Pulse Width Modulation (PWM)

Materials:

- 2 Series Mixed Series Oscilloscope (MSO)
- Arduino UNO
- Potentiometer
- Breadboard
- Jumper wires

Procedure:

Task 1: Varying Duty Cycles

1. Pulse Width Modulation (PWM) is a technique used to control the amount of power delivered to an electrical device by varying the width of the pulses in a signal. In PWM, a digital signal alternates between on and off states, and the ratio of the on-time to the total period of the signal (known as the duty cycle) determines the average power delivered. This lab will use an Arduino UNO to generate PWM signals to observe on the 2 Series MSO.

2. Begin by attaching the channel 1 probe to a PWM pin on the Arduino UNO (Ex: Pin 3). Make sure to attach the probe ground to one of the GND pins on the Arduino. Write an Arduino script that declares the PWM pin and outputs PWM signals with a duty cycle of 25%, 50%, 75% and 90%. Upload the script to the Arduino and turn on channel 1 on the oscilloscope to observe the PWM signal.

3. Capture a period with a few PWM signals. To measure the duty cycle of the signals, tap the "Cursors" button to show the two cursors. First, measure the period of the signal by user cursor A to measure the first rising edge and cursor B to measure the next rising edge. Record this period in Table 1. Measure the section of the signal that is active or high using the two cursors and record that in Table 1. Calculate the duty cycle by using the following formula:

$$duty\ cycle = \frac{time\ high}{period} \times 100\%$$

Repeat these steps for all four duty cycles and record the results in Table 1.

Coded Duty Cycle	Period	Time High	Calculated Duty Cycle
25%			
50%			
75%			
90%			

Table 1. Measured period, time high and calculated duty cycles.

4. What is the percent error of the coded duty cycles compared to the measured? What is the frequency of the PWM signal? What would happen if the duty cycle were 0% or 100%?

Task 2: Adding a Potentiometer

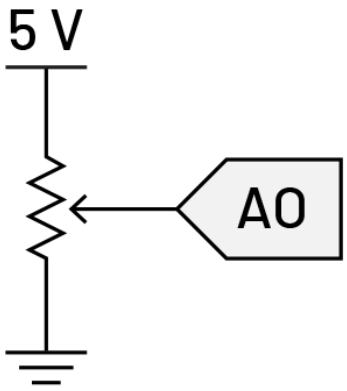

Figure 1. Potentiometer wiring diagram.

1. Add a potentiometer to the circuit in used in Task 1 by attaching the potentiometer to the 5V, GND and A0 pins on the Arduino as shown in Figure 1. The varying voltage drop across the potentiometer will act as an analog input to control the PWM duty cycle.

2. Write a script that continuously outputs a PWM signal to pin 3 based on the reading on the potentiometer. Add a short delay in between each reading. Make sure to map the potentiometer reading to the PWM range.

3. Turn on channel 1 and make sure the probe is still attached to pin 3 and the Arduino GND. Adjust the horizontal scale to show a few periods of the PWM signal. Rotate the potentiometer back and worth to see the duty cycle increase and decrease. Does the signal completely turn off and on? What are some applications that PWM can be used?

Instructor Notes:

Figure 2. Using the cursors to measure the period of the PWM signal.

Coded Duty Cycle	Period	Time High	Calculated Duty Cycle
25%	2.044 ms	507.82 us	24.8%
50%	2.044 ms	1.014 ms	49.6%
75%	2.044 ms	1.529 ms	74.8%
90%	2.044 ms	1.833 ms	89.7%

Table 2. Answered period, time high and calculated duty cycles.

Arduino Code for Task 1:

```
const int PWMPin = 3;

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

void loop(){
analogWrite(PWMPin, 64); //25% duty cycle
delay(2000);
analogWrite(PWMPin, 128; //50% duty cycle
delay(2000);
analogWrite(PWMPin, 192); //75% duty cycle
delay(2000);
analogWrite(PWMPin, 230); // 90% duty cycle
delay(2000);
}
```

Arduino Code for Task 2:

```
const int potPin = A0;  // Analog input pin for the potentiometer
const int pwmPin = 3;   // PWM output pin

void setup(){
 // Initialize the PWM pin as an output
 pinMode(pwmPin, OUTPUT);
}

void loop(){
 // Read the potentiometer value (0 to 1023)
 int potValue = analogRead(potPin);

 // Map the potentiometer value to a PWM duty cycle (0 to 255)
 int pwmValue = map(potValue, 0, 1023, 0, 255);

 // Output the PWM signal
 analogWrite(pwmPin, pwmValue);

 // Wait for a short time before reading the potentiometer again
 delay(10);
}
```