RAMAIAH
Institute of Technology

Department of Information Science & Technology

BE

IS45 – Microprocessor

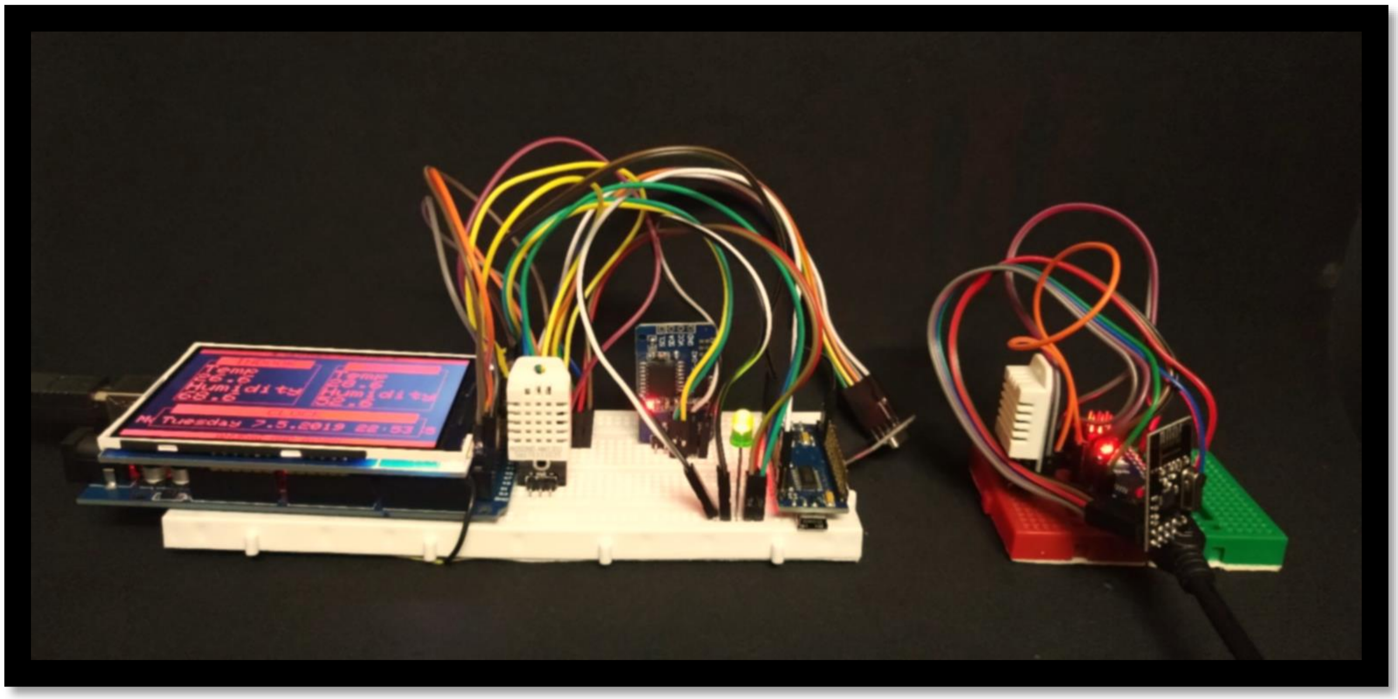
Wireless Weather Station Using Transmitter and Receiver

Submitted By: 1MS17IS145 – R. Anurag Pillai

Faculty In-Charge: George Philip C (Associate Professor)

RAMAIAH INSTITUTE OF TECHNOLOGY
(AUTONOMOUS INSTITUTE, AFFILIATED TO VTU)
Bangalore – 560054
2018-2019

Wireless Weather Station Using Transmitter and Receiver



ABSTRACT

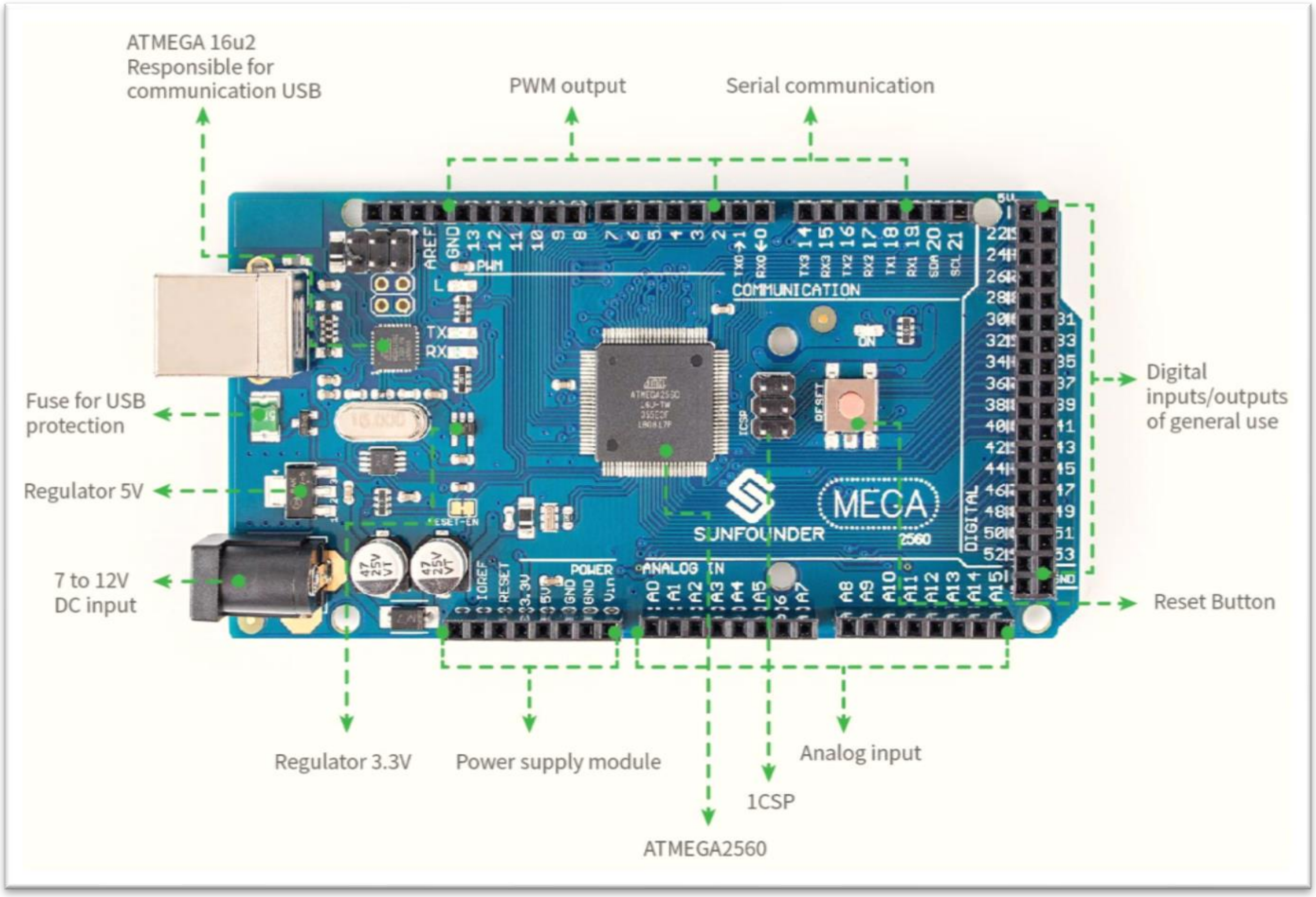
A Weather station is a device that collects data related to the weather and environment using many different sensors. We can measure many things like Temperature and Humidity along with displaying real time. In our variation, we have a transmitter station to send data from a remote location to the main station with a receiver at an indoor location. In the Weather Station that we built, we are going to measure Temperature and Humidity in two locations and display the current date and time. Building a Wireless Weather Station was a great learning experience. When we finished building this project, we have a better understanding of how wireless communications work, how sensors work, and how powerful the Arduino platform can be. With this project as a base and the experience gained, we would be able to easily build more complex projects in the future.

AIM

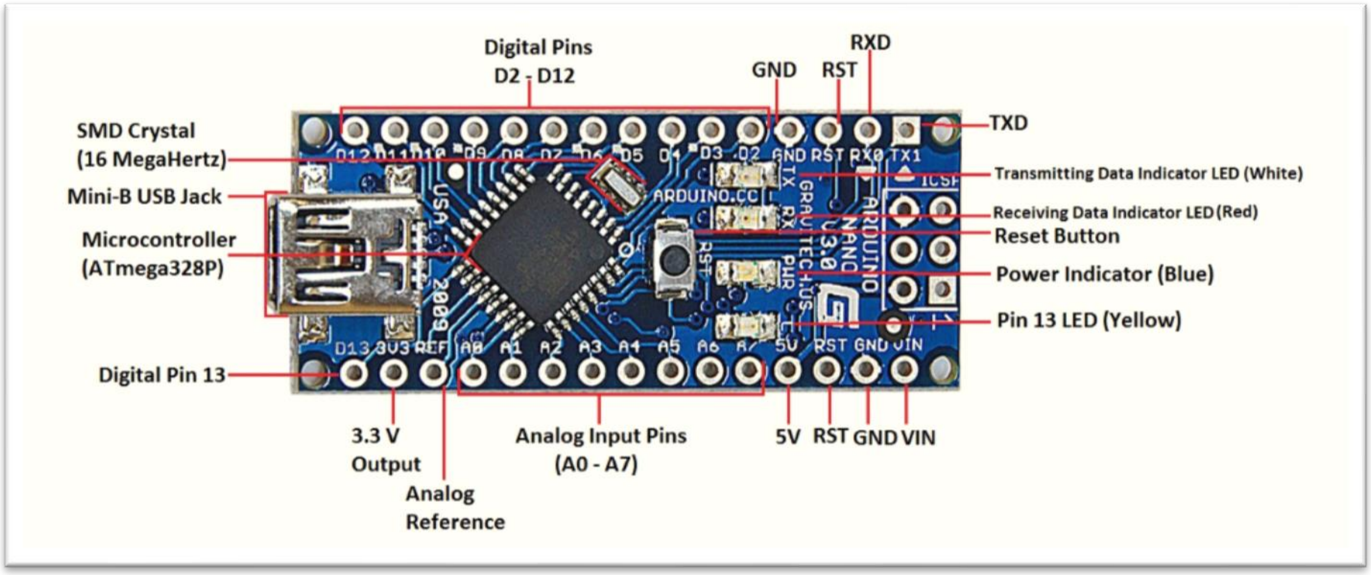
To construct wirelessly connected devices capable of transmitting and receiving data from one location to another and display it to the user at an indoor location.

DEVICES USED

Development Boards Used:



- **Arduino Mega:** Used for Receiver which is to be placed at an easily accessible location for the user to get all data such as temperature & humidity of that location as well as a remote location along with date & time, and displayed in real time.
  <u>Reasons for selecting Arduino Mega</u>:
  - ✓ *A large number of Analog and digital pins for interfacing with display shield, along with other sensors.*
  - ✓ *Easily scalable for implementing extra sensors for future upgrade.*
  - ✓ *Cheaper than Arduino Due.*
  - ✓ *Higher Performance because of the ATmega2560 processor.*

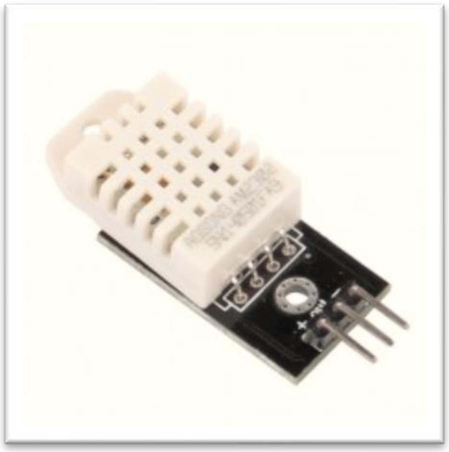


- **Arduino Nano:** Used for Transmitter which is to be placed at difficult to access location for transmitting data such as temperature & humidity of that location to the unit which is located indoor.
  Reasons for selecting Arduino Nano:
  - ✓ *Compact size and lightweight.*
  - ✓ *Easy to use, handle, and placement at a remote location.*
  - ✓ *A reasonable number of Analog and Digital pins to work with.*
  - ✓ *Really Cheap and needs smaller breadboard for all sensors to be used.*
  - ✓ *Performance comparable to Arduino Uno because of ATmega328P processor.*

Sensors & Modules Used:

- **DHT22:**
  - ✓ Temperature and Humidity Sensor.
  - ✓ Units: 2 (1-For Transmitter and 1-For Receiver).
  - ✓ Substitute available in the market: DHT11, AM2302.

  Specifications & Features:
  - ✓ *Compact size and lightweight.*
  - ✓ *Low cost.*
  - ✓ *3 to 5V power and I/O.*
  - ✓ *2.5mA max current use during conversion.*
  - ✓ *0-100% humidity readings with 2-5% accuracy.*
  - ✓ *-40 to 125°C temperature readings with ±0.5°C accuracy.*


DHT22

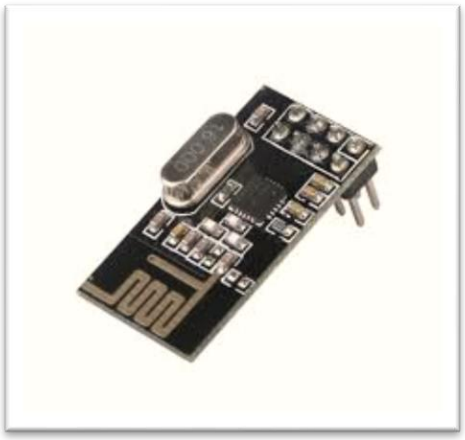
- **DS3231:**
  - ✓ Real Time Clock Module.
  - ✓ Units: 1 (1-For Receiver).
  - ✓ Have a slot for button cell, to store time even when the device is switched off.

  Specifications & Features:
  - ✓ *Real-Time Clock Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the Week, and Year, with Leap-Year Compensation Valid Up to 2100.*
  - ✓ *Accuracy ±2ppm from 0°C to +40°C.*
  - ✓ *Accuracy ±3.5ppm from -40°C to +85°C.*
  - ✓ *Digital Temp Sensor Output: ±3°C Accuracy.*

DS3231

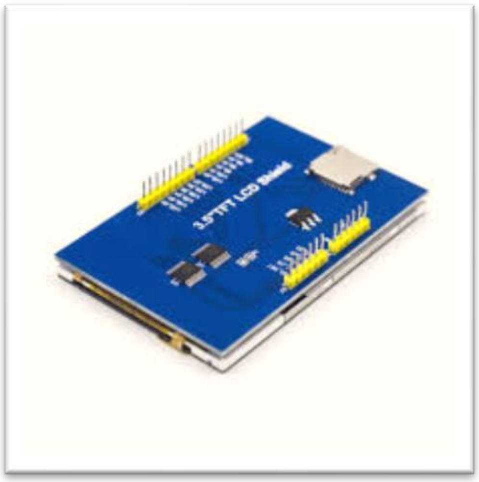
- **NRF24L01:**
  - ✓ Wireless transceiver module.
  - ✓ Units: 2 (1-For Transmitter and 1-For Receiver)
  - ✓ Works on SPI Communication via SPI pins.

  Specifications & Features:
  - ✓ *Uses the 2.4 GHz band.*
  - ✓ *Operate with baud rates from 250 kbps up to 2 Mbps.*
  - ✓ *Ranges up to 100 meters.*
  - ✓ *Up to 125 different channels with 6 addresses per channel.*


NRF24L01

Other Components and Tools Used:

- 1 - Large Breadboard
- 2 - Small Breadboards
- 1 - ILI9586 Display Shield (3.5" TFT)
- 1 - LED
- 1 - Arduino Nano (To fulfill last minute requirement for Buck-Convertor/Logic-Level-Convertor)
- 1 - USB 2.0 Type A Male to Type B Male cable.
- 1 - USB 2.0 Type A Male to Type B Mini Male cable.
- Few Jumper Wires (M-M, M-F, F-F)

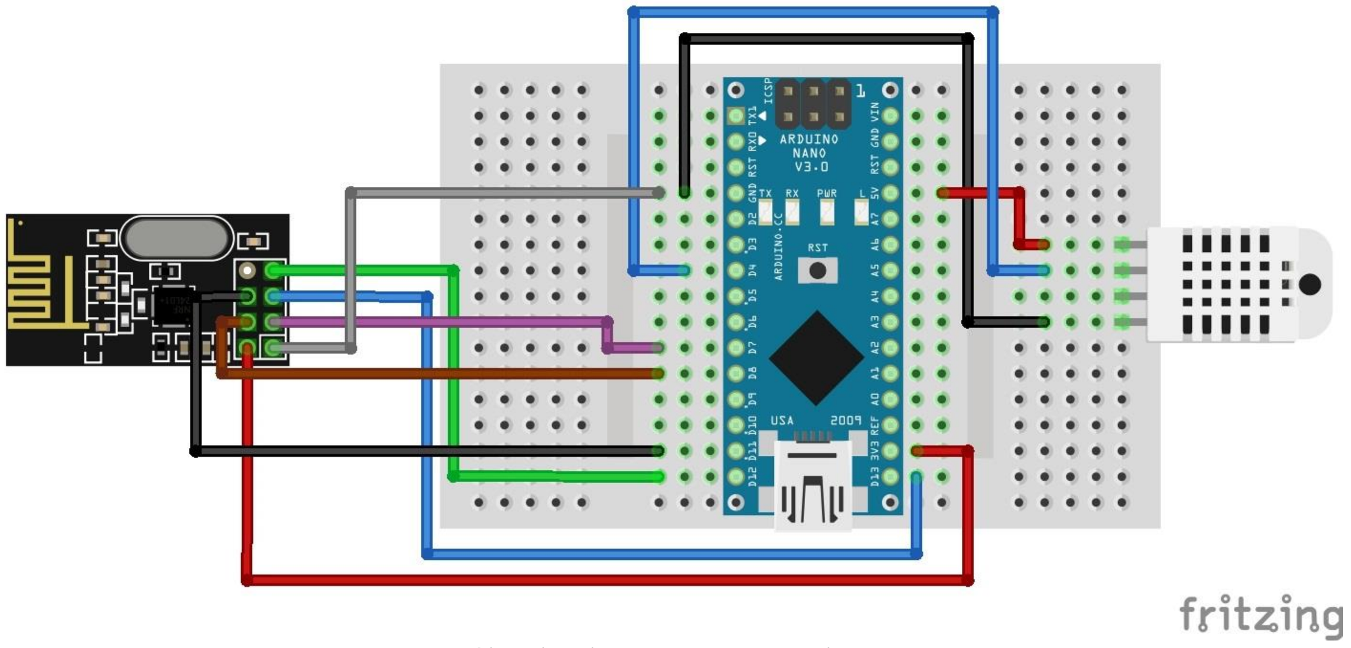


ILI9586 Display Shield

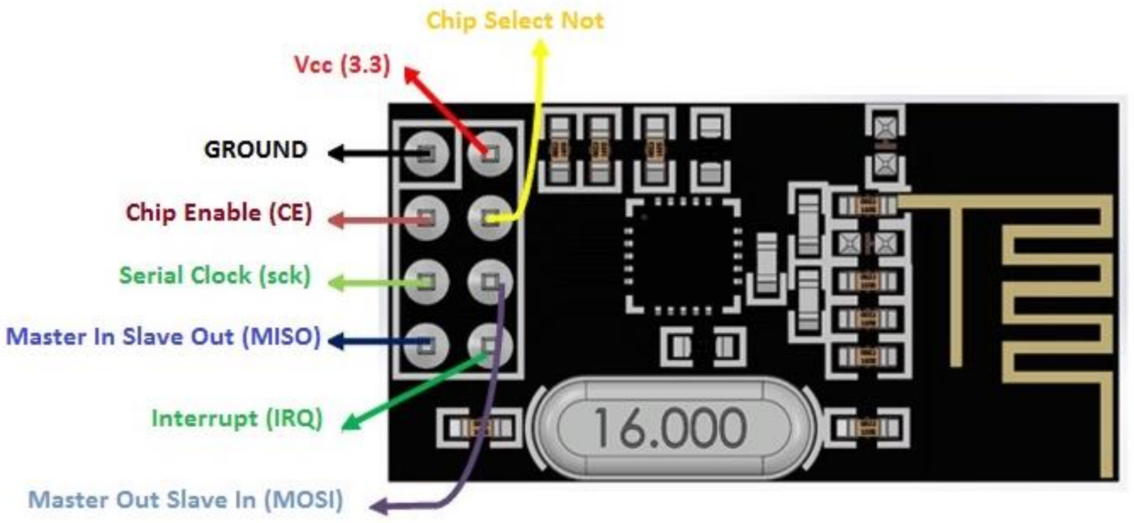
PROCEDURE AND WORKING

Transmitter

We designed a simple transmitter Using Arduino Nano with DHT22 sensor and sends data to receiver in indoor location using NRF24L01 wireless communication via SPI communication.
We designed determined the connections for the Transmitter using open-source electrical designing application – Fritzing and tested our codes over simulation before implimenting.
Fritzing diagram for the simulation is given below, the same connections was the best for our project.



Circuit Diagram : Transmitter



NRF24L01 Pinout with SPI pins laid out

We used following **libraries** for the transmitter:

- DHT.h (for DHT22 Sensor)
- SPI.h (for SPI interfacing)
- RF24.h (for NRF24L01 module)

```
#include "DHT.h"
#include <SPI.h>
#include "RF24.h"
```

```
struct package
{
  float temperature ;
  float humidity ;
} data;
```

Structure - **data** is used for data transmission between receiver and transmitter.

Initialization & void setup() block

```
DHT dht(DHTPIN, DHTTYPE);

RF24 myRadio (7, 8);
const byte address[6] = "00001";

void setup()
{
    Serial.begin(9600);
    pinMode(led_pin, OUTPUT);
    dht.begin();
    myRadio.begin();
    myRadio.setChannel(115);
    myRadio.setPALevel(RF24_PA_MAX);
    myRadio.setDataRate(RF24_250KBPS);
    myRadio.openWritingPipe(address);
    myRadio.stopListening();
    delay(1000);
}
```

For **DHT Sensor,** all we had to do was create object of **dht library** and assing the pin used and type of sensor used. Then for **dht.begin()** in void setup is used for getting data continously

For **NRF24L01 module,** data is send by the means of structure. Next we need to create a byte array which will represent the address, or the so called pipe through which the two modules will communicate. we need to include the basic **SPI** and the newly installed **RF24 libraries** and create an RF24 object. The two arguments here are the CSN and CE pins. we need to initialize the radio object and using the function we set the address of the receiver to which we will send data. Then using the **myRadio.setPALevel()** function we set the Power Amplifier level. Next we have the **myRadio.stopListening()** function which sets module as transmitter.

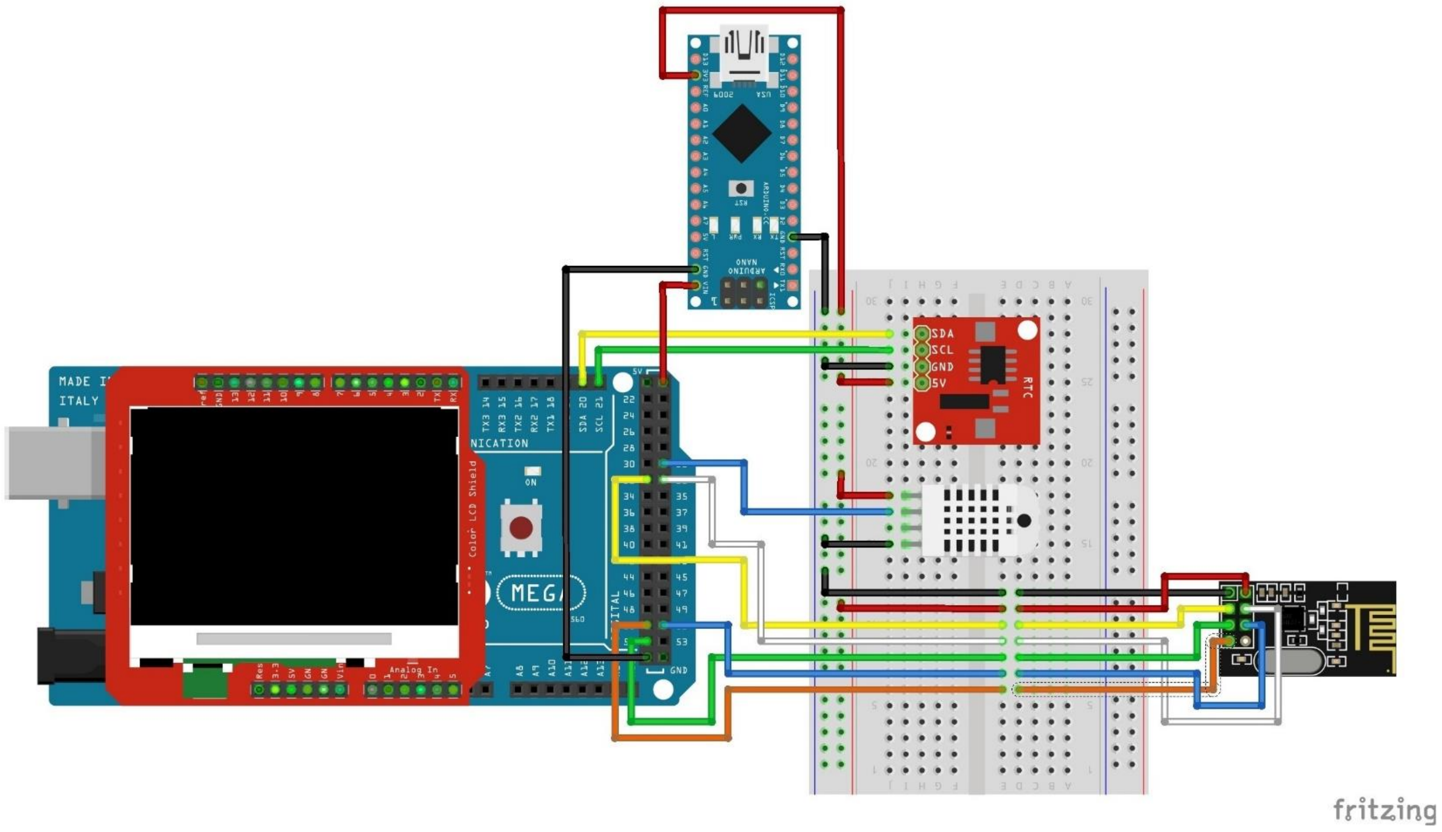
void loop() block

we are storing humidity and temperature using **readHumidity()** and **readTemperature().** Then these value are written off to the Receiver side using **myRadio.write()** function

```
void loop()
{
  digitalWrite(led_pin, LOW);
  data.humidity = dht.readHumidity();
  data.temperature = dht.readTemperature();
  Serial.println(data.temperature);
  Serial.println(data.humidity);
  myRadio.write(&data, sizeof(data));
  digitalWrite(led_pin, HIGH);
  delay(1000);
}
```

Receiver

Just like Transmitter, We designed a simple Receiver Using Arduino Mega with DHT22 sensor, DS3231 RTC module and 3.5" TFT LCD Display Sheild, and receive data from Transmitter at remote location using NRF24L01 wireless communication via SPI communication and display it on the TFT display
Like Before, we designed Receiver circuit on Fritzing before implimenting for proper wiring. Fritzing diagram for the simulation is given below, which was later used for the project



Circuit Diagram : Transmitter

The code of the Receiver is almost similar to Transmitter, but using reading pipe instead of **writing pipe, startListening()** and **checkForWirelessData()** with reference to **DHT** and **NRF24L01** modules. However, main difference comes up because of **Display Shield** (using **LCDWIKI_GUI.h** & **LCDWIKI_KBV.h** libraries) and **DS3231** RTC module (using **Sodaq_DS3231.h** library). Functioning of different module has been described in brief is given below:

*For **DS3231 Module :***

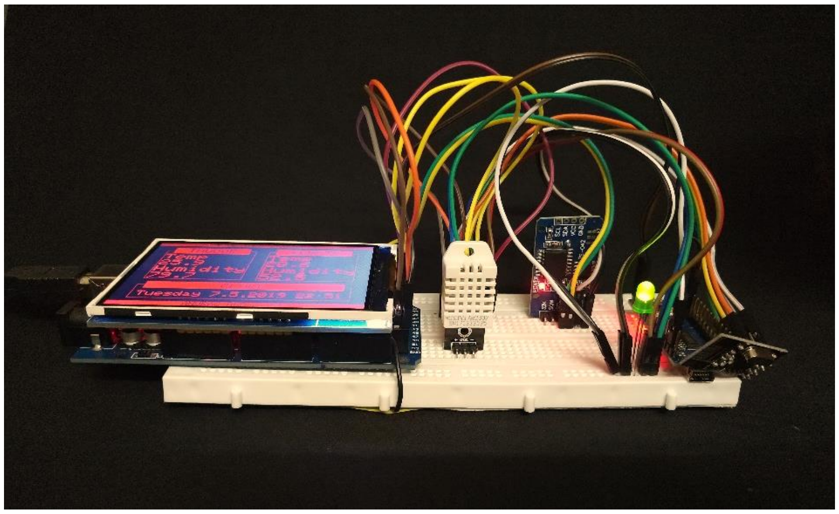
- **setRTCTime()** function is used once in setup block to set date & time of the clock.
- **getTime()** function is used to get time from DS3231 using functions from Sodaq_DS3231.h library

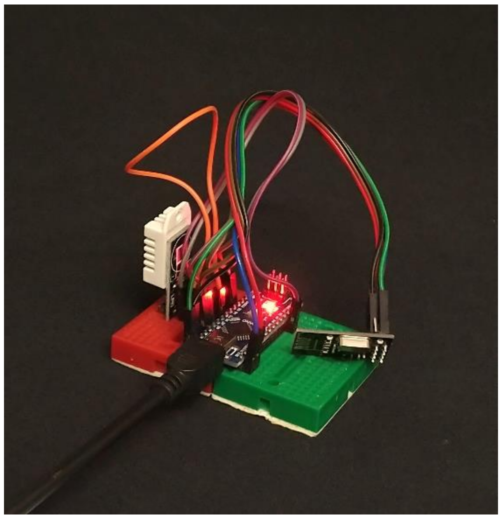
For **ILI9586 Display Shield :**

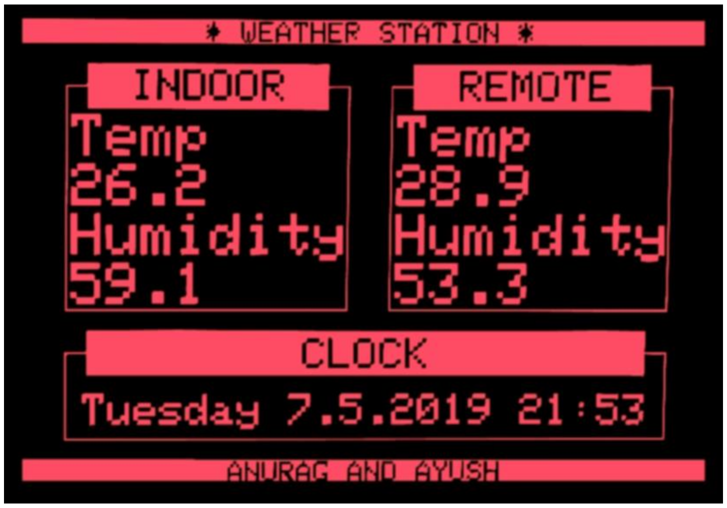
- **UI_setup()** function is used in setup block to setup the screen which is not to be refreshed again and again.
- **show_string()** function is used to print string at particular location on the screen.
- **printIndoorTemperature() & printIndoorHumidity()** functions are used to get data from DHT22 sensor at indoor base and compare with previous values. If the values are different, the value is changed on the screen. This is done to reduce flickering of the screen, while printing data.
- **printRemoteTemperature() & printRemoteHumidity()** functions are used to get data using NRF24L01 from remote base and compare with previous received values. If the values are different, the value is changed on the screen. This is done for same reason as mentioned above.

RESULT

The constructed devices worked properly and functioned according to our aims with good, low-latency wireless performance. Images of its working is given below.


Receiver Base


Transmitter Base



Output Screen at Transmitter Base

APPENDIX

Transmitter Code

```
#include "DHT.h"
#include "RF24.h"
#include <SPI.h>
#define DHTPIN 4
#define DHTTYPE DHT22
#define led_pin 13

struct package
{
  float temperature ;
  float humidity ;
} data;

RF24 myRadio (7, 8);
DHT dht(DHTPIN, DHTTYPE);
const byte address[6] = "00001";

void setup()
{
	Serial.begin(9600);
	pinMode(led_pin, OUTPUT);
	dht.begin();
	myRadio.begin();
	myRadio.setChannel(115);
	myRadio.setPALevel(RF24_PA_MAX);
	myRadio.setDataRate(RF24_250KBPS);
	myRadio.openWritingPipe(address);
	myRadio.stopListening();
	delay(1000);
}
void loop()
{
      digitalWrite(led_pin, LOW);  // Flash a light to show transmitting
      data.humidity = dht.readHumidity();
      data.temperature = dht.readTemperature();
      Serial.println(data.temperature);
      Serial.println(data.humidity);
      myRadio.write(&data, sizeof(data));
      digitalWrite(led_pin, HIGH);
      delay(1000);
}
```

Receiver Code

```
#include <LCDWIKI_GUI.h>
#include <LCDWIKI_KBV.h>
#include <Sodaq_DS3231.h>
#include "DHT.h"
#include "RF24.h"

#define DHTPIN 31
#define DHTTYPE DHT22

// DHT object initialization -- Temperature and Humidity Sensor
DHT dht(DHTPIN, DHTTYPE);

// RF24 object initialization -- Wireless Communitication Module
RF24 myRadio (34,33);
const byte address[6] = "00001";

// LCDWIKI_KBV object initialization -- TFT Display Module
LCDWIKI_KBV my_lcd(ILI9486,A3,A2,A1,A0,A4);
int16_t xmax = my_lcd.Get_Display_Height()-1;
int16_t ymax = my_lcd.Get_Display_Width()-1;

// Structure and Variable for data from Remote Location
struct package
{
  float temperature ;
  float humidity ;
} data;

float previousRemoteHumidity = 0.1;
float previousRemoteTemperature = 0.1;
float remoteHumidity = 0.0;
float remoteTemperature = 0.0;

// Variables for data from Indoor Location
float previousIndoorHumidity = 0;
float previousIndoorTemperature = 10;
float indoorHumidity = 0;
float indoorTemperature = 0;

// Variables for Clock Module
String dateString;
String hours;
```

```
int minuteNow=0;
int minutePrevious=0;

// Only for first time to setup clock time
void setRTCTime()
{
  // Adjust date-time as defined 'dt'
  // Year, Month, Day, Hour, Minutes, Seconds, Day of Week
  DateTime dt(2019, 4, 27, 18, 57, 15, 6);
  rtc.setDateTime(dt);
}

// Function to print string on display
void show_string(uint8_t *str, int16_t x, int16_t y, uint8_t csize, uint16_t fc, uint16_t bc, boolean
mode)
{
    my_lcd.Set_Text_Mode(mode);
    my_lcd.Set_Text_Size(csize);
    my_lcd.Set_Text_colour(fc);
    my_lcd.Set_Text_Back_colour(bc);
    my_lcd.Print_String(str,x,y);
}

// --- Main Display ---
void UI_setup(void)
{
  my_lcd.Set_Draw_color(230, 35, 35);

  // header
  my_lcd.Fill_Rectangle(0, 0, xmax, 15);
  show_string("* WEATHER STATION *", CENTER, 1, 2, 0x0000, 0, 1);

  // footer
  my_lcd.Fill_Rectangle(0, ymax-15, xmax, ymax);
  show_string("ANURAG AND AYUSH", CENTER, ymax-14 , 2, 0x0000, 0, 1);

  // indoor
  my_lcd.Draw_Rectangle(30, 45, (xmax+1)/2 - 15, ymax-120);
  my_lcd.Fill_Rectangle(45, 30, (xmax+1)/2 - 30, 60);
  show_string("INDOOR", 75, 34, 3, 0x0000, 0, 1);
  show_string("Temp", 34, 65 , 4, 0xE165, 0, 1);
  show_string("Humidity", 34, 133 , 4, 0xE165, 0, 1);

  // outdoor
  my_lcd.Draw_Rectangle((xmax+1)/2 + 15, 45, xmax-30, ymax-120);
  my_lcd.Fill_Rectangle((xmax+1)/2 + 30, 30, xmax-45, 60);
  show_string("REMOTE", (xmax+1)/2 + 62, 34, 3, 0x0000, 0, 1);
  show_string("Temp", (xmax+1)/2 + 19, 65 , 4, 0xE165, 0, 1);
  show_string("Humidity", (xmax+1)/2 + 19, 133 , 4, 0xE165, 0, 1);

  // clock
  my_lcd.Draw_Rectangle(30, ymax-90 , xmax-30 , ymax-30 );
  my_lcd.Fill_Rectangle(45, ymax-105 , xmax-45 , ymax-75 );
  show_string("CLOCK", CENTER, ymax-100, 3, 0x0000, 0, 1);
}

// ----- CLOCK -----
String getDayOfWeek(int i)
{
  switch(i)
  {
    case 1:   return "Monday";    break;
    case 2:   return "Tuesday";   break;
    case 3:   return "Wednesday"; break;
    case 4:   return "Thursday";  break;
    case 5:   return "Friday";    break;
    case 6:   return "Saturday";  break;
    case 7:   return "Sunday";    break;
    default:  return "Monday";    break;
  }
}

void getTime()
{
   DateTime now = rtc.now();  //get the current date-time
   minuteNow = now.minute();
   if(minuteNow!=minutePrevious)
   {
    dateString = getDayOfWeek(now.dayOfWeek())+" ";
    dateString = dateString+String(now.date())+"."+String(now.month());
    dateString= dateString+"."+ String(now.year());
    minutePrevious = minuteNow;
    hours = String(now.hour());
    if(now.minute()<10)
      hours = hours+":0"+String(now.minute());
    else
      hours = hours+":"+String(now.minute());
    String dateAndTime = dateString+" "+hours;
    Serial.println(dateAndTime);
    uint8_t buf[50];
    dateAndTime.toCharArray(buf,50);
    my_lcd.Set_Draw_color(0, 0, 0);
    my_lcd.Fill_Rectangle(31, ymax-74 , xmax-31 , ymax-31 );
    show_string(buf, CENTER, ymax-61, 3, 0xE165, 0, 1);
   }
}

// --- Indoor Temperature ---
void printIndoorTemperature()
{
  indoorTemperature = dht.readTemperature();
  String temperature = String(indoorTemperature);;
  if(indoorTemperature != previousIndoorTemperature & temperature!=" NAN")
  {
    temperature[4]=' ';//'C';
    Serial.println("Indoor Temperature : "+temperature);
    uint8_t buf[6];
    temperature.toCharArray(buf,6);
    my_lcd.Set_Draw_color(0, 0, 0);
    my_lcd.Fill_Rectangle(31, 98, (xmax+1)/2 - 16, 132);
    show_string(buf , 34, 99 , 4, 0xE165, 0, 1);
    previousIndoorTemperature = indoorTemperature;
  }
}

// --- Indoor Humidity ---
void printIndoorHumidity()
{
  previousIndoorHumidity = indoorHumidity;
  indoorHumidity = dht.readHumidity();
  String humidity = String(indoorHumidity);
  if(indoorHumidity != previousIndoorHumidity & humidity!=" NAN")
  {
    humidity[4]=' ';//'%';
    Serial.println("Indoor Humidity : "+humidity);
    uint8_t buf[6];
    humidity.toCharArray(buf,6);
    my_lcd.Set_Draw_color(0, 0, 0);
    my_lcd.Fill_Rectangle(31, 166, (xmax+1)/2 - 16, 198);
    show_string(buf, 34, 167 , 4, 0xE165, 0, 1);
    previousIndoorHumidity = indoorHumidity;
  }
}

// --- Getting data from Remote Communication ---
```

```
void checkForWirelessData()
{
  if ( myRadio.available())
    {
      myRadio.read( &data, sizeof(data) );
      previousRemoteTemperature = remoteTemperature;
      previousRemoteHumidity = remoteHumidity;
      remoteTemperature = data.temperature;
      remoteHumidity = data.humidity;
    }
}

// --- Remote Temperature ---
void printRemoteTemperature()
{
  String temperature;
  if(remoteTemperature != previousRemoteTemperature)
  {
      temperature = String(remoteTemperature);
      temperature[4]=' ';//'C';
      Serial.println("Remote Temperature : "+temperature);
      uint8_t buf[6];
      temperature.toCharArray(buf,6);
      my_lcd.Set_Draw_color(0, 0, 0);
      my_lcd.Fill_Rectangle((xmax+1)/2 + 16, 98, xmax-31, 132);
      show_string(buf, (xmax+1)/2 + 19, 99 , 4, 0xE165, 0, 1);
      previousRemoteTemperature = remoteTemperature;
  }
}

// --- Remote Humidity ---
void printRemoteHumidity()
{
  String humidity;
  if(remoteHumidity != previousRemoteHumidity)
  {
    if(remoteHumidity == 0.0 && remoteTemperature == 0.0) //We just booted up
      humidity = "---";
    else
    {
      humidity = String(remoteHumidity,1);
      humidity[4]=' ';//'%';
      Serial.println("Remote Humidity : "+humidity);
      uint8_t buf[6];
      humidity.toCharArray(buf,6);
      my_lcd.Set_Draw_color(0, 0, 0);
      my_lcd.Fill_Rectangle((xmax+1)/2 + 16, 166, xmax-31, 198);
      show_string(buf, (xmax+1)/2 + 19, 167 , 4, 0xE165, 0, 1);
      previousRemoteHumidity = remoteHumidity;
    }
  }
}

void setup()
{
    Serial.begin(9600);
    my_lcd.Init_LCD();
    Serial.println(my_lcd.Read_ID(), HEX);
    my_lcd.Fill_Screen(0x0);
    my_lcd.Set_Rotation(1);
    UI_setup();
    rtc.begin();
    dht.begin();
    myRadio.begin();
    myRadio.setChannel(115);
    myRadio.setPALevel(RF24_PA_MAX);
    myRadio.setDataRate( RF24_250KBPS ) ;
    myRadio.openReadingPipe(1, address);
    myRadio.startListening();
    delay(100);
}

void loop()
{
    getTime();
    printIndoorTemperature();
    printIndoorHumidity();
    checkForWirelessData();
    printRemoteTemperature();
    printRemoteHumidity();
}
```

BIBLIOGRAPHY & SOURCES

- Robu.in
- instructable.com
- lcdwiki.com
- https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial
- https://howtomechatronics.com/tutorials/arduino/arduino-ds3231-real-time-clock-tutorial
- https://learn.adafruit.com/dht
- NRF24L01 Library: https://github.com/TMRh20/RF24
- DHT22 Library: https://github.com/adafruit/DHT-sensor-library
- DS3231 Library: https://github.com/SodaqMoja/Sodaq_DS3231
- Udemy - Tech Explorations: Arduino Step by Step
- Google Images
- Wikipedia