From: Dr. Randy Hoover
To: CENG447L/547L
Subject: Lab 0: Setting Up Atmel Studio and working with GPIO

Setting up Atmel Studio

Start by following the link https://www.microchip.com/mplab/avr-support/atmel-studio-7 and downloading the latest release of AVR studio (Atmel Studio 7.0 sp1 (build 1931).

The download is fairly large ($\sim 700MB$). The installation was successful on a Windows 7 machine, which is what's recommended for this lab.

Download and Set Up WinAVR

Follow the link to download WinAVR http://winavr.sourceforge.net

Download the setup file and install it. Note down the directory where WinAVR was installed, as this will be used later on

Setting up The Compiler

After both pieces of software have been set up, launch Atmel Studio.

After launching, click on Tools>External Tools to create a new external tool.

Change the name to whatever you want, and in the Command field, navigate to your WinAVR directory>bin and choose avrdude.exe.

In the Arguments field, copy the same line in command and replace the .exe file extension with .conf, insert one blank space after that followed by the line (do not copy and paste – it may break things):

-F -v -p m328p -c arduino -P\\.\(COM PORT) -b 115200 -U
flash:w:"$(ProjectDir)\Debug\$(TargetName).hex":i

Replace (COM PORT) with the COM PORT your device will be using.

Blinking the on-board LED

Create a new project selecting "c/c++ executable" and "Atmega328P" as your device.

Important: Start your program by defining the CPU frequency: #define F_CPU 16000000

Import the AVR I/O library and delays library using the following #include files at the top of your project:

#include <avr/io.h>

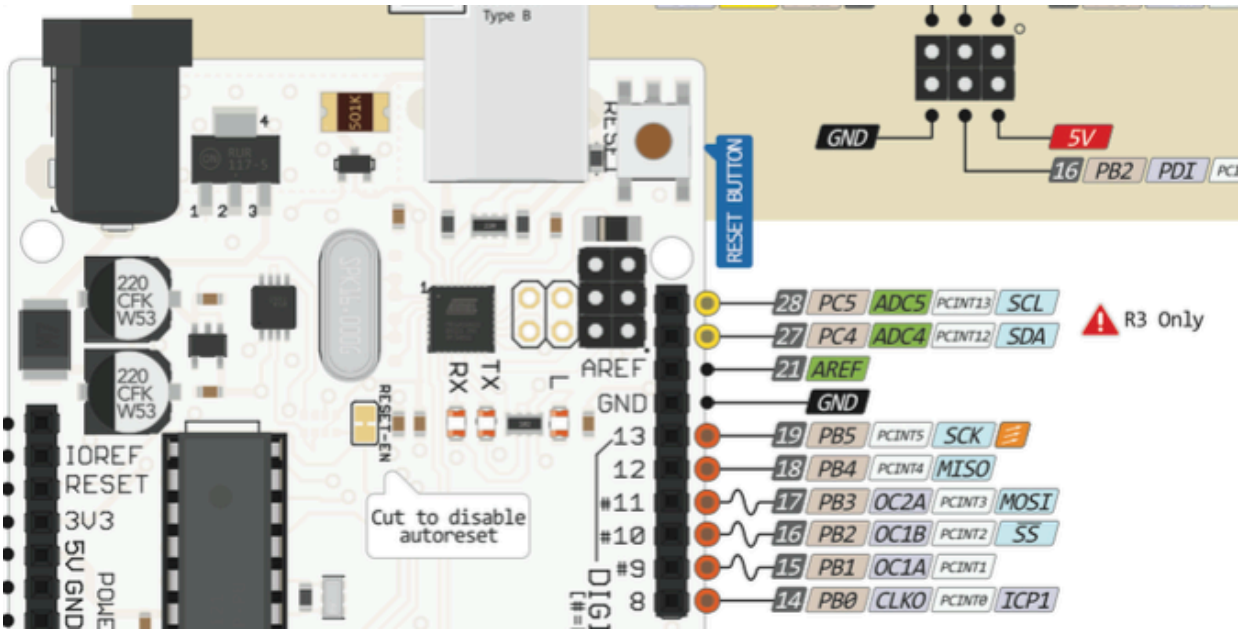
#include <util/delay.h>

You will need to use the function "_delay_ms(x)" to delay a specific amount of milliseconds.

Part 1:

The LED on board is wired to PORTB Pin as seen from the pinout below:

From: Dr. Randy Hoover
To: CENG447L/547L
Subject: Lab 0: Setting Up Atmel Studio and working with GPIO



To verify we can write, compile, and upload (program) code to our robot, we'll start by blinking the on-board LED at a rate of 1Hz (1 toggle/s).

You'll need to modify the Data Direction Register (DDRB) at the beginning of your main function to make the pins you are using output pins. Writing a logic 1 makes them output whereas writing a logic 0 makes them input pins.

WARNING: DO NOT GO NEAR PINS 6 AND 7 AS THEY ARE USED BY THE CRYSTAL OSCILLATOR OR PINS 0 AND 1 AS THEY ARE CONNECTED TO YOUR USART (USB).

From: Dr. Randy Hoover
To: CENG447L/547L
Subject: Lab 0: Setting Up Atmel Studio and working with GPIO
Here is some "sample" code to bling an LED on the Elegoo Robot Controller to get you started.

```
/*
 Program to make the blue LED blink on the Sparkfun Redboard.
 Modified by: Dr. Randy C. Hoover 1/25/17
 For use with ATMega 328p running at 16MHz
*/

#include <avr/io.h>
#include <util/delay.h>
int main(void)
{
    DDRB = 0xFF;            /* make the LED pin an output */
    for(;;){
        char i;
        for(i = 0; i < 10; i++){
            _delay_ms(60);  /* max is 262.14 ms / F_CPU in MHz */
        }
        PORTB ^= 0b00100000;    /* toggle the LED */
    }
    return 0;               /* never reached */
}
```

Deliverables

After successfully completing Part 1, video your LED blinking, capture some screen shots of AtmelStudio installed and working (if you use Linux or OSX capture screen shots of the environment you are using) and post it to your webpage.