Developing Smart and Affordable Home Automation System using Arduino-based Framework

By
Ankit Raj
Student ID: 0661545
Email: araj@lakeheadu.ca

Supervised By
Dr. Sabah Mohammed
Email: mohammed@lakeheadu.ca

A Project submitted to
The Faculty of Graduates Studies
Lakehead University

In partial fulfilment of the requirement of the degree of
Master of Science in Computer Science

Department of Computer Science
Lakehead University

Acknowledgement

I would like to express my sincere gratitude to my Supervisor Dr. Sabah Mohammed for his continuous support towards my study and research. His guidance, patience, and motivation have played a vital role in the success of this journey. I could not have had a better supervisor.

I would also like to thank Dr. Jinan Fiaidhi, Dr. Vijay Mago, Dr. Maurice Benson and Dr. Ruizhong Wei who have been excellent professors that I had the privilege and pleasure to learn from while at Lakehead University. They made it a rewarding learning experience.

A special thanks to Radhika Goenka Dalmia, for not just being a wonderful friend but also a great support, I could count on. I am also thankful to Siddhant Shukla for helping me with all technical abnormalities that I faced in this project. I would also like to thank Shikha Malik, for being a friend who constantly gave me reality checks, and, Loveena Lucia Stephen who helped with her experience with Computer Science. They made a rather challenging journey much easier.

Thanks to my colleagues and friends at Lakehead International for helping me out in difficult times. Special thanks to Mr. Shashank Kala and Ms. Sapna Prabhugaonkar for encouraging me to pursue higher education and also for the understanding they have shown through a rough but fulfilling journey. Thanks to them, I have learned a lot more than just Computer Science in this endeavor.

Most of all, I am deeply grateful to my family for placing enormous trust and belief in me. They have always supported me with my decisions. Everything that I am, I owe it to them.

Contents

List of Tables

List of Figures

Abstract

The living standards of the modern society along with human behavior and thinking are changing dramatically with the advancement of technology, and the idea of a simple home is changing into a smart home. The popularity of home automation has been increasing vastly in recent years due to much higher affordability and simplicity. Being able to control aspects of our houses, and for having the feature to respond automatically to events, it is becoming more and more popular and necessary due to security and cost purposes.

This project proposes to implement an integrated home automation system using Arduino UNO as the master controller. It provides the concept to build a low cost-effective home control and monitoring system with the assistance of an integrated web server with internet protocol (IP) connectivity for access of equipment and devices remotely using ThingSpeak – a web-based app. This 3-Console system does not require a dedicated workstation as compared to other similar systems and offers a new communication protocol for monitoring and controlling the home environment with more than just switching functionality through a local server. The Arduino consoles control sensors and actuators that monitor a specified location and take action based on defined parameters like temperature, humidity etc. It also logs data captured by the sensor and projects it over the website, so as to provide access to users who are not on the same network of the connected devices. The data is secured from unauthentic access through the ThingSpeak app. The console can also send alerts if it detects an abnormality. The console sends a tweet whenever it detects the abnormal drift in temperature of the defined location. It also sends a tweet about the specific interaction with the actuators.

CH. 1: Introduction

CHAPTER 1: INTRODUCTION

1.1. OVERVIEW

The use of the word "Internet" in the term "Internet of Things" that stands for the vision of co-related communication can be seen either as a simple metaphor that means the way people use the Web today, in the same way, things will soon also communicate with each other, use services, provide data and thus generate added value – or it can be interpreted in a more strict technical sense, postulating that an IP protocol stack will be used by smart things (or at least by the ―proxies, their representatives on the network). Over the years, IOT have proven its significance and came out as a revolution across the globe that had started its reign since past few years and now gradually capitalizing the whole world.

The goal of this project is to develop a home automation system that uses concepts and principles of IOT to interact with sensors and actuators wirelessly to provide the smart home facility. The project aims on building a prototype for the aforementioned system as each component of the system has potential for further research in itself.

This chapter begins with a formal introduction of Internet of Things (IOT) and the history and evolution of IOT from its genesis to modern day. We then discuss main approaches used to tackle IOT tasks. Lastly, we describe in brief some of the main applications of IOT in the real world.

1.2. WHAT IS IOT

The Internet of Things (IoT) is a phenomenon that is rapidly gaining attention in the case of modern wireless telecommunications. The basic idea of this concept is the persistent existence around us of a variety of things or objects – such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbours to reach common goals **[Giusto et al. 2010].** The real strong point of IOT idea is its impact, that, it could have on several aspects of everyday-life and behaviour of potential users. From the perspective of a private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. In this context, aided living, e-health, enhanced learning are only a few examples of possible application scenarios in which the new paradigm will play a leading role in the near future. Similarly, from the perspective of an entrepreneur, the most apparent consequences will be equally visible in fields such as, automation and industrial manufacturing,

logistics, business/process management, intelligent transportation of people and goods. Due to its strong impact, IoT is included by the US National Intelligence Council in the list of six ‘‘Disruptive Civil Technologies” with potential impacts on US national power **[Santucci 2010]**. NIC foresees that ‘‘by 2025 Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more”. It highlights future opportunities that will arise, starting from the idea that ‘‘popular demand combined with technology advances could drive widespread diffusion of an Internet of Things (IoT) that could, like the present Internet, contribute invaluably to economic development”. The possible threats deriving from a widespread adoption of such a technology are also stressed. Indeed, it is emphasized that ‘‘to the extent that everyday objects become information security risks, the IoT could distribute those risks far more widely than the Internet has to date”.**[Santucci 2010]**

1.2.1. IOT Architecture

IOT architecture includes different categories of technologies supporting IOT. It works to show how various technologies relate to each other and communicates the scalability, modularity and configuration of IOT deployments in different scenarios. Below mentioned is the diagram that shows IOT architecture.[1]

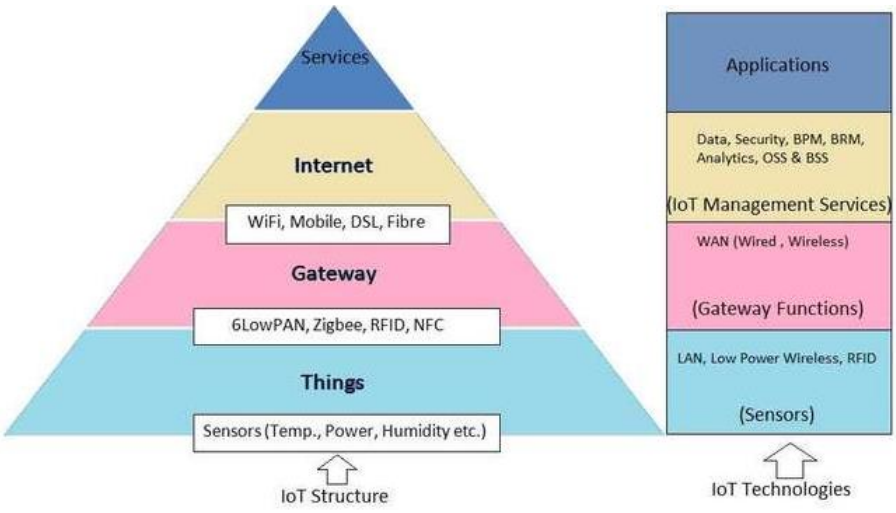


Figure 1.1: IoT Architecture

The lowest layer of architecture is defined by the smart objects integrated with sensors. The sensors facilitate the interconnection of the physical and digital worlds allowing real-time information to be collected and processed. The trimness of equipment has given effective sensors to be created in significantly smaller structures which are coordinated into objects in the physical world. There are different kinds of sensors for a variety of purposes. The

[1] http://www.rfwireless-world.com/images/IoT-structure-and-technologies.jpg

sensors have the ability to take measurements such as temperature, air quality, movement, and electricity. A sensor can measure the corporal property and change it into a signal that can be understood by a digital device. Sensors are categorized according to their unique purpose such as environmental sensors, body sensors, home appliance sensors and vehicle sensors, etc. Most sensors require connectivity to the sensor aggregators (gateways).

Post Sensor-layer come Gateway-Layer that includes all kinds of network definitions that helps the sensors with the connectivity. It can be of any form of LANs, such as, Ethernet or WiFi connections or PAN such as Bluetooth, ZigBee and Ultra-Wideband. Current network often tied with very different protocols, have been used to support various machine to machine networks and application.

After Gateway, comes Service management layer that is majorly defined by the internet. The service management renders the processing of data or information possible through security controls, analytics, and device management. This layer provides the features of business and process rule engines. IOT provides interaction of devices and systems together providing information in the form of events such as the temperature of goods, current location and traffic data. Some of these events need sorting or routing to post-processing systems such as taking periodic sensory data, while others require a response to the immediate situations such as reacting to emergencies on patient's health conditions. The rule engines result the establishment of decision logic and activate interactive and automated processes to facilitate a more reactive IOT system.

The top-most layer in the IOT architecture is the Application layer. There are various applications that can take advantage from IOT. Applications can be verticalized ones that are particular to a specific industry segment, and different applications, for example, Armada Administration, Resource Following, and Reconnaissance can cut over various industry parts.

1.3. HISTORY OF IOT

Ever since the birth of the internet in 1989, connecting "Things" in the internet began widely. Trojan Room coffee pot is possibly the first application of this kind **[Guptaet al. 2010].**In 1990 John Romkey created the first Internet 'device'[2], a toaster that could be turned on and off over the Internet. Wear-Cam[3] was invented in 1994 by Steve Mann. It had a near-real-time performance using a 64-processor system. Paul Saffo's[4] gave the first brief description about sensors and their future course of action in 1997. In 1999 The Internet of Things term was coined by Kevin Ashton, executive director of the AutoIDCentre, MIT. They also invented a

[2] https://romkey.com/
[3] http://wearcam.org/myview.html
[4] http://www.saffo.com/essays/sensors-the-next-wave-of-infotech-innovation/

global RFID-based item identification system in the same year[5]. As a major leap in commercializing IoT, in 2000 electronics giant LG announced its plans of revealing a smart refrigerator that would determine itself whether or not the food items stored in it are replenished. In 2003 RFID was deployed at a massive level in US army in their Savi program. The same year saw retail giant Walmart to deploy RFID in all its shops across the globe to a greater extent. In 2005 mainstream publications like The Guardian, Scientific American and Boston Globe cited many articles about IoT and its future course. "In 2008 a group of companies launched the IPSO Alliance to promote the use of Internet Protocol (IP) in networks of "smart objects" and to enable the Internet of Things. In 2008 the FCC approved the usage of the "white space spectrum". Finally, the launch of IPv6 in 2011 triggered massive growth and interests in this field" **[Suresh et al. 2013]**. After that, IT giants like Cisco, IBM and Ericson took a lot of educational and commercial initiatives with IoT. The IoT technology can be simply explained as a connection between humans – computers – things. All the equipment's we use in our day to day life can be controlled and monitored using the IoT. A majority of the process is done with the help of sensors in IoT. Sensors are deployed everywhere and these sensors convert raw physical data into digital signals and transmit them to its control centre **[Yuet al. 2013]**. By this way we can monitor environmental changes remotely from any part of the world via internet. This systems architecture would be based on context of operations and processes in real-time scenarios. In home automation every electrical switch box would be connected to a Smartphone (or sometimes a remote) so that it could be operated remotely. But such a scenario doesn't need a processor and a storage device installed in every switch box. It just needs a sensor to capture signals and process it (mostly switching ON/OFF). So this systems architecture varies depending on the context of its application.

1.4. IoTAPPLICATIONS IN DIFFERENT INDUSTRIAL SECTORS

Below mentioned are some of the IOT applications that are currently in use by various industry sectors.

1.4.1. Supply Chain

Dynamic ordering management tool is the best implementation of IOT in Supply chain management. Conventionally, the order-management in the warehouse uses multiple types of commodities to satisfy independent customer demands. The dynamic ordering tool can identify the types of commodities and decompose the order picking process to distributed sub-tasks based on area divisions. The application will plan the delivery routes centrally before activating order pickers for the delivery. Using executable algorithms in active tags, the tags can choose the best paths for the order pickers to take, as well as paths that are

[5] http://www.rfidjournal.com/articles/view?4986

within their responsible areas. This results in a more optimized order processing, time savings and lower cost of delivery.

1.4.2. Government

- **Crowd Control during Emergencies and Events:** The crowd control application will allow relevant government authorities to estimate the number of people gathering at event sites and determine if necessary actions need to be taken during an emergency. The application would be installed on mobile devices and users would need to agree to share their location data for the application to be effective. Using location-based technologies such as cellular, Wi-Fi and GPS, the application will generate virtual ―heat maps of crowds which can be combined with sensor information obtained from street cameras, motion sensors and officers on patrol to evaluate the impact of the crowded areas and help to inform the emergency vehicles about the best possible routes to take.
- **Intelligent Lamp-posts:** The intelligent lamp-posts or Smart-Lamps is a network of streetlamps that are tied together in a WAN which can be controlled and monitored from a central point, by the city or a third party. It captures data such as ambient temperature, visibility, rain, GPS location and traffic density which can be fed into applications to manage road maintenance operations, traffic management, and vicinity mapping.

1.4.3. Retail

Shopping Assistants applications can be used to locate appropriate items for shoppers and provide recommendations of products based on consumer preferences, in the retail sector. The application can be used through the shopper's personal mobile devices such as tablets and phones, and provide shopping recommendations based on the profile and current mood of the shopper. With the help of context-aware computing services, the application confines data feeds such as locations of products and types of stores, either from the shopping-complex or mall, websites or open API if the mall allows it. Additionally, the application tries to match the user's shopping requirements or prompts the user for any specific choice or preferences, e.g., ―What would you like to buy today? In order to help user to find a particular product in the mall, the application guides the user from the current location to the destination, using local-based technology such as Wi-Fi 33 embedded on the user's mobile phone.

1.4.4. Healthcare

- **Elderly Family Member Monitoring:** This application creates the freedom for the elderly to move around outdoors with family members being able to monitor their presence. The elderly sometimes lose their way or are unable to identify familiar surroundings to recall their way back home. The application can be a tiny piece of a wearable device such as a coil-on-chip tag attached to the elderly. This tag will be equipped with location-based sensors to report

the paths that the wearer has traveled. It can emit signals to inform family members if the wearer ventures away from predetermined paths. It can also detect deviations in their daily routines. Family members can also track the location of their elderly online via the user interface (UI) application.

- **Continuous Patient Monitoring:** It is an expansion of the Elderly Family Member Monitoring application. This application, however, needs the support from the medical services companies. This application requires the use of medical body sensors to monitor vital body conditions such as heartbeat, temperature and sugar levels. The application examines the current state of the patient's health for any abnormalities and can predict if the patient is going to encounter any health problems. Analytics such as predictive analytics and CEP can be used to extrapolate information to compare against existing patterns and statistics to make a judgment.

1.4.5. Transportation

- **Special Needs and Elderly Transportation Assistant:** The transportation assistant application serves to address the group of commuters with special needs and who require assistance as they commute using public transportation e.g., using the public train service, the transport assistant will inform the nearest transport staff so they can provide special assistance such as audio and visual services and physical assistance for the passengers.
- **Distributed Urban Traffic Control systems:** Distributed Urban Traffic Control systems enable the tracking of car locations in real time and provide an appropriate traffic management response to handle road conditions. It can be used in times of emergency such as setting up of fast lane corridors for emergency services, i.e., ambulances, police cars, and fire brigades, to pass through during heavy traffic conditions.

1.4.6. Energy Management

- **Facilities Energy Management Facilities:** Energy management involves the use of a combination of advanced metering and IT and operational technology (OT) that is capable of tracking, reporting and alerting operational staff in real time or near real time. They provide dashboard views of energy consumption levels, with varying degrees of granulation, and allow data feeds from a wide range of building equipment and subsystems.
- **Home Energy Management/Consumer Energy Management:** Home energy management (HEM) optimizes residential energy consumption and production. Solutions include software tools that analyze energy usage and home-area network (HAN) energy management sensors that respond to variable power prices. A combination of these solutions contributes towards reducing overall carbon emissions for homes.
- **Cloud Computing:** IOT connects billions of devices and sensors to create new and innovative applications. In order to support these applications, a platform is required, which is reliable, elastic and agile. Cloud computing is one of the prominent platforms to support IOT. Cloud computing is an architecture that arranges various technology capabilities such as multi-

tenancy, automated provisioning and usage accounting while relying on the Internet and other connectivity technologies like richer Web browsers to realize the vision of computing delivered as a utility. Cloud computing is seen as a growing adoption with three commonly deployed cloud service models namely Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). For example, in IaaS, the use of hardware such as sensors and actuators can be made available to consumers as cloud resources. Platform as a Service (PaaS) can facilitate users with a platform to access IOT data and tool to customize or develop IOT applications (or host-acquired IOT applications). Software as a Service (SaaS) can be offered on top of the PaaS solutions to offer the provider's own SaaS platform for specific IOT domains.

In the next chapter we will discuss the main concepts used in this project – namely working with sensors, interacting with various IOT boards, challenges involved and results achieved so far concluding with the research problem definition.

CH. 2: Literature Review

CHAPTER 2: LITERATURE REVIEW

As innovation is developing, things that encompass us in our day by day lives are beginning to be able to share information over the Web. With this development, it is never again the case that no one but people can work the gadgets associated with the Web. These gadgets are presently ready to gather and offer sensorial information that can be controlled by sensor inputs. They likewise enable you to control huge information examination, screen frameworks, and even influence gadgets to cooperate for a typical reason. A new era has begun, the era of Internet of Things! **[De Sousa 2015]**

Following this vision, Arduino presented the Arduino Uno board; the most used and documented board of the whole Arduino family. An Arduino Uno Board can be programmed to read and control sensors and actuators, being an interesting tool for sensorial data collection. ***Arduino Uno***[6] is a microcontroller board based on the ATmega328P (datasheet). It has 14 computerized input/output pins (of which 6 can be utilized as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset catch. It contains everything expected to help the microcontroller; just interface it to a PC with a USB link or power it with an AC-to-DC adapter or battery to begin.

This chapter focuses on explaining various types of sensors that could be used to establish a well-organized system of IoT using Arduino UNO, and also describes few experiments that demonstrate the working of these sensors.

2.1. WORKING WITH SENSORS

The Internet of Things (IoTs) can be depicted as interfacing ordinary items like PDAs, Internet TVs, sensors and actuators to the World Wide Web where the gadgets are shrewdly connected together empowering new types of correspondence amongst things and individuals, and between things themselves. Building IoTs has advanced altogether over the years and has added another measurement to the universe of data and correspondence advances. As indicated by **[Swan 2012]**, in 2008, the quantity of associated gadgets outperformed associated individuals and it has been evaluated by Cisco that by 2020 there will be 50 billion associated gadgets which is seven times the total populace. Presently anybody, from whenever and anyplace can have availability for anything and it is normal that these associations will broaden and make a completely propelled dynamic system of IoTs. The advancement of the IoTs will affect various divisions, from remote sensors to nanotechnology.

[6] https://store.arduino.cc/usa/arduino-uno-rev3

A sensor is a device that identifies or measures a physical property and records, demonstrates, or generally reacts to it. It is a device that converts signals from one energy domain to electrical domain. We live in a world of Sensors. We can discover diverse sorts of sensors in our homes, workplaces, cars and so on attempting to make our lives less demanding by turning on the lights by detecting our presence, altering the room temperature, recognize smoke or fire, make us delectable espresso, open garage doors when our car is close to the door and numerous different work. The following diagram shows various available sensors[7]:

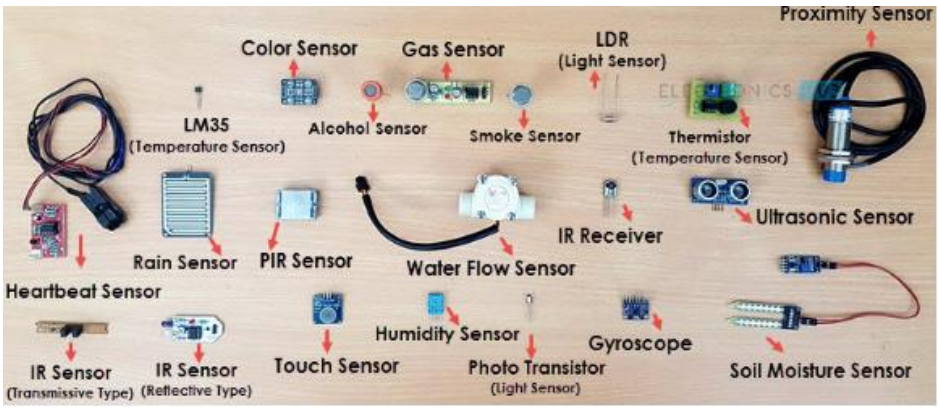


Figure 2.1 Different types of Sensors available in market today

Sensors play a vital role in enabling smart objects to participate in IoT (Internet of Things). It makes it easier to collect accurate data that one could only imagine to be accurate if collected otherwise. Sensors can be used to play with various types of inputs or outputs. There are several classifications of sensors made by various authors and experts. Some are extremely basic and some are very complex. The accompanying grouping of sensors may as of now be utilized by a specialist in the subject, however, this is an extremely basic characterization of sensors.

In the primary characterization of the sensors, they are segregated into Active and Passive. Active Sensors are those which require an outer excitation signal or a power signal. Inactive/Passive Sensors, on the other hand, don't require any external power signal and directly generates output.

The other kind of categorization depends on the methods for detection used in the sensor. Some of the means of detection are Electric, Biological, Chemical and Radioactive etc.

Another classification depends on conversion i.e. the input and the output. Few of these general phenomena are Photoelectric, Thermoelectric, Electrochemical, Electromagnetic, Thermo-optic, etc.

[7] https://www.electronicshub.org/different-types-sensors/

The last categorization of the sensors is Analog and Digital Sensors. Analog Sensors produce an analog output i.e. continuous output signal with respect to the quantity being measured. Digital Sensors, unlike Analog Sensors, work with discrete or advanced information. The information in digital sensors, which is utilized for transformation and transmission, is discrete in nature.

2.1.1. Working with Analog Inputs

The analog inputs are used to convert voltage level into a digital value that can be stored and processed in the computer. In today's world of mass-digitization, analog inputs hold a very significant role. One can only process the real world value by a computer or any machine if they manage to control the analog inputs. Arduino UNO comes with 6 Analog input pins as mentioned earlier in this chapter. One can use any of the sensors that can manage analog input to give a desired list of actions.

The following experiment explains the concept of handling analog inputs using an Arduino Uno board[8]:

- **Overview:**

Experiment_Figure 2.1 Potentiometer

In this experiment, a variable resistor (a potentiometer)[9], we read its value using one analog input of an Arduino board and we change the blink rate of the built-in LED accordingly. The potentiometer is the electrical type of transducer or sensor and it is of resistive type because it works on the principle of change of resistance of the wire with its length. The resistor's analog value is read as a voltage because this is how the analog inputs work.

[8] https://www.dropbox.com/s/sf5d6gqyu3tlljf/IMG_3058.MOV?dl=0

[9] https://www.brighthubengineering.com/hvac/47389-what-is-potentiometer-potentiometer-used-as-the-transducer-or-sensor/

- **Specification:**

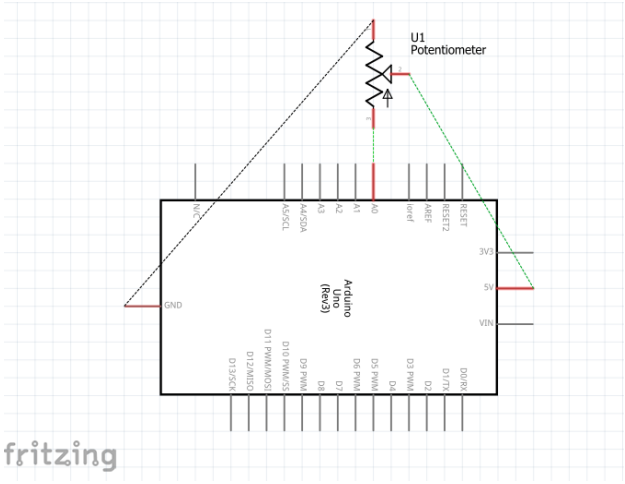
Product Name: Potentiometer;
Resistance Value: 10K ohm;
Adjustment Type: Top Adjustment

- **Hardware Required:**

Material Diagram	Material Name	Quantity
	10KΩ Potentiometer	1
	USB cable	1
	Arduino UNO R3	1
	Breadboard	1
	Jumper Wires	Several

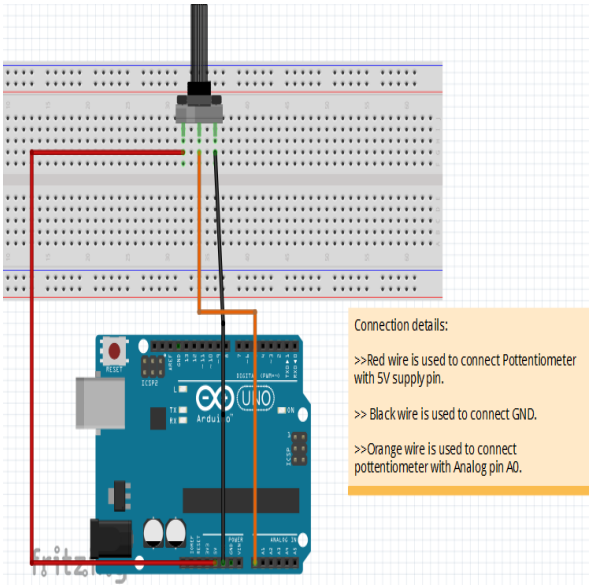
Table_Experiment 2.1 Material Required for Experiment 1

- **Connection:**
  - **a. Schematic:**



Experiment_Figure 2.2 Schematic diagram of connection

b. Connection Diagram:



Experiment_Figure 2.3 Fritzing connection Diagram

Note: The middle pin of the potentiometer is connected to the analog port 0(A0).

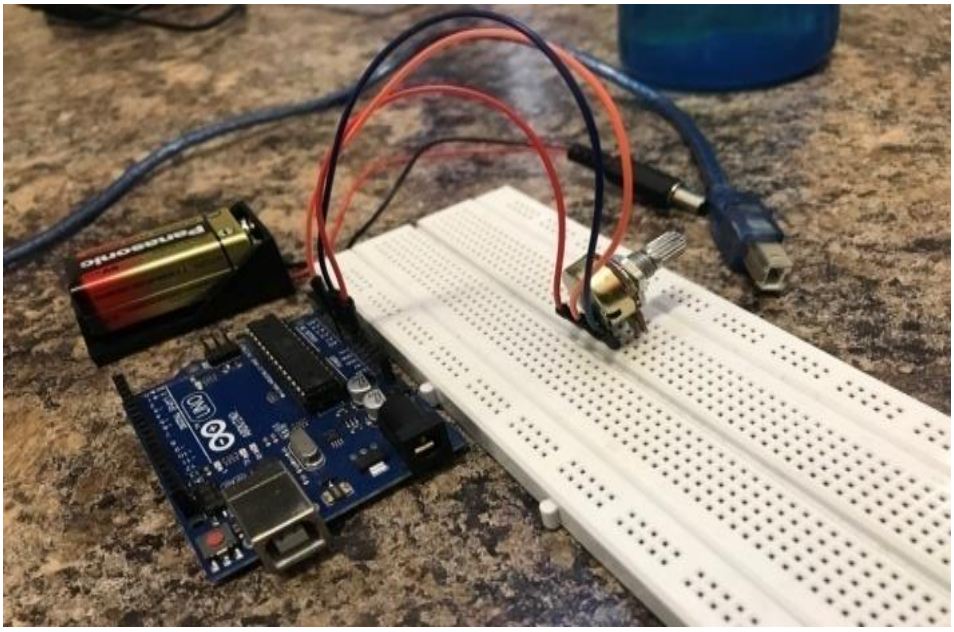
- **Sample Code:**

Ch_2_Ex_1.inc

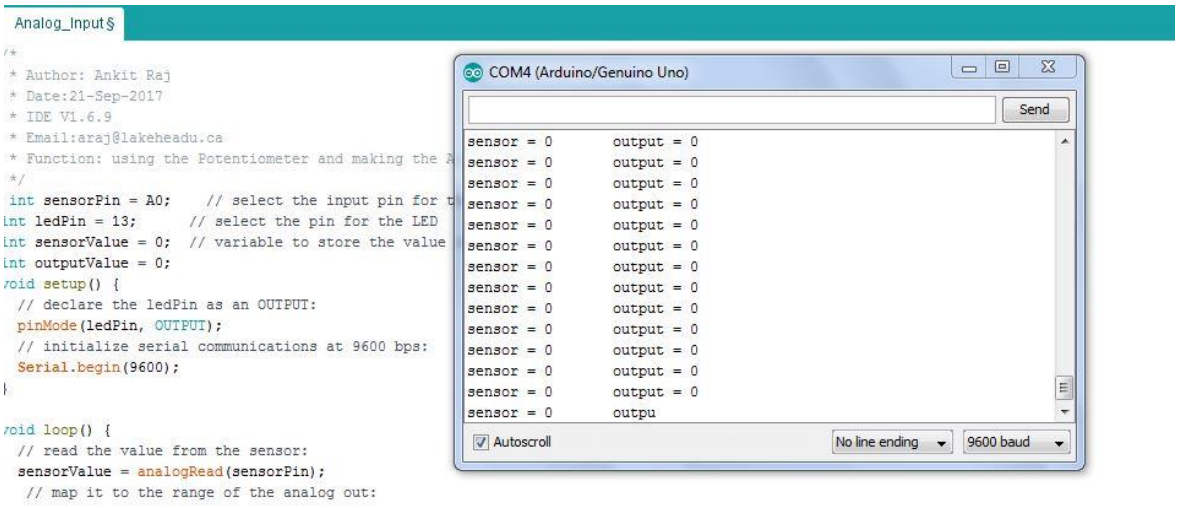
```
/*
 * Author: Ankit Raj
 * Date:21-Sep-2017
 * IDE V1.6.9
 * Email:araj@lakeheadu.ca
 * Function: using the Potentiometer and making the Arduino board to read the analog inputs
 */
 int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor
int outputValue = 0;
void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
   // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  analogWrite(ledPin, outputValue);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
```

Experiment_Figure 2.4 Sample code

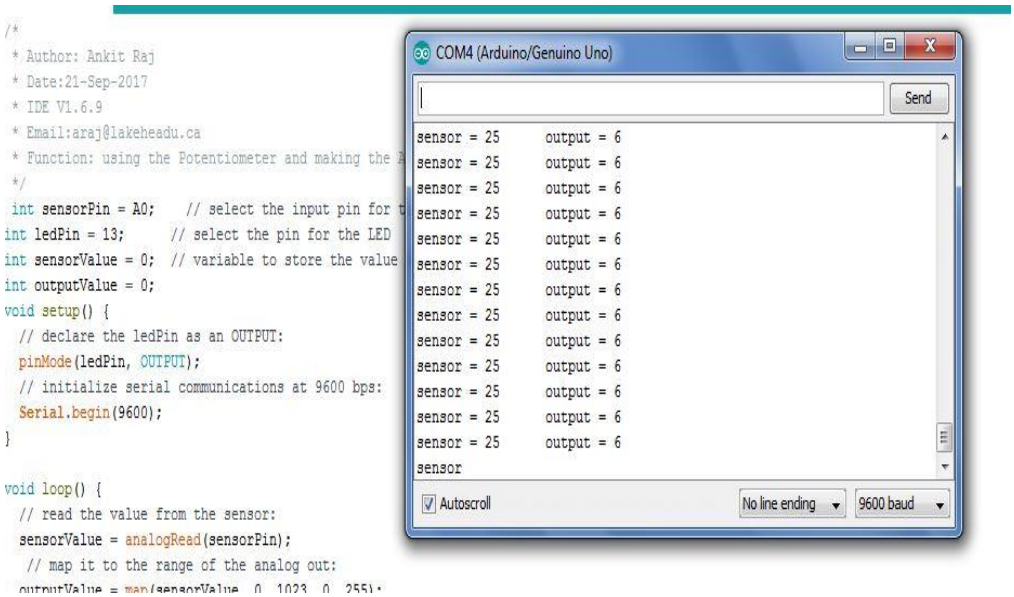
- **Set-Up Picture:**



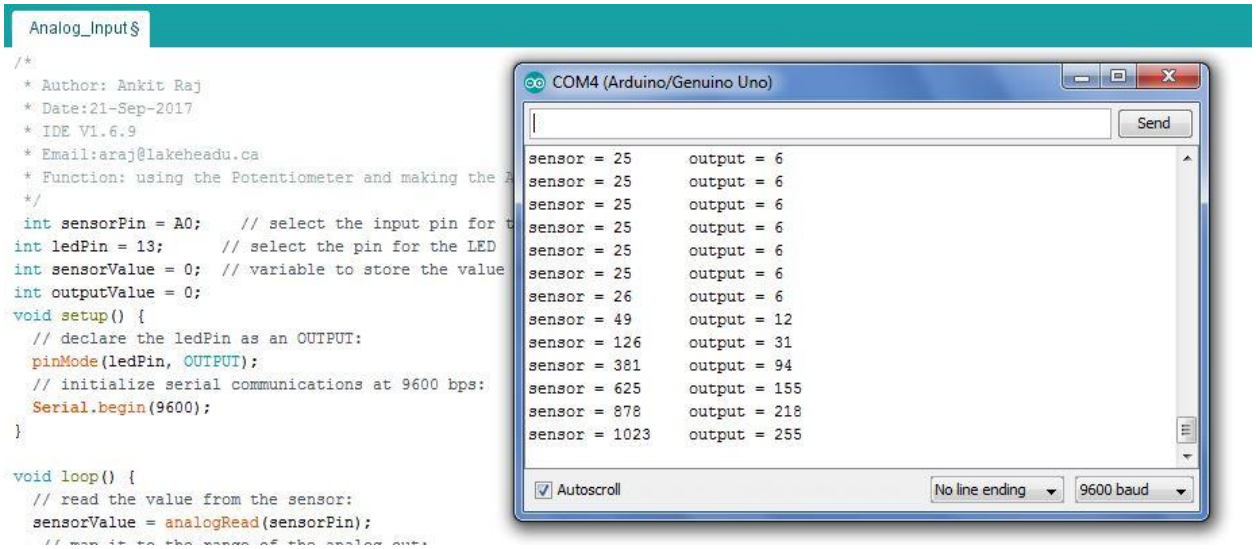
Experiment_Figure 2.5 The setup



Experiment_Figure 2.6 Output for Analog inputs on Serial monitor when Potentiometer is set to 0.



Experiment_Figure 2.7 Value of output changed as the potentiometer is rotated.



Experiment_Figure 2.8 Value changing as per the potentiometer.

- **Language Reference:**

The code keywords used in the program were : pinMode(), OUTPUT, INPUT, HIGH, LOW, delay(), map(), Serial.begin(), Serial.print(), analogRead(), digitalWrite(), analogWrite()

- **Application Effect:**

The analog input of Potentiometer is read and the blink rate of the built-in LED is changed accordingly.

2.1.2. Handling Multiple Outputs

The reason why IoT is considered the most revolutionary innovation in the world is because it gives or provides us with various systems that can deal with real time values and manage various devices from one device. IoT world forms a connection between all physical objects, for example, individuals, cars, tables, seats, etc., in such a way that one can characterize every one of them in the virtual world and empower connectivity between every one of them anyplace and whenever. IoT can be connected in numerous applications, for example, aviation, supply-chain, media communications **[Sundmaeker et al. 2010; Uckelmann et al. 2011]**. There are a few advances that empower that association, for example, Radio Frequency ID (RFlD) and Wireless Sensor Network (WSN). Blending both of these advances can revolutionize IoT in real world.

However, these advancements could be unequipped for giving a security to their applications. Because of the low calculation capacities of the RFID and WSN there is a need to assemble a lightweight security cipher that can fit these gadgets. Outlining a cipher with lightweight properties is always challenging because the designer needs to adapt to the trade off between accomplishing vigorous security with ease and upgraded execution **[Eisenbarth et al. 2007]**. Accomplishing a cipher plan that incorporates all these three goals is so difficult. Cipher algorithm can be either symmetric or asymmetric. Symmetric cipher depends on a mix of arithmetic and cryptographic rules that for the most part utilize simple primitives, for example, permutation, substitution, rotation, bit-wise XOR, shift etc. On the other hand, asymmetric ciphers are based on hard to solve mathematical problems and usually depend on hard primitives such as modular addition, subtraction, modular multiplication, variable length rotations, and so on. Therefore, symmetric cipher is easier to implement in low computation devices **[Charlwood et al. 1998]**. Symmetric cipher is separated into two kinds: one is block cipher and the other is a stream cipher. The stream cipher is more suited to low calculation gadgets since it utilizes little memory size and runs speedier than the block cipher. The greater part of the accessible stream cipher endeavoured to imitate the possibility of the One Time Pad (OTP) **[Stallings 2002]**. The OTP idea relies upon having a random key that is the same size as the plaintext then XORs both of them to generate the cipher text. The quality of the OTP originates from the randomness of the key. Outlining such a figure that can deliver a genuine irregular key is considered as a troublesome issue. The accessible stream ciphers are either about producing a True Random Generator (TRG) or Pseudo Number Generators (PNG). Implementing the TRG is not that practical a solution since it depends on the physical phenomena such as thermal noise. Be that as it may, the PNG relies upon scientific arrangements. Therefore it is a more practical solution. Stream cipher can be either software oriented (SWO) or hardware oriented (HWO) security solution. The vast majority of the hardware-oriented solutions depend on utilizing Shift Registers (SR). SR solutions are easy to execute in hardware and can be utilized to make PNG that can produce a protected cryptographic key.

The following experiment shows managing the output through shift-registers using Arduino Uno[10]:

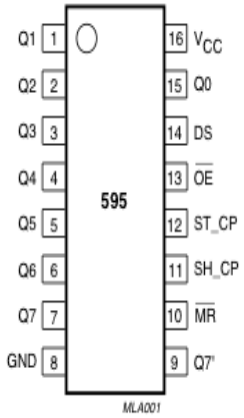
- **Overview:**

Experiment_Figure 2.9 Shift Register

In this experiment, Shift register is used to control eight LED with an Arduino without needing to give up 8 output pins.

- **Specification:**

Product Name: SN74HC595 (Shift register)

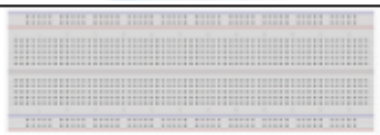


PINS 1-7, 15	Q0 " Q7	Output Pins
PIN 8	GND	Ground, Vss
PIN 9	Q7"	Serial Out
PIN 10	MR	Master Reclear, active low
PIN 11	SH_CP	Shift register clock pin
PIN 12	ST_CP	Storage register clock pin (latch pin)
PIN 13	OE	Output enable, active low
PIN 14	DS	Serial data input
PIN 16	Vcc	Positive supply voltage

Experiment_Figure 2.10 Specification of Shift-Register

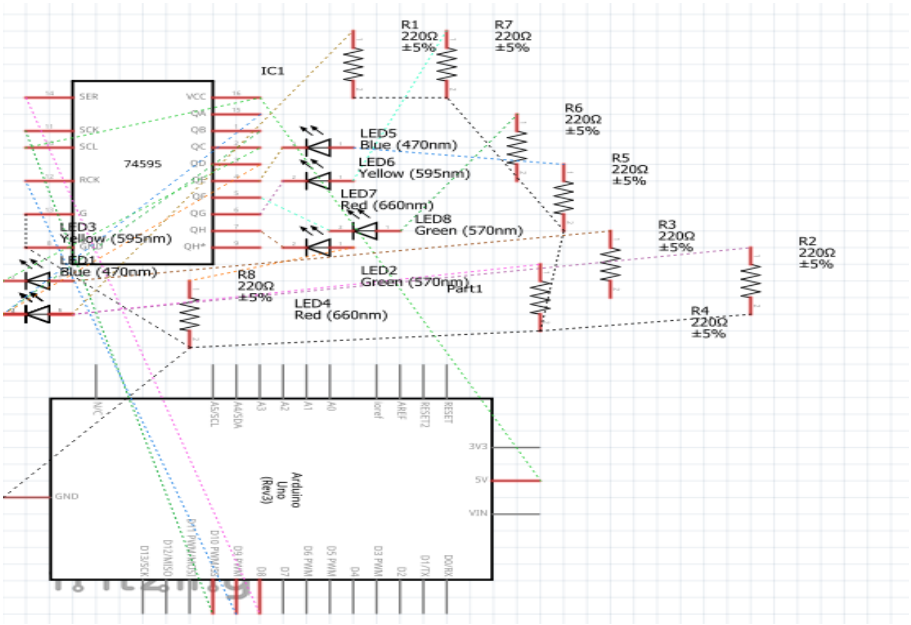
[10] https://www.dropbox.com/s/5g3jy496ql1adn4/Chapter2_Experiment2.MOV?dl=0

- **Hardware Required:**

Material Diagram	Material Name	Quantity
	Shift register SN74HC595	1
	USB cable	1
	Arduino UNO R3	1
	Breadboard	1
	Jumper Wires	Several
	LED	8
	220Ω RESISTORS	8

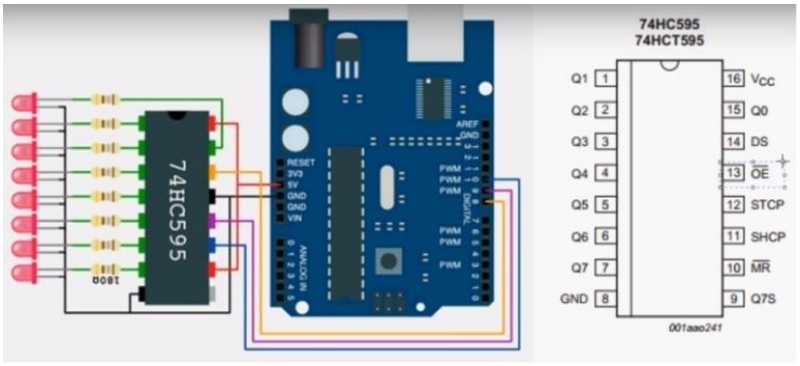
Table_Experiment 2.2 Material Required for Experiment 2

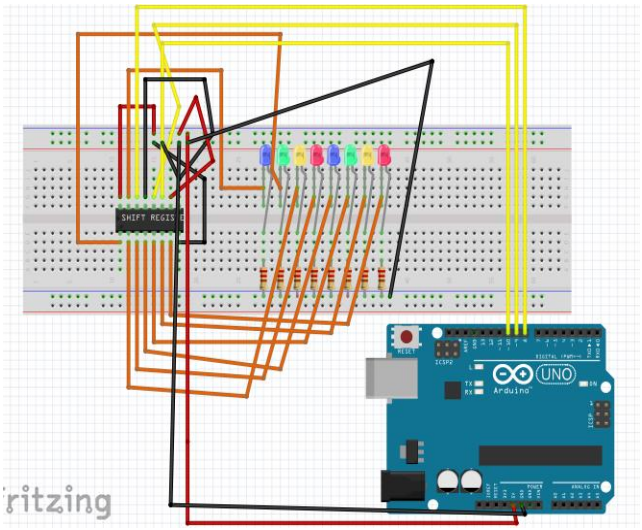
- **Connection:**
  - **a. Schematic**



Experiment_Figure 2.11 Schematic connection for SR

b. Connection Diagram:





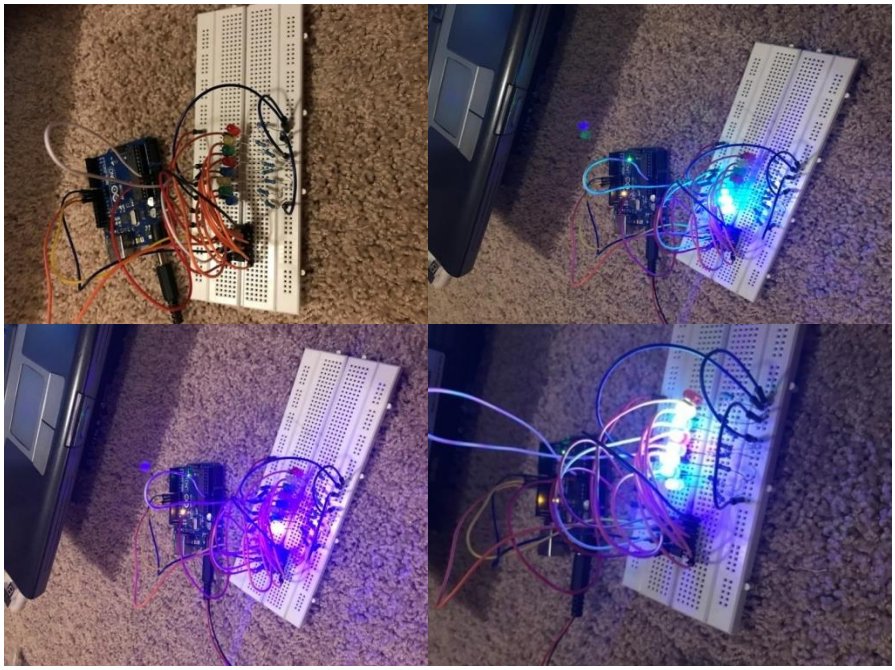
Experiment_Figure 2.12 Connection diagram SR - Arduino (Fritzing)

- **Sample Code:**

```
/*
 * Author: Ankit Raj
 * Date:15-October-2017
 * IDE Arduino V1.6.9
 * Email:araj@lakeheadu.ca
 * Function: Controlling input/output of LEDs using Shift register.
 */
 // This is the working code for SN74HC595
#define DS_pin 8
#define STCP_pin 9
#define SHCP_pin 10

boolean registers[8];
int time1 = 300;

void writeReg() {
  digitalWrite(STCP_pin, LOW);

  for (int i=7; i>=0; i--)
  {
    digitalWrite(SHCP_pin, LOW);
    digitalWrite(DS_pin, registers[i]);
    digitalWrite(SHCP_pin, HIGH);
  }
  digitalWrite(STCP_pin, HIGH);
}

void setup() {
  // put your setup code here, to run once:

  pinMode(DS_pin, OUTPUT);
  pinMode(STCP_pin, OUTPUT);
  pinMode(SHCP_pin, OUTPUT);

  writeReg();
}

void loop() {

  for (int i=0; i<=7; i++)
  {
    registers[i]=HIGH;
    delay(time1);
    writeReg();
  }

  for (int i=7; i>=0; i--)
  {
    registers[i]=LOW;
    delay(time1);
    writeReg();
  }

  delay(1000);
}
```

- Experiment_Figure 2.13 Sample code

- **Set-Up Picture:**



Experiment_Figure 2.14 The setup and the processing of SR

- **Language Reference:**

Keywords used in the program were: pinMode(), OUTPUT, HIGH, LOW, delay(), writeReg();

- **Application Effect:**

3 LED ports can be used to control the eight IO. One can observe all the LEDs turn on or turn off regularly.

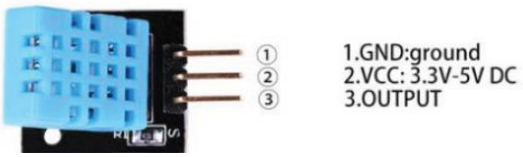
2.1.3. Handling Environmental Inputs

With increasing demands of smart homes and smart cities, handling environmental data on real time is really necessary. We have often heard that sometimes the appliances that are responsible for maintaining a comfortable temperature (like air-conditioners) often behave abnormal and make things difficult for the user to handle last minute.

Sensors which we are going to discuss later here can be used to alert the user of these incidents. Also, if we take a larger area of implication, say a city, weather alerts are the most frequently used system by people. With IoT, one can imagine a system that could maintain a stable temperature for a large area by constantly interacting with a device to maintain a uniform temperature. Not to forget, that the mentioned system involves a lot of cost and high maintenance. But even a small system where the sensor is used to alert the user about changing climatic condition is also a part of IoT. Additionally, temperature and humidity sensors can also be used by civil engineers to help them with construction metrics **[Barroca et al. 2013]**. Since, our whole chapter is based on how Arduino is making this possible, it is important for us to know how efficiently the UNO board can interact with these kinds of sensors.

The following experiment shows how Arduino UNO board can be used to read the values of DHT11 (Temperature and Humidity Sensor):

- **Overview:**

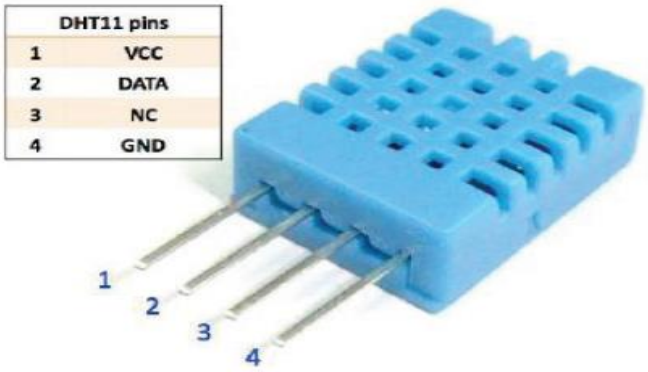


Experiment_Figure 2.15 DHT11

In this experiment, DHT11 – Temperature and Humidity sensor is used to read temperature and humidity of existing ambience and plot it on the monitor.

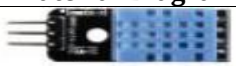
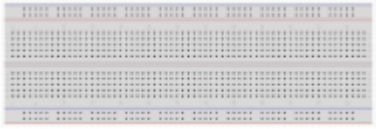
- **Specification:**

Product Name: DHT11



Experiment_Figure 2.16 Specification - DHT11 pins

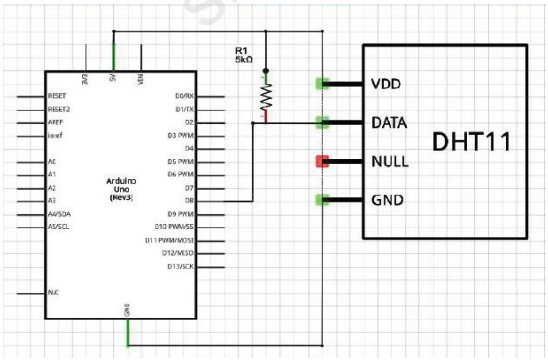
- **Hardware Required:**

Material Diagram	Material Name	Quantity
	**Temperature-Humidity Sensor (DHT11)**	**1**
	**USB cable**	**1**
	**Arduino UNO R3**	**1**
	**Breadboard**	**1**
	**Jumper Wires**	**3**

Table_Experiment 2.3 Handling Temperature and Humidity Sensor

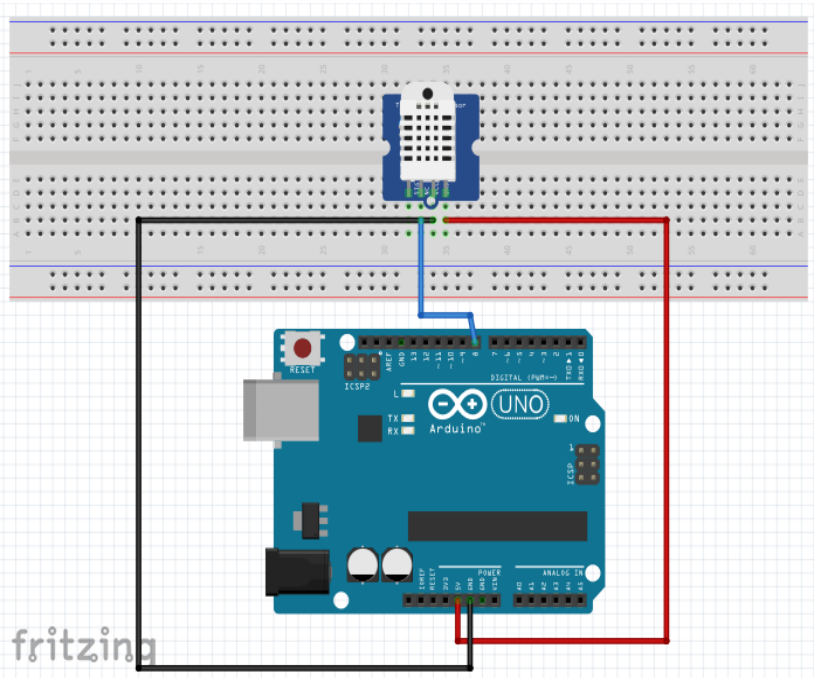
- **Connection:**

a. Schematic

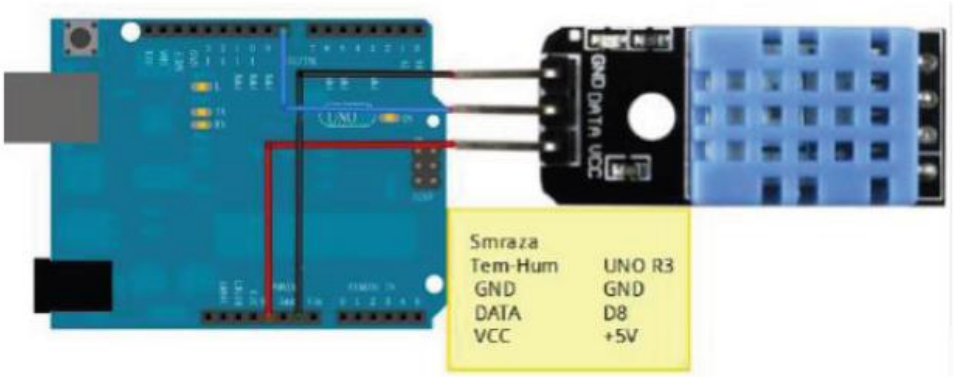


Experiment_Figure 2.17 Schematic Connection – DHT11

b. Connection Diagram:



Experiment_Figure 2.18: Connection Diagram DHT11-UNO



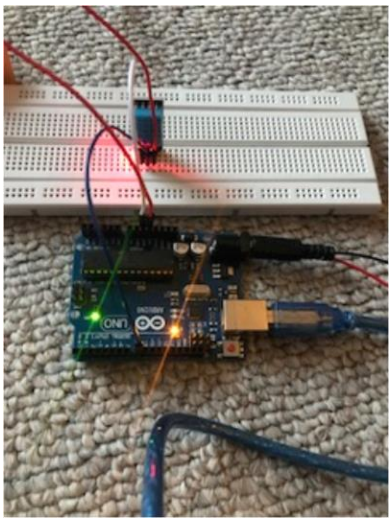
Experiment_Figure19 DHT11_Connection Explanation

- **Sample Code:**

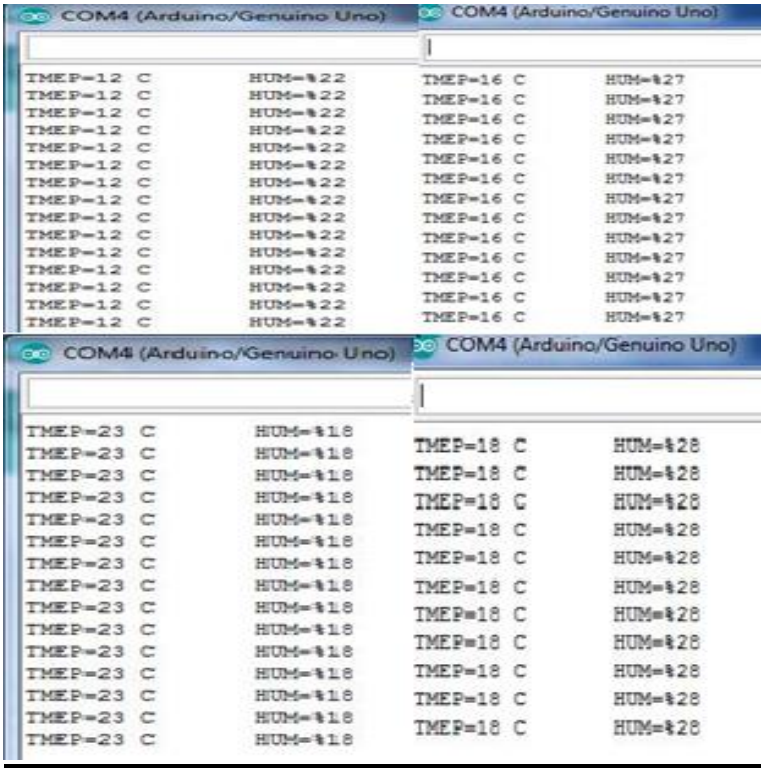
```
/*
 * Author: Ankit Raj
 * Date:2017/11/29
 * IDE V1.6.9
 * Email:araj@lakeheadu.ca
 * Function:working of sensor to capture temperature and humidity
 * Tips:Open the serial port
 *      Add Dht11 to your libary
 */
#include <dht11.h>
dht11 DHT11;
#define DHT11PIN 8
void setup(void)
{
  DHT11.read(DHT11PIN);
  Serial.begin(9600);
}
void loop(void)
{
  Serial.print("TMEP=");
  Serial.print(DHT11.temperature);
  Serial.print(" C");
  Serial.print('\t');
  Serial.print("HUM=%");
  Serial.println(DHT11.humidity);
  delay(500);
}
```

Experiment_Figure 2.20 Sample Code: DHT11

- **Set-Up Picture:**



Experiment_Figure 2.21 Setup - DHT11



Experiment_Figure 2.22 The reading on plotter changes with varying temperature of the room

Note: The above mentioned experiment was carried out non-stop for 4hrs and the recordings were noted by changing the temperature of pre-installed thermostat of the room.

- **Language Reference**

A separate library was used (**dht11.zip**).
The keywords used in the program were: Serial.begin(), DHT11.read (), Serial.print(), delay()

- **Application Effect**

Open the serial port monitor; one will see some different value return by DHT11.

2.1.4. Controlling the movements

One of the enhanced features of IoT is its contribution to robotics. IoT provides various concepts that help in developing computer-assisted instrumentation to design various laboratories **[Kuan et al. 2016].** These concepts can also be used to design remote driven wheel-chairs for disabled. However, one can only achieve that if they can store or transfer the information of co-ordinates in/from the computer.

The following experiment shows a basic example of how Arduino UNO boards can be used to read the analog inputs of a joystick:

- **Overview:**

Experiment_Figure 2.23 Joystick

In this experiment, Joystick module is used and its analog value is displayed over monitor

- **Specification:**

Pin Definition:

UNO R3	Joystick module
GND	GND
5V	+5V
A0	VRx
A1	VRy
D13	SW

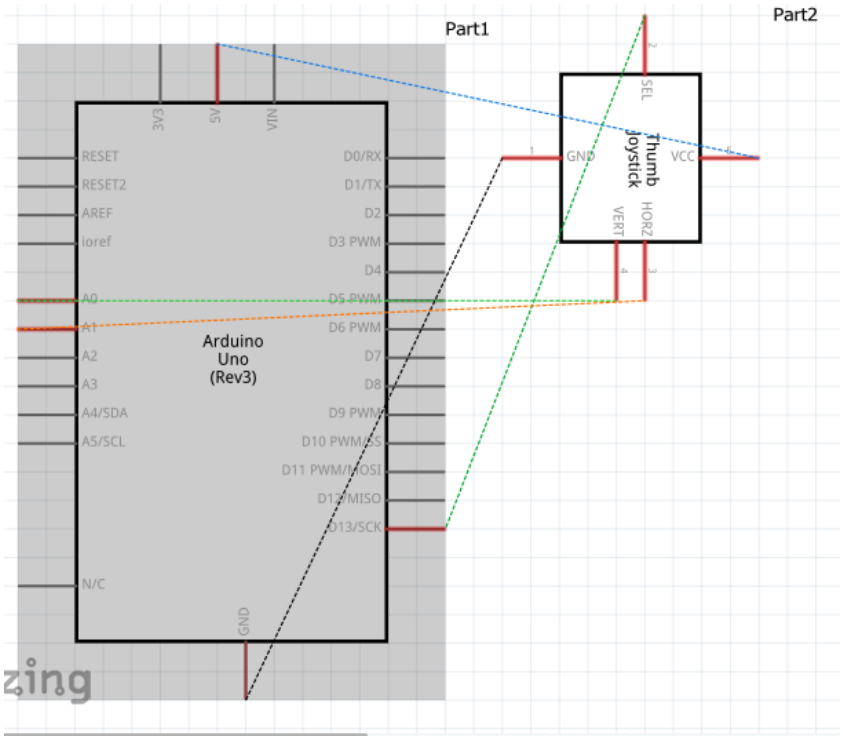
Experiment_Figure 2.24 Pin definition– Joystick

- **Hardware Required:**

Material Diagram	Material Name	Quantity
	**Joystick module**	**1**
	**USB cable**	**1**
	**Arduino UNO R3**	**1**
	**Male to female Jumper Wires**	**5**

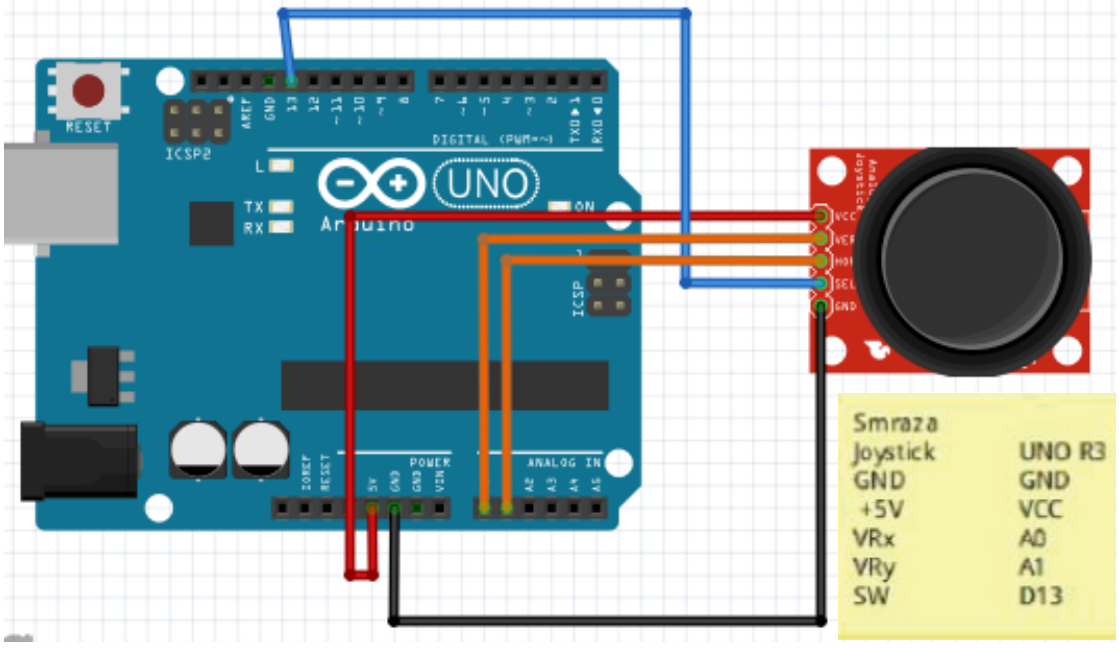
Table_Experiment 2.4 Material Required Joystick Experiment

- **Connection**
  - **a. Schematic**



Experiment_Figure 2.25 Schematic Diagram - Joystick - UNO

  - **b. Connection Diagram:**



Experiment_Figure 2.26 Connection Diagram Joystick-UNO

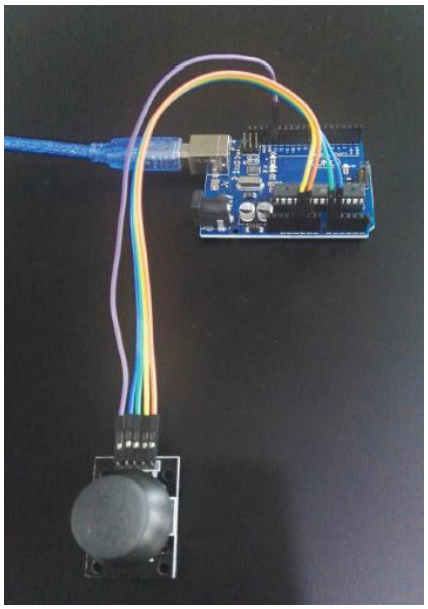
- **Sample Code:**

2_joystick_exp4.i

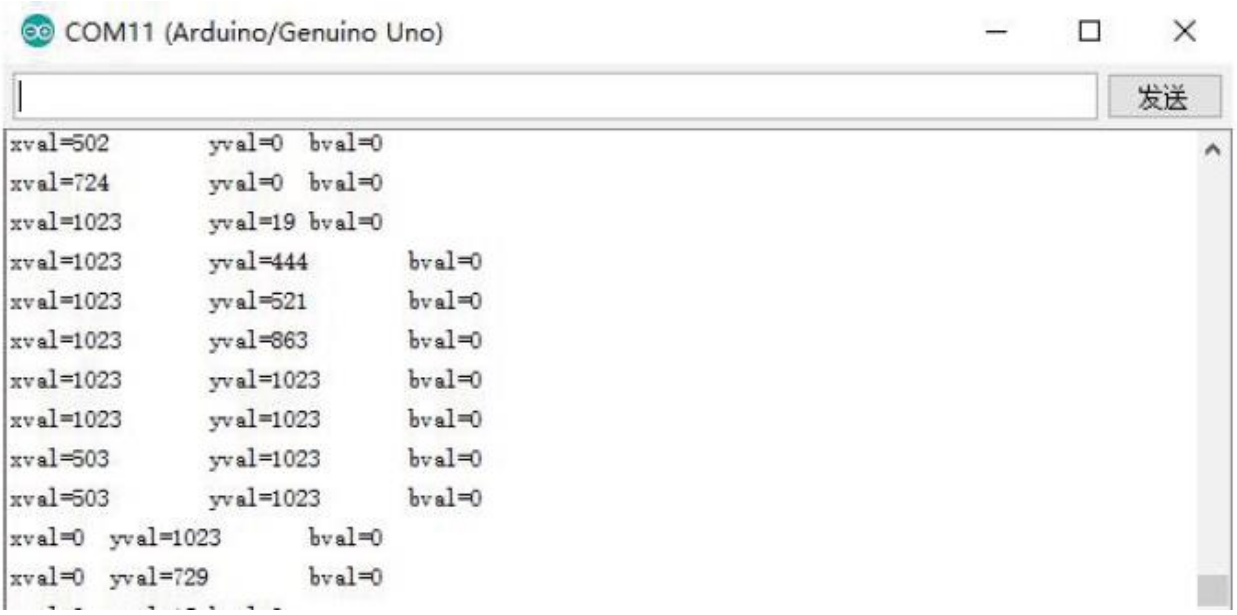
```
 * Email:araj@lakeheadu.ca
 * Function: to use Joystick module
 */
int xpotPin = A0;  //Set port
int ypotPin = A1;
int bpotPin = 13;
int xval=0;    //Initialize a variable
int yval=0;
int bval=0;
void setup()
{
  pinMode(xpotPin,INPUT);
  pinMode(ypotPin,INPUT);
  pinMode(bpotPin,INPUT);
  Serial.begin(9600);
}
void loop()
{
  xval = analogRead(xpotPin);    //Read Values from the xpotPin
  yval = analogRead(ypotPin);
  bval = digitalRead(bpotPin);
  Serial.print("xval=");
  Serial.print(xval);
  Serial.print('\t');
  Serial.print("yval=");
  Serial.print(yval);
  Serial.print('\t');
  Serial.print("bval=");
  Serial.println(bval);
  delay(200);
}
```

Experiment_Figure 2.27 Sample Code UNO-Joystick

Set-Up Picture:



Experiment_Figure 2.28 The Setup: Joystick-UNO



Experiment_Figure 2.29 The reading on plotter changes with varying value of module based on the thumb pressure on module

- **Language Reference:**

Keywords that were used in the program: pinMode(), digitalRead(), analogRead(), Serial.

- **Application Effect**

Open the serial port monitor, and turning the joystick, one will see some different values by return.

2.1.5. Working with Audio Sensors

Other than light (in reference with LEDs), sound is also one of the factors that is put into consideration when it comes to read real time data. Sound alarms are basic examples of showing how sensors have become integral part of security systems. Other than alarms, sensors are also used by devices to read the voice (audio inputs) and act accordingly.

The following experiment is the basic example of how Arduino UNO boards are used to display the analog values captured by the BigSound module (audio sensor) based on the sound captured through microphone.

- **Overview:**

Experiment_Figure 2.30BigSound Module (An Audio Sensor)

In this experiment, Bigsound module is used and its analog value is displayed over monitor based on the sound captured through microphone.

- **Specification:**

Pin Definition:

UNO R3	Bigsound
A0	A0
GND	GND
5V	+
D8	D0

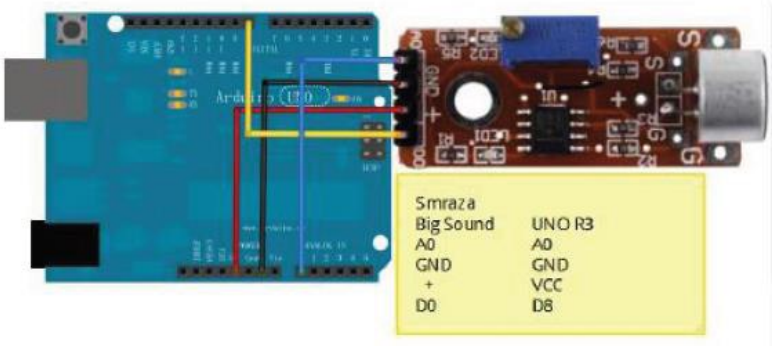
Experiment_Figure 2.31 Pin Definition - BigSound Module

- **Hardware Required:**

Material Diagram	Material Name	Quantity
	**Bigsound module**	**1**
	**USB cable**	**1**
	**Arduino UNO R3**	**1**
	**Male to female Jumper Wires**	**5**

Table_Experiment 2.5: Material Requirement - To display readings of an Audio Sensor

- **Connection:**
  a. **Connection Diagram:**



Experiment_Figure 2.32 Connection Diagram BigSound Module- Arduino UNO

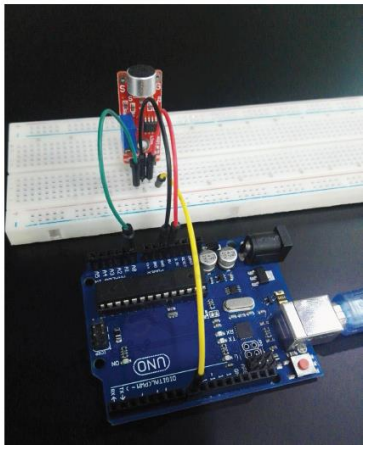
- **Sample Code:**

2_exp5_sound.ii

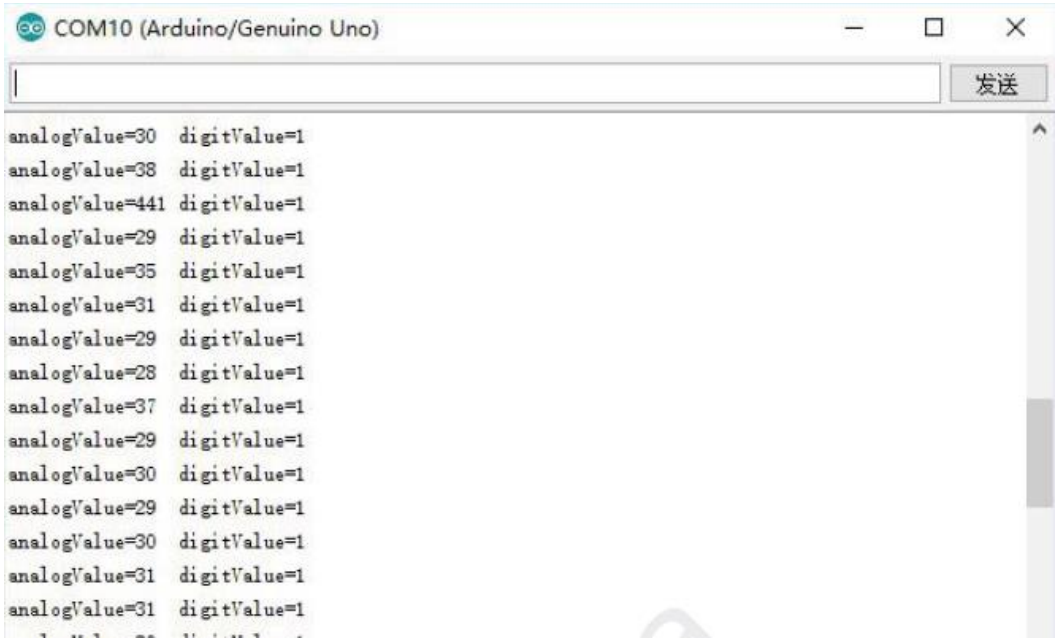
```
/*
 * Author: Ankit Raj
 * Date:2017/11/29
 * Email:araj@lakeheadu.ca
 * Function: to use Bigsound module
 */
int analog_sensor  = A0; // select the input pin for the potentiometer
int digit_sensor = 8; // select the input pin for the potentiometer
int analogValue ; // value from the analog input pin
int digitValue ;  //  value from the digit input pin
void setup ()
{
pinMode (digit_sensor, INPUT);
Serial.begin (9600);
}
void loop ()
{
  analogValue = analogRead (analog_sensor);
  digitValue=digitalRead(digit_sensor);
  Serial.print("analogValue=");
  Serial.print(analogValue);
  Serial.print('\t');
  Serial.print("digitValue=");
  Serial.println (digitValue);
  delay(300);
}
```

Experiment_Figure 2.33 Sample Code: To use BigSoundModule

- **Set-Up Picture:**



Experiment_Figure 2.34 The Setup: BigSound Module with Arduino UNO



Experiment_Figure 2.35 The reading on plotter changes with varying value of module based on the voice on microphone

- **Language Reference:**

Keywords that were used in the program: pinMode(), digitalRead(), analogRead(), Serial.

- **Application Effect**

Open the serial port monitor, and talking to microphone, one will see some different values (Analog and digital) by return.

2.2. RESEARCH PROBLEM

In all the above-mentioned scenarios (experiments), the only limitation we found was the inability of Arduino UNO to work wirelessly. The solution is to implement WiFi or a radio technique so that the board is made capable of interacting with interface wirelessly. Furthermore, to meet the mass-requirement of globalization, it is also necessary that the boards are able to interact with each other wirelessly.

The approach for this project would be made to make Arduino UNO interact wirelessly so that it can be used efficiently in home automation. In coming chapters, we will discuss all the techniques that can help to achieve the target of IoT. It will be seen that how Arduino UNO boards can be accessed wirelessly and can also communicate with various sensors in real-time.

CH. 3: Methodology

CHAPTER 3: METHODOLOGY

3.1.CONCEPTS AND DEFINITIONS

The main idea of this project is to create a smart home automation based on sensors using Arduino as the master controller. In previous chapter we came across the point where it was necessary for Arduino board to interact wirelessly with devices so as to achieve the project goal. But before we dive in to discuss the ways of implementation, it's better to first introduce ourselves with "Smart Home Automation" and different ways to interact with sensors and actuators.

3.1.1. Smart Home Automation

Smart-Home automation is an IoT concept that provides a way that all devices and appliances will be networked together to provide us with seamless control over all aspects of our home and more. It is a concept of automation of housework or household activity. Home automation has emerged quickly and is already in the trend nowadays. Increase in comfort and more rational use of energy and other resources are its main benefits. Generally, the home automation is very complex and carries a high cost along with it. Thus, an attempt is made to keep it less complex and cost effective by using Arduino as master controller.

The challenge faced while using Arduino UNO, as discussed in previous chapter, is that it was unable to interact wirelessly (i.e. WiFi or Radio). In order to facilitate WiFi feature, we need a WiFi-shield to be mounted over the board or ESP8266 chip, which is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller and also have 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to Wi-Fi. To make things a bit simple, we are using WeMos D1 (R2).

WeMos D1 is an Arduino UNO compatible WiFi board based on ESP8266EX. It has following feature:-

- 11 digital input/output pins, all pins have interrupt/pwm/I2C/one-wire supported(except for D0)
- 1 analog input(3.2V max input)
- Micro USB connection
- Power jack, 9-24V power input.
- Compatible with Arduino
- Compatible with nodemcu

In order to facilitate radio feature with Arduino UNO, NRF24L01 is used. NRF24L01 is a single chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a

fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced ShockBurst™ protocol engine. By the help of a SPI interface, output power, frequency channels, and protocol setup can be easily programmed. Current consumption of NRF24L01 is very low, only 9.0mA at an output power of -6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable. In order to facilitate Wi-Fi feature ESP8266 is used. ESP8266 ESP-01 is a Serial WIFI Transceiver Module. It is a cheap and easy way to connect any small microcontroller platform, like Arduino, wirelessly to Internet. ESP8266 has powerful on-board processing and storage capabilities that allow it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area. ESP-01 WIFI Transceiver Module is addressable over SPI and UART, making this an exceptionally easy choice for anyone wanting to build an IoT. One can use AT commands to connect to WiFi networks and open TCP connections without need to have TCP/IP stack running in their own microcontroller: One can simply connect any microcontroller to this module and start pushing data up to the Internet.

3.1.2. Modes of Communication

There is numerous ways to communicate with IoT. Other than wired connection following are the ways of communicating with IoT:

- WiFi
- Radio or Remote.

In order to achieve the successful communication, various techniques are required, which are as follows:

- SPI
- Serial Port
- I^2C

3.1.2.1. Serial Peripheral Interface (SPI)

Serial Peripheral Interface (SPI)[11]is an interface bus that helps to send data between microcontrollers and small peripherals. This interface uses isolated clock and data lines, along with a select line to choose the device one wishes to communicate with.SPI works in a somewhat unique way. It's a "synchronous" data bus, which implies that it utilizes isolate lines for data and a "clock" that keeps the two sides in culminating match up. The clock is an oscillating sign that advises the recipient precisely when to test the bits on the data line. This could be the rising or

[11]https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

falling edge of the clock flag; the datasheet will indicate which one to use. At the point when the receiver identifies that edge, it will promptly take a gander at the data line to peruse the following bits. Since the clock is sent alongside the data, indicating the speed isn't critical, despite the fact that gadgets will have the top speed at which they can work.

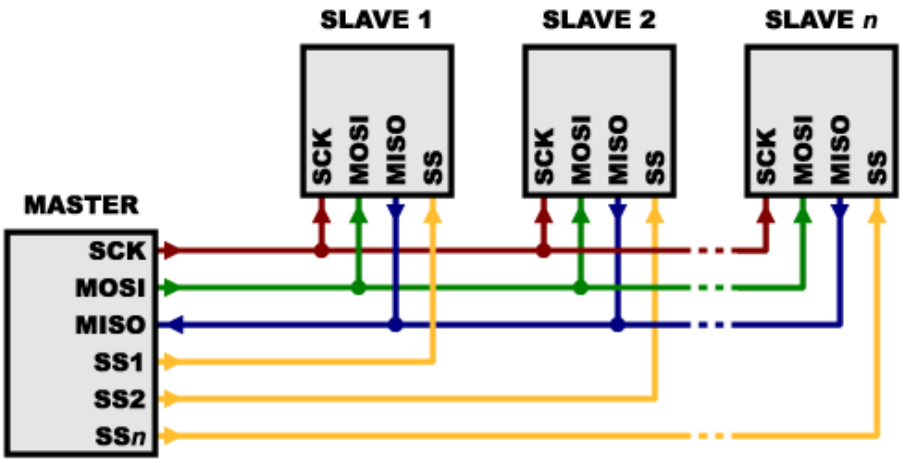


Figure 3.2 SPI Master-Slave concept

3.1.2.2. Serial Port

A serial communication interface through which information transfers in or out one bit at a time is termed as serial port. The asynchronous port (generally with TX and RX lines), which is most common port, has no control over data transmission and could not guarantee that both sides are running at precisely the same rate.

There could be an issue when two systems with a little different clock try to communicate with each other as computers normally rely on everything being synchronized to a single “clock”.

To solve this problem, asynchronous serial connections add extra start and stop bits to each byte that help the receiver sync up to data as it arrives. There should also be agreement on same transmission speed in advance. Slight differences in the transmission rate aren’t a problem because the receiver re-syncs at the start of each byte.

3.1.2.3.Inter-integrated Circuit (I^2C)

The Inter-integrated Circuit (I^2C)[12] Protocol is a protocol that provides communication between multiple chips (digital integrated circuits) with one or more master chips. I2C requires only two wires, as asynchronous serial, but those two wires can hold up to 1008 slave devices.

[12]https://learn.sparkfun.com/tutorials/i2c

Additionally, unlike SPI, I2C can support a multi-master framework, enabling in excess of one master to communicate with all devices on the transport (in spite of the fact that the master devices can't communicate with each other over the communication bus and should alternate utilizing the communication lines).

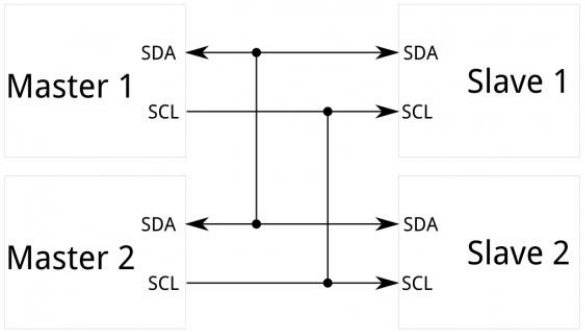


Figure 3.3 I2C Master-Slave concept

3.1.3. Libraries Used

The Arduino environment can be extended through the use of libraries, just like most programming platforms. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data.

For using a library in a sketch, one can select it from **Sketch>Import Library** option under the toolbar of IDE. A number of libraries come installed with the IDE, but one can also download or create their own.

3.1.3.1. Standard Libraries

- **EEPROM** - reading and writing to "permanent" storage
- **Ethernet / Ethernet 2** - for connecting to the internet using the Arduino Ethernet Shield, Arduino Ethernet Shield 2 and Arduino Leonardo ETH
- **Firmata** - for communicating with applications on the computer using a standard serial protocol.
- **GSM** - for connecting to a GSM/GRPS network with the GSM shield.
- **LiquidCrystal** - for controlling liquid crystal displays (LCDs)
- **SD** - for reading and writing SD cards
- **Servo** - for controlling servo motors
- **SPI** - for communicating with devices using the Serial Peripheral Interface (SPI) Bus
- **SoftwareSerial** - for serial communication on any digital pins. Version 1.0 and later of Arduino incorporate Mikal Hart's NewSoftSerial library as SoftwareSerial.
- **Stepper** - for controlling stepper motors

- **TFT** - for drawing text , images, and shapes on the Arduino TFT screen
- **WiFi** - for connecting to the internet using the Arduino WiFi shield
- **Wire** - Two Wire Interface (TWI/I2C) for sending and receiving data over a net of devices or sensors.

3.1.3.2. Custom libraries for Arduino

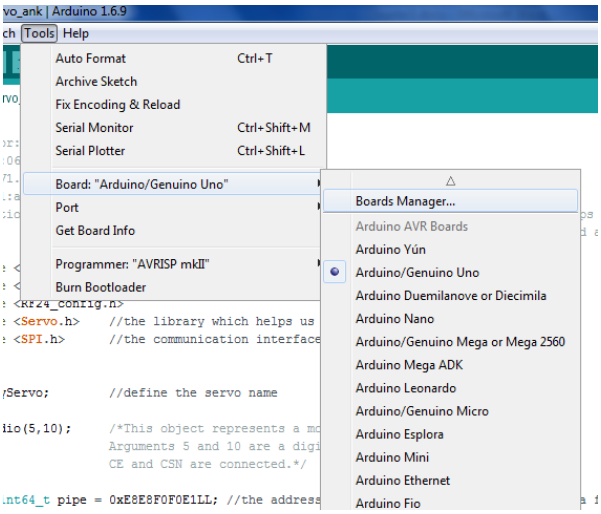
- **ESP8266WiFi**–This library is used to enable ESP8266 micro-chip on the board that enables the board to connect with the WiFi.
- **nRF24L01** – The library holds the driver details for NFR24LO1 to enable the radio feature for Arduino board so that communication can happen remotely by controlling the radio modem.

3.1.4. Preparing the Environment

A. Install the following as per the instruction on the corresponding site in the same order as below:

i. Arduino 1.6.9 https://www.arduino.cc/en/Main/OldSoftwareReleases

ii. ESP8266- Go under **tools – boards**<name of currently connected board>**- Boards Manager** and then search "ESP8266" and select **install**.



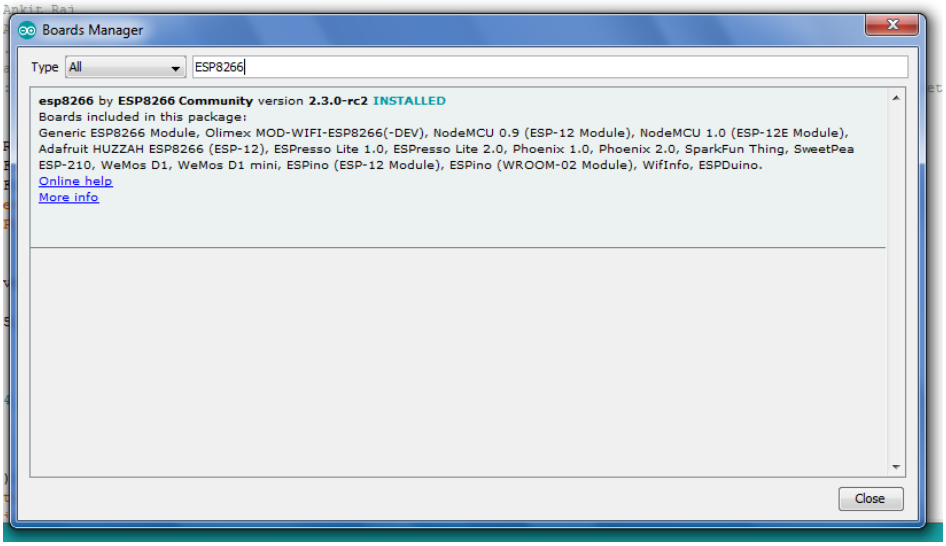


Figure 3.4 Library Manager

B. Attached the required custom libraries –

After adding the custom library by going under **sketch – include library – add .zip Library**, one need to include the library to the IDE by going under **>sketch – include library – <name of the custom library>**(as shown in figure below)

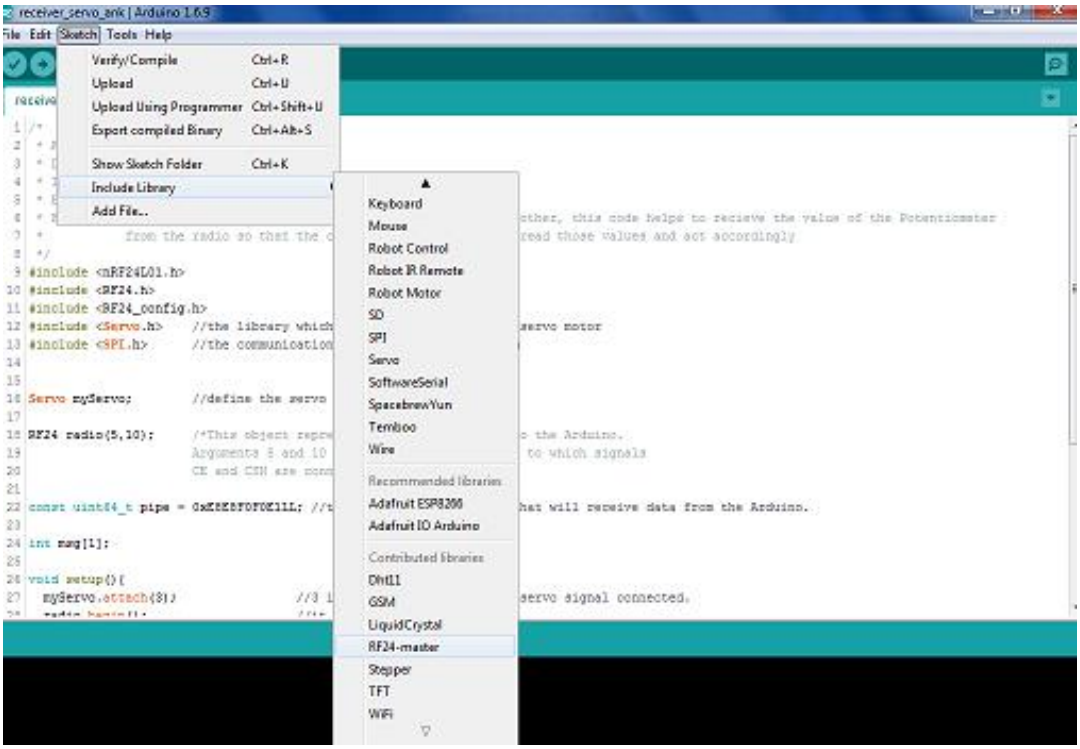


Figure 3.5 Inclusion of Custom Library

3.2. Implementation Methodology

In this project our main aim is to propose model for home automation system using Arduino as master controller. Proposed system architecture is shown in figure below:-

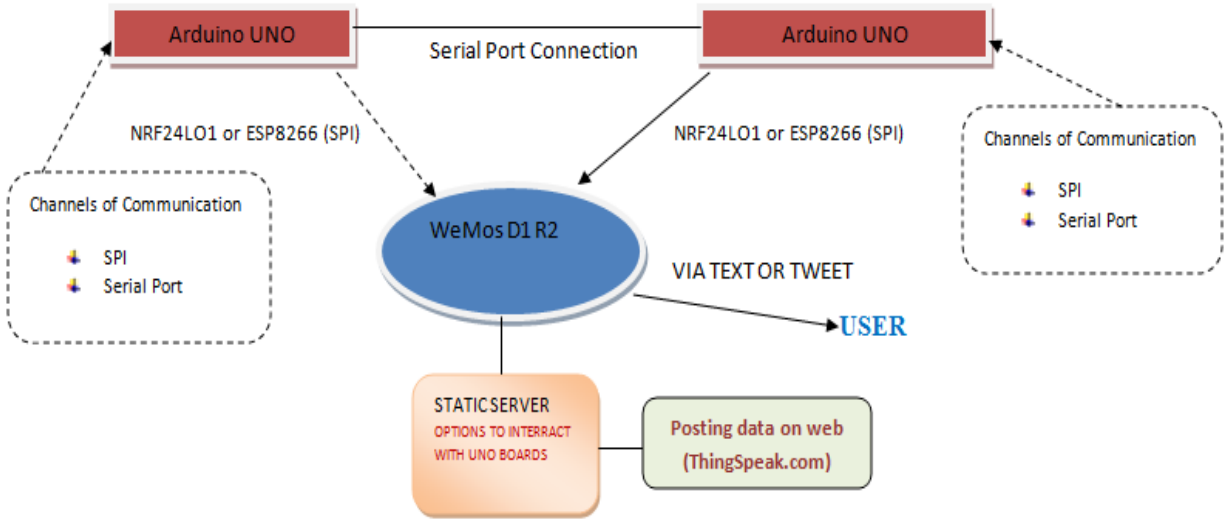


Figure 3.6 System Architecture – Methodology

From this model, an attempt is made to interact with sensors and actuators wirelessly to carry out certain routine and simultaneously keeping the user aware of their functioning status via text or tweet.

There are three modes of this project, described as follows:

A. **Monitoring the sensor values/readings:** Arduino enables users to read the sensor values as it convert the analog input signal into digital values. In this case the user would be getting a temperature or voltage value so as to carry out the desired task.
B. **Controlling Actuators:** Arduino gateway can trigger various actions through actuators. In this case, based on the values returned from the sensors, various actions would be performed like switching On/Off of LEDs or rotating of servo-motor.
C. **Pushing Notification to User (VIA Text or Tweet):** A function of pushing notifications of status of actuators or sensors through text or tweet would also be present so that the user (who is not around) can be updated accordingly about the device-interaction or performance.

Following are the steps to follow in order to validate the concept of methodology:

i. Do the required setup by connecting the sensors, actuators, Arduino as desired.
ii. After set up, upload the code to the respective board, by connecting it to computer (IDE).

iii. Disconnect all the connection between the boards and computer and let it work as per the wireless interaction of devices with the board.

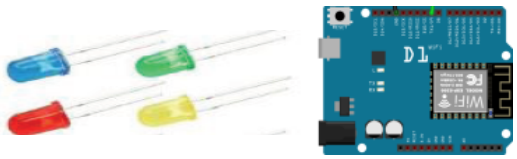
3.2.1. Experimental Evaluation

To achieve the desired goal of this methodology it is necessary to check whether the boards can:

- Communicate with actuators/sensors Wirelessly
- Communicate with other boards wirelessly

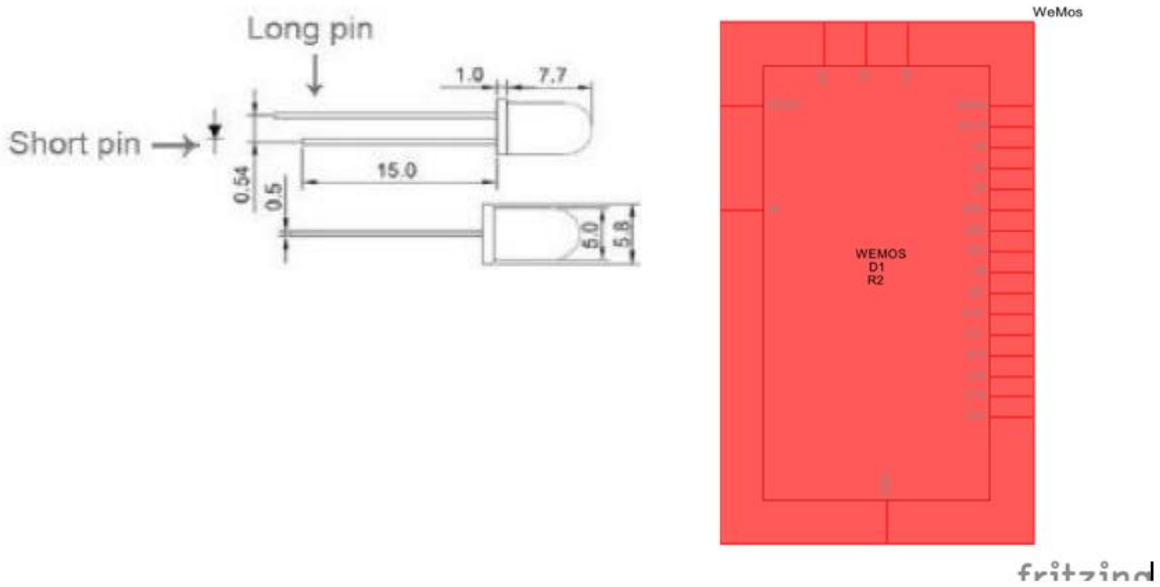
A. Following experiment[13] demonstrates how Arduino board is used to control actuators wirelessly.

- **Overview:**



Experiment_Figure 3.36 LEDs and WeMosD1 R2

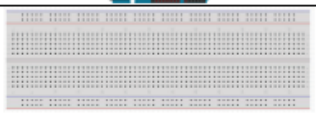
- **Specification:**



Experiment_Figure 3.37 Pin Specification of LED and WeMos D1 R2

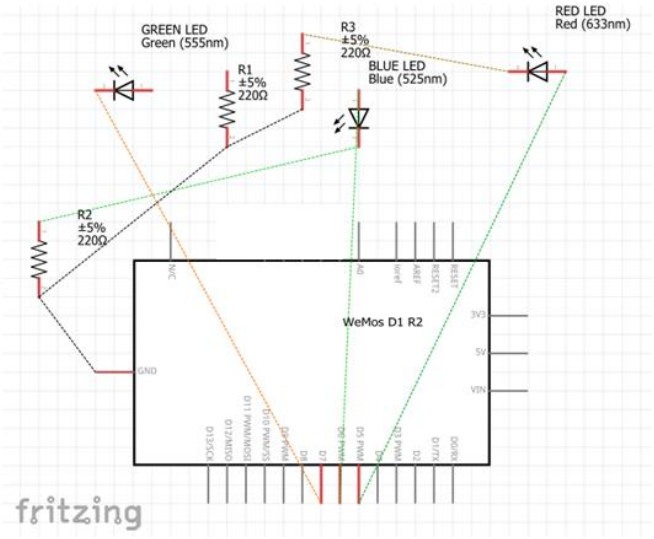
[13] https://www.amazon.ca/clouddrive/share/vck7GKUqwKrRUA7meriKJkQKaxE6ENx3YLDjS4Rd71N

- **Hardware Required:**

Material Diagram	Material Name	Quantity
	220/330Ω Resistor	3
	Red LED	1
	Green LED	1
	Blue LED	1
	USB cable	1
	WeMos D1 R2	1
	Breadboard	1
	Jumper Wires	Several
	Battery	1

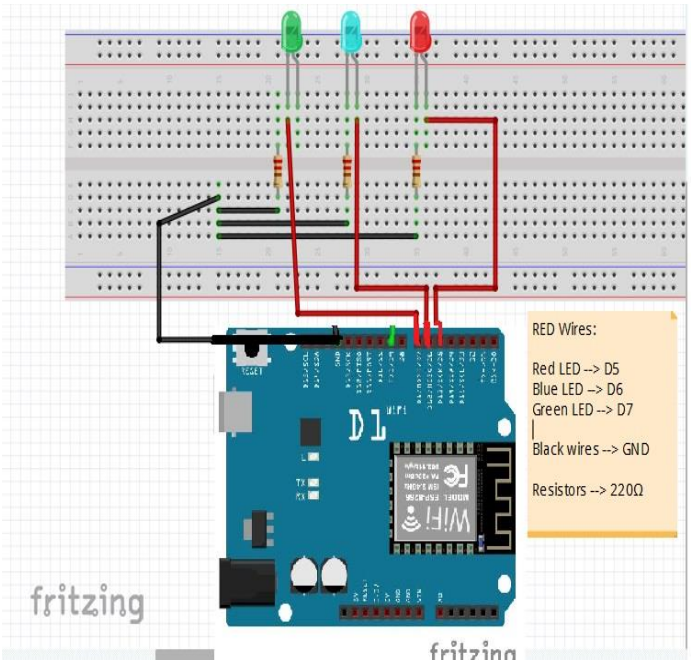
Table_Experiment 3.6 Material Requirement: Wireless Interaction with LED

- **Connection**
  - **a. Schematic**



Experiment_Figure 3.38 Schematic Connection of LEDs and WeMos D1 R2

b. Connection Diagram:



Experiment_Figure 3.39 Connection Diagram - LEDs and WeMos D1 R2

- **Sample Code:**

```
ɔter 3 _ Experiment 1

/*
Created By : Ankit Raj
Student Id: 0661545
Date: 25-Feb-2018
email: araj@lakeheadu.ca
Description: To interact with LEDs wirelessly
 */

//This example will set up a static IP - in this case 192.168.0.110

#include <ESP8266WiFi.h>

const char* ssid = "<SSid_Name>"; // I have hidden the actual wi-fi SSID and Passw
const char* password = "password";

int redled =D5;    // initialize digital pin.
int blueled =D6;
int greenled =D7;
WiFiServer server(80);
IPAddress ip(192, 168, 0, 110); // where the desired IP Address
IPAddress gateway(192, 168, 0, 1); // set gateway to match my network

void setup() {
  Serial.begin(115200);
  delay(10);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(redled, OUTPUT);
pinMode(blueled, OUTPUT);
pinMode(greenled, OUTPUT);
  digitalWrite(redled, LOW);
digitalWrite(blueled, LOW);
digitalWrite(greenled, LOW);


Serial.print(F("Setting static ip to : "));
Serial.println(ip);

// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(" Your Wi-Fi ");
IPAddress subnet(255, 255, 255, 0); // set subnet mask to match your network
WiFi.config(ip, gateway, subnet);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

// Start the server
server.begin();
Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL : ");
Serial.print("http://");
Serial.print(WiFi.localIP());
```

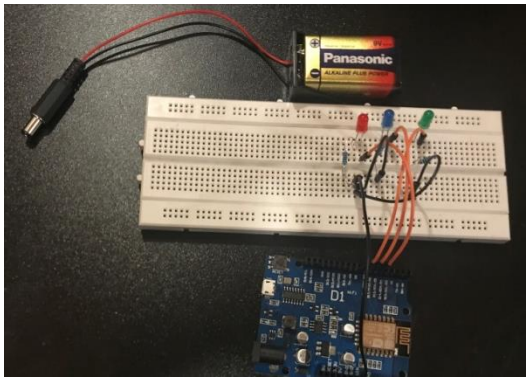
- Experiment_Figure 3.40 -a Sample Code - Wireless Interaction of LEDs

```
  Serial.print("http://");
  Serial.print(WiFi.localIP());
  Serial.println("/");

}

void loop() {
   digitalWrite(LED_BUILTIN, LOW);
  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Wait until the client sends some data
  Serial.println("new client");
  while(!client.available()){
    delay(1);
  }

  // Read the first line of the request
  String request = client.readStringUntil('\r');
  Serial.println(request);
  client.flush();

  // Initializing the values for all three pins

  int valueRed = LOW;
  int valueBlue = LOW;
  int valueGreen = LOW;
    if (request.indexOf("/LEDR=ON") != -1) {
   digitalWrite(redled, HIGH);
    valueRed = HIGH;
  }
  if (request.indexOf("/LEDR=OFF") != -1){
    digitalWrite(redled, LOW);
    valueRed = LOW;
  }
```

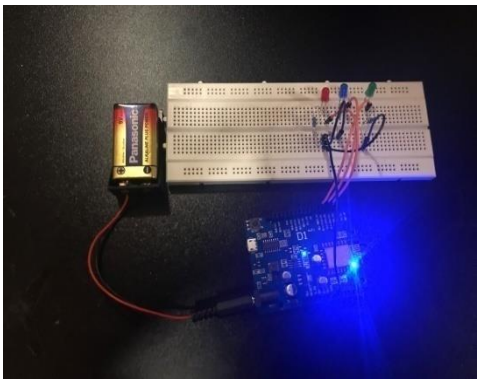
```
  // Return the response
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println(""); //  do not forget this one
  client.println("<!DOCTYPE HTML>");
  client.println("<html>");
  client.println("<body bgcolor=\"/#00FF00\"/>");
  client.print("<center><h1><u>Please click on respective link to operate the on-board LEDs</u></h1></cen

  client.println("<br><br>");
  client.println("Click <a href=\"/LEDR=ON\">here</a> turn the RED LED ON<br>");
  client.println("Click <a href=\"/LEDR=OFF\">here</a> turn the RED LED OFF<br>");
  client.println("<br><br>");
  client.println("Click <a href=\"/LEDB=ON\">here</a> turn the BLUE LED ON<br>");
  client.println("Click <a href=\"/LEDB=OFF\">here</a> turn the BLUE LED OFF<br>");
  client.println("<br><br>");
  client.println("Click <a href=\"/LEDG=ON\">here</a> turn the GREEN LED ON<br>");
  client.println("Click <a href=\"/LEDG=OFF\">here</a> turn the GREEN LED OFF<br>");
  client.println("</body>");
  client.println("</html>");

  delay(1);
  Serial.println("Client disconnected");
  Serial.println("");

}
```

Experiment_Figure 3.41 -b Sample Code - Wireless Interaction of LEDs

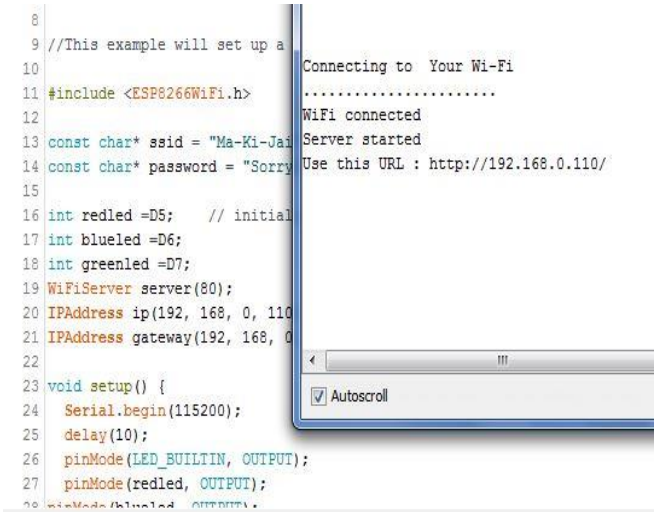
- **Set-Up Picture:**



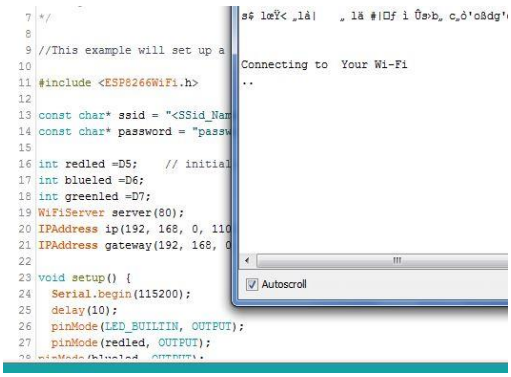
Experiment_Figure 3.42 The Setup – WeMos



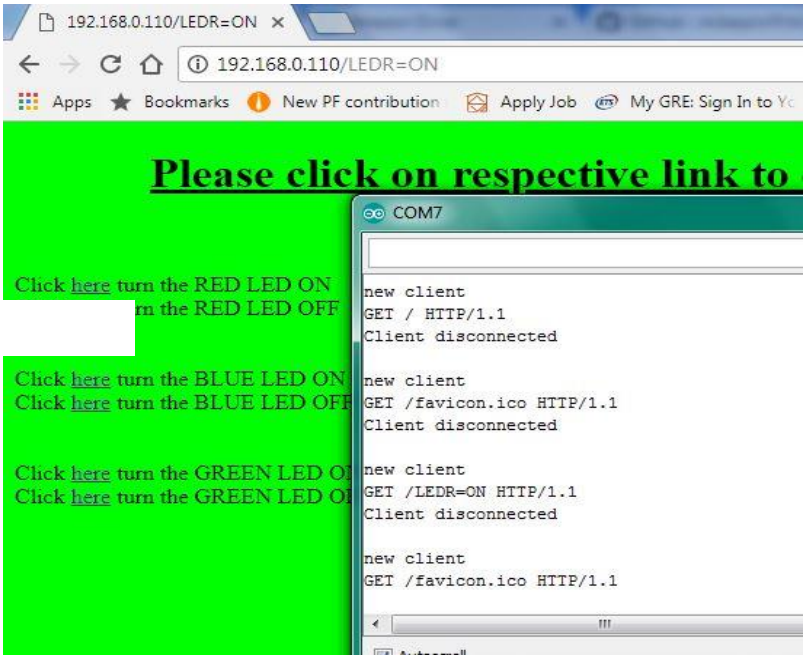
Experiment_Figure 3.43 When connected to external power



Experiment_Figure 3.44 When WiFi is connected it displays the URL for static server so as to interact with the board wirelessly.

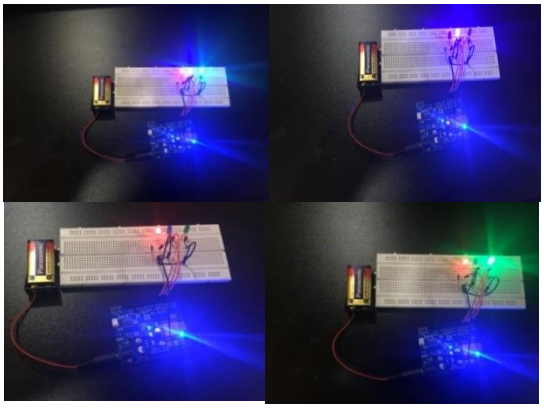


Experiment_Figure 3.45Monitor confirming the attempt to connect to wifi



Experiment_Figure 3.46The monitor confirming the request being sent by server to the board that acts as a client.

Experiment_Figure 3.47The static server screen with desired options to interact with the LEDs wirelessly.



Experiment_Figure 3.48 Different Pattern of LEDs as per the instruction passed by user through static web server

- **Language Reference:**

Following Library was used:
ESP8266WiFi *(#include <ESP8266WiFi.h>)*

The code keywords that were used in the program: WiFiServer.h, IPAddress , WiFi.config(), WiFi.begin(), WiFi.status(), WiFi.localIP(), WiFiClient, client.available(), client.readStringUntil, client.flush(), pinMode(), OUTPUT, INPUT, HIGH, LOW, delay() , Serial.begin(),Serial.print() , digitalWrite() .

- **Application Effect:**

With this application one can interact with LEDs that are connected with WeMos D1R2 board that is wirelessly connected to a server from where user can pass the desired commands to handle these lights.

B. The other technique needed to communicate with IoT is Radio. Following experiment[14] confirms that two boards can interact with themselves wirelessly:

- **Overview:**

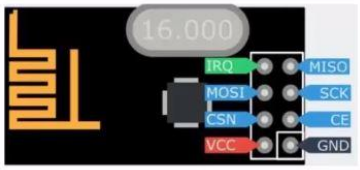
Experiment_Figure 3.49 NFR24LO1

[14] https://www.amazon.ca/clouddrive/share/thuwi7HX0JgUN2m4RmXQZ6DEvQYF7bpu9DBFKZRHfeT

In this experiment, two Arduino UNO boards are used in order to interact wirelessly with each other through a RF transceiver IC - nRF24L01.

- **Specification:**

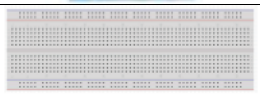
The module communicates with the controller via SPI interface.



VCC - power - from 1.9 up to 3.3 V
GND - ground
MOSI - SPI serial data input
MISO - SPI serial data output
SCK - SPI clock
CSN - low state on this pin indicates that with this module the controller wants to communicate.
CE - signal activating receiving and transmitting. In receive mode, the high state indicates that he wants to receive. In transmit mode, pulse sends one packet of data.
IRQ - interrupt output. It does a low state pulse when data is waiting to receive, or when the data was properly sent.

Experiment_Figure 3.50 Pin Specification: NFR24LO1

- **Hardware Required:**

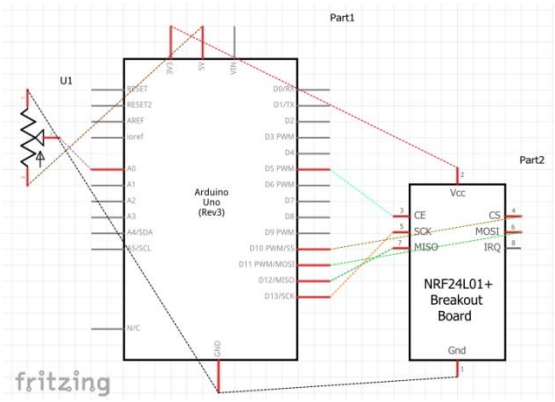
Material Diagram	Material Name	Quantity
	nRF24L01	2
	10KΩ Potentiometer	1
	Servo Motor	1
	USB cable	2
	Arduino UNO R3	2
	Breadboard	1
	Jumper Wires	Several
	Battery	1

Table_Experiment 3.7 Material Required for Radio interaction

- **Connection**

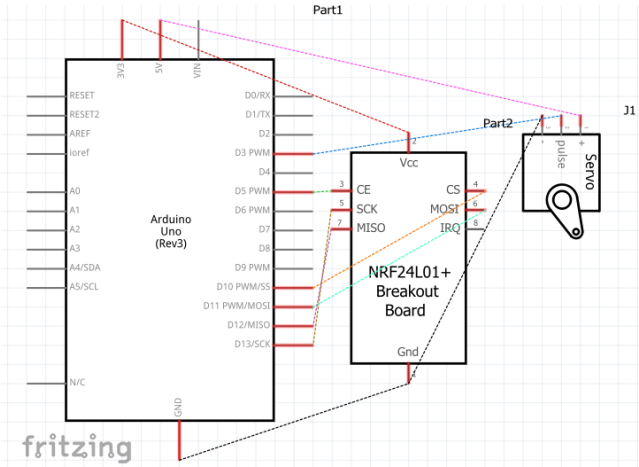
a. Schematic

Transmitter : The connection includes nRF24L01and Potentiometer along with Arduino →



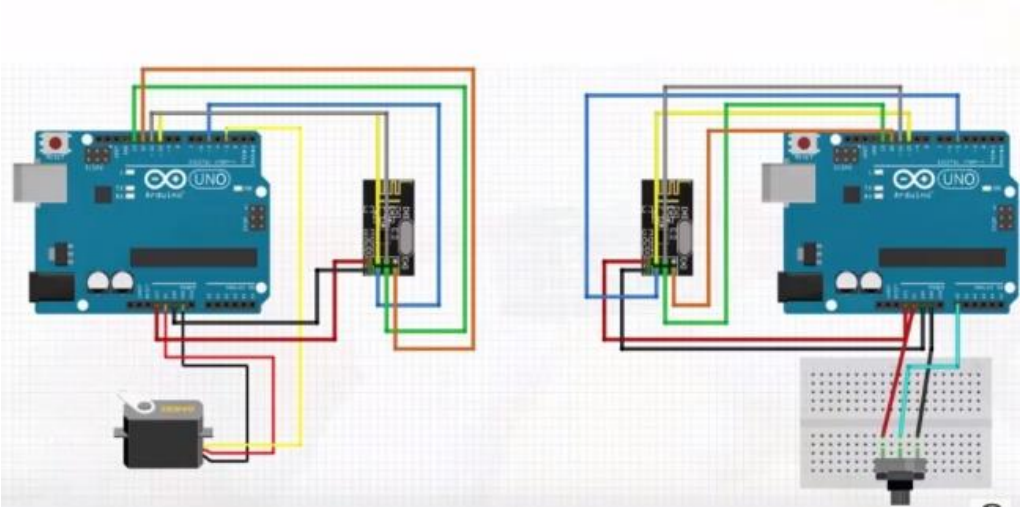
Experiment_Figure 3.51 Transmitter: The connection includes nRF24L01 and Potentiometer along with Arduino

Receiver : The connection includes nRF24L01and servo motor along with Arduino →



Experiment_Figure 3.52 Receiver: The connection includes nRF24L01 and servo motor along with Arduino

b. Connection Diagram:



Experiment_Figure 3.53 Connection Diagram: Transmitter & Receiver

- **Sample Code:**

Rx.ino

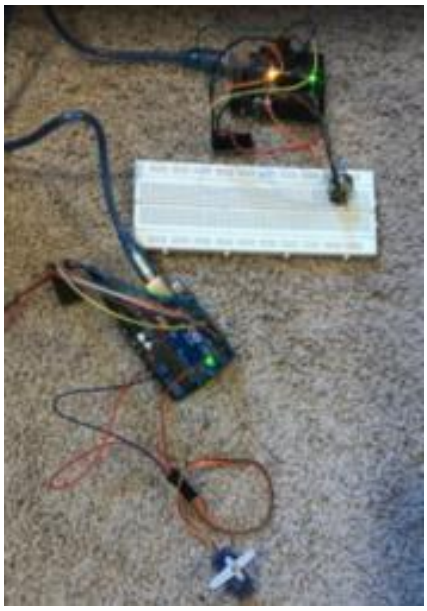
```
 * Email:araj@lakeheadu.ca
 * Function: using nRF24L01 two arduino boards in
 *           from the radio so that the connected
 */
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <Servo.h>      //the library which helps u
#include <SPI.h>        //the communication interfa


Servo myServo;          //define the servo name

RF24 radio(5,10);       /*This object represents a
                        Arguments 5 and 10 are a di
                        CE and CSN are connected.*/

const uint64_t pipe = 0xE8E8F0F0E1LL; //the addre

int msg[1];

void setup(){
  myServo.attach(3);                //3 is a digi
  radio.begin();                    //it activate
  radio.openReadingPipe(1, pipe);   //determines
  radio.startListening();           //enable rece
  }

void loop(){
  if(radio.available()){            //checks whet
    bool done = false;              //returns a
```

Experiment_Figure 3.54 Code Sample a: Receiver

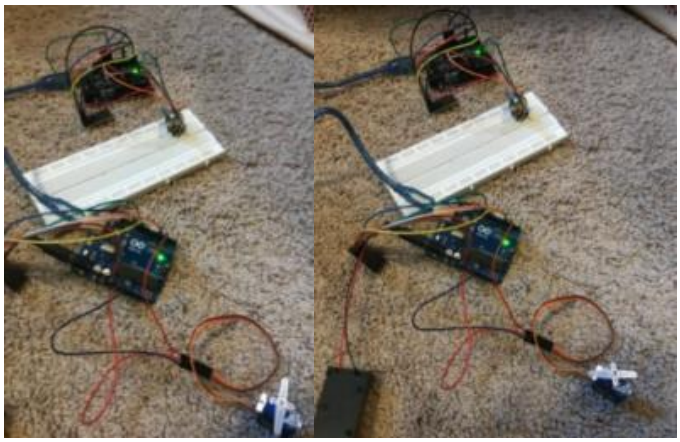
```
 * Author: Ankit Raj
 * Date:06-April-2018
 * IDE V1.6.9
 * Email:araj@lakeheadu.ca
 * Function: using nRF24L01 two arduino bo
 *           to the radio so as the servo
 */

#include <nRF24L01.h>
#include <RF24.h>    //the library which he
#include <RF24_config.h>

#include <SPI.h>                          //th

int msg[1];
int sensorPin = A0;     // select the input
RF24 radio(5,10);                         //5
int sensorValue = 0;  // variable to store
int outputValue = 0;
const uint64_t pipe = 0xE8E8F0F0E1LL; //th


void setup(void){
  radio.begin();                          //it
  radio.openWritingPipe(pipe);            //se
  Serial.begin(115200);
}

void loop(void){
```

Experiment_Figure 3.55 Code Sample b: Transmitter

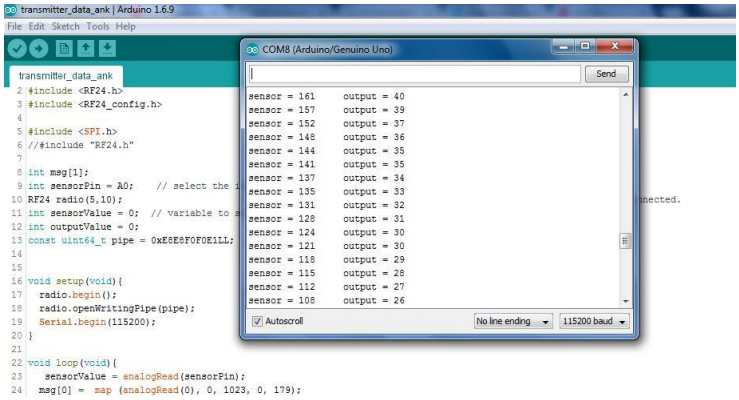
- **Set-Up Picture:**



Experiment_Figure 3.56The Setup: Tx & Rx



Experiment_Figure 3.57Rotating shaft of motor as per the values of potentiometer



Experiment_Figure 3.58Different values being transmitted to receiver for subsequent direction of motor shaft.

- **Language Reference**

Following Libraries were used:

nRF24L01.h *(#include <nRF24L01.h>)*	-- the library holds the driver details
RF24.h *(#include <RF24.h>)*	-- the library which helps us to control the radio modem
RF24_config.h *(#include <RF24_config.h>)*	-- the library which helps us to make radio modem compatible with Arduino
SPI.h*(#include <SPI.h>)*	-- the communication interface with the modem
Servo.h (#include <Servo.h>)	-- the library which helps us to control the servo motor

The code keywords used in the program were:uint64_t (to hold the address of the modem), radio.begin() (it activates the modem), radio.openWritingPipe(pipe)(sets the address of the receiver to which the program will send data), Serial.begin(), radio.write () , radio.available() (checks whether any data have arrived at the address of the modem), myServo.attach(<pin_number>);(<pin_number> is a digital pin to which servo signal connected), radio.begin() (it activates the modem), radio.openReadingPipe(1, pipe) (determines the address of our modem which receive data), radio.startListening()(enable receiving data via modem).

- **Application Effect:**

With this application one can interact with different Arduino boards that are connected with nRF24L01, which is capable of transmitting and receiving data wirelessly, and carry out specific operation. Like in this experiment the direction of the shaft implemented over servo-motor is being controlled by potentiometer wirelessly.

3.2.2. Conclusion

With both experiments mentioned above it is clear that we can use Arduino UNO boards for wireless interaction with devices and it can be used in home automation. In the next chapter we will discuss the prototype of the model and its implementation. The main goal of this project is to make a home automation system that interacts with sensors and actuators keeping Arduino UNO as master controller.

CH. 4: Project Implementation and Results

CHAPTER 4: PROJECT IMPLEMENTATION AND RESULTS

The end of previous chapter describes the methodology and the concept needed to develop the home automation system as per the project goal. In the previous chapter we discussed the method of interacting wirelessly with sensors using radio (NRF24L01), but WeMos D1 R2 which is acting as the third console in this project has a limitation with communicating through radio channels. Thus, implementation of communicating through WiFi (ESP8266) is approached in this project which would be discussed later in this chapter. But, before we discuss the various phases of implementation, we need to discuss the hardware required in this project and their respective specifications.

4.1. REQUIRED HARDWARE AND SPECIFICATIONS

Material Diagram	Material Name	Quantity
	ESP8266 (ESP-01)	2
	3.3V voltage Regulator (LM317)	2
	Red LED	3
	Green LED	3
	Blue LED	1
	220/330Ω Resistors	Several
	USB cable	2
	Temperature-Humidity Sensor (DHT11)	1
	WeMos D1 R2	1
	Arduino UNO R3	2
	Breadboard	1
	Jumper Wires	Several
	Battery	1

Table 4.1: Material Required

This project consists of one sensor – DHT11 (Temperature and Humidity sensor) and various LEDs acting as actuators. Above mentioned are the materials required to build the home automation system, as described under the project goal, which interacts with the sensors and actuators wirelessly keeping Arduino UNO as master controller. Details of the major component for this project are mentioned below.

- **Arduiono UNO R3:**

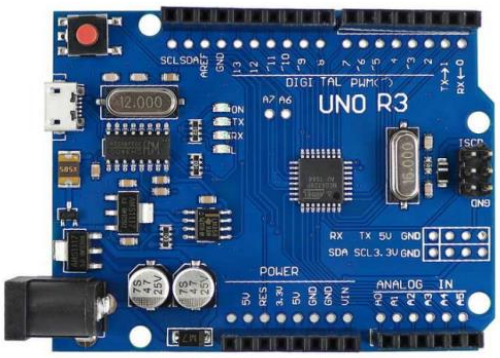


Figure 4.7: Arduino UNO R3

As explained in previous chapter (Ref: Chapter-2), ***Arduino Uno***[15] is a microcontroller board based on the ATmega328P (datasheet). It has 14 computerized input/output pins (of which 6 can be utilized as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset catch. It contains everything expected to help the microcontroller; just interface it to a PC with a USB link or power it with an AC-to-DC adapter or battery to begin. This project use two of these boards as Console-1 and Console-2 to implement the Phase-I.

- **WeMos D1 R2:**

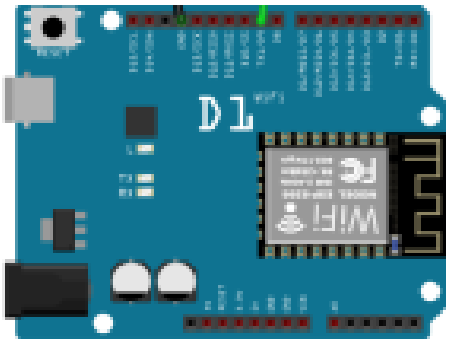


Figure 4.8: WeMos D1 R2

WeMos D1 R2 is used as a third console in order to implement Phase –II and Phase –III. WeMos D1 is an Arduino UNO compatible WiFi board based on ESP8266EX, features of which have already been discussed in previous chapter.

[15] https://store.arduino.cc/usa/arduino-uno-rev3

ESP8266 ESP-01:

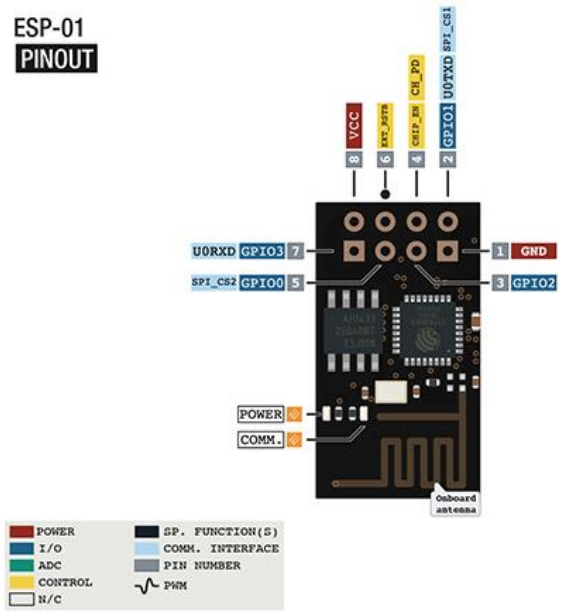


Figure 4.3: ESP-01

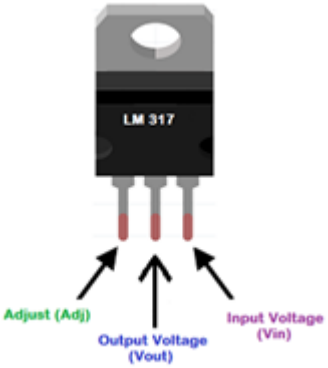


Figure 4.4: LM317

ESP-01 is used to enable wireless communication on one of the UNO boards (console-2). It uses a separate 3.3V voltage regulator LM317 (Figure 4.4) as ESP-01 does not support 5V.

DHT-11:

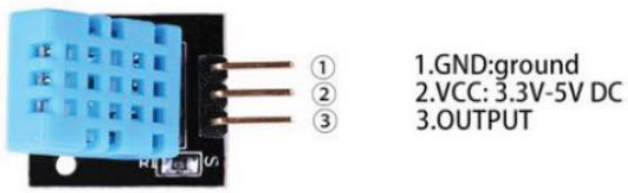


Figure 4.5: DHT-11

DHT-11 is a temperature and humidity sensor that is a high performance 8-bit microcontroller. It uses 5V power supply to sense, measure and regularly report the relative humidity in air along with temperature.

4.2. PHASES OF IMPLEMENTATION

For the development of this project we use Arduino IDE and few custom libraries. We have already discussed the method of adding the custom libraries in previous chapter.

There are three phases of this project. Below mentioned are the details of these phases.

4.2.1. Phase I – Two Board Communication

In the first phase, two Arduino UNO boards (Console-1 and Console-2) communicate serially. Console -1 captures the temperature and humidity data through DHT-11 sensor and send it to Console-2. Console-2 reads the temperature and humidity separately and based on a pre-defined range (Temperature is stable when it ranges between 18 degrees to 25 degrees) decide whether the temperature is stable or not stable. If the temperature is less than (or equal to) 25 and is more than 18 degrees it sends “STABLE” as acknowledgement to Console-1 and lights up green LED. If the temperature is more than 25 degrees or less than 18 degrees then it sends “NTSTBL” as acknowledgement to Console-1 and lights up red LED.

Console-1 also acknowledges the data received from Console-2 through LEDs. If it receives “STABLE”, it lights up green LED. If Console-1 receives “NTSTBL” from Console-2 then red LED gets ON. In case of serial lag, where the string character sent from Console-2 is not properly received, blue LED gets ON.

For enabling the second phase, Console-2 is connected to ESp-01 to transfer the data to Console-3. Following is the connection diagram:

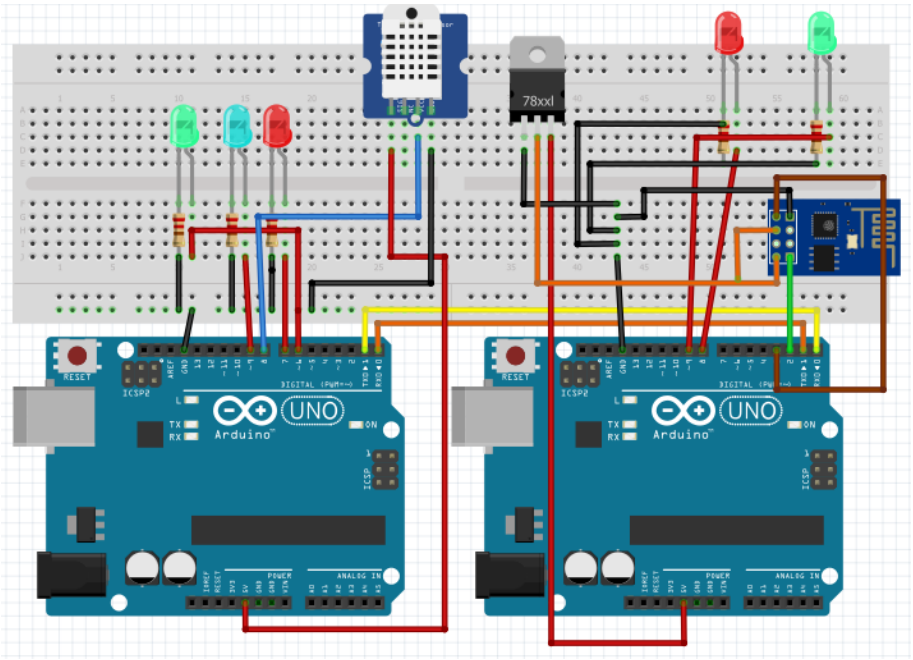


Figure 4.6: Serial Communication between Console 1 & Console 2

```
if(i>0){
 sbh=String(recstr);
 //Serial.print(sbh);
 if(sbh.equals("STABLE")){
     digitalWrite(rledPin, LOW);
     digitalWrite(gledPin, HIGH);
     digitalWrite(bledPin, LOW);
     delay(500);
 }
   if(sbh.equals("NTSTBL")){
     digitalWrite(rledPin, HIGH);
     digitalWrite(gledPin, LOW);
     digitalWrite(bledPin, LOW);
     delay(500);
 }
   if((!(sbh.equals("STABLE")))||((!(sbh.equals("NTSTBL"))))){
     digitalWrite(rledPin, LOW);
     digitalWrite(gledPin, LOW);
     digitalWrite(bledPin, HIGH);
     delay(500);
 }

}
```

Code Sample 4.1: Console 1 code of concept for acknowledging data sent from Console 2

Libraries Used: DHT11.h and SoftwareSerial.h.

For initiating next phase, WiFiClient.h, WiFiServer.h, WiFiUdp.h, libraries are used.

```
#include <SoftwareSerial.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
char mystr[]="STABLE";
char mystr2[]="NTSTBL";
const int gledPin =  9;      // the number of the LED pin
const int rledPin =  8;
String ank;
int Lov;
char recstr[5]; //Initialized variable to store recieved data

SoftwareSerial mySerial(0, 1);//(Rx,Tx)

String data;
SoftwareSerial esp(2, 3);// RX & TX For ESP-01
```

Code Sample 4.2: Console 2 concept of receiving data from Console 1 through serial ports and then broadcasting it to Console 3 wirelessly using ESp8266

```
void connectWifi() {
String cmd = "AT+CWJAP=\"" +ssid+"\",\"" + password + "\"";
esp.println(cmd);
delay(4000);
if(esp.find("OK")) {
esp.println("Connected!");
}
else {
connectWifi();
esp.println("Cannot connect to wifi"); }
}
```

Code Sample 4.3: Console 2 concept of connecting to WiFi of home router using ESP8266

```
void httppost () {
esp.println("AT+CIPSTART=\"TCP\",\"" + server1 + "\",80");//start a TCP connection.
if( esp.find("OK")) {
esp.println("TCP connection ready");
} delay(1000);
String postRequest =
"POST " + uri + " HTTP/1.0\r\n" +
"Host: " + server1 + "\r\n" +
"Accept: *" + "/" + "*\r\n" +
"Content-Length: " + data.length() + "\r\n" +
"Content-Type: application/x-www-form-urlencoded\r\n" +
"\r\n" + data;
String sendCmd = "AT+CIPSEND=";//determine the number of caracters to be sent.
esp.print(sendCmd);
```

Code Sample 4.4: Console 2 concept of broadcasting Console 1 data received at its end to the static server created by WiFi of home router using ESP8266

4.2.2. Phase II – Three Board Communication

The Console-2 from previous phase connects with the available WiFi network through ESP-01. The Console-2 connects to the WiFi and transmits the data (temperature and humidity) received from Console-1 on the server so that Console-3 connects to the server and receive the same data at its end. This result into a three board communication as desired for achieving the goal of the project.

Console-3 creates a local server to broadcast the data received from Console-2. Console-3 interacts with another set of LEDs based on the data recorded at its end (RED for temp>25 [temp<18] and GREEN for temp<25 [temp >18]). Following is the connection diagram:

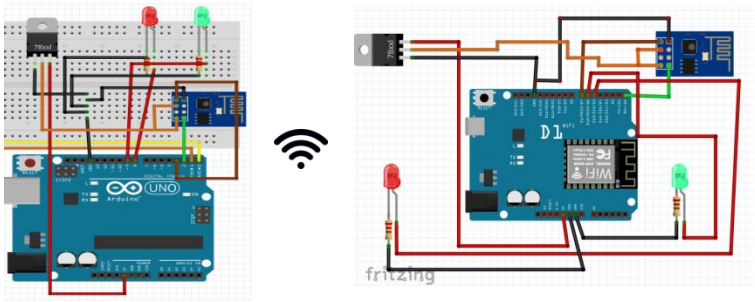


Figure 4.7: Communication using WiFi between Console 2 & Console 3

Libraries Used: ESP8266WiFi.h,WiFiClient.h, WiFiServer.h, WiFiUdp.h and SoftwareSerial.h.

```
// Connect to WiFi network
Serial.println();
Serial.println();
Serial.print("Connecting to ");
Serial.println(" Your Wi-Fi ");
IPAddress subnet(255, 255, 255, 0); // set subnet mask to match your network
WiFi.config(ip, gateway, subnet);
WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");

// Start the server
server.begin();
Serial.println("Server started");

// Print the IP address
Serial.print("Use this URL : ");
Serial.print("http://");
Serial.print(WiFi.localIP());
Serial.println("/");
```

Code Sample 4.5: Console 3 concept of creating a static local server

```
String  readWifiSerialMessage(){
 char value[100];
 int index_count =0;
 while(wifiSerial.available()>0){
   value[index_count]=wifiSerial.read();
   index_count++;
   value[index_count] = '\0'; // Null terminate the string
 }
 String str(value);
 str.trim();
 return str;
}
```

Code Sample 4.6: Console 3 concept of reading the data sent from Console 2 wirelessly using ESP-01

4.2.3. Phase III – Alerting the user

The most defining phase of this project is the third phase. This phase is comprised of all such cases that user should be made aware of. In this case, since the interaction is with the temperature sensor thus user is made aware of the current temperature and humidity, also alerted if the temperature is over the room temperature. In order to facilitate this feature, Console-3 attempts to connect with ThingSpeak.com and post the temperature and humidity data on the website. Additionally, when the temperature gets over room temperature range, Console-3 uses the ThingTweet API of ThingSpeak to post tweet to the user.

```
#include <ESP8266WiFi.h>
 #include <Adafruit_Sensor.h>
 #include<SoftwareSerial.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#include <WiFiUdp.h>
String temp[2];
String hum[2];
  int valueRed = LOW;
  int valueGreen = LOW;
SoftwareSerial wifiSerial(7,0);      // RX, TX for Receiving msg from ESP-01 of Console 2

String API = "MGWRT5S8QFWTGMSS";// For Tweet
String apiKey = "9WYW3LQR4PAG6AYC"; // For data log
```

Code Sample 4.7: Console 3 concept of broadcasting data received from console 2 to the static and Web server including Twitter by using WiFi of home router

```
 //****************For Things Speak Data Log************//


          if (client.connect("api.thingspeak.com",80)) {
          String postStr = apiKey;
          postStr +="&field1=";
          postStr += String(t);
          postStr +="&field2=";
          postStr += String(h);
          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);

          }
          client.stop();
```

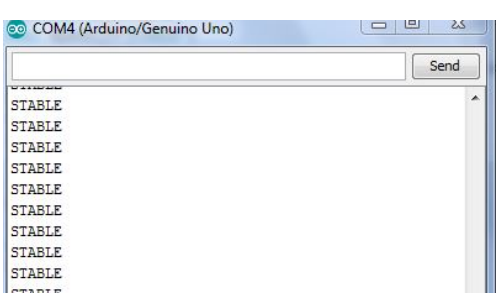
Code Sample 4.8: Console 3 concept of posting the data on ThingSpeak.com

```
//****************For Sending Tweets************//
if (client.connect("184.106.153.149", 80)){
   if(((t>=18)&&(t<=25))&& (request.indexOf("/LEDR=ON") != -1)) {
    tweet = "Stablizing_Temp_Measures_Are_Being_Taken.";
client.print("GET /apps/thingtweet/1/statuses/update?key=" + API + "&status=" + tweet + " HTTP/1.1\r\n");
client.print("Host: api.thingspeak.com\r\n");
client.print("Accept: */*\r\n");
client.print("User-Agent: Mozilla/4.0 (compatible; esp8266 Lua; Windows 7 Ultimate)\r\n");
client.print("\r\n");
   }
   else  if(((t<18)||(t>25)) && (request.indexOf("/LEDR=ON") != -1)){
    tweet = "Temperature_Not_stable_You_Need_To_Visit_The_Place_To_Fix_This_Issue.";
client.print("GET /apps/thingtweet/1/statuses/update?key=" + API + "&status=" + tweet + " HTTP/1.1\r\n");
client.print("Host: api.thingspeak.com\r\n");
client.print("Accept: */*\r\n");
client.print("User-Agent: Mozilla/4.0 (compatible; esp8266 Lua; Windows 7 Ultimate)\r\n");
client.print("\r\n");
   }
   if(((t>=18)&&(t<=25))&& (request.indexOf("/LEDR=OFF") != -1)) {
    tweet = "Issue_resolved_Temperature_Is_Under_Control!";
client.print("GET /apps/thingtweet/1/statuses/update?key=" + API + "&status=" + tweet + " HTTP/1.1\r\n");
client.print("Host: api.thingspeak.com\r\n");
client.print("Accept: */*\r\n");
client.print("User-Agent: Mozilla/4.0 (compatible; esp8266 Lua; Windows 7 Ultimate)\r\n");
client.print("\r\n");
   }
```

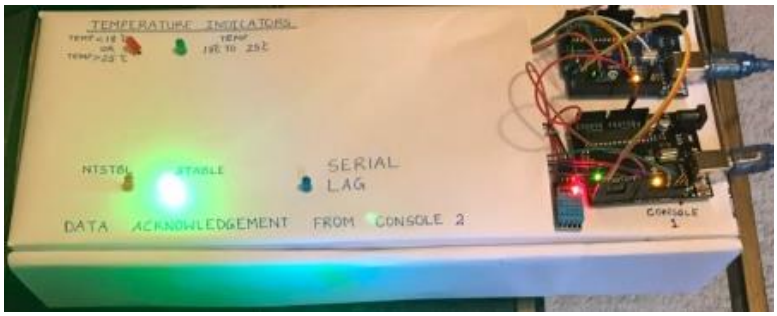
Code Sample 4.9: Console 3 concept of posting tweets using API key of Twitter

4.2. Results

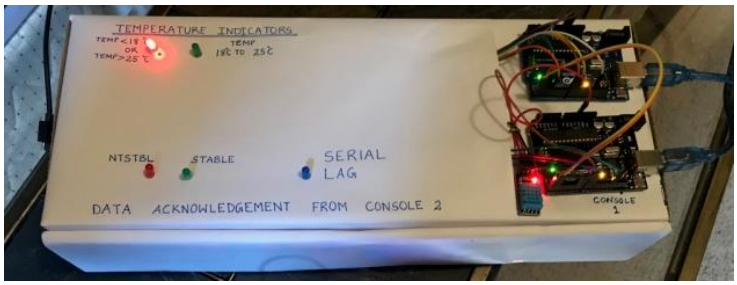
- Phase I Results:-

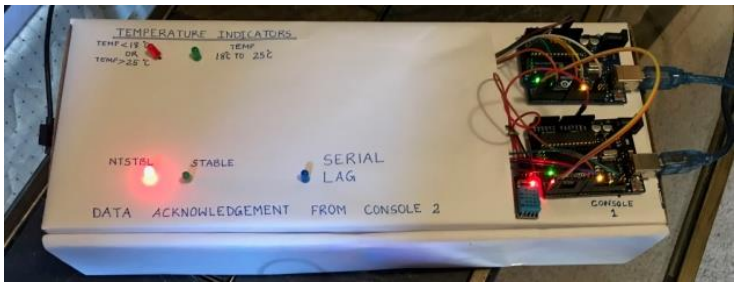


Experiment_Figure 4.1: When Temperature is stable and is acknowledged by Console 2 simultaneously

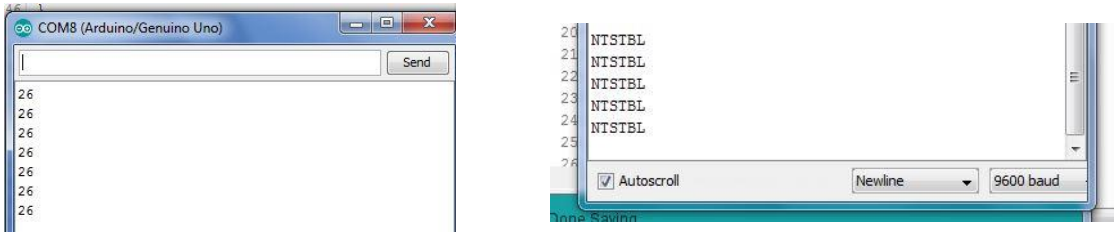


Experiment_Figure 4.2: Console 1 acknowledging the message from Console 2





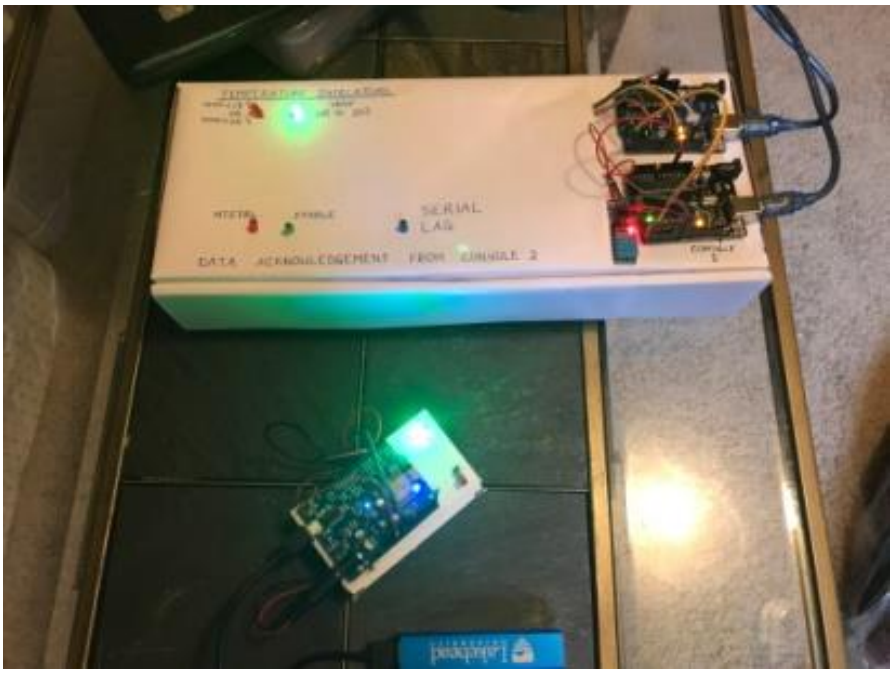
Experiment_Figure 4.3 - a: When Temperature is greater than room temperature (25 degrees)



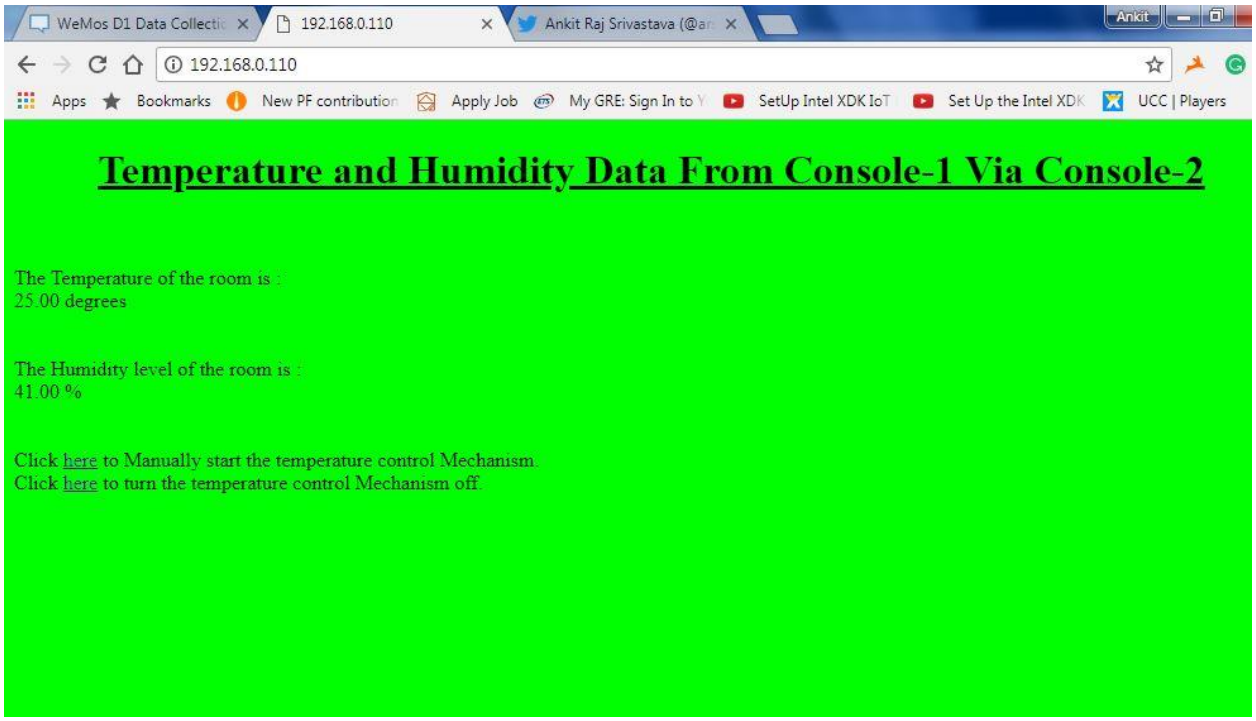
Experiment_Figure 4.3 - b: When Temperature is greater than room temperature (25 degrees)

Experiment_Figure 4.4: When there is a Serial-Lag

- Phase II Results:



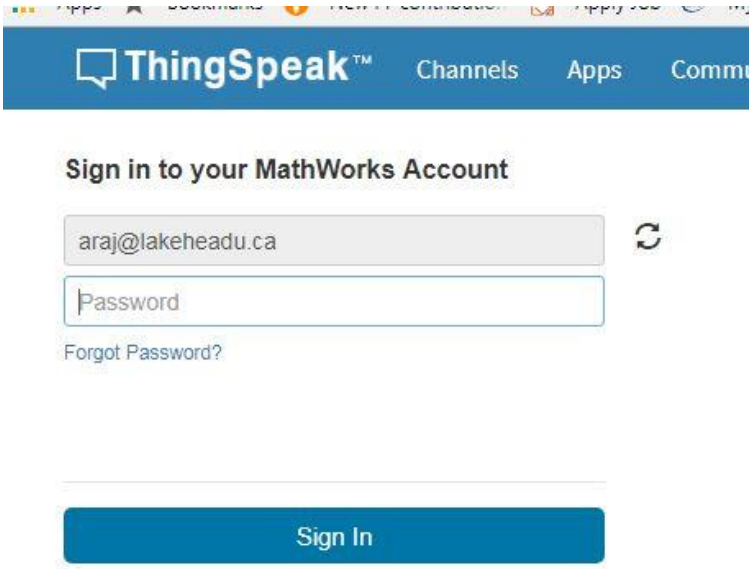
Experiment_Figure 4.5 - a: Console-3 acknowledging the stable Temperature through Green LED and creating a Static Server (Local) recording Temperature and Humidity sent from Console -1 via Console-2



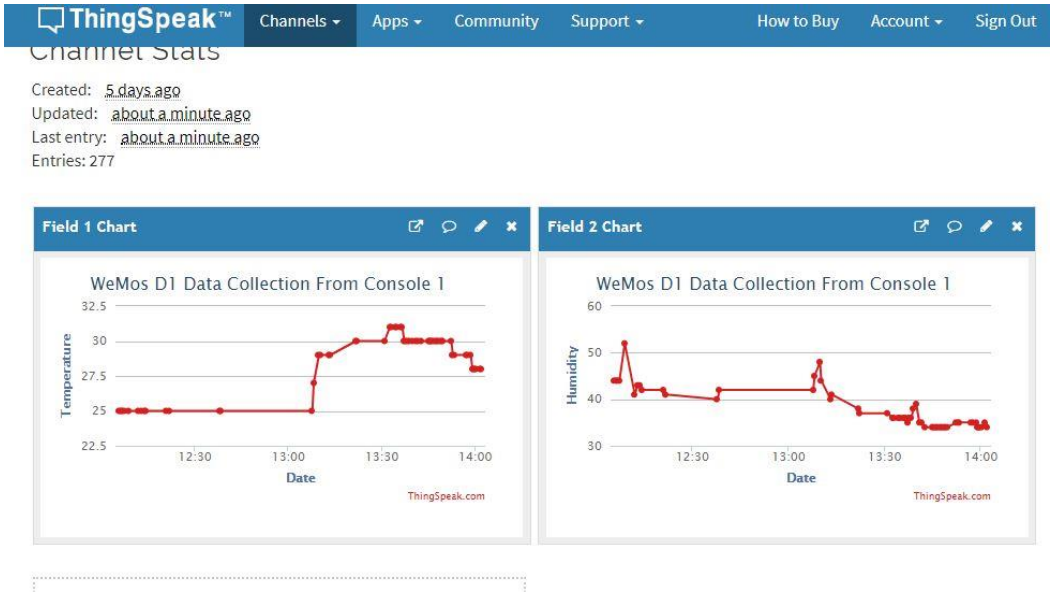
Experiment_Figure 4.5 - b: Console-3 acknowledging the stable Temperature through Green LED and creating a Static Server (Local) recording Temperature and Humidity sent from Console -1 via Console-2

- Phase III Results:

Experiment_Figure 4.6 - a: Login Screen for ThingSpeak.com in order to view data posted by Console -3 on website



Experiment_Figure 4.6 - b: Login Screen for ThingSpeak.com in order to view data posted by Console -3 on website



Experiment_Figure 4.7: Data Posted on Web Server of Thingspeak.com

Experiment_Figure 4.8: Several Tweets posted based on the interaction with Sensor

The above mentioned results show that the desired goal of creating smart home automation based on a sensor using Arduino as the master controller is achieved.

CH. 5: Conclusion

5.1. Conclusion
5.2. Future Developments

5.1. CONCLUSION

Building home automation system using Arduino UNO is challenging, as the board itself has a lot of limitation. However, one can still build an efficient system of home automation from UNO boards by using some additional accessories like ESP8266 and WeMos D1 R2 (WiFi enabled Arduino UNO). In this project, ESP-01 is used to facilitate with ESP8266, which is very economical but is the lower version of ESP8266. One can use ESP-12 to overcome the challenges faced in ESP-01, especially with the voltage regulator. WeMos D1 board too offered few challenges as it does not support Radio communication, which in return would have made this project more economical.

Considering all challenges faced, this project is still a good example of creating a smart home automation based on a sensor using ArduinoUNO as the master controller.

5.2. FUTURE DEVELOPMENTS

In spite of having a limited scope (due to challenges offered by WeMos D1 R2), this project still have a scope of further advancements in future. Following are few:

- **Global Interaction with Sensors:** This project offers to interact with the sensors locally (needs the user to be connected with the home WiFi). However, it also offers the scope of advancement so as to make user capable of interacting with the sensor from anywhere. For instance, user can post Tweet about switching off the lights or any possible actuators.

- **Adding Home Security:** Just like temperature sensors, there are also few bio-metric sensors available in the market which can be incorporated in this project so as to add security feature. For instance, user would be alerted if someone accesses any actuators linked with the sensor without proper authentication.

References:

1. Barroca, N., Borges, L.M., Velez, F.J., Monteiro, F., Górski, M. and Castro-Gomes, J., 2013. Wireless sensor networks for temperature and humidity monitoring within concrete structures. Construction and Building Materials, 40, pp.1156-1166.
2. Charlwood, S. and James-Roxby, P., 1998, August. Evaluation of the XC6200-series architecture for cryptographic applications. In International Workshop on Field Programmable Logic and Applications (pp. 218-227). Springer, Berlin, Heidelberg.
3. De Sousa, M., 2015. Internet of Things with Intel Galileo. Packt Publishing Ltd.
4. Eisenbarth, T. and Kumar, S., 2007. A survey of lightweight-cryptography implementations. IEEE Design & Test of Computers, 24(6).
5. Giusto, D., Iera, A., Morabito, G. and Atzori, L. eds., 2010. The internet of things: 20th Tyrrhenian workshop on digital communications. Springer Science & Business Media.
6. Gupta, V. and Simmons, D.G., 2010. Building the web of things with sun SPOTs. JavaOne.
7. Kuan, W.H., Tseng, C.H., Chen, S. and Wong, C.C., 2016. Development of a computer-assisted instrumentation curriculum for physics students: Using LabVIEW and Arduino platform. Journal of Science Education and Technology, 25(3), pp.427-438.
8. Santucci, G., 2010. The internet of things: Between the revolution of the internet and the metamorphosis of objects. Vision and Challenges for Realising the Internet of Things, pp.11-24.
9. Stallings, W., 2002. High-speed networks and internets: performance and quality of service. Pearson Education India.
10. Sundmaeker, H., Guillemin, P., Friess, P. and Woelfflé, S., 2010. Vision and challenges for realising the Internet of Things. Cluster of European Research Projects on the Internet of Things, European Commision, 3(3), pp.34-36.
11. Suresh, P., Daniel, J.V., Parthasarathy, V. and Aswathy, R.H., 2014, November. A state of the art review on the Internet of Things (IoT) history, technology and fields of deployment. In Science Engineering and Management Research (ICSEMR), 2014 International Conference on (pp. 1-8). IEEE.
12. Swan, M., 2012. Sensor mania! The internet of things, wearable computing, objective metrics, and the quantified self 2.0. Journal of Sensor and Actuator Networks, 1(3), pp.217-253.
13. Uckelmann, D., Harrison, M. and Michahelles, F., 2011. An architectural approach towards the future internet of things. In architecting the internet of things (pp. 1-24). Springer, Berlin, Heidelberg.
14. Yu, R. and Watteyne, T., 2013. Reliable, Low Power Wireless Sensor Networks for the Internet of Things: Making Wireless Sensors as Accessible as Web Servers. Linear Technology.