degrees and rotate by 20 degrees each time the method is called. This is done by calling the servo.write()[[8]](#endnote-8) method from the servo library and supplying the angle to it as an argument.

After each call (a rotation of 20 degrees has occurred), it calls the method detectPlant() to determine whether the ultrasonic can detect an object (plant). The detectPlant() method uses the ultrasonic sensor to detect an object within the appropriate distance. This code was re-used from a previous assignment[[9]](#endnote-9). Based on the 3D design, the appropriate distance for a plant is 4-6 centimetres from the system. If a plant is detected within this distance, the servo will temporarily stop rotating to allow the system to water the plant. The blue LED will turn on to simulate watering.

When the servo reaches an angle of 180 degrees, that is, it’s maximum, it will stop for a certain amount of time before repeating the cycle in reverse. This is done by calling the method delayWatering(). The same method is called when the servo rotates back to an angle of 0, that is, it’s minimum. The amount of delay time between cycles is changed as follows:

1. The initial value of the potentiometer is read and mapped to a time in milliseconds when the device is turned on.
2. The current value of the potentiometer is read after each servo rotation in case the user changes the state of the potentiometer.
3. If the current value is equal to initial value, no action is taken.
4. If the current value is not equal to initial value, the initial value is changed to the current value.
5. This value is used as the amount of time to wait between watering cycles.

To convert the value read from the potentiometer to a value of time in milliseconds, the following steps are taken:

1. The potentiometer reading is mapped from a value between 0 and 1023, to a value between 1 and 12. Thus, the user can specify a time interval between 1-12 hours to wait in between each watering cycle.
2. The mapped value is converted to milliseconds by the equation: delayTime = mappedValue \* 60 \* 60 \* 1000. To further explain, a value between 1 and 12 is multiplied by 60 to convert it from hours to minutes. This is converted to seconds by multiplying by 60 again and finally, it is multiplied by 1000 to convert it to milliseconds. The reason for this is so that the native “delay()” method can be used.
3. To cycle through the RGB spectrum (from green to red) while waiting, the delay time is divided by the number of values in the spectrum (256 to include the 0). This provides the delay in milliseconds that the Arduino must wait before changing the RBG value. This is done in a method called “delayWatering()”.

The “delayWatering()” method will initialize the green colour channel of the RGB LED to 255 and red, blue, and counter to zero, because it will only cycle the spectrum after watering is finished. After the appropriate delay, green will decrease by one, red will increase by one, and counter will increase by one as well. The method “changeRgbByValue()” is re-used from a previous assignment, and it changes the value of an RGB LED based on three integers values red, green, and blue. The method loops until the counter reaches 255. At this point, red will be 255 and green will be zero. The loop will terminate because the delay time has ended, and the system will induce another cycle. That is, the servo will start rotating to water the plants again.

# Design and Construction

The system consists of three main parts namely, the physical body, the circuit and electronics and the software written in C. Each of these categories has a design aspect which is described in detail below.

## Body

The body of the system was specific to its purpose and thus had specific requirements. For example, the upper section had to rotate using the servo. So, the upper section needed a bed for the servo rudder to rest in. For this reason, the parts were designed using measurements from the electronic components and 3D printed.

The body consisted of the following parts:

1. The base.

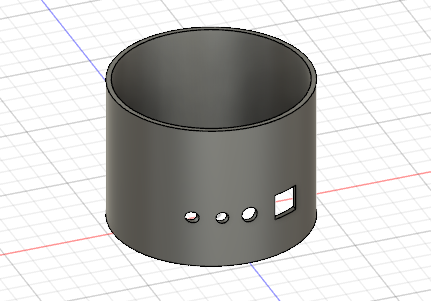


Figure 1 illustrates the 3D design of the base. This is the container for the system and has the required fittings for the potentiometer, two LEDs and the button. This is the biggest component because it is the structural integrity of the system.

1. The Lid

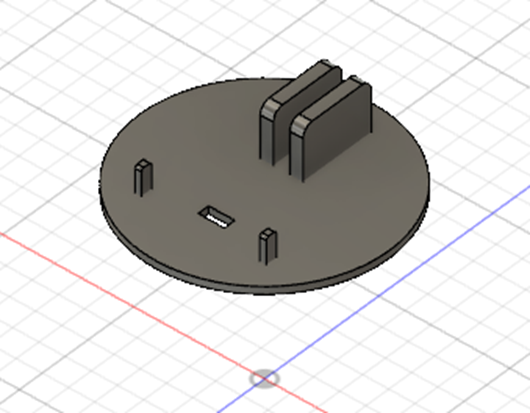


Figure 2 illustrates the lid of the system. The lid is the component between the solid foundation and the rotating section of the system. The lid has fittings designed for the ultrasonic sensor. It has a bay for the ultrasonic sensor to rest in as well has a space for its wires to connect through.

1. The Arm

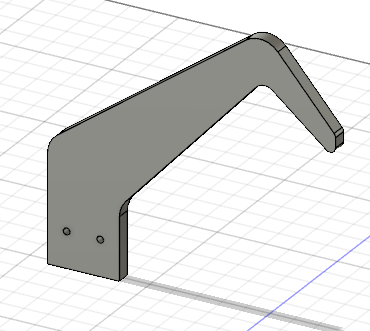


Figure 3 illustrates the arm. This connects to the lid and is responsible for watering the plants. The arm has two screw holes to secure it to the lid.

1. The Servo Bay

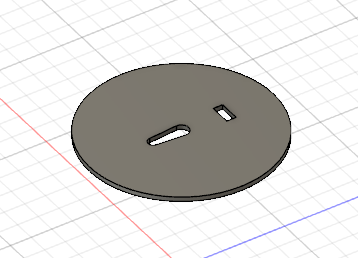


Figure 4 illustrates the servo bay. This component is responsible for attaching the lid to the base such that the lid can rotate whilst the base does not. It has fittings for the servo rudder and a space for the ultrasonic sensor wires. It is fixed to the underside of the lid such that the space for the ultrasonic sensor wire align.

## Circuit

The circuit represents the electrical components and how they are connected to the Arduino. The Schematic diagram of the circuit can be found below.

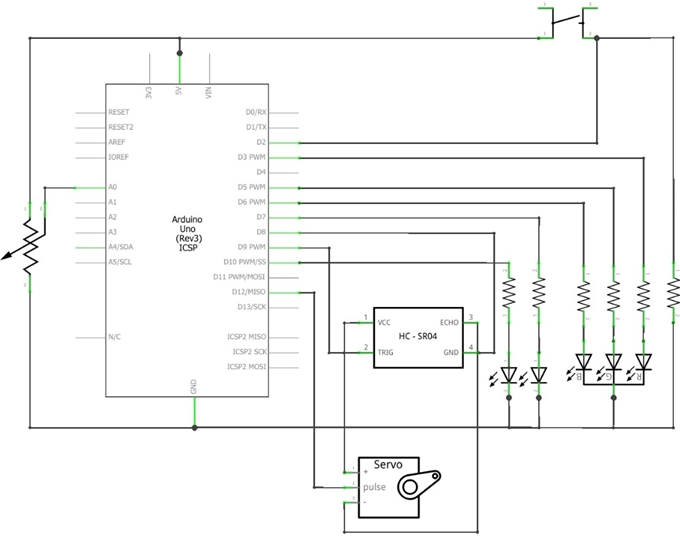


Figure 5 illustrates how the components are connected to the Arduino. All components have a live and ground connection. Each RGB channel is connected to a separate pin. The servo is connected to pin 12, the ultrasonic sensor is connected to pins 8 and 9, the potentiometer is connected to pin A0, the normal LEDs are connected to pins 10 and 7 and finally, the RGB LED is connected to pins 3, 5 and 6.

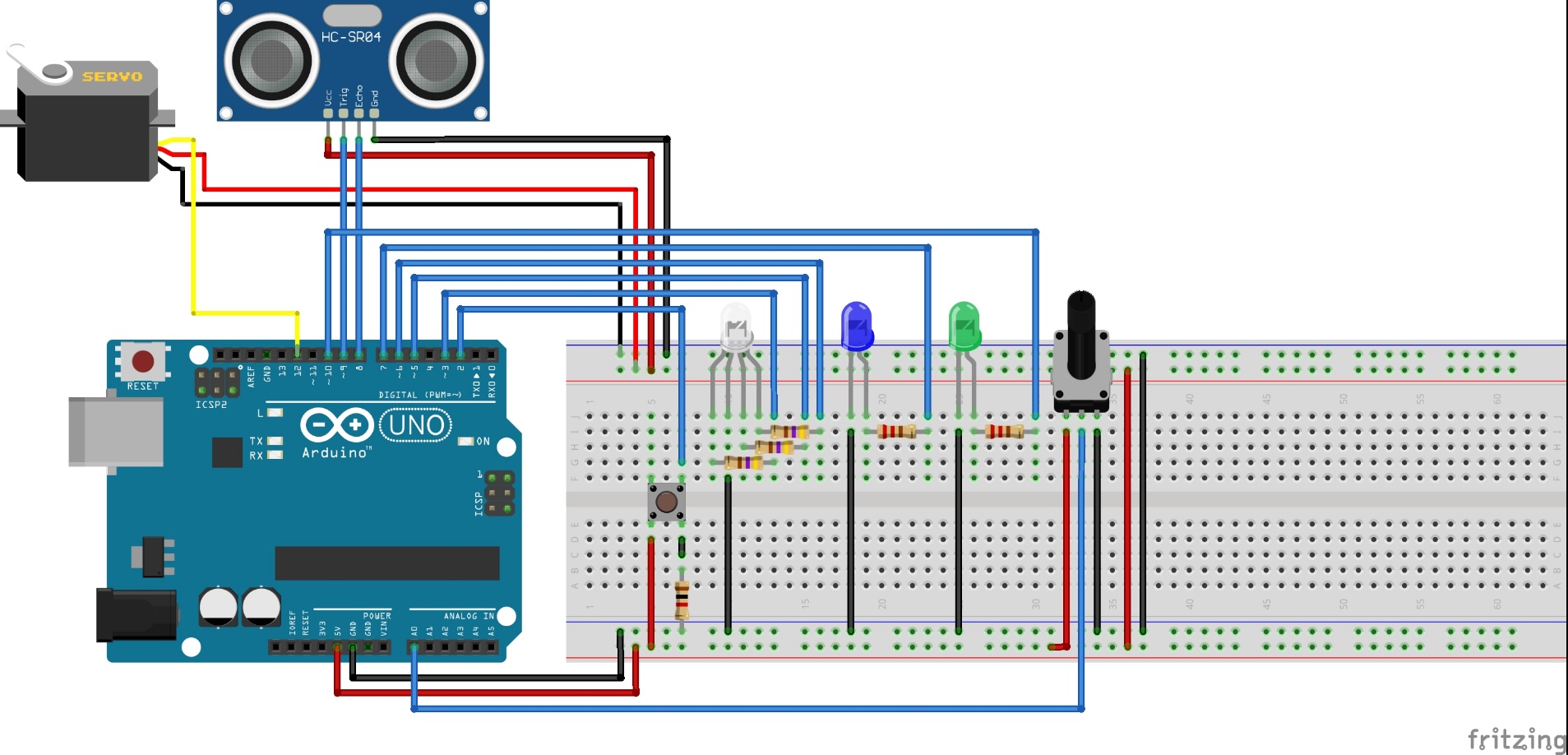


Figure 6 illustrates a graphical example of the components connected to the Arduino.

## Software

The software aspect of the system features under the design section due to the state machine ideology that is embedded within it. Each function that the system offers is a separate state in the software. For example, when the Arduino is plugged in, but the power button has not been pressed, it is in the “turned off” state.

The state diagram of the system is presented below:

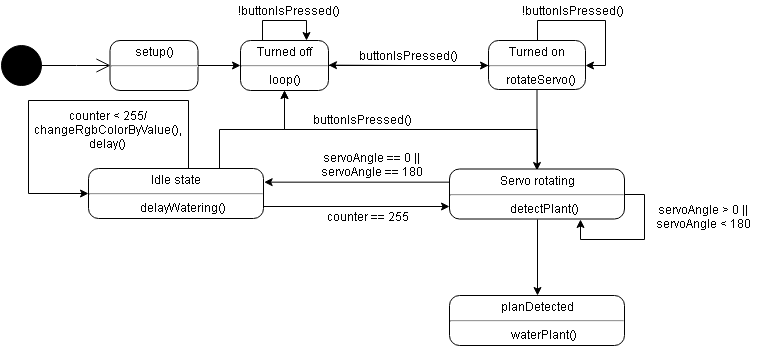


Figure 7 describes the state machine of the system. On initialization, the system is in the turned off state. When the power button is pressed, the state is changed to “Turned On”. From the “Turned On” state, the system proceeds to the “Servo Rotating” state, unless the user presses the power button. In this case, the state is changed to “Turned Off”. In the “Servo Rotating” state, the system will proceed into the “Plant Detected” state if a plant is detected. If no plant is detected, the system remains ins the “Servo Rotating state” if the servo angle is less than 180 degrees. Once this condition is false, the system will move into the “Idle” state where it will remain until the counter is 256. At this point it will return to the “Servo Rotating” state. It is also possible for the system to go from the Idle state to the “Turned Off” state should the user press the power button.

# References

Some of the components and techniques used required some research first. Below is a list of all the resources used to complete this assignment.

1. <https://www.arduino.cc/en/Reference/Servo> [↑](#endnote-ref-1)
2. <https://www.arduino.cc/reference/en/language/structure/further-syntax/define/> [↑](#endnote-ref-2)
3. <https://www.arduino.cc/reference/en/language/structure/sketch/setup/> [↑](#endnote-ref-3)
4. <https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/> [↑](#endnote-ref-4)
5. <https://www.arduino.cc/en/Reference/ServoAttach> [↑](#endnote-ref-5)
6. <https://www.arduino.cc/en/serial/begin> [↑](#endnote-ref-6)
7. [https://www.arduino.cc/en/Reference/Loop](https://www.arduino.cc/en/Reference/Loop?setlang=it) [↑](#endnote-ref-7)
8. <https://www.arduino.cc/en/Reference/ServoWrite> [↑](#endnote-ref-8)
9. <http://www.robot-electronics.co.uk/htm/arduino_examples.htm#SRF04%20Ultrasonic%20Ranger>

   10 <https://www.youtube.com/watch?v=_B3gWd3A_SI&t=95s> [↑](#endnote-ref-9)