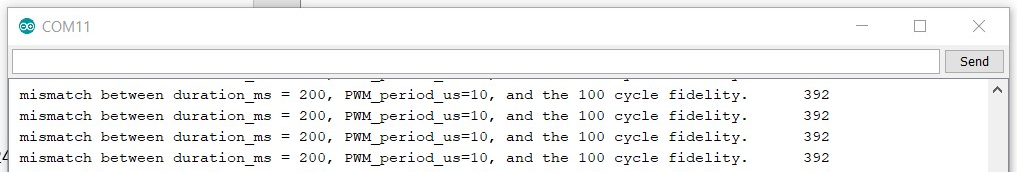
**Quiz 2**

Code for this quiz is given in the github repository for the course. The relevant file is **02\_24-100pwm.c** Direct link: <https://github.com/ntmoore/physics333_spring20/blob/master/02_24-100pwm.c>

The example file contains a function, *pwm\_100()*, that implements a PWM algorithm with 1 out of 100 fidelity (rather than 1 out of 1000 fidelity in the old *ntm\_pwm()* function – see line 50 specifically in the source code).

The code also has some basic error checking that’s enabled if line 3, “#define WARNINGS” is active (and not commented out). This line basically turns on the lines encapsulated by the “#ifdef WARNINGS” and “#endif” blocks (lines 29 and 43).

1. With **#define Warnings** active (uncommented) the code as written sends the following error message to the serial monitor (check it yourself!)



This is not how the error message should work, because (line 37),

(200\*1000)%100\*10 = 0, not 392.

Question: what’s going on? Hint, *long* might be useful.

**The Arduino Uno uses 16 bit integers, which gives them a range from -32,768 to 32,767**

**200\*1,000 = 200,000 which is much bigger than 32,767. The number is wrapping around to the negative end, and overflowing over and over to what eventually looks like 392.**

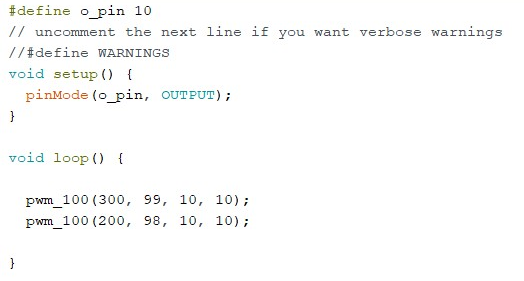
**Using a long instead extends this range to roughly -2 billion to positive 2 billion, which is plenty for this calculation.**

1. Please fix the conditional in line 37 so that the error is:
   1. active for a function call of pwm\_100(200, 99, 7, 10);
   2. not active for a function call of pwm\_100(300, 99, 6, 10);

if ((duration\_ms \* 1000) % (100 \* PWM\_period\_us) = 500) {

**This is a special case solution that doesn’t really fit what this if is supposed to do but it seems to me that both of these call would trip the error, assuming the numbers aren’t too big.**

1. Turn off the warnings by commenting out line 3, *#define WARNINGS*

Then, imagine that the code consists of: 

This file is given in the gitub repo if you need it, <https://github.com/ntmoore/physics333_spring20/blob/master/02_24_quiz_p3-4.c>

What will the signal look like if you average it out by feeding the signal to a big-enough capacitor and resistor, ie , in series to ground?

**The difference between 99 and 98 duty cycle is so minimal it likely wouldn’t even be seen but if you got really close, with a big enough capacitor, I imagine it would look like a sine wave around 5V with a very very small peak to peak, and a non symmetric cycle of 300 ms on the upper half and 200 ms on the lower half. By most accounts though it would just look like a regular DC signal just under 5Volts**

1. Assuming 5.0v is the Arduino’s high logic level, with what tolerance can your PWM specify analog output voltage values? Or, in other words, what average output voltage difference corresponds to the step between power=99 and power=98?

**Assuming a period of 100ms, allowing for exactly each of the 100 duty steps expected in the parameters of this function, it would be 5Volts / 101 steps = ~.05V. If your period was less than 100ms, multiple duty cycle values would correspond to the same power levels. If instead the duty was specified by a real number 0.0 through 1.0, your pwm resolution would depend heavily on the period of your wave.**

**See my Lab 02 Report for a more elaborate explanation of this.**