White Paper
Laboratory of Internet of Things

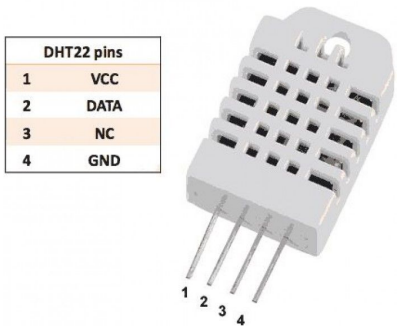
Lishchynska Ruslana
2019

Introduction

The Internet of Things refers to billion of physical devices around the world that are now connected to the Internet in order to collect and manipulate the data. Thanks to cheap microcontrollers and wireless network it is possible nowadays to turn anything into the part of Internet of Things[1]. This white paper is describing the prototype of a smart thermostat. IoT device that will be described later reads the temperature and humidity values from DHT-22 sensor, sends data to ThingSpeak platform, displays real-time values and is controlled from Virtuino simulating the thermostat response on LCD display. For this purpose, Arduino code is written and compiled using Arduino IDE.

Working parts

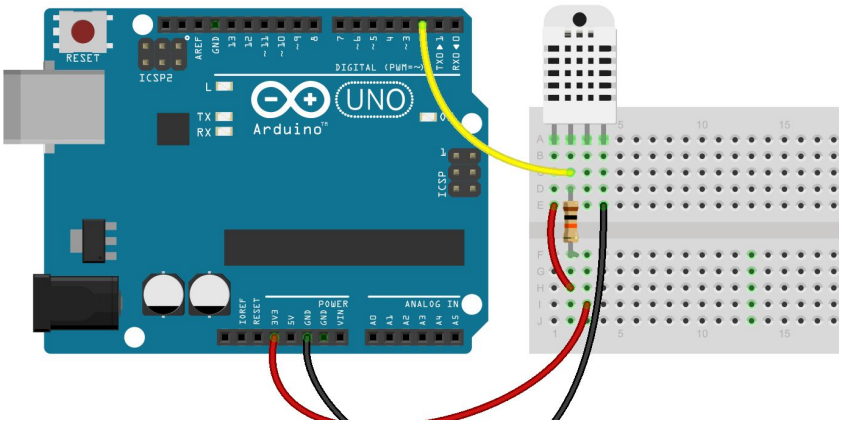
DHT-22 Digital-output relative humidity & temperature sensor/module. The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air, and spits out a digital signal on the data pin (no analog input pins needed).



Picture 1 - DHT-22[2]

It is fairly simple to use, but requires careful timing to grab data. Downside of this sensor is that it is possible to get new data from it once every 2 seconds, so when using Adafruit library, sensor readings can be up to 2 seconds old[3].

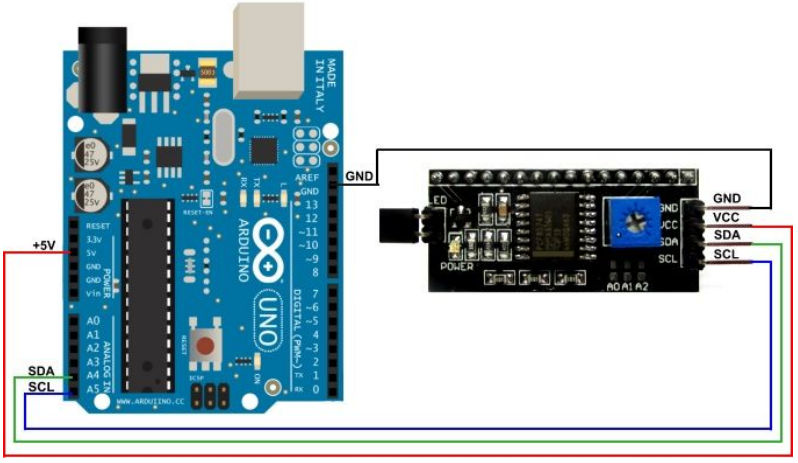
It is simple to connect, the first VCC pin to 3-5V power, the second data pin to data input pin and the fourth gnd pin to the ground.



Picture 2 - DHT-22 to Arduino connection scheme[4]

LCD Display Module. The LCD1602 display module is a very popular and inexpensive LCD display. The display consists of two rows of 16 characters each. It interfaces through a parallel data interface and has an LED backlight. In operation data is sent down the parallel data lines for the display. There are two types of data that can be sent to the display. The first type of data are the ASCII characters which are to be displayed on the display. The other type of data are the control characters that are used to activate the various display functions[5]. Connection to Arduino is made by I2C Bus. The bus uses four connections, two of them are for power:

- **Power** – This can be either 5 Volts or 3.3 volts, depending upon the application.
- **Ground** – The ground connection.
- **SDA** – Serial Data. This line is used for both transmit and receive.
- **SCL** – Serial Clock. This is a timing signal supplied by the Bus Master device.

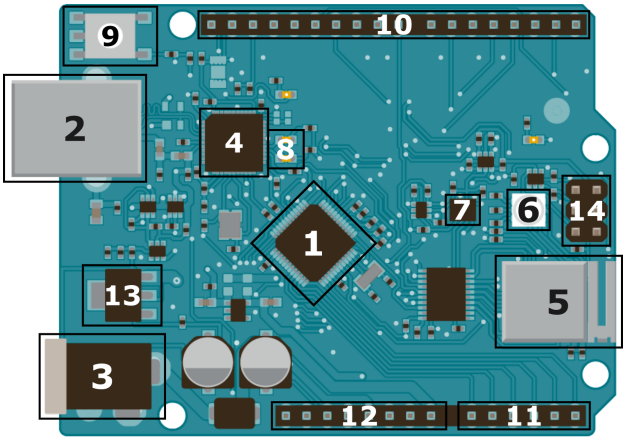


Picture 3 - LCD I2C Bus pins and communication with Arduino[6]

In I2C communications there is the concept of Master and Slave devices. There can be multiples of each but there can only be one Master at any given moment. In most Arduino applications one Arduino is designated Master permanently while the other Arduinos and

peripherals are the Slaves. The Master transmits the clock signal which determines how fast the data on the bus is transferred.

Arduino Uno WiFi Rev2. The Arduino Uno WiFi Rev2 is an Arduino Uno with an integrated WiFi module. The board is based on the Microchip MEGA4809 with an ESP32 u-blox NINA-W13 WiFi Module integrated. The NINA-W13 Module is a self contained System-on-Chip (SoC) with integrated TCP/IP protocol stack that can give access to WiFi network (or the device can act as an access point)[7].



Picture 4 - Arduino Uno WiFi Rev2 Structure[8]

Arduino Uno WiFi Rev 2 contains:

1. Micro-controller (ATMEGA4809). It offers 48KB programmable flash memory, 6144 Bytes SRAM, 256 Bytes EEPROM. ATmega4809 microcontroller operates between 1.8 to 5.5 volt. It contains a total of 48 input/output pins.
2. USB Connector. To load user program to Arduino using Arduino IDE and to power the Arduino Board.
3. Power Port. The power port is useful for supplying power Arduino from an external source. It is possible to use an AC to DC adapter or battery to power it. 2.1mm center-positive plug is used to power the Arduino Board. Recommended input power Voltage is 7-12V.
4. ISP Flash and USB Controller. Arduino UNO WiFi has Atmel ATmega32U4 as ISP Flash and USB Controller chip. ISP Flash stands for In-system programming Flash. It is the main bridge between Arduino USB and Micro-controller.
5. WiFi Module (NINA-W102). WiFi support is enabled with the u-blox NINA-W102 Module.
6. RGB LED. Arduino UNO WiFi has built-in RGB LED which can be programmable.
7. IMU (Inertial Measurement Unit). It contains onboard LSM6DSMTR IMU which is helpful for Accelerometer, Gyroscope, Temperature, and 3 Axis Sensor.
8. RX/TX LED. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer.

9. Master Reset Button. This Button is useful for resetting the Arduino UNO board and restarting the program from the origin. Pressing the Master Reset Button, it sends the logical pulse to the reset pin of microcontroller.
10. Digital I/O Pins. Arduino UNO WiFi board contains 14 Digital I/O Pins numbered from D0 to D13. In which 6 of them are PWM (Pulse-Width Modulation) Digital I/O Pins. Pin no 3, 5, 6, 9, 10, and 11 are having a (~) symbol on it are PWM Pins. PWM pins simulate analog output. Digital I/O Pins are useful to take Digital Input or provide Digital Output.
11. Analog Input Pins. The board contains 6 Analog Input Pins numbered from Pin A0 to Pin A5. Analog Input Pins are useful to take the signal from analog sensors and convert it into a digital value. The pins measure the voltage not current because it has a very high value of internal resistance. So, the value of the current is much smaller as compared to the voltage.
12. Power Pins
    - 12.1. Vin – The input voltage to the Arduino board when it's using an external power source.
    - 12.2. 5V – This pin outputs a regulated 5V volt supply generated by the onboard regulator.
    - 12.3. 3V3 – This pin outputs a regulated 3.3 volt supply generated by the onboard regulator.
    - 12.4. GND – Ground pins.
    - 12.5. IOREF – This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage. It selects the appropriate power source or enables voltage translators on the outputs to work with the 5V or 3.3V.
13. Voltage Regulator. The main work of the Voltage Regulator is to regulate the input and output voltage of the Arduino UNO board. It also regulates the voltage for 5V and 3V3 pins. If someone tries to supply power more than 12V, it may overheat and damage the board.
14. ICSP Header Pins. ICSP is In-Circuit Serial Programming also called In-System Programming (ISP). It is used to program AVR microcontrollers. You can use the Arduino ISP to upload sketches directly on the AVR-based Arduino boards without the need for the bootloader. ICSP Header Pins contains six pins MISO, +Vcc, SCK, MOSI, Reset, GND.

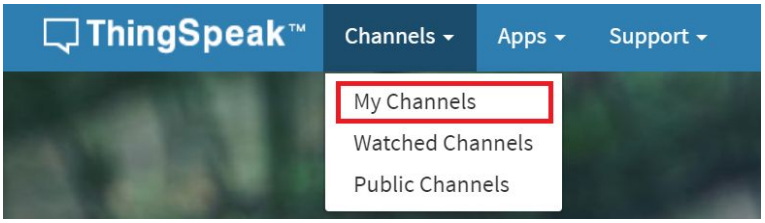
ThingSpeak. ThingSpeak™ is an IoT analytics platform service that allows programmers to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by the devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak software engineers can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics[9].

Virtuino. Virtuino is an Human-Machine Interface (HMI) platform for IoT servers, Arduino ESP and similar boards, MQTT brokers, PLCs and Modbus servers.

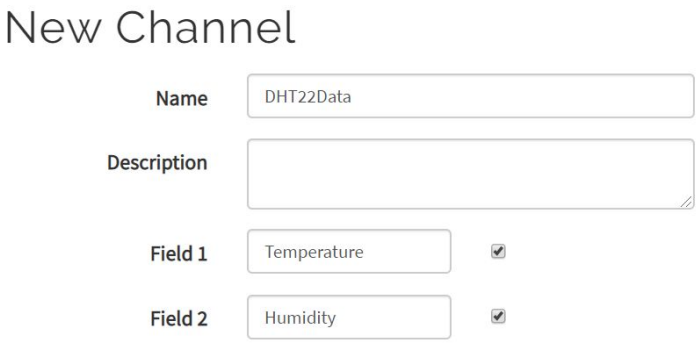
Tutorials

ThingSpeak Tutorial. Create a Channel:

1. Sign in to ThingSpeak using your MathWorks account, or create a new MathWorks account.
2. Click **Channels** -> **My Channels**



3. On the **Channels** page click **New Channel**.
4. Check the boxes next to Fields 1-2. Enter channel setting values:
    a. Name: DHT22Data
    b. Field 1: Temperature
    c. Field 2: Humidity

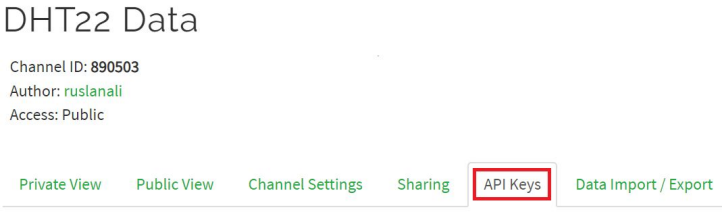


5. Click **Save Channel** on the bottom of this page.
   You now see these tabs:

- **Private View**: This tab displays information about your channel that only you can see.
- **Public View**: If you choose to make your channel publicly available, use this tab to display selected fields and channel visualizations.
- **Channel Settings**: This tab shows all the channel options you set at creation. You can edit, clear, or delete the channel from this tab.
- **Sharing**: This tab shows channel sharing options. You can set a channel as private, shared with everyone (public), or shared with specific users.
- **API Keys**: This tab displays your channel API keys. Use the keys to read from and write to your channel.
- **Data Import/Export**: This tab enables you to import and export channel data[10].

Post data from DHT-22 to ThingSpeak:

1. Go to the API Keys section on the earlier created channel.

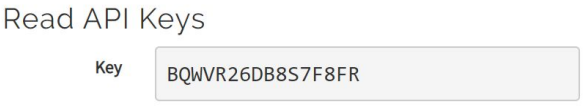


2. Record your write API key.

3. Program Arduino.

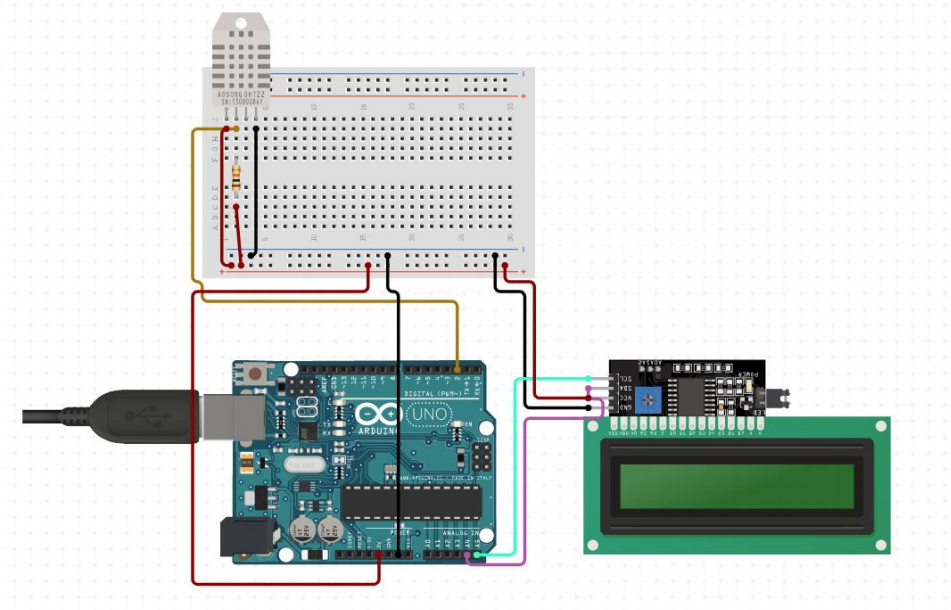
Read data from ThingSpeak to Arduino:

1. Go to the API Keys section on the earlier created channel.
2. Record your read API key.



3. Program Arduino.

Arduino + DHT22 + LCD connection circuit:



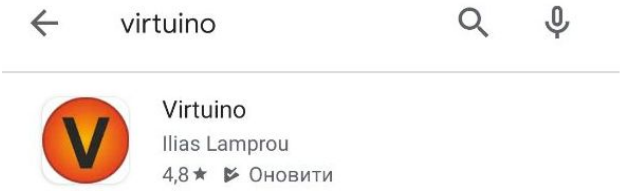
Programming Arduino:

1. Download the latest version of Arduino IDE.

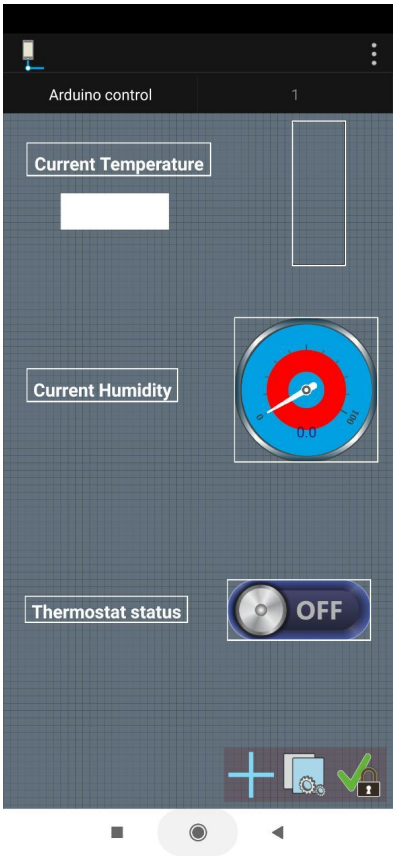
2. Add the needed libraries:
    a. Select **Sketch -> Include Library -> Manage Libraries**.
    b. Select WiFiNINA, ThingSpeak, DHT, SPI, Wire, hd44780 to add to Arduino sketch.
3. Select the appropriate port and board in the Arduino IDE.
4. Create the application. Open a new window in the Arduino IDE, and save the file. Add the code provided in the Source Code section of this White Paper. Be sure to change the wireless network information, the channel IDs, the read API key, and the write API key.
   Note: The Channels that was used to complete this project are public, you can use not your own keys, but the provided in the code.

Virtuino Setup and Configuration:

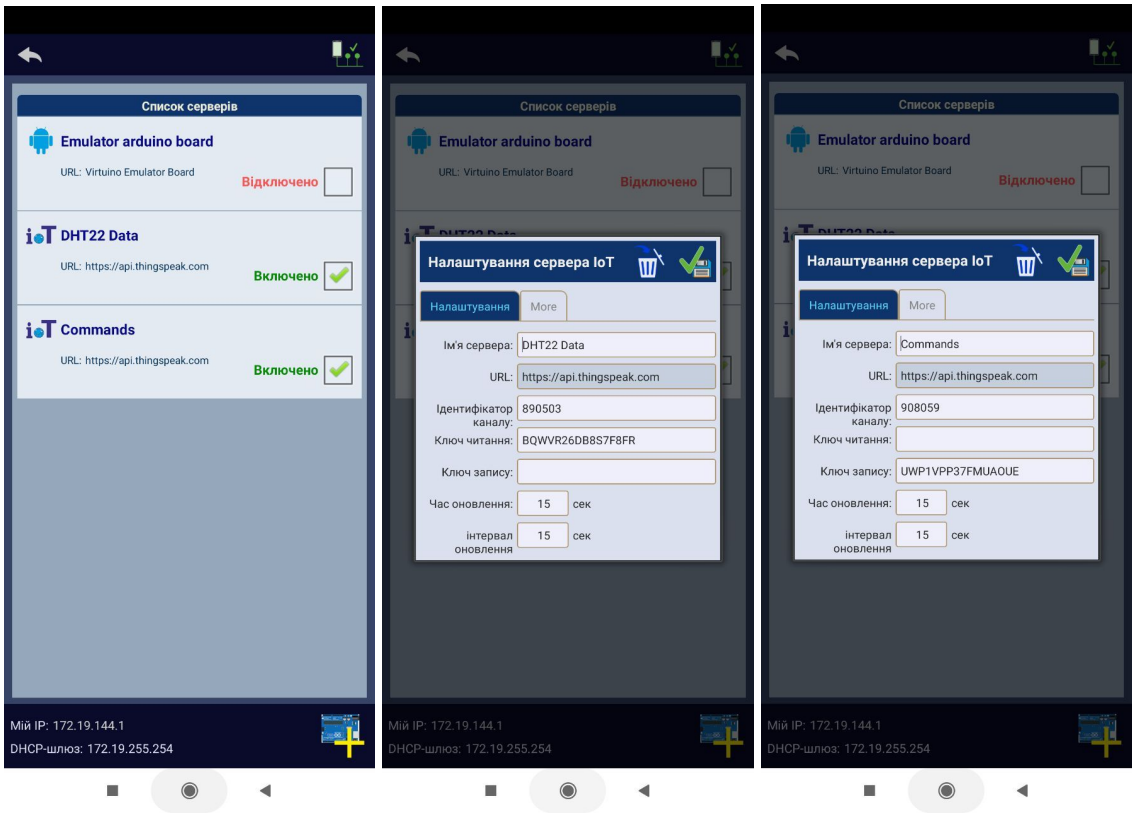
1. Download Virtuino app from Play Market.



2. Add all the needed components for mobile application and establish connection between each of them and respective field on the ThingSpeak channel.

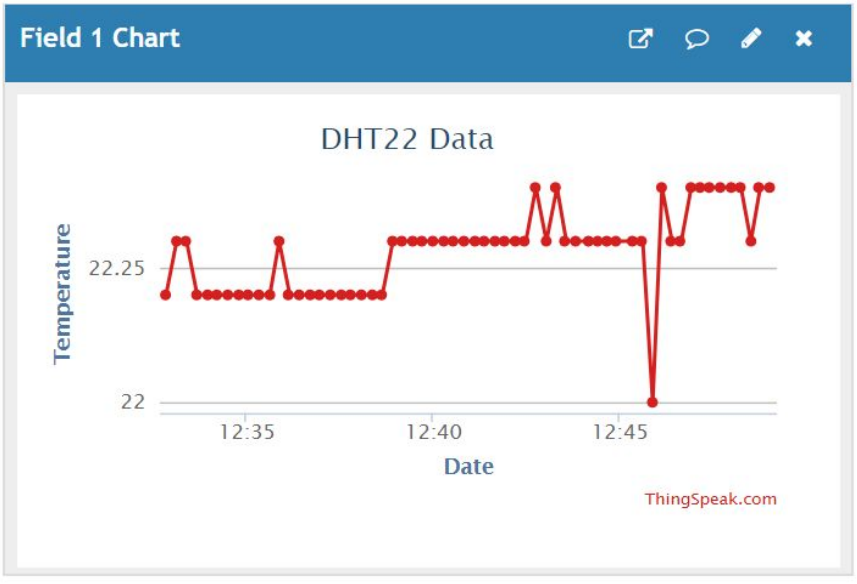


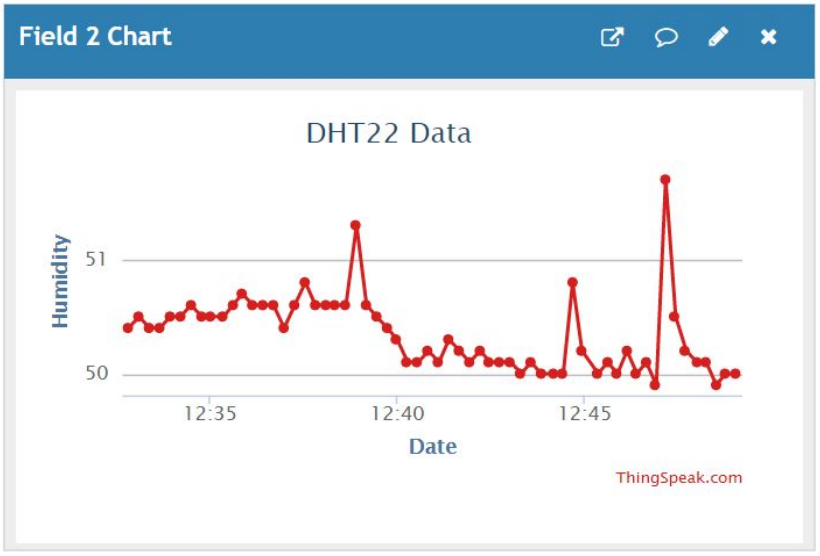
3. Setup the connection between mobile app and ThingSpeak servers.



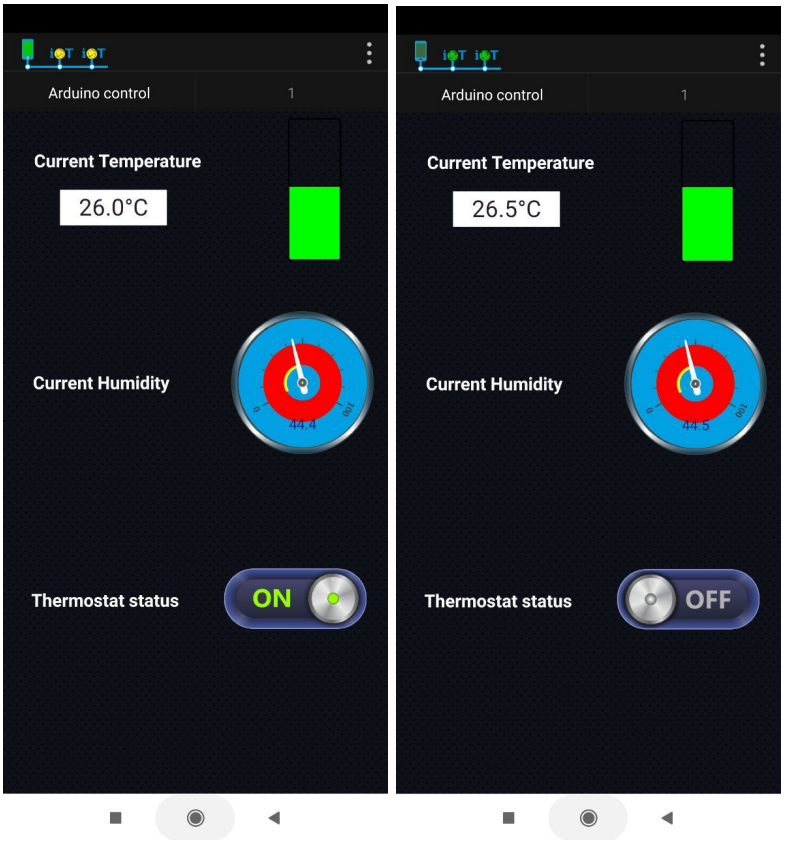
Results

Temperature and Humidity monitoring in the cloud:

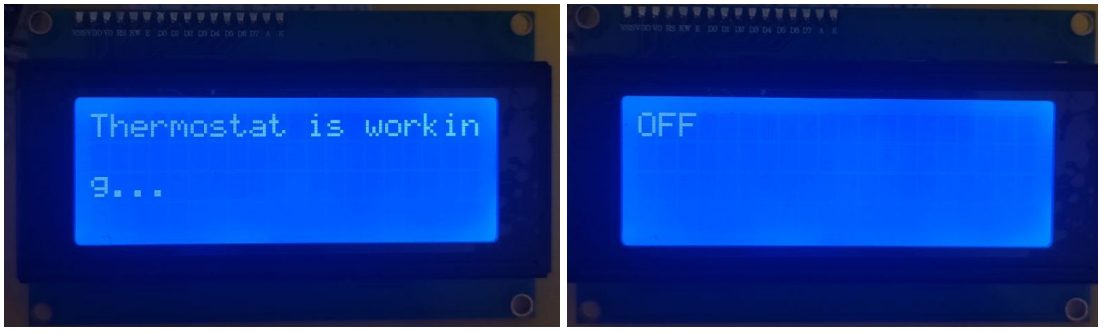




Controlling Arduino from Virtuino app:



Thermostat simulation:



References

1. https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/
2. http://www.robotpark.com/DHT22-Temperature-and-Humidity-Sensor
3. https://www.adafruit.com/product/385
4. https://www.fontenay-ronan.fr/wi-fi-temperature-and-humidity-sensor-arduino-dht22-esp8266-lcd/
5. https://dronebotworkshop.com/lcd-displays-arduino/
6. https://modernelectronics.com.pk/product/i2c-lcd-module/
7. https://www.arduino.cc/en/Guide/ArduinoUnoWiFiRev2
8. https://iot-guider.com/arduino/hardware-basics-of-arduino-uno-wifi-rev2/
9. https://thingspeak.com/pages/learn_more
10. https://it.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html

Source code

```
#include <WiFiNINA.h>
#include <DHT.h>
#include <ThingSpeak.h>
#include <SPI.h>

/*------------------------LCD-------------------------------*/
#include <Wire.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h>
hd44780_I2Cexp lcd;
const int LCD_COLS = 20;
const int LCD_ROWS = 4;
/*----------------------------------------------------------*/

/*------------------------DHT SENSOR------------------------*/
#define DHTPIN 2         // DHT data pin connected to Arduino pin 2
#define DHTTYPE DHT22     // DHT 22 (or AM2302)
DHT dht(DHTPIN, DHTTYPE); // Initialize the DHT sensor
/*----------------------------------------------------------*/

/*-----------------------ThingSpeak-------------------------*/
char thingspeakAddress[] = "api.thingspeak.com";
unsigned long channelID = 890503;
unsigned long commandsChannelID = 908059;
String apiKey = "BBOMQTFH9H3JK8XM"; //API write key from ThingSpeak data
channel
const char* readAPIKey = "BQWVR26DB8S7F8FR"; //API read key from ThingSpeak
data channel
const char* writeAPIKeyCommands = "UWP1VPP37FMUAOUE"; //API write key from
ThingSpeak commands channel
const char* ssid = "RL"; //SSID of your wifi
const char* password = "zxcvbnm1605"; //password of your wifi
int duration=5;//delay between each data measure and uploading
```

```
/*-----------------------------------------------------------*/

/*-----------------------------------------------------------*/
unsigned int tempField = 1; //ThingSpeak Field ID
unsigned int humField = 2; //ThingSpeak Field ID
long lastReadTime = 0;
float responseValue = 0;
/*-----------------------------------------------------------*/

int status = WL_IDLE_STATUS;
WiFiClient client; //Start client

void setup()
{
  Serial.begin(115200);

  dht.begin();

  lcd.begin(LCD_COLS, LCD_ROWS);

  ThingSpeak.begin(client);

  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv != "1.0.0") {
    Serial.println("Please upgrade the firmware");
  }

  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    Serial.println(ssid);
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, password); // Connect to WPA/WPA2 network. Change
this line if using open or WEP network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected");
  }
}

void loop()
{
  char t_buffer[10];
  char h_buffer[10];
  int statusCode = 0;
```

```
  //Read temperature and humidity values from DHT sensor:
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  String t=dtostrf(temp,0,5,t_buffer);
  String h=dtostrf(hum,0,5,h_buffer);
  //Update values on ThingSpeak platform
  updateThingSpeak("field1="+t+"&field2="+h);
  delay(1000);

  if (millis() - lastReadTime > 15000) {
    float value = ThingSpeak.readFloatField(commandsChannelID, 1); //read
command to the thermostat from ThingSpeak
    Serial.println("New ThingSpeak command = " + String(value));
    if (value == 0) {
      lcd.clear();
      lcd.print("OFF");
    }
    else {
      lcd.clear();
      lcd.print("Thermostat is working...");
    }
    responseValue = value;
    lastReadTime = millis();
  }
}

void updateThingSpeak(String tsData)
{
    if (client.connect(thingspeakAddress, 80)){
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");
    client.print(tsData);
    if (client.connected())
    {
      Serial.println();
    }
    else {
      Serial.println("Connection to ThingSpeak failed.");
      Serial.println();
    }
  }
}
```