**Maharaja Agrasen Institute of Technology**

*To whom it may concern*

We, Shubham Aggarwal, Enrollment No. 02014802814 and Atmaja Chowdhury, Enrollment No. 03414802814, students of Bachelors of Technology (ECE), a class of 2014-2018, Maharaja Agrasen Institute of Technology, Delhi hereby declare that the In-House Training Project report entitled **‘IoT-Based Climate Controller’** is an original work , under the guidance of **Mr P.K Chaudhry**, **Asst Prof.** Department of **Electronics and Communication Engineering,** Maharaja Agrasen Institute of Technology Rohini and the same has not been submitted to any other Institute for the award of any other degree.

Date: 07.04.2017

Place: New Delhi, India

**Signature of Faculty In-charge**

**ACKNOWLEDGMENT**

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. I consider myself privileged to express gratitude and respect towards all those who guided us through the completion of this project.

We would also like to thank Prof. P.K Choudhary, our faculty guide for giving his valuable inputs into the project. Furthermore, thanking Prof. S.K. Kundu, our faculty coordinator for his guidance.

We are also grateful to Dr Neelam Sharma, Head of the Department Electronics and Communication Engineering at Maharaja Agrasen Institute of Technology Rohini,  for giving us the support and encouragement that was necessary for the completion of this Project.

**ABSTRACT**

Internet of Things, popularly known as IoT, has expanding applications in every field. It is mostly defined as the interconnection of different devices via use of electronics or equipment’s such as sensors or actuators. IoT has the capability to solve our day-to-day problems as well as large-scale problems of environment or warfare.

Basically, IoT allows the elements in the network architecture to be accessed remotely, thus integrating the physical world into the virtual (network) layer. When IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids, virtual power plants, smart homes, intelligent transportation and smart cities.

Home automation system is one of the most important applications of IoT in our day-to-day lives. From automatic burglar alarm system to water level detector, a lot of equipment installed in today’s homes could see elements of electronics embedded in it. It is an ever expanding field in IoT application and our project is based on the larger ecosystem of smart homes.

Our project, titled ‘IoT based Climate Controller’ is an inter-network between heater, air-conditioner and our smartphone to control the temperature of the room using these elements. The main purpose of the system is to eliminate the need of manual intervention using remote control etc. and automate the working of the elements in the network.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| S.No. | Content | Page No. |
| 1. | Objective | 5 |
| 2. | Elements of IoT-based climate controller | 7 |
| 3. | Working | 17 |
| 4. | Observations | 22 |
| 5. | Result and Conclusion | 24 |
| 6. | References | 26 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Figure No. | Name of the figure | Page No. |
| Fig. 1 | Pin configuration of general Arduino microcontroller | 10 |
| Fig. 2 | Block Diagram of Arduino Uno | 11 |
| Fig. 3 | Block Diagram of ESP Module | 13 |
| Fig. 4 | Diagram of working of single relay switch | 14 |
| Fig 5 | Graph showing variations in temperature and humidity in different days | 16 |
| Fig.6 | Graph showing variations in heat index and speed of sound in different days | 17 |

**LIST OF PHOTOGRAPHS**

|  |  |  |
| --- | --- | --- |
| Photo No. | Name of the photograph | Page No. |
| Photo 1. | ARDUINO UNO | 11 |
| Photo 2. | ESP8266 MODULE | 14 |
| Photo 3. | DHT11 SENSOR | 14 |
| Photo 4. | SONGLE SINGLE PHASE RELAY | 17 |
| Photo 5. | IoT based climate controller with Arduino and Single relay switch | 25 |
| Photo 2. | Snapshot of app showing different temp. for different days | 25 |

**CHAPTER 1**

**OBJECTIVE**

1.1 INTRODUCTION

IoT-based climate controller uses various elements of electronics such as microcontroller, switch, sensors etc. to achieve a real-life climate controller using the temperature-controlling equipments of the house. In this case, we use only one temperature controlling equipments: heater or the air-conditioner or fan or any cooling device. The user would control the equipments via data of temperature, humidity etc. to switch on any one equipment at a time. This eliminates the human intervention to switch on/off the fan/heater when needed.

Secondly, the monitoring of weather is really helpful in various applications like in critical scientific systems or for simulation purposes. In other fields like agriculture, disaster management and medical suited environments. Weather sensing is one of the major functions in aerospace applications to check suited weather environments of other planets too. The need for this project came from the support of a fact of very low popular devices and instruments are available that can provide you live weather results. On top of that requirement of accessing it anywhere.

Weather forecasting is done using predicting the weather and values obtained from sensors or instruments. We human use an approach of algorithms having certain or no input and valid output. Considering there is nothing random in nature and everything around us follows a particular pattern.

On the basis of these weather forecasting patterns people can take precautions on even harsh weather conditions. The wireless arduino weather station has a capability of working on low power. Hence it is not much dependent on power source.

1.2 OBJECTIVE

To create an IoT-based climate controller and weather station using Arduino microcontroller and ESP8266 module.

1.3 EQUIPMENTS USED

1. Arduino Uno microcontroller
2. ESP8266 module
3. DHT11 Sensor
4. Single/Double relay switch
5. Single-strain wire
6. PCB board etc.

**CHAPTER 2**

**ELEMENTS IN IOT-BASED MICROCONTROLLER**

* 1. Arduino microcontroller

Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on our computer, used to write and upload computer code to the physical board.

 Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – we can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduinos have the majority of these components in common:

Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from our computer or a wall power supply that is terminated in a barrel jack.

**We should not** use a power supply greater than 20 Volts as we will overpower (and thereby destroy) our Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts.

Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on our Arduino are the places where we connect wires to construct a circuit (probably in conjuction with a breadboard and some wire). They usually have black plastic ‘headers’ that allow us to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

* **GND** : Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground our circuit.
* **5V & 3.3V**: As we might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
* **Analog**: The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor
* **Digital** : Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
* **PWM**: We may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM).
* **AREF**: Stands for Analog Reference. Most of the time we can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Arduino has a reset button. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if our code doesn’t repeat, but we want to test it multiple times.

Power LED Indicator

Just beneath and to the right of the word “UNO” on our circuit board, there’s a tiny LED next to the word ‘ON’. This LED should light up whenever we plug our Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong.

TX RX LEDs

TX is short for transmit, RX is short for receive. These pins are responsible for [serial communication](https://learn.sparkfun.com/tutorials/serial-communication). In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we’re loading a new program onto the board).

Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit . Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC’s from the ATMEL company. This can be important, as we may need to know the IC type (along with our board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC.

Voltage Regulator

The voltage regulator is not actually something we can (or should) interact with on the Arduino. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don’t apply power to Arduino to anything greater than 20 volts.

PIN CONFIGURATION OF ARDUINO

Power

The Arduino/Genuino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply from 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

* Vin. The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). We can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
* 5V.This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage our board. We don't advise it.
* 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
* GND. Ground pins.
* IOREF. This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

Memory

The ATmega328 has 32 KB (with 0.5 KB occupied by the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](https://www.arduino.cc/en/Reference/EEPROM)).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode()](https://www.arduino.cc/en/Reference/PinMode), [digitalWrite()](https://www.arduino.cc/en/Reference/DigitalWrite), and [digitalRead()](https://www.arduino.cc/en/Reference/DigitalRead) functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50k ohm. A maximum of 40mA is the value that must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

* Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
* External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
* PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
* LED: 13. There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
* TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.  
There are a couple of other pins on the board:

* AREF. Reference voltage for the analog inputs. Used with analogReference().
* Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

Arduino/Genuino Uno has a number of facilities for communicating with a computer, another Arduino/Genuino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](https://www.arduino.cc/en/Guide/Windows#toc4). The Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. 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 (but not for serial communication on pins 0 and 1).

A [Software Serial library](https://www.arduino.cc/en/Reference/SoftwareSerial) allows serial communication on any of the Uno's digital pins.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino/Genuino Uno board is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino Software (IDE) uses this capability to allow we to upload code by simply pressing the upload button in the interface toolbar. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

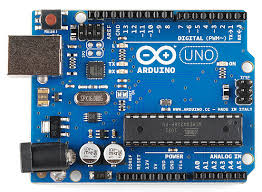


Photo 1. ARDUINO UNO

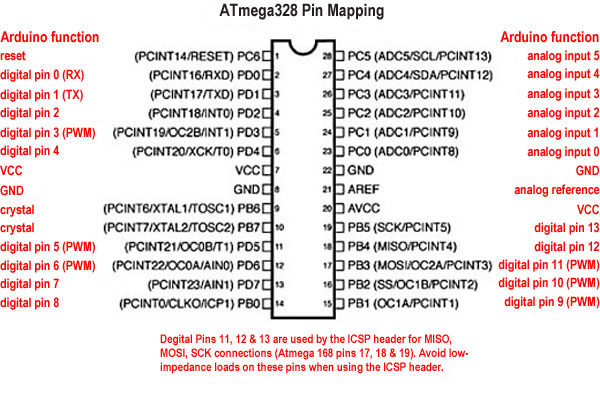


Fig 1. Pin configuration of general Arduino microcontroller

Fig 2. Block Diagram of Arduino Uno

ESP8266 module

[3]The ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to our WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, we can simply hook this up to our Arduino device and get about as much WiFi-ability as a WiFi Shield offers .The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs. Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area.

**Features:**

* 802.11 b/g/n
* Wi-Fi Direct (P2P), soft-AP
* Integrated TCP/IP protocol stack
* +19.5dBm output power in 802.11b mode
* Power down leakage current of <10uA
* 1MB Flash Memory
* Integrated low power 32-bit CPU could be used as application processor

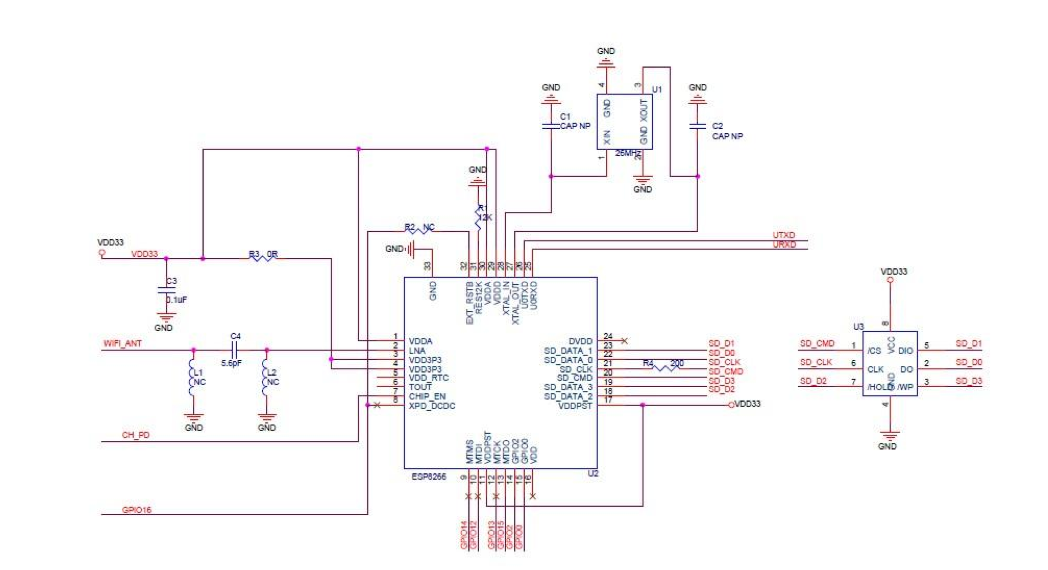


Fig 3. Block Diagram of ESP module

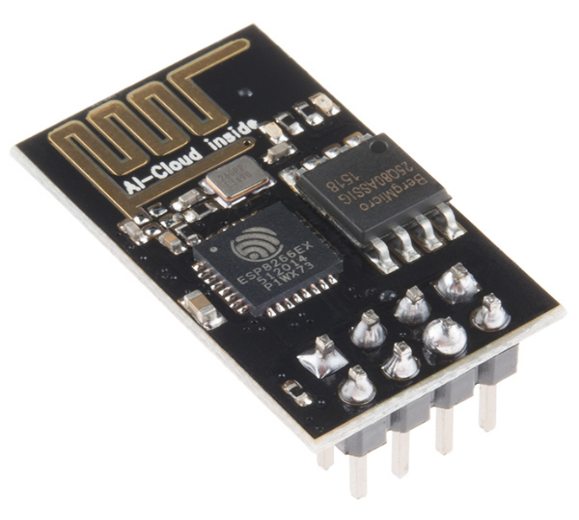


Photo 2. ESP8266 MODULE

* 1. DHT11 Sensor

[4] DHT11 is a basic low cost temperature and humidity sensor. In this project we’ve connected DHT11 sensor to the digital pin 7 of arduino. It consists of 4 pins from left to right Vcc, Data, NC(not connected) and GND. There are mainly three pins which are used. Connecting the ground on ground of arduino and Vcc to 5v output of Arduino.

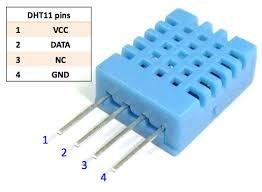


Photo 3. DHT11 SENSOR

* 1. Single/Double Relay Switch

[5]A relay is an [electromagnetic](http://www.explainthatstuff.com/magnetism.html) switch operated by a relatively small [electric](http://www.explainthatstuff.com/electricity.html) current that can turn on or off a much larger electric current. The heart of a relay is an electromagnet (a coil of wire that becomes a temporary [magnet](http://www.explainthatstuff.com/magnetism.html) when electricity flows through it). We can think of a relay as a kind of electric [lever](http://www.explainthatstuff.com/toolsmachines.html): switch it on with a tiny current and it switches on ("leverages") another appliance using a much bigger current. As the name suggests, many sensors are incredibly *sensitive* pieces of [electronic](http://www.explainthatstuff.com/electronics.html) equipment and produce only small electric currents. But often we need them to drive bigger pieces of apparatus that use bigger currents. Relays bridge the gap, making it possible for small currents to activate larger ones. That means relays can work either as switches (turning things on and off) or as amplifiers (converting small currents into larger ones)

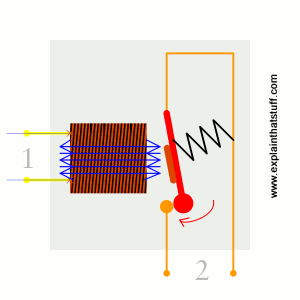


Fig. 4 Diagram of working of a single relay switch

When power flows through the first circuit (1), it activates the electromagnet (brown), generating a magnetic field (blue) that attracts a contact (red) and activates the second circuit (2). When the power is switched off, a [spring](http://www.explainthatstuff.com/how-springs-work.html) pulls the contact back up to its original position, switching the second circuit off again.

This is an example of a "normally open" (NO) relay: the contacts in the second circuit are not connected by default, and switch on only when a current flows through the magnet. Other relays are "normally closed" (NC; the contacts are connected so a current flows through them by default) and switch off only when the magnet is activated, pulling or pushing the contacts apart. Normally open relays are the most common.



Photo 4. SONGLE SINGLE PHASE RELAY

**SOFTWARE USED:**

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them.  
  
Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors.

The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

**CHAPTER 3**

**WORKING**

3.1 OPERATION

* 1. With the help of DHT11 sensor, the device measure various parameters such as speed of sound, humidity, temperature etc.
  2. This data is sent to the user’s smartphone via ESP module in an app pre-installed in the smartphone.
  3. Then according to the threshold temperature set the device sends message and starts up the machine attached to it. If the temperature is greater than upper threshold temperature, the fan starts.

The device works by taking readings from various sensors at different pins in arduino microcontroller. For this purpose we’ve used an arduino compatible .

All the sensors are connected using a breadboard. We’ve used DHT11 temperature sensor to get the temperature and humidity readings connected to digital pin 7 on board for input signals. It gives us continuous reading of surrounding environment in the range of two to three seconds

the arduino connect to the local internet connection providers and connect.

Its job is to transmit the data to a website linked to it and visualize the data over there for every twenty seconds. The website for this project is an open source IOT(Internet of Things) website named Thingspeak by a community of Mathworks. Thingspeak website provide API write key and API read key for each of its own purpose.

secondly , we have used a single phase electromagnetic relay .

When the Dth11 sensor sends data to arduino of the current temperature then the code in arduino compares the current temperaure with a set temperature "T". If the current temperauture is less than "T" then, it sends a high signal to relay and relay behaves as open loop and if it is more than "T" then, it sends a low signal and relay behaves as closed loop and the corresponding electric device connected to relay runs.

Thingspeak is an open data platform for the Internet of Things. It sends data to the cloud. Using this we can analyze and visualize our data. Finally on the basis of those we can react or trigger an action. It provides real time data collection and other devices and technologies like Particle Photon, Raspberry Pi, Twitter, Electric Imp etc.

On the basis of this project, we calculated the following Fields:

1. Humidity

2. Temperature

3. Heat Index

4. Speed of Sound

* 1. **CODE**

#include <ESP8266WiFi.h>

// www.arduinesp.com

//

// Plot DTH11 data on thingspeak.com using an ESP8266

// April 11 2015

// Author: Jeroen Beemster

// Website: www.arduinesp.com

#include <DHT.h>

float sound;

// replace with your channel’s thingspeak API key,

String apiKey = "RI5288HWE5CFESMS";

const char\* ssid = "TP-LINK\_30A26C";

const char\* password = "Ayush123@";

const char\* server = "api.thingspeak.com";

#define DHTPIN 2 // what pin we’re connected to

DHT dht(DHTPIN, DHT11,15);

WiFiClient client;

void setup() {

Serial.begin(115200);

delay(10);

dht.begin();

WiFi.begin(ssid, password);

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

}

void loop() {

if(WiFi.status() != WL\_CONNECTED){

WiFi.begin(ssid,password);

}

float h = dht.readHumidity();

float t = dht.readTemperature();

float hic = dht.computeHeatIndex(t,h, false);

if (isnan(h) || isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return;

}

sound = 331.3 + (0.606\*t);

// code for relay ,to swith on the appliance

if ( t <= 34 )

{digitalWrite (11, HIGH) };

else (t >= 34 )

{digitalWrite (11, LOW) };

//code ends here for relay

if (client.connect(server,80)) { // "184.106.153.149" or api.thingspeak.com

String postStr = apiKey;

postStr +="&field1=";

postStr += String(t);

postStr +="&field2=";

postStr += String(h);

postStr +="&field3=";

postStr += String(hic);

postStr +="&field4=";

postStr += String(sound);

postStr += "\r\n\r\n";

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(postStr.length());

client.print("\n\n");

client.print(postStr);

Serial.print("Temperature: ");

Serial.print(t);

Serial.print(" degrees Celcius Humidity: ");

Serial.print(h);

Serial.print("Heat Index");

Serial.print(hic);

Serial.print("speed of sound :");

Serial.print(sound);

Serial.println("% send to Thingspeak");

}

client.stop();

Serial.println("Waiting…");

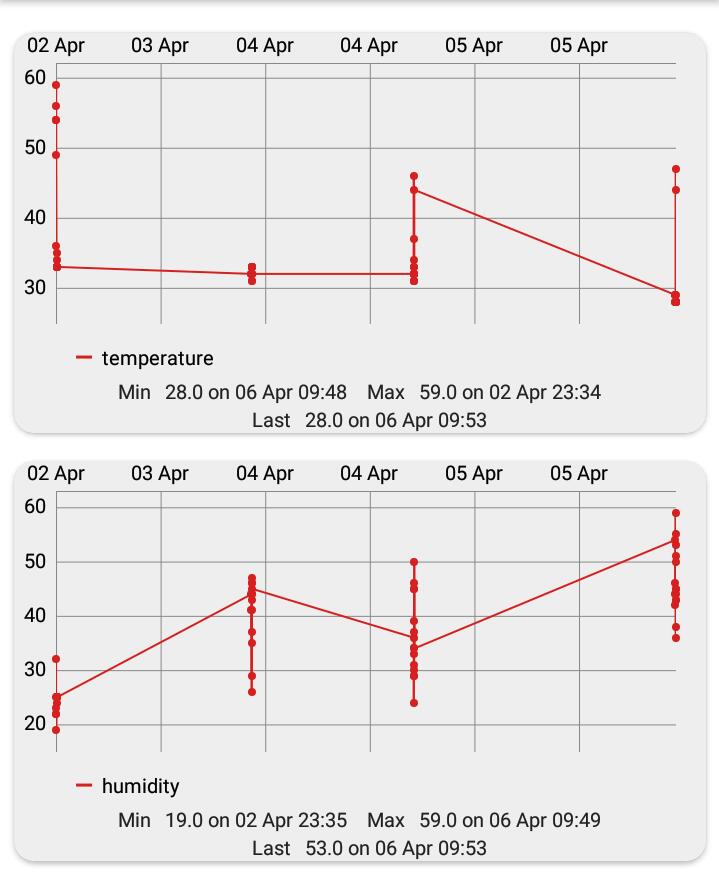
// thingspeak needs minimum 15 sec delay between updates

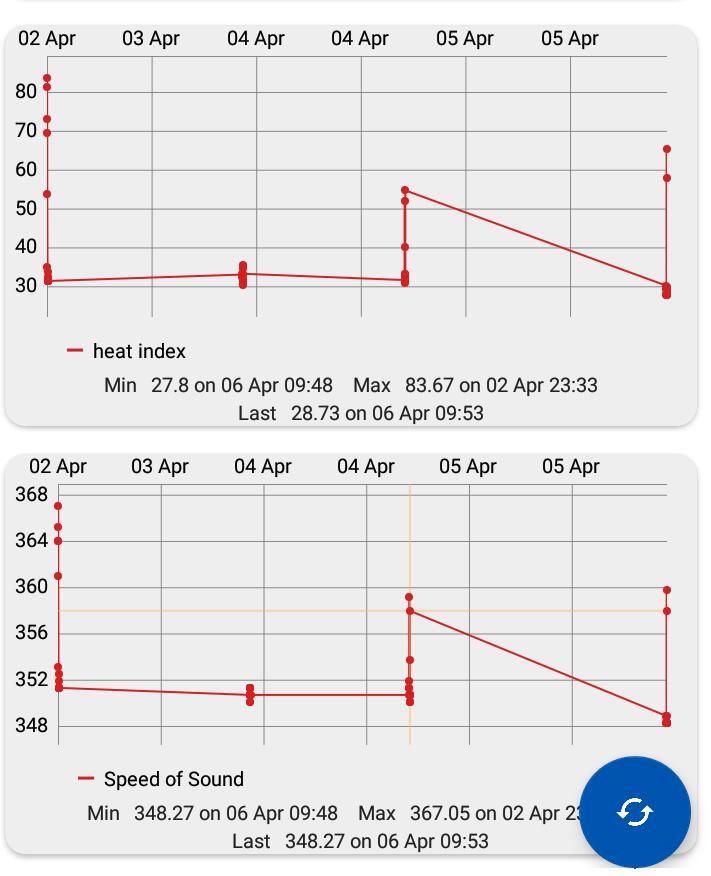
delay(15000);

}

**CHAPTER 4**

**OBSERVATIONS**

The graphs show variations for different parameters in different days from 2nd April to 5th April 2017: 



**CHAPTER 5**

**RESULT AND CONCLUSION**

5.1 RESULT

IoT-based climate controller is constructed and operates at a threshold temperature of 34 Celsius.

5.2 CONCLUSION AND DISCUSSION

From this experiment, we have been able to successfully demonstrate a climate controller using a microcontroller and a relay switch. This project, when scaled up to connect various temperature controlling devices can be easily installed at home to automatically control them without human intervention.

Some of the advantages of this project are as follows:

* The switch can be changed to two-phase or multi-phase relay switch to control two or more devices simultaneously.
* The devices can be switched on/off without any manual intervention.
* Real-time data on parameters such as temperature, humidity, speed of sound etc. is available easily.
* The climate-controller is inexpensive and scalable, thus making is a viable option to install in every home.

We can build on this project by adding multi-phase switches for controlling more devices. Adding different threshold temperatures for switching on/off different devices can also be done in this project to make it more versatile and useful.

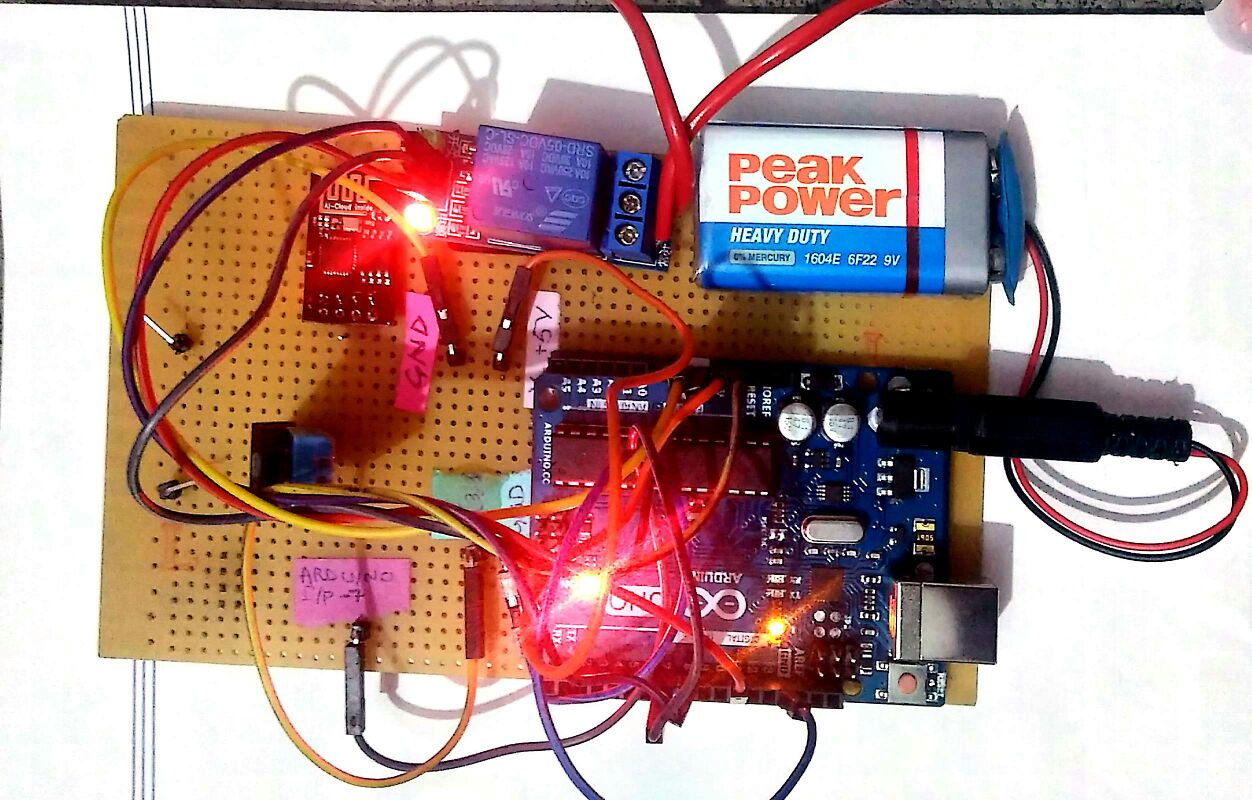


Photo 5. IoT-based climate controller with Arduino Uno and single-relay switch

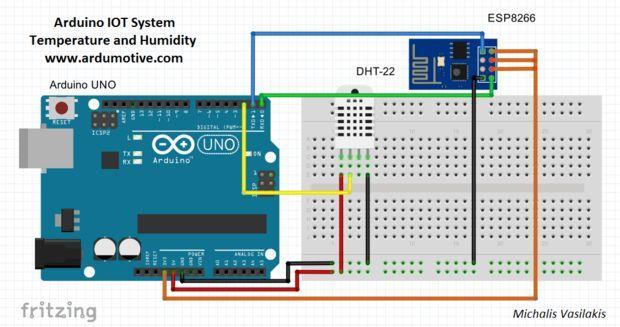
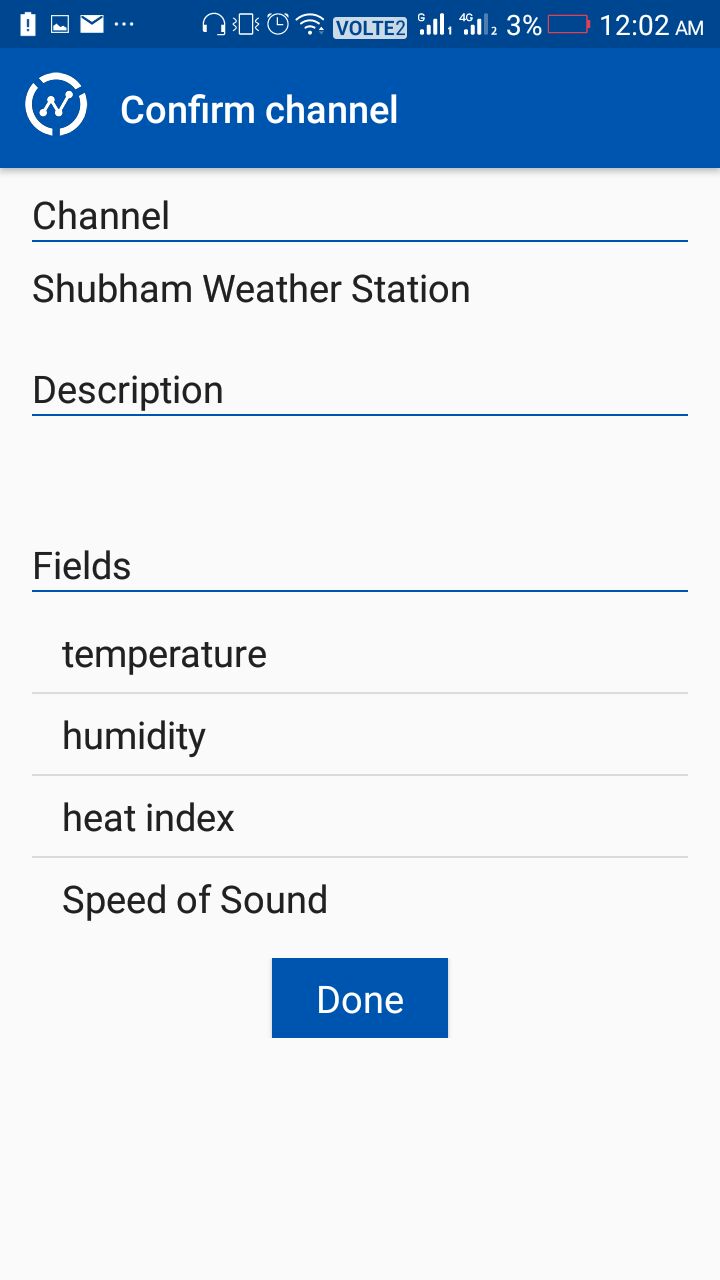
 

Photo 6.Snapshot of app showing observation w.r.t. different fields for each day

**REFERENCES**

[1] <https://learn.sparkfun.com/tutorials/what-is-an-arduino>

[2]Arduino datasheet

[3] <https://www.sparkfun.com/products/13678>

[4] <https://bandg.com/nz/product/dst-800-sensor/>

[5] <http://www.explainthatstuff.com/howrelayswork.html>