MS101 Makerspace

Expt. 4: Arduino Familiarization and Motor Control

Department of Electrical Engineering
Indian Institute of Technology, Bombay

February 2025

Arduino Nano Board

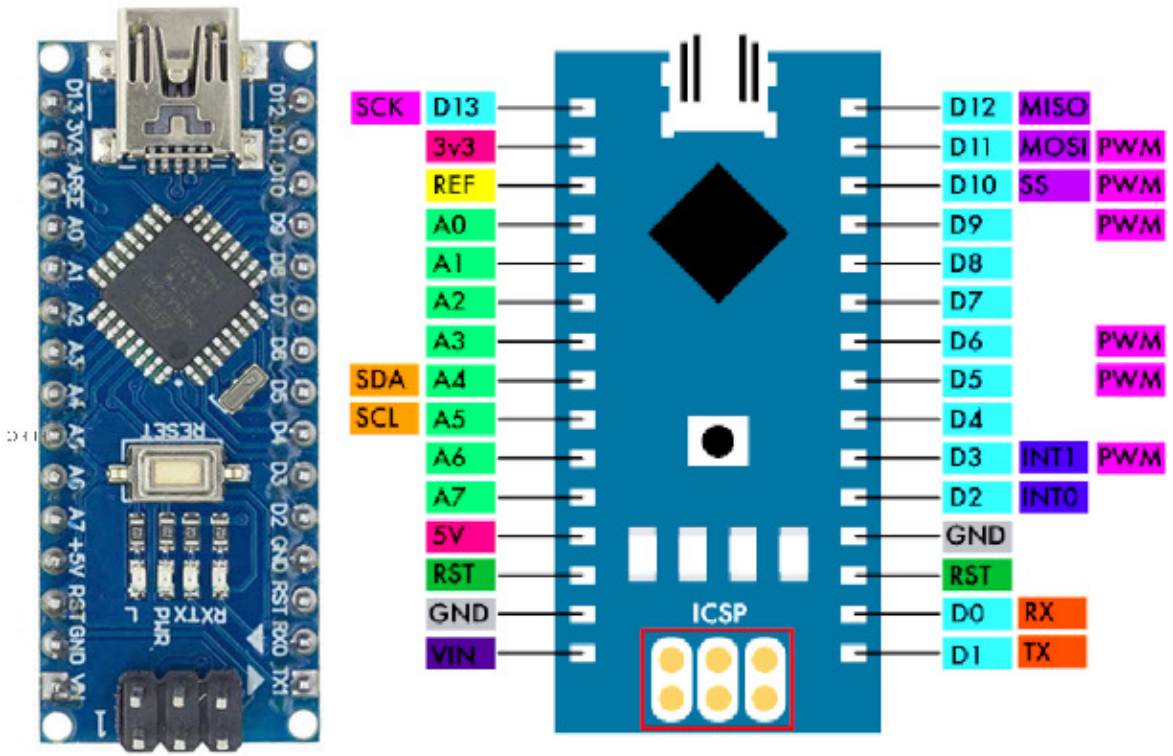


Diagram showing pins

Arduino Nano uses a USB mini connector to connect to a laptop/PC.

It derives power through the USB.

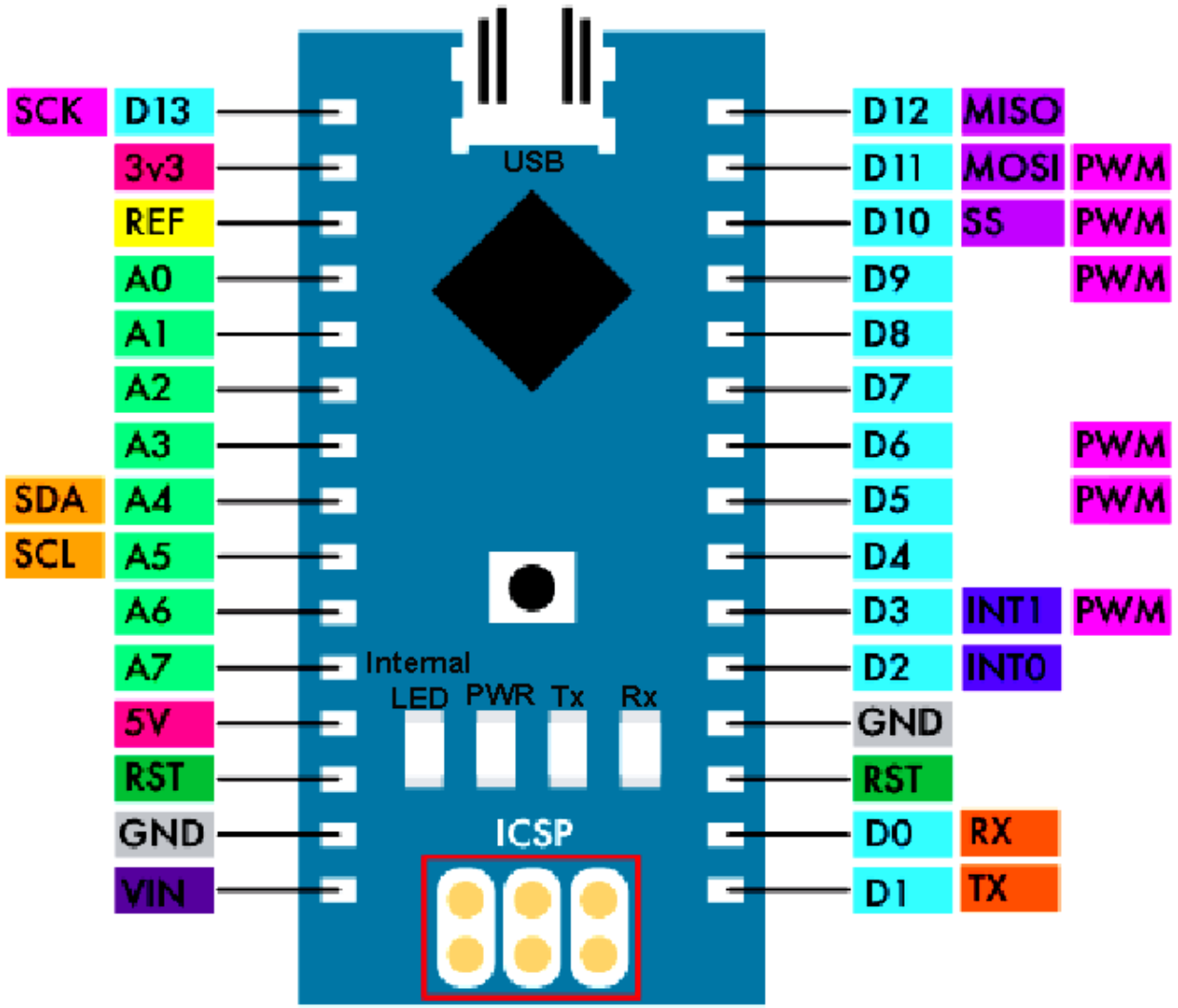
The board can be plugged directly into a breadboard for trial circuits.

Some of the inexpensive nano boards need special drivers (CH 340) with Windows. The driver can be downloaded from:
https://www.wch-ic.com/downloads/CH341SER_ZIP.html
A tutorial for installing it is available at:
https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all

Powering up the Arduino board

Power to the Arduino Nano board may be supplied:

1. Through the USB connector. (Nano uses a mini USB connector)
2. Through the Vin pin (bottom left in the figure to the right).



The “power on” LED will light up when power is applied.

Programming Arduino Boards

- Arduino boards are programmed using an Integrated Development Environment (IDE) running on a laptop or PC.
- Detailed instructions were put up on moodle for installing the IDE depending on your operating system.
- YOU SHOULD HAVE INSTALLED THE IDE ON YOUR LAPTOP BEFORE COMING TO THE LAB.
  Note that the current version of the IDE is version 2.3.0.
- In order to communicate with your board, the IDE needs to know
  1. The type of your board (Uno/Nano/. . .), and
  2. The serial line you will use for connecting the card to the laptop/PC.
- The write up on moodle provides all details which are required for setting up these two parameters for the IDE.

Programming Arduino Boards: Some terminology

Arduino programs which follow a pattern suitable for event-driven applications are called *sketches*.

- A sketch has two parts – the initialization part which is run first (and only once), and the repetitive part which runs in a perpetual loop.
- You need to define two functions (called “setup” and “loop”) which provide these two parts. These functions receive no parameters from the calling code and return nothing to the caller (void type).
- These functions are to be written in Arduino flavour of C or C++.
- Many pre-defined functions for specific actions are available as libraries. These functions can be called from your **setup( )** or **loop( )** code, which makes it easy to write software for applications.

Experiments for Lab. 4

This experiment has two parts: part A is for familiarization with Arduino board, its Integrated Development Environment (IDE) and software. Part B is for controlling motors using Arduino boards.

In part 4A, we shall carry out 5 simple experiments using Arduino.

1. LED Blink experiment for Digital output – to be done before coming to the lab to test your board and IDE installation.
2. LED fade experiment for Pulse Width Modulated outputs.
3. Switch reading (digital input) and serial output,
4. Reading voltage output from a potentiometer as an illustration of analog input.

For this lab, we'll use ready made example programs available in the Arduino IDE with minimal modifications.

Experiments for Lab. 4

Before starting the experiments, it is assumed that

1. you have read this document and an earlier file (Introduction to Arduino) that we had put on moodle.
2. You have acquired an Arduino board.
3. You have installed and checked the IDE according to your operating system.
4. You have checked the Arduino board by connecting it to your laptop using an appropriate type of cable – USB A on the laptop side and USB B on micro-controller side for Arduino Uno and a mini USB connector for Arduino Nano.
5. You have run the blink experiment (described next) **before** coming to the lab.

Connect the board to your laptop through the USB cable and start the IDE according to the conventions of your OS (for example, by double clicking on the program name).

Expt. 4 – IDE set up

- In the IDE, click on the down arrow in the window labelled as “Select Board” and choose to “Select Other Board and Port”.
- In the search window, type Nano and the board option for Arduino Nano will come up. Click on it and a tick mark will appear against the name.
  (You could also scroll down the alphabetical list of boards to select your board).
- Click on the serial port shown on the right half of the window. A tick mark will appear here as well. Click on OK to select these choices.
- The code for a previously run program may be already displayed in the editing window. However, we shall begin with a fresh ready-made program.

Expt. 4A-1 – Digital Output and delay: Blinking an LED

Objective: Familiarization with Arduino and its IDE.
This experiment does not require any external circuit or instruments. You should perform it **before** coming to the lab to check that your board is working properly and the IDE is installed correctly.

- Click on the File tab and choose “Examples->01-Basics->Blink. A fresh edit window will open with the sketch for Blink loaded in it.
- Read through the comments to understand what the program is supposed to do and notice the function calls used.
- Click on the “Sketch” tab and choose to verify/compile. The IDE responds with “Done compiling”. (A short cut for this is control-R).
- Click on the “Sketch” tab again and choose to “Upload”. The IDE responds with “Done uploading”. (A short cut for this is control-U).
- The $\rightarrow$ icon on top can be used to compile and upload in one go.
- The Blink program starts running. Admire your handiwork lovingly.

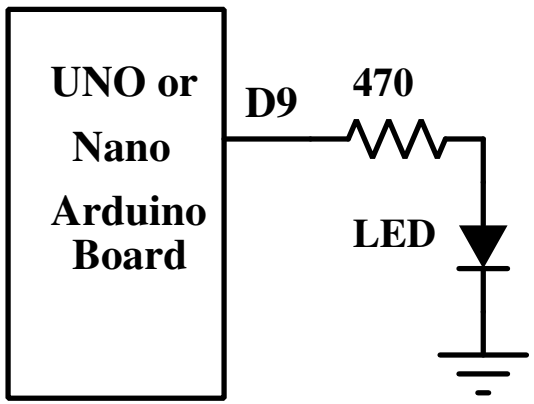
Expt. 4A – Digital Output and delay: Blinking an LED

Now we'll tinker with the sketch . . .
This part will be done in the lab.

- Change the on-time/off-time to 500 ms/1500 ms in the sketch by modifying the appropriate numbers in the code.
- Compile and upload again and run it to observe a different ratio of on and off time for the LED.
- Change the on time/off-time to 1500 ms/500 ms, recompile, upload and run.
- Call your TA to participate in your excitement . . .
  (who may spoil the fun somewhat by asking you about the function calls used and the structure of the program).
- Have your lab book signed for completion of Experiment 4A-1.

Expt. 4A-2 – Pulse Width Modulation: Fade

Objective: Use of Pulse width modulation (PWM) to control power given to an external device.
This experiment requires making a small circuit on your breadboard.



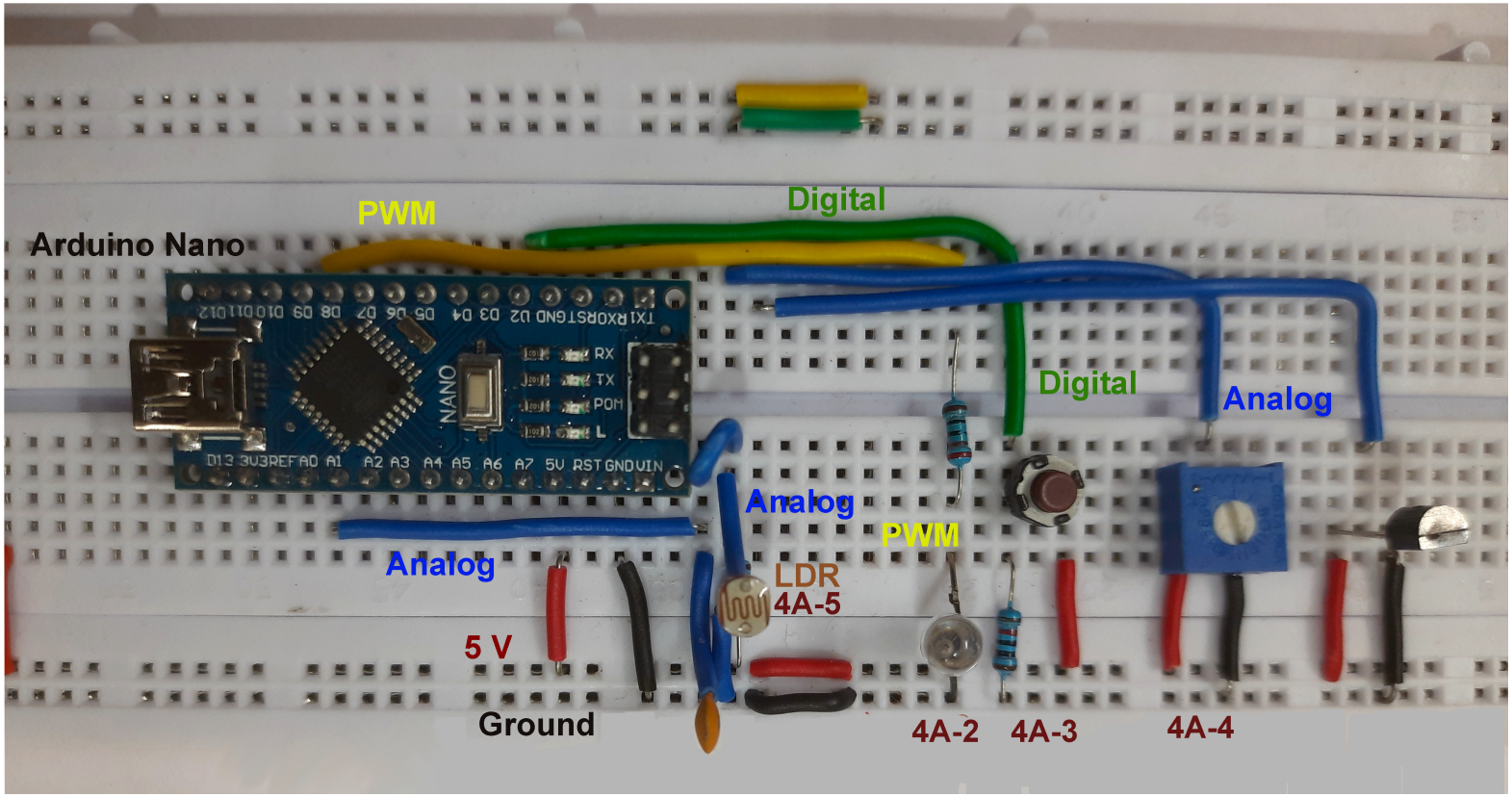
The on-board LED is connected to pin 13, which does not support Pulse Width Modulation. Hence the need for this circuit.

- Connect a wire from Digital pin D-9 on your UNO/Nano board to the breadboard.
- Connect a resistor (any value between 220Ω to 470Ω will do) from this point to the anode of an LED (its longer lead).
- Return the cathode of the LED (its shorter lead) to a long line of the breadboard for the ground connection.
- Connect a (black) wire from this long line to the GND terminal of Arduino (in the group of pins labelled as “power”).

Done!

Expt. 4A-2 – Pulse Width Modulation: Fade

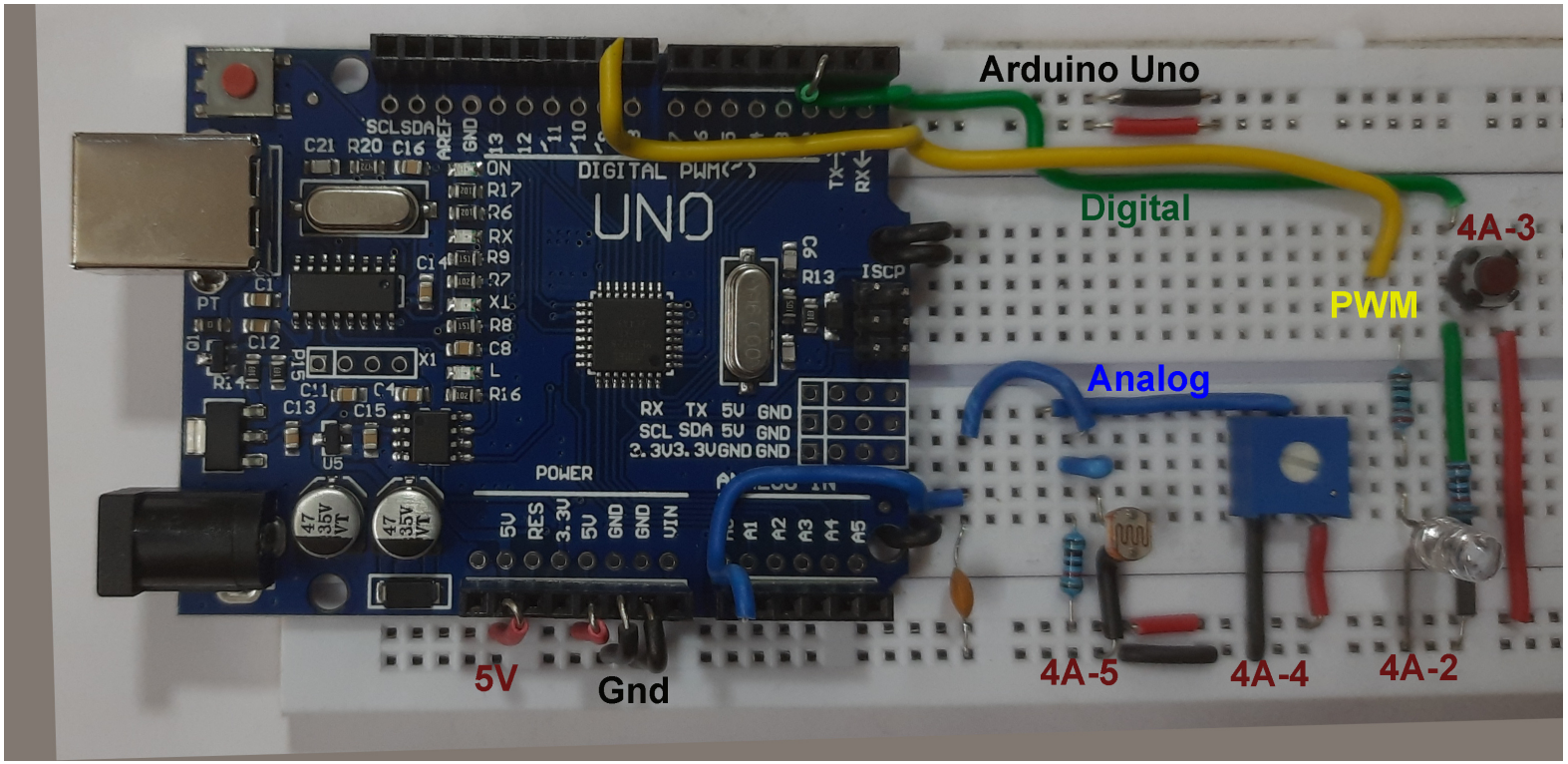
This is how I wired up all parts of expt. 4 on a breadboard, using a Nano board.



(Beware – due to parallax, wires may appear to go to the wrong pins on Arduino – follow the text, not what you perceive from the figure.)
Wire colours: Red: 5V, Black: Ground,
Green: Digital, Yellow: PWM, Blue: Analog

Expt. 4 with an Uno board

The photograph shows all parts of expt. 4 wired up using an Uno board.



(Beware – due to parallax, wires may appear to go to the wrong pins on Arduino – follow the text, not what you perceive from the figure.)
Wire colours: Red: 5V, Black: Ground,
Green: Digital, Yellow: PWM, Blue: Analog

Expt. 4A-2 – Pulse Width Modulation: Fade

- Click on the File tab and choose “Examples->01-Basics->Fade. A fresh edit window will open with the sketch for Fade loaded in it.
- Read through the comments to understand what the program is supposed to do and notice the function calls used.
- Click on the “Sketch” tab and choose to verify/compile. The IDE responds with “Done compiling”.
- Click on the “Sketch” tab again and choose to “Upload”. The IDE responds with “Done uploading”.
- Again, you could click on the $\rightarrow$ icon to combine the compile and upload operations.
- The Fade program starts running.
  Watch the LED on the breadboard fade in and out. View the waveform on pin D9 using the Digital Storage Oscilloscope (DSO).

Expt. 4A-2 – Pulse Width Modulation: Fade

- Modify the sketch to use a different fading increment.
- Re-compile, upload and run. Watch the effect of slower and faster fading on the LED.
- Define separate values for increment/decrement amount so that the rate of fading is not symmetrical during fade in and fade out phases. (For this, you will have to modify the program code).

For code modification, you will define two values for fade amounts before the setup( ) function.

```
int brightness = 0; // how bright the LED is
int fd1 = 5; // how many points to fade the LED by
int fd2 = 20;
int fadeAmount = fd1; // Initial fading rate
```

(Only the changed part of the program has been described above)

Expt. 4A-2 – Pulse Width Modulation: Fade

In the loop( ) function, you have to check for two termination conditions:

```
 //change the brightness for next time:
brightness = brightness + fadeAmount;
// reverse the direction of the fading
// when you reach max brightness
 if (brightness > 255) {
     brightness = 255-fd2;
    fadeAmount = -fd2;
  }

 if (brightness < 0) {
   brightness = fd1;
  fadeAmount = fd1;
}
// wait for 50 milliseconds per brightness change
delay(50);
```

Expt. 4A-2 – Pulse Width Modulation: Fade

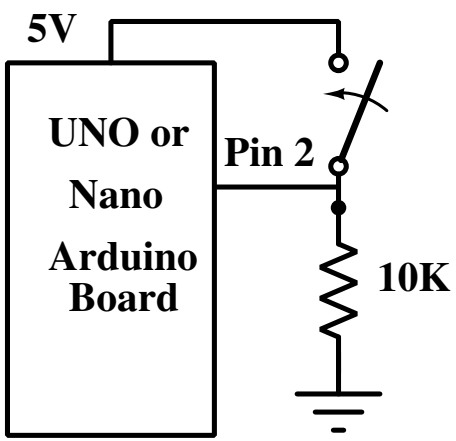
After modifying the code,

- Re-compile, re-upload, re-run, and rejoice.
- View the waveform on pin 9 using the DSO.
- Draw this in your note book. (Use waveforms with dotted/dashed lines to show how the pulse width is changing).
- Call the RA/TA to show your program running with asymmetrical fading rate and get the completion of Experiment 4A-2 signed.

Do not dismantle this circuit when you go to the next part!

Expt. 4A-3 – Digital Input, Serial Output

We need to make another simple circuit for this experiment.



Digital pin 2 will be used as the digital input. The circuit shows a switch, but we can simulate the switch by just touching a wire manually.

Connect the ground pin (among the power pins of Arduino) to the long ground line of your breadboard. Plug in a 10KΩ resistor between ground and any group of shorted 5 pins.
(Any value between 10KΩ and 100KΩ will do)

Connect a wire from digital pin 2 to the top of this resistor. Connect a wire from the 5V output in the group of pins marked “power” on Arduino.

We’ll touch the other end of this wire by hand to the top of the resistor to simulate a switch.

Expt. 4A-3 – Digital Input, Serial Output

- Click on the File tab and choose “Examples->01-Basics->DigitalReadSerial.
  A fresh edit window will open with the sketch for DigitalReadSerial loaded in it.
- Read through the comments to understand what the program is supposed to do and notice the function calls used. Pay particular attention to the way the serial link will be established between the laptop and Arduino.
- Click on the “Sketch” tab and choose to verify/compile. The IDE responds with “Done compiling”.
- Click on the “Sketch” tab again and choose to “Upload”. The IDE responds with “Done uploading”.

Expt. 4A-3 – Digital Input, Serial Output

- To see the measured values, open the serial monitor.
  For this, click on the Tools tab and then on the Serial Monitor choice. This will open a window below the edit window to show serial communication between the Laptop and Arduino.
- Watch the string of ‘0’s appearing in the serial monitor. This is because Arduino is reading a Low voltage on pin 2 through the grounded 10KΩ resistor.
- Press the push button (or touch the wire you had connected from the 5V output pin on Arduino to the top of 10KΩ resistor). You will see a series of ‘1’s now in the serial monitor window.

On the top right of IDE window are two icons.

These can be used to invoke the digital monitor and digital plotter provided by the IDE.

Expt. 4A-3 – Digital Input, Serial Output

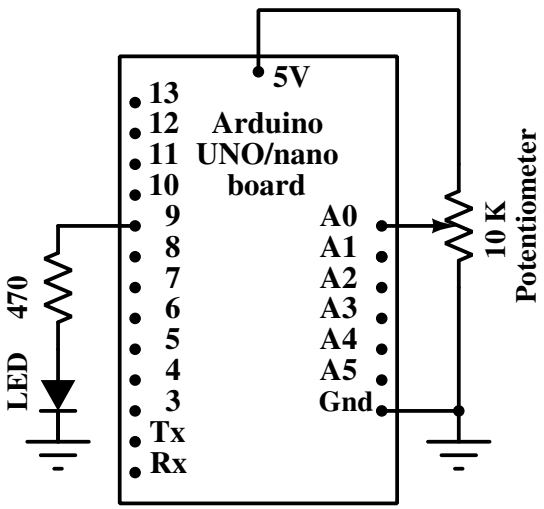
- Press and release the push button (or touch and remove the 5V wire from the top of the 10KΩ resistor) and check that the digital value read on pin 2 changes from ‘0’ to ‘1’ and back.
- Call your TA and claim that you have learnt:
  1. how to read digital input,
  2. how to send data through the serial line, and
  3. how to plot incoming data from Arduino on the laptop.

(The TA may be tempted to test whether you have actually learnt it)!

Get your lab book signed for completion of Experiment 4A-3.

Expt. 4A-4 – Analog Input, Serial Output

Another experiment . . . another circuit.
You are now veterans of making circuits on the bread board.
So just make the circuit shown below without detailed instructions.



- Do not forget to connect the 5V and Ground lines to the 5V and ground pins in the “power” group of pins on the Arduino board.
- The breadboard compatible potentiometer covers the nearby pins – so make a note of the label on the group of 5 pins on the breadboard to which each of its terminals is connected.

The LED connected to pin D9 through a (≈ 470)Ω resistor should still be there. (If you removed these by mistake, connect these again as shown).

Expt. 4A-4 – Analog Input, Serial Output

- Click on the File tab and choose “Examples->03.Analog->AnalogInOutSerial. A fresh edit window will open with the sketch for AnalogInOutSerial loaded in it.
- Read through the comments to understand what the program is supposed to do and notice the function calls used.
- Pay particular attention to the way the range of 10bit ADC on AVR 328P is mapped to a range of 8bits (0 to 255), and how a serial link is established between the laptop and Arduino.
- Click on the “Sketch” tab and choose to verify/compile. The IDE responds with “Done compiling”. (Short cut: control-R).
- Click on the “Sketch” tab again and choose to “Upload”. (Short cut: control-U). The IDE responds with “Done uploading”.

Expt. 4A-4 – Analog Input, Serial Output

- Open the serial monitor. For this, click on the Tools tab and then on the Serial Monitor choice. (You can also click on the icon at the top right of the IDE). This will open a window below the edit window to show serial communication between the Laptop and Arduino.
- Adjust the position of the potentiometer by rotating the slider using a small screw driver. Watch the intensity of the LED being controlled by the position of the potentiometer.
- Also see the sensor output values (ADC outputs with a 10 bit range and the mapped values to 8 bit ranges) being sent through the serial port. These appear in the serial monitor. You can also use the plotter function to see how these change when you turn the potentiometer slider.

Show the operation to your TA and explain how it all works. Get your notebook signed for observations of experiment 4A-4.

Motor Speed Control with PWM using an Arduino Board

In this part of the experiment, we'll learn how to control the speed of a motor using an Arduino board.

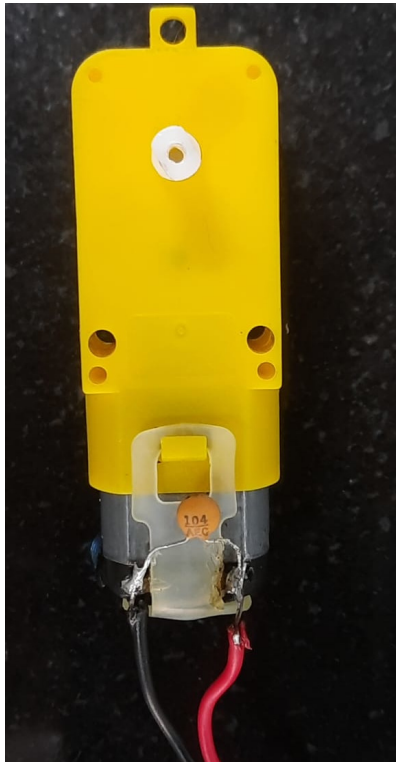
We need a motor driver card and battery operated (BO) motors for this part.

First, we have to assemble the set up. The various steps for this are:

- Soldering a capacitor and wires to the Battery Operated (BO) motor.
- Mounting the motor on a stage and fixing an encoder disk on its axis.
- Mounting a reflective sensor with an LED and photo-diode which gives digital outputs of '1'/'0' as the encoder disk rotates.
- Connecting the motor to its driver and connecting the driver to Arduino Nano card.

Soldering a Capacitor across BO Motor Terminals

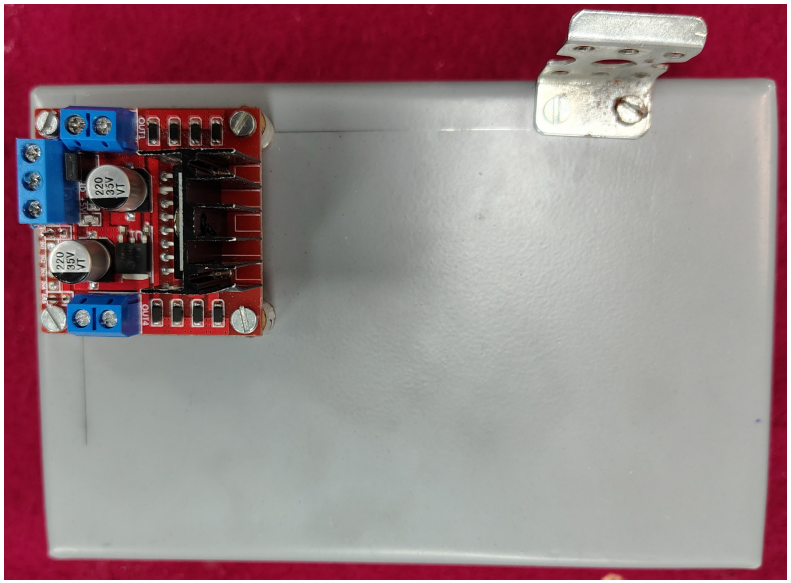
- Before we connect the motor to our circuits, we need to solder a 0.1μF ceramic capacitor across its terminals to suppress noise.
- We also need to solder wires to the motor terminals, so that we can connect it to the motor driver.



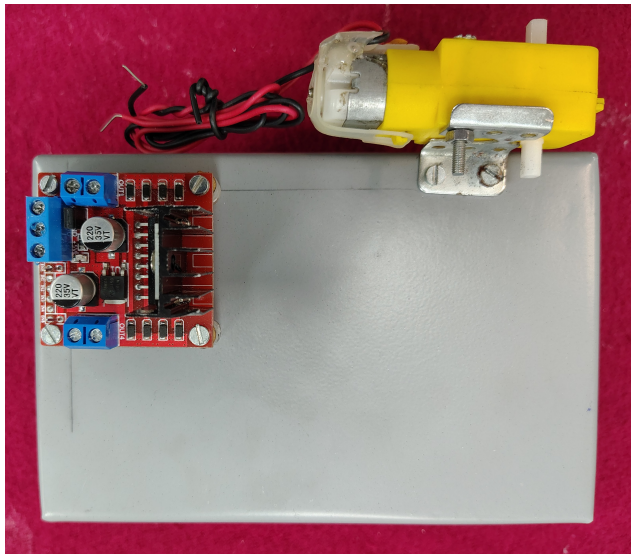
Solder the capacitor and wires to a BO motor.
Seek help from your TA/EE staff if you are unable to solder these neatly.

Mounting the motor on a platform

You are given a platform with the motor driver mounted on it and a bracket to mount the BO motor.



Platform with mounted motor driver and bracket

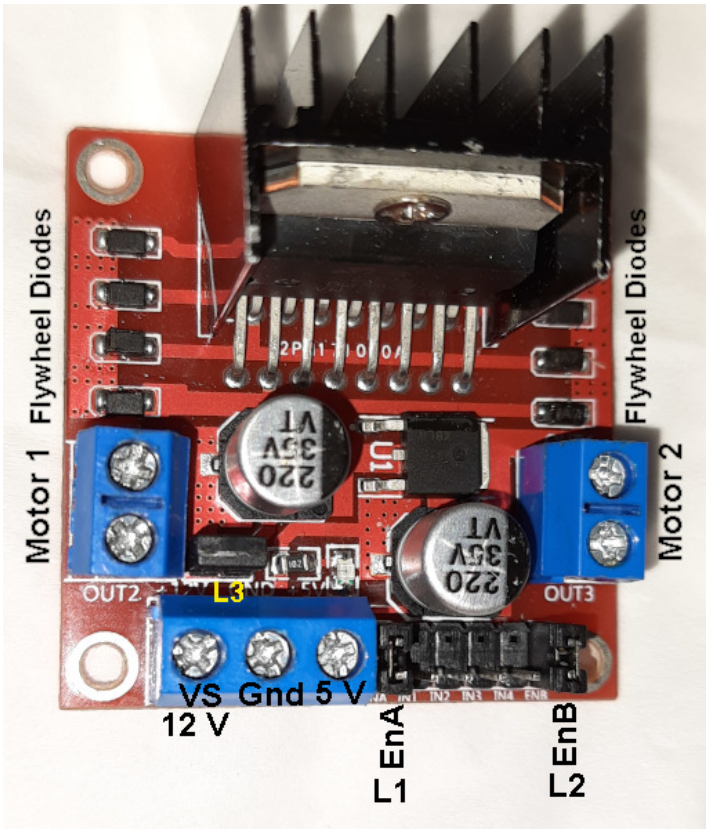


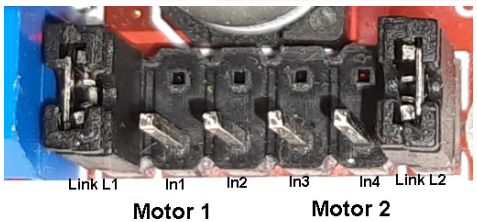
BO motor mounted on the platform

- Mount the BO motor on the bracket using nuts and bolts provided to you, as shown in the figure above.
- If your motor has a single shaft, it should be pointing outwards.
- Wires soldered to the motor will be connected to the motor driver board.

Configuring the Motor Driver Board

We shall use the motor driver L298. This board can drive two BO motors, but we are going to use only one motor in this experiment.

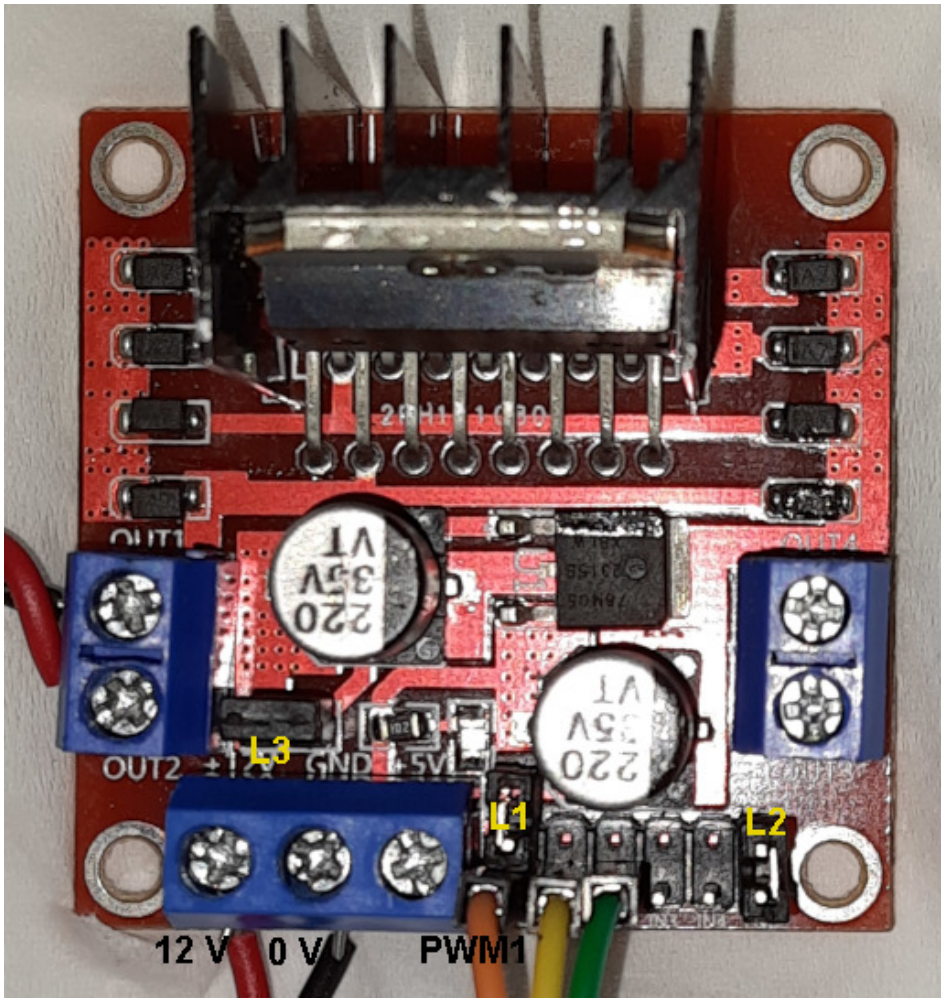




The board has 3 “**links**”. (A link is a little plastic part with metal inside, used to connect two pins to each other). Locate the three links:

1. Link L1 is located just to the right of the power block on the bottom, between the power block and In1 connection.
2. Link L2 is at the bottom right just above the PWM2 pin.
3. Link L3 is at left, just above the power block and to the right of OUT2 motor connection.

Using Links to configure the motor driver



While removing a link, if you just take it off, you are likely to lose it.

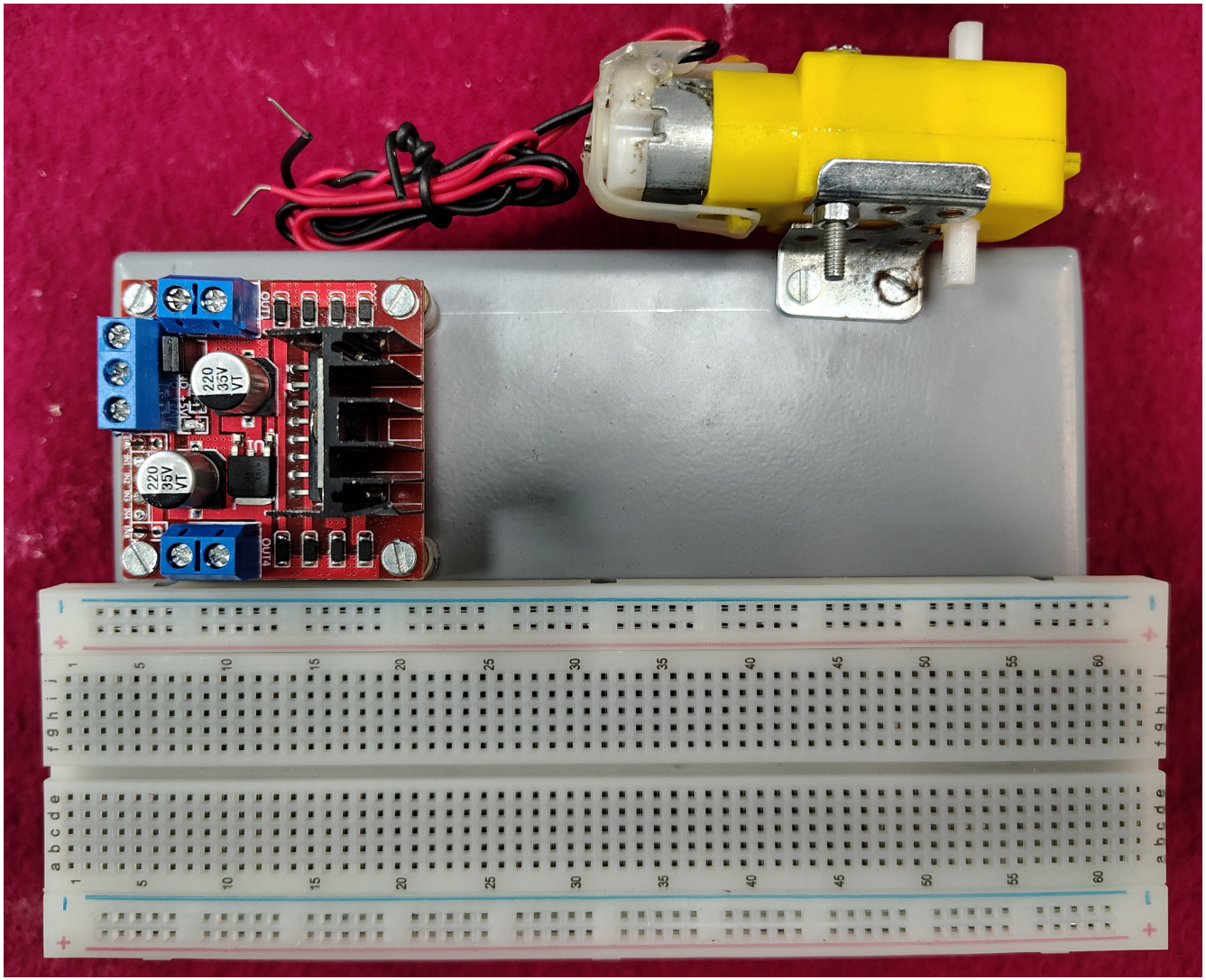
To remove a link, pull it out and plug it back such that it covers only one of the two pins over which it was originally placed.

(See link L1 which has been moved up by one position.
This exposes the pin previously shorted by the link and now has an orange wire labelled PWM1 connected to it.)

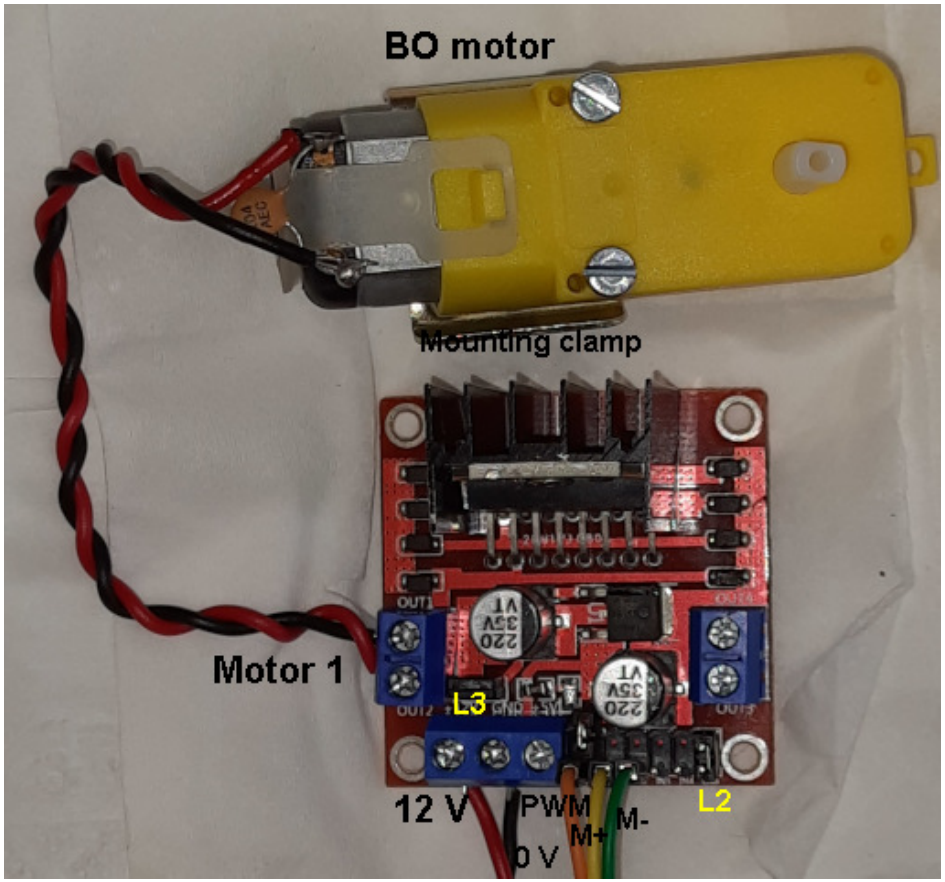
Breadboard for connecting Arduino Nano

We shall use an Arduino Nano board for controlling the motor.

The Nano card will be plugged into a breadboard, which will be placed on the platform as shown in the figure.
The breadboard can be stuck to the platform using a double sided sticking tape.

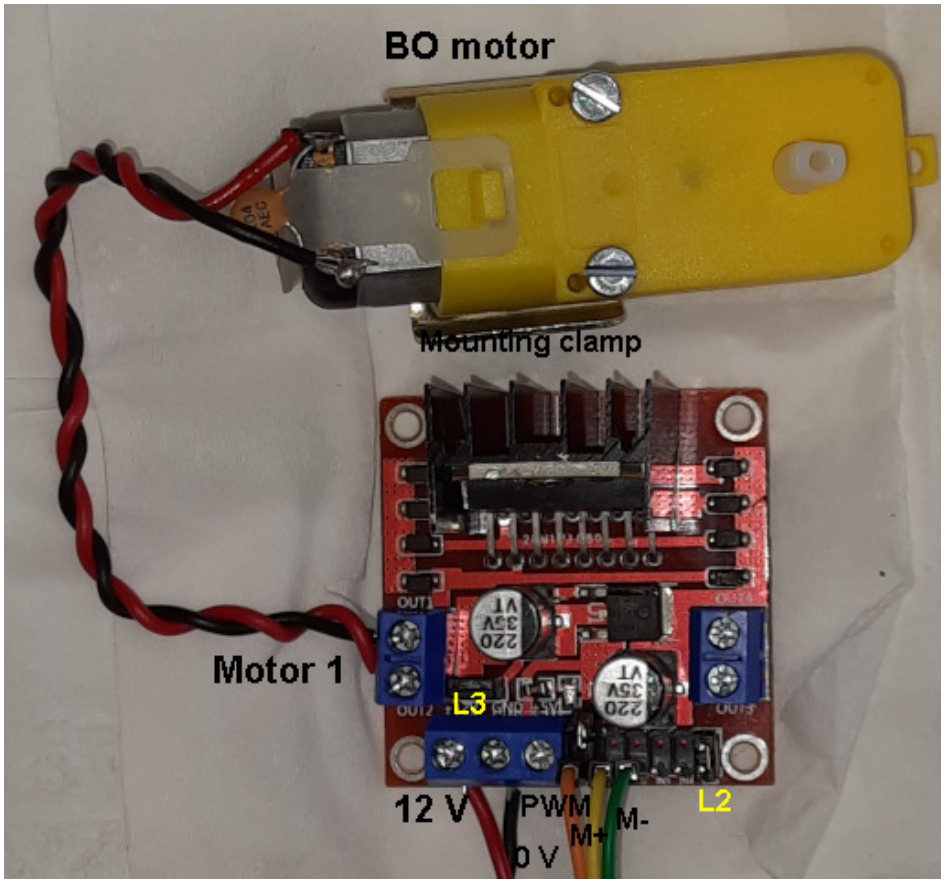


Connecting Motor and Power to the Driver Board



- The motor leads (soldered earlier) should be connected to Motor 1 terminals (OUT1 and OUT2) on the L298 motor driver board.
- Connect a DC power supply (programmed to 12 V) to the V_S and Ground terminal blocks on the L298 card. The third terminal (meant for 5 V DC) **should be left open**.
- Connect the ground terminal of L298 card to the ground pin of Arduino.

Connections between Motor Driver and Arduino



- Remove Link L1. (The link connects the enable input for motor A to 5 V generated by on-board 5 V regulator).
- The pin (which was connected to on-board 5 V through the link earlier) should now be connected to pin 6 of Arduino for PWM control.
- Connect the next two pins (IN1 and IN2) on the motor driver to pins 10 and 11 of Arduino. (Yellow and Green wires in the picture).

Female to male hook up wires should be used for these connections.

Connection Check List

Links: L3 in; L1 removed.

Power:

Driver card	Power supply
V_S (10-12 V)	Supply+
Ground	Supply−
5 V Terminal	**Open**

Motor to Driver card

OUT1, OUT2 - Motor wires

Motor card to Arduino

Driver Card	Arduino UNO/nano
Ground	Gnd
Enable A	Pin 6
IN1	Pin 10
IN2	Pin 11

After making all the connections, have these checked by your TA.

Sketch for Rotating the Motor

We are all set to rotate the motor.

We shall use the provided sketch (sketch-expt4B-1.ino) to do this. The sketch defines a PWM value in the code (at the end of the sketch) and runs the motor with this PWM value.

If the defined PWM value is negative, it rotates the motor in the opposite direction.

- Connect your Arduino to the laptop **without powering on the motor driver.**
  (Keithley Power supply output should be OFF).
- Enter, compile and upload the provided sketch (sketch-expt4B.ino).

Rotating the Motor

We'll use 12 V power from the Keithley power supply (Channel 1 or 2).

- Set the voltage on the selected channel to 12 V.
- Program Iset to 1A on this channel and turn ON the output.

The power LED on motor driver should glow.
If it does not, turn everything off and re-check your connections.

If all is well, **your motor will rotate!**

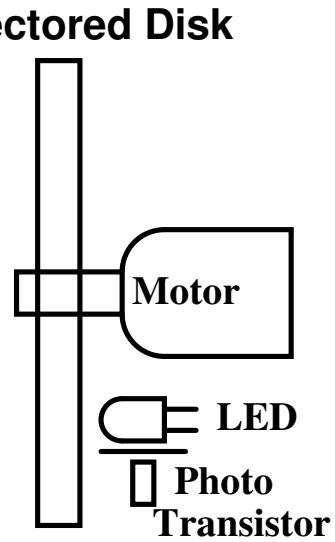
Effect of PWM Value on Motor Speed

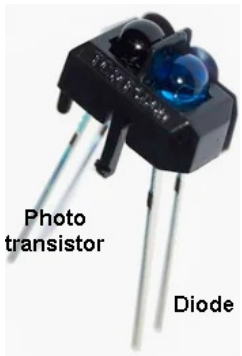
When you have stopped jumping for joy, call the TA and show your success to him/her. Have your lab book signed that your motor connections are right and the motor is rotating.

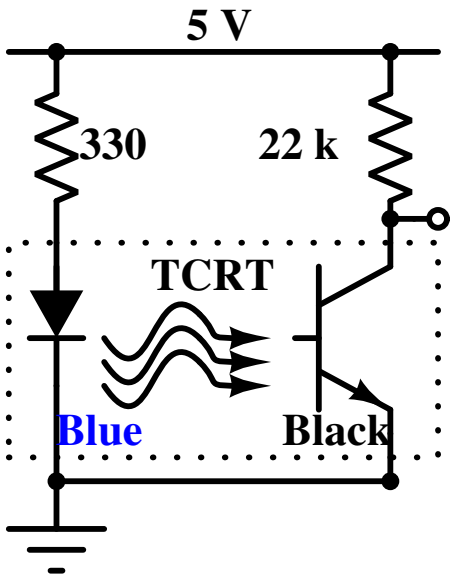
Try various positive and negative values of PWM.
(Absolute value of PWM has to be ≤ 255).
For this you will have to edit the sketch, altering the value of pwmValue near the end of the sketch every time, and re-uploading the binary to Arduino.

After that, Disconnect the power from the motor driver –
We have more work to do . . .

Measuring Rotation Speed

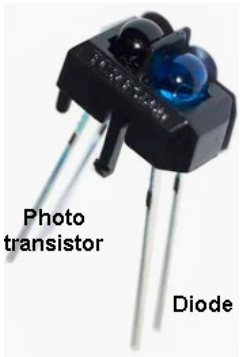


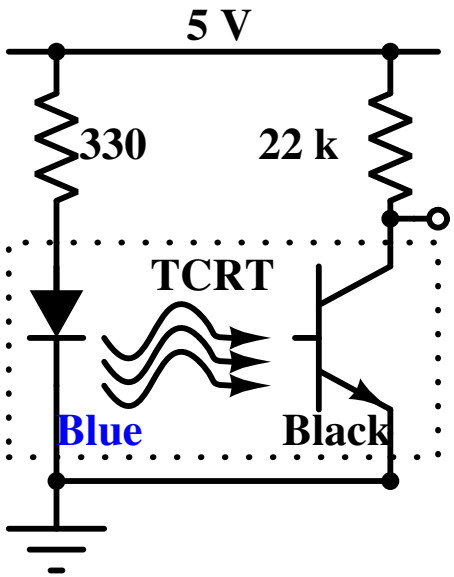


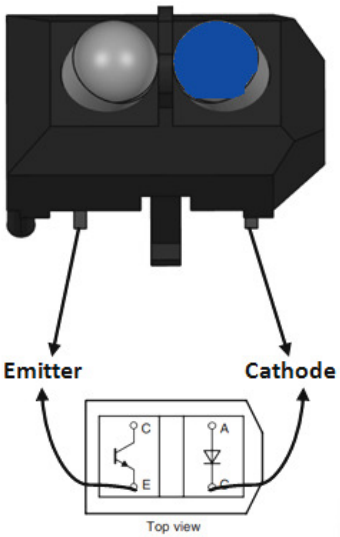


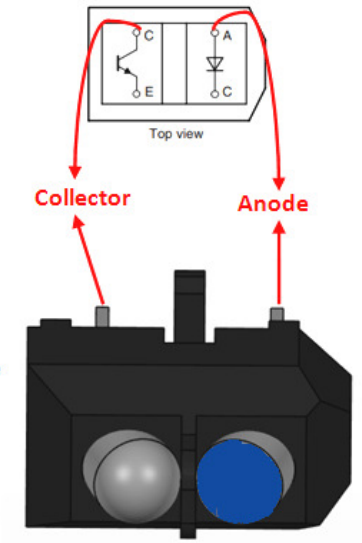
- Mount the encoder disk with black and white sectors on your motor
- We shall use an infrared LED/photo-transistor pair (TCRT sensor) to detect the rotation of this disk.
- The circuit for the TCRT sensor is shown on the left. This circuit is given to you as a pre-soldered PC board.
- You only have to connect the power and ground pins from the PC board to your circuit.

Measuring Rotation Speed









- Connect the power and ground terminals of the TCRT sensor circuit board to 5V and Gnd terminals of Arduino Nano.
- Run the sketch sketch-expt4B-1.ino and Power the motor on as before so that the encoder disk rotates.
- Place the TCRT sensor next to the slots on the rotating encoder disk.
- Observe the waveform at the TCRT output on the oscilloscope.
- Draw the waveform in your record book.

Measuring Rotation Speed

- Due to light beam reflection by rotating sectors on the encoder disk, you will see a (nearly) square waveform on the oscilloscope.
- The time period of this waveform is the time taken for one slot to move into the location of the previous slot.
- Since there are 8 white sectors on the disk, the time for one full rotation is 8 times the time period of the waveform.
- The reciprocal of this is the speed in revolutions per second.
- Speed is often specified in revolutions per minute (RPM). That is just 60 times the revolutions per second that we had computed.

Rotation Speed vs PWM Plot

- Modify the PWM value in the given sketch (sketch-expt4B-1.ino) from 32 to 224 in steps of 32.
- Notice that at low PWM values, the motor may not rotate at all. This is normal.
- Compute Rotation Speed for each value of PWM. Enter these values in your note book as a table and have these signed by your TA.
- Plot Rotation Speed vs the applied PWM.

After this, enter, compile and upload the second sketch given to you. (sketch-expt4B-2.ino).
See the motor accelerate in one direction, decelerate, then accelerate in the other direction and decelerate to idle condition.
See how this is accomplished in the code in the given sketch.