1.1 LED

Outcomes: After this first exercise you should be familiar with the solderless breadboards, LED's and datasheets.

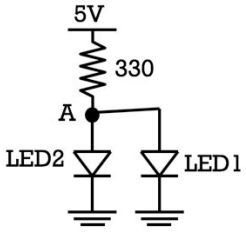
1.1.1 Choose an LED from your kit. What ranges of resistance will work well and why? [Hint: use the LED datasheets from Digikey.] Here are the LED's in your kit with Digikey.com part number and description:

- 160-1610-ND LTL2T3TBK5 5mm blue LED
- 160-1029-ND LTE-4208 EMITTER IR 5MM 940NM CLEAR
- 160-1128-ND LTL-4224 5mm red LED
- 160-1132-ND LTL-4238 5mm green LED

Calculate the smallest resistor that you can safely use with this LED in series with 5V power. **Submit a copy of the page from the datasheet you used highlighting the specification you used to determine this value along with the calculation you used to get it. The lecture on LED's will be relevant here.**

1.1.2 Use a breadboard and hook up the LED you chose in series with a resistor that has a value close to but higher than the value you calculated above. Connect it to 5V and ground and observe the brightness and verify that the LED does not get too hot, smoke or explode. Increase the resistance (using more resistors in series or resistors with larger values) and find what value of resistance will give a very dim but still visible light. **Submit the value of resistance you observed and calculate the current that gives this dim light.**

1.1.3 Take two LED's with different colors. Put them in parallel with a 330 ohm resistor as in the figure below. Measure the voltage at point A. **Submit what you observe: include which LED's you chose; which LED would you expect to be brighter based on the specifications; which LED is brighter; what values on the two different datasheets for each LED explains the voltage at point A?**



1.1.4 Take the same circuit, move LED1 to be in series with LED2. Measure the voltage at point A and the point between the two LED's. **Submit what you observe: include the values you read and the values you expect based on the datasheet.**

1.1.5 Pretend you are designing an LED powered light source. This could be a flashlight (tight beam) or a desk lamp (wide area). Choose a type of light you want for your lamp - Color, brightness, (wide or tight beam), cost, size. Go to www.digikey.com Find the brightest or cheapest or smallest LED that suits your needs. **Submit a write up that describes the reason you chose it, and how it is optimal (either**

bright cheap or small). Full points if the TA's cannot find a more optimal solution in less than 4 minutes.

1.2 TEENSY INTRODUCTION

Outcomes: After this section of the lab you should be familiar with modifying and compiling C code and downloading it to the Teensy. You should also become familiar with controlling the digital output of GPIO pins on a microcontroller by using registers.

We don't want you using Arduino with the Teensy. While there are advantages to Arduino, there are also benefits to understanding C and this is what we want you to get out of this. You will use the Arduino interface later in this class.

Go through the three "Getting started" steps on the following website installing the C compilers and downloading software. http://medesign.seas.upenn.edu/index.php/Guides/Teensy

Tips for Macs

When you get to the C compiler page (Step #3) on the pjrc website the link to the AVR Mac Pack is expired. You can use the following link: https://www.obdev.at/products/crosspack/download.html

Tips on navigating in a terminal/command window on Windows machines

> *cd "folder to go to"*,

this folder can be the full path *"C:/blah blah"* or referenced from the current folder > *cd subfolder* quotes are needed if there are spaces.

> *cd ..*	takes you one folder up.
> *cd /*	takes you back to the root drive.
<Tab>	can be used for auto complete.
> *ls*	lists the what is in the current directory.

Tips for Linux

Please email the teaching staff if you need to use Linux.

First Test

Now that you have a system setup you can start uploading code to your Teensy. Find `Blinky.zip` on Canvas, copy and unzip it in a directory you would like to use (**NOTE: this blinky program is different from the one you may find on the pjrc website**) . Inside you will find a makefile which contains the directions the computer will use for transferring your code into something that the teensy can use. Inside the SRC (source) folder you should put your `.c` files. Inside the INC (include) folder you should put your `.h` (header) files. Open a terminal or command prompt and navigate to that directory. Type "make" and press enter. If everything is working correctly it should create a `.hex` file and a folder with a couple of other files. If you get an error, this is pretty common and is why we supplied files that we know work so you can try to debug it in your setup and don't have to worry that it is the files you are

using. If you do have an error, check the internet to see if you can figure out the problem, if you get stuck you can post to piazza, be sure to include what OS you are using. Once you have generated the `.hex` file you should connect your teensy via the miniUSB cable to your computer and press the button on it (this starts it listening to your computer). Open the *teensy loader* you downloaded in "getting started step #2" open the `.hex` file (far left button on the loader window), load it (next button to the right), and reboot the teensy (one more button to the right, or by pressing the hardware button). Your teensy should now be blinking at 0.5 Hz.

Interfacing with Pins

Solder header pins to the Teensy board you have received so you may use it in the protoboard. Be sure to solder the pins while being held in the protoboard otherwise the pins may be the wrong angle and the board will be ruined. This short video on soldering the teensy will be helpful: https://www.youtube.com/watch?v=XYIqK4uzNYQ&t=78s

Digital Output

Now it is time to get your hands into the code. Select a pin other than PD6 and configure it to blink an LED you hook up on the protoboard. The GPIO page on medesign summarizes some of the pins on the ATMEGA32 http://medesign.seas.upenn.edu/index.php/Guides/MaEvArM-gpio Also, the pin description in Section 2.2 in the datasheet is another reference you can use to help you choose.

1.2.1 What registers are you changing and why?

1.2.2 Attach an LED to this pin with appropriate additional hardware. **Submit a drawing of your circuit and justify your additional hardware choice.**

1.2.3 Create a variable in your code that you can use to change the frequency the LED blinks. At what frequency does the LED start to appear to be continually on even though it is blinking very quickly? **Submit your (well commented) C code for this portion.**

1.2.4 Create a variable in your code so you can easily change the duty cycle (100% duty cycle is always on, 0% duty cycle is always off, 50% is half on half off). **Submit your (well commented) C code for this portion.**

1.2.5 How many LEDs can you turn on from one pin of the Teensy? **Submit a picture of your set up and a description of the issues limiting the number of LEDs.**

1.3 TIMERS

Outcomes: After this section of the lab you should be able to create PWM output on a GPIO pin using the internal timer and practice with an oscilloscope .

1.3.1 Use the timer of the ATMEGA32u4 to get an LED to blink at 200Hz. Verify the driving frequency on a scope connecting the ground and probe wires. **Submit a photo or screenshot of the oscilloscope screen showing the 200Hz signal and the code that generates this.**

1.3.2 Using the timer registers, adjust the system clock pre-scaler. Does the output change as you would expect, why or why not? What is the default system clock frequency? **Submit your answers.**

1.3.3 Use the PWM functions of the ATMEGA32u4 timer to do the same thing you did in 1.2.4. Explain which timer options you used for generating the PWM. Show that you can generate 0% and 100% duty cycle. **Submit your (well-commented) code, and your explanation. Also submit a short video of the system working.**

1.4 PRACTICE WITH LOOPS

Outcomes: After this section of the lab you should be familiar with writing `for` loops, subroutines and variables in C code.

1.4.1 Create code to make the external LED pulse. The pulse should start at 0 intensity, take 1 second to increase in intensity until it is full brightness, then 1 second to decrease in brightness and repeat.

Take that code and change it so that time to increase and the time to decrease is variable. **Submit a video of repeating asymmetric pulses (0.3 second to full intensity and 0.7 seconds to 0 intensity) along with well commented code and circuit diagram. Look on Canvas for sample videos of what this should look like.**

1.4.2 Use subroutines so that the pulsing of the LED with variable increasing and decreasing intensity is easy to call from another routine. Modify the code above so that the maximum intensity can also be changed. Create a routine that causes the LED to blink as if it were a heartbeat (e.g. lub dub). That is the LED percentage intensity *i* should follow the pattern below at time *t* seconds but smoothly interpolated intensities between each value:

t=0 i = 0
t=0.1 i = 100
t=0.5 i = 0
t=0.6 i = 50
t=1.0 i = 0
t=3.0 i = 0
t=3.1 i = 100
t=3.5 i = 0
t=3.6 i = 50
t=4.0 i = 0

Submit a video and your (well commented) C code for this portion. Look on Canvas for sample videos of what this should look like.

1.4.3 **Extra Credit:** Modify the above code so that the heartbeat stays at a constant frequency, but the maximum intensity slowly fades as if the heartbeat is getting weaker. Have it take 20 beats before it is reduced to 0 intensity. **Show a TA your LED blinking; answer his/her questions, and get signed off.**

(NOTE YOU MUST BE CHECKED OFF BY A TA BEFORE THIS PART WILL BE CONSIDERED FINISHED). Submit a short video and the commented code.

1.5 RETROSPECTIVE

This section is not graded. It is used to inform the teaching staff for future labs.

1.5.0 Briefly submit an estimate of the hours you spent on each section of this lab. Indicate any troubling spots (areas that could be improved in instruction) or other thoughts you may have about this lab or logistics for the course thus far.