ELEC3607/9607 Project Group29

Watering Garden

Qizhang Li SID:460452436

Yuechuan Tao SID:470268177

THE UNIVERSITY OF SYDNEY

School of Electrical and Information Engineering

STUDENT PLAGIARISM: COURSEWORK – POLICY AND PROCEDURE

COMPLIANCE STATEMENT

LABORATORY WORK

I certify that:

(1) I have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure*;

(2) I understand that failure to comply with the *Student Plagiarism: Coursework Policy and Procedure* can lead to the University commencing proceedings against me for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

(3) the Work that I record in my logbook or have marked will be solely the work of my lab partner and myself. All measurements, calculations, and other Work will be done during the assigned laboratory session unless otherwise noted. The Work will not include any material previously submitted for credit in any unit of study.

Name(s): Qizhang Li, Yuechuan Tao

Signature(s): 李启璋, 陶悦川

Date: 2018. 6.15

Unit of Study for which Work will be submitted: ELEC 3607 Assignment

ELEC3607/9607 Milestone IV

Project Implementation Summary Form

Category	List of items, with web link or page number where item is described
Open source compliance Are you publishing your code as open source? Explain what license you are using, or link to where you have published your project.	Publish the code at GitHub https://github.com/viperliwow/elec3607project with MIT license.
Platforms List the platforms you have used, including processor architecture (ARM, AVR, x86, etc.), programming languages (Python, C, C++, etc.), and IDEs. If you are not using Arduino Due, justify your choice of platform.	Arduino DUE (Atmel SAM3X8E) Arduino Phone (x86/ARM) C language for programming in Arduino IDE
Sensors and inputs List your hardware inputs (push buttons, analog sensors, I2C or SPI sensors including accelerometers, etc.) and mention on which page each input is described.	Moisture sensor(page3) Ultrasonic sensor(page4)
Outputs List any mechanical, visual or other type of outputs controlled by your hardware (motors, magnets, LED, 7-Segs, LCD, etc.)	LCD(page4) Servo(page4)

and mention on which page each input is described.	
Connectivity List any protocols used for communicating between your main controller and any other microcontroller excluding sensors or actuators (USART, UART over USB, Bluetooth 2, Bluetooth 4, Wi-Fi, ZigBee, etc.) and mention on which page each input is described.	UART and Bluethooth2.0(page5)
Graphical User Interface List any GUI used for interfacing users (local LCD, Phone app, Desktop app, Web app, Web dashboard, etc.) and mention on which page each input is described.	LCD(page4) Bluetooth APP on Android phone(page5)
Algorithms / Logic List substantial control algorithms (state-machines, real-time operating system, filesystems, feedback algorithms, mathematical transformations, etc.) and mention on which page each input is described.	Real-time operation system(page7)
Physical case and other mechanical consideration List what casing or mechanical	Water bottle and straw (pages10)

systems have been used in your project (3d prints, pipes, cardboard mechanics, etc.) and mention on which page each input is described.	

Abstract

In this project, we are going to design a house-used automatic gardening system. The system can automatic watering the garden considering the moisture of the soil as well as weather factors. The water flowing rate will be controlled to ensure the irrigation quality. And all the system running situation will be shown in LCD screen. Besides, the house owner can also remotely control the system by using the apps in mobile phone. The system is generally successfully tested. This report will focus on introducing the hardware and software design, the debug process as well as the future improvement.

1. Background

Consuming most of the day time in working, many citizens have rare chance to pay attention to their garden and leaving the plants deserted. Some people may hire gardener to take care of their garden, which consumes a large amount of money. However, along with the emerging of the embedded system, realization of automatic irrigation to the garden can be achieved with the automatic watering garden. The system not only can automatic watering the garden when the soil is dry but also will consider the weather data so that it can be used in outdoor gardening. In this way, the plants in the garden can be well irrigated even though they are neglected by their owner due to heavy working burden.

2. Aim

The system is controlled by Arduino DUE, which is the micro-controller. The peripheral circuit includes moisture sensor circuit, ultrasound sensor circuit, servo, LCD display circuit as well as Bluetooth. The moisture sensor will sense the humidity of the soil and send the data to the micro-controller. At the same time, the Bluetooth will send the weather information from the mobile phone to micro-controller. The micro-controller will combine the data of current soil moisture and the future weather to decide whether to water or not. If the plants should be watered, the servo will rotate. In this way, it can prevent the problem of over watering during the wet weather condition. The ultrasound is used to sense the water level in the tank. Different water level will cause the water flowing rate changing accordingly. To ensure the fixed water flowing rate, the servo should stay at different angles. The moisture of the soil and the water level will be shown on the LCD screen. The user can also control the servo by using Bluetooth app on their mobile phone to realize remote control.

3. Hardware Design

3.1 System structure

The system structure diagram is shown as Fig 3.1

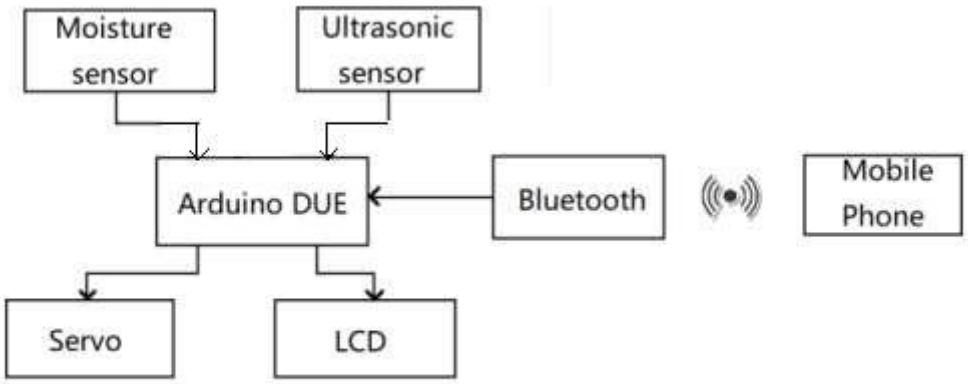


Fig 3.1 system structure diagram

In this system, these components are used: a moisture sensor, an ultrasonic sensor, a Bluetooth shield, an Arduino DUE, a servo and a LCD screen. A water bottle and a straw are used for modelling the water tank and the watering pipe.

3.2 Micro-controller

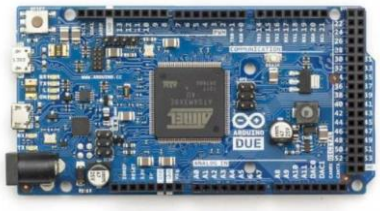


Fig 3.2 Arduino DUE

The micro-controller we chose is Arduino DUE which is an ARM based micro-controller. In this project, Arduino DUE play the role of a “brain”. It will receive data from sensors as well as Bluetooth shield. Then Arduino DUE will gather the information and make the decision of watering by controlling the servo. It also will print the real time situation on LCD screen and send to the mobile phone via Bluetooth.

3.3 Moisture sensor

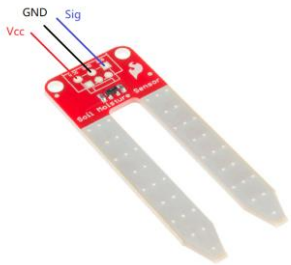


Fig 3.3 Moisture sensor

The type of the moisture sensor we use is “SF-SEN-13322” which is mainly for soil moisture. There are three pins for the sensor, connecting to 5V voltage, ground and the output signal. The resistance of the two legs on the sensors will change based on its surface moisture, changing the output voltage at the same time. Thus it can be used to sense the moisture.

3.4 Ultrasonic sensor

Fig 3.4 Ultrasonic sensor

The ultrasonic sensor is used to measure the water level in the tank. The type of ultrasonic sensor we used is “HC - SR04”. There are four pins for this module. It is noticeable that there is a trig pin and an echo pin in the module. A short 10uS pulse should be supplied to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo sending the pulse through echo pin. The micro-controller can measure the time interval between sending trigger signal and receiving echo signal and then convert it into the distance.

3.5 Servo

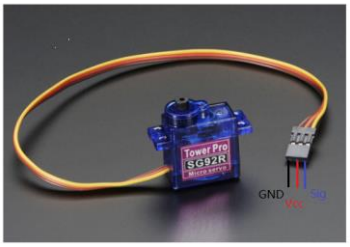


Fig 3.5 Servo

The servo is used to control whether water or not and the water flowing rate. The type of servo we used is “AF-169”. It has three pins. One should be connected to the ground, one should be connected to Vcc and the left one is the signal input pin. To drive the servo, a PWM pulse should be input to the servo. According to the different cycles, the servo will rotate to different position, and hence the control of the irrigation can be realized.

3.6 LCD

LCD is used to display the humidity and the water level in this project. The type of the LCD we used is “16x2” LCD shield with keyboard. The LCD shield can be installed

on Aruino directly. The LCD has 16 pins and besides ground and Vcc, it will occupy 7 pins of Arduino DUE (D4, D5, D6, D7, D8 and D9). It also has 6 buttons among which one is for resetting. The other buttons will provide different analog signals and A0 pin on the Arduino DUE will read these signals. And hence these buttons can be used for shifting different interfaces of LCD.

3.7 Bluetooth shield

Fig 3.6 Bluetooth

The Bluetooth is used for sending the weather information from mobile phone wirelessly and it can also be used to realize remote control. The Bluetooth shield can be installed on Arduino directly. The tx pin of Bluetooth shield should be connected to rx2 pin of Arduino while the rx pin of Bluetooth shield should be connected to tx2 pin of Arduino.

4. Operational scenario

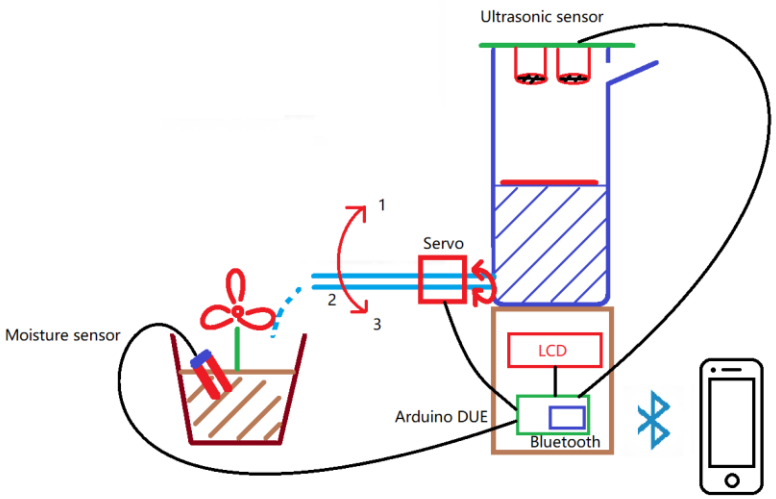


Fig 4.1 Operational scenario

Figure 4.1 is the operation scenario for our system. From the diagram, we can see that Arduino and Bluetooth are installed together. Bluetooth can communicate with mobile phone wirelessly. All the black lines are the wire connection. Sensors send data to micro-controller, and micro-controller will give order to servo. The servo can drive the pipe to control the watering process. When the pipe is at position 1, it will not water. When it is at position 2 it begins to water. To increase the watering flow, the

pipe can reach position 3.

5. Software

In the software programming design, we choose to use C language. We first test all the module separately and finally combine them together. The whole coding will be seen in the appendix.

5.1 Flow chart

The flow chart will be shown as Fig

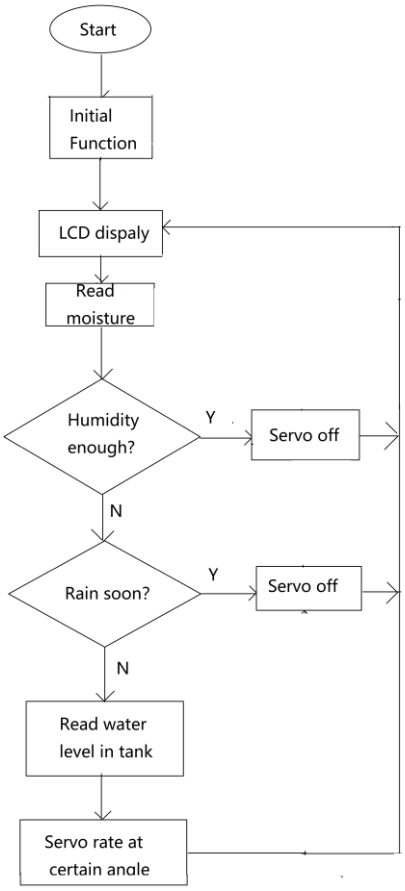


Fig 5.1 Flow chart

5.2 Moisture sensor

Sensing the soil moisture is a simple part, since the sensor will send an analog signal. So what we should do is let the micro-controller read this analog signal as following[4]:

```
Moisture_value= analogRead(Moisture);
```

We use timer counter to sense the moisture during every same interval. Time counter 1

and channel 0 is chosen. And the interval time rc is set as “rc = VARIANT_MCK / 128 / 9600” .

5.3 Ultrasonic sensor

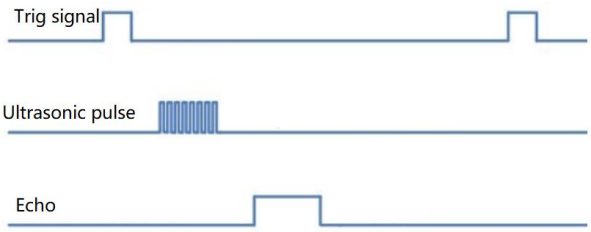


Fig 5.2 Timing diagram

The timing diagram of ultrasonic is shown as Fig 5.2.

Firstly, we should use the digital pin of Arduino to provide the Trig pin with a high signal of at least 10μs as following[3]:

```
digitalWrite(TrigPin, LOW);
delayMicroseconds(2);
digitalWrite(TrigPin, HIGH);
delayMicroseconds(10);
digitalWrite(TrigPin, LOW);
```

After the trigger, the module will automatically send 8 ultrasonic pulses of 40KHz and automatically detect whether there is a signal return as well as raise the echo pin. This step is done automatically by the module.

If signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return time. At this point, we can use the pulseIn() function to get the the duration of the high level, which is the time from when the ultrasonic wave is transmitted to return. To calculate the real distance, $\text{Distance (m)} = \frac{\text{Time (s)} \times \text{V (m/s)}}{2}$, where V=340m/s. Then transfer the unit we need as

$$\text{Distance(cm)} = \frac{\text{Time (us)} \times 340 \times 10^{-4}\ \text{(cm/us)}}{2}$$

$$\text{Distance (cm)} = \text{Time (us)} \times 0.0170\ \text{(cm/us)} \approx \frac{\text{Time(us)}}{58\ \text{(us/cm)}}$$

So, the code should be programmed as :

```
distance = (pulseIn (EchoPin, HIGH)/58 )-1;
```

5.4 Servo

The servo is controlled by a PWM pulse. For different duty cycle, it will rotate different angles. So, in one way, we can create a PWM pulse by setting high then setting low. However, there is another method to make it easier. We can use the library of servo.h[1]. Firstly, we create servo object to control a servo and attach the servo on one of the digital pin to the servo object.

```
Servo myservo;

void setup() {
  myservo.attach(9);
}
```

Under different situation, the micro-controller should tell servo to rotate to different direction:

`myservo.write(Position);` "Position" is a variable which decided by the moisture, weather and the water level in the tank.

For the water flowing rate, we applied a simple open loop control shown as Fig 5.1

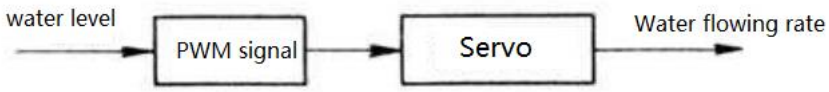


Fig 5.3 Open loop control for water flowing rate

5.5 LCD

LCD is used to display the moisture and water level. We can use the library of LiquidCrystal.h[2] which can make the problem become easier. With the library of LiquidCrystal.h, we do not need write I2C and transfer the lattice by ourselves.

First we set the pin used on LCD panel by using "LiquidCrystal lcd(8, 9, 4, 5, 6, 7); ". Every time we want to print words on the screen, the LCD cursor position should be set as "lcd.setCursor(x,y);", where x represents row and y represents column. Then " lcd.print("XXXXX")" can be used to print the words.

The LCD shield also contains a keyboard. By pushing different buttons, the interface of LCD can be shifted. The four buttons shares A0, which is an analog read pin of Arduino, which means by pushing different buttons, different value of analog signal will send to micro-controller. For example:

```
adc_key_in = analogRead(0);
if (adc_key_in > 1000) return btnNONE;
 else if (adc_key_in < 50)   return btnRIGHT;
   else if (adc_key_in < 380)  return btnUP;
    else if (adc_key_in < 790)  return btnDOWN;
     else if (adc_key_in < 1000)  return btnLEFT;
```

When no button is pushed, Arduino will receive 1023, while when the right button is pushed, the value of Arduino read is smaller than 50. So, in our code, if no button is

pushed, the moisture will be displayed. If the right button is pushed, the water level will be displayed.

```
switch (lcd_key){

      case btnRIGHT:{
           lcd.print("Water    ");
             lcd.setCursor(9,1);
           lcd.print(Water_remain);
              delay(10);
           break;
      }
       case btnNONE:{
             lcd.print("Moisture  ");
              lcd.setCursor(9,1);
            lcd.print(Moisture_value);
               delay(10);
             break;
```

5.6 Bluetooth2.0

To guarantee the Bluetooth working properly, we should first set the connection of the Bluetooth, including its baud rate, working mode (master or slave), name , Permission of the connection of paired device, Auto-connection forbidden, pin, etc. Then the micro-controller can receive chars from mobile phone through Bluetooth.

```
char recvChar;
     if(blueToothSerial.available())
     {
         recvChar = blueToothSerial.read();
         //Serial.print(recvChar);
       for (int i=0;i<10;i++)
          {if (recvChar ==  rainbuffer[i])
            rainhour=i;
           }
```

“ if (blueToothSerial.available())” is used to check if there's any data sent from the remote bluetooth shield.

6. Test result

When the moisture level is high or it will rain in 3 hours, the watering will not be processed, shown as Fig 6.1

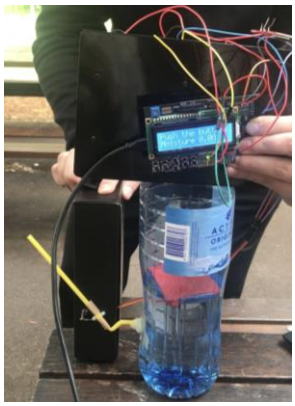


Fig 6.1 Servo off

When the soil is dry and the it will not rain in 3 hours, the watering will be process automatically. The rotating angle will be decided on the water level in the tank , shown as Fig 6.2 and Fig 6.3.

Fig 6.2 Servo on (20°)

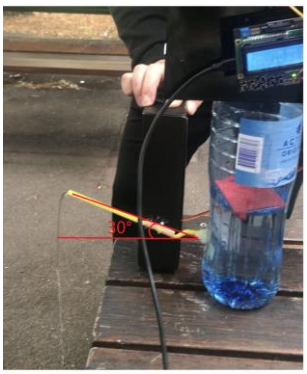


Fig 6.3 Servo on (30°)

The LCD will display the moisture and water level, shown as

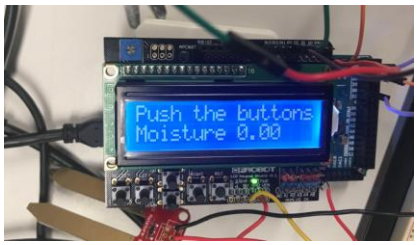


Fig 6.4 Moisture level (Dry)

Fig 6.5 Moisture level (Wet)

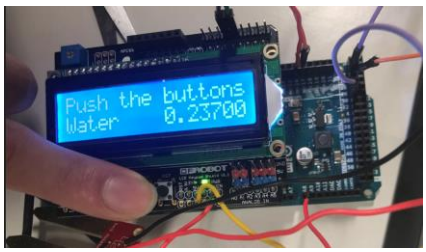


Fig 6.6 Water level

7. Further improvement

During the project, we found several problems which can be improved and other superior functions which can be realized in the future.

(1) The irrigation method

In this project, we used servo to drive the irrigation as mentioned before. However this method has several limits. For example, the water tank should be installed at the relatively high position, since it is using the gravity of the water to irrigate. Besides, using servo to drive the pipe will reduce the life span of the pipe causing leaking problem. So, to make the device robust and practical, we can use the electromagnetic valve instead of the servo. And the servo can be used to control the water flowing rate.

(2) The water flowing rate control

In the control process, we provide a method to control the flowing rate of the water however it is a open loop control. To make it more precise to control, a close loop control with PID controller can be applied. Therefore, a flow sensor should be introduced to provide feedback to the micro-controller, shown as Fig 7.1.

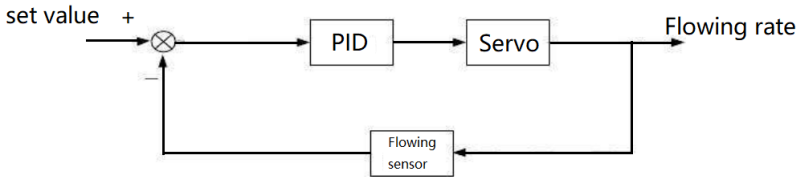


Fig 7.1 Closed loop control of water flowing rate

(3) The wireless data transmission

As for information transmission, Bluetooth has been applied. However, to transmit the data, we have to type the weather information on the mobile phone manually or design an app by ourselves. However, if we use WIFI module, this function can be realized more easily. It can get the weather information from the mobile phone and transmit data to micro-controller through WIFI, which realize true automation.

8. Conclusion

In this project, we have developed an outdoor irrigation system which can realize automatic irrigation according to current soil moisture and weather information. On the other hand, the control of water flowing rate is accomplished. The device is equipped with various peripheral circuits, which make the device intuitive and easy to operate. The whole system is built into real hardware and it is successfully designed and tested. However, during the debug process, we have encountered several problems and some factors can be improved in the future work. From this project, we are more familiar with the basic knowledge about embedded system, such as timer counter, UART, ADC and etc. To most important, during the experimental set up, we should learn how to debug our circuit to find out the problems when the result dose not comply with the expectation. Through measuring voltage and waveform, examining the connection, debugging the code we can exclude the problem step by step and finally find the solution.

9. Reference

[1] Arduino Servo library: https://github.com/arduino-libraries/Servo
[2] Arduino LCD library: https://github.com/arduino-libraries/LiquidCrystal
[3] Sample code for ultrasonic sensor: http://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/
[4] Moisture sensor: https://github.com/sparkfun/Soil_Moisture_Sensor

Appendix I Code:

```
#include <Servo.h>                          //using servo.h library for control servo
#define Moisture A8
#include <LiquidCrystal.h>                  //using library to control LCD
#define blueToothSerial      Serial2        //set serial2(USART1) as blueToothSerial
char rainbuffer[] ="0123456789";
char recvChar;
int rainhour;
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);                  // select the pins used on the LCD panel
Servo servo29;

int lcd_key         = 0;
int adc_key_in    = 0;
#define Moisture A8
#define btnRIGHT    0
#define btnUP         1
#define btnDOWN      2
#define btnLEFT      3
#define btnSELECT 4
#define btnNONE      5

const int TrigPin=22;
const int EchoPin=24;


float distance;
int a;
float leng=30.0;
float Water_remain;
float sum;
float cm;
int Position;
float Moisture_value;

int read_LCD_buttons(){                         // read the buttons to select mode
       adc_key_in = analogRead(0);              // read the value from the sensor
       if (adc_key_in > 1000) return btnNONE;
         else if (adc_key_in < 50)      return btnRIGHT;
            else if (adc_key_in < 380)    return btnUP;
```

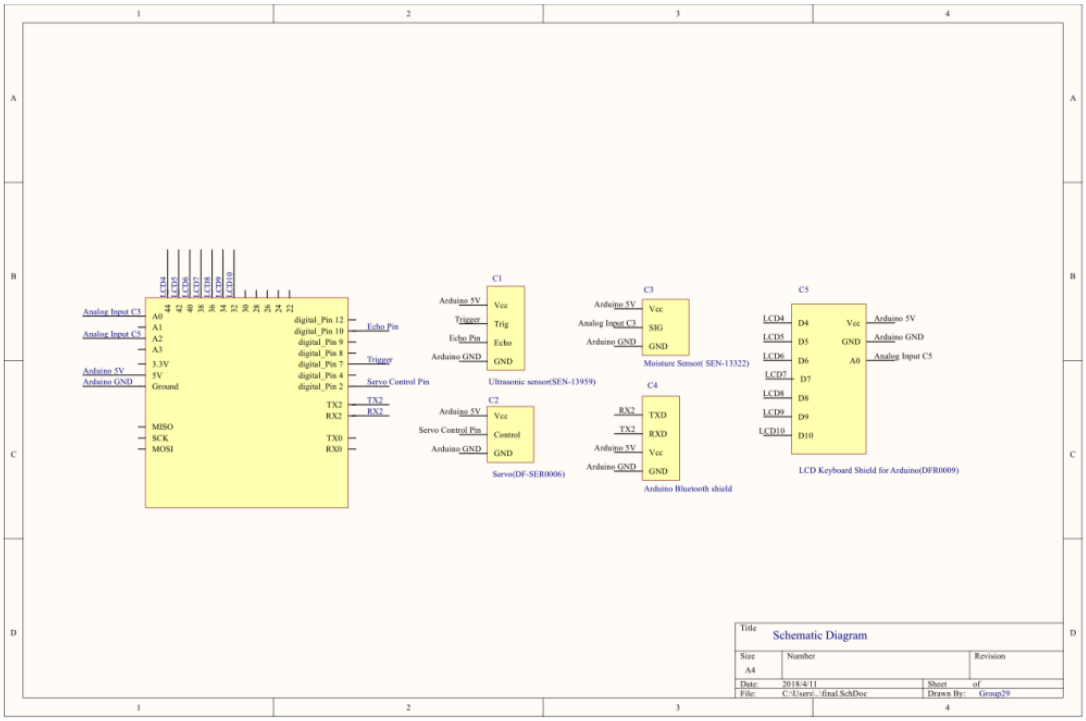
```
            else if (adc_key_in < 790)    return btnDOWN;
              else if (adc_key_in < 1000)    return btnLEFT;

      return btnNONE;                          // when all others fail, return this.
}

void setup() {
   lcd.begin(18, 2);                      // start the library
   lcd.setCursor(0,0);                    // set the LCD cursor      position
   lcd.print("Push the buttons");    // print a simple message on the LCD
  Serial.begin(9600);
  pinMode(TrigPin,OUTPUT);
  pinMode(EchoPin,INPUT);
  servo29.attach(30);                    //setup the servo
  pinMode(Moisture, INPUT);
   setupBlueToothConnection();
   //initalize TC
   pmc_set_writeprotect(false);
   pmc_enable_periph_clk(ID_TC1);
   TC_Configure(TC1, 0, TC_CMR_WAVE | TC_CMR_WAVSEL_UP_RC |
TC_CMR_TCCLKS_TIMER_CLOCK4);
   const uint32_t rc = VARIANT_MCK / 128 / 9600;
   TC_SetRC(TC1, 0, rc);
   TC_Start(TC1, 0);
   //enable TC interupt
   TC1->TC_CHANNEL[1].TC_IER = TC_IER_CPCS;
   NVIC_EnableIRQ(TC1_IRQn);
}

void loop() {
     Serial.print("rain ");
     Serial.println(rainhour);
     Serial.print("distance    ");
     Serial.println(distance);
     Serial.print("moisture    ");
     Serial.println(Moisture_value);

     char recvChar;
            if(blueToothSerial.available())
            {//check if there's any data sent from the remote bluetooth shield
                  recvChar = blueToothSerial.read();
                  //Serial.print(recvChar);
               for (int i=0;i<10;i++)
                   {if (recvChar ==    rainbuffer[i])
```

```
                    rainhour=i;
                   }

      // Serial.println(rainhour);
           }



  digitalWrite(TrigPin,LOW);
  delayMicroseconds(2);
  digitalWrite(TrigPin,HIGH);
  delayMicroseconds(10);
  digitalWrite(TrigPin,LOW);
  distance = (pulseIn (EchoPin,HIGH)/58 )-1;

  delay(1000);
   // Serial.println(distance);

  Water_remain=1-distance/leng;        // calculate the remain water


if (Moisture_value < 512    && rainhour>3)
 {if (Water_remain<0.3)
     {Position= 30; }                        //turn the servo to 30degree
       else if (Water_remain<0.6)
       {Position= 20;}                      //turn the servo to 20degree
        else
        Position=10;                        //turn the servo to 10degree

 }
 else
    Position= 0;                           //turn the servo off
    servo29.write(Position);

    lcd.setCursor(0,1);                    // move to the begining of the second line
    lcd_key = read_LCD_buttons();     // read the buttons
 switch (lcd_key){                       // depending on which button was pushed, we
perform an action

          case btnRIGHT:{                  //    push button "RIGHT" and show the word
on the screen
                lcd.print("Water     ");
                  lcd.setCursor(9,1);                  // move cursor to second line "1"
```

```
and 9 spaces over
                lcd.print(Water_remain);          // display seconds elapsed since
power-up
                   delay(10);
                break;
          }
          case btnLEFT:{
                lcd.print("LEFT     "); //    push button "LEFT" and show the word on the
screen
                break;
          }
          case btnUP:{
                lcd.print("UP       ");    //    push button "UP" and show the word on the
screen
                break;
          }
          case btnDOWN:{
                lcd.print("DOWN    ");    //    push button "DOWN" and show the word on
the screen
                break;
          }
          case btnSELECT:{
                lcd.print("SELECT");    //    push button "SELECT" and show the word
on the screen
                break;
          }
          case btnNONE:{
                lcd.print("Moisture    ");    //    No action    will show "None" on the
screen
                  lcd.setCursor(9,1);                  // move cursor to second line "1"
and 9 spaces over
                lcd.print(Moisture_value);          // display seconds elapsed since
power-up
                   delay(10);
                break;

          }
    }

}

void setupBlueToothConnection()
{
      blueToothSerial.begin(38400);                                    // Set BluetoothBee
```

```
BaudRate to default baud rate 38400
      blueToothSerial.print("\r\n+STWMOD=0\r\n");                  // set the bluetooth
work in slave mode
      blueToothSerial.print("\r\n+STNA=garden\r\n");                 // set the bluetooth
name as "yuy"
      blueToothSerial.print("\r\n+STOAUT=1\r\n");                  // Permit Paired
device to connect me
      blueToothSerial.print("\r\n+STAUTO=0\r\n");                  // Auto-connection
should be forbidden here
      blueToothSerial.print("\r\n+STPIN=1234\r\n");                // set pin as
1234(default value is 0000 or 1234)
      delay(2000);                                                  // This delay is
required.
      blueToothSerial.print("\r\n+INQ=1\r\n");                     // make the slave
bluetooth inquirable
      Serial.println("The slave bluetooth is inquirable!");
      delay(2000);                                                  // This delay is
required.
      blueToothSerial.flush();
}

void TC1_Handler() {
   TC_GetStatus(TC1, 0);
   Moisture_value= analogRead(Moisture);
}
```

Appendix II Scheme diagram:


Analog Input C3
Analog Input C5
Arduino 5V
Arduino GND
A0
A1
A2
A3
3.3V
5V
Ground
MISO
SCK
MOSI
digital_Pin 12
digital_Pin 10
digital_Pin 9
digital_Pin 8
digital_Pin 7
digital_Pin 4
digital_Pin 2
TX2
RX2
TX0
RX0
Echo Pin
Trigger
Servo Control Pin
TX2
RX2
C1
Arduino 5V
Trigger
Echo Pin
Arduino GND
Vcc
Trig
Echo
GND
Ultrasonic sensor(SEN-13959)
C2
Arduino 5V
Servo Control Pin
Arduino GND
Vcc
Control
GND
Servo(DF-SER0006)
C3
Arduino 5V
Analog Input C3
Arduino GND
Vcc
SIG
GND
Moisture Sensor(SEN-13322)
C4
RX2
TX2
Arduino 5V
Arduino GND
TXD
RXD
Vcc
GND
Arduino Bluetooth shield
C5
LCD4
LCD5
LCD6
LCD7
LCD8
LCD9
LCD10
D4
D5
D6
D7
D8
D9
D10
Vcc
GND
A0
Arduino 5V
Arduino GND
Analog Input C5
LCD Keyboard Shield for Arduino(DFR0009)
Title
Schematic Diagram
Size
A4
Number
Revision
Date:
2018/4/11
Sheet
of
File:
C:\Users\...\final.SchDoc
Drawn By:
Group29