|  |
| --- |
| [Type the company name] |
| Microcontroller Boards |
| 8051(AT89C51RD2), PIC16877A, ARM(LPC1768), Ardurino UNO, Node MCU, Raspberry pi |

|  |
| --- |
| Richa Bajpai  5/30/2019 |

Arduino Uno

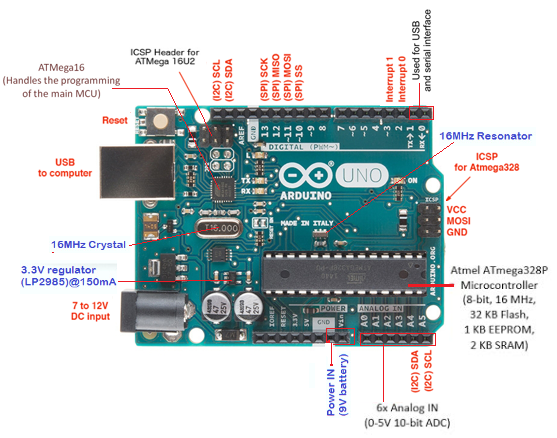


Arduino boards are able to read analog or digital input signals from different sensors and turn it into an output such as activating a motor, turning LED on/off, connect to the cloud and many other actions.

Arduino does not need an extra piece of hardware (called a programmer) in order to load a new code onto the board. You can simply use a USB cable.

Arduino IDE uses a simplified version of C++ programming.

PinOut:



The **crystal oscillator** helps Arduino in dealing with time issues. Arduino calculate time, by using the crystal oscillator 16MHz.

**Reset** i.e., start your program from the beginning. You can reset the UNO board in two ways.

1. By using the reset button on the board.
2. By connecting an external reset button to the Arduino pin labelled RESET.

**Analog pins**

The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor.

**ICSP pin**

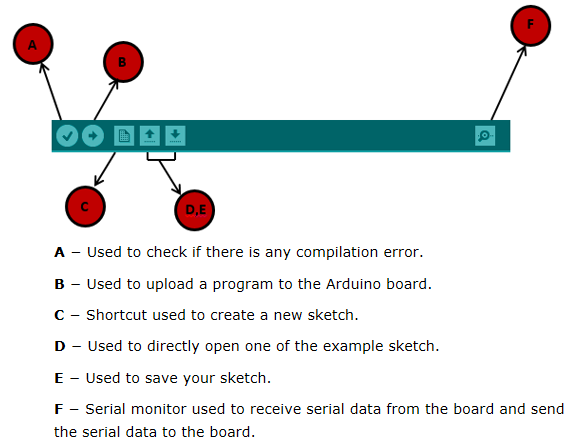
ICSP is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus.

**TX and RX LEDs**

The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

**AREF**

AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.



Arduino programs can be divided in three main parts: **Structure, Values** (variables and constants), and **Functions**.

**Structure**. Software structure consist of two main functions −

* Setup( ) function: this function is called when a sketch starts. It initializes and sets the initial values, and runs only once, after each power up or reset of the Arduino board.
* Loop( ) functionThis function loops consecutively, allowing the program to change and respond. Used to actively control the Arduino board.

**Data types** used during Arduino programming are:

1. **Void** keyword is used only in function declarations.
2. **Boolean** holds one of two values, true or false. It takes one byte of memory.
3. **Character** takes up one byte of memory that stores a character value. Character literals are written in single quotes like this: 'A' and for multiple characters, **Strings** use double quotes: "ABC".

millis() function:

This function is used to return the number of milliseconds at the time, the Arduino board begins running the current program i.e(it returns milliseconds from the start of the program.). This number overflows i.e. goes back to zero after approximately 50 days.

Sensors

# LM35 precision centigrade temperature sensor:

Provides analog output, the ADC of arduino used for converting analog value to digital approximation: using: ADC value= sample\* 1024/refrence voltage (+5V).

void loop() {

temp = analogRead(tempPin);

// read analog volt from sensor and save to variable temp

temp = temp \* 0.48828125;

// convert the analog volt to its temperature equivalent

Serial.print("TEMPERATURE = ");

Serial.print(temp); // display temperature value

Serial.print("\*C");

Serial.println();

delay(1000); // update sensor reading each one second

}

Temperature=temperature \* ((5/1024)\*100)

HC-SR04 Ultra sonic sensor:

send a short pulse (10us high level signal, it automatically sends eight 40 kHz pulses) to trigger pin and then wait for the echo pulse and measure how long it was high.

Test distance = (high level time × velocity of sound (340M/S) / 2,

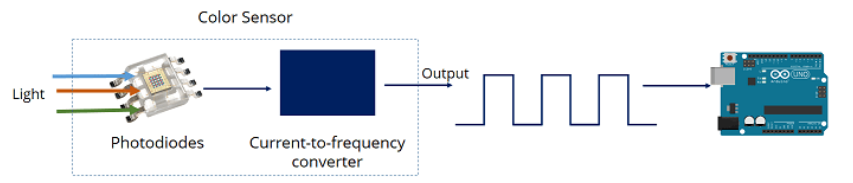
Divide the number of microseconds it was high by 58 and you have the distance in centimetres ( use 148 for inches).

Colour sensor (TCS230/TCS3200):

TCS3200 color sensor can detect a wide variety of colors based on their wavelength. This sensor is specially useful for color recognition. It also contains four white LEDs that light up the object in front of it. If you take a closer look at the TCS3200 chip you can see the different filters.







STM32 NUCELO F401RE

1. Based on ARM processor with 84MHz clock
2. 512 Kb flash (ROM) memory
3. RICS architecture
4. I-code,D-code and System buses helps in communication between cortex-M processor and the internal interfaces
5. The cpu(processors) acts as master and all other peripherals are the slave to it they communicate with each other using buses.
6. Data accusations blocks= ADC and DAC

Steps to work with stm:

Connect the board with mini usb cable

Download the st-link009 driver to setup the virtual com port (can be seen on the device manager)

: https://www.st.com/content/st\_com/en/products/development-tools/

Upgrade firmware

ARM uses its own native compiler

Install keil MDK by selecting to download MDK lite : <http://www2.keil.com/mdk5>

DFP(device firmware packages): used for internal device peripherals whereas BSP(board support package): used for external peripherals.

STM cube\_mx is a code generator, powerful tool to generate code by using user defined configurations

Writing code:

Open Keil MDK application -> projects -> new project ->give file name ->save->select device ->STM32F446RE.

A manage runtime environment pops up select: Device->startup check the box ->click resolve at bottom left corner -> click ok

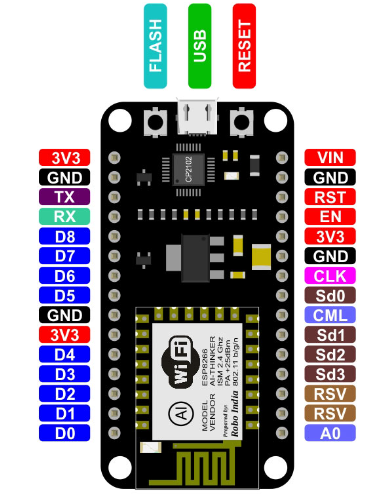
The target so created consists of source group where we store our c program files , CMSIS and Device which consists of essential system startup files. These files will be executed first before the c code.

NodeMCU (

NodeMCU is an open source IoT platform. It is a firmware which runs on ESP8266 Wi-Fi SoC (system on chip) from Espressif Systems. It has on board wi-fi available through which IoT applications becomes easy to build.

It has got Micro USB slot that can be directly connected to the computer.

It has got CP2102 USB to serial converter.



Setup steps:

1. Install CP2102 (USB to Serial Converter), driver for the same can be downloaded from: [https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers](https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers%20)

Table of Contents

**ESP82661**

Type chapter title (level 2)2

Type chapter title (level 3)3

**Blynk4**

Type chapter title (level 2)5

Type chapter title (level 3)6

**ZIGBEE7**

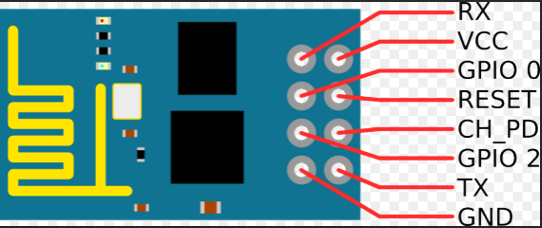
ESP8266

ESP8266 WiFi Module offers complete networking solutions to our DIY (Do-it-yourself) and IoT (Internet of Things) projects. It provides WiFi connectivity to any microcontroller through its full TCP/IP Stack.

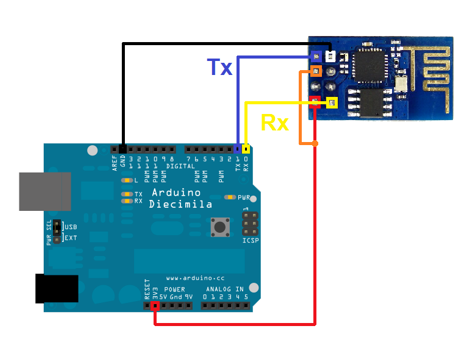
The ESP8266 WiFi module and the microcontroller can be interface through UART (Tx-Rx pins) and with the help of a wide range of AT Commands, the Microcontroller can control the ESP Module.

ESP8266 WiFi module is basically a microcontroller developed and manufactured by Espressif Systems which is a company based out of shanghai, China.

Pin out:



Connection with Arduino uno:



**ESP8266 can operate in three different modes: Wi-Fi station, Wi-Fi access point, and both at the same time.**

AT commands:

The AT Commands of the ESP8266 WiFi Module are responsible for controlling all the operations of the module like restart, connect to WiFi, change mode of operation and so forth.

In order to type in AT commands on ESP8266 (to check if its working fine) using Arduino IDE’s serial monitor :

1. Short CH-PD and Vcc pin and provide 3.3V to it.
2. Upload a blank sketch onto arduino by removing the Tx -Rx pins.
3. Next connect the Tx-Tx pin and Rx-Rx pins .
4. Select baud rate of serial monitor at 115200 and select NL&CR (newline & carriage return) first drop-box in serial monitor.
5. Type in AT commands.

ESP8266 AT Commands can be classified into four types:

* Test
* Query
* Set
* Execute

Basic useful AT commands:

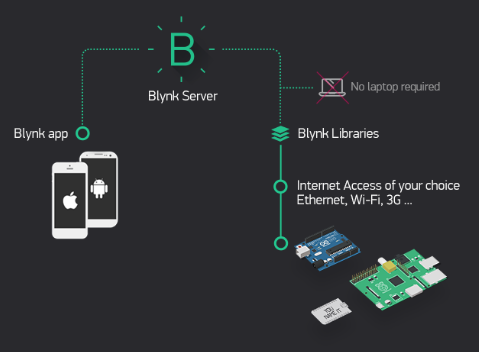
Blynk



Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, vizualize it and do many other cool things.

There are three major components in the platform:

* Blynk App - allows to create interfaces for your projects using various widgets provided.
* Blynk Server - responsible for all the communications between the smartphone and hardware. You can use the Blynk Cloud or run your [private Blynk server](http://docs.blynk.cc/#blynk-server) locally. It’s open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
* Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outcoming commands.



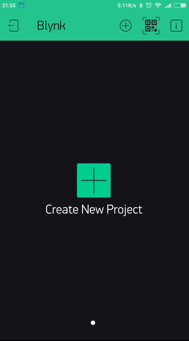
* Every time you press a Button in the Blynk app, the message travels to the Blynk Cloud, where it finds its way to your hardware (using ESP8266 or any other wifi device) and vice versa(opposite direction communication.

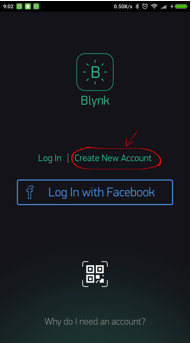
Connection to the cloud can be done using:

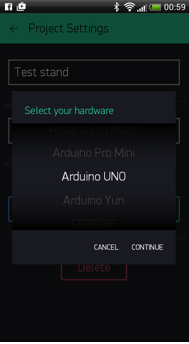
1. WiFi
2. Bluetooth and BLE
3. Ethernet
4. USB (Serial)
5. GSM

Blynk works over the Internet. This means that the hardware you choose should be able to connect to the internet. Some of the boards, like Arduino Uno will need an Ethernet or Wi-Fi Shield to communicate, others are already Internet-enabled: like the ESP8266, Raspberri Pi with WiFi dongle, Particle Photon or SparkFun Blynk Board. But even if you don’t have a shield, you can connect it over USB to your laptop or desktop.

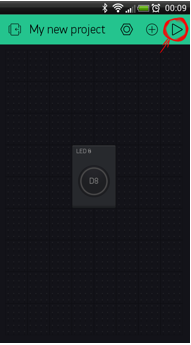
Steps for setting up Blynk wiget on app:

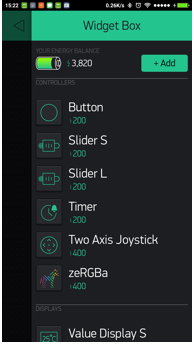


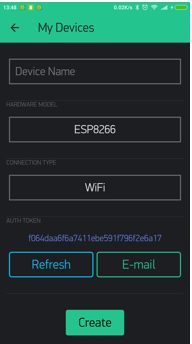


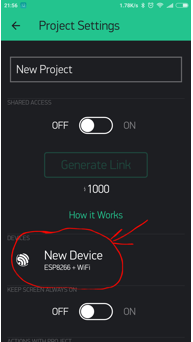


1. Create a Blynk Account:After you download the Blynk App, you’ll need to create a New Blynk account.
2. Create a New Project
3. Choose Your Hardware









Auth Token is a unique identifier which is needed to connect your hardware to your smartphone. Every new project you create will have its own Auth Token. You’ll get Auth Token automatically on your email after project creation. You can also copy it manually. Click on devices section and selected required device

Add a Widget: Each Widget has it’s own settings. Tap on the widget to get to them.

Run The Project: When you are done with the Settings - press the PLAY button. This will switch you from EDIT mode to PLAY mode where you can interact with the hardware. While in PLAY mode, you won’t be able to drag or set up new widgets, press STOP and get back to EDIT mode.

Important Note:

To send notification that consists of string and integer use:

int count=5;

Blynk.notify("count= " + String(count) + "\r\n");

We need to typecast int count as string to be transmitted successfully.

Here the notification received on the app is count=5

Zigbee

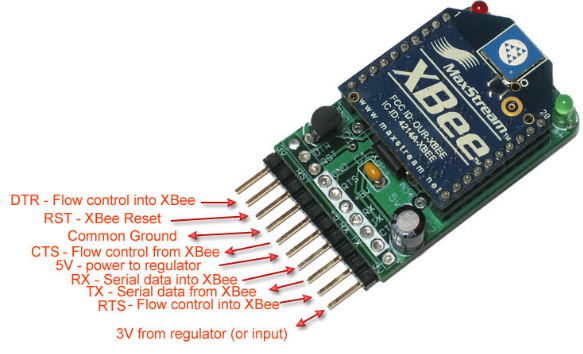


Zigbee is a mesh communication protocols that sits on top of IEEE 802.15.4 PHY(physical layer). It has lower power consumption and lower data transfer rate (250kbit/s).

**ZigBee** is standard protocol for wireless networking. While **XBee** is a product that supports various wireless communication protocol, including ZigBee, Wi-Fi (Wi-Fly module), 802.15.4, 868 MHz module etc.

Xbee is a small transceiver(Tx-Rx) sends serial data fed to it by arduino. Xbee module can act as a coordinator/router/end device. Required to be configured in desired mode. Supports point to point and point to multipoint n/w(unicast/broadcast)

Any receiving radio (receiver) has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.



Zigbee forms a PAN(personal area network).

A zigbee module can be configured as a coordinator(first established, setups a new PAN) , a router or as end device by using **XCTU** software for windows.

While configuring the xbee using XCTU connect TX(uno) to Tx(Xbee) and Rx(uno) to Rx(Xbee) poer and groud and RST to GND.

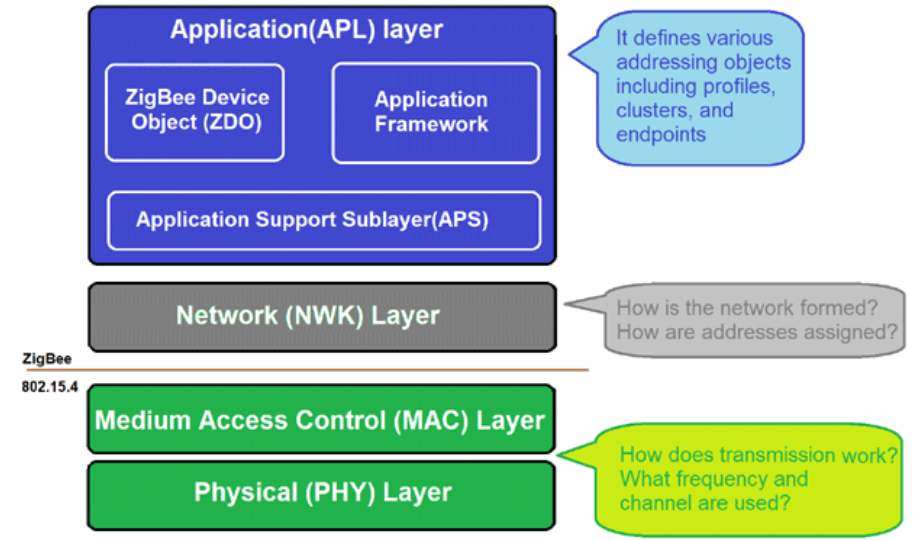
PAN is setup by coordinator, once setup router and end devices can join the network.

|  |  |  |
| --- | --- | --- |
| Coordinator ( C ) | Router ( R ) | End Device ( E ) |
| First in the network and it sets up the PAN | Second to join | Last to join |
| Assists data routing | Assists data routing | Does not assists data routing as it Is the last element in the n/w |
| Allows join requests from R and E | Allows other R and E to join | Does not allow other devices to join the PAN |
| Powered by AC mains suppy | Powered by AC mains supply | Battery powered |
| Never goes to sleep | Never goes to sleep | Sleeps to save battery power |
| Selects channel and PAN ID. Has PAN ID=0 | Receives PAN ID from C | Gets PAN ID from C |

PAN ID used to communicate between zigbee devices, it is 16 bit number and is assigned to other devices by coordinator when they join the n/w.

**ZigBee Architecture:**

There are **major four layers available** in ZigBee stack which are physical layer, Media access layer, Network layer and application layer.



1. **Network Layer** adds routing capabilities that allows RF data packets to traverse multiple devices (multiple "hops") to route data from source to destination (peer to peer). (director/route signs)
2. **MAC layer** manages RF data transactions between neighboring devices (point to point). The MAC includes services such as transmission retry and acknowledgment management and collision avoidance techniques.(information center/ ticket checker)
3. **Physical layer:** It defines how devices are connected to make a network; it defines the output power, number of channels and transmission rate. (architect)

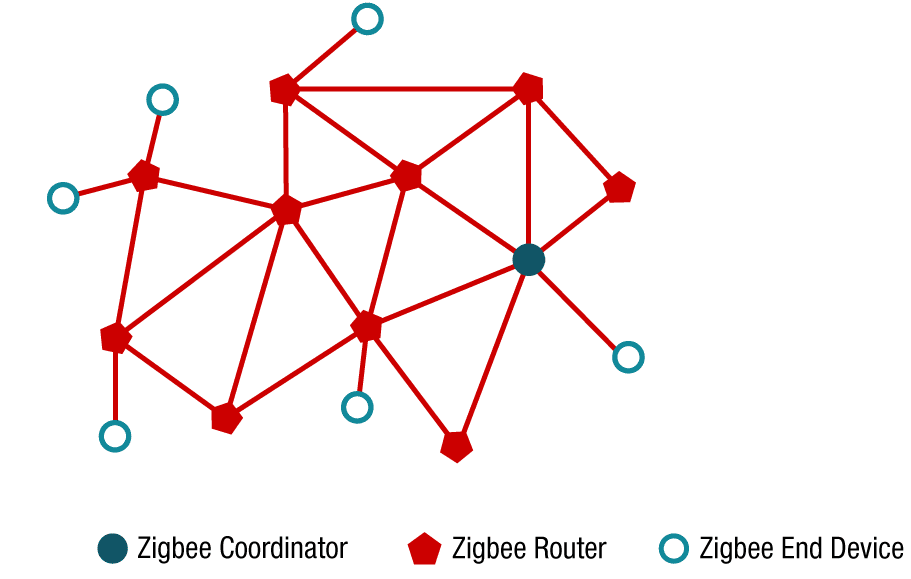
Network can be thought as a combination of software and hardware which is capable of sending data from one location to another. Hardware is responsible for carrying the signals from one point of network to another. Software consists of instruction sets that make it possible to work as we expect.

Zigbee uses mesh routing.Routing protocol used by zigbee- AODV(adhoc on demand distance vector) protocol.

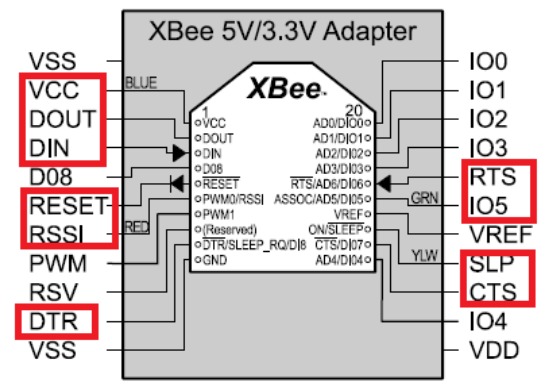
Zigbee transmission in 2 ways: Unicast or broadcast

**Broadcast:**

Data is propagated in the whole network such that all nodes receive the transmission. To achieve this the coordinator and all routers that receive a broadcast transmission will retransmit the packet three times.

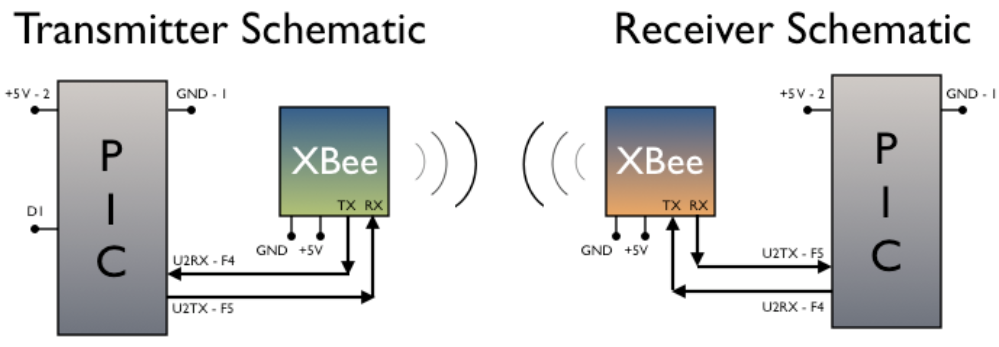


Unicast: Route data from a source device to destination device.



There are two types of addresses Static address (64-bit address) : unique device number and Dynamic address (16-bit address) : unique local address when a device joins a zigbee network.

Zigbee is connected with the microcontroller using UART (Tx-Rx) pins.



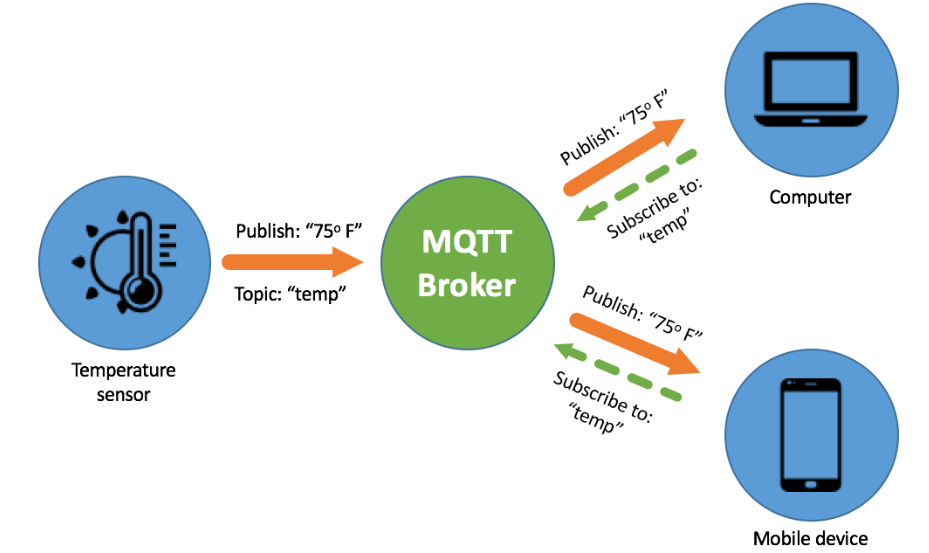
XBee AT Commands:

1. **AT (TEST):** to check the module is responding OK as reply confirms the same.
2. **ATDH:** Destination Address High. To configure the upper 32 bits of the 64-bit destination address DL and DH combined gives you 64 bit destination address.
3. **ATDL:** Destination Address Low. This again for configuring the lower 32 bits of the 64-bit destination address.
4. **ATID:** This command changes the PAN ID
5. **ATWR:** Write. Write parameter values to non-volatile memory so that parameter modifications persist through subsequent resets. Note: Once WR is issued, no additional characters should be sent to the module until After the "OK\r" response is received.
6. **ATRE (Restore Defaults):** Restores factory settings to the module, is very useful if the module does not responds

MQTT protocol



* MQTT stands for MQ Telemetry Transport.
* MQTT is widely used in IoT (Internet of Things) embedded applications, where every sensor is connected to a server and we have access to control them over the internet.
* *MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It* is lightweight publish-subscribe based messaging protocol. It is developed on the base of TCP/IP protocol.
* It allows remote location devices to connect, subscribe, publish etc. to a specific topic on the server with the help of message broker.
* MQTT Broker/Message broker is a module in between the sender and the receiver. It is an element for message validation, transformation and routing.
* The broker is responsible for distributing messages to the interested clients (subscribed clients) of their interested topic.

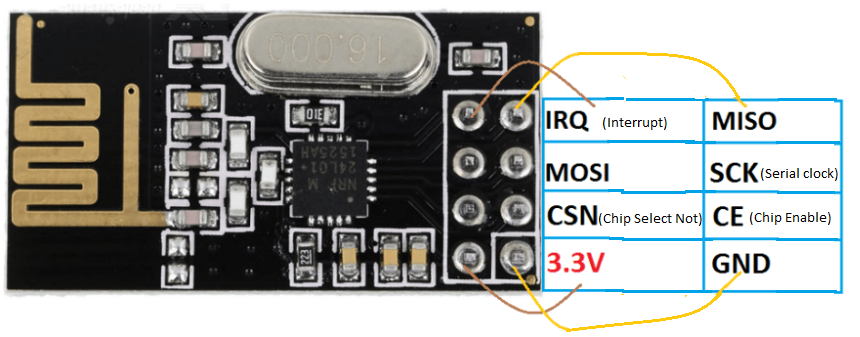


TCP/IP port 1883 is reserved for MQTT.

Additional security can be added by an application encrypting data which it sends and receives, but it is not built-in to the protocol, in order to keep it simple and lightweight.

**NRF24L01**

1. NRF24L01 is basically a wireless transceiver, which is used to send and receive data by using radio waves.
2. It is a single chip transceiver module.
3. It uses SPI protocol for transmitting data. Its data transmission speed is up to 2Mbps.
4. It sends and receives data at an operating radio frequency of 2.4 to 2.5 GHz ISM band.
5. Its range can reach to 100meters if used in open space using antenna.
6. The module consists of a frequency generator, shock burst mode controller, power amplifier, crystal oscillator modulator, and demodulator.
7. When transmitting power is zero dBm it uses only 11.3 mA of current(even lesser than LED), while during receiving mode, it uses 13.5 mA of current.
8. This module is designed for long distance and fast transmission of data.
9. Its high air data rate combined with power saving mode makes it very favorable for ultra-low power applications.
10. Its power supply range is 1.9 to 3.6 V. It is designed to operate at 3.3 volts. But other pins are 5v tolerant (i.e can be directly connected to Arduino).
11. It has an address range of 125 and it can communicate with six other modules.
12. Its nominal current is 50 mA. Its operating current is 250 mA.
13. Its Maximum Pipeline or node is six.
14. This module is a by default half- duplex fabricated module and it has the capability to send and receive data simultaneously.
15. By using this NRF24L01 module we can send a message to a particular receiver.
16. We can receive a message from some particular sender.
17. During sending the message through this module, we will have to specify the message sender’s and receiver’s address.
18. Also we will have to specify the size of that particular message, which we are going to transmit through this module.
19. When received a particular message then, we will stop the communication first and will read it, and then send it. So in such situations, we have to perform the switching while sending or receiving data through this module.
20. NRF24l01 has total 8 pins.



The CE and CSN pins are used to selected the mode of operation of NRF24L01: active, command or transmit mode

Important Concepts

**Bit Rate and Baud Rate:**

An analogy to illustrate the concept of bauds and bits. In transportation, a **baud is** comparable to a **bus**, a **bit** analogous to a **passenger**. A bus can carry multiple passengers. If 1000 buses go from one point to another carrying only one passenger (the driver), then 1000 passengers are transported. However, if each bus carries twenty passengers (suppose), then 20000 passengers are transported. In this case, busses determine traffic not the number of passengers consequently broader highways are needed. Likewise, the number of bauds determines the required bandwidth, not the number of bits.

One byte is equal to 8 bits, but serial connections send a start and stop bit to identify the start and end to a particular byte to the receiving system. Thus, 10 bits are needed to send one character.

Making the number smaller (e.g. Serial.begin(300)) has the Arduino send data more slowly. Increasing it, say to 57600 will send data faster. Both the sending system and the receiving system need to agree on what speed to use.

**Bootloader:**

The bootloader is basically a **.hex** file that runs when you turn on the board.

It does two things.

First, it looks around to see if the computer is trying to program it. If it is, it grabs the program from the computer and uploads it into the ICs memory (in a specific location so as not to overwrite the bootloader). That is why when you try to upload code, the Arduino IDE resets the chip. This basically turns the IC off and back on again so the bootloader can start running again.

Second, If the computer isn't trying to upload code, it tells the chip to run the code that's already stored in memory. Once it locates and runs your program, the Arduino continuously loops through the program and does so as long as the board has power.

The default bootloader of the Arduino UNO is the **optiboot bootloader**.

**Embedded Operating Systems:**

Common O.S for resource constrained devices (embedded devices) are: FreeRTOS (for ARM, PIC, AVR, MSP430) , Mbed OS (by ARM for ARM processor), Mynewt OS (for ARM cortex M, MIPS and RISCv processor), Zephyr, RIOT, Contiki OS, Tiny OS.