PES UNIVERSITY

IEEE Robotics & Automation Society
PES UNIVERSITY
Student Chapter

Arduino

Robofest Workshop

What we cover

- Introduction
- Blinking an LED
- Reading from the serial monitor (LED morse code implementation)
- PWM - Making an automatic brightness lamp
- Making an Arduino car using a motor driver
- Making it a wireless car using an HC-05 bluetooth module
- Using pyserial to send commands from a python code

Introduction

After gaining appreciable knowledge of low level drivers, an ideal path would be to get into image processing, followed by ROS for those willing to dive deep into robotics.

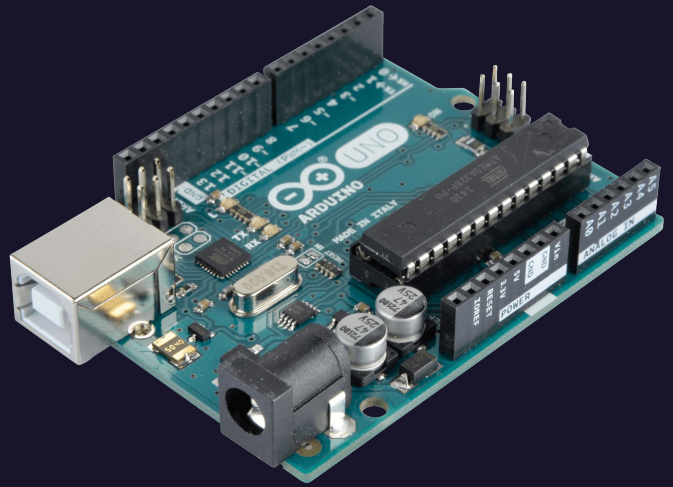
Introduction

Almost all robotics applications require low level drivers to complete the communication that makes the robot perform as desired.

With an increasing use of image processing and ROS in robotic systems, it is highly recommended to understand the low level workings of robotics, to be able to accurately implement these concepts while working on projects.

What is Arduino?

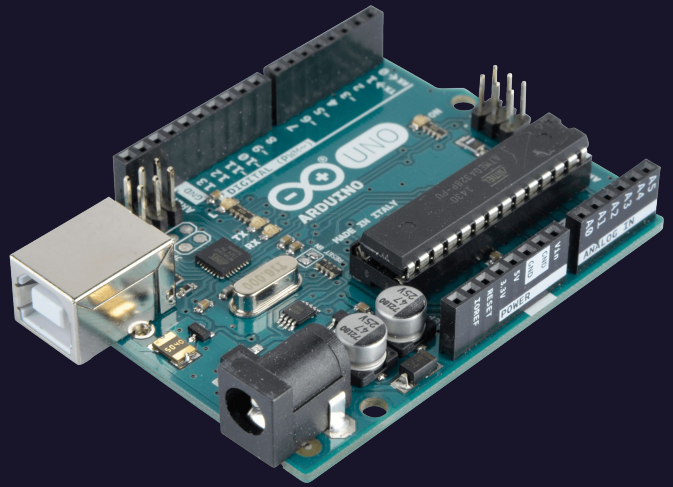
Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It’s intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.



What is Arduino?

There are many different Arduino boards, like Arduino Uno, Arduino Nano, Arduino Micro, Lilypad, etc.

We'll primarily be looking into Arduino Uno.

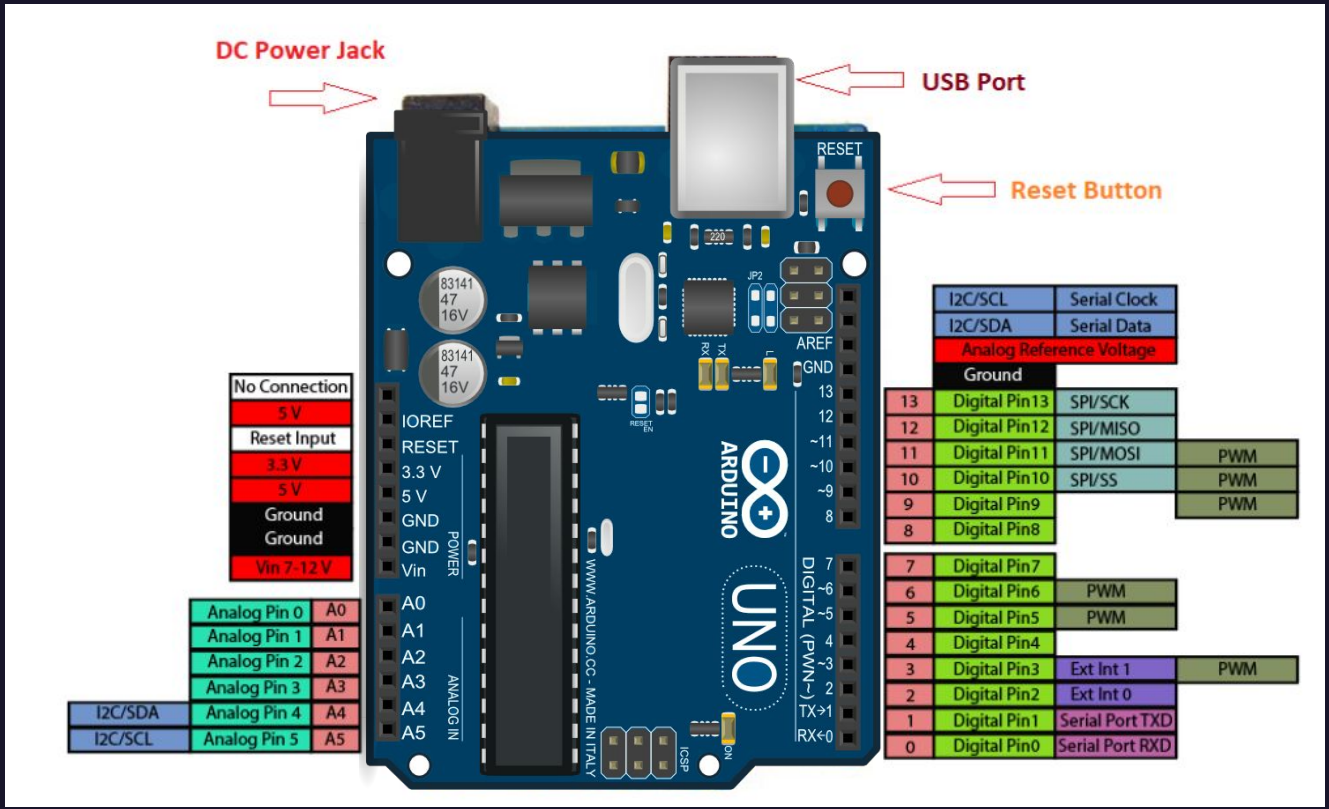


Communication Protocols

1. Universal Asynchronous Receiver Transmitter (UART)
2. Serial Peripheral Interface (SPI)
3. Inter-integrated circuit (I2C)

Link for more information:

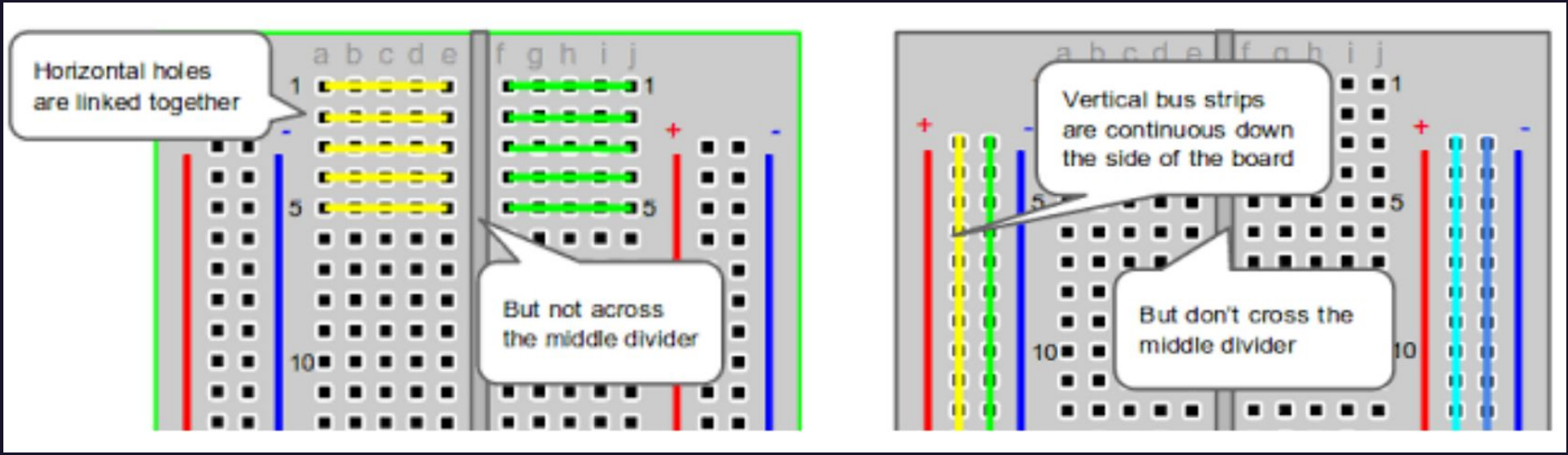
https://www.deviceplus.com/arduino/arduino-communication-protocols-tutorial/


DC Power Jack
USB Port
Reset Button
RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
IOREF
RESET
3.3 V
5 V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
WWW.ARDUINO.CC - MADE IN ITALY
ICSP
No Connection
5 V
Reset Input
3.3 V
5 V
Ground
Ground
Vin 7-12 V
Analog Pin 0 A0
Analog Pin 1 A1
Analog Pin 2 A2
Analog Pin 3 A3
I2C/SDA Analog Pin 4 A4
I2C/SCL Analog Pin 5 A5
I2C/SCL Serial Clock
I2C/SDA Serial Data
Analog Reference Voltage
Ground
13 Digital Pin13 SPI/SCK
12 Digital Pin12 SPI/MISO
11 Digital Pin11 SPI/MOSI PWM
10 Digital Pin10 SPI/SS PWM
9 Digital Pin9 PWM
8 Digital Pin8
7 Digital Pin7
6 Digital Pin6 PWM
5 Digital Pin5 PWM
4 Digital Pin4
3 Digital Pin3 Ext Int 1 PWM
2 Digital Pin2 Ext Int 0
1 Digital Pin1 Serial Port TXD
0 Digital Pin0 Serial Port RXD

Structure


SENSOR
A2D
CONVERTER
PROCESSOR
D2A
CONVERTER
ACTUATOR
Arduinos don’t have this. That’s why we use Pulse Width Modulation (PWM)

Breadboards

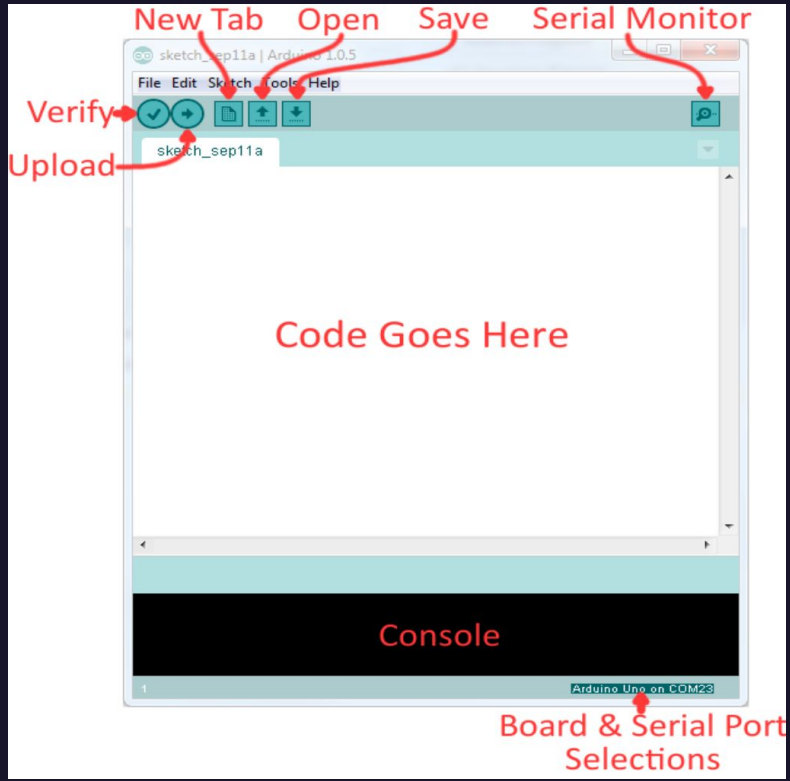


Arduino IDE installation

Available software for Windows, Linux, and Mac OS X

Link for installation:

https://www.arduino.cc/en/software



Coding

We use C++, but this is very different from most C++ varieties. There is an extensive use of conditional statements (while, if else, for loops), but because of its high level of abstraction, it’s comparatively simple to use.

Since this is a beginner level course, and there isn’t extensive coding, having less or no prior knowledge of coding is not an issue.

If you don’t own an Arduino yet

- Head over to https://www.tinkercad.com/, create an account and go to your dashboard, click on circuits on the side nav and then finally click on create new circuit.
- The Circuit Design is pretty straight forward! Search for the component on the right side and drag and drop.

If you don’t own an Arduino yet

- Once the required circuit is designed, click on the “Code”, click on the “Blocks” and choose “Text”.
- Make the required changes in the code and hit “Start Simulation”.

Some important methods and functions in Arduino

- void setup(): runs once when you press reset or power the board.
- void loop(): runs over and over again forever.
- pinMode(): initialize pin as input or output.
- delay(): wait for a certain amount of time (in milliseconds).

Some important methods and functions in Arduino

- digitalWrite(): set a pin as HIGH or LOW.
- digitalRead(): read a digital pin’s state.
- analogWrite(): write an analog pin’s PWM value.
- analogRead(): read an analog pin’s state.

Serial Monitor

The serial monitor is present in the Arduino IDE, at the top right corner. Outputs are generally sent there, and inputs are read from it. This uses serial communication, and needs to start by setting the data rate in bits per second using ‘Serial.begin(baud_rate)’. The acceptable rates are listed as a drop down menu in the serial monitor. However, the most general use case is a baud rate of 9600.

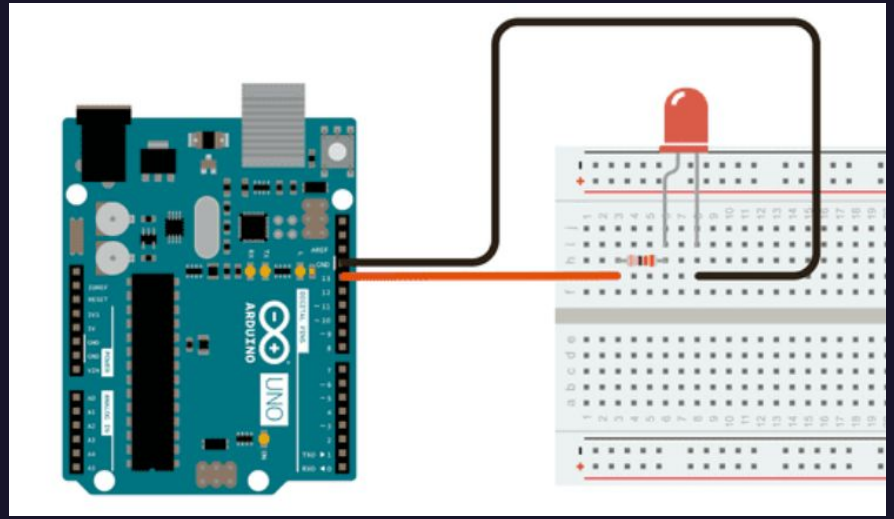
Serial Monitor

‘Serial.print()’ prints the argument in the serial monitor, and ‘Serial.println()’ prints it followed by a new line character.

‘Serial.read()’ is used to read from the serial monitor.

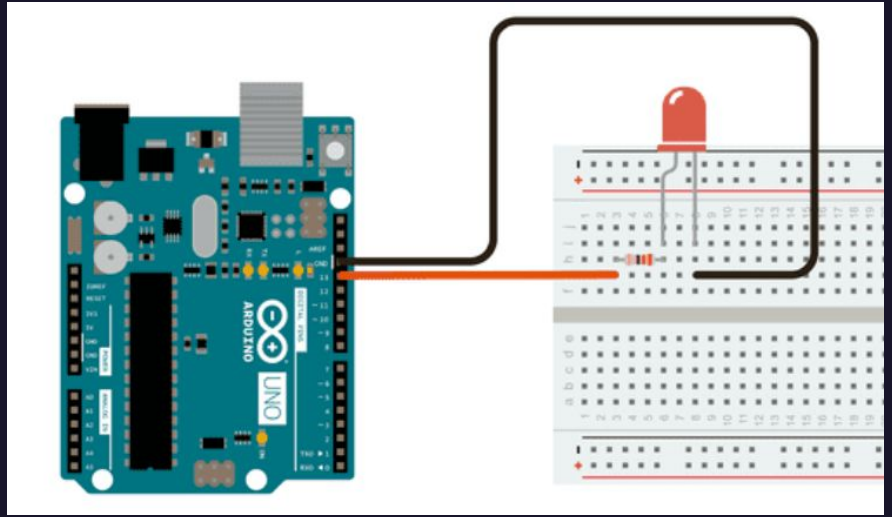
Blinking LED - Wiring

As you can see, the longer end of the LED is connected to pin number 13, while the shorter end is connected to ground. A resistor is connected to regulate the flow of current.



Blinking LED - Wiring

You can connect the LED to any pin, and pin 13 has an in-built resistance. For better understanding of the wiring and the code, it is recommended that you use another pin.



Blinking LED - Code

Whichever pin you connect your LED to, you will have to initialize that pin as an output pin in void setup.

To make the LED blink, you will need to write the voltage level as HIGH first, then LOW in void loop. Make sure to give delays to be able to notice the blinking.

Blinking LED - Code

Now connect your Arduino and go to ‘Tools’ in the IDE and select the appropriate board and port number. Upload the code.

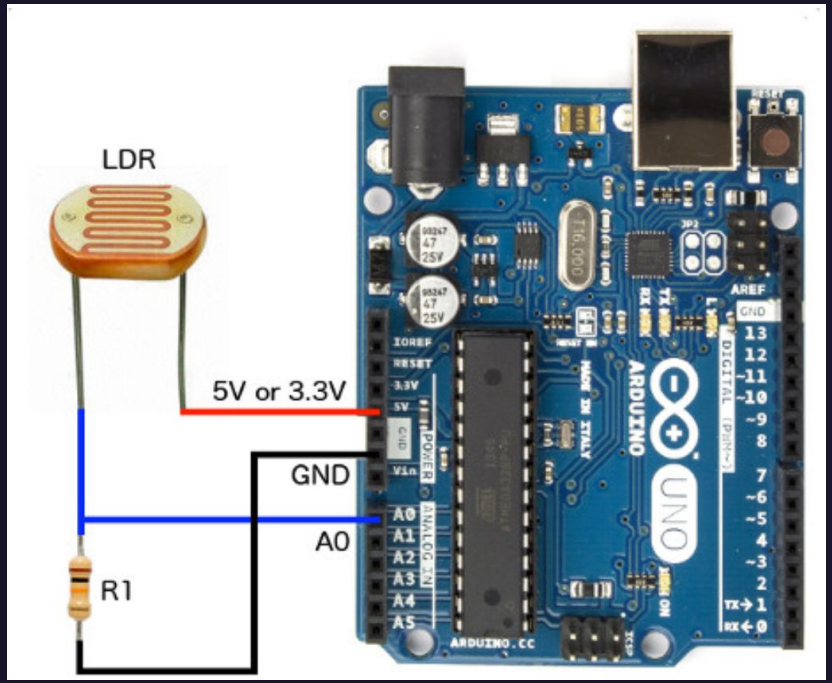
Congratulations on completing your first project on an Arduino!

Morse communication light

Now you can try and implement these concepts by making a morse communication light. Taking input from the serial monitor, the LED can blink for either 100ms or 1000ms based on whether the code reads a ‘.’ or a ‘-’ respectively.

LDR

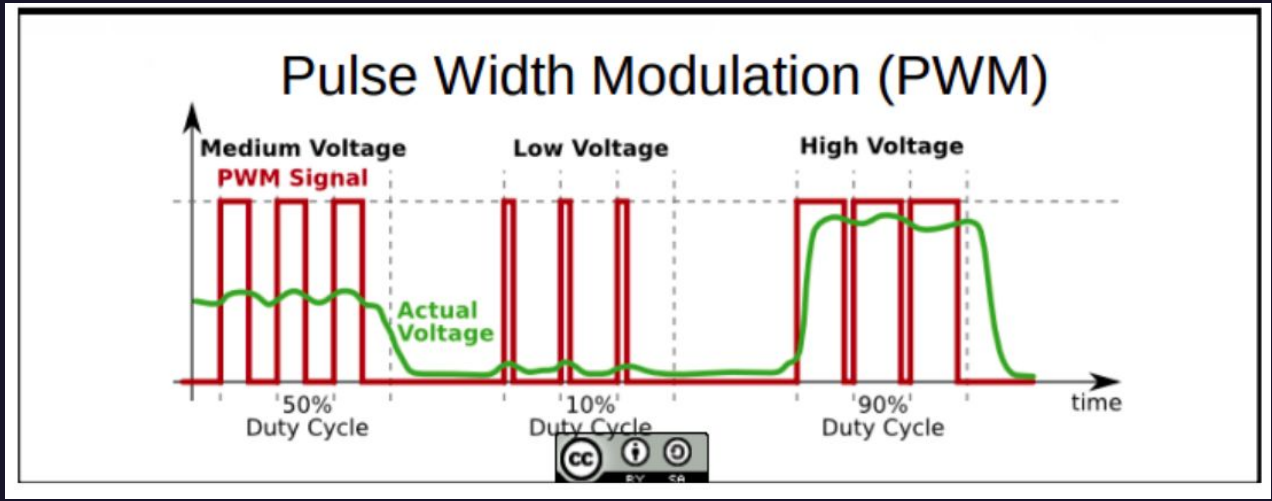
- One pin to 5V
- The other pin to a resistor that connects to ground (needed to bring change in the voltage rather than just current)
- The other pin of the LDR also connects to one of the analog pins to send input signal



Pulse Width Modulation

Arduinos don’t have a digital to analog converter. Hence, to regulate the actuators to perform as desired, we use the concept of pulse width modulation, which is basically sending the HIGH and LOW values at different duty cycles, allowing the signal to behave like an ‘analog’ signal.

There are some pins in the arduino board marked ‘~’, which accept PWM signals.

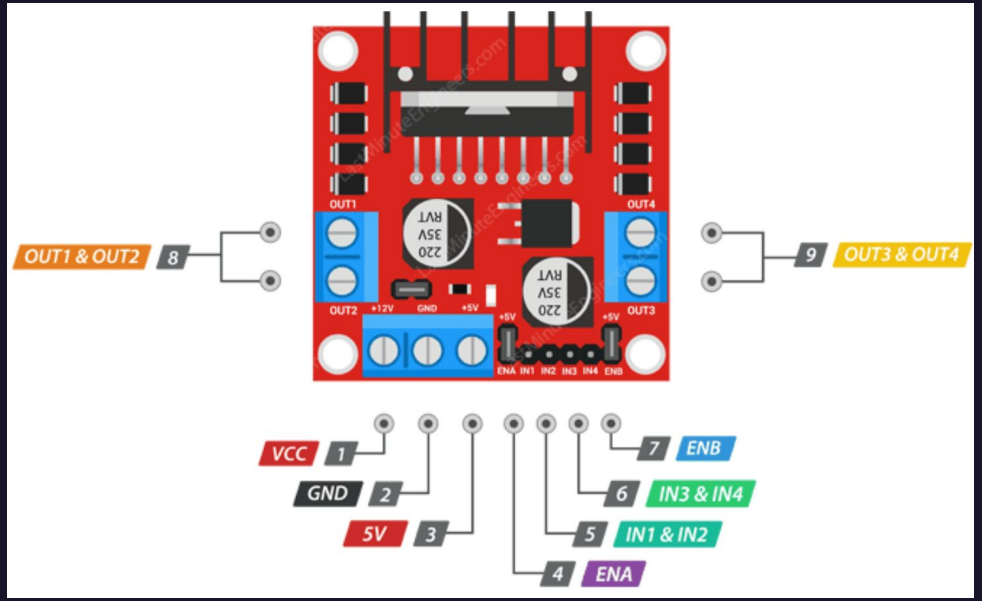


Automatic Lamp

Now we can try implementing this using an LDR and an LED. A pretty simple and elegant circuit, which would light up the LED based on the amount of light in the surroundings, kind of how our phones' 'auto-brightness' works. An LDR gives different values based on the surrounding light, and an approximation could be made by printing these values in the serial monitor.

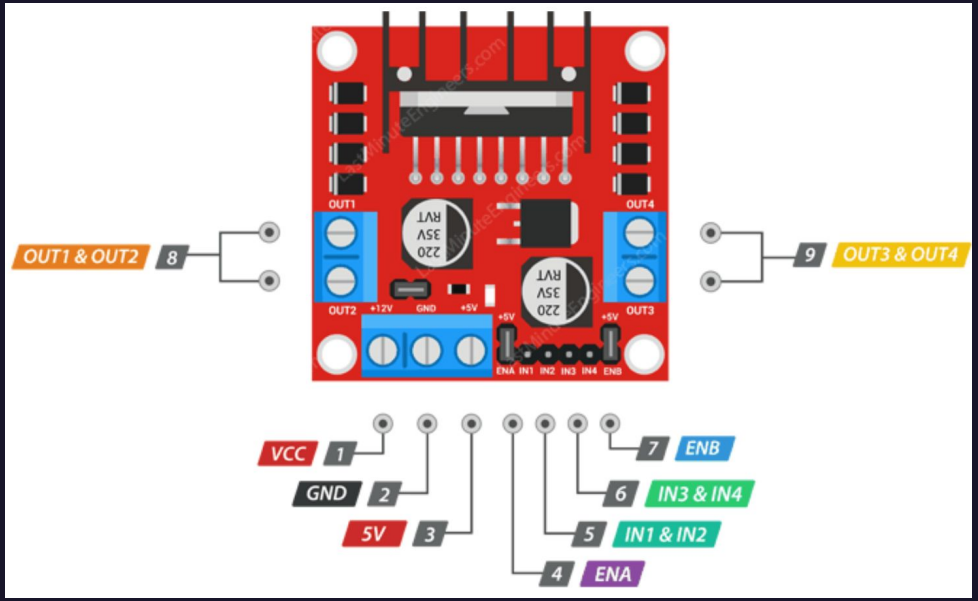
Motor Driver

This is a module for motors that allows you to control the speed and direction of two motors simultaneously.



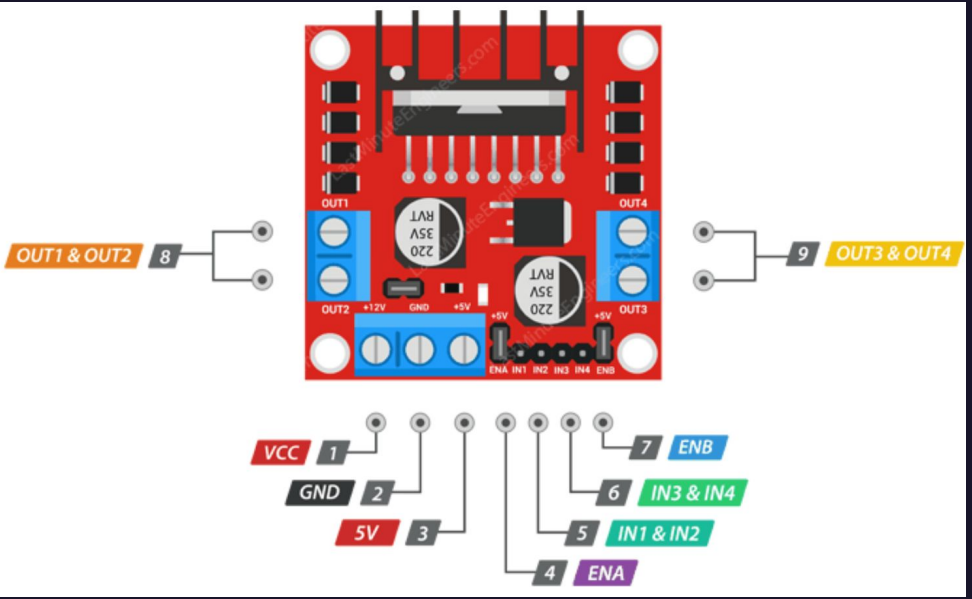
Motor Driver

OUT1, OUT2, OUT3 and OUT4 connect to the two motors, and IN1, IN2, IN3 and IN4 connect with the Arduino.



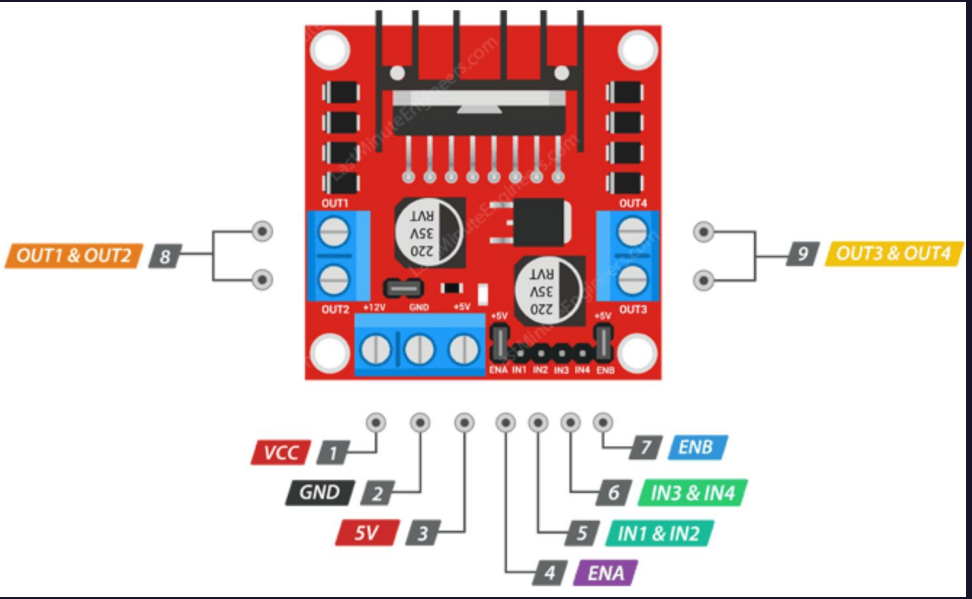
Motor Driver

The ENA and ENB pins can be connected to the Arduino for controlling the speed of the motors.



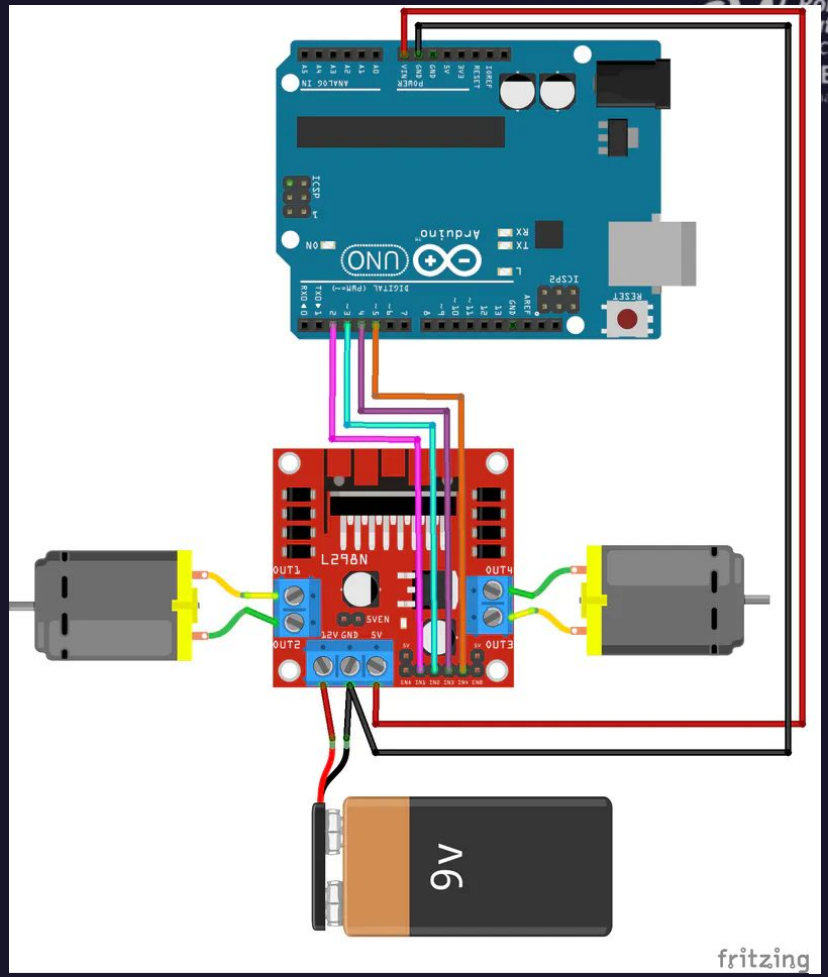
Motor Driver

The VCC pin is used to power the motor driver (12V), and GND is ground. The 5V pin is a 5V output that can be used to power the Arduino.



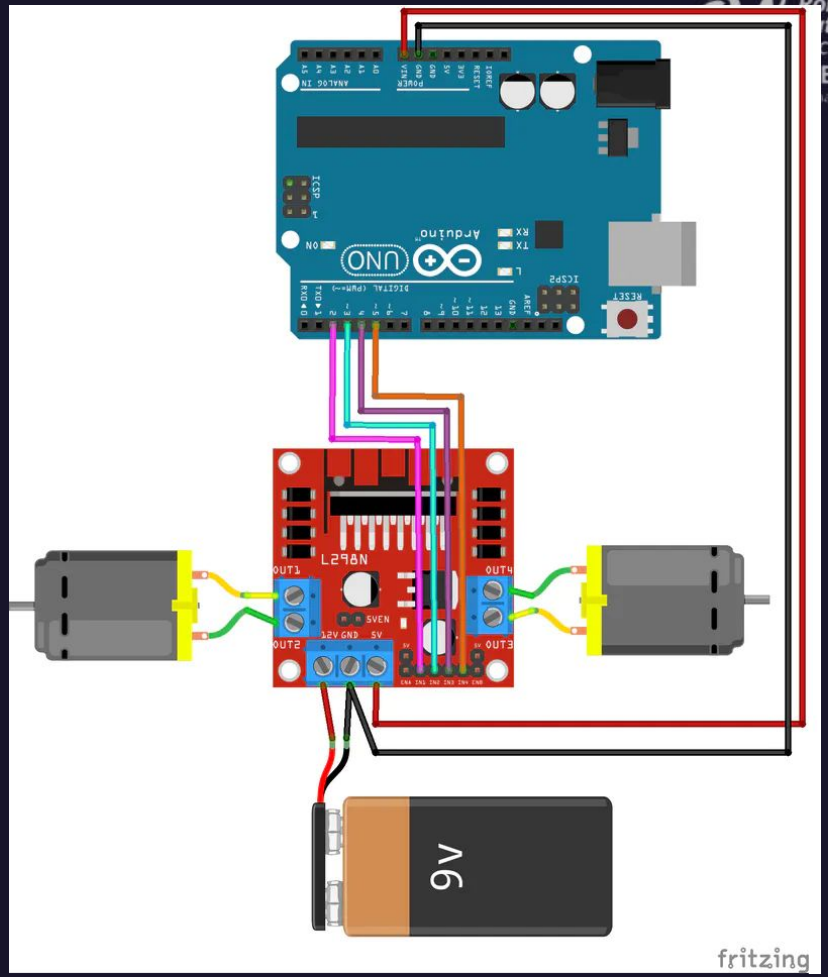
Motor Driver - Wiring

Connecting the ENA and ENB pins are optional, and would need to be connected to PWM pins since they control the speed of the motors. They haven't been connected in this figure.



Motor Driver - Wiring

One important thing to remember is that the ground of the Arduino has to be connected with the ground of the motor driver for it to work.



Motor Driver - Code

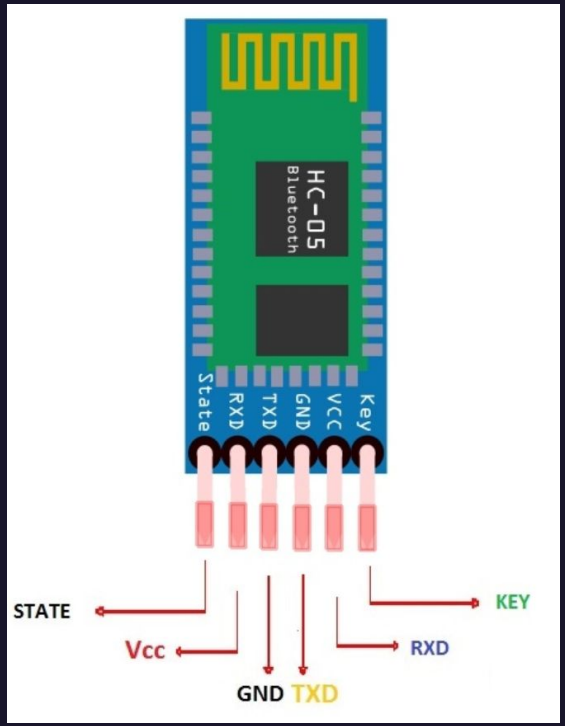
IN1 and IN2 are used to control one motor, and IN3 and IN4 are used to control the other. If one pin receives a HIGH signal whereas the corresponding other pin receives a LOW signal, the motor will rotate clockwise or anticlockwise based on which pin received HIGH. If both pins receive HIGH or LOW, then the motor does not move.

Motor Driver - Code

The way to go about the code would be to set the pins of the Arduino that connect to the motor driver as OUTPUT pins, and take an input from the serial monitor, and based upon the input received, set those pins as HIGH or LOW accordingly, using if statements. Remember to set the pins to LOW after a small delay. The delay you give depends upon how long you want each command to execute.

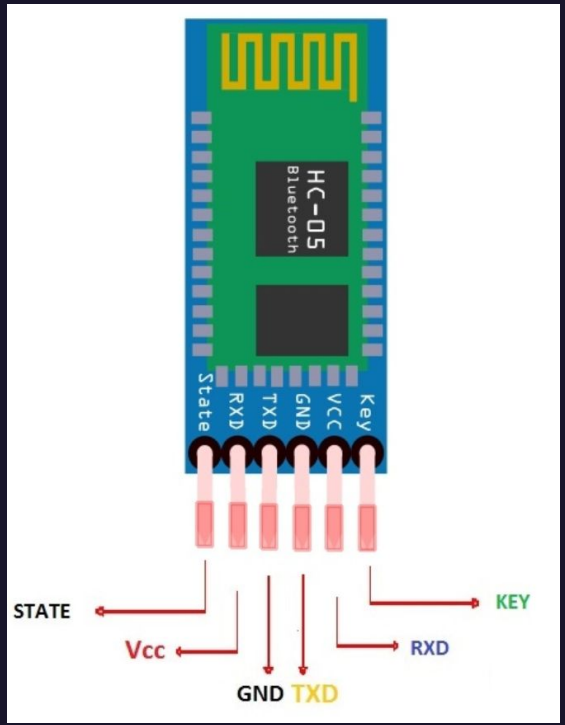
Bluetooth Module (HC-05)

This allows the arduino to communicate using bluetooth, i.e., primarily allowing it to take input from and giving output to connected devices.



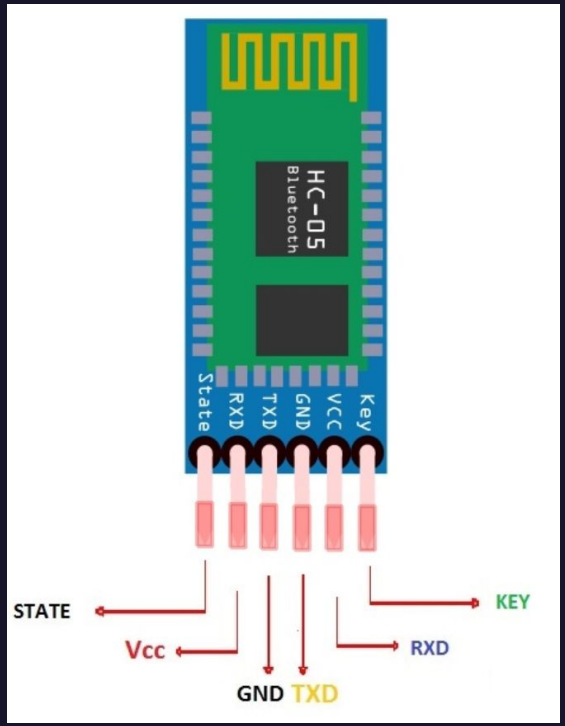
Bluetooth Module (HC-05)

The VCC pin is connected to a 5V power source, GND to ground, and the RXD and TXD pins are connected to the TXD and RXD pins of the Arduino respectively.



Bluetooth Module (HC-05)

One important point to remember is that when uploading a code to the Arduino, make sure that the RXD and TXD pins of the Arduino are free.



Bluetooth Car

Connecting the HC-05 and the motor driver to the Arduino and uploading the code completes the build of the bluetooth car. The next step is to install any bluetooth terminal app (ex: BT Terminal) on your phone and connect your phone’s bluetooth to your HC-05 and pass commands to the Arduino via the app.

PES UNIVERSITY

IEEE Robotics & Automation Society
PES UNIVERSITY
Student Chapter

Pyserial

```
import serial

ser = serial.Serial('PORT', baudrate = 9600, timeout = 1)

i = ""
while i != 'done':
    i = input()
    ser.write(i.encode())
```