The goal of Lab 1 is to provide you practice with circuits, LEDs, C programming, and the Itsy Bitsy Micro Controller. There are four tasks which work with different aspects of the course material. Please read the instructions carefully and as always start early!

Towne 195 is staffed at hours listed on the course calendar on the MEAM 5100 Canvas Page. If you run into issues or have questions please do not hesitate to ask a member of the teaching team or post on EdDiscussion.

Pick up Equipment:

- Pick up your ItsyBitsy microcontroller in Towne 195 during staffed office hours.

What is Due:

- A typed report answering the questions outlined below.
- When asked for directly, links to videos uploaded to YouTube within the report
- When helpful but not asked for directly, photos in the report and videos uploaded to YouTube are appreciated.
- A TA checkoff for section 1.4.3

Due Date: 02/04/2025

Submission is on Gradescope.

1.1 LED

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

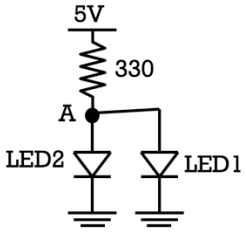
1.1.1 Choose an LED and a resistor from the ministore. What ranges of resistance values will work well and why? [*Hint*: use the LED datasheets from Digikey.] Use one of the three LED's listed below available in the GM lab with Digikey.com part number and description:

- 160-1133-ND 5mm yellow LED, LTL-4253
- 160-1128-ND 5mm red LED, LTL-4224
- 160-1709-ND LED 3MM YELLOW TRANSPARENT, LTL-1CHYE

Calculate the smallest value resistor that you can safely use with the chosen 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 any two LED's with two different colors different from the ones you used in 1.1.1. 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. Increase the power supply voltage if necessary to get them to turn on. Measure two voltages, one between the resistor and 1st LED 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 will be like a desk lamp (wide area (not focused beam). Go to www.digikey.com Find the brightest or cheapest or smallest that suits your needs for this lamp. **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 ItsyBitsy INTRODUCTION

Outcomes: After this section of the lab you should be familiar with modifying and compiling C code and downloading it to the ItsyBitsy. 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 ItsyBitsy. 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.

https://medesign.seas.upenn.edu/index.php/Guides/ItsyBitsy32u4

Go through the "Getting started" steps installing the C compilers and downloading software.

Tips on navigating in a terminal/command window

> *cd "folder to go to"*,

this folder can be the full path *"C:/blah blah"* (on Windows) 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 files in the current directory.

First Test

Now that you have a system setup you can start uploading code to your ItsyBitsy.

1. Find Blinky.zip on Canvas, copy and unzip it in a directory you would like to use. Inside the directory you will find a *makefile* which contains the directions the computer will use for transferring your code into something that the ItsyBitsy 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.
2. Open a terminal or command prompt and navigate to the directory you just unzipped.
3. Type "make" and press enter.
    a. If everything is working correctly it should create a .hex file and a folder with a couple of other files.
    b. Typically, in the debugging process, you will get an error during compilation. **This is why we supplied files that we know work so you can try to debug your setup and don't have to worry that it is the setup mechanism.**
    c. In the future, 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 Ed Discussion. Please be sure to include what OS you are using.
4. Once you have generated the .hex file you should connect your ItsyBitsy via the microUSB cable to your computer. Follow the "Uploading your code" section on https://medesign.seas.upenn.edu/index.php/Guides/Atmega32u4-starting.
5. If things have been setup correctly, you should see the onboard LED blinking 1 second on 1 second off.

Solder header pins to the ItsyBitsy 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 permanently misaligned ruining the board. (Note you may need to use some force to push the pins into the protoboard). This short video on soldering the teensy board will be helpful:

- https://www.youtube.com/watch?v=XYIqK4uzNYQ&t=78s

Use the same soldering principles for your ItsyBitsy board. In addition to the video, here are a few critical safety tips (for you and for the equipment)

- Burn Risk: Soldering irons are very hot. Keep them in the soldering stand. Keep the soldering area clean – clutter could cause the wires to get tangled and lead to burns.
- Health Risk: Lead Based solder (blue barrels) tends to work a bit better than lead free solder. But it has lead in it. Always wash your hands after soldering. Use the filter and exhale while soldering to avoid breathing fumes. Do not solder with your face right over your work, it is better to be off to the side a bit.

- DO NOT TURN THE HEAT ALL THE WAY UP ON THE SOLDERING IRON. Around 700-750°F (70-75 on the dial) works well. If you turn the heat up too high, it will oxidize the tip of the soldering iron so that it no longer works.

When in doubt please ask your classmates or TAs for help! YOUR TA WILL INSPECT YOUR SOLDERED PINS DURING THE TA CHECKOFF. You will be asked to fix any pins that do not look good.

Digital Output

Now it is time to get your hands into the code. Select a pin from Port B (i.e., labeled PB#) and configure it to blink an LED you hook up on the protoboard. The pinout page on medesign summarizes some of the pins on the ATmega32U4 https://medesign.seas.upenn.edu/index.php/Guides/Atmega32u4-pinout. 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 from the ministore to this pin with appropriate additional hardware. **Submit a drawing of your circuit using *circuitlab* and justify any component values you use.**

1.2.3 Using the `_delay_ms()` routine, create code that will blink an LED (you may modify the Blinky example to start with). **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). Verify with an oscilloscope that the duty cycle changes appropriately. Ensure your code handles all cases including the special cases of 0% where the LED is completely off and 100% where it is completely on. **Submit your (well commented) C code for this portion.**

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 reading and writing to registers.

1.3.1 Instead of delay functions, use a timer of the ATmega32U4 to get an LED to blink at 20Hz. (Note: don't use OCR registers which we will use later). Verify the driving frequency on a scope. **Submit a photo of the oscilloscope screen showing the 20Hz signal and the code that generates this.**

1.3.2 Using the timer registers, adjust the system clock pre-scaler. (Note: don't use OCR registers which we will use later). 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 (the WGM modes of timer registers) of the ATmega32U4 timer to do the same as 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 (from part 1.2) pulse. The pulse should start at 0 intensity, take 0.5 seconds to increase in intensity until it is full brightness, then 0.5 seconds 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 0.6 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, large then small intensity – ala 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 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 beat 20 times 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) YOUR TA WILL ALSO CHECK THAT YOUR SOLDER IS SUFFICIENT AND GIVE GUIDANCE ON HOW TO FIX IF NOT. 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.