HARDWARE REQUIRED

1. Arduino Nano – 1 pcs
2. Arduino Nano USB Cable – 1 pcs
3. Ultrasonic Sensor Module HC-SR04 - 2 pcs
4. Rain sensor Module – 1 pcs
5. Sim 800l GSM Module – 1 pcs
6. Neo 6m GPS Module - 1 Pcs
7. 9v Battery - 3 pcs
8. Battery snapper - 3 pcs
9. B20 Buzzer excel - 1 pcs
10. Jumper wire (Female to Female) - As per requirement
11. Mini Rocker switch – 1 pcs
12. Wire – As per requirement
13. Pipe or Stick
14. Arduino IDE Software

HARDWARE DESCRIPTION

1.Arduino Nano

Synopsis Description

Arduino is an open source microcontroller board. The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328P (Arduino Nano 3.x).The microcontroller on the board is programmed using Arduino software.

The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards or breadboards and other circuit. The Microcontrollers are typically programmed using a dialect of features from programming language C & C++. In addition to using traditional compiler tool chains, the Arduino project provides an integrated development environment (IDE) bases on the processing language project.

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 kOhms. 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: 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. See the attachInterrupt() function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- LED: 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 is it 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 (documentation on the Wiring website).

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.

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide 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 SoftwareSerial 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. To use the SPI communication, please see ATmega328 datasheet.

Programming

The Arduino Nano can be programmed with the Arduino software (download). Select "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board).
The ATmega328 on the Arduino Nano comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol.
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

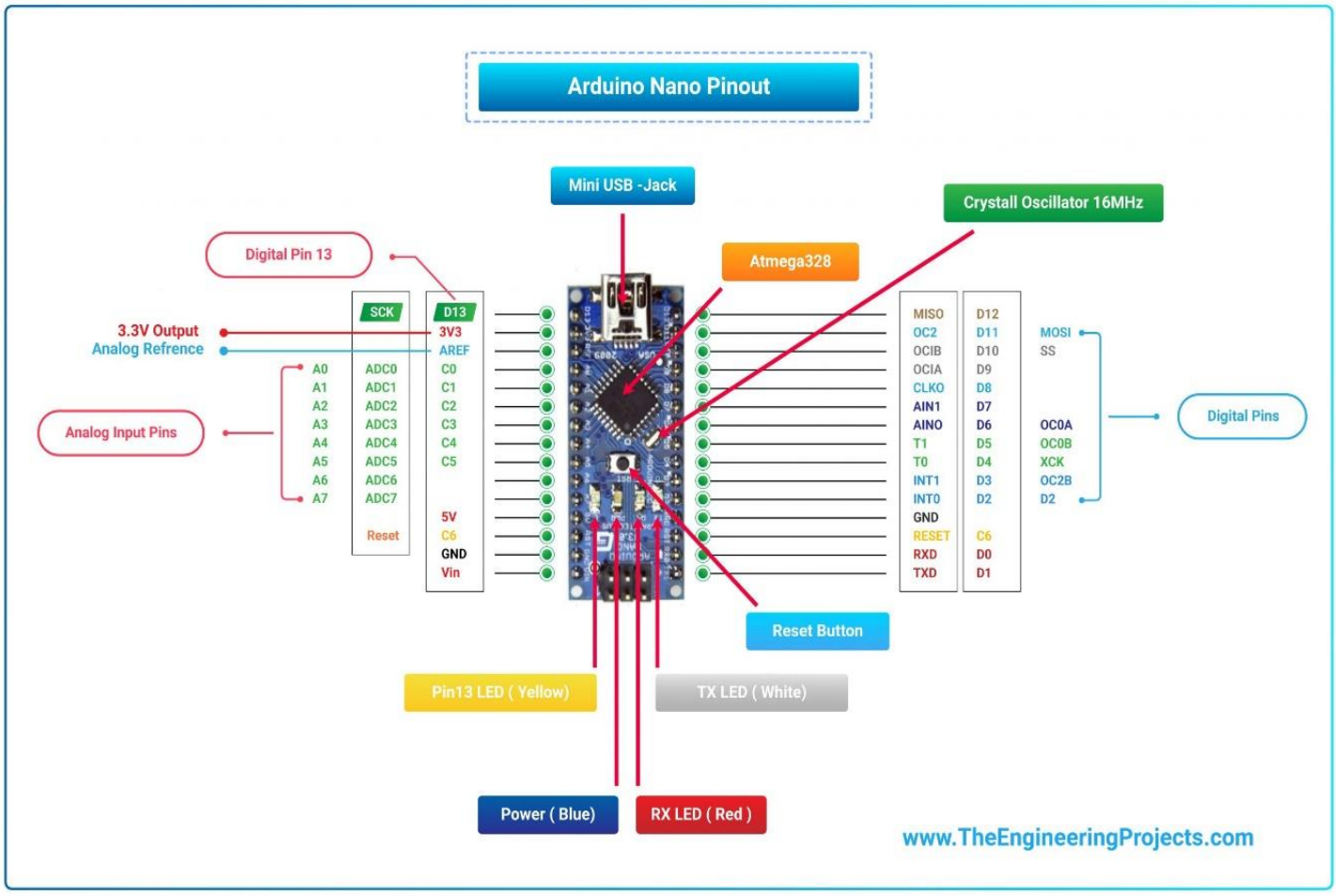
Automatic (Software) Reset

> Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of

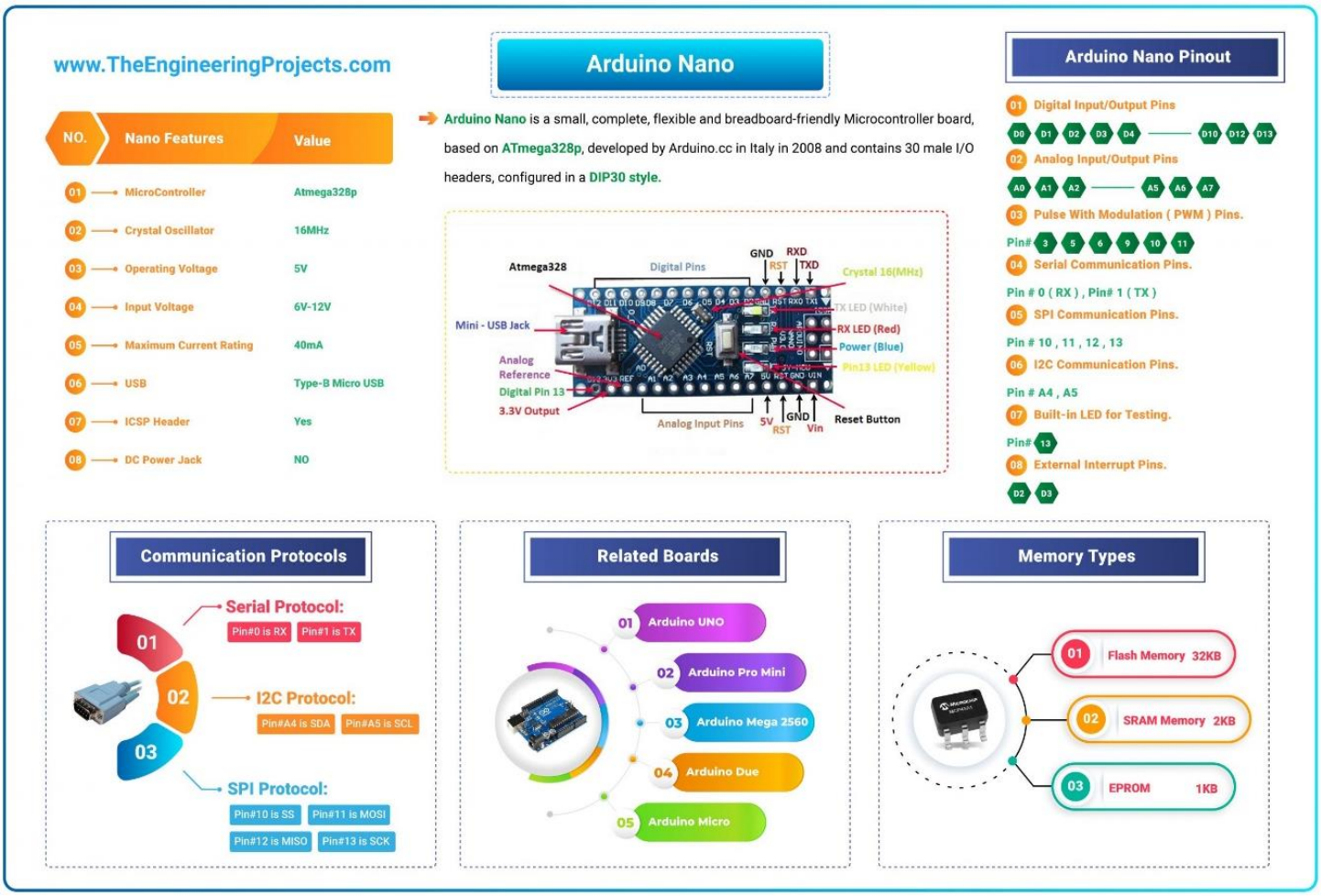
the FT232RL is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Nano is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Nano. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

Brief Description :- Arduino nano in very detailed explanation with Graphics

Arduino Nano Pinout, datasheet, drivers & applications. It is a Microcontroller board developed by arduino.cc and based on Atmega328p / Atmega168. Arduino boards are widely used in robotics, embedded systems, automation, Internet of Things (IoT) and electronics projects. These boards were initially introduced for the students and non-technical users but nowadays Arduino boards are widely used in industrial projects.



Here’s the figure showing key points of Arduino Nano:



- Here’s the table showing important features of Arduino Nano:

1	Microcontroller	Atmega328p
2	Crystal Oscillator	16MHz
3	Operating Voltage	5V
4	Input Voltage	6V-12V
5	Maximum Current Rating	40mA
6	USB	Type-B Micro USB
7	ICSP Header	Yes
8	DC Power Jack	No

Here’s the quick overview of Arduino Nano Pinout:

1	D0 – D13	Digital Input / Output Pins.
2	A0 – A7	Analog Input / Output Pins.
3	Pin # 3, 5, 6, 9, 10, 11	Pulse Width Modulation (PWM) Pins.
4	Pin # 0 (RX) , Pin # 1 (TX)	Serial Communication Pins.
5	Pin # 10, 11, 12, 13	SPI Communication Pins.
6	Pin # A4, A5	I2C Communication Pins.
7	Pin # 13	Built-In LED for Testing.
8	D2 & D3	External Interrupt Pins.

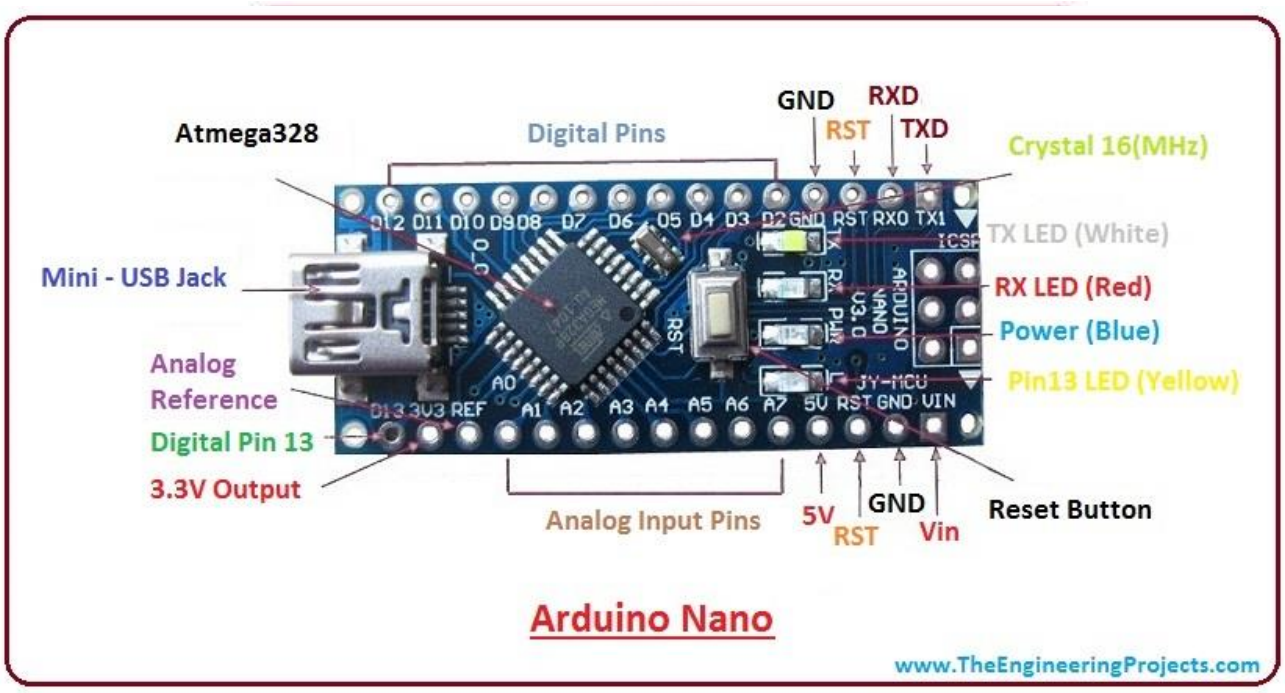
- Arduino Nano offers three types of communications protocols, shown in the below table:

6	Serial Port	1 (Pin#0 is RX, Pin#1 is TX).
7	I2C Port	1 (Pin#A4 is SDA, Pin#A5 is SCL).
8	SPI Port	1 (Pin#10 is SS, Pin#11 is MOSI, Pin#12 is MISO, Pin#13 is SCK).

- Here's the memory details present in Arduino Nano:

7	Flash Memory	32KB
8	SRAM Memory	2KB
7	EEPROM	1KB

Introduction to Arduino Nano



- **Arduino Nano** is a small, complete, flexible and breadboard-friendly Microcontroller board, based on **ATmega328p**, developed by Arduino.cc in Italy in 2008 and contains 30 male I/O headers, configured in a **DIP30 style**.
- **Arduino Nano Pinout** contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.
- It is programmed using **Arduino IDE**, which can be downloaded from Arduino Official site.
- Arduino Nano is simply a smaller version of Arduino UNO, thus both have almost the same functionalities.
- It comes with an **operating voltage of 5V**, however, the input voltage can vary from **7 to 12V**.
- Arduino Nano's **maximum current rating is 40mA**, so the load attached to its pins shouldn't draw current more than that.
- Each of these Digital & Analog Pins is assigned with multiple functions but their main function is to be configured as **Input/Output**.

- Arduino Pins are acted as **Input Pins** when they are interfaced with sensors, but if you are driving some load then we need to use them as an **Output Pin**.
- Functions like **pinMode()** and **digitalWrite()** are used to control the operations of digital pins while **analogRead()** is used to control analog pins.
- The analog pins come with a total **resolution of 10-bits** which measures the value from 0 to 5V.
- Arduino Nano comes with a **crystal oscillator of frequency 16 MHz**. It is used to produce a clock of precise frequency using constant voltage.
- There is one limitation of using Arduino Nano i.e. it doesn't come with a **DC power jack**, which means you can not supply an external power source through a battery.
- This board doesn't use standard USB for connection with a computer, instead, it comes with **Type-B Micro USB**.
- The tiny size and breadboard-friendly nature make this device an ideal choice for most applications where the size of the electronic components is of great concern.
- **Flash memory is 16KB or 32KB** that all depends on the Atmega board i.e Atmega168 comes with 16KB of flash memory while Atmega328 comes with a flash memory of 32KB. Flash memory is used for storing code. The 2KB of memory out of total flash memory is used for a bootloader.
- The **SRAM memory of 2KB** is present in Arduino Nano.
- Arduino Nano has an **EEPROM memory of 1KB**.

- The following figure shows the specifications of the Arduino Nano board.

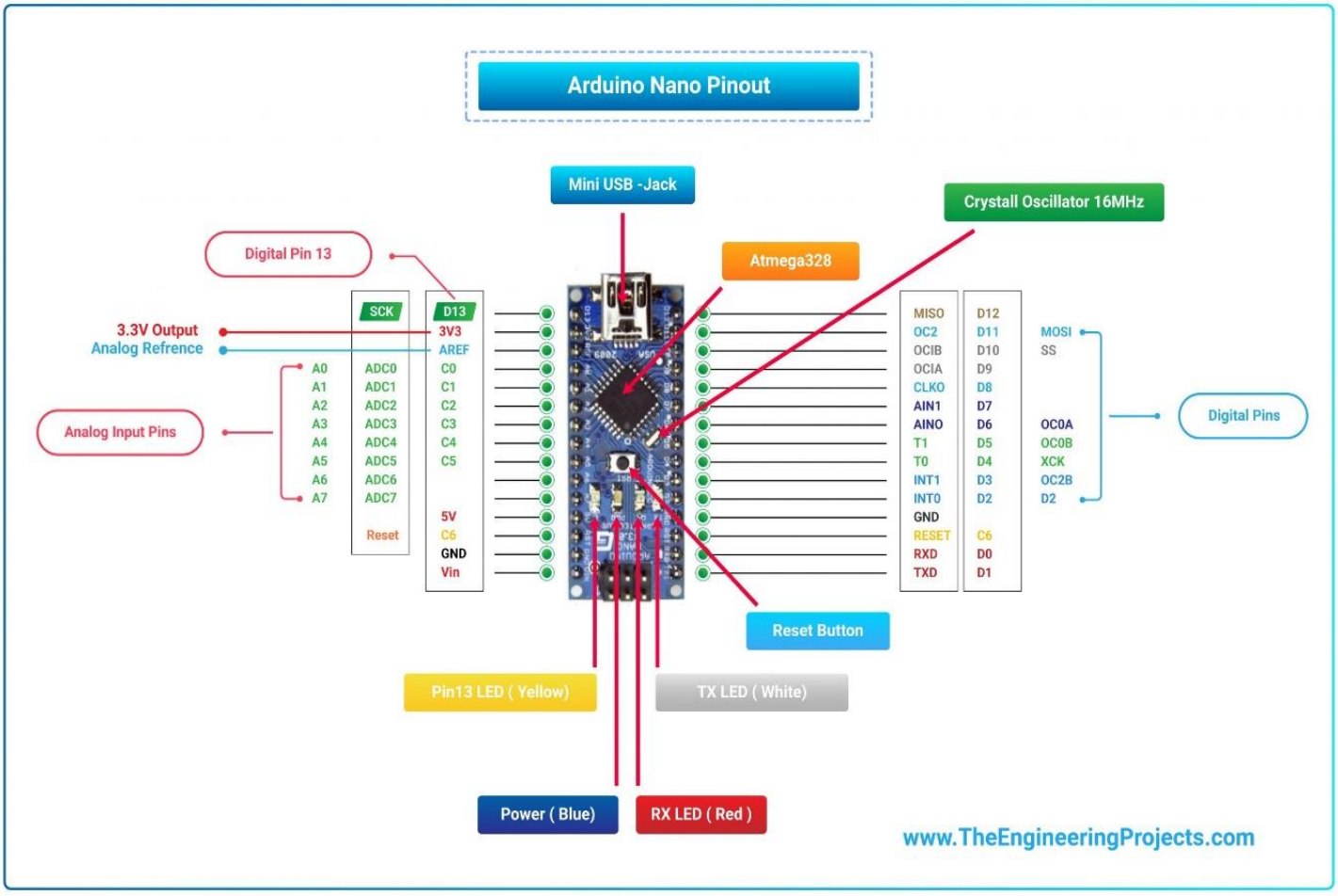
Arduino Nano Spacification

The following figure shows the specifications of the Arduino Nano board.

Microcontroller	Atmega328/Atmega168
Operating Voltage	5V
Input Voltage	7 - 12 V
Digital I/O Pins	14
PMW	6 Out of 14 Digital Pins
Max. Current Rating	40mA
USB	Mini
Analog Pins	8
Flash Memory	16KB or 32KB
SRAM	1KB or 2KB
Crystal Oscillator	16 MHz
EEPROM	512byte or 1KB
USART	Yes

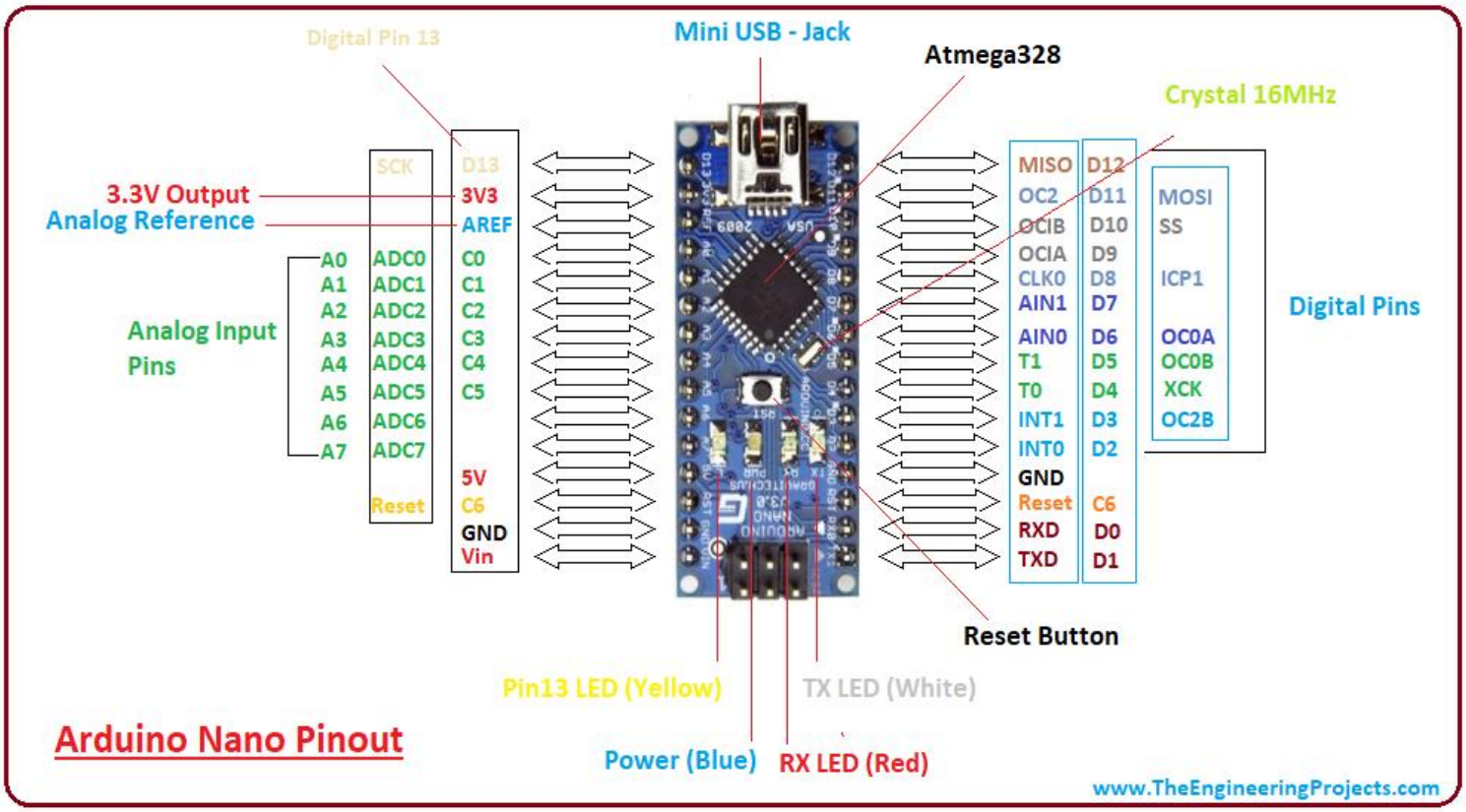
- It is programmed using Arduino IDE which is an Integrated Development Environment that runs both offline and online.
- No prior arrangements are required to run the board. All you need is a board, mini USB cable and Arduino IDE software installed on the computer.
- USB cable is used to transfer the program from the computer to the board.
- No separate burner is required to compile and burn the program as this board comes with a built-in boot-loader.

Now, let's have a look at Arduino Nano Pinout in detail:



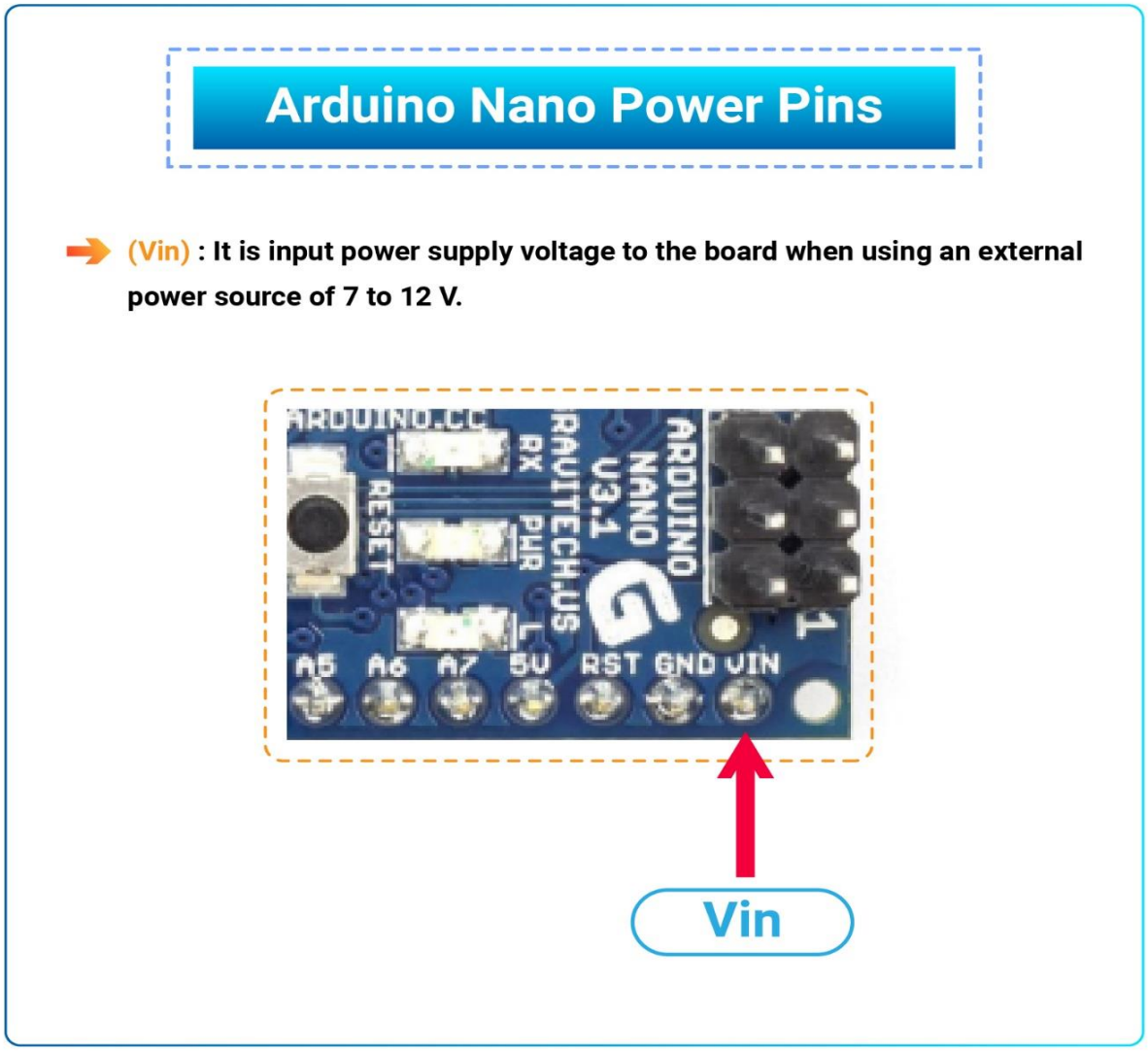
Arduino Nano Pinout

- The following figure shows the pinout of the Arduino Nano Board:



- Each pin on the Nano board comes with a specific function associated with it.
- We can see the analog pins that can be used as an analog to a digital converter, where A4 and A5 pins can also be used for I2C communication.
- Similarly, there are 14 digital pins, out of which 6 pins are used for generating PWM.

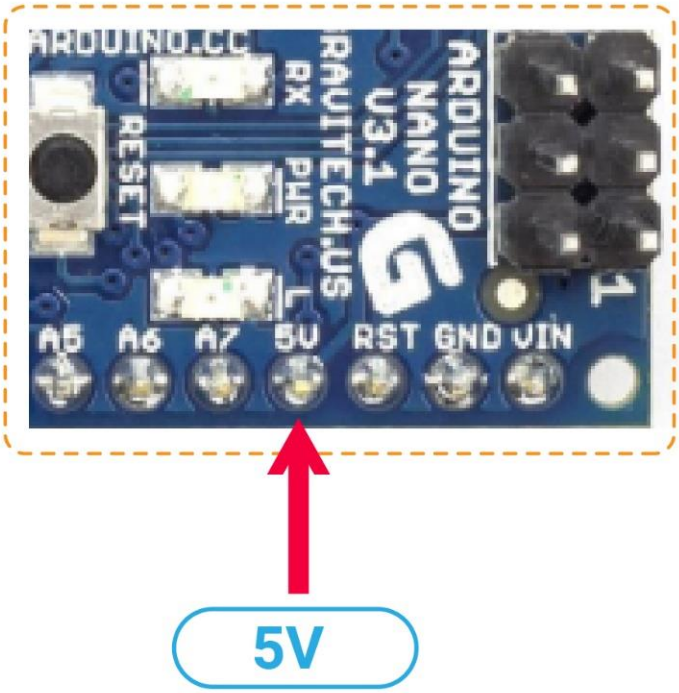
Let’s have a look at the Arduino Nano Pinout in detail:



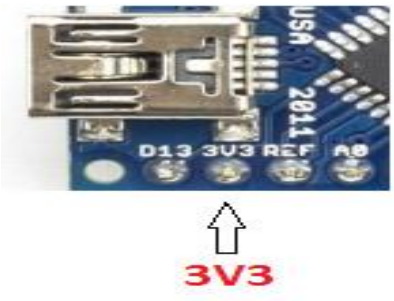
- **Vin:** It is input power supply voltage to the board when using an external power source of 7 to 12 V.
- **5V:** It is a regulated power supply voltage of the board that is used to power the controller and other components placed on the board.

Arduino Nano Power Pins

5V: It is a regulated power supply voltage of the board that is used to power the controller and other components placed on the board..

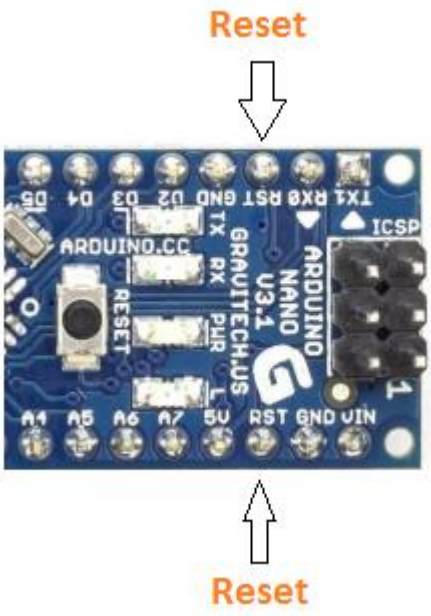


- **3V3:** This is a minimum voltage generated by the voltage regulator on the nano board.



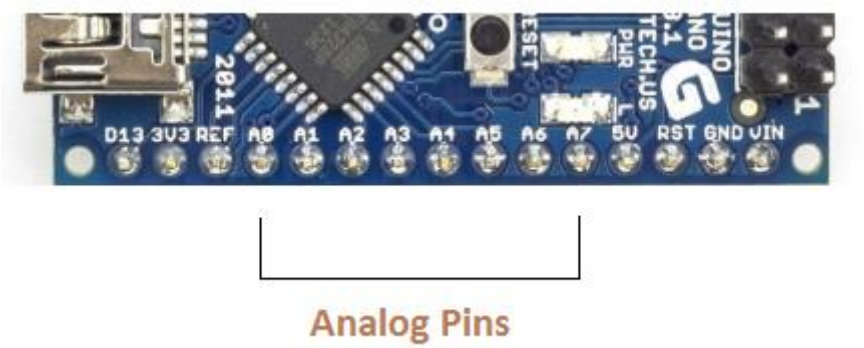
- **GND Pin:** These are the ground pins on the board.
- There are multiple ground pins on the board that can be interfaced accordingly when more than one ground pin is required.

- **Reset Pin:** Arduino Nano has 2 reset pins incorporated on the board, making any of these **Reset pins LOW** will reset the microcontroller.



- **Pin#13:** A built-in LED is connected to pin#13 of nano board.
- This LED is used to check the board i.e. it's working fine or not.
- **AREF:** This pin is used as a reference voltage for the input voltage.

- **Analog Pins:** There are 8 analog pins on the board marked as **A0 – A7**.



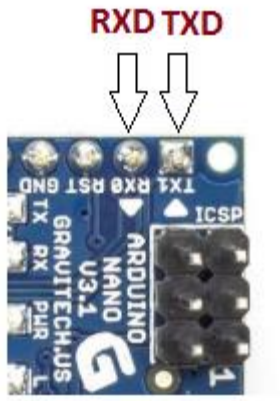
- These pins are used to measure the analog voltage ranging between **0 to 5V**.

- **Digital Pins**: Arduino Nano has 14 digital pins starting from D0 to D13.
- These digital pins are used for interfacing third-party digital sensors and modules with Nano board.
- **PWM Pins:** Arduino Nano has 6 PWM pins, which are Pin#3, 5, 6, 9, 10 and 11. (All are digital pins)
- These pins are used to generate an 8-bit PWM (Pulse Width Modulation) signal.
- **External Interrupts:** Pin#2 and 3 are used for generating external interrupts normally used in case of emergency, when we need to stop the main program and call important instructions.
- The main program resumes once interrupt instruction is called and executed.

- **Serial Pins:** These pins are used for serial communication where:

    1. Pin#0 is RX used for receiving serial data.

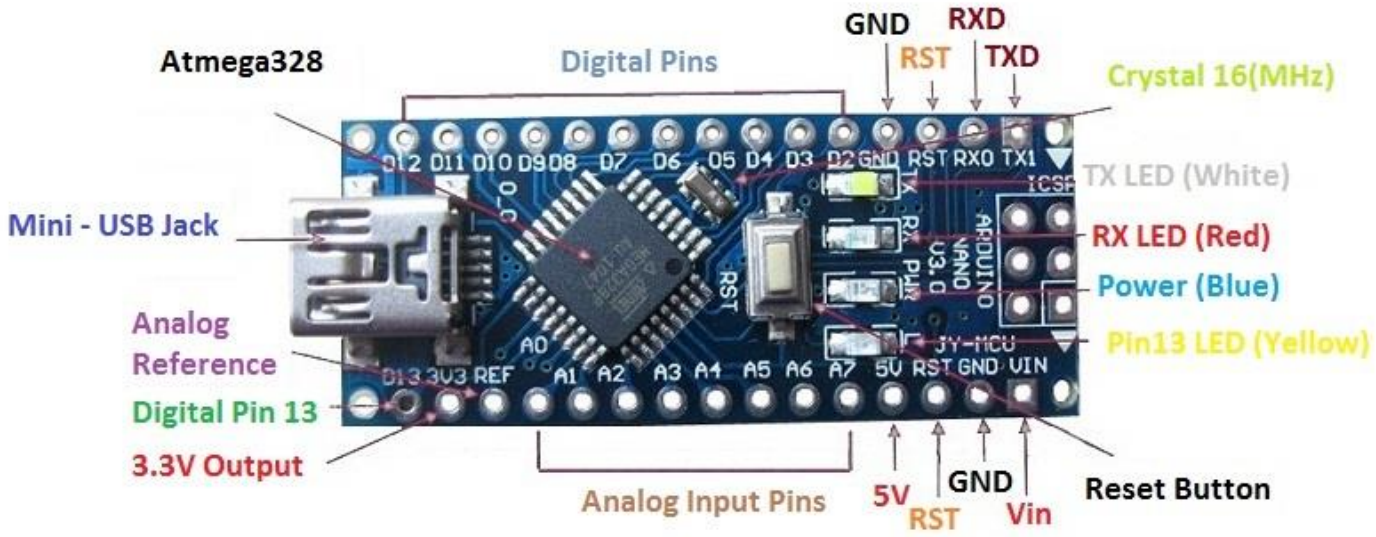
    2. Pin#1 is Tx used for transmitting serial data.



- **SPI Protocol:** Four pins 10(SS->Slave Select), 11(MOSI -> Master Out Slave In), 12(MISO -> Master In Slave Out) and 13(SCK -> Serial Clock) are used for SPI (Serial Peripheral Interface) Protocol.

- SPI is an interface bus and is mainly used to transfer data between microcontrollers and other peripherals like sensors, registers, and SD cards.

- **I2C Protocol:** I2C communication is developed using A4 and A5 pins, where **A4 represents the serial data line (SDA)** which carries the data and **A5 represents the serial clock line (SCL)** which is a clock signal, generated by the master device, used for data synchronization between the devices on an I2C bus.



Arduino Nano

Arduino Nano Programming & Communication

- The Nano board comes with the ability to set up communication with other controllers and computers.
- The serial communication is carried out by the digital pins, Pin 0(Rx) and Pin 1(Tx) where Rx is used for receiving data and Tx is used for the transmission of data.
- The serial monitor is added to the Arduino IDE, which is used to transmit textual data to or from the board.
- FTDI drivers are also included in the software which behaves as a virtual com port to the software.
- The Tx and Rx pins come with an LED which blinks as the data is transmitted between FTDI and USB connection to the computer.
- Arduino Software Serial Library is used for carrying out serial communication between the board and the computer.
- Apart from serial communication the Nano board also supports I2C and SPI communication. The Wire Library inside the Arduino Software is accessed to use the I2C bus.
- The Arduino Nano is programmed by Arduino Software called IDE which is a common software used for almost all types of board available. Simply download the software and select the board you are using.
- Uploading code to Arduino Nano is quite simple, as there's no need to use any external burner to compile and burn the program into the controller and you can also upload code by using ICSP (In-circuit serial programming header).
- Arduino board software is equally compatible with Windows, Linux or MAC, however, Windows are preferred to use.

Applications of Arduino Nano

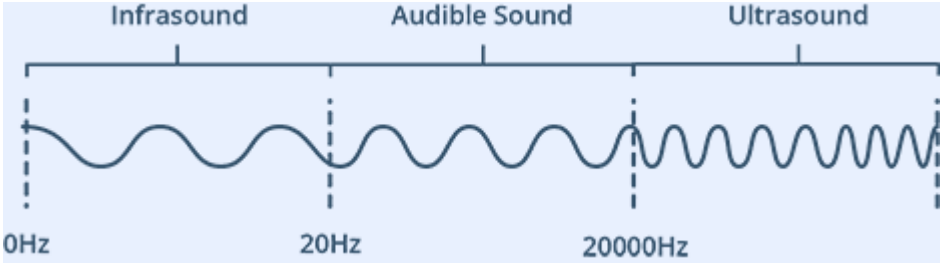
Arduino Nano is a very useful device that comes with a wide range of applications and covers less space as compared to other Arduino boards. Breadboard-friendly nature makes it stand out from other boards. Following are the main applications of Arduino Nano:

- Engineering Students' Projects.
- Medical Instruments
- Industrial Automation
- Android Applications
- GSM Based Projects
- Embedded Systems
- Automation and Robotics
- Home Automation and Defense Systems
- Virtual Reality Applications

HC-SR04 Ultrasonic Sensor Hardware Overview

What is Ultrasound?

Ultrasound is high-pitched sound waves with frequencies higher than the audible limit of human hearing.



Human ears can hear sound waves that vibrate in the range from about 20 times a second (a deep rumbling noise) to about 20,000 times a second (a high-pitched whistling). However, ultrasound has a frequency of over 20,000 Hz and is therefore inaudible to humans.

An ultrasonic sensor is a device that can measure the distance to an object by using sound waves. It measure distance by sending out a sound wave at a specific frequency and listening for that sound wave to bounce back. The ultrasonic transmitter an ultrasonic wave this wave travel in air and when it gets object by any material it gets reflected back toward the sensor this reflected wave is observed by the ultrasonic receiver module.

At its core, the HC-SR04 Ultrasonic distance sensor consists of two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses. The receiver listens for the transmitted pulses. If it receives them, it produces an output pulse whose width can be used to determine the distance the pulse travelled. As simple as pie!

The sensor is small, easy to use in any robotics project and offers excellent non-contact range detection between 2 cm to 400 cm (that's about an inch to 13 feet) with an accuracy of 3mm. Since it operates on 5 volts, it can be hooked directly to an Arduino or any other 5V logic microcontrollers.

Here are complete specifications:

Operating Voltage	DC 5V
Operating Current	15mA
Operating Frequency	40KHz
Max Range	4m
Min Range	2cm
Ranging Accuracy	3mm
Measuring Angle	15 degrees
Trigger Input Signal	10µS TTL pulse
Dimension	45 x 20 x 15mm

HC-SR04 Ultrasonic Sensor Pinout

HC-SR04 Pinout

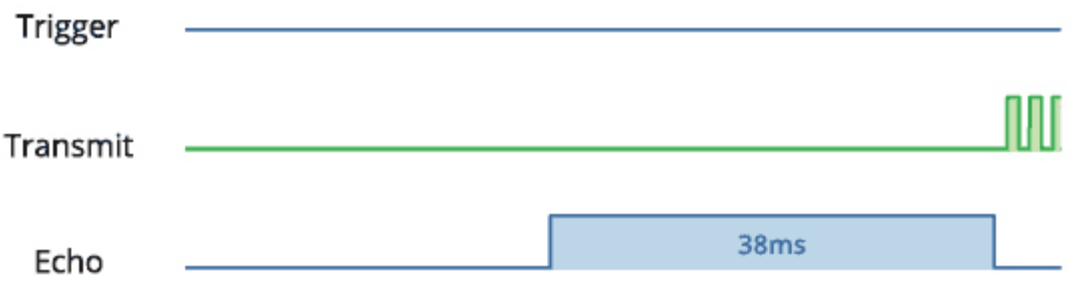
VCC is the power supply for HC-SR04 Ultrasonic distance sensor which we connect the 5V pin on the Arduino.
Trig (Trigger) pin is used to trigger the ultrasonic sound pulses.
Echo pin produces a pulse when the reflected signal is received. The length of the pulse is proportional to the time it took for the transmitted signal to be detected.
GND should be connected to the ground of Arduino.

How Does HC-SR04 Ultrasonic Distance Sensor Work?

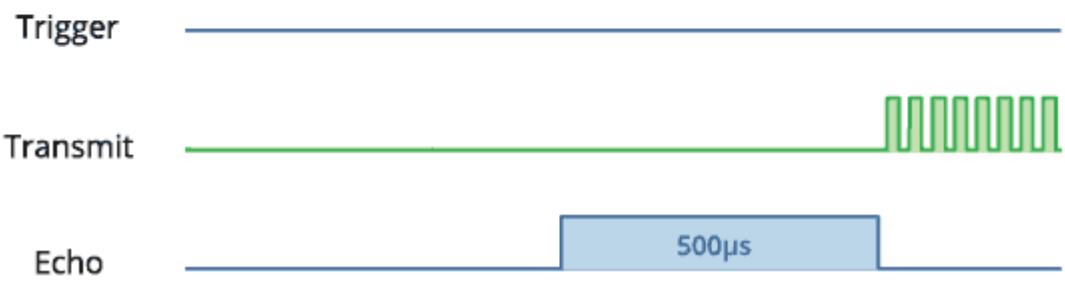
It all starts, when a pulse of at least 10 µS (10 microseconds) in duration is applied to the Trigger pin. In response to that the sensor transmits a sonic burst of eight pulses at 40 KHz. This 8-pulse pattern makes the "ultrasonic signature" from the device unique, allowing the receiver to differentiate the transmitted pattern from the ambient ultrasonic noise.

The eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the Echo pin goes HIGH to start forming the beginning of the echo-back signal.

In case, If those pulses are not reflected back then the Echo signal will timeout after 38 mS (38 milliseconds) and return low. Thus a 38 mS pulse indicates no obstruction within the range of the sensor.



If those pulses are reflected back the Echo pin goes low as soon as the signal is received. This produces a pulse whose width varies between 150 µS to 25 mS, depending upon the time it took for the signal to be received.



The width of the received pulse is then used to calculate the distance to the reflected object. This can be worked out using simple distance-speed-time equation, we learned in High school. In case you forgot, an easy way to remember the distance, speed and time equations is to put the letters into a triangle.

Distance = Speed x Time

$$\text{Time} = \frac{\text{Distance}}{\text{Speed}}$$

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}}$$

Let's take an example to make it more clear. Suppose we have an object in front of the sensor at an unknown distance and we received a pulse of width 500 µS on the Echo pin. Now let's calculate how far the object from the sensor is. We will use the below equation.

$$\text{Distance} = \text{Speed} \times \text{Time}$$

Here, we have the value of Time i.e. 500 µs and we know the speed. What speed do we have? The speed of sound, of course! Its 340 m/s. We have to convert the speed of sound into cm/µs in order to calculate the distance. A quick Google search for "speed of sound in centimeters per microsecond" will say that it is 0.034 cm/µs. You could do the math, but searching it is easier. Anyway, with that information, we can calculate the distance!

$$\text{Distance} = 0.034\ \text{cm}/\mu\text{s} \times 500\ \mu\text{s}$$

But this is not done! Remember that the pulse indicates the time it took for the signal to be sent out and reflected back so to get the distance so, you'll need to divide your result in half.

$$\text{Distance} = (0.034\ \text{cm}/\mu\text{s} \times 500\ \mu\text{s}) / 2$$

$$\text{Distance} = 8.5\ \text{cm}$$

So, now we know that the object is 8.5 centimetres away from the sensor.

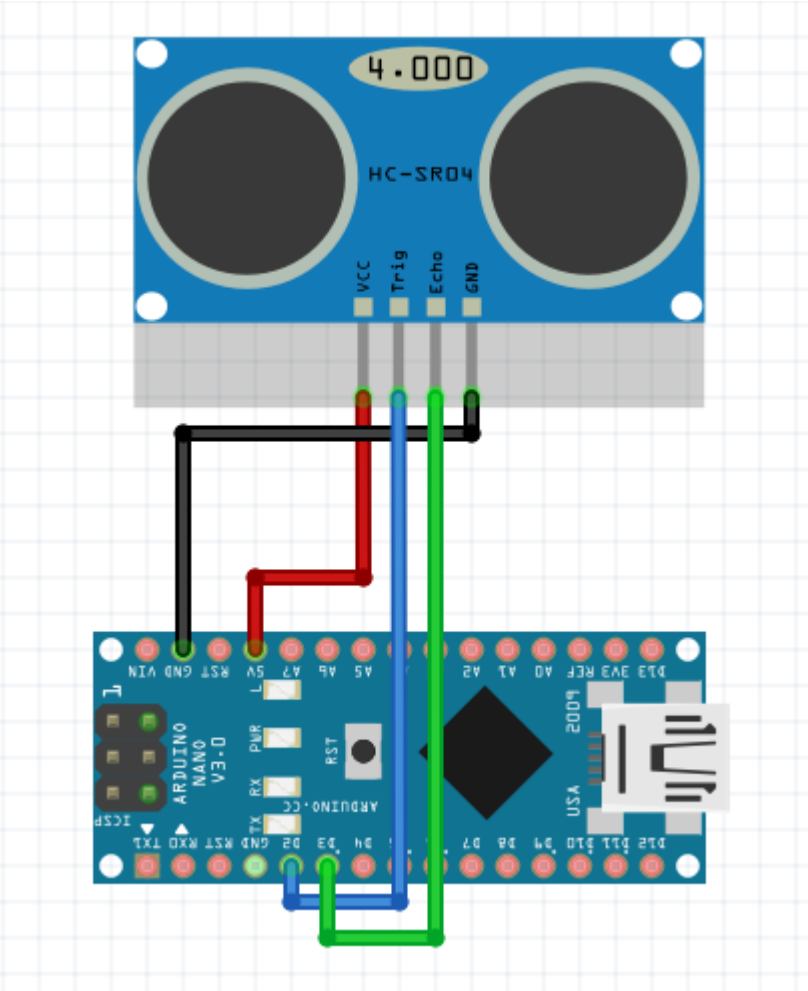
Wiring – Connecting HC-SR04 to Arduino Nano

Now that we have a complete understanding of how HC-SR04 ultrasonic distance sensor works, we can begin hooking it up to our Arduino.

Connecting the HC-SR04 to the Arduino is pretty easy.

Connect one GND pin of Arduino Nano with the GND of HC-SR04.
Connect 5V of Arduino Nano with the VCC pin of HC-SR04.

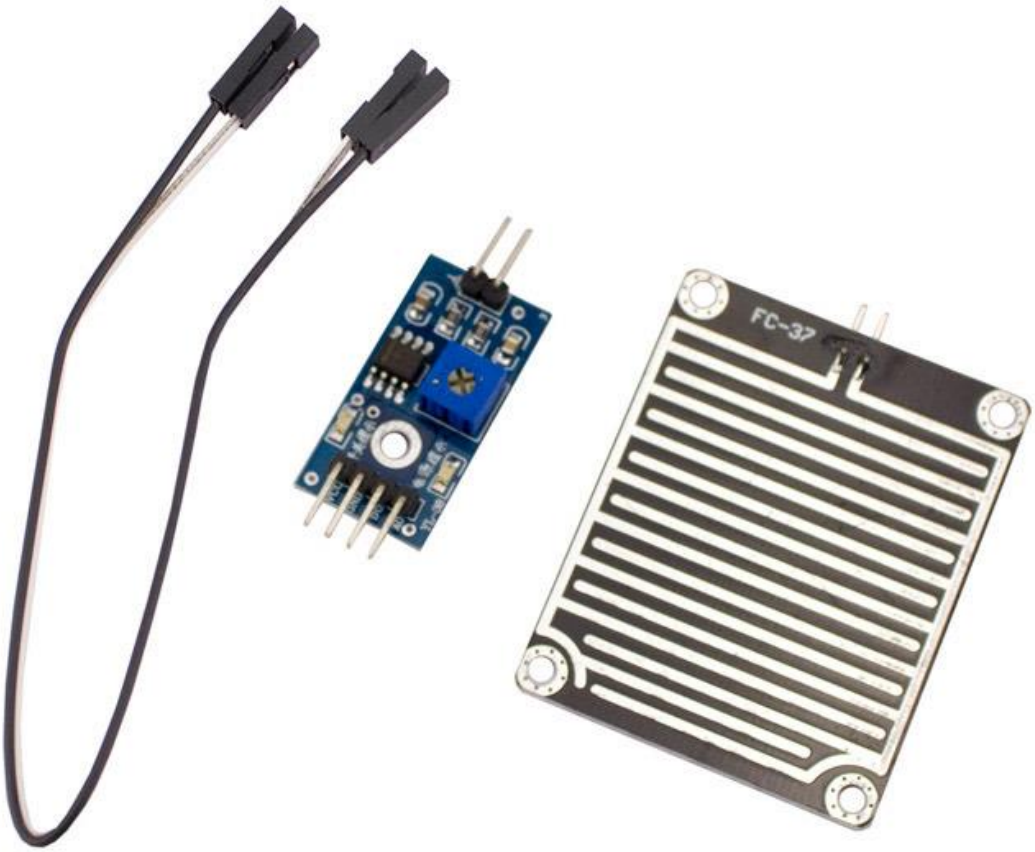
Connect ECHO pin of HC-SR04 to the D3 pin of Arduino Nano.
Connect the TRIG pin of HC-SR04 to the D2 pin of Arduino Nano.



3.5 Rain Sensors

The rain sensor module is and easy tool for rain detection. It can be used as a switch when raindrop falls through raining board and also for measuring rainfall intensity. The module features, a rain board and the control board that separated for more convenience power indicator LED and an adjustable sensitivity through a potentiometer.

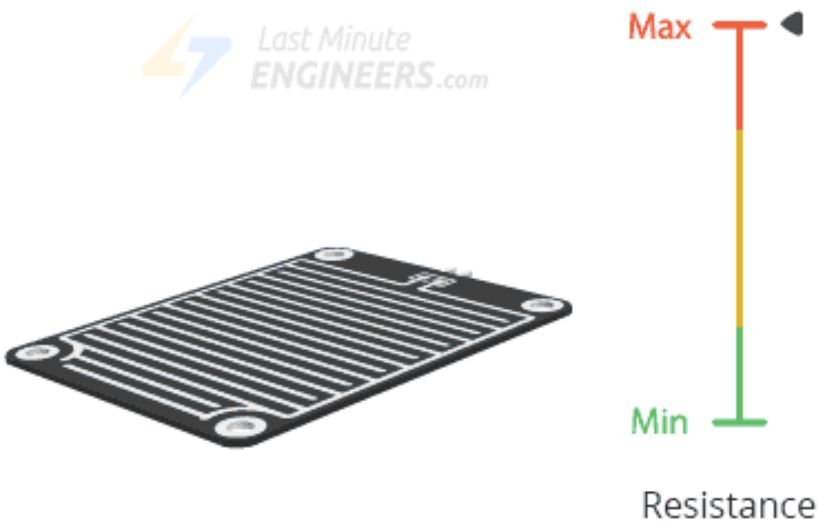
The rain sensor detects water that completes the circuits on its sensor board printed leads. The sensor board acts as a variable resistor that will change from 100 ohms when wet to 2M ohms when dry. In short the wetter the board the more current that will conducted.



How Rain Sensor works?

The working of the rain sensor is pretty straightforward.

The sensing pad with series of exposed copper traces, together acts as a variable resistor (just like a potentiometer) whose resistance varies according to the amount of water on its surface.



This resistance is inversely proportional to the amount of water:

- The more water on the surface means better conductivity and will result in a lower resistance.
-
- The less water on the surface means poor conductivity and will result in a higher resistance.

The sensor produces an output voltage according to the resistance, which by measuring we can determine whether it's raining or not.

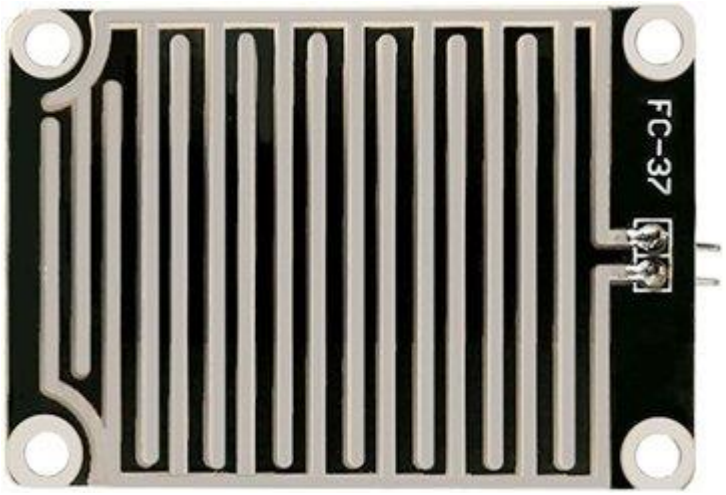
Hardware Overview

A typical rain sensor has two components.

The Sensing Pad

The sensor contains a sensing pad with series of exposed copper traces that is placed out in the open, possibly over the roof or where it can be affected by rainfall.

Usually, these traces are not connected but are bridged by water.



The Module

The sensor also contains an electronic module that connects the sensing pad to the Arduino.

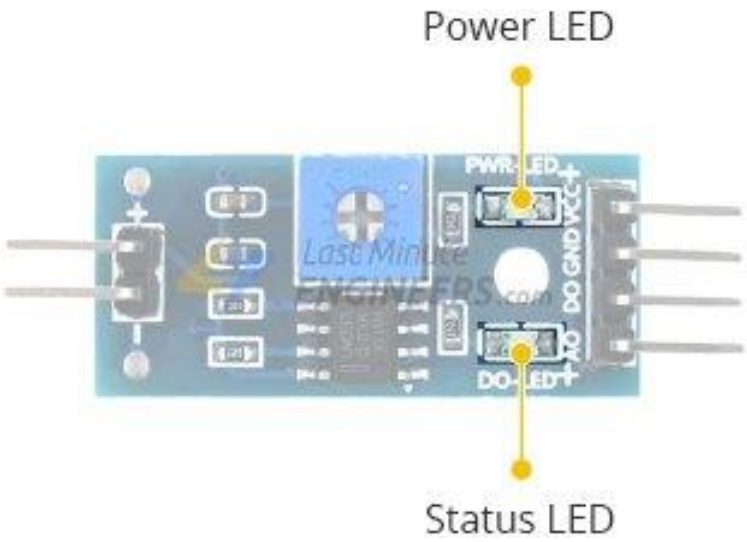
The module produces an output voltage according to the resistance of the sensing pad and is made available at an Analog Output (AO) pin.

The same signal is fed to a LM393 High Precision Comparator to digitize it and is made available at an Digital Output (DO) pin.

The module has a built-in potentiometer for sensitivity adjustment of the digital output (DO).

You can set a threshold by using a potentiometer; So that when the amount of water exceeds the threshold value, the module will output LOW otherwise HIGH.

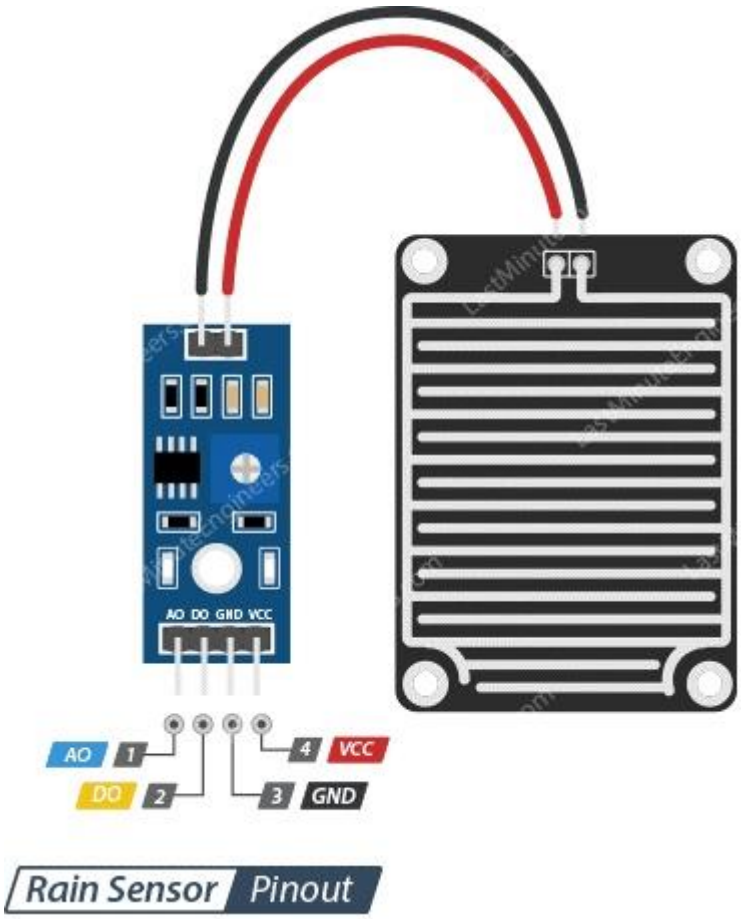
Rotate the knob clockwise to increase sensitivity and counterclockwise to decrease it.



Apart from this, the module has two LEDs. The Power LED will light up when the module is powered. The Status LED will light up when the digital output goes LOW.

Rain Sensor Pinout

The rain sensor is super easy to use and only has 4 pins to connect.



AO (Analog Output) pin gives us an analog signal between the supply value (5V) to 0V.

DO (Digital Output) pin gives Digital output of internal comparator circuit. You can connect it to any digital pin on an Arduino or directly to a 5V relay or similar device.

GND is a ground connection.

VCC pin supplies power for the sensor. It is recommended to power the sensor with between 3.3V – 5V. Please note that the analog output will vary depending on what voltage is provided for the sensor.

Wiring Rain Sensor with Arduino

Connect one GND pin of Arduino Nano with the GND of Rain Sensor.
Connect 5V of Arduino Nano with the VCC pin of Rain Sensor.
Connect AO pin of Rain Sensor to the A4 pin of Arduino Nano.

However, one commonly known issue with these sensors is their short lifespan

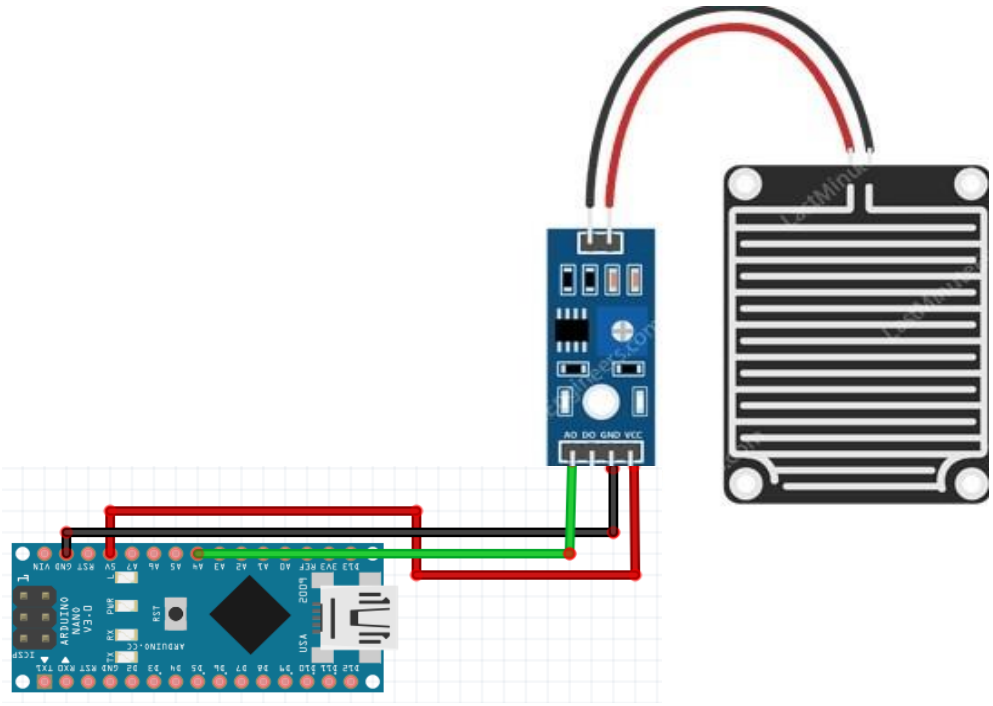
when exposed to a moist environment. Having power applied to the sensing pad constantly, speeds the rate of corrosion significantly.

To overcome this, we recommend that you do not power the sensor constantly, but power it only when you take the readings.

An easy way to accomplish this is to connect the VCC pin to a digital pin of an Arduino and set it to HIGH or LOW as as per your requirement.

Also the total power drawn by the module (with both LEDs lit) is about 8 mA, so it is okay to power the module off a digital pin on an Arduino.

The following illustration shows the wiring.



SIM800L GSM Module

SIM800L GSM/GPRS module is a miniature GSM modem, which can be integrated into a great number of IoT projects. You can use this module to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to internet through GPRS, TCP/IP, and more! To top it off, the module supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

GSM (Global system for mobile communication) is a digital mobile telephony system that is widely used all over the world. A GSM module requires a SIM (Subscriber's identity module) card to be operated and operated over a network range subscribed by the network operated. It can be connected to an Arduino through cable or Bluetooth connection. GSM module can be communicated to PIC-microcontroller using normal serial USART protocol.

GSM is a mobile communication modem; it is stand for global system mobile communication (GSM). The idea of GSM was developed at Bell 1970.It is widely used mobile communication system in the world. GSM is open and digital cellular technology used for transmitting mobile voice and data services operated at the 850 MHz,900 MHz,18000 MHz and 1900MHz frequency band.

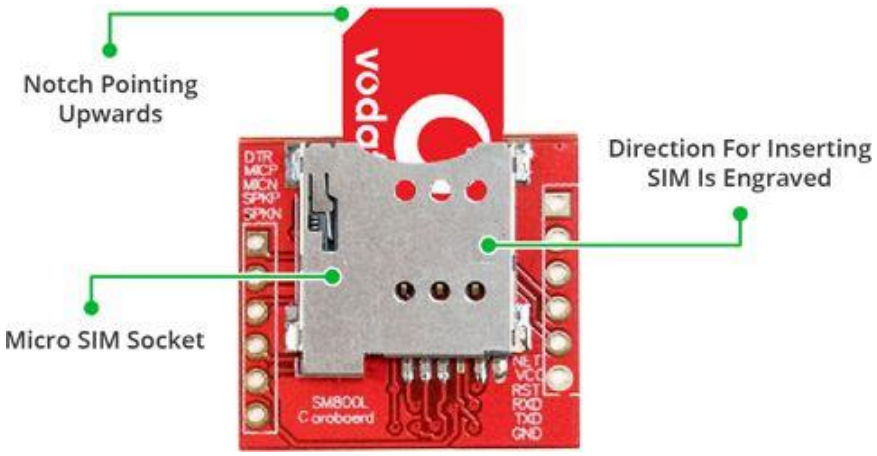
GSM system developed a digital system using Time division multiple access (TDMA) technique for communication purposed. The digital system has an ability to carry 64 kbps to 120mbps of data rates.

Hardware Overview of SIM800L GSM/GPRS module

At the heart of the module is a SIM800L GSM cellular chip from SimCom. The operating voltage of the chip is from 3.4V to 4.4V, which makes it an ideal candidate for direct LiPo battery supply. This makes it a good choice for embedding into projects without a lot of space.

All the necessary data pins of SIM800L GSM chip are broken out to a 0.1" pitch headers. This includes pins required for communication with a microcontroller over UART. The module supports baud rate from 1200bps to 115200bps with Auto-Baud detection.

The module needs an external antenna to connect to a network. The module usually comes with a Helical Antenna and solders directly to NET pin on PCB. The board also has a U.FL connector facility in case you want to keep the antenna away from the board.



There’s a SIM socket on the back! Any activated, 2G micro SIM card would work perfectly. Correct direction for inserting SIM card is normally engraved on the surface of the SIM socket.

This module measures only 1 inch^2 but packs a surprising amount of features into its little frame. Some of them are listed below:

- Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900
- Connect onto any global GSM network with any 2G SIM
- Make and receive voice calls using an external 8Ω speaker & electret microphone
- Send and receive SMS messages
- Send and receive GPRS data (TCP/IP, HTTP, etc.)
- Scan and receive FM radio broadcasts
- Transmit Power:
    - Class 4 (2W) for GSM850
    - Class 1 (1W) for DCS1800
- Serial-based AT Command Set
- FL connectors for cell antennae
- Accepts Micro SIM Card

LED Status Indicators

There is an LED on the top right side of the SIM800L Cellular Module which indicates the status of your cellular network. It'll blink at various rates to show what state it's in:

Blink every 1s

The module is running but hasn't made connection to the cellular network yet.

Blink every 2s

The GPRS data connection you requested is active.

Blink every 3s

The module has made contact with the cellular network & can send/receive voice and SMS.

Selecting Antenna

An antenna is required to use the module for any kind of voice or data communications as well as some SIM commands. So, selecting an antenna could be a crucial thing. There are two ways you can add an antenna to your SIM800L module.

The first one is a Helical GSM antenna which usually comes with the module and solders directly to NET pin on PCB. This antenna is very useful for projects that need to save space but struggles in getting connectivity especially if your project is indoors.

The second one is any 3dBi GSM antenna along with a U.FL to SMA adapter which can be obtained online for less than $3. You can snap-fit this antenna to small u.fl connector located on the top-left corner of the module. This type of antenna has a better performance and allows putting your module inside a metal case – as long the antenna is outside.

Supplying Power for SIM800L module

One of the most important parts of getting the SIM800L module working is supplying it with enough power.

Depending on which state it's in, the SIM800L can be a relatively power-hungry device. The maximum current draw of the module is around 2A during transmission burst. It usually won't pull that much, but may require around 216mA during phone calls or 80mA during network transmissions. This chart from the datasheet summarizes what you may expect:

Modes	Frequency	Current Consumption
Power down		60 uA
Sleep mode		1 mA
Stand by		18 mA
Call	GSM850	199 mA
	EGSM900	216 mA
	DCS1800	146 mA
	PCS1900	131 mA
GPRS		453 mA
Transmission burst		2:00 AM

Since SIM800L module doesn't come with onboard voltage regulator, an external power supply adjusted to voltage between 3.4V to 4.4V (Ideal 4.1V) is required. The power supply should also be able to source 2A of surge current, otherwise the module will keep shutting down. Here are some options you can consider to correctly power your GSM module.

3.7v Li-Po Battery (Recommended)

One of the cool things about Li-Po batteries is that their voltage is generally in the range of 3.7V – 4.2V, perfect for SIM800L Module. Any 1200mAh or larger sized Lithium ion/polymer battery is best since it can provide the correct voltage range even during 2 Amp spikes.

DC-DC Buck Converter

Any 2A-rated DC-DC buck converter like LM2596 would work. These are much more efficient than a liner voltage regulator like LM317 or LM338.

Warning:

You should be very careful to not to disconnect the GND before the VCC and always connect GND before VCC. Otherwise, the module can use the low voltage serial pins as ground and can get destroyed instantly.

SIM800L GSM Module Pinout

The SIM800L module has total 12 pins that interface it to the outside world. The connections are as follows:

SIM800L Pinout

NET is a pin where you can solder Helical Antenna provided along with the module.

VCC supplies power for the module. This can be anywhere from 3.4V to 4.4 volts. Remember connecting it to 5V pin will likely destroy your module! It doesn't even run on 3.3 V! An external power source like Li-Po battery or DC-DC buck converters rated 3.7V 2A would work.

RST (Reset) is a hard reset pin. If you absolutely got the module in a bad space, pull this pin low for 100ms to perform a hard reset.

RxD (Receiver) pin is used for serial communication.

TxD (Transmitter) pin is used for serial communication.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

RING pin acts as a Ring Indicator. It is basically the 'interrupt' out pin from the module. It is by default high and will pulse low for 120ms when a call is received. It can also be configured to pulse when an SMS is received.

DTR pin activates/deactivates sleep mode. Pulling it HIGH will put module in sleep mode, disabling serial communication. Pulling it LOW will wake the module up.
MIC± is a differential microphone input. The two microphone pins can be connected directly to these pins.
SPK± is a differential speaker interface. The two pins of a speaker can be tied directly to these two pins.

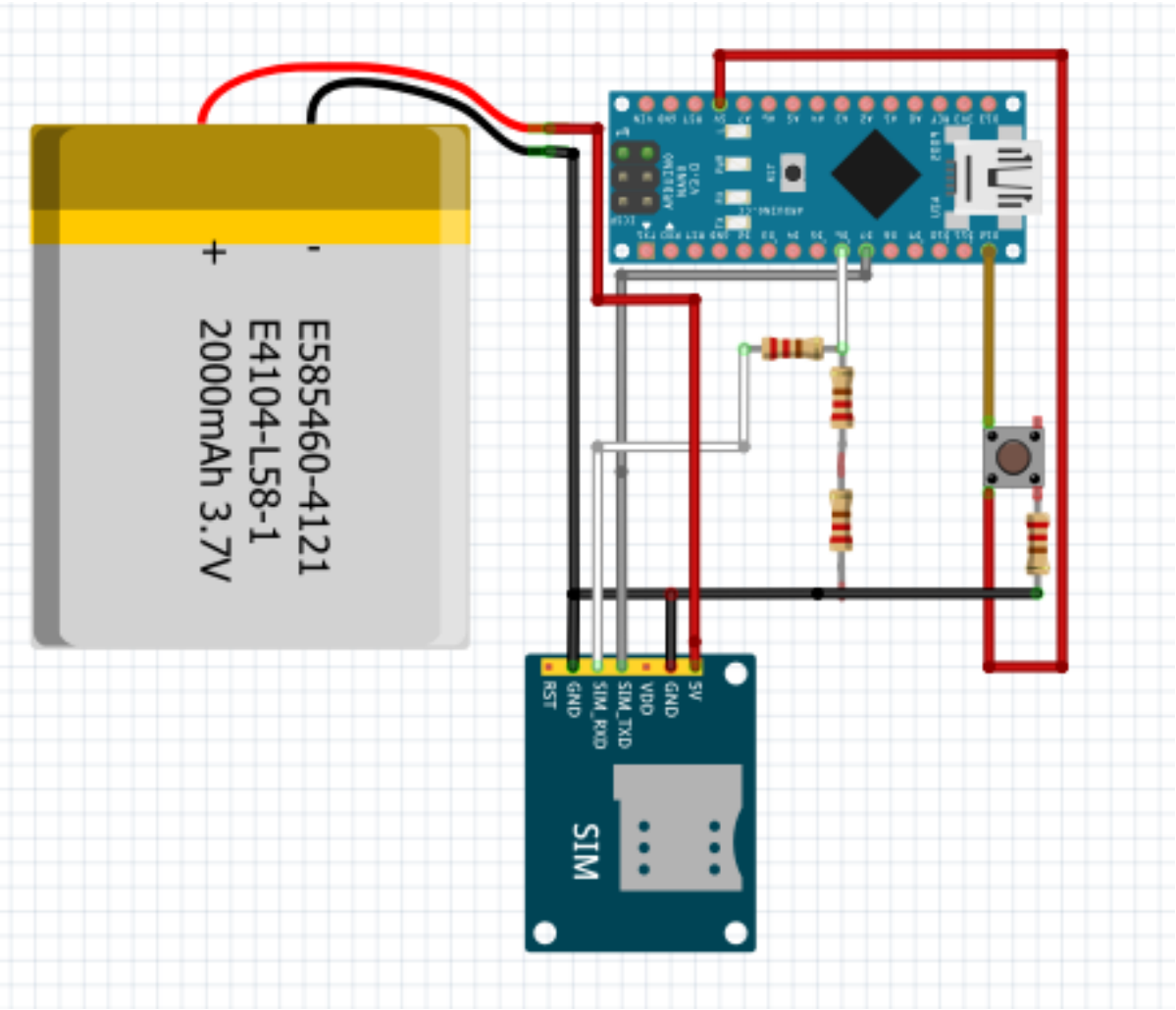
Wiring – Connecting SIM800L GSM module to Arduino NANO

Now that we know everything about the module, we can begin hooking it up to our Arduino!

Start by soldering/connecting the antenna, insert fully activated Micro SIM card in the socket. Now, connect Tx pin on module to digital pin 7 on Arduino as we'll be using software serial to talk to the module.

We cannot directly connect Rx pin on module to Arduino's digital pin as Arduino Nano uses 5V GPIO whereas the SIM800L module uses 3.3V level logic and is NOT 5V tolerant. This means the Tx signal coming from the Arduino Uno must be stepped down to 3.3V so as not to damage the SIM800L module. There are several ways to do this but the easiest way is to use a simple resistor divider. A 10K resistor between SIM800L Rx and Arduino digital pin D6, and 20K between SIM800L Rx and GND would work fine.

Now we are remaining with the pins that are used to supply power for the module. As you have multiple choices for powering up the module, we can use what suits you bests. Here we are using one 3.7 V Li Ion 2200mAh Li-Po Battery.



Wiring SIM800L GSM GPRS Module with Arduino Nano

TIP

In case you are using LM2596 buck converter to power up the module, remember to common all the ground in the circuit.

Once you have everything hooked up you are ready to go!

Sim 800l GSM Module is Control through Arduino Nano by commands called “AT Commands”.

They are as follow :-

AT – It is the most basic AT command. It also initializes Auto-baud’er. If it works you should see the AT characters echo and then OK, telling you it’s OK and it’s understanding you correctly! You can then send some commands to query the module and get information about it such as

AT+CSQ – Check the ‘signal strength’ – the first # is dB strength, it should be higher than around 5. Higher is better. Ofcourse, it depends on your antenna and location!

AT+CCID – get the SIM card number – this tests that the SIM card is found OK and you can verify the number is written on the card.

AT+CREG? Check that you’re registered on the network. The second # should be 1 or 5. 1 indicates you are registered to home network and 5 indicates roaming network. Other than these two numbers indicate you are not registered to any network.

ATI – Get the module name and revision

AT+COPS? – Check that you’re connected to the network, in this case BSNL

AT+COPS=? – Return the list of operators present in the network.

AT+CBC – will return the lipo battery state. The second number is the % full (in this case its 93%) and the third number is the actual voltage in mV (in this case, 3.877 V)

AT+CMGF=1 – Selects SMS message format as text. Default format is Protocol Data Unit (PDU)

AT+CMGS=+ZZxxxxxxxxxx – Sends SMS to the phone number specified. The text message entered followed by a 'Ctrl+z' character is treated as SMS. 'Ctrl+z' is actually a 26^{th} non-printing character described as 'substitute' in ASCII table. So, we need to send 26_{DEC} ($1A_{HEX}$) once we send a message.

AT+CMGF=1 – Selects SMS message format as text. Default format is Protocol Data Unit (PDU)

AT+CNMI=1,2,0,0,0 – specifies how newly arrived SMS messages should be handled. This way you can tell the SIM800L module either to forward newly arrived SMS messages directly to the PC, or to save them in message storage and then notify the PC about their locations in message storage.

Its response starts with +CMT: All the fields in the response are comma-separated with first field being phone number. The second field is the name of person sending SMS. Third field is a timestamp while forth field is the actual message.

ATD+ +ZZxxxxxxxxxx; – Dials a specified number. The semicolon (;) modifier at the end separates the dial string into multiple dial commands. All but the last command must end with the semicolon (;) modifier.

ATH – Hangs up the call

ATA – Accepts incoming call.

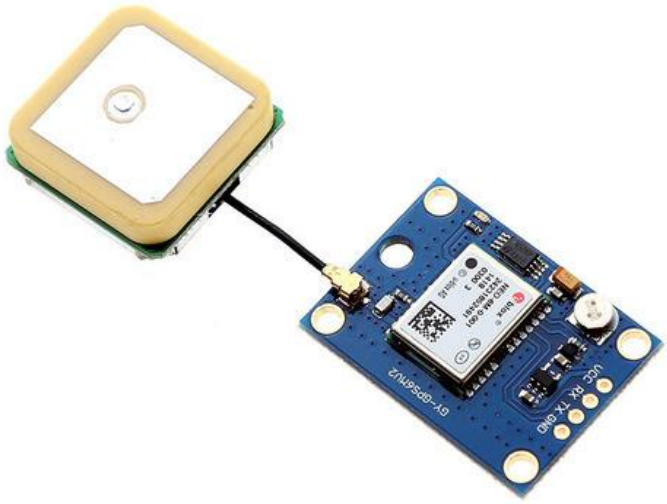
ATH – Hangs up the call. On hanging up the call it sends NO CARRIER on the serial monitor indicating call couldn't connect.

GPS(Global positioning System)is a satellite navigation system used to determined the ground position of an object.GPS technology was first used by the United state military in 1960s and expanded into civilian use over the next few decades. Today, GPS receiver are included in many commercial products, such as automobiles, Smartphone, exercise watches etc.

GPS system include 24 satellite deployed in space about 12000 miles (19300 kilometre) above the earth's surface. The orbit earth one every 12 hours at extremely fast pace of roughly 7000 miles per hour. The satellites are evenly spread so that satellites are accessible via direct line-of-sight anywhere on the globe. The navigation messages are broadcast at a rate of 50 bits per second. Utilizing this collocation of data, GPS receiver in order to generate position data.

Arduino project ability to sense locations with NEO-6M GPS Module that can track up to 22 satellites and identifies locations anywhere in the world. It may serve as a great launch pad for anyone looking to get into the world of GPS.

They are low power (suitable for battery powered devices), inexpensive, easy to interface with and are insanely popular among hobbyists.

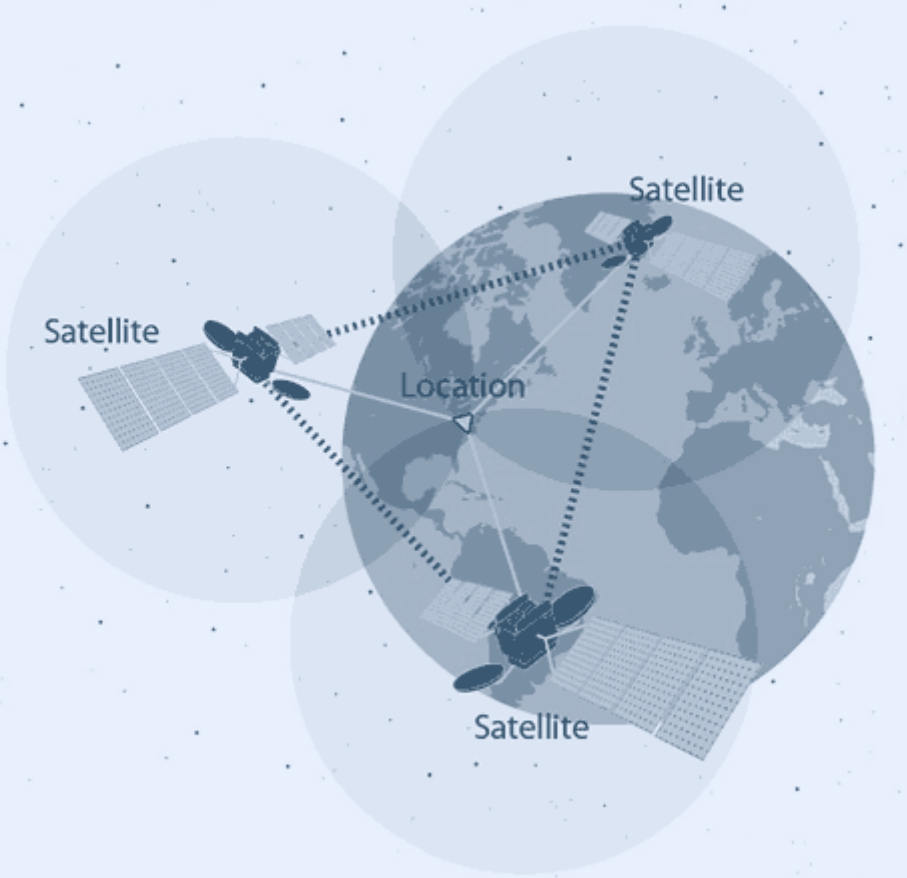


NEO 6m GPS Module

How does GPS work?

GPS receivers actually work by figuring out how far they are from a number of satellites. They are pre-programmed to know where the GPS satellites are at any given time.

The satellites transmit information about their position and the current time in the form of radio signals towards the Earth. These signals identify the satellites and tell the receiver where they are located.



The receiver then calculates how far away each satellite is by figuring out how long it took for the signals to arrive. Once it has information on how far away at least three satellites are and where they are in space, it can pinpoint your location on Earth.

This process is known as Trilateration.

Hardware Overview of NEO-6M GPS Module

NEO-6M GPS Chip

At the heart of the module is a NEO-6M GPS chip from u-blox. The chip measures less than the size of a postage stamp but packs a surprising amount of features into its little frame.

It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current.

Unlike other GPS modules, it can do up to 5 location updates a second with 2.5m Horizontal position accuracy. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second.

One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch.

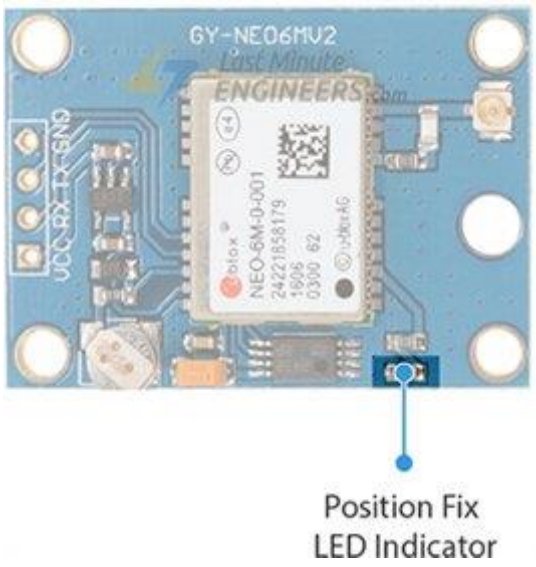
The necessary data pins of NEO-6M GPS chip are broken out to a 0.1″ pitch headers. This includes pins required for communication with a microcontroller

over UART. The module supports baud rate from 4800bps to 230400bps with default baud of 9600.

Here are complete specifications:

Receiver Type	50 channels, GPS L1(1575.42Mhz)
Horizontal Position Accuracy	2.5m
Navigation Update Rate	1HZ (5Hz maximum)
Capture Time	Cool start: 27sHot start: 1s
Navigation Sensitivity	-161dBm
Communication Protocol	NMEA, UBX Binary, RTCM
Serial Baud Rate	4800-230400 (default 9600)
Operating Temperature	-40°C ~ 85°C
Operating Voltage	2.7V ~ 3.6V
Operating Current	45mA
TXD/RXD Impedance	510Ω

Position Fix LED Indicator



There is an LED on the NEO-6M GPS Module which indicates the status of Position Fix. It'll blink at various rates depending on what state it's in:

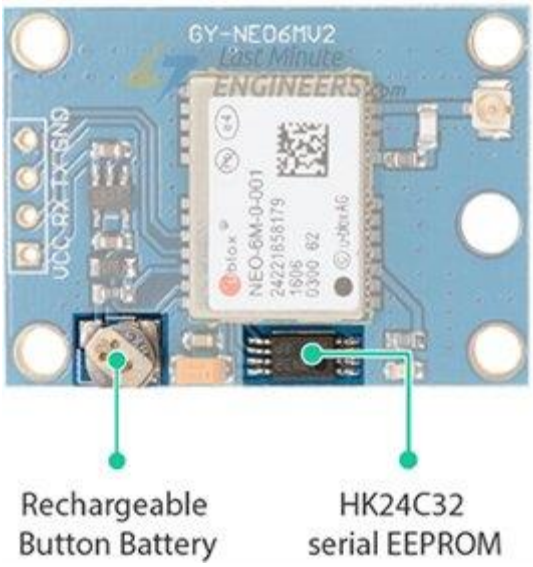
- No Blinking – It's searching for satellites.
- Blink every 1s – Position Fix is found(The module can see enough satellites).

3.3V LDO Regulator

The operating voltage of the NEO-6M chip is from 2.7 to 3.6V. But the good news is that, the module comes with MIC5205 ultra-low dropout 3V3 regulator from MICREL.

The logic pins are also 5-volt tolerant, so we can easily connect it to an Arduino or any 5V logic microcontroller without using any logic level converter.

Battery & EEPROM



The module is equipped with an HK24C32 two wire serial EEPROM. It is 4KB in size and connected to the NEO-6M chip via I2C.

The module also contains a rechargeable button battery which acts as a super-capacitor.

An EEPROM together with battery helps retain the battery backed RAM (BBR). The BBR contains clock data, latest position data(GNSS orbit data) and module configuration. But it's not meant for permanent data storage.

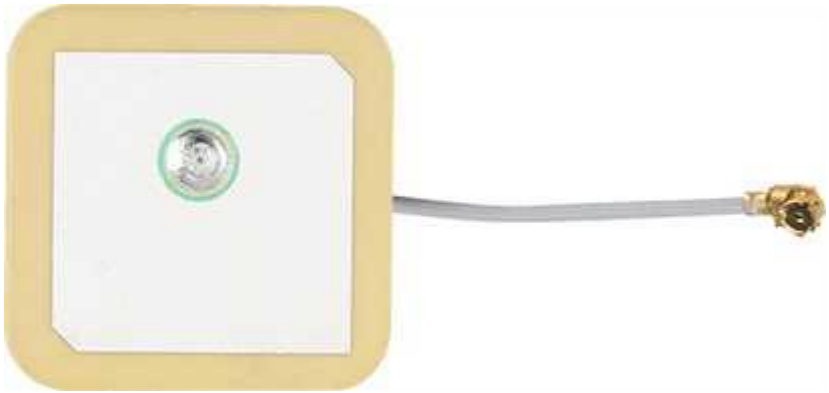
As the battery retains clock and last position, time to first fix (TTFF) significantly reduces to 1s. This allows much faster position locks.

Without the battery the GPS always cold-start so the initial GPS lock takes more time.

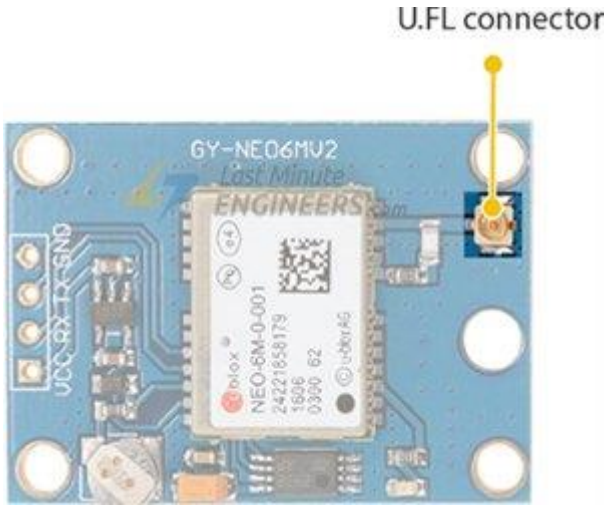
The battery is automatically charged when power is applied and maintains data for up to two weeks without power.

Antenna

An antenna is required to use the module for any kind of communication. So, the module comes with a patch antenna having -161 dBm sensitivity.



You can snap-fit this antenna to small U.FL connector located on the module.



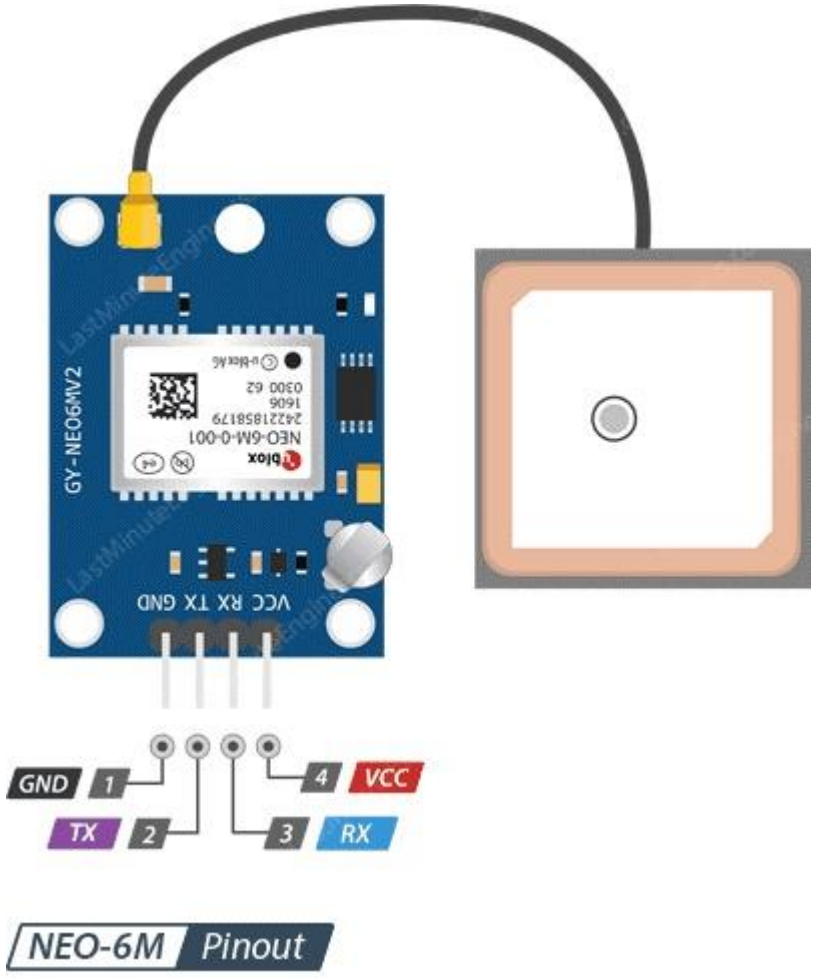
Patch antenna is great for most projects. But if you want to achieve more sensitivity or put your module inside a metal case, you can also snap on any 3V active GPS antenna via the U.FL connector.

TIP

U.FL connectors are small, delicate and are not rated for strain. To prevent damaging the U.FL connection, we recommend threading the U.FL cable through the mounting hole, then attach the U.FL connectors.

NEO-6M GPS Module Pinout

The NEO-6M GPS module has total 4 pins that interface it to the outside world. The connections are as follows:



GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

TxD (Transmitter) pin is used for serial communication.

RxD (Receiver) pin is used for serial communication.

VCC supplies power for the module. You can directly connect it to the 5V pin on the Arduino.

Wiring NEO-6M GPS module with Arduino Nano

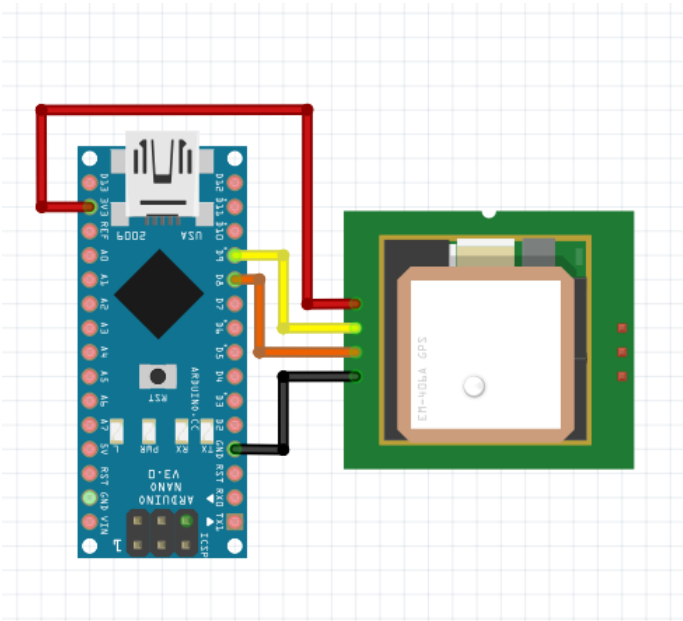
Now that we know everything about the module, we can begin hooking it up to our Arduino!

Start by connecting the patch antenna to the U.FL connector. Remember to thread the U.FL cable through one of the mounting holes for robust connection.

The module usually comes with header pins unsoldered. So, you'll need to solder them.

Now, connect Tx and Rx pin on module to digital pin#9 and #8 respectively on Arduino; as we'll be using software serial to talk to the module.

Next, connect VCC pin to the 3.3V pin on the arduino and GND to ground.

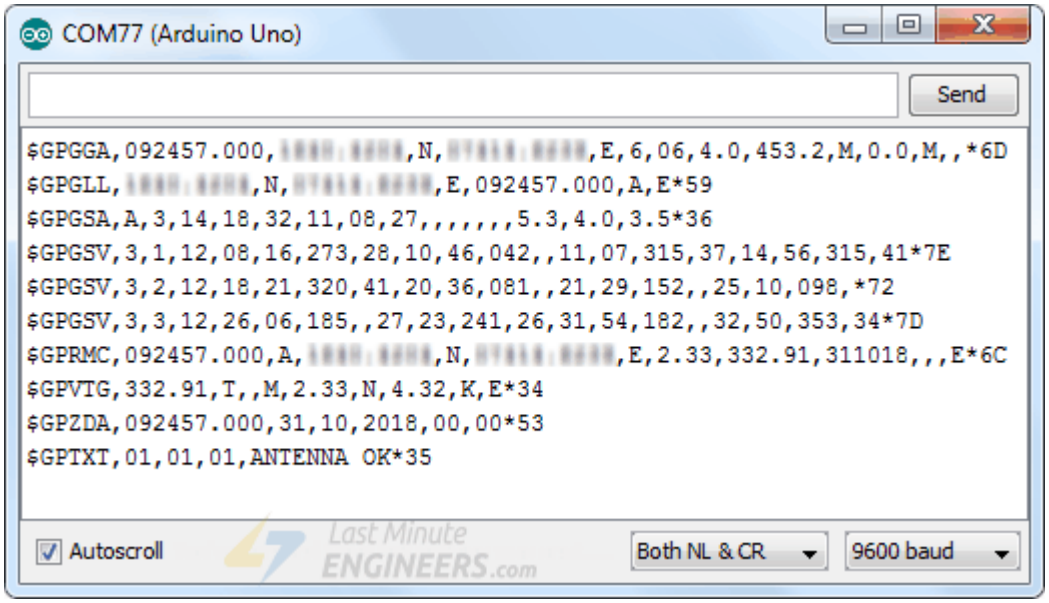


Once you have everything hooked up you are ready to go!

Arduino Code – Reading GPS Data

The best thing about any GPS receiver is that they start spitting out data as soon as you turn them ON.

Upload the program and open up the serial monitor from the Arduino IDE. Remember to select 9600 baud. You should see text like the following:



The data you are getting over a serial interface are actually NMEA sentences.

NMEA is an acronym for the National Marine Electronics Association. This is a standard message format for Nearly all GPS receivers.

The NMEA standard is formatted in lines of data called sentences. Each sentence is comma separated to make it easier to parse by computers and microcontrollers.

These NMEA sentences are sent out at an interval called the update rate.

NEO-6M GPS module updates this information once per second(1Hz frequency) by default. But you can configure it for up to 5 updates per second(5Hz frequency).

Parsing NMEA Sentences

There are many sentences in the NMEA standard, the most common ones are:

- $GPRMC (Global Positioning Recommended Minimum Coordinates) provides the time, date, latitude, longitude, altitude and estimated velocity.
- $GPGGA sentence provides essential fix data which provide 3D location and accuracy data.

Let's take an example of $GPRMC NMEA sentence from a GPS receiver.

$GPRMC, 123519, A, 4807.038, N, 01131.000, E,022.4, 084.4, 230394, 003.1, W*6A

$	Every NMEA sentence starts with $ character.
GPRMC	Global Positioning Recommended Minimum Coordinates
123519	Current time in UTC – 12:35:19
A	Status A=active or V=Void.
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
022.4	Speed over the ground in knots
084.4	Track angle in degrees True
220318	Current Date – 22rd of March 2018
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

Let’s take an example of $GPGGA NMEA sentence.

$GPGGA, 123519, 4807.038, N, 01131.000, E, 1, 08, 0.9, 545.4, M, 46.9, M, , *47

$	Starting of NMEA sentence.
GPGGA	Global Positioning System Fix Data
123519	Current time in UTC – 12:35:19
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
1	GPS fix
08	Number of satellites being tracked
0.9	Horizontal dilution of position
545.4,M	Altitude in Meters (above mean sea level)
46.9,M	Height of geoid (mean sea level)
(empty field)	Time in seconds since last DGPS update
(empty field)	DGPS station ID number
*47	The checksum data, always begins with *

For more information about NMEA sentences and what data they contain, check out gpsinformation.org

Arduino Code – TinyGPS Library

Often for our projects, we need to parse NMEA sentences into useful information. To simplify our work, we have a library called TinyGPS++ library.

This library does a lot of heavy lifting required for receiving data from GPS modules, such as reading and extracting useful data in the background. So, we don't need to worry about icky parsing work.

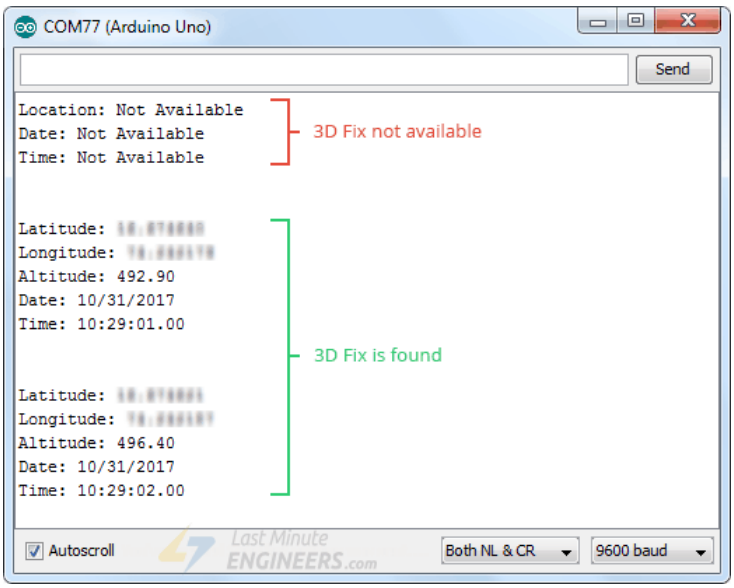
Thanks to Mikal Hart for his great contribution. His website Arduiniana.org has a full overview of all of the capabilities of the TinyGPS++ library.

To install it, open the Arduino IDE, go to Sketch > Include Library > Add .ZIP Library, and then select the TinyGPSPlus ZIP file that you just downloaded. If you need more details on installing a library, visit this Installing an Arduino Library tutorial.

Once you have the library installed, you can copy below sketch into the Arduino IDE.

The following test sketch will print the location information(Latitude, Longitude & Altitude) and UTC(Date & Time) on the serial monitor. Try the sketch out; and then we will explain it in some detail.

This is how the output looks like on the serial monitor.



Other Useful Functions In TinyGPS++ Library

There are a few useful functions you can use with TinyGPS++ object. Few of them are listed below:

- `gps.speed.value()` function returns current ground speed in 100ths of a knot.

- `gps.course.value()` function returns current ground course in 100ths of a degree.

- `gps.satellites.value()` function returns the number of visible, participating satellites.

- `gps.hdop.value()` function returns horizontal diminution of precision.

- If you want to know how old an object's data is, call its `age()` method, which returns the number of milliseconds since its last update. If this returns a value greater than 1500 or so, it may be a sign of a problem like a lost fix.

- If you want to extract data from any other NMEA sentence. You can use library's custom extraction functionality by telling TinyGPS++ the sentence name and the field number you are interested in, like this:`TinyGPSCustom magneticVariation(gps, "GPRMC", 10)`And you can query it just like the others:`magneticVariation.value()`

Buzzer

A buzzer is a small yet efficient component to add sound features to add sound to our project system. It is very small and compact 2 pin structure. Buzzer is in the lower portion of the audible frequency range of 20 Hz to 20 KHz. This is accomplished by covering an electric, oscillating signal in the audible range, into mechanical energy.

Buzzer

Wiring of Buzzer with Arduino

Connect GND pin of Arduino Nano with the Negative pin of buzzer.

Connect D5 pin of Arduino Nano with the possitive pin of buzzer.

Jumper Wire

A jumper wire is an electrical wire that has connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wire are typically with breadboards and other prototyping tools in order to make it easy to change a circuit as needed. Individual jumper wires are fitted by inserting their end connectors into the slots provided in a breadboard the header connected of a circuit board or piece of test equipment. Jumper wire are in three version :-

1. male to male,
2. male to female
3. female to male.

Jumper wire

SOFTWARE DESCRIPTION

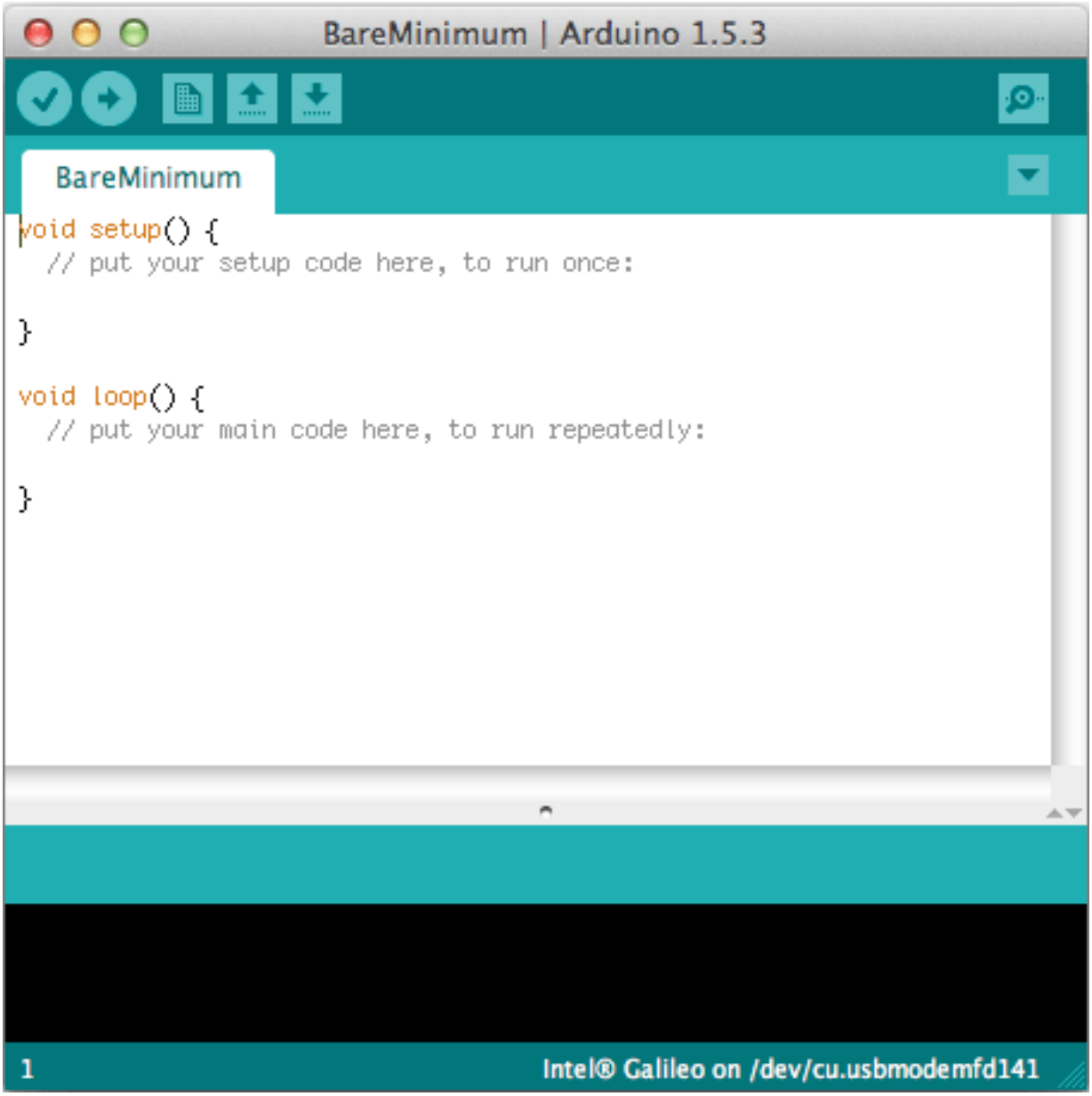
Arduino IDE

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, Mac OS and Linux) that is written in the programming language java. It is used and uploads programs to Arduino compatible boards.

The Arduino IDE supports the language C and C++ using special rules of codes structuring. The Arduino IDE supplies a software library from the wiring project which provides many common input and output input basic functions, for starting the sketch and the main program loop that are compiled and linked with a program

Arduino IDE is an open source that is mainly used for writing and compiling the code into the Arduino module. It is official software making code compilation too easy that even a common person with no prior technical knowledge can get their feet with the learning process. A different range of Arduino modules available including Arduino Uno, Arduino mega, Arduino Nano, and many more. Each of them consist a microcontroller on the board that is actually programmed and accepts the information in the form of code. The IDE environment mainly contains two basic parts: Editor and compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino module.

Arduino IDE


BareMinimum | Arduino 1.5.3
BareMinimum
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
1
Intel® Galileo on /dev/cu.usbmodemfd141

Arduino Project

SMART BLIND STICK WITH WATER DETECTION AND LOCATION TACKING MECHANISM

A survey by WHO (World Health Organization) carried out in 2011 estimates that in the world, about 1% of the human population is visually impaired (about 70 million people) and amongst them, about 10% are fully blind (about 7 million people) and 90% (about 63 million people) with low vision. The main problem with blind people is how to navigate their way to wherever they want to go. Such people need assistance from others with good eyesight. As described by WHO, 10% of the visually impaired have no functional eyesight at all to help them move around without assistance and safely. This study proposes a new technique for designing a smart stick to help visually impaired people that will provide them navigation. The conventional and archaic navigation aids for persons with visual impairments are the walking cane (also called white cane or stick) and guide dogs which are characterized by a many imperfections. The most critical shortcomings of these aids include: essential skills and training phase, range of motion, and very insignificant information communicated been communicated. Our approach modified this cane with some electronics components and sensors, the electronic aiding devices are designed to solve such issues. The ultrasonic sensor, water sensor, buzzer, GPS and GSM are used to record information about the presence of obstacles on the road. Ultrasonic sensors have the capacity to detect any obstacle within the distance range of 2 cm-450 cm. Therefore whenever there is an obstacle in this range it will alert the user. Water sensor is used to detect if there is water in path of the user. Most blind guidance systems use ultrasound because of its immunity to the environmental noise. With the rapid advances of modern technology both in hardware and software it has become easier to provide intelligent navigation system to the visually impaired. Also, high-end technological solutions have been introduced recently to help blind persons navigate independently. Whenever the user wants to locate it, such a person will press a button on remote control and buzzer will ring, then the person can get the idea of where the stick is placed. Vision is the most important part of human physiology as 83% of information human being gets from the environment is via sight. The 2011 statistics by the World Health Organization (WHO) estimates that there are 70 million people in the world living with visual impairment, 7 million of which are blind and 63 million with low vision. The conventional and oldest mobility aids for persons with visual impairments are characterized with many limitations. Some inventions also require a separate power supply or navigator which makes the user carry it in a bag every time they travel outdoor. These bulky designs will definitely make the user to be exhausted.

OBJECTIVES

Visually impaired people are the people who find it difficult to recognize the smallest detail with healthy eyes. The objectives of this research work include as follows:

1. To design an assistive technology for visually impaired people that can detect obstacles and provide alternative routes for the blind

2. To alarm the user through vibration to determine the obstacles direction sources

3. To help the user find his stick when he mistakenly loses it somewhere. Through this smart blind stick, visually impaired people will have so much of assistance.

4. In case of any problem, with the function of Global Positioning System (GPS), we can track their whereabouts.

BLOCK DIAGRAM

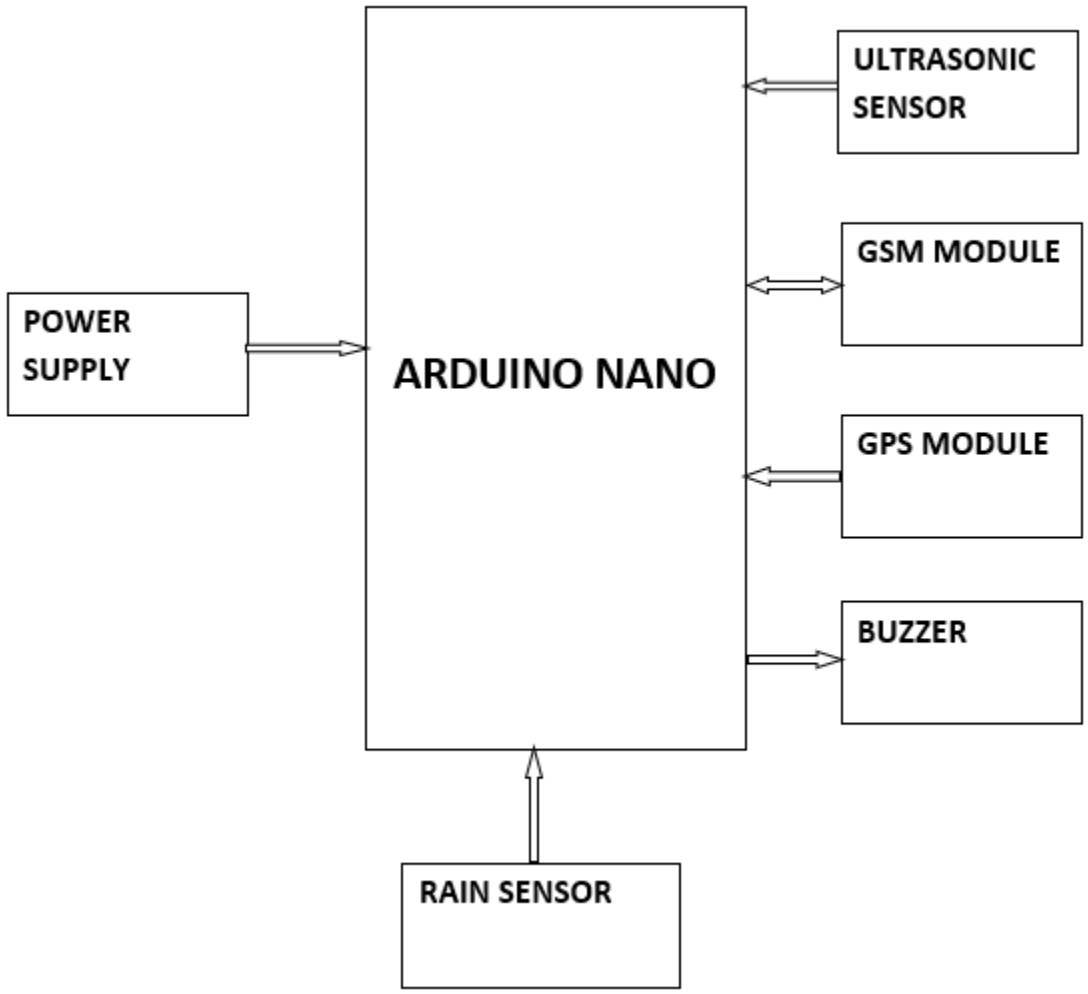


Figure:- Block diagram Smart Blind Stick

Block diagram description

The above block diagram shows the block diagram of our project. A DC 9voltage is supply though the power source to the Nano Arduino. All the sensor are interface with the Nano Arduino. Ultrasonic sensor is connected to the Arduino Nano then processes this data and calculates if the obstacle is close enough. If the obstacles are close the Nano Arduino sends a signal to sound a buzzer. It also detected and sound a different buzzer if it where detects water and alerts the blind.

Circuit Diagram

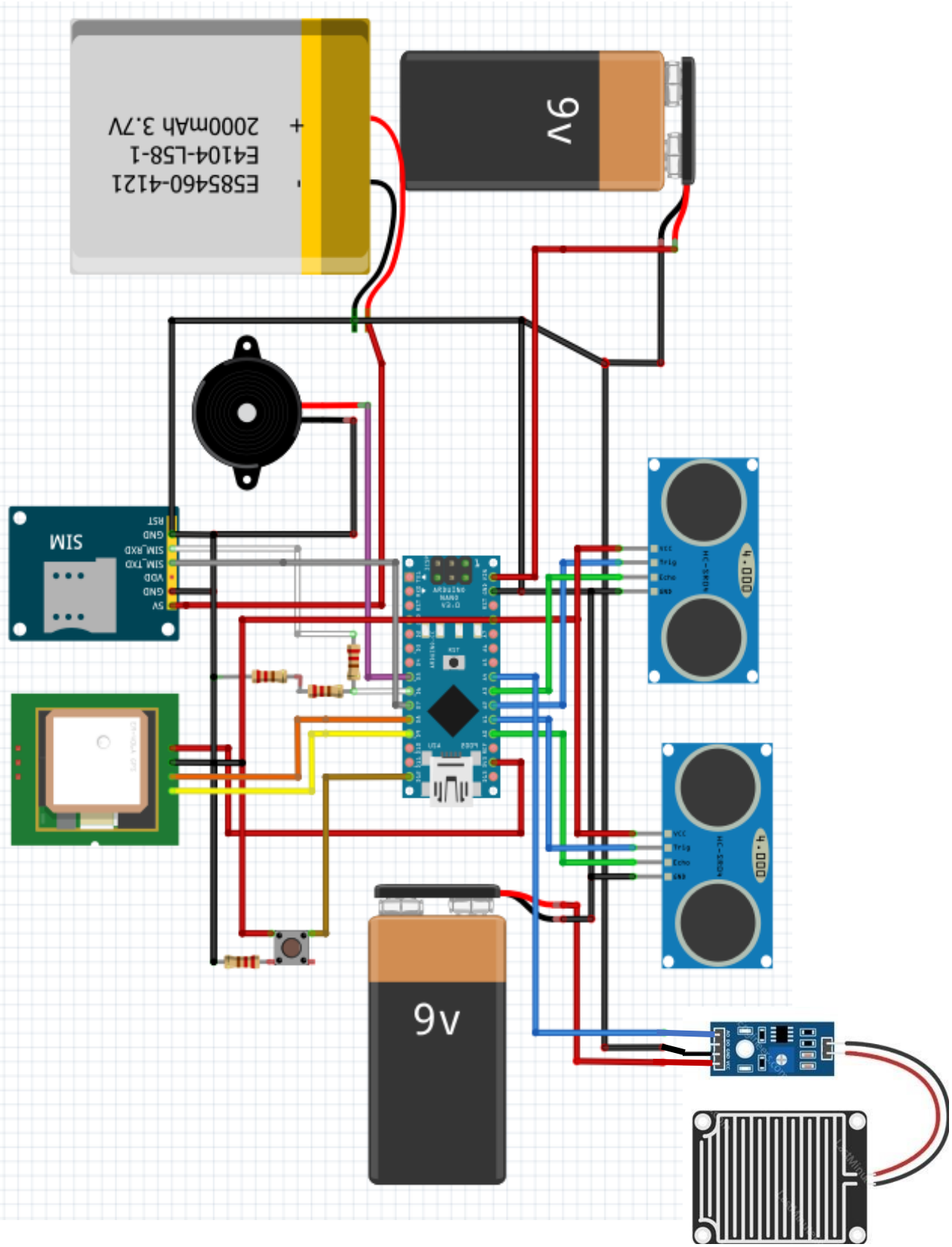


Figure:-Circuit diagram

Circuit diagram description :-

The above figure shows the circuit diagram of Smart Blind Stick With Water Detection and Location Tacking mechanism. We can see an **Arduino Nano** is used to control all the sensors.

The complete system is powered by three 9V batteries where two of them regulated to +5V for Rain Sensor and GPS. One 9V battery is for Arduino Nano at default and one 3.7V 2200 mA LI-Ion battery for Sim 800l GSM Module.

Both **Ultrasonic Sensors** are powered by 5V pin of Arduino Nano. and the trigger1 and Echo1 pin of First **Ultrasonic Sensor** is connected to Arduino Nano pin A1 and A0 as shown above. And second Ultrasonic Sensor is connected to the trigeer2 and echo2 to the A2 and A3. GND of both Ultrasonic sensor is common GND to Arduino Nano.

The **Rain sensor is power by the 5v in VCC by 9v battery** regulated to +5V **and Gnd is connected to GND of Arduino, AO pin of the Rain sensor is connected** to Arduino Nano **A4.**

GPS Module connection are as follows Vcc is connected to 3.3v or regulated 5V, Ground is to GND, Rx pin is to D8 of Arduino nano and Tx is to D9.

GSM Module

GSM Module is used to send the SMS. In our project we use to GSM Module connection are as follows Vcc is connected to 3.7v Li-Ion battery and GND to GND of Arduino Nano and Rx is to D6 and Tx is to D7 of Arduino Nano.

The output of the board is given by the **Buzzer.** The negative of buzzer is connected to GND of Arduino Nano and the positive of buzzer is connected to pin D5 of Arduino Nano.

WORKING

In this project we use Arduino nano buzzer, Ultrasonic Sensor Module, Rain Sensor Module, GSM Module, GPS Module.

The basic working principle of the Ultrasonic Sensor module was already described. This Ultrasonic Sensor sends a signal through a trigger pin which when blocked by an obstacle returns high at the echo pin.

The time taken between this interval is used to calculate the distance in cm. The buzzer produces a sound that keeps on increasing as the distance from obstacle gets less after a certain set minimum distance limit.

Its has Rain Drop sensor which detect the water and send signal to Arduino nano then Arduino nano activate the buzzer.

Its also has GPS Module which give the coordinates of blind person and that is send to any mobile number by using GSM module when the blind person press the button which is connected to the D12 of Arduino Nano.

This button is an emergency button and send sms in following manner.

SOURCE CODE

```
// End of Program

#include <SoftwareSerial.h>
#include <TinyGPS++.h>

const int buttonpin = 12;
int ButtonValue;
float lattitude, longitude;

float a[2];

float *p;

SoftwareSerial gpsSerial(8, 9);

SoftwareSerial gsmSerial(6, 7);

TinyGPSPlus gps;

const int trigPin1 = A1;

const int echoPin1 = A0;

long duration1;

int distance1;

const int trigPin2 = A2;

const int echoPin2 = A3;

long duration2;

int distance2;

void setup()

{

  pinMode(trigPin1, OUTPUT);

  pinMode(echoPin1, INPUT);

  pinMode(5, OUTPUT);

  pinMode(A4, INPUT);
```

```
  Serial.begin(9600);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(buttonpin, INPUT);
  Serial.begin(9600);
  delay(1000);
  gpsSerial.begin(9600);
  delay(1000);
  gsmSerial.begin(9600);
  delay(1000);
  Serial.print("----Tracking----\n");
  Serial.print("***Location***\n");
  gsmSerial.println("AT+CNMI=2,2,0,0,0");
  delay(3000);
  Serial.print("Initializing……\n");
  delay(2000);
  Serial.print("==== System Ready ==== \n ");
  delay(1000);
}
void loop()
{
  digitalWrite(trigPin1, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin1, HIGH);
```

```
  delayMicroseconds(10);

  digitalWrite(trigPin1, LOW);

  duration1 = pulseIn(echoPin1, HIGH);

  distance1 = duration1 * 0.034 / 2;

  Serial.print("Distance1: ");

  Serial.println(distance1);

  digitalWrite(trigPin2, LOW);

  delayMicroseconds(2);

  digitalWrite(trigPin2, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin2, LOW);

  duration2 = pulseIn(echoPin2, HIGH);

  distance2 = duration2 * 0.034 / 2;

  Serial.print("Distance2: ");

  Serial.println(distance2);
  ButtonValue = digitalRead(buttonpin);

  if ((distance1 < 30 && distance1 > 20) || (distance2 < 30 && distance2 >
20))
  {

    digitalWrite(5, HIGH);

    delay(1000);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);

    delay(1000);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);
```

```
    delay(1000);

    digitalWrite(5, LOW);
  }

  else if ((distance1 < 20 && distance1 > 10) || (distance2 < 20 && distance2
> 10))
  {

    digitalWrite(5, HIGH);

    delay(500);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);

    delay(500);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);

    delay(500);

    digitalWrite(5, LOW);
  }

  else if ((distance1 < 10 && distance1 > 0) || (distance2 < 10 && distance2 >
0))
  {

    digitalWrite(5, HIGH);

    delay(100);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);

    delay(100);

    digitalWrite(5, LOW);

    digitalWrite(5, HIGH);

    delay(100);
```

```
  digitalWrite(5, LOW);
}

digitalWrite(5, LOW);

int sensorValue = digitalRead(A4);

if (sensorValue == 0)
{

  digitalWrite(5, HIGH);

  delay(1500);

  digitalWrite(5, LOW);

  digitalWrite(5, HIGH);

  delay(1500);
}

else if (ButtonValue == 1)

{

  Serial.println("Button Pressed");
  delay(1000);
  Serial.println(ButtonValue);
  delay(2000);

  SendMessage();
}

if (gsmSerial.available() > 0)

  Serial.write(gsmSerial.read());

while (gsmSerial.available())

{

  gsmSerial.read();
}

while (Serial.available())

{
```

```
    Serial.read();
  }
}

float *get_gps()

{

  gpsSerial.listen();

  Serial.println("INSIDE get_gps");

  while (1)
  {

    while (gpsSerial.available() > 0)

    {
      gps.encode(gpsSerial.read());
    }

    if (gps.location.isUpdated())

    {

      Serial.print("LAT=");
      Serial.println(gps.location.lat(), 6);

      Serial.print("LONG=");
      Serial.println(gps.location.lng(), 6);

      lattitude = gps.location.lat();

      longitude = gps.location.lng();

      break;
    }
  }

  a[0] = lattitude;

  a[1] = longitude;

  return a;
}

void SendMessage()
```

```
{
  Serial.println("\n==== Sending SMS...... ======");

  gsmSerial.println("AT+CMGF=1\r"); //Sets the GSM Module in Text Mode

  delay(1000); // Delay of 1000 milli seconds or 1 second

  gsmSerial.println("AT+CMGS=\"+917987460461\"\r"); // Replace x with mobile
number

  delay(1000);

  gsmSerial.println("I am in problem plz help!! \n"); // The SMS text you want
to send

  delay(1000);

  p = get_gps();

  gsmSerial.listen();

  Serial.print("Your position is : ");

  gsmSerial.print("Location is : ");

  Serial.print("LATTITUDE=");
  Serial.print(*p, 6);
  Serial.print("LONGITUDE=");
  Serial.print(*(p + 1), 6);

  gsmSerial.print("LATTITUDE=");
  gsmSerial.print(*p, 6); // The SMS text you want to send
  gsmSerial.print("\nLONGITUDE=");
  gsmSerial.print(*(p + 1), 6);

  gsmSerial.print("\nhttps://www.google.com/maps/search/?api=1&query=");
  gsmSerial.print(*p, 6);
  gsmSerial.print(",");         // The SMS text you want to send
  gsmSerial.print(*(p + 1), 6); // The SMS text you want to send

  delay(100);

  gsmSerial.println((char)26);
  Serial.println("\n+++++ Text Sent. +++++");
}

// End of Program
```

ADVANTAGES & LIMITATIONS

Advantage of blind stick is listed below:-

1. This system consists of different type of the sensor, which is used to measure the distance and alert the blind people.

2. It is simple to use and is affordable.

3. This system can navigate the location of the blind people when they find themselves in danger or some adverse situations.

4. Smart blind stick is robust and the stick is light, portable and reliable.

5. It consumes low power which makes it feasible to use.

Limitations of blind stick are listed below:-

1. Pits and bumps of the road cannot be detected using this device.

2. Smart stick is unfoldable.

3. As the sensor is sensitive, it should be handled in utmost care and prevented from contact with water.

APPLICATIONS

Following are the application of Smart Blind Stick

Facilitates the visually-impaired people through various user-friendly features such as water detection, Navigation, Obstacle alert and communication.

FUTURE IMPROVEMENTS

The certain modification of sensor and programming we can detect the small pits and dumps of road.

Water resistant can be eliminated

PROBLEM FORMULATION

While researching for this project, many problems were there. First problem was the unavailability of the materials required for the project. Because of this, much of the time was invested to search for them and talk to people outside Biratnagar. Secondly, there were some issues regarding interface. Other than that, resources could be utilized to make the research better.

CONCLUCSION

It is worth mentioning at this point that the aim of the of this study is design and implementation of a smart walking stick for the blind has been fully achieved. The smart stick as a basic platform for the coming generation of more adding devices to help the visually impaired to navigate safely both indoor and outdoor. It is effective and affordable. It leads the good result in detecting the obstacles on the path of the user in a range. This project offers low cost, reliable, portable, low power consumption and robust technology for navigation with obvious short response time. In this project, different types of sensors and other component with the light weight and the rain sensor is used to detect the water.

Biblography

https://www.theengineeringprojects.com/2018/06/introduction-to-arduino-nano.html

https://lastminuteengineers.com/arduino-sr04-ultrasonic-sensor-tutorial/

https://lastminuteengineers.com/rain-sensor-arduino-tutorial/

https://electronicsworkshops.com/2020/06/14/smart-blind-stick-using-gsm-module-gsm-moduleultrasonic-sensor-and-rain-sensor/