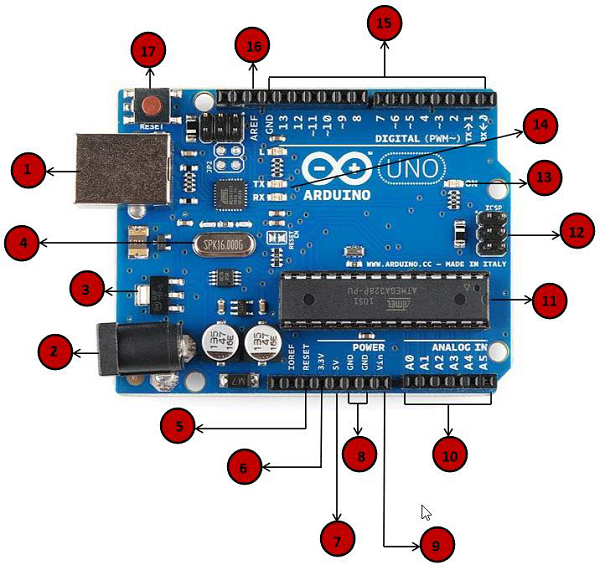
ARDUINO

Arduino is an open-source electronics platform based on easy-to-use hardware and software.

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

### Why Arduino?

* Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50.
* Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
* Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
* Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the [breadboard version of the module](https://www.arduino.cc/en/Main/Standalone) in order to understand how it works and save money.



|  |  |
| --- | --- |
| Power USB | **Power USB**  Arduino board can be powered by using the USB cable from your computer. All you need to do is connect the USB cable to the USB connection (1). |
| Barrel Jack | **Power (Barrel Jack)**  Arduino boards can be powered directly from the AC mains power supply by connecting it to the Barrel Jack (2). |
| Voltage Regulator | **Voltage Regulator**  The function of the voltage regulator is to control the voltage given to the Arduino board and stabilize the DC voltages used by the processor and other elements. |
| Crystal Oscillator | **Crystal Oscillator**  The crystal oscillator helps Arduino in dealing with time issues. How does Arduino calculate time? The answer is, by using the crystal oscillator. The number printed on top of the Arduino crystal is 16.000H9H. It tells us that the frequency is 16,000,000 Hertz or 16 MHz. |
| Arduino Reset | **Arduino Reset**  You can reset your Arduino board, i.e., start your program from the beginning. You can reset the UNO board in two ways. First, by using the reset button (17) on the board. Second, you can connect an external reset button to the Arduino pin labelled RESET (5). |
| Pins | **Pins (3.3, 5, GND, Vin)**   * 3.3V (6) − Supply 3.3 output volt * 5V (7) − Supply 5 output volt * Most of the components used with Arduino board works fine with 3.3 volt and 5 volt. * GND (8)(Ground) − There are several GND pins on the Arduino, any of which can be used to ground your circuit. * Vin (9) − This pin also can be used to power the Arduino board from an external power source, like AC mains power supply. |
| Analog pins | **Analog pins**  The Arduino UNO board has six analog input pins A0 through A5. These pins can read the signal from an analog sensor like the humidity sensor or temperature sensor and convert it into a digital value that can be read by the microprocessor. |
| Main microcontroller | **Main microcontroller**  Each Arduino board has its own microcontroller (11). You can assume it as the brain of your board. The main IC (integrated circuit) on the Arduino is slightly different from board to board. The microcontrollers are usually of the ATMEL Company. You must know what IC your board has before loading up a new program from the Arduino IDE. This information is available on the top of the IC. For more details about the IC construction and functions, you can refer to the data sheet. |
| ICSP pin | **ICSP pin**  Mostly, ICSP (12) is an AVR, a tiny programming header for the Arduino consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often referred to as an SPI (Serial Peripheral Interface), which could be considered as an "expansion" of the output. Actually, you are slaving the output device to the master of the SPI bus. |
| Power LED indicator | **Power LED indicator**  This LED should light up when you plug your Arduino into a power source to indicate that your board is powered up correctly. If this light does not turn on, then there is something wrong with the connection. |
| TX and RX LEDs | **TX and RX LEDs**  On your board, you will find two labels: TX (transmit) and RX (receive). They appear in two places on the Arduino UNO board. First, at the digital pins 0 and 1, to indicate the pins responsible for serial communication. Second, the TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process. |
| Digital I/O | **Digital I/O**  The Arduino UNO board has 14 digital I/O pins (15) (of which 6 provide PWM (Pulse Width Modulation) output. These pins can be configured to work as input digital pins to read logic values (0 or 1) or as digital output pins to drive different modules like LEDs, relays, etc. The pins labeled “~” can be used to generate PWM. |
| AREF | **AREF**  AREF stands for Analog Reference. It is sometimes, used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins. |

Local Variables

Variables that are declared inside a function or block are local variables. They can be used only by the statements that are inside that function or block of code. Local variables are not known to function outside their own. Following is the example using local variables –

Void setup () {

}

Void loop () {

int x,y;

int z; Local variable declaration

x = 0;

y = 0; actual initialization

z = 10;

}

### Global Variables

Global variables are defined outside of all the functions, usually at the top of the program. The global variables will hold their value throughout the life-time of your program.

A global variable can be accessed by any function. That is, a global variable is available for use throughout your entire program after its declaration.

The following example uses global and local variables –

Int T, S;

float c = 0; Global variable declaration

Void setup () {

}

Void loop () {

int x, y;

int z; Local variable declaration

x = 0;

y = 0; actual initialization

z = 10;

}

## **Structure:**

* Setup( ) function
* Loop( ) function

## **Arduino Code:**

void setup() {

char my\_str[6]; // an array big enough for a 5 character string

Serial.begin(9600);

my\_str[0] = 'H'; // the string consists of 5 characters

my\_str[1] = 'e';

my\_str[2] = 'l';

my\_str[3] = 'l';

my\_str[4] = 'o';

my\_str[5] = 0; // 6th array element is a null terminator

Serial.println(my\_str);

}

void loop() {

}

int n[10]; // n is an array of 10 integers

void setup () {

}

void loop () {

for (int i=0; i<10; ++i) // initialize elements of array n to 0 {

n[ i ] = 0; // set element at location i to 0

Serial.print (i);

Serial.print (‘\r’);

}

for (int j = 0; j < 10; ++j) // output each array element's value {

Serial.print (n[j]);

Serial.print (‘\r’);

}

}

int button = 5; // button connected to pin 5

int LED = 6; // LED connected to pin 6

void setup () {

pinMode(button , INPUT\_PULLUP);

// set the digital pin as input with pull-up resistor

pinMode(button , OUTPUT); // set the digital pin as output

}

void setup () {

If (digitalRead(button ) == LOW) // if button pressed {

digitalWrite(LED,HIGH); // turn on led

delay(500); // delay for 500 ms

digitalWrite(LED,LOW); // turn off led

delay(500); // delay for 500 ms

}

}