# **HARDWARE REQUIRED**

1. Arduino Nano – 1 pcs
2. Arduino Nano USB Cable – 1 pcs
3. IR Sensor module - 2 pcs
4. 5v Relay module - 1 pcs
5. 9v Battery - 1 pcs
6. Battery snapper - 1 pcs
7. Jumper wire (Female to Female) - As per requirement
8. Jumper wire (male to Female) - As per requirement
9. Mini Rocker switch – 1 pcs
10. LDR Sensor 1pcs
11. Bulb – As per requirement
12. Bulb Holder – As per requirement
13. Wire – As per requirement
14. Arduino IDE Software
15. Other materials

**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](http://www.arduino.org/software)). 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.

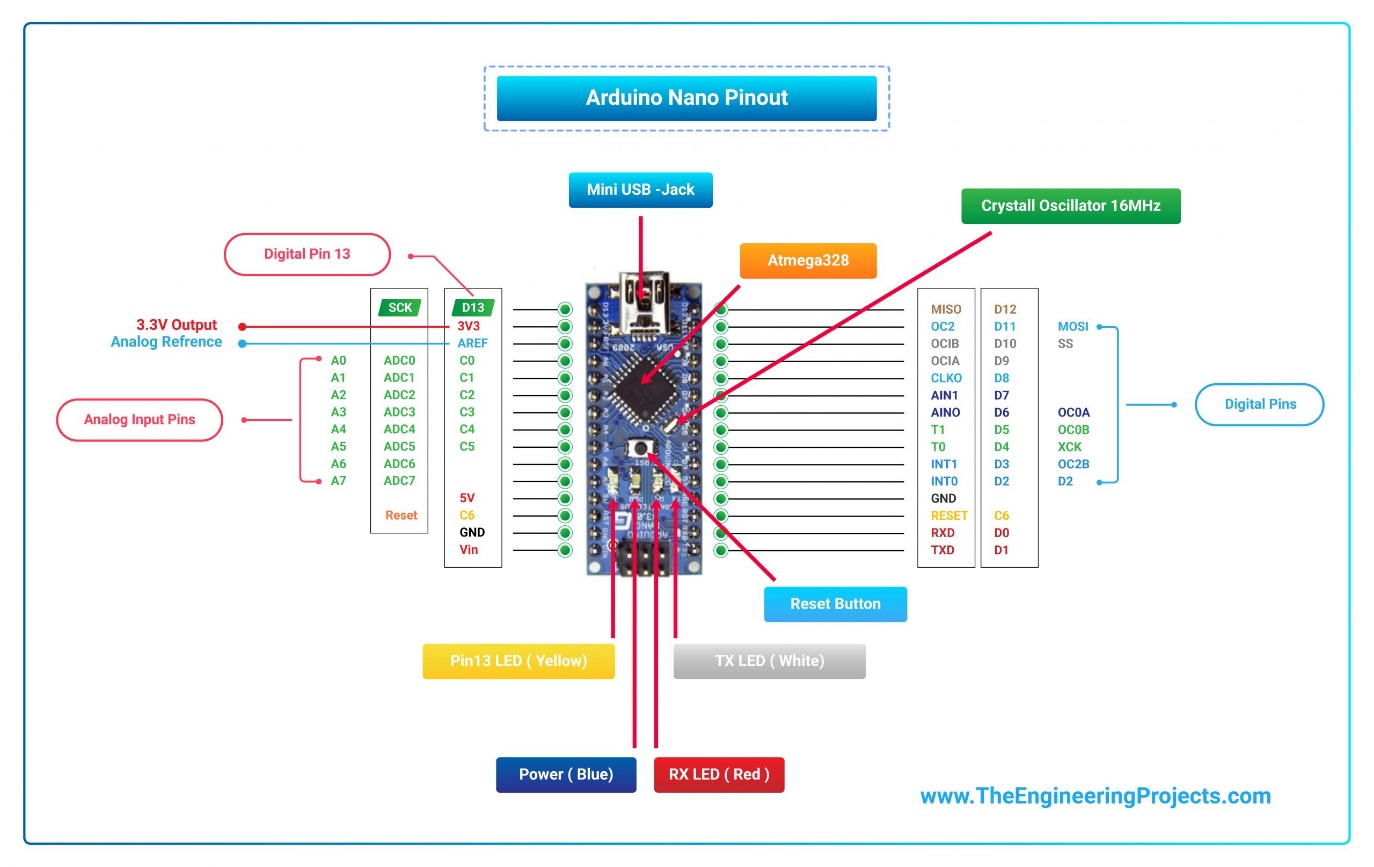
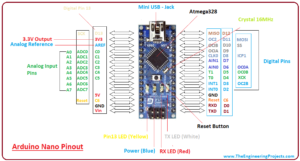
**Automatic (Software) Reset**

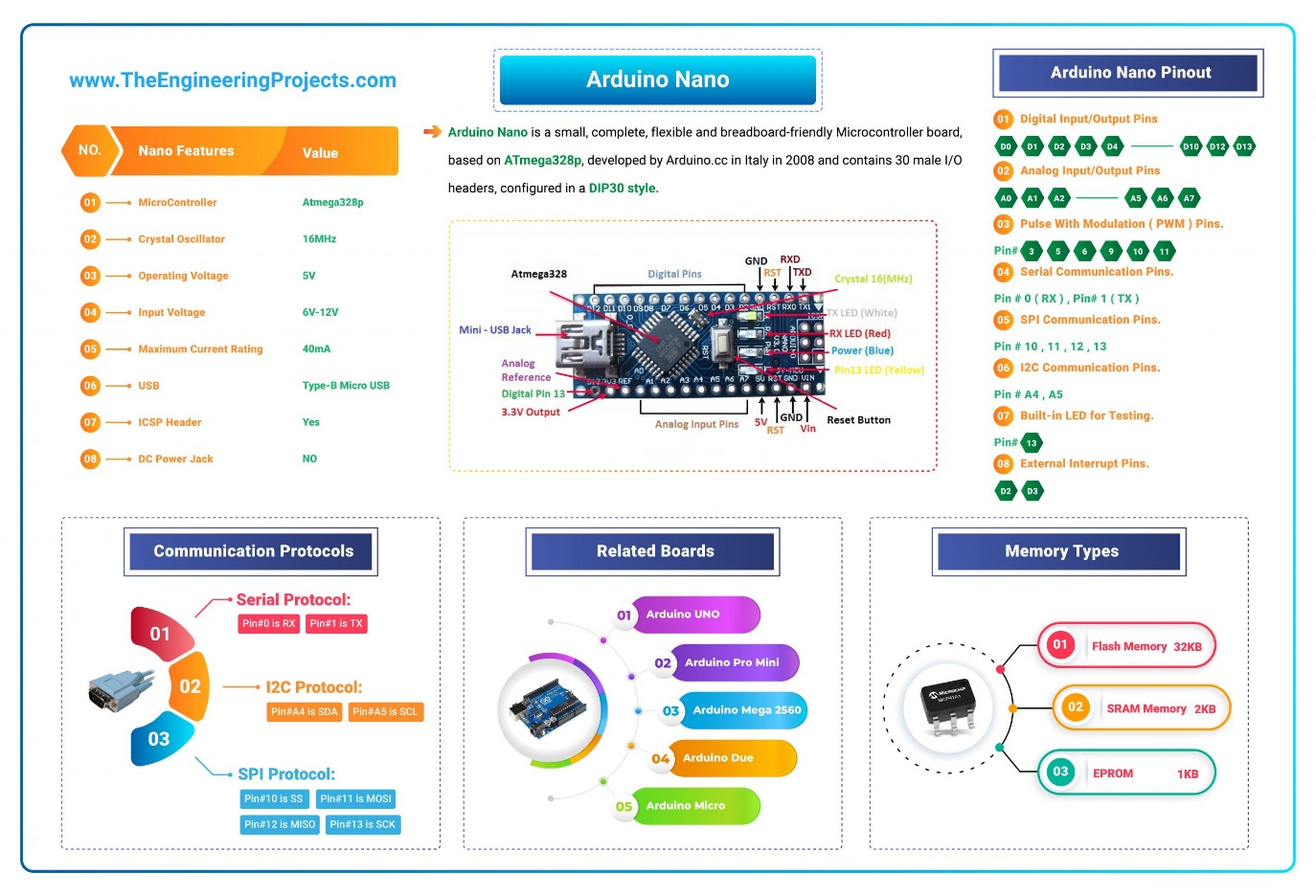
Rather then requiring a physical press of the reset button before an upload, the Arduino Nano is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the FT232RL is connected to the reset line of the ATmega328 via a 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](https://www.arduino.cc/) and based on [Atmega328p](https://www.theengineeringprojects.com/2017/08/introduction-to-atmega328.html) / [Atmega168](https://www.theengineeringprojects.com/2018/09/introduction-to-atmega168.html).

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.

[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/Arduino-Nano-Pinout-01-scaled.jpg)[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-13-1.png)

[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/Arduino-Nano-01-scaled.jpg)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:

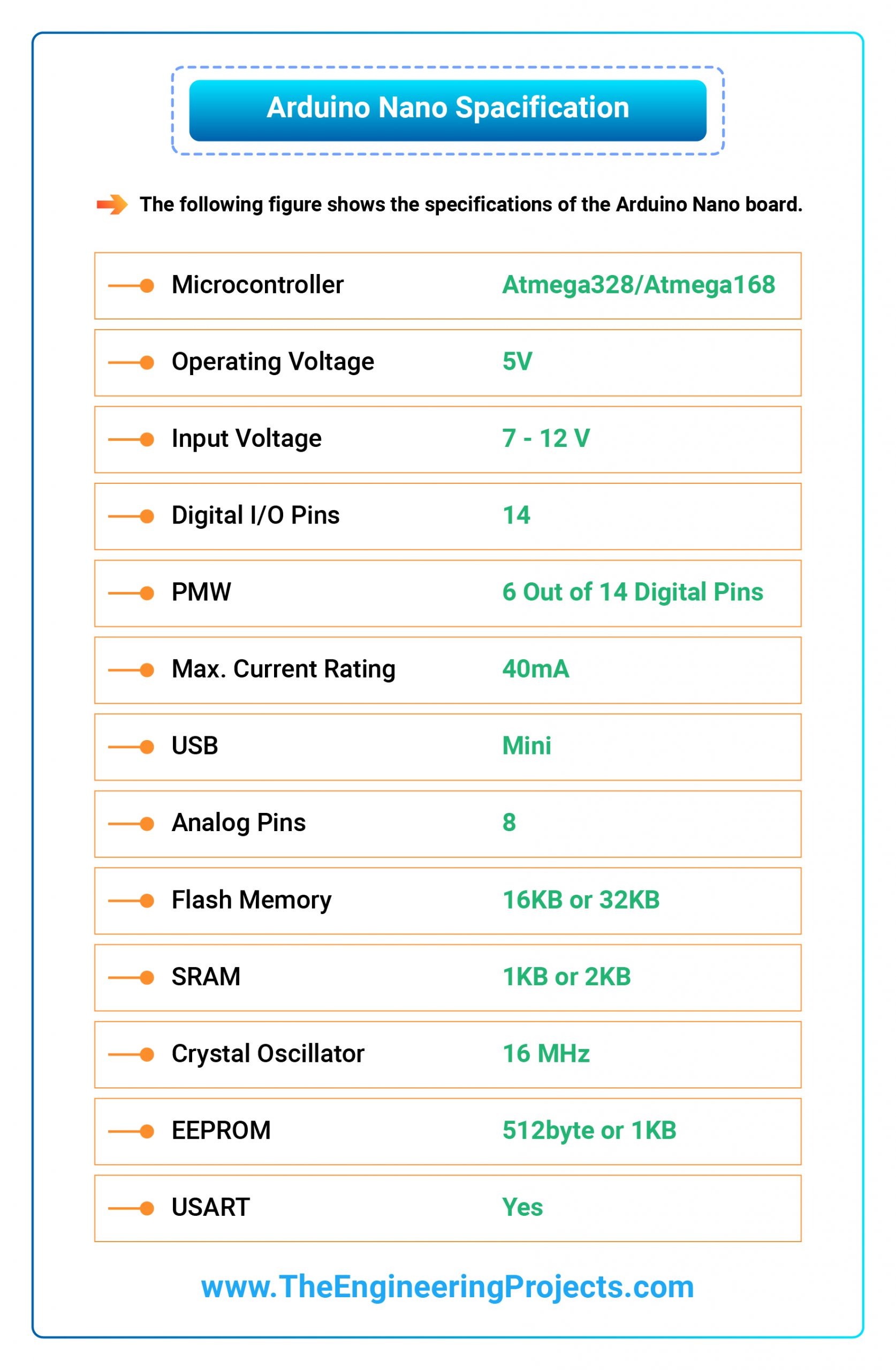
|  |  |  |
| --- | --- | --- |
| 6 | Serial Port | 1 (Pin#0 is RX, Pin#1 is TX). |
| 7 | I2C Port | 1 (Pin#A4 is SDA, Pin#A5 is SCL). |
| 8 | SPI Port | 1. (Pin#10 is SS, Pin#11 is MOSI, Pin#12 is MISO, Pin#13 is SCK). |

* Here’s the memory details present in Arduino Nano:

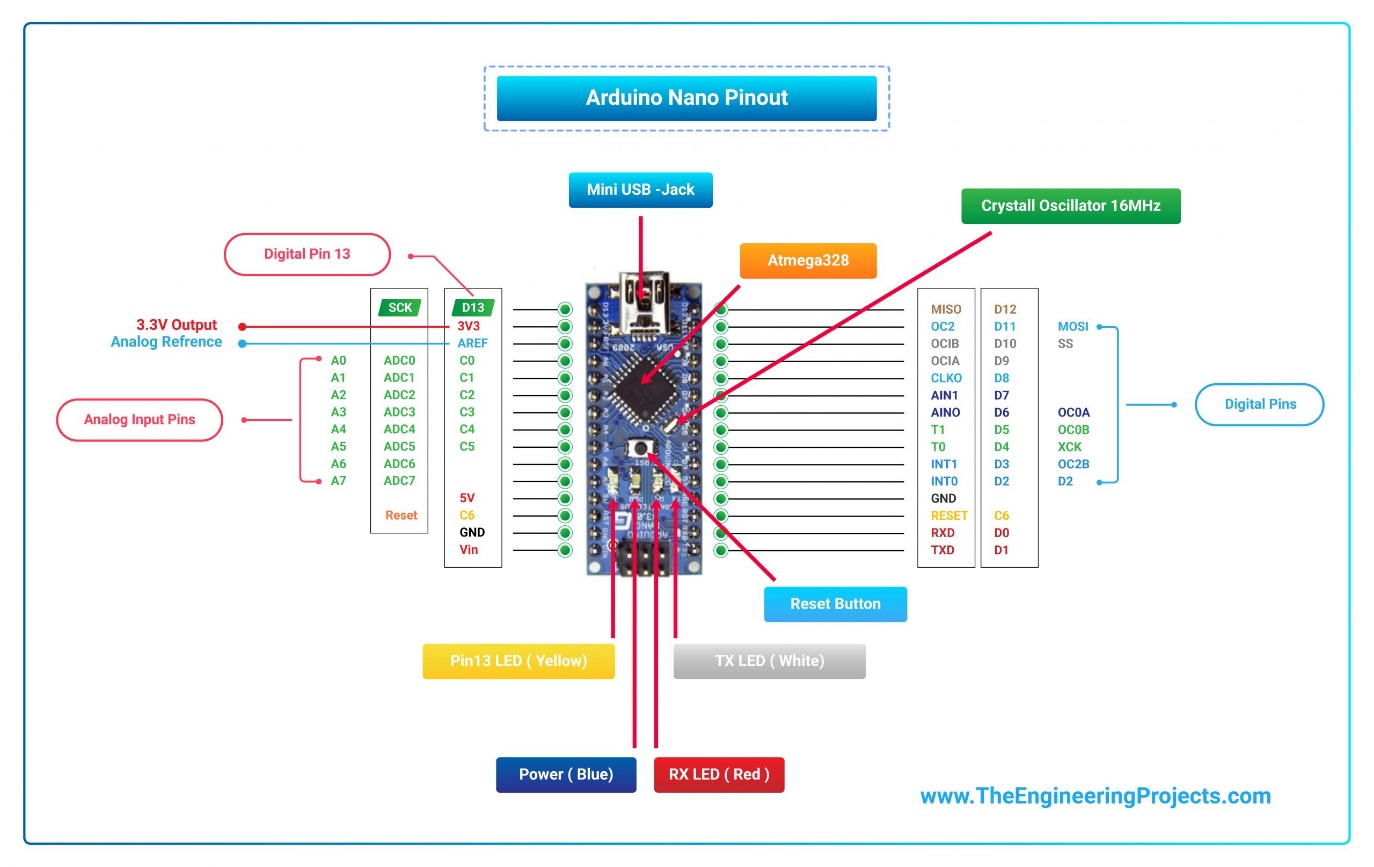
|  |  |  |
| --- | --- | --- |
| 7 | Flash Memory | 32KB |
| 8 | SRAM Memory | 2KB |
| 7 | EEPROM | 1KB |

## [Introduction to arduino nano, intro to arduino nano, pin diagram of arduino nano, applications of arduino nano, arduino nano pinout, difference between Arduino nano and Arduino uno, arduino nano specificationsIntroduction to arduino nano, intro to arduino nano, pin diagram of arduino nano, applications of arduino nano, arduino nano pinout, difference between Arduino nano and Arduino uno, arduino nano specifications](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-4-1.jpg)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.

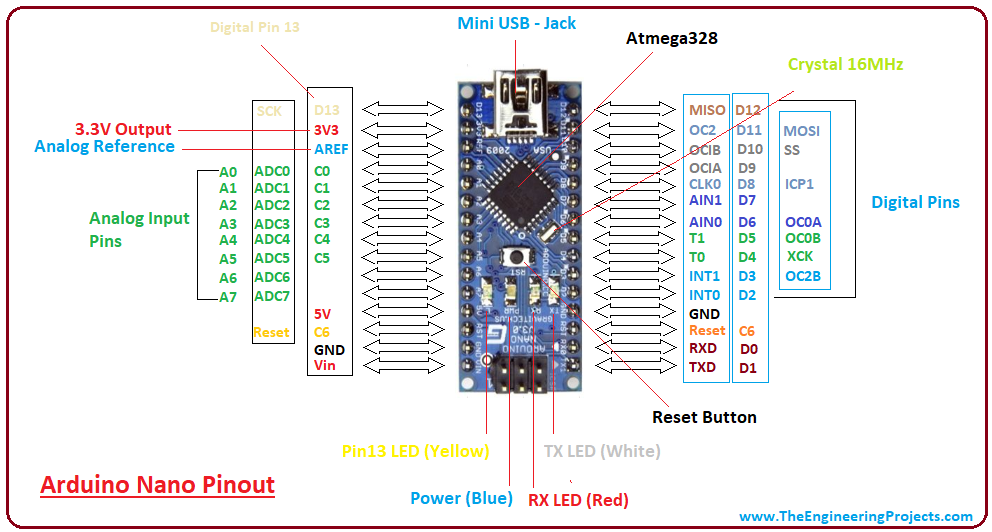
[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/01-Arduino-Nano-Specification-01-scaled.jpg)

* 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.

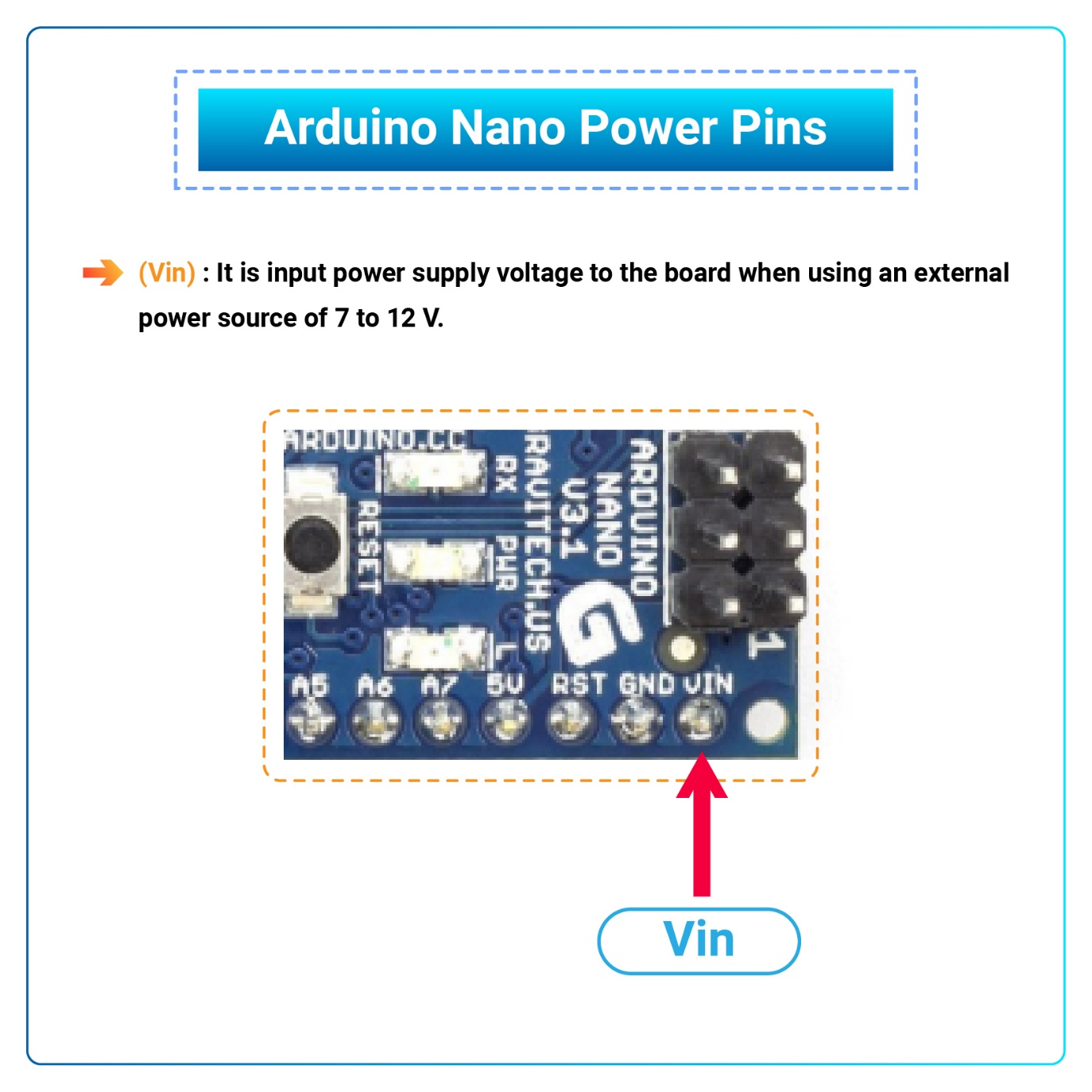
[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/Arduino-Nano-Pinout-01-scaled.jpg)Now, let’s have a look at Arduino Nano Pinout in detail:

## Arduino Nano Pinout

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

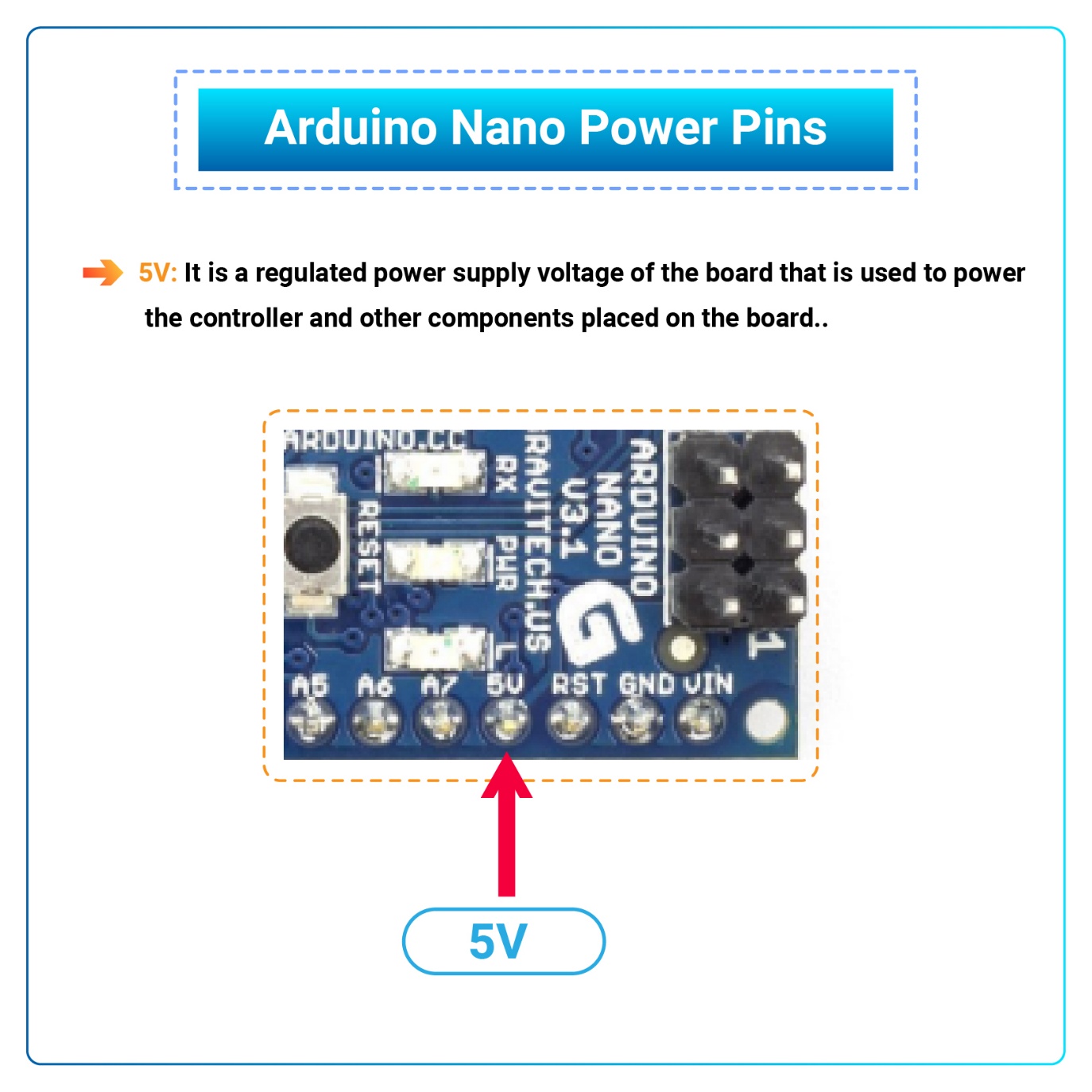
[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-13-1.png)

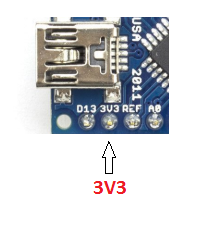
* Each pin on the Nano board comes with a specific function associated with it.
* We can see the analog pins that can be used as an analog to a digital converter, where A4 and A5 pins can also be used for I2C communication.
* Similarly, there are 14 digital pins, out of which 6 pins are used for generating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

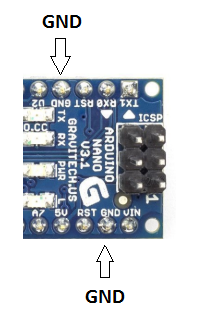
[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/Vin-01.jpg)Let’s have a look at the Arduino Nano Pinout in detail:

### Arduino Nano Power Pins

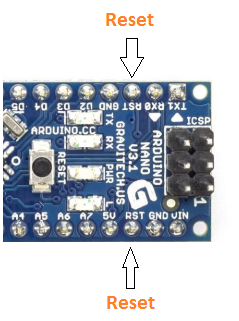
* **Vin:** It is input power supply voltage to the board when using an external power source of 7 to 12 V.
* **5V:** It is a regulated power supply voltage of the board that is used to power the controller and other components placed on the board.

[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/5V-01.jpg)

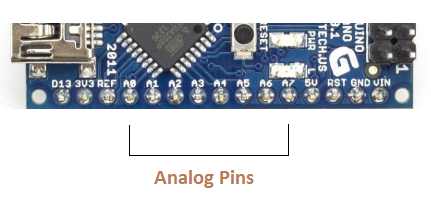
* [](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-8.png)**3V3:** This is a minimum voltage generated by the voltage regulator on the nano board.
* **GND Pin:** These are the ground pins on the board.
* There are multiple ground pins on the board that can be interfaced accordingly when more than one ground pin is required.

[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-9.png)

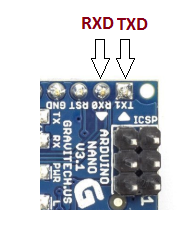
### Arduino Nano Function Pins

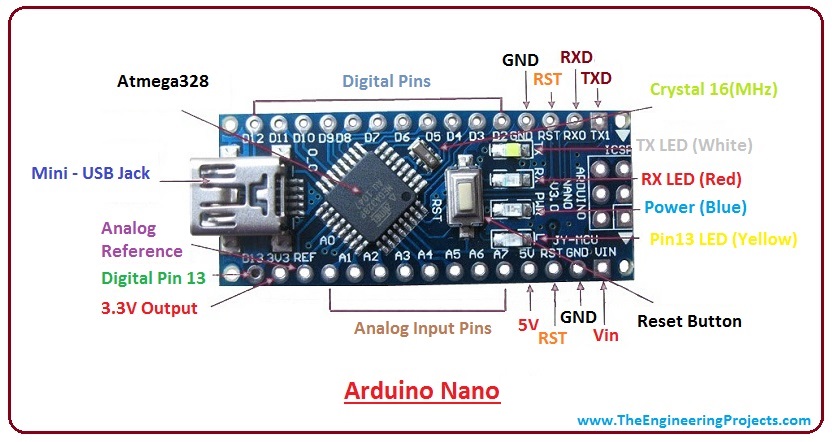
* [](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-10.png)**Reset Pin:** Arduino Nano has 2 reset pins incorporated on the board, making any of these **Reset pins LOW** will reset the microcontroller.
* **Pin#13:** A built-in LED is connected to pin#13 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.

### Arduino Nano I/O Pins

* [](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-11.png)**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.

### Nano Pinout for Communication Protocols

* **Serial Pins:** These pins are used for serial communication where:
  1. Pin#0 is RX used for receiving serial data.
  2. [](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-12.png)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.

[](https://www.theengineeringprojects.com/wp-content/uploads/2018/06/introduction-to-arduino-nano-4-1.jpg)

## 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](https://www.theengineeringprojects.com/2016/10/what-is-embedded-systems.html)
* Automation and Robotics
* Home Automation and Defense Systems
* Virtual Reality Applications

**IR Sensor Module**

**Hardware Overview**

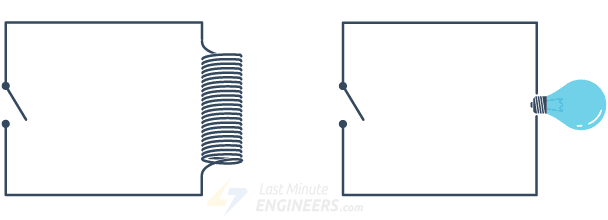
Sometimes you want your Arduino to control AC powered devices like lamps, fans or other household devices. But because the Arduino operates at 5 volts, it cannot directly control these higher voltage devices.

That’s where the relay module comes in. You can use a relay module to control the AC mains and Arduino to control the relay.

## How Do Relays Work?

A relay is an **electromagnetic switch** operated by a relatively small current that can control much larger current.

Here’s a simple animation illustrating how the relay uses one circuit to switch on another circuit.



Initially the first circuit is switched off and no current flows through it until something (either a sensor or switch closing) turns it on. The second circuit is also switched off.

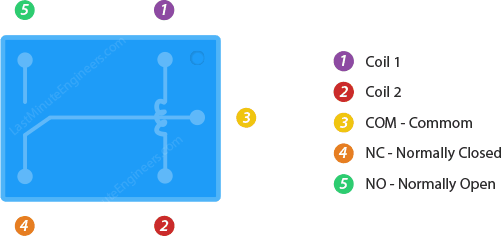
When a small current flows through the first circuit, it activates the electromagnet, which generates a magnetic field all around it.

The energized electromagnet attracts a contact in the second circuit toward it, closing the switch and allowing a much bigger current to flow through the second circuit.

When the current stops flowing, the contact goes back up to its original position, switching the second circuit off again.

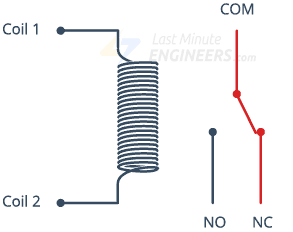
## Relay Basics

Typically the relay has 5 pins, three of them are high voltage terminals (NC, COM, and NO) that connect to the device you want to control.



The mains electricity enters the relay at the common (COM) terminal. While use of NC & NO terminals depends upon whether you want to turn the device ON or OFF.

Between the remaining two pins (coil1 and coil2), there is a coil that acts like an electromagnet.



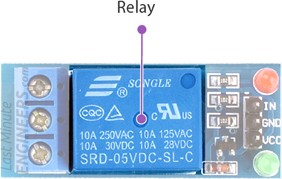
When current flows through the coil, the electromagnet becomes charged and moves the internal contacts of the switch. At that time the normally open (NO) terminal connects to the common (COM), and the normally closed (NC) terminal becomes disconnected.

When current stops flowing through the coil, the internal contact returns to its initial state i.e. the normally closed (NC) terminal connects to the common (COM), and the normally open (NO) terminal reopens.

This is known as a single pole, double throw switch (**SPDT**).

## One Channel Relay Module

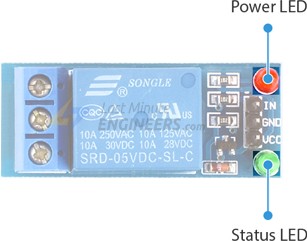
For this tutorial, we are going to use one channel relay module. However there are other modules with two, four and eight channels. You can choose the one that best suits your needs.



This module is designed for switching only a single high powered device from your Arduino. It has a relay rated up to 10A per channel at 250VAC or 30VDC.

### **LEDs**

There are two LEDs on the relay module indicating the position of the relay.



The Power LED will light up when the module is powered. The Status LED will light up when the relay is activated.

### **Output Terminal Block**

We have three channels of the relay broken out to blue screw pin terminals. The channels are labeled for their function: common (COM), normally closed (NC), and normally open (NO)

The names explain the state of the channel with relation to the switch at rest.



**COM** (Common): This is the pin you should connect to the signal (mains electricity in our case) you are planning to switch.

**NC** (Normally Closed): A normally closed configuration is used when you want to turn off the relay by default. In this configuration the relay is always closed and remains closed until you send a signal from the Arduino to the relay module to open the circuit.

**NO** (Normally Open): A normally open configuration works the other way in which the relay is always open until you send a signal from the Arduino to the relay module to close the circuit.

### **Control Pins**

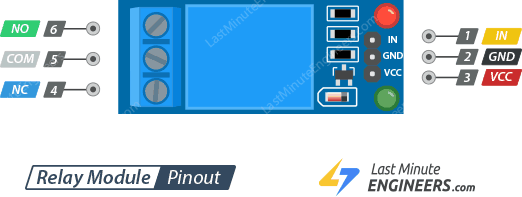
On the other side of the module, there are three pins – a Ground pin and a VCC pin to power the module and an input pin IN to control the relay.



The input pin is active low, meaning the relay will be activated when you pull the pin LOW and it will become inactive when you pull the pin HIGH.

## One Channel Relay Module Pinout

Let’s have a look at the pinout of one channel relay module.



### **Control Pins:**

IN pin is used to control the relay. It is an active low pin, meaning the relay will be activated when you pull the pin LOW and it will become inactive when you pull the pin HIGH.

GND is the ground connection.

VCC pin supplies power to the module.

### **Output Terminal:**

COM pin is connected to the signal you are planning to switch.

NC pin is connected to the COM pin by default, unless you send a signal from the Arduino to the relay module to break the connection.

NO pin is open by default, unless you send a signal from the Arduino to the relay module to make the connection.

**IR Sensor Module**

**Hardware Overview**

1. Signal -> shorted with VCC
2. GND -> GND
3. VCC -> 5V

**LDR**

**Hardware Overview**

A **Light Sensor** generates an output signal indicating the intensity of light by measuring the radiant energy that exists in a very narrow range of frequencies basically called “light”, and which ranges in frequency from “Infra-red” to “Visible” up to “Ultraviolet” light spectrum.

The light sensor is a passive devices that convert this “light energy” whether visible or in the infra-red parts of the spectrum into an electrical signal output. Light sensors are more commonly known as “Photoelectric Devices” or “Photo Sensors” because the convert light energy (photons) into electricity (electrons).

Photoelectric devices can be grouped into two main categories, those which generate electricity when illuminated, such as Photo-voltaics or Photo-emissives etc, and those which change their electrical properties in some way such as Photo-resistors or Photo-conductors. This leads to the following classification of devices.

* • Photo-emissive Cells – These are photodevices which release free electrons from a light sensitive material such as caesium when struck by a photon of sufficient energy. The amount of energy the photons have depends on the frequency of the light and the higher the frequency, the more energy the photons have converting light energy into electrical energy.
* • Photo-conductive Cells – These photodevices vary their electrical resistance when subjected to light. Photoconductivity results from light hitting a semiconductor material which controls the current flow through it. Thus, more light increase the current for a given applied voltage. The most common photoconductive material is Cadmium Sulphide used in LDR photocells.
* • Photo-voltaic Cells – These photodevices generate an emf in proportion to the radiant light energy received and is similar in effect to photoconductivity. Light energy falls on to two semiconductor materials sandwiched together creating a voltage of approximately 0.5V. The most common photovoltaic material is Selenium used in solar cells.
* • Photo-junction Devices – These photodevices are mainly true semiconductor devices such as the photodiode or phototransistor which use light to control the flow of electrons and holes across their PN-junction. Photojunction devices are specifically designed for detector application and light penetration with their spectral response tuned to the wavelength of incident light.

## The Photoconductive Cell

A **Photoconductive** light sensor does not produce electricity but simply changes its physical properties when subjected to light energy. The most common type of photoconductive device is the Photoresistor which changes its electrical resistance in response to changes in the light intensity.

Photoresistors are Semiconductor devices that use light energy to control the flow of electrons, and hence the current flowing through them. The commonly used Photoconductive Cell is called the **Light Dependent Resistor** or **LDR**.

### The Light Dependent Resistor



**Typical LDR**

As its name implies, the **Light Dependent Resistor** (LDR) is made from a piece of exposed semiconductor material such as cadmium sulphide that changes its electrical resistance from several thousand Ohms in the dark to only a few hundred Ohms when light falls upon it by creating hole-electron pairs in the material.

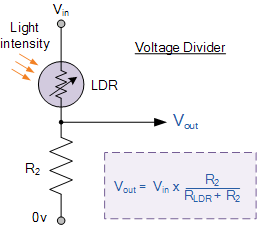
The net effect is an improvement in its conductivity with a decrease in resistance for an increase in illumination. Also, photoresistive cells have a long response time requiring many seconds to respond to a change in the light intensity.

Materials used as the semiconductor substrate include, lead sulphide (PbS), lead selenide (PbSe), indium antimonide (InSb) which detect light in the infra-red range with the most commonly used of all photoresistive light sensors being **Cadmium Sulphide** (Cds).

Cadmium sulphide is used in the manufacture of photoconductive cells because its spectral response curve closely matches that of the human eye and can even be controlled using a simple torch as a light source. Typically then, it has a peak sensitivity wavelength (λp) of about 560nm to 600nm in the visible spectral range.

### ldr photocellThe Light Dependent Resistor Cell

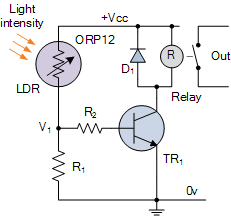
The most commonly used photoresistive light sensor is the **ORP12** Cadmium Sulphide photoconductive cell. This light dependent resistor has a spectral response of about 610nm in the yellow to orange region of light. The resistance of the cell when unilluminated (dark resistance) is very high at about 10MΩ’s which falls to about 100Ω’s when fully illuminated (lit resistance).

To increase the dark resistance and therefore reduce the dark current, the resistive path forms a zigzag pattern across the ceramic substrate. The CdS photocell is a very low cost device often used in auto dimming, darkness or twilight detection for turning the street lights “ON” and “OFF”, and for photographic exposure meter type applications.

Connecting a light dependant resistor in series with a standard resistor like this across a single DC supply voltage has one major advantage, a different voltage will appear at their junction for different levels of light.

The amount of voltage drop across series resistor, R2 is determined by the resistive value of the light dependant resistor, RLDR. This ability to generate different voltages produces a very handy circuit called a “Potential Divider” or **Voltage Divider Network**.

As we know, the current through a series circuit is common and as the LDR changes its resistive value due to the light intensity, the voltage present at VOUT will be determined by the voltage divider formula. An LDR’s resistance, RLDR can vary from about 100Ω in the sun light, to over 10MΩ in absolute darkness with this variation of resistance being converted into a voltage variation at VOUT as shown.

One simple use of a Light Dependent Resistor, is as a light sensitive switch as shown below.

**LDR Switch**

This basic light sensor circuit is of a relay output light activated switch. A potential divider circuit is formed between the photoresistor, LDR and the resistor R1. When no light is present ie in darkness, the resistance of the LDR is very high in the Megaohms (MΩ) range so zero base bias is applied to the transistor TR1 and the relay is de-energised or “OFF”.

As the light level increases the resistance of the LDR starts to decrease causing the base bias voltage at V1 to rise. At some point determined by the potential divider network formed with resistor R1, the base bias voltage is high enough to turn the transistor TR1 “ON” and thus activate the relay which in turn is used to control some external circuitry. As the light level falls back to darkness again the resistance of the LDR increases causing the base voltage of the transistor to decrease, turning the transistor and relay “OFF” at a fixed light level determined again by the potential divider network.

By replacing the fixed resistor R1 with a potentiometer VR1, the point at which the relay turns “ON” or “OFF” can be pre-set to a particular light level. This type of simple circuit shown above has a fairly low sensitivity and its switching point may not be consistent due to variations in either temperature or the supply voltage. A more sensitive precision light activated circuit can be easily made by incorporating the LDR into a “Wheatstone Bridge” arrangement and replacing the transistor with an Operational Amplifier as shown.

### light activated switchLight Level Sensing Circuit

In this basic dark sensing circuit, the light dependent resistor LDR1 and the potentiometer VR1 form one adjustable arm of a simple resistance bridge network, also known commonly as a Wheatstone bridge, while the two fixed resistors R1 and R2 form the other arm. Both sides of the bridge form potential divider networks across the supply voltage whose outputs V1 and V2 are connected to the non-inverting and inverting voltage inputs respectively of the operational amplifier.

The operational amplifier is configured as a Differential Amplifier also known as a voltage comparator with feedback whose output voltage condition is determined by the difference between the two input signals or voltages, V1 and V2. The resistor combination R1 and R2 form a fixed voltage reference at input V2, set by the ratio of the two resistors. The LDR – VR1 combination provides a variable voltage input V1 proportional to the light level being detected by the photoresistor.

As with the previous circuit the output from the operational amplifier is used to control a relay, which is protected by a free wheel diode, D1. When the light level sensed by the LDR and its output voltage falls below the reference voltage set at V2 the output from the op-amp changes state activating the relay and switching the connected load.

Likewise as the light level increases the output will switch back turning “OFF” the relay. The hysteresis of the two switching points is set by the feedback resistor Rf can be chosen to give any suitable voltage gain of the amplifier.

The operation of this type of light sensor circuit can also be reversed to switch the relay “ON” when the light level exceeds the reference voltage level and vice versa by reversing the positions of the light sensor LDR and the potentiometer VR1. The potentiometer can be used to “pre-set” the switching point of the differential amplifier to any particular light level making it ideal as a simple light sensor project circuit.

**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

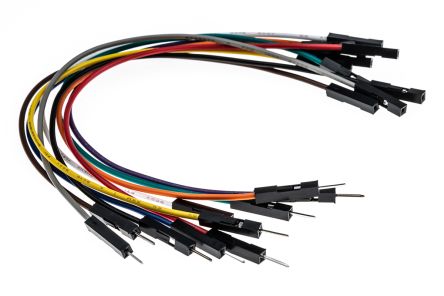
Connect D12 pin of Arduino Nano with the Positive pin of buzzer.

Connect GND pin of Arduino Nano with the Negative 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**

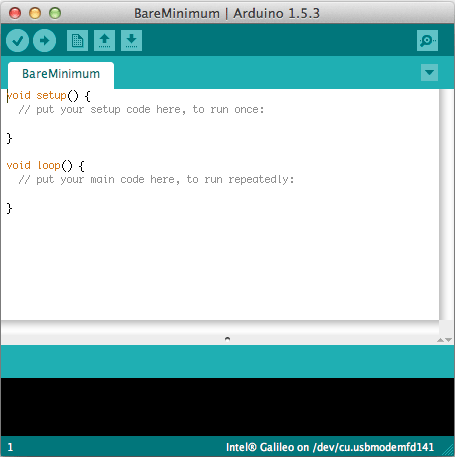
**Arduino IDE**

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**

****

## Arduino Project

# Laser Security System

Laser based Security System is a type of security and alarm system that uses laser light and a light sensor. A security system protects our homes, offices, banks, lockers etc. from intrusion and unauthorised access. There are different types of security systems available and laser based security system is an important and efficient type.

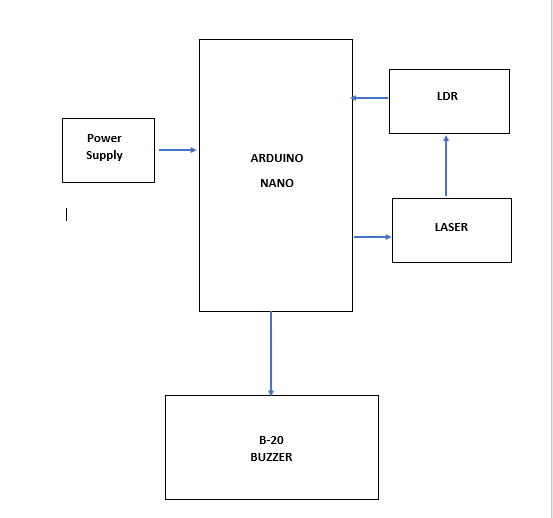
A Laser security system can acts as a standalone system, which makes some sound or noise when it detects any irregular activity, or can be part of a much bigger security and home automation system, which can send messages, call the owner etc.

In this project, we have designed a simple DIY lased based security system, which acts as a tripwire like security system and triggers an alarm when the laser in interrupted.

**OBJECTIVES**

* Laser Security System can be used in safety lockers in our homes, where even if the locker’s code is hacked, it acts as an additional layer of security.
* Apart from security systems, this laser based setup can also be used to check if pets or babies crossed a certain boundary.

**BLOCK DIAGRAM**

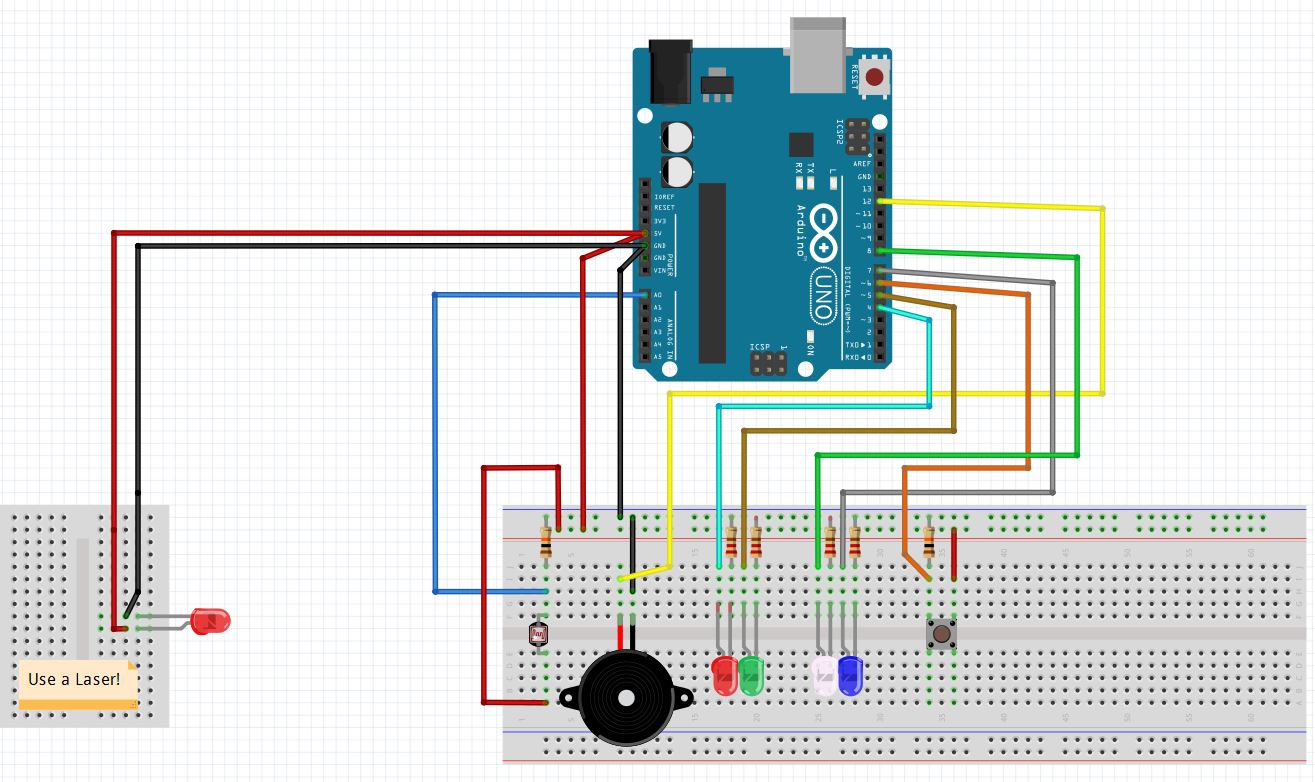
**Figure:- Block diagram Laser Security System**

**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**

**Circuit diagram description :-**

******

***Figure:-Circuit diagram***

The above figure shows the circuit diagram of Laser security System . We can see an **Arduino Uno** is used to control all the sensors however UNO can be replaced with Nano and we used

The complete board is powered by a 9V battery which is connected to VCC in of Arduino nano.

The **Laser module** is power by the 5v in VCC and Gnd is grounded and signal is connected to VCC.

The laser module focus laser on LDR and if disturbed then signal is noted by Arduino and it activates the buzzer.

The LDR one led is connected to A0 of Arduino and another leg is to positive.

All LEDS negative are grounded with resistance of 220 ohm and positive terminal is connected is as followed.

* Blue Positive to D7 pin of Arduino.
* White Positive to D8 pin of Arduino.
* Green Positive to D5 pin of Arduino.
* Red Positive to D4 pin of Arduino.

Switch is connected with D6 of Arduino.

The output of the board is given by the **Buzzer**which is connected to pin D12 the with the positive of buzzer and negative of buzzer is connected to GND.

## WORKING

## ****SOURCE CODE****

//Start of program

const int triggeredLED = 7;

const int triggeredLED2 = 8;

const int RedLED = 4;

const int GreenLED = 5;

const int inputPin = A0;

const int speakerPin = 12;

const int armButton = 6;

boolean isArmed = true;

boolean isTriggered = false;

int buttonVal = 0;

int prev\_buttonVal = 0;

int reading = 0;

int threshold = 0;

const int lowrange = 2000;

const int highrange = 4000;

void setup()

{

    pinMode(triggeredLED, OUTPUT);

    pinMode(triggeredLED2, OUTPUT);

    pinMode(RedLED, OUTPUT);

    pinMode(GreenLED, OUTPUT);

    pinMode(armButton, INPUT);

    digitalWrite(triggeredLED, HIGH);

    delay(500);

    digitalWrite(triggeredLED, LOW);

    calibrate();

    setArmedState();

}

void loop()

{

    reading = analogRead(inputPin);

    int buttonVal = digitalRead(armButton);

    if ((buttonVal == HIGH) && (prev\_buttonVal == LOW))

    {

        setArmedState();

        delay(500);

    }

    if ((isArmed) && (reading < threshold))

    {

        isTriggered = true;

    }

    if (isTriggered)

    {

        for (int i = lowrange; i <= highrange; i++)

        {

            tone(speakerPin, i, 250);

        }

        for (int i = highrange; i >= lowrange; i--)

        {

            tone(speakerPin, i, 250);

        }

        digitalWrite(triggeredLED, HIGH);

        delay(50);

        digitalWrite(triggeredLED, LOW);

        delay(50);

        digitalWrite(triggeredLED2, HIGH);

        delay(50);

        digitalWrite(triggeredLED2, LOW);

        delay(50);

    }

    delay(20);

}

void setArmedState()

{

    if (isArmed)

    {

        digitalWrite(GreenLED, HIGH);

        digitalWrite(RedLED, LOW);

        isTriggered = false;

        isArmed = false;

    }

    else

    {

        digitalWrite(GreenLED, LOW);

        digitalWrite(RedLED, HIGH);

        tone(speakerPin, 220, 125);

        delay(200);

        tone(speakerPin, 196, 250);

        isArmed = true;

    }

}

void calibrate()

{

    int sample = 0;

    int baseline = 0;

    const int min\_diff = 200;

    const int sensitivity = 50;

    int success\_count = 0;

    digitalWrite(RedLED, LOW);

    digitalWrite(GreenLED, LOW);

    for (int i = 0; i < 10; i++)

    {

        sample += analogRead(inputPin);

        digitalWrite(GreenLED, HIGH);

        delay(50);

        digitalWrite(GreenLED, LOW);

        delay(50);

    }

    do

    {

        sample = analogRead(inputPin);

        if (sample > baseline + min\_diff)

        {

            success\_count++;

            threshold += sample;

            digitalWrite(GreenLED, HIGH);

            delay(100);

            digitalWrite(GreenLED, LOW);

            delay(100);

        }

        else

        {

            success\_count = 0;

            threshold = 0;

        }

    } while (success\_count < 3);

    threshold = (threshold / 3) - sensitivity;

    tone(speakerPin, 196, 250);

    delay(200);

    tone(speakerPin, 220, 125);

}

//End of Program

**ADVANTAGES & LIMITATIONS**

### Advantages

* The circuit, construction and setup for the Laser Security System is very simple.
* If used with a battery, the laser security system can work even when there is a power outage.

### Disadvantages

* The laser security system works only if the laser is obstructed. If the intruder passes without obstructing the laser, it is considered as a failure.
* In order to secure a larger area, we need more lasers and corresponding sensors.

**APPLICATIONS**

* Laser Security System can be used in safety lockers in our homes, where even if the locker’s code is hacked, it acts as an additional layer of security.
* Apart from security systems, this laser based setup can also be used to check if pets or babies crossed a certain boundary.

**FUTURE IMPROVEMENTS**

 The certain modification of sensor and programming we can detect the high speed object.

**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 college. Secondly, there were some issues regarding interface. Other than that, resources could be utilized to make the research better.

**CONCLUCSION**