

**Computer Systems Engineering Technology**

**CST 120 – Embedded C Programming**

|  |  |
| --- | --- |
| Lab 01 – Mr. Blinky - Hello World | Name: Chris Thomas\_ |
|  |  |
|  | |
| Possible Points: 100 | |
|  | |
| Due: Friday, April 1, 2022 : 5PM | |

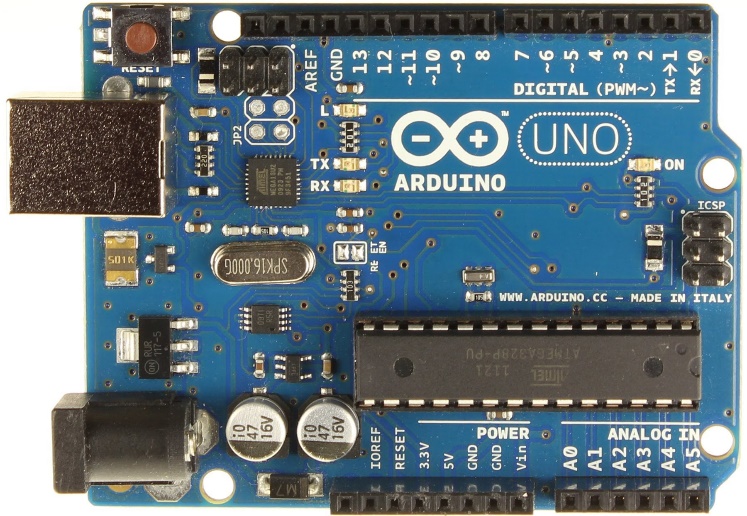
# Instructions

This is a three part lab. The first part will be a tutorial introducing you to the hardware and software we will be using this term. The second part will be an expansion on this introduction tutorial. When finished upload your code to Canvas. Make sure to do a Build->clean before creating a .zip file. The file should contain the entire project and be named first\_last\_lab#.zip

# Part 1

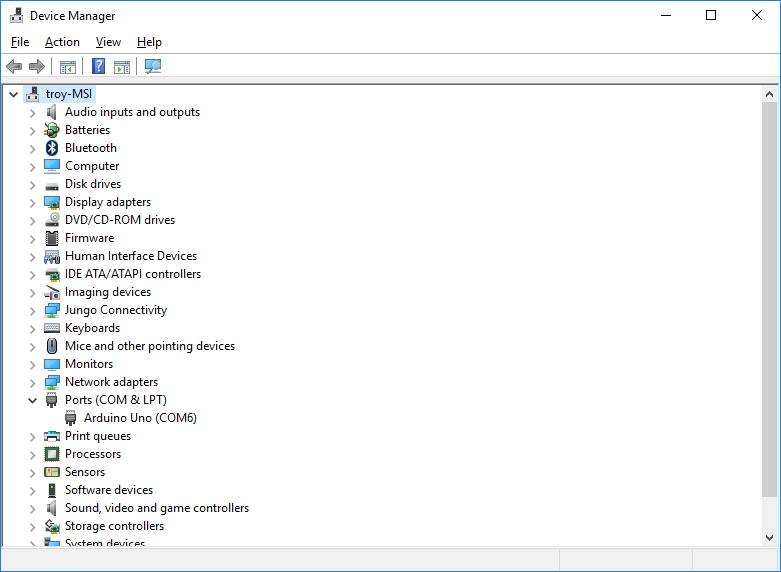
Download the zip file avrdude.zip from the Canvas Supplemental Material module. Unzip the file and place the unzipped folders in C:\avrdude\.

The next step will be to plug in your Arduino UNO board into the computer using the provided USB cable.

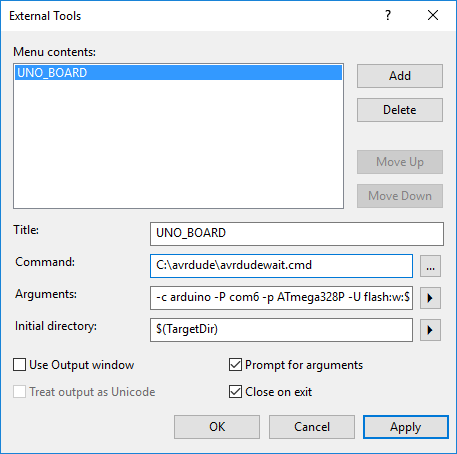


Once plugged in we need to go to windows device manager and make note of a few things. First we need to make sure the board was found by windows. The second thing we need to do is note what COM port the board is using. NOTE that in this example the UNO Board is on COM6. The following screen capture shows where we can find this information. Record the information below.

COM PORT COM4



Now start the development environment. Go to the start menu and find Microchip/Atmel Studio 7.0 and execute that program. Once the development IDE is open we need to do a couple more things. The first is to click on Tools->external tools. We need to add our own tool as defined below.



Command: C:\avrdude\avrdudewait.cmd

Arguments: -c arduino -P com6 -p ATmega328P -U flash:w:$(TargetName).hex:i

\*\*\*Note: make sure com port above matches the one you recorded earlier\*\*\*

Initial Directory: $(TargetDir)

Make sure the checkboxes are checked as in the given picture. Then click OK

Now we need to create a project for our UNO board.

File->New->Project then make sure to select a GCC C Executable Project. Give it a path on your Z drive and a name (Lab1p1). Under device family select ATmega. Then select ATmega328p for the device. Click OK.

Now we need to add some code to our project.

/\*

\* Lab1p1.c

\*

\* Created:

\* Author : \*/

#define *F\_CPU* 16000000UL

#include <avr/io.h>

#include <util/delay.h>

int main(void)

{

DDRB = (1 << DDB5);// pin 5 Port B selected as output

while(1) /\* Loop turns LED (connected to Pin 5 Port B) on for 1 second then off for 1 second \*/

{

PORTB = (1 << DDB5);

*\_delay\_ms*(1000);

PORTB = 0;

*\_delay\_ms*(1000);

}

}

Now build your code (Build -> Build Solution F7). Once built successfully we need to download to our board. Select Tools->UNO\_BOARD (The command we created earlier). Verify the com port is correct then click OK. The code will be downloaded into the board and start running. You should see the on board LED blink one second on, one second off.

Now let’s examine the first code above.

#define *F\_CPU* 16000000UL

The above line of code tells the compiler the clock rate of the processor. This is necessary to calculate the delay loops. If you comment out this line and rebuild and download your code what happens? Record your answer below:

What Happened? The light is blinking much faster than 1 second.

Take your code back to the original. Now let’s look at the next few lines.

#include <avr/io.h>

#include <util/delay.h>

int main(void)

These are just including the libraries we need and creating the entry point for our program. <avr/io.h> allows us to talk to the GPIO pin blinking the LED. <util/delay.h> provides the \_delay\_ms() function. This is a software delay loop.

while(1)

{

}

Now let’s look at what is happening here. We are saying while(1). 1 is TRUE so this loop will run forever. Isn’t this a bad thing? In embedded systems most of forever.

Now let’s take a look at a few more things.

DDRB = (1 << DDB5);

The above line sets the pin for the LED as an output port. This is necessary since a GPIO line can be either an input or an output. We need to set the pin as an output to make it turn the LED on and off.

**DDRB – The Port B Data Direction Register**



The DDB5 is just a #define DDB5 5, so this line shifts a 1 to the 5th bit of the DDRB register (DDB5 = (1 << 5)). A 1 sets that particular pin to be an output. To save power in embedded systems we always leave unused pins as inputs.

Now let’s look at the next section of code.

PORTB = (1 << DDB5);

*\_delay\_ms*(1000);

PORTB = 0;

*\_delay\_ms*(1000);

Let’s look at the first line PORTB = (1 << DDB5);. This line of code is doing the same sort of thing we did before except it is setting the output of the pin to be a 1. So we shift a 1 to bit 5 of the register setting that bit to a 1.

**PORTB – The Port B Data Register**



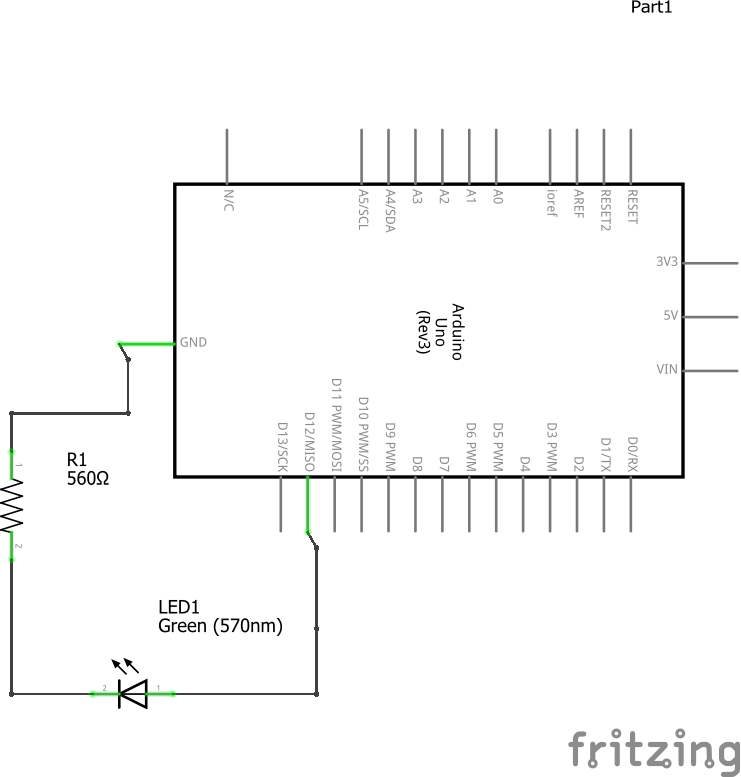
The line PORTB = 0; sets it back to a 0. We can get away with setting the whole register to a 0 since we are not using anything else and the default values are 0. The *\_delay\_ms*(1000); is just a software delay loop calculated based on the clock frequency of the controller.

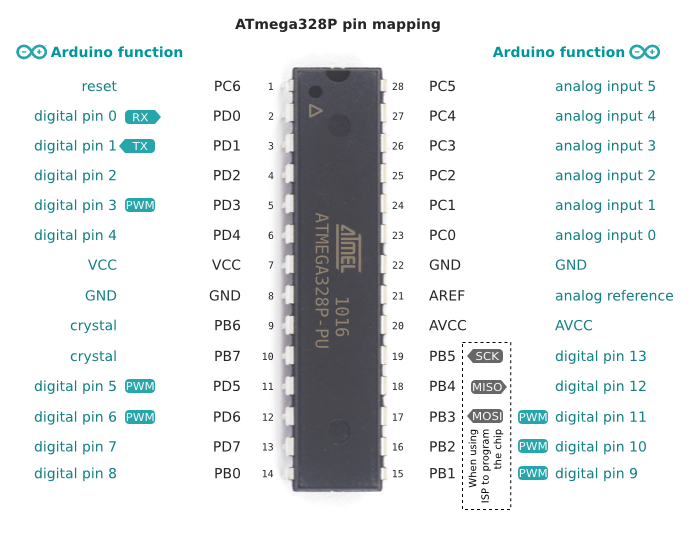
Play with changing the delays of the loop and become familiar with the environment. Once this is completed call Lab Instructor over for checkoff of part 1. Be prepared to answer some questions.

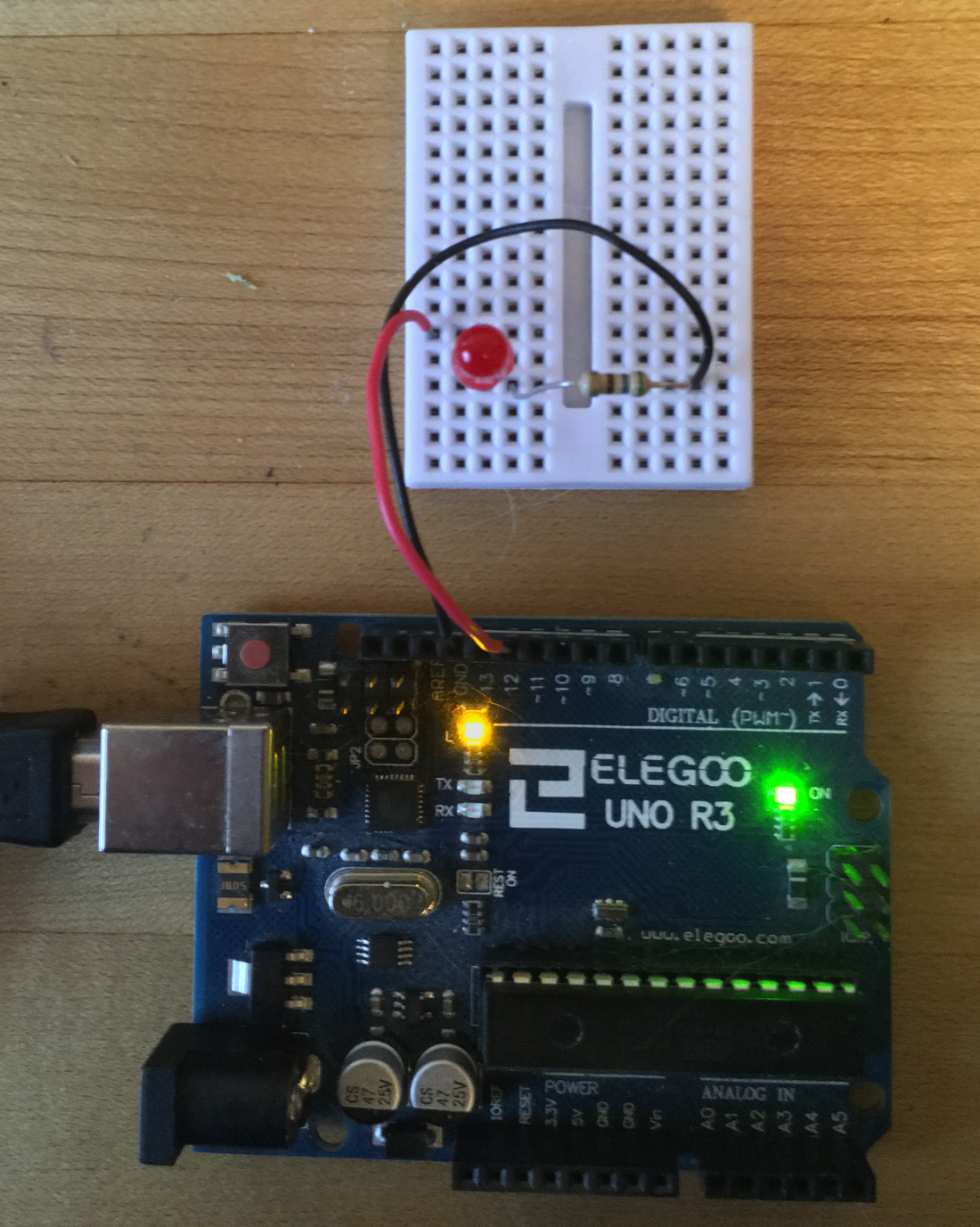
PART 1 CHECKOFF\_\_\_\_\_\_\_\_\_\_\_\_3.14159\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Part 2

For part 2 of this lab we are going to wire an additional circuit to the Arduino board and reproduce our blinky light using an external LED. The circuit we are going to produce is given in the following schematic. See last page of lab for an explanation of symbols. Looking at the schematic we can look and see which port the pin we are going to use is on. This is digital pin 12 (D12/MISO) on the UNO board. The pin mapping diagram on the next page shows us that it is on PB4.This is Port B pin 4.3

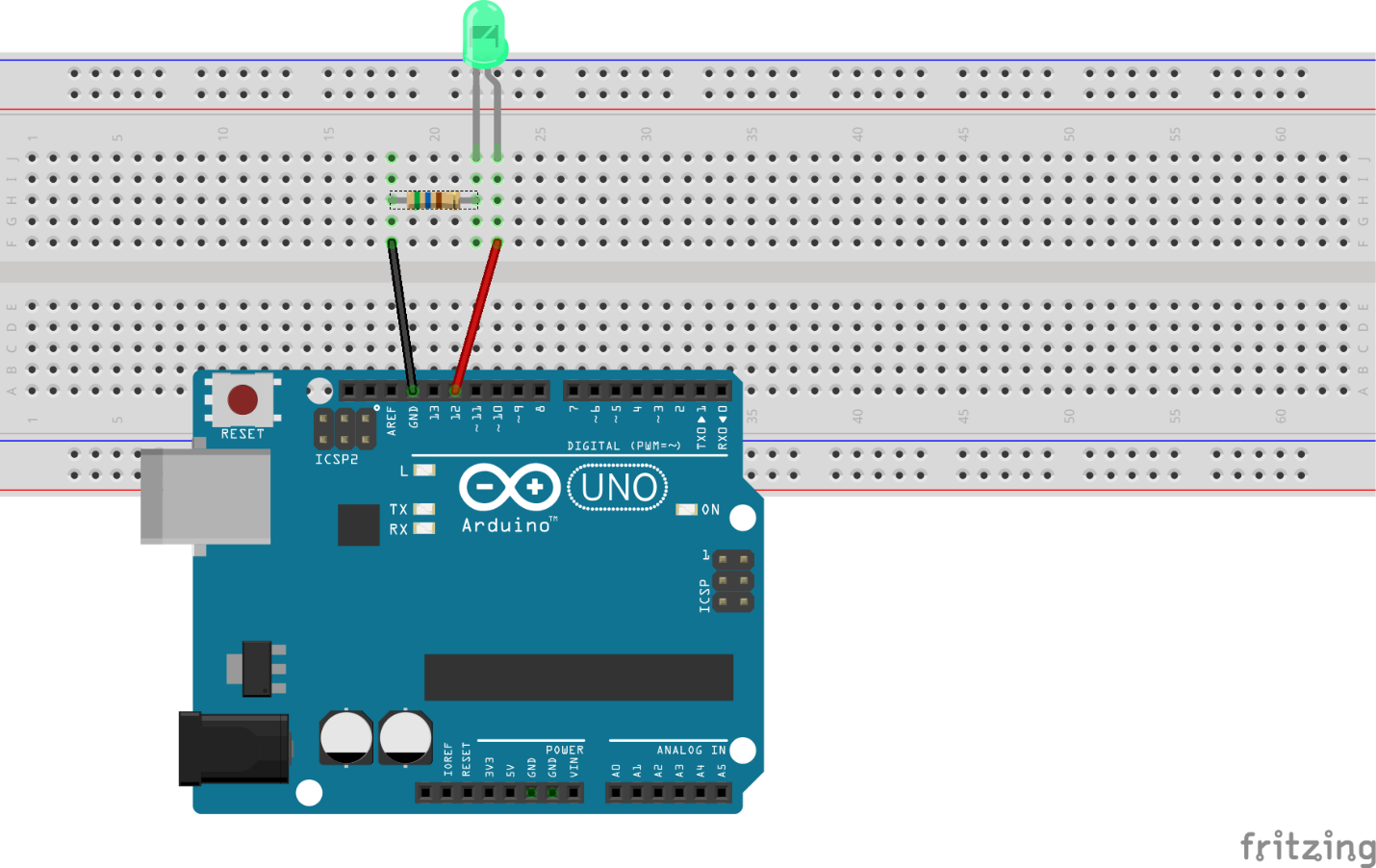






In the picture above the Red Wire is connected to Digital Output 12 from the UNO Board and to one side of the LED (the side with the longer lead). The Black Wire is connected to Ground (GND) of the UNO Board and one side of the resistor.

Get out your breadboard, wires, LEDs and resistors and wire up the circuit as shown below. **Make sure you power your board off by unplugging the USB cable before wiring the circuit.**



Have the lab instructor check your circuit before continuing.

Using what we did in Part 1 of the Lab, create a new project (lab1\_p2). Modify your code from Part 1 to blink the external LED that is now wired into our circuit. Demonstrate this to the lab instructor.

Lab Instructor Checkoff\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3.14159\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You now have two LEDs that you can blink under software control.

**Part 3**

Create a new project (lab1\_p3) for the final part of the lab. Modify your code from Lab 2 so that you are blinking the on board LED and the external LED, where the on board LED is blinking once a second and the external LED is blinking once every two seconds.

Part 3 Lab Instructor Checkoff\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3.14159\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Turning it in

Now you will need to upload your project (lab1\_p3) to the Canvas Assigments site. First you must do a Build->Clean on your project (this will remove any unnecessary intermediate files). To do this create a .zip file of your project and name it as follows. Firstname\_lastname\_lab#.zip. Upload this file to the Student Upload section on the Canvas site.

For this week we won’t worry about coding style. In the future your labs must follow the CSET Coding style or you will receive points deducted.

