ENME 461 Lab 1- Introducing the Arduino Microcontroller

1. Introduction

This lab is an introduction to the Arduino microcontroller family and some basic functions of the microcontroller. The lab consists of three guided projects that will serve to familiarize users with basic process and two exercise projects that users will complete on their own.

To complete the lab, show the 3 guided projects and 2 exercise projects to your teaching assistant.

2. Objectives

- To develop a basic understanding of microcontroller fundamentals.
- To develop familiarity with breadboards and basic electric and electronic components.
- To create basic functional microcontroller programs.
- To control basic electronic components.
- To use serial communication to read sensor inputs to the microcontroller.

3. Guided Project One- LED Blink

This project, perhaps the most basic of all microcontroller projects, uses the Arduino microcontroller to blink an LED on and off.

3.1. Equipment

The equipment needed for this project are included in the Arduino budget packs at each station.

- UNO microcontroller
- Breadboard
- One Red LED
- 100 Ω resistor (See Figure 1 for the band colour codes.)
- Jumper wires
- USB cable connection

3.2. Program Concept

The digital pins on the Arduino board (pins 0 through 13) can be programmed to be inputs or outputs. This is accomplished in the Arduino sketch (program) using the pinMode() command.

These pins can also be programmed to be powered On or Off, respectively HIGH or LOW in the Arduino Sketch, using the digitalWrite() command. The On state corresponds to a voltage of +5V sent to the pin and the off state represents a voltage of zero.

With this information, we can set one digital pin, 2 for instance, to be an output and be powered on and off repeatedly. If we connect an LED to this pin, we can power the LED on and off, creating a repeating blinking pattern.

3.3. Light-Emitting Diode (LED)

The light emitting diode is a diode that converts electrical energy into light. Diodes allow current to flow only in one direction, therefore it is important to understand the polarity of an LED connection to a DC voltage source.

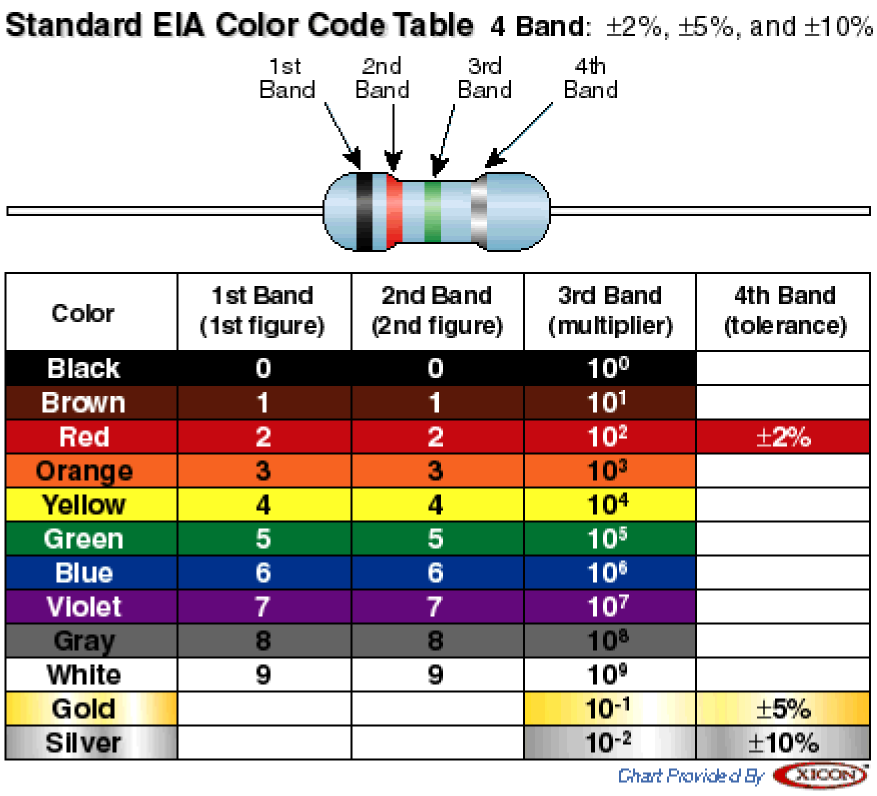
The ***positive*** side of the LED is called the ***Anode*** and has a ***long*** leg. The ***negative*** side of the LED is the ***Cathode*** and has a ***short*** leg.

To prevent damage to the LED, the current drawn by the LED must be limited. Therefore, we always use an LED in series with a resistor to reduce the current flow into the LED.

3.4. Project LED Blink Circuit

Complete the LED Blink circuit as shown in breadboard and schematic views in Figure 2.

- Note that Pin 2 is connected to the positive power rail of the breadboard using a red wire. The negative power rail on the breadboard is grounded using a black wire. This helps with circuit organization.
- Recall that, on the breadboard, power rails that run horizontally along the length of the board are electrically connected. Also recall that the terminal strips (vertical rows in the two mid-sections of the breadboard) are connected at each row (a through e are connected) and (f through j are connected), but separated by the crevice in the middle.
- The 100 Ω resistor is in series with the LED. See Figure 1 for the band colour codes.
- The positive (anode, long leg) of the LED is connected to the positive side of the circuit and the negative (cathode, short leg) of the LED is grounded.
- Note the schematic circuit representation of the LED.



Standard EIA Color Code Table 4 Band: ±2%, ±5%, and ±10%

1st Band
2nd Band
3rd Band
4th Band
Chart Provided By XICON

Color	1st Band (1st figure)	2nd Band (2nd figure)	3rd Band (multiplier)	4th Band (tolerance)
Black	0	0	10^0	
Brown	1	1	10^1	
Red	2	2	10^2	±2%
Orange	3	3	10^3	
Yellow	4	4	10^4	
Green	5	5	10^5	
Blue	6	6	10^6	
Violet	7	7	10^7	
Gray	8	8	10^8	
White	9	9	10^9	
Gold			10^{-1}	±5%
Silver			10^{-2}	±10%

Figure 1- Colour code chart for resistor bands. The value has the unit Ω

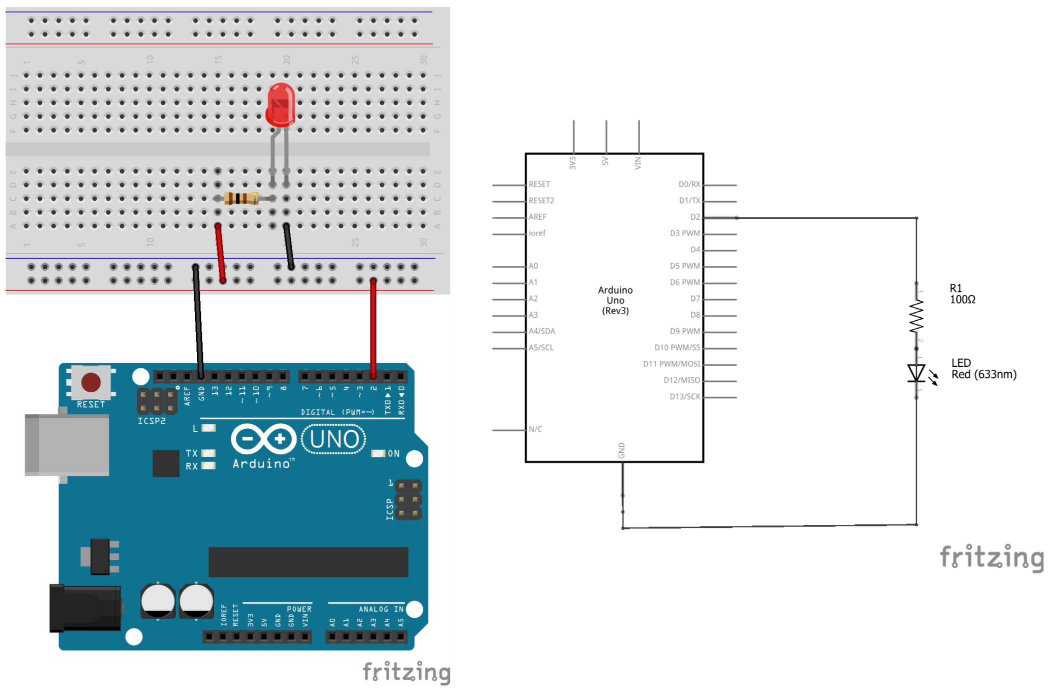


Figure 2- Breadboard and schematic views of the LED blink circuit

3.5. LED Blink Program

Open the Arduino IDE on the computer. Create a new sketch from File→New.

Recall that each Arduino program (sketch) consists of two main parts; setup() function and loop() function.

3.5.1. setup()

In the setup() function, we set pin modes, initialize variables and use libraries. This setup() function is run only once, as explained on the sketch.

Within the two curly brackets, we set the digital pin 2 to be a digital output. The syntax is as follows:

```
pinMode(2, OUTPUT); // initializes pin 2 as a digital output
```

- Letter case is important.
- Each command is terminated by a semicolon.
- The // indicates comments. These are not executed.

3.5.2. loop()

The loop() function loops through the commands within it indefinitely. In the loop() function, **within the two curly brackets**, we first turn pin 2 on which sends a 5V voltage to pin 2, turning the LED on. The syntax is as follows:

```
digitalWrite(2, HIGH); // sets the voltage level to HIGH or on
```

We then implement a delay() function. This pauses the program for the amount of time in milliseconds specified, keeping the LED on. The syntax is as follows:

```
delay(1000); // pauses program execution for 1000 milliseconds (1 sec)
```

We then turn the pin 2 off, setting its voltage to zero which turns the LED off. The syntax is as follows:

```
 digitalWrite(2, LOW); // sets the voltage level to LOW or off
```

We then implement another delay() function. This again pauses the program, keeping the LED off for another second.

```
delay(1000); // pauses program execution for 1000 milliseconds (1 sec)
```

This completes the sketch. We can now verify the sketch and upload it to the Arduino.

3.6. Save, Verify and Upload

Saving the sketch creates an *.ino file on the computer. Save your sketch and give it a representative name. Connect the board to the computer using the USB cable. Note that the ON LED on the board lights up as does a second LED marked L.

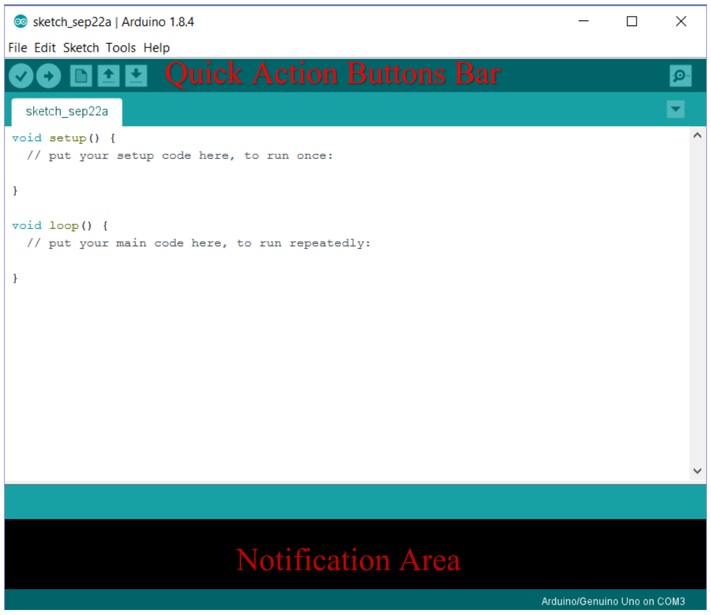
Verify checks the sketch for errors and compiles it, converting the human readable sketch into a machine language *.hex file. To verify a sketch, use the Verify button on the Quick Action Buttons Bar as shown below in Figure 3. Once verified, messages will appear in the Notification Area.

To upload a sketch to the Arduino for the very first time, we select the board and the serial port through which communication is done with the Arduino. Board selection is shown in Figure 3. The boards in the lab are the UNO type.

To select the port, follow Figure 5. Select the serial port next to which Arduino UNO appears (may be a different com number than pictures).

Finally, we upload the sketch to the Arduino using the upload button on the Quick Action Buttons Bar from Figure 3. During upload, a status bar indicates the upload progress and the TX and RX LED's on the board blink intermittently.

The program should now run indefinitely.


sketch_sep22a | Arduino 1.8.4
File Edit Sketch Tools Help
Quick Action Buttons Bar
sketch_sep22a
Notification Area
Arduino/Genuino Uno on COM3

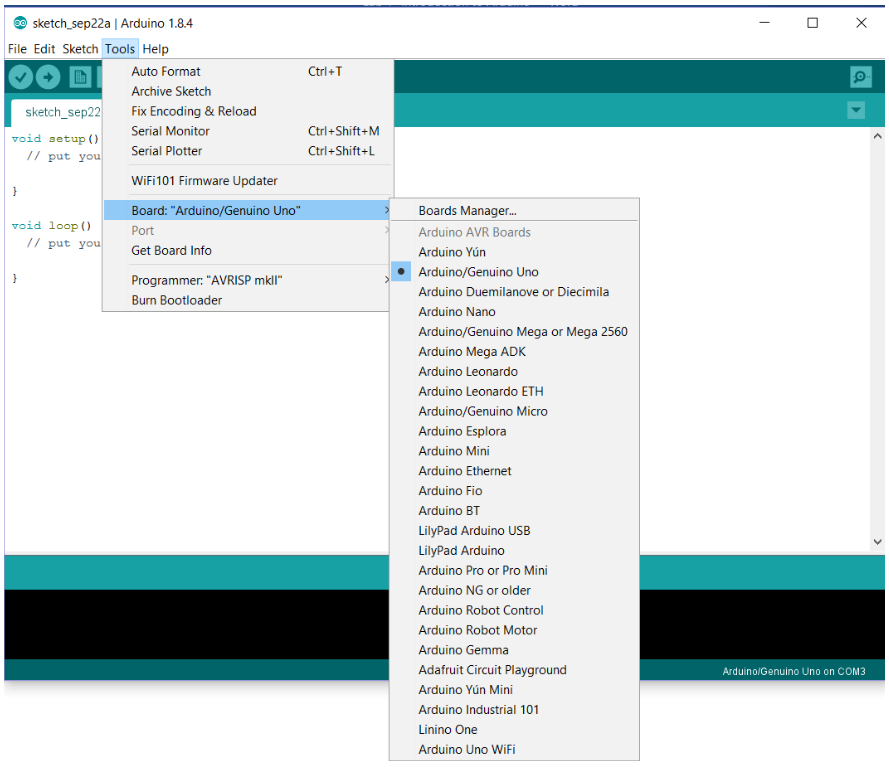
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 3- Location of Verify and Upload on the Quick Buttons Bar



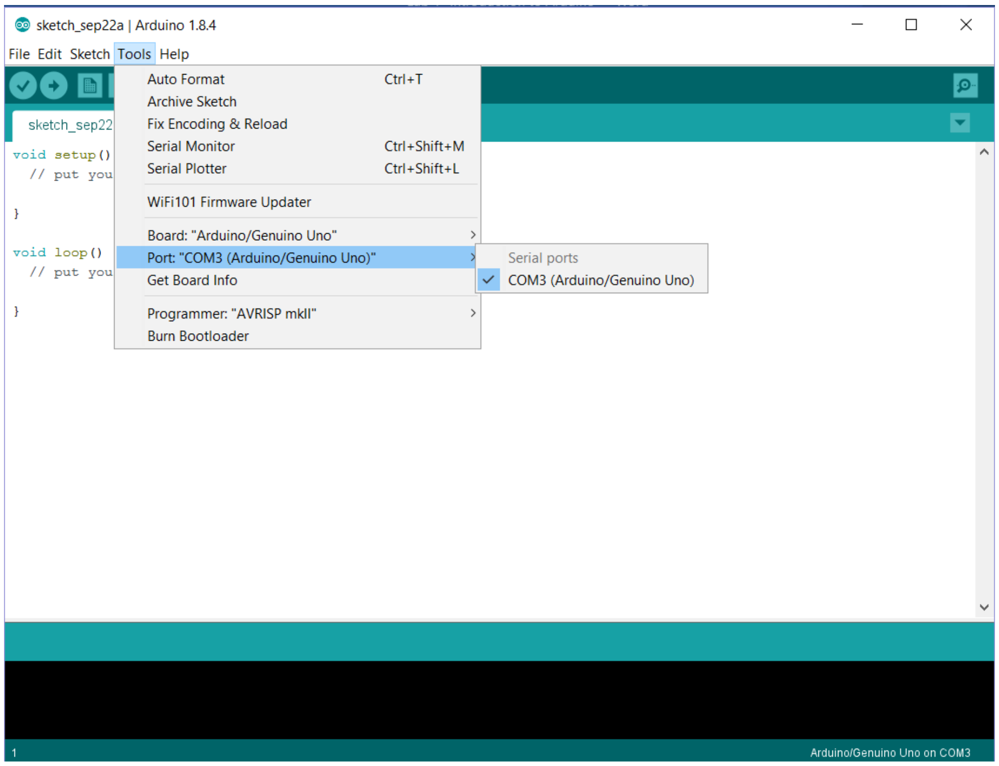
```
void setup()
  // put you

}

void loop()
  // put you

}
```

Figure 4- Selecting the Arduino UNO Board


sketch_sep22a | Arduino 1.8.4
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM3 (Arduino/Genuino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
✓ COM3 (Arduino/Genuino Uno)
sketch_sep22
1
Arduino/Genuino Uno on COM3

```
void setup()
  // put you

}

void loop()
  // put you

}
```

Figure 5- Selecting the Serial port (choose the port where Arduino UNO appears)

4. Guided Project Two- LED and Push-Button Control

This project will allow the Arduino to detect if a push-button is pressed. The press of a push-button is confirmed by turning an LED on.

4.1. Equipment

We use the same equipment as before with the addition of a push-button from the packs and a 1kΩ resistor (Google).

4.2. Pushbutton

The operation of a push-button is simple. As shown in Figure 6, legs A and D are electrically connected as are legs B and C. When the button is pressed, all legs are essentially connected.

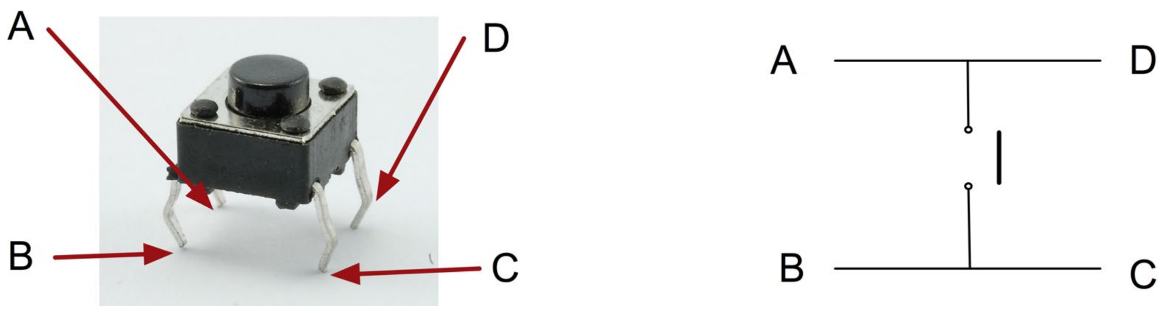


Figure 6- Push-button operation

4.3. Project Concept

We now use one digital pin (pin 8 for instance) as an input and have the Arduino read whether it is On (HIGH) or Off (LOW). We connect this pin to the 5V source using the push button so that the press of a button will turn it on or off. We also connect an LED to pin 2 as before. If pin 8 is On (push button pressed), we turn pin 2 and the LED on.

4.4. LED Push-Button Circuit

Before attempting to create a new circuit, we disconnect the USB cable from the computer.

The circuit and schematic for the LED push button are shown in Figure 7 below.

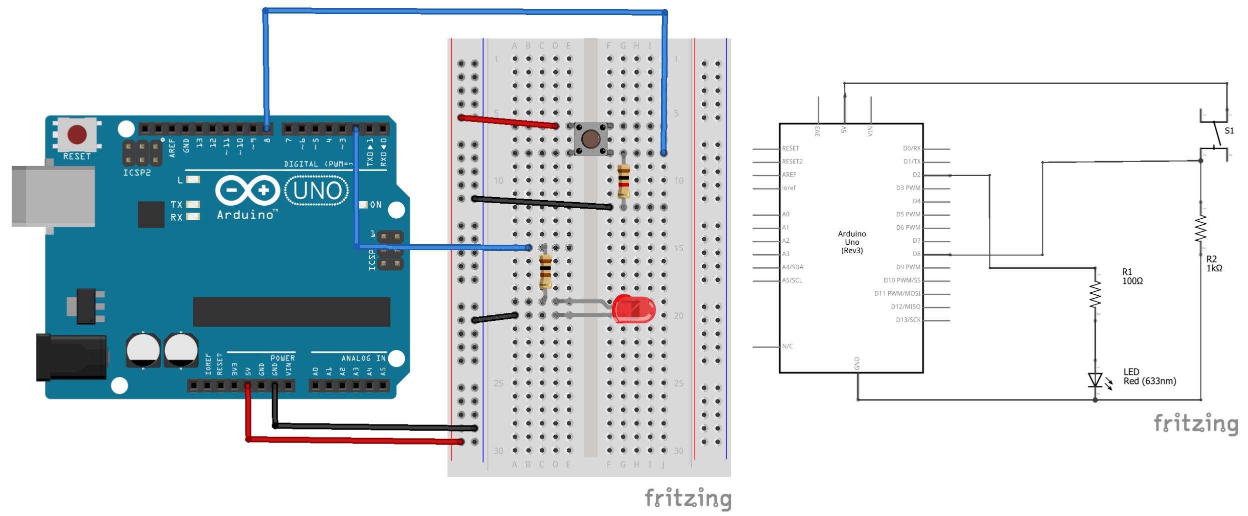


Figure 7- Circuit and schematic for LED push-button

4.5. LED Push-Button Program

This time, let us create names for the pins we intend to use. We will use global variables to describe pins 2 and 8. Global variables are those that are used by all functions in an Arduino sketch (this comes from the C programming language on which the Arduino language is based). Global variables are defined at the beginning and are not inside any functions.

Local variables are those that are confined to the function in which they are used. These are defined within a function and are limited to that function. The C programming language requires all variables to be declared; this means that their type should be defined before use.

We define two global variables to represent the pins; “ledPin” and “pushbuttonPin”. These are integers since they represent pin numbers. The “ledPin” is initialized to 2 since it is connected to pin 2. The syntax is as follows:

```
int ledPin = 2; // ledPin is an integer that takes the value 2, representing the pin number
```

Define and initialize “pushbuttonPin” similarly to represent pin 8.

Define a third global integer variable “buttonState” in order to represent the state of the push-button and initialize it to 0.

Any time we need to refer to pin 2, we now write ledPin. Any time we would like to refer to pin 8, we replace it with pushbuttonPin.

4.5.1. setup() function

In the setup() function, similar to project 1, use the pinMode command to define ledPin to be an output pin. Define pushbuttonPin to be an input pin. The syntax is similar.

4.5.2. loop() function

In the loop() function, we now need to read the state of the digital pushButtonPin. This is done using the digitalRead command as follows:

```
buttonState = digitalRead(pushbuttonPin); // Read the state of the digital pin 8 to which the push-button is
connected.
```

We then write an if loop which determines if the value just read is HIGH (push-button pressed) or LOW (push-button not pressed). If the push button is pressed, we turn pin 2 on which lights up the LED. If not, we turn pin 2 off. The if loop is as follows:

```
if (buttonState == HIGH) {
        digitalWrite(ledPin, HIGH); // power ledPin and turn ledPin on if button is pressed
 }
 else {
        digitalWrite(ledPin, LOW); //do not power ledPin if button is not pressed
 }
```

This completes the program.

4.6. Upload to Arduino

Save, verify and upload to Arduino as before.

5. Guided Project Three- Potentiometer

In this simple experiment, we recreate the blinking LED project, but use a potentiometer to vary the interval between blinks. We also use the serial monitor to view the digital-converted voltage that the Arduino is reading.

5.1. Equipment

Same as Project 1 with the addition of one 1KΩ potentiometer. The potentiometers included in the packs available in the lab are 1KΩ (designated T102) and 10KΩ (designated T103).

5.2. Potentiometer

A potentiometer is a variable resistor that changes resistance with the rotation of a knob or the movement of a slider. The 1KΩ potentiometer resistance can be varied by turning its knob between 0 and 1KΩ.

Potentiometers have three pins. The outer pins are connected to the voltage source and ground. The middle pin is connected to a wiper which provides the variable resistance. The potentiometer and its schematic operation are shown in Figure 8. Pins A and B can be connected to voltage source and ground. By turning the knob, the potentiometer provides variable resistance between pin A and W as well as between W and B.

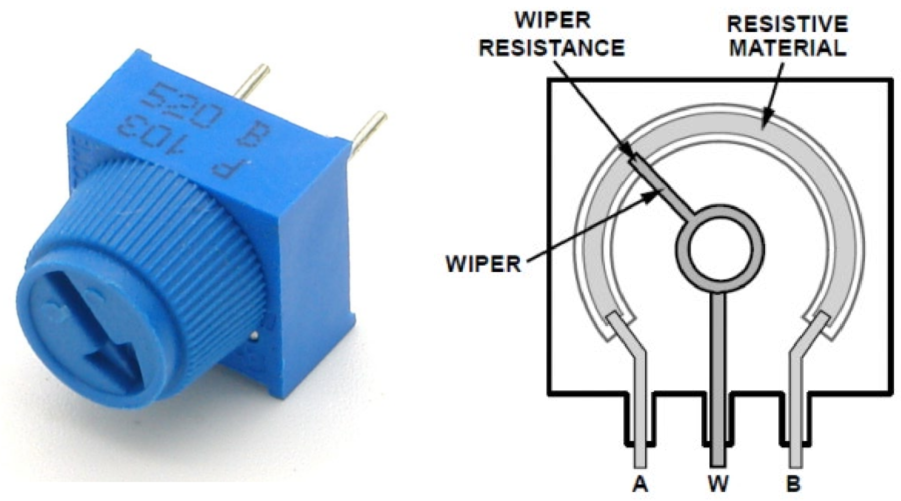


Figure 8- A potentiometer and its schematic operation

5.3. Project Concept

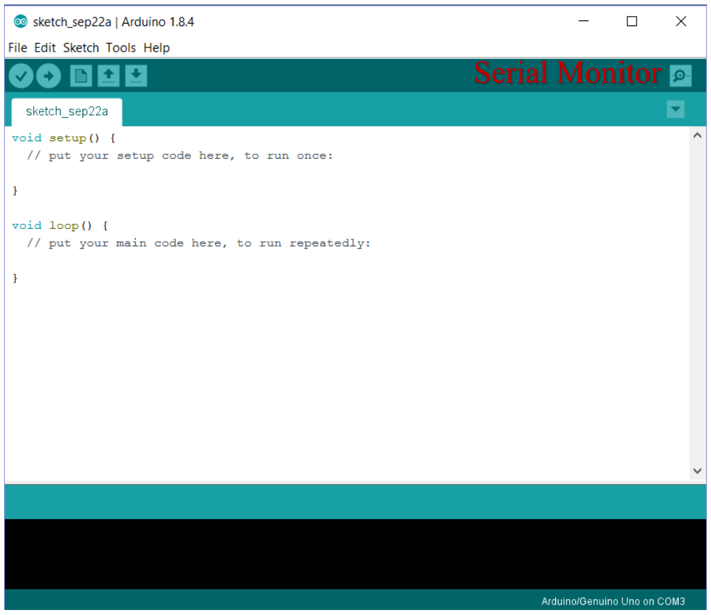
Since a potentiometer can provide variable resistance between the middle pin and the end pins, this indicates that the voltage at pin W can be varied by rotating the knob. Therefore, we can connect the middle pin W of a potentiometer to an analog Arduino input, read the voltage and use this number as the delay time between LED blinks.

5.4. Analog to Digital Conversion

The Arduino board has a built-in 6 channel, 10-bit analog to digital converter. This ADC maps an input analog voltage between 0 and 5V to integers between 0 and 1023. In the software, this is incorporated in the analogRead(pin) function.

5.5. Serial Communication

Communication between Arduino and the computer through the USB cable is based on the serial protocol. Serial refers to when data is streamed between two pints one bit at a time. Arduino has a built-in Serial Monitor which can be used to observe the communication and also to interact with Arduino. The Serial Monitor is on the right end of the Quick Action Buttons Bar as shown in Figure 9.


sketch_sep22a | Arduino 1.8.4
File Edit Sketch Tools Help
Serial Monitor
sketch_sep22a
Arduino/Genuino Uno on COM3

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 9- Serial Monitor on Quick Action Bar

To use the Serial Monitor, we first have to set the rate of communication. This is done using the following command in the setup() function.

```
Serial.begin(9600);  // set up serial communication at 9600 bps
```

The rate of communication is called baud rate and is measured in bits per second. 9600 is a common baud rate.

In the loop(), we can then print information using, for instance, the following "print line" command:

```
Serial.println(value);  // prints the value to Serial Monitor with ending line break
```

5.6. Circuit and Circuit Schematic

The circuit and its schematic representation are shown in Figure 10.

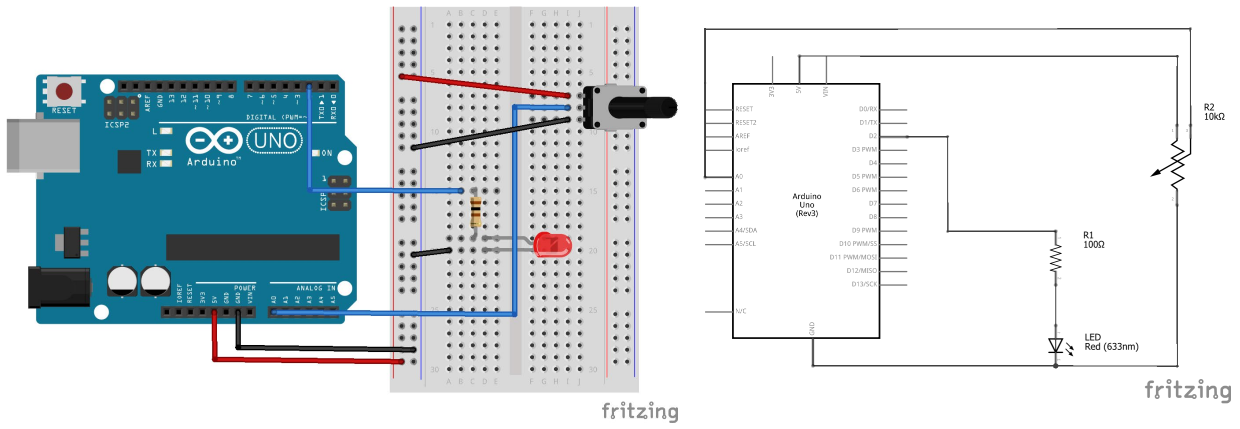


Figure 10- Circuit and schematic of potentiometer and LED

5.7. Program

5.7.1. <u>Variable initialization</u>

We define three global variables for the LED pin, the potentiometer pin and the value read from the potentiometer.

ledPin refers to the pin to which the LED is connected and this is initialized to 2. Define this as before.

```
int ledPin = 2
```

The second global variable represents the pin connected to the potentiometer. We call this variable potPin and set its value to A0 (or simply 0). It is an integer.

```
int potPin = A0; // potPin represents the Analog pin A0
```

The third global variable is the integer representing the value read from the potentiometer. Let us call this potVal. Define potVal as an integer as before.

```
int potVal; //Defines potVal as an integer
```

5.7.2. <u>setup()</u>

In the setup() function, set ledPin to be output. Also set the baud rate of communication to 9600 as explained in Section 5.5.

We do not need to set the pin A0 to be input as this is done by the program when analogRead is called in the loop() function.

5.7.3. loop()

The first step is to read the voltage on the potentiometer pin which was defined as potPin, map it to digital and and store it in the defined integer variable potVal. This is done using the analogRead function as follows:

```
potVal = analogRead(potPin); // Reads the analog potentiometer pin and stores its value in potVal.
```

We then print this value to Serial Monitor as explained in Section 5.5

```
  Serial.println(potVal); // print line the potentiometer
```

Finally, use this value as the delay() value in the blink process. This is similar to the last four lines of the program from the LED Blink, except we use potVal in the delay.

5.8. Upload to Arduino

Save, verify and upload as usual, remembering to open the Serial Monitor.

6. Exercise Project One

The Arduino budget pack includes two transparent LEDs. Create a circuit and write an appropriate program that repeats the following pattern.

- One LED turns on, delay one second;
- The other LED turns on, delay one second
- Turn off all LEDs
- One LED blinks five times, one second on, one second off, red turns off
- The other LED blinks five times, one second on, one second off, green turns off
- Note that each LED must be made series with a resistor.
- Hint: Use while or if loops. Many online references explain the syntax and the process.

7. Exercise Project Two

The Arduino packs in the lab include a photoresistor. This is a sensor that changes resistance based on the intensity of light. The resistance of a photoresistor decreases as the intensity of light received by it increases. Create a circuit and an appropriate program that turns an LED on when the photoresistor is covered (by hand or a piece of paper).

- Hint: First read the digitally-mapped voltage reading from the photoresistor in ambient lab light, similar to Project three. Then use this number to construct an if loop which turns an LED on if voltage falls significantly below this number.