Arduino Uno Microcontroller

Interfacing the Arduino Uno to LEDs and buttons

Developed and written by: Simon Hudson

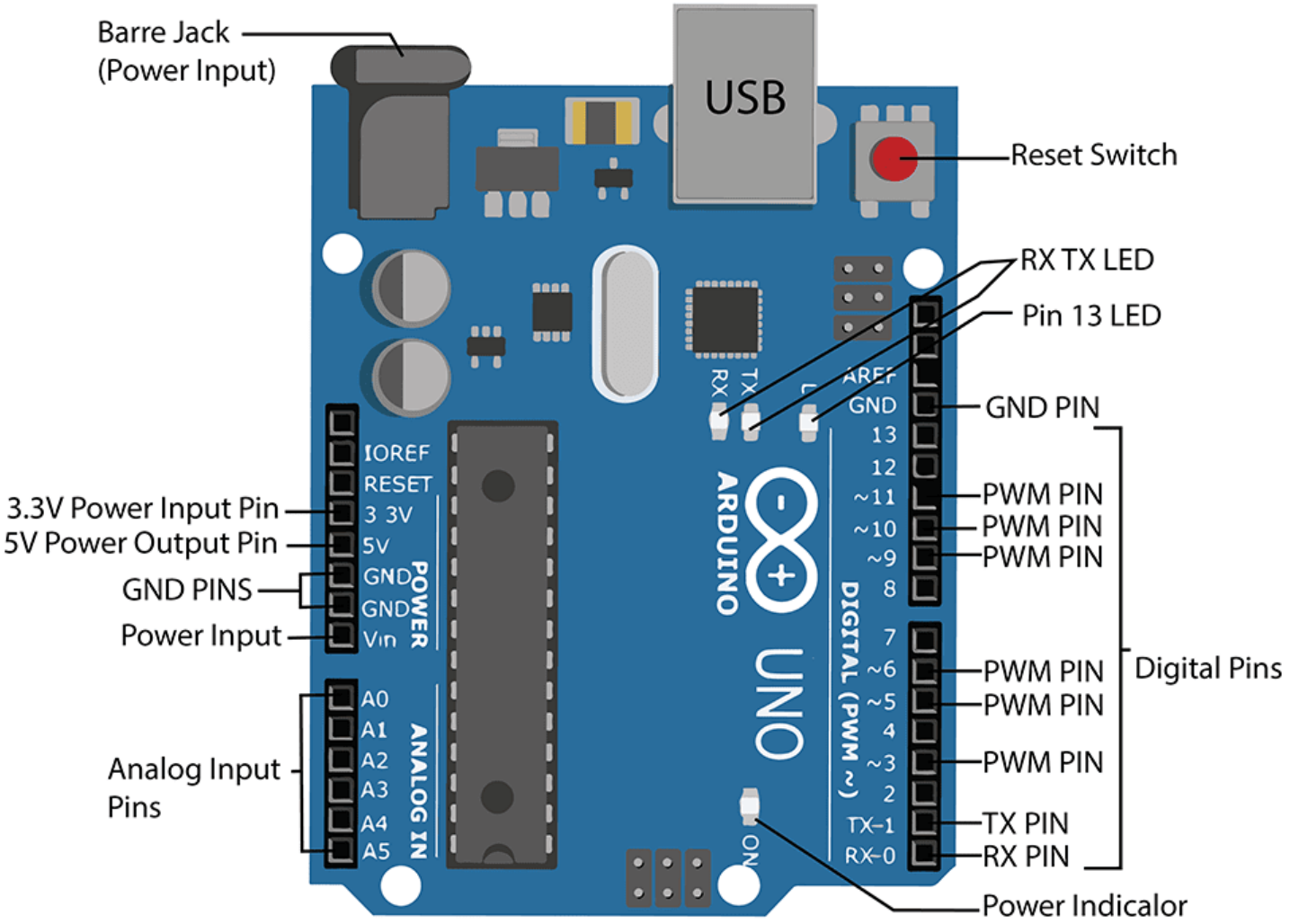
Learning Objectives for this lesson

- Description of the Arduino Uno Input / Output (I/O) pins.
- Interfacing the Arduino Uno to a LED, push-button and potentiometer.
- Description of the concept of Pulse Width Modulation (PWM).
- Using PWM concepts to control brightness in LED downlights – Leading and Trailing edge brightness control.
- Using the Arduino Digital I/O ports to create PWM.
- Writing, uploading and executing simple Arduino Uno Sketches for interfacing to a Led and potentiometer to demonstrate PWM.
- More practice on writing advanced Sketches.

Arduino Uno - Digital I/O pins

- The Uno has 14 Digital Input / Output pins.
- Each Digital I/O pin has a **maximum** drive current of 40 milliamps. However, it is recommended to supply only **20 milliamps** (max) as **continuous** current.
- The Digital Pins can be configured in a variety of ways:
  - Digital Input (all 14 pins)
  - Digital Output (all 14 pins)
  - PWM (Pulse Width Modulation) Output (only 6 pins)
  - Serial Tx / Rx (communications) (2 dedicated pins)
- Note, some of the pins support multiple functions !

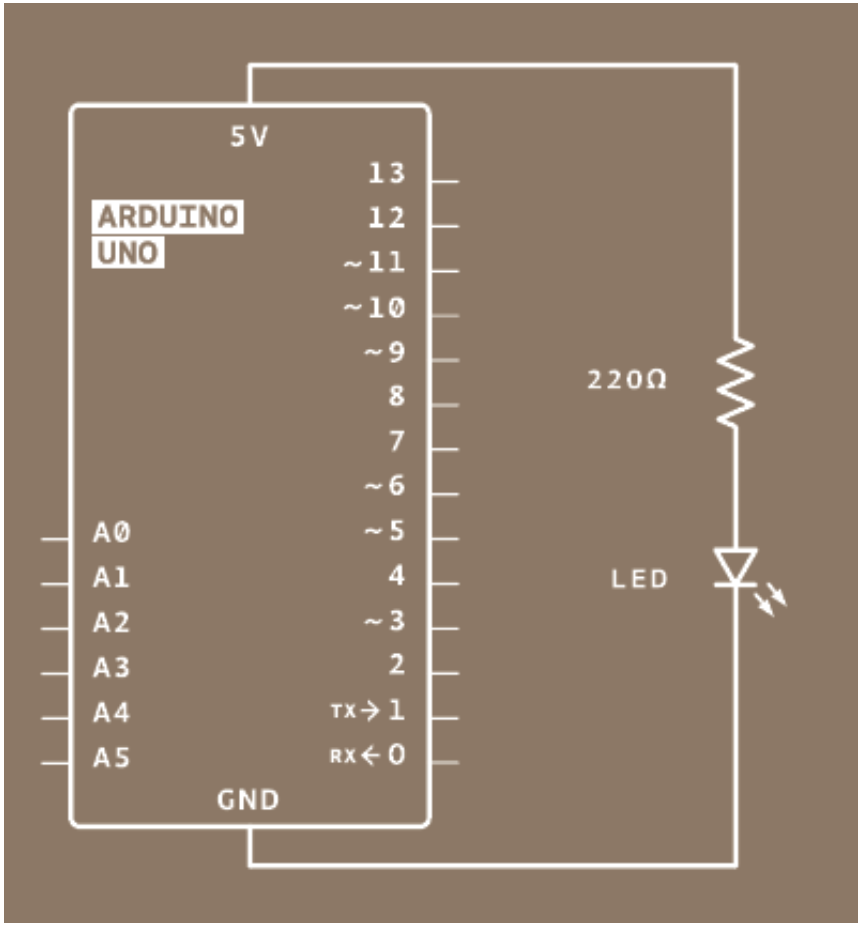
Arduino Uno


Barre Jack
(Power Input)
USB
Reset Switch
RX TX LED
Pin 13 LED
GND PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
TX PIN
RX PIN
Digital Pins
3.3V Power Input Pin
5V Power Output Pin
GND PINS
Power Input
Analog Input
Pins
Power Indicalor
IOREF
RESET
3 3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
DIGITAL (PWM ~)
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
TX
RX
L
ON

Digital Output LED interfacing

- The Digital Output pins can be configured for driving an external LED.
- We use a current limiting resistor to ensure the LED is supplied with sufficient current to switch the LED ON but not too much current to cause damage.
- A current of 14 milliamps is reasonable and this can be achieved using a limiting current resistor of value 220 ohms.

LED pinout and typical circuit



Current flowing (I) = V / R

= 5 – 2(Vf) / 220 = 13.6 milliamps

(Remember: The LED drops 2 volts approx. when switched ON)

Controlling LED from a Digital Output pin

- To control the LED from a specific Digital Output pin we need to write a Sketch to perform the following:
- Define which pin we want to use and specify the pin as an output. We use the function ***pinMode()*** to do this.
- Write the data (High) or (Low) to the port to turn ON or OFF the LED using the function ***digitalWrite()***.
- We can also include the function ***delay()*** if we want the program to loop and blink the LED at a specific flash rate.

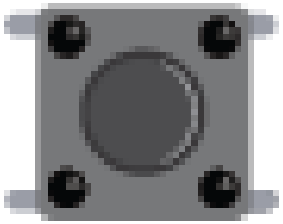
Code example for flashing an External LED connected to pin 13 at 1 second rate…

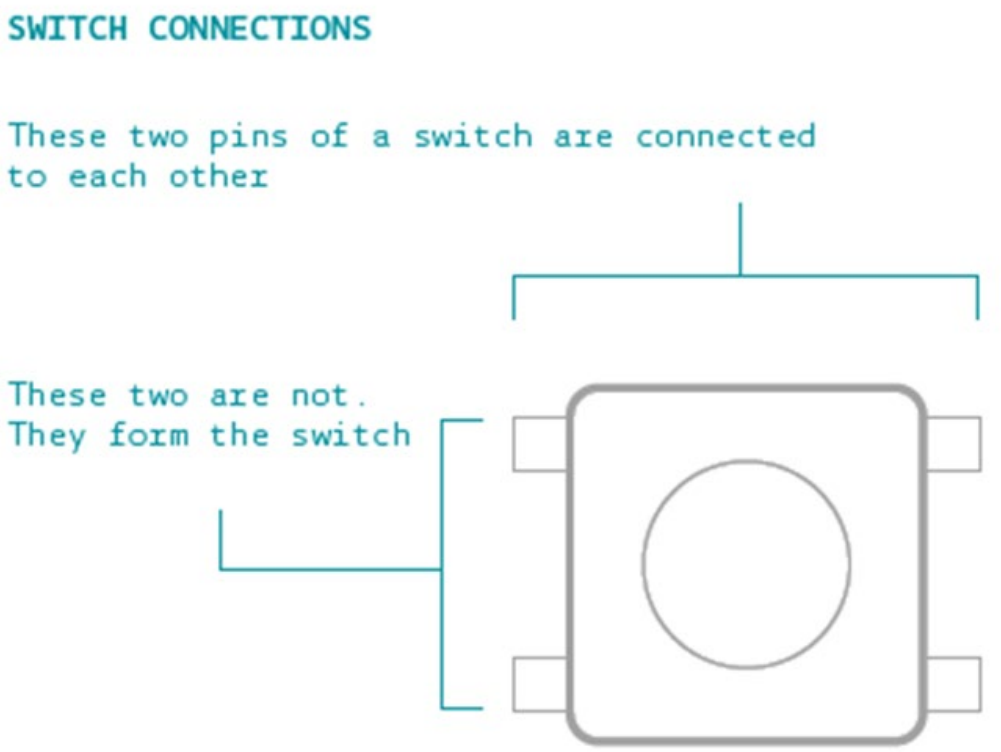
Example Code

```
The code makes the digital pin 13 `OUTPUT` and Toggles it `HIGH` and
`LOW`

    void setup() {
      pinMode(13, OUTPUT);    // sets the digital pin 13 as output
    }

    void loop() {
      digitalWrite(13, HIGH); // sets the digital pin 13 on
      delay(1000);            // waits for a second
      digitalWrite(13, LOW);  // sets the digital pin 13 off
      delay(1000);            // waits for a second
    }
```

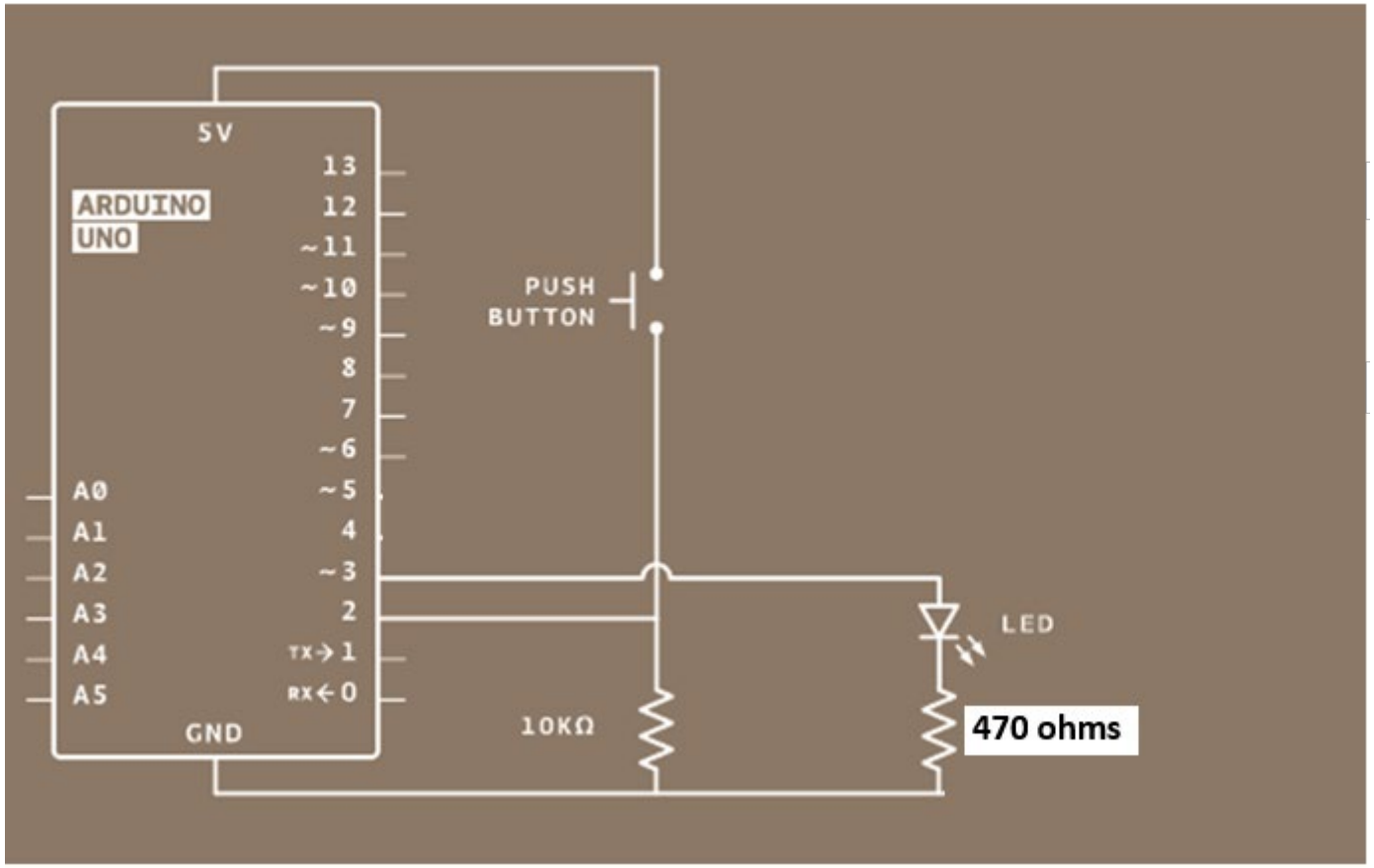
Digital Input Button interfacing

- The Digital I/O pins can also be configured as inputs for interfacing to a push-button.



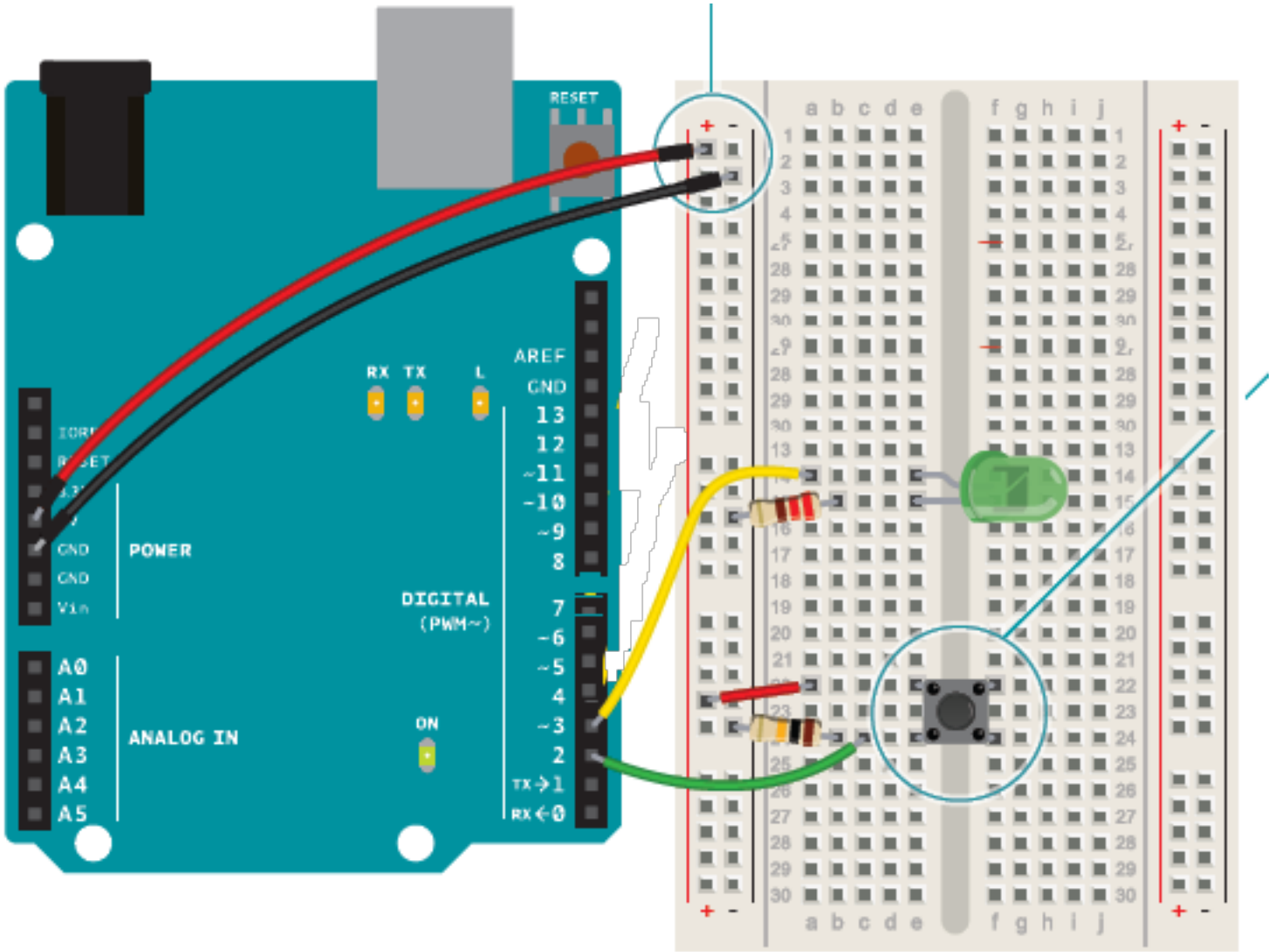


- To interface a button, we need to install a “pull- down” resistor.
- The pull-down resistor will ensure the digital input does not ‘float’ to an unknown voltage. A suitable value for pull-down resistor is 10 kohms.



- In this circuit, if PUSH BUTTON is **NOT** pressed the digital input pin 2 is pulled-down to a logic LOW.
- If the PUSH BUTTON **is** pressed the digital input pin 2 will now be logic HIGH.

Built Circuit



Controlling LED from using a push-button input

- To control the LED by using the push button we need to write a Sketch to perform the following:

- Define which pin we want to use for the push button input and specify the pin as an **input**. We use the function ***pinMode()*** to do this.

- Read the data (High) or (Low) from the push button input pin using the function ***digitalRead()***.

- Note, there is only one parameter in the ***digitalRead()*** function that is, the pin number.

- So, we must first define a variable to store the value of data we have read from the digital input pin when we used the function ***digitalRead()***.

Code example of Reading a push button digital input

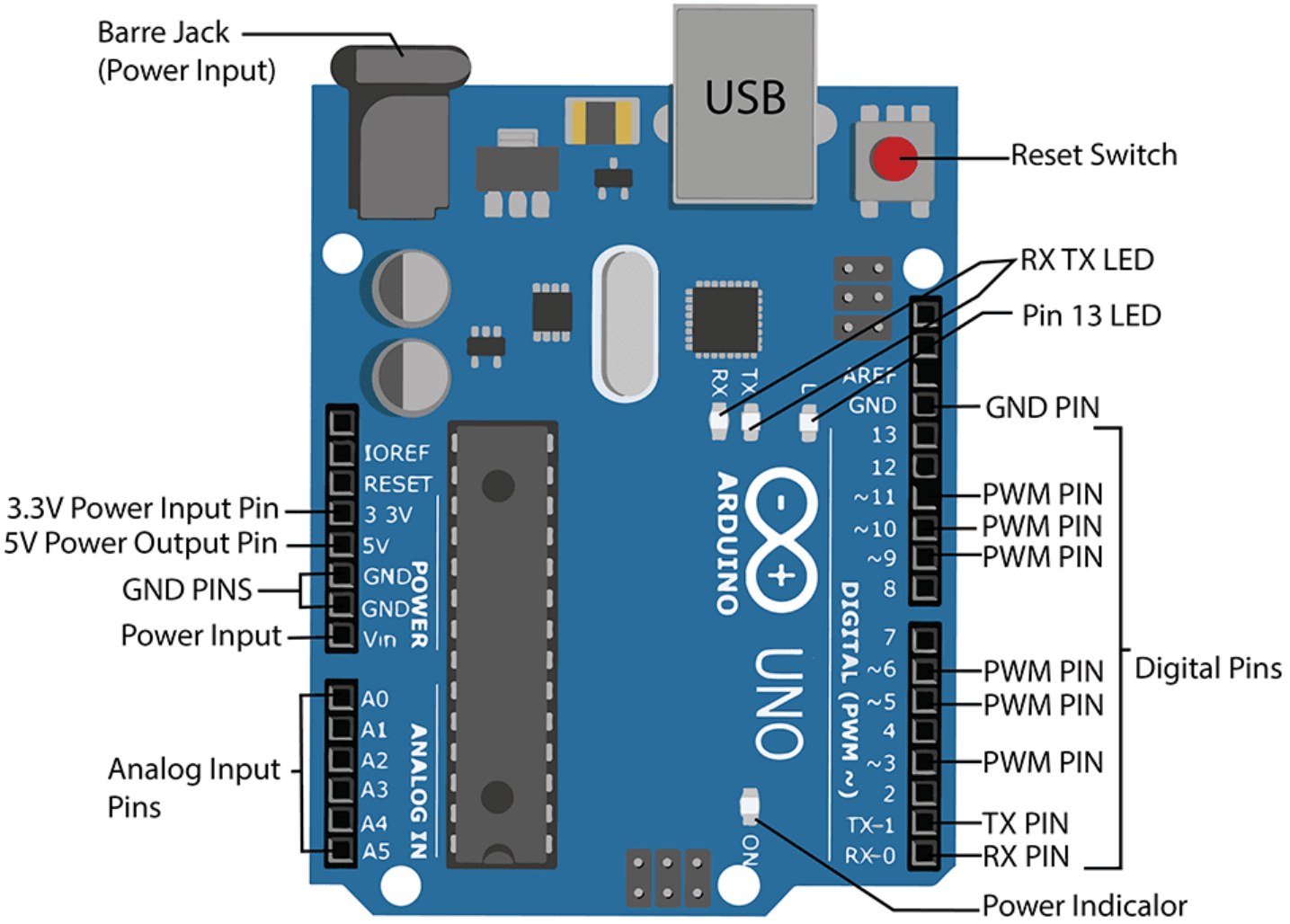
- Note, how three integer variables (**ledPin, inPin** and **val**) have been defined before the **setup()** function.

```
int ledPin = 13;  // LED connected to digital pin 13
    int inPin = 7;    // pushbutton connected to digital pin 7
    int val = 0;      // variable to store the read value

    void setup() {
      pinMode(ledPin, OUTPUT);  // sets the digital pin 13 as output
      pinMode(inPin, INPUT);    // sets the digital pin 7 as input
    }

    void loop() {
      val = digitalRead(inPin);   // read the input pin
      digitalWrite(ledPin, val);  // sets the LED to the button's value
    }
```

Arduino Uno - Analogue Input pins

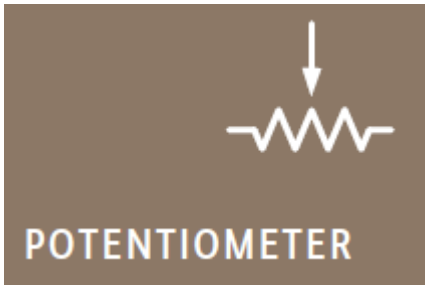
- The Uno has 6 analogue input pins – labelled A0 to A5.
- The analogue input can accept a range of input voltages from 0 to 5 volts.
- The input resolution is 10-bits.
- The code generated by the analogue input ranges from 0 to 1023. This means the following:
  - An analogue input voltage of **0** volts will generate a code of **0**.
  - An analogue input voltage of **2.5** volts will generate a code of **511**.
  - An analogue input voltage of **5** volts will generate a code of **1023.**

Arduino Uno

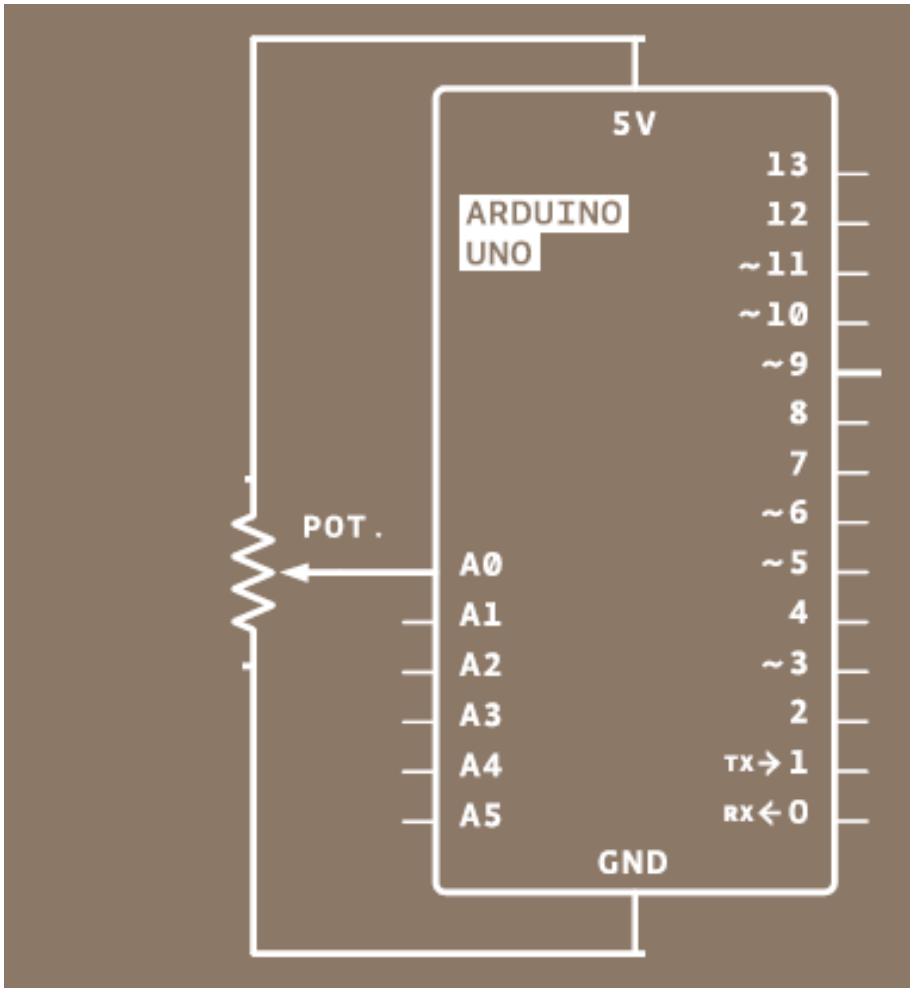

Barre Jack
(Power Input)
USB
Reset Switch
RX TX LED
Pin 13 LED
GND PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
PWM PIN
TX PIN
RX PIN
Digital Pins
3.3V Power Input Pin
5V Power Output Pin
GND PINS
Power Input
Analog Input
Pins
Power Indicalor
IOREF
RESET
3 3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
DIGITAL (PWM ~)
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
TX
RX
L
ON

Creating an Analogue input voltage

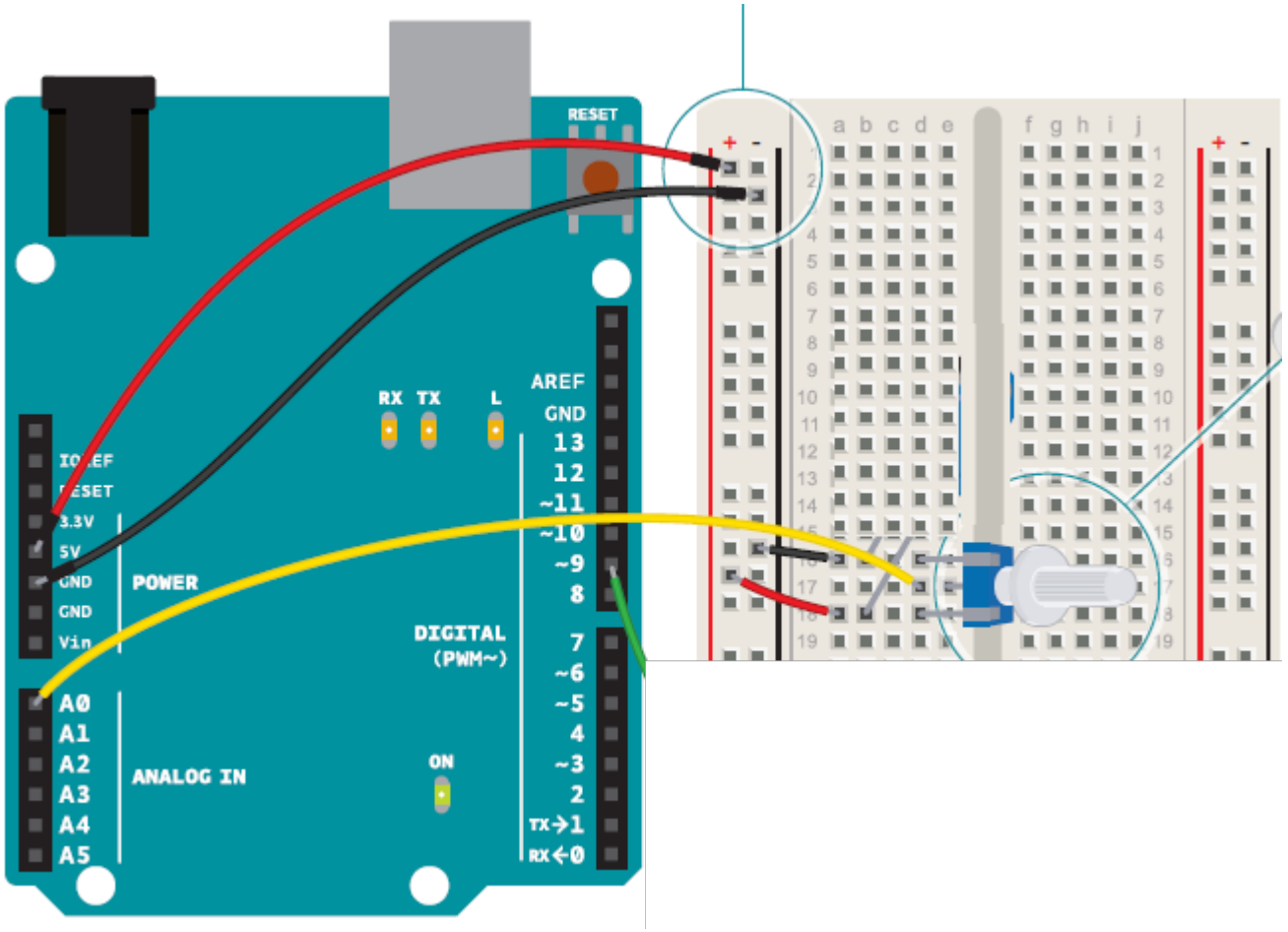
- We can use a **potentiometer** (**Pot**) as a voltage divider circuit to create an analogue voltage at input A0.
- A potentiometer is a variable resistor with three pins. Two of the pins are connected to the ends of a fixed resistor. The middle pin, or wiper, moves across the resistor, dividing it into two halves.
- When the external sides of the potentiometer are connected to voltage and ground, the middle leg will give the difference in voltage as you turn the knob. Often referred to as a pot.



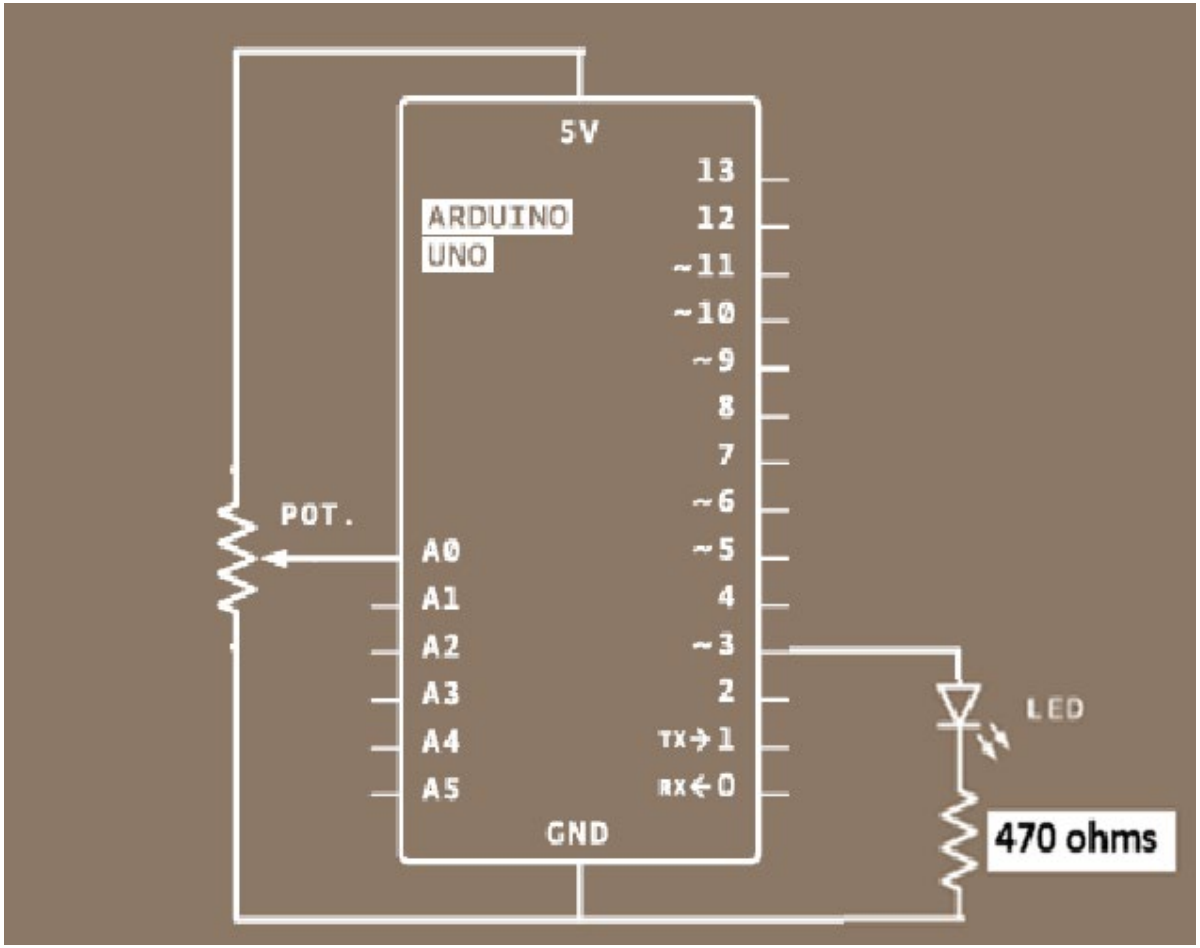
Circuit showing how a Pot is used



Build of circuit showing how to connect a Potentiometer



Circuit diagram to control a LED using the potentiometer



Controlling LED from Pot connected to A0

- To control the LED by using the potentiometer connected to analogue input A0, we need to write a Sketch to perform the following:

- We must first connect the middle pin of the potentiometer to analogue input pin A0 on the Arduino. The two outside pins of the potentiometer are connected to **GND** and **5V**.

- To read the analogue voltage at pin A0 we use the function ***analogRead(A0)***.

- Note, there is only one parameter in the ***analogRead(A0)*** function that is, the pin number – which we have decided as A0.

- So, we must first define a variable to store the value of data we have read from the analogue input pin when we used the function ***analogRead(A0)***.

Code example of using a Pot to control the flashing rate of an external LED

```
/*
  This is a simple Sketch to read a potentiometer on
  the Analog Input Pin A0.
  The value of the Pot is used to set the
  flash rate for an external LED
  connected to pin 3

*/

 int PotValue;    // variable to hold potentiometer value
 const int LED1 = 3;   // assign digital I/O pin 3 to LED

void setup() {
 pinMode(LED1, OUTPUT);
}

void loop() {
  PotValue = analogRead(A0);    //read the input from A0 and store it in a variable
    delay(PotValue);
    digitalWrite(LED1, HIGH);    // Turn LED1 ON
    delay(PotValue);
    digitalWrite(LED1, LOW);   // Turn LED1 OFF
 }
```

Pulse Width Modulation (PWM)

- **P**ulse **W**idth **M**odulation (PWM) is a digital technique to control a signal by repeatedly toggling a signal between a HIGH and a LOW state in a consistent pattern.
- PWM can be used to control average voltage of a signal by varying the width of a pulse, or the time it is "on," within a fixed period. It effectively creates an analog-like output from a digital signal.

Digital to Analog Conversion:

PWM allows digital control circuits to effectively simulate analog voltage levels by varying the duration of pulses.

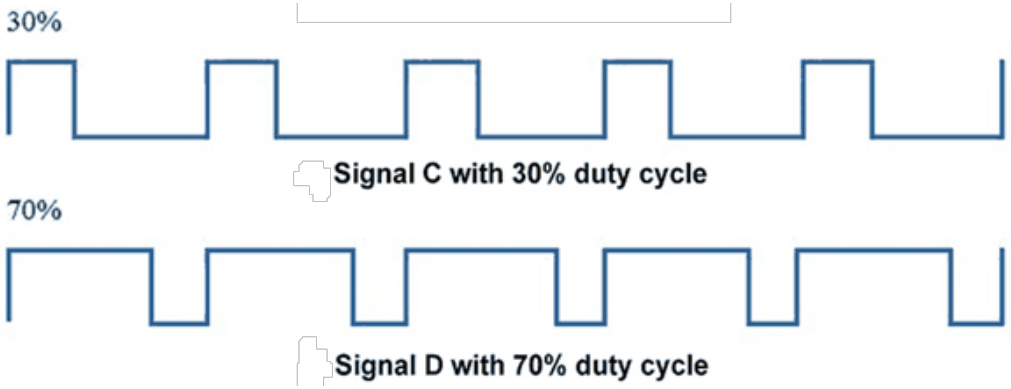
Advantages:

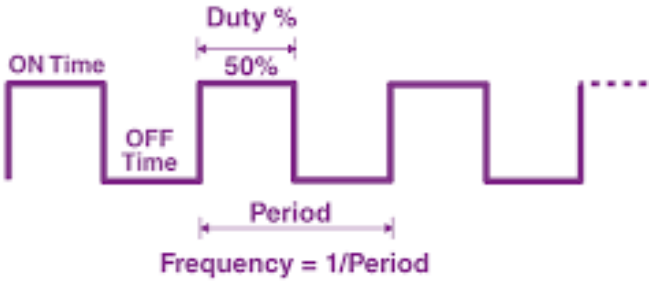
PWM is energy-efficient, precise, and can be used to control power delivery to devices.

Pulse Width Modulation (PWM)

Duty Cycle:

The "on" time of the pulse is called the pulse width, and the ratio of the pulse width to the total period is known as the duty cycle.





Frequency:

PWM signals also have a frequency, which is how often the pulses are switched on and off.

Applications:

PWM is used in many applications, including controlling motor speeds and adjusting the brightness of LEDs.

PWM and Domestic LED downlights

- The dimmers used to control the brightness of domestic LEDs in the home use a type of PWM.
- Dimmers are categorised as either Leading Edge or Trailing Edge in design.
- Leading Edge dimmers are used for halogen and incandescent globes. Trailing edge is the preferred type for LEDs.
- The dimmers use electronics and Triacs to control the amount of time the AC cycle is provided to the LED.

Leading edge and Trailing edge dimming

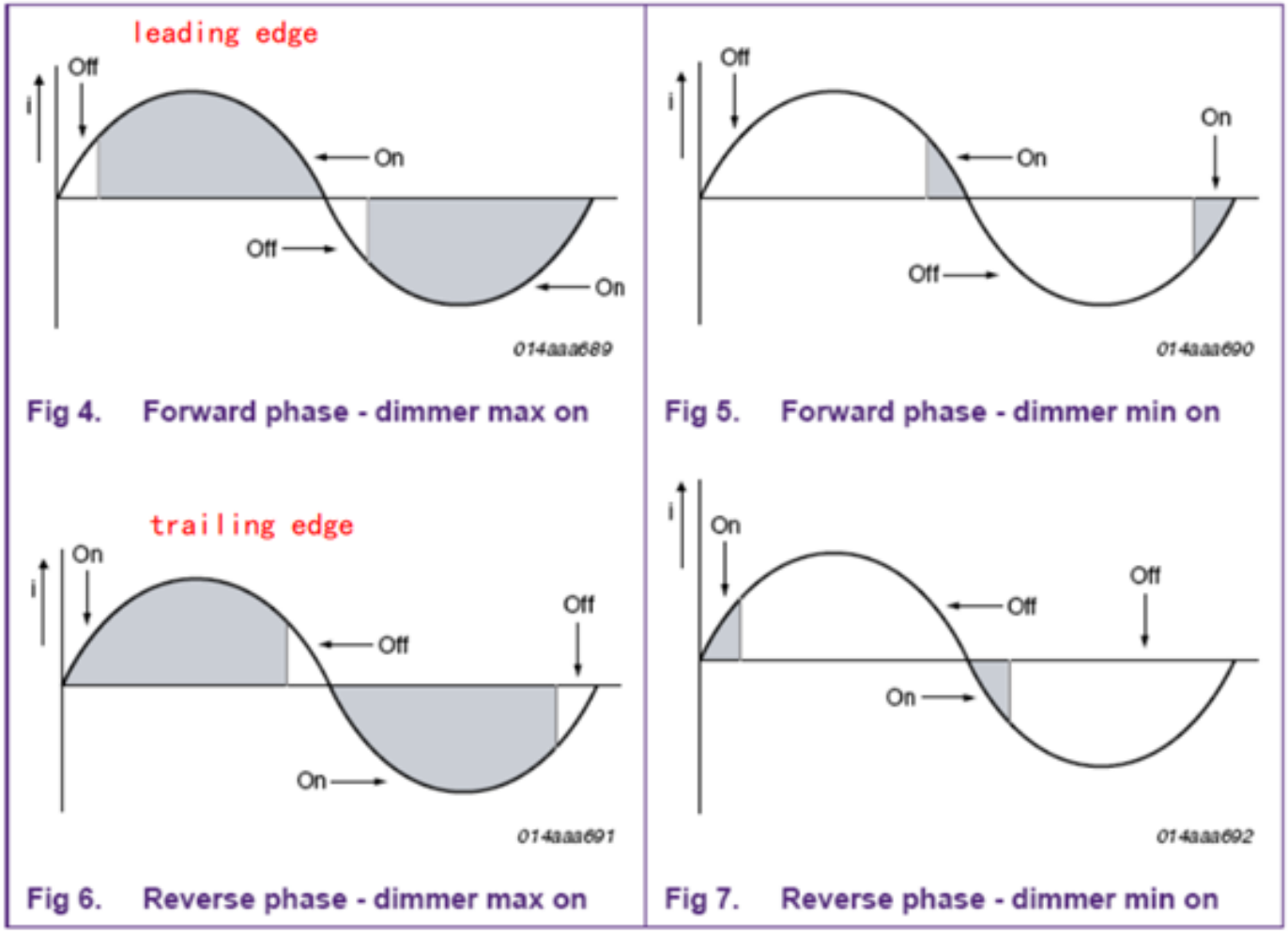


Fig 4. Forward phase - dimmer max on

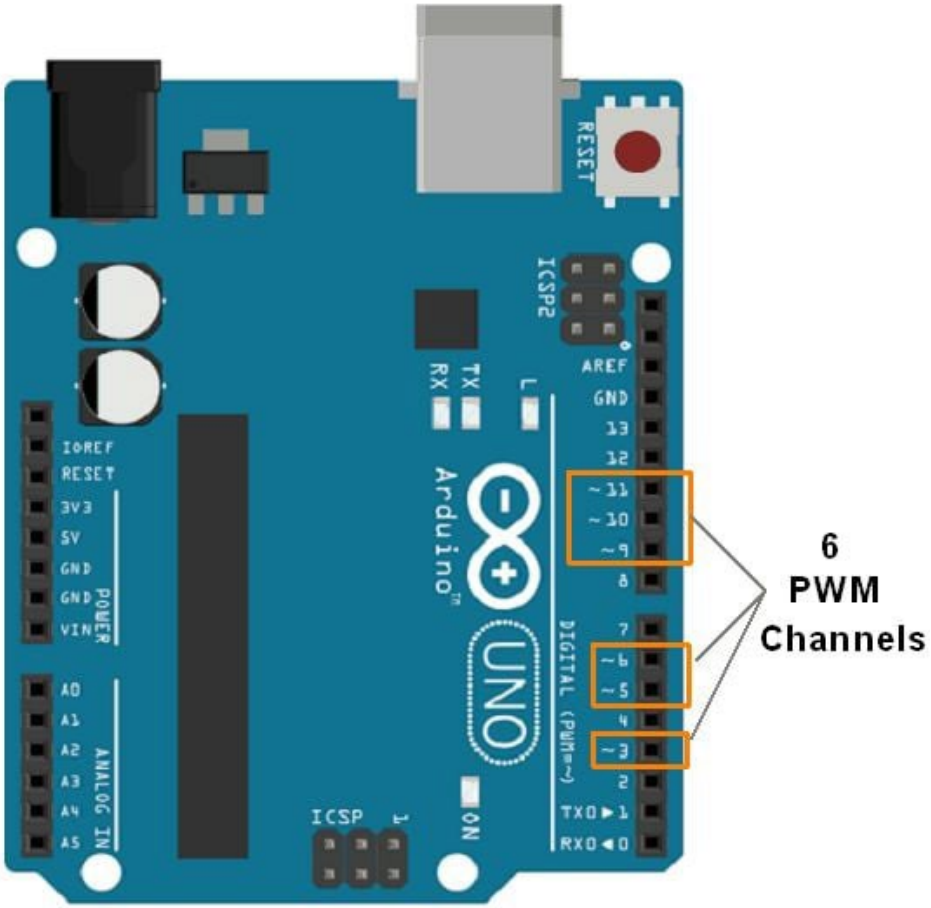
Fig 5. Forward phase - dimmer min on

Fig 6. Reverse phase - dimmer max on

Fig 7. Reverse phase - dimmer min on

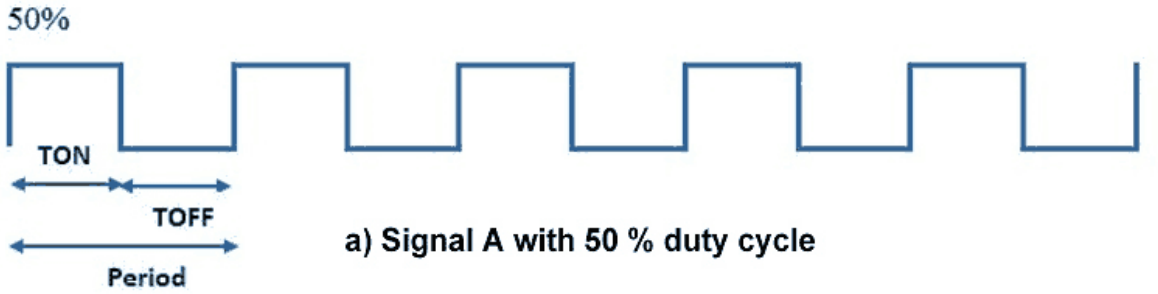
PWM and the Arduino

- The Arduino Uno has 6 8-bit PWM channels assigned to Digital I/O pins 3, 5, 6, 9, 10, and 11. Note, these are the only pins that can be used for PWM.



PWM and the Arduino

- The PWM signal uses a square wave with different duty cycles as shown below:



a) Signal A with 50 % duty cycle

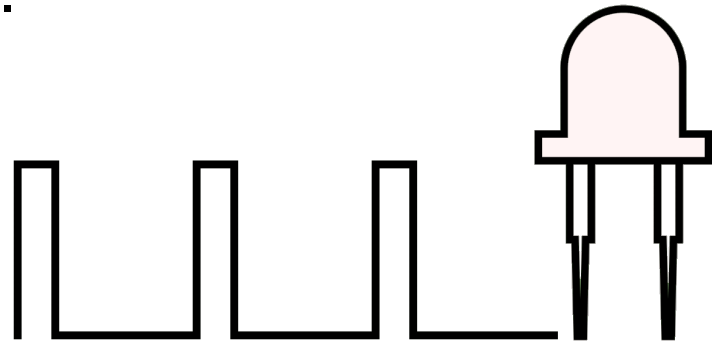
b) Signal B with 10% duty cycle

c) Signal C with 30% duty cycle

d) Signal D with 70% duty cycle

PWM in the Arduino

- It is important to have a high enough frequency (small period!) so that the LED does not appear to blink when the duty cycle changes.
- For example, if the period was 1 second, then a 50% duty cycle would cause the LED to simply ‘blink’.
- With Arduino the frequency is 490 Hz (period = 2 millisecs) so with 50% duty cycle the LED will appear as half bright (and not blink).



Controlling external LED brightness with PWM

- To control an external LED brightness using PWM we need to write a Sketch to perform the following:

- Define which Digital I/O pin we want to use for a PWM output to the LED. Note, you can **only** use pins **3, 5, 6, 9, 10, and 11.** Specify the pin you want to use as an **output**. We use the function ***pinMode()*** to do this.

- Write the data to represent brightness as a value between 0 and 255 using the function ***analogWrite()***. The value 0 would mean OFF and value 255 would mean fully ON.

- Let us look at how to use ***analogWrite()***….

analogWrite()

Last revision • 26/09/2024

Description

Writes an analog value (**PWM wave**) to a pin. Can be used to light a LED at varying brightnesses or drive a motor at various speeds. After a call to `analogWrite()`, the pin will generate a steady rectangular wave of the specified duty cycle until the next call to `analogWrite()` (or a call to `digitalRead()` or `digitalWrite()`) on the same pin.

Syntax

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

Parameters

- `pin`: the Arduino pin to write to. Allowed data types: `int`
- `value`: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: `int`

Example Code…

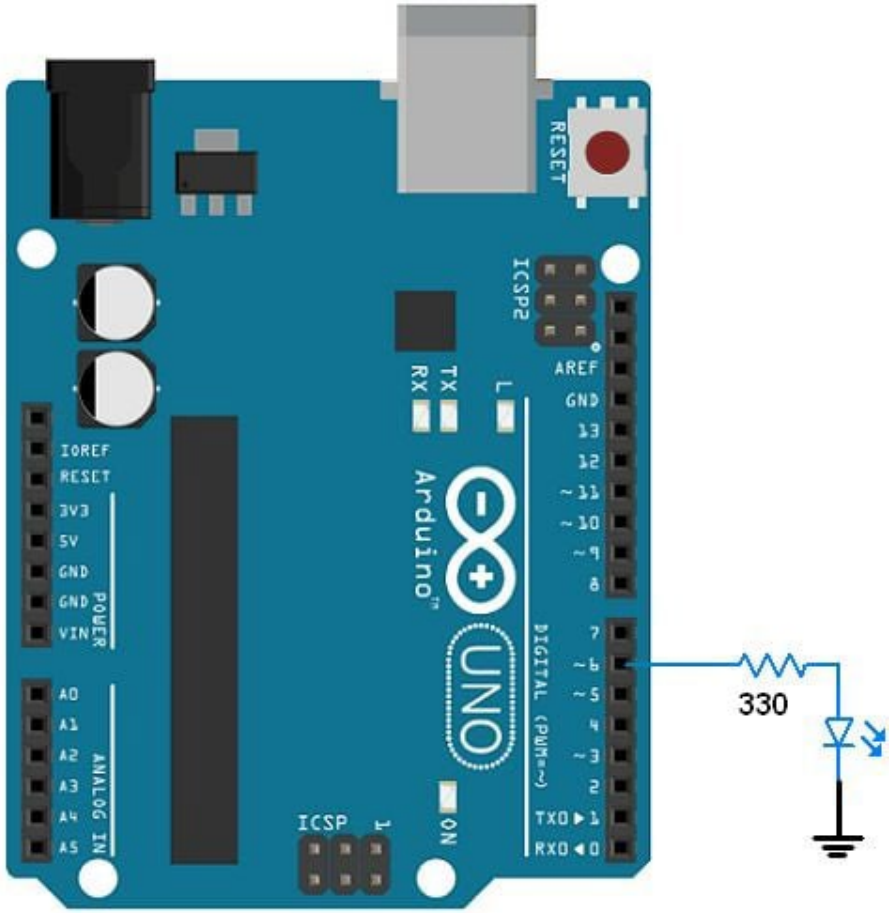
Sets the output to the LED proportional to the value read from the potentiometer (connected to A3).

```
int ledPin = 9;      // LED connected to digital pin 9
    int analogPin = A3;   // potentiometer connected to analog pin 3 (A3)
    int val = 0;          // variable to store the read value

    void setup() {
      pinMode(ledPin, OUTPUT);  // sets the pin as output
    }

    void loop() {
      val = analogRead(analogPin);  // read the input pin
      analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWri
    }
```

Student Activity 1 - Design Specification

- Build a circuit and write the C Sketch to perform the following:
  - Drive an external LED from the digital I/O pin 6.
  - The digital pin 6 is to be used in PWM.
  - The program is to slowly increase the brightness of the LED from 0 to 255 and repeat the process indefinitely.
  - Use the function ***delay(30)*** so that you can see the effect of the brightness.
  - Use an if-else statement and a counter to step up through the values of 0 to 255.

Circuit to build



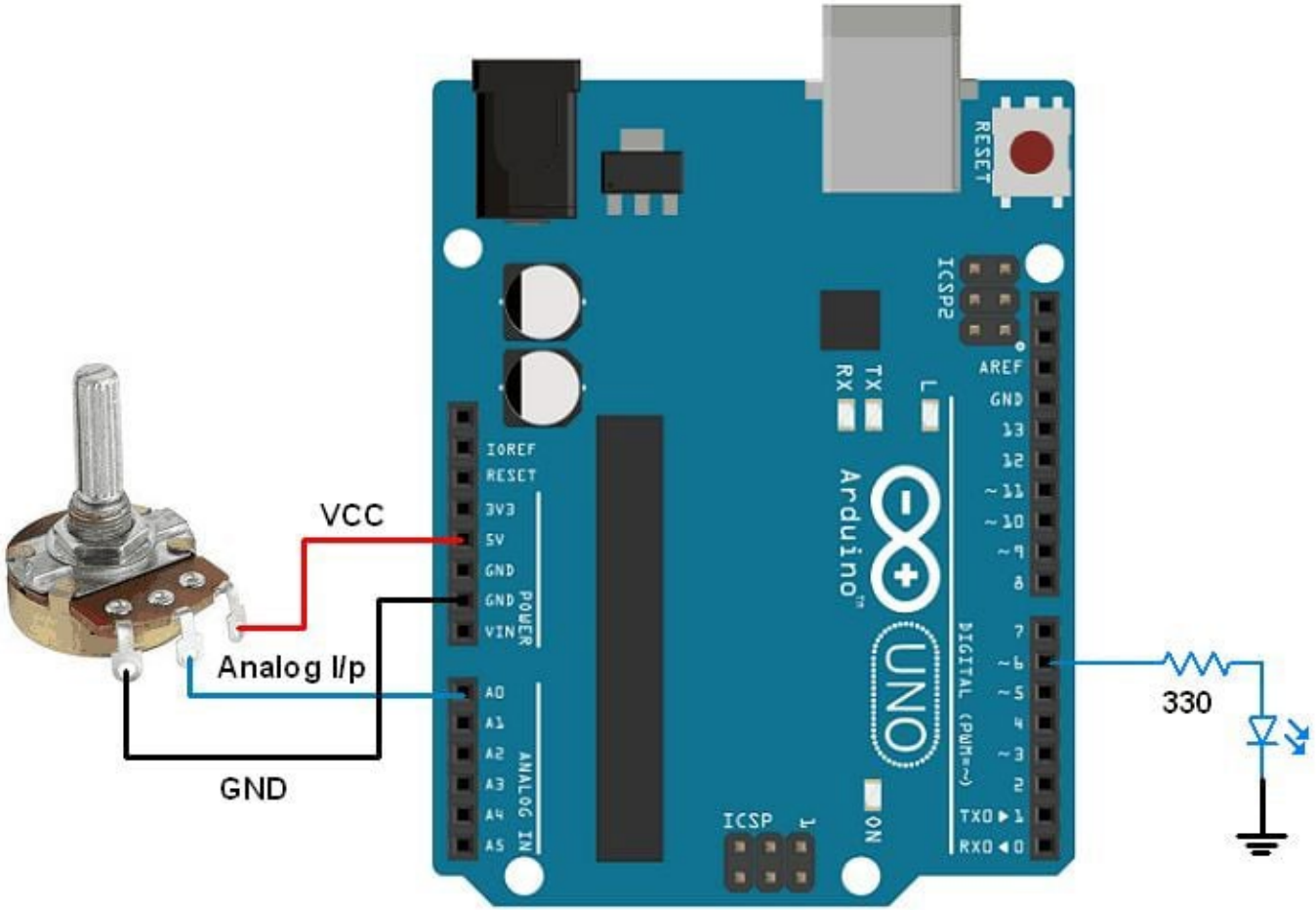
Student Activity 2 - Design Specification

- Using your same circuit in Activity 1, write the C Sketch to perform the following:
  - Drive an external LED from the digital I/O pin 6.
  - The digital pin 6 is to be used in PWM.
  - The program is to slowly **fade in and out** the brightness of the LED from 0 to 255. That is, LED increase up to maximum then slowly decreases back to zero and then repeat the process indefinitely.
  - Use the function ***delay(30)*** so that you can see the effect of the brightness.
  - Use an if-else statement and a counter to step up (and down) through the values of 0 to 255.

Student Activity 3 - Design Specification

- Build a circuit and write the C Sketch to perform the following:
  - Drive an external LED from the digital I/O pin 6.
  - The digital pin 6 is to be used in PWM.
  - Connect a Potentiometer to analogue input pin A0.
  - The program is to allow control of the brightness of the LED by using the Potentiometer – from the LED completely OFF to fully turned ON.
  - The program will also output the values of the potentiometer to the Serial Monitor port so you can observe what the optimal range of pot values is best for controlling the brightness.

Circuit to build



Student Activity 3v2 - Design Specification

- Using your same circuit in Activity 3, modify the C Sketch to include the ***map()*** function.

- Choose the values for map that provide the best optimal control of the brightness of the LED from your Potentiometer.

Arduino Photo Resistor

- The Arduino Uno kit also includes a photoresistor. This component changes resistance value depending upon the intensity of the light that falls on its face.

- In full bright sun light, the resistance is less than **80 ohms**. In darkness the resistance increases to more than **70 kohms** !!

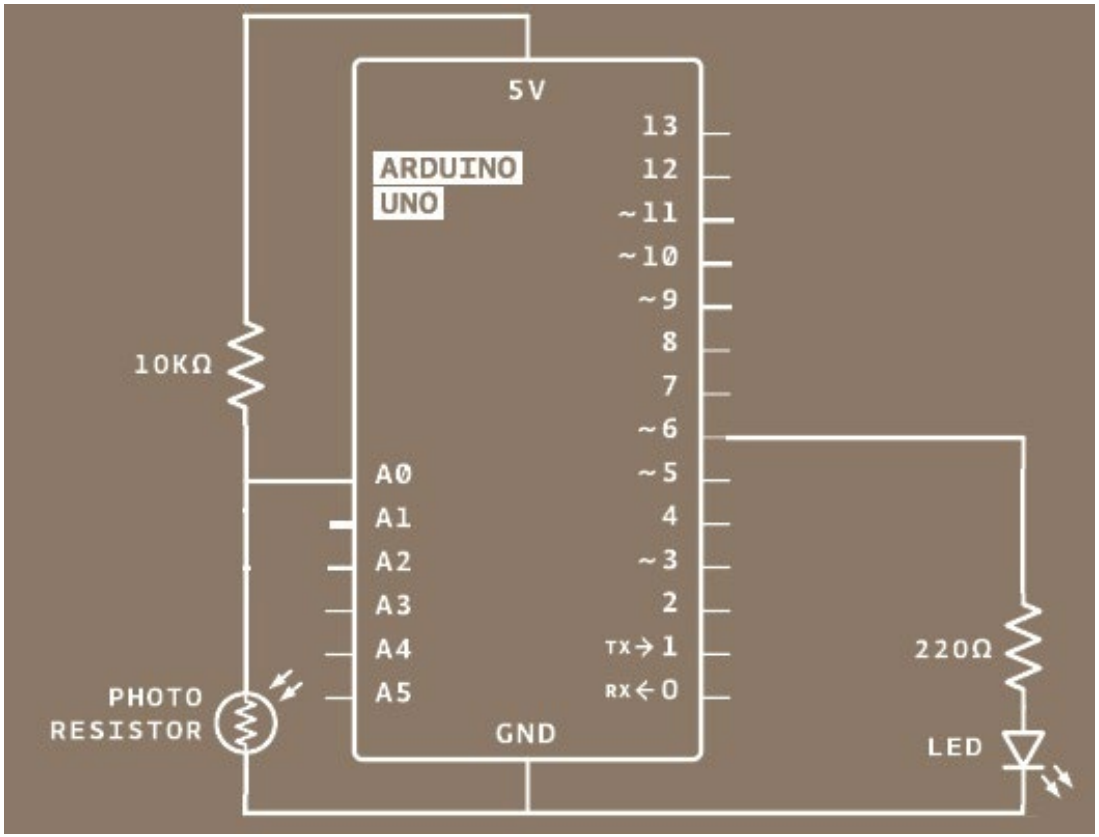
- We could replace the Pot connected to the analogue input A0 in Activity 3, with the photoresistor and make a circuit that controls the intensity of the LED dependent upon the background light level. This could be used as an automatic light switch to turn garden outdoor LEDs on when the sun sets.

- Let us do an Activity to test this idea…

Student Activity 4 - Circuit to Build

- Build the following circuit on your Arduino breadboard. You will need to remove the Pot from A0 and replace with the photoresistor as shown.

- Remember, you want the LED to be bright when the room is dark !!



Student Activity 4 – Code to write

- Write a C Sketch that turns the LED ON when the background room light is dark and turns the LED OFF when there is plenty of background light.
- You can use your Activity 3 Sketch for the code.
- Test your Sketch by covering the photoresistor sensor to see if the LED brightness changes.
- You may want to include a call to the ***map()*** function in your Sketch to obtain the best optimal performance of your automatic dimming circuit.

Extra Practice Activities….

- Writing and fault-finding Sketches is the best way to gain Skills in C programming in the Arduino IDE.

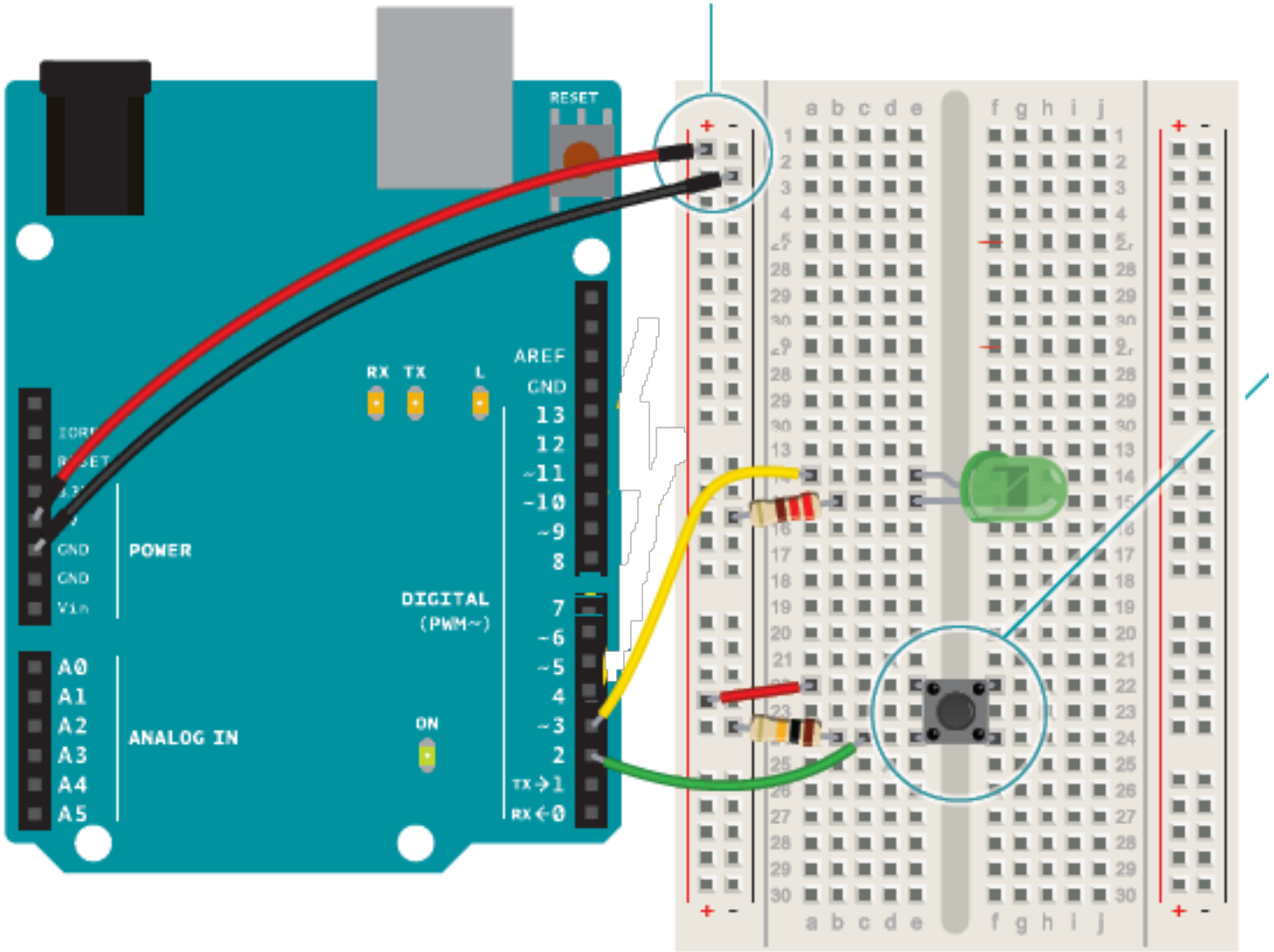
- Here are some practice activities from the previous lesson which you should now try to attempt..

Student Activity 5

- Build a circuit and write the C Sketch to perform the following:
  - Drive an external LED from the digital I/O pin 3.
  - The push-button input is to be connected to digital I/O pin 2.
  - If the push-button is pressed **momentarily** once the LED will be turned ON continually.
  - If the push-button is pressed again **momentarily** the LED will be turned OFF continually.
  - If the push-button is **held down** the LED flashes at a rate of 0.2 seconds until you release the button.
  - Use an **if-else statement** to decide if the push button has been pressed.

** IMPORTANT POINT ** You will need to decide how you would detect a **momentary** push of a button !!

Student Activity 5 – Built Circuit



Student Activity 5v2 – (Advanced)

- Build a circuit and write the C Sketch to perform the following:
  - Drive an external LED from the digital I/O pin 3.
  - The push-button input is to be connected to digital I/O pin 2.
  - If the push-button is pressed **momentarily** once the LED will be turned ON continually.
  - If the push-button is pressed again **momentarily** the LED will be turned OFF continually.
  - If the push-button is **held down** then nothing happens to the current status of the LED.

(that is, if the LED is ON then it stay ON and if it is OFF then it stays OFF).

  - Use **if-else statements** to decide if the push button has been pressed.

** IMPORTANT POINT ** You will need to decide how you would detect a **momentary** push of a button !!

Student Activity 6 - Design Specification

- Build a circuit and write the C Sketch to perform the following:
  - Drive three external LEDs from the digital I/O pins 10, 11 and 12.
  - Use analogue input A0 to interface to a potentiometer. This will produce a voltage from 0 v to 5 volts at the input pin A0.
  - ALL LEDs will be turned off if the analogue input is less than 1 volts
  - LED1 only will be turned ON if the analogue input is from 1 to 2 volts.
  - LED2 only will be turned ON if the analogue input is from 2 to 3 volts.
  - LED3 only will be turned ON if the analogue input is from 3 to 4 volts.
  - ALL LEDs will be turned ON if the analogue input is more than 4 volts.
  - Use several **if statements** to decide which LEDs to turn on and off.
  - Output the **voltage** to Serial port to 2 decimal points

  (Note, you may need to use a ***Float*** command to give you the right values)

Hint: 1 volt = 205 code, 2 volts = 409 code, 3 volts = 614 code, 4 volts = 818 code

Student Activity 6 – A hint when using *If* statements

- If statements can also use logical operators in their expressions. Look at this example using the logical AND operator:

```
if (PotValue > 30  &&  PotValue < 60)  {
        digitalWrite(LEDX, HIGH) ;
                                    }
```

- This code will only turn LEDX to ON if the value of PotValue is between the range 30 to 60.

- That is, PotValue must be larger than 30 and smaller than 60.