Module 05

Introduction to Arduino Programming

Introduction

What is Arduino?

Arduino is an **open-source electronics prototype 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. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the **Arduino programming language** (based on Wiring), and the **Arduino Software (Integrated Development Environment),** based on Processing.

- Open source Based electronic Programmable(micro controller) and softer IDE
- Accept analog and digital signals and give desirable output
- No Extra hardware required to load a program in to the controller board

The Arduino board consists of sets of **analog and digital I/O** (Input / Output) pins, which are further interfaced to breadboard, **expansion boards**, and other circuits. Such boards feature the model, **Universal Serial Bus (USB)**, and serial communication interfaces, which are used for loading programs from the computers.

Why Arduino?

Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux.

Important Advantages of Arduino

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows. IDE Software operate on any operating system.
- **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. Don't need to external programmer (Burner)
- **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 in order to understand how it works and save money.
- Programming ease, Easy to learn
- Huge community

Features

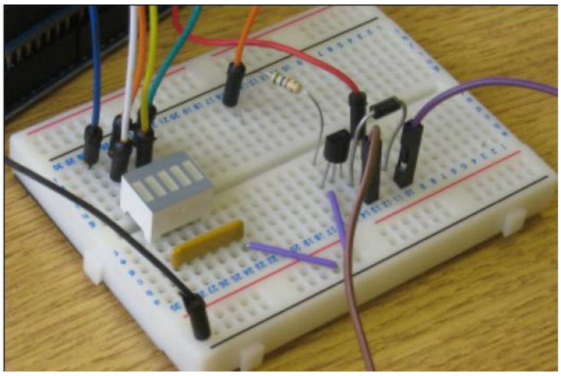
The features of Arduino are listed below:

- Arduino programming is a simplified version of C++, which makes the learning process easy.
- The Arduino IDE is used to control the functions of boards. It further sends the set of specifications to the microcontroller.
- Arduino does not need an extra board or piece to load new code.
- Arduino can read analog and digital input signals.
- The hardware and software platform is easy to use and implement.

Prototype

Bread boards

Most of the prototyping can be done on what are called solderless breadboards. They enable you to build components together into a circuit with just a push-fit connection, which also means you can experiment with different options quickly and easily.



Components of Arduino

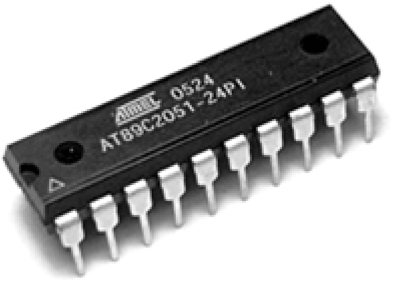
1. Sensors and Actuators

▪ **Sensors**: Sensors are the ways of getting information into your device, finding out things about your surroundings.

▪ **Actuators**: Actuators are the outputs for the device—the motors, lights, and so on, which let your device do something to the outside world.

2. Microcontroller

The most essential part of the Arduino is the Microcontroller.



- Microcontroller is small and low power computer. Most of the microcontrollers have a RAM (Random Access Memory), CPU (Central Processing Unit), and a memory storage like other computer systems.
- It has very small memory of 2KB (two Kilobytes). Due to less memory, some microcontrollers are capable of running only one program at a time.
- It is a single chip that includes memory, Input/Output (I/O) peripherals, and a processor.
- The GPIO (General Purpose Input Output) pins present on the chip help us to control other electronics or circuitry from the program.

3. Arduino IDE

The IDE makes the traditional projects even easier and simpler. The USB cable is used to load the program or sketch on the specific Arduino board.

The IDE application is suitable for Windows, Mac OS X, and Linux. It supports the programming language C and C++. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software.

4. Arduino Boards

There are variety of Arduino board used for different purposes. The board varies in I/O pins, size, etc. The various components present on the Arduino boards are Microcontroller, Digital Input/Output pins, USB Interface and Connector, Analog Pins, Reset Button, Power button, LED's, Crystal Oscillator, and Voltage Regulator. Some components may differ depending on the type of board.

the popular Arduino boards.

- Arduino UNO
- Arduino Nano
- Arduino Mega
- Arduino Due
- Arduino Bluetooth

5. Shields

Arduino shields are the boards, which are plugged over the Arduino board to expand its functionalities. There are different varieties of shields used for various tasks, such as Arduino motor shields, Arduino communication shields, etc.

The advantages of using Arduino shields are listed below:

- It adds new functionalities to the Arduino projects.
- The shields can be attached and detached easily from the Arduino board. It does not require any complex wiring.
- It is easy to connect the shields by mounting them over the Arduino board.
- The hardware components on the shields can be easily implemented.

Types of Shields

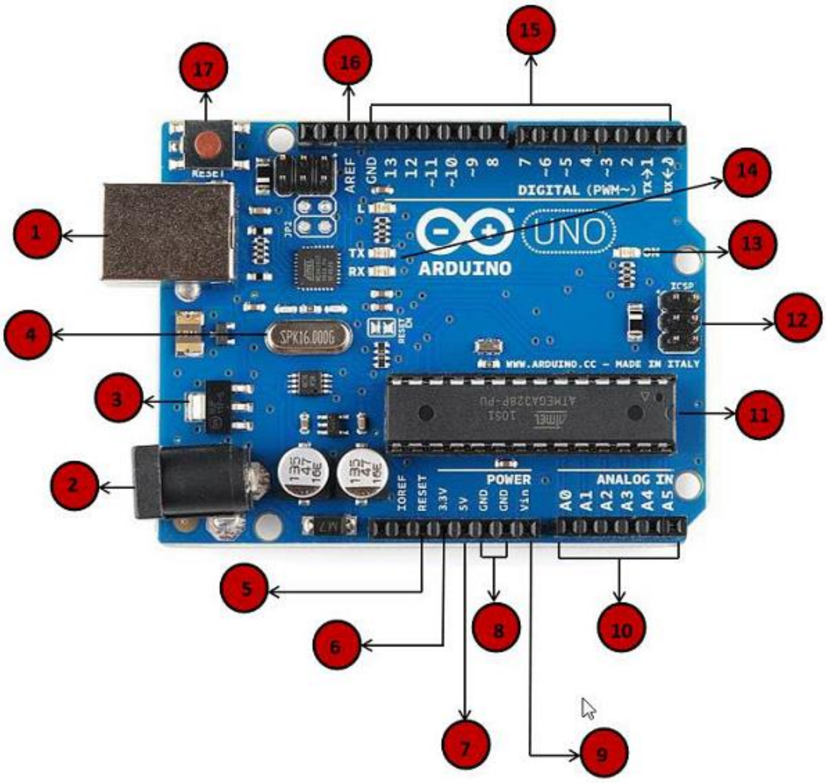
- Ethernet shield
- Xbee Shield
- Proto shield
- Relay shield
- Motor shield
- LCD shield
- Bluetooth shield
- Capacitive Touchpad Shield

Arduino Boards

Arduino is an easy-to-use open platform to create electronics projects. Arduino boards play a vital role in creating different projects. It makes electronics accessible to non-engineers, hobbyists, etc.

The various components present on the Arduino boards are Microcontroller, Digital Input/output pins, USB Interface and Connector, Analog Pins, Reset Button, Power button, LED's, Crystal Oscillator, and Voltage Regulator. Some components may differ depending on the type of board.

The most standard and popular board used over time is Arduino UNO.



Components

Power USB Arduino board can be powered by using the USB cable from your computer.

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 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.

Arduino Reset You can reset your Arduino board, i.e., start your program from the beginning.

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 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 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.

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.

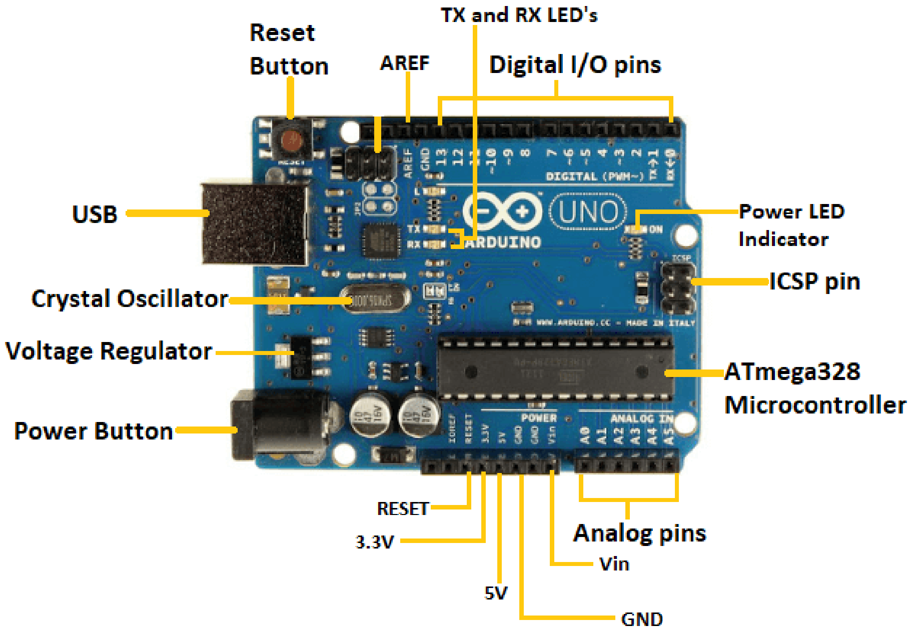
TX and RX LEDs The TX and RX led (13). The TX led flashes with different speed while sending the serial data. The speed of flashing depends on the baud rate used by the board. RX flashes during the receiving process.

Digital I/O 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.

Arduino UNO

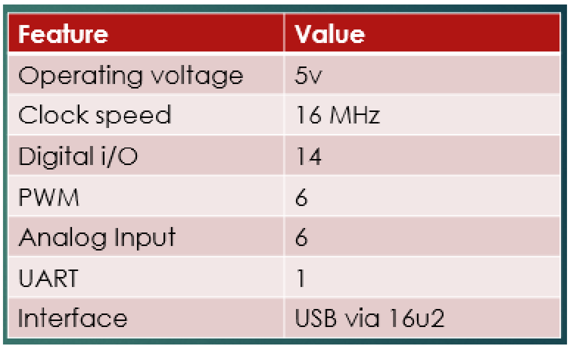
The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It was named as UNO to label the first release of Arduino Software. It was also the first USB board released by Arduino. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms.



Components on Board

- Power supply : USB or Power barrel jack
- Voltage Regulator
- LED Power indicator
- Tx-RX LED Indicator
- Output power, Ground
- Analog input Pins
- Digital Input Pins



Feature	Value
Operating voltage	5v
Clock speed	16 MHz
Digital i/O	14
PWM	6
Analog Input	6
UART	1
Interface	USB via 16u2

Technical Specifications of Arduino UNO

The technical specifications of the Arduino UNO are listed below:

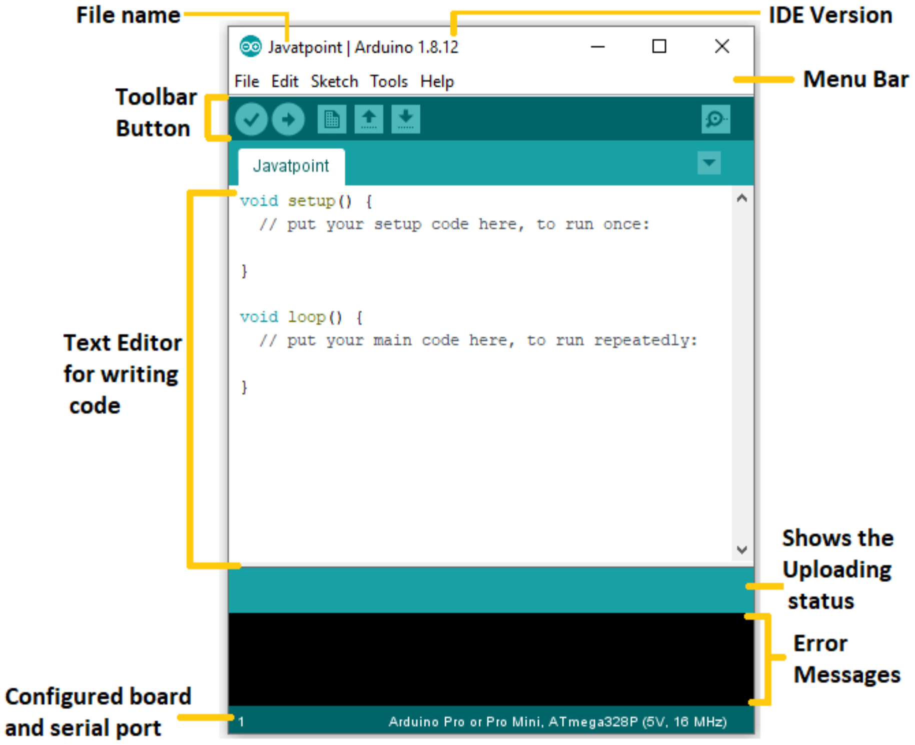
- There are 20 Input/Output pins present on the Arduino UNO board. These 20 pis include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable pins.
- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz.

- It also has a Arduino integrated WiFi module. Such Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V.
- Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB.

Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment.

- Arduino IDE is a software that is used to Program the Arduino board
- Based on variations of C and C++ Programming Language
- It can be freely downloaded from Arduinos official website into a PC


File name
IDE Version
Javatpoint | Arduino 1.8.12
File Edit Sketch Tools Help
Menu Bar
Toolbar Button
Javatpoint
Text Editor for writing code
Shows the Uploading status
Error Messages
Configured board and serial port
1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino Pro or Pro Mini, ATmega328P (5V, 16 MHz)

The program or code written in the Arduino IDE is often called as **sketching.** We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

We do not require any additional hardware to upload our sketch using the Arduino **Bootloader**. A Bootloader is defined as a small program, which is loaded in the microcontroller present on the board. The LED will blink on PIN 13.

BASIC set up

- Power the board by connecting it in to a PC via USB cable
- Launch the Arduino IDE
- Set the board type and the port for the board
- Tools- Board- select your board
- Tools – Port- select – your –port

Arduino Coding Basics

The initial step to start with Arduino is the IDE download and installation.

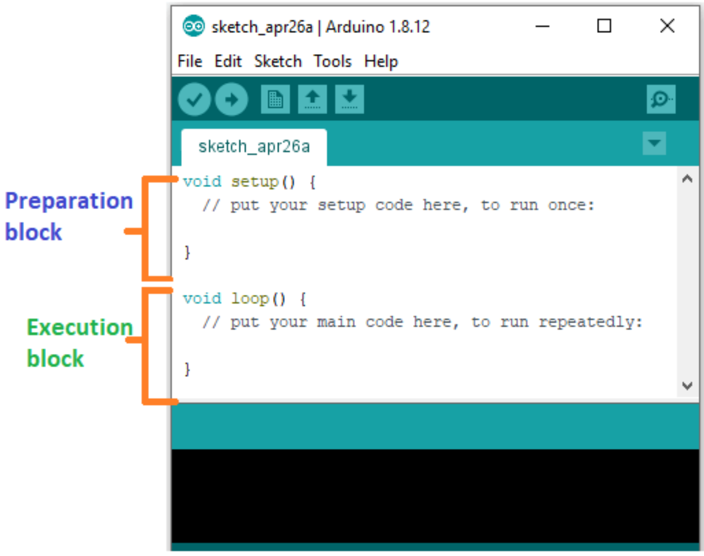
Brackets There are two types of brackets used in the Arduino coding, which are listed below

- **Parentheses ( )** The parentheses brackets are the group of the arguments, such as method, function, or a code statement. These are also used to group the math equations.
- **Curly Brackets { }** The statements in the code are enclosed in the curly brackets. We always require closed curly brackets to match the open curly bracket in the code or sketch.

Line Comment There are two types of line comments, which are listed below:

- // Single line comment
- / * Multi - line comment */

Coding Screen



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

What is Setup? What type of code is written in the setup block?

It contains an initial part of the code to be executed. The pin modes, libraries, variables, etc., are initialized in the setup section. It is executed only once during the uploading of the program and after reset or power up of the Arduino board.

Zero setup () resides at the top of each sketch. As soon as the program starts running, the code inside the curly bracket is executed in the setup and it executes only once.

What is Loop? What type of code is written in the Loop block?

The loop contains statements that are executed repeatedly. The section of code inside the curly brackets is repeated depending on the value of variables.

Bare minimum code

```
void setup() {
 // put your setup code here, to run once: }
```

setup : It is called only when the Arduino is powered on or reset. It is used to initialize variables and pin modes

void loop() {
 // put your main code here, to run repeatedly: }

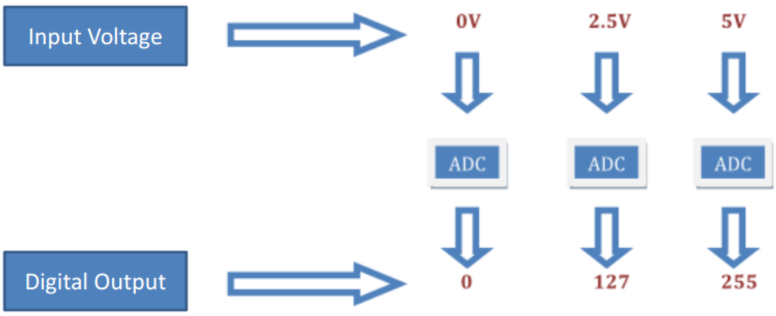
loop : The loop functions runs continuously till the device is powered off. The main logic of the code goes here. Similar to while (1) for micro-controller programming.

PinMode

- A pin on Arduino can be set as input or output by using pinMode function.
- pinMode(13, OUTPUT); // sets pin 13 as output pin
- pinMode(13, INPUT); // sets pin 13 as input pin

Analog to Digital Conversion

An analogue-to-digital converter (ADC) lets you measure varying voltages. Microcontrollers often have a number of these converters built in. They will convert the voltage level between 0V and a predefined maximum (often the same 5V or 3.3V the processor is running at, but sometimes a fixed value such as 1V) into a number, depending on the accuracy of the ADC. The Arduino has 10-bit ADCs, which by default measure voltages between 0 and 5V. A voltage of 0 will give a reading of 0; a voltage of 5V would read 1023 (the maximum value that can be stored in a 10-bits); and voltages in between result in readings relative to the voltage. 1V would map to 205; a reading of 512 would mean the voltage was 2.5V; and so on.



- It is continuous range of voltage values (not just 0 or 5V)
- Because our microcontroller only understands digital.

Reading/Writing Analog Values

analogRead(A0); // used to read the analog value from the pin A0

analogWrite(2,128);

DAC, or digital-to-analogue converter

DACs let you generate varying voltages from a digital value but are less common as a standard feature of microcontrollers. This is due to a technique called pulse-width modulation (PWM), which gives an approximation to a DAC by rapidly turning a digital signal on and off so that the average value is the level you desire.

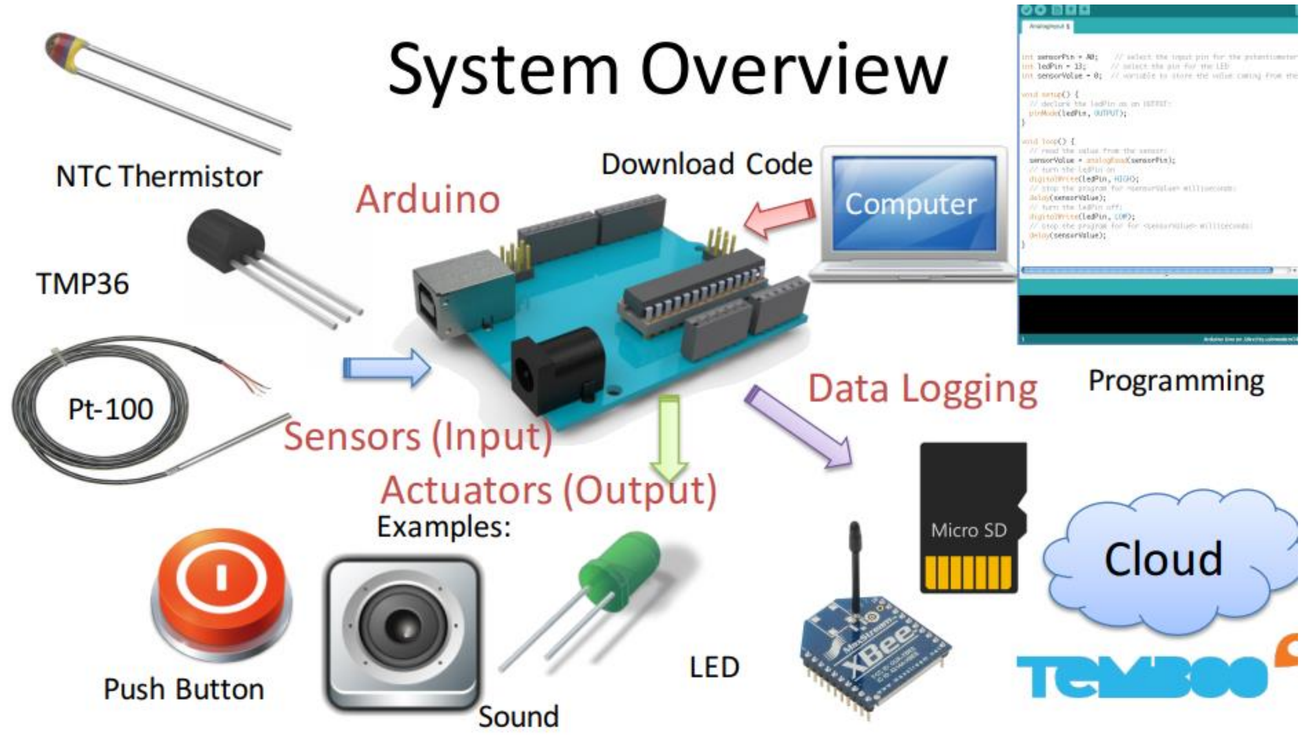
Reading/writing digital values

digitalWrite(13, LOW); // Makes the output voltage on pin 13 , 0V

digitalWrite(13, HIGH); // Makes the output voltage on pin 13 , 5V

int buttonState = digitalRead(2); // reads the value of pin 2 in buttonState

Integration of Sensors and Actuators with Arduino



```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

SENSORS

Pushbuttons and switches, which are probably the simplest sensors, allow some user input. Potentiometers (both rotary and linear) and rotary encoders enable you to measure movement.

- Electronic Elements
- Connects Physical/Quantity/Measurements into electrical signals
- Can be analog or digital

Types of sensors

- Temperature
- Humidity
- Compass
- Light
- Sound
- Motion sensors

ACTUATORS

An Actuators a type of motor for moving or controlling a mechanism or system. It is operated by a source of energy, typically electric current, hydraulic fluid pressure, or pneumatic pressure, and converts that energy into motion. An actuator is the mechanism by which a control system acts upon an environment.

Interfacing Sensors with Arduino Microcontroller

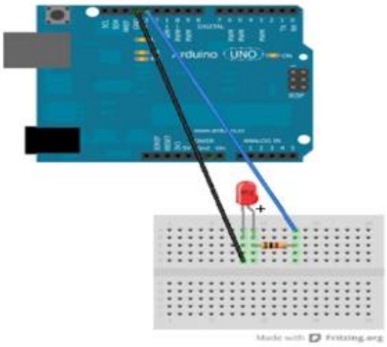
Attach one leg of LDR to 5V and another leg attach 110Kregister with that leg of LDR connected to A0Attach another leg of register to the ground

Sample Code in Arduino

```
        const int sensorMin = 0;     // sensor minimum, discovered through experiment
        const int sensorMax = 600;    // sensor maximum, discovered through experiment

        void setup() {   // initialize serial communication:   Serial.begin(9600);   }

        void loop() {   // read the sensor:   int sensorReading = analogRead(A0);   // map the sensor range to
        a range of four options:   int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

        switch (range) {   case 0:   // your hand is on the sensor

        Serial.println("dark");    break;   case 1:   // your hand is close to the sensor

        Serial.println("dim");    break;   case 2:   // your hand is a few inches from the sensor
        Serial.println("medium");    break;   case 3:   // your hand is nowhere near the sensor
        Serial.println("bright");    break;   }   delay(1);      // delay in between reads for stability }
```

Components of Various Projects

The Bread Board, Resistors, LED's

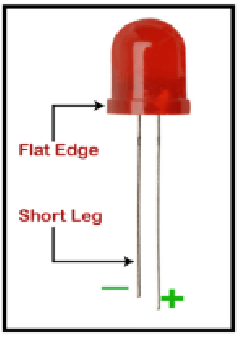


Wiring an LED with Arduino

Note: Additional 1K resistor should be used to prevent damage to the pins of the LED if it is reversed.

The long leg of the LED is connected to the pin 13 the short leg is to ground (GND)

Code for LED Blink



```
Int ledPin = 13;

Void setup() {

        pinMode(ledPIn,OUTPUT); }

Void loop() {

        digitalWrite(ledPin, HIGH);

        delay(500); // need delay to see the leds blinking

        digitalWrite(ledPin, LOW);

        delay(500);

}
```

Wiring a switch with Arduino

The resistor in this setup is called pull-down because it pulls the pin voltage down to GND (0V) if the switch is open.

Push button switch

10K Resistor, 5V, GND, D2 (max input 5V)

Digital input with Arduino

Code

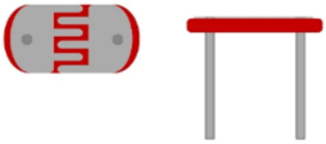
```
Int sensorPin = 2; // button switch

Void setup(){

        serial,begin(9600;

        pinMode(sensorPin, INPUT);

}

Void loop(){

Int sensorValues = digitalRead(sensorPin);

Serial.println(sensorValue);

}
```

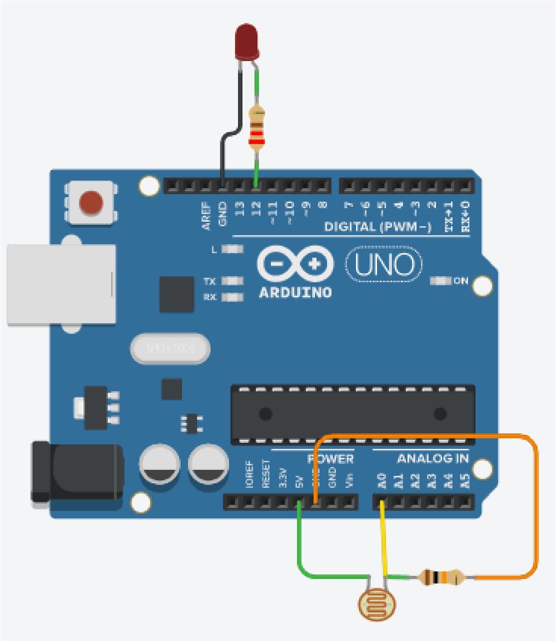
Arduino LDR

Arduino LDR (Light Dependent Resistor) project uses the photoresistor to light an LED. The LED will light up whenever there is dark or no light over the sensor.

What is photoresistor?

It is defined as a light-controlled resistor, which is also called as LDR. It is a variable resistor that controls the resistance in accordance with the received light intensity. It means, the resistance decreases as intensity of light increases.





An LDR can be used as a simple analog sensor

Code

```
Int sensorPin = A0 // anogue sensor pin

Void setup() {

Serial.begin(9600);

}

Void loop() {

Int sensorValue = analogRead(sensorPin);

Serial.println(sensorValue);

}
```

Implementation of IoT

What is Arduino in IoT?

Arduino acts as the brain of the system and processes the data from the sensor. Arduino is an open source hardware platform that is readily available for hobbyists & enthusiasts across the globe to build projects. It comes with an ATMEGA microcontroller that processes the data and facilitates the proper working of the IoT system. And the beauty is that the Arduino can be programmed ‘n’ number of times making it possible for you to build various types of IoT projects just by changing a simple code.

Nowadays a new emerging technology is Internet of Things (IoT): this post describes how to create Internet of things with Android and Arduino. In IoT project, all the physical objects (things) are connected together using internet infrastructure. Arduino board is one of the most important object in this ecosystem.

Arduino IoT Cloud

Arduino IoT Cloud is an application that helps makers build connected objects in a quick, easy and secure way. You can connect multiple devices to each other and allow them to exchange real-time data. You can also monitor them from anywhere using a simple user interface.

Arduino IoT Cloud is fully integrated in the Arduino Create ecosystem, you will be able to generate a template code in Arduino IoT Cloud and then edit and upload it to your board using the Arduino Web Editor.