

AUTOMATED WIRELESS WEATHER LOGGER STATION

BY

AKPAN EDIDIONG EMMANUEL

150808057

**A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF PHYSICS,
FACULTY OF SCIENCE, UNIVERSITY OF LAGOS, AKOKA, YABA LAGOS,
NIGERIA IN PARTIAL FULFILLMENT FOR THE AWARD OF BACHELOR OF
SCIENCE (B.SC.) DEGREE IN APPLIED PHYSICS (ELECTRONICS)**

NOVEMBER, 2019.

CERTIFICATION

This is to certify that this project work was carried out by **AKPAN EDIDIONG EMMANUEL** with matric number **150808057** of the Department of Physics, Faculty of Science University of Lagos, Akoka, Yaba Lagos, Nigeria in partial fulfillment for the award of Bachelor of science (B.Sc.) degree in Applied Physics (electronics) under the supervision of Dr. B. Olugbon.

DR. B. OLUGBON

(PROJECT SUPERVISOR)

.....

SIGNATURE AND DATE

DR. O. I. OLUSOLA

(PROJECT COORDINATOR)

.....

SIGNATURE AND DATE

PROF. E. O. OYEYEMI

(HEAD OF DEPARTMENT)

.....

SIGNATURE AND DATE

DEDICATION

This project is dedicated to God Almighty, the author and finisher of my faith, who gave me the grace, strength and enablement to start and finish this program successfully.

I also dedicate this this study to my loving Father, Mr. Sunday Akpan and to the whole family of Akpan.

ACKNOWLEDGEMENT

I am really grateful to God Almighty who gave me the strength and knowledge to sustain me throughout this project

I want to thank my supervisor, Dr. B. Olugbon for her supportive and disciplined efforts at every point in time in the course of this project. I am thankful for her constant corrections and guidance in making this project successful.

To my H.O.D Prof. E. O. Oyeyemi, my course adviser Dr. Humphrey and other members of staff and lecturers in the department of Physics I say a big thank you to you all.

I also want to appreciate my dad, Mr. Sunday Akpan who single handedly funded this project, supported me in prayers, cash, calls, messages, encouragements, advice, cash again and supports that made this project a reality. May God bless you with long life and the best of health to enjoy the fruits of your labor.

Special thanks also goes to my siblings, course mates, friends, well-wishers for their consistent concern, moral supports and prayers, may God richly bless you all.

Lastly, I thank myself for a job well done.

TABLE OF CONTENTS

CERTIFICATION	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS	ix
ABSTRACT	1
CHAPTER ONE	2
1.1 General Introduction	2
1.2 Elements of Weather	2
1.3 General Description of the Automated Wireless Weather Logger Station (AWWLS) ...	4
1.4 STATEMENT OF PROBLEM	4
1.5 SIGNIFICANCE OF STUDY	4
1.6 AIM AND OBJECTIVES OF STUDY	5
1.7 SCOPE OF STUDY	5
CHAPTER TWO	7
2.1 A Brief History of Weather Observation	7
2.2 Conceptual Framework	9
2.3 Future Developments	10
CHAPTER THREE	11
3.1 System Operation, Design and Implementation.....	11
3.2 Design Requirements, Circuit Components and Functions	12
3.2.1 DHT22 Temperature and Humidity Sensor	13
3.2.2 Arduino NANO Microcontroller.....	17
3.2.3 NRF24L01 Transceiver Module	23

3.2.4 DS3231 RTC Module.....	31
3.2.5 SSD1306 0.96 Inch OLED Screen Display	34
3.2.6 SD Card Module.....	37
3.2.7 Custom Printed Circuit Boards (PCBs).....	40
3.2.8 Power Jack.....	41
3.2.9 Power Switch.....	42
3.2.10 18650 Li-ion Battery	43
3.3 System Design and Implementation.....	45
CHAPTER FOUR.....	50
4.1 Introduction	50
4.2 Results	50
4.3 Discussion	54
CHAPTER FIVE	55
5.1 Summary and Conclusion	55
5.2 Recommendations	55
REFERENCES	57
APPENDIX.....	58

LIST OF FIGURES

Figure 3.1 Block diagram of the Arduino Wireless Weather Logger Station (AWWLS).....	11
Figure 3.2 The DHT11 and DHT22 Temperature and Humidity sensors	14
Figure 3.3 A DHT22 Sensor consisting of a Humidity sensing component and NTC	15
Figure 3.4 The Humidity sensing component having two electrodes and a moisture holding substrate between them	16
Figure 3.5 A Graphical Plot showing the Resistance against the Temperature in the NTC sensor of DHT22	17
Figure 3.6 The Arduino Interface Development Environment with C++ Program Language	18
Figure 3.7 The Arduino NANO microcontroller board	19
Figure 3.8 The pinout of the Arduino Nano board	21
Figure 3.9 NRF24L01 Transceiver Module with Antenna.....	23
Figure 3.10 NRF24L01 Transceiver Module without Antenna.....	24
Figure 3.11 The Channel Frequency for theNRF24L01 Transceiver Module.....	26
Figure 3.12 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 1	27
Figure 3.13 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 2	28
Figure 3.14 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 3	29
Figure 3.15 The NRF24L01 Transceiver module pinout.....	30
Figure 3.16 The DS3231 Real Time Clock Module	31
Figure 3.17 A Graphical representation of the compensated frequency of TCXO of a DS3231 RTC module.....	32
Figure 3.18 A CR2032 Battery holder attached to its back can hold a 20mm 3v lithium coin-cell CR2032 battery	33
Figure 3.19 DS3231 RTC module Pinout.....	34
Figure 3.20 SSD1306 0.96 Inch OLED Screen Display	35
Figure 3.21 A 1K memorySSD1306 0.96 Inch OLED Screen Display with pages, segments and data SSD1306 0.96 OLED Screen Display Module Pinout.....	36

Figure 3.22 SSD1306 0.96 OLED Screen Display Module Pinout	36
Figure 3.23 SD Card Module	37
Figure 3.24 Two main components on the SD Card Module	38
Figure 3.25 Micro SD Card Module Pinout.....	39
Figure 3.26 A designed PCB for the AWWLS circuit diagram (https://www.easyeda.com).....	40
Figure 3.27 The manufactured PCB design for the AWWLS circuit diagram	41
Figure 3.28 A Female Connector Power Jack.....	42
Figure 3.29 A Power Switch	43
Figure 3.30 18650 Lithium ion Battery	43
Figure 3.31 A Dual 18650 Lithium ion Battery Holder.....	44
Figure 3.32 A 10 micro-farad Capacitor	44
Figure 3.33 A 220 ohms Resistor.....	44
Figure 3.34 Soldering header pins to the PCB.....	45
Figure 3.35 Placing the Circuit components on the PCBs (https://www.howtomechatronics.com)	46
Figure 3.36 The Circuit components for the Indoor and Outdoor PCBs	47
Figure 3.37 The Arduino IDE and the included Libraries	48
Figure 3.38 The Indoor and Output System of the Automated Wireless Weather Logger Station (https://www.howtomechatronics.com)	49
Figure 4.1 Declaring the File name of the Logged data as firstday.csv and initializing the SD card to begin logging of its retrieved data	50
Figure 4.2 The Serial monitor displaying the logged data at real time at the interval of one second every hour showing the Day, Date, Time, Temperature in °C and Humidity in %	51
Figure 4.3 Sample of the data stored in a text file	52
Figure 4.4 A plot of the Temperature and Humidity against Time.....	53
Figure 4.5 A plot of the Temperature and Humidity against Time.....	53

LIST OF ABBREVIATIONS

μA	Micro-Ampere
μS	Micro-second
AC	Alternating Current
ACK	Acknowledgement packet
ARD	Auto-Retransmit-Delay
AWWLS	Automatic Wireless Weather Logger Station
CE	Chip Enable
CMOS	Complementary Metal-Oxide Semiconductor
CSN	Chip Select Not
dB m	Decibel meter
DC	Direct Current
DHT22	Digital High Technology for Temperature and Humidity sensor
DS3231 RTC	Digital Single 3231 Real Time Clock
FTDI	Future Technology Devices International
GDDRAM	Graphic Display Data Random Access Memory
GFSK	Gaussian Frequency-Shift Keying
GHz	Giga-Hertz
GND	Ground
GPS	Global Positioning System
IC	Integrated Circuit
IDE	Integrated Development Environment
IRQ	Interrupt pin
ISM	Indian Service Machine
Kbps	Kilobytes per second
LCD	Liquid Crystal Display
LI-ION	Lithium-ion
mA	Milli-Ampere
Mbps	Megabytes per second

MHz	Megs-Hertz
MISO	Master In Slave Out
MOSI	Master Out Slave In
nA	Nano-Ampere
NRF24L01	Multiple Transmitter Single Receiver
NTC	Negative Temperature Coefficient
OLED	Organic Light Emitting Diode
PCB	Printed Circuit Board
PWM	Pulse Width Modulation
RTD	Resistance Temperature Detectors
RX	Receiving Pin
SCK	Serial Clock pin
SCL	Serial Clock
SDA	Serial Data Access
SPI	Serial Peripheral Interface
SQW	Square Wave output pin
TCXO	Temperature Compensated Crystal Oscillator
TTL	Transistor-Transistor Logic
TWI	Two Wire Interface
TX	Transmitting Pin
UART	Universal Asynchronous Reception and Transmission
USB	Universal Serial Bus
V	Volts
VCC	Voltage Collector-Collector power supply

ABSTRACT

Climate change is the leading cause of natural disaster in the world. Air temperature, humidity, wind speed and direction including air pressure readings are the best weather elements and parameters that could promote early detection of climate changes. A number of weather stations in Nigeria are purchased overseas and contain several missing data points in the logged data. The costs of acquisition of these instruments and unavailability of data are clear limitations in the field of meteorology today. This project employed the use of wireless networks in the design and implementation of an automatic weather station which detects, measures and logs temperature and relative humidity. This unit is an improvement in terms of cost and reliability. Temperature and humidity sensors were mounted on printed circuit boards which were designed with a programmed Arduino board and connected through a wireless medium. The wireless feature was responsible for the transmission of the measured readings. A display unit was also included in the design. Due to unavailability of temperature and humidity data at the weather station in the Department of Physics, Faculty of Science, University of Lagos during the deployment of the equipment, the data readings could not be validated.

CHAPTER ONE

INTRODUCTION

1.1 General Introduction

Weather and climate are among the foremost factors which determine how a society develops in a geographical region. Weather usually describes a particular event or condition for a short period of time such as hours or days whereas climate refers to the behavior of the atmosphere in a given place over many years. Weather also includes current atmospheric conditions such as the temperature, precipitation, humidity and wind (Muhammad, 2015; Ahmad, 2014; Nisha et al., 2015; Moje et al., 2017). Weather reports, and data are usually gathered and collated and used to serve as a precautionary measure against natural disasters such as hurricane, flood and drought (Abubakar and Sulaiman, 2018). These data collated over a long period of time can be used to predict climate change and future trends i.e. the data can identify the pattern of climate change, how and when they occur, as well as their cause and prevention (Rameshbabu et al., 2018).

1.2 Elements of Weather

The atmospheric conditions of a place at any time is expressed by a combination of several elements. Some of the elements include

- **Temperature:** The temperature of place shows how cold or hot the atmosphere is. It is measured with a thermometer that measures in units of degrees Celsius. Temperature is an important factor in determining the weather because it influences and controls other elements of the weather.

- **Humidity:** This is the measurable amount of moisture in the air of the atmosphere or simply the amount of water vapor present in the atmosphere. It is measured with a hygrometer. There are three types of humidity which are the Relative Humidity, Absolute and Specific Humidity. The Relative humidity, which is our major concern, is the ratio of the amount of moisture in the air at a certain temperature to the maximum amount of moisture that the air can retain at the same temperature. Meaning that the relative humidity is measured by how much of the moisture capacity of the air is being used. It is expressed as a percentage and is highest during a rain downpour, usually reaching a 100 percent.
- **Air Pressure:** Air pressure is the weight resting on the earth's surface. This weight exerted by the air is the atmospheric pressure, and all pressure systems have direct impact on precipitation. Some places dominated by low pressure tend to be moist while those dominated by high pressure are dry. The instrument used for measuring this element of weather is a barometer.
- **Precipitation:** As an element of weather, it determines whether the outdoor activities are suitable and if the water levels of lakes and rivers will rise. As an element of climate, precipitation is a long-term predictable factor of a region's makeup. For instance, a deserted environment may experience a storm (weather) though it remains a typically dry area (climate). Its measuring instrument is the Pluviometer.
- **Cloudiness:** Cloud are suspended water in the atmosphere, as they are the most obvious feature in the sky. The clouds give us a clue about what is going on in the atmosphere and how the weather might change in the hours or days to come. Each cloud forms in a different way and each brings its own kind of weather. They are the base for precipitation and they provide protection from the rays of the sun (Hocking, 2000).

1.3 General Description of the Automated Wireless Weather Logger Station (AWWLS)

The Automated Wireless Weather Logger Station (AWWLS) is a device that is employed in climatology monitoring using a microcontroller and sensor to monitor and record atmospheric conditions like temperature and humidity. The logged data can then be carefully analyzed to provide useful information of the weather conditions of a particular environment and be used to prepare reports to the community and general public.

1.4 STATEMENT OF PROBLEM

Weather stations available in some parts of Nigeria have weather instruments and tools that are manually used to attain the readings of the atmospheric conditions. In the locations where more sophisticated weather monitoring facilities are available, the equipment are usually sourced overseas and are expensive. When these equipment breakdown (which is quite common), maintenance can be a problem which can result to abandoning of the equipment, outright loss of invested funds, resources, time, energy and man-power in this direction. Indigenous input in this area is lacking. The AWWLS is a contribution to introduce a functional low cost but accurate and durable temperature and humidity sensor device that is portable which can also log data that can be retrieved or accessed for analysis and dissemination locally.

1.5 SIGNIFICANCE OF STUDY

Human activity is influenced by weather conditions. Therefore, monitoring weather conditions can help greatly in controlling human activity. The weather change of a particular area will differ from

another, as such it is important to monitor and study the pattern of the weather at the different locations, compare and contrast the patterns.

1.6 AIM AND OBJECTIVES OF STUDY

The aim of this project is to build an Automated Wireless Weather Logger Station that senses and logs the temperature and humidity values of an environment. The objectives of this study are to:

1. Acquire the components of the AWWLS such as the Arduino NANO Microcontroller, DHT22 temperature and humidity sensor, 0.96' inch OLED display screen, DS3231 RTC as described in Chapter three.
2. Program the Arduino NANO Microcontroller to take measurements of temperature and humidity at real time using the DHT22 sensor.
3. Log the data readings of temperature and humidity for both Indoor and Outdoor units for every second of the hour for a period of six to eight hours (6-8 hours) to obtain a plot of temperature and humidity against varying time.
4. Compare the data obtained from the AWWLS with other weather recorded readings at other locations.

1.7 SCOPE OF STUDY

The scope of the study will be confined and conducted in the serene environment of the Mariere Hall, University of Lagos Akoka Yaba. The study will focus on collecting Temperature and

humidity weather data alongside the varying time difference from the surrounding environment and storing the data on a data storage medium that can be easily accessible.

CHAPTER TWO

LITERATURE REVIEW

2.1 A Brief History of Weather Observation

Modern weather tools or instruments did not begin development until the 1400's. Before this, weather observation was extremely rudimentary - mostly based on the appearance of the sky and the feel of the air. Much of the development of these weather tools were not just necessitated by agriculture, but also due to an increase in sea travel. Because storms at sea can be deadly, and ships were propelled by the wind, the ability to predict weather conditions relevant to sailing was extremely important.

Weather observation began to be developed in the early mid-1400's. In 1441, a Korean crown prince named Munjong of Joseon developed the first standard rain gauge (Acurite, 2019). This rain gauge was then popularized across the country by his father, Sejong the Great. More developments in weather monitoring included the first water thermometer in 1593, the first practical humidity measurement system in 1664, and the first barometer in 1643. Over time, these weather tools were further improved and refined.

The National Oceanic and Atmospheric Administration (NOAA) was established in the 1800's, and it still provides most of the weather information for the United States. But many people enjoy setting up their own home weather stations for weather information that is more specific to their location. Home weather stations have always been particularly useful for farmers, who need to be able to anticipate the weather and the needs of their crops.

Agriculture has always been heavily dependent on the weather and weather forecasts, both for its control on the quantity and quality of a harvest and its effect on the farmer's ability to work the land and graze his stock. Also, water resources depend critically not just on rainfall, but also other weather phenomenon that drives together plant growth, photosynthesis and evaporation. Just as pollen and seed dispersal in the atmosphere are driven almost entirely by the weather, so too is the direction and distance of travel of atmospheric pollution.

In the early 1800's and 1900's, home weather stations typically consisted of a few specialized analog tools. Humidity gauges measure the moisture content in the air, while rain gauges and barometers help determine previous and future rainfall. This information could then be tracked in order to identify trends. Today, there are digital weather stations that can better consolidate and report information so it is easy to view and understand. Some weather stations can connect to smartphone apps or online services so people can access their weather information from anywhere. Modern weather observers may use this information to manage their gardens, monitor their farms, or simply as a hobby. Digital weather stations tend to be more accurate and easier to use (Acurite, 2019). Weather monitoring is also important not just in defining the present climate, but also detecting changes in the climate and providing the data to input into models which enables us to predict the future changes in our environment. Online weather reports are often based on weather stations located at airports, which most times are miles away from home, but a personal weather station gives a more detailed and localized information that is central to the home and neighborhood.

As the same way that news reporting has moved to the Internet, so has a lot of weather reporting. Often, information can be sourced from multiple areas and analyzed in order to give a more

detailed picture of the current weather condition. This is important especially in areas that are likely to have emergency conditions, such as extreme storms, hurricanes, or tornadoes. The demand for these data, usually on an hourly or more frequent timescale, has increasingly been met by the development and widespread deployment of automatic weather stations over the last 30 years.

2.2 Conceptual Framework

According to Hendrik Zophel, he used a Platin PT100 temperature probe to measure temperature whose basis was a resistance varying according to temperature, and for humidity it's varies with capacitance. Its constraint was found in the complexity of the circuitry connections and its high cost. Also, Jason Eric Box, he used a Vaisala 50-YC for measuring temperature as well, but had a constraint is in the consistent error in readings it gave when put out in direct sunlight. In the Meteorological Department of Department of Bangladesh, their data acquisition system lagged because of the old and ancient equipment they used, and their maintenance was not so easy that's why they required the expertise of a technician's control.

The quality of these weather stations and the data obtained from them is constrained by their high cost of production, ineffectiveness and short life span. Thus, the AWWLS stands out as a digitalized and modernized system for weather monitoring and report or analysis that will be efficient, portable in structure, have a data acquisition system, as well as easy maintenance and cost.

2.3 Future Developments

One of the most significant developments which has enabled more automatic weather stations to become a common and significant item in environmental studies has been the development of economical, low-power solid state data loggers. These will continue to become more user-friendly, easy to use, powerful and able to record more variables with easy access, and greater volumes of data at similar or lower prices than today. The sensors used and designed for these systems have undergone incremental improvement over the years, leading to a steady improvement in performance and reliability accompanied by a fall in price. The number and variety of sensors on automated weather systems is also more likely to increase.

The new generation of humidity sensor should be less sensitive to chemical interference, and have improved long term stability and negligible hysteresis. This trend should continue as alternatives to other sensors that are likely to appear. Visibility monitors are increasingly common on road monitoring stations. The most likely additions are those monitoring chemical variables, not only in the atmosphere (e.g. ozone, SO₂ and other industrial pollutants), but also in rainfall and even in the soil. The newly developed circuitry boards should be fabricated and scaled up even much smaller than it is now, as technology now advances more into nanotech, the future of integrated processor chip technology. This aids more compatibility and portability (Mark and Geoff, 2011).

CHAPTER THREE

METHODOLOGY

3.1 System Operation, Design and Implementation

The setup of the AWWLS design and implementation is based on two separate operating weather stations which are the INDOOR and OUTDOOR. Both weather stations are connected by a wireless NRF24L01 Transceiver module as shown in the Figure 3.1

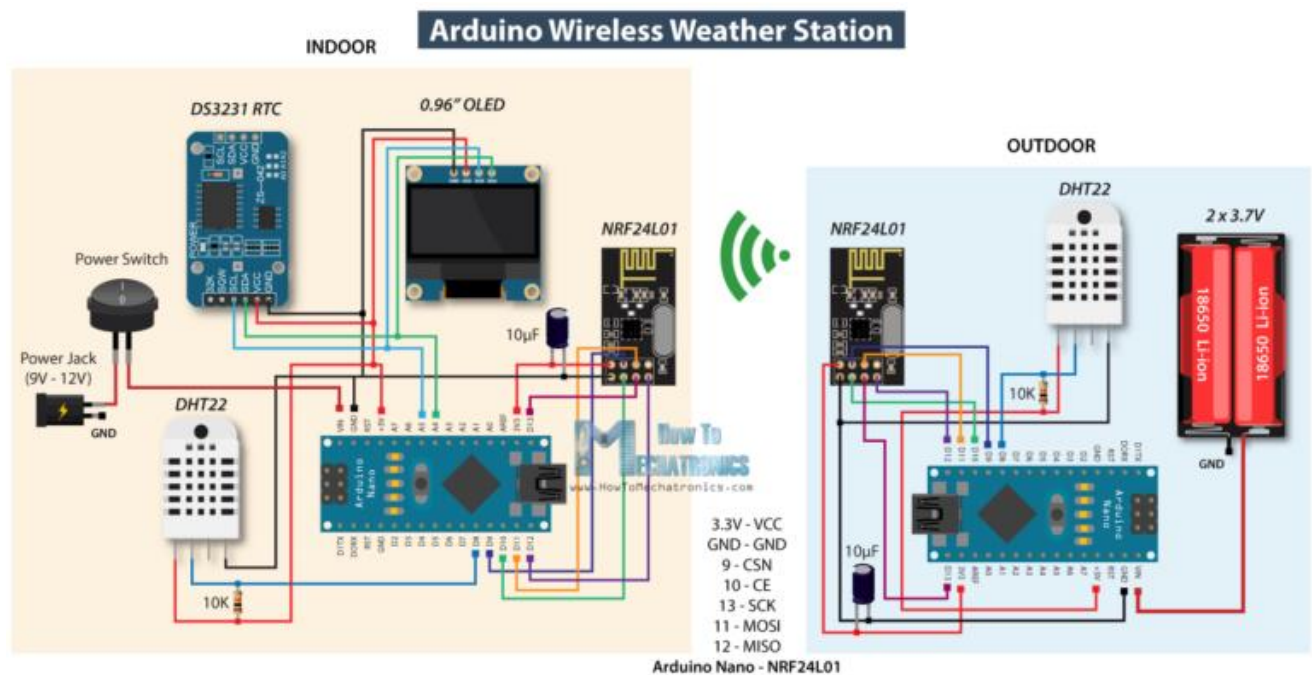


Figure 3.1 Block diagram of the Arduino Wireless Weather Logger Station (AWWLS)

The OUTDOOR weather station has a mounted DHT22 sensor that detects the temperature and humidity of the outdoor environment. The Arduino NANO microcontroller then processes the temperature and humidity data and activates the wireless NRF24L01 Transceiver which then transmits these data to the INDOOR unit. The NRF24L01 Receiver mounted in the INDOOR unit activates the 0.96' inch OLED screen and displays the received data (Dejan, 2018).

The AWWLS also has a good storage mechanism feature which is the SD Card Module that stores both the Indoor and Outdoor Temperature and Humidity.

3.2 Design Requirements, Circuit Components and Functions

The AWWLS design is comprised of a couple of Integrated Circuit components for both the Indoor and Outdoor weather stations as listed below.

1. DHT22 Temperature and Humidity Sensor
2. Arduino NANO Microcontroller
3. NRF24L01 Transceiver Module
4. DS3231 RTC Module
5. SSD1306 0.96 Inch OLED Screen Display
6. SD Card Module
7. Custom Printed Circuit Boards (PCBs)
8. Power Jack
9. Power Switch
10. 18650 Li-ion Battery
11. Battery Holder

12. Capacitor


13. Resistor.

3.2.1 DHT22 Temperature and Humidity Sensor

The Digital Temperature and Humidity Sensor uses a capacitive humidity sensor and a thermistor to measure the surrounding air and is of two types which are DHT11 and DHT22 sensors as shown in Figure 3.2. These sensors are common in electronics because they are very cheap but still provides great performance. The sensors have their specifications and their differences.

The DHT22 is a more expensive version which has better specifications, its temperature measuring range is from -40 to +125 degrees Celsius with ± 0.5 degrees accuracy, while the DHT11 temperature range is from 0 to 50 degrees Celsius with ± 2 degrees accuracy. Also, the DHT22 sensor has better humidity measuring range, from 0 to 100% with 2-5% accuracy, while the DHT11 humidity range is from 20 to 80% with 5% accuracy (Dejan, 2018).

There are two specifications where the DHT11 is better than the DHT22. That's the sampling rate which for the DHT11 is 1Hz or one reading every second, while the DHT22 sampling rate is 0.5Hz or one reading every two seconds and also the DHT11 has a smaller body size. The operating voltage of both sensors is from 3 to 5volts, while the max current used when measuring is 2.5mA.



The image shows two digital temperature and humidity sensors. On the left is the DHT11, a blue plastic module with a white sensor component and four pins. On the right is the DHT22, a white plastic module with a more complex sensor component and four pins. Between them is a logo for 'How To MECHATRONICS' with the website 'www.HowToMechatronics.com'.

DHT11		DHT22	
0 - 50°C / ± 2°C	<i>Temperature Range</i>	-40 - 125 °C / ± 0.5 °C	
20 - 80% / ± 5%	<i>Humidity Range</i>	0 - 100 % / ± 2-5%	
1Hz (one reading every second)	<i>Sampling Rate</i>	0.5 Hz (one reading every two seconds)	
15.5mm x 12mm x 5.5mm	<i>Body Size</i>	15.1mm x 25mm x 7.7mm	
3 - 5V	<i>Operating Voltage</i>	3 - 5V	
2.5mA	<i>Max Current During Measuring</i>	2.5mA	

Figure 3.2 The DHT11 and DHT22 Temperature and Humidity sensors

The Working Principle of DHT22 Temperature Sensor

As said earlier, these sensors consist of a humidity sensing component, an NTC temperature sensor (usually called a Thermistor) and an IC on the back side of the sensor as shown in Figure 3.3. The NTC stands for “Negative Temperature Coefficient”, they are resistors with negative temperature coefficient which means that the resistance decreases with increasing temperature. They are primarily used as resistive temperature sensors and current-limiting devices. The temperature sensitivity coefficient is about five times greater than that of silicon temperature sensors usually called ‘silistors’ and about ten times greater than those of resistance temperature detectors (RTDs). They are in the range from -55°C to 200°C . An NTC thermistor is thermally sensitive and its resistance exhibits a large, precise and predictable decrease as the core temperature of the resistor increases over the operating temperature range.

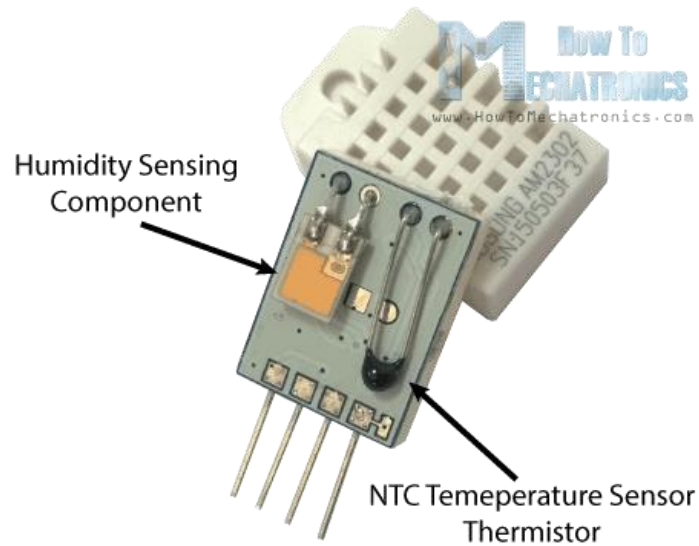


Figure 3.3 A DHT22 Sensor consisting of a Humidity sensing component and NTC

Now for measuring humidity, they use the humidity sensing component which has two electrodes with moisture holding substrate between them as shown in Figure 3.4. So as the humidity changes, the conductivity of the substrate changes or the resistance between these electrodes changes. This change in resistance is measured and processed by the Integrated Circuit (IC) which makes it ready to be detected by a microcontroller like the Arduino NANO.

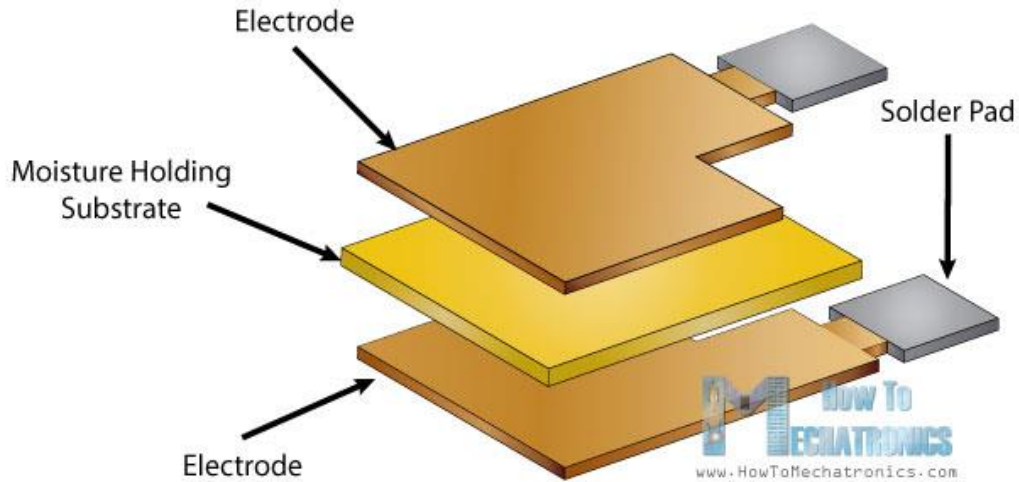


Figure 3.4 The Humidity sensing component having two electrodes and a moisture holding substrate between them

On the other hand, for measuring temperature, it uses the NTC thermistor as it is a variable resistor that changes its resistance with the change of the temperature. This thermistor is made by sintering of semi-conductive materials such as ceramics or polymers in order to provide larger changes in the resistance with just small changes in the temperature, meaning that the resistance decreases with increase of the temperature as shown in the graphical plot of resistance against temperature in Figure 3.5

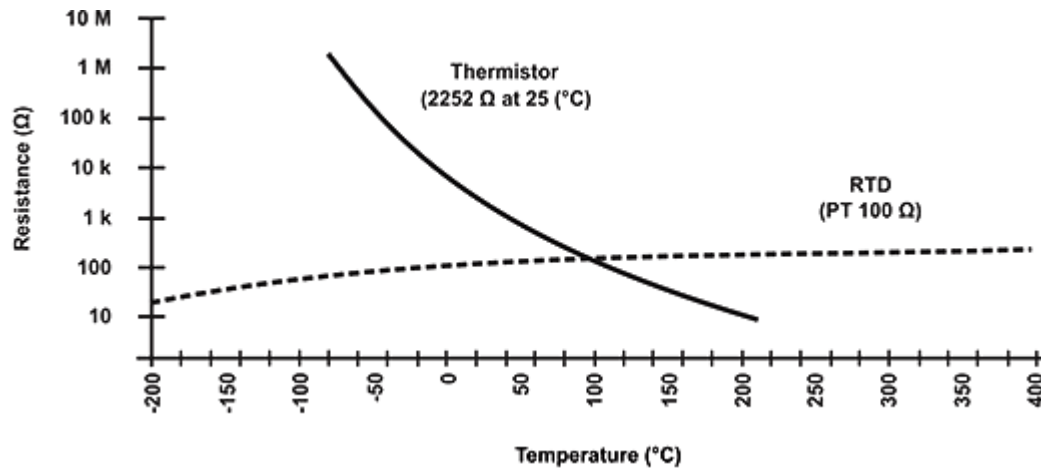


Figure 3.5 A Graphical Plot showing the Resistance against the Temperature in the NTC sensor of DHT22

3.2.2 Arduino NANO Microcontroller

Firstly, the Arduino is an open-source platform used for building electronics projects. The Arduino consists of both a physical programmable circuit board known as a microcontroller, and a piece of software or IDE (Integrated Development Environment) that is used to write and upload programmable codes from the computer to the physical board. The Arduino does not need a separate piece of hardware in order to load new codes onto the board, only a simple USB cable is required. The Arduino IDE uses a simplified version of C++ program language, making it easy to learn as shown in the Figure 3.6

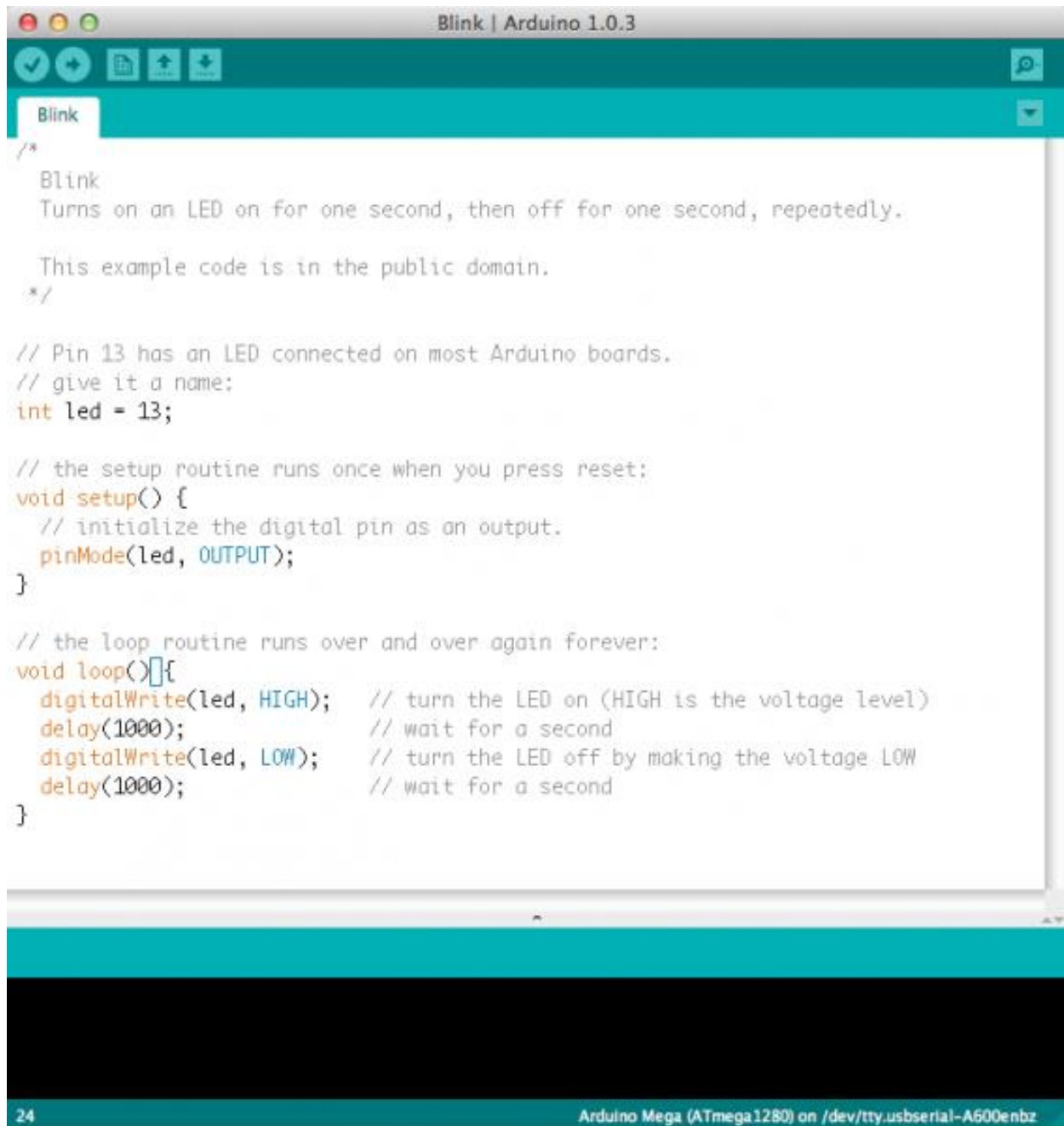


Figure 3.6 The Arduino Interface Development Environment with C++ Program Language

Now the Arduino has a couple of microcontroller boards that are widely used in robotics, embedded systems and electronic projects where automation is an essential part of the system.

Some of these microcontroller boards are: The Arduino UNO, Arduino Mega, Arduino LilyPad, Arduino Leonardo, Arduino NANO, etc. In this case, we make use of the Arduino NANO.

The Arduino NANO

The Arduino Nano is small, compact, complete and breadboard-friendly board based on the ATmega328 like the Arduino Uno only in functionality but different in size as shown in Figure 3.7

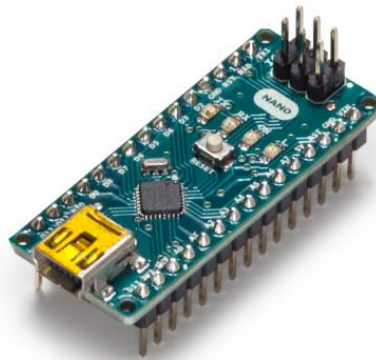


Figure 3.7 The Arduino NANO microcontroller board

The Arduino Nano can be described in terms of;

Power

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Memory

The ATmega328 has 32 KB, (also with 2 KB used for the bootloader). The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano can be used as an input or output, using `pinMode()`, `digitalWrite()`, and `digitalRead()` functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kilo Ohms. In addition, some pins have specialized functions:

- **Serial:** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- **External Interrupts:** For pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM:** Pins 3, 5, 6, 9, 10, and 11 provide 8-bit PWM output with the `analogWrite()` function.
- **SPI:** Pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED:** For pin 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Nano has 8 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upper end of their range using the `analogReference()` function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- **I2C:** A4 (SDA) and A5 (SCL). Support I2C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with `analogReference()`.
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

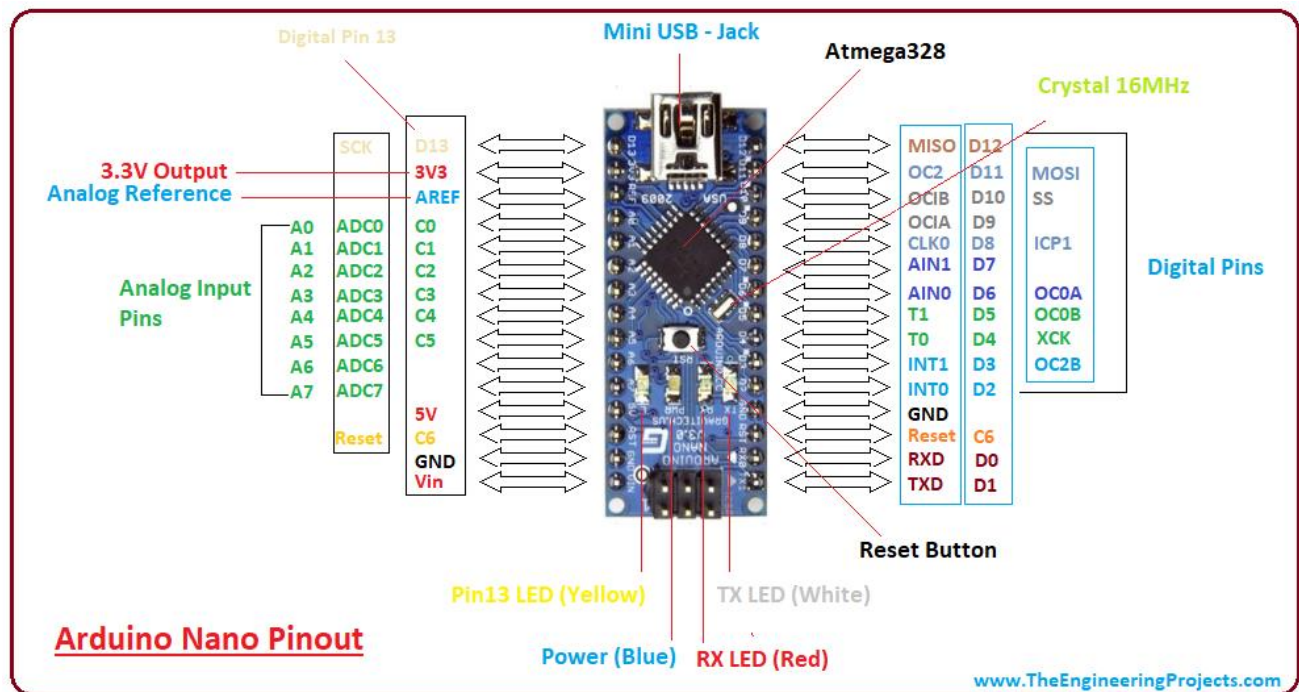


Figure 3.8 The pinout of the Arduino Nano board

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328P provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual COM port to software on the computer. The Arduino software

includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Nano's digital pins. The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus.

Applications

Arduino Nano is a very useful device that comes with a wide range of applications and covers less space as compared to other Arduino boards. Its breadboard friendly nature makes it stand out from other boards. The following are the applications of the board;

- Arduino Metal Detector
- Real-Time Face Detection
- Medical Instruments
- Industrial Automation
- Android Applications
- GSM Based Projects
- Embedded Systems
- Automation and Robotics
- Home Automation and Defense Systems
- Virtual Reality Applications

3.2.3 NRF24L01 Transceiver Module

There are two types of NRF24L01 Transceiver Module;

- The NRF24L01 with antenna – This type of Transceiver module has a wireless coverage of about 1 Kilometer in open space as shown in Figure 3.9

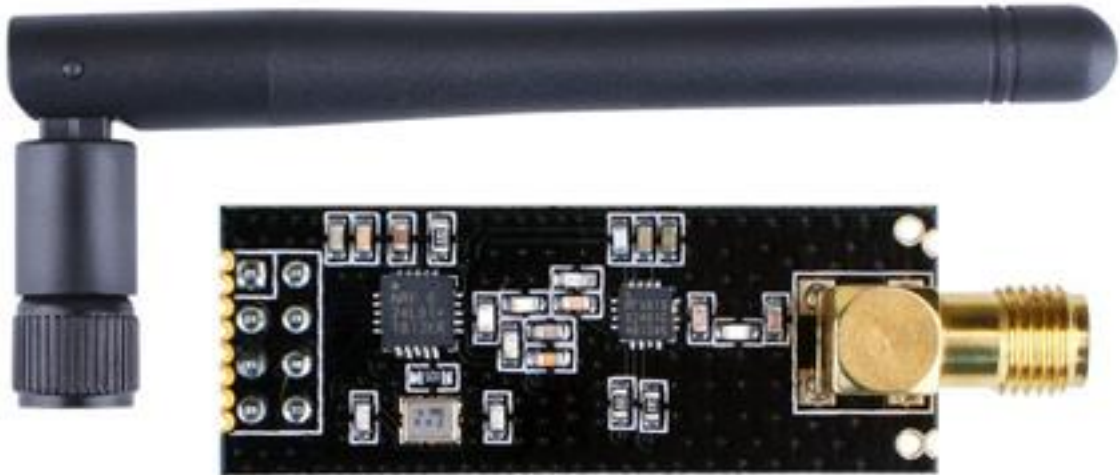


Figure 3.9 NRF24L01 Transceiver Module with Antenna

- The NRF24L01 without antenna – This is the type of Transceiver module we are working with and it has a wireless coverage of about 50 meters in open space.

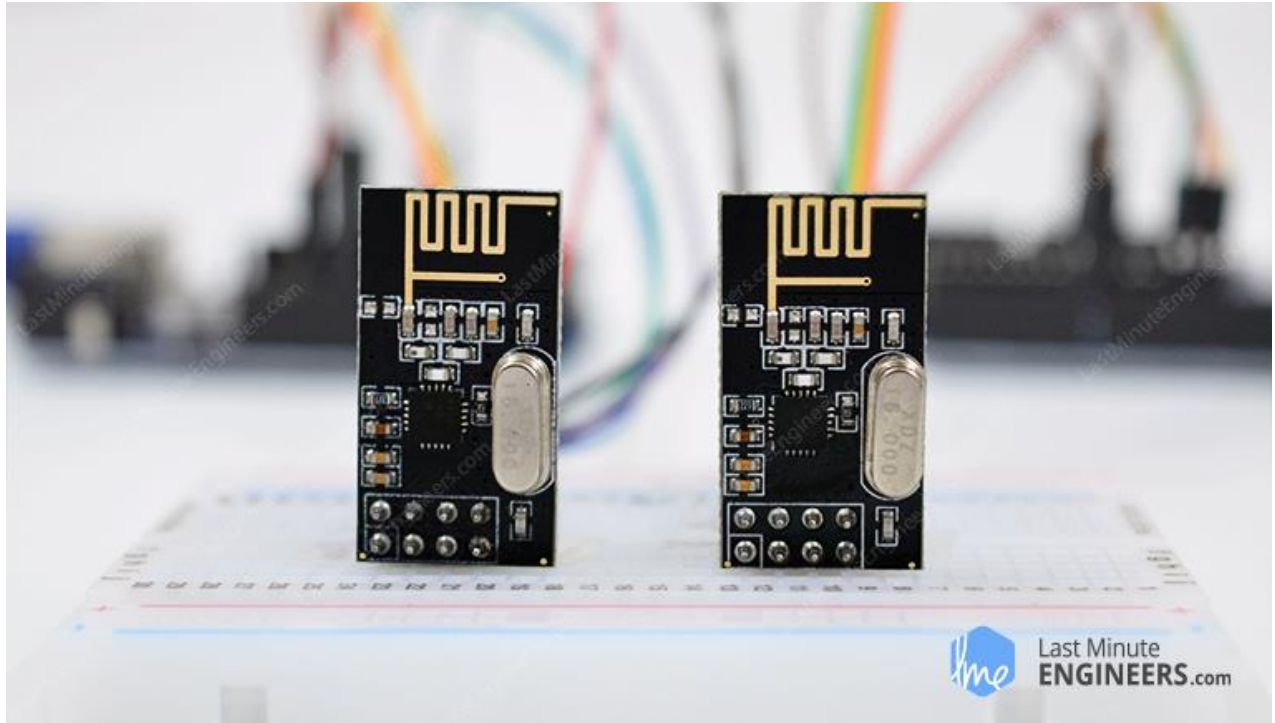


Figure 3.10 NRF24L01 Transceiver Module without Antenna

From the Fig.3.14 above, the nRF24L01 Transceiver module can operate as a transmitting device as well as a Receiving device because both the transmitter and receiver are integrated on the same IC. Its' hardware is reviewed in terms of;

Radio-Frequency

The nRF24L01+ transceiver module is designed to operate in 2.4 GHz worldwide ISM frequency band and uses GFSK modulation for data transmission. The data transfer rate can be one of 250kbps, 1Mbps and 2Mbps.

Power Consumption

The operating voltage of the module is from **1.9 to 3.6V**, but the good news is that the **logic pins are 5-volt tolerant**, so we can easily connect it to an Arduino or any 5V logic microcontroller

without using any logic level converter. The module supports programmable output power 0dBm, -6dBm, -12dBm or -18dBm and consumes around **12mA during transmission** at 0dBm, which is even lower than a single LED. And best of all, it consumes 26 μ A in standby mode and 900nA at power down mode. That's why they're the go-to wireless device for low-power applications.

SPI Interface

The nRF24L01+ transceiver module communicates over a 4-pin Serial Peripheral Interface (**SPI**) with a maximum data rate of **10Mbps**. All the parameters such as frequency channel (125 selectable channels), output power (0dBm, -6dBm, -12dBm or -18dBm), and data rate (250kbps, 1Mbps, or 2Mbps) can be configured through SPI interface. The SPI bus uses a concept of a Master and Slave, in most common applications our Arduino is the Master and the nRF24L01+ transceiver module is the Slave. Unlike the I2C bus the number of slaves on the SPI bus is limited, on the Arduino Uno you can use a **maximum of two SPI slaves** i.e. two nRF24L01+ transceiver modules.

Table 1 NRF24L01 Transceiver Module Specification

Frequency Range	2.4GHz ISM Band
Maximum Air Rate	2Mb/s
Modulation Format	GFSK
Maximum Output Power	0dBm
Operating Supply Voltage	1.9v to 3.6v
Maximum Operating Current	13.5Ma
Minimum Current (Standby mode)	26 μ A
Logic Inputs	5v Tolerant
Communication Range	800+ m (line of sight)

Mode of Operation of the NRF24L01 Transceiver Module

RF Channel Frequency

The nRF24L01 Transceiver module transmits and receives data on a certain frequency called **Channel**. Also in order for two or more transceiver modules to communicate with each other, they need to be on the same channel. This channel could be any frequency in the 2.4 GHz ISM band or to be more precise, it could be between 2.400 to 2.525 GHz (2400 to 2525 MHz).

Each channel occupies a bandwidth of less than 1MHz. This gives us 125 possible channels with 1MHz spacing. So, the module can use 125 different channels which give a possibility to have a network of 125 independently working modems in one place. This is shown in Figure 3.11

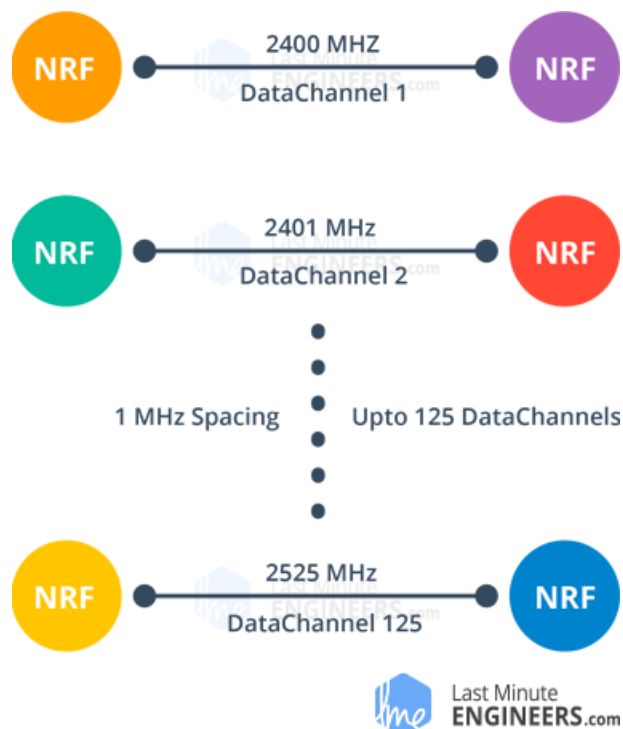


Figure 3.11 The Channel Frequency for theNRF24L01 Transceiver Module

Mode of Transaction between two NRF24L01 Transceiver Modules

Now there are three cases by which nRF2L01 module sends and receives data and these cases are considered below;

CASE 1: Transaction with acknowledgement and Interrupt - This is a positive case whereby the transmitter starts a communication by sending a data packet to the receiver. Once the whole packet of data is transmitted, it waits (around 130 μ s) for the acknowledgement packet (ACK packet) to receive. When then receiver then receives the packet, it sends ACK packet to the transmitter. On the receiving the ACK packet the transmitter asserts an Interrupt (IRQ) signal to indicate the new data is available. This is illustrated below in Figure 3.12



Figure 3.12 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 1

CASE 2: Transaction with data packet lost - This is a negative case where a retransmission is needed due to loss of the packet of data transmitted. After the packet is transmitted, the transmitter waits for the ACK packet to receive. If the transceiver doesn't get it within the Auto-Retransmit-Delay (ARD) time, the packet is retransmitted. When the retransmitted packet is received by the receiver, the ACK packet is transmitted which in turn generates the Interrupt at the transmitter. This is illustrated in the pictorial diagram below in Figure 3.13

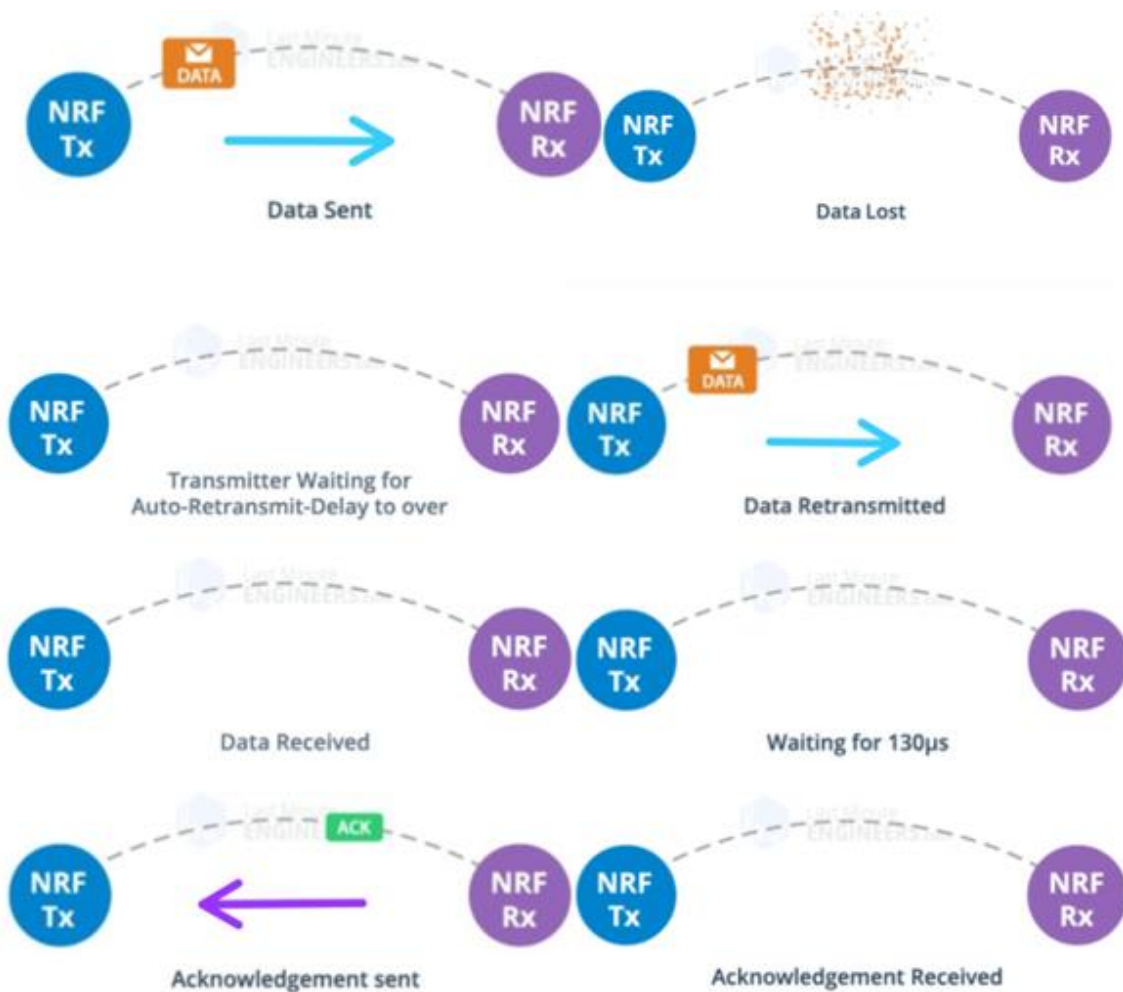


Figure 3.13 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 2

CASE 3: Transaction with acknowledgement lost - This is also a negative case where a retransmission is needed due to the loss of the ACK packet. Here, even if the receiver receives the packet of data in the first attempt, due to the loss of ACK packet the transmitter thinks the receiver has not gotten the packet at all. So, after the Auto-Retransmit-Delay time is over, it retransmits the packet. Now when the receiver receives the packet containing the same ID as previous, it discards it and sends ACK packet again. This is illustrated in the pictorial diagram below in Figure 3.14

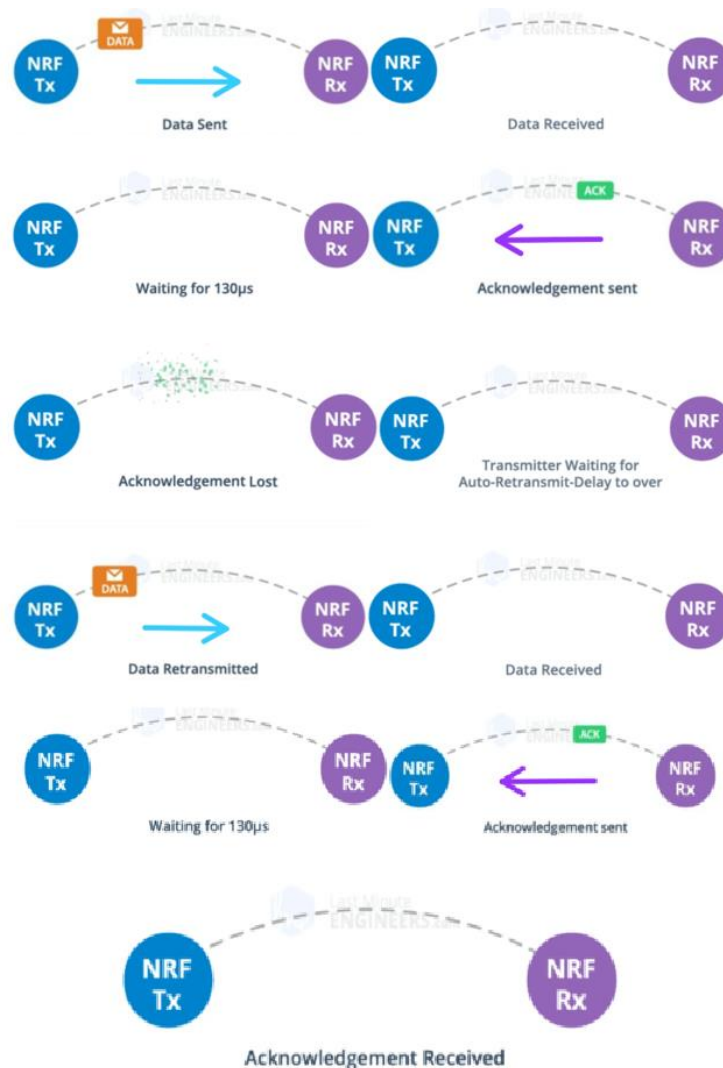


Figure 3.14 A pictorial diagram for how data is being transmitted and received between two NRF24L01 modules for Case 3

The NRF24L01 Transceiver Module Pinout

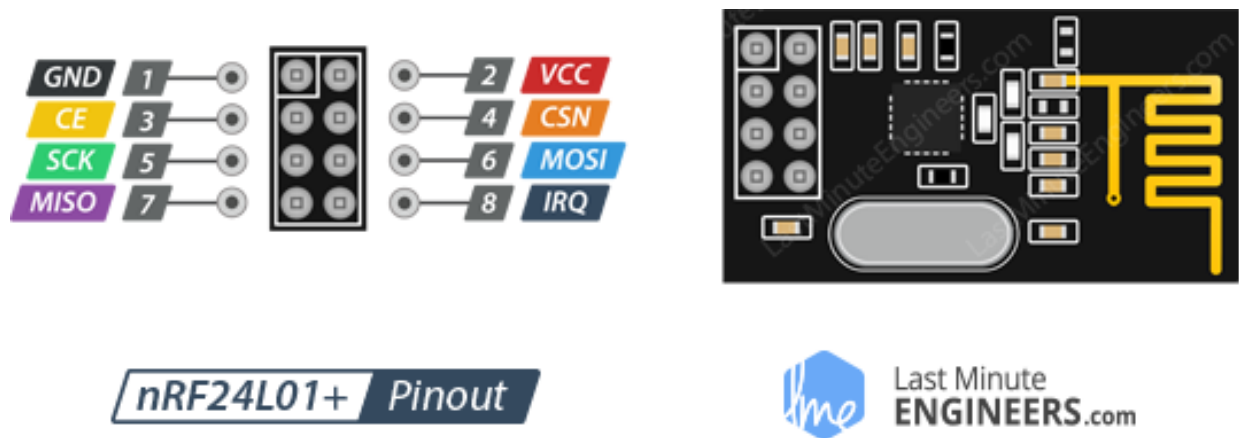


Figure 3.15 The NRF24L01 Transceiver module pinout

From Figure 3.15 above,

GND- is the Ground Pin. It is usually marked by encasing the pin in a square so it can be used as a reference for identifying the other pins.

VCC- supplies power for the module and this can be anywhere from 1.9 to 3.9volts.

CE (Chip Enable)- this is an active-HIGH pin when selected, the NRF24L01 will either transmit or receive, depending upon which mode it is currently in.

CSN (Chip Select Not)- is an active-LOW pin and is normally kept HIGH. When this pin goes low, the NRF24L01 begins listening on its SPI port for data and processes it accordingly.

SCK (Serial Clock)- this accepts clock pulses provided by the SPI bus master.

MOSI (Master out Slave in)- is an SPI input to the NRF24L01.

MISO (Master in Slave out)- is an SPI output from the NRF24L01.

IRQ- is an interrupt pin that can alert the master when new data is available to process.

Applications of the NRF24L01 Transceiver module

- Applicable in wireless home automation systems
- For Drone remote control systems
- Remote sensors for pressure and alarm systems
- Also applicable in Robot control and monitoring

3.2.4 DS3231 RTC Module

Real Time Clock or RTC is a time-keeping device in the form of an Integrated Circuit and it is an integral component of many critical applications and devices like Servers, GPS, Data Loggers, e.t.c. The DS3231 is an RTC IC that was developed by Maxim Integrated, an extremely accurate module with communication over I2C interface which can be interfaced with any microcontroller.

The DS3231 RTC module in Figure 3.16 keeps track of the time in seconds, minutes, hours, day, date, month and year information. That even the date at the end of the month is automatically adjusted for months with lesser days than 31 days, including corrections for leap year.

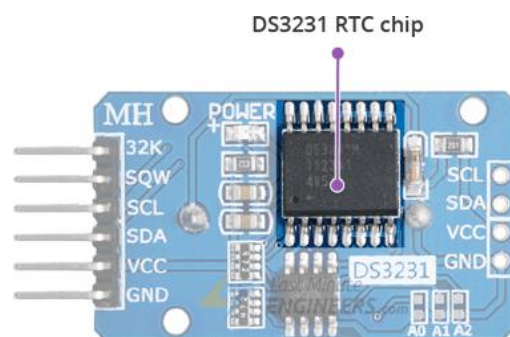


Figure 3.16 The DS3231 Real Time Clock Module

An interesting feature of the DS3231 RTC module is that it has an integrated crystal oscillator and a temperature sensor and hence doesn't need an external crystal, because most RTC modules come with an external 32kHz crystal for time-keeping but the problem with these crystals is that the external temperature can affect their oscillation frequency. So this inbuilt crystal oscillator usually called TCXO – Temperature Compensated Crystal Oscillator, can avoid such slight drifts in the crystal because it's highly immune to the external temperature changes. This sensor compensates the frequency changes by adding or removing clock ticks so that the time-keeping stays on track. That's the reason TCXO provides a stable and accurate reference clock and maintains the RTC to within ± 2 minutes per year accuracy. And this is illustrated graphically below;

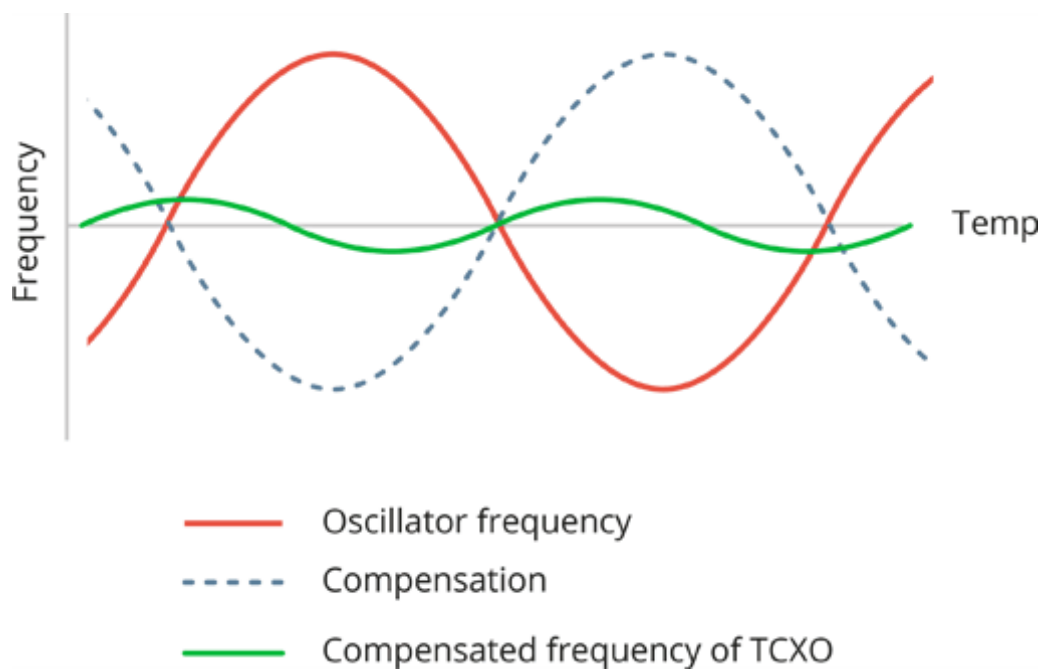


Figure 3.17 A Graphical representation of the compensated frequency of TCXO of a DS3231 RTC module

Battery Backup

The DS3231 RTC module also has a battery input that maintains an accurate time when the main power to the device is interrupted. The built-in power-sense circuit continuously monitors the status of VCC to detect power failures and automatically switches to the backup supply.

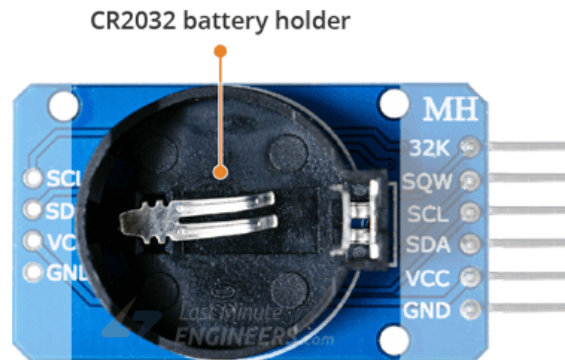


Figure 3.18 A CR2032 Battery holder attached to its back can hold a 20mm 3v lithium coin-cell CR2032 battery

So assuming the CR2032 battery is fully charged with a capacity of 220mAh and the DS3231 RTC chip consumes a minimum of 3 μ A, the battery can keep the RTC module running for a minimum of 8 years without an external 5V power supply.

$$220\text{mAh}/3\mu\text{A} = 73333.34 \text{ hours} = 3055.56 \text{ days} = 8.37 \text{ years}$$

DS32321 RTC Module Pinout

The DS3231 RTC module has a total of 6 pins that interfaces outside and the connections are as follows;

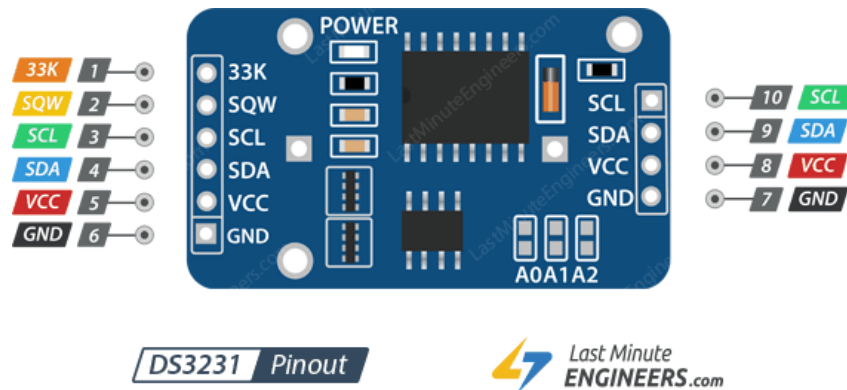


Figure 3.19 DS3231 RTC module Pinout

32k – Pin outputs a stable (temperature compensated) and accurate reference clock

SQW – Pin outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This is done so that it can be further used as an interrupt due to alarming conditions in many time-based applications.

SCL – This is a serial clock pin for I2C interface

SDA – This is a serial data pin for I2C interface

VCC – Pin supplies power for the module. It can be anywhere between 3.3v to 5.5v

GND – This is the ground pin.

3.2.5 SSD1306 0.96 Inch OLED Screen Display

The OLEDs, Organic Light Emitting Diodes are super-light, almost paper-thin, theoretically flexible display screens that produce brighter and crisper pictures than the Liquid Crystal Display screen (LCD). This screen display contains a powerful single chip which is the CMOS OLED driver controller – SSD1306. It communicates with the microcontroller in multiple ways include I2C and SPI. The SSD1306 OLED screen display comes in different sizes and colors, like the

128x64 OLED display, 128x32 OLED display, with white OLEDs, Blue OLEDs and Dual color OLEDs.



Figure 3.20 SSD1306 0.96 Inch OLED Screen Display

An OLED display works without a backlight because it makes its own light. This is why the display has such a high contrast, extremely wide viewing angle and can display deep black levels. Now the absence of backlight significantly reduces the power required to run the OLED. On an average, the display uses about 20mA current, although it depends on how much of the display is in it.

The operating voltage of the SSD1306 controller is from 1.65V to 3.3V while the OLED panel requires about 7 to 15V supply voltage. All these different power requirements are sufficed using internal charge pump circuitry. This makes it possible to connect it to an Arduino or any 5V logic microcontroller easily without using any logic level converter.

Regardless of the size of the size of the OLED module, the SSD1306 driver has a built-in 1KB Graphic Display Data RAM (GDDRAM) for the screen which holds the bit pattern to displayed. The 1K memory area is organized in 8 pages (0 to 7). Each page contains 128 columns/segments (block 0 to 127). And each column can store 8 bits of data (from 0 to 7). Each bit represents a

particular OLED pixel on the screen which can be turned ON and OFF programmatically. The 128x64 OLED screen displays all the contents of RAM whereas 128x32 OLED screen displays only 4 pages (half content) of the RAM. The whole 1K memory with pages, segments and data is shown in Figure 3.21 below.

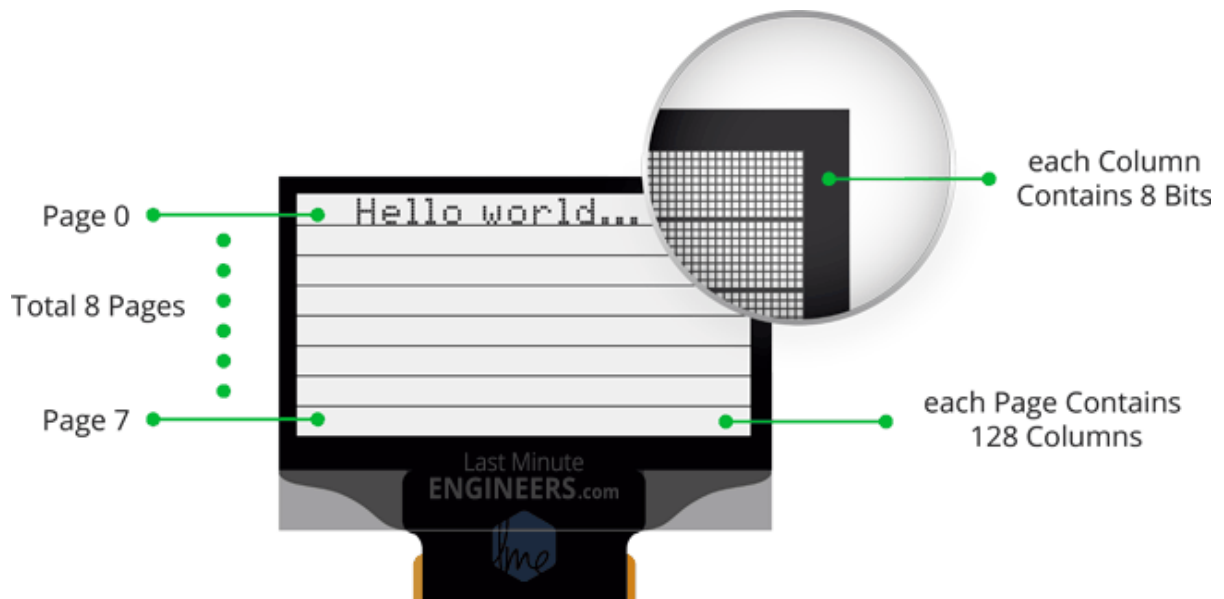


Figure 3.21 A 1K memory SSD1306 0.96 Inch OLED Screen Display with pages, segments and data SSD1306 0.96 OLED Screen Display Module Pinout

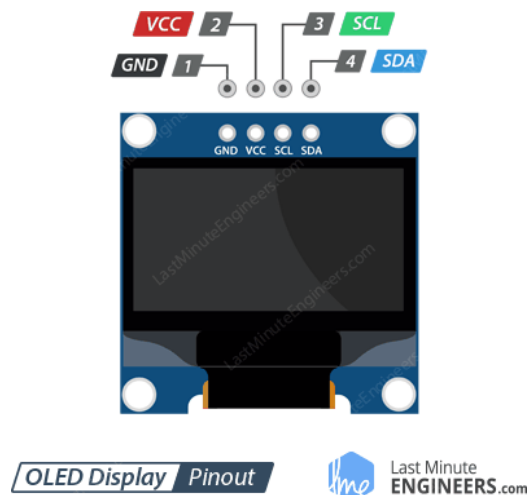


Figure 3.22 SSD1306 0.96 OLED Screen Display Module Pinout

GND – This pin is connected to the ground of the Arduino Nano.

VCC – This is the power supply for the display which is connected to the 5V pin on the Arduino Nano.

SCL – This is a serial clock pin for I2C interface.

SDA – This is a serial data pin for I2C interface.

3.2.6 SD Card Module

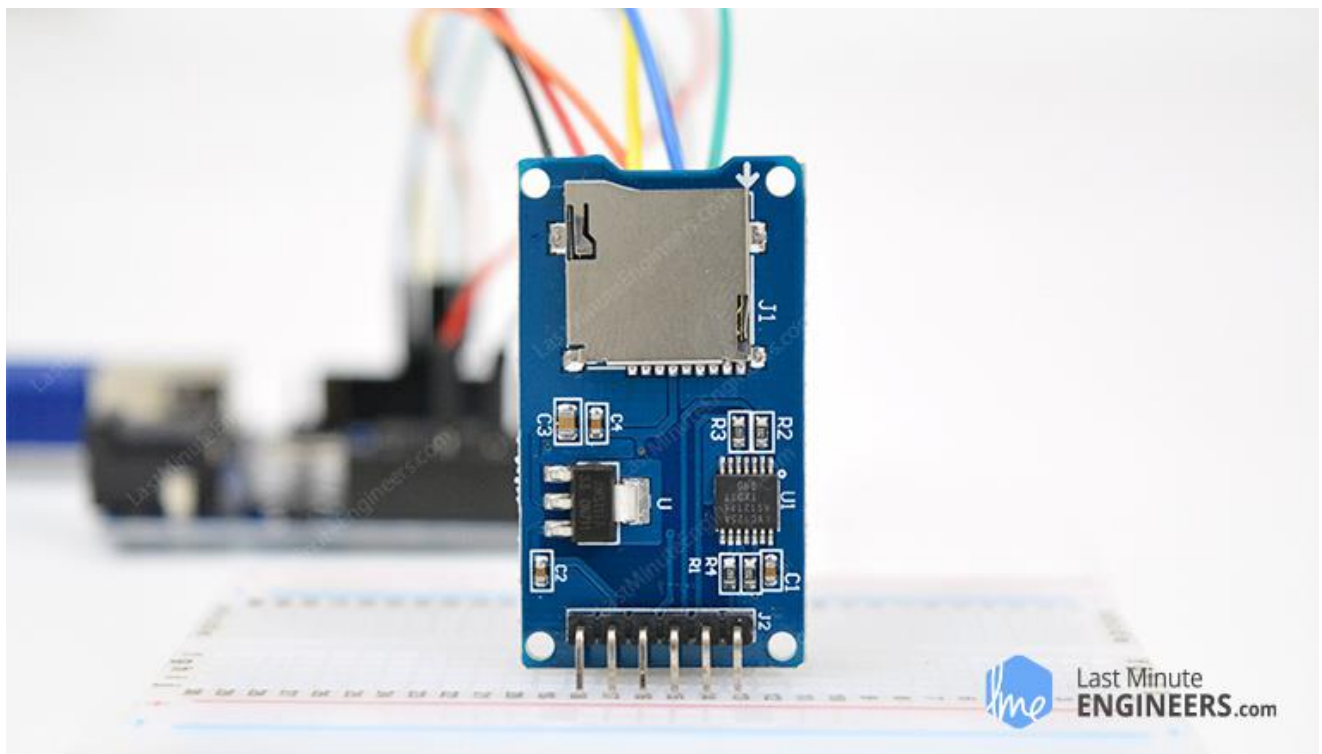


Figure 3.23 SD Card Module

The Micro SD card module contains two main components that make it easy to log data which are; the Level Shifter and the Voltage Regulator as shown in Figure 3.24

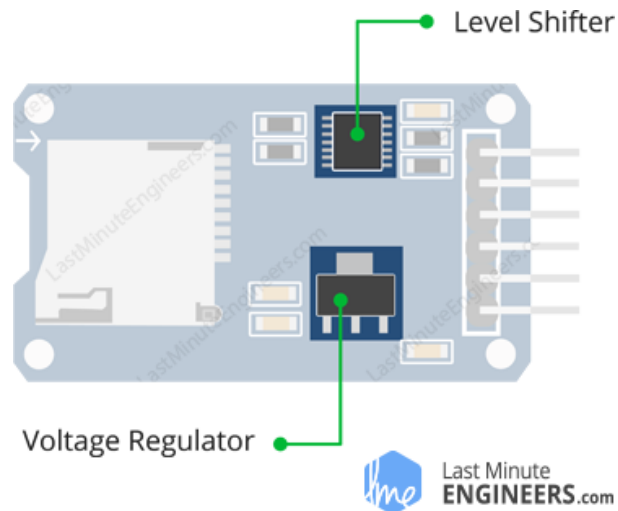


Figure 3.24 Two main components on the SD Card Module

The Operating voltage of any standard micro SD card is 3.3V and as such it cannot be connected directly to circuits that use 5V logic. In fact, any voltages exceeding 3.6V will permanently damage the micro SD card. That's why the module has an onboard ultra-low dropout regulator that will drop any voltage ranging from 3.3V – 6V down to ~3.3V.

There is also a 74LVC125A chip on the module that converts the interface logic from 3.3V – 5v to 3.3V. This is called Level Shifting. That means the SD card module can interact with both 3.3V and 5V microcontrollers like the Arduino Nano.

The SD card module also has two interface – the SPI and SDIO mode. The SDIO mode is faster and used in mobile phones, digital cameras and so on. But it is more complex and requires signing non-disclosure documents. Also, the SD card module is based on lower speed and less overhead which makes the SPI mode a better interface to use with any microcontroller.

Micro SD Card Module Pinout

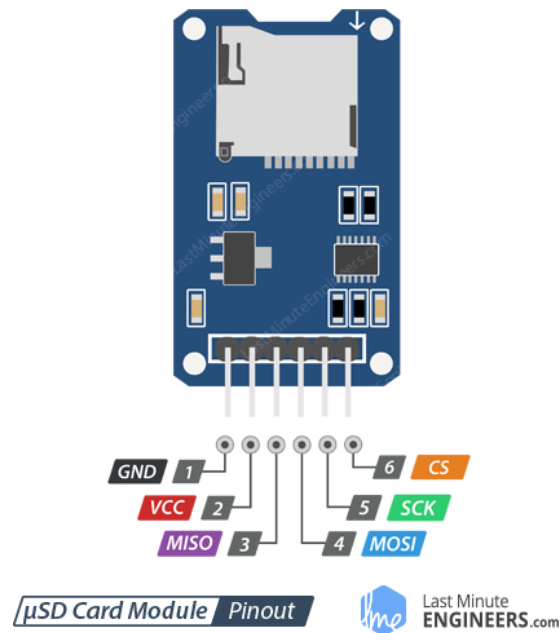


Figure 3.25 Micro SD Card Module Pinout

VCC – Pin supplies power to the module and is connected to 5V pin on the Arduino Nano.

GND – This is connected to the ground of the Arduino Nano.

MISO (Master In Slave Out) – This is the SPI output from the Micro SD Card Module.

MOSI (Master Out Slave In) – This is the SPI input from the Micro SD Card Module.

SCK (Serial Clock) – Pin accepts clock pulses which synchronizes data transmission generated by the Arduino Nano.

SS (Save Select) – Pin is used by the Arduino (Master) to enable and disable specific devices on SPI bus.

3.2.7 Custom Printed Circuit Boards (PCBs)

PCBs are Printed Circuit Boards that houses the electrical components or any system and keeps them organized, so they are built for circuit components to be soldered and placed on. Custom PCBs are convenient light weight boards that are built to suit the project or designed system unlike the white generic breadboards that are heavier and used in general purposes.

In order to design a PCB that can be manufactured and printed, an “EasyEDA” free online circuit design software tool was used to design in specification of the circuit diagram as shown from the beginning of this chapter for both the Indoor and Outdoor unit which has the design specifications and is as shown in the Figure 3.26 below;

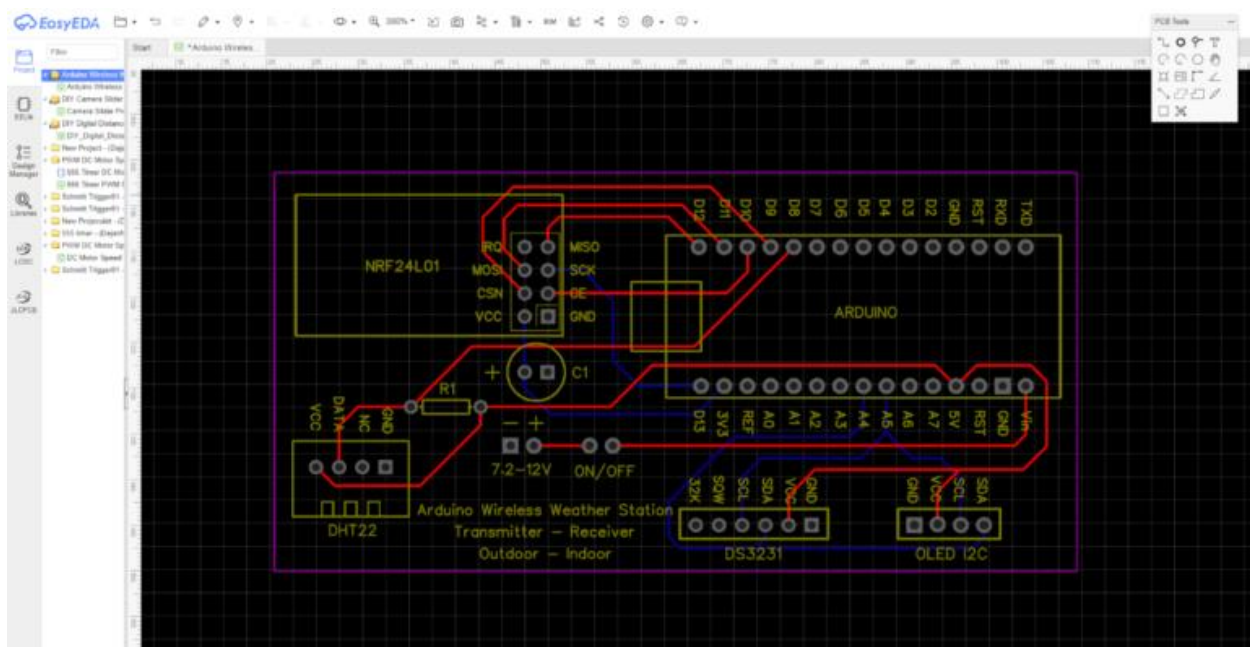


Figure 3.26 A designed PCB for the AWWLS circuit diagram (<https://www.easyeda.com>)

The designed PCB is now exported in a Gerber file and sent to a manufacturing company like JLCPCB to print and produce; and then can be ordered. The ordered PCBs would then look like this in Figure 3.27 when gotten.



Figure 3.27 The manufactured PCB design for the AWWLS circuit diagram

3.2.8 Power Jack

This is a 3 pin Printed Circuit Board (PCB) mount type Direct Current (DC) Female Connector also known as a DC Barrel Jack. It can be soldered on PCBs to be used as power sockets. These Barrel connectors can be plugged into power by using an Alternating Current (AC) Adaptor. Adaptors vary in power ratings and voltages of 9V, 12V and so on.

Some of these female barrel connectors or Jack have an additional contact that allows its application to detect whether a power supply is plugged into the barrel jack or not, thus allowing the device to bypass batteries and save battery life when running on external power.

From the diagram below in Figure 3.28, the 3 pins extending from the Jack are to be connected or soldered to connecting wires where the pin located at the back is connected to the positive voltage (+V), while the other 2 pins are connected to ground of the PCB.

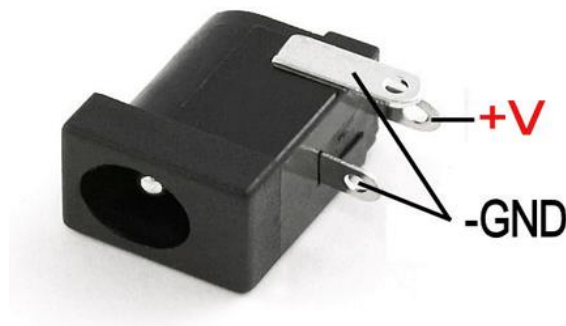


Figure 3.28 A Female Connector Power Jack

3.2.9 Power Switch

The Power Switch as shown below in Figure 3.28, is simply a circuit breaker, it allows or breaks current from flowing through. It has 3 terminals – the Ground terminal (negative), the positive terminal (Battery or power source) and the circuit breaker terminal. By soldering connecting wires to both the ground and positive terminals and plugging the power source, the switch can now be used to control the flow of current by breaking or allowing, a simple ON and OFF switching operation.

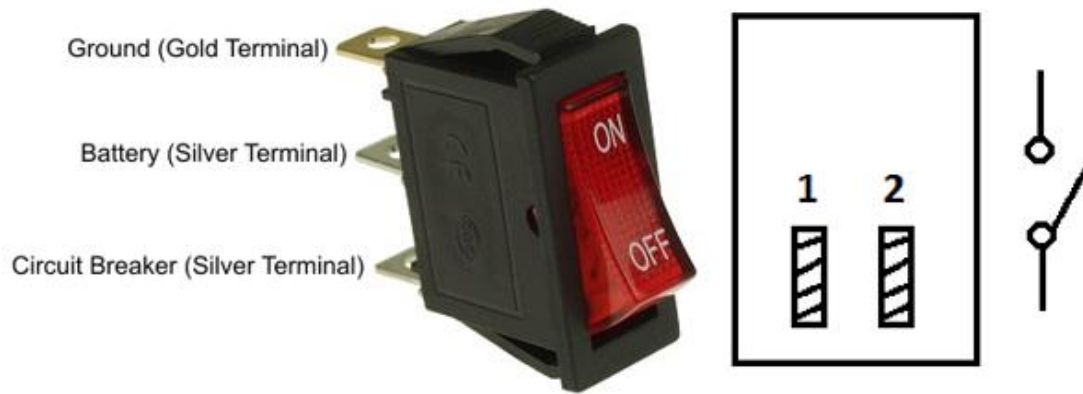


Figure 3.29 A Power Switch

3.2.10 18650 Li-ion Battery

The 18650 is a Lithium ion rechargeable battery, usually called 18650 cell. The 18650 has a voltage of 3.7V and has between 1800mAh and 3500mAh (mili-amp-hours). The 18650 has a charge time of about 4 hours. The 18650 Lithium ion battery can be used in flashlights, electronic devices, laptops, vaping and even electric vehicles.



Figure 3.30 18650 Lithium ion Battery

Other listed components are shown below;



Figure 3.31 A Dual 18650 Lithium ion Battery Holder



Figure 3.32 A 10 micro-farad Capacitor



Figure 3.33 A 220 ohms Resistor

3.3 System Design and Implementation

Having discussed the Circuitry components of the system and its functions, we begin coupling of the components and to begin with, we solder header pins to the custom Printed Circuit Boards (PCBs) as shown in the Figure 3.34

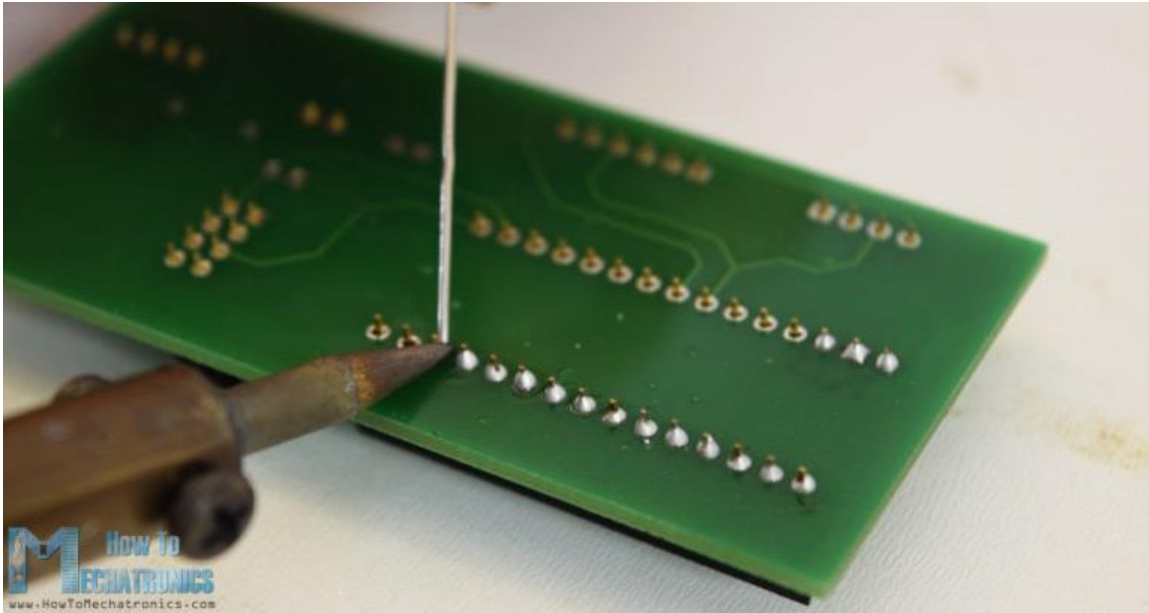


Figure 3.34 Soldering header pins to the PCB

After which, we start placing each circuit component carefully on both PCBs because they have the same schematics, that is for the Indoor and Outdoor systems as shown in Figure 3.35

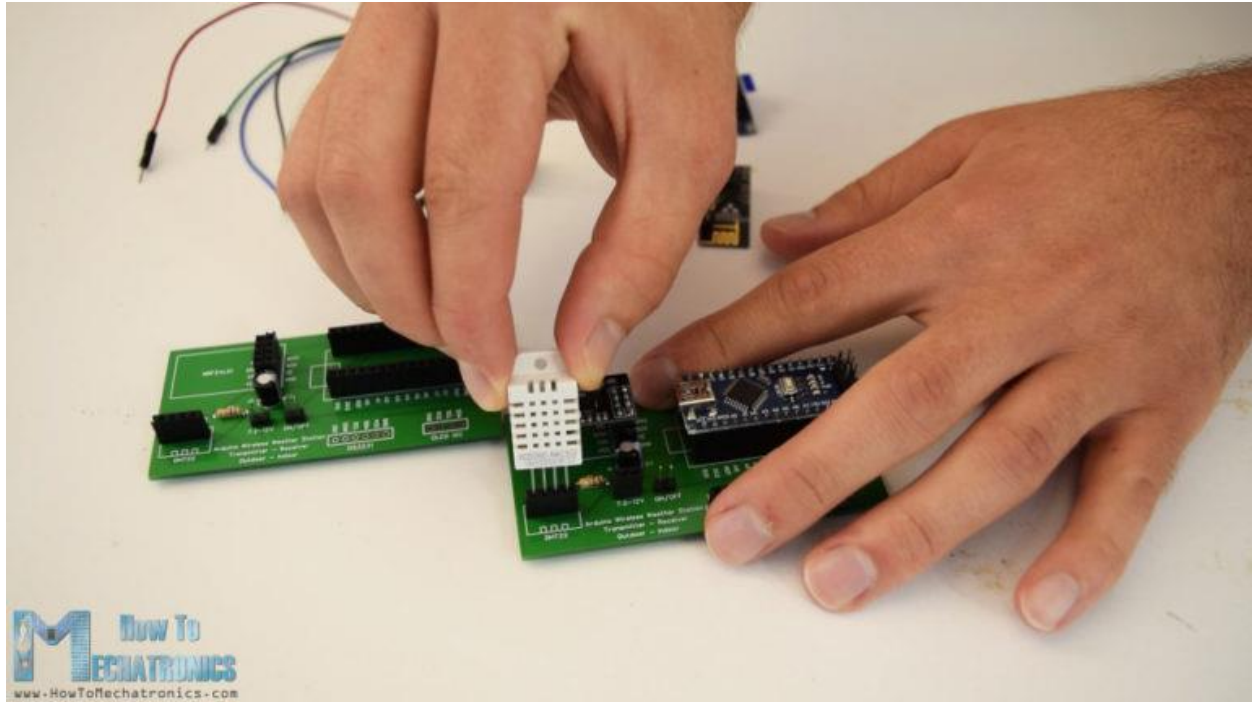


Figure 3.35 Placing the Circuit components on the PCBs (<https://www.howtomechatronics.com>)

On the Indoor board is placed the Arduino NANO, DHT22 sensor, DS3231 RTC module, the NRF24L01 Transceiver, the SSD1306 0.96' inch screen Display as well as the capacitor to keep the power supply stable and the pull up resistor to keep the DHT22 working properly (Dejan, 2018).

Also, on the Outdoor board is placed another set of Arduino NANO, DHT22 sensor, NRF24L01 Transceiver, the capacitor and resistor and this is all powered by 2 18650 Lithium ion batteries of 3.7V each and this is as shown in Figure 3.36

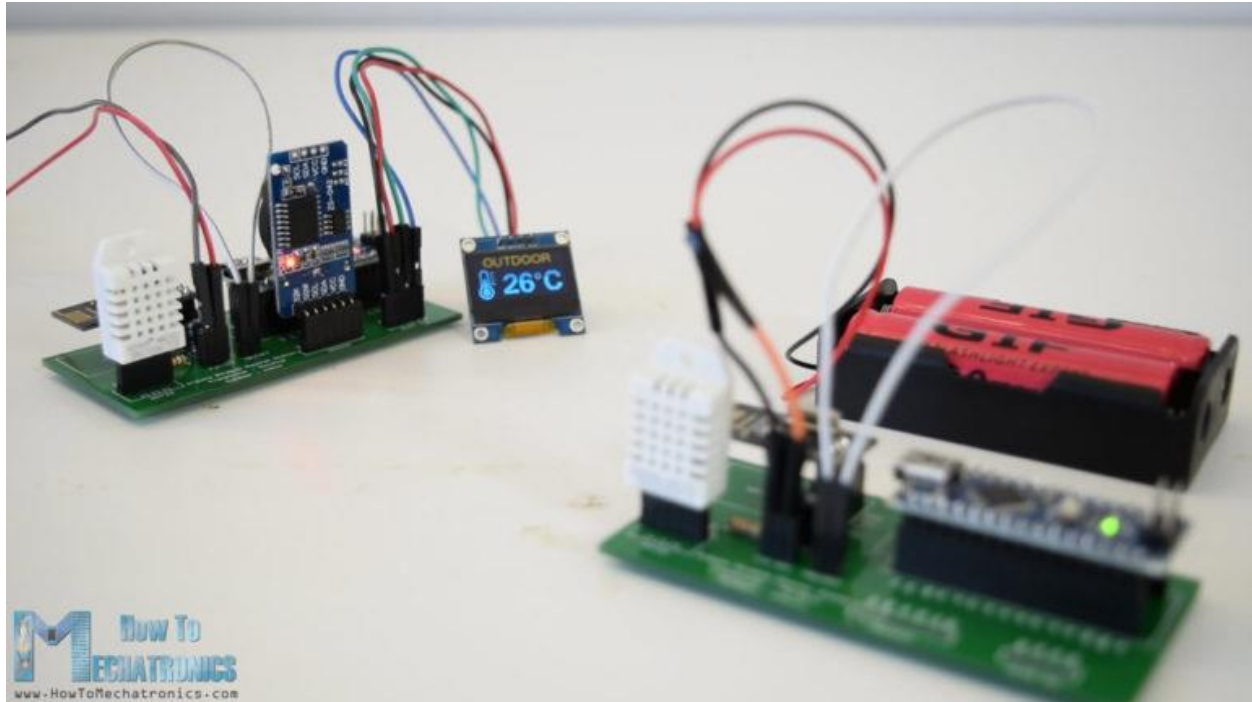


Figure 3.36 The Circuit components for the Indoor and Outdoor PCBs

CODE Implementation

In order to make the system work and begin to sense and get its readings, a programmable code is written, compiled and uploaded to the Arduino NANO to begin to process the information. These codes are written on the Arduino Integrated Development Environment (IDE) on a computer system and adequate libraries are added as well that are necessary for the compilation of the codes. Libraries such as DHT.h library for the DHT22 sensor, DS3231.h library for the DS3231 RTC module, U8G2LIB.h for the 0.96" OLED screen display, SD.h library for the SD card module and the RF24.h library for the NRF24L01 Transceiver module. The actual code can be found in the Appendix.

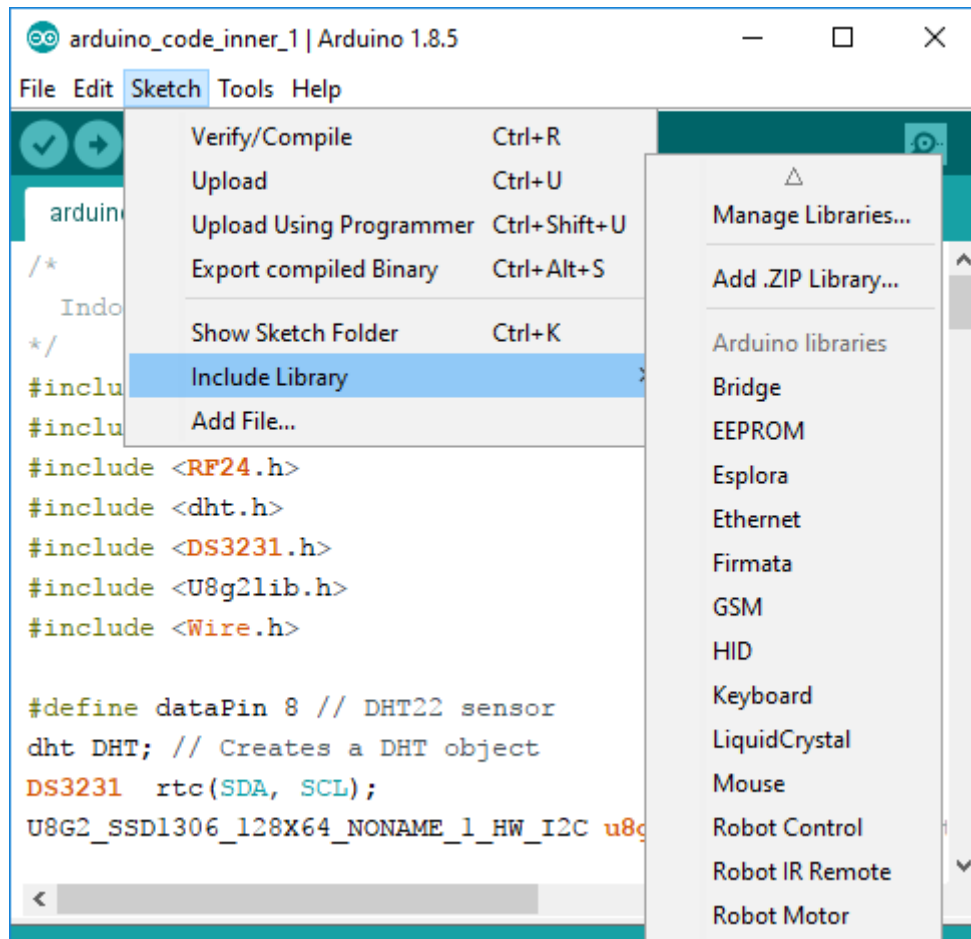


Figure 3.37 The Arduino IDE and the included Libraries

After the code is compiled and uploaded, the INDOOR and OUTDOOR units start to operate and is as shown in Figure 3.38 below;



Figure 3.38 The Indoor and Output System of the Automated Wireless Weather Logger Station
(<https://www.howtomechatronics.com>)

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

The temperature and humidity data in the SD card can be saved in a text (.txt) file format or in a (.csv) format in Excel Spreadsheet. In this project, the data was saved in Excel Sheet format. In the SD card module code initialization setup, the file type and format is declared. This initializes the SD card and the Arduino NANO begins to write the collected data from the sensors to it as shown in Figure 4.1

```
File myFile = SD.open("firstday.csv", FILE_WRITE);  
if (myFile) {
```

```
  Initializing SD card...card initialized.
```

Figure 4.1 Declaring the File name of the Logged data as firstday.csv and initializing the SD card to begin logging of its retrieved data

4.2 Results

After the SD card is been initialized as displayed on the serial monitor of the Arduino IDE, it immediately begins to log the data and display it as well on the serial monitor. The retrieved data for the temperature and humidity are logged at 1 second intervals. The data are arranged by Date,

Time, Temperature (°C), and Humidity (%). This is as shown on the serial monitor in Figure 4.2 below

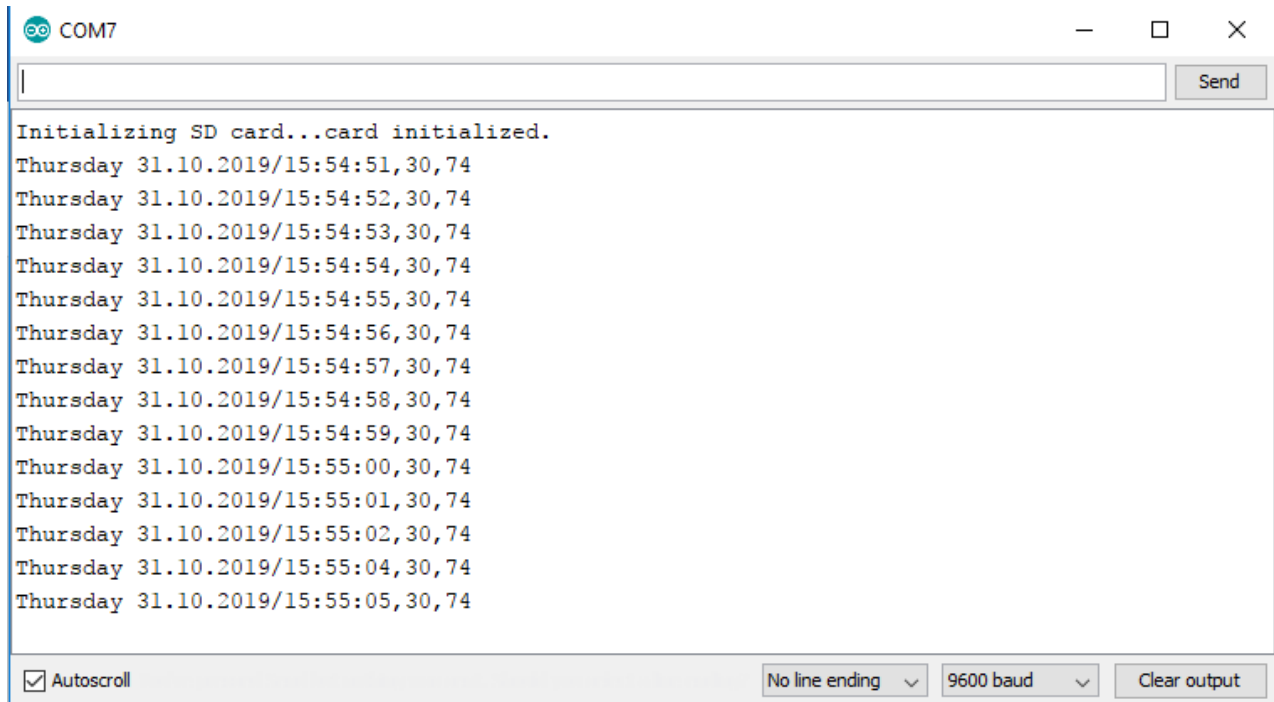


Figure 4.2 The Serial monitor displaying the logged data at real time at the interval of one second every hour showing the Day, Date, Time, Temperature in °C and Humidity in %

The data can then be transferred to an Excel Spreadsheet software for easy accessibility and usability. First, the micro-SD card is ejected from its module and placed in an adapter connected to a computer system for transfer its data.

A sample of the data stored in a text file is shown in Figure 4.3

Date and Time	Temperature (°C)	Humidity(%)
Thursday 31.10.2019/03:11:29	30	88
Thursday 31.10.2019/03:11:30	30	88
Thursday 31.10.2019/03:11:31	30	88
Thursday 31.10.2019/03:11:32	30	88
Thursday 31.10.2019/03:11:34	30	88
Thursday 31.10.2019/03:11:35	30	88
Thursday 31.10.2019/03:11:36	30	88
Thursday 31.10.2019/03:11:37	30	88
Thursday 31.10.2019/03:11:38	30	88
Thursday 31.10.2019/03:11:39	30	88
Thursday 31.10.2019/03:11:40	30	88
Thursday 31.10.2019/03:11:41	30	88
Thursday 31.10.2019/03:11:42	30	88
Thursday 31.10.2019/03:11:43	30	88
Thursday 31.10.2019/03:11:44	30	88
Thursday 31.10.2019/03:11:45	30	88
Thursday 31.10.2019/03:11:46	30	88
Thursday 31.10.2019/03:11:47	30	88
Thursday 31.10.2019/03:11:48	30	88
Thursday 31.10.2019/03:11:49	30	88
Thursday 31.10.2019/03:11:50	30	88
Thursday 31.10.2019/03:11:51	29	87
Thursday 31.10.2019/03:11:52	29	87
Thursday 31.10.2019/03:11:53	30	88
Thursday 31.10.2019/03:11:54	30	88
Thursday 31.10.2019/03:11:56	30	88
Thursday 31.10.2019/03:11:57	29	87
Thursday 31.10.2019/03:11:58	29	87

Figure 4.3 Sample of the data stored in a text file

The temperature and humidity data logged for Thursday 31 October 2019, 03:11 – 11:00 UT is plotted and shown in Figure 4.4 and Figure 4.5

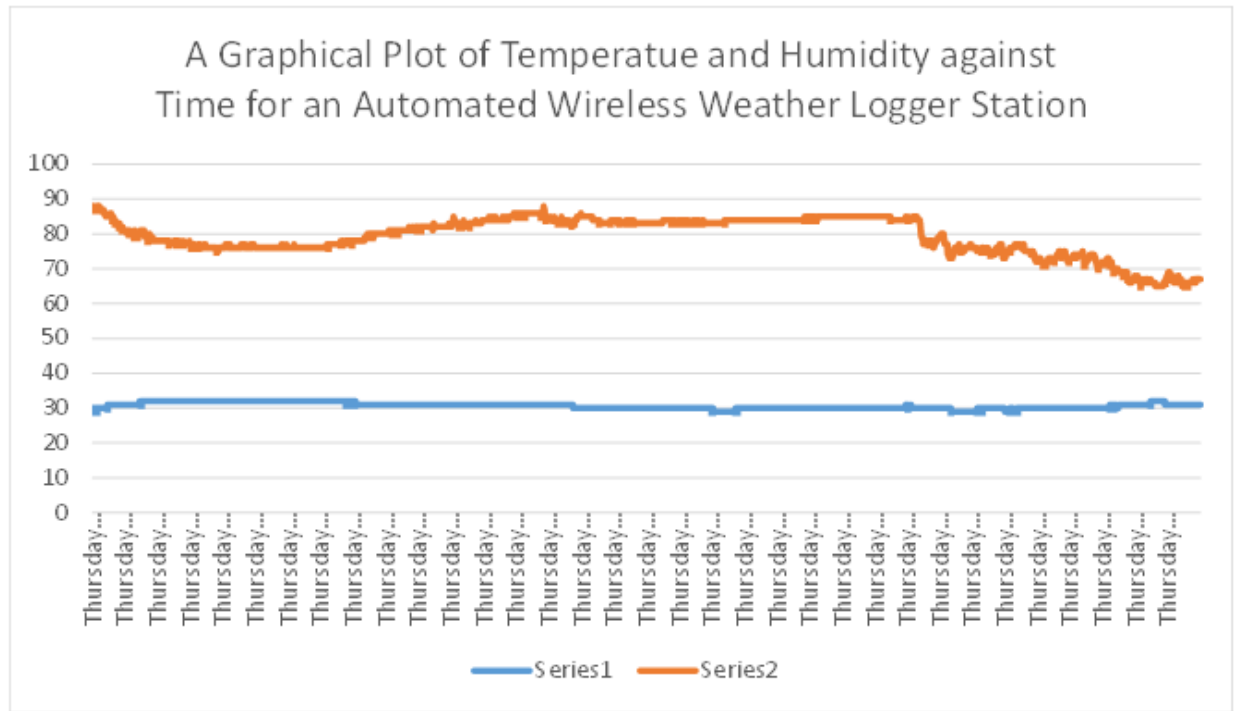


Figure 4.4 A plot of the Temperature and Humidity against Time

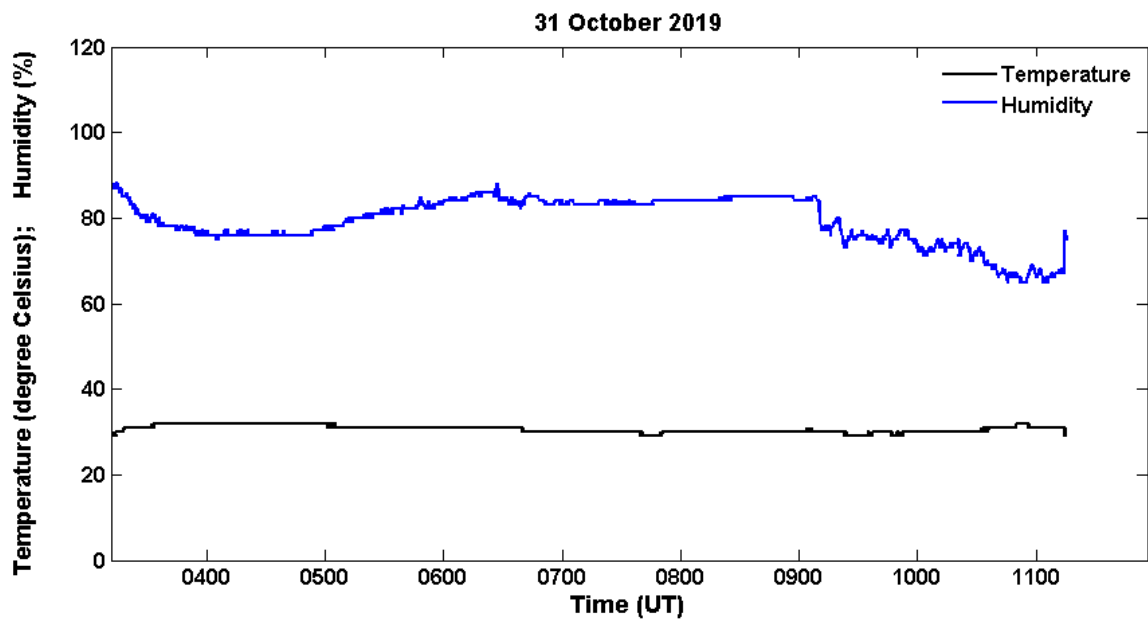


Figure 4.5 A plot of the Temperature and Humidity against Time

4.3 Discussion

The data shown in Figure 4.4 and Figure 4.5 was collected over 7 hours (03:11 – 11:11 UT). An estimate of over 27,600 data points were recorded. In Figure 4.4 and Figure 4.5, it can be noticed that the peak in humidity occurred between (03:11 – 03:34 UT) and as time progressed, humidity began to fall while temperature increased. This observation can be attributed to/explained by the fact that as the atmospheric heat from the sun in the surrounding environment begins to rise as the day breaks, the water vapor in the air begins to drop. Meaning that Humidity reduces as the temperature increases. It also shows that only at rainy days will the relative humidity be at its maximum (100%) and at sunny days it drops. The average temperature during this interval was computed as $\sim 30.6^{\circ}\text{C}$ while the average humidity was $\sim 79.3\%$.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Summary and Conclusion

This project demonstrated the implementation of AWWLS to read temperature and humidity values from sensors in both indoor and outdoor units. The data were logged in an SD card in real time and a sample plot from the measurements was shown.

The developed Automated Wireless Weather Logger Station is simple, stable, low cost, and of low power consumption.

5.2 Recommendations

A major drawback during the course of this project was that the data acquired by the AWWLS could not be validated by data acquired from other standard weather monitors. The only weather station in the vicinity where the AWWLS was deployed, in the Department of Physics University of Lagos (UNILAG), did not have data available during the period of investigation. It is therefore recommended that the temperature and humidity sensors in the weather station at the Department of Physics, UNILAG be fixed or replaced in the shortest possible time. Now based on the AWWLS developed system, it is also recommended that;

1. Metrology Agency Technical experts resort to designing and building real time automatic weather station such as the AWWLS which provides easy and remote accessibility because of its simplicity in design structure and mobility.

2. More values for temperature and humidity readings can be logged for several days and months to see the variations in temperature rise and fall, and properly analyze and give a structured report on weather condition for the month or year
3. Apart from temperature and humidity reports, other elements of weather can also be retrieved to give a well detailed report. Elements such as Air pressure and water level precipitate as discussed earlier in chapter one. These elements can be detected by their respective sensors – the pressure sensor and water level sensor. These sensors can equally be added to the design setup of the AWWLS to perform its weather operations.

REFERENCES

- Abubakar, I. M. and M.B. Sulaiman, (2018)** Micro-controller Based Mobile Weather Monitor System, *American Journal of Embedded Systems and Applications* **6(1)**, 23-29.
- Acurite (2019).** History of Home Weather Station. Retrieved from <https://www.acurite.com/learn/history-of-home-weather-stations/>
- Ahmad N. B. (2014)** Wireless Weather Station by using Zigbee. Retrieved from <https://eprints.utm.edu.my/16234/>
- Dejan (2018).** Arduino Wireless Weather Station. Retrieved from <https://howtomechatronics.com/tutorials/arduino/arduino-wireless-weather-station-project/>
- Rameshbabu, K., Misay, Mangesthu, Melkapu, Belete, Goshuion, (2018)** Design & Implementation of Automatic Weather Station with wireless Sensor network system. *Journal of Emerging Technologies and Innovative Research* **5(6)**, 493-502.
- Last Minute Engineers, (2019)** Interfacing Micro SD card Module with Arduino. Retrieved from <https://lastminuteengineers.com/arduino-micro-sd-card-module-tutorial/>
- Mark, R. and W. Geoff, (2011)** Weather Monitoring. Retrieved from <https://www.awemagazine.com/article/weather-monitoring-june-2011--99/>
- Muhammad, A. B. (2015)** Hydrology Study and trend of Rainfall event at Ump Gambang. Retrieved from <http://umpir.ump.edu.my/id/eprint/12155/>
- Nisha, G., G. Varsha, K. Sonali, T. Archana, (2015)** Zigbee Based Weather Monitoring System. *The International Journal of Engineering and Science* **4(4)**, 61-66.

Moje, R. K., A. Komal, B. Supriya, (2017) A Zigbee Based Smart Wireless Sensor Network for Monitoring an Agricultural Environment. *International Journal of Advance Research and Innovative Ideas in Education* **3(2)**, 4527-4533.

Hocking, W. K. (2000) The Instruments of Metrology. Retrieved from
<http://www.physics.uwo.ca/~whocking/p103/instrum.html>

APPENDIX

The Outdoor AWWLS code

```
/*  
  Outdoor unit - Transmitter  
*/  
  
#include <SPI.h>  
#include <nRF24L01.h>  
#include <RF24.h>  
#include <dht.h>  
#include <LowPower.h>  
  
#define dataPin 8 // DHT22 data pin  
dht DHT; // Creates a DHT object  
RF24 radio(10, 9); // CE, CSN  
const byte address[6] = "00001";  
char thChar[32] = "";  
String thString = "";  
  
void setup() {  
  Serial.begin(115200);  
  radio.begin();  
  radio.openWritingPipe(address);  
  radio.setPALevel(RF24_PA_MIN);  
  radio.stopListening();  
}  
  
void loop() {  
  int readData = DHT.read22(dataPin); // Reads the data from the sensor  
  int t = DHT.temperature; // Gets the values of the temperature  
  int h = DHT.humidity; // Gets the values of the humidity  
  thString = String(t) + String(h);  
  thString.toCharArray(thChar, 12);  
  Serial.println(thChar);  
}
```

```

// Sent the data wirelessly to the indoor unit
for (inti = 0; i<= 3; i++) { // Send the data 3 times
radio.write(&thChar, sizeof(thChar));
delay(50);
}
// Sleep for 2 minutes, 15*8 = 120s
for (intsleepCounter = 15; sleepCounter> 0; sleepCounter--)
{
LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
}
}

```

The Indoor AWWLS code

```

/*
Indoor unit - Receiver
*/

#include <SPI.h>

#include <nRF24L01.h>

#include <RF24.h>

#include <dht.h>

#include <DS3231.h>

#include <U8g2lib.h>

#include <Wire.h>

#define dataPin 8 // DHT22 sensor

dht DHT; // Creates a DHT object

```

```

DS3231 rtc(SDA, SCL);

U8G2_SSD1306_128X64_NONAME_1_HW_I2C      u8g2(U8G2_R0,      /*      reset=*/
U8X8_PIN_NONE);

RF24 radio(10, 9); // CE, CSN

const byte address[6] = "00001";

char text[6] = "";

int readDHT22, t, h;

String inTemp, inHum, outTemp, outHum;

String rtcTime, rtcDate;

int draw_state = 0;

unsigned long previousMillis = 0;

long interval = 3000;

#define Temperature_20Icon_width 27

#define Temperature_20Icon_height 47

static const unsigned char Temperature_20Icon_bits[] U8X8_PROGMEM = {

    0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0x00, 0x00, 0x80, 0x7f, 0x00, 0x00,

    0xc0, 0xe1, 0x00, 0x00, 0xe0, 0xc0, 0x01, 0x00, 0x60, 0x80, 0xf9, 0x03,

    0x60, 0x80, 0x01, 0x00, 0x60, 0x80, 0x01, 0x00, 0x60, 0x80, 0x79, 0x00,

    0x60, 0x80, 0x01, 0x00, 0x60, 0x80, 0x01, 0x00, 0x60, 0x80, 0xf9, 0x03,

    0x60, 0x80, 0x01, 0x00, 0x60, 0x80, 0x01, 0x00, 0x60, 0x8c, 0x79, 0x00,

    0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0xf9, 0x03,

    0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x79, 0x00,

    0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0xf9, 0x03,

```

```

0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x01, 0x00, 0x60, 0x9e, 0x01, 0x00,
0x70, 0x9e, 0x03, 0x00, 0x38, 0x1e, 0x07, 0x00, 0x18, 0x3e, 0x0e, 0x00,
0x1c, 0x3f, 0x0c, 0x00, 0x0c, 0x7f, 0x18, 0x00, 0x8c, 0xff, 0x18, 0x00,
0x8e, 0xff, 0x38, 0x00, 0xc6, 0xff, 0x31, 0x00, 0xc6, 0xff, 0x31, 0x00,
0xc6, 0xff, 0x31, 0x00, 0x8e, 0xff, 0x38, 0x00, 0x8c, 0xff, 0x18, 0x00,
0x0c, 0x7f, 0x1c, 0x00, 0x3c, 0x1c, 0x0e, 0x00, 0x78, 0x00, 0x06, 0x00,
0xe0, 0x80, 0x07, 0x00, 0xe0, 0xff, 0x03, 0x00, 0x80, 0xff, 0x00, 0x00,
0x00, 0x1c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00

```

```
};
```

```
#define Humidity_20Icon_width 27
```

```
#define Humidity_20Icon_height 47
```

```

static const unsigned char Humidity_20Icon_bits[] U8X8_PROGMEM = {
    0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00, 0x00, 0x70, 0x00, 0x00,
    0x00, 0x70, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0xdc, 0x00, 0x00,
    0x00, 0xdc, 0x01, 0x00, 0x00, 0x8e, 0x01, 0x00, 0x00, 0x86, 0x03, 0x00,
    0x00, 0x06, 0x03, 0x00, 0x00, 0x03, 0x07, 0x00, 0x80, 0x03, 0x06, 0x00,
    0x80, 0x01, 0x0c, 0x00, 0xc0, 0x01, 0x1c, 0x00, 0xc0, 0x00, 0x18, 0x00,
    0xe0, 0x00, 0x38, 0x00, 0x60, 0x00, 0x30, 0x00, 0x70, 0x00, 0x70, 0x00,
    0x30, 0x00, 0xe0, 0x00, 0x38, 0x00, 0xc0, 0x00, 0x18, 0x00, 0xc0, 0x01,
    0x1c, 0x00, 0x80, 0x01, 0x0c, 0x00, 0x80, 0x03, 0x0e, 0x00, 0x80, 0x03,
    0x06, 0x00, 0x00, 0x03, 0x06, 0x00, 0x00, 0x03, 0x07, 0x00, 0x00, 0x07,
    0x03, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x06, 0x03, 0x00, 0x00, 0x06,
    0x63, 0x00, 0x00, 0x06, 0x63, 0x00, 0x00, 0x06, 0x63, 0x00, 0x00, 0x06,

```



```

0xe3, 0x00, 0x00, 0x06, 0xc7, 0x00, 0x00, 0x06, 0xc6, 0x01, 0x00, 0x07,
0x86, 0x03, 0x00, 0x03, 0x0e, 0x1f, 0x00, 0x03, 0x0e, 0x1e, 0x80, 0x01,
0x1c, 0x00, 0xc0, 0x01, 0x38, 0x00, 0xe0, 0x00, 0x78, 0x00, 0x70, 0x00,
0xf0, 0x00, 0x38, 0x00, 0xe0, 0x07, 0x1f, 0x00, 0x80, 0xff, 0x0f, 0x00,
0x00, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00

};

void setup() {

  radio.begin();

  radio.openReadingPipe(0, address);

  radio.setPALevel(RF24_PA_MIN);

  radio.startListening();

  u8g2.begin();

  rtc.begin();

  // The following lines can be uncommented to set the date and time

  //rtc.setDOW(SATURDAY); // Set Day-of-Week to SUNDAY

  //rtc.setTime(10, 16, 0); // Set the time to 10:16:00 (24hr format)

  //rtc.setDate(5, 10, 2019); // Set the date to September 25th, 2019

}

void loop() {

  if (radio.available()) {

    radio.read(&text, sizeof(text)); // Read incoming data

    outTemp = String(text[0]) + String(text[1]) + char(176) + "C"; // Outdoor Temperature

    outHum = String(text[2]) + String(text[3]) + "%"; // Outdoor Humidity

```

```

}

unsigned long currentMillis = millis();

if (currentMillis - previousMillis > interval) {

    previousMillis = currentMillis;

    u8g2.firstPage();

    do {

        switch (draw_state ) {

            case 0: drawDate(); break;

            case 1: drawInTemperature(); break;

            case 2: drawInHumidity(); break;

            case 3: drawOutTemperature(); break;

            case 4: drawOutHumidity(); break;

        }

    } while ( u8g2.nextPage() );

    draw_state++;

    if (draw_state > 4) {

        draw_state = 0;

    }

}

}

void drawDate() {

    String dowa = rtc.getDOWStr();

    dowa.remove(3);

```

```

rtcDate = dow + " " + rtc.getDateStr();

u8g2.setFont(u8g2_font_timB14_tr);

u8g2.setCursor(0, 15);

rtcTime = rtc.getTimeStr(); // DS3231 RTC time

rtcTime.remove(5);

u8g2.print(rtcDate);

u8g2.setFont(u8g2_font_fub30_tf);

u8g2.setCursor(8, 58);

u8g2.print(rtcTime);

}

void drawInTemperature() {

  readDHT22 = DHT.read22(dataPin); // Reads the data from the sensor

  t = DHT.temperature; // Gets the values of the temperature

  inTemp = String(t) + char(176) + "C";

  u8g2.setFont(u8g2_font_helvR14_tr);

  u8g2.setCursor(24, 15);

  u8g2.print(F("INDOOR"));

  u8g2.setFont(u8g2_font_fub30_tf);

  u8g2.setCursor(36, 58);

  u8g2.print(inTemp);

  u8g2.drawXBMP( 0, 17, Temperature_20Icon_width, Temperature_20Icon_height,
Temperature_20Icon_bits);

}

```

```

void drawInHumidity() {

    h = DHT.humidity; // Gets the values of the humidity

    inHum = String(h) + "%";

    u8g2.setFont(u8g2_font_helvR14_tr);

    u8g2.setCursor(24, 15);

    u8g2.print(F("INDOOR"));

    u8g2.setFont(u8g2_font_fub30_tf);

    u8g2.setCursor(36, 58);

    u8g2.print(inHum);

    u8g2.drawXBMP(    0,    17,    Humidity_20Icon_width,    Humidity_20Icon_height,
Humidity_20Icon_bits);

}

void drawOutTemperature() {

    u8g2.setFont(u8g2_font_helvR14_tr);

    u8g2.setCursor(12, 15);

    u8g2.print(F("OUTDOOR"));

    u8g2.setFont(u8g2_font_fub30_tf);

    u8g2.setCursor(36, 58);

    u8g2.print(outTemp);

    u8g2.drawXBMP(    0,    17,    Temperature_20Icon_width,    Temperature_20Icon_height,
Temperature_20Icon_bits);

}

void drawOutHumidity() {

```

```

u8g2.setFont(u8g2_font_helvR14_tr);

u8g2.setCursor(12, 15);

u8g2.print(F("OUTDOOR"));

u8g2.setFont(u8g2_font_fub30_tf);

u8g2.setCursor(36, 58);

u8g2.print(outHum);

u8g2.drawXBMP(    0,    17,    Humidity_20Icon_width,    Humidity_20Icon_height,
Humidity_20Icon_bits);

}

```

The SD card logging code

```

#include <dht.h>

#include <SD.h>

#include <SPI.h>

#include <DS3231.h>

#define dataPin 8 // DHT22 sensor

dht DHT; // Creates a DHT object

File myFile;

DS3231 rtc(SDA, SCL);

Time t;

int chipSelect = 4; // Pin 10 on Arduino Uno

int readDHT22, temp, h;

void setup() {

    Serial.begin(9600);

```

```

Serial.print("Initializing SD card...");

// see if the card is present and can be initialized:

if (!SD.begin(chipSelect)) {

    Serial.println("Card failed, or not present");

    // don't do anything more:

    return;

}

Serial.println("card initialized.");

rtc.begin();

// The following lines can be uncommented to set the date and time
//rtc.setDOW(THURSDAY);    // Set Day-of-Week to SUNDAY
//rtc.setTime(3, 10, 0);    // Set the time to 10:16:00 (24hr format)
//rtc.setDate(31, 10, 2019); // Set the date to September 25th, 2019
//
}

void loop() {

    readDHT22 = DHT.read22(dataPin); // Reads the data from the sensor

    temp = DHT.temperature; // Gets the values of the temperature

    h = DHT.humidity; // Gets the values of the humidity

    String dow = rtc.getDOWStr();

    String rtcDate = dow + " " + rtc.getDateStr();

    String rtcTime = rtc.getTimeStr();

    File myFile = SD.open("firstday.csv", FILE_WRITE);

```

```
if (myFile) {  
  
    myFile.println(String(rtcDate)+"/"+String(rtcTime)+ "," + String(temp) + "," + String(h));  
  
    Serial.println(String(rtcDate)+"/"+String(rtcTime)+ "," + String(temp) + "," + String(h));  
  
    myFile.close();  
  
    // print to the serial port too:  
  
}  
  
// if the file didn't open, print an error:  
  
else {  
  
    Serial.println("error opening firstday.csv");  
  
}  
  
delay(1000);  
  
}
```