University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 160 Name: pwm_main.c
Atmega 3 PWM Due: see http://ece160.org

Write a program to change the color of an RGB LED. Your program should query the three potentiometers and, based on the potentiometer (pot) setting, adjust the brightness of the Red (Channel 0), Green (Channel 1), and Blue (Channel 3) LED. When the pot is fully counter clockwise, the associated color should be completely off. When the pot is fully clockwise, the associated color should be completely on.

Analog Inputs:

You can read in an analog input on any of the analog pins (port C) of the Atmega. To do this you call *analogRead(). analogRead()* takes one argument, the pin to read in and returns a 10 bit value (0-1023 or 0-3FF) representing the voltage on the pin.

Pot	ADC Channel
Leftmost, with buttons at bottom	Channel 2
Middle	Channel 1
Rightmost, with buttons at bottom	Channel 0

READING ANALOG VALUES:

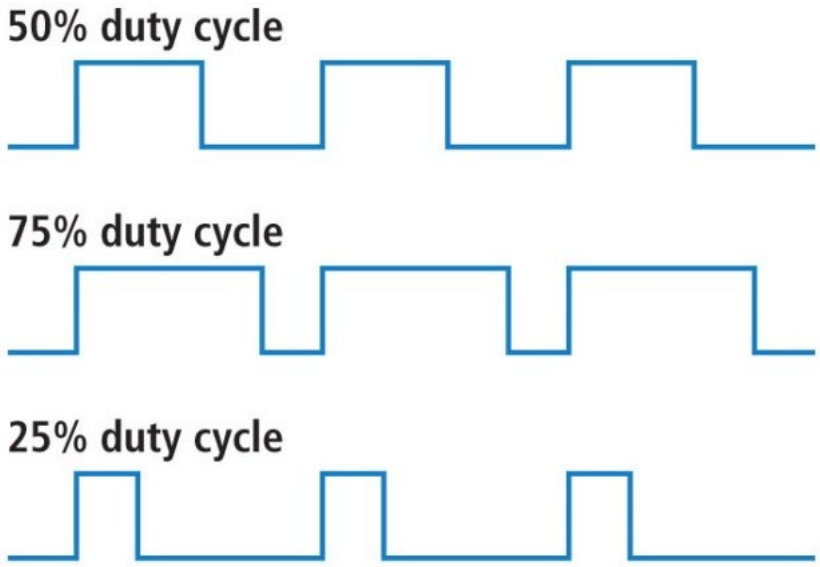
I am providing you with two “magic” routines to read the setting of the potentiometers

```
uint16_t AnalogRead16(uint8_t channel)
{
	ADMUX=64+channel;ADCSRA=135;ADCSRB=0;DIDR0=0;ADCSRA|=64;while(!(ADCSRA&16));return
ADC;
}
uint8_t AnalogRead8(uint8_t channel)
{
	ADMUX=96+channel;ADCSRA=135;ADCSRB=0;DIDR0=0;ADCSRA|=64;while(!(ADCSRA&16));return
ADC;
}
```

You may read the pots with 10 bit precision or 8 bit precision.
Channel is which pot to read, as described in the table above.

PWM:

PWM or Pulse Width Modulation, is a way of making an analog like waveform from a digital output. This works by varying the length of time the pulse is on vs the length of time the pulse is off (called the duty cycle). Doing this cycle at a high frequency (250 kHz in the case of the ATMega) makes the square wave (average out) at an analog voltage. Varying the duty cycle changes the analog voltage produced. Below are some examples of different duty cycles.



(From SparkFun)

PWM on the Atmega:

I am providing you with an AnalogWrite() function to do PWM.
chan is 0, 1, or 2, or Red, Green, or Blue LED
val is a value where 0 is off, and 255 is full on

```
void AnalogWrite(uint8_t chan, uint8_t val)
{
	static bool bRi=false,bGi=false,bBi=false;
	if(!bRi&&chan==0){DDRD|=1<<6;TCCR0A=131;TCCR0B=3;bRi=true;}
	if(!bGi&&chan==1){DDRB|=1<<1;TCCR1A=129;TCCR1B=11;bGi=true;}
	if(!bBi&&chan==2){DDRB|=1<<3;TCCR2A=131;TCCR2B=4;bBi=true;}
	if(chan==0)OCR0A=val;if (chan==1)OCR1A=val;if(chan==2)OCR2A=val;
}
```

YOU STILL NEED TO SET THE COMMON BIT (C3, both DDR and PORT)