ME133 - Lab 2 Guide

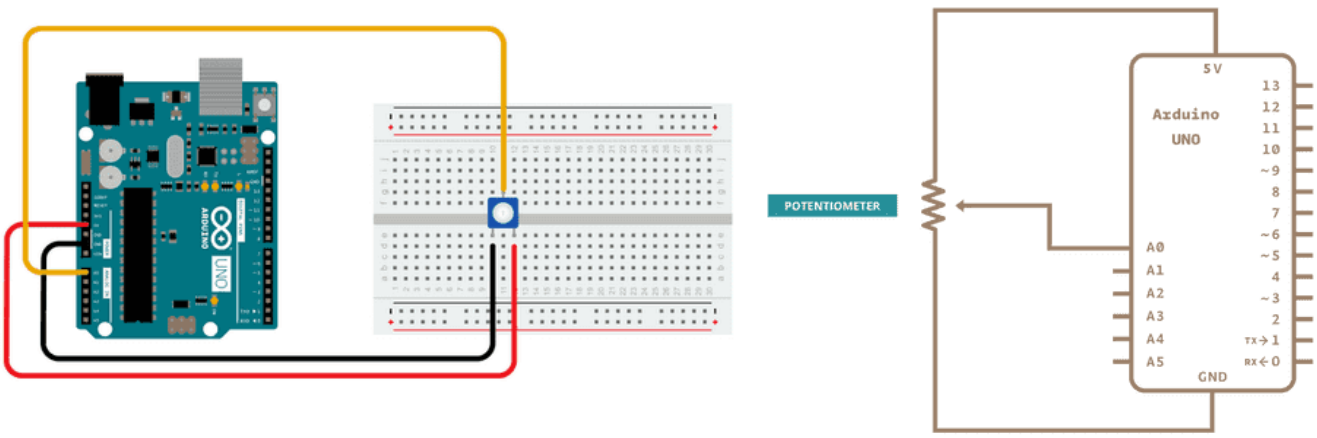
Serial.begin(speed) & Serial.print() & Serial.println()

"Speed" in Serial.begin means data rate in bits per second for serial data transmission, when using "Serial Monitor" or "Serial Plotter" from **Tools**, we should set the same data rate to generate communication. To view data in monitor or plotter, we need Serial.print() or Serial.println(). The "ln" means new line, so that println() will print the content first and change line, but print() will not. By using this kind of function, we could print text or data as we need. For example, "Serial.print("My Age = ");" will print out "My Age = " directly, "Serial.println(agedata);" will print the number we set on "agedata" and change a new line.

analogRead(pin)

Open example from Arduino IDE: File -> Examples -> Basics -> AnalogReadSerial and connect circuit below.

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

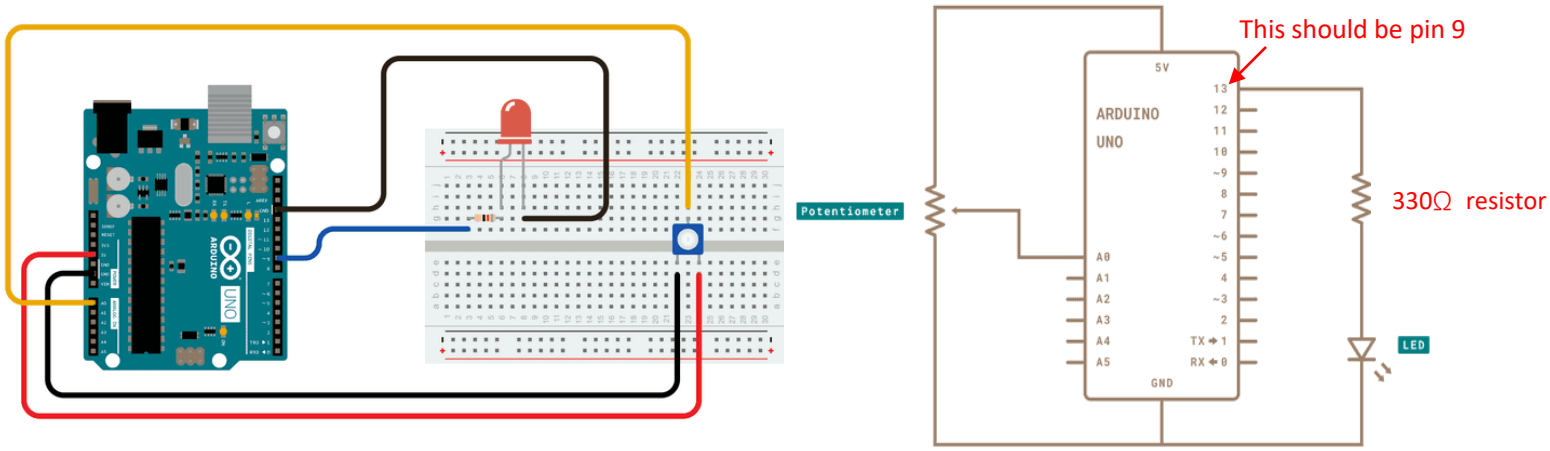


This code will print out integer values from 0 to 1023 instead of 0 to 5 volts. To transfer the integer value to voltage value, we can multiply 0.0049, which means 5 volts per 1024 units.

analogWrite(pin) & map(value, fromLow, fromHigh, toLow, toHigh)

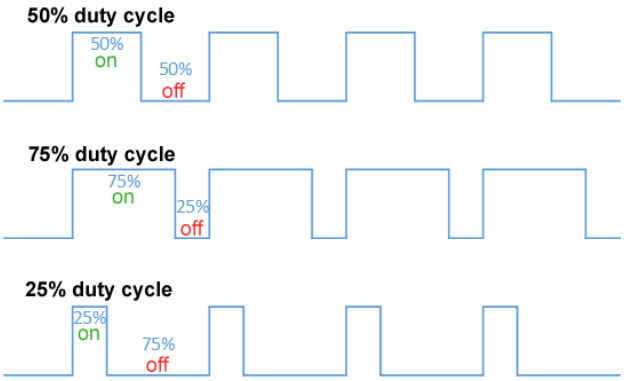
Open example from Arduino IDE: File -> Examples -> Analog -> AnalogInOutSerial and connect circuit below.

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital
  // converter to settle after the last reading:
  delay(2);
}
```



analogWrite will let the pin generate a pulse-width modulation (PWM) signal to control motor speed or light brightness. The PWM signal is a method of reducing the average power by adjusting on and off time, which is called duty cycle. The term duty cycle describes the proportion of 'on'

time to the regular interval or 'period' of time; a low duty cycle corresponds to low power, because the power is off for most of the time.

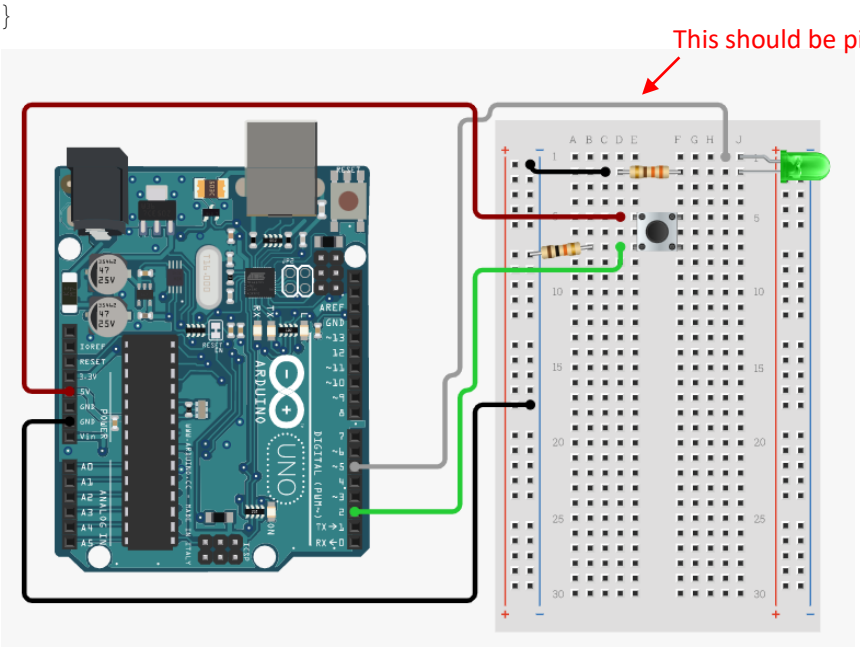


In Arduino, the duty cycle ranges from 0 to 255 representing 0 to 100%. To transfer data from analogRead to analogWrite, we could use map( ) function to re-map “0 – 1023” to “0 – 255”.

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)

Here is an example code from this website.

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```



The pin for interrupting should be set as “INPUT_PULLUP” mode in “setup”. The logic is once the pin has one of the modes, the system starts running function “ISR” (or blink in the example)

There are four modes:

- **LOW** to trigger the interrupt whenever the pin is low,
- **CHANGE** to trigger the interrupt whenever the pin changes value
- **RISING** to trigger when the pin goes from low to high,
- **FALLING** for when the pin goes from high to low.

The Due, Zero and MKR1000 boards allow also **HIGH** to trigger the interrupt whenever the pin is high.