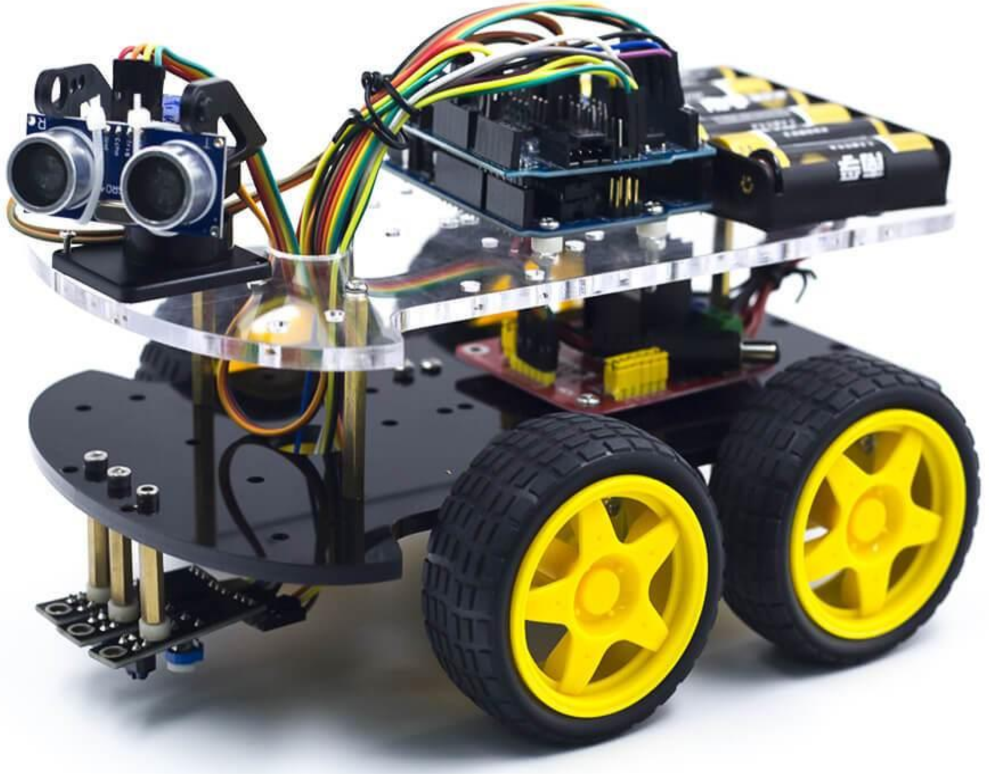
NSDC – Junior Skills Championship Mobile robotics

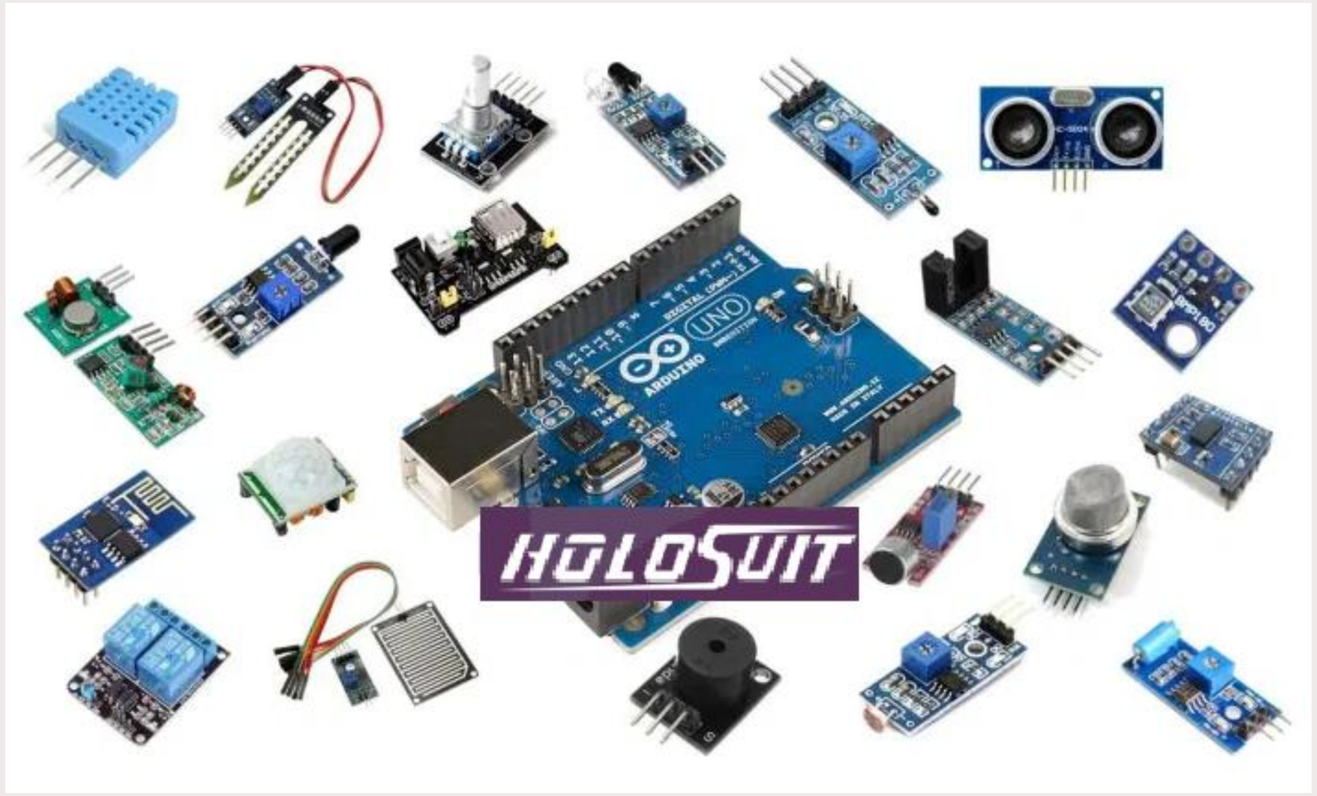


HOLOWORLD

HOLOWORLD

Agenda – Day 2

- Sensors
  - IR Sensor and working
  - PIR Sensor and working
  - Ultrasound sensor and working
- Arduino interface with sensors
- Arduino with motor using driver IC's.
  - L293D motor driver IC
  - L298N motor driver IC
  - Arduino Interfacing Motor with motor driver ICs



Functions in Arduino UNO

Functions allow structuring the programs in segments of code to perform individual tasks. The typical case for creating a function is when one needs to perform the same action multiple times in a program.

- Functions helps to conceptualize the program.
- Functions codify one action in one place so that the function only has to be thought about and debugged once.
- This also reduces chances for errors in modification
- Functions make the whole sketch smaller and more compact
- They make it easier to reuse code in other programs by making it modular, and using functions often makes the code more readable.

There are two required functions in an Arduino sketch or a program i.e. setup () and loop(). Other functions must be created outside the brackets of these two functions.

Function name

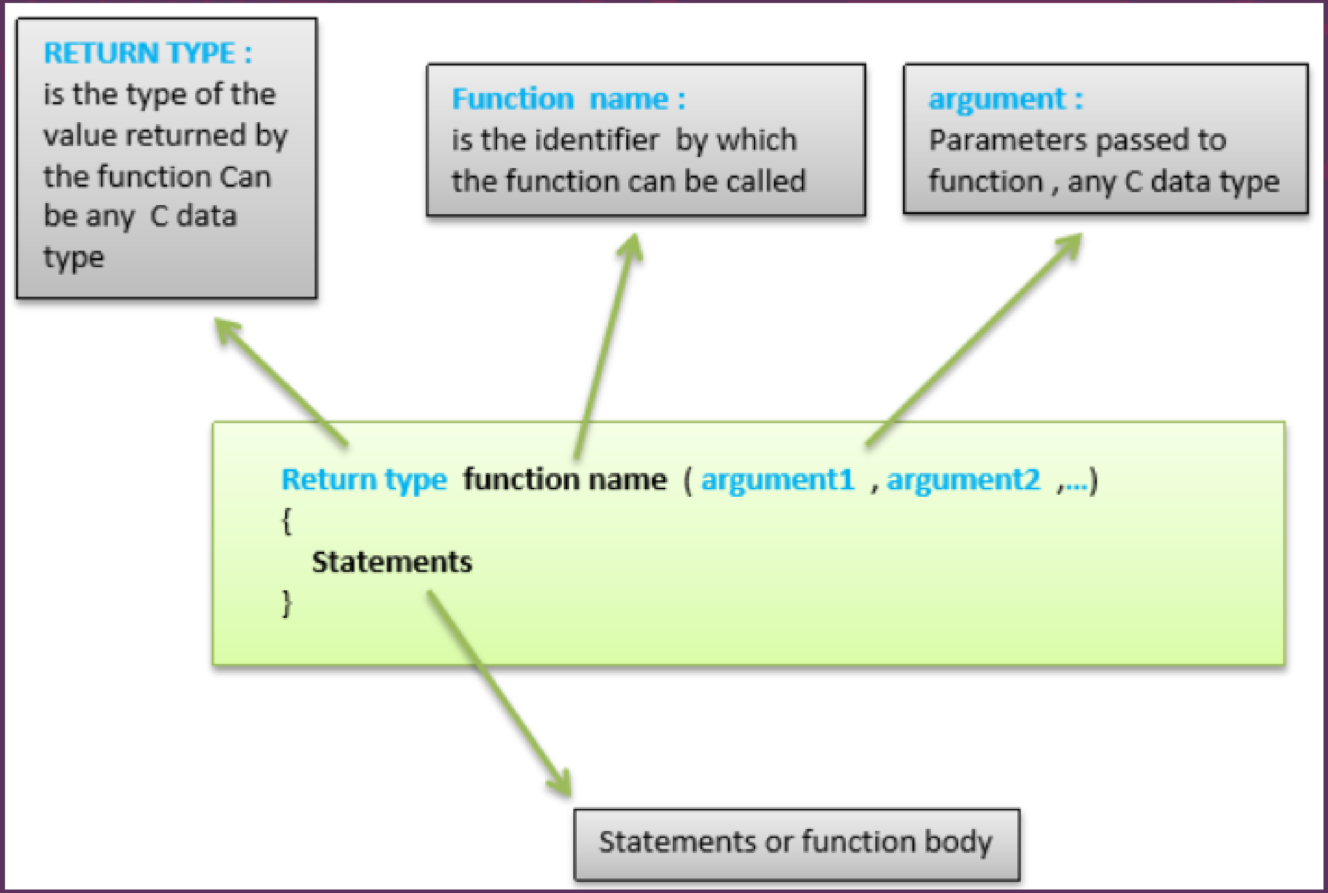
It consists of a name specified to the function. It represents the real body of the function.

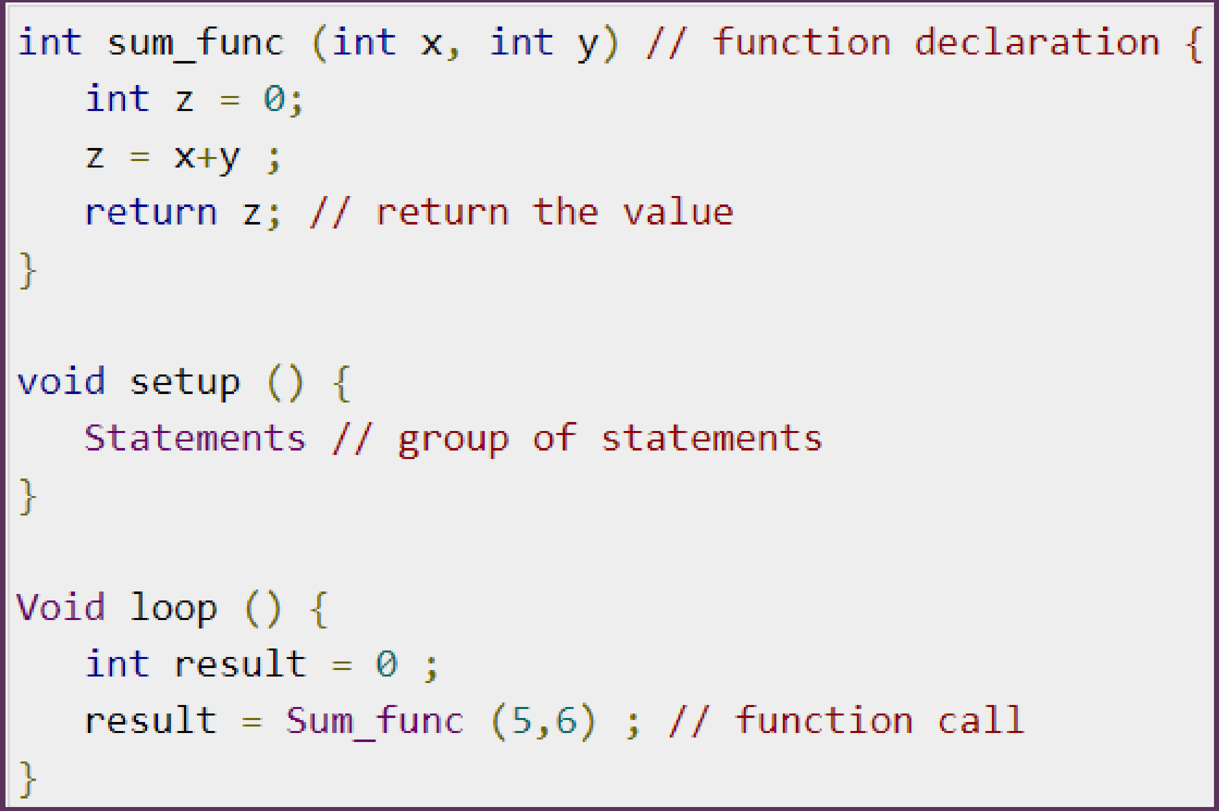
Function parameter

It includes the parameters passed to the function. The parameters are defined as the special variables, which are used to pass data to a function.

The function must be followed by **parentheses ( )** and the **semicolon ;**

The actual data passed to the function is termed as an argument.


RETURN TYPE :
is the type of the value returned by the function Can be any C data type
Function name :
is the identifier by which the function can be called
argument :
Parameters passed to function , any C data type
Return type function name (argument1 , argument2 ,...)
{
Statements
}
Statements or function body



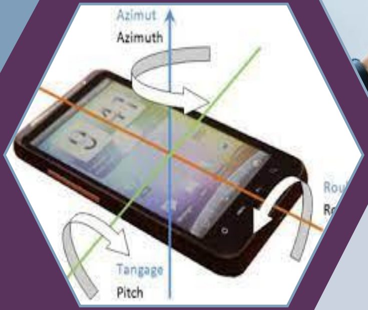
```
int sum_func (int x, int y) // function declaration {
   int z = 0;
   z = x+y ;
   return z; // return the value
}

void setup () {
   Statements // group of statements
}

Void loop () {
   int result = 0 ;
   result = Sum_func (5,6) ; // function call
}
```

HOLOWORLD

Sensor Applications

Motion Sensor

Touch Sensor



Automatic Sanitizer Dispenser

Parking Sensors

Temperature Control Sensor

Motion Sensor LEDs

Air bags

Fingerprint Sensor

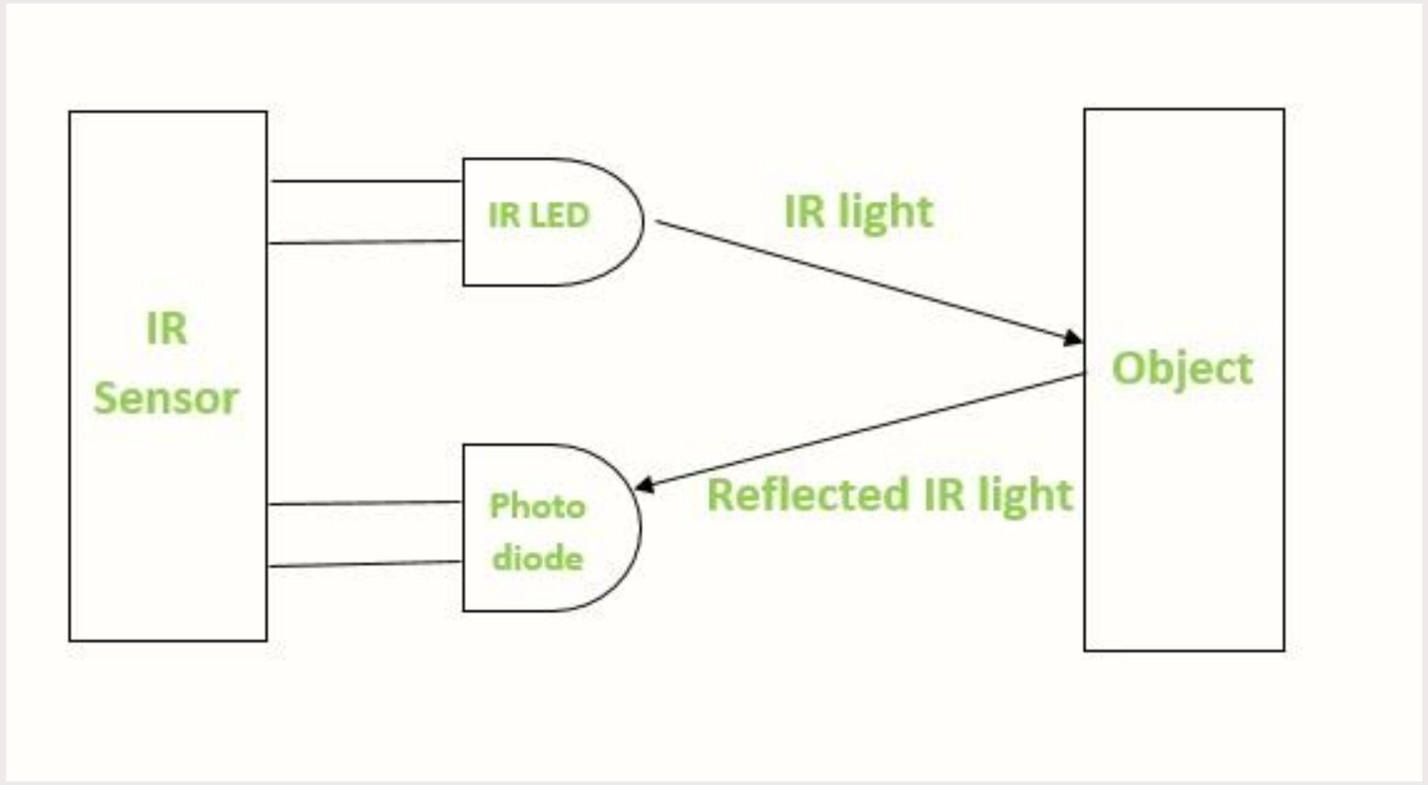
HOLOWORLD

IR sensor

- An infrared sensor is an electronic device, that emits in order to sense some aspects of the surroundings.

- An IR sensor senses in object as well as detects the motion.

- These types of sensors measure only infrared radiation, which cannot be viewed in human eye



HOLOWORLD

IR sensor

IR Sensor Working Principle:

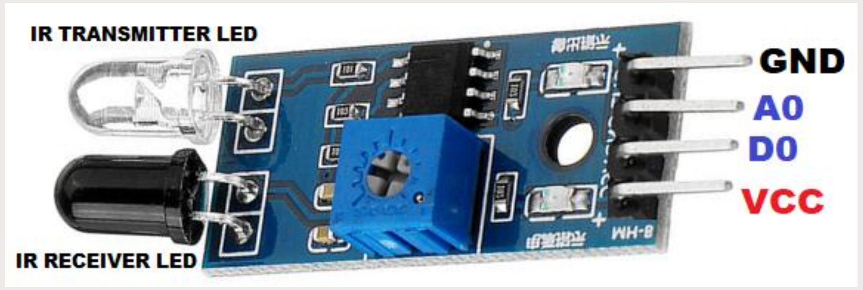
There are different types of infrared transmitters depending on their wavelengths, output power and response time. An IR sensor consists of an IR LED and an IR Photodiode, together they are called as Photocoupler or Optocoupler.

- **IR Transmitter or IR LED**

  Infrared Transmitter is a light emitting diode (LED) which emits infrared radiations called as IR LED's. Even though an IR LED looks like a normal LED, the radiation emitted by it is invisible to the human eye.

- **IR Receiver or Photodiode**

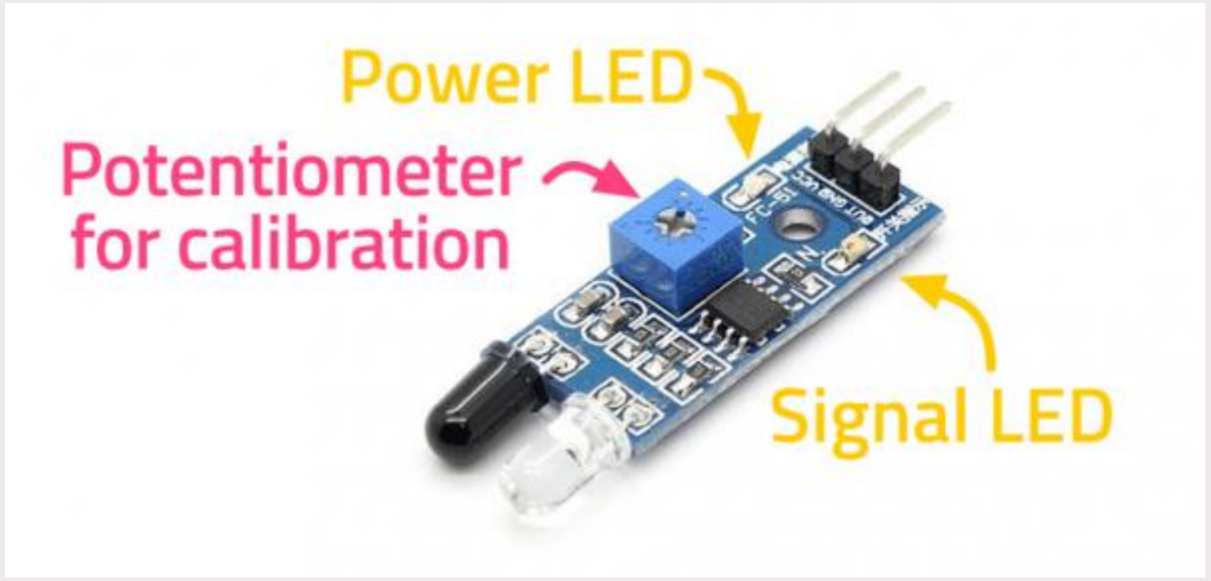
  Infrared receivers or infrared sensors detect the radiation from an IR transmitter. IR receivers come in the form of photodiodes and phototransistors.



Pin Name	Description
VCC	Power Supply Input
GND	Power Supply Ground
DO	Digital output (High or Low)
A0	Analog output (0 - 1023)

IR sensor

- The potentiometer is used to calibrate the output of the sensor according to the requirement.
- Then the light emitted by the IR LED is incident on the photodiode after hitting an object, the resistance of the photodiode falls from a huge value.

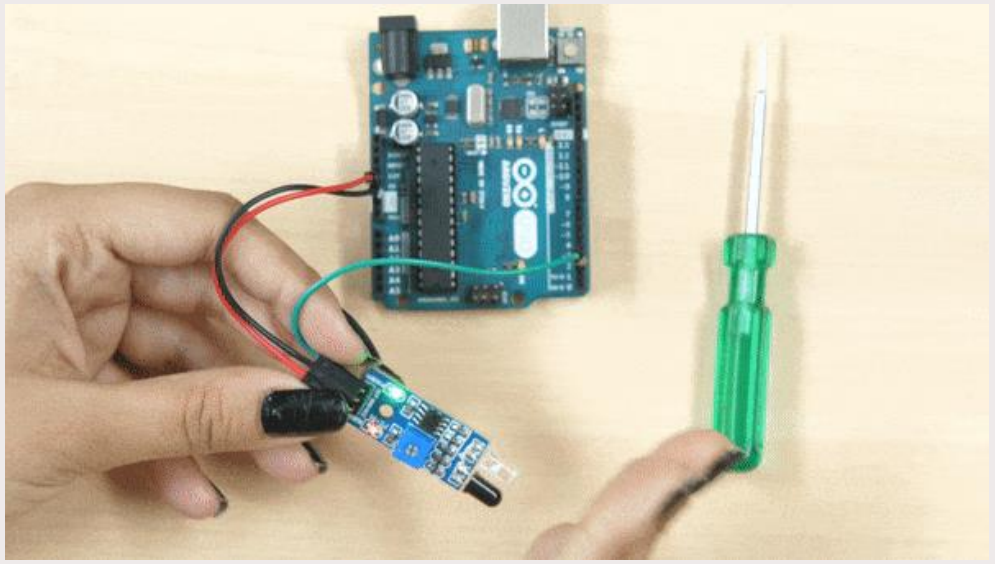


Pin Name	Description
VCC	Power Supply Input
GND	Power Supply Ground
DO	Digital output (High or Low)

Arduino with sensors

Calibrating the IR Sensor

- If the LED turns ON when you bring your hand near to the IR Sensor and turns OFF when you take your hand away, indicates that the IR Sensor is working properly
- What if the LED does not turn ON or OFF when necessary. Then, its the time when you need to calibrate your IR Sensor



- Keep your hand at some distance from the Sensor approximately 15cm. If the LED is OFF, then you need not do anything. Otherwise, turn the potentiometer using a screwdriver in the anti-clockwise direction until the LED turns OFF.
- Similarly, keep your hand close to the IR Sensor approximately 5cm. If the LED is ON, the sensor is perfectly calibrated. Otherwise, turn the potentiometer in the clockwise till the LED turns ON.
- Repeat the above two steps until your IR works perfectly fine.

Types of IR Sensor

There are two types of IR sensors are available and they are,

- Active Infrared Sensor
- Passive Infrared Sensor

- **Active Infrared Sensor**

Active infrared sensors consist of two elements: infrared source and infrared detector. . Infrared detectors include photodiodes or phototransistors. The energy emitted by the infrared source is reflected by an object and falls on the infrared detector.

- **Passive Infrared Sensor**

Passive infrared sensors are basically Infrared detectors. Passive infrared sensors do not use any infrared source and use only detector.

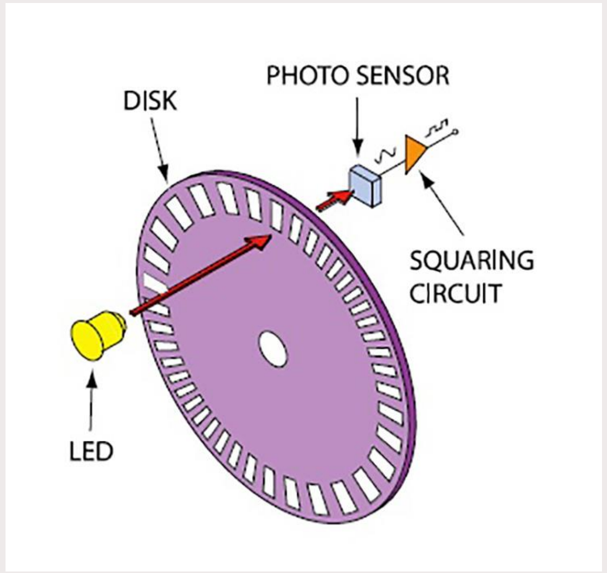
Difference between PIR Sensor and Motion Sensor

- The motion sensor: Able to detect the movement of people or objects.
- In most applications, these sensors are mainly used to detect human activities in a specific area.
- PIR is only one of the technical methods to detect motion, so we will say PIR sensor is a **subset of the motion sensor**.
- PIR sensor is small in size, cheap in price, low-power consumption and very easy to understand, which makes it quite popular.
- A lot of merchants will add “motion” between PIR sensor for the convenience of beginners.

HOLOWORLD

IR sensor

- The speed sensor is used for synchronizing the speed of multiple motors
- Thermal radiation sensor
- The Line tracing in robots
- Night vision glasses
- IR imaging



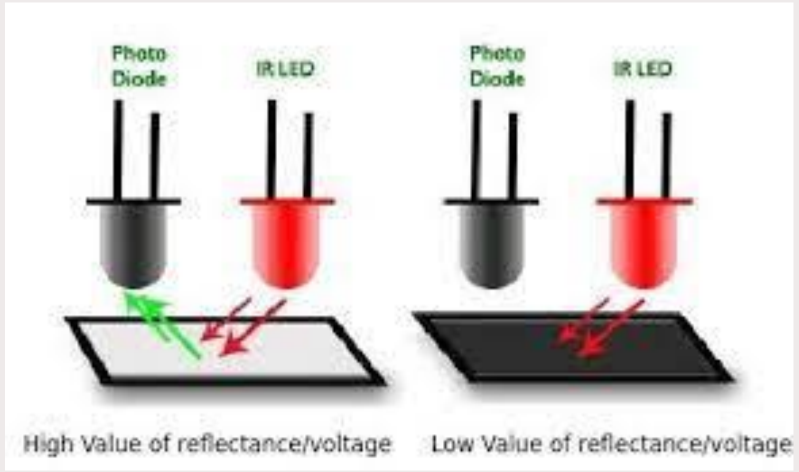
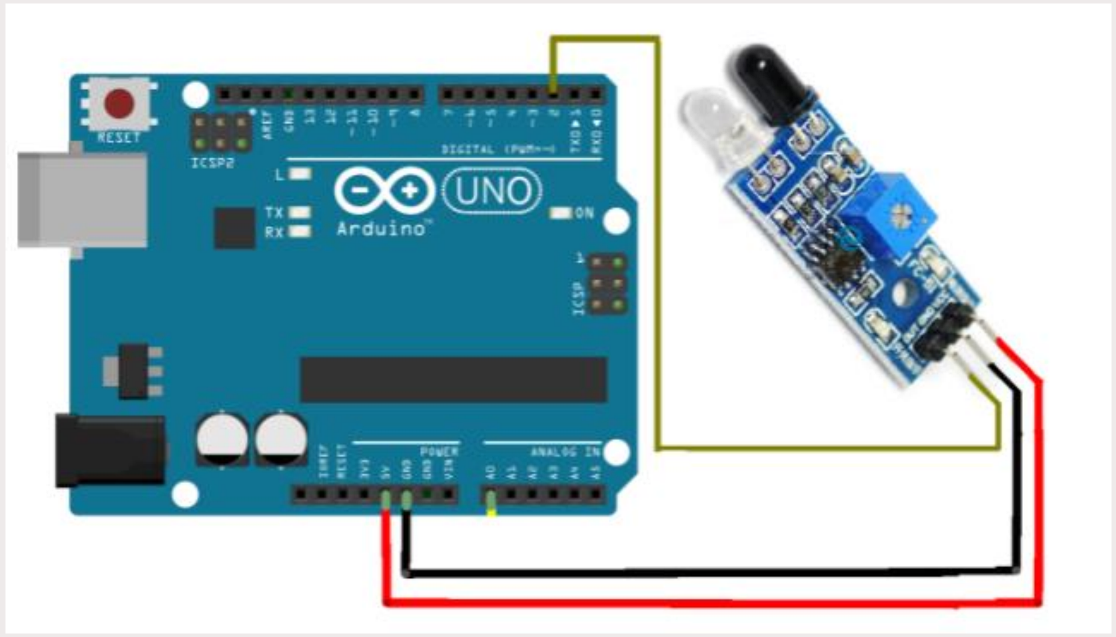


Photo Diode
IR LED
Photo Diode
IR LED
High Value of reflectance/voltage
Low Value of reflectance/voltage

Arduino with sensors

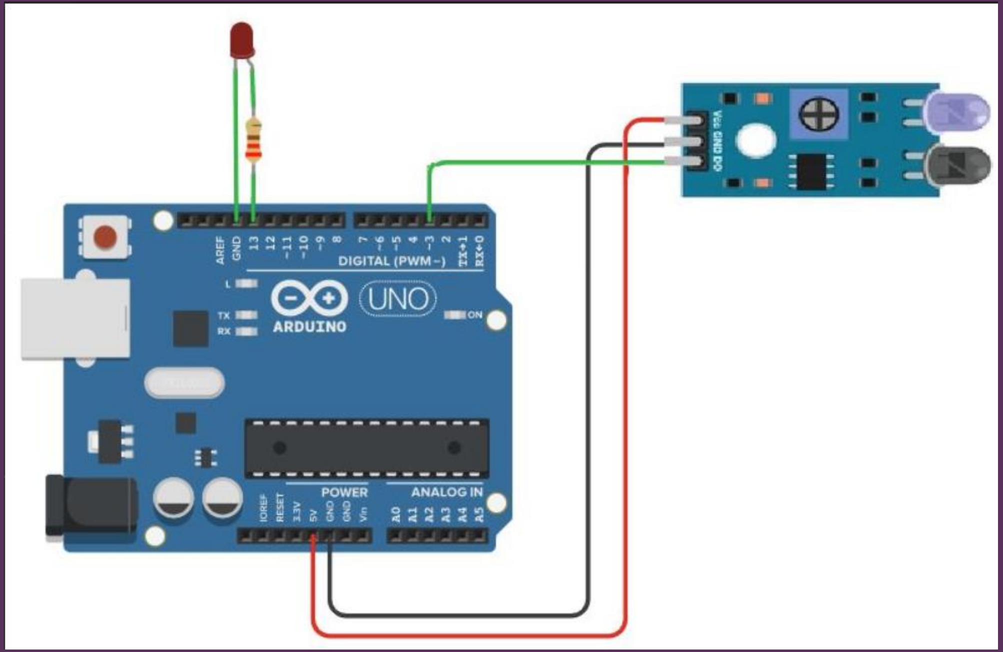
Interfacing the IR Sensor

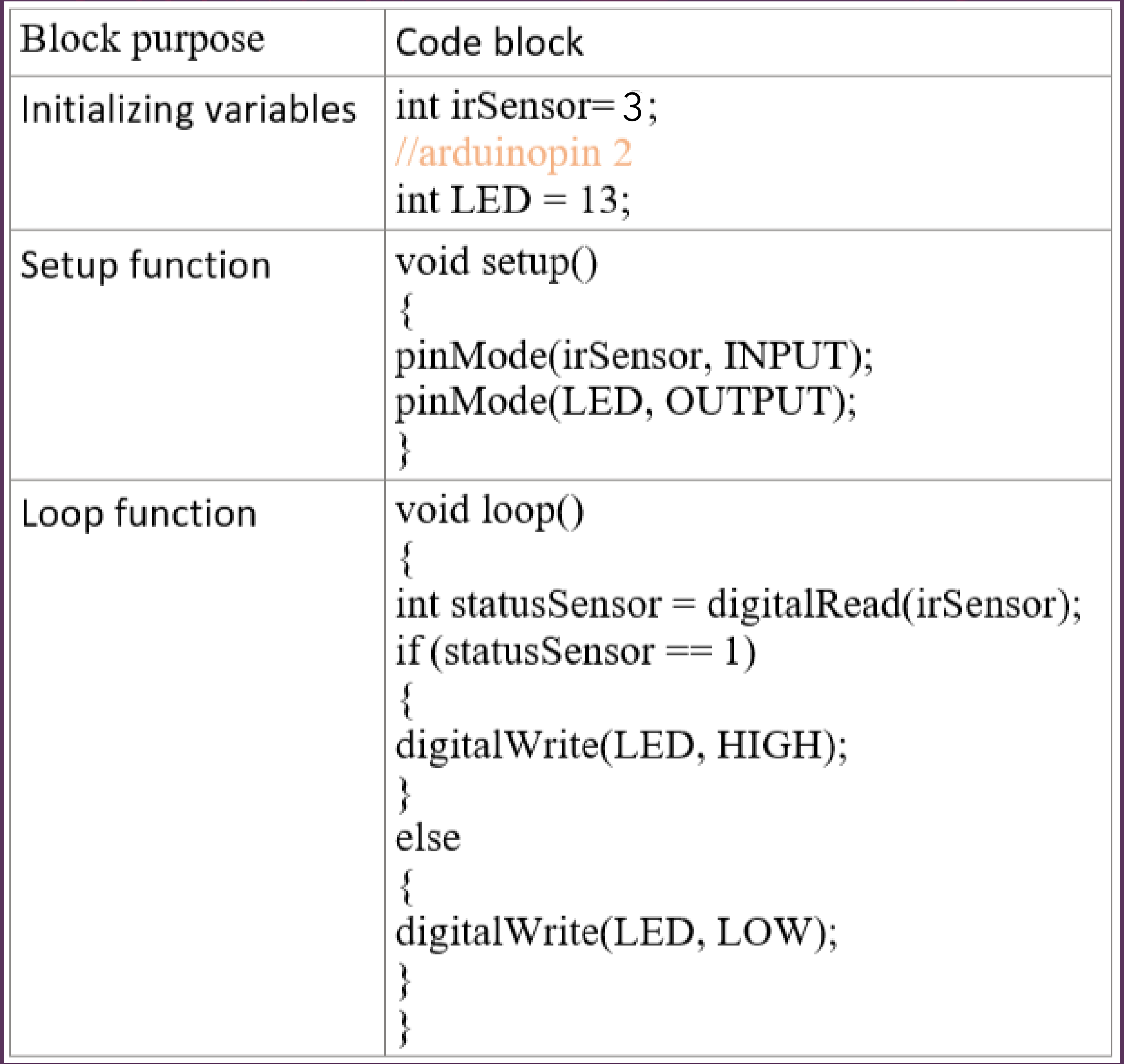
- Connect the sensor’s VCC pin to Uno’s 5V pin using a red male-to-female jumper cable
- Next, connect the sensor’s ground pin to Uno’s ground pin using a black male-to-female jumper cable
- Then, connect the sensors out pin to Uno’s digital pin 2 using a green male-to-female jumper cable



Interface IR sensor with LED

Components: IR sensor, LED, Arduino UNO, Jumper wires.





Block purpose	Code block

Initializing variables	int irSensor= 3;
	//arduinopin 2
	int LED = 13;

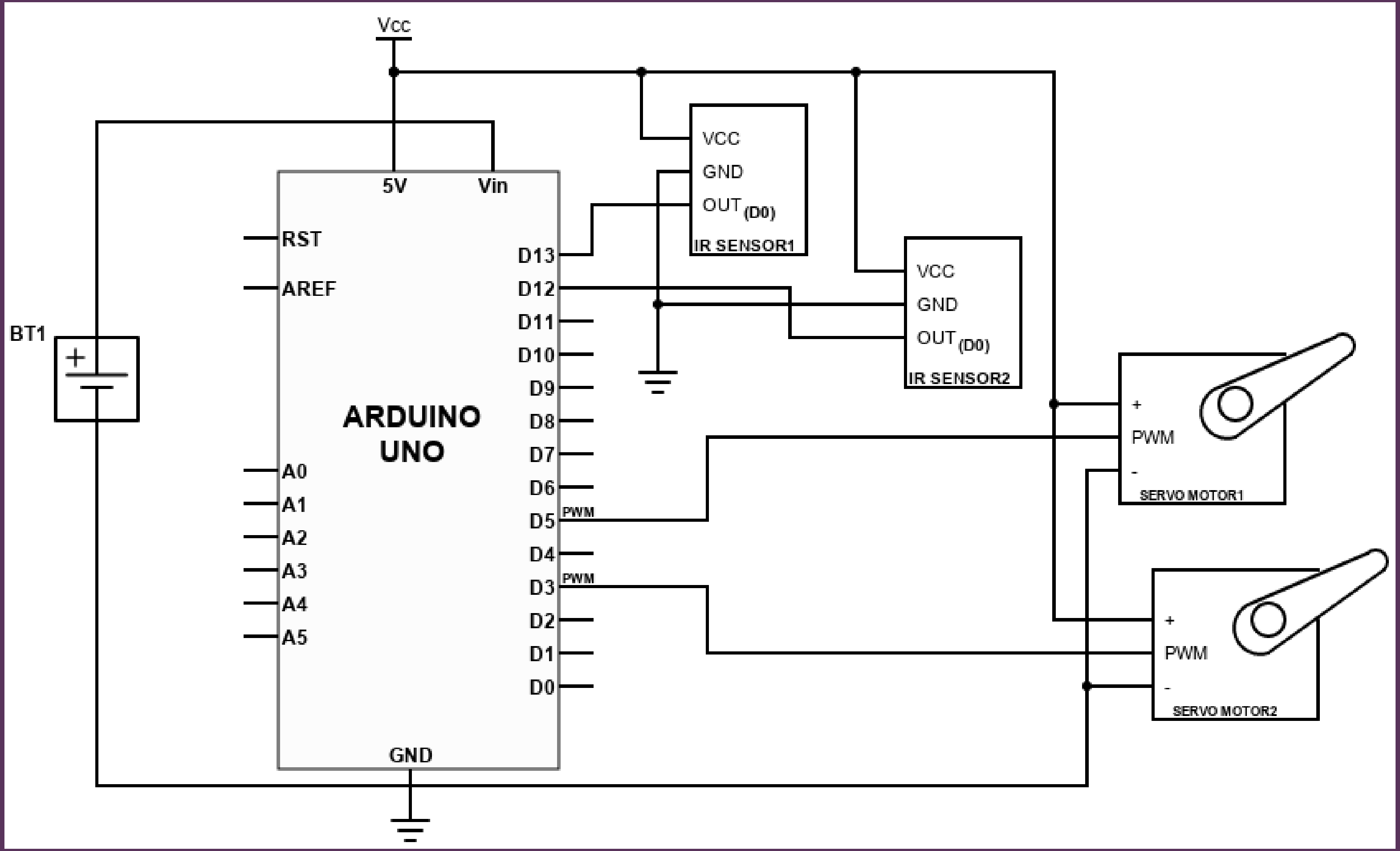
Setup function	void setup()
	{
	pinMode(irSensor, INPUT);
	pinMode(LED, OUTPUT);
	}

Loop function	void loop()
	{
	int statusSensor = digitalRead(irSensor);
	if (statusSensor == 1)
	{
	digitalWrite(LED, HIGH);
	}
	else
	{
	digitalWrite(LED, LOW);
	}
	}

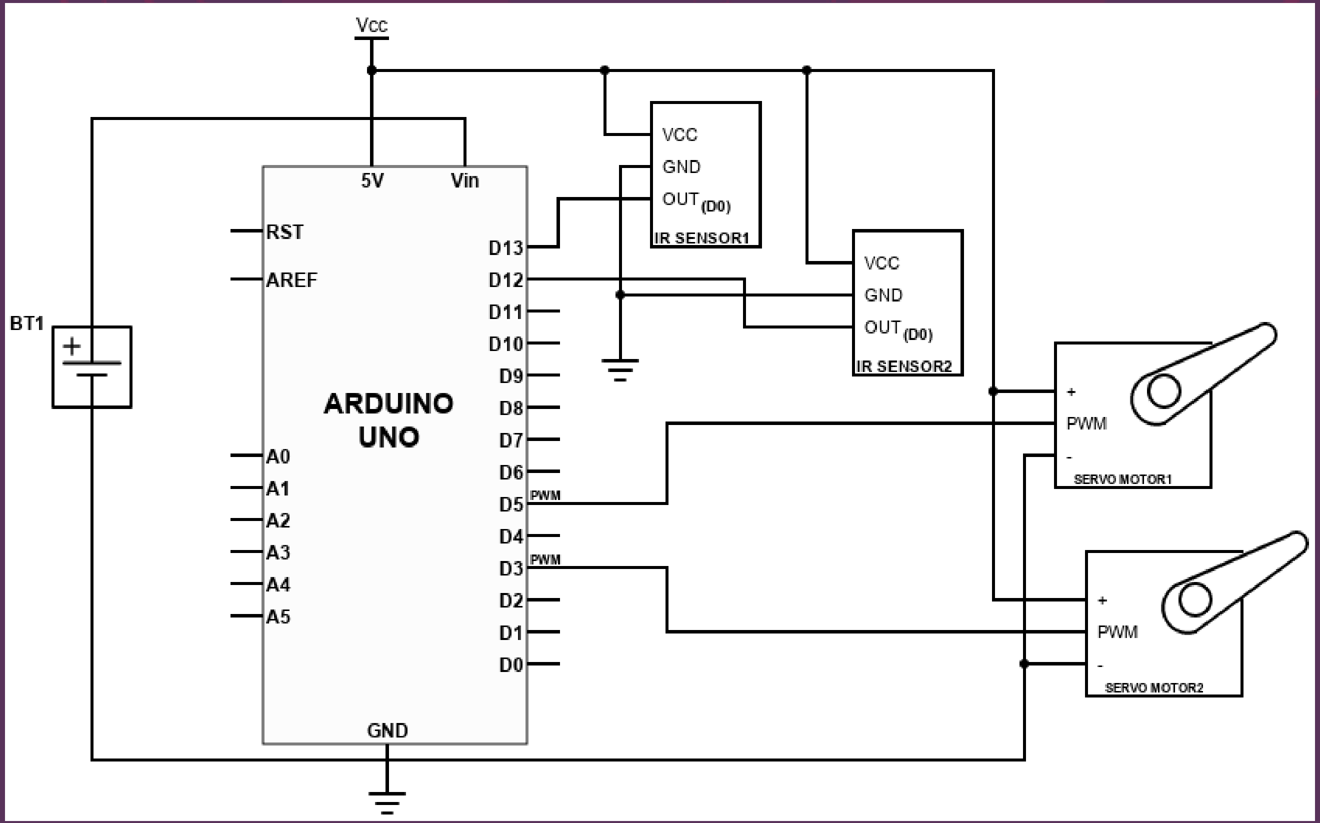
Let us design a circuit for train detection.

Components: Arduino UNO, IR based proximity sensors, LED, Wires, USB 2.0 Cable, Jumper wires, Servo motors

Gates should be closed when the IR sensor is HIGH.



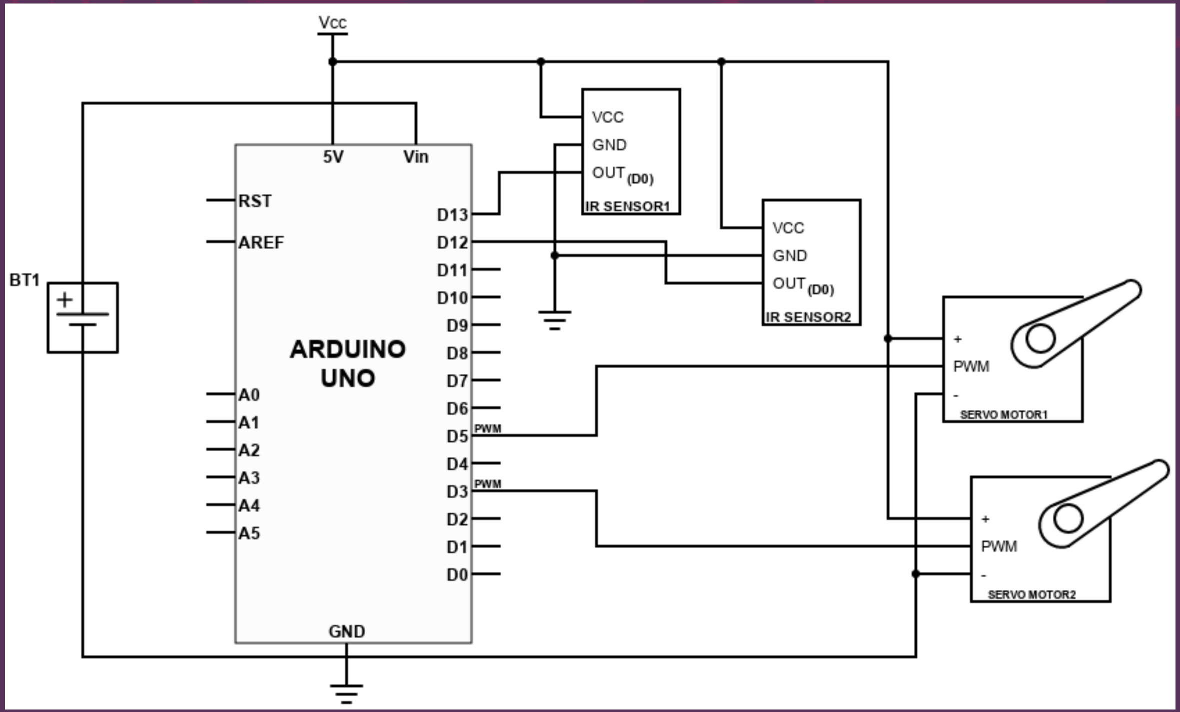
Gates should be closed when the IR sensor is HIGH.

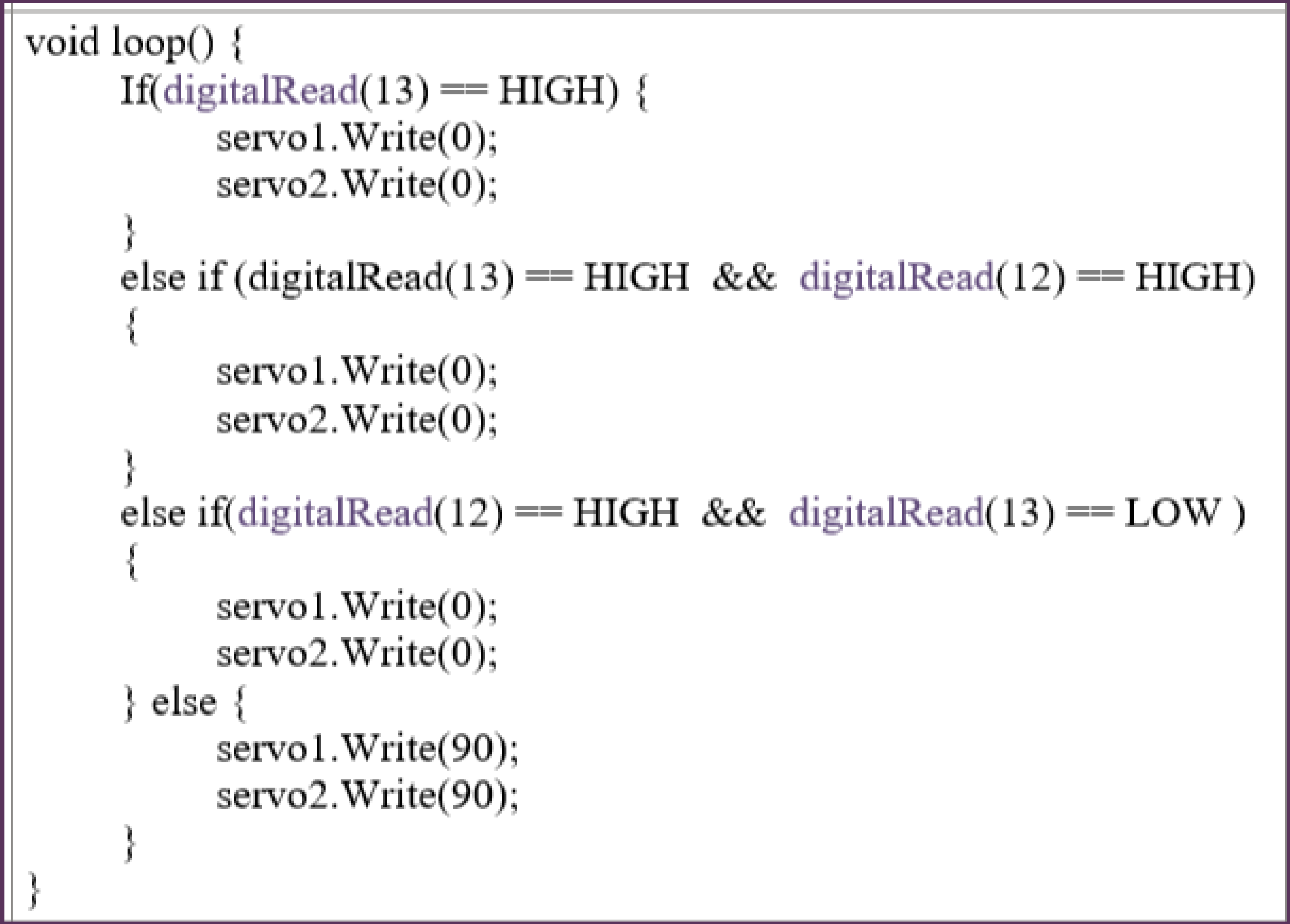


Block purpose	Code block

Initializing variables	`#include <Servo.h>`
	`Servo servo1, servo2 ;`
	`int sensor1 = 13;`
	`int sensor2 = 12;`
	`int servo1_position=0;`
	`int servo2_position=0;`

Setup function	`void setup() {`
	`        Serial.begin(9600);`
	`        pinMode(sensor1,INPUT);`
	`        pinMode(sensor2,INPUT);`
	`        servo1.attach(5);`
	`        servo2.attach(3);`
	`}`

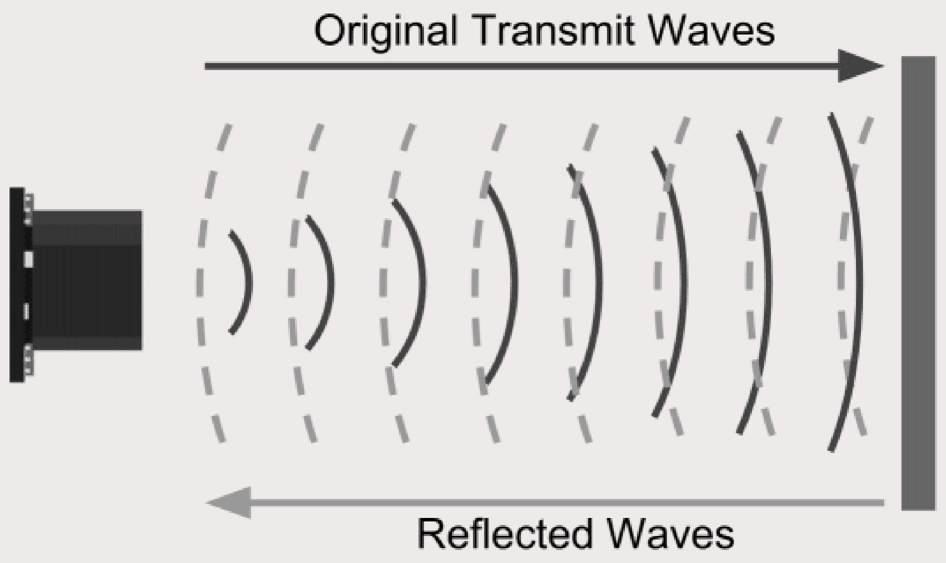




```
void loop() {
        If(digitalRead(13) == HIGH) {
                servo1.Write(0);
                servo2.Write(0);
        }
        else if (digitalRead(13) == HIGH  &&  digitalRead(12) == HIGH)
        {
                servo1.Write(0);
                servo2.Write(0);
        }
        else if(digitalRead(12) == HIGH  &&  digitalRead(13) == LOW )
        {
                servo1.Write(0);
                servo2.Write(0);
        } else {
                servo1.Write(90);
                servo2.Write(90);
        }
}
```

Ultrasound sensor

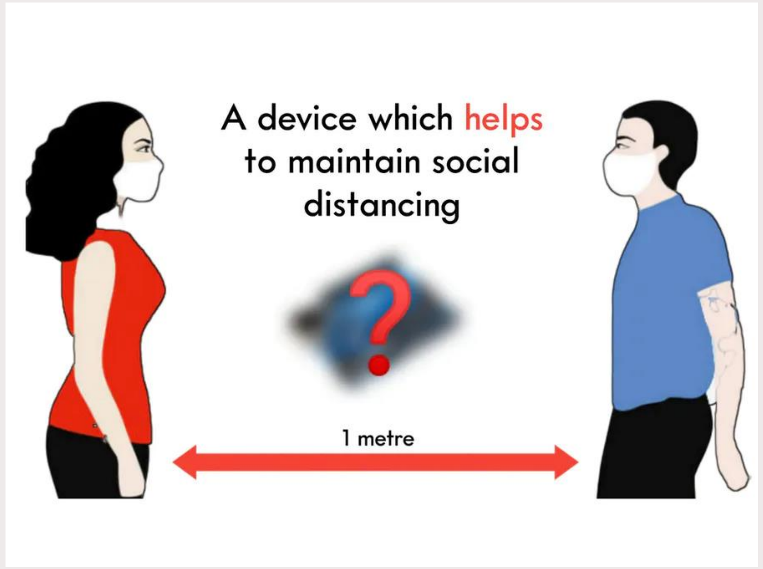
- An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves and converts the reflected sound into an electrical signal.
- Ultrasonic waves travel faster than the speed of audible sound
- Ultrasonic sensors have two main components: the transmitter and the receiver

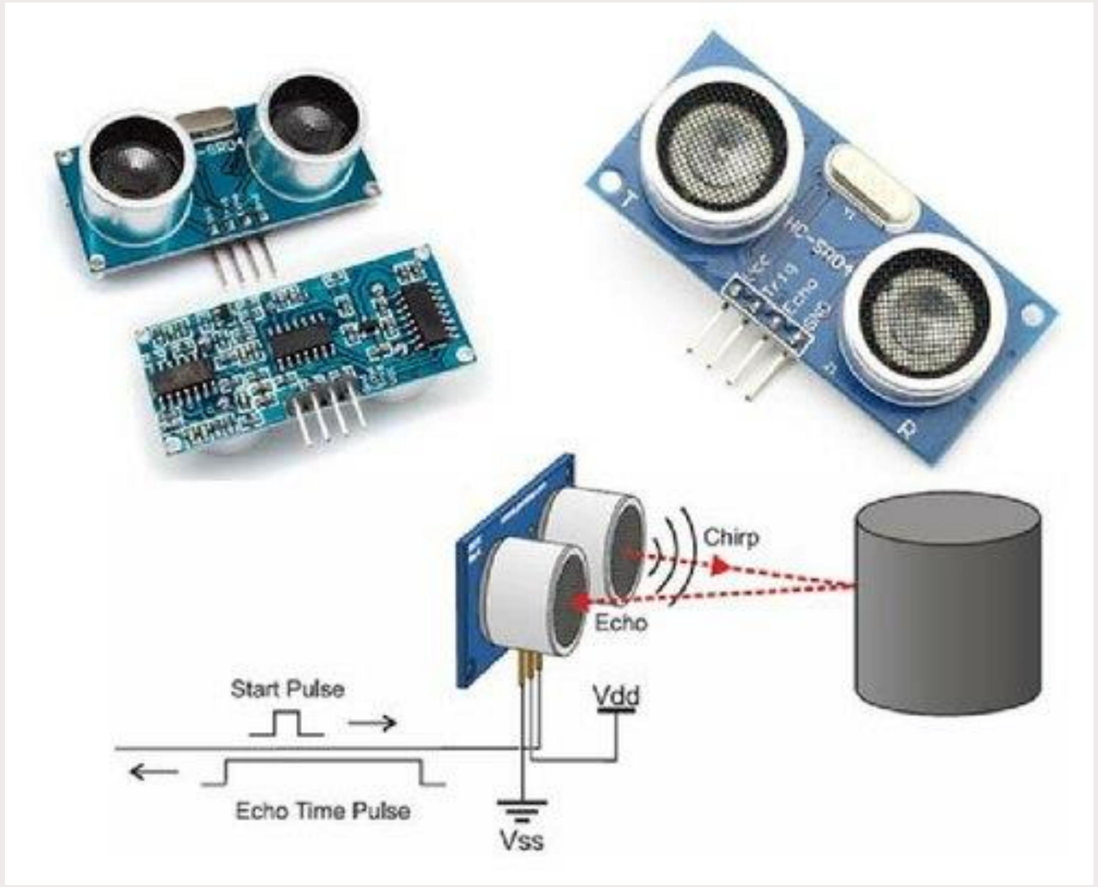


HOLOWORLD

Ultrasound sensor

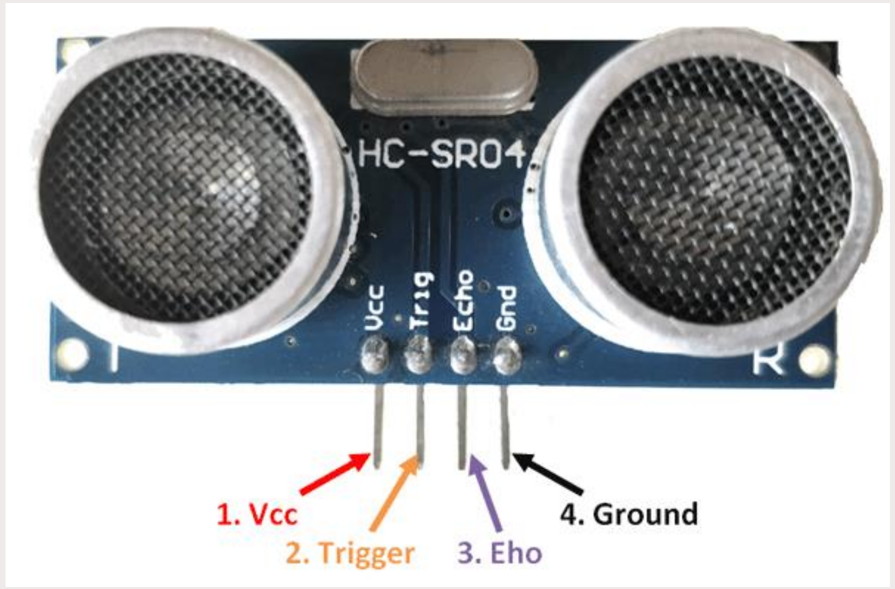
- An ultrasonic sensor can measure the distance of a target object.
- This feature is used here in this experiment to alert somebody if an object comes too close.
- For instance, it can be used to build a COVID Social Distancing Alarm.





Ultrasound sensor

Pin Configuration



Pin Number	Pin Name	Description
1	VCC	The VCC pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period which will be equal to the time taken for the US wave to return to the sensor.
4	Ground	This pin is connected to the Ground of the system.

Ultrasound sensor

How to use the HC-SR04 Ultrasonic Sensor

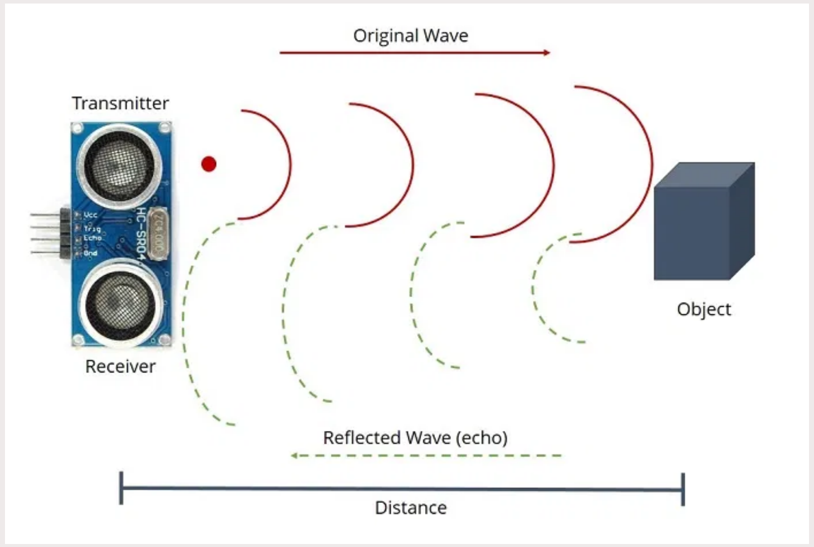
- The word 'sonic' refers to sound. Ultrasonic waves are those that travel much faster than sound.
- The sensor emits ultrasonic waves and once these are reflected back, the sensor lets us know using one of its pins going low.
- We can then measure the time difference it took for the wave to return and use this to calculate the signal.
- This is done using the formula,

  D = ½ * T *V

  where, D is the distance

  T is the time and

  V is the speed of sound (343 m/s in air).

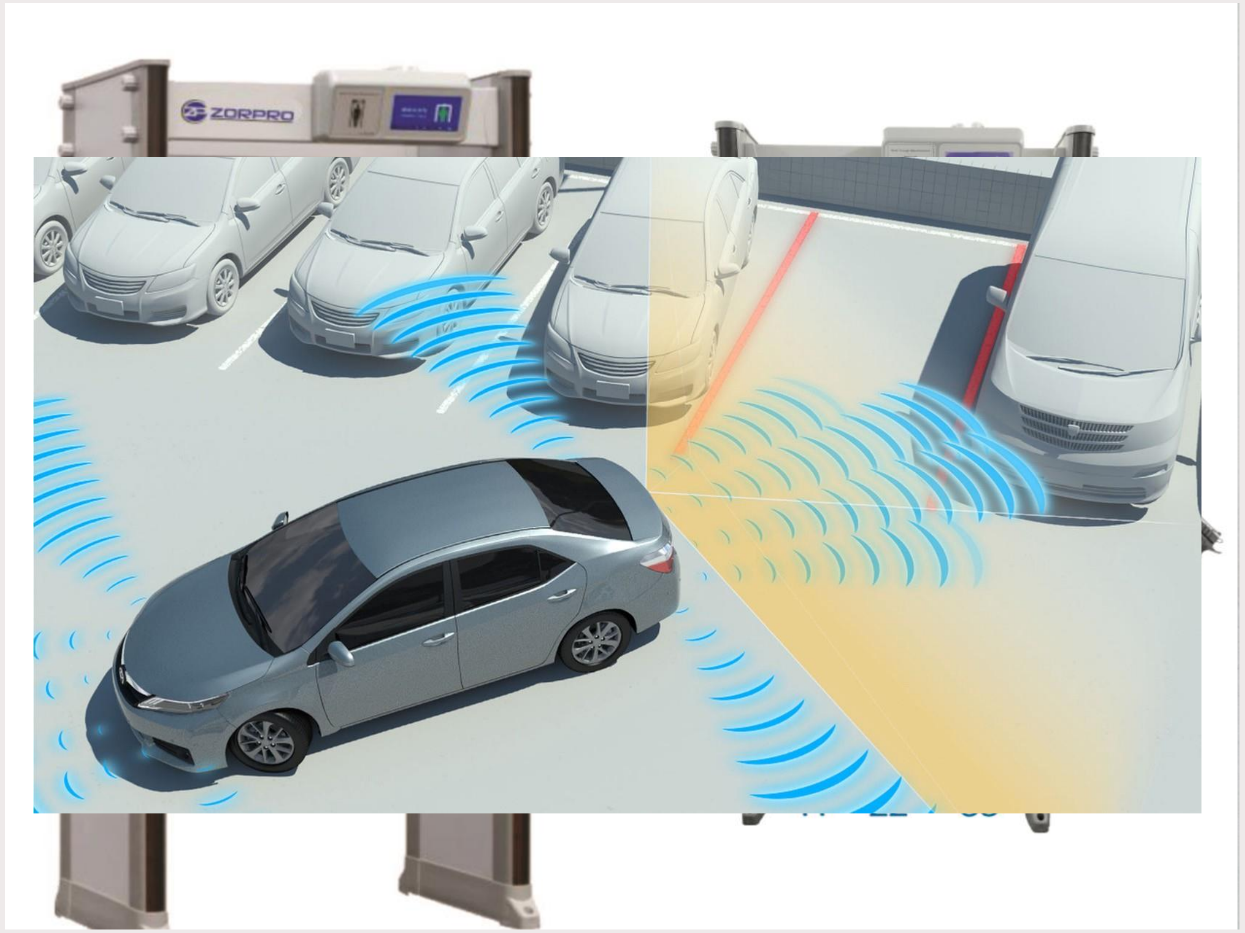


It needs to be divided by 2 because the sound waves first get emitted and then get reflected back.

Ultrasound sensor

Ultrasound Sensor Applications

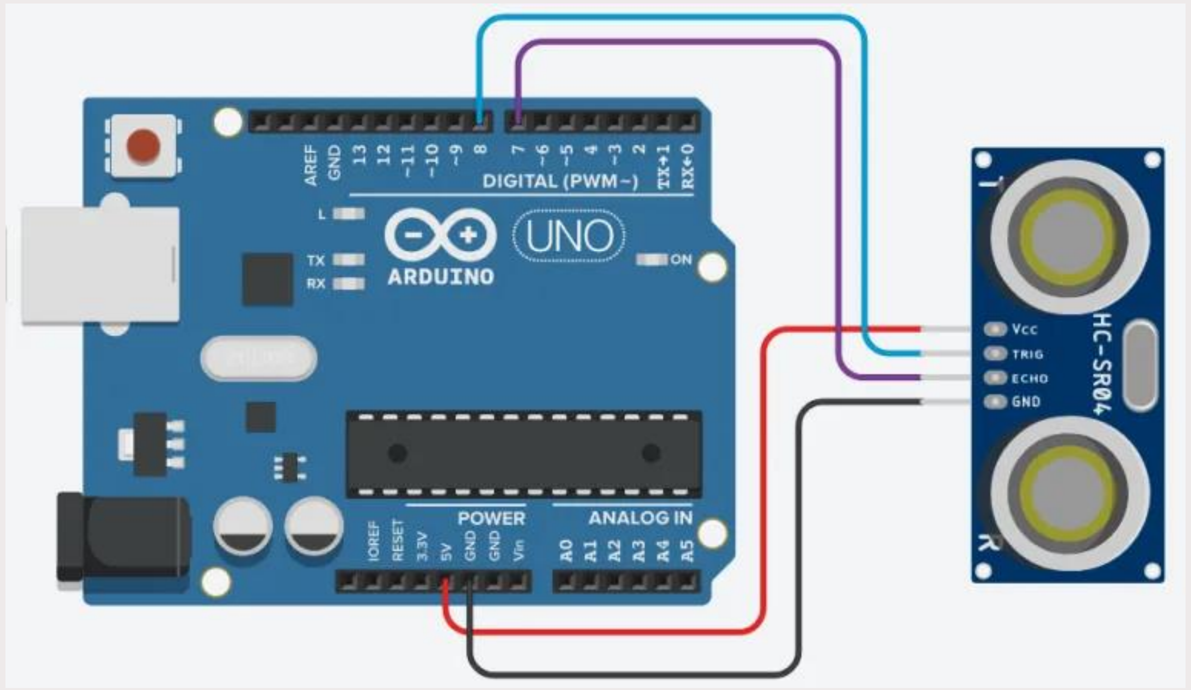
- Robotic sensing
- Liquid level control
- Counting people / people detection
- Presence detection
- Box Sorting using a Multi-Transducer System.
- Easy Control of Trash Collection Vehicles.
- Pallet Detection with Forklifts.
- Bottle Counting on Drink Filling Machines



Arduino with sensors

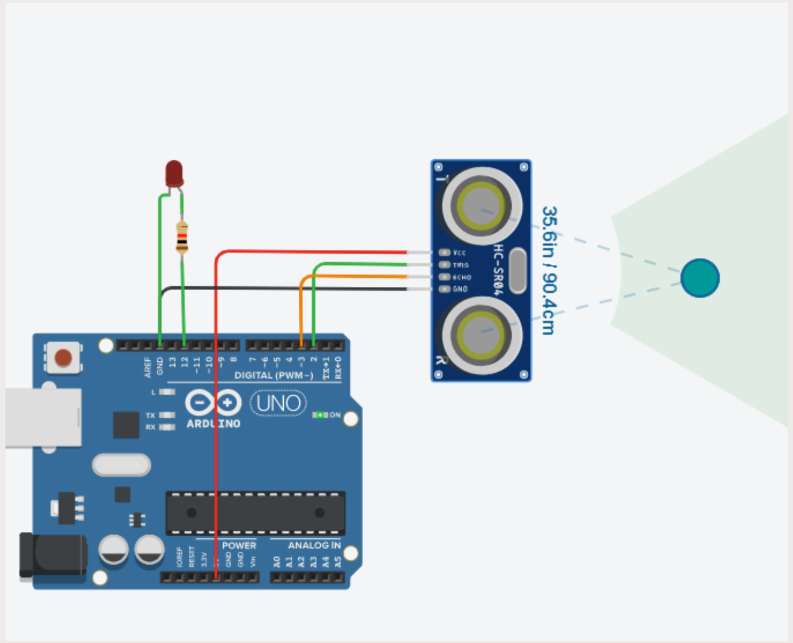
Interfacing ultrasound sensor with Arduino

- An ultrasonic sensor is an electronic component used to find the range of a targeted object by emitting ultrasonic waves (sound waves).
- This sensor mainly consists of two parts, a transducer that produces ultrasonic sound waves and another that listens for its echo.



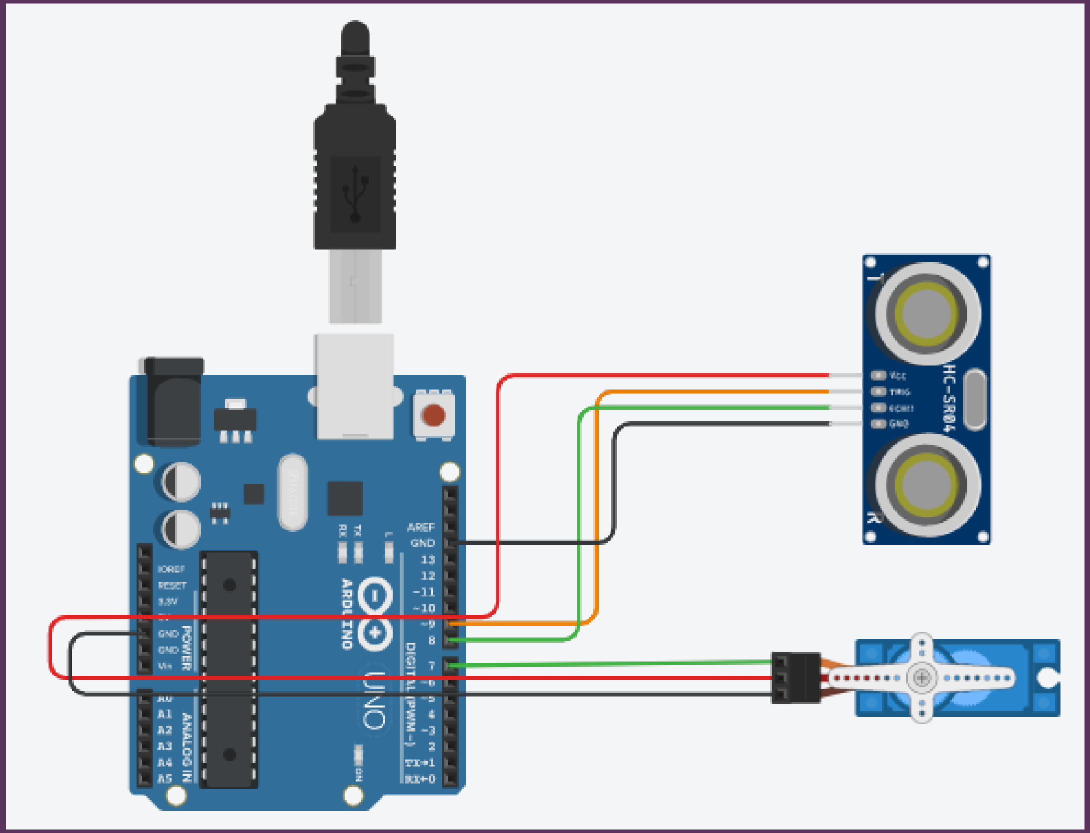
Arduino with sensors

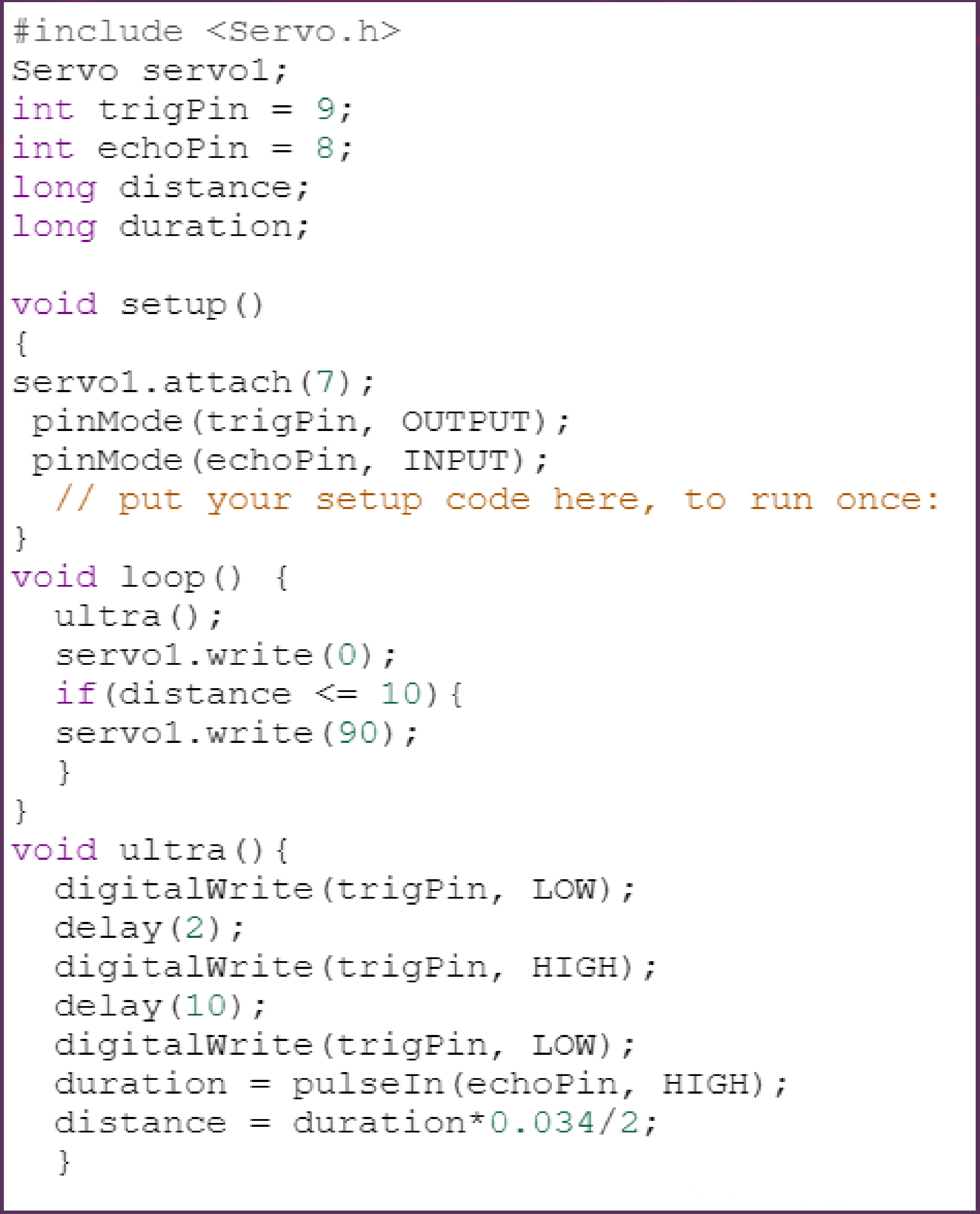
Interfacing ultrasound sensor with Arduino



```
long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);
  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  /* Reads the echo pin, and returns the
  sound wave travel time in microseconds*/
  return pulseIn(echoPin, HIGH);
}

void setup()
{
  pinMode(12, OUTPUT);
}

void loop()
{
  if (0.01723 * readUltrasonicDistance(2, 3) <= 100) {
    digitalWrite(12, HIGH);
  } else {
    digitalWrite(12, LOW);
  }
  delay(10);
  // Delay a little bit to improve simulation performance
}
```

Ultrasonic sensor with servo motor

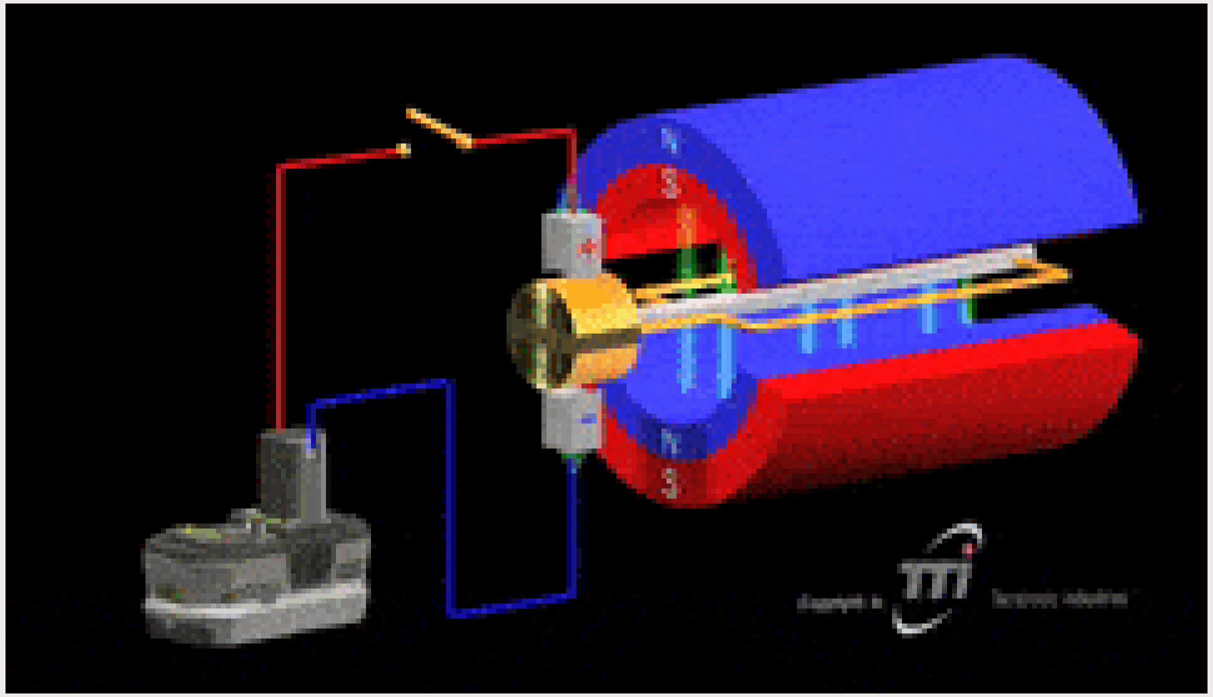




```
#include <Servo.h>
Servo servo1;
int trigPin = 9;
int echoPin = 8;
long distance;
long duration;

void setup()
{
servo1.attach(7);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
  // put your setup code here, to run once:
}
void loop() {
  ultra();
  servo1.write(0);
  if(distance <= 10){
  servo1.write(90);
  }
}
void ultra(){
  digitalWrite(trigPin, LOW);
  delay(2);
  digitalWrite(trigPin, HIGH);
  delay(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration*0.034/2;
  }
```

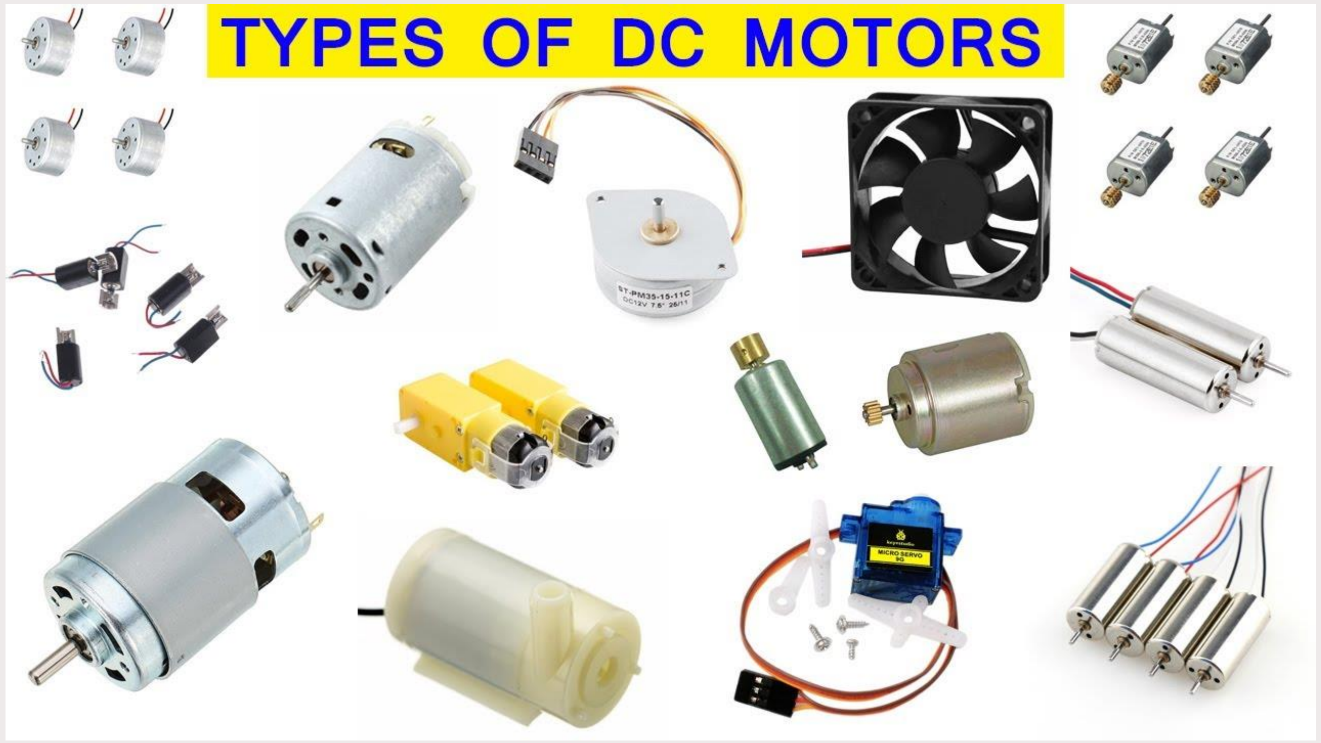
Arduino with motor using driver IC’s

- A DC Motor is a type of electric motor that converts DC electrical power to mechanical power i.e., a DC supply is converted to rotation or movement.
- DC motors are one of the commonly used motors in different applications like electronic toys, power tools, portable fans, etc.



Arduino with motor using driver IC’s

- DC Motors are further classified into different types like series, shunt and compound and each type is used in different areas of applications.
- Some DC motors are also used in Robotic and Industrial applications for their easy control and precision.

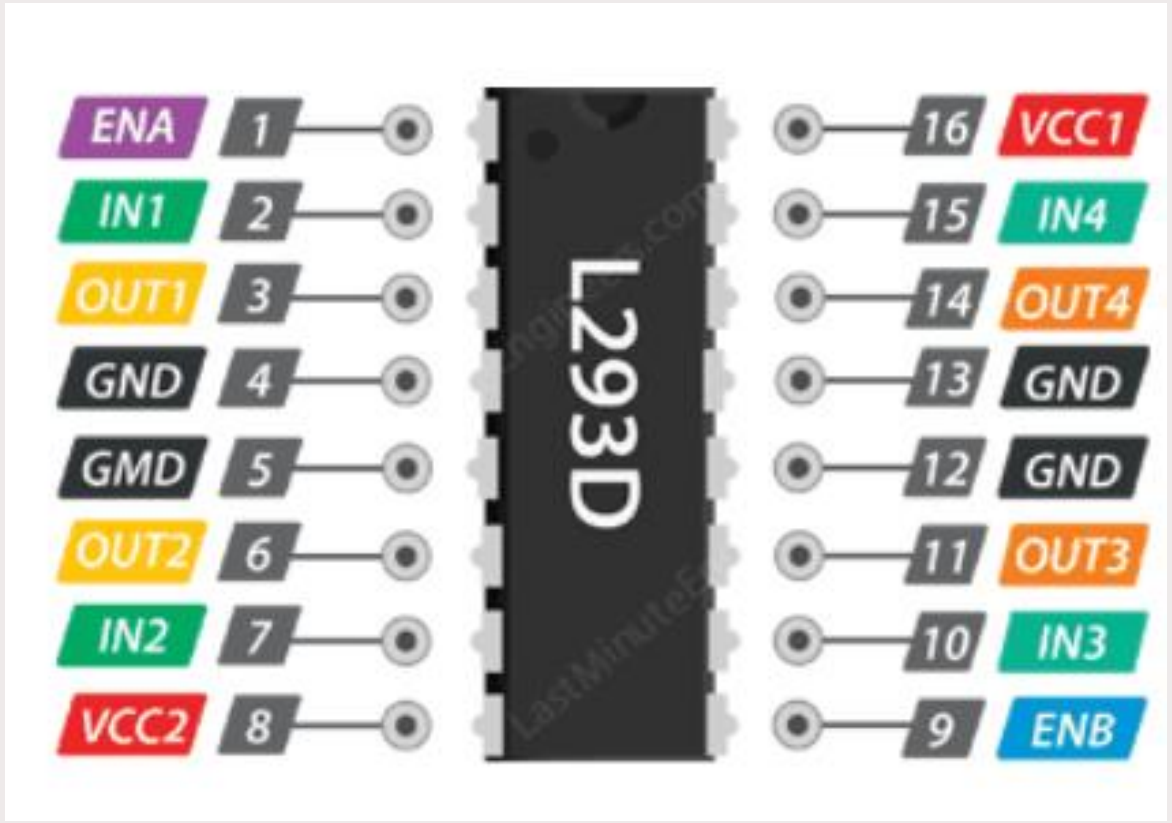


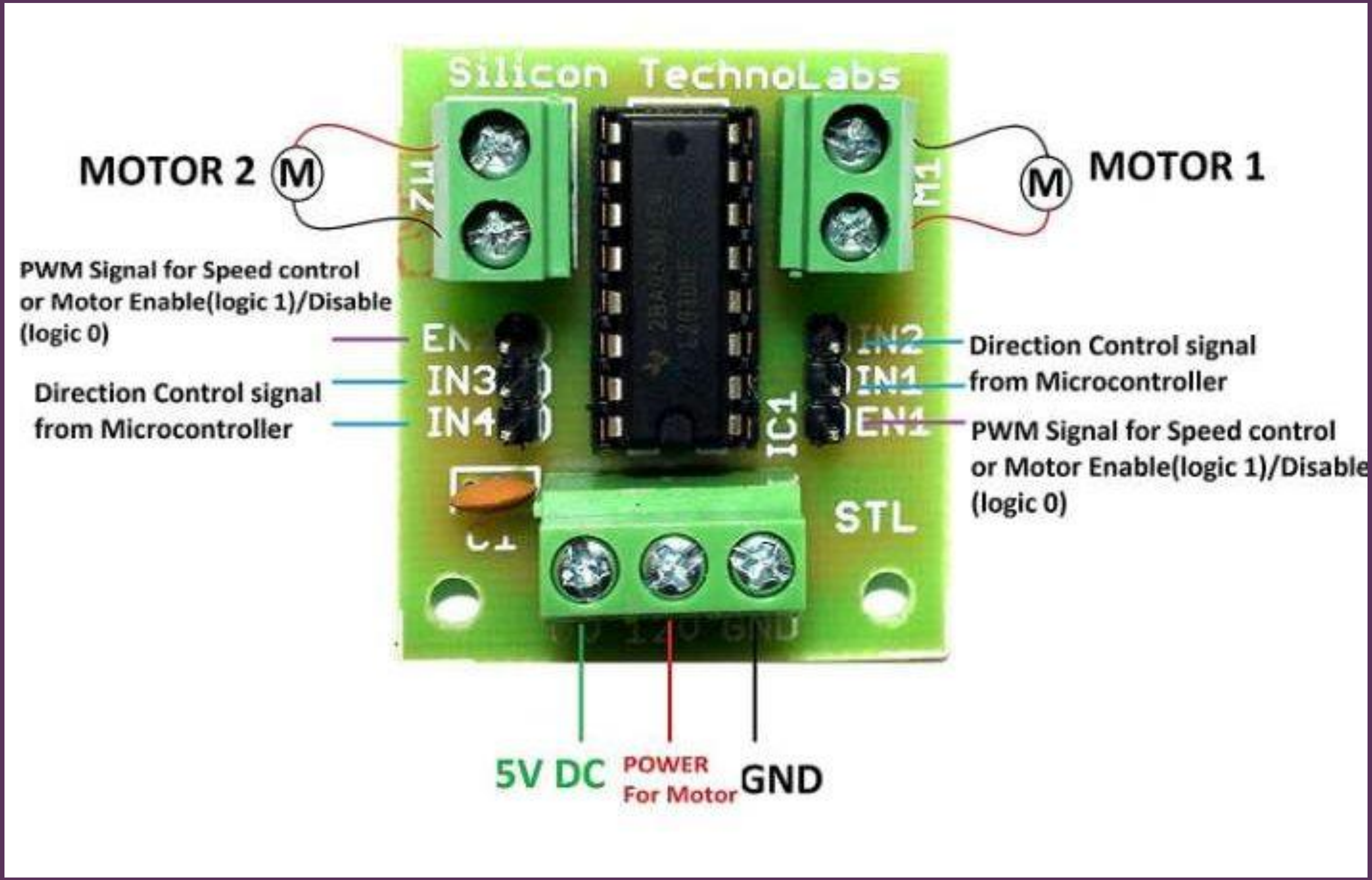
Arduino with motor using driver IC’s

L293D Motor Driver IC (H bridge)

IN1	IN2	Spinning Direction
Low(0)	Low(0)	**Motor OFF**
High(1)	Low(0)	**Forward**
Low(0)	High(1)	**Backward**
High(1)	High(1)	**Motor OFF**

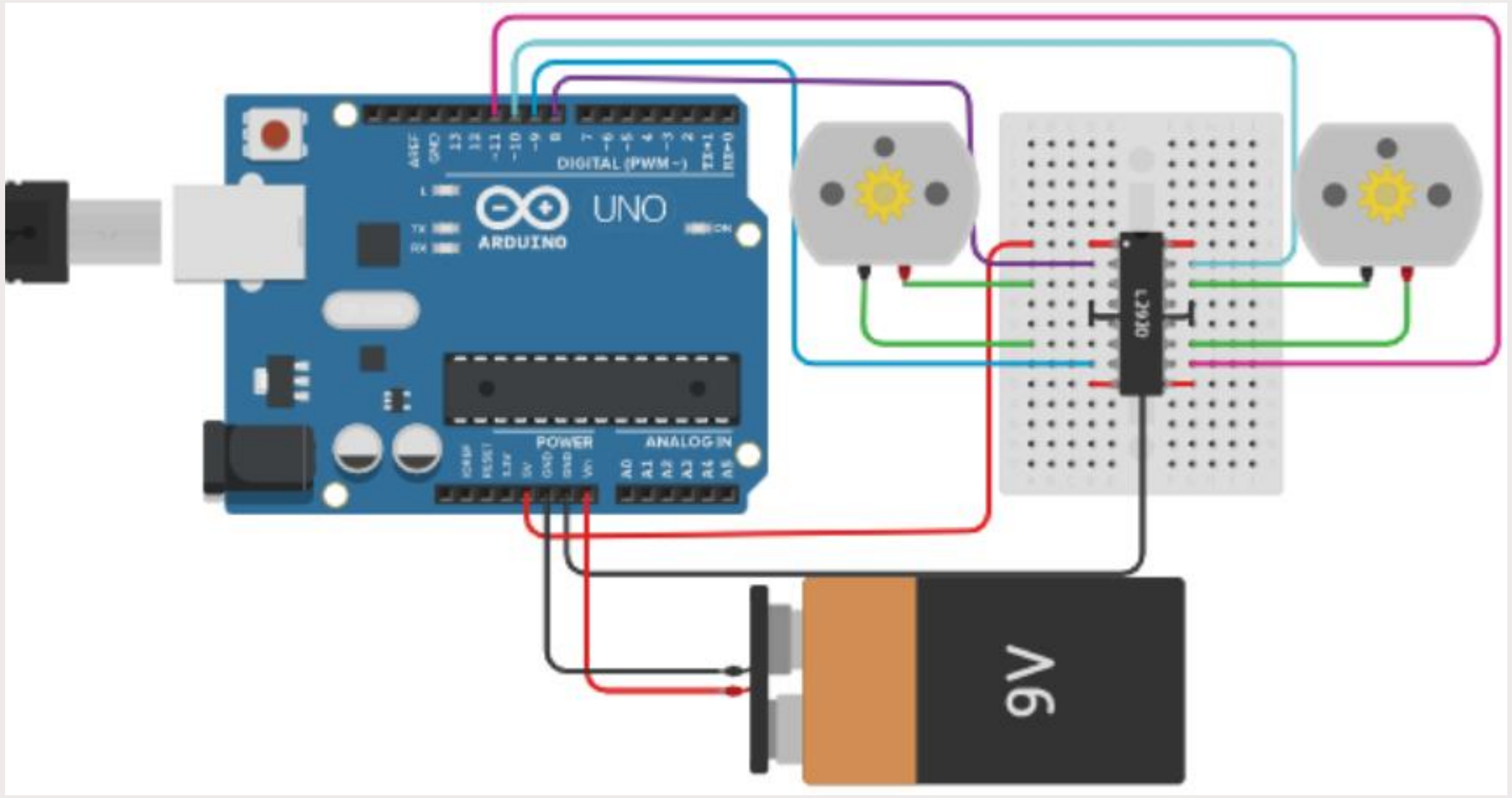
IN3	IN4	Spinning Direction
Low(0)	Low(0)	**Motor OFF**
High(1)	Low(0)	**Forward**
Low(0)	High(1)	**Backward**
High(1)	High(1)	**Motor OFF**

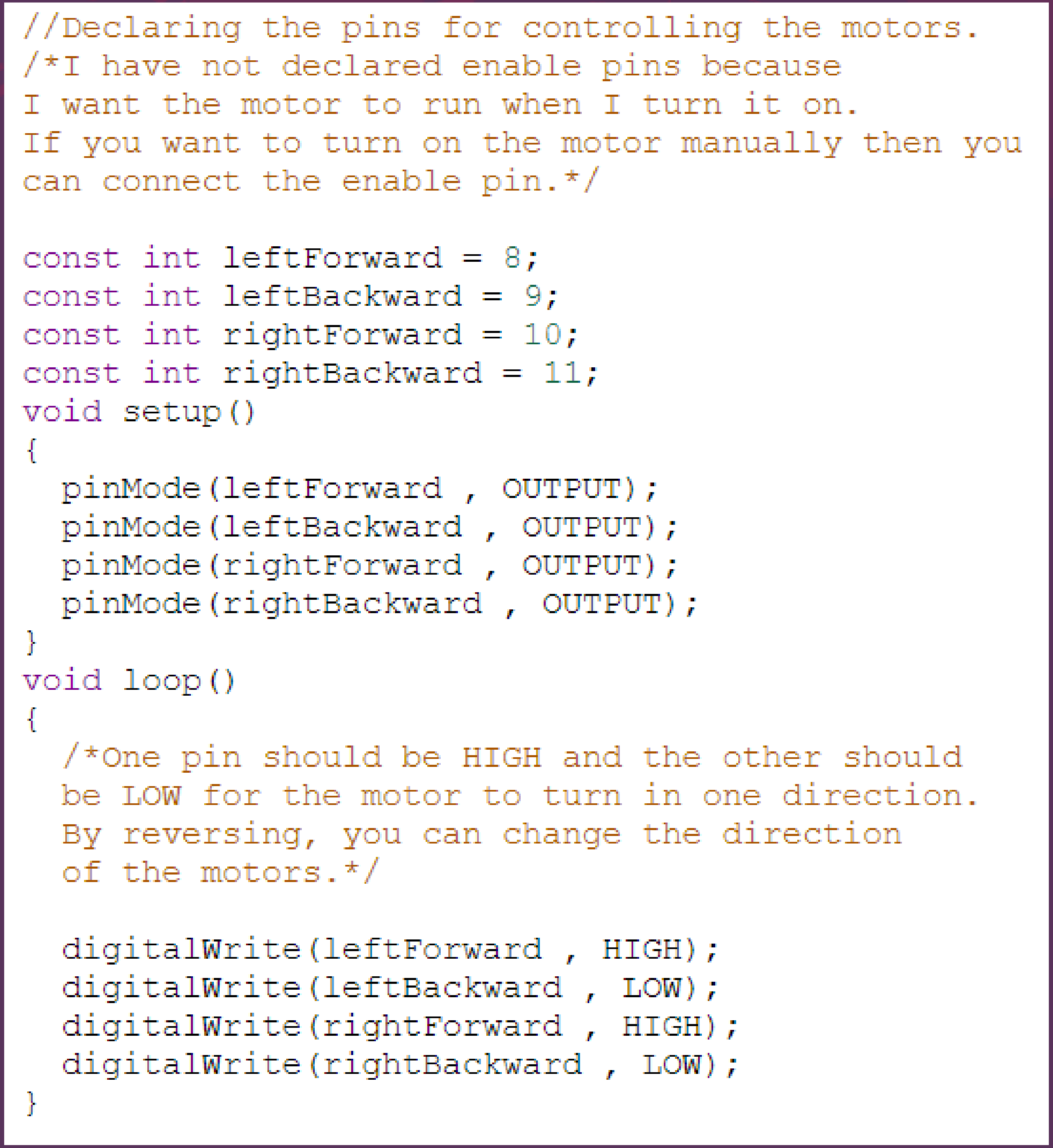



Silicon TechnoLabs
MOTOR 2
M
MOTOR 1
M
M2
M1
PWM Signal for Speed control or Motor Enable(logic 1)/Disable (logic 0)
EN2
IN3
IN4
Direction Control signal from Microcontroller
IN2
IN1
EN1
Direction Control signal from Microcontroller
PWM Signal for Speed control or Motor Enable(logic 1)/Disable (logic 0)
IC1
C1
STL
5V DC
POWER For Motor
GND

Arduino with motor using driver IC’s

Wiring L293D motor driver IC with Arduino UNO

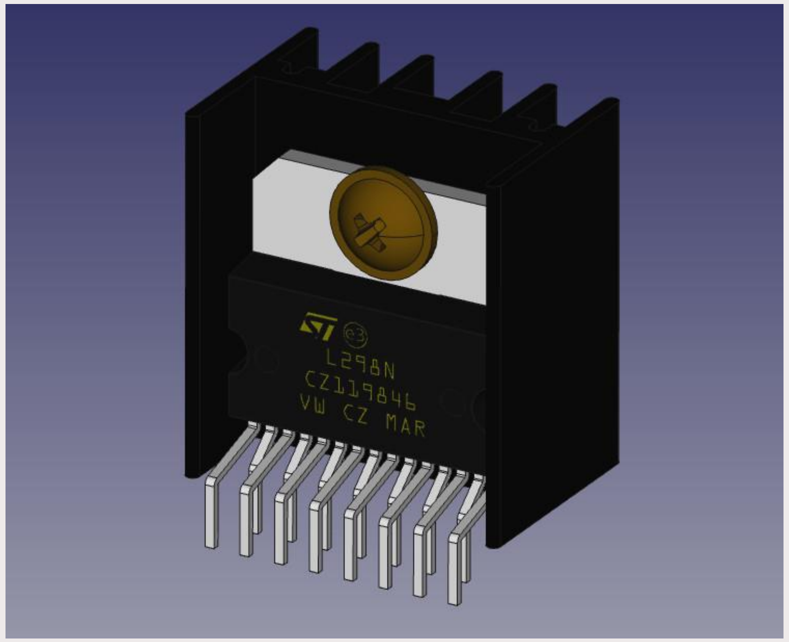


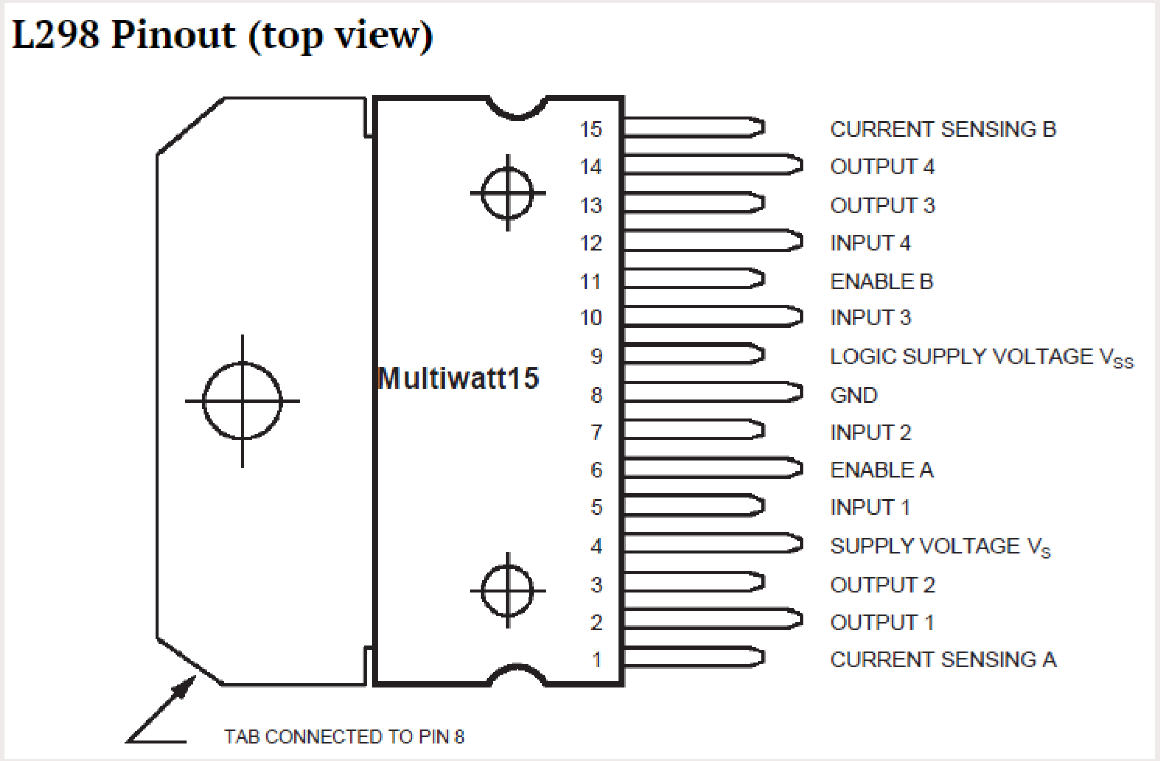


```
//Declaring the pins for controlling the motors.
/*I have not declared enable pins because
I want the motor to run when I turn it on.
If you want to turn on the motor manually then you
can connect the enable pin.*/

const int leftForward = 8;
const int leftBackward = 9;
const int rightForward = 10;
const int rightBackward = 11;
void setup()
{
  pinMode(leftForward , OUTPUT);
  pinMode(leftBackward , OUTPUT);
  pinMode(rightForward , OUTPUT);
  pinMode(rightBackward , OUTPUT);
}
void loop()
{
  /*One pin should be HIGH and the other should
  be LOW for the motor to turn in one direction.
  By reversing, you can change the direction
  of the motors.*/

  digitalWrite(leftForward , HIGH);
  digitalWrite(leftBackward , LOW);
  digitalWrite(rightForward , HIGH);
  digitalWrite(rightBackward , LOW);
}
```

L298N Motor Driver IC (H bridge)

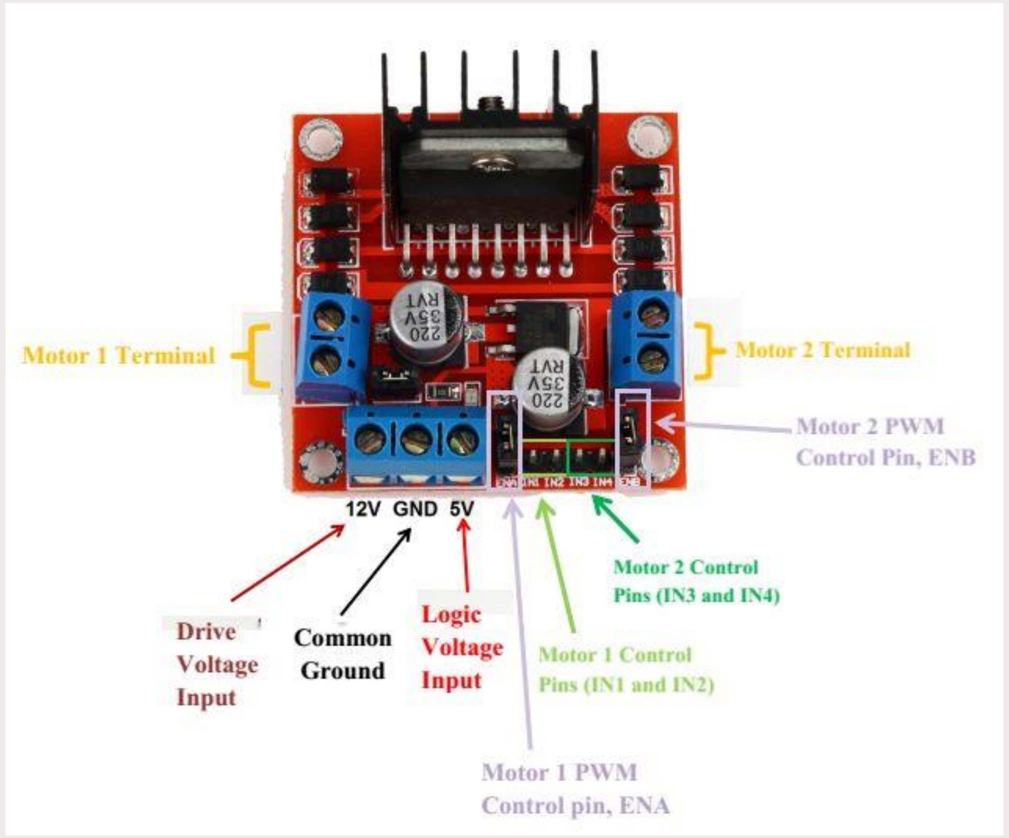
- The L298 can control the speed and direction of DC motors and stepper motors and can control two motors simultaneously
- Its current rating is 2A for each motor. At these currents, however, you will need to use heat sinks.





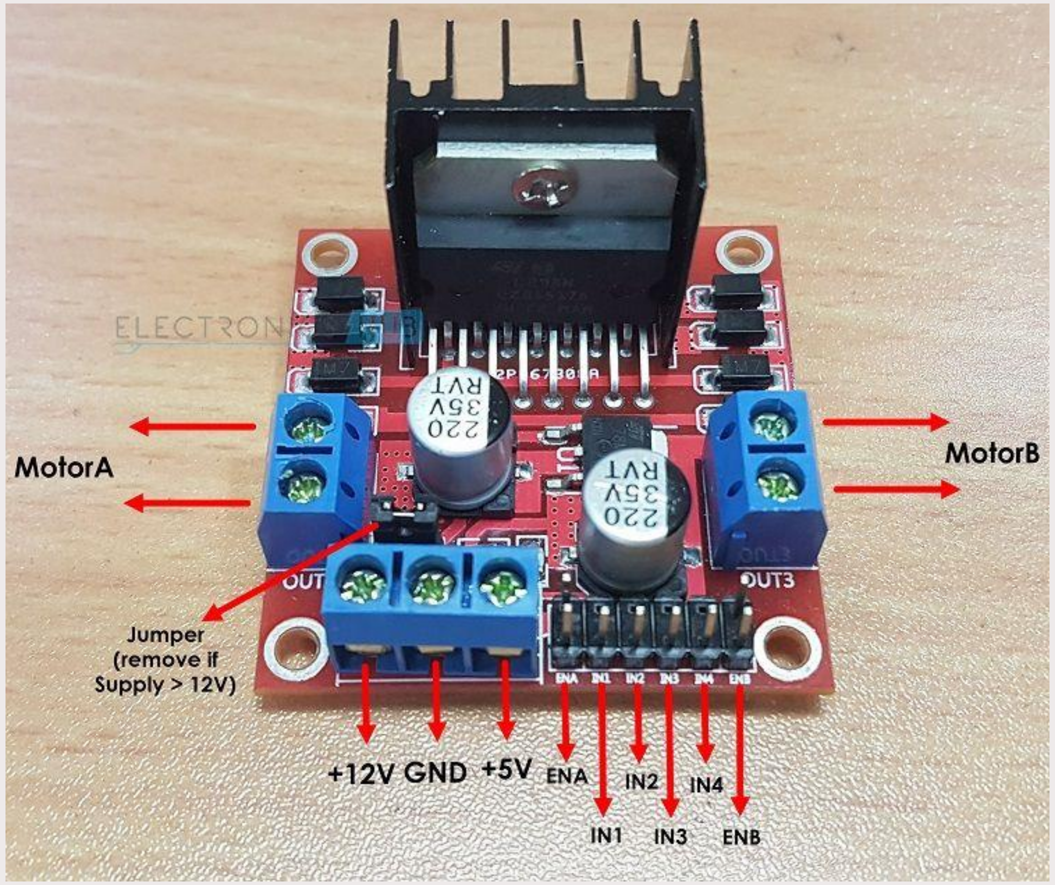
L298N Motor Driver IC (H bridge)

- The module has two screw terminal blocks for the motor 1 and 2, and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output
- We have Enable pins for Motor1 & 2 used to control the speed using PWM signals
- The module have an onboard 5V regulator which is either enabled or disabled using a jumper


Motor 1 Terminal
Motor 2 Terminal
Motor 2 PWM Control Pin, ENB
12V GND 5V
Drive Voltage Input
Common Ground
Logic Voltage Input
Motor 2 Control Pins (IN3 and IN4)
Motor 1 Control Pins (IN1 and IN2)
Motor 1 PWM Control pin, ENA

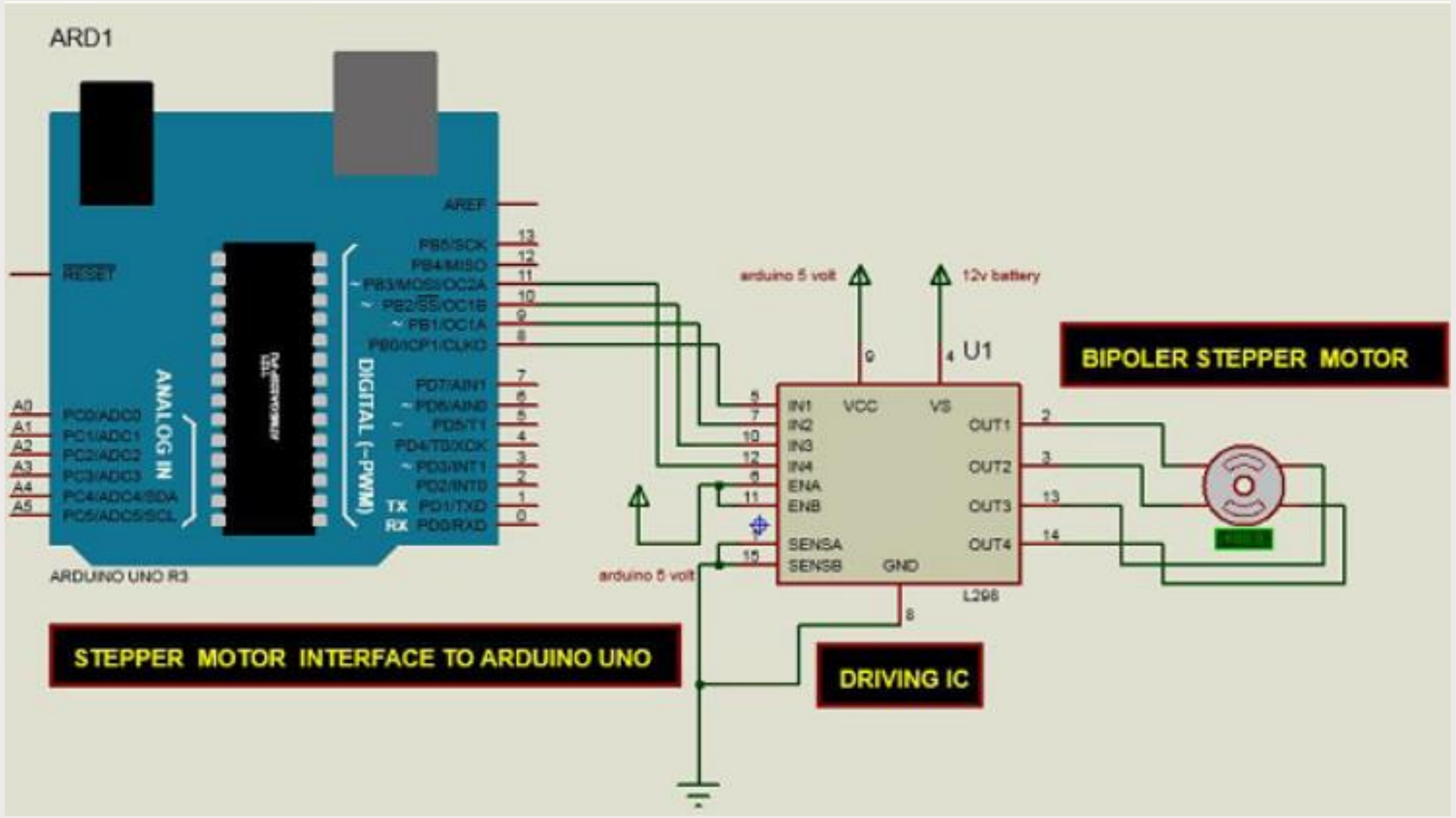
Line Tracing Robot

Motor 1		Motor 2		
IN1	IN2	IN3	IN4	Direction
0	0	0	0	Stop
1	0	1	0	Forward
0	1	0	1	Reverse
1	0	0	1	Left
0	1	1	0	Right



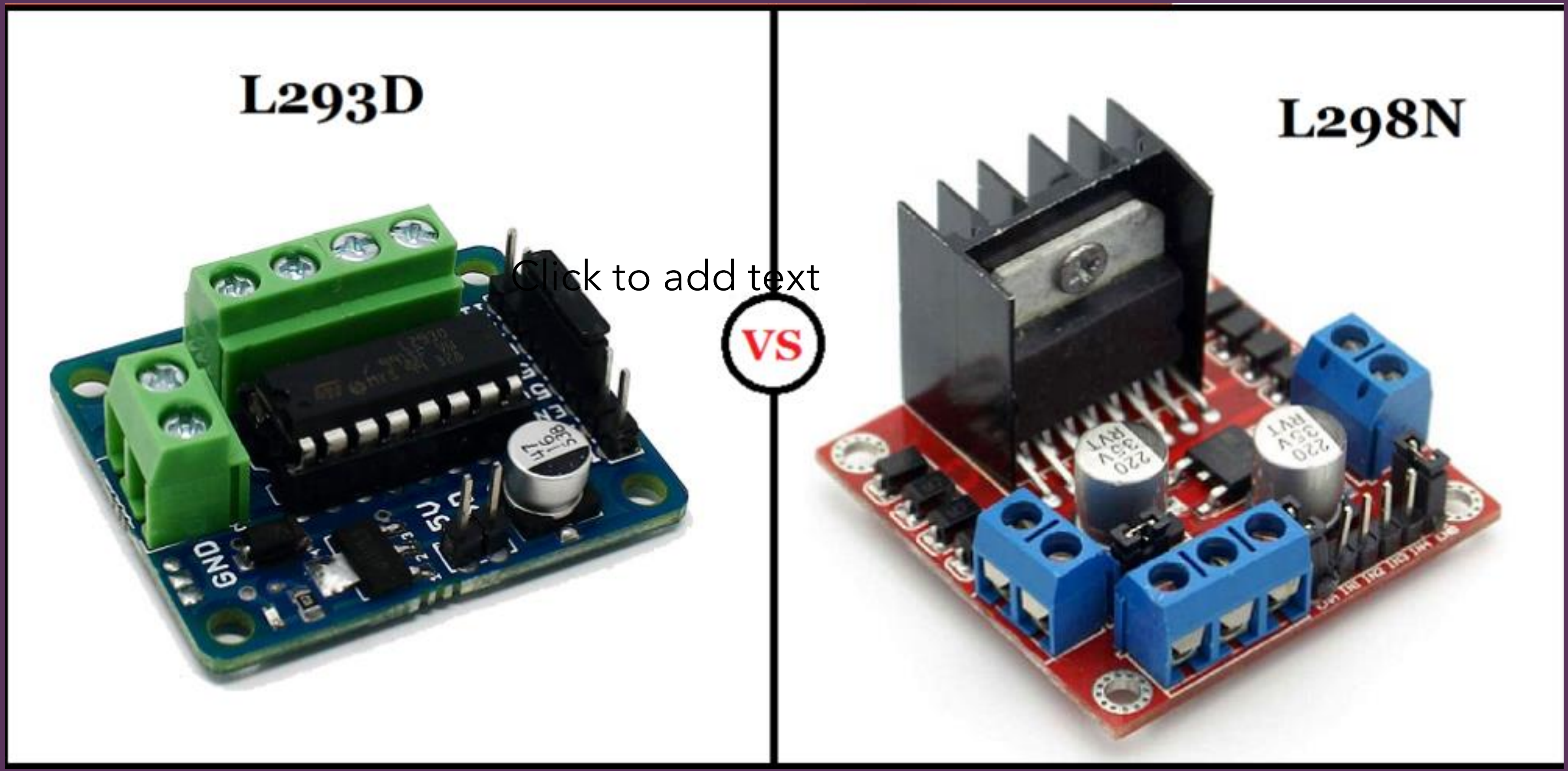
Arduino with motor using driver IC’s

L298N Motor Driver IC (H bridge)



```
#include <Stepper.h>
const int stepsPerRevolution = 90;
// change this to fit the number of steps per revolution
// for your motor
// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
   // set the speed at 60 rpm:
   myStepper.setSpeed(5);
   // initialize the serial port:
   Serial.begin(9600);
}

void loop() {
   // step one revolution in one direction:
   Serial.println("clockwise");
   myStepper.step(stepsPerRevolution);
   delay(500);
   // step one revolution in the other direction:
   Serial.println("counterclockwise");
   myStepper.step(-stepsPerRevolution);
   delay(500);
}
```

Difference Between L293D and L298N Motor Driver



Click to add text

The basic difference between L293D and L298N Motor Driver:

- L293D Drivers Operates at 4.5V to 36V whereas L298N can be Operates at up to 46V.

- Maximum 600mA Current can be drawn through both channels of L293D whereas L298N Motor Driver can draw up to 2A from both channels. Hence, the heat sink is provided in L298 motor drivers.

- L293 is a quadruple motor driver that uses a half-H driver while L298 is a dual full-H driver, i.e, in L293 all four input-output lines are independent while in L298, a half-H driver cannot be used independently, only a full H driver has to be used.

- L293D is suitable for small current drawing motors like BO motor, DC geared motors up to 500 RPM, and small stepper motors which take less current up to 600mA at their highest torque rating. WhereasL298N has the advantage of higher output current up to 2A and therefore it is suitable for high torque and high RPM motors like Johnson motors and high torque DC Geared motors.

The Arduino has two common functions **setup()** and **loop(),** which are called automatically in the background. The code to be executed is written inside the curly braces within these functions.

void setup() - It includes the initial part of the code, which is executed only once. It is called as the **preparation block**.

void loop() - It includes the statements, which are executed repeatedly. It is called the **execution block**.

But sometimes, we need to write our own functions.

Let's start writing the functions.

Function Declaration

The method to declare a function is listed below:

Function return type

We need a return type for a function. For example, we can store the return value of a function in a variable.

We can use any data type as a return type, such as **float, char**, etc.