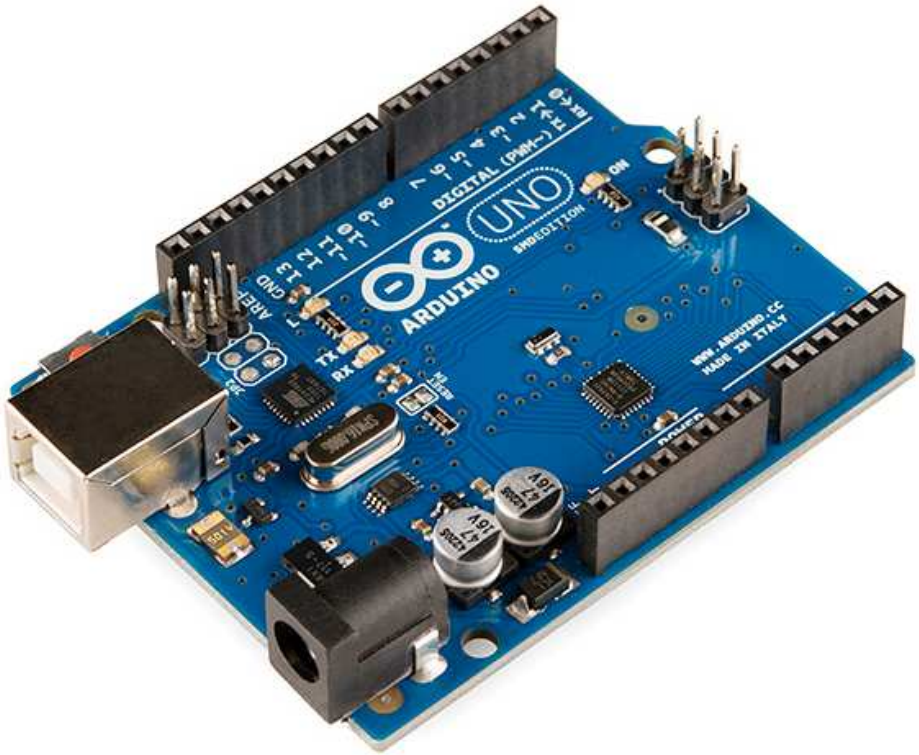
Expt. 4: Arduino board Familiarization

Dinesh Sharma
Joseph John
P.C. Pandey
Kushal Tuckley

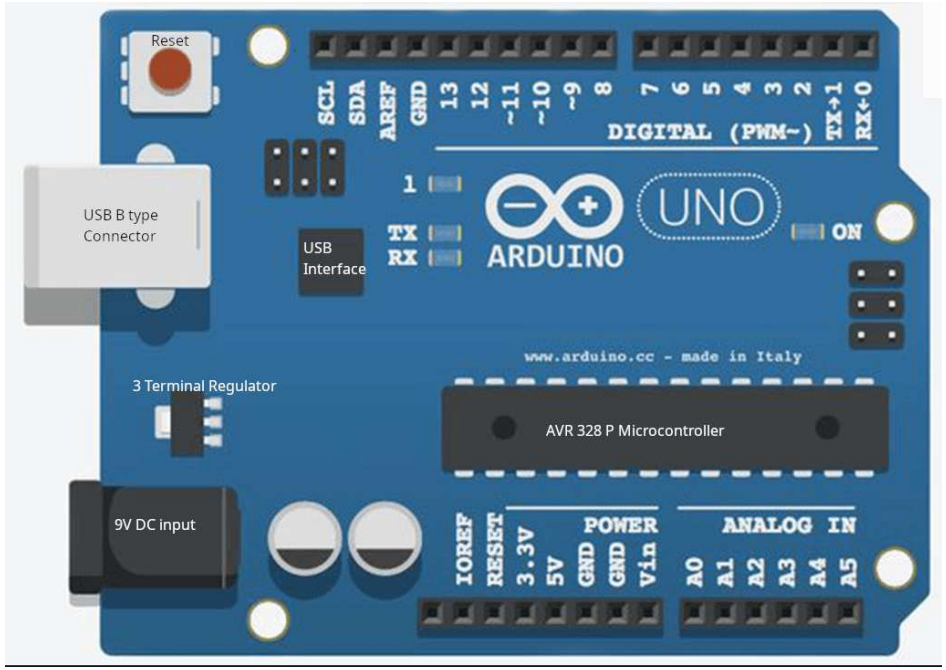
Department of Electrical Engineering
Indian Institute of Technology, Bombay

April 5, 2023

Arduino Uno R3 Board

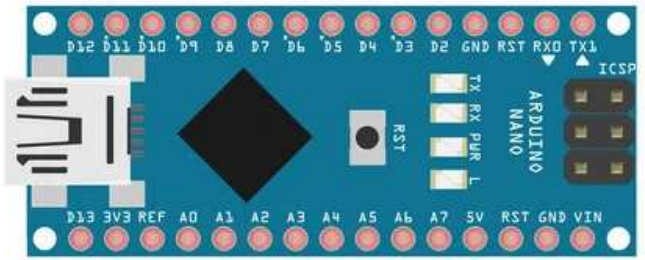


(Arduino Uno
with a soldered Surface Mount
micro-controller chip)



(Diagram showing pins and a
socketed μC chip)

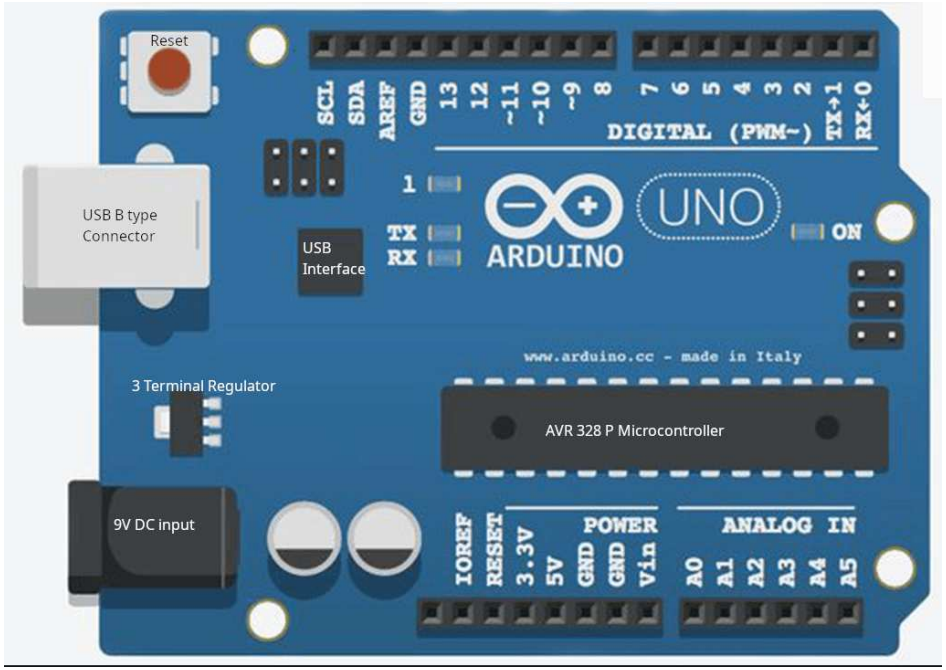
Arduino Nano Board



(Diagram showing pins)

Nano uses a USB mini connector to connect to a laptop/PC.
There is no connector for a wall plug type battery eliminator.
It can be plugged in directly into a breadboard for trial circuits.
Functionally, it is quite similar to Arduino Uno. However some of the inexpensive nano boards need special drivers.

Powering up the Arduino board



The Arduino Uno board may be powered in one of 3 ways; whereas there are 2 options for powering the Nano board.

1. Through the USB connector. (Uno uses a USB B type, Nano uses mini USB)
2. Through the Vin pin (in the group of pins marked as power).
3. Using a wall plug type battery eliminator. (Typically 9V; Not available on Nano).

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 2.0.4.
- In order to communicate with your board, the IDE needs
  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 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

We shall carry out 5 simple experiments using Arduino.

- A: LED Blink experiment for Digital output – to be done before coming to the lab to test your board and IDE installation.
- B: LED fade experiment for Pulse Width Modulated outputs.
- C: Switch reading (digital input) and serial output,
- D: Reading voltage output from a potentiometer as an illustration of analog input, and
- E: Using a light dependent resistor (LDR) to sense light intensity.

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

In the next lab, we shall write our own programs.

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 UNO and the board option for Arduino Uno will come up. Click on it and a tick mark will appear against the name. (You could just scroll down the alphabetical list of boards to select your board – but UNO will be found near the end!).
- 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.
- Notice that the skeleton of a possible program is already displayed in the editing window. However, we shall begin with a few ready-made programs.

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

This program does not require any external circuit or instruments. You should perform this experiment 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 Blink program starts running. Admire your handiwork lovingly.

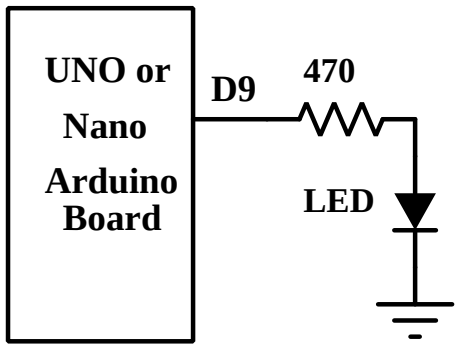
Expt. 4A – Digital Output and delay: Blinking an LED

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 5-A.

Expt. 4B – Pulse Width Modulation: Fade

This experiment requires making a small circuit on your breadboard.



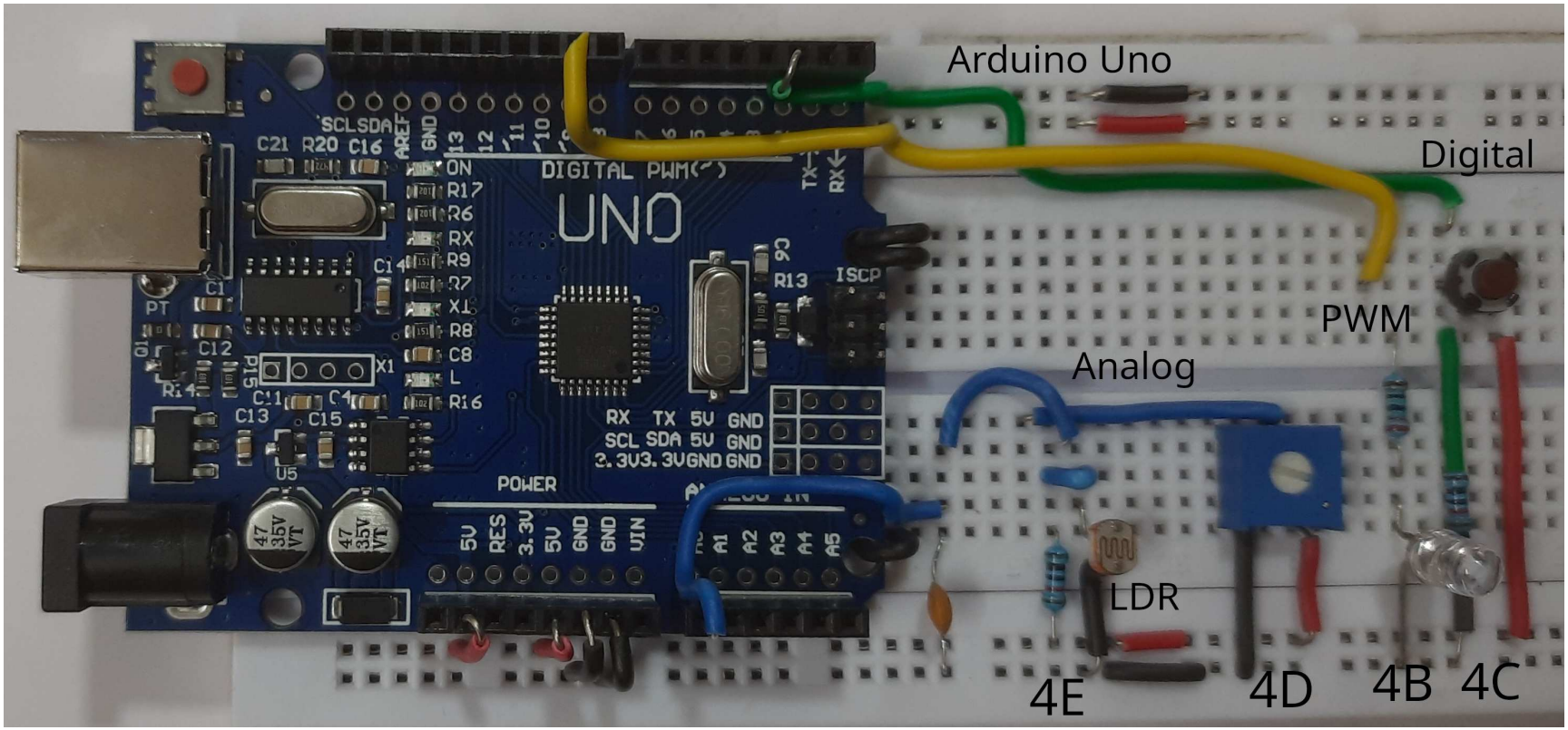
The on-board LED is connected to pin 13, which does not support Pulse Width Modulation. Hence the need for this circuit.

- 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. 4B – Pulse Width Modulation: Fade

This is how I wired up all parts of Expt. 4 on a breadboard.



(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. 4B – 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”.
- 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. 4B – 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 in 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. 4B – 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);\\
```

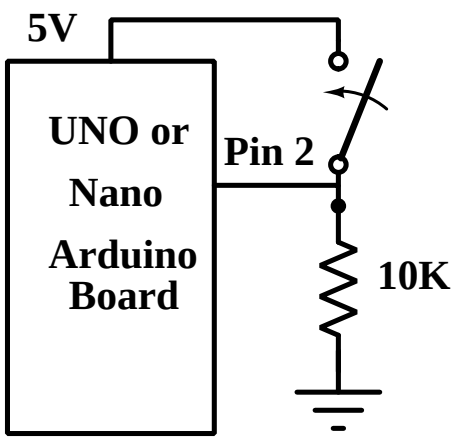
Expt. 4B – Pulse Width Modulation: Fade

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.
- Call the RA/TA to show your program running with asymmetrical fading rate and get the completion of Experiment 4B signed.

Expt. 4C – 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. 4C – 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. 4C – 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.

Expt. 4C – Digital Input, Serial Output

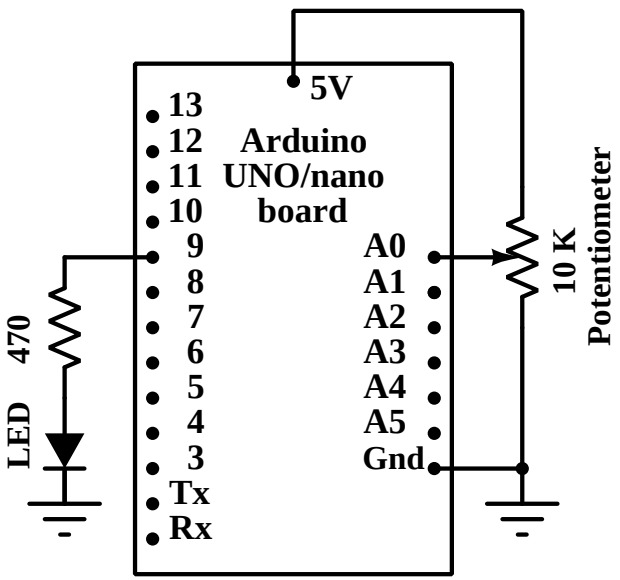
- Press and release the push button (or touch and remove the 5V wire from the top of the 10KΩ resistor) and check that the digital value read on pin 2 changes from ‘0’ to ‘1’ and back.
- Call your TA and claim that you have learnt:
  1. how to read digital input and
  2. how to send data through the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

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

Get your lab book signed for completion of Experiment 4C.

Expt. 4D – 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 note the group of 5 pins on the breadboard to which each of its terminals is connected.
- An LED is connected to pin D9 through a $(\approx 470)\Omega$ resistor.

Expt. 4D – 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".

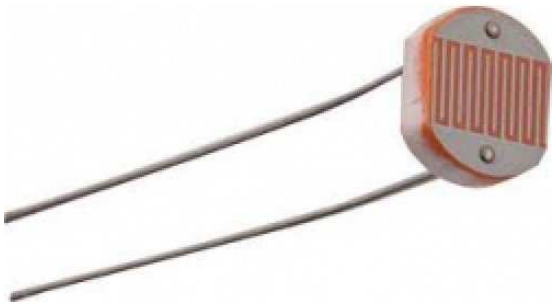
Expt. 4D – Analog Input, Serial Output

- 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.

Show the operation to your TA and explain how it all works. Get your notebook signed for experiment 4D.

Light detection using a Light Dependent Resistor

We'll use an LDR for detecting light intensity.

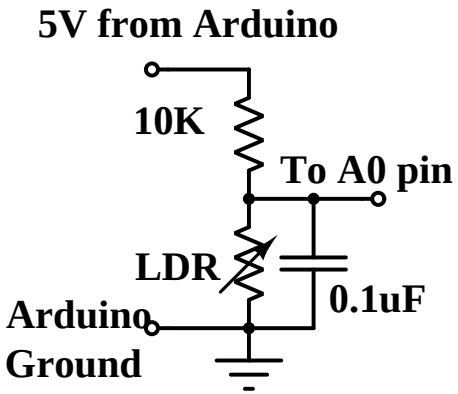


- An LDR is a resistor whose resistance value changes with light intensity.
- The LDR uses a semiconductor instead of a metal or carbon as the resistive material.

- When light falls on the resistor, electron-hole pairs are generated. Since more current carriers are available now, the current increases for a given applied voltage.
- Thus the resistance of an LDR **decreases** with increasing light intensity.
- We can use the LDR as one arm of a potential divider circuit and measure the output voltage to measure the light intensity.

Light detection using a Light Dependent Resistor

Make the potential divider circuit as shown below, on your breadboard. The only change from the circuit for experiment 4D is replacing the potentiometer with the potential divider circuit shown below.



- Notice the 0.1μF capacitor used for bypassing noise.
- 5V and Ground are taken from the Arduino board. The voltage divider output is connected to the analog input pin A0.

We shall use the same program as was used in experiment 4D.

Leave the LED connected to pin 9 through the 470Ω resistor as in experiment 4D.

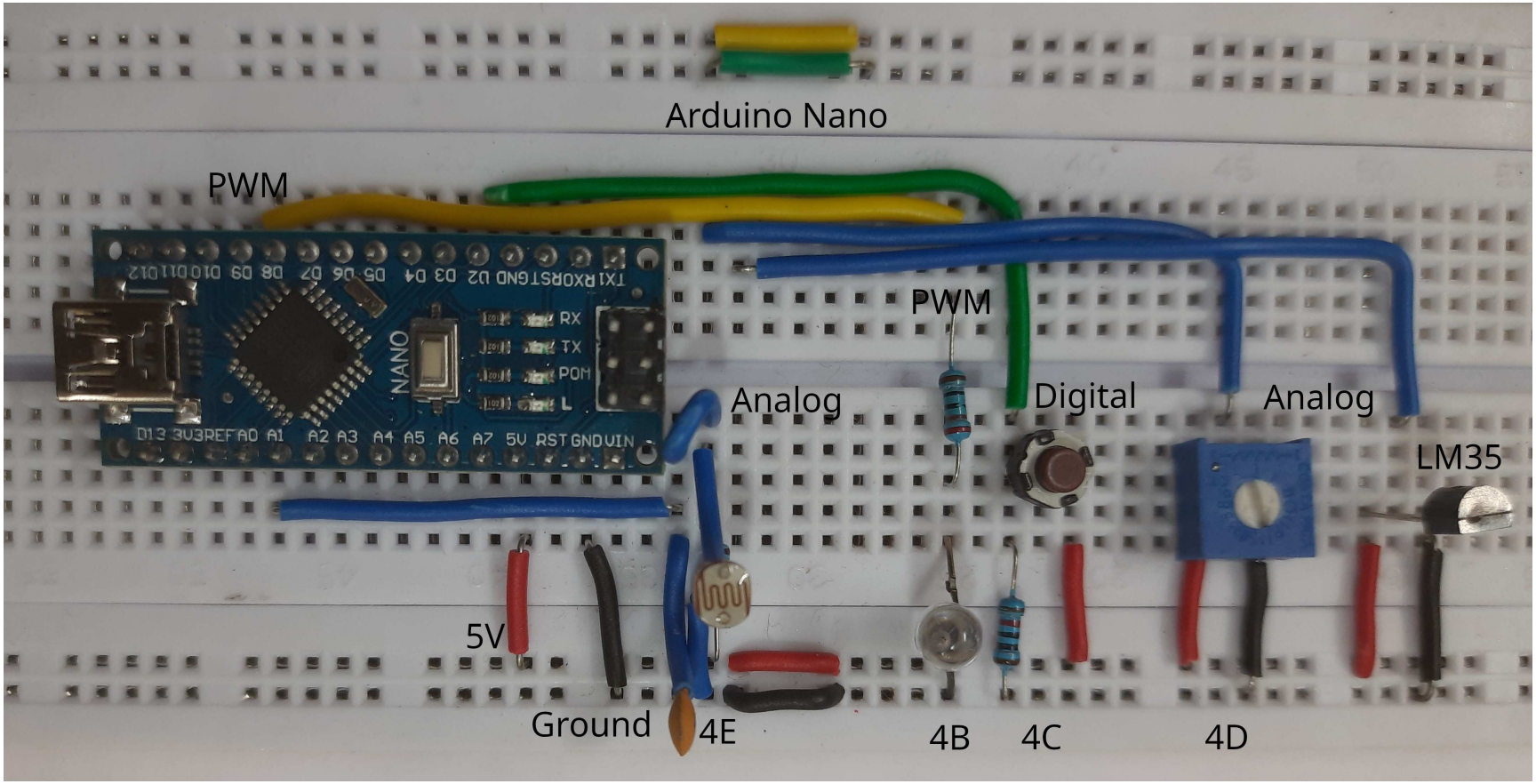
Expt. 4E – Light Intensity Detection

- compile, upload and run your sketch to send the voltage value to the serial monitor.
- Notice that the voltage will decrease as the light intensity increases.
- The effect of light intensity variation will cause the LED driven from pin D9 to be brighter when the light is less and dimmer when there is more light.
- Cover the LDR with your hand and note the output voltage.
- Shine light from your mobile on the LDR and note the output voltage.

Note down and Show your readings to the TA and have these signed.

Can we use this arrangement to adjust the light intensity in a hall?

Expt. 4 with 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