

Nagendra Dangi

Monitoring environmental parameters: humidity and temperature using Arduino based microcontroller and sensors

Microcontroller based building monitoring system

Helsinki Metropolia University of Applied Sciences

Bachelor's Degree

Degree Programme in Environmental Engineering

Thesis

14.11.2017

Author(s) Title	Nagendra Dangi Monitoring environmental parameters: humidity and temperature using Arduino based microcontroller and sensors
Number of Pages Date	25 pages + 11 appendices 14 November 2017
Degree	Bachelor's of Engineering
Degree Programme	Environmental Engineering
Specialisation option	Renewable Energy Technology
Instructor(s)	Antti Tohka, Principal Lecturer and Supervisor Dr. Minna Paananen-Porkka (Language Advisor)
<p>Arduino is a newly developed open source hardware and software system. Yet it has drawn the attention of a large technology community. Its less technical design and affordable cost are basic features that enlarges its large volume of use, where the compatibility with many other electronics and possibility of extension are interesting characteristics that increase the range of its use. Arduino hardware is simply a motherboard which can be used to make interacting objects with suitable computer programming (IDE-Integrated Development Environment).</p> <p>The idea of this thesis was to build an Arduino-based embedded device for monitoring environmental variables: humidity and temperature and to study its performance in different temperature and humidity. The device was built using the microcontroller Arduino and sensors, which could sense the temperature and amount of moisture inside a building and provide information in a serial monitor and a liquid crystal display. Out of many clones and different available microcontroller boards, Arduino Uno was used in this thesis project.</p> <p>This thesis has two parts-a theoretical part which gives a basic introduction to the materials and equipment used during the project and the second part provides step-wise process for connection and circuitry. The project was successful to meet pre-determined goals, implementation was possible with the help of the Arduino book, previous work of another student and related internet sites where most of the information is available.</p>	
Keywords	Arduino, microcontroller, monitoring, IDE, codes

Acknowledgement

I would like to acknowledge Antti Tohka (HOD, Metropolia UAS) for his instructions and supervision throughout the process of writing this thesis. My sincere thanks go to Dr. Minna Paananen-Porkka (Senior Lecturer), for her academic guidance. I am grateful to all my teachers, family and friends. Special thanks to Gopal Dangi for his priceless support and Bijay Karki for his assistance.

November 2017,
Nagendra Dangi

Contents

Abstract

Acknowledgement

List of Figures	1
List of Graphs	1
Abbreviations	1
1 Introduction	2
2 Background	3
3 Materials and Method	4
3.1.1 Arduino	4
3.1.2 Sensors	5
3.1.3 IDE	8
3.2 Setup/connection	9
4 Result and analysis	19
5 Conclusion	22
6 References	23

Appendices

Appendix 1. Datasheet Arduino, DHT11

Appendix 2. Serial Monitor readings

List of Figures

Figure 1: Arduino Uno.....	5
Figure 2: DHT11 Sensor	7
Figure 3: Circuit diagram for DHT11	8
Figure 4: Arduino IDE	9
Figure 5: Wiring DHT11 Sensor and LCD display to Arduino Uno.....	11
Figure 6: Sketches for displaying Humidity and Temperature readings in LCD	12
Figure 7: Sketches for displaying Humidity and Temperature readings in LCD	13
Figure 8: Temperature and Humidity display in LCD	15
Figure 9: Schematic diagram for wiring DHT11 sensor with Arduino.....	16
Figure 10: Sketches for DHT11 to give the readings in serial monitor	17
Figure 11: Serial monitor readings	19

List of Graphs

Graph 1: Humidity and temperature plotted against time (Inside the building).....	20
Graph 2: Humidity and temperature plotted against time (outside the building).....	21
Graph 3: Humidity and temperature plotted against time (Inside the building).....	21

Abbreviations

IDE	Integrated Development Environment
LCD	Liquid Crystal Display
LED	Light-Emitting diode
PWM	Pulse-width modulation
HVAC	Heating, Ventilation and air conditioning
ICSP	in Circuit Serial Programming
V	Voltage
RH	Relative Humidity
USB	Universal Serial Bus
Ω	Ohm

1 Introduction

The number of embedded devices that can interact with environment are already connected to internet, and it is estimated that the number reaches 50 billion by 2020 (Kouhia, 2016). The growth of such interacting objects achieved this staggering pace with the development of microcontroller based easy-to-use designed system which are replacing old systems designed with complicated electronic circuits.

Arduino is a microcontroller board which functions as a tiny computer; it is a platform where creation and development of interacting objects is possible with required programming software. The Arduino software IDE (Integrated Development Environment) provides space to write codes in the language (programming languages C, C++) that Arduino board understands and responds to. Inexpensiveness, easy-to-use design and flexibility for advance modifications are some features of the microcontroller based Arduino hardware and software that are making its range of use wider. One of the most important factor that affects its increasing range of use is its freedom of use. Both the Arduino hardware and the software are open source. Which means that one can easily use the ideas generated by others in their work and modify them without anyone's authorization. It can be used by anyone to do anything they want to do with it (Banzi, ei pvm). Arduino boards are designed in such a way that one without prior knowledge of electronics or previous experience of programming can use information from other people's work and build their own interactive object that can sense the environment and control it. It comes with a cheap price which is a crucial factor that makes Arduino accessible to many students, hobbyists and teachers and ultimately a new revolution of innovation in electronics (Banzi, ei pvm).

This thesis is an academic work made in the final year of environmental engineering studies. The purpose of the thesis was to build an Arduino based embedded device for building monitoring (monitoring environmental variables temperature and humidity) and to study the characteristics of its performance. To study its performance characteristics, Arduino was tested in three different temperature and humidity conditions. The different environmental conditions were created by using the device in NTP (normal room temperature/humidity), outdoor temperature/humidity and the insulated wooden box. The result is shown graphically in 'Result and Analysis' section. This thesis includes practical steps to follow for developing the device that can measure temperature and humidity of building or surrounding and display readings in a LCD display (liquid crystal display) and serial monitor as well using Arduino board and sensors. The developed system is useful

in monitoring two variables- temperature and humidity- in a building, laboratory and greenhouse. In this thesis, the required equipment, connections and circuit diagrams with necessary codes were compiled in a step by step order.

The Materials and Methods chapter contains the details of the apparatus used and procedures followed for creating a building monitoring system. An attempt was made to explain the circuit diagrams and methodologies as simply as possible, which makes the procedure easy to follow and repeat even without any experience in programming and electronics.

2 Background

The monitoring of environmental variables such as temperature, pressure and humidity has a long history of development and the variables have shown significant impact in the productivity of plant growth, the quality of food industry and the efficiency of many temperature and humidity-sensitive equipment (Vleeschouwer, et al., 2017). The monitoring of temperature and humidity of laboratories, storages, halls, school and hospitals is important with respect to health and hygiene. The reliable measurement and monitoring is crucial in this competitive era of technology.

Arduino, the open source hardware has shown ability to meet the need of accurate and real-time monitoring and controlling of environmental variables. The Arduino user community is a forum where many people can share their ideas, use each other's work and modify them to innovate and advance many different interacting objects. Arduino is used in a wide range of projects to develop objects that can interact with people or environment and internet.

The materials required are easily available, cheap in price and easy to use with the help of available open source information. Arduino has been used to build robots, drones, remote controllers, monitoring devices and many interesting objects which is a one big step towards making the world more automated and sustainable. Arduino can be told to do such things in appropriate language that Arduino understand: C, C++.

The projects related to monitoring environmental variables are simple and common but always with great importance; many projects on greenhouse building monitoring and

household plant monitoring are available (Akami, et al., 2015). This thesis was expected to be helpful in learning electronics and programming as well as documentation process.

3 Materials and Method

3.1.1 Arduino

The heart part of the building monitoring system; the Arduino is defined in Wikipedia as “an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world” (Anon., 2017). In other words, it can also be defined as a single-board microcontroller for building digital objects and interactive devices. Arduino is designed to sense the environment and/or surrounding by receiving input signal through sensors and communicates with its surrounding through actuators. An actuator could be a simple LED (light emitting diode), a motor or sensors, ethernet or some other electronics depending on the project (kushner, 2011). The Arduino hardware are available in many format and design enabling different features. The programming is based on hardware wiring. The Arduino software can be run on Windows, Linux or Mac OS (Sandhu, 2016).

The Arduino can be programmed to work stand-alone, with computer or other electronic devices; which can be done with Arduino software which generally termed as IDE (Integrated Development Environment). The detail about IDE is given in chapter 3.1.3. Since the Arduino hardware and software is an open source, there are already many clones of Arduino hardware available with many exciting features, the board used in this thesis project is Arduino Uno which is shown in Figure 1.

Arduino Uno board is a microcontroller board based on Atmel Atmega328 8-bit microprocessor. There are 14 digital input and output pins; 6 of which can be used as pulse-width (PWM) outputs. It has 6 Analog inputs and a 16 MHz quartz crystal or oscillator. Arduino Uno board has USB (universal serial bus) cable to connect to a computer, a power jack, an ICSP (In Circuit Serial Programming) header and a reset button (Anon., 2017). In Italian “Uno” means one and the board was named so because the Arduino Uno board is the first in a series of USB Arduino boards. The block diagram and other features of Arduino Uno are provided in Appendix 1.

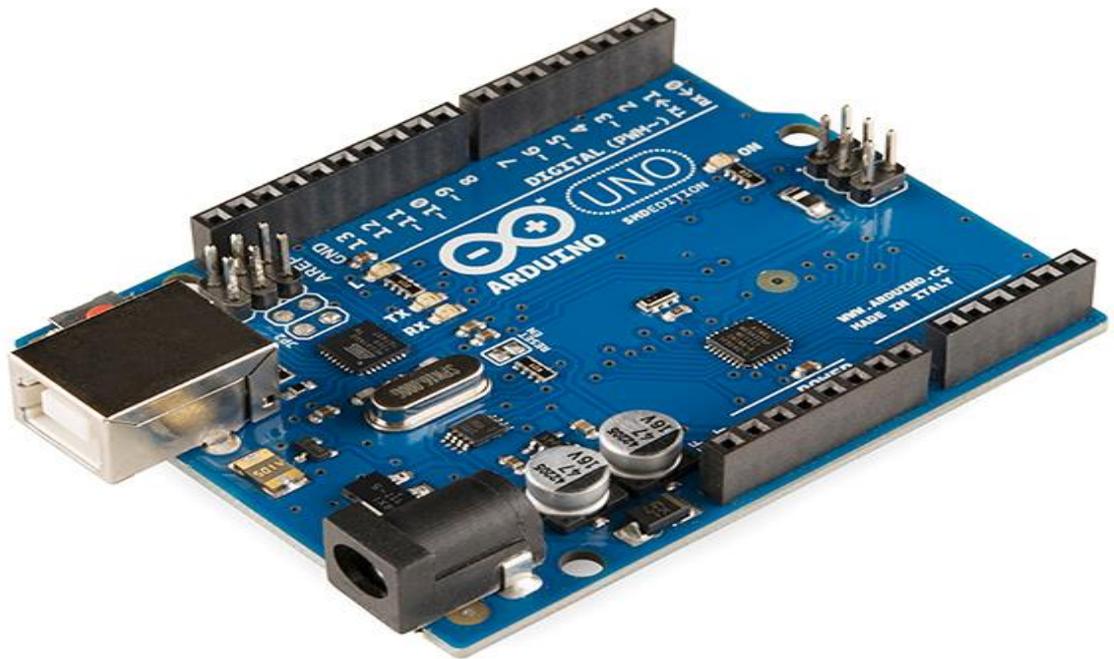


Figure 1: Arduino Uno

3.1.2 Sensors

A sensor is an electronic device that converts a change in physical phenomenon into an electrical signal. It can send the information to computers or other electronic devices. Therefore, it is a part of the interface between the environment or physical world and the electronics (Kenny, 2005). The function of a sensor is to respond to an input physical signal and to convert it into an electrical signal (voltage). It is a semiconductor device that is designed to respond on change in their resistive or capacitive property depending upon the type of sensor. The performance of sensor is characterized by: Transfer function, Sensitivity, Range, Accuracy/Uncertainty, Hysteresis, Linearity, Noise, Resolution and Bandwidth (Wilson, 2005).

Sensors are used in many objects and places, for example, touch-sensitive phone screen, motion sensitive light switch and several applications. The use of sensors has been expanded widely with the development of microcontrollers. The sensors are used in robotics, airplanes and aerospace, cars and many other applications (Blaauw, et al., 2016).

The sensors used in this work is temperature and humidity sensor-DHT11. The sensor-DHT11 is an Analog sensor designed to sense the physical change in heat and moisture when exposed in air with suitable wiring and programming.

Its small size, cheap price, low power consumption, quick responses are the characteristics for being one of the best choices for many users. The sensor DHT11 is applicable in HVAC (heating, ventilation and air conditioning), it can be used in testing and inspecting equipment and consumer goods. It is also applicable to use in building a weather station or a humidity regulator. The use of DHT11 sensor has shown its usefulness measuring and controlling temperature and humidity in home appliances, medical and many other sector (Anon., 2017)

The Figure 2 shows the DHT11 temperature and humidity sensor. The sensor DHT11 has following performance range and accuracy. A detailed datasheet is included in Appendices.

Measurement range:

Temperature:	0 to 50°C
Humidity:	20 to 90 % RH

Accuracy:

Temperature:	±2 %
Humidity:	±5 %

Where the operating Voltage remains between 3V to 5.5V (Anon., 2017)

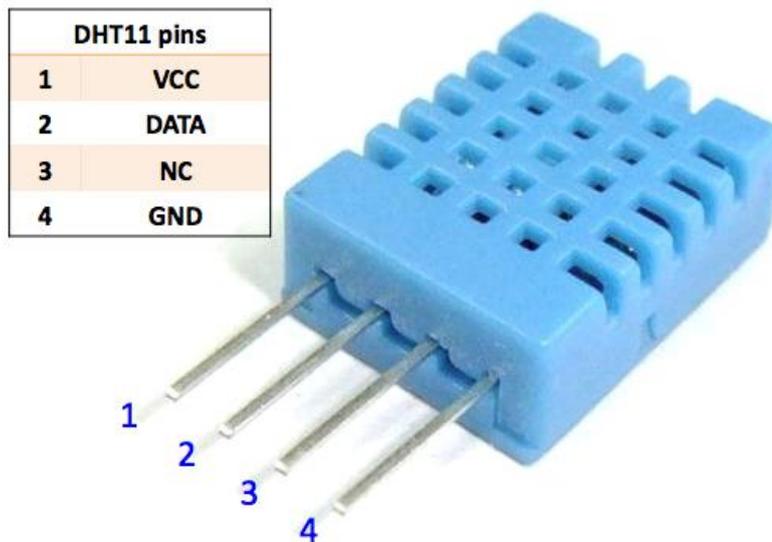


Figure 2: DHT11 Sensor

Humidity is defined as the amount of water vapour contained in air. Usually, it is expressed as absolute humidity, dew point and relative humidity. The sensor used in this thesis project, DHT1, is designed to measure humidity in terms of relative humidity (RH). Relative humidity (RH) is the ratio of the amount of water vapour content of the air to the saturated moisture level at the same pressure or temperature:

$$RH = \frac{\rho_w}{\rho_s} \times 100\% ,$$

Where RH is relative humidity, ρ_w is the density of water vapour, and ρ_s is the density of water vapour at saturation.

The sensor DHT11 detects moisture in the air by measuring the electrical resistance between electrodes. It is fabricated with a moisture holding substrate. When substrate absorbs moisture, ionization takes place and results in the increase in conductivity between the electrodes. The relative humidity is proportional to the change in resistance between electrodes due to moisture absorbed.

The typical circuit diagram for DHT11 is shown in Figure 3 and the detailed datasheet for DHT11 is given in Appendix 2.

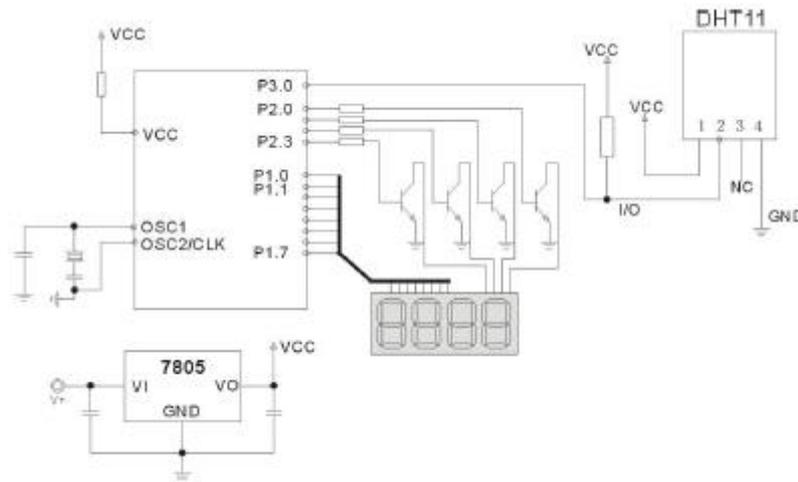


Figure 3: Circuit diagram for DHT11

3.1.3 IDE

The brain part of the building monitoring system, the Arduino IDE (integrated development environment), is a software development environment or software application for Arduino where users can write different kind of computer programs and test. The user can write codes in IDE in a language which an Arduino understands, i.e. C, C++. The program (codes) written in IDE, when uploaded into the Arduino microcontroller determines what and how the system works. The Arduino IDE comes with a 'built-in code parser' that studies the validity of the written codes before sending it to the Arduino. The compilation and translation work is done in IDE after checking the validity of codes. After translating the code, the IDE uploads the program to the Arduino microcontroller (Kouhia, 2016).

IDE software includes the set of different programs that are ready for being tested on the device. Just like in other programming platform, Arduino IDE can also be extended with the use of libraries; the IDE installation includes the installation of number of libraries (Anon., 2017).

The software page where Arduino codes are written looks like as shown in Figure 4. It has two main functions 'setup()' function and 'loop()' functions. The setup part is where the codes should be written so that the program runs and the loop part is where the codes should be written so that the program runs with repetition until the power off or reset button is pushed. It allows users to program and edit Arduino to do anything they like to do with it. Depending upon the feature of different boards, the IDE enables communication with Arduino board through USB (Kouhia, 2016). The following figure shows the screen capture of Arduino IDE.

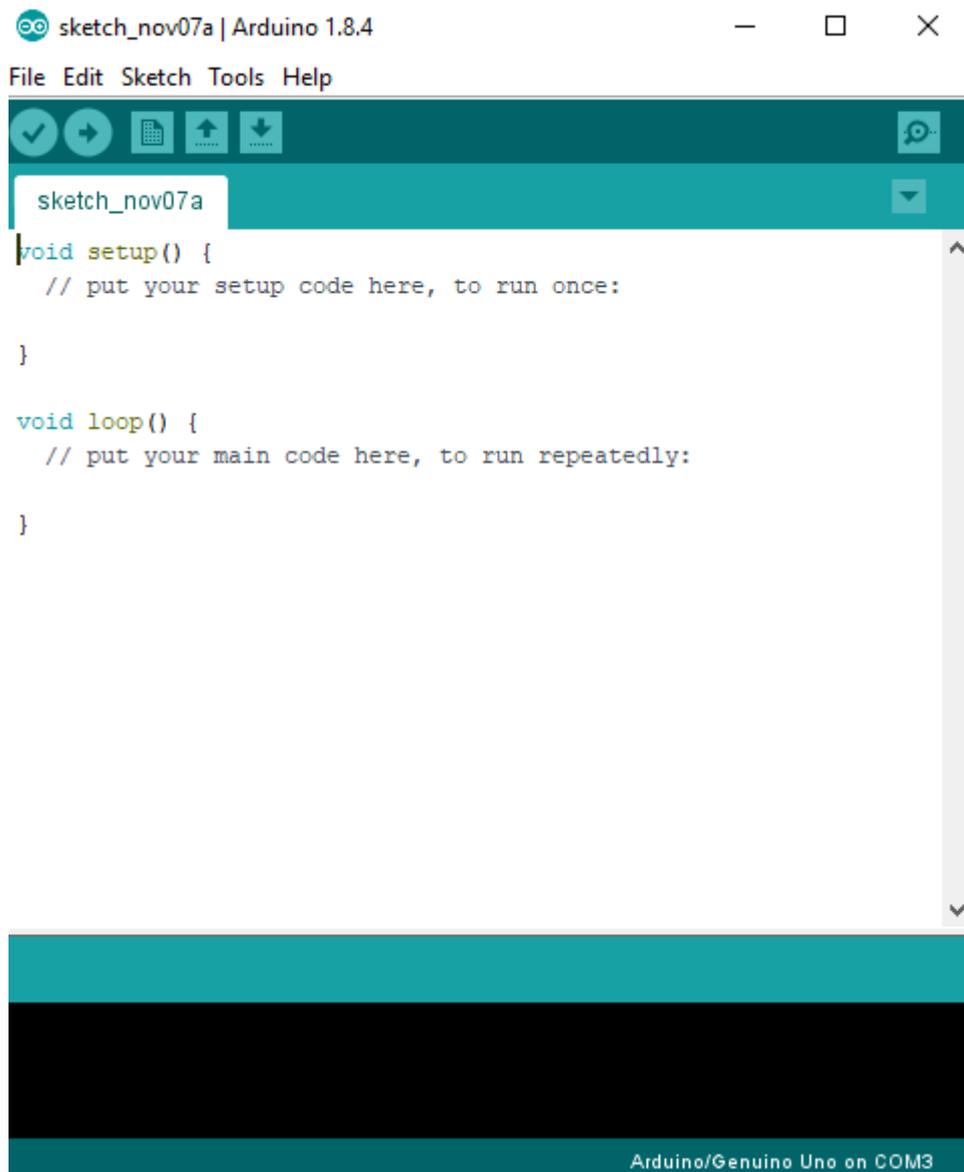


Figure 4: Arduino IDE

3.2 Setup/connection

Arduino can be programmed to give the readings (measurement data) in serial monitor and/or in the LCD display. The wiring and connection is one of the most important part of electronics to work it properly. The required materials for the project are listed below:

- an Arduino board (Arduino Uno)
- a breadboard
- a sensor (DHT11)
- a 220 Ω resistor

- a potentiometer
- jumper wires
- an LCD Display

The required wiring and pins to connect the LCD display and the sensor with Arduino is given below (Circuitbasics, 2017):

- Arduino GND Breadboard -ve power rail
- Arduino 5v Breadboard +ve power rail
- DHT11 -ve pin Breadboard -ve power rail
- DHT11 +ve pin Breadboard +ve power rail
- DHT11 S Arduino Analog pin A0
- LCD 1 Breadboard -ve power rail
- LCD 2 Breadboard +ve power rail
- LCD 3 Potentiometer centre pin
- LCD 4 Arduino Digital pin 12
- LCD 5 Breadboard -ve power rail
- LCD 6 Arduino Digital 11
- LCD 11 Arduino Digital Pin 5
- LCD 12 Arduino Digital Pin 4
- LCD 13 Arduino Digital Pin 3
- LCD 14 Arduino Digital Pin 2
- LCD 15 220 Ω (ohm) resistor and the other pole of 220 Ω resistor to Breadboard positive power rail
- LCD 16 Breadboard -ve power rail
- Potentiometer +ve Breadboard negative power rail

Schematically, the connections for LCD display and DHT11 sensor with Arduino Uno can be shown as in following Figure 5 (Circuitbasics, 2017).

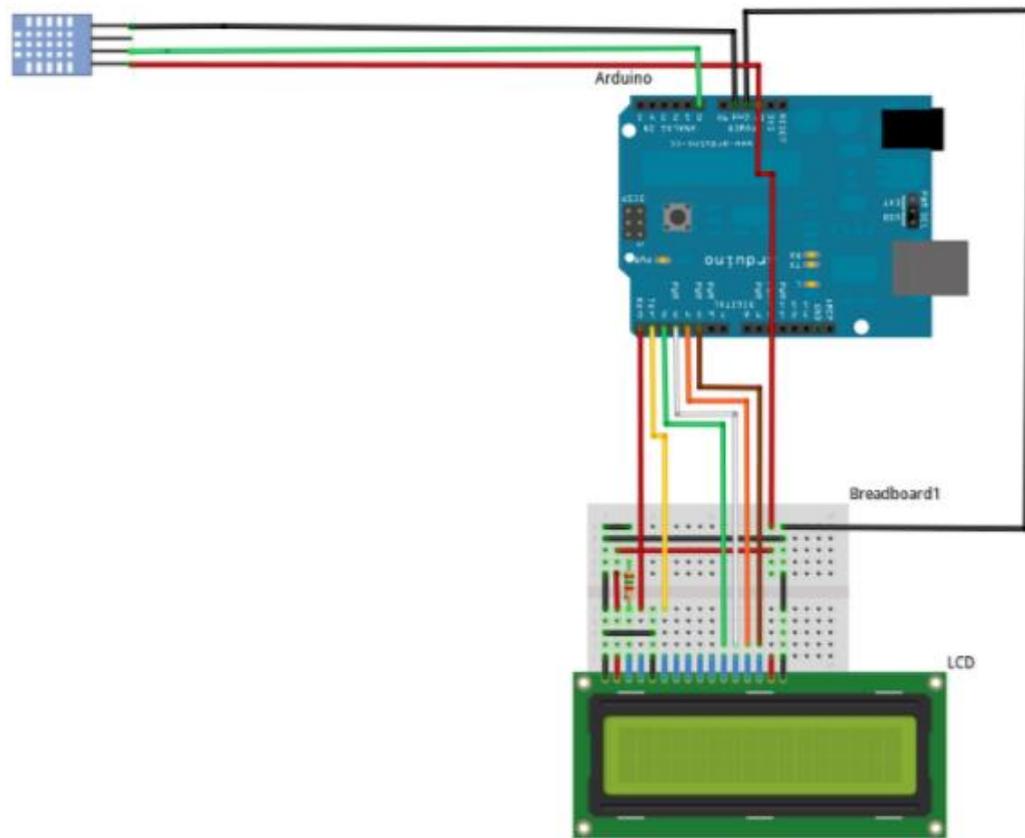


Figure 5: Wiring DHT11 Sensor and LCD display to Arduino Uno

When all the connections and wiring are done, the code should be written in IDE and the codes written in IDE tells the Arduino to function so that the measurement obtained from sensor can be read in LCD display.

The necessary codes for displaying temperature and humidity readings in LCD display are given below after the Figure 6 and Figure 7, which show the codes while program running in IDE.

```

sketch_lcdtemp §
#include <LiquidCrystal.h>
#include <dht.h>
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#define dht_dpin A0 //no ; here. Set equal to channel sensor is on
int val;
int tempPin = 1;
dht DHT;

void setup() {
  lcd.begin(16,2);
  // put your setup code here, to run once:
  lcd.print("Booting Up");
  Serial.begin(9600);
  delay(300); //Let system settle
  Serial.println("Humidity and temperature\n\n");
  delay(700); //Wait rest of 1000ms recommended delay before
  //accessing sensor
}

void loop() {
  // put your main code here, to run repeatedly:
  DHT.read11(dht_dpin);
  lcd.setCursor(0,1);
  val = analogRead(tempPin);
  float mv = ( val/1024.0)*5000;
  float cel = mv/10;
  float farh = (cel*9)/5 + 32;

  lcd.print("TEMP = ");
  lcd.setCursor(0,1);
}

```

Done uploading.

Sketch uses 5516 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 339 bytes (16%) of dynamic memory, leaving 1709 bytes for local variables.

Figure 6: Sketches for displaying Humidity and Temperature readings in LCD

```

sketch_lcdtemp $
delay(300); //Let system settle
Serial.println("Humidity and temperature\n\n");
delay(700); //Wait rest of 1000ms recommended delay before
//accessing sensor
}

void loop() {
    // put your main code here, to run repeatedly:
    DHT.read11(dht_dpin);
    lcd.setCursor(0,1);
    val = analogRead(tempPin);
    float mv = ( val/1024.0)*5000;
    float cel = mv/10;
    float farh = (cel*9)/5 + 32;

    lcd.print("TEMP = ");
    lcd.setCursor(0,1);
    lcd.print(cel);
    lcd.print(farh);
    // lcd.print(farh,"*F")
    delay(500);

    lcd.print("Humidity | = ");
    lcd.print(DHT.humidity);
    lcd.print("% ");
    delay(500);

    lcd.print("temperature = ");
    lcd.print(DHT.temperature);
    lcd.println("C ");
}

Done uploading.

Sketch uses 5516 bytes (17%) of program storage space. Maximum is 32256 bytes.
Global variables use 339 bytes (16%) of dynamic memory, leaving 1709 bytes for local

```

Figure 7: Sketches for displaying Humidity and Temperature readings in LCD

The programming codes displaying humidity and temperature readings in LCD with Arduino and DHT11 sensor are given below. The codes were developed with the help of an original source (Circuitbasics, 2017). It is given in a format that can work by simply copying and pasting in IDE.

```

// to include LCD display
#include <LiquidCrystal.h>
// to include DHT11 sensor
#include <dht.h>
// introduction to pin connected
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#define dht_dpín A0
int val;
int tempPin = 1;
dht DHT;

void setup() {
  lcd.begin (16,2);
  // the codes written here is to run the programme once
  lcd.print("Booting Up");
  Serial.begin(9600);
  delay(300); //Let system settle
  Serial.println("Humidity and temperature\n\n");
  delay(700); // delay helps the system cool down by have pause
}

void loop() {
  // the codes written here is to run programme repeatedly until the power off or
  //stopped.
  DHT.read11(dht_dpín);
  lcd.setCursor(0,1);
  val = analogRead(tempPin);
  float mv = ( val/1024.0)*5000;
  float cel = mv/10;
  float farh = (cel*9)/5 + 32;

  lcd.print("TEMP = ");
  lcd.setCursor(0,1);

```

```
lcd.print(cel);  
lcd.print(farh);  
// lcd.print(farh,"*F")  
delay(500);// delay sets the frequency  
  
lcd.print("Humidity = ");  
lcd.print(DHT.humidity);  
lcd.print("% ");  
delay(500);  
  
lcd.print("temperature = ");  
lcd.print(DHT.temperature);  
lcd.println("C ");  
}
```

After writing, the codes given above should be verified by IDE and when the verification completes the program is ready to be uploaded in Arduino. While the programme is running, the LCD display shows the reading as shown in Figure 8 .

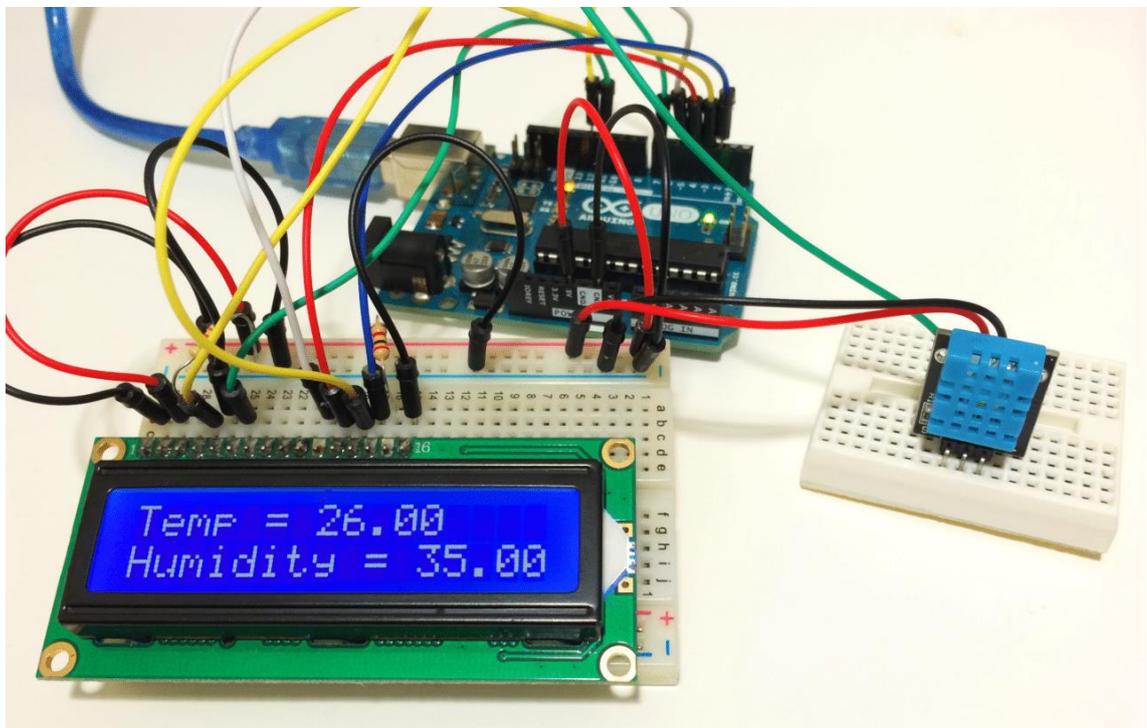


Figure 8: Temperature and Humidity display in LCD

The wiring for connecting the DHT 11 sensor with Arduino to display reading in serial monitor is given below (Circuitbasics, 2017). The data obtained in serial monitor can be exported to other analysing tools.

<u>DHT11 Pin</u>	<u>Arduino Pin</u>
• DHT11 positive pin	Arduino 5v
• DHT11 negative pin	Arduino GND
• DHT11 (S)	Arduino Analog pin A0

The following Figure 9 is the schematic diagram for wiring DHT11 sensor with Arduino to give the data readings in serial monitor (Circuitbasics, 2017).

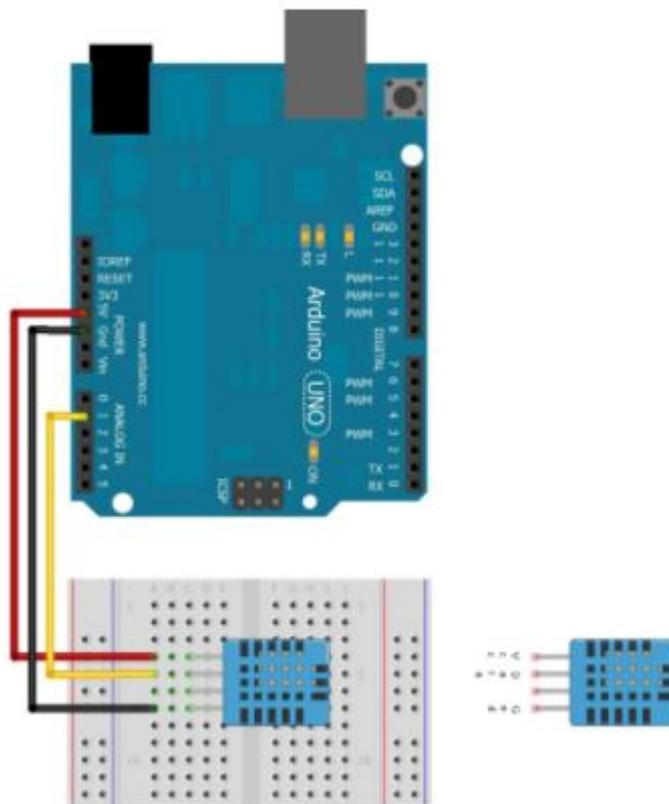


Figure 9: Schematic diagram for wiring DHT11 sensor with Arduino.

After wiring the necessary codes for to write in IDE for humidity and temperature sensor (DHT11) to give data readings in serial monitor are shown below the following Figure 10 which is also included in the codes section right after the Figure 10:



```

sketch_HT $
#include <dht.h>
// to include dht11 sensor

#include <dht.h>

#define dht_dpin A0 //no ; here. Set equal to channel sensor is on

dht DHT;

void setup(){
  Serial.begin(9600);
  delay(300); //Let system settle
  Serial.println("Humidity and temperature\n\n");
  delay(700);
}

void loop(){

  DHT.read11(dht_dpin); // To get the readings.

  Serial.print("Current humidity = ");
  Serial.print(DHT.humidity);
  Serial.print("% ");
  Serial.print("temperature = ");
  Serial.print(DHT.temperature);
  Serial.println("C ");
  delay(5000); // To get the data in regular interval

```

Figure 10: Sketches for DHT11 to give the readings in serial monitor

The above codes and their brief explanation is given below (Circuitbasics, 2017). The codes given below can be directly copied and pasted in IDE to run it.

```

include <dht.h> // to include the DHT11 sensor to system
#define dht_dpin A0

```

```
dht DHT;
// to write the codes to run the program once
void setup(){
  Serial.begin(9600);//
  delay(300);// to let the system settle down
  Serial.println("Humidity and temperature\n\n");
}
// writing codes to run the program repeatedly
void loop(){
  DHT.read11(dht_dp1n);

  Serial.print("Current humidity = ");
  Serial.print(DHT.humidity);
  Serial.print("% ");
  Serial.print("temperature = ");
  Serial.print(DHT.temperature);
  Serial.println("C ");
}
```

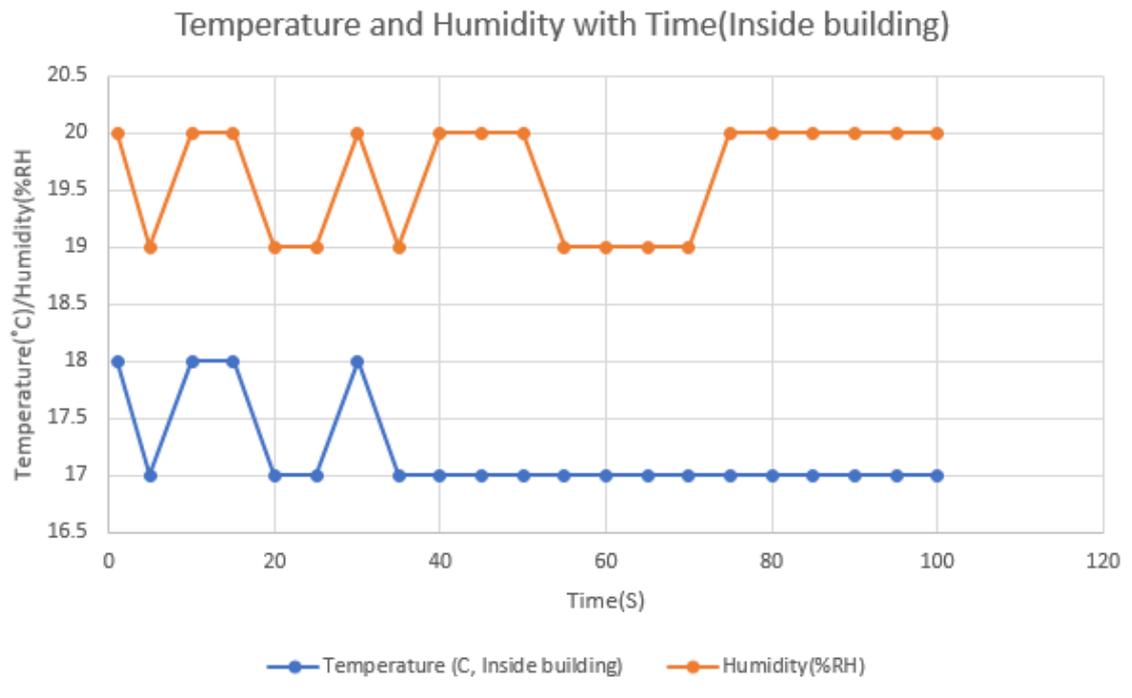
When these codes are written in IDE and uploaded to Arduino, the humidity and temperature measurement starts to appear in serial monitor which is shown in Figure 11. The serial monitor gives the humidity and temperature measurement result in a frequency set as delay() while writing codes.

After wiring and writing the code, the program was run and the built device was successful in measuring humidity and temperature. The LCD display and serial monitor readings are shown in Graph 1 Graph 2 and Graph 3 below.

To study the performance characteristics of the used Arduino-based humidity and temperature sensor, the test was done in three different conditions. The first test was done inside an environmental laboratory room and the other two tests were done inside a box and outside laboratory, respectively. To create a different temperature and humidity, the sensor was kept inside an insulated wooden box and for the 3rd test the measurement was taken outside of the building.

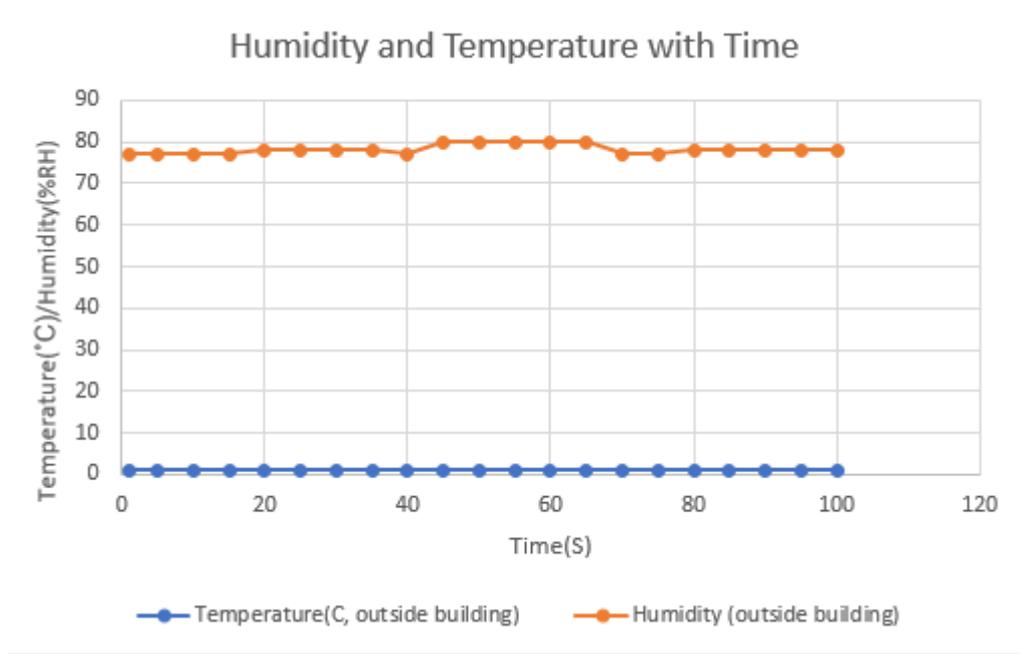
The measurement data was analysed using excel, the original data in serial monitor is given in Appendix 2

The following graphs show humidity and temperature plotted against time.



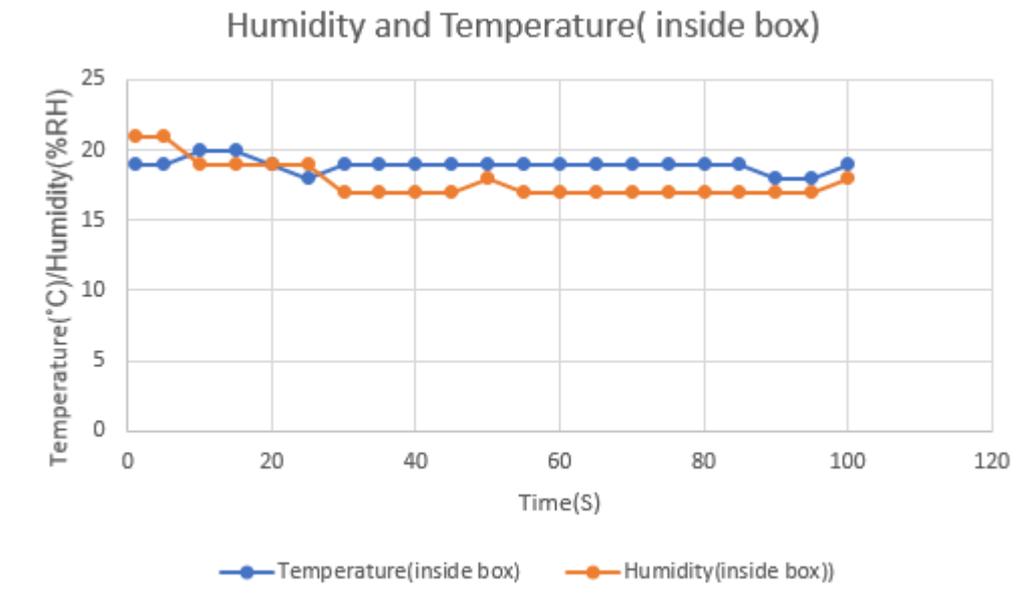
Graph 1: Humidity and temperature plotted against time (Inside the building)

The sensor was again used to measure temperature and humidity outside of the building and the measurement results plotted against time are shown below.



Graph 2: Humidity and temperature plotted against time (outside the building)

The final test was done inside a wooden box. The measurement results are shown against time in the following graph.



Graph 3: Humidity and temperature plotted against time (Inside the box)

The device is upgradable. The pressure sensor can be included. It can be programmed to work offline using Arduino shield. The measurement data can be accessible in other

analysing and programming tools (Excel, MATLAB). The Arduino microcontrollers' platform of development has created a rapid growth of innovations which has resulted in the availability of more advanced Arduino boards and its clones. The measurement was compared to available data and it was accurate and reliable. The cost of the project is less than 60 euro.

5 Conclusion

The work was successful in building a monitoring device which works as a thermometer for measuring temperature and humidity inside a building; it is capable of measuring humidity and temperature outdoors. Compared to expensive sensor, the Arduino-based monitoring system successfully reduces the power consumption, cost and complexity of the process. The performance of the system was accurate and reliable with some error in measurement and limitations of the used sensor.

Arduin- based monitoring devices are the new possibilities for developing smart devices freely with small budget and simple work. The accelerating race of advanced technology outdates the technology used in Arduino Uno in no time; advanced software working similarly are available. The Arduino is programmed to use a USB cable to connect to computer while there are many other boards available with different features.

The project was interesting and was practically helpful to learn to use microcontrollers (Arduino), programming language C and basic electronics. This was a very helpful project in learning and understanding the world of microcontrollers, and using microcontrollers in real life. The thesis project was a platform to advance the technique of research, test and documentation that was learnt throughout the studies in the degree programme.

6 References

Akami, P., Oke, A. & Akpomiemie, O., 2015. Impact of environmental factors on building project. *ScienceDirect*, 11(1), pp. 91-97.

Anon.,2017. *ArduinoUnoRev3*. [Online]
Available at: <https://store.arduino.cc/arduino-uno-rev3>
[Accessed 12 11 2017].

Anon.,2017. *Arduino-Introduction*. [Online]
Available at: <https://www.arduino.cc/en/guide/introduction>
[Accessed 20 10 2017].

Anon.,2017. *Arduino-Wikipedia*. [Online]
Available at: <https://en.wikipedia.org/wiki/Arduino>
[Accessed 10 10 2017].

Anon.,2017. *DHT11.pdf*. [Online]
Available at: <https://akizukidenshi.com/download/ds/aosong/DHT11.pdf>
[Accessed 12 11 2017].

Banzi, M., n.d. *How Arduino is open-sourcing imagination*. s.l.: TED Talk.

BeanDevice, n.d. *Datasheet-wireless-temp-humidity-BeanDevice-ONE-TH.pdf*. [Online]
Available at: http://www.beanair.com/wa_files/Datasheet-wireless-temp-humidity-BeanDevice-ONE-TH.pdf
[Accessed 17 11 2017].

Blaauw, F. et al., 2016. Let's get Physical-An intuitive and generic methodd to combine sensor technology with ecological momentary assessments. *SciencceDirecct*, Volume 63, pp. 141-149.

Circuitbasics, 2017. *How to set up the DHT11 humidity sensor*. [Online]
Available at: <http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/>
[Accessed 25 10 2017].

Kenny, D. T., 2005. Basic Sensor Technology. In: J. S. Wilson, ed. *Sensor Technology Handbok*. s.l.:s.n.

Kouhia, E.-P., 2016. *Development of an Arduino based Embedded system*, s.l.: s.n.

kushner, D., 2011. *The Making of Arduino-IEEE Spectrum*. [Online] Available at: <https://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino> [Accessed 10 11 2017].

McRoberts, M., 2010. Beginning Arduino. In: *Beginning Arduino*. s.l.:s.n., p. 172.

Sandhu, R., 2016. *What is Arduino? A full definition*. [Online] Available at: <https://www.lifewire.com/what-is-arduino-2495652> [Accessed 12 11 2017].

Therma-Stor, n.d. *Relative Humidity and Your Home-Therma-Stor, LLC*. [Online] Available at: <http://www.thermastor.com/information/relative-humidity-and-your-home.aspx> [Accessed 17 11 2017].

Vleeschouwer, K. D., Loey, Van, A. & hendrickx, M. E., 2017. The Effect of High Pressure-High Temperature Processing Conditions on Acrylamide Formation and Other maillard Reaction Compounds. *Journal of Agricultural and food chemistry*, Volume 2010, 58(22), pp. 11740-11748.

Wilson, J. S., 2005. *Sensor Technology Handbook*. Oxford: Elsevier Inc.

Appendices

Appendix 1: Data sheet of Arduino

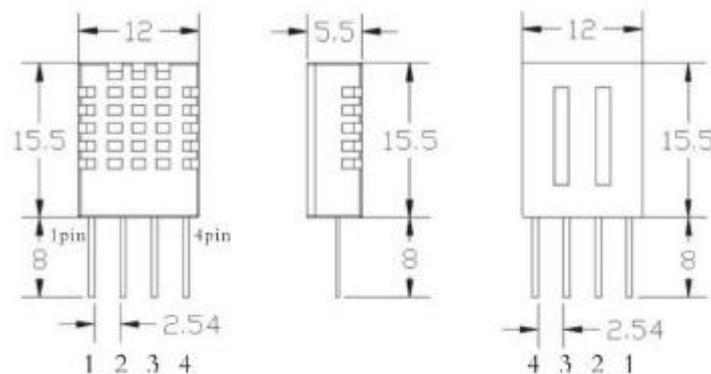
Features of Arduino Uno

- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 23 Programmable I/O Lines
 - 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF
- Operating Voltage:
 - 1.8 - 5.5V
- Temperature Range:
 - -40°C to 85°C
- Speed Grade:
 - 0 - 4MHz@1.8 - 5.5V, 0 - 10MHz@2.7 - 5.5.V, 0 - 20MHz @ 4.5 - 5.5V
- Power Consumption at 1MHz, 1.8V, 25°C
 - Active Mode: 0.2mA
 - Power-down Mode: 0.1µA
 - Power-save Mode: 0.75µA (Including 32kHz RTC)

Pin out: ATmega48A/PA/88A/PA/168A/PA/328/P

Appendix 2: Data sheet of DHT 11

Size and dimension of DHT11



DHT11 product parameters

Relative humidity

Resolution: 16Bit

Repeatability: $\pm 1\%$ RH

Accuracy: At 25°C $\pm 5\%$ RH

Interchangeability: fully interchangeable

Response time: 1 / e (63%) of 25°C 6s

1m / s air 6s

Hysteresis: $< \pm 0.3\%$ RH

Long-term stability: $< \pm 0.5\%$ RH / yr in

Temperature

Resolution: 16Bit

Repeatability: $\pm 0.2^\circ\text{C}$

Range: At 25°C $\pm 2^\circ\text{C}$

Response time: 1 / e (63%) 10S

Electrical Characteristics

Power supply: DC 3.5 ~ 5.5V

Supply Current: measurement 0.3mA standby 60 μ A

Sampling period: more than 2 seconds

Pin Description

1, the VDD power supply 3.5 ~ 5.5V DC

2 DATA serial data, a single bus

3, NC, empty pin

4, GND ground, the negative power

Data Timing Diagram:

Data format:

The 8bit humidity integer data + 8bit the Humidity decimal data +8 bit temperature integer data + 8bit fractional temperature data +8 bit parity bit.

○Parity bit data definition

“8bit humidity integer data + 8bit humidity decimal data +8 bit temperature integer data + 8bit temperature fractional data” 8bit checksum is equal to the results of the last eight.

Example 1: 40 data is received:

<u>0011 0101</u>	<u>0000 0000</u>	<u>0001 1000</u>	<u>0000 0000</u>	<u>0100 1101</u>
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate;

$0011\ 0101 + 0000\ 0000 + 0001\ 1000 + 0000\ 0000 = 0100\ 1101$

Received data is correct;

Humidity: $0011\ 0101 = 35H = 53\%RH$

Temperature: $0001\ 1000 = 18H = 24^{\circ}C$

Example 2: 40 data is received:

<u>0011 0101</u>	<u>0000 0000</u>	<u>0001 1000</u>	<u>0000 0000</u>	<u>0100 1001</u>
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate;

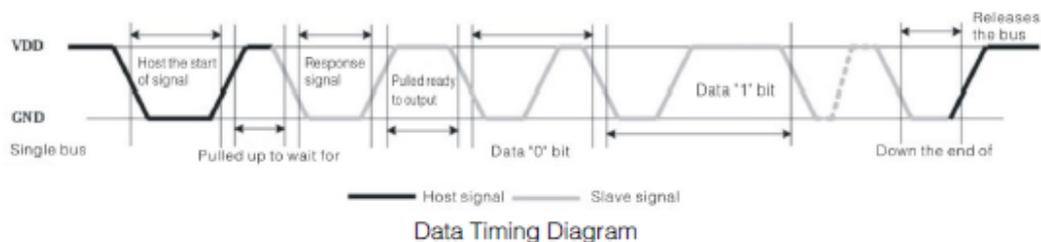
$0011\ 0101 + 0000\ 0000 + 0001\ 1000 + 0000\ 0000 = 0100\ 1101$

$01001101 \neq 0100\ 1001$

The received data is not correct, give up, to re-receive data.

○Data Timing Diagram

User host (MCU) to send a signal, DHT11 converted from low-power mode to high-speed mode, until the host began to signal the end of the DHT11 send a response signal to send 40bit data, and trigger a letter collection. The signal is sent as shown.



Serial point readings:

 COM3 (Arduino/Genuino Uno)

```
Current humidity = 20.00% temperature = 18.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 18.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 18.00C
Current humidity = 20.00% temperature = 18.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 18.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 20.00% temperature = 17.00C
Current humidity = 20.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
Current humidity = 19.00% temperature = 17.00C
```

 Autoscroll

