

Lab 2: Analog Output(PWM), Timers and Analog Input

Professor Campisi

Kiersten Page

Setting up the Analog Signal:



The screenshot shows the Arduino IDE interface with a window titled "lab_2 | Arduino 1.8.10". The code editor contains the following C++ code:

```
void setup() {  
  // put your setup code here, to run once:  
  Serial.begin(9600);  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  float aval;  
  long x;  
  x = millis();  
  aval = abs(3 * sin(2 * 3.141592654 * x / 1000));  
  delay(1);  
  Serial.println(aval);  
}
```

Below the code editor, a status bar indicates "Save Canceled." and provides memory usage information: "Sketch uses 3670 bytes (11%) of program storage space. Maximum is 32256 bytes. Global variables use 200 bytes (9%) of dynamic memory, leaving 1848 bytes free." The bottom status bar shows the line number "14" and the board name "Arduino Uno on /dev/cu.usbmodem14101".

Figure 1: Code Setup

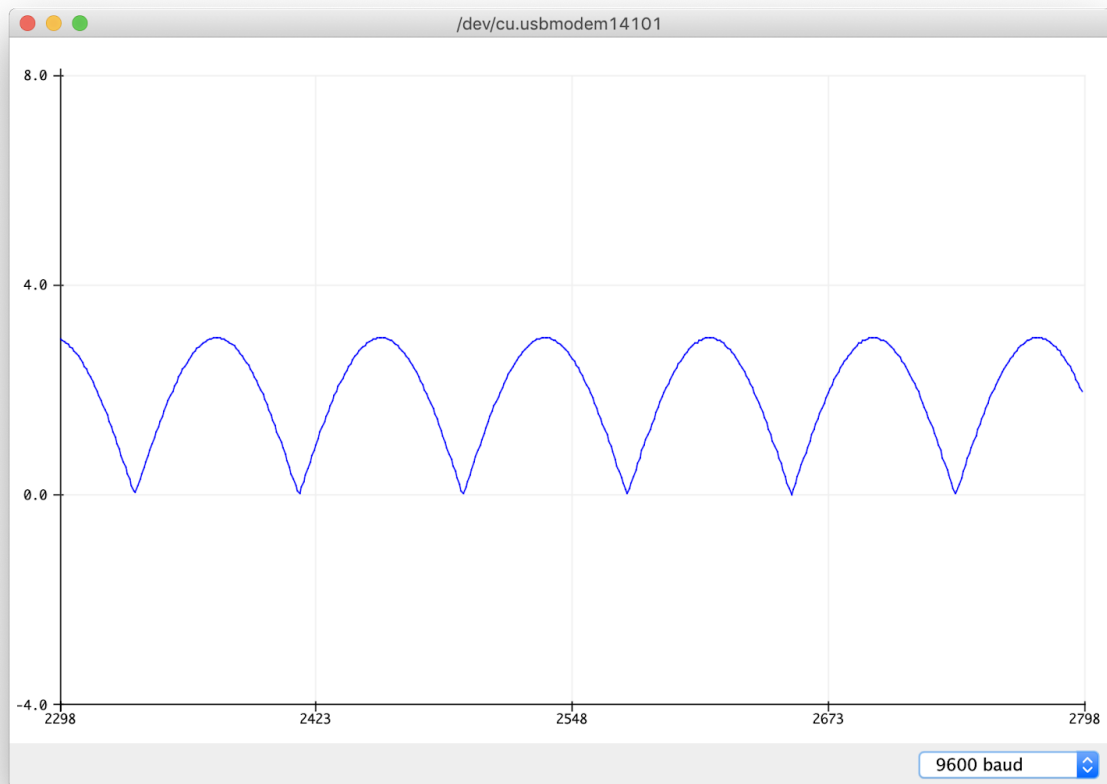


Figure 2: Sinusoid on serial plotter

Setting up the PWM signal:

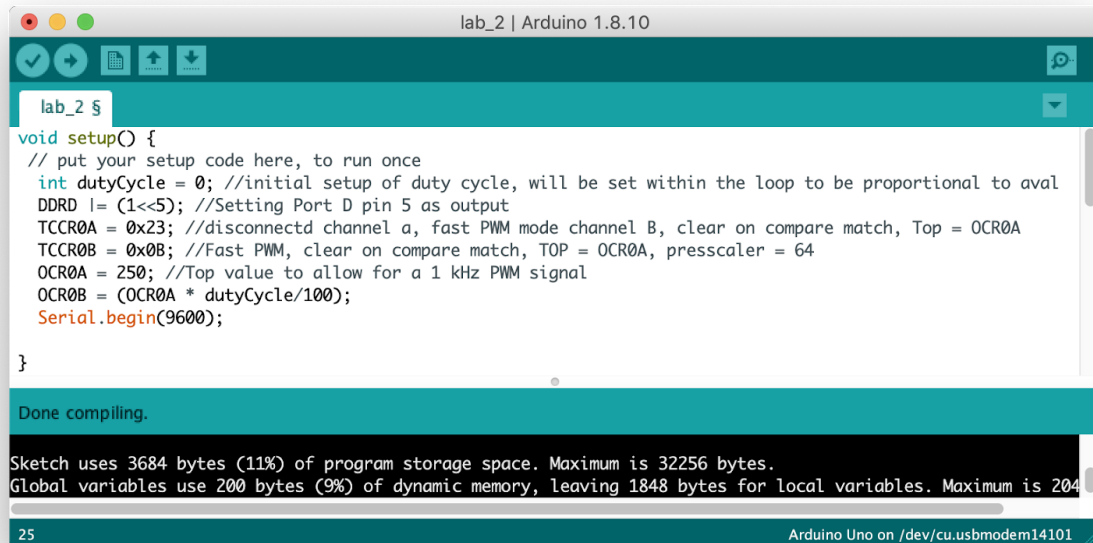
(1) Explain which timer must be used for this PWM setup.

- (a) We will use timer 0 which is an 8-bit timer for this PWM setup as it allows for the actual frequency to be 1kHz. Timer 2 could also be used.

(2) Show the code that sets up these registers and explain your bit selections

- (a) For the DDRD, pin 5 is set to 1 so that it is registered as an output. TCCR0A is set to 0x23 such that channel A is disconnected, channel B is set to fast PWM mode and clear on compare match, and the top is set to OCR0A. TCCR0B is set to 0x0B such that channel B is in fast PWM mode and clear on compare match,

the top is OCR0A is set to 250 to allow for a 1 kHz signal. OCR0B is set to be proportional to the set duty cycle.



```
lab_2 $  
void setup() {  
  // put your setup code here, to run once  
  int dutyCycle = 0; //initial setup of duty cycle, will be set within the loop to be proportional to aval  
  DDRD |= (1<<5); //Setting Port D pin 5 as output  
  TCCR0A = 0x23; //disconnectd channel a, fast PWM mode channel B, clear on compare match, Top = OCR0A  
  TCCR0B = 0x0B; //Fast PWM, clear on compare match, TOP = OCR0A, prescaler = 64  
  OCR0A = 250; //Top value to allow for a 1 kHz PWM signal  
  OCR0B = (OCR0A * dutyCycle/100);  
  Serial.begin(9600);  
}
```

Done compiling.

Sketch uses 3684 bytes (11%) of program storage space. Maximum is 32256 bytes.
Global variables use 200 bytes (9%) of dynamic memory, leaving 1848 bytes for local variables. Maximum is 2048 bytes.

25 Arduino Uno on /dev/cu.usbmodem14101

Figure 3: Code that sets up the registers.

(3) Have your lab instructor verify using an oscilloscope (channel 1) that output pin 5 is indeed the correct PWM and has the correct characteristics as previously described.

(a) Due to COVID-19, this portion of the lab was unable to be completed.

Setting up the Demodulator:

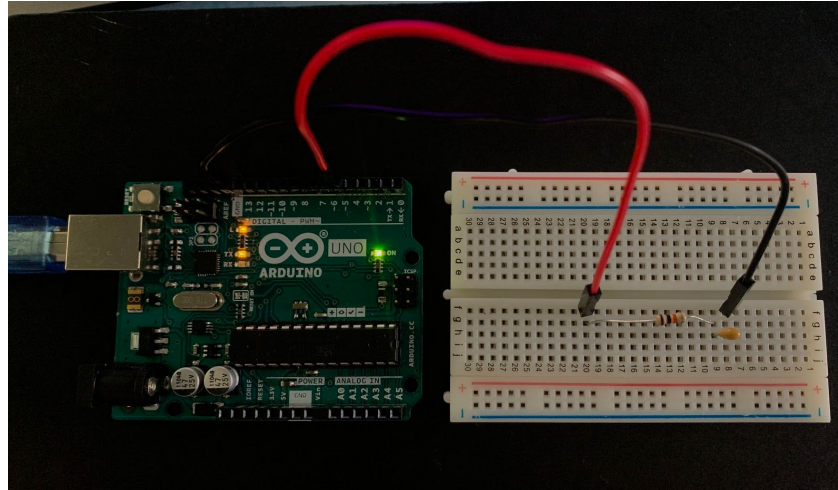


Figure 4: Single pole RC low-pass filter

(4) Indicate the frequency of the PWM signal, the frequency of the analog signal, and the cutoff frequency ($1/RC$) of the filter. Confirm that the values are adequate.

- (a) PWM signal frequency: 1 kHz
- (b) Analog Signal Frequency: 1Hz
- (c) Cutoff Frequency of the filter: 100 Hz
- (d) These values are all adequate as the PWM signal frequency is much more than $1/RC$, and the analog signal frequency is much less than $1/RC$


(5) Show the lab instructor the trace and the P-P and frequency values for channel 2.

Is this analog signal a good representation of the original analog signal? Explain.

- (a) Due to COVID-19, this portion of the lab was unable to be completed.

Setting up the analog input

(6) Show the lab instructor the code to set up the 5 registers described above



The screenshot shows the Arduino IDE interface with a file named 'lab_2'. The code in the editor is as follows:

```
DDRC &= (0<<0); //set port c pin 0 as an input
ADMUX = 0x40; //vcc as reference, right justified
ADCSRA = 0xA4; //16 prescaler
ADCSRB = 0x00; //free running mode
ADCSRA |= (1<<6); //start conversion
DIDR0 |= (1<<0); //disable digital access to pin A0

}
```

Below the code editor, a status bar indicates 'Done compiling.' and provides memory usage details: 'Sketch uses 3728 bytes (11%) of program storage space. Maximum is 32256 bytes. Global variables use 200 bytes (9%) of dynamic memory, leaving 1848 bytes for local variables. Maximum is 2048 bytes.' The bottom status bar shows '22' and 'Arduino Uno on /dev/cu.usbmodem14101'.

Figure 5: Register configuration

(7) Show the lab instructor your final code and the serial output described above

```
lab_2 §
void setup() {
  // put your setup code here, to run once
  int dutyCycle = 50; //initial setup of duty cycle, will be set within the loop to be proportional to aVal
  DDRD |= (1<<5); //Setting Port D pin 5 as output
  TCCR0A = 0x23; //disconnectd channel a, fast PWM mode channel B, clear on compare match, Top = OCR0A
  TCCR0B = 0x0B; //Fast PWM, clear on compare match, TOP = OCR0A, prescaler = 64
  OCR0A = 250; //Top value to allow for a 1 kHz PWM signal
  OCR0B = (OCR0A * dutyCycle/100);
  Serial.begin(9600);
  DDRC &= (0<<0); //set port c pin 0 as an input
  ADMUX = 0x40; //vcc as reference, right justified
  ADCSRA = 0xA4; //16 prescaler
  ADCSRB = 0x00; //free running mode
  ADCSRA |= (1<<6); //start conversion
  DIDR0 |= (1<<0); //disable digital access to pin A0
}

void loop() {
  // put your main code here, to run repeatedly:
  float aVal;
  long x;
  x = millis();
  aVal = abs(3 * sin(2 * 3.141592654 * x / 1000));
  delay(1);
  int dutyCycle = aVal;
  OCR0B = (OCR0A * dutyCycle/100);
  unsigned short ADCVal;
  if(ADLAR==0){
    unsigned short *ADCData;
    unsigned short ADCVal;
    ADCData=(unsigned short *)0x78;
    ADCVal=(*ADCData & 0x3FF);
  }
  else if(ADLAR==1){
    unsigned short *ADCData;
    unsigned short ADCVal;
    ADCData=(unsigned short *)0x78;
    ADCVal=(*ADCData & 0xFFC0) >> 6;
  }
  float fADCVal;
  fADCVal=((float)ADCVal)/1023 * 3;
  Serial.print(abs(aVal)); //Original rectified sinusoid
  Serial.print(" ");
  Serial.println(fADCVal); //Analog voltage measured from ADC
}

9 Arduino Uno on /dev/cu.usbmodem14101
```

Figure 6: Final Code

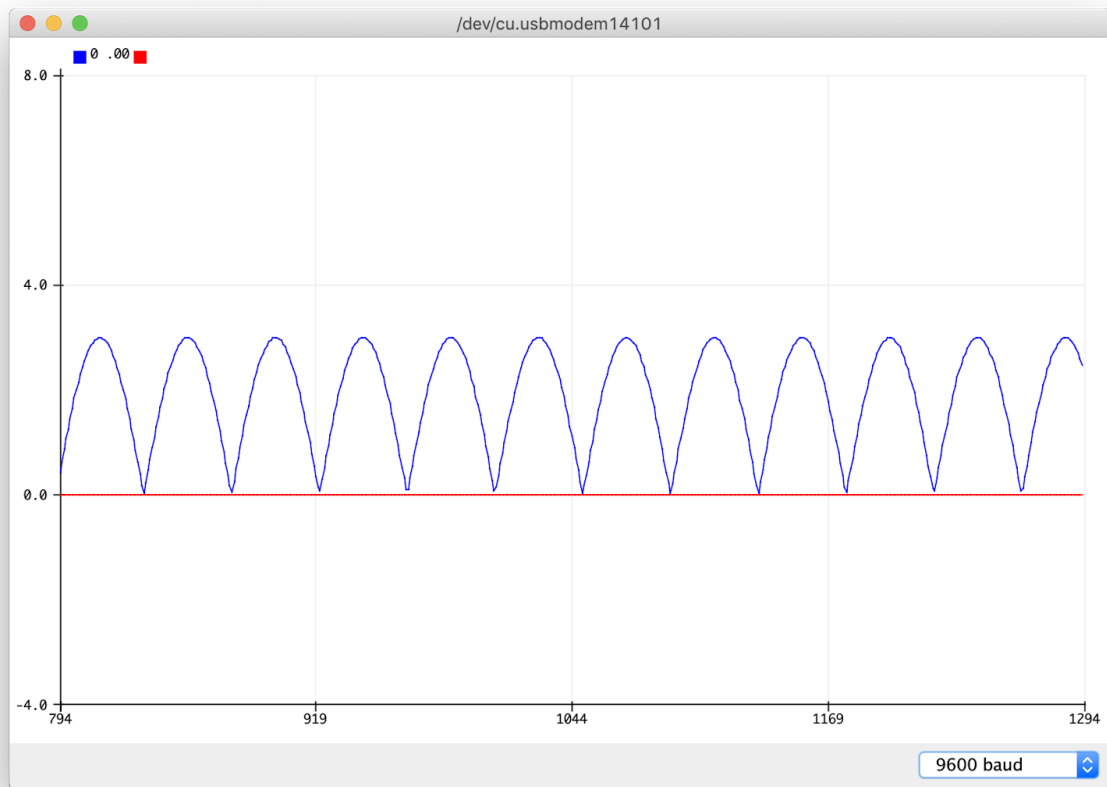


Figure 7: Serial Output of final code

Note: The data points do not have the same approximate value. This is most likely due to the fact that while the “aval” value was being plotted to the serial plotter, the fADCVAL was not which is most likely due to a coding error. If the coding error wasn’t present, then the two values should be quite close to one another.