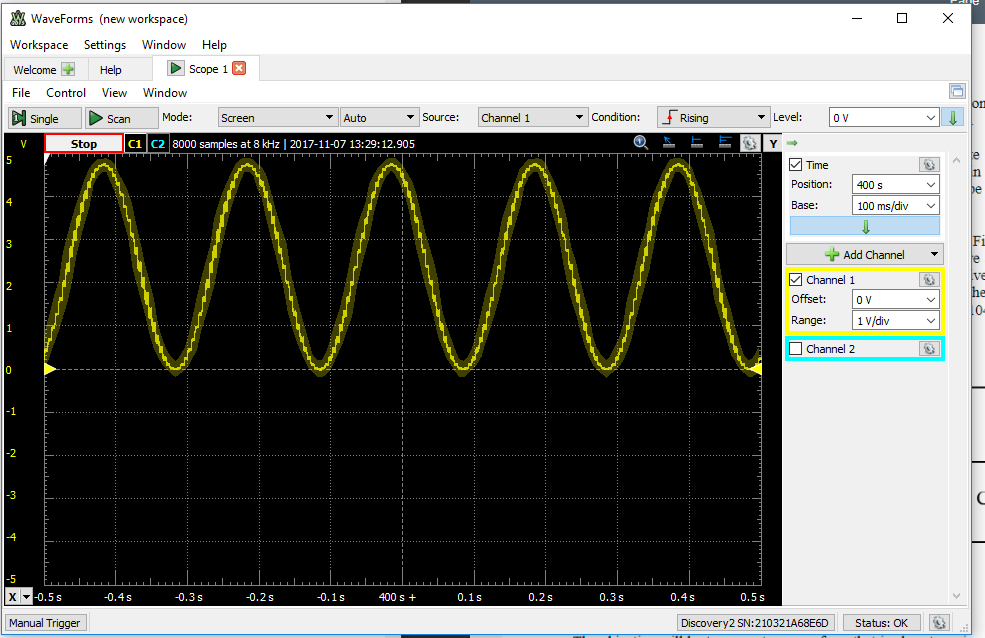
Objective

* The objective of this lab was to gain understanding of Digital to Analog Converters and the effect of the RC Circuit and the period of the Pulse Width Modulation on the output waveform.

Parts

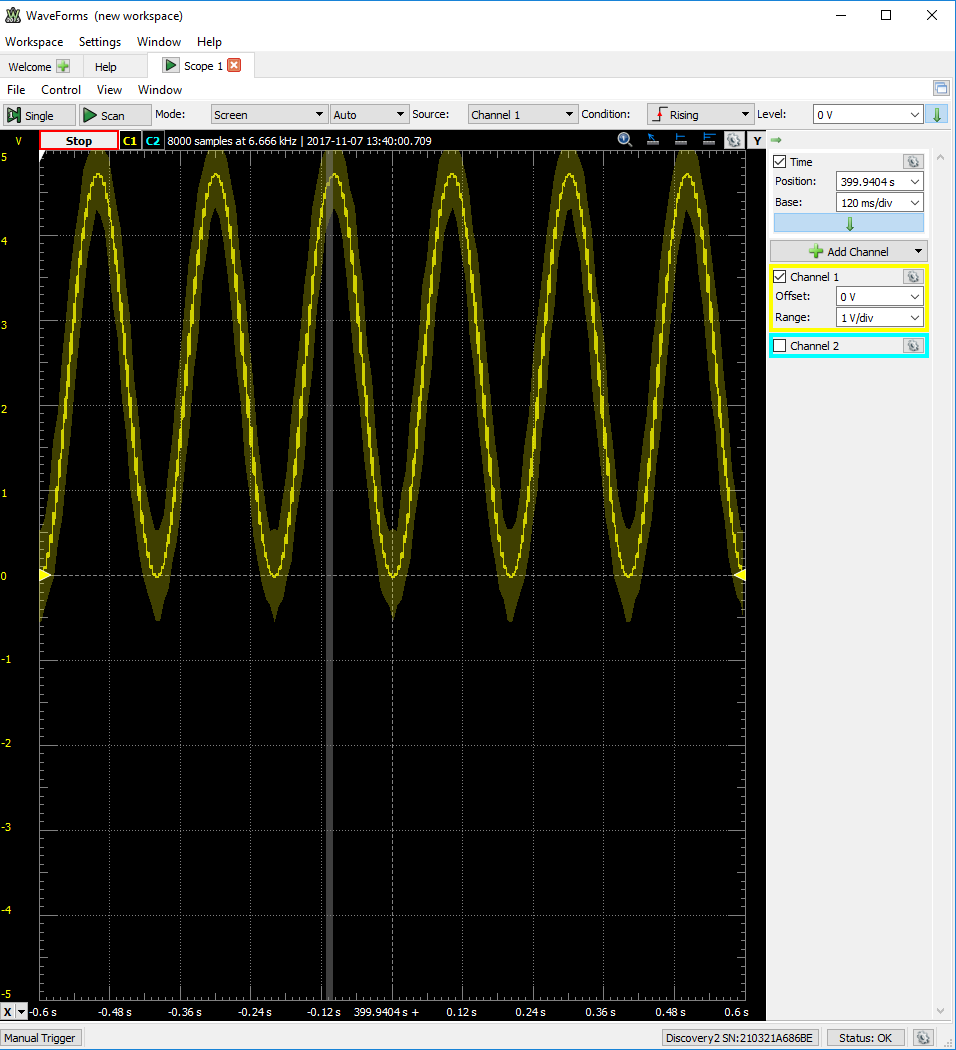
1. For the first part, we set up a circuit using a 5 Ohm resistor and a 0.1 micro Farad capacitor. We set the frequency in the code to 5 Hz. We then set up the analog discovery as shown in the lab descriptions Figure 10-1. We then captured the waveform generated and demonstrated it to our lab instructor. **Figure 1**, shows the waveform I captured in part 1. The code for this is depicted in **Appendix A**.



**Figure 1:** This is the waveform generated by the code from the prelab (with the frequency set to

5 Hz). As you can see, the wave is slightly jagged, and not very smooth.

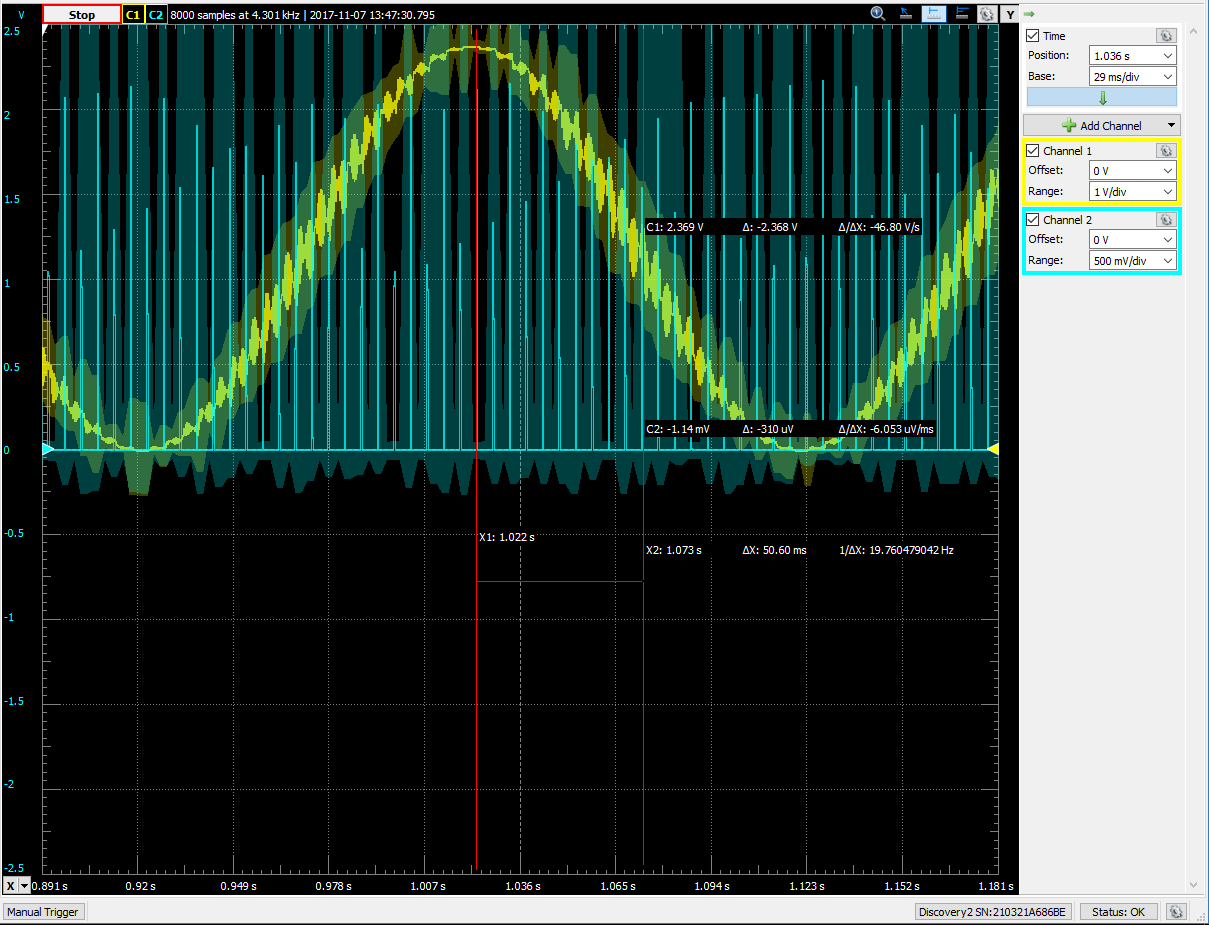
1. For the second part, we changed the capacitance to .2 micro Farads. **Figure 2** depicts what the waveform generated looks like. The code for this is depicted in **Appendix A**.



**Figure 2:** Here we can see a slight variation in the consistency of the wave. It is a bit smoother,

but due to stretching and resizing the original image captured, it may not appear so. Raising the capacitance should have softened the appearance of the wave.

1. For part three, we connect V2 of the AnaDisc to pin 13. We then measured the number of the pulses per second and the length of the average pulse. We calculated the number of pulses per second to be approximately 200p/s. The length of the ISR’s run time varies and was inconsistent. **Figure 3** depicts what the AnaDisc displayed. The code for this is depicted in **Appendix A**.



**Figure 3:** Here we can use these measurements to calculate the pulses per second.

1. For the fourth and final part, we replaced the RC Circuit with the LED Circuit shown in the labs Figure 10-2. We then set the frequency to .5 Hz and demonstrated it to the lab instructor. The code for this is depicted in **Appendix B**.

**Appendix A (Parts 1 – 3)**

#include <TimerOne.h>

#include <MsTimer2.h>

#define ANALOGWRITE

// Pins used

#define AnalogPin 0

#define AnalogOutputPin 10

float Time;

int Frequency = 5;

float Period;

float x = 0;

//Interupt service routine

void pwm\_ISR()

{

digitalWrite(13, HIGH); //Turns LED on

Time += 5e-3; //Advance time by sample interval

if (Time > Period)

{

Time -= Period; //Keeping check on Time so it won't grow too large

} //end if

x = 511\*sin(6.2831853\*Frequency\*Time) + 512;

digitalWrite(13, LOW); //Turns LED off

} //end ISR function

void setup()

{

MsTimer2::set(5, pwm\_ISR); //sets the time in ms

MsTimer2::start(); //Enables the interrupt

Timer1.initialize(250); //Initializes a timers period in microseconds

Timer1.pwm(10, 512); //Generates a PWM waveform on pin 10

pinMode(13, OUTPUT); //Set pin 13 to ouput for LED

pinMode(10, OUTPUT); //Set pin 10 to output for PWM

}

void loop()

{

// put your main code here, to run repeatedly:

}

**Appendix B (Part 4)**

#include <TimerOne.h>

#include <MsTimer2.h>

#define ANALOGWRITE

// Pins used

#define AnalogPin 0

#define AnalogOutputPin 10

float Time;

float Frequency = .5;

float Period = 1.0 / Frequency;

float x = 0;

//Interupt service routine

void pwm\_ISR()

{

digitalWrite(13, HIGH); //Turns LED on

Time += 5e-3; //Advance time by sample interval

if (Time > Period)

{

Time -= Period; //Keeping check on Time so it won't grow too large

} //end if

x = 511\*sin(6.2831853\*Frequency\*Time) + 512;

Timer1.setPwmDuty(10, x);

digitalWrite(13, LOW); //Turns LED off

} //end ISR function

void setup()

{

MsTimer2::set(5, pwm\_ISR); //sets the time in ms

MsTimer2::start(); //Enables the interrupt

Timer1.initialize(250); //Initializes a timers period in microseconds

Timer1.pwm(10, 512); //Generates a PWM waveform on pin 10

pinMode(13, OUTPUT); //Set pin 13 to ouput for LED

pinMode(10, OUTPUT); //Set pin 10 to output for PWM

}

void loop()

{

// put your main code here, to run repeatedly:

}