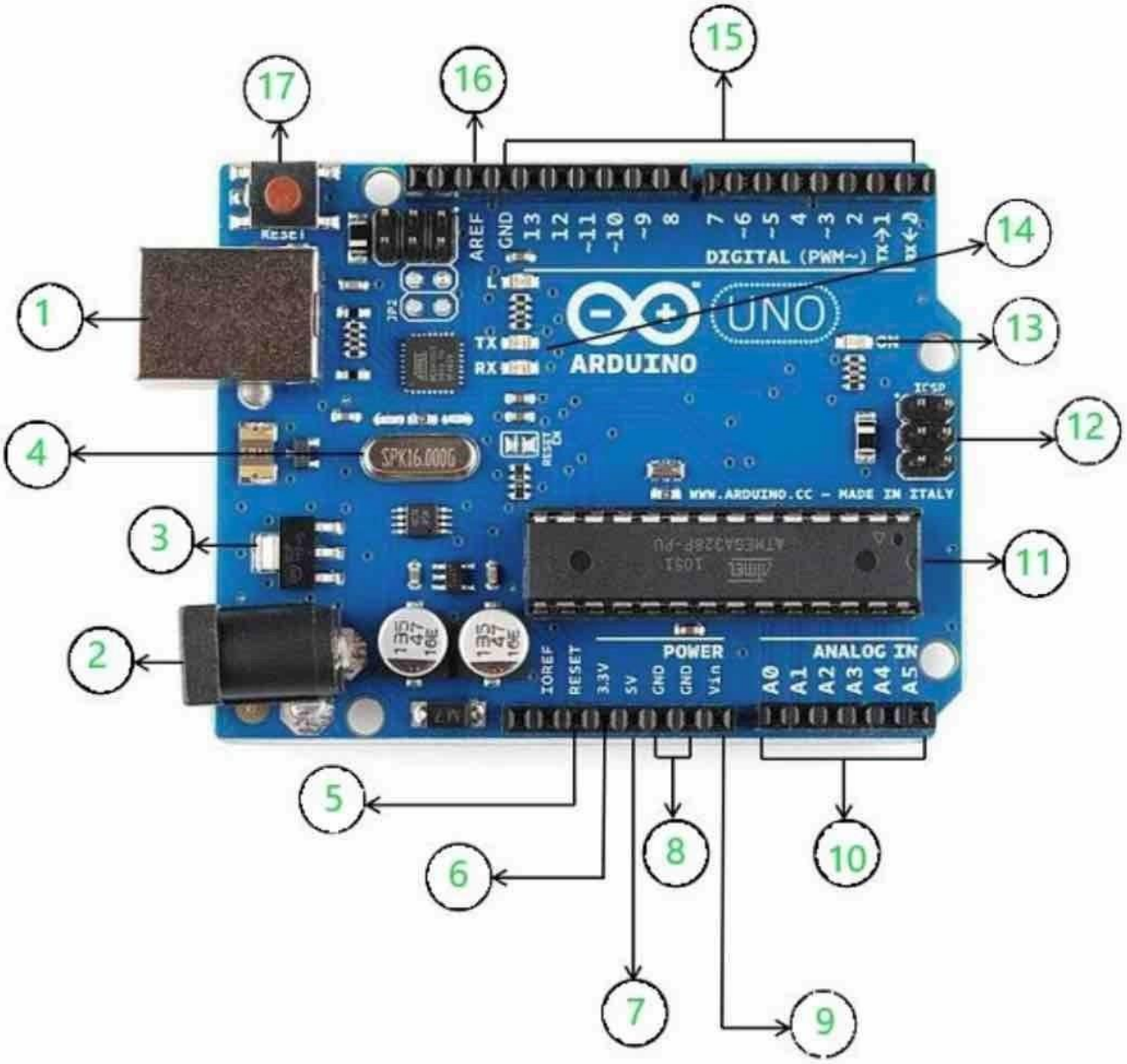
HAND GESTURE CONTROLLED ROBOTIC ARM

Arduino – Basics and Design

An Arduino board is a one type of microcontroller-based kit. The first Arduino technology was developed in the year 2005 by David Cuartielles and Massimo Banzi. The designers thought to provide easy and low-cost board for students, hobbyists and professionals to build devices. Arduino board can be purchased from the seller or directly we can make at home using various basic components. The best examples of Arduino for beginners and hobbyists includes motor detectors and thermostats, and simple robots. In the year 2011, Adafruit industries expected that over 3lakhs Arduino boards had been produced. But 7 lakhs boards were in user's hands in the year 2013. Arduino technology is used in many operating devices like communication or controlling.

Now that you know the origin of Arduino, it is essential to get yourself acquainted with the hardware that Arduino as a company offers. One of the main reasons for Arduino being so accessible and affordable across the globe is because all of the Arduino hardware is opensource. Being open-source has a plethora of advantages- anyone can access the design and build of the device and make improvements; anyone can use the same hardware design to create their product lineup. Since Arduino is open source, it has its own devoted community that strives to help the core company develop and improve its hardware products. Another significant advantage of being opensource, especially in the case of hardware, is that local companies can create replicas of the products, making it more accessible and affordable to the local consumers as it avoids hefty customs and shipping charges. All of these advantages contribute to Arduino being so widespread, affordable and ever improving.
It is necessary to know that Arduino doesn't necessarily offer just one piece of hardware, it provides a range of boards, each of which caters to a different level of expertise and have different use-cases altogether. Arduino Uno is one of the most basic and popular boards that Arduino offers. This is because it features an ATMega328 microcontroller that is both cheap and powerful enough for most basic beginner-level projects. Once you're familiar with Arduino IDE, you can move up to boards with more powerful and sophisticated

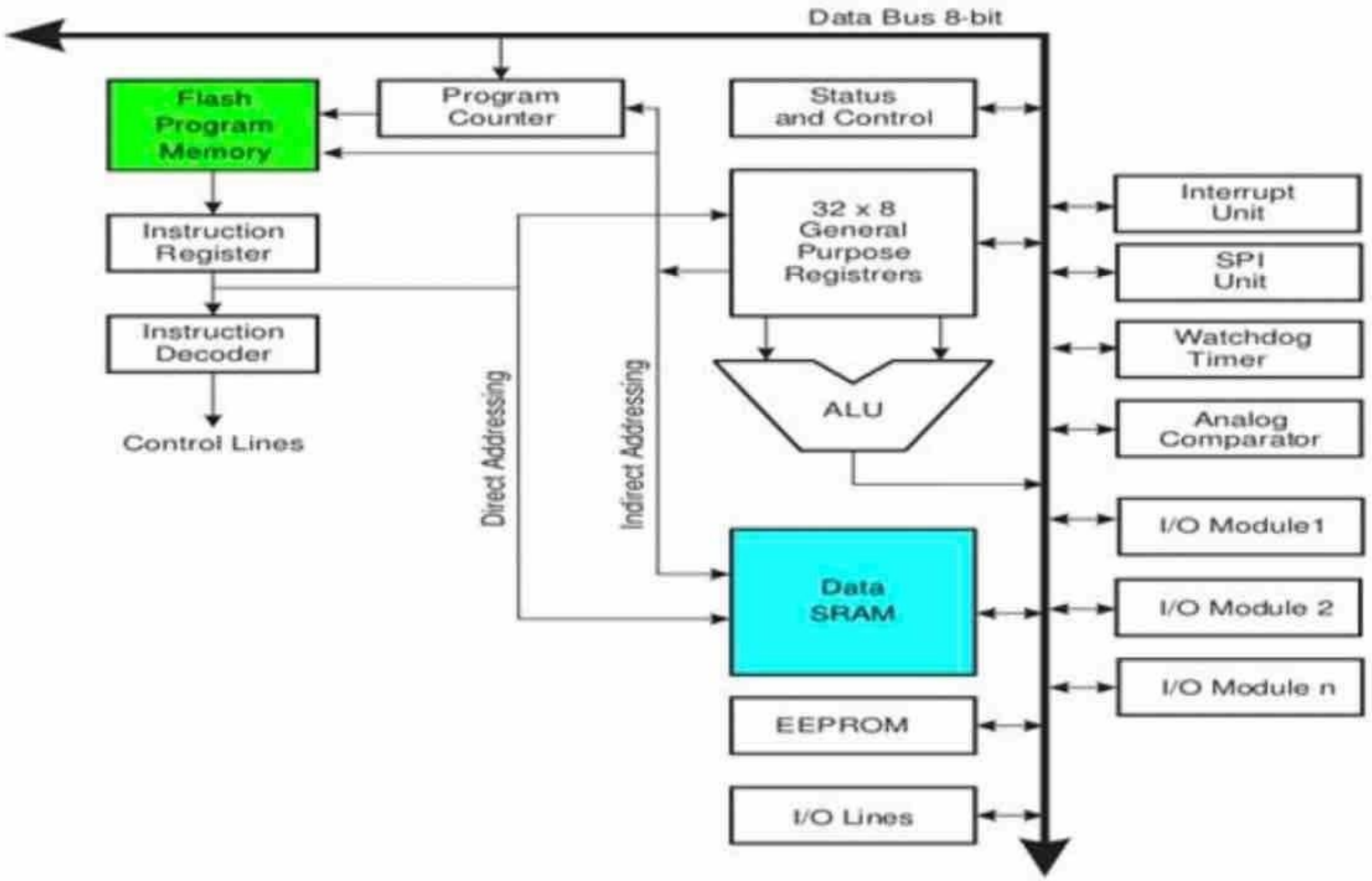
chipsets like the MKR range which is concerned with IoT applications and inter compatibility, or the Nano range which as the name suggests is designed to keep the form factor as small as possible while packing most of the features and power of the full-sized boards.



Using the above image as a reference, the labeled components of the board respectively are-

USB: can be used for both power and communication with the IDE
Barrel Jack: used for power supply
Voltage Regulator: regulates and stabilizes the input and output voltages
Crystal Oscillator: keeps track of time and regulates processor frequency
Reset Pin: can be used to reset the Arduino Uno
3.3V pin: can be used as a 3.3V output **5V pin:**
can be used as a 5V output

GND pin: can be used to ground the circuit
Vin pin: can be used to supply power to the board
Analog pins(A0-A5): can be used to read analog signals to the board
Microcontroller (ATMega328): the processing and logical unit of the board
ICSP pin: a programming header on the board also called SPI
Power indicator LED: indicates the power status of the board **RX and TX LEDs:** receive (RX) and transmit (TX) LEDs, blink when sending or receiving serial data respectively
Digital I/O pins: 14 pins capable of reading and outputting digital signals; 6 of these pins are also capable of PWM
AREF pins: can be used to set an external reference voltage as the upper limit for the analog pins
Reset button: can be used to reset the board



Arduino Architecture

Arduino Architecture:

Arduino's processor basically uses the Harvard architecture where the program code and program data have separate memory. It consists of two memories- Program memory and the data memory. The code is stored in the flash program memory, whereas the data is stored in the data memory. The Atmega328 has 32 KB of flash memory for storing code (of which 0.5 KB is used for the bootloader), 2 KB of SRAM and 1 KB of EEPROM and operates with a clock speed of 16MHz.

How to program an Arduino?

The most important advantage with Arduino is the programs can be directly loaded to the device without requiring any hardware programmer to burn the program. This is done because of the presence of the 0.5KB of Bootloader which allows the program to be burned into the circuit. All we have to do is to download the Arduino software and writing the code. The Arduino tool window consists of the toolbar with the buttons like verify, upload, new, open, save, serial monitor. It also consists of a text editor to write the code, a message area which displays the feedback like showing the errors, the text console which displays the output and a series of menus like the File, Edit, Tools menu.

5 Steps to program an Arduino

- Programs written in Arduino are known as sketches. A basic sketch consists of 3 parts
    1. Declaration of Variables
    2. Initialization: It is written in the setup () function.
    3. Control code: It is written in the loop () function.

- The sketch is saved with. ino extension. Any operations like verifying, opening a sketch, saving a sketch can be done using the buttons on the toolbar or using the tool menu.
- The sketch should be stored in the sketchbook directory.

- Chose the proper board from the tool's menu and the serial port numbers.
- Click on the upload button or chose upload from the tool's menu. Thus, the code is uploaded by the bootloader onto the microcontroller

Few of basic Arduino functions are:

- **digitalRead**(pin): Reads the digital value at the given pin.
- **digitalWrite** (pin, value): Writes the digital value to the given pin.
- **pinMode** (pin, mode): Sets the pin to input or output mode.
- **analogRead**(pin): Reads and returns the value.
- **analogWrite** (pin, value): Writes the value to that pin.
- **serial. begin** (baud rate): Sets the beginning of serial communication by setting the bit rate.

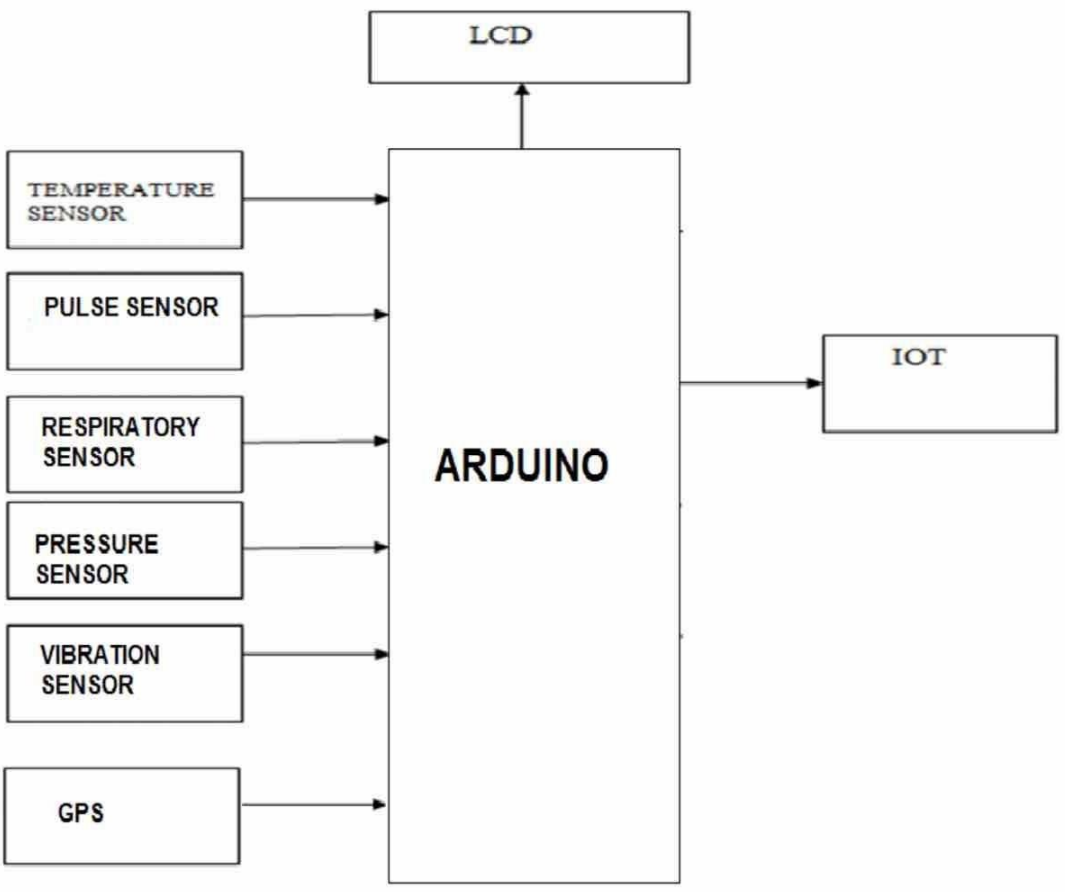
How to Design your own Arduino?

We can also design our own Arduino by following the schematic given by the Arduino vendor and also available at the websites. All we need are the following components- A breadboard, a led, a power jack, an IC socket, a microcontroller, few resistors, 2 regulators, 2 capacitors.

- The IC socket and the power jack are mounted on the board.
- Add the 5v and 3.3v regulator circuits using the combinations of regulators and capacitors.
- Add proper power connections to the microcontroller pins.
- Connect the reset pin of the IC socket to a 10K resistor. • Connect the crystal oscillators to pins 9 and 10
- Connect the led to the appropriate pin.
- Mount the female headers onto the board and connect them to the respective pins on the chip.
- Mount the row of 6 male headers, which can be used as an alternative to upload programs.
- Upload the program on the Microcontroller of the readymade Arduino and then pry it off and place back on the user kit.

7 Reasons why Arduino is being preferred these days

1. It is inexpensive
2. It comes with an open-source hardware feature which enables users to develop their own kit using already available one as a reference source.
3. The Arduino software is compatible with all types of operating systems like Windows, Linux, and Macintosh etc.
4. It also comes with open-source software feature which enables experienced software developers to use the Arduino code to merge with the existing programming language libraries and can be extended and modified.
5. It is easy to use for beginners.
6. We can develop an Arduino based project which can be completely stand alone or projects which involve direct communication with the software loaded in the computer.
7. It comes with an easy provision of connecting with the CPU of the computer using serial communication over USB as it contains built in power and reset circuitry.



So, this is some basic idea regarding an Arduino. You can use it for many types of applications. For instance, in applications involving controlling some actuators like motors, generators, based on the input from sensors.

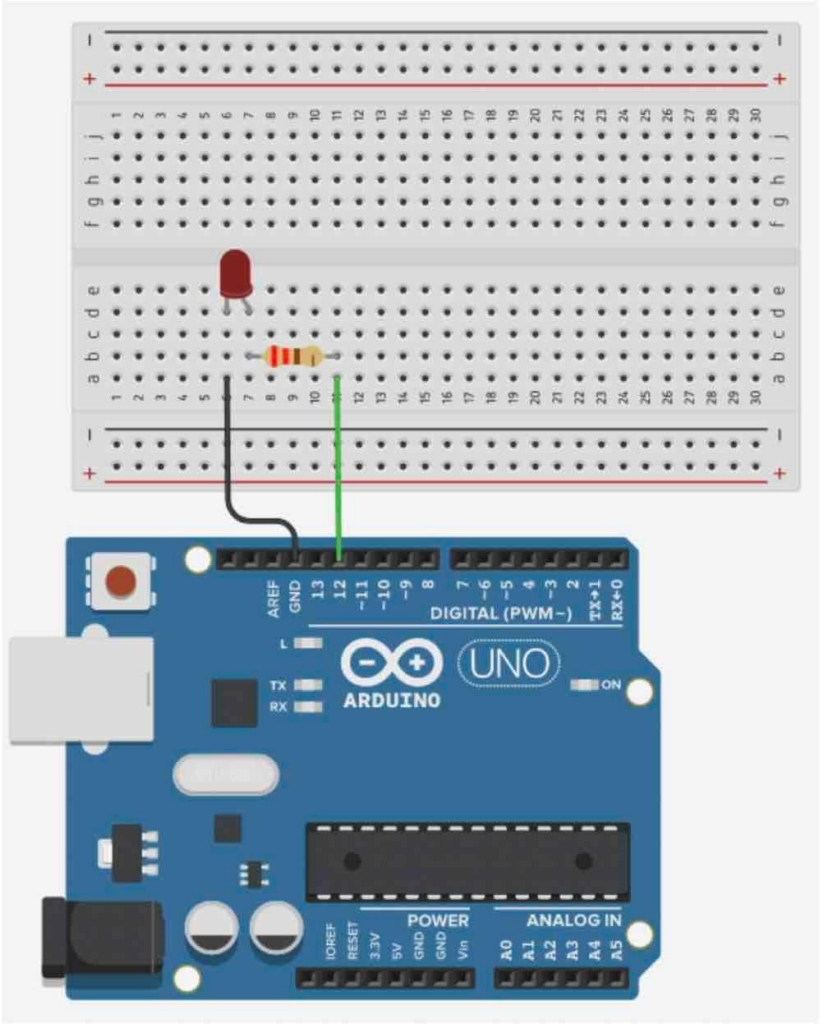
Arduino LED

First, you will setup your circuit with an Arduino board and an LED, and then discover different ways to control the LED.I will use Arduino Uno for the examples, but the instructions here apply to any Arduino board. If you want to just try the LED code on your Arduino, without doing the circuit, well, good news! You can just use the internal LED on pin 13 if you want

Create the Arduino LED circuit

- Arduino board.
- LED (any color, I will use red).
- Breadboard.
- 220 Ohm resistor (more info on the value later on).
- Some males to female wires.

Build the circuit



How to build the circuit:

- First make sure that the Arduino is powered off (no USB cable plugged to anything).
- Check the LED, you will see that one of the legs is shorter than the other one.
- Plug the shorter leg of the LED to a hole on the breadboard. Connect that leg to a GND pin of the Arduino, using a black cable if possible (convention for GND).
- Plug the longer leg of the LED to a different hole, on a different and independent line of the breadboard.
- Add a 220 Ohm resistor between this longer leg and a digital pin of the Arduino, using an additional colored wire (no red, no black) for convenience.

  Here we choose digital pin 12 on the Arduino Uno.

Arduino code to power on an LED

Power on the LED with digitalWrite ()

Let's start with the simplest thing. Let's say you just want to power on the LED when the Arduino program starts. This code will help us understand the setup and how digital pins work.

```
1.  #define LED_PIN 12
2.
3.  void setup()
4.  {
5.    pinMode(LED_PIN, OUTPUT);
6.    digitalWrite(LED_PIN, HIGH);
7.  }
8.
9.  void loop() {}
```

This code is very short and will just power on the LED on pin 12. Let's analyze it.

#Define LED_PIN 12

First, and this is a best practice, we create a define for the pin number. This way, anytime we need to use this digital pin, we can write LED_PIN instead of the hard-coded number. In the future, if you ever need to use a different digital pin (for example 11), then you just need to change the number here and it will update it everywhere in your program.

```
3.  void setup()
4.  {
5.    pinMode(LED_PIN, OUTPUT);
```

We enter the void setup () function, which will be executed once at the beginning of the program.

Before we can actually use a digital pin, we need to set a mode. Basically, you have 2 modes: output (if you want to control a component), or input (if you want to read some information from a component). Here, we want to control the LED, so we choose output. For that, we use the pinMode () function with 2 parameters:

- Pin number: here we use the LED_PIN that we previously defined.
- Mode: for output mode, we have to use OUTPUT (all uppercase).

After the execution of this line, the digital pin 12 will be set as output, and we can control the LED.

```
6.    digitalWrite(LED_PIN, HIGH);
7.  }
```

Now, to control the LED, it's very simple. We need to use the digitalWrite () function with 2 parameters:

- Pin number: again, we use the defined LED_PIN.
- State: we have 2 choices here. HIGH to power on the LED, and LOW to power it off.

So, after this line, the LED will be powered on.

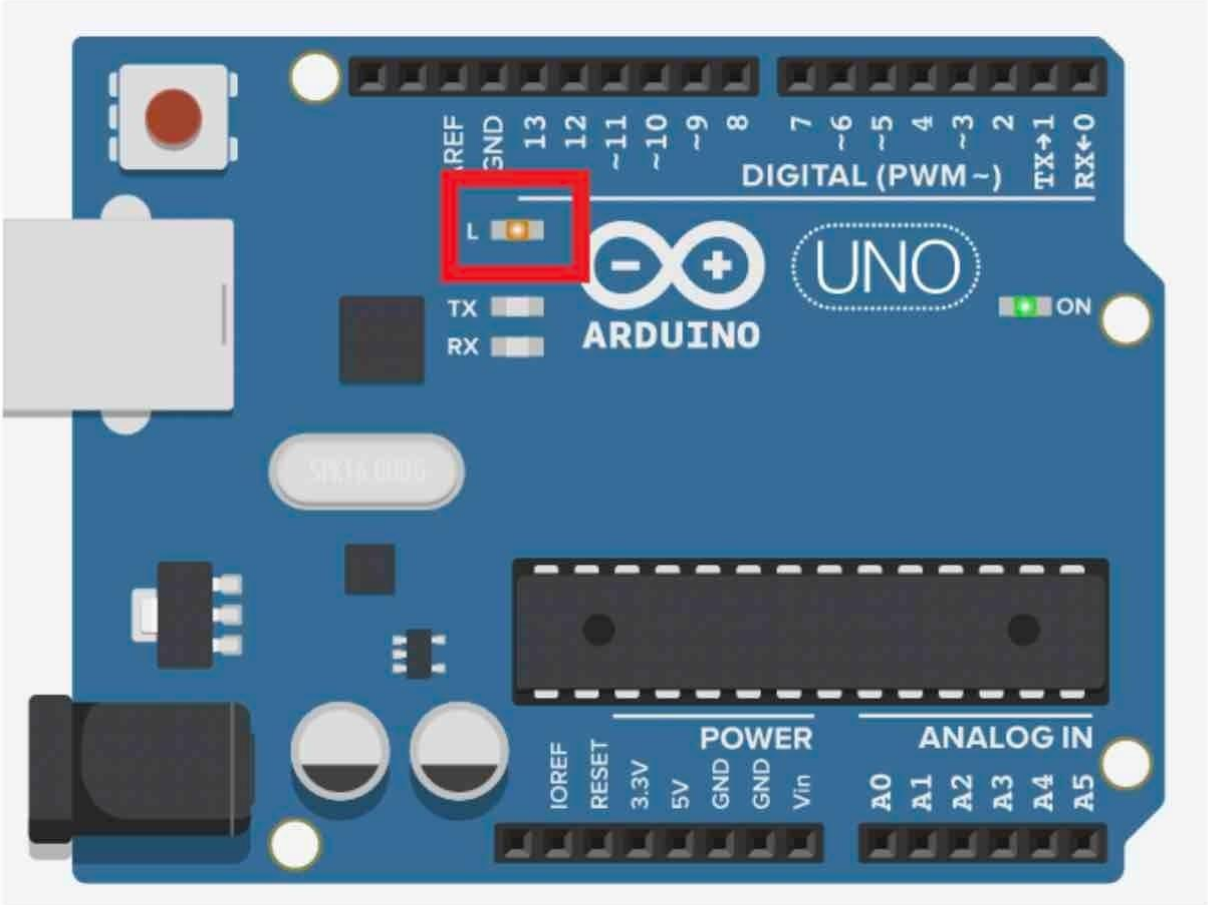
Note: it's important to call pinMode () before digitalWrite (), otherwise the LED won't be powered on because the mode is not set. **void** loop () {}

After the void setup () function is finished, we enter the void loop () and this function will be executed again and again, until you power off the Arduino.

Here we didn't write anything in the void loop () because we just want to power on the LED and that's it.

Test using only the built-in Arduino LED

If you don't want to build the circuit, or want to test with something even simpler, you can use the built-in LED on the Arduino, which is soldered on digital pin 13.



You can use LED_BUILTIN which is already predefined for this LED.

```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, HIGH);
}

void loop() {}
```

And you'll see the built-in LED powered on. Note: the location of the LED can vary depending on the type of your Arduino board.

Make the blink LED example

Now that you have the circuit and the code to setup the LED, let's do something a bit more interesting.

Let's make the LED blink, which means that we are going to:

1. Power on the LED,
2. wait,
3. Power off the LED,
4. wait,
5. Go back to 1.

Here's the code to do that (using our LED connected to digital pin 12).

```
#define LED_PIN 12

void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
}
```

And in the void loop (), we handle the blink mechanism. First, we power on the LED with digitalWrite () and HIGH, then we wait using delay (). The delay () function will block the program for a given amount of time (in milliseconds). So, here, we block the program for 500 milliseconds, or 0.5 second. After that, we power off the LED with digitalWrite () and LOW and wait again for 0.5 seconds using delay (). This way, we have a complete cycle of “LED powered on” + “LED powered off”.

Once we exit the void loop (), it is called again, and thus the LED is powered on again, etc etc. This blink example is maybe one of the most common examples, but also super useful when you get started with Arduino. If you correctly understand the circuit and the code, you’ve already made big progress.

Make the LED fade in/fade out

Let’s try something more interesting with PWM. Here we want to make the LED fade in (which means the brightness will slowly increase until the max), and then fade out (brightness will slowly decrease), so we can create a nice effect with the LED.

```
#define LED_PIN 11

void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  for (int i = 0; i <= 255; i++) {
    analogWrite(LED_PIN, i);
    delay(10);
  }

  for (int i = 255; i >= 0; i--) {
    analogWrite(LED_PIN, i);
    delay(10);
  }
}
```

PWM functionality with Arduino LED

As for now, you've seen that you can control an LED with 2 states: HIGH (fully on with 5V), or LOW (fully off with 0V). What if you want to control the brightness of the LED, for example to 40% of the max brightnessThat's where the PWM functionality is useful.

Very basically put, a PWM will allow you to output a voltage which is a percentage of the max voltage. So, if you want 40% brightness, you'd have to provide 40% of the voltage, which is 40% * 5V = 2V. Even if that sounds complicated, you will see that it's super easy to do in the code. First, we update the define line with the pin number

Let's write the code to power on the LED, but with just 40% of the max voltage.

```
#define LED_PIN 11

void setup()
{
  pinMode(LED_PIN, OUTPUT);
  analogWrite(LED_PIN, 102);
}

void loop() {}
```

11 instead of 12. As you can see this is very practical, as we don't have to change the number anywhere else in the code.

Then, instead of using digitalWrite (), we use analogWrite () with 2 parameters:

- Digital pin, here LED_PIN as previously defined.
- Percentage of the voltage, scaled to 0-255 (one byte of data). If we want 40%, we need to do 255 * 0.4 = 102. This will correspond to an output of 5V * 0.4 = 2V.

Arduino code for 3 LEDs

Let's make a very simple application: we want first the red LED to be powered on, then the yellow one, then the green one, and back to the red one.

```
#define LED_PIN_1 11
#define LED_PIN_2 10
#define LED_PIN_3 9

void setup()
{
  pinMode(LED_PIN_1, OUTPUT);
  pinMode(LED_PIN_2, OUTPUT);
  pinMode(LED_PIN_3, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN_1, HIGH);
  digitalWrite(LED_PIN_2, LOW);
  digitalWrite(LED_PIN_3, LOW);
  delay(1000);
  digitalWrite(LED_PIN_1, LOW);
  digitalWrite(LED_PIN_2, HIGH);
  digitalWrite(LED_PIN_3, LOW);
  delay(1000);
  digitalWrite(LED_PIN_1, LOW);
  digitalWrite(LED_PIN_2, LOW);
  digitalWrite(LED_PIN_3, HIGH);
  delay(1000);
}
```

Nothing really complicated: for the setup we just duplicate the code for each new LED. We add a define and we setup the mode with pinMode ().

Then, well you can use digitalWrite () on each LED – and analogWrite () if the pin is PWM compatible – whenever you want. In this case we just power on each LED, once at a time, with a 1 second delay.

ABSTRACT - HAND GESTURE CONTROLLED ROBOTIC ARM

This paper presents an application on how to control a robotic arm with the hand movement of the operator. Hand movement controlled robotic arms can move more naturally by following the movement of the operator's hand. A transmitter circuit, powered by an nRF24L01 breakout board and Arduino Pro Mini, is attached to a glove. The circuit also contains an IMU and flex sensor. For this project, a DIY flex sensor was used Now-a-days robotic arm is used in various areas such as military, defence, medical surgeries, pick and place function in industrial automation applications. Based on the gesture of human hands the robotic arm moves and performs the task and this system replicates the actions of human hands. The arm is very flexible and can be made suitable in places where the environment is not safe for humans like firework manufacturing industry, bomb diffusing etc.

1) INTRODUCTION

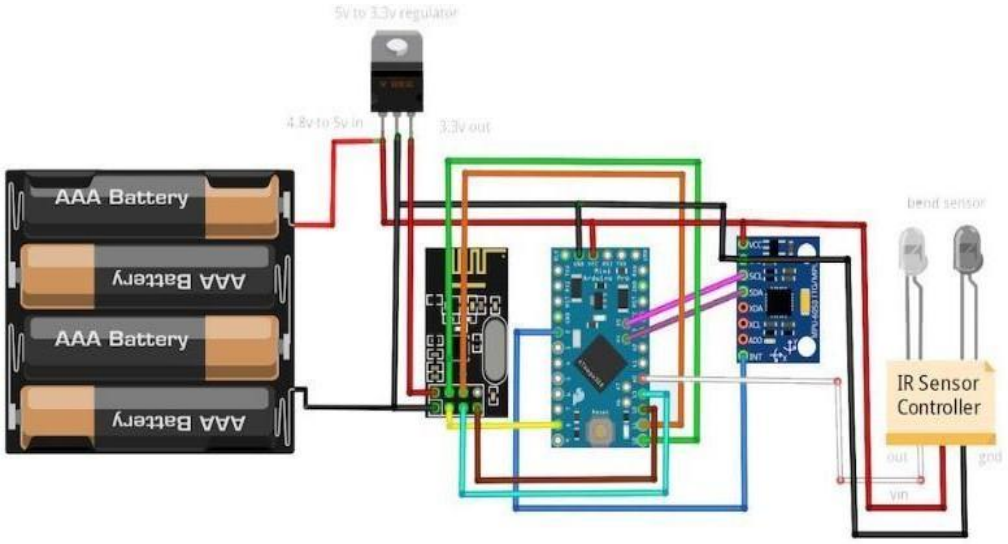
Robotics is a current emerging technology in the field of science Many universities throughout the world are engaged in this endeavour. In the upcoming years, civilization will benefit much from the new, burgeoning sector of robotics. These days, a wide variety of wireless robots are being produced and used for a variety of purposes. Instead of physically controlling it with a traditional remote controller, we have created a robotic arm that is operated and controlled wirelessly with the use of hand gestures. Signals are transmitted to the robot by an auto device installed to the gloves. Depending on distant finger and hand gestures, the robot moves and behaves in certain ways. The robot follows the actions of the fingers and hand to travel up, down, left, or right, pick objects up from one location, and retain them at the desired location.

1.1) PROBLEM STATEMENT

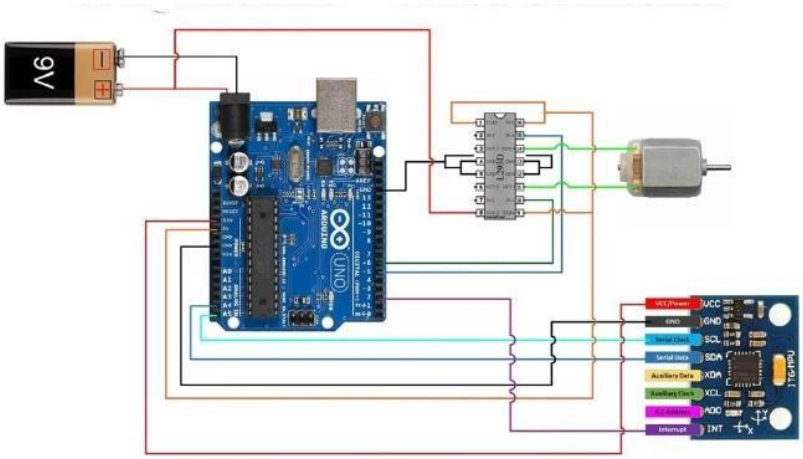
Gesture means the movement of hand and face of humans. The main objective of this project is to control the robotic arm using human gestures. The hand gesture controlled robotic arm will function by a wearable hand glove holding the transmitter circuit, from which the movements of the hand can be used as the input for the movement of the robotic arm. The basic idea is to develop a system which can recognize the human interaction with it to accomplish certain designated tasks. In our project, the transmitter circuit in the glove captures the movements of the hand with the help of sensors and converts the raw mechanical data into electrical form.

CIRCUIT DIAGRAM

FRITZING DIAGRAM FOR MPU6050:



FRITZING DIAGRAM FOR ROBOTIC ARM:



HARDWARE COMPONENTS:

1. DC motor:

A DC motor (**Direct Current motor**) is the most common type of motor. DC motors normally have just two leads, one positive and one negative. If you connect these two leads directly to a battery, the motor will rotate. If you switch the leads, the motor will rotate in the opposite direction.

2. Robotic arm kit

3. Touch sensor:

4. MPU6050 Gyroscope:

5. Arduino pro mini-5V/16MHz

6. Arduino Uno

7. NRF24L01 breakout board: A breakout board “breaks out” these pins onto a printed circuit board that has its own pins that are spaced perfectly for a solderless breadboard, giving you easy access to use the integrated circuit. There is all type of breakout boards – but most of them are for different types of sensors, for example: accelerometers, ultrasonic distance sensors, RFID tag sensors.

SOFTWARE COMPONENTS:

1. Arduino IDE

WORKING MECHANISM:

Step 1-Boards:

Two microcontroller boards were taken up as per the requirement of this project:

1. Arduino pro mini-

Contains an ATmega328 microcontroller running at 16 MHz with external resonator (0.5% tolerance). Has a 5V regulator, max 150mA output, Dc input 5V up to 12V, a 16 MHz crystal oscillator.

2. Arduino Uno

Arduino UNO is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button Step 2- Sensors: Touch Sensors:

Touch Sensors are the electronic sensors that can detect touch. They operate as a switch when touched. These sensors are used in lamps, touch screens of the mobile, etc… Touch sensors offer an intuitive user interface. Touch sensors work similar to a switch. When they are subjected to touch, pressure or force they get activated and acts as a closed switch. When the pressure or contact is removed, they act as an open switch.

In this gesture controlled Robotic Arm, touch sensor is used to control the gripper of the robotic arm. When the sensor obtains input, the servo motor attached to the gripper rotates and it opens.

MPU6050 Gyroscope:

It is used for the movement of the robotic arm in x- and y-axis. This sensor has a 3axis accelerometer, a 3-axis gyroscope, and an in-built temperature sensor. It can be used to measure parameters like Acceleration, Velocity, Orientation, Displacement, etc.

Features:

- Communication: I2C protocol with configurable I2C Address
- Input Power Supply: 3-5V
- In-built temperature sensor

Step 3- Communication:

To get the most out of the system, it was better to transfer the data wirelessly between the Robotic arm and the glove.

Algorithm:

The MPU6050 sensor measures change in its position and this data is sent to the Arduino Uno through the analog pins. The received data is sent to a second Arduino through serial communication. And the communication medium is bluetooth. In order to achieve Bluetooth connection Bluetooth module is used. The two Bluetooth is paired up by their unique address and it is also configured such that it does not pair with any other device. So, the X and Y axis value changes of the MPU is processed in the range of 0 to 180 to process the DC motor. These values are sent to the receiver Arduino and the servo motors are driven based on the angle received from the transitter.

Step 4- Wheel Drive mechanism:

It is a basic 2 DC motor mechanism used to provide drive. They are attached to a railing which is used to create a 2-wheel structure similar to an axle.

Functioning:

A small object is placed near the arm at a certain distance within its reach. The operator stands at a distance and moves his hand accordingly. the driving motors receives the captured hand movement gestures and follows it. Once the arm is near the object, the operator uses the touch sensor to open the arm's gripper to pick up the object. The robotic arm is then moved to another desired position and then lowered.

This allows the user to command and direct the robotic arm from a distance, often up to 200 metres, in any way they see fit.

FEATURES COVERED:

The MPU6050 module can be moved in all the three axes.

Very strong and stable wireless communication upto 10 meters

Easy to use as we just have to wear the gloves installed with MPU6050 in order to use

Long battery life as we are just using DC motors.

CODE:

```
#include <Arduino.h>
#include <Wire.h>
#include <MPU6050_tockn.h>

// Create an MPU6050 object
MPU6050 mpu(Wire);

// Define motor control pins using 'const int' for good practice
const int motor1pin1 = 2;
const int motor1pin2 = 3;
const int motor2pin1 = 4;
const int motor2pin2 = 5;

// Set a threshold for movement. The robot won't move unless tilted enough.
const int threshold = 4000;

void setup() {
  Serial.begin(9600);

  // Start I2C communication FIRST
  Wire.begin();

  // Initialize the MPU6050 sensor
  mpu.begin();
```

```
  // Calibrate the sensor (optional but recommended)
  Serial.println("Calibrating MPU6050, do not move the sensor...");
  mpu.calcGyroOffsets(true);
  Serial.println("Calibration complete!");

  // Set motor pins as OUTPUT
  pinMode(motor1pin1, OUTPUT);
  pinMode(motor1pin2, OUTPUT);
  pinMode(motor2pin1, OUTPUT);
  pinMode(motor2pin2, OUTPUT);

  // Ensure motors are stopped initially
  stopMotors();
}

void loop() {
  // Update the sensor data
  mpu.update();

  // Get raw accelerometer values
  float accX = mpu.getAccX();
  float accY = mpu.getAccY();

  // --- Main Control Logic ---

  // Forward
  if (accX > threshold) {
    Serial.println("Forward");
    moveForward();
  }
  // Backward
  else if (accX < -threshold) {
    Serial.println("Backward");
    moveBackward();
  }
  // Right
```

```
  else if (accY > threshold) {
    Serial.println("Right");
    turnRight();
  }
  // Left
  else if (accY < -threshold) {
    Serial.println("Left");
    turnLeft();
  }
  // Stop
  else {
    Serial.println("Stop");
    stopMotors();
  }

  // A small delay to prevent spamming the serial monitor, but fast enough for
control
  delay(50);
}

// --- Motor Control Functions for Readability ---

void moveForward() {
  digitalWrite(motor1pin1, HIGH);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, HIGH);
  digitalWrite(motor2pin2, LOW);
}

void moveBackward() {
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, HIGH);
}

void turnLeft() {
```

```
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, HIGH); // Motor 1 backward
  digitalWrite(motor2pin1, HIGH); // Motor 2 forward
  digitalWrite(motor2pin2, LOW);
}

void turnRight() {
  digitalWrite(motor1pin1, HIGH); // Motor 1 forward
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);  // Motor 2 backward
  digitalWrite(motor2pin2, HIGH);
}

void stopMotors() {
  digitalWrite(motor1pin1, LOW);
  digitalWrite(motor1pin2, LOW);
  digitalWrite(motor2pin1, LOW);
  digitalWrite(motor2pin2, LOW);
}



}
```

Functionality of code:

The void setup() is **the first function to be executed in the sketch and it is executed only once**. It usually contains statements that set the pin modes on the Arduino to OUTPUT and INPUT.

This is where the bulk of your Arduino sketch is executed. The program starts directly after the opening curly bracket ({), runs until it sees the closing curly bracket (}), and jumps back up to the first line in loop() and starts all over.

The MPU6050's output is in the form of X axis and Y axis. Based on this we can program the motors to move forward or backward based on the angle of movement of the module. However, it is important to import MPU6050 module in order to use functionality. The conditional statement of IF ELSE is used to verify the position of x and

y axis respectively

FUTURE SCOPE:

- Hand gesture controlled industrial grade robotic arms can be developed

- Entertainment applications – Most videogames today are played either on game consoles, arcade units or PCs, and all require a combination of input devices. Gesture recognition can be used to truly immerse a player in the game world like never before.

- Automation systems – In homes, offices, transport vehicles and more, gesture recognition can be incorporated to greatly increase usability and reduce the resources necessary to create primary or secondary input systems like remote controls, car entertainment systems with buttons or similar. These are only a few of the settings and circumstances where gesture recognition technology can be used, and as is clear, it can fundamentally alter how we engage with the world around us, both at home and in public spaces for business.

LITERATURE SURVEY:

In the below paper, the author has developed a system (robotic arm) prepared by him, which is operated and controlled by capturing the gestures made by hand movements. The author operates the robotic arm wirelessly through an auto device attached to the gloves put on hands rather than manually controlling it through a conventional remote controller.

The arm is a TYPE-C robot, programmable, servo-controlled with continuous or point to point trajectories. It mimics the movement of the glove worn by the user. Its grabber functions with the use of flex sensors attached to the centre most finger, the degree of movement of the finger determines the angle of the mini servo which is attached to the gears controlling the movement of the grabber.

The arm contains 2 microcontroller boards (Arduino duemilanove, Arduino mega). It has an accelerometer sensor (that measures the forces applied on the sensor in all 3 directions), a gyroscope sensor (to measure the degree of rotation in all 3 axes) and a flex sensor (that is used for grabbing purposes in the Robotic Arm)

Keywords

Arduino Pro Mini, Transmitter circuit, Robotic, flex sensor, breakout board

REFERENCES:

1. P. P. Sarker, F. Abedin and F. N. Shimim, "R3Arm: Gesture controlled robotic arm for remote rescue operation," 2017 IEEE Region 10 Humanitarian Technology

Conference (R10-HTC), 2017, pp. 428-431, doi: 10.1109/R10-

HTC.2017.8288991

URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8288891&isnumber=8288891

2. Verma, S., 2013. Hand gestures remote controlled robotic arm. Advance in Electronic and Electric Engineering, 3(5), pp.601-606.

URL: http://www.ripublication.com/aeee/80_pp%20%20%20%20601-606.pdf

Verma, S., 2013. Hand gestures remote controlled robotic arm. Advance in Electronic and Electric Engineering, 3(5), pp.601-606.