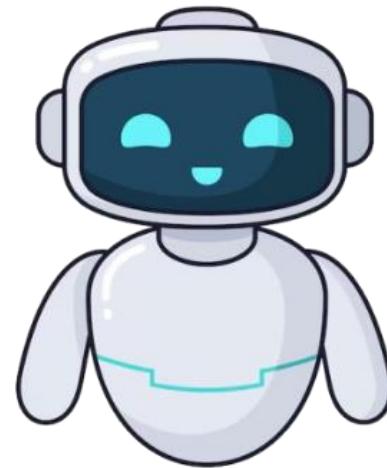


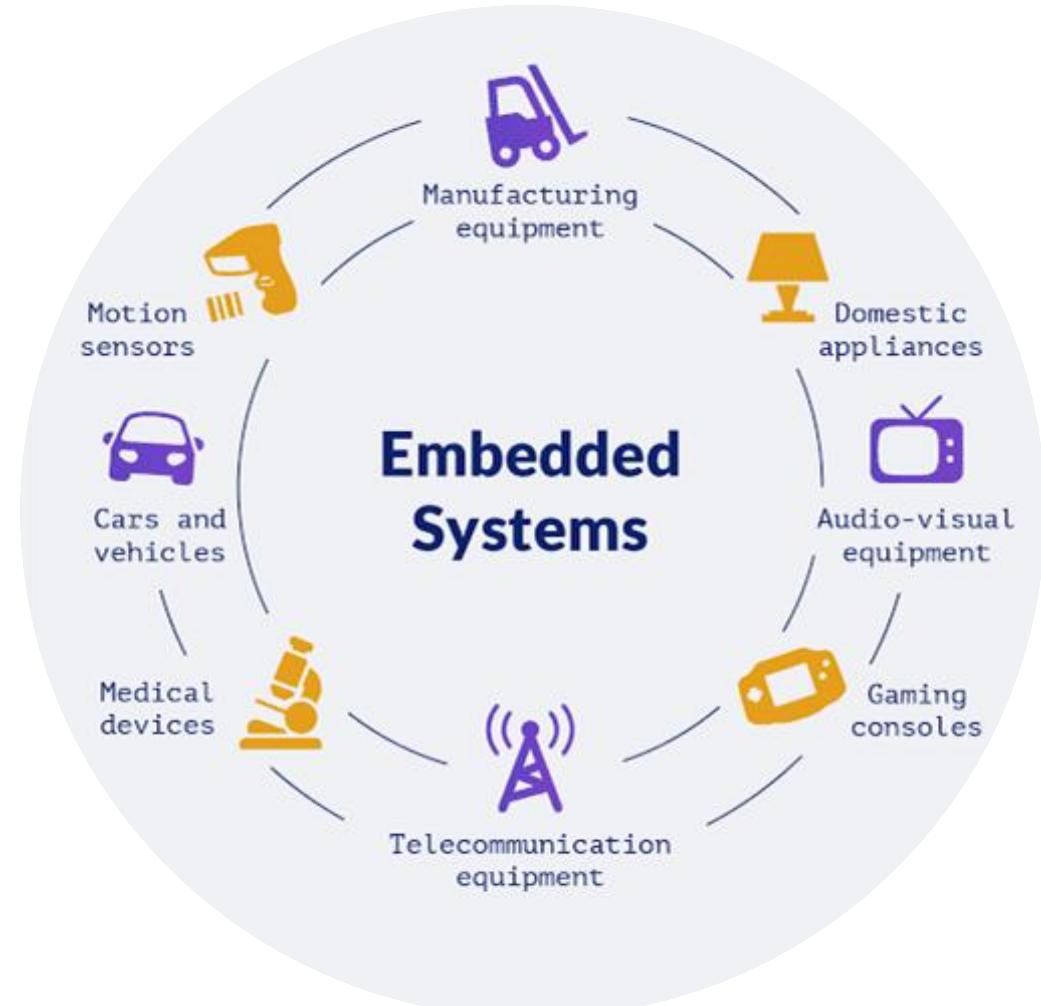
# Advanced Arduino Course



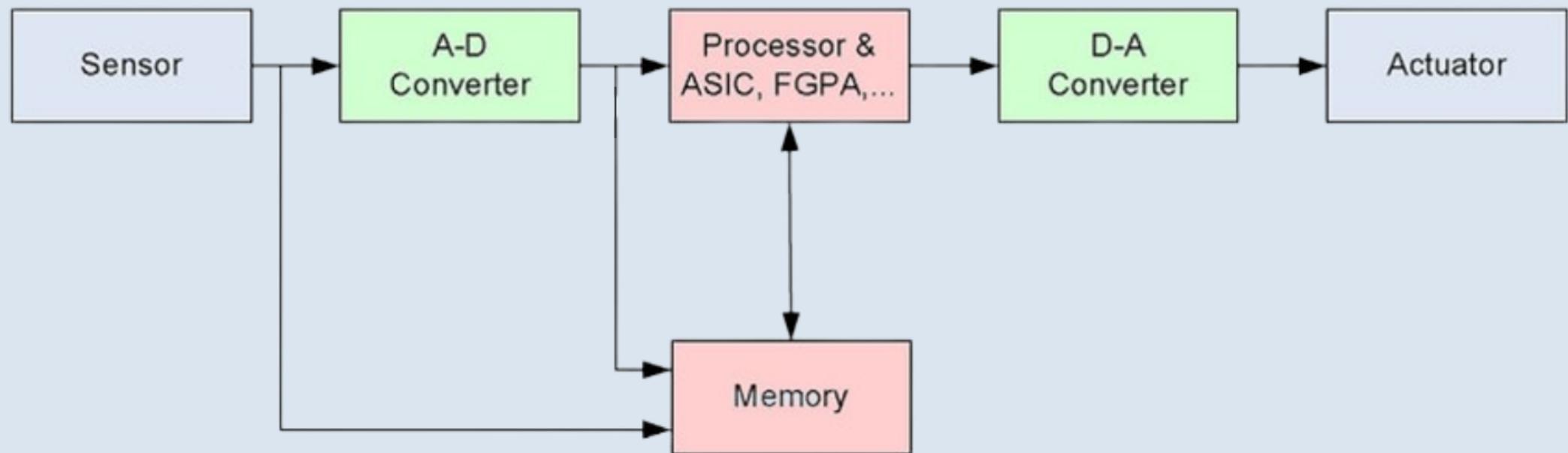
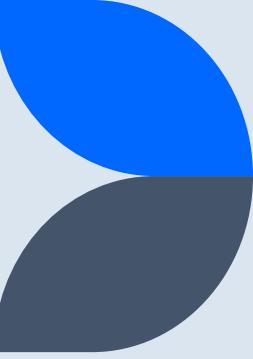
Prepared by: Qossay Abu Rida  
IEEE Robotics and Automation Society



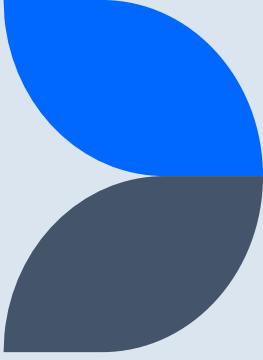
# Introduction



# Embedded systems (Symbolic-Based)



# Arduino Hardware



## Arduino Boards

- Microcontroller-based boards with various input/output capabilities.
- Examples: Arduino Uno, Arduino Mega, Arduino Nano.

## Key Components

- **Microcontroller:** The brain of the board (e.g., ATmega328 on Arduino Uno).
- **Digital Pins:** Used for digital input/output (e.g., LEDs, buttons).
- **Analog Pins:** Used for analog input (e.g., sensors).
- **Power Supply:** Can be powered via USB or external power source.
- **Communication Ports:** Serial, I2C, SPI for interfacing with other devices.

# ESP32 Hardware

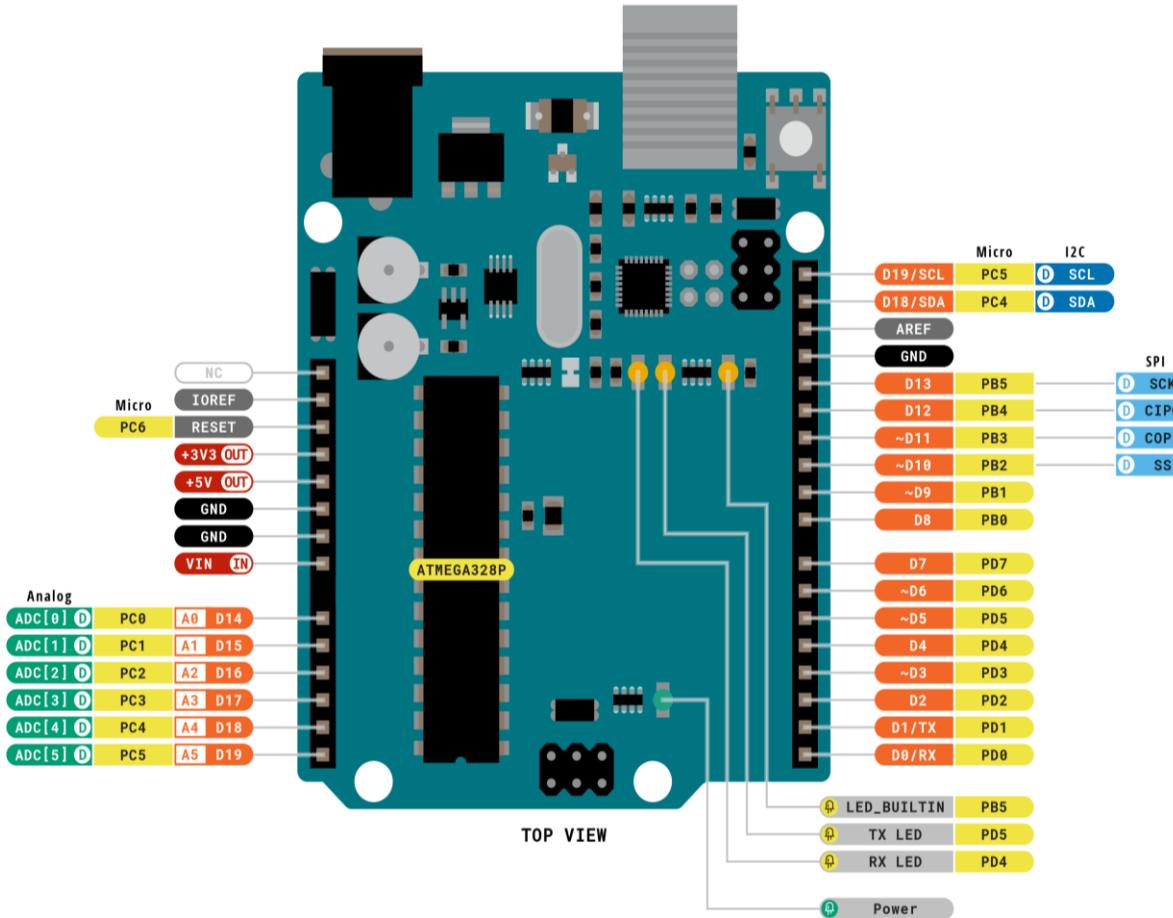
## ESP32 Overview

- A low-power Wi-Fi and Bluetooth microcontroller with various input/output capabilities.
- Developed by Espressif Systems, commonly used in IoT applications.

## Key Components

- **Microcontroller:** Dual-core Tensilica Xtensa LX6 processor with up to 240 MHz clock speed.
- **Digital Pins:** Supports multiple GPIO pins for digital input/output operations.
- **Analog Pins:** Features ADC (Analog-to-Digital Converter) and DAC (Digital-to-Analog Converter).
- **Power Supply:** Operates at 3.3V, powered via USB or external Li-ion battery.
- **Wireless Connectivity:** Built-in Wi-Fi (802.11 b/g/n) and Bluetooth (Classic & BLE) for wireless communication.
- **Communication Ports:** Supports UART, I2C, SPI, PWM, and CAN protocols for external device interfacing.

# Arduino Uno board

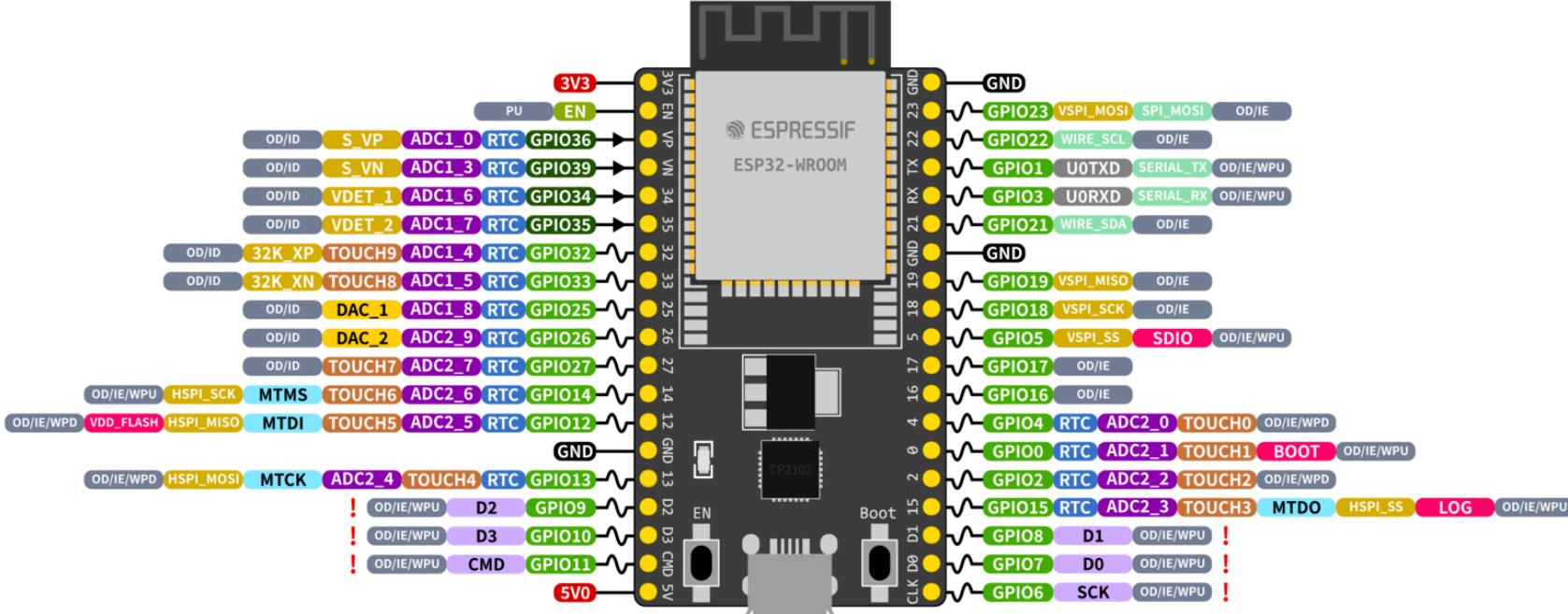


Legend:	■ Digital	■ I2C
■ Power	■ Analog	■ SPI
■ Ground	■ Main Part	■ Analog



ARDUINO UNO REV3  
SKU code: A000066  
Pinout  
Last update: 6 Oct, 2022

# ESP-32 board



ESP32 Specs

32-bit Xtensa® dual-core @240MHz  
Wi-Fi IEEE 802.11 b/g/n 2.4GHz  
Bluetooth 4.2 BR/EDR and BLE  
520 KB SRAM (16 KB for cache)  
448 KB ROM  
34 GPIOs, 4x SPI, 3x UART, 2x I2C,  
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,  
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

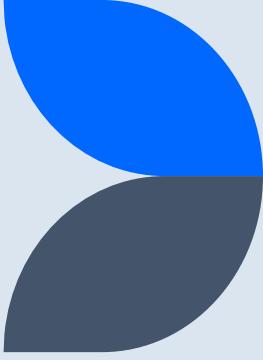


GPIO STATE

<b>RTC</b>	RTC Power Domain (VDD3P3_RTC)	<b>WPU:</b> Weak Pull-up
<b>GND</b>	Ground	<b>WPD:</b> Weak Pull-down
<b>PWD</b>	Power Rails (3V3 and 5V)	<b>PU:</b> Pull-up
	Pin Shared with the Flash Memory	<b>IE:</b> Input Enable (A)
!	<i>Can't be used as regular GPIO</i>	<b>ID:</b> Input Disabled (A)
		<b>OE:</b> Output Enable (A)
		<b>OD:</b> Output Disabled (A)

# Arduino Software

(use for Arduino UNO & ESP32)



## Arduino IDE

- Integrated Development Environment for writing, compiling, and uploading code to Arduino boards.
- Available for Windows, Mac, and Linux.

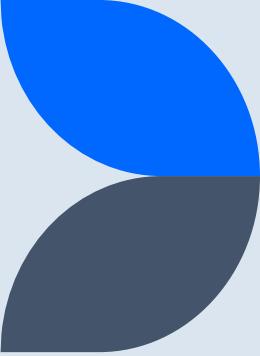
## Key Features

- **Code Editor:** Write and edit Arduino sketches (programs).
- **Compiler:** Convert sketches into machine code that the microcontroller can execute.
- **Uploader:** Transfer the compiled code to the Arduino board via USB.
- **Serial Monitor:** Communicate with the board and debug sketches using serial communication.

## Programming Language

- Based on C/C++ with simplified syntax and functions.
- Uses libraries to add functionality (e.g., controlling motors, reading sensors).

# How to Program Arduino & ESP 32



## Setup Function:

- **Purpose:** Initializes settings.
- **Runs Once:** This code runs once when the Arduino is powered on or reset.

## Loop Function:

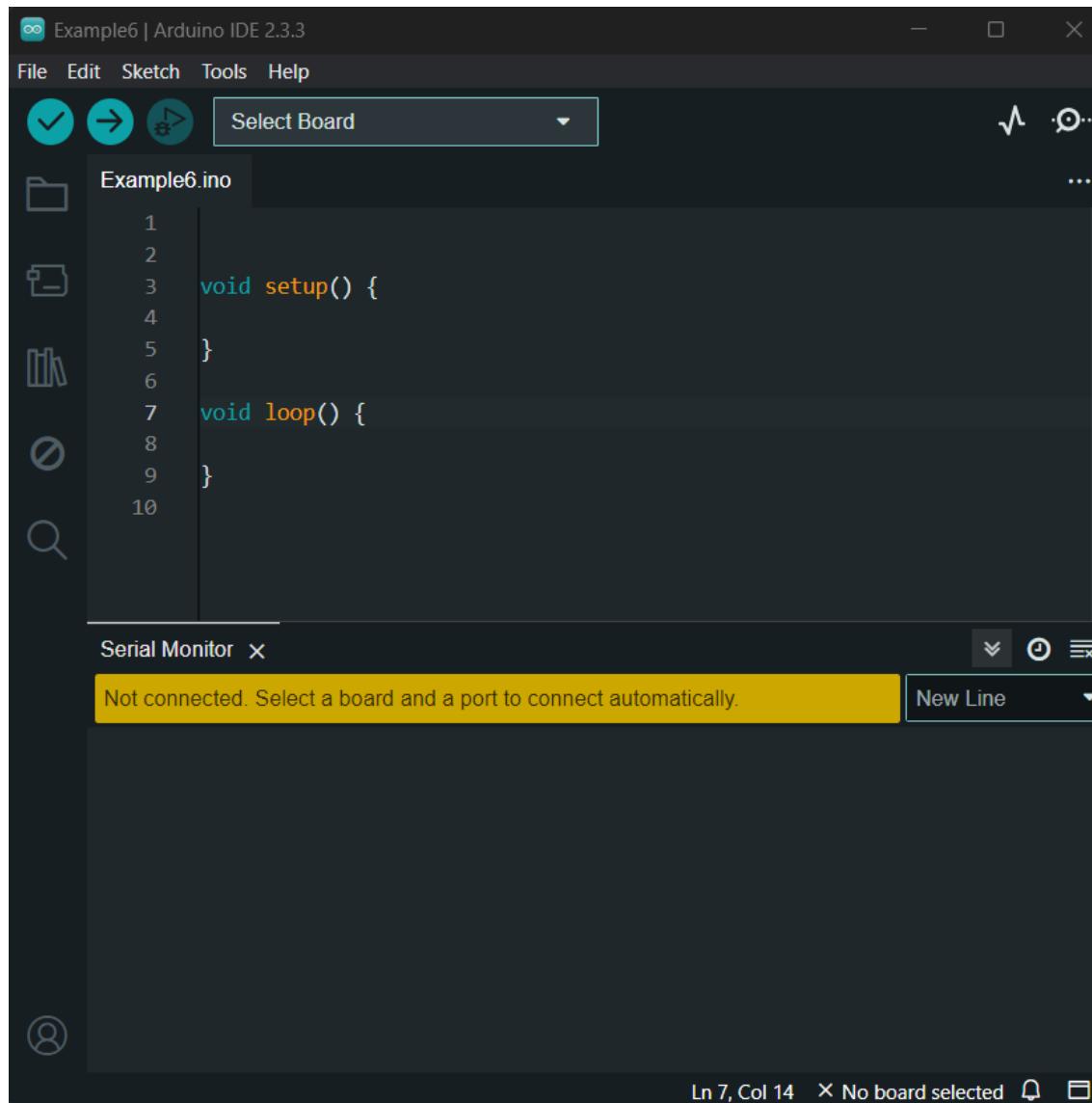
- **Purpose:** Contains the main logic of the program.
- **Runs Repeatedly:** This code runs in a loop after the setup function has completed.



```
void setup() {  
    // Initialization code  
}
```

```
void loop() {  
    // Main code  
}
```

# Arduino IDE simple sketch.



# Common Functions Used in Arduino Programming

## **pinMode()**:

- **Purpose:** Configures a specified pin to behave either as an input or an output.

## **digitalWrite()**:

- **Purpose:** Sets a specified digital pin to HIGH or LOW.

## **digitalRead()**:

- **Purpose:** Reads the value from a specified digital pin, either HIGH or LOW.

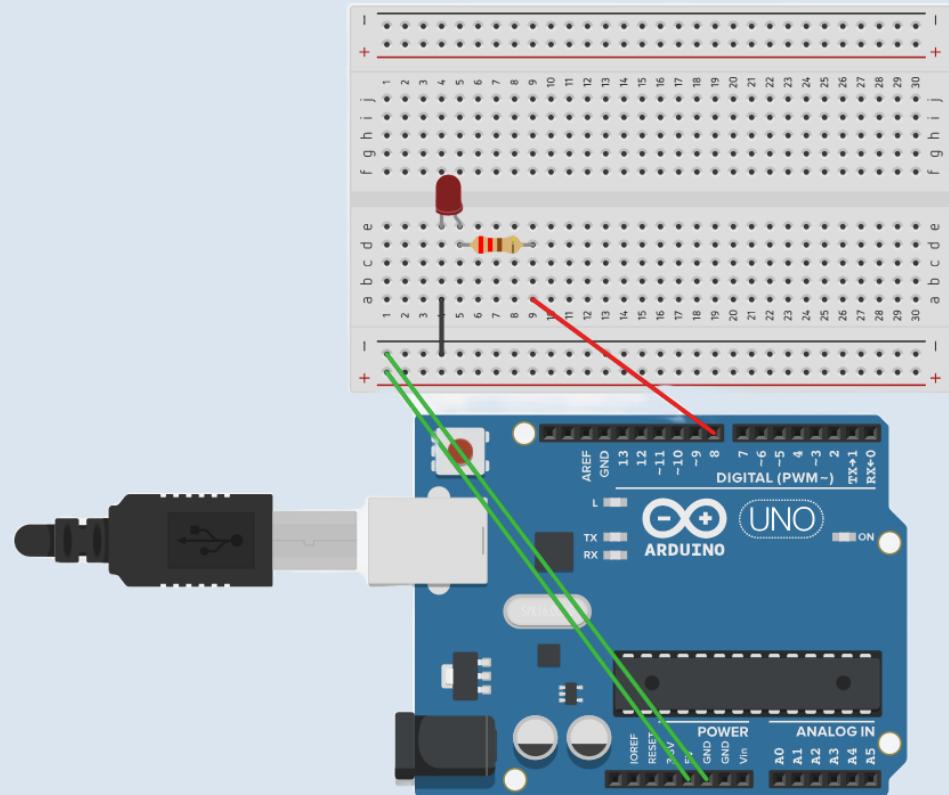


```
pinMode(pin, mode);
```

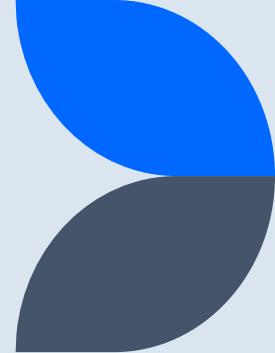
```
digitalWrite(pin, value);
```

```
int value = digitalRead(pin);
```

# Example:



# Common Functions Used in Arduino Programming



## analogRead():

- **Purpose:** Reads the value from a specified analog pin.

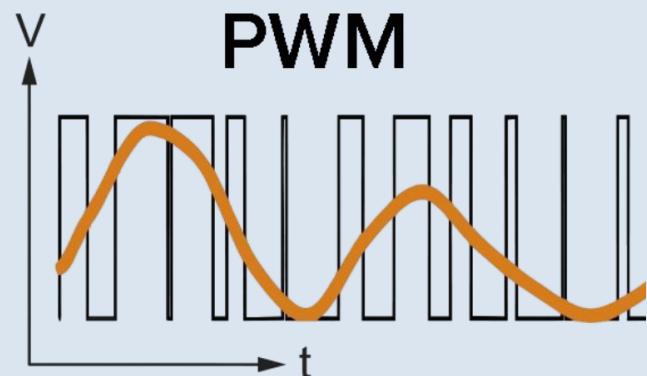
## analogWrite():

- **Purpose:** Writes an analog value (PWM wave) to a specified pin.

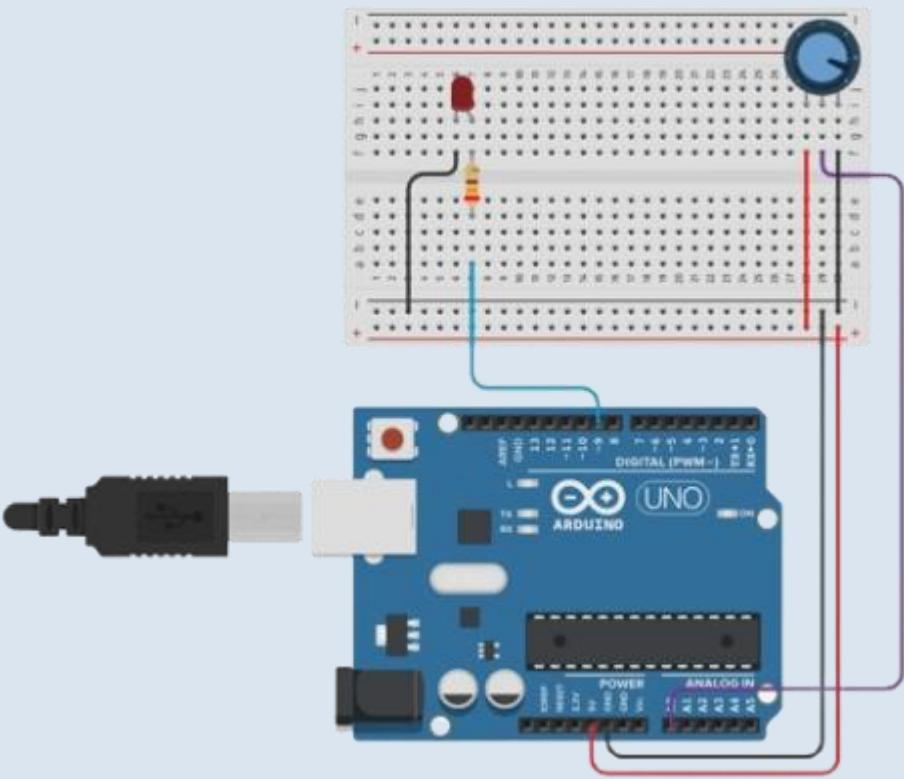


```
int value = analogRead(pin);
```

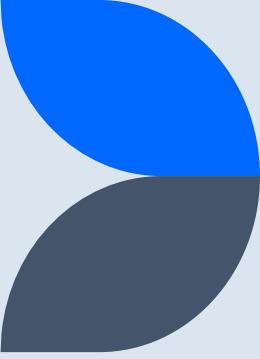
```
analogWrite(pin, value);
```



# Example:



# Common Functions Used in Arduino Programming



## delay():

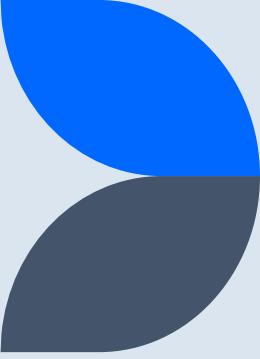
- **Purpose:** Pauses the program for a specified number of milliseconds.

## millis():

- **Purpose:** Returns the number of milliseconds since the Arduino board began running the current program.

```
delay(milliseconds);  
unsigned long currentTime = millis();
```

# Common Functions Used in Arduino Programming



## **Serial.begin():**

- **Purpose:** Sets the data rate in bits per second (baud) for serial data transmission.

## **Serial.print():**

- **Purpose:** Prints data to the serial port as human-readable ASCII text.

## **Serial.println():**

- **Purpose:** Prints data to the serial port as human-readable ASCII text, followed by a carriage return character.

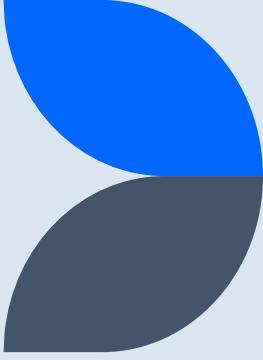


```
Serial.begin(baudRate);
```

```
Serial.print(data);
```

```
Serial.println(data);
```

# Interrupts



## Definition

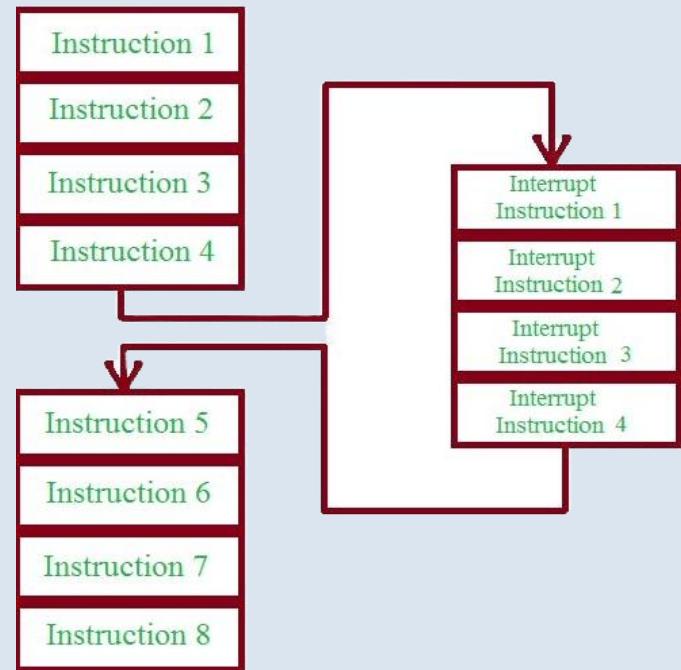
- Interrupts are signals that temporarily halt the main program execution to immediately run a special function called an Interrupt Service Routine (ISR).

## Purpose

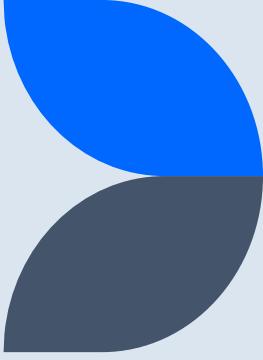
- To handle time-critical tasks, respond to external events, or process real-time signals without delay.

## Benefits

- Allows for efficient handling of tasks without continuously checking for events within the main loop.



# Using Interrupts in Arduino



## Attaching an Interrupt:

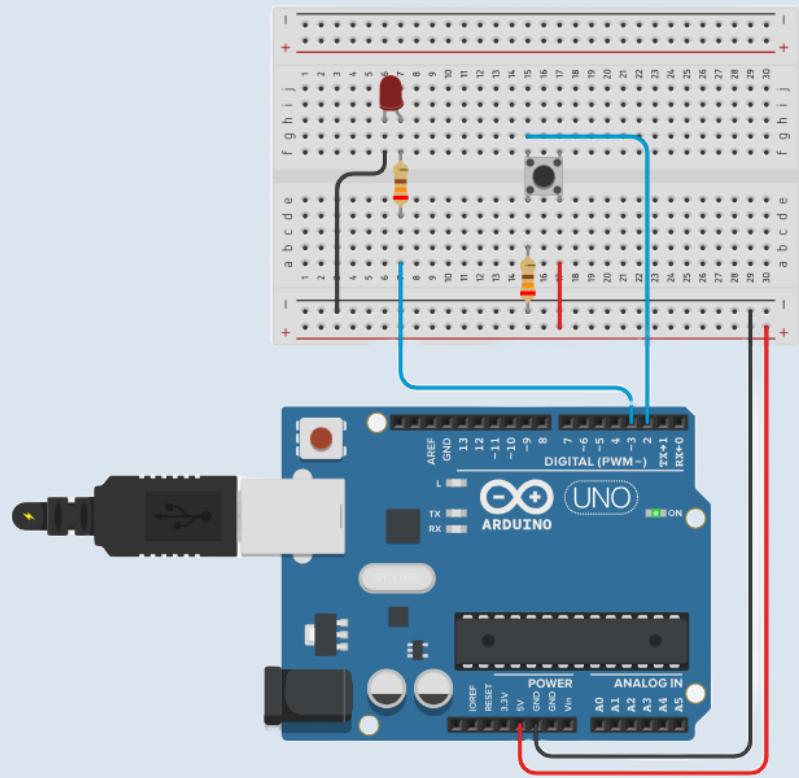
- Use the attachInterrupt() function to specify which pin and condition will trigger the interrupt, and which ISR to call.

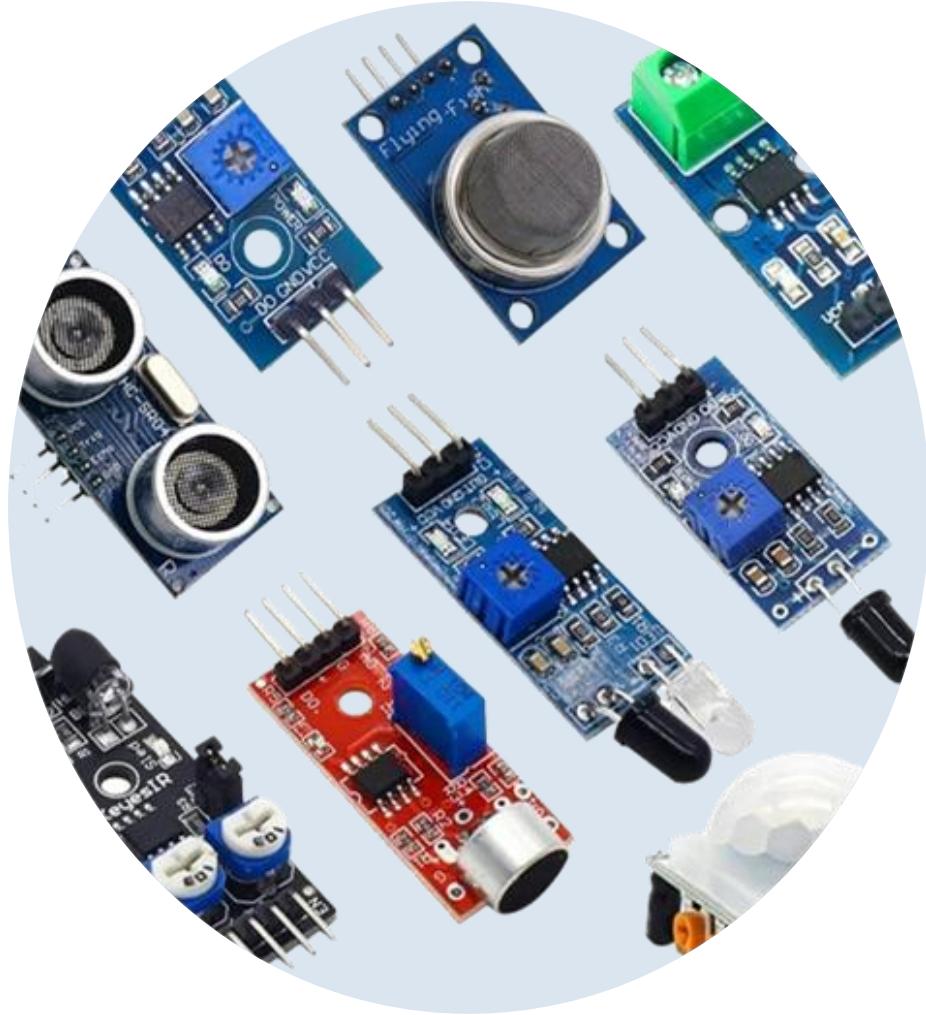


```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

- Parameters:
  - pin: The pin number where the interrupt is attached.
  - ISR: The name of the function to call when the interrupt occurs.
  - mode: The condition that triggers the interrupt (LOW, CHANGE, RISING, FALLING).

# Example:





# Sensors

# Ultrasonic Sensor

## Introduction:

- **Definition:** Ultrasonic sensors use sound waves to measure the distance between the sensor and an object.
- **Common Model:** HC-SR04.

## Working Principle:

- **Sound Waves:** Sends out an ultrasonic wave and measures the time it takes for the echo to return.
- **Distance Calculation:** Uses the speed of sound to calculate distance based on the time it took for the echo to return.



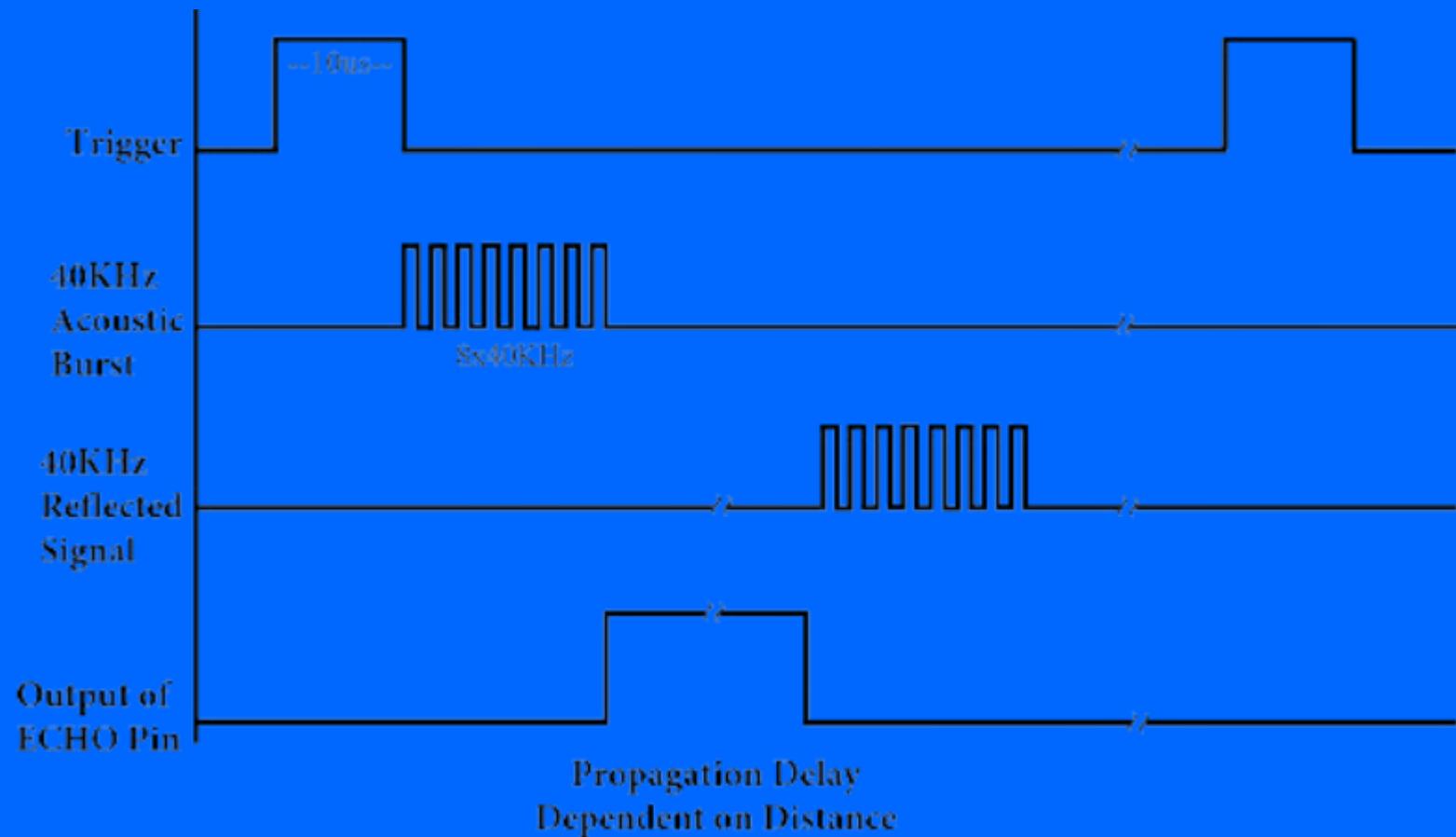
# Pin Configuration

**VCC:** +5V Power Supply

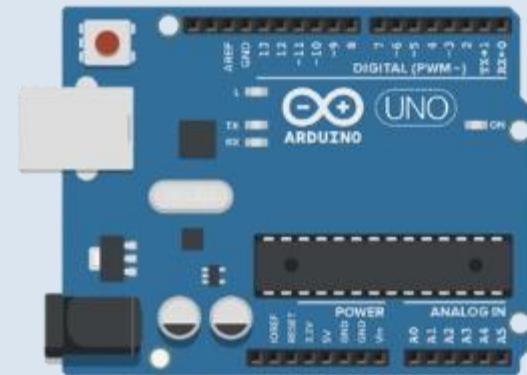
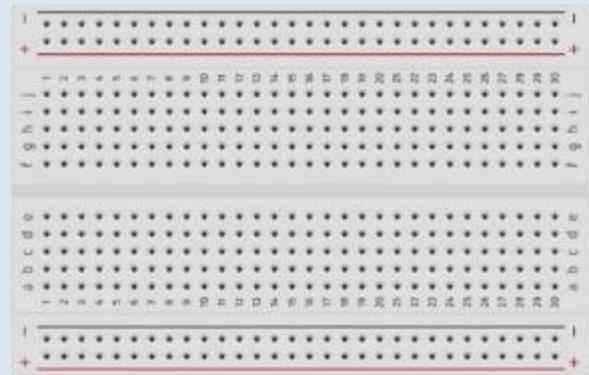
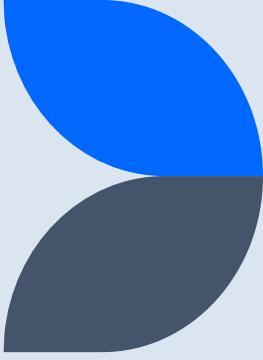
**GND:** Ground

**Trig:** Trigger Pin

**Echo:** Echo Pin



# Example:



# IR Sensor

## Introduction:

- **Definition:** Infrared (IR) sensors use infrared light to detect objects or measure the distance between the sensor and an object.
- **Common Model:** TCRT5000.

## Working Principle:

- **Infrared Light:** Emits an infrared light beam and detects the reflection from an object.
- **Distance Calculation:** Measures the intensity of the reflected light to determine the presence or distance of an object.

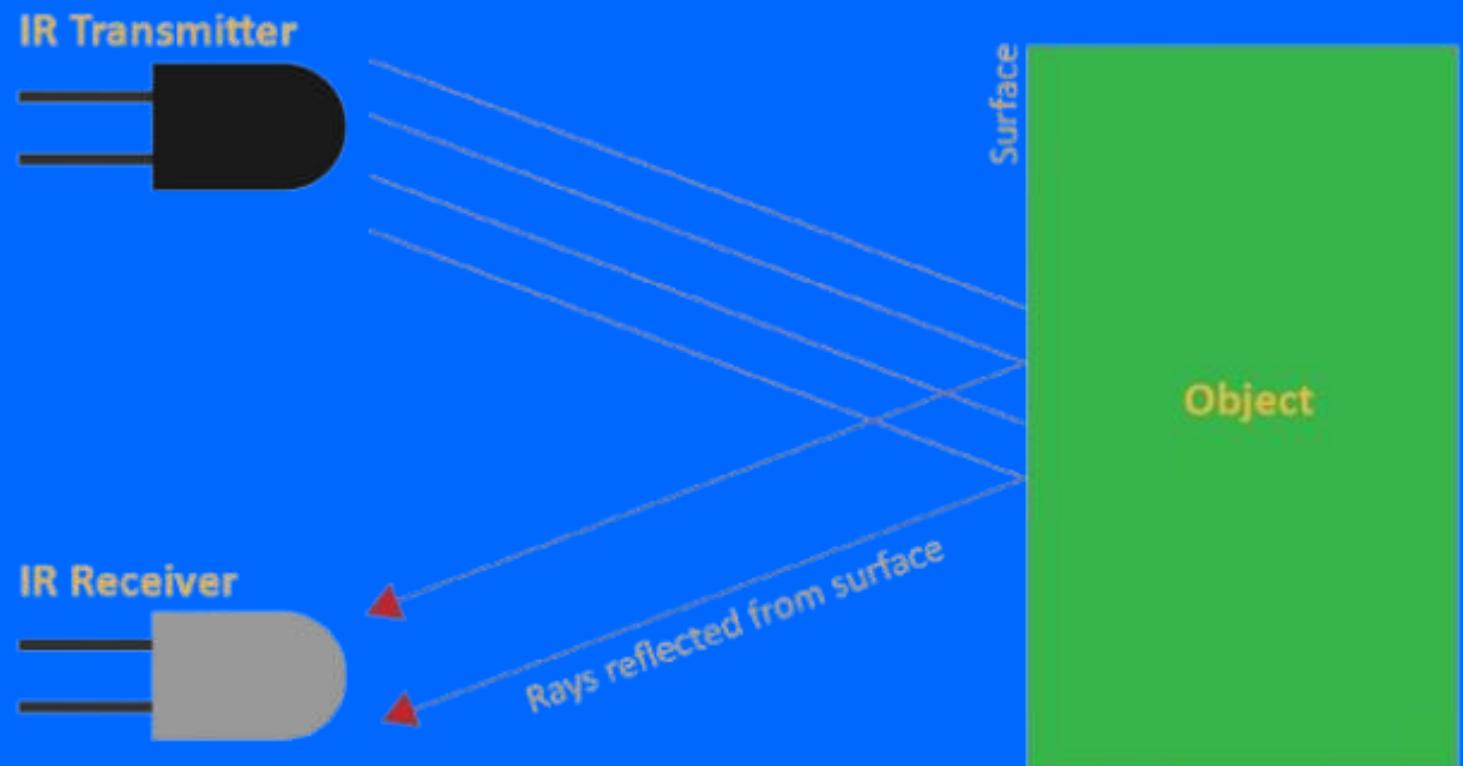


## Pin Configuration

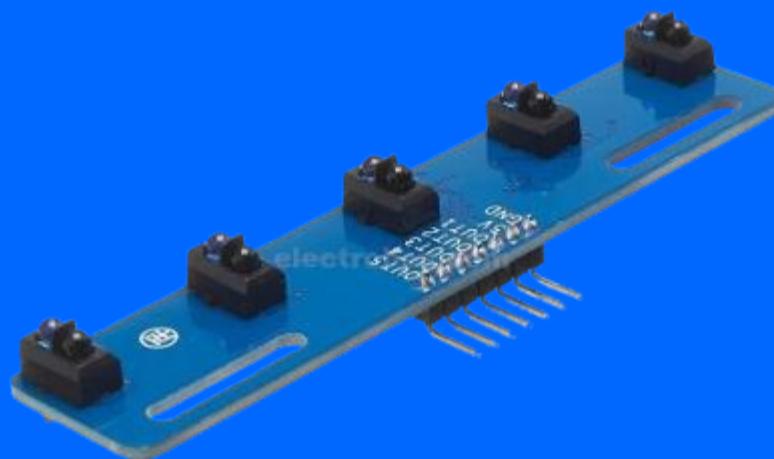
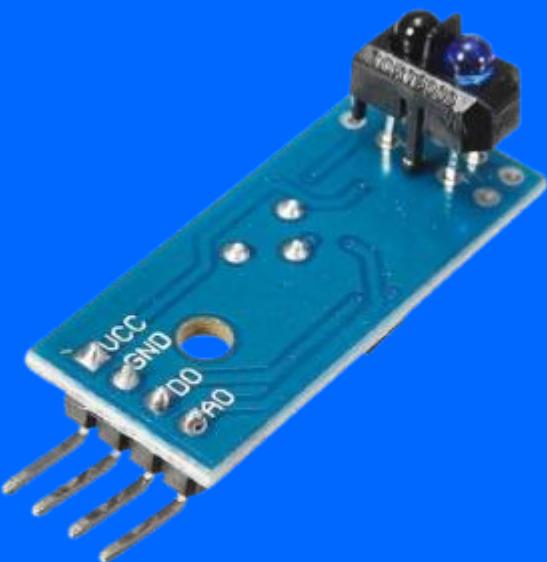
**VCC:** +5V Power Supply

**GND:** Ground

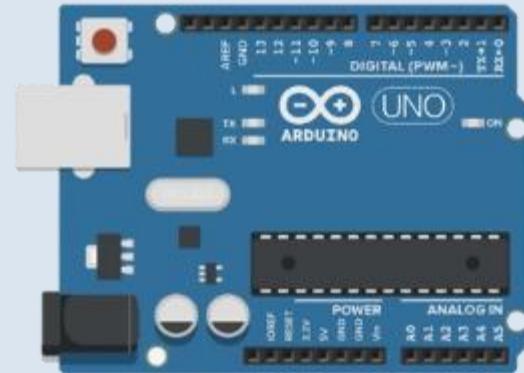
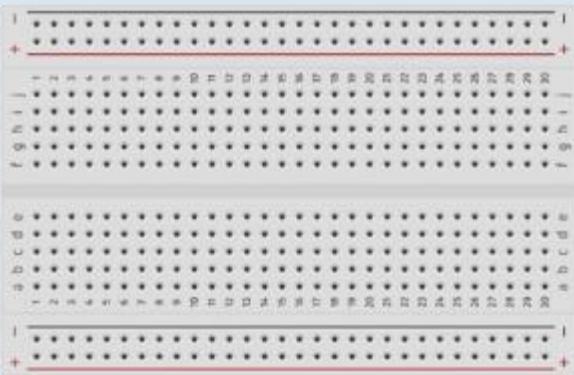
**OUT:** Output pin



## Other models for the IR sensor



# Example:



# LIDAR Sensor

## Introduction:

- **Definition:** The VL6180X is a laser range finder (micro LIDAR) that uses time-of-flight (ToF) technology to measure the distance between the sensor and an object with high precision.
- **Common Model:** VL6180X.

## Working Principle:

- **Laser Emission:** Emits a laser pulse toward the target object.
- **Time-of-Flight Measurement:** Measures the time it takes for the laser pulse to reflect off the object and return to the sensor.
- **Distance Calculation:** Uses the speed of light to calculate the distance based on the time-of-flight of the laser pulse.

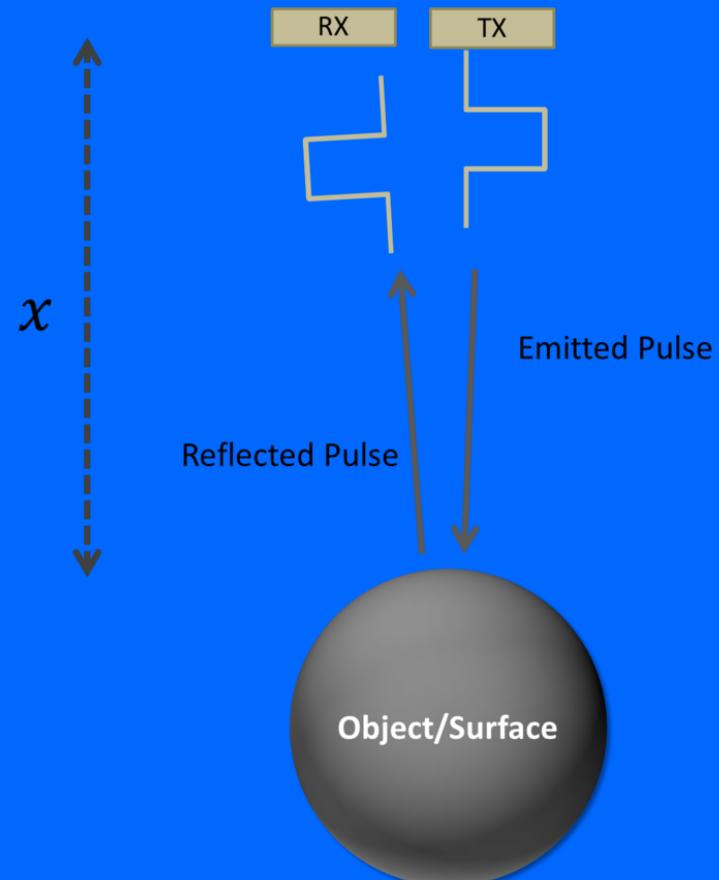


## Pin Configuration

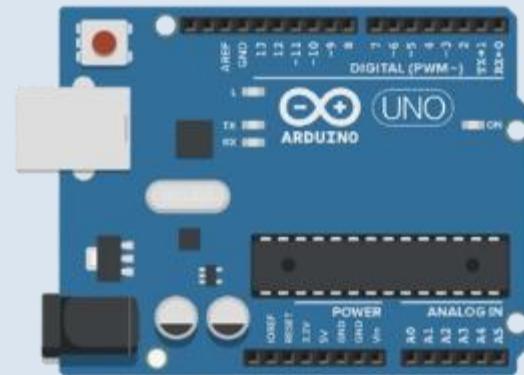
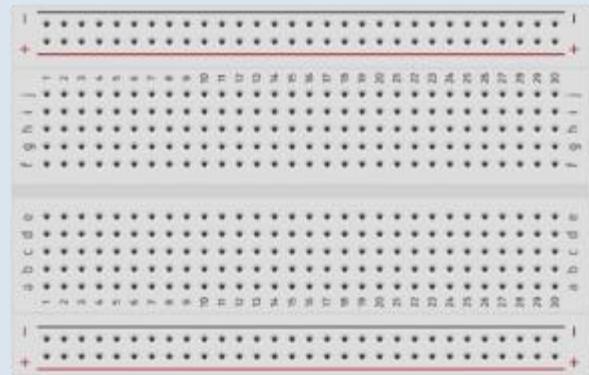
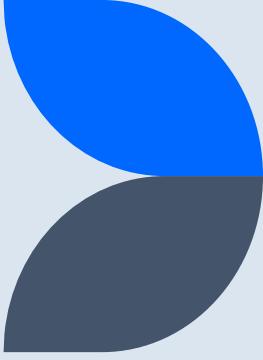
**VCC:** +5V Power Supply

**GND:** Ground

**I2C PIN:** Communication with the sensor



# Example:



# Motor



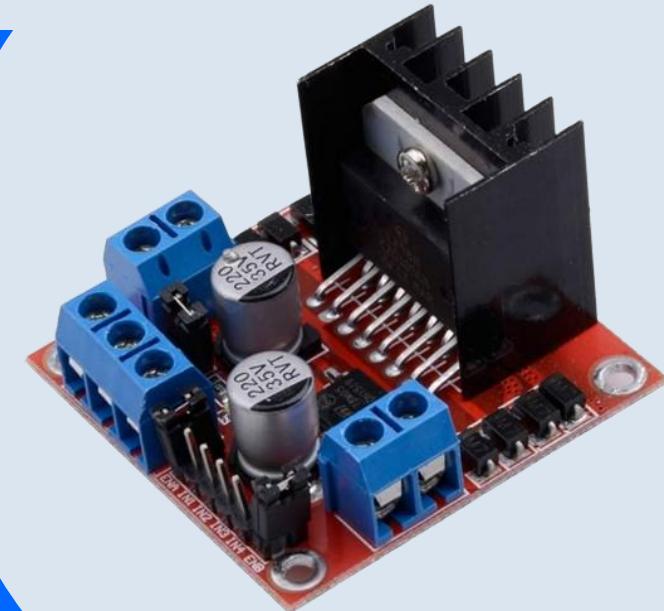
# H-Bridge

## Introduction:

- **Definition:** An H-Bridge is an electronic circuit that enables a voltage to be applied across a load in either direction.
- **Common Use:** Motor control, allowing motors to run forward or backward.

## Working Principle:

- **Configuration:** Consists of four switches (transistors or MOSFETs) that control the direction of current flow.
- **Direction Control:** By closing different pairs of switches, the direction of the current and thus the motor's direction can be controlled.



## Pin Configuration

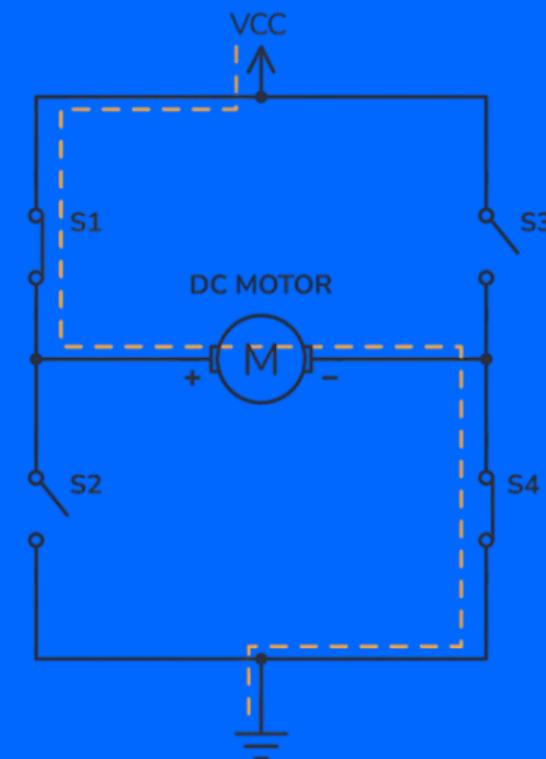
**IN1, IN2, IN3, IN4:** Input pins to control the switches.

**ENA, ENB:** Enable pins to activate the H-Bridge channels.

**OUT1, OUT2, OUT3, OUT4:** Output pins connected to the motor.

**VCC:** Voltage supply for the H-Bridge.

**GND:** Ground.**12V:** Power supply for the motors.



# Examples



# Example 1 => Blinking LED

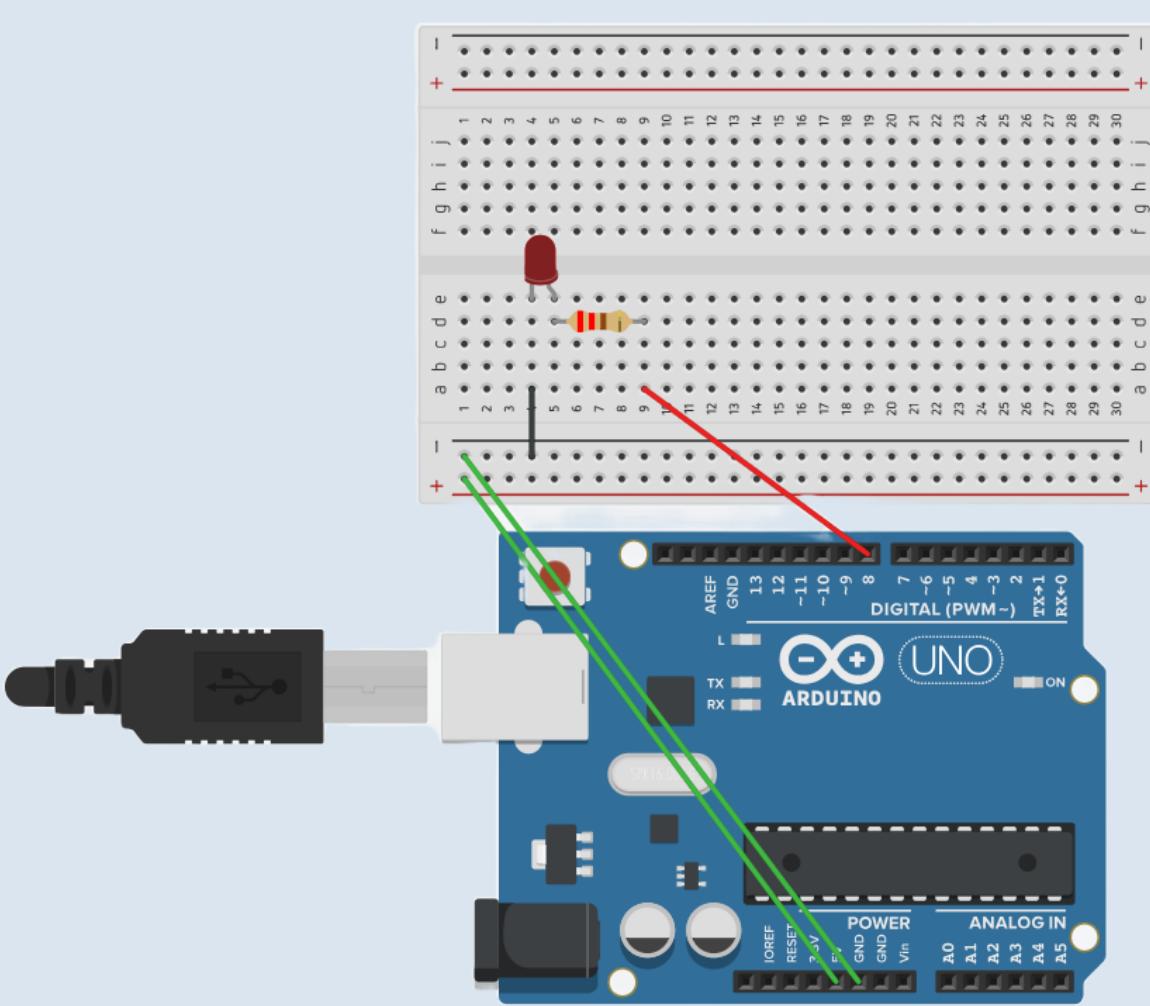


Blinking LED using Delay function:

```
int ledPin=8;

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

void loop(){
    digitalWrite(ledPin,HIGH);
    delay(1000);
    digitalWrite(ledPin,LOW);
    delay(1000);
}
```



# Example 2 => Blinking LED



Blinking LED using millis function:

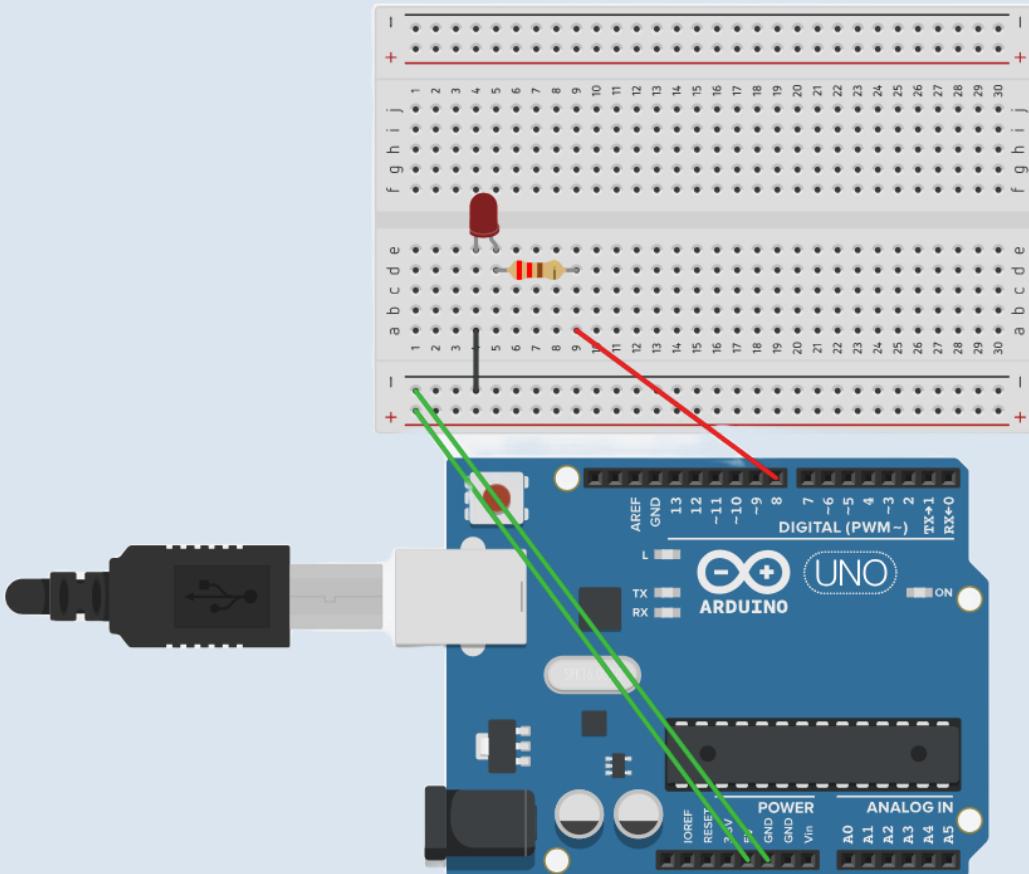
```
int ledPin = 8;
unsigned long previousMillis = 0;
const long interval = 1000;

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

void loop() {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
        previousMillis = currentMillis;

        if (digitalRead(ledPin) == LOW)
            digitalWrite(ledPin, HIGH);
        else
            digitalWrite(ledPin, LOW);
    }
}
```

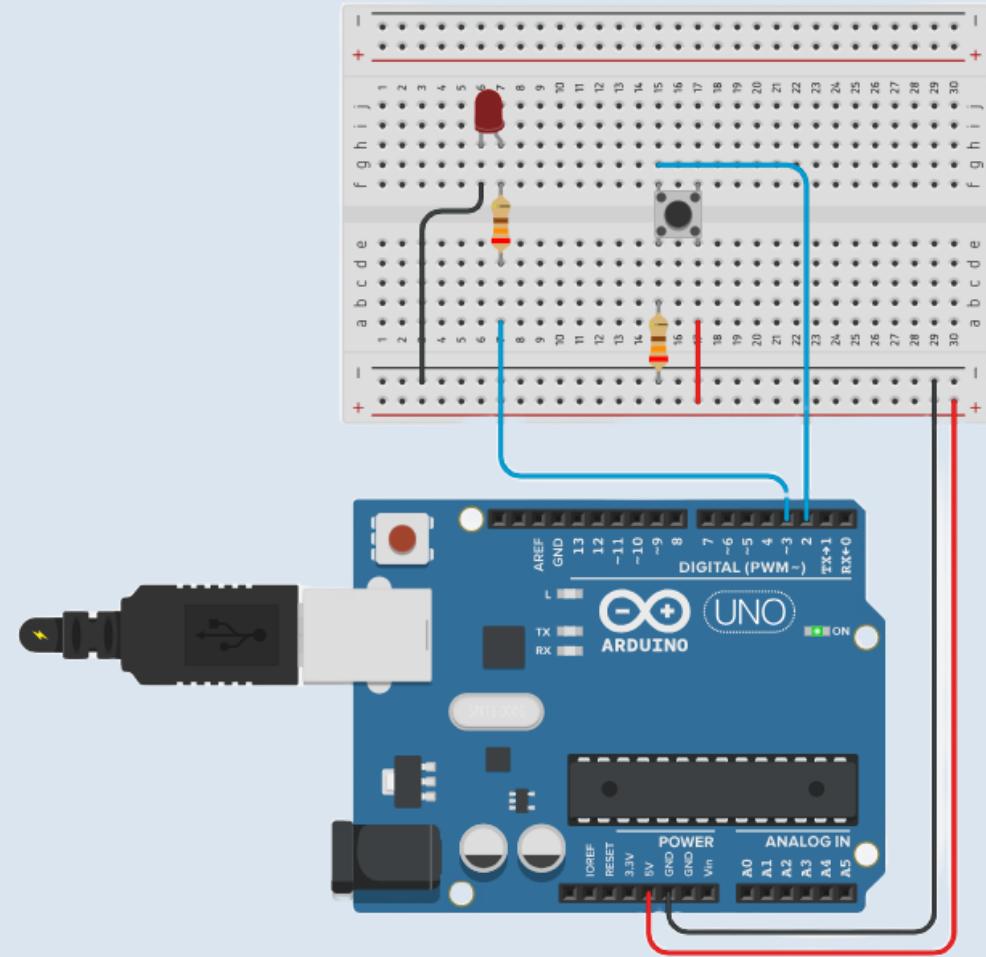


# Example 3 => Toggle LED

Toggle LED without interrupts:



```
● ● ●  
  
const int Switch = 2, LED = 3;  
int state = 0, LEDstate=0;  
  
void setup(){  
    pinMode(Switch, INPUT);  
    pinMode(LED, OUTPUT);  
}  
  
void loop(){  
    if (state == 0 && digitalRead(Switch) == HIGH) {  
        state = 1;  
        LEDstate=!LEDstate;  
    }  
    if (state == 1 && digitalRead(Switch) == LOW) {  
        state = 0;  
    }  
    digitalWrite(LED, LEDstate);  
}
```



# Example 4 => Toggle LED

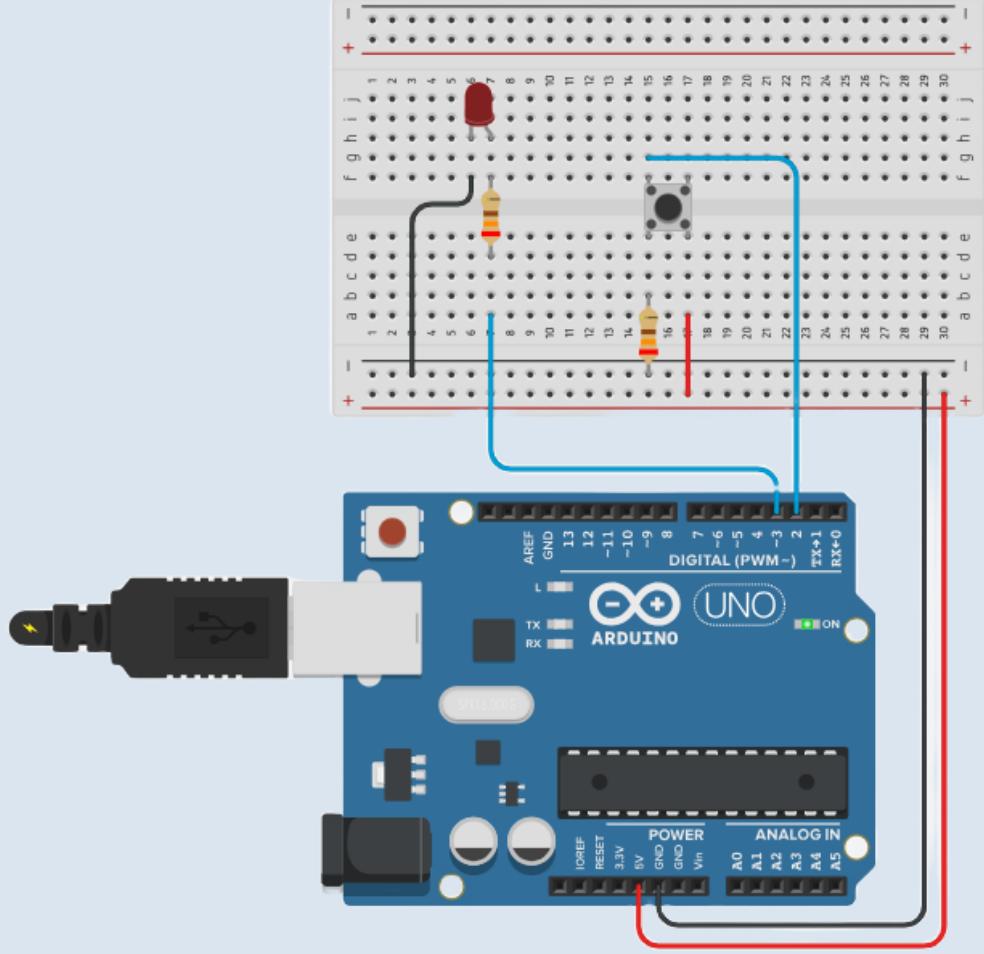
Toggle LED with interrupts:

```
const int buttonPin = 2;
const int ledPin = 3;
volatile int LEDstate = LOW;

void setup() {
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(buttonPin),
                    toggleLED, RISING);
}

void loop() {}

void toggleLED() {
    LEDstate = !LEDstate;
    digitalWrite(ledPin, LEDstate);
}
```



# Example 5 => Toggle LED

Toggle LED with interrupts and avoid bouncing effect:



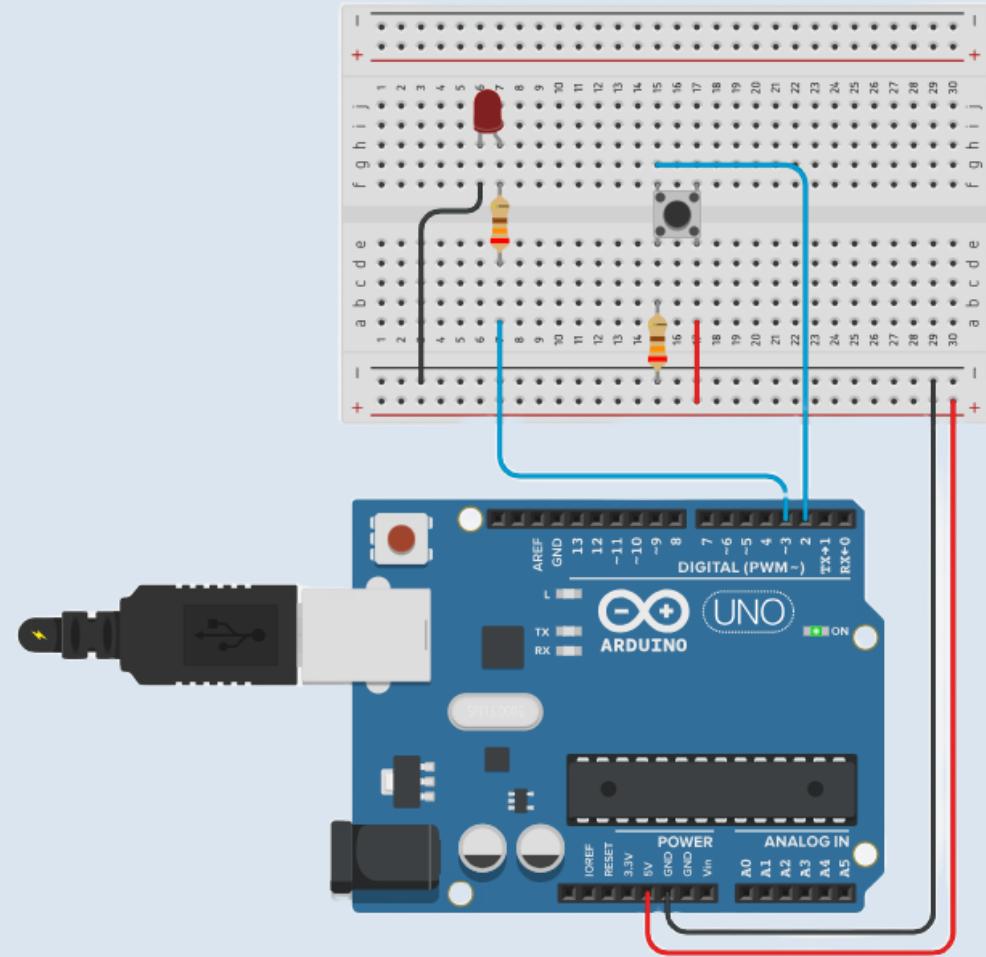
```
● ● ●

const int buttonPin = 2;
const int ledPin = 3;
volatile int LEDstate = LOW;
volatile unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 50;

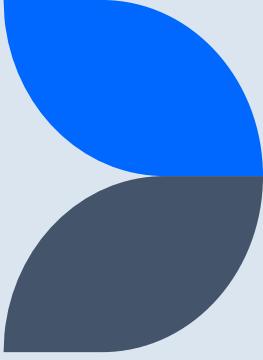
void setup() {
    pinMode(buttonPin, INPUT);
    pinMode(ledPin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(buttonPin),
                    toggleLED, RISING);
}

void loop() {}

void toggleLED() {
    unsigned long currentTime = millis();
    if (currentTime - lastDebounceTime
        > debounceDelay) {
        LEDstate = !LEDstate;
        lastDebounceTime = currentTime;
        digitalWrite(ledPin, LEDstate);
    }
}
```



# Example 6 => Controlling LED



Creating an interactive system using an Arduino that involves the following components:

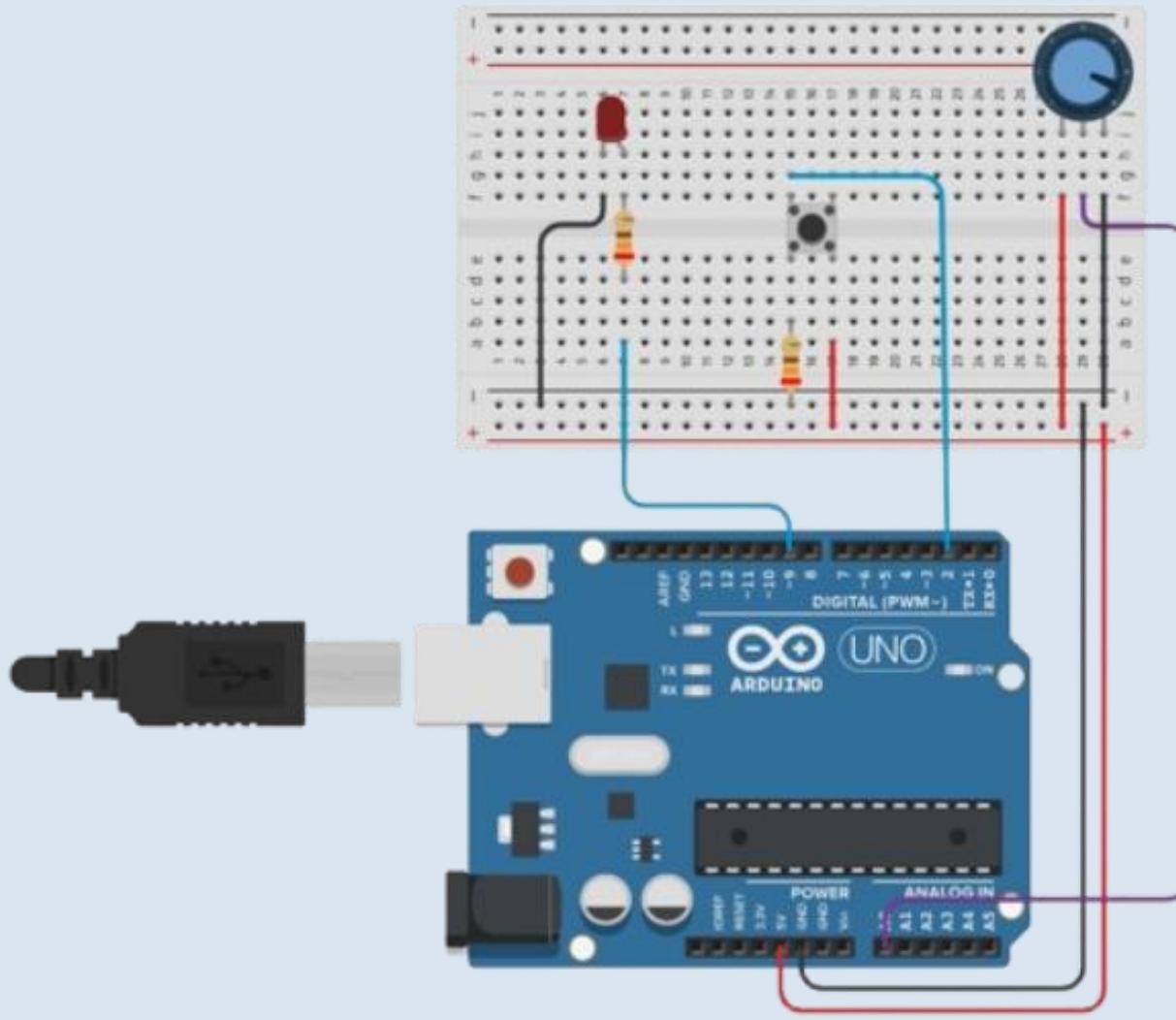
- Potentiometer
- Button
- LED

Requirements:

- The potentiometer should control the brightness of the LED.
- Button should toggle the LED on and off.
- Use interrupts to handle the button presses efficiently.



# Example 6 => Controlling LED



TIN  
KER  
CAD

# Example 6 => Controlling LED

```
● ● ●

const int potPin = A0;
const int ledPin = 9;
const int button1Pin = 2;

volatile bool ledState = false;
int potValue = 0;
int brightness = 0;

void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(button1Pin, INPUT);

    attachInterrupt(digitalPinToInterrupt(button1Pin), toggleLED, RISING);
}

void loop() {

    potValue = analogRead(potPin);
    brightness = map(potValue, 0, 1023, 0, 255);

    if (ledState) {
        analogWrite(ledPin, brightness);
    } else {
        analogWrite(ledPin, 0);
    }

    delay(10); // Small delay to avoid rapid changes
}

void toggleLED() {
    ledState = !ledState;
}
```



# Example 7 => Controlling Array of LEDs

## Unsolved example

You need to create an Arduino system that controls four LEDs based on the value of a potentiometer.

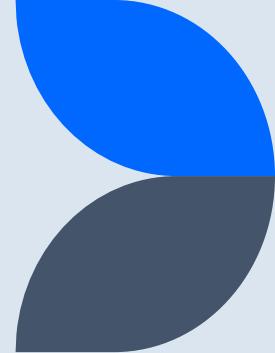
Requirements: Use a potentiometer to determine how many LEDs should be turned on.

The system should work as follows:

- If the potentiometer value is between 0 and 255, turn on 1 LED.
- If the potentiometer value is between 256 and 511, turn on 2 LEDs.
- If the potentiometer value is between 512 and 767, turn on 3 LEDs.
- If the potentiometer value is between 768 and 1023, turn on all 4 LEDs.

# Example 8 => LED Reaction Game

## Unsolved example

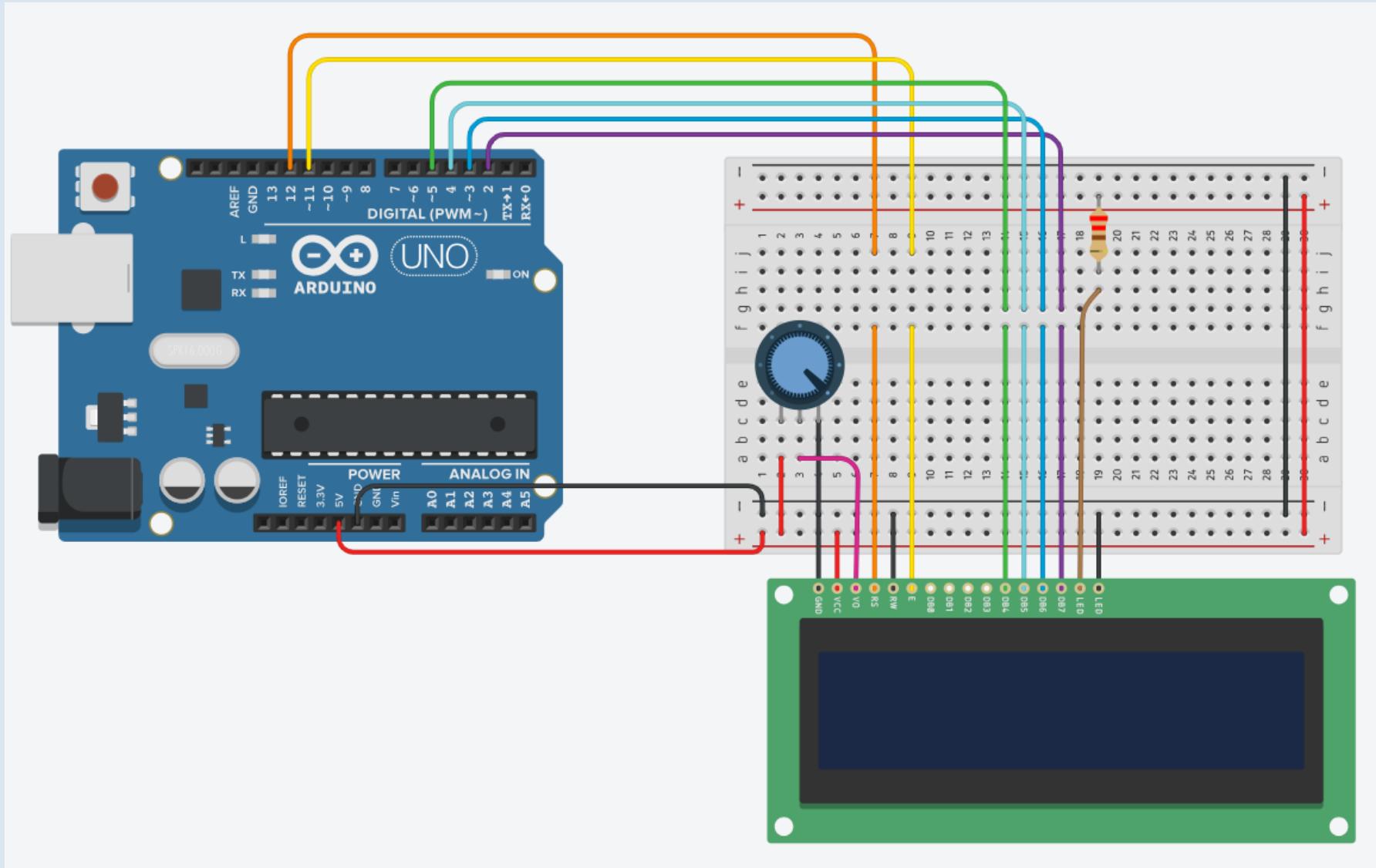


Design an Arduino-based reaction game that uses LEDs, a potentiometer, and a button. The goal of the game is to test the player's reaction time.

Requirements:

- Use a potentiometer to set the difficulty level of the game.
- An LED should light up randomly after a delay.
- The player must press a button as quickly as possible when the LED lights up.
- The system should display the player's reaction time using LEDs.
- The game should reset automatically after displaying the reaction time.

# Example 9 => Print Your Name



TIN  
KER  
CAD

# Example 9 => Print Your Name

```
#include <LiquidCrystal.h>

// LiquidCrystal(rs, enable, d4, d5, d6, d7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

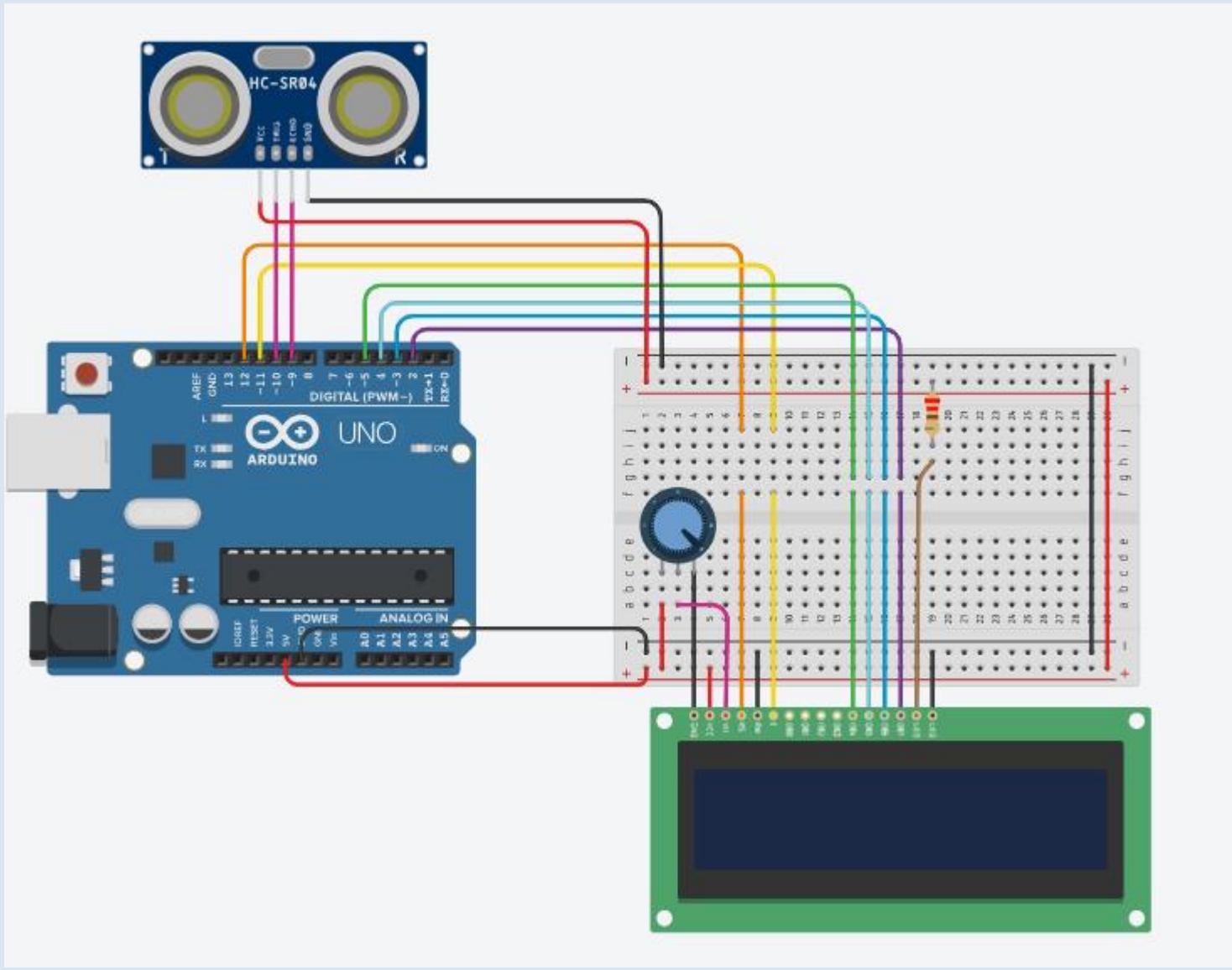
void setup()
{
    lcd.begin(16, 2);

    lcd.setCursor(0, 0);
    lcd.print("Qossay Abu Rida");
}

void loop(){}
```



# Example 10 => Distance Measurement



T  
I  
N  
K  
E  
R  
C  
A  
D

# Example 10 => Distance Measurement

```
#include <LiquidCrystal.h>

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
const int trigPin = 10;
const int echoPin = 9;

void setup() {
    lcd_1.begin(16, 2);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    lcd_1.print("Distance: ");
}

void loop() {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

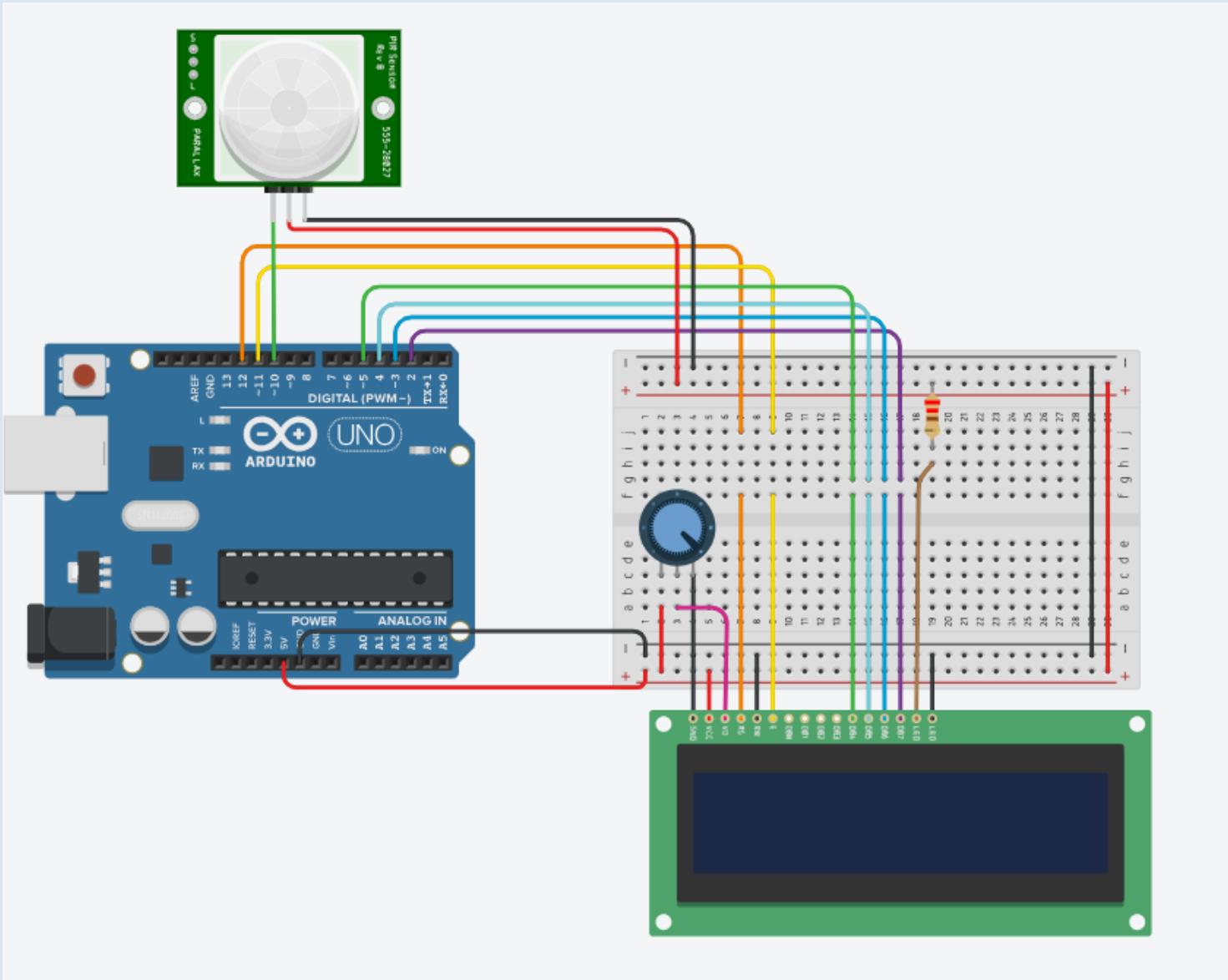
    long duration = pulseIn(echoPin, HIGH);
    float distance = duration * 0.034 / 2;

    lcd_1.setCursor(0, 1);
    lcd_1.print("          "); // Clear previous reading
    lcd_1.setCursor(0, 1);
    lcd_1.print(distance);
    lcd_1.print(" cm");
    delay(500);
}
```

The speed of sound in air is approximately 343 meters per second or 0.0343 centimeters per microsecond.



# Example 11 => Motion detection



T  
I  
N  
K  
E  
R  
C  
A  
D

# Example 11 => Motion detection

```
#include <LiquidCrystal.h>

LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
const int pirPin = 10;

void setup() {
    lcd_1.begin(16, 2);
    pinMode(pirPin, INPUT);
    lcd_1.print("PIR Sensor:");
}

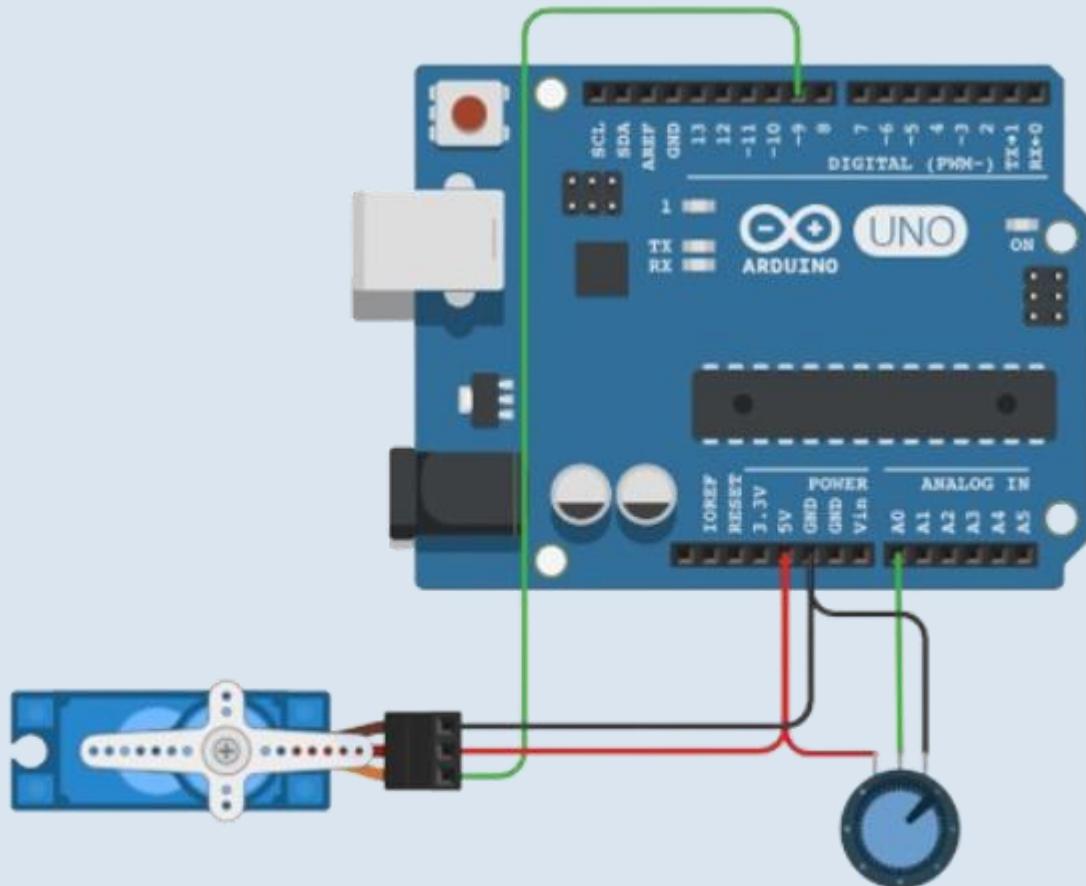
void loop() {
    int pirState = digitalRead(pirPin);

    lcd_1.setCursor(0, 1);
    if (pirState == HIGH)
        lcd_1.print("Motion detected");
    else
        lcd_1.print("No motion      ");

    delay(500);
}
```



# Example 12 => Servo Motors



TIN  
KER  
CAD

# Example 12 => Servo Motors

```
#include <Servo.h>

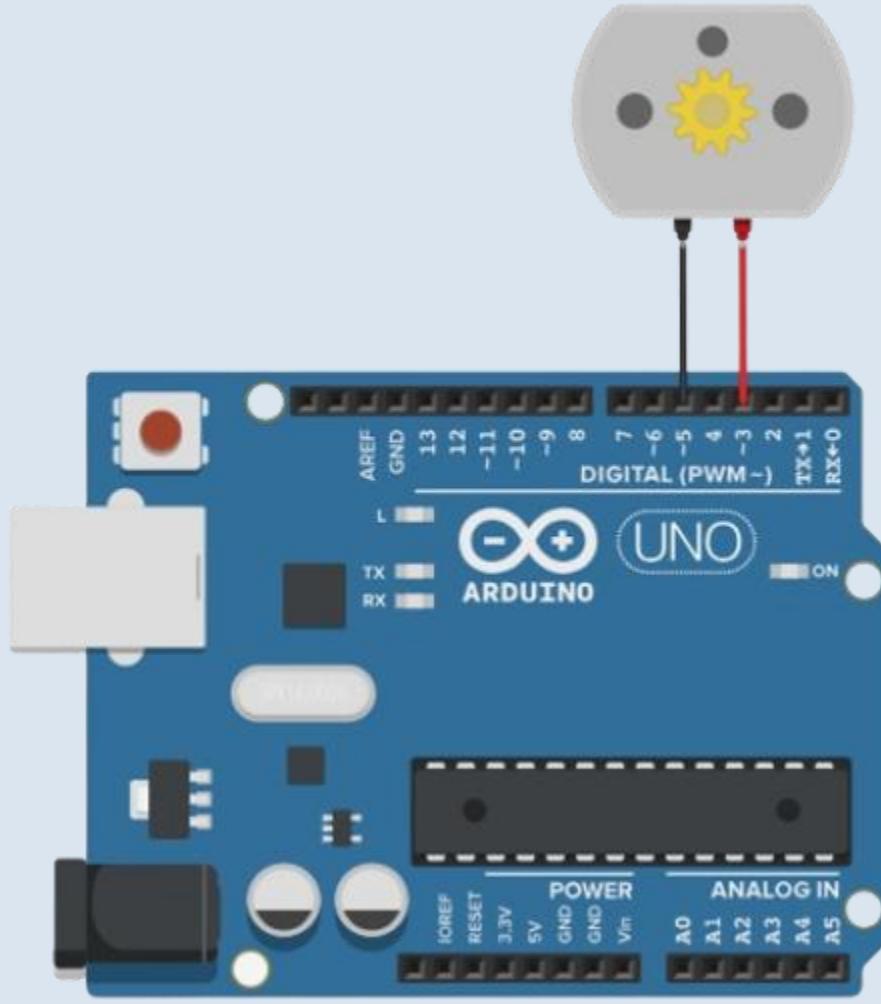
Servo myservo;
int value;
double angle;

void setup()
{
    Serial.begin(9600);
    myservo.attach(9);
}

void loop()
{
    value = analogRead(A0);
    angle = map(value, 0, 1023, 0, 180);
    Serial.println(angle);
    myservo.write(angle);
    delay(15);
}
```



# Example 13 => DC Motors



TIN  
KER  
CAD

# Example 13 => DC Motors

```
● ● ●

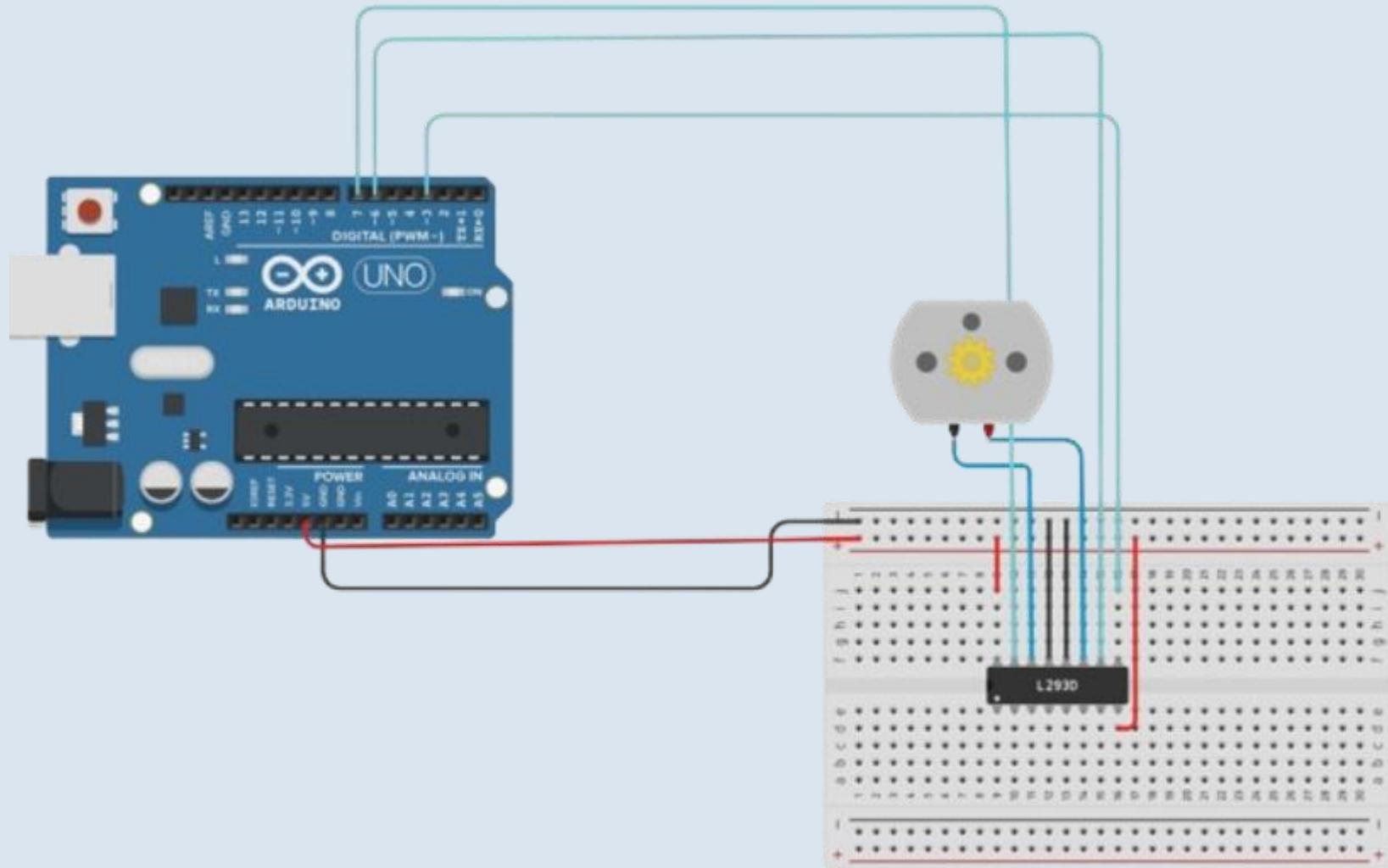
int motorPin1 = 3;
int motorPin2 = 5;

void setup() {
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
}

void loop() {
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
    delay(2000);
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, HIGH);
    delay(2000);
}
```



# Example 14 => H-Bridge



T I N  
K E R  
C A D

# Example 14 => H-Bridge

```
● ● ●

int enA = 3;
int in1 = 6;
int in2 = 7;

void setup() {
    pinMode(enA, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
}

void loop() {
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    analogWrite(enA, 255);
    delay(2000);
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    delay(2000);
}
```



# Contact us

