Appendix A.
Temperature control program coded with the Arduino IDE for the microcontroller ArduinoUNO® and the SEEED® relay shield v2.1.

```
// Include libraries for one wire, Dallas temp, SD card and real time clock
#include <OneWire.h>
#include <DallasTemperature.h>
#include <SD.h>
#include <DS3231.h>
// spi settings
// MOSI, MISO, SCLK Set by default
//port 10 assigned to chip select (slave select)
int pinCS = 10;
float refresh_rate = 0.0;
int powPin = 8;
int relay1 = 5;
int relay2 = 7;
// Init the DS3231 using the hardware interface
DS3231 rtc(SDA, SCL);
 // Data wire is plugged into port 2 on the Arduino
#define ONE_WIRE_BUS 2
#define TEMPERATURE_PRECISION 9
// Setup a oneWire instance to communicate with any OneWire devices (not just //Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
// arrays to hold device addresses
DeviceAddress insideThermometer, outsideThermometer;
void setup(void)
{
  //relay1 y relay2 salida "output"
  pinMode(relay1,OUTPUT);
  pinMode(relay2,OUTPUT);
  // start serial port
  Serial.begin(9600);
  // Initialize the rtc object
  rtc.begin();
  Serial.println("Iniciando tarjeta");
   //CS pin is an output
  pinMode(pinCS, OUTPUT);
  //Card will draw power from pin 8, so set it high
  pinMode(powPin, HIGH);
  digitalWrite(powPin, HIGH);
  //check if card is ready
   if(!SD.begin(pinCS))
    {
     Serial.println("Error, verifica que hay tarjeta...");
     return;
    }
      Serial.println("Acceso otorgado");

    //read configuration information (COMMANDS.txt)
    File commandFile = SD.open("COMMANDS.txt");
    if (commandFile)
    {
      Serial.println("Leyendo archivo");
      float decade = pow(10, (commandFile.available() - 1));
      while(commandFile.available())
```

```
    {
      float temp = commandFile.read() -('0');
      refresh_rate = temp*decade+refresh_rate;
      decade = decade/10;
    }
    Serial.print("Ciclos de = ");
    Serial.print(refresh_rate);
    Serial.println(" ms");
    commandFile.close();
  //Serial.println("Temperatura *C");
  }
  else
  {
    Serial.println("Error no se puede leer el archivo de comandos.");
    return;
  }
 File logFile = SD.open("log.csv",FILE_WRITE);
 if(logFile)
  {
 logFile.println("   ");
 String header = ("Fecha, Hora, Temp 1, Temp 2");
 logFile.println(header);
 logFile.close();
 //Serial.println(header);
  }
  else
  {
   Serial.println("Error");
  }
// Start up the library
 sensors.begin();
 // locate devices on the bus
 Serial.print("Localizando dispositivos...");
 Serial.print("Dispositivos encontrados ");
 Serial.print(sensors.getDeviceCount(), DEC);
 Serial.println(" dispositivos.");
 // report parasite power requirements
 Serial.print("parasite current: ");
 if (sensors.isParasitePowerMode()) Serial.println("ON");
 else Serial.println("OFF");
 // method 1: by index
 if (!sensors.getAddress(insideThermometer, 0)) Serial.println("Unable to find address for Device 0");
 if (!sensors.getAddress(outsideThermometer, 1)) Serial.println("Unable to find address for Device 1");
// show the addresses we found on the bus
 Serial.print("Dispositivo 0 direccion: ");
 printAddress(insideThermometer);
 Serial.println();
 Serial.print("Dispositivo 1 direccion: ");
 printAddress(outsideThermometer);
 Serial.println();
 // set the resolution to 9 bit
 sensors.setResolution(insideThermometer, TEMPERATURE_PRECISION);
 sensors.setResolution(outsideThermometer, TEMPERATURE_PRECISION);
 Serial.print("Dispositivo 0 Resolucion: ");
 Serial.print(sensors.getResolution(insideThermometer), DEC);
 Serial.println();
 Serial.print("Dispositivo 1 Resolucion: ");
 Serial.print(sensors.getResolution(outsideThermometer), DEC);
 Serial.println();
 Serial.println("   Fecha    ,    Hora    , Temp1       Temp2");
```

```
}
// function to print a device address
void printAddress(DeviceAddress deviceAddress)
{
  for (uint8_t i = 0; i < 8; i++)
  {
    // zero pad the address if necessary
    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}

// function to print the temperature for a device
void printTemperature(DeviceAddress deviceAddress)
{
  float tempC = sensors.getTempC(deviceAddress);
}
// function to print a device's resolution
void printResolution(DeviceAddress deviceAddress)
{
  Serial.print("Resolucion: ");
  Serial.print(sensors.getResolution(deviceAddress));
  Serial.println();
}
// main function to print information about a device
void printData(DeviceAddress deviceAddress)
{
  Serial.print("Dispositivo Direcccion: ");
  printAddress(deviceAddress);
  Serial.print(" ");
  printTemperature(deviceAddress);
  Serial.println();
}
void loop(void)
{
  sensors.requestTemperatures();
  float Temp1 = sensors.getTempCByIndex(0);
  float Temp2 = sensors.getTempCByIndex(1);
 String dataString = (String(rtc.getDateStr()) + "  ,  " + String(rtc.getTimeStr())+"  ,  "+ String(Temp1) + "    ,    "
+ String(Temp2));
{
  Serial.println(dataString);
}
     //Serial.println("DONE");
File logFile = SD.open("log.csv", FILE_WRITE);
   if(logFile)
   {
      logFile.println(dataString);
     logFile.close();
   }
   else
   {
     Serial.println("Error");
   }
  //Si la temperatura está arriba de 28 abre interruptor general
   if (Temp2 < 28)
    digitalWrite(relay2, HIGH);
    else
    digitalWrite(relay2, LOW);
  // Si la temperatura está por debajo de 40 C enciende la bomba de agua
```

```
  if (Temp1 < 40)
   digitalWrite(relay1, HIGH);
   else
   digitalWrite(relay1, LOW);
 delay (10000);
}
```