***An Industry Oriented major project***

***on***

**IOT BASED WEATHER MONITORING USING NODEMCU**

**Submitted in partial fulfillment for the award of the degree of**

## BACHELOR OF TECHNOLOGY

**In**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

**K. ELENDRA CHARY - 19Q91A0430**

**MD. RIJWAN - 19Q91A0437**

**CH. AVINASH - 19Q91A0417**

**B. SAIDEEP - 20Q95A0405**

**Under the Esteemed Guidance of**

**Mr.M.SHIVA KUMAR**

**Associate Professor**

**Dept. of Electronics And Communication Engineering, MRCE**



## DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

**MALLA REDDY COLLEGE OF ENGINEERING**

**(**Approved by AICTE- Permanently Affiliated to JNTU Hyderabad) Accredited by NBA & NAAC, Recognized under section 2(f) &12(B) of UGC New Delhi: ISO 9001:2015 certified Institution

Maisammaguda, Dhullapally (Post via Kompally), Secunderabad- 500100

## 2022- 2023

**MALLA REDDY COLLEGE OF ENGINEERING**

**(**MALLA REDDY GROUP OF INSTITUTIONS)

**(**Approved by AICTE- Permanently Affiliated to JNTU Hyderabad) Accredited by NBA & NAAC, Recognized under section 2(f) &12(B) of UGC New Delhi: ISO 9001:2015 certified Institution

Maisammaguda, Dhulapally (Post via Kompally), Secunderabad- 500100



**CERTIFICATE**

This is to certify that the mini project report on “ **IOT BASED WEATHER MONITORING USING NODEMCU”** is successfully done by the following students of Department of Electronics And Communication Engineering of our college in partial fulfillment of the requirement for the award of B.Tech degree in the year 2019- 23.The results embodied in this report have not been submitted to any other University for the award of any diploma or degree.

**K. ELENDRA CHARY - 19Q91A0430**

**MD. RIJWAN - 19Q91A0437**

**CH. AVINASH - 19Q91A0417**

**B. SAIDEEP - 20Q95A0405**

**INTERNAL GUIDE HOD PRINCIPAL**

**Mr. M.Shiva kumar Mr M. Shiva Kumar Dr M. Sreedhar Reddy**

**Associate Professor Associate Professor Professor**

Submitted for the viva voice examination held on

**INTERNAL EXAMINAR EXTERNAL EXAMINAR**

**ACKNOWLEDGEMENT**

First and foremost, we would like to express our immense gratitude towards our institution Malla Reddy College of Engineering, which helped us to attain profound technical skills in the field of Electronics & Communication Engineering, there by fulfilling our most cherished goal.

We are pleased to thank **Sri Ch. Malla Reddy,** our Founder, Chairman **MRGI**, **Sri Ch. Mahender Reddy**, Secretary, **MRGI** for providing this opportunity and support throughout the course.

It gives us immense pleasure to acknowledge the perennial inspiration of **Dr.M.Sreedhar Reddy** our beloved principal for his kind co-operation and encouragement in bringing out this task.

We would like to thank **Dr. T. V. Reddy** our vice principal **Mr. M. Shiva Kumar** HOD, ECE Department for their inspiration adroit guidance and constructive criticism for successful completion of our degree

We convey our gratitude to **Mr. P.BALA SURESH REDDY** Assistant Professor, our project coordinator for their valuable cordinaton and guidelines. We would like to thank **Mr.M.SHIVA KUMAR** Associate Professor our internal guide, for his valuable suggestions and guidance during the exhibition and completion of this Finally, we avail this opportunity to express our deep gratitude to all staff who have contribute their valuable assistance and support making our project success.

**DECLARATION**

We hereby declare that industry oriented mini project report entitled **“ IOT BASED WEATHER MONITORING USING NODEMCU”** is a genuine project work carried out by us, in B. Tech (Electronics and Communication Engineering, Malla Reddy College of Engineering, Kompally, Hyderabad) degree course of Jawaharlal Nehru Technological University, Hyderabad and has not been submitted to any other courses or university for award of any degree by us.

## Signature of the Student

**K. ELENDRA CHARY - 19Q91A0430**

**MD. RIJWAN - 19Q91A0437**

**CH. AVINASH - 19Q91A0417**

**B. SAIDEEP - 20Q95A0405**

Date: Place:

**ABSTRACT**

The system proposed in this paper is an advanced solution for monitoring the weather conditions at a particular place and make the information visible anywhere in the world. The technology behind this is Internet of Things (IOT), which is an advanced and efficient solution for connecting the things to the internet and to connect the entire world of things in a network. Here things might be whatever like electronic gadgets, sensors and automotive electronic equipment. The system deals with monitoring and controlling the environmental conditions like temperature, relative humidity and CO level with sensors and sends the information to the web page and then plot the sensor data as graphical statistics. The data updated from the implemented system can be accessible in the internet from anywhere in the world.

It is a system that involves in acquiring weather and environment data using advanced electronic

sensors and sending them to a web server via internet for real time weather monitoring and storage of data for future analysis and study.

**CONTENTS**

**CHAPTER 1**

#### **1 INTRODUCTION**

* 1. Existing system and proposed system 06
  2. Block diagram 07
  3. Hardware tools 08

# [CHAPTER 2](file:///C:\Users\shrav\AppData\Local\Microsoft\Windows\INetCache\IE\60F2S7RL\batch-10_documentation%5b1%5d.docx#_bookmark2)

#### **2 Literature Review**

#### 2.1Basic Introduction Of IOT 09

# CHAPTER 3

#### **3**. **TECHNOLOGY USED**

**Hardware components**

3.1Node MCU 10

3.2 Modules 11

3.3 Digital 11

3.4 Analog 12

3.5 Output and input signals 14

**Software Components**

3.6 Software tips 19

3.7 Arduino Ide 20

3.8 Things to remember about digital 23

3.9 Things to remember about analog 24

**CHAPTER 4**

**4 HARDWARE DESCRPITION**

4.1 Node MCU 28

4.1.1Specifications30

4.1.2 Pin diagram 31

4.2 DHT11.SENSOR 32

4.2.1 working 31

4.3 Humidty sensor 33

4.4 rain sensor 38

4.5.Buzzer 39

4.5.1 Electromechanical 36

4.5.2 Piezoelectric 38

4.5.3 Mechanical 36

4.6LCD 42

**CHAPTER 5**

**5 SOFTWARE IMPLEMENTATION**

5.1Think Thing speak 50

5.2 Arduino software Ide 53

5.2.1 Definition 54

5.3 Embedded C 55

5.3.1 Basics of Embedded C 55

5.3.2 Programming embedded systems 56

5.3.3 Factors for selecting the programming language 57

**CHAPTER 6**

**6 SOURCE CODE AND RESULT**

6.1Source code 58

6.2 Result 61

**CHAPTER 7**

**7 CONCLUSION**

7.1Conclusion 62

7.2 Reference 63

**LIST OF FIGURES**

1.1 Block diagram 03

2.1 Conceptual Layered Iot Architecture 07

4.1 Node MCU 26

4.2 Pin diagram 29

4.3 DHT11.sensor 30

4.4 Humidity sensor 33

4.6 Buzzer pin configuration 35

4.7 Piezoelectric Buzzer 36

4.8 Water Level indicator Buzzer 39

4.9 I2C LCD 42

4.10 LCD Configuration 43

5.1 Input Entering window 47

5.2 Output Showing window 47

# CHAPTER – 1

# INTRODUCTION

Climate plays an important role in human life the unprecedented growth of industries and vehicular traffic have seriously affected the purity of clean air and environment [1]. Satellite weather report system gives condition of present which does not give the exact condition of the particular place. The building sector offers a great potential for the energy savings, where it is necessary to have accurate weather data in the exact location where the building is being built in order to improve the calibration of energy simulation programs [2]. By develop a controlling local weather reporting system with ESP32 and Wemos D1 mini microcontroller can minimize the error in weather forecast system at exact location. A precision agriculture and farming can be defined as the art and science of using technology to improve crop production [3]. Even though water is a scarce resource, overall 50% of water is wasted in agriculture due to the improper scheduling of irrigation [4]. In this context, the realtime monitoring of water usage in the fields can prevent misuse of water [4]. Use of technology in the field of agriculture plays important role in increasing the production as well as in reducing the extra man power efforts, some of the researches tried for betterment of farmers and provides the systems that use technologies which are helpful for increasing the agriculture yield [5]. Difficulty to monitor weather parameters through offline system such as agriculture zone during certain hazardous envy and critical situations where the people need to check manually the weather condition at the places and it will take time unless it is online system. In the evolving generation of wireless technology, the concept of smart cities and IOT has given a new remark in the world. One such remark leads towards the online smart weather station system [6]. The weather parameters should be able displaying, analysing and monitoring system using ThingSpeak that connect user with internet that visible anywhere in the world. To analyse and monitoring system using ThingSpeak that connect user with internet that visible anywhere in the world. Internet of Things (IoT) is playing a leading role in providing solutions to many applications with the support of software, internet and embedded systems [7]. There is a necessity in security or alarming system that give warning and alerting when there is a bad condition at the place. The existing technologies are developed using microcontrollers like Arduino, Node MCU etc. and ARM processors like Raspberry Pi [7]. So, machine learning techniques achieved better performance than traditional statistical methods in learning without being expressly customized [8]. Data measured by the stations could be used for various purposes, such as: air quality management to reduce pollutant gases in the local atmosphere and climate monitoring for a better yields of the region crops [9]. To making alarming system this project has uses the Blynk apps to send a notification when the weather at bad condition to the user and also the buzzer sound at the weather station to put the user at home alert about the situation outside.

limatic change and environmental monitoring have

received much attention recently. Man wants to stay updated

about the latest weather conditions of any place like a

college campus or any other particular building. Since the

world is changing so fast so there should be the weather

stations. Here in this paper we present a weather station that

is very helpful for any places. This weather station is based

on IOT (internet of things). It is equipped with

environmental sensors used for measurements at any

particular place and report them in real time on cloud. To

accomplish this we used Arduino Uno and different

environmental sensors like DHT11, soil moisture sensor and

rain drop sensor .The sensors constantly sense the weather

parameters and keeps on transmitting it to the online web

server over a wifi connection. The weather parameters are

uploaded on the cloud and then provides the live reporting

of weather information. This paper also focuses on the IOT

application in the new generation of environmental

information and provides a new paradigm for environmental

monitoring in future. The system has been development

particularly in the view of building smart city by giving the

weather update of any particular place like a particular office

or room

limatic change and environmental monitoring have

received much attention recently. Man wants to stay updated

about the latest weather conditions of any place like a

college campus or any other particular building. Since the

world is changing so fast so there should be the weather

stations. Here in this paper we present a weather station that

is very helpful for any places. This weather station is based

on IOT (internet of things). It is equipped with

environmental sensors used for measurements at any

particular place and report them in real time on cloud. To

accomplish this we used Arduino Uno and different

environmental sensors like DHT11, soil moisture sensor and

rain drop sensor .The sensors constantly sense the weather

parameters and keeps on transmitting it to the online web

server over a wifi connection. The weather parameters are

uploaded on the cloud and then provides the live reporting

of weather information. This paper also focuses on the IOT

application in the new generation of environmental

information and provides a new paradigm for environmental

monitoring in future. The system has been development

particularly in the view of building smart city by giving the

weather update of any particular place like a particular office

or room

* + **EXISTING SYSTEM**
* WSN deals with more no of inexpensive wireless sensor, it is used for collecting, storing and processing the input data and send it to server.
* WSN gateway is used to get the data from a WSN nodes by anywhere and anytime, it is used to receive,
* prepare and display in your measurement data. We used arduino uno without wifi module.

**1.2 PROPOSED SYSTEM**

* .Measuring Environmental Parameters (Temperature, pressure, humidity etc…) like gas sensors, moisture sensora etc..
* Sensors are used with different range and different uses.
* Sensor Data Acquisition and Decision Making.

Intelligent Environment in this portion collected information will be stored in cloud and it sent do end user.

* it is inbuilt wifi module (ESP8266) .

**1.3 BLOCK DIAGRAM**

DHT11

NODE MCU

BUZZER

RAIN SENSOR

I2C LCD

**Fig 1.1- Block Diagram**

**1.4 HARDWARE TOOLS**

* NODE MCU
* Rain sensor
* DHT11 sensor
* I2C LCD
* buzzer

**1.5 SOFTWARE TOOLS**

* + Arduino IDE
  + IOT

**CHAPTER -2**

**LITERATURE REVIEW**

In today’s world many pollution monitoring systems are designed by different environmental parameters. Existing system model is presented IOT based Weather monitoring and reporting system where you can collect, process, analyze, and present your measured data on web server. Wireless sensor network management model consists of end device, router, gateway node and management monitoring center. End device is responsible for collecting wireless sensor network data, and sending them to parent node, then data are sent to gateway node from parent node directly or by router. After receiving the data from wireless sensor network, gateway node extracts data after analyzing and packaging them into Ethernet format data, sends them to the server. Less formally, any device that runs server software could be considered a server as well. Servers are used to manage network resources. The services or information provided through the Internet that are connected through LAN and made available for users via smart phones, web browser or other web browser devices to make the system more intelligent, adaptable and efficient.

# 2.1 BASIC INTRODUCTION OF IOT

IoT is short for Internet of Things. As the name implies, the Internet of Things is the Internet of things connected. There are two meanings here. First , the core and foundation of the Internet of Things is still the Internet, which is an extension and expansion network based on the Internet; second, its user-side extended and expanded information exchange and communication between objects. The Internet of Things is the third wave of the development of the world’s information industry after computers and the Internet.

Until now, there isn’t an accurate and recognized definition of the Internet of Things. This is mainly due to: First, the theoretical system of the Internet of Things has not been fully established, and it has not been recognized deep enough. Second, because the Internet of Things is closely related to the Internet, mobile communication networks, and sensor networks, researchers in these fields have different starting points and end points for thinking about the Internet of Things, and there is no consensus in the short term.

According to the industry information we have observed, the Internet of Things is an information carrier based on the Internet and traditional telecommunications networks, allowing all independently addressable common physical objects interconnected. It has three characteristics: common object equipment, autonomous terminal interconnection, and universal service intelligence.



**Fig 2.1 Conceptual Layered Iot Architecture**

**CHAPTER -3**

**TECHNOLOGY USED**

**3.1 NODE MCU:**

Microcontroller board based on the atmega328p (datasheet). Ithas14digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button as shown in Fig.2.

It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst-case scenario you can replace the chip for a few dollars and start over again. “Uno" means one in Italian and was chosen to mark the release of Arduino Software (IDE) 1.0. The Uno board and version 1.0 of Arduino Software (IDE) were the reference versions of Arduino, now evolved to newer releases.

The Uno board is the first in a series of USB Arduino boards, and the reference model for the Arduino platform; for an extensive list of current, past or outdated boards see the Arduino index. The Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. Clock speed of Arduino UNO is 16MHz.The Arduino/Genuine Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.

**3.2 MODULES:**

**ARDUINO UNO:**

setup (): A capacity present in each Arduino sketch. Run once before the loop () function. Frequently used to set pin mode to info or yield. The setup () function resembles like:

void setup( ){

//code goes here

}

input: A pin mode that admissions data.

output: A pin mode that sends data.

HIGH: Electrical sign present (5V for Uno). Additionally, ON or True in Boolean rationale.

LOW: No electrical sign present (0V). Additionally OFF or False in Boolean rationale.

DigitalRead: Get a HIGH or LOW perusing from a pin previously proclaimed as an info. 8

DigitalWrite: Assign a HIGH or LOW an incentive to a pin previously pronounced as anoutput .

AnalogRead: Get an incentive between or including 0 (LOW) and 1023 (HIGH). This permits you to get readings from simple sensors or interfaces that have multiple states.

AnalogWrite: Assign an incentive between or including 0 (LOW) and 255 (HIGH). This allows you to set output to a PWM esteem rather than simply HIGH or LOW.

PWM: Stands for Pulse-Width Modulation, a strategy for copying a simple sign through an advanced pin. An incentive between or including 0 and 255. Utilized with analogWrite.

**3.3 DIGITAL**

An electronic sign communicated as paired code that can be either the presence or nonattendance of current, high and low voltages or short heartbeats at a specific recurrence. People see the world in simple, however robots, PCs and circuits utilize Digital. A computerized signal is a sign that has just two states. These states can shift contingent upon the sign, however basically characterized the states are ON or OFF, never in the middle. In the realm of Arduino, Digital signs are utilized for everything except for Analog Input.

Contingent upon the voltage of the Arduino the ON or HIGH of the Digital sign will be equivalent to the framework voltage, while the OFF or LOW sign will consistently rise to 0V. This is an extravagant method of saying that on a 5V Arduino the HIGH signals will be a little under 5V and on a 3.3V Arduino the HIGH signals will be somewhat under 3.3V. To get or impart Digital signs the Arduino utilizes Digital pins # 0 - # 13. You may likewise arrangement your Analog In pins to go about as Digital pins. To set up Analog In pins as Digital pins utilize the order:

pinMode(pinNumber, esteem);

where pinNumber is an Analog pin (A0 – A5) and worth is either INPUT or OUTPUT. To arrangement Digital pins utilize a similar order yet reference a Digital pin for pinNumber rather than an Analog In pin. Computerized pins default as info, so truly you just need to set them to OUTPUT in pinMode. To peruse these pins utilize the order:

digitalRead(pinNumber);

**Things To Remember About Digital:**

* Digital Input/Output utilizes the Digital pins, however Analog In pins can be utilized as Digital
* To get a Digital sign use: digitalRead(pinNumber);
* To impart a Digital sign use: digitalWrite(pinNumber, esteem);
* Digital Input and Output are in every case either HIGH or LOW.

The entirety of the electrical signs that the Arduino works with are either Analog or Digital. It

is critical to comprehend the distinction between these two kinds of sign and how to control

the data these signs speak to.

**3.4 ANALOG:**

People see the world in simple. All that we see and hear is a ceaseless transmission of data to our faculties. The temperatures we see are rarely 100% hot or 100% cold, they are continually changing between our scopes of satisfactory temperatures. (What's more, in the event that they are out of our scope of adequate temperatures, at that point what are we doing there?) This ceaseless stream is the thing that characterizes simple information. Advanced data, the correlative idea to Analog, gauges simple information utilizing just ones and zeros. In the realm of Arduino an Analog sign is essentially a sign that can be HIGH (on), LOW (off) or anything in the middle of these two states.

This implies an Analog sign has a voltage esteem that can be anything somewhere in the range of 0V and 5V (except if you play with the Analog Reference pin). Simple permits you to send yield or get contribution about gadgets that run at rates just as here and there. The Arduino does this by inspecting the voltage signal shipped off these pins and contrasting it with a voltage reference signal (5V). Contingent upon the voltage of the Analog sign when contrasted with the Analog Reference signal the Arduino then doles out a mathematical incentive to the sign somewhere close to 0 (0%) and 1023 (100%).

The computerized arrangement of the Arduino would then be able to utilize this number in figurings and portrayals. To get Analog Input the Arduino utilizes Analog pins # 0 - # 5. These pins are intended for use with segments that yield Analog data and can be utilized for Analog Input. There is no arrangement fundamental, and to peruse them utilize the order:

analogRead(pinNumber);

where pinNumber is the Analog In pin to which the Analog part is associated. The analogRead order will restore a number including or somewhere in the range of 0 and 1023. The Arduino additionally has the ability to yield a computerized signal that goes about as an Analog sign, this sign is called Pulse Width Modulation (PWM). Advanced Pins # 3, # 5, # 6,

# 9, # 10 and #11 have PWM capacities. To yield a PWM signal utilize the order:

analogWrite(pinNumber, esteem);

where pinNumber is a Digital Pin with PWM abilities and worth is a number between 0 (0%)

and 255 (100%). On the Arduino UNO PWM pins are meant by a ~ sign. For more data on

PWM see the PWM worksheets or S.I.K. circuit 12.

**Things To Remember About Analog:**

* Analog Input utilizes the Analog In pins, Analog Output utilizes the PWM pins
* To get an Analog sign use: analogRead(pinNumber);
* To impart a PWM sign use: analogWrite(pinNumber, esteem);
* Analog Input esteems range from 0 to 1023 (1024 qualities since it utilizes 10 pieces, 210)
* PWM Output esteems range from 0 to 255 (256 qualities since it utilizes 8 pieces, 28)

The entirety of the electrical signs that the Arduino works with are either info or yield. It is critical o comprehend the distinction between these two sorts of sign and how to control the data these signs speak to.

**3.5 OUTPUT SIGNALS:**

13Analog input to the Arduino pins is consistently Digital, anyway there are two unique kinds of Digital Output; ordinary Digital Output and Pulse Width Modulation Output (PWM). Yield is just conceivable with Digital pins # 0 - # 13.

The Digital pins are preset as Output pins, so except if the pin was utilized as an Input in a similar sketch, there is no motivation to utilize the pinMode order to set the pin as an Output. Should a circumstance emerge where it is important to reset a Digital pin to Output from Input

utilize the order:

pinMode(pinNumber, OUTPUT);

where pinNumber is the Digital pin number set as Output. To impart a Digital Output sign

utilize the order:

digitalWrite(pinNumber, esteem);

where pinNumber is the Digital pin that is yielding the sign and worth is the sign. While

yielding a Digital sign worth can be either HIGH (On) or LOW (Off).

analogWrite(pinNumber, esteem);

where pinNumber is a Digital Pin with PWM abilities and worth is a number between 0 (0%)

and 255 (100%).

For more data on PWM see the PWM worksheets or S.I.K. circuit 12.

Yield can be shipped off a wide range of gadgets, however it is dependent upon the client to

sort out which sort of Output signal is required, connect the equipment and afterward type the

right code to appropriately utilize these signs.

**Things To Remember About Output:**

* Output is consistently Digital.
* There are two sorts of Output: ordinary Digital or PWM (Pulse Width Modulation).

To impart an Output sign use analogWrite(pinNumber, esteem); (for simple) or digitalWrite(pinNumber, esteem); (for computerized)

* Output pin mode is set utilizing the pinMode order: pinMode(pinNumber, OUTPUT);
* Regular Digital Output is in every case either HIGH or LOW.
* PWM Output shifts from 0 to 255.

The entirety of the electrical signs that the Arduino works with are either information or yield.

It is critical to comprehend the contrast between these two sorts of sign and how to control the

data these signs speak to.

**INPUT SIGNALS :**

Simple Input enters your Arduino through the Analog In pins # 0 - # 5. These signs begin from simple sensors and interface gadgets. These simple sensors and gadgets use voltage levels to impart their data rather than a straightforward yes (HIGH) or no (LOW). Therefore you can't utilize a computerized pin as an information pin for these gadgets. Simple Input pins are utilized uniquely for getting Analog signs. It is simply conceivable to peruse the Analog Input sticks so there is no order vital in the arrangement( ) capacity to set up these pins for input. To peruse the Analog Input pins utilize the order:

analogRead(pinNumber);

where pinNumber is the Analog Input pin number. This capacity will restore an Analog Input perusing somewhere in the range of 0 and 1023. A perusing of zero relates to 0 Volts and a perusing of 1023 compares to 5 Volts. These voltage esteems are radiated by the simple sensors and interfaces. On the off chance that you have an Analog Input that could surpass Vcc + 5V you may change the voltage that 1023 relates to by utilizing the Aref pin. This pin sets the greatest voltage boundary your Analog Input pins can peruse. The Aref pin's preset worth is 5V.

Computerized Input can enter your Arduino through any of the Digital Pins # 0 - # 13. Advanced Input signals are either HIGH (On, 5V) or LOW (Off, 0V). Since the Digital pins can be utilized either as info or yield you should set up the Arduino to utilize these pins as contributions to your arrangement( )work. To do this sort the order:

pinMode(pinNumber, INPUT);

inside the wavy sections of the arrangement( ) work where pinNumber is the Digital pin number you wish to announce as an information. You can change the pinMode insider savvy( )work on the off chance that you need to switch a pin to and from among info and yield, however it is typically set in the arrangement( )capacity and left immaculate tuned in( )work. To peruse the Digital pins set as data sources utilize the order:

digitalRead(pinNumber);

where pinNumber is the Digital Input pin number.

Info can emerge out of a wide range of gadgets, however every gadget's sign will be either Analog or Digital, it is dependent upon the client to sort out which sort of information is required, connect the equipment and afterward type the right code to appropriately utilize these signs.

2x3 pin header Used to program Atmega with bootloader. The number 1 on the two sides of the board shows link pin1 position. Used to transfer draws on Atmega ICs without bootloader (accessible just in Arduino IDE variants 0011 and 0012). 3 pins jumper When in position 2-3, this jumper empowers sequential association (through X1 connector) to/from PC/gadgets. Utilize this as default position. When in position 1-2, it handicap

sequential correspondence, and empowers outer draw down resistors on pin0 (RX) and pin1 (TX).

Utilize this just to forestall commotion on RX (that appears to be approaching information to Atmega), that occasionally makes sketch not beginning. While eliminating this jumper, sequential correspondence is impaired, and pin0 and pin1 fill in as a typical (gliding) advanced pin. Valuable when more advanced pins are required, yet just when sequential correspondence isn't essential. Outside draw down/pull-up resistor is required.

**JP4**

2 pins jumper When in position 1-2, this jumper empowers auto reset include, helpful while transferring a sketch to Arduino, resetting Atmega consequently. It makes superfluous to squeeze reset button (S1) when transferring portrays. Be certain that PC COM Port speed is set to 19200bps in any case auto reset won't work appropriately. Whenever eliminated, cripples auto reset include. Extremely valuable to forestall undesired Atmega reset when

utilizing outlines that needs sequential correspondence.

Auto reset works with DTR heartbeat on sequential pin4. Once in a while Arduino faculties a DTR beat while associating X1 (chronic connector) and a few virtual products sends a DTR beat when it begins or when it shuts, that makes Atmega reset when not wanted. S1 Material catch This catch resets Atmega, to restart transferred sketch or to plan Arduino to get a sketch through chronic connector (when auto reset isn't dynamic).

**LEDS**

Characteristic leds POWER drove Turns on when Arduino is fueled through DC1, +9v. pin or +5v. pin. RX drove Blinks while accepting information from PC/gadget through sequential association. TX drove Blinks when sending information to PC/gadget through sequential association. L drove This drove is associated with computerized pin13 with a current limiter resistor (that doesn't influence pin13). Helpful to test portrays. It isn't unexpected to squint while bootloading as well.

**POWER PINOUT:**

6 pin header

RST pin

Makes Atmega reset when associated with GND. Helpful for Shield Boards, or to associate

outside reset.

NC pin

This pin isn't associated in Arduino S3v3. Arduino Diecimila has a 3.3 volts pin similarly

situated.

+9v. pin

At the point when Arduino DC1 is fueled (with battery or DC connector), this pin is utilized as Vout, with a similar voltage provided on DC1 (see DC1), less 0,7 volts. The all out provided current relies upon outer force source limit When Arduino DC1 isn't controlled, +9v. pin can be utilized as Vin, interfacing it to an outside controlled force source (+7 to +20 volts) and associating 0v. pin to outside force source GND. For this situation,

+5v. pin can be utilized as Vout, providing +5 volts. +5v.

pin When Arduino DC1 is controlled (with battery or DC connector), +5v. pin supplies +5 volts as a Vout pin. The absolute provided current relies upon Voltage Regulator (7805 supplies up to 1A). This applies just to +5v. pin: Atmega in/out pins just supplies max. 40mA on each pin. At the point when Arduino DC1 isn't controlled, this pin can be utilized as Vin, associating it to a managed +5v. what's more, associating 0v. pin to control source GND. For this situation, +9v. pin is dormant. 0v. pin

(GND) Two 0v. sticks between +5v. what's more, +9v.

/One

0v. pin

adjacent to AREF pin. At the point when Arduino DC1 is controlled, 0v. pin supplies 0 volts reference (GND) for +5v. pin and +9v. pin. At the point when DC1 isn't controlled, and Arduino is fueled through +5v. pin or +9v. pin, 0v. pin must be utilized as GND reference,

interfacing it to the outside force source GND.

GND pin

see 0v. pin (GND).

AREF pin The AREF can be set to AVcc (default), inward 2.56 volts (Atmega8), inner 1.1 volts (Atmega168), or outside AREF. If there should arise an occurrence of AVcc or interior AREF, AREF pin can be utilized to connect na outside capacitor to decouple the sign, for better clamor execution. If there should be an occurrence of outer AREF, AREF pin is utilized to append the outside reference voltage. Recollect that it is important to change de wires (wiring.c document), and re-transfer sketch, prior to interfacing outside voltage to AREF .

**3.6 SOFTWARE TIPS:**

While bootloadingna Atmega8 chip with Arduino 0010, there is an order (- i800) that makes bootloader defer 10 minutes. Thus, in the event that you need to utilize bootloader, use order line rather than IDE, eliminating "– i800" order and adding "– F" order, or use Arduino 0007 IDE. To transfer draws Arduino 0010 turns out great.

**ARDUINO S3v3 NEW FEATURES:**

* Full viable with Shield Boards (Version 2 is the main Arduino Board not viable with Shield Boards as a result of ICSP header wrong position, and tall parts);
* AVcc LP channel to lessen commotion level on ADC;
* auto reset include;
* auto reset empower/impair jumper, to dodge not wanted reseting;
* arduinoDiecimila viable reset pin;
* pin13 locally available drove, with current limiter resistor;
* TX and RX locally available leds;
* power drove with suitable current limiter resistor (less 20mA of comsumption);

jumper to impair sequential correspondence and to empower RX outer draw down resistor, to evade "RX skimming blunder". This element permits to utilize computerized pin0 and pin1 as an ordinary pin, when sequential correspondence isn't required

* all comparative segments (diodes, semiconductors, leds, capacitors) has a similar board direction (to commits simpler to mount with less errors);
* no wires between cushions, more space between wires, bigger wires, bigger cushions (better for drawing, binding and penetrating, with no shortcircuits, patching extensions or open wires in erosion);
* just 3 wire spans; electrolitic capacitor (in sequential to TTL circuit) changed to bipolar sort (to keep away from rearranged voltage issue when sequential link isn't associated);
* All jumpers are correct point type, to permit Shield Boards use.

**3.7 ARDUINO IDE:**

Download Arduino Integrated Design Environment (IDE) here (Most late form: 1.6.5):

https://www.arduino.cc/en/Main/Software

This is the Arduino IDE whenever it's been opened. It opens into a clear sketch where you can begin programming right away. To begin with, we ought to arrange the board and port settings to permit us to transfer code. Interface your Arduino board to the PC through the USB link.

**UPLOADING BLINK:**

One common procedure to check whether the board you're using is correctly founded is to upload the “Blink” sketch. This sketch is included with all Arduino IDE releases and might be accessed by the Filepull-down menu and visiting Examples, 01.Basics, then select Blink. Standard Arduino Boards include a surface-mounted LED labeled “L” or “LED” next to the “RX” and “TX” LEDs, that's connected to digital pin 13. This sketch will blink the LED at an everyday interval, and is a straightforward thanks to confirm if your board is ready up properly and you were successful in uploading code. Open the “Blink” sketch and press the “Upload” button within the upper-left corner to upload “Blink” to the board.

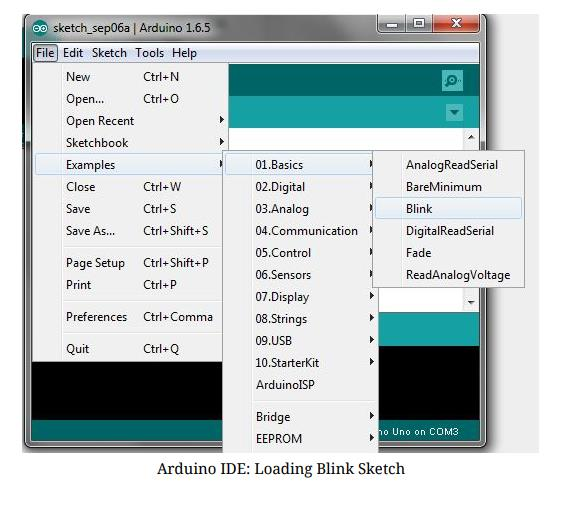
1.Download and introduce Arduino IDE (https://www.arduino.cc/en/Main/Software)

2. Plug in your Arduino Board

3. Select the correct board in the IDE (Tools>Boards>Arduino Uno)

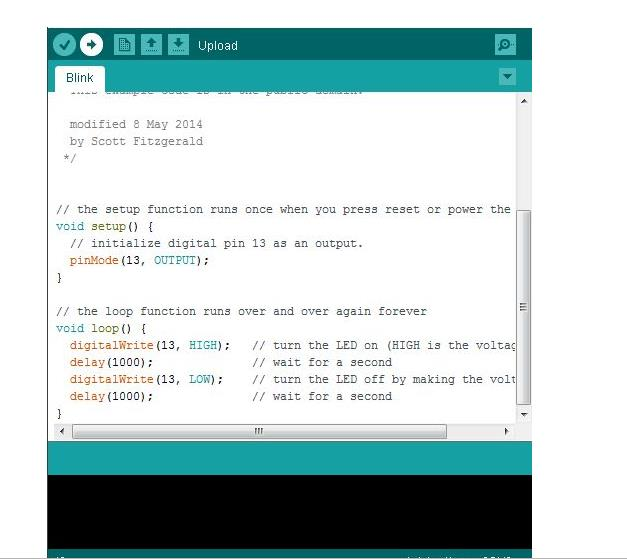
4. Select the correct COM port (Tools>Port>COMx (Arduino Uno))

5. Open the "Blink" sketch (File>Examples>Basics>01.Blink)



6. Press the Upload catch to transfer the program to the board

7. Affirm that your board is functioning true to form by noticing LED



Arduino has bunches of network backing and documentation. Your smartest option when running into unforeseen issues is to look through online for help. You should have the option to discover a discussion where somebody had a similar issue you are having, and somebody encouraged them fix it. In the event that you don't discover results, take a stab at altering your pursuit, or post on the Arduino discussions.

**3.8 ANDROID APPLICATION:**

Control your Arduino with voice commands using an Android smartphone. The App works by pressing the microphone button, then it will wait for you to say a command. The app will then display the word's that you've stated and will send data strings to the Arduino and arduino generates the PWM control signal for the motor driver and the speed and direction of the DC motor is controlled.

**DIGITAL:**

An electronic signal transmitted as binary code that can be either the presence or absence of current, high and low voltages or short pulses at a particular frequency. Humans perceive the world in analog, but robots, computers and circuits use Digital. A digital signal is a signal that has only two states. These states can vary depending on the signal, but simply defined the states are ON or OFF, never in between. In the world of Arduino, Digital signals are used for everything with the exception of Analog Input. Depending on the voltage of the Arduino the ON or HIGH of the Digital signal will be equal to the system voltage, while the OFF or LOW signal will always equal 0V. This is a fancy way of saying that on a 5V Arduino the HIGH signals will be a little under 5V and on a 3.3V Arduino the HIGH signals will be a little under 3.3V. To receive or send Digital signals the Arduino uses Digital pins # 0 - # 13. You may also setup your Analog In pins to act as Digital pins. To set up Analog In pins as Digital pins use the command:

pinMode(pinNumber, value);

where pinNumber is an Analog pin (A0 – A5) and value is either INPUT or OUTPUT. To setup Digital pins use the same command but reference a Digital pin for pinNumber instead of an Analog In pin. Digital pins default as input, so really you only need to set them to OUTPUT in pinMode. To read these pins use the command:

digitalRead(pinNumber);

where pinNumber is the Digital pin to which the Digital component is connected. The digitalRead command will return either a HIGH or a LOW signal. To send a Digital signal to a pin use the command:

digitalWrite(pinNumber, value);

where pinNumber is the number of the pin sending the signal and value is either HIGH or

LOW.

The Arduino also has the capability to output a Digital signal that acts as an Analog signal, this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, # 5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

analogWrite(pinNumber, value);

where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

**3.8.1 THINGS TO REMEMBER ABOUT DIGITAL:**

* Digital Input/Output uses the Digital pins, but Analog In pins can be used as Digital
* To receive a Digital signal use: *digitalRead(pinNumber);*
* To send a Digital signal use: *digitalWrite(pinNumber, value);*
* Digital Input and Output are always either HIGH or LOW
* All of the electrical signals that the Arduino works with are either Analog or Digital.
* It is extremely important to understand the difference between these two types of signal and
* how to manipulate the information these signals represent.

**ANALOG:**

Humans perceive the world in analog. Everything we see and hear is a continuous transmission of information to our senses. The temperatures we perceive are never 100% hot or 100% cold, they are constantly changing between our ranges of acceptable temperatures. (And if they are out of our range of acceptable temperatures then what are we doing there?) This continuous stream is what defines analog data. Digital information, the complementary concept to Analog, estimates analog data using only ones and zeros.

In the world of Arduino an Analog signal is simply a signal that can be HIGH (on), LOW (off) or anything in between these two states.This means an Analog signal has a voltage value that can be anything between 0V and 5V (unless you mess with the Analog Reference pin). Analog allows you to send output or receive input about devices that run at percentages as well as on and off. The Arduino does this by sampling the voltage signal sent to these pins and comparing it to a voltage reference signal (5V).

Depending on the voltage of the Analog signal when compared to the Analog Reference signal the Arduino then assigns a numerical value to the signal somewhere between 0 (0%) and 1023 (100%). The digital system of the Arduino can then use this number in calculations and sketches. To receive Analog Input the Arduino uses Analog pins # 0 - # 5. These pins are designed for use with components that output Analog information and can be used for Analog Input. There is no setup necessary, and to read them use the command:

analogRead(pinNumber);

where pinNumber is the Analog In pin to which the the Analog component is connected. The analogRead command will return a number including The Arduino also has the capability to output a digital signal that acts as an Analog signal, this signal is called Pulse Width Modulation (PWM). Digital Pins # 3, # 5, # 6, # 9, # 10 and #11 have PWM capabilities. To output a PWM signal use the command:

*analogWrite(pinNumber, value);*

where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). On the Arduino UNO PWM pins are signified by a ~ sign. For more information on PWM see the PWM worksheets or S.I.K. circuit 12.

**3.8.2 THINGS TO REMEMBER ABOUT ANALOG:**

* Analog Input uses the Analog In pins, Analog Output uses the PWM pins
* To receive an Analog signal use: analogRead(pinNumber);
* To send a PWM signal use: analogWrite(pinNumber, value);
* Analog Input values range from 0 to 1023 (1024 values because it uses 10 bits, 210)
* PWM Output values range from 0 to 255 (256 values because it uses 8 bits, 28)

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

**OUTPUT SIGNALS:**

Output to the Arduino pins is always Digital, however there are two different types of Digital Output; regular Digital Output and Pulse Width Modulation Output (PWM). Output is only possible with Digital pins # 0 - # 13. The Digital pins are preset as Output pins, so unless the pin was used as an Input in the same sketch, there is no reason to use the pinMode command to set the pin as an Output. Should a situation arise where it is necessary to reset a Digital pin to Output from Input use the command:

pinMode(pinNumber, OUTPUT);

where pinNumber is the Digital pin number set as Output. To send a Digital Output signal use the command:

digitalWrite(pinNumber, value);

where pinNumber is the Digital pin that is outputting the signal and value is the signal. When outputting a Digital signal value can be either HIGH (On) or LOW (Off).

*analogWrite(pinNumber, value);*

where pinNumber is a Digital Pin with PWM capabilities and value is a number between 0 (0%) and 255 (100%). For more information on PWM see the PWM worksheets or S.I.K. circuit 12. Output can be sent to many different devices, but it is up to the user to figure out which kind of Output signal is needed, hook up the hardware and then type the correct code to properly use these signals.

**3.8.3 THINGS TO REMEMBER ABOUT OUTPUT:**

* Output is always Digital
* There are two kinds of Output: regular Digital or PWM (Pulse Width Modulation)
* To send an Output signal use *analogWrite(pinNumber, value);* (for analog) or *digitalWrite(pinNumber, value);* (for digital)
* Output pin mode is set using the pinMode command: *pinMode(pinNumber, OUTPUT);*
* Regular Digital Output is always either HIGH or LOW
* PWM Output varies from 0 to 255

All of the electrical signals that the Arduino works with are either input or output. It is extremely important to understand the difference between these two types of signal and how to manipulate the information these signals represent.

**INPUT SIGNALS:**

Analog Input enters your Arduino through the Analog In pins # 0 - # 5. These signals originate from analog sensors and interface devices. These analog sensors and devices use voltage levels to communicate their information instead of a simple yes (HIGH) or no (LOW). For this reason you cannot use a digital pin as an input pin for these devices. Analog Input pins are used only for receiving Analog signals. It is only possible to read the Analog Input pins so there is no command necessary in the setup( ) function to prepare these pins for input. To read the Analog Input pins use the command:

*analogRead(pinNumber);*

where pinNumber is the Analog Input pin number. This function will return an Analog Input reading between 0 and 1023. A reading of zero corresponds to 0 Volts and a reading of 1023 corresponds to 5 Volts. These voltage values are emitted by the analog sensors and interfaces. If you have an Analog Input that could exceed Vcc + .5V you may change the voltage that 1023 corresponds to by using the Aref pin. This pin sets the maximum voltage parameter your Analog Input pins can read. The Aref pin's preset value is 5V. Digital Input can enter your Arduino through any of the Digital Pins # 0 - # 13. Digital Input signals are either HIGH (On, 5V) or LOW (Off, 0V).

# CHAPTER -4

# HARDWARE COMPONENTS

## 4.1 NODEMCU:

****

**Fig 4.1- NODEMCU**

**NodeMCU** is an open-source firmware for which open-source prototyping board designs are available. The name “**NodeMCU**” combines “node” and “MCU” (micro-controller unit). The term “**NodeMCU**” strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source. **Nodemcu ESP8266** and **Nodemcu ESP32** are becoming very popular and are almost used in more then 50% IoT based projects today.

The firmware uses the Lua scripting language. The firmware is based on the eLua project and built on the **Espressif Non-OS SDK for ESP8266**. It uses many open source projects, such as lua-cjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit [**ESP32**](https://www.electroniclinic.com/esp32-arduino-ide-board-manager-installation-espressif-esp32-wroom/) has also been implemented.

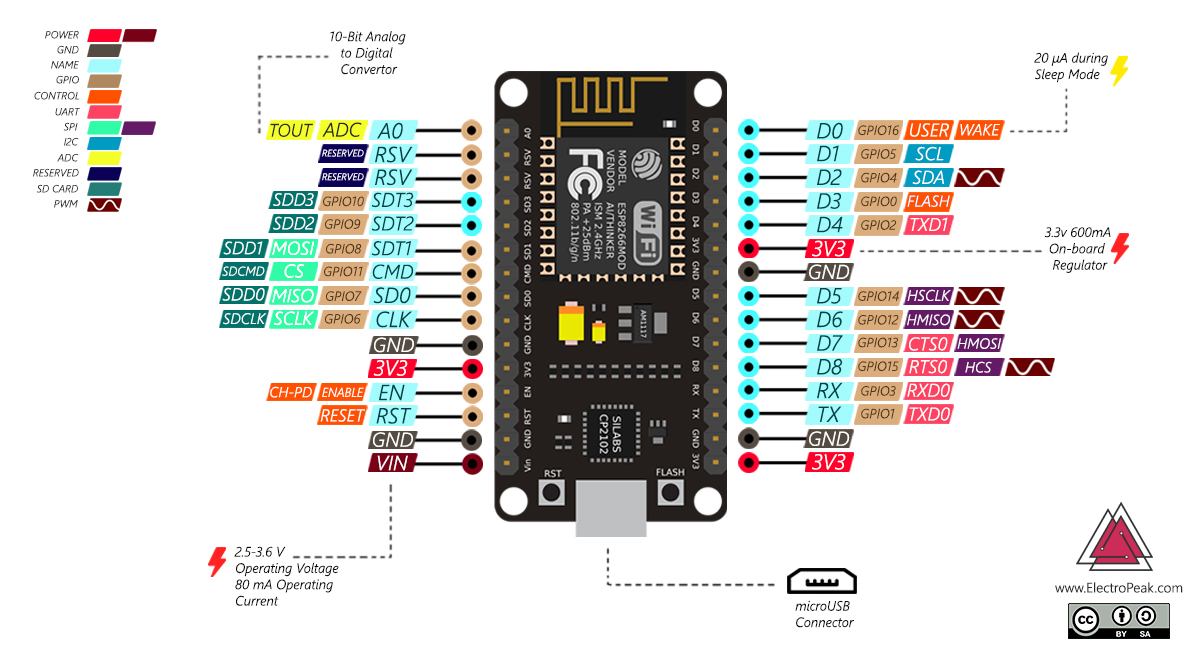
The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

### **4.1.1 NodeMCU Development Board Pinout Configuration:**

|  |  |  |
| --- | --- | --- |
| **Pin Category** | **Name** | **Description** |
| Power | Micro-USB, 3.3V, GND, Vin | **Micro-USB:** NodeMCU can be powered through the USB port    **3.3V:** Regulated 3.3V can be supplied to this pin to power the board    **GND:** Ground pins    **Vin:**External Power Supply |
| Control Pins | **EN, RST** | The pin and the button resets the microcontroller |
| Analog Pin | A0 | Used to measure analog voltage in the range of 0-3.3V |
| GPIO Pins | GPIO1 to GPIO16 | NodeMCU has 16 general purpose input-output pins on its board |
| SPI Pins | SD1, CMD, SD0, CLK | NodeMCU has four pins available for SPI communication. |
| UART Pins | TXD0, RXD0, TXD2, RXD2 | NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program. |
| I2C Pins |  | NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C. |

### **4.1.2 Nodemcu Esp8266 Specifications & Features:**

* Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 16
* Analog Input Pins (ADC): 1
* UARTs: 1
* SPIs: 1
* I2Cs: 1
* Flash Memory: 4 MB
* SRAM: 64 KB
* Clock Speed: 80 MHz
* USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
* PCB Antenna
* Small Sized module to fit smartly inside your IoT projects



**Fig 4.2- Pin Diagram**

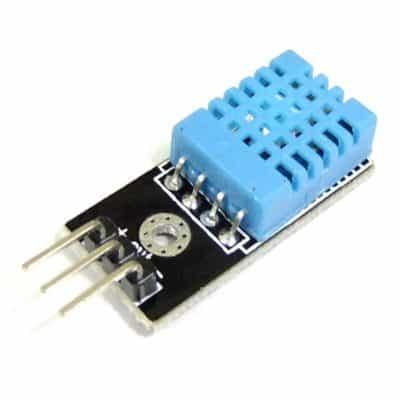
**4.2 DHT11 sensor:**

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature.  The humidity sensing [capacitor](https://www.elprocus.com/construction-of-capacitor-with-working/) has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second.  DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and micro-controller.



4.3 Fig.DHT11 sensor

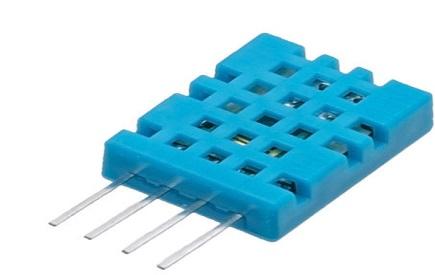
### **4.2.1 Working Principle**

DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature.  The humidity sensing [capacitor](https://www.elprocus.com/construction-of-capacitor-with-working/) has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers.

The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz .i.e. it gives one reading for every second.  DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and micro-controller.

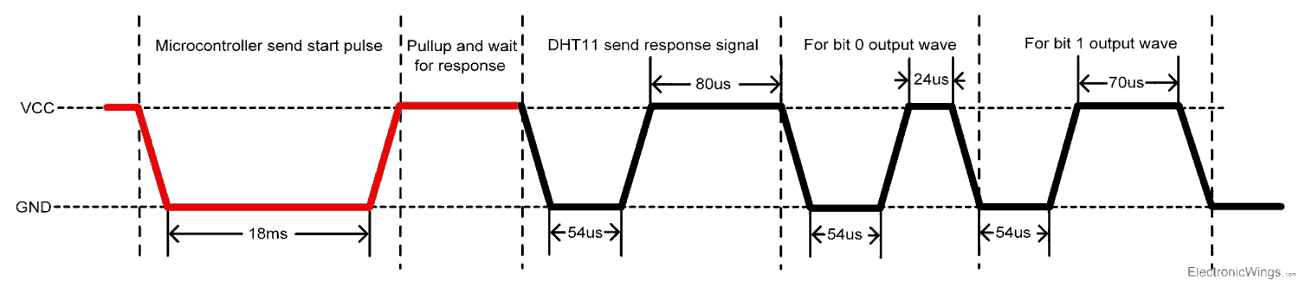


### **4.4 fig.Humidity sensor**

### **4.2.2**

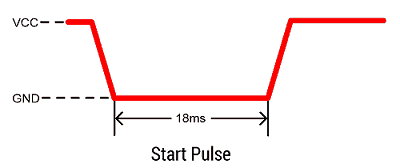
**HUMIDITY (DHT11):**

* DHT11 uses only one wire for communication. The voltage levels with certain time value defines the logic one or logic zero on this pin.
* The communication process is divided in three steps, first is to send request to DHT11 sensor then sensor will send response pulse and then it starts sending data of total 40 bits to the microcontroller.



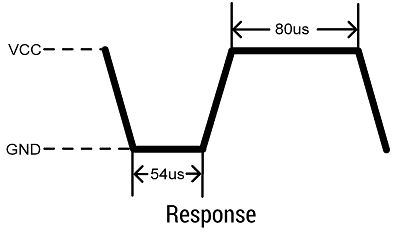
**Communication process**

Start pulse (Request)



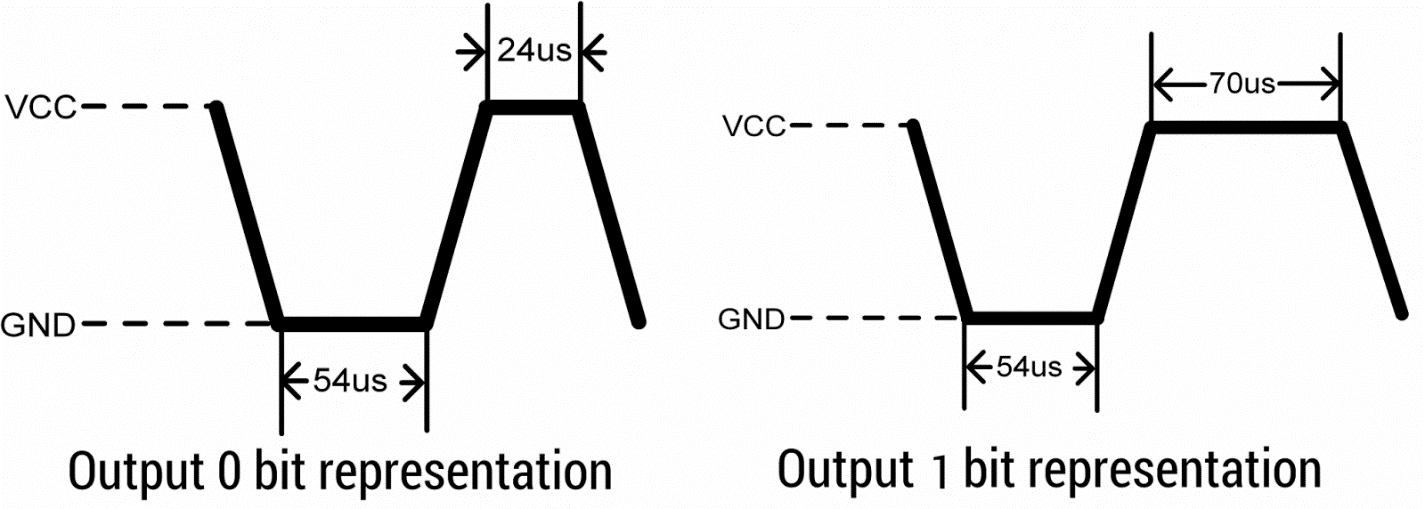
* To start communication with DHT11, first we should send the start pulse to the DHT11 sensor.
* To provide start pulse, pull down (low) the data pin minimum 18ms and then pull up, as shown in diag.

**Response**



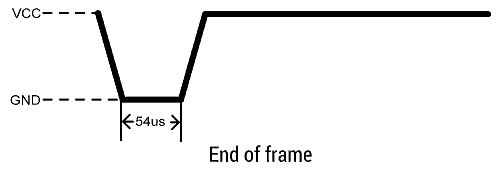
* After getting start pulse from, DHT11 sensor sends the response pulse which indicates that DHT11 received start pulse.
* The response pulse is low for 54us and then goes high for 80us.

**Data**



* After sending the response pulse, DHT11 sensor sends the data, which contains humidity and temperature value along with checksum.
* The data frame is of total 40 bits long, it contains 5 segments (byte) and each segment is 8-bit long.
* In these 5 segments, first two segments contain humidity value in decimal integer form. This value gives us Relative Percentage Humidity. 1st 8-bits are integer part and next 8 bits are fractional part.
* Next two segments contain temperature value in decimal integer form. This value gives us temperature in Celsius form.
* Last segment is the checksum which holds checksum of first four segments.
* Here checksum byte is direct addition of humidity and temperature value. And we can verify it, whether it is same as checksum value or not. If it is not equal, then there is some error in the received data.
* Once data received, DHT11 pin goes in low power consumption mode till next start pulse.

**End of frame**



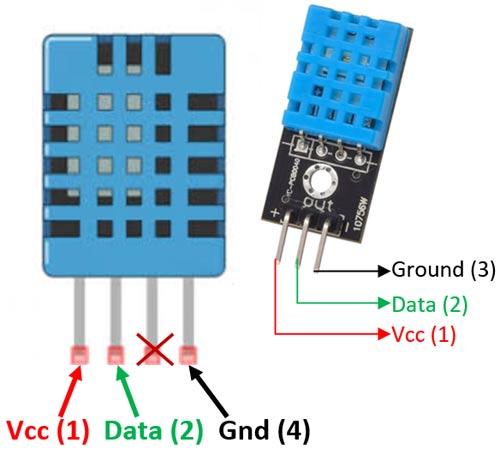
* After sending 40-bit data, DHT11 sensor sends 54us low level and then goes high. After this DHT11 goes in sleep mode.

**DHT11 vs DHT22**

Two versions of the DHT sensor, they look a bit similar and have the same pinout, but have different characteristics and specifications:

[**DHT11**](http://www.adafruit.com/products/386)

* Ultra-low cost
* 3 to 5V power and I/O
* 2.5mA max current use during conversion (while requesting data)
* Good for 20-80% humidity readings with 5% accuracy
* Good for 0-50°C temperature readings ±2°C accuracy
* No more than 1 Hz sampling rate (once every second)
* Body size 15.5mm x 12mm x 5.5mm
* 4 pins with 0.1" spacin



## Applications

* + This sensor is used in various applications such as measuring humidity and temperature values in heating, ventilation and air conditioning systems.
  + Weather stations also use these sensors to predict weather conditions.  The humidity[sensor](https://en.wikipedia.org/wiki/Humidity) is used as a preventive measure in homes where people are affected by humidity.  Offices, cars, museums, greenhouses and industries use this sensor for measuring humidity values and as a safety measure.

**4.4 RAIN SENSOR**:

A rain sensor is one kind of switching device which is used to detect the rainfall. It works like [a switch](https://www.elprocus.com/sound-activated-switch/) and the working principle of this sensor is, whenever there is rain, the switch will be normally closed.

### rain-sensor

### RAIN SENSOR MODULE:

The rain sensor module/board is shown below. Basically, this board includes nickel coated lines and it works on the resistance principle. This [sensor module](https://www.elprocus.com/pir-sensor-circuit-with-working/) permits to gauge moisture through analog output pins & it gives a digital output while moisture threshold surpasses.

This sensor is a resistive dipole, and based on the moisture only it shows the resistance. For example, it shows more resistance when it is dry and shows less resistance when it is wet.

### Pin Configuration

The pin configuration of this sensor is shown below. This sensor includes four pins which include the following.

* Pin1 (VCC): It is a 5V DC pin
* Pin2 (GND): it is a GND (ground) pin
* Pin3 (DO): It is a low/ high output pin
* Pin4 (AO): It is an analog output pin

### Specifications

The specifications of the rain sensor include the following.

### Applications:

The applications of rain sensor include the following.

* This sensor is used as a water preservation device and this is connected to the [irrigation system](https://www.elprocus.com/smart-irrigation-system-using-iot/) to shut down the system in the event of rainfall.
* This sensor is used to guard the internal parts of [an automobile](https://www.elprocus.com/different-microcontrollers-used-in-automobiles/) against the rainfall as well as to support the regular windscreen wiper’s mode.
* This sensor is used in specialized satellite communications aerials for activating a rain blower over the opening of the aerial feed, to get rid of water droplets from the mylar wrap to keep pressurized as well as dry air within the waveguides.

**4.4 BUZZER**:

There are many ways to communicate between the user and a product. One of the best ways is audio communication using a buzzer IC. So during the design process, understanding some technologies with configurations is very helpful. So, this article discusses an overview of an audio signaling device like a beeper or [a buzzer and its working](https://www.elprocus.com/8-candidate-quiz-buzzer-circuit-using-8051-microcontroller/) with applications.

## 4.4.1 What Is A Buzzer?

An audio signaling device like a beeper or buzzer may be electromechanical or [piezoelectric](https://www.elprocus.com/what-is-a-piezoelectric-material-working/) or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.





**Fig 4.6-Buzzer Pin Configuration**

The **pin configuration of the buzzer** is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the ‘+’ symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the ‘-‘symbol or short terminal and it is connected to the GND terminal. The history of an electromechanical buzzer and piezoelectric is discussed below.

#### **4.4.2 Electromechanical:**

This buzzer was launched in the year 1831 by an American Scientist namely Joseph Henry but, this was used in doorbells until they were eliminated in 1930 in support of musical bells, which had a smooth tone

#### **.3 Piezoelectric:**

These buzzers were invented by manufacturers of Japanese & fixed into a broad range of devices during the period of 1970s – 1980s. So, this development primarily came due to cooperative efforts through the manufacturing companies of Japanese. In the year 1951, they recognized the Application Research Committee of Barium Titanate that allows the corporations to be cooperative competitively & bring about numerous piezoelectric creations

**Fig4.7- Piezoelectric Buzzer**

### **4.4.4 Specifications:**

The **specifications of the buzzer** include the following.

* Color is black
* The frequency range is 3,300Hz
* Operating Temperature ranges from – 20° C to +60°C
* Operating voltage ranges from 3V to 24V DC
* The sound pressure level is 85dBA or 10cm
* The supply current is below 15Ma
  + 1. **Types of Buzzer:**

A buzzer is available in different types which include the following.

* Piezoelectric
* Electromagnetic
* Mechanical
* Electromechanical
* Magnetic

#### **Piezoelectric:**

As the name suggests, the piezoelectric type uses the piezoelectric ceramic’s piezoelectric effect & pulse current to make the metal plate vibrate & generate sound. This kind of buzzer is made with a resonance box, multi resonator, piezoelectric plate, housing, impedance matcher, etc. Some of the buzzers are also designed with [LEDs](https://www.elprocus.com/explain-different-types-leds-working-applications-engineering-students/). The multi resonator of this mainly includes ICs and transistors. Once the supply is given to this resonator, it will oscillate and generates an audio signal with 1.5 to 2.kHz. The impedance matcher will force the piezoelectric plate to produce sound.

#### **4.4.5.2 Electromagnetic:**

This type of buzzer is made with a magnet, solenoid coil, oscillator, housing, vibration diaphragm, and magnet. Once the [power supply](https://www.elprocus.com/regulated-power-supply-circuit-working-applications/) is given, the oscillator which produces the audio signal current will supply throughout the solenoid coil to generate a magnetic field.

Sometimes, the vibration diaphragm will vibrate & generates sound under the magnet & solenoid coil interaction. The frequency range of this ranges from 2 kHz to 4kHz.

#### **4.4.5.3 Mechanical:**

These types of buzzers are subtypes of electromagnetic, so the [components](https://www.elprocus.com/basic-components-used-electronics-electrical/) used in this type are also similar. But the main difference is that the vibrating buzzer is placed on the outside instead of the inside.

#### **4.4.5.4 Electromechanical:**

The designing of these types of buzzers can be done with a bare metal disc & an electromagnet. The working principle of this is similar to magnetic and electromagnetic. It generates sound throughout the disc movement & magnetism.

#### **4.4.5.5 Magnetic:**

Like a piezo type, magnetic is also used to generate a sound but they are different due to core functionality. The magnetic type is more fixed as compared to the piezo type because they work through a magnetic field. Magnetic buzzers utilize an electric charge instead of depending on piezo materials to generate a magnetic field, after that it permits another element of the buzzer to vibrate & generate sound.  
The applications of magnetic buzzers are similar to the piezo type in household devices, alarms such as watches, clocks & keyboards.

### **Working Principle**

The working principle of a buzzer depends on the theory that, once the voltage is given across a piezoelectric material, then a pressure difference is produced. A piezo type includes piezo crystals among two conductors. Once a potential disparity is given across these crystals, then they thrust one [conductor](https://www.elprocus.com/what-is-an-acsr-conductor-types-and-its-advantages/) & drag the additional conductor through their internal property. So this continuous action will produce a sharp sound signal.

### **4.4.6.1 Mounting Configurations**

The mounting configurations of buzzers include the following.

* Panel Mount
* Wire Leads
* Screw Terminals
* Through Hole
* Spring Contact
* Surface Mount

#### **How to use a Buzzer?**

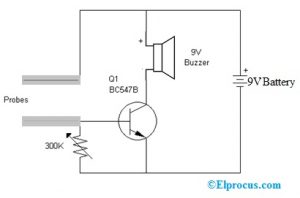
A buzzer is an efficient component to include the features of sound in our system or project. It is an extremely small & solid two-pin device thus it can be simply utilized on breadboard or PCB. So in most applications, this component is widely used.

There are two kinds of buzzers commonly available like simple and readymade. Once a simple type is power-driven then it will generate a beep sound continuously. A readymade type looks heavier & generates a Beep. Beep. Beep. This sound is because of the internal oscillating circuit within it.

This buzzer uses a DC power supply that ranges from 4V – 9V. To operate this, a 9V [battery](https://www.elprocus.com/what-is-lead-acid-battery-types-working-its-applications/) is used but it is suggested to utilize a regulated +5V/+6V DC supply. Generally, it is connected through a switching circuit to switch ON/OFF the buzzer at the necessary time interval.

### **Buzzer Circuit Diagram:**

The**circuit diagram of the water level indicator** **using the buzzer** is shown below. This circuit is used to sense or detect the water level within the tank or washing machine or pool, etc. This circuit is very simple to design using few components such as a transistor, buzzer, 300K variable resistor, and power supply or 9V battery.



**Fig 4.8-Water Level Indicator Buzzer**

Once the two probes of the circuit are placed in the tank, it detects the level of water. Once the water level exceeds the fixed level, then it generates a beep sound through a buzzer connected to the circuit. This circuit uses a [BC547B NPN transistor](https://www.elprocus.com/bc547-transistor-working-and-its-applications/) however we can also use any general-purpose transistor instead of using 2N3904/2N2222.

This water level sensor circuit working is very simple and the transistor used within the circuit works as a switch. Once the two probes notice the water level within the tank, then the transistor turns ON & the voltage begins flowing throughout the transistor to trigger the buzzer.

#### **4.4.9 How to Choose a Buzzer?**

While choosing a buzzer or speaker, many principles need to consider like the following.

* Size of the product
* Consumption of Current
* Type of terminal
* Frequency Voltage
* Volume
* Type
* AC/DC Voltage
* The tone is Continuous/Pulsed
* Fixing – Pins, Leads/Surface Mount
* Output of Sound
* Feedback Option
* Piezo Elements

**4.4.10 Advantages:**

The**advantages of a buzzer** include the following.

* Simply Compatible
* Frequency Response is Good
* Size is small
* Energy Consumption is less
* The Range of Voltage usage is Large
* Sound Pressure is high

### **4.4.11 Disadvantages:**

The **disadvantages of the buzzer** include the following.

* Controlling is a little hard
* Generates Annoying Sound
* Training is necessary to know how to repair the condition without just turning off.

### **4.4.12 Applications:**

The **applications of the buzzer** include the following.

* Communication Devices
* Electronics used in Automobiles
* Alarm Circuits
* Portable Devices
* Security Systems
* Timers
* Household Appliances
* Electronic Metronomes
* Sporting Events
* Annunciator

And the above mentioned are the applications of buzzer, according to the usage the buzzers are placed in several places in appliences and etc.

Buzzer plays a vital role in where it is used as above mentioned of its applications.

* 1. **LCD:**

The I2C 1602 LCD module is a 2 line by 16character display interfaced to an I2C daughter board. The I2C interface only requires 2 data connections, +5 VDC and GND to operate Viewable area Adjustable by potentiometer on I2c interface 80mm x 36mm x 20 mm. Now, with only 3 pins from microcontroller, you can display message on this LCD. Compared to parallel LCD which required at least 6 pins of I/O, this LCD offer more cost effective solution. The LCD display is four lines by 20 characters and provides basic text wrapping so that your text looks right on the display.

|  |
| --- |
|  |
| **FIG 4.9- I2C LCD**    **Fig 4.9.1 I2C LCD Display** |  |

# Specifications:

# 

**Fig 4.10- LCD Configurtion**

I2C Adderss range :0\*2 to0\*27

Operating :5vdc

Balcklight :white

**4.5.2 Feature:**

* 5V powered 4 x 20
* SPI communication
* Minimum 3 Pins interface to microcontroller
* Compatible with all types of microcontrollers
* Suitable for hobbyists and experts
* Back light and contrast control is available

Size: 99mm x 60mm x 10mm

## Pin/Control Description

## 

## Fig 4.11- Pin/ Control Description

## 

## CHAPTER 5

## SOFTWARE IMPLEMENTATION

## 5.1 THING SPEAK

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend. IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation. At a high level, many IoT systems can be described using the diagram below.

On the left, we have the smart devices (the “things” in IoT) that live at the edge of the network. These devices collect data and include things like wearable devices, wireless temperatures sensors, heart rate monitors, and hydraulic pressure sensors, and machines on the factory floor. In the middle, we have the cloud where data from many sources is aggregated and analyzed in real time, often by an IoT analytics platform designed for this purpose. The right side of the diagram depicts the algorithm development associated with the IoT application.

Here an engineer or data scientist tries to gain insight into the collected data by performing historical analysis on the data. In this case, the data is pulled from the IoT platform into a desktop software environment to enable the engineer or scientist to prototype algorithms that may eventually execute in the cloud or on the smart device itself. An IoT system includes all these elements. ThingSpeak fits in the cloud part of the diagram and provides a platform to quickly collect and analyze data from internet connected sensors.

**5.1.1 Thing Speak (IOT):**

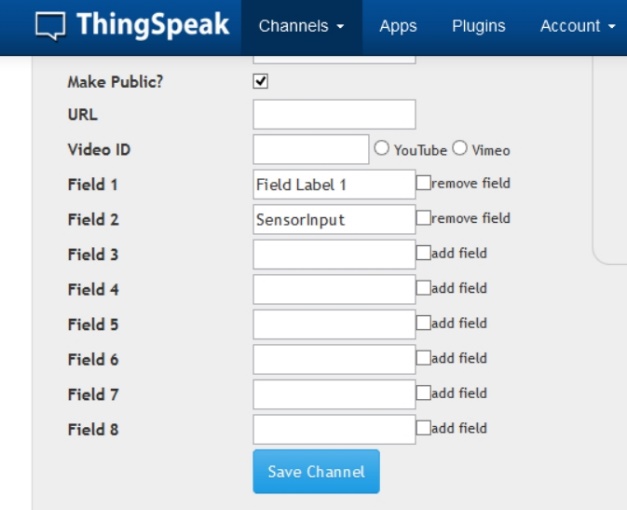
Thing Speak is IoT Cloud platform where you can send sensor data to the cloud. You can also analyze and visualize your data with MATLAB or other software, including making your own applications. The Thing Speak service is operated by Math Works. In order to sign up for Thing Speak, you must create a new Math Works Account or log in to your existing Math Works Account Thing Speak is free for small non-commercial projects.

Thing Speak includes a Web Service (REST API) that lets you collect and store sensor data in the cloud and develop Internet of Things applications. It works with Arduino, Raspberry Pi and MATLAB (premade libraries and APIs exists) But it should work with all kind of Programming Languages, since it uses a REST API and HTTP.

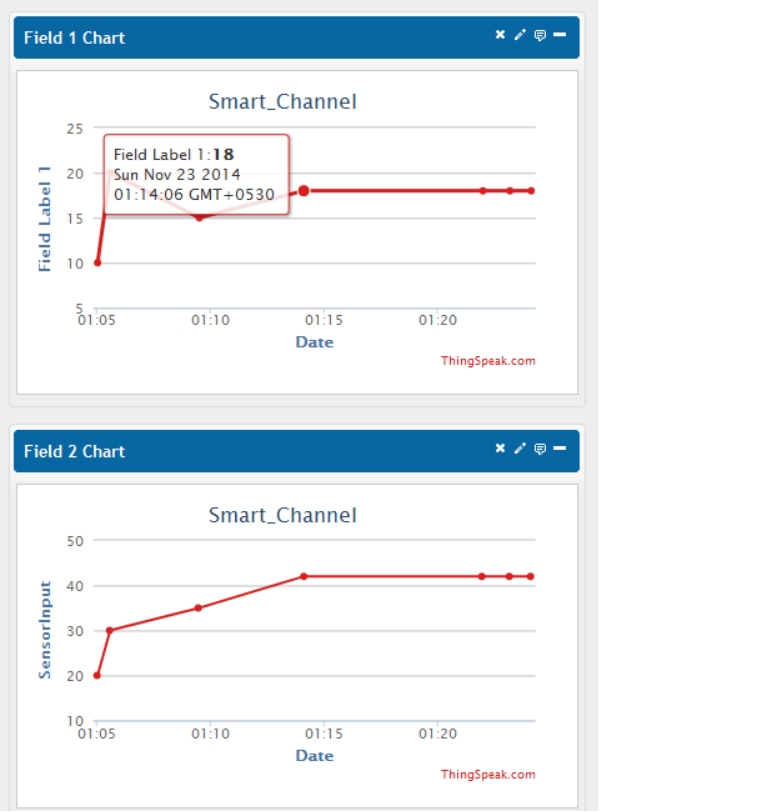
Thing Speak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in the form of charts, ability to create plugins and apps for collaborating with web services, social network and other APIs. We will consider each of these features in detail below.

The core element of Thing Speak is a ‘Thing Speak Channel’. A channel stores the data that we send to Thing Speak and comprises of the below elements:

* 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
* 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
* 1 status field - A short message to describe the data stored in the channel.



**Fig 5.1- Input Entering Window**



**Fig 5.2- Output Showing Windows**

**5.1.2 Thing Speak Key Features:**

ThingSpeak allows you to aggregate, visualize and analyze live data streams in the cloud. Some of the key capabilities of ThingSpeak include the ability to:

* Easily configure devices to send data to ThingSpeak using popular IoT protocols.
* Visualize your sensor data in real-time.
* Aggregate data on-demand from third-party sources.
* Use the power of MATLAB to make sense of your IoT data.
* Run your IoT analytics automatically based on schedules or events.
* Prototype and build IoT systems without setting up servers or developing web software.
* Automatically act on your data and communicate using third-party services like Twilio® or Twitter®.

**5.2 ARDUINO SOFTWARE IDE**

**5.2.1 DEFINITION:**

* The Arduino Integrated Development Environment (IDE) is a cross- platform application (for [Windows,](https://en.wikipedia.org/wiki/Windows) [macOS,](https://en.wikipedia.org/wiki/MacOS) [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++.](https://en.wikipedia.org/wiki/C%2B%2B_(programming_language))It is used to write and upload programs to [Arduino](https://en.wikipedia.org/wiki/Arduino) compatible boards, but also, with the help of third-party cores, other vendor development boards. The source code for the IDE is released under the [GNU General Public License,](https://en.wikipedia.org/wiki/GNU_General_Public_License) version The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring.
* The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable [cyclic executive](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain,](https://en.wikipedia.org/wiki/GNU_toolchain) also included with the IDE. The Arduino IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.
* Arduino IDE is a derivative of the [Processing IDE](https://en.wikipedia.org/wiki/Processing_(programming_language)), however as of version 2.0, the Processing IDE will be replaced with the [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)-based Eclipse Their IDE framework. With the rising popularity of Arduino as a software platform, other vendors started to implement custom open source compilers and tools (cores) that can build and upload sketches to other [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller) that are not supported by arduino’s official line of microcontrol

## 5.3 EMBEDDED C

**5.3.1 BASICS OF EMBEDDED C**

* Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems. So, in this article, we will see some of the Basics of Embedded C Program and the Programming Structure of Embedded C. Embedded C is perhaps the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++, Python etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability. Before digging in to the basics of Embedded C Program, we will first take a look at what an Embedded System is and the importance of Programming Language in Embedded Systems.
  + 1. **PROGRAMMING EMBEDDED SYSTEMS**

As mentioned earlier, Embedded Systems consists of both Hardware and Software.If we consider a simple Embedded System, the main Hardware Module is the Processor. The Processor is the heart of the Embedded System and it can be anything like a Microprocessor, Microcontroller, DSP, CPLD (Complex Programmable Logic Device) or an FPGA (Field Programmable Gated Array).All these devices have one thing in common: they are programmable i.e., we can write a program (which is the software part of the Embedded System) to define how the device actually works.

Embedded Software or Program allow Hardware to monitor external events (Inputs/ Sensors) and control external devices (Outputs) accordingly. During this process, the program for an Embedded System may have to directly manipulate the internal architecture of the Embedded Hardware (usually the processor) such as Timers, Serial Communications Interface, Interrupt Handling, and I/O Ports etc.

From the above statement, it is clear that the Software part of an Embedded System is equally important as the Hardware part. There is no point in having advanced Hardware Components with poorly written programs (Software).

There are many programming languages that are used for Embedded Systems like Assembly (low-level Programming Language), C, C++, JAVA (high-level programming languages), Visual Basic, JAVA Script (Application level Programming Languages), etc.

In the process of making a better embedded system, the programming of the system plays a vital role and hence, the selection of the Programming Language is very important.

* + 1. **FACTORS FOR SELECTING THE PROGRAMMING LANGUAGE**

The following are few factors that are to be considered while selecting the Programming Language for the development of Embedded Systems.

* + - 1. **Size:** The memory that the program occupies is very important as Embedded Processors like Microcontrollers have a very limited amount of ROM (Program Memory).
      2. **Speed**: The programs must be very fast i.e., they must run as fast as possible. The hardware should not be slowed down due to a slow running software.
      3. **Portability:** The same program can be compiled for different processors.

# CHAPTER – 6

# SOURCE CODE AND RESULT

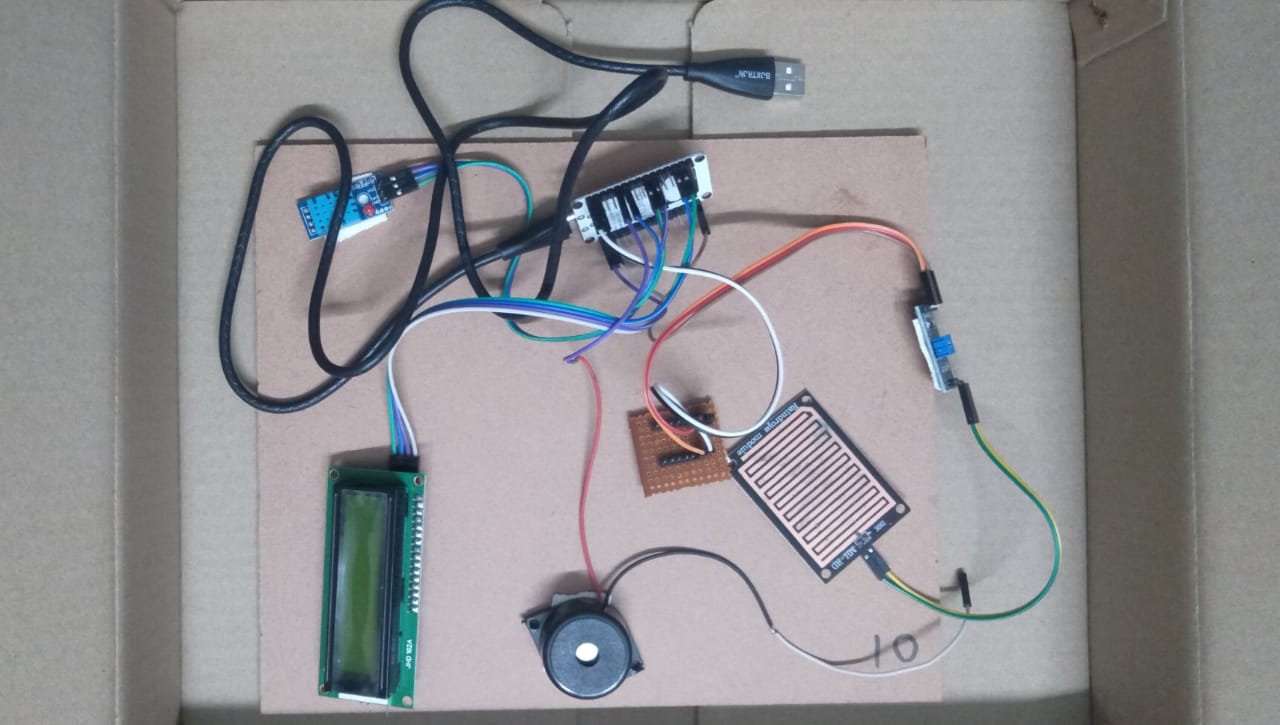
## SOURCE CODE

1. #define SW\_VERSION " ThinkSpeak.com" // SW version will appears at innitial LCD Display
2. #include <Wire.h>
3. #include <LiquidCrystal\_I2C.h>
4. // Set the LCD address to 0x27 for a 16 chars and 2 line display
5. LiquidCrystal\_I2C lcd(0x3F, 16, 2);
6. #include <ESP8266WiFi.h>
7. #include <WiFiClientSecure.h>
8. #include <DHT.h>
9. #define DHTPIN D7 // what digital pin we're connected to
10. #define DHTTYPE DHT11 // DHT 22 (AM2302), AM2321)
11. DHT dht(DHTPIN, DHTTYPE);
12. const char\* MY\_SSID = "chary's Oppo";
13. const char\* MY\_PWD = "12345678";
14. WiFiClient client;
15. const char\* TS\_SERVER = "api.thingspeak.com";
16. String TS\_API\_KEY = "KL7ZWFR9VAQIJH6E";
17. int rain=A0;
18. int buzzer=D8;
19. void setup() {
20. pinMode(buzzer,OUTPUT);
21. pinMode(rain,INPUT);
23. Serial.begin(9600);
24. dht.begin();
25. lcd.begin();
26. // Turn on the blacklight and print a message.
27. lcd.backlight();
28. lcd.print("welcome");
29. connectWifi();
30. // put your setup code here, to run once:
31. }
32. void connectWifi()
33. {
34. Serial.print("Connecting to " + \*MY\_SSID);
35. WiFi.begin(MY\_SSID, MY\_PWD);
36. while (WiFi.status() != WL\_CONNECTED)
37. {
38. delay(1000);
39. Serial.print(".");
40. }
41. Serial.println("");
42. Serial.println("WiFi Connected");
43. Serial.println("");
45. }
46. void sendDataTS(void)
47. {
48. float h = dht.readHumidity();
49. // Read temperature as Celsius (the default)
50. float t = dht.readTemperature();
51. // Read temperature as Fahrenheit (isFahrenheit = true)
52. float f = dht.readTemperature(true);
53. // Check if any reads failed and exit early (to try again).
54. if (isnan(h) || isnan(t) || isnan(f)) {
55. Serial.println(F("Failed to read from DHT sensor!"));
56. return;
57. }
58. // Compute heat index in Fahrenheit (the default)
59. float hif = dht.computeHeatIndex(f, h);
60. // Compute heat index in Celsius (isFahreheit = false)
61. float hic = dht.computeHeatIndex(t, h, false);
62. Serial.print(F("Humidity: "));
63. Serial.println(h);
64. delay(1000);
65. lcd.clear();
66. lcd.print("Humidity: ");
67. lcd.print(h);
68. delay(1000);
69. if(h>45)
70. {
71. digitalWrite(buzzer,HIGH);
72. delay(1000);
73. }
74. else{
75. digitalWrite(buzzer,LOW);
76. delay(1000);
77. }
79. Serial.print(F("Temperature: "));
80. Serial.println(t);
81. delay(1000);
82. lcd.clear();
83. lcd.print("Temperature:");
84. lcd.print(t);
85. delay(1000);
86. if(t>30)
87. {
88. digitalWrite(buzzer,HIGH);
89. delay(1000);
90. }
91. else{
92. digitalWrite(buzzer,LOW);
93. delay(1000);
94. }
95. int x=analogRead(rain);
96. Serial.print("rain value :");
97. Serial.println(x);
98. delay(1000);
99. lcd.clear();
100. lcd.print("rain value:");
101. lcd.print(x);
102. delay(1000);
104. if(x>475)
105. {
106. Serial.print("havey rain");
107. digitalWrite(buzzer,HIGH);
108. delay(1000);
109. }
110. else{
111. digitalWrite(buzzer,LOW);
112. delay(1000);
113. }

116. if (client.connect(TS\_SERVER, 80))
117. {
118. String postStr = TS\_API\_KEY;
119. postStr += "&field1=";
120. postStr += String(x);
121. postStr += "&field2=";
122. postStr += String(t);
123. postStr += "&field3=";
124. postStr += String(h);
125. postStr += "\r\n\r\n";
126. client.print("POST /update HTTP/1.1\n");
127. client.print("Host: api.thingspeak.com\n");
128. client.print("Connection: close\n");
129. client.print("X-THINGSPEAKAPIKEY: " + TS\_API\_KEY + "\n");
130. client.print("Content-Type: application/x-www-form-urlencoded\n");
131. client.print("Content-Length: ");
132. client.print(postStr.length());
133. client.print("\n\n");
134. client.print(postStr);
135. delay(1000);
136. }
137. client.stop();
138. }
139. void loop() {
140. sendDataTS();
141. delay(10);
142. }

## RESULT

These are the outputs which are observed for our project while under working.



**Fig 6.1**  **weather monitoring kit**

**CHAPTER- 7**

**7.1 CONCLUSION:**

As the conclusion this project have cleared the objective that to build a system that can monitored weather parameter by wireless system and IoT. The Sensor station and Weather station will be communicated by hotspot Wi-Fi and it is limited in areas covered but still better in communication via wireless. The value that been recorded from google sheet and Table 1, 2 and 3 it seen that the weather at particular place has different condition from the exact condition with the accuracy of weather reporting system and forecast system data has been compared. It says that weather reporting system is more accurate than forecast system.

**REFERENCES:**

* [1] M. H. Asghar, A. Negi, and N. Mohammadzadeh, “Principle application and vision in internet of things (iot),” in International Conference on Computing, Communication Automation, May 2015, pp. 427–431.
* [2] A. Gheith, R. Rajamony, P. Bohrer, K. Agarwal, M. Kistler, B. L. W. Eagle, C. A. Hambridge, J. B. Carter, and T. Kaplinger, “Ibm bluemix mobile cloud services,” IBM Journal of Research and Development, vol. 60, no. 2-3, pp. 7:1–7:12, March 2016.
* [3] S. Gangopadhyay and M. K. Mondal, “A wireless framework for environmental monitoring and instant response alert,” in 2016 International Conference on Microelectronics, Computing and Communications (MicroCom), Jan 2016, pp. 1–6.
* [4] H. Saini, A. Thakur, S. Ahuja, N. Sabharwal, and N. Kumar, “Arduino based automatic wireless weather station with remote graphical application and alerts,” in 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN), Feb 2016, pp. 605–609. [5] A. Lage and J. C. Correa, “Weather station with cellular communication network,”,

**FUTURE SCOPE:**

* The IOT Based weather Informative System will proposed to Real time Applications. It doesn’t need of any data centers physically because of we are creating a data Server in Cloud so that it doesn’t require any datacenter further. So It Reduces the Cost of Equipment.
* Many of the Innovative Researchers are interested towards The IOT based Real time Applications. So this System will helps to the Researcher for their further investigation of weather details.
* The IOT Based Weather Informative System not only displays the weather parameters like Temperature, Altitude, Humidity and Pressure etc., but it also displays the weather location, Industry, Time and other weather information from this we can forecasts the weather details.