Arduino Sketch:

```
// Define the LED pin
int ledPin = 13;
// Setup function, runs once when the sketch starts
void setup() {
// Initialize the digital pin as an output
pinMode(ledPin, OUTPUT);
}
// Loop function, runs repeatedly
void loop() {
// Turn the LED on (HIGH)
digitalWrite(ledPin, HIGH);
// Wait for 0.1 second
delay(100);
// Turn the LED off (LOW)
digitalWrite(ledPin, LOW);
// Wait for another 0.1 second
delay(100);
}
```

Arduino Sketch:

```
#include<SoftwareSerial.h>
SoftwareSerial sim8001(0,1);
#define button1 7
bool button_state;
void setup()
{
pinMode(button1,INPUT_PULLUP);
sim8001.begin(9600);
Serial.begin(9600);
delay(100);
}
void loop()
{
button_state=digitalRead(button1);
if(button_state==LOW)
{
Serial.println("Button pressed");
delay(200);
SendSMS();
}
if(sim8001.available())
Serial.write(sim8001.read());
}
void SendSMS()
{
Serial.println("Sending SMS...");
sim8001.print("AT+CMGF=1\r");
delay(100);
sim8001.print("AT+CMGS=\"+9865533668\"\r");
delay(100);
sim8001.print("SIM8001 is working");
delay(100);
sim8001.pr1awA2A34`int((char)26);
delay(100);
sim8001.println();
Serial.println("Text Sent.");
delay(400);
}
```

Arduino Sketch:

```
char inputByte;
void setup() {
Serial.begin(9600);
pinMode(13,OUTPUT);
}
void loop() {
while(Serial.available()>0){
inputByte= Serial.read();
Serial.println(inputByte);
if (inputByte=='1'){
digitalWrite(13,HIGH);
}
else if (inputByte=='0'){
digitalWrite(13,LOW);
}
}
}
```

Program

```
void setup() {
Serial.begin(9600); // Set the baud rate to 9600
}
void loop() {
// Send data to XBee module
Serial.println("Hello XBee!");
// Receive data from XBee module
if (Serial.available() > 0) {
String receivedData = Serial.readString();
Serial.println("Received data: " + receivedData);
}
delay(1000); // Delay for 1 second
```

Program:

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
LED_Red = 7
LED_Yellow= 11
GPIO.setup(LED_Red, GPIO.OUT)
GPIO.setup(LED_Yellow, GPIO.OUT)
while 1:
GPIO.output(LED_Red, True)
time.sleep(0.2)
GPIO.output(LED_Yellow, True)
time.sleep(.1)
GPIO.output(LED_Red, False)
time.sleep(.1)
GPIO.output(LED_Yellow, False)
time.sleep(.1)
```

Program:

```
#!/usr/bin/python
import spidev
import time
import os
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)
# Define GPIO to LCD mapping
LCD_RS = 15
LCD_E = 16
LCD_D4 = 7
LCD_D5 = 11
LCD_D6 = 12
LCD_D7 = 13
# Define sensor channels
temp_channel = 0
'''
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
# 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
temp = ((data * 330)/float(1023))
temp = round(temp,places)
return temp
# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(.2)
while 1:
temp_level = ReadChannel(temp_channel)
temp = ConvertTemp(temp_level,2)
# Print out results
lcd_string("Temperature ",LCD_LINE_1)
lcd_string(str(temp),LCD_LINE_2)
time.sleep(.1)
```

PROGRAM:

```
#!/usr/bin/python
import time
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
'''
define pin for lcd
'''
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005
delay = 1
buzzer=37
GPIO.setup(buzzer, GPIO.OUT)
# Define GPIO to LCD mapping
LCD_RS = 7
LCD_E = 11
LCD_D4 = 12
LCD_D5 = 13
LCD_D6 = 15
LCD_D7 = 16
IR_Sensor = 18
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(IR_Sensor, GPIO.IN) # DB7
# 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)
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(2)
# Define delay between readings
delay = 5
```

```
while 1:
# Print out results
if GPIO.input(IR_Sensor):
lcd_string("Obstacle Detected ",LCD_LINE_1)
time.sleep(1)
else:
lcd_string("Obstacle Removed ",LCD_LINE_1)
time.sleep(1)
```

PROGRAM:

```
# !/usr/bin/env python3
# Modules
from goto import *
import time
import var
import pio
import resource
import spidev
import RPi.GPIO as GPIO
import urllib.request
import requests
# 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):
temp = ((data * 330)/float(1023))
temp = round(temp,places)
return temp
def thingspeak_post(temp,moisture_level,motor_status,rain_data):
URl='https://api.thingspeak.com/update?api_key='
#Enter Your Private Key here
```

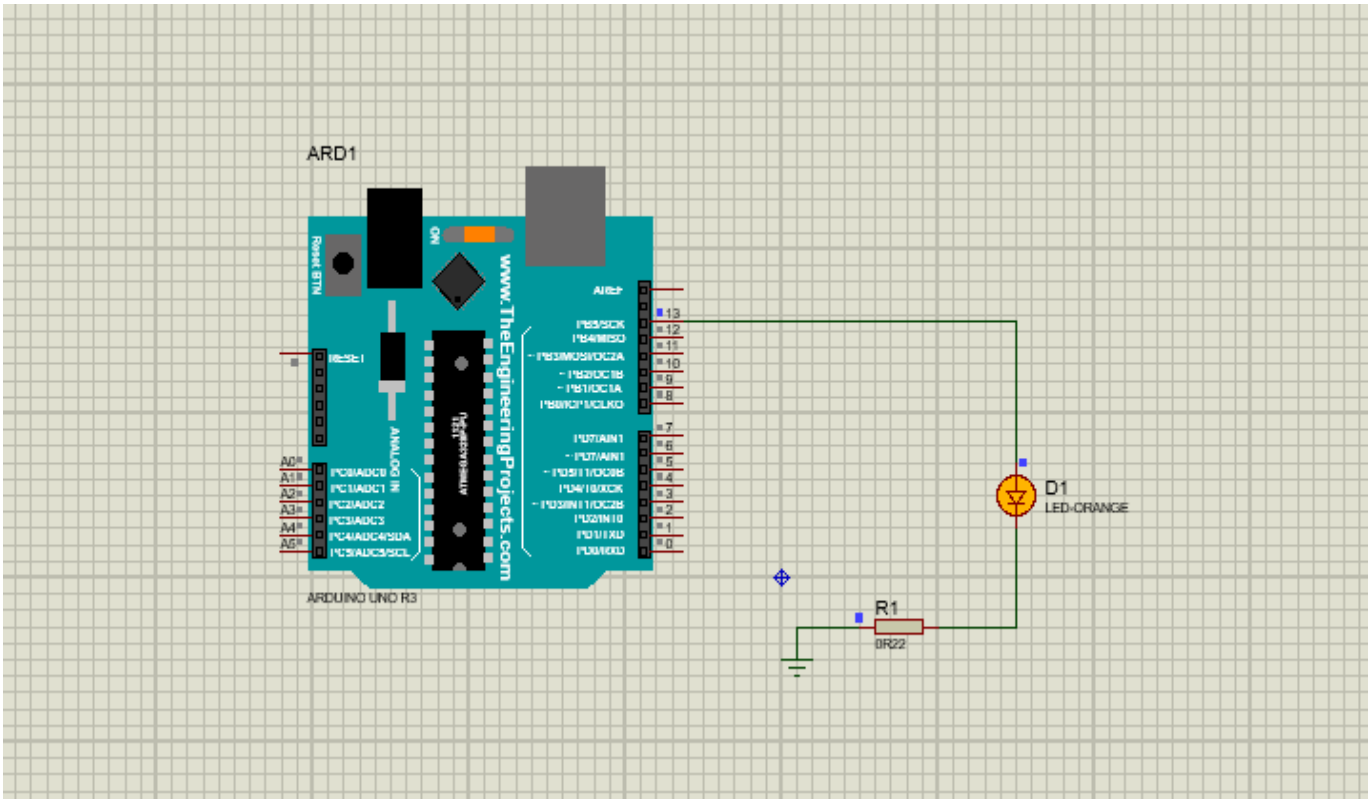
```
KEY='24AOZ5TLM9UHE5BO'
HEADER='&field1={}&field2={}&field3={}&field4={}'.format(temp,moisture_l
evel,motor_st
atus,rain_data)
NEW_URL=URl+KEY+HEADER
print(NEW_URL)
data=urllib.request.urlopen(NEW_URL)
print(data)
# Define delay between readings
delay = 5
lcd_init()
lcd_string("welcome ",LCD_LINE_1)
time.sleep(1)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
lcd_string("Smart Irrigation",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(0.5)
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(0.5)
```

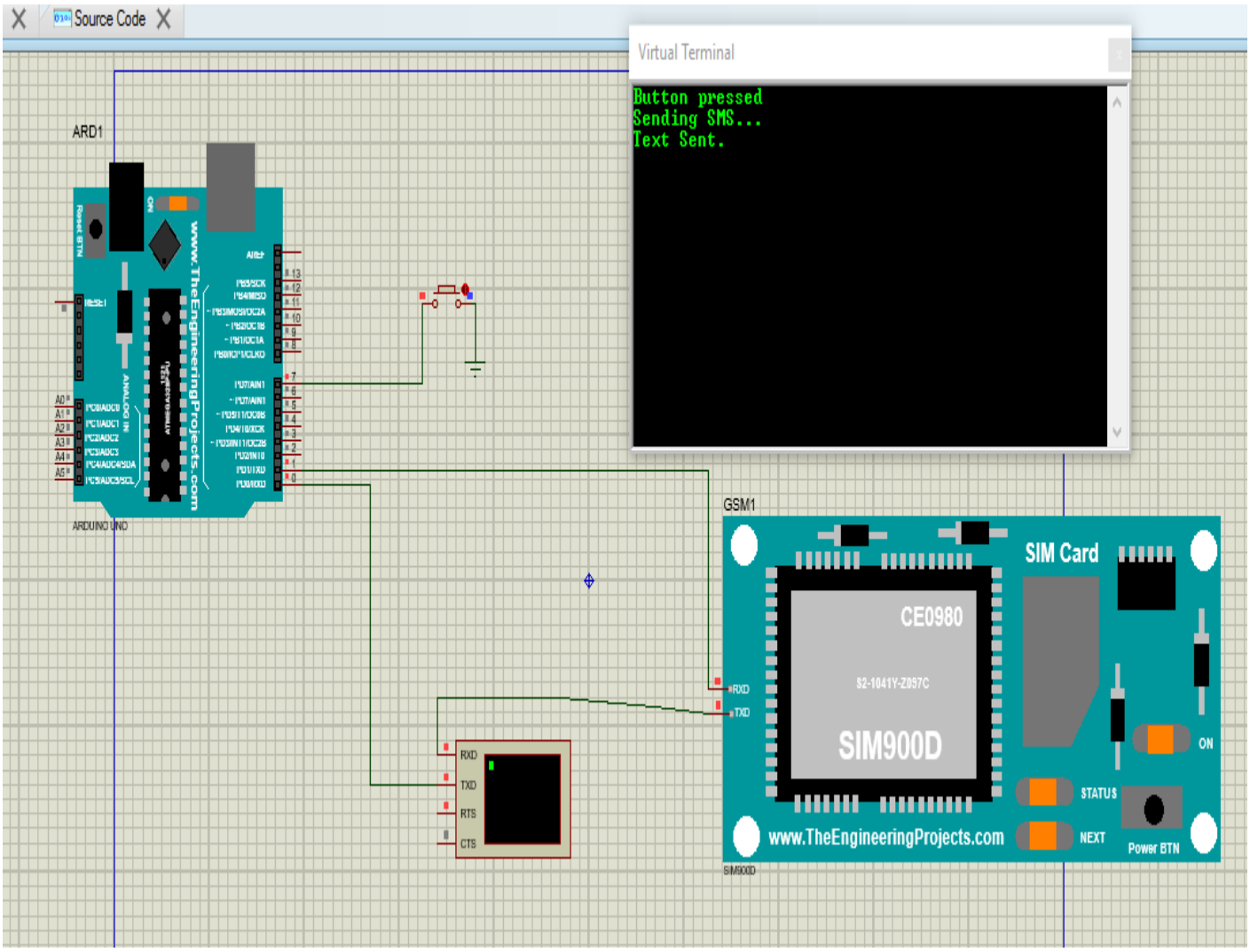
```
rain_data = GPIO.input(Rain_sensor)
#Send data on thing speak server
thingspeak_post(temp,moisture_level,motor_status,rain_data)
if((temp > 25) and (moisture_level < 100) and (rain_data != True)) :
GPIO.output(Relay_pin, True)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
lcd_string("Motor Start ",LCD_LINE_1)
pio.uart.println("AT")
pio.uart.println("AT+CMGF=1")
pio.uart.println("AT+CMGS=\"+919865533668\"\r")
pio.uart.println("Motor Started")
motor_status = 1
time.sleep(0.5)
else:
GPIO.output(Relay_pin, False)
lcd_byte(0x01,LCD_CMD) # 000001 Clear display
lcd_string("Motor Stop ",LCD_LINE_1)
if(rain_data == True):
lcd_string("Rain Detected ",LCD_LINE_2)
pio.uart.println("AT")
pio.uart.println("AT+CMGF=1")
pio.uart.println("AT+CMGS=\"+919865533668\"\r")
pio.uart.println("Motor Stop")
motor_status = 0
time.sleep(0.5)
pass
# Command line execution
if __name__ == '__main__' :
main()
```

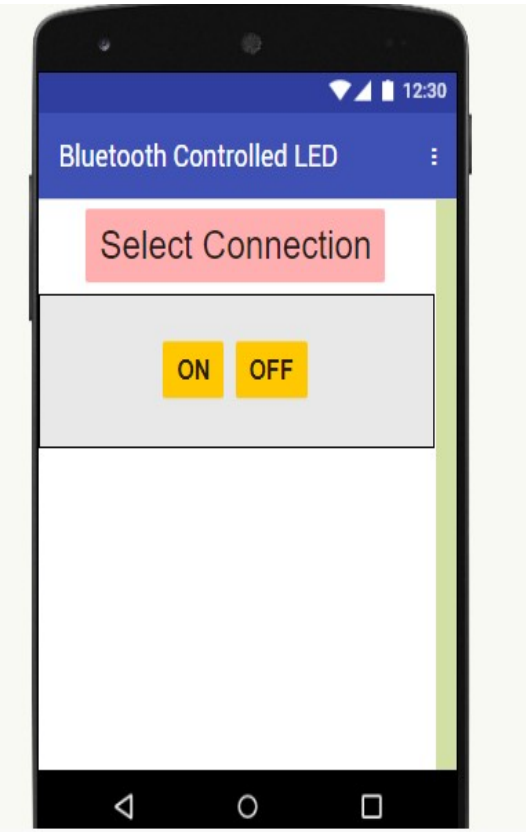
Program:

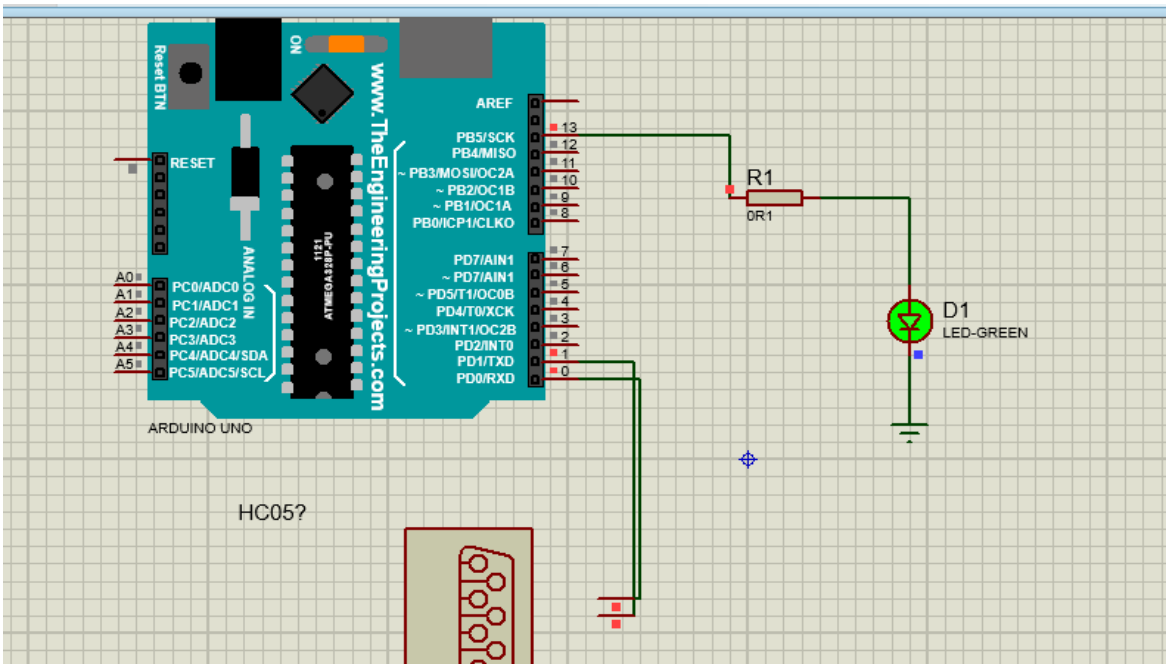
```
# Main.py file generated by New Project wizard
from goto import *
import time
import var
import pio
import resource
import spidev
import RPi.GPIO as GPIO
import urllib.request
import requests
# Peripheral Configuration Code (do not edit)
#---CONFIG_BEGIN---
import cpu
import FileStore
import VFP
def peripheral_setup () :
# Peripheral Constructors
pio.cpu=cpu.CPU ()
pio.storage=FileStore.FileStore ()
pio.server=VFP.VfpServer ()
pio.storage.begin ()
pio.server.begin (0)
# Install interrupt handlers
def peripheral_loop () :
pio.server.poll ()
#---CONFIG_END---
# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)
def thingspeak_post(temp):
URl='https://api.thingspeak.com/update?api_key='
#Enter Your Private Key here
KEY='24AOZ5TLM9UHE5BO'
HEADER='&field1={}'.format(temp)
NEW_URL=URl+KEY+HEADER
print(NEW_URL)
data=urllib.request.urlopen(NEW_URL)
print(data)
# Define delay between readings
# Main function
def main () :
# Setup
peripheral_setup()
peripheral_loop()
# Infinite loop
while 1 :
temp = 32
```

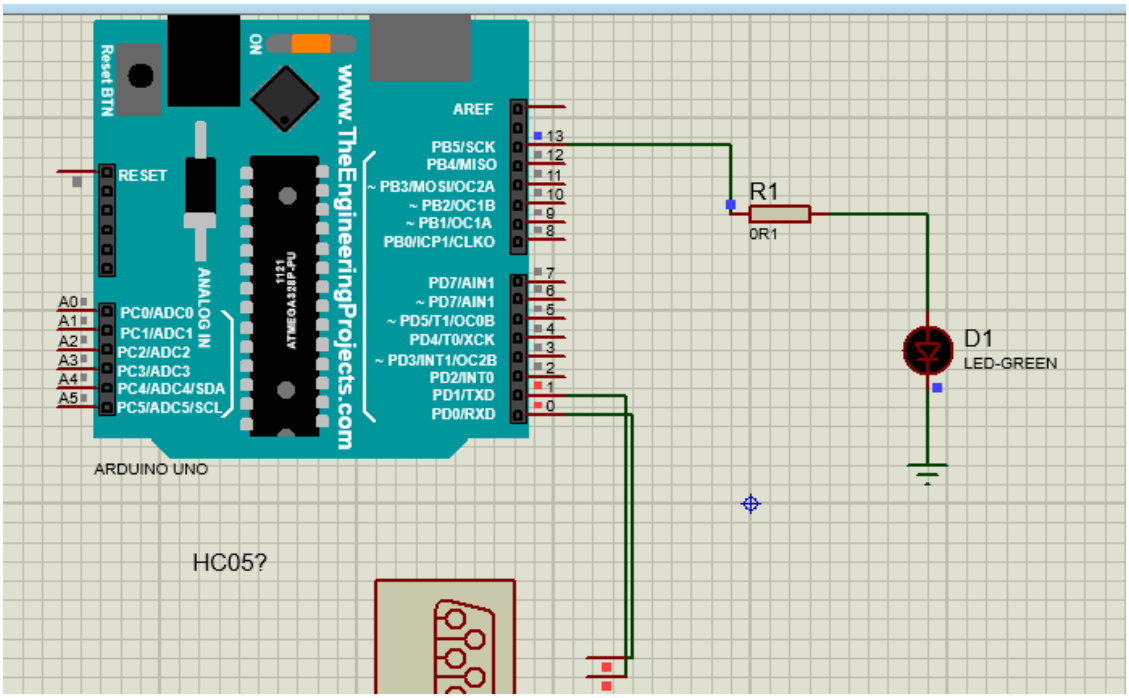
```
#Send data on thing speak server
thingspeak_post(temp)
pass
# Command line execution
if __name__ == '__main__' :
main()
```

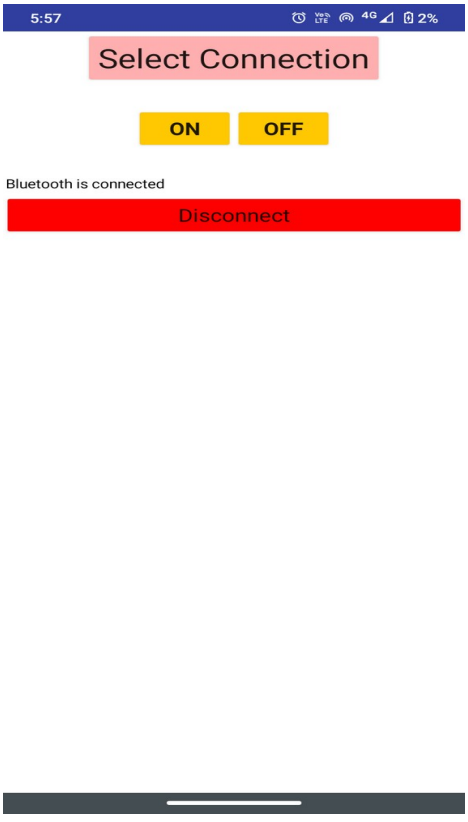

ARD1
Reset BTN
ON
www.TheEngineeringProjects.com
ANALOG IN
ARDUINO UNO R3
D1
LED-ORANGE
R1
0R22

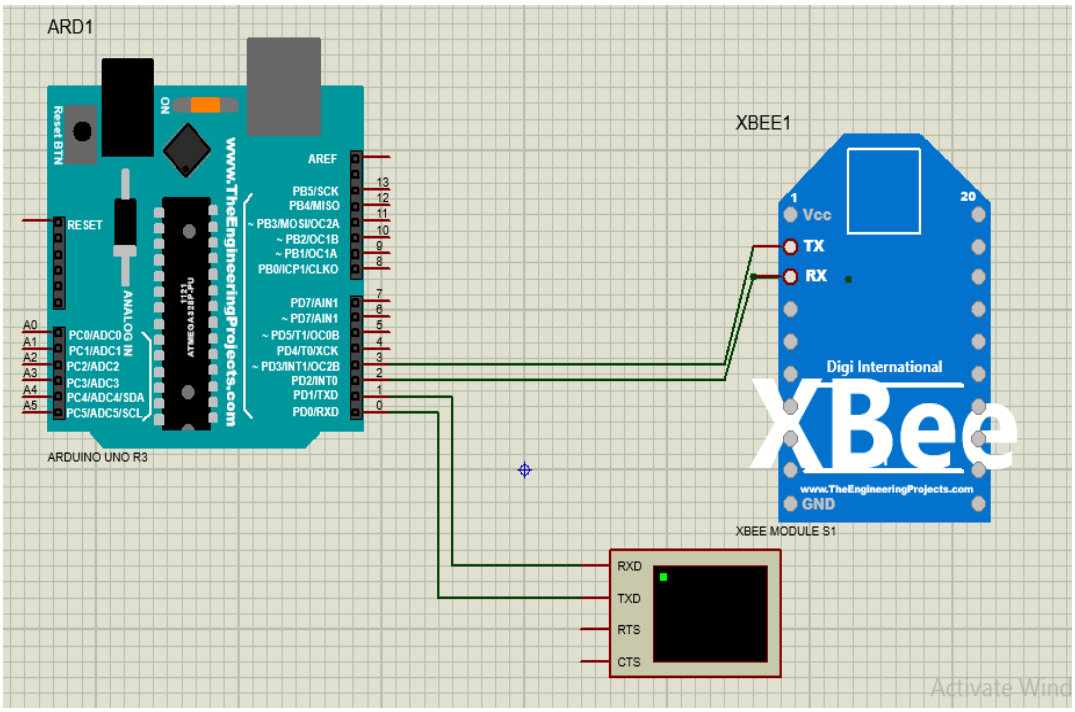

Source Code
Virtual Terminal
Button pressed
Sending SMS...
Text Sent.
ARD1
ARDUINO UNO
www.TheEngineeringProjects.com
GSM1
SIM Card
CE0980
S2-1041Y-Z097C
SIM900D
RXD
TXD
ON
STATUS
NEXT
Power BTN
www.TheEngineeringProjects.com
RXD
TXD
RTS
CTS

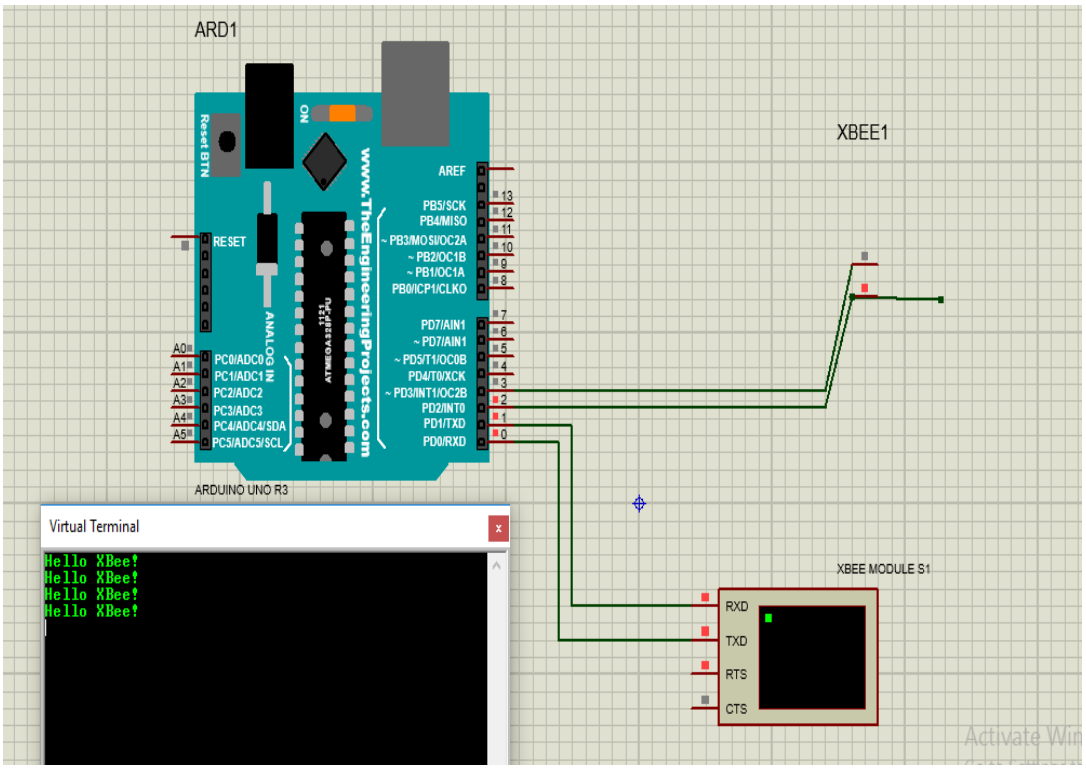

12:30
Bluetooth Controlled LED
Select Connection
ON
OFF

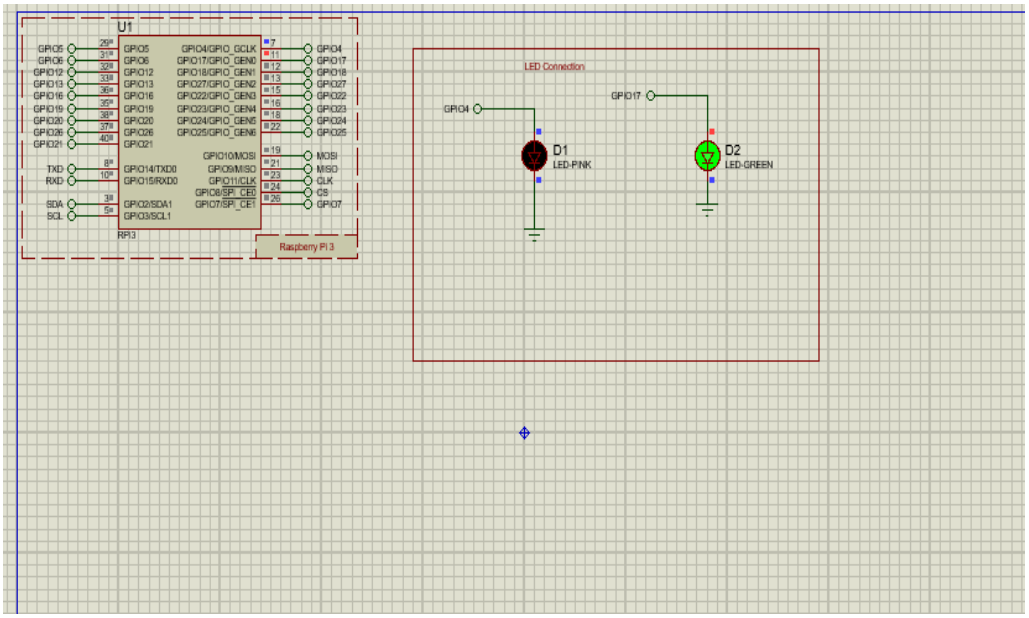

Reset BTN
ON
RESET
ANALOG IN
www.TheEngineeringProjects.com
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ARDUINO UNO
R1
0R1
D1
LED-GREEN
HC05?

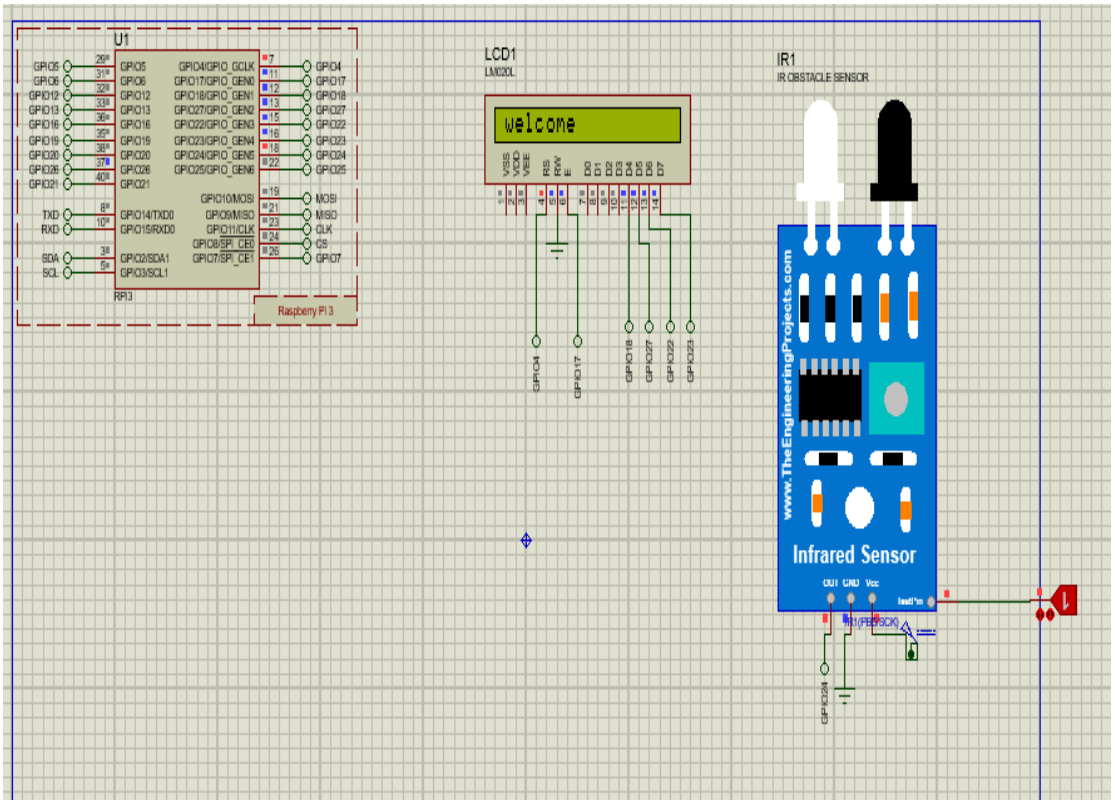

Reset BTN
ON
www.TheEngineeringProjects.com
AREF
RESET
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
13
12
11
10
9
8
R1
0R1
ANALOG IN
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
ATMEGA328P-PU
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
7
6
5
4
3
2
1
0
D1
LED-GREEN
ARDUINO UNO
HC05?

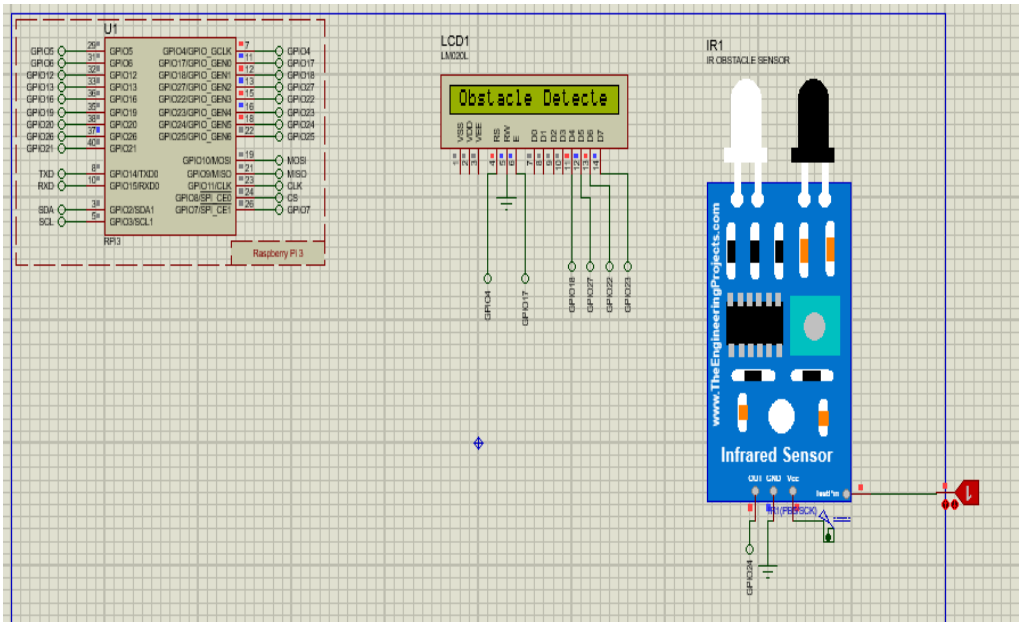

5:57
2%
Select Connection
ON
OFF
Bluetooth is connected
Disconnect

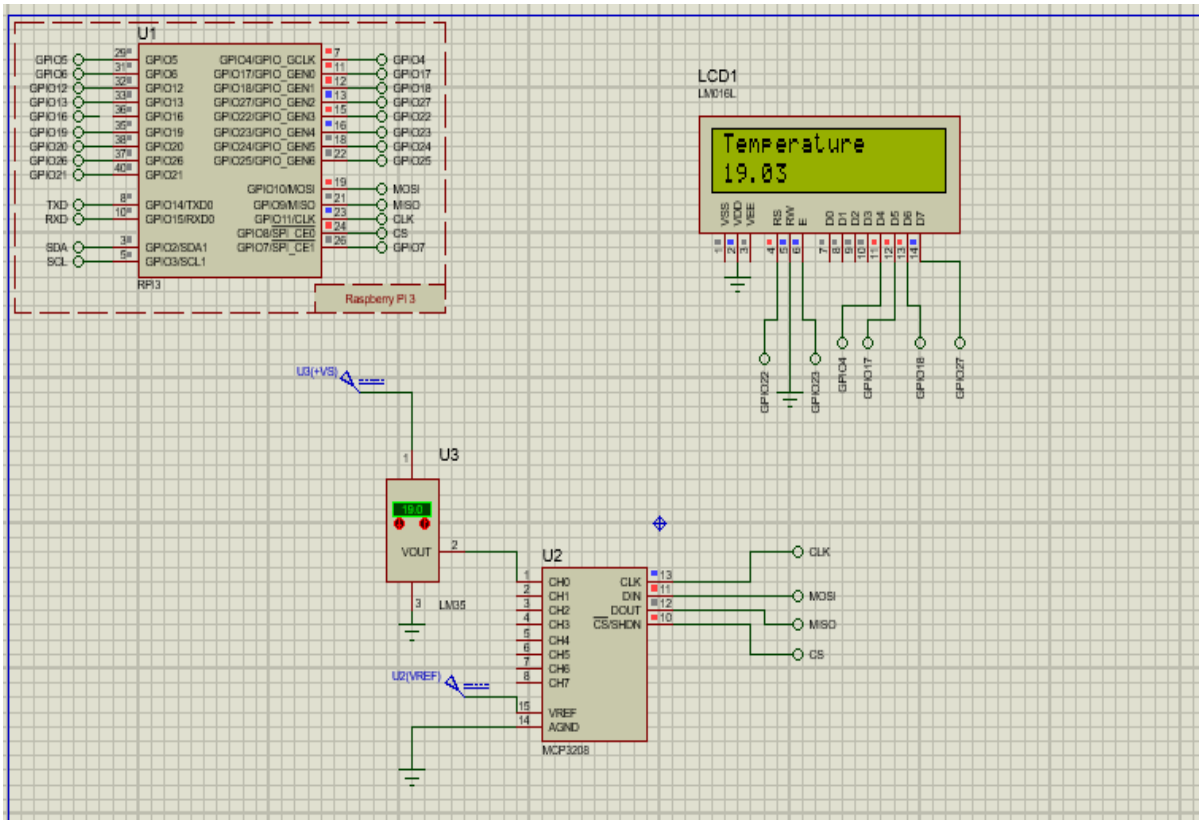

ARD1
Reset BTN
ON
RESET
ANALOG IN
AREF
PB5/SCK
PB4/MISO
~ PB3/MOSI/OC2A
~ PB2/OC1B
~ PB1/OC1A
PB0/ICP1/CLKO
PD7/AIN1
~ PD7/AIN1
~ PD5/T1/OC0B
PD4/T0/XCK
~ PD3/INT1/OC2B
PD2/INT0
PD1/TXD
PD0/RXD
A0
A1
A2
A3
A4
A5
PC0/ADC0
PC1/ADC1
PC2/ADC2
PC3/ADC3
PC4/ADC4/SDA
PC5/ADC5/SCL
www.TheEngineeringProjects.com
ARDUINO UNO R3
XBEE1
1
20
Vcc
TX
RX
GND
Digi International
XBee
www.TheEngineeringProjects.com
XBEE MODULE S1
RXD
TXD
RTS
CTS

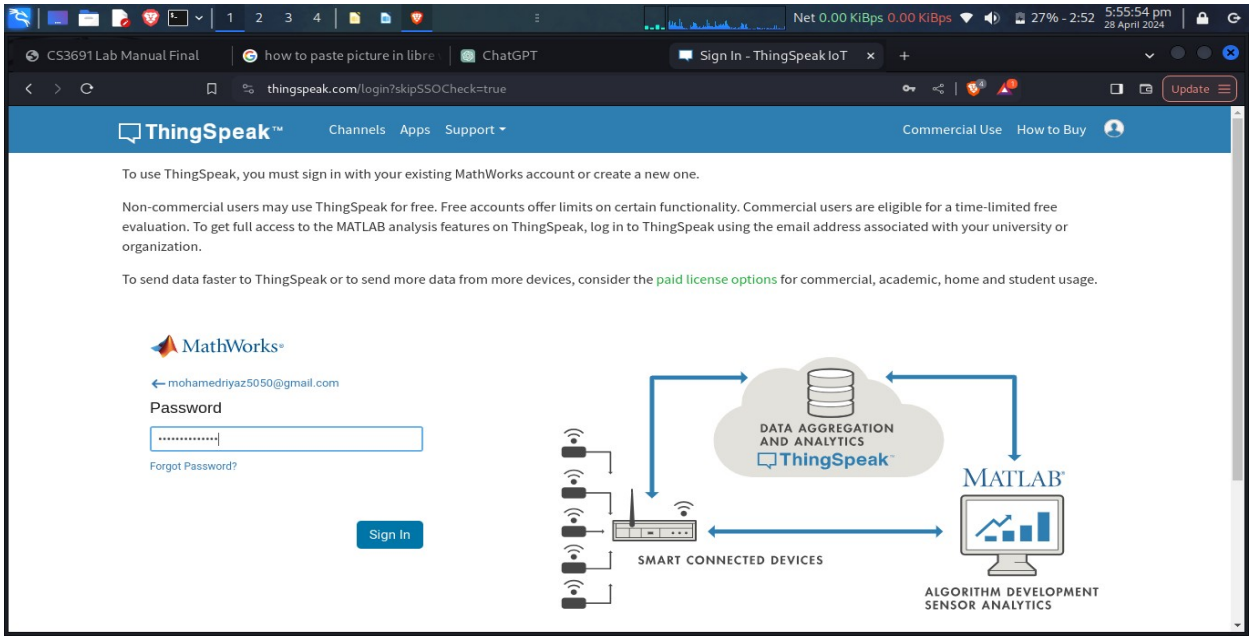

ARD1
XBEE1
ARDUINO UNO R3
Virtual Terminal
Hello XBee!
Hello XBee!
Hello XBee!
Hello XBee!
XBEE MODULE S1
RXD
TXD
RTS
CTS

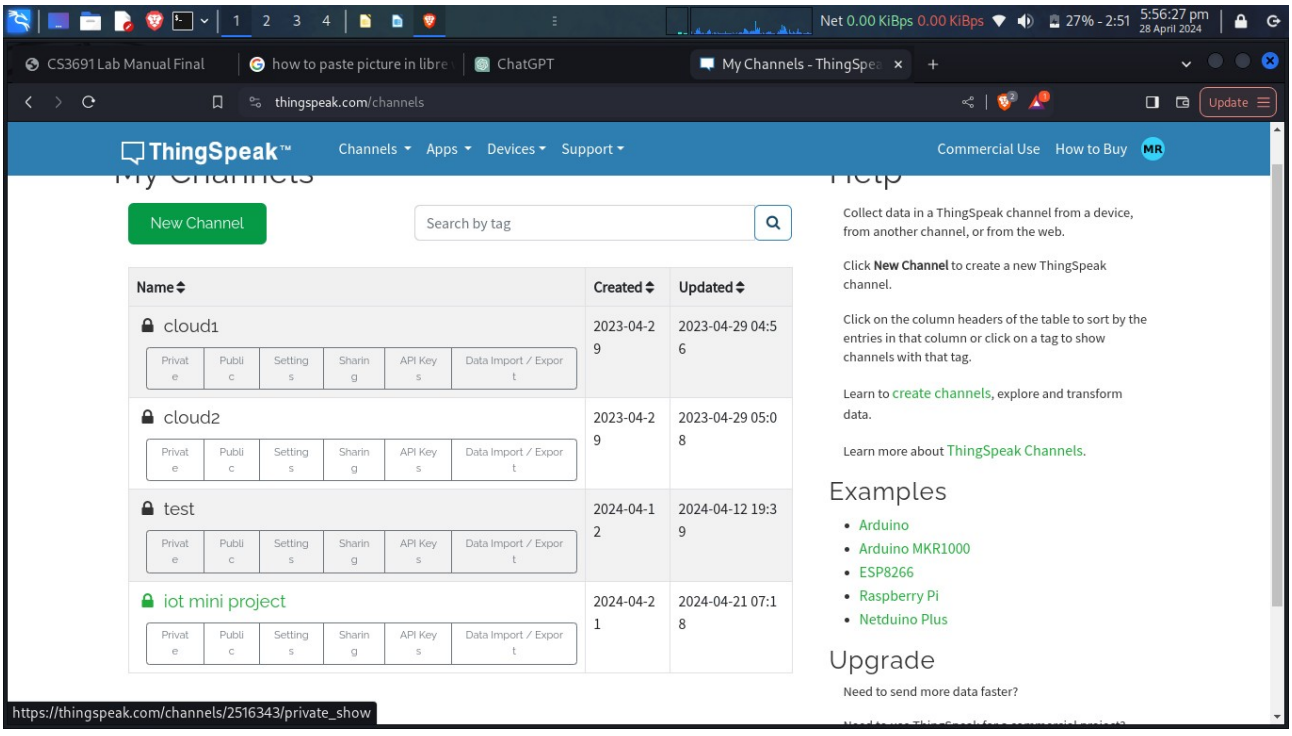

U1
GPIO5
GPIO6
GPIO12
GPIO13
GPIO16
GPIO19
GPIO20
GPIO26
GPIO21
TXD
RXD
SDA
SCL
GPIO4/GPIO_GCLK
GPIO17/GPIO_GEN0
GPIO18/GPIO_GEN1
GPIO27/GPIO_GEN2
GPIO22/GPIO_GEN3
GPIO23/GPIO_GEN4
GPIO24/GPIO_GEN5
GPIO25/GPIO_GEN6
GPIO10/MOSI
GPIO9/MISO
GPIO11/CLK
GPIO8/SPI_CE0
GPIO7/SPI_CE1
GPIO14/TXD0
GPIO15/RXD0
GPIO2/SDA1
GPIO3/SCL1
MOSI
MISO
CLK
CS
RP3
Raspberry Pi 3
LED Connection
GPIO4
GPIO17
D1
LED-PINK
D2
LED-GREEN


U1
LCD1
LM020L
welcome
IR1
IR OBSTACLE SENSOR
www.TheEngineeringProjects.com
Infrared Sensor
OUT GND Vcc
RP3
Raspberry Pi 3


U1
LCD1
LM020L
Obstacle Detecte
IR1
IR OBSTACLE SENSOR
www.TheEngineeringProjects.com
Infrared Sensor
OUT GND Vcc
RP3
Raspberry Pi 3


U1
RPI3
Raspberry PI 3
LCD1
LM016L
Temperature
19.03
U3(+VS)
U3
VOUT
LM35
U2(VREF)
U2
MCP3208
CLK
MOSI
MISO
CS


Net 0.00 KiBps 0.00 KiBps
27% - 2:52
5:55:54 pm
28 April 2024
CS3691 Lab Manual Final
how to paste picture in libre
ChatGPT
Sign In - ThingSpeak IoT
thingspeak.com/login?skipSSOCheck=true
Update
ThingSpeak™
Channels
Apps
Support
Commercial Use
How to Buy
To use ThingSpeak, you must sign in with your existing MathWorks account or create a new one.
Non-commercial users may use ThingSpeak for free. Free accounts offer limits on certain functionality. Commercial users are eligible for a time-limited free evaluation. To get full access to the MATLAB analysis features on ThingSpeak, log in to ThingSpeak using the email address associated with your university or organization.
To send data faster to ThingSpeak or to send more data from more devices, consider the paid license options for commercial, academic, home and student usage.
MathWorks
mohamedriyaz5050@gmail.com
Password
Forgot Password?
Sign In
DATA AGGREGATION AND ANALYTICS
ThingSpeak
MATLAB
SMART CONNECTED DEVICES
ALGORITHM DEVELOPMENT
SENSOR ANALYTICS


Net 0.00 KiBps 0.00 KiBps
27% - 2:51
5:56:27 pm
28 April 2024
CS3691 Lab Manual Final
how to paste picture in libre
ChatGPT
My Channels - ThingSpea
thingspeak.com/channels
Update
ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
MR
New Channel
Search by tag
Name
Created
Updated
cloud1
2023-04-29
2023-04-29 04:56
cloud2
2023-04-29
2023-04-29 05:08
test
2024-04-12
2024-04-12 19:39
iot mini project
2024-04-21
2024-04-21 07:18
Private
Public
Settings
Sharing
API Keys
Data Import / Export
Help
Collect data in a ThingSpeak channel from a device, from another channel, or from the web.
Click New Channel to create a new ThingSpeak channel.
Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.
Learn to create channels, explore and transform data.
Learn more about ThingSpeak Channels.
Examples
Arduino
Arduino MKR1000
ESP8266
Raspberry Pi
Netduino Plus
Upgrade
Need to send more data faster?
https://thingspeak.com/channels/2516343/private_show

ThingSpeak™ Channels Apps Devices Support Commercial Use How to Buy MR

Channel ID: **2516343**
Author: mwa0000029558987
Access: Private

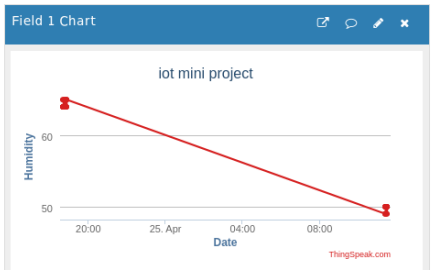
Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

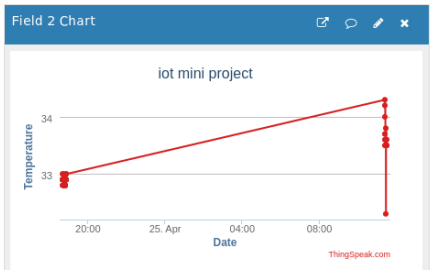
Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

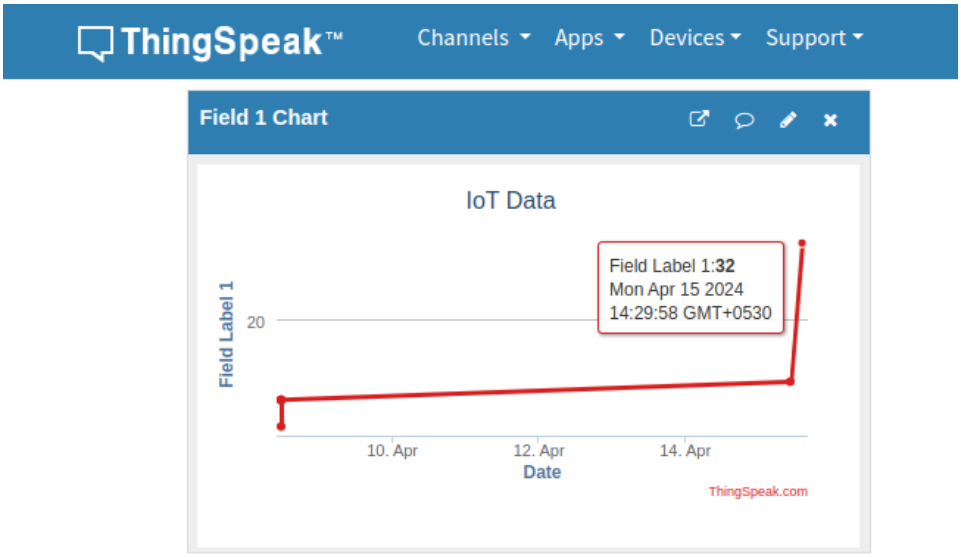
Channel 4 of 4

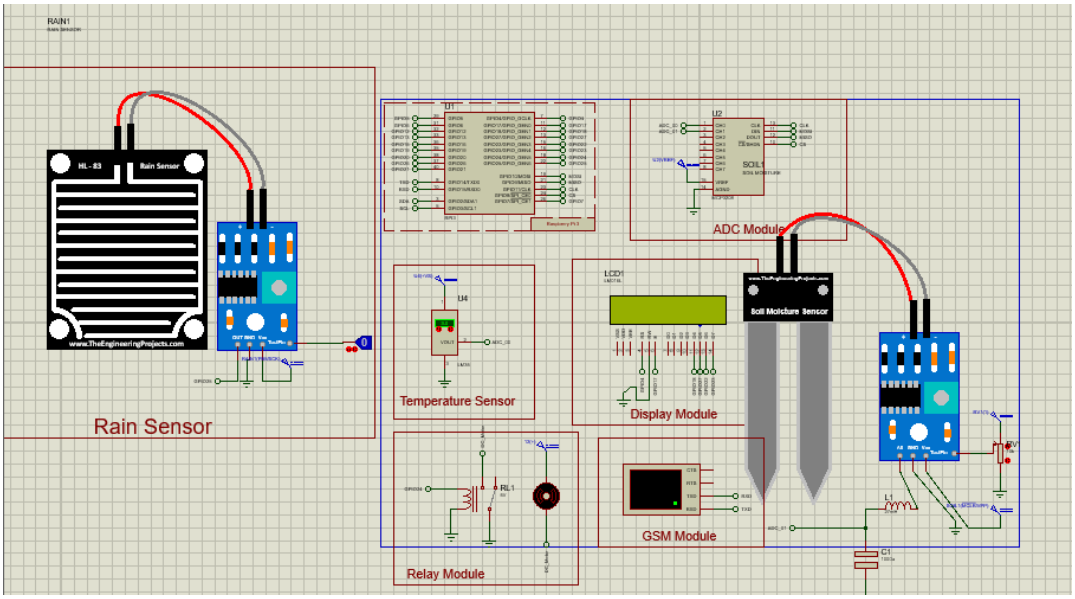
Channel Stats

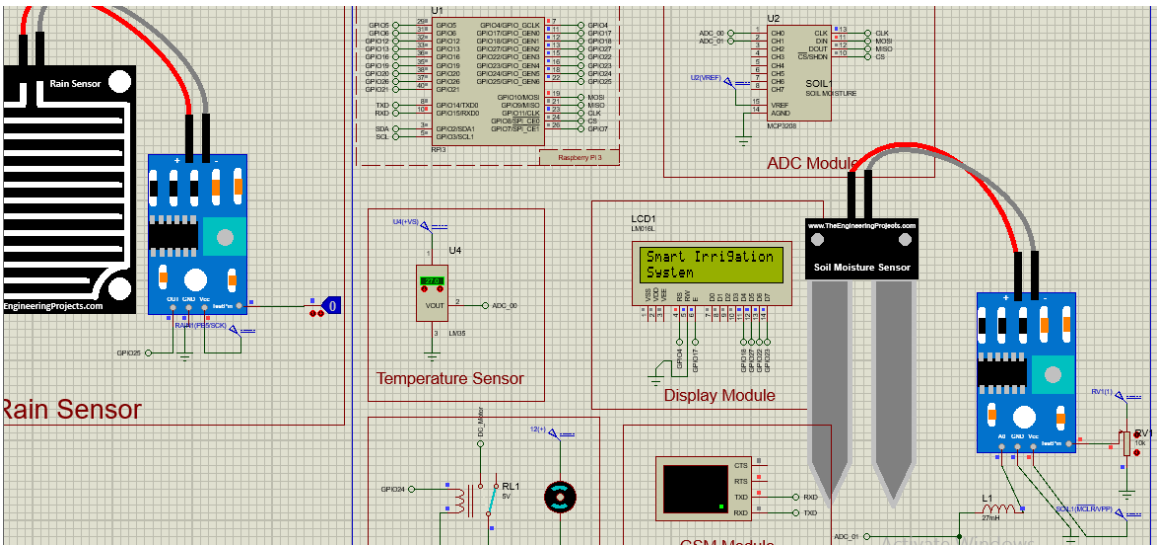
Created: 7 days ago
Last entry: 3 days ago
Entries: 254

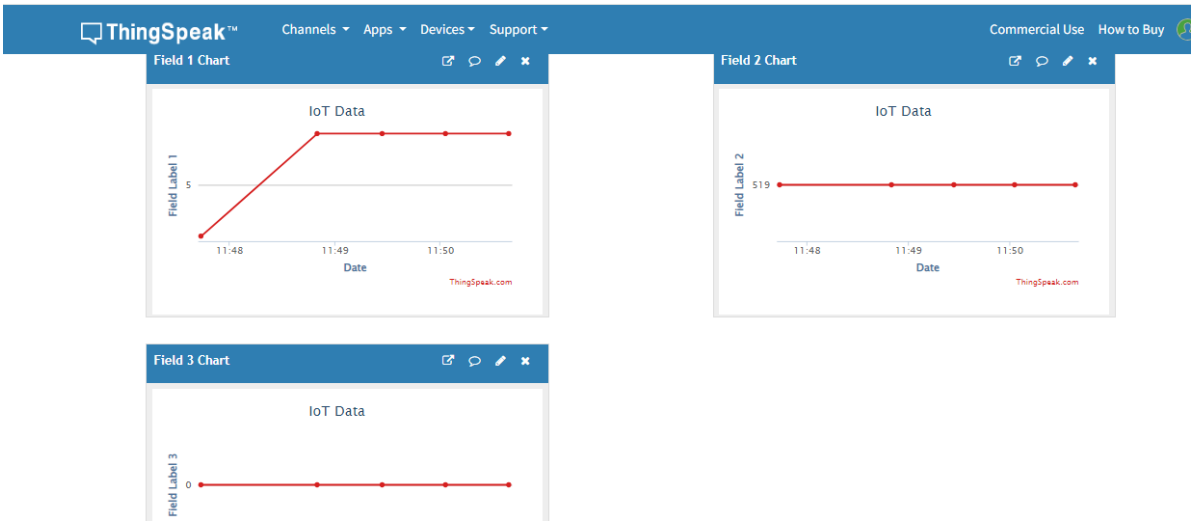





ThingSpeak™
Channels
Apps
Devices
Support
Field 1 Chart
IoT Data
Field Label 1:32
Mon Apr 15 2024
14:29:58 GMT+0530
Field Label 1
20
10. Apr
12. Apr
14. Apr
Date
ThingSpeak.com


RAIN1
HL - 83
Rain Sensor
www.TheEngineeringProjects.com
Rain Sensor
U1
U2
ADC Module
SOIL1
SOIL MOISTURE
U4
Temperature Sensor
LCD1
Display Module
Soil Moisture Sensor
RL1
Relay Module
GSM Module
L1
C1


Rain Sensor
U1
U2
ADC Module
SOIL1
SOIL MOISTURE
U4
Temperature Sensor
LCD1
Smart Irrigation
System
Display Module
www.TheEngineeringProjects.com
Soil Moisture Sensor
RL1
GSM Module
L1


ThingSpeak™
Channels
Apps
Devices
Support
Commercial Use
How to Buy
Field 1 Chart
IoT Data
Field Label 1
5
11:48
11:49
11:50
Date
ThingSpeak.com
Field 2 Chart
IoT Data
Field Label 2
519
11:48
11:49
11:50
Date
ThingSpeak.com
Field 3 Chart
IoT Data
Field Label 3
0