Mini Project Report

on

IOT Based Weather Monitoring and Reporting System

Submitted by

1032190705 -Zeeshan Mujawar (PA18)
1032190723-Chaitanya Nirfarake (PA19)
1032190915 -Aditya Lanjewar (PA29)
1032190930-Shreyanshu Mane (PA30)

Under the guidance of

Prof. Smita Khole

School of Computer Engineering and Technology
MIT World Peace University, Kothrud,
Pune 411 038, India

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

MIT-WPU
।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

CERTIFICATE

This is to certify that

1032190705 -Zeeshan Mujawar (PA18)
1032190723-Chaitanya Nirfarake (PA19)
1032190915 -Aditya Lanjewar (PA29)
1032190930-Shreyanshu Mane (PA30)

of T. Y. B. Tech. successfully completed Mini Project in

IOT Based Weather Monitoring and Reporting System

to my satisfaction and submitted the same during **Trimester VII, Academic Year 2021-22** as part of **Embedded and Internet of Things Laboratory** course.

Prof. Smita Khole
Course Teacher

Prof. Vrushali Kulkarni
Head of School

Place: SCET, MIT-WPU, Pune
Date: 04-10-2021

Table of Contents

ABSTRACT

Internet of Things (IoT) is adding value to products and applications in the recent years. The connectivity of the IoT devices over the network has widely reduced the power consumption, robustness, and connectivity to access data over the network. IoT is powering many frontiers of industries and is seen as a promising technology to take Big Data Analytics to a level higher. Weather monitoring system as a module is an issue among IoT research community and it has been widely addressed. A new weather monitoring system is developed using various sensors connecting to Raspberry Pi. The implementation and data visualization on the data collected are discussed in this paper in detail. Weather parameters like temperature, humidity and rain are monitored and visualized in graphical means using the Raspberry Pi as server and data accessed over the intranet or internet in a specified subnet or world wide web. The data visualization is provided as result and proves to be a robust framework for analyzing weather parameters in any geographical location studying the effect of these varying parameters like temperature, moisture, and rain on the crops.

Chapter 1: Introduction

Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given location. Human beings have attempted to predict the weather informally for millennium and formally since the nineteenth century. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere on a given place and using scientific understanding of atmospheric processes to project how the atmosphere will evolve on that place.

Weather is driven by air pressure differences between one place and another, temperature, and moisture. These pressure and temperature differences can occur due to the sun angle at any spot, which varies by latitude from the tropics and many other minute details. The atmosphere is a chaotic system, so small changes to one part of the system can grow to have large effects on the system. This makes it difficult to accurately predict weather more than a few days in advance, though weather forecasters are continually working to extend this limit through the scientific study of weather, meteorology.

Existing technology mainly focus on controlling and monitoring of different activities. These are increasingly emerging to reach the human needs. An efficient environmental monitoring system is required to monitor and assess the conditions in case of exceeding the prescribed level of parameters. Sensors are placed at different locations to collect the data to predict the behavior of a particular area of interest. The main aim of this report is to design and implement an efficient monitoring system through which the required parameters are monitored remotely using internet and the data gathered from the sensors are stored in the cloud and to project the estimated trend on the web browser. The value from the cloud is updated at each and every moment. The crops are cultivated, and the soil are tested mainly the moisture is measured. Thus, we can cultivate different crops at a particular area.

Chapter 2: Related Work

2.1 Literature Survey

Weather Monitoring System has been done extensively using IoT devices in the past. The wide range of literatures discussed provide information like a survey. The existing technologies are developed using microcontrollers like Arduino, Node MCU etc and ARM processors like Raspberry Pi. The implementation proposed in shows that a Raspberry Pi based weather monitoring system is developed using PM 2.5 sensors. Apart from temperature, pressure and humidity, rain is also monitored. This is not purely Raspberry Pi based implementation, but a combination of Pi and Arduino Nano. An IoT node based framework was proposed by [P. P. Ray, "Internet of Things Cloud Based Smart Monitoring of Air Borne PM 2.5 Density Level," in International conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016, pp. 995–999]. This methodology uses various sensor and monitor it periodically and stores in a cloud server. A raspberry Pi based weather monitoring system is proposed in [S. S. Sarnin, A. Akbar, W. N. W. Mohamad, A. Idris, N. f. Naim, and N. Yaacob, "Maleficent Mirror with ALEXA Voice Services as an Internet of Things Implement Using Raspberry Pi 3 Model B," in TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 1202–1207]. This gets basic parameters like temperature, humidity, and few other parameters. A weather monitoring system for agriculture was developed by [Y. Alif, K. Utama, Y. Widianto, Y. Hari, and M. Habiburrahman, "Design of Weather Monitoring Sensors and Soil Humidity in Agriculture Using Internet of Things (IoT)," Trans. Mach. Intell. Artif. Intell., vol. 7, no. 1, pp. 10–20, 2019] using Node MCU. This uses a temperature and humidity sensor and soil moisture sensor for monitoring weather with respect to agriculture. A higher resilience based weather monitoring system using microcontroller was developed by [D. Strigaro and M. Cannata, "Boosting a Weather Monitoring System in Low Income Economies Using Open and Non-Conventional Systems : Data Quality Analysis," Sensors, vol. 19, no. 5, pp. 1–22, 2019.]. This was done mainly to target underdeveloped and developing economies. This used temperature, humidity, and rain gauge systems to obtain data in real time.

Chapter 3: Proposed Work

3.1 Problem Statement:

Weather monitoring system being very hand for better yield from rural agriculture lands has the issue of higher cost. The hard drive-based data logging facility requires a separate computer setup for its operation and many a times, the data stored cannot be manipulated in a useful mean. These two problems are the primary concerns when you consider a weather monitoring system and we have come up with cost effective innovative solution to provide the layman's weather monitoring system.

The Problem found in most weather Stations recently all the weather Stations Consists of their Own Data Centre to Access and send the information to Display devices. Each data center needs crores to build their own data center in the particular place. IoT Based Weather System acts as Weather Station, and it update the Data Centre in Cloud. So, by using IoT Based Weather Monitoring System we can solve the cost of equipment problem and, we can also access the information remotely through internet Devices and Websites.

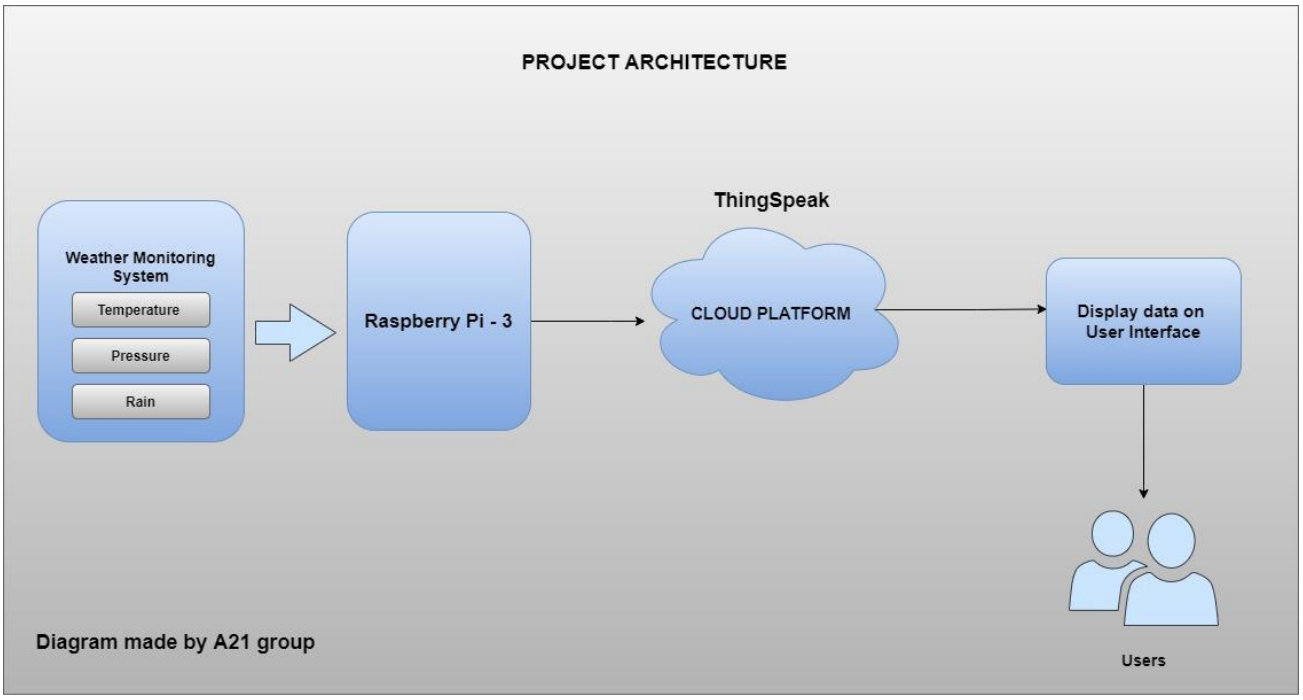
3.2 Social Relevance:

Our objective is to help farmers, especially rural farmers get the best out of their farmlands by having a better understanding of the upcoming weather to better prepare their fields and crop for the upcoming situations.

Rural India suffers a lot from unpredictable weathers each year, resulting in massive losses. Agriculture giving the highest employment in India needs to grow a lot technologically advanced which currently comes at a higher cost which might not be in capacity of the general rural population. Thus, we want to try and bring this into the reaches of the common.

3.3 Architecture:

The architecture of the system implementation is given in figure below. It shows the hardware connected to the raspberry Pi and the software used in managing the data collected using the sensors. The scope of this framework is to store the data in a cloud platform. Data collected is made available to be seen on ThingSpeak website as well as mobile application.

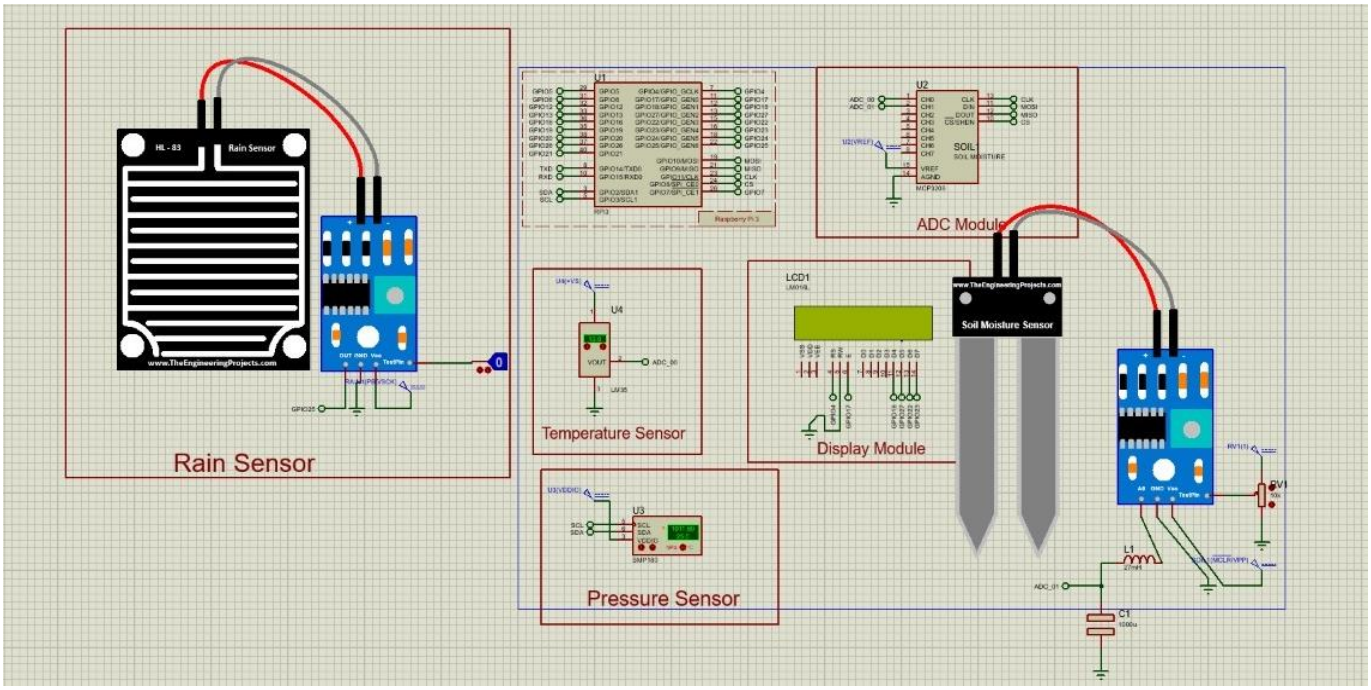


The sensors are connected to the Raspberry Pi's input terminals as per the specifications. ThingSpeak for IoT devices is used for remote data viewing. Python is used to code for the interfacing the sensors, extraction and storing of data from the sensors.

Raspberry Pi's takes the data from all the connected sensors and uploads it to the ThingSpeak Cloud Platform for viewing and analysis of collected data, like temperature, moisture, and rain.

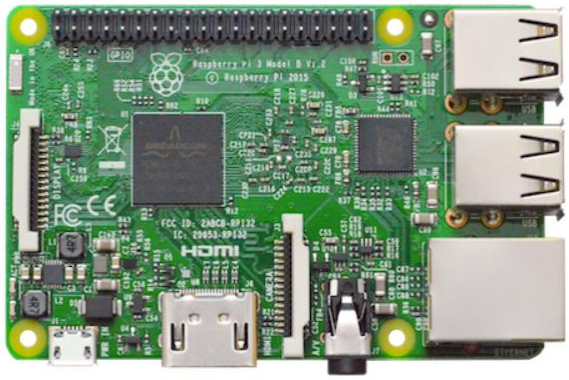
3.4 Hardware and Software requirements:

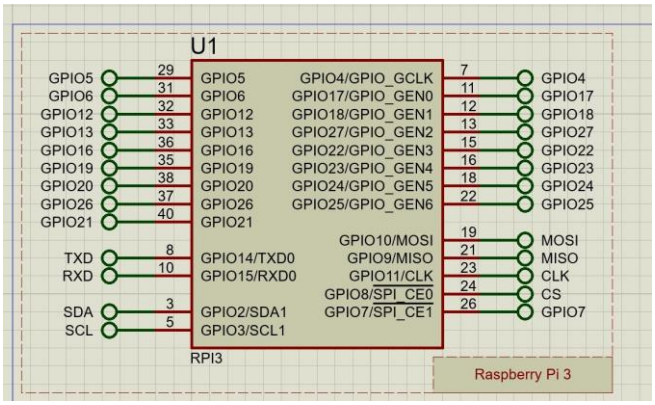
Circuit Diagram:



Raspberry Pi 3

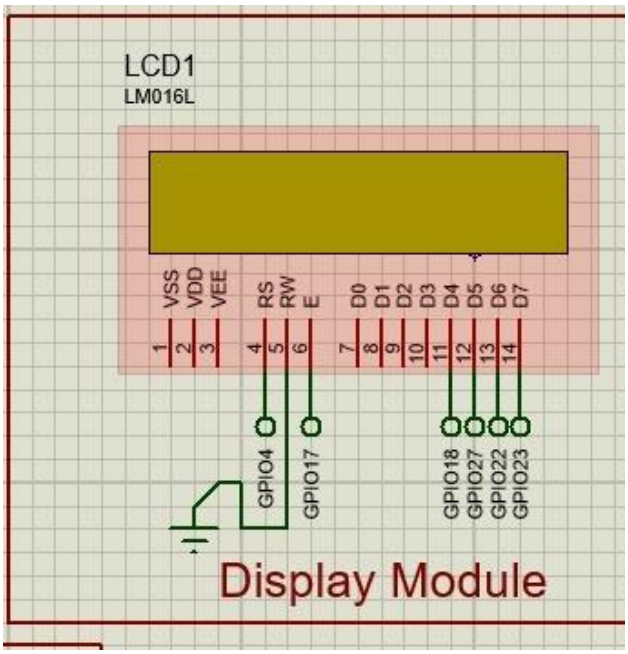
Raspberry Pi 3 Model B is the latest iteration of the world's most popular single board computer. It provides a quad-core 64-bit ARM Cortex-A53 CPU running at 1.2GHz, four USB 2.0 ports, wired and wireless networking, HDMI and composite video output, and a 40-pin GPIO connector for physical interfacing projects.





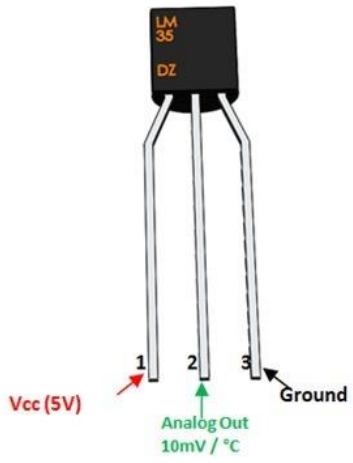
LCD LM016L

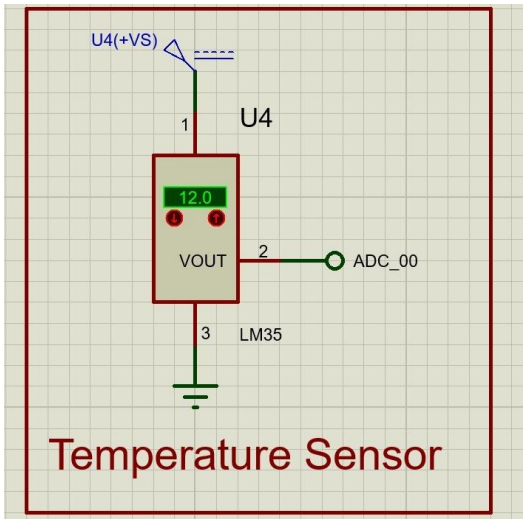
- 16 Character x 2 lines
- Built-in control LSI HD44780 type
- +5V single power supply



Pin No.	Symbol	Level	Function	
1	V_{SS}	–	0V	Power supply
2	V_{DD}	–	+5V	
3	V_O	–	–	
4	RS	H/L	L: Instruction code input H: Data input	
5	R/W	H/L	H: Data read (LCD module→MPU) L: Data write (LCD module←MPU)	
6	E	H, H→L	Enable signal	
7	DB0	H/L	Data bus line Note (1), Note (2)	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

LM35 Temperature Sensor



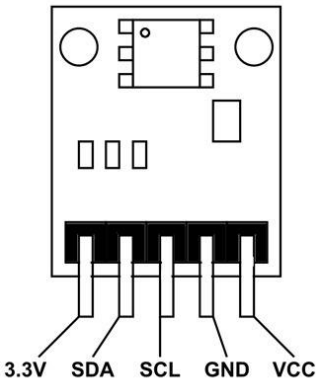


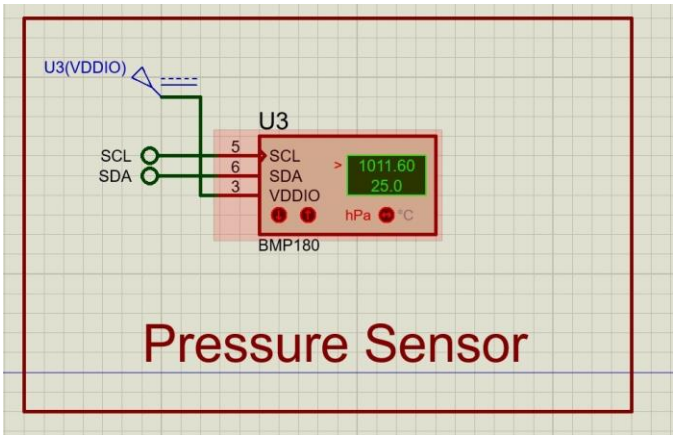
1. Vcc: Input voltage is +5V for typical applications
2. Analog Out: There will be increase in 10mV for raise of every 1°C. Can range from -1V(-55°C) to 6V(150°C)
3. Ground: Connected to ground of circuit

LM35 Regulator Features:

- Minimum and Maximum Input Voltage is 35V and -2V respectively. Typically, 5V.
- Can measure temperature ranging from -55°C to 150°C
- Output voltage is directly proportional (Linear) to temperature (i.e.) there will be a rise of 10mV (0.01V) for every 1°C rise in temperature.
- ±0.5°C Accuracy
- Drain current is less than 60uA
- Low-cost temperature sensor
- Small and hence suitable for remote applications

BMP180 Pressure Sensor

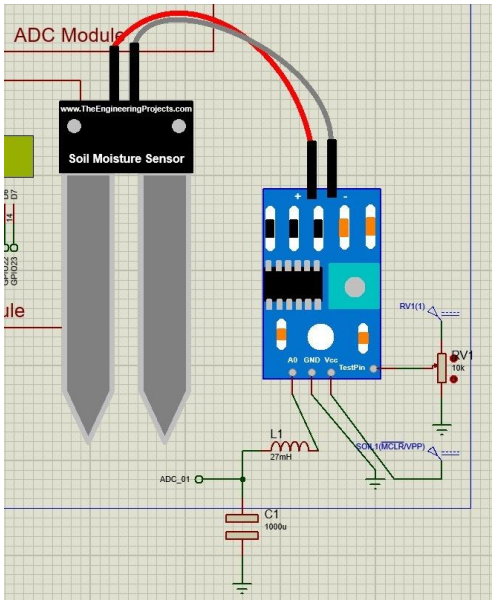




VCC: Connected to +5V
GND: Connected to ground.
SDA: Serial Data pin (I2C interface)
SCL: Serial Clock pin (I2C interface)
3.3V: If +5V is not present. Can power module by connecting +3.3V to this pin.
BMP180 MODULE Features

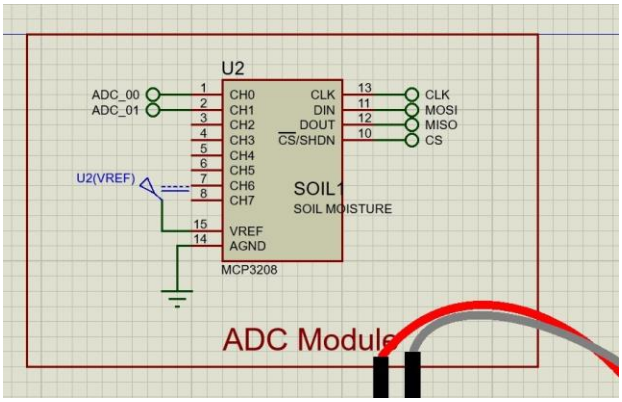
- Can measure temperature and altitude.
- Pressure range: 300 to 1100hPa
- High relative accuracy of ±0.12hPa
- Can work on low voltages
- 3.4Mhz I2C interface
- Low power consumption (3uA)
- Pressure conversion time: 5msec
- Potable size

Soil Moisture Module:



The soil moisture module is used to measure the soil moisture levels.

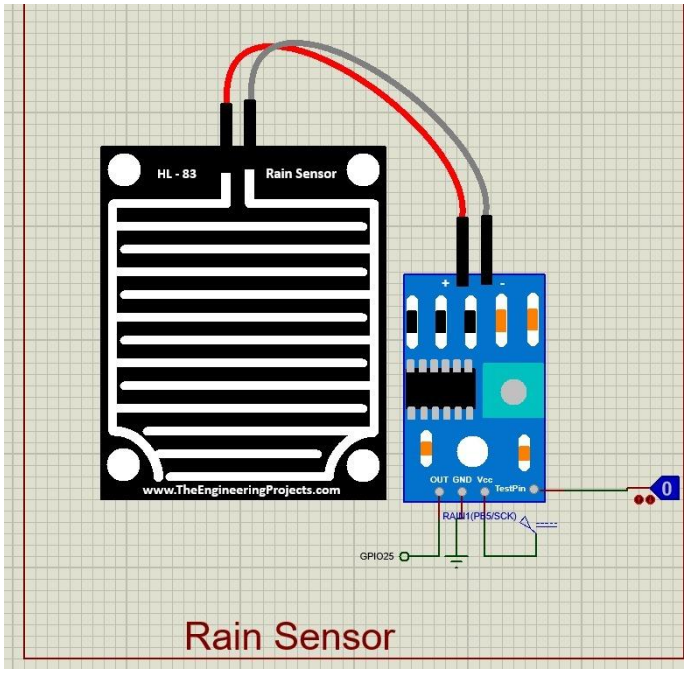
ADC Module:



Name	Function
V_{DD}	+2.7V to 5.5V Power Supply
DGND	Digital Ground
AGND	Analog Ground
CH0-CH7	Analog Inputs
CLK	Serial Clock
D_{IN}	Serial Data In
D_{OUT}	Serial Data Out
$\overline{CS}$/SHDN	Chip Select/Shutdown Input
V_{REF}	Reference Voltage Input

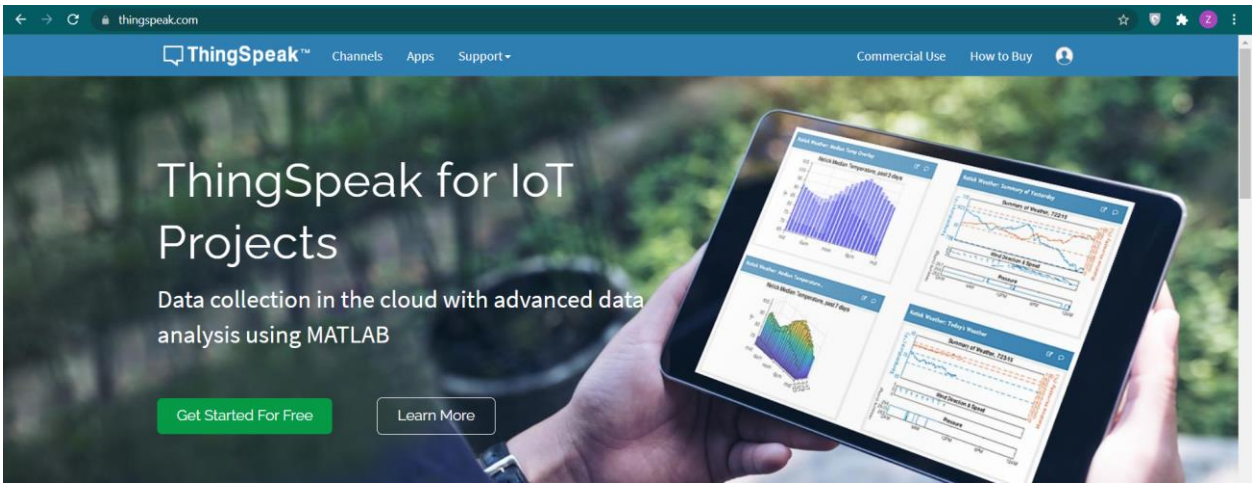
The ADC module, or the analog to digital convertor, is used to convert analog signals from the sensor to digital signals.

Rain Sensor:



The rain sensor is used to report if rain is currently pouring or not.

ThingSpeak Cloud Platform:



ThingSpeak is an IoT analytics platform sevice that allows us to aggregate, visualise and analyise live data streams in the cloud.

Python Program:

```
# !/usr/bin/env python3

#!/usr/bin/env python3

# Main.py file generated by New Project wizard

# Created:   Sat Jun 26 2021
# Processor: RPI3
# Compiler:  Python 3 (Proteus)

# Modules
from goto import *
import time
import var
import pio
import resource
import spidev
import RPi.GPIO as GPIO
import urllib.request
import requests
import smbus
from ctypes import c_short

# Peripheral Configuration Code (do not edit)
#---CONFIG_BEGIN---
import cpu
import FileStore
import VFP
import Ports

def peripheral_setup () :
# Peripheral Constructors
 pio.cpu=cpu.CPU ()
 pio.storage=FileStore.FileStore ()
 pio.server=VFP.VfpServer ()
 pio.uart=Ports.UART ()
 pio.storage.begin ()
 pio.server.begin (0)
# Install interrupt handlers

def peripheral_loop () :
 pass

#---CONFIG_END---
```

```
# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Define GPIO to LCD mapping
LCD_RS = 4
LCD_E  = 17
LCD_D4 = 18
LCD_D5 = 27
LCD_D6 = 22
LCD_D7 = 23
Relay_pin= 24
Rain_sensor = 25
# Define sensor channels
temp_channel  = 0
Moisture_channel =1

'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1



GPIO.setup(LCD_E, GPIO.OUT)  # E
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7
GPIO.setup(Relay_pin, GPIO.OUT) # Motor_1
GPIO.setup(Rain_sensor, GPIO.IN)
# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False
LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

'''
Function Name :lcd_init()
Function Description : this function is used to initialized lcd by sending the different commands
'''
```

```
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)
'''
Function Name :lcd_byte(bits ,mode)
Fuction Name :the main purpose of this function to convert the byte data into bit and send to lcd port
'''
def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command

  GPIO.output(LCD_RS, mode) # RS

  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
```

```
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()
'''
Function Name : lcd_toggle_enable()
Function Description:basically this is used to toggle Enable pin
'''
def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
'''
Function Name :lcd_string(message,line)
Function  Description :print the data on lcd
'''
def lcd_string(message,line):
  # Send string to display

  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)




# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
  adc = spi.xfer2([1,(8+channel)<<4,0])
  data = ((adc[1]&3) << 8) + adc[2]
  return data


# Function to calculate temperature from
# TMP36 data, rounded to specified
```

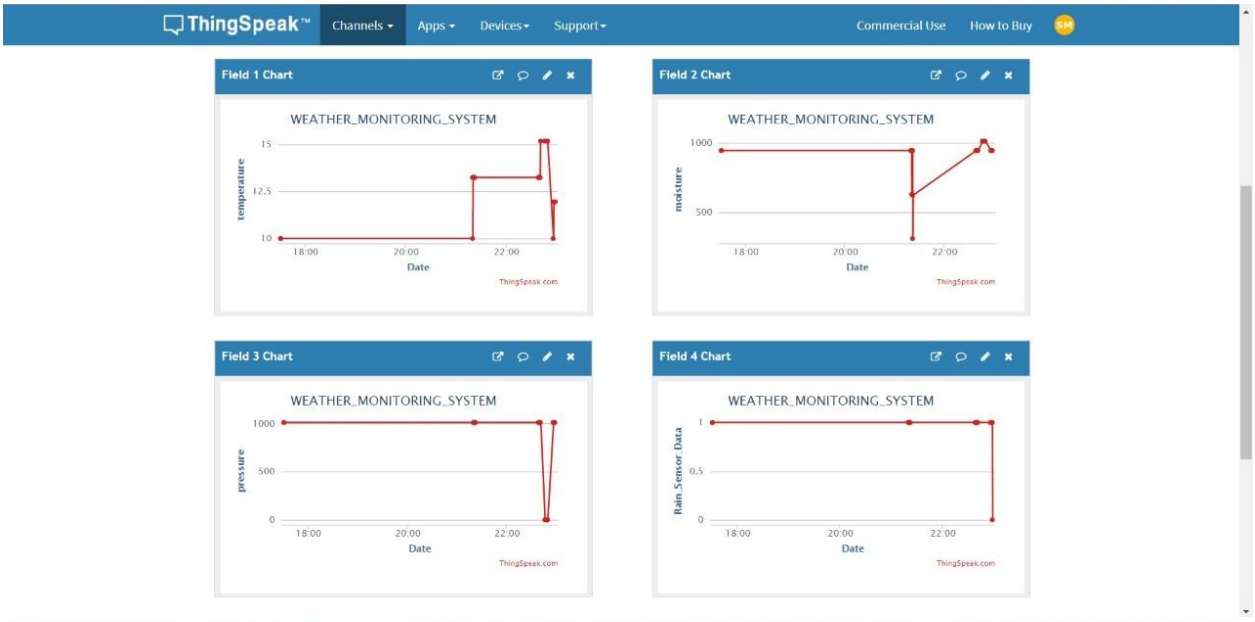
```
# number of decimal places.
def ConvertTemp(data,places):

  # ADC Value
  # (approx)  Temp  Volts
  #    0      -50    0.00
  #   78      -25    0.25
  #  155        0    0.50
  #  233       25    0.75
  #  310       50    1.00
  #  465      100    1.50
  #  775      200    2.50
  # 1023      280    3.30

  temp = ((data * 330)/float(1023))
  temp = round(temp,places)
  return temp

def thingspeak_post(temp,moisture_level,pressure,rain_data):
    URl='https://api.thingspeak.com/update?api_key='
    #Enter Your Private Key here
    KEY='BY8H20WNMWL94ZH1'

HEADER='&field1={}&field2={}&field3={}&field4={}'.format(temp,moisture_level,pressure,rain_data)
    NEW_URL=URl+KEY+HEADER
    print(NEW_URL)
    data=urllib.request.urlopen(NEW_URL)
    print(data)

DEVICE = 0x77 # Default device I2C address

#bus = smbus.SMBus(0)  # Rev 1 Pi uses 0
bus = smbus.SMBus(1) # Rev 2 Pi uses 1

def convertToString(data):
  # Simple function to convert binary data into
  # a string
  return str((data[1] + (256 * data[0])) / 1.2)

def getShort(data, index):
  # return two bytes from data as a signed 16-bit value
  return c_short((data[index] << 8) + data[index + 1]).value

def getUshort(data, index):
  # return two bytes from data as an unsigned 16-bit value
  return (data[index] << 8) + data[index + 1]
```

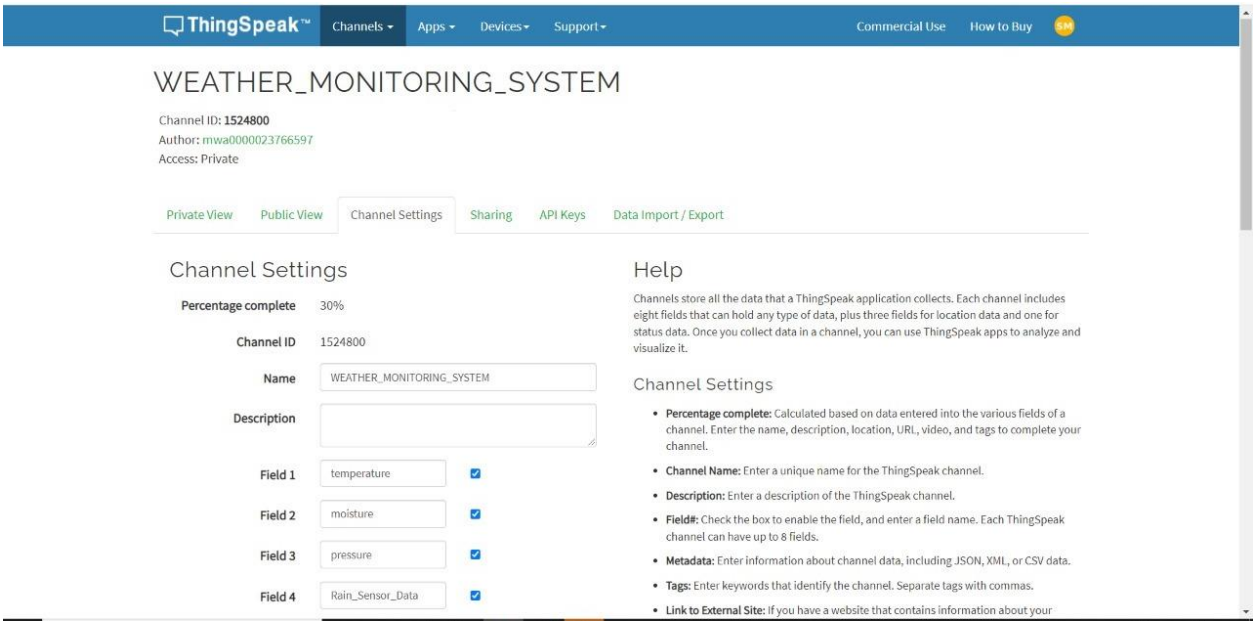
```
def readBmp180Id(addr=DEVICE):
  # Chip ID Register Address
  REG_ID     = 0xD0
  (chip_id, chip_version) = bus.read_i2c_block_data(addr, REG_ID, 2)
  return (chip_id, chip_version)

def readBmp180(addr=DEVICE):
  # Register Addresses
  REG_CALIB  = 0xAA
  REG_MEAS   = 0xF4
  REG_MSB    = 0xF6
  REG_LSB    = 0xF7
  # Control Register Address
  CRV_TEMP   = 0x2E
  CRV_PRES   = 0x34
  # Oversample setting
  OVERSAMPLE = 3    # 0 - 3

  # Read calibration data
  # Read calibration data from EEPROM
  cal = bus.read_i2c_block_data(addr, REG_CALIB, 22)

  # Convert byte data to word values
  AC1 = getShort(cal, 0)
  AC2 = getShort(cal, 2)
  AC3 = getShort(cal, 4)
  AC4 = getUshort(cal, 6)
  AC5 = getUshort(cal, 8)
  AC6 = getUshort(cal, 10)
  B1  = getShort(cal, 12)
  B2  = getShort(cal, 14)
  MB  = getShort(cal, 16)
  MC  = getShort(cal, 18)
  MD  = getShort(cal, 20)

  # Read temperature
  bus.write_byte_data(addr, REG_MEAS, CRV_TEMP)
  time.sleep(0.005)
  (msb, lsb) = bus.read_i2c_block_data(addr, REG_MSB, 2)
  UT = (msb << 8) + lsb

  # Read pressure
  bus.write_byte_data(addr, REG_MEAS, CRV_PRES + (OVERSAMPLE << 6))
  time.sleep(0.04)
  (msb, lsb, xsb) = bus.read_i2c_block_data(addr, REG_MSB, 3)
  UP = ((msb << 16) + (lsb << 8) + xsb) >> (8 - OVERSAMPLE)
```

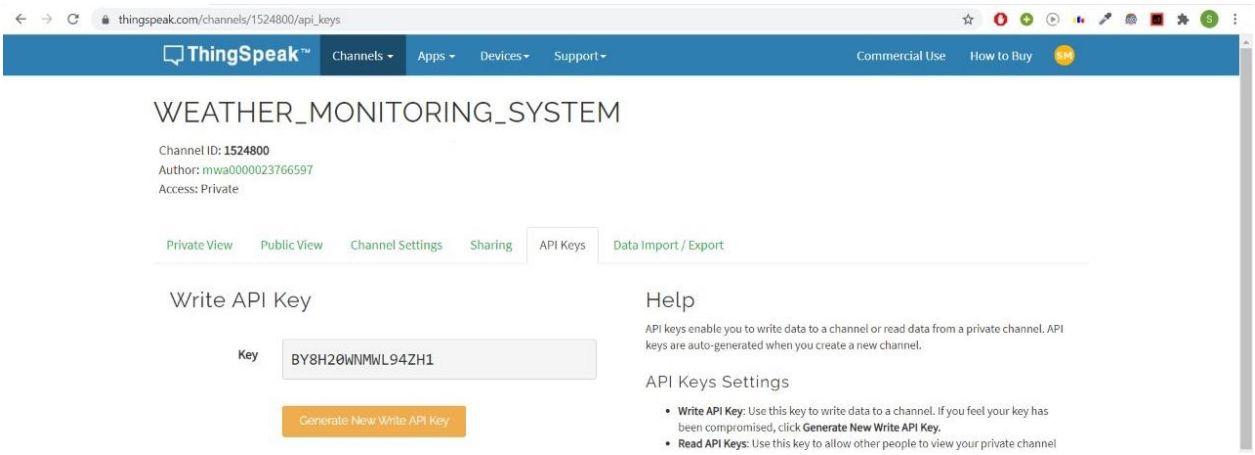
```
  # Refine temperature
  X1 = ((UT - AC6) * AC5) >> 15
  X2 = (MC << 11) / (X1 + MD)
  B5 = X1 + X2
  #temperature = int(B5 + 8) >> 4

  # Refine pressure
  B6  = B5 - 4000
  B62 = int(B6 * B6) >> 12
  X1  = (B2 * B62) >> 11
  X2  = int(AC2 * B6) >> 11
  X3  = X1 + X2
  B3  = (((AC1 * 4 + X3) << OVERSAMPLE) + 2) >> 2

  X1 = int(AC3 * B6) >> 13
  X2 = (B1 * B62) >> 16
  X3 = ((X1 + X2) + 2) >> 2
  B4 = (AC4 * (X3 + 32768)) >> 15
  B7 = (UP - B3) * (50000 >> OVERSAMPLE)

  P = (B7 * 2) / B4

  X1 = (int(P) >> 8) * (int(P) >> 8)
  X1 = (X1 * 3038) >> 16
  X2 = int(-7357 * P) >> 16
  pressure = int(P + ((X1 + X2 + 3791) >> 4))

  return (pressure/100.0)


# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(1)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
lcd_string("Weather Monitoring ",LCD_LINE_1)
lcd_string("System ",LCD_LINE_2)
time.sleep(1)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
```

```
# Main function
def main () :
# Setup
 peripheral_setup()
 peripheral_loop()
 #Motor Status
 motor_status = 0
# Infinite loop
 while 1 :

  temp_level = ReadChannel(temp_channel)
  temp       = ConvertTemp(temp_level,2)

  # Print out results
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  lcd_string("Temperature  ",LCD_LINE_1)
  lcd_string(str(temp),LCD_LINE_2)
  time.sleep(0.1)

  moisture_level = ReadChannel(Moisture_channel)
  # Print out results
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  lcd_string("Moisture Level  ",LCD_LINE_1)
  lcd_string(str(moisture_level),LCD_LINE_2)
  time.sleep(0.1)

  #Rain Sesnor Data
  rain_data = GPIO.input(Rain_sensor)

  #Pressure Sensor Data
  (chip_id, chip_version) = readBmp180Id()
  pressure=readBmp180()
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  lcd_string("Pressure Value ",LCD_LINE_1)
  lcd_string(str(pressure),LCD_LINE_2)
  time.sleep(0.1)

  #Send data on thing speak server
  thingspeak_post(temp,moisture_level,pressure,rain_data)

  pass
# Command line execution
if __name__== '__main__' :
   main()
```

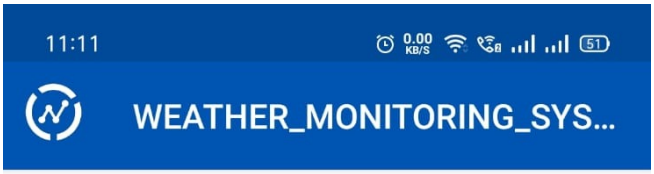
Results Obtained:
(On the ThingSpeak website)

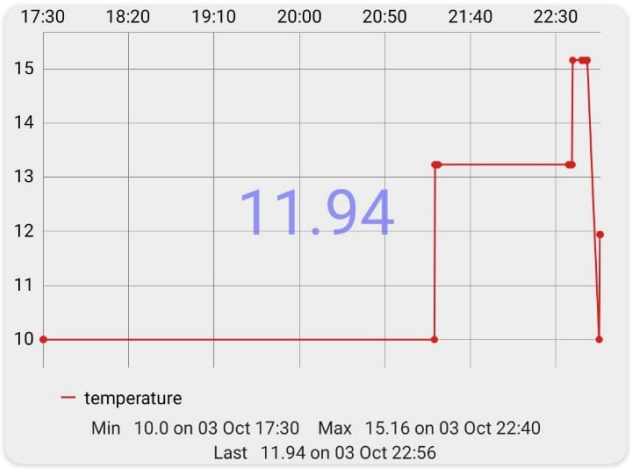


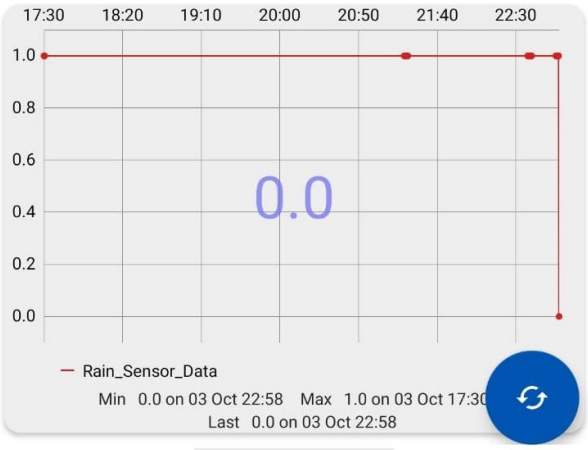




(On the ThingSpeak Mobile Application)







Chapter 4: Conclusion

The weather monitoring system built is meant to be cost effective and economical even for small, remote, and backward areas of the country. The implementation of weather monitoring system using Raspberry Pi is done as per the specifications above and the data insights are generated in web-based portal. The access to this data is available in the intranet with the current level of implementation and it could be made public when the data is made to store in cloud servers or other sources on the internet. This proposed system is one of the most compact unit for measuring weather parameters. This device in multiple nodes can be connected to the internet from various locations of study. This connectivity will aid the user to monitor the weather metrics over an IOT platform from the cloud.

Chapter 5: References

I. https://www.researchgate.net/profile/Palaniappan-Thillainathan/publication/328782869_IoT_Based_Weather_Monitoring_System/links/5be29902299bf1124fc079a4/IoT-Based-Weather-Monitoring-System.pdf
II. https://www.researchgate.net/profile/Ferdin-Joe-John-Joseph/publication/332799164_IoT_Based_Weather_Monitoring_System_for_Effective_Analytics/links/5cca497892851c8d2213fd91/IoT-Based-Weather-Monitoring-System-for-Effective-Analytics.pdf
III. https://developer.android.com/things/hardware/raspberrypi
IV. http://ijcrme.rdmodernresearch.com/wp-content/uploads/2015/06/CP-040.pdf
V. https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf
VI. https://www.analog.com/media/en/technical-documentation/data-sheets/AD9461.pdf
VII. https://datasheetspdf.com/datasheet/LM016L.html#:~:text=Description-,LCD,44H%20x%20120%20(max.)
VIII. https://www2.ece.ohio-state.edu/~passino/LM35.pdf
IX. https://www.mathworks.com/help/thingspeak/
X. https://circuitdigest.com/microcontroller-projects/raspberry-pi-iot-weather-station-to-monitor-temperature-humidity-pressure