**Laboratory 1 – Analog output from the Arduino**

**Introduction:**

The Arduino has 14 digital input pins (0-13), 6 analog input pins (0-5). But it does not have a real method of producing an analog output voltage. To control most physical systems, it is necessary to produce an analog output voltage (for example, to control the setting of a heating system, or vary the base voltage on a transistor etc).

In today’s lab, we will look into techniques of producing an analog output voltage from the Arduino .   
You will have to recall some of the analog electronics techniques learnt in earlier labs to optimize the solution for today’s lab.

**Part A: Background and buildup: analogWrite() function**

Looking at the Arduino language reference, you will find, surprisingly, that there does exist a function analogWrite(pin,value) that should allow you to write an effective analog value to a few selected pins (that normally function as digital I/O pins)

1. Which pins can be used by the analogWrite(pin,value)function?  
   **3,5,6,9,10,11 – easily found from looking at documentation of analogWrite function**
2. What is the range of allowed values for the variable value?  
   **0 to 255 (corresponds to the duty cycle of PWM)**

The language reference mentions that ‘PWM’ – pulse width modulation is used to create a digital square wave of adjustable duty cycle corresponding to value on the set pin.

To see how this results in a ‘sort of’ analog output, it is simplest to write a small program using analogWrite(value,pin) with value and pin set within the allowed parameter range. Observe the output obtained at pin on the DSO and answer the following questions.

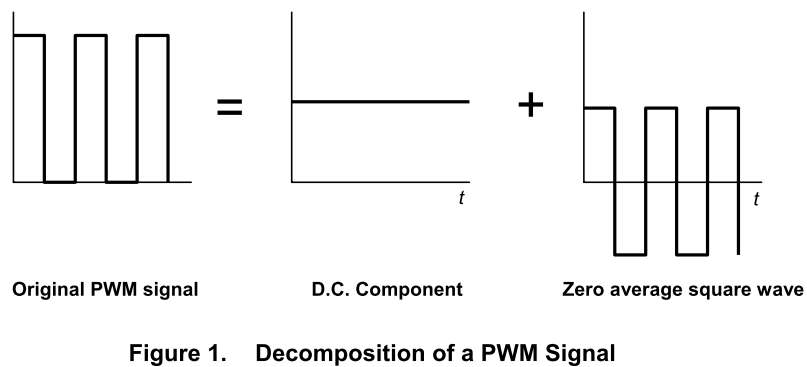
1. What is the shape and frequency of the waveform observed? ***Square wave, 490 Hz***
2. What changes as you vary the value from 0 to 255?   
   Pick ~ 5 representative values in increasing order  
   **Questions 3 and 4 are demo questions for TA to examine – student should be able to demonstrate varying duty cycle. Program is quite simple (must be commented!)**
3. Use the DSO’s math functions to determine RMS value of the voltage measured at the set pin. You will notice that the RMS value increases as the value set by the analogWrite(pin,value) increases.
4. Combine your observations from Q.3, Q.4, Q.5 to determine the   
   *Resolution* ***~1/255\*5 = 0.02V***(defined as the least count, or the smallest step value between 2 settings,   
    note that this may be different over the range) **Resolution varies as SQRT(duty cycle)** *Range* \_\_\_\_\_***0 to 5 V***\_\_ (defined as the minimum and maximum voltage values possible)  
   of the analogWrite function.

For example, if an application requires to produce an analog output voltage in the range   
0 – 5V in steps of 0.01V, would you be able to accomplish the task with analogWrite(value,pin) over the complete range 0 -5V ?

**Part B: Optimizing the analog output from the Arduino**

As you will have noted in the observations for Q.3,4,5,6 of Part A, the analogWrite function does not produce a real analog output. It produces a pulse width modulated digital square wave, which, when averaged over long periods approximates to a proportional analog voltage value.

This *suggests* that the output signal has two components as shown in Fig 1

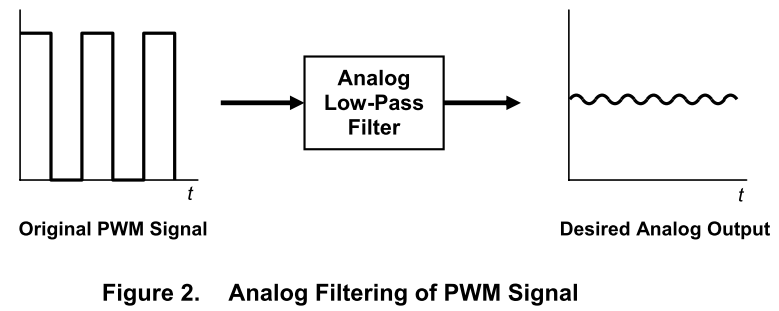


0

5V

It should therefore be possible to *filter* the PWM output signal and obtain a truly analog voltage output.

Determine what type of filter is to be put in to the following block diagram:



Analog  
Filter  
????

1. Filter type (simple passive filter):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***RC low-pass***
2. First guess at passive filter design parameters given the measured frequency of the PWM output:  
     
   ***fC = 1 / 2πRC where fC ~ 10 Hz ≪ 1 kHz (PWM frequency) C = 0.1 μF, R = 100kΩ***
3. Put together such a filter on a bread-board. Use analogWrite(pin,value) function. Connect the output of pin to the input of the filter, and demonstrate the output for a few (~5) increasing settings of value in your program.  
   Draw a detailed time trace diagram of the observed filter output here, noting especially the magnitude of any residual ripple in your filtered analog output.

***Demo for TA – observe ripple magnitude left over from insufficient filtering***

1. Check and demonstrate if the filtered analog output voltage spans the full range 0 – 5V

Yes / No – if not why not? ***YES  
 RC filter will attenuate above fc (at ~ 10\*10Hz) by 1/(1 + 1k \* R \* C ) ~ 1/10 – so ripple is highly attenuated***

**Part C: Optimizing the filtered analog output from the Arduino   
 (Signal Uncertainty)**

Consider the resolution requirements stated in Part A, Q.6 and the final measurements of the filtered output in Part B, Q.3

Putting the two calculations and observations together, put in values on the following diagram to determine the total uncertainty of your filtered analog signal .

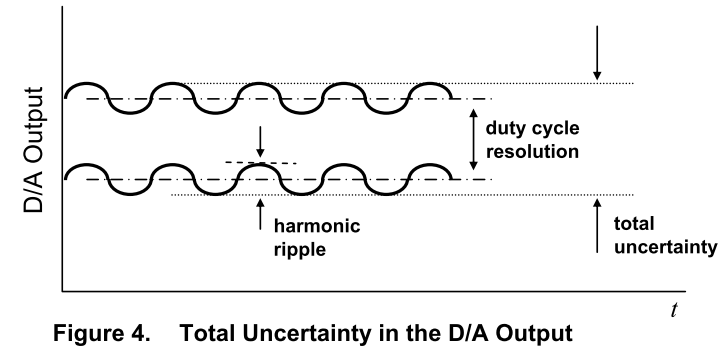
***If setpoint is x;  
actual value will be:***

***x ± 0.02V ± ripple***

***latter two terms =uncertainty***

***± 0.02V → fixed resolution***

***± ripple → can be improved by adding 2nd filter stage***



Q.1

With this level of signal uncertainty, are you able to satisfy the requirement of producing an analog output voltage in the range 0 – 5V with an accurate minimum resolution of 0.02 V?

***Almost, resolution is 0.02V – depending on RC filter values chosen, ripple may be ≪ 0.02V so***

***total uncertainty would be close to 0.02V***

Q.2 How would you decrease the total uncertainty, i.e. improve accuracy of the filtered analog output?

*Hint: your duty cycle resolution is obviously limited by the internal ATmega hardware, so there is only one other parameter in Figure 4 that you can improve! Also, RC component values chosen for passive filter must be realistic!*

***To keep within ~ 0.1 μF and ~ 100kΩ range, must add second filter stage instead of increasing  
values of R,C for the one stage RC filter***

**Part D: Optimizing the filtered analog output from the Arduino   
 (Drive capability)**

As we have learnt earlier in analog electronics, impedances matter when you introduce passive filter blocks in the signal chain. The Arduino is driving a digital (0-5V) signal into your analog filter(s). It *probably* has sufficient capability to drive the input impedance of the analog filter(s).

However when an external device of unknown (possibly very small) input impedance is connected to the output of the filter, the analog output voltage will likely get distorted due to impedance mismatching. Add an active element to the output of your circuit to make it capable of driving any load impedance connected to its output.

(+0.5 bonus points for extra things you can do with the active stage)

**Optional G can be put in to increase range of   
analog output beyond 0 – 5V**

**0.5 marks bonus:**

**Note opamp driver can also**

**extend range of analog out**

**to negative values with ± V**

**using DC bias at + terminal of  
opamp**

***Ideal solution:***

Arduino

1

RC

RC

G

PWM

/14.5

/1+1

/2

/0.5 \* 5

/1+1

A.1-A.5

A.6

B.1,2

B.3,4

/2.5

C.1

/1.5

C.2

/2

D

Total