Bryce Monaco
CPE 301-1001
Lab 9
April 12, 2017

Assignment Description:

Lab 09 had us modifying a prewritten program which took analog input and reported the value to a desktop program connected through the serial bus. We had to modify the program to function without using built in library functions so this meant removing any Arduino Analog functions as well as the C math library data types t_div and the calls associated to that. The program itself, in all versions, was meant to take input from a photoresistor and display the change in values on a desktop oscilloscope program which took in data through the serial bus.

Problems Encountered:

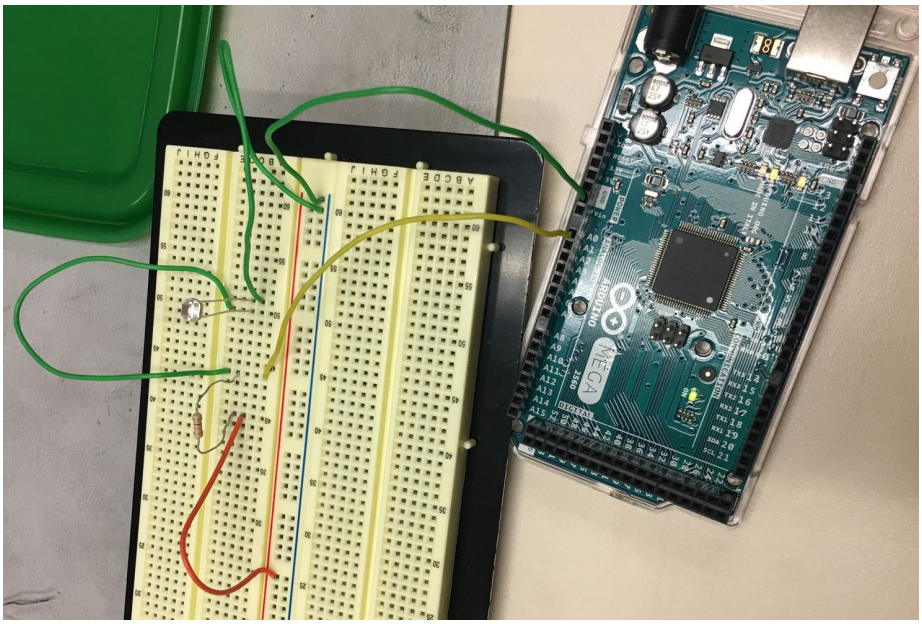
I had a lot of small issues in this lab that all came together to make the program not work when it should have. One of the biggest (and easiest to fix) was that the power supply was off when I had assumed it was on. Another issue was that a partner and I were sharing the power supply so we had the power supply input into my partner's power rails on his bread board and then I just ran jumps from his rails to my rails, this created a weird issue where my Arduino would sometimes report values that were being affected by his own photoresistor. Our last big issue was that both of us had the circuit wired up wrong, which was entirely our fault since we didn't realize that it was written on the board. All of these things came together to create strange bugs with the reported values.

Outside of hardware issues, I had a small bug in my program which was the result of a misconception. I had assumed that we only needed to initialize and start the ADC once and then it would just continuously report values but I was incorrect and later realized that I had to initialize and start it every single time that I wanted to read a new 10 bit value.

Lessons Learned:

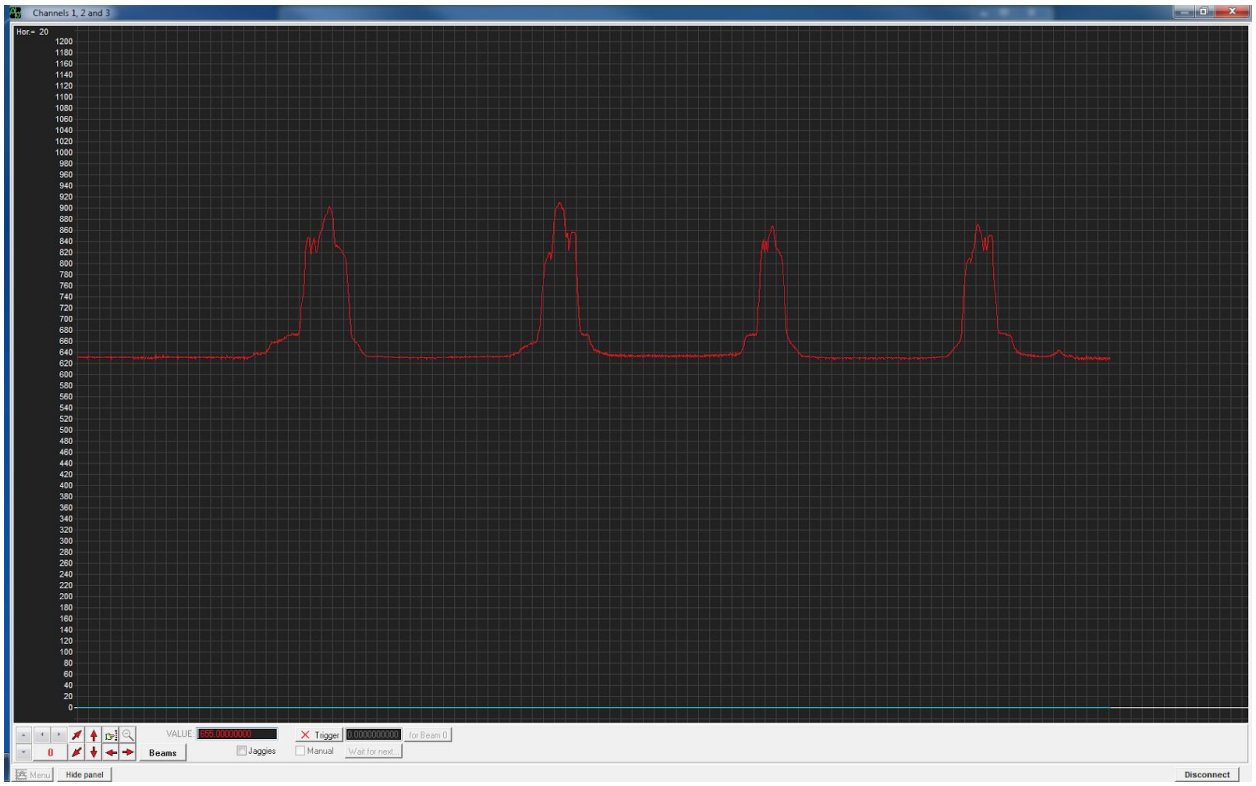
This lab reminded me of the importance to check the board for any relevant information, it also reminded me of the importance to fully read the relevant register and bit descriptions to fully understand how a system works.

Description of Completed Lab:



The completed physical setup, not shown is the USB cable to the desktop to allow for serial output.

As written in the Assignment Description section, the final goal of this lab was a system where we used a desktop oscilloscope program to monitor change in voltage caused by changes in light from a photoresistor using the serial bus to report values captured with the Arduino ADC.



Example output on the oscilloscope program showing a change in voltage as I moved my hand quickly over the photoresistor four times.

ADCSerialPrint Code:

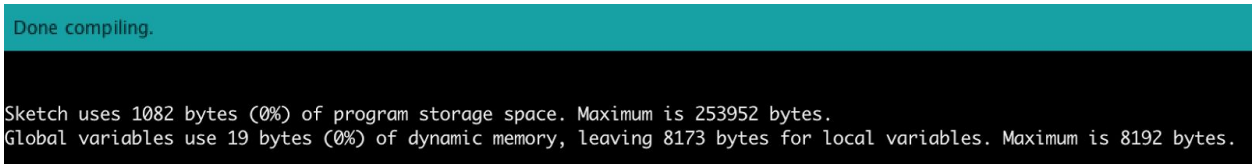
```
/*
    ArduinoPrintADC.ino

    Original Author: Seb Madgwick
    Modified By: Bryce Monaco
    Version 2.0 //2.0 since I consider 1.0 to be the original program

    Original File:
https://github.com/xioTechnologies/Serial-Oscilloscope/blob/master/ArduinoP
rintADC/ArduinoPrintADC.ino

 */

//Serial Pointers
volatile unsigned char *myUCSR0A = (unsigned char *) 0xC0;
volatile unsigned char *myUCSR0B = (unsigned char *) 0xC1;
volatile unsigned char *myUCSR0C = (unsigned char *) 0xC2;
volatile unsigned int  *myUBRR0  = (unsigned int *)  0xC4;
volatile unsigned char *myUDR0   = (unsigned char *) 0xC6;
```

```
//ADC Pointers
volatile unsigned char *myADCSRA = (unsigned char *) 0x7A;
volatile unsigned char *myADCSRB = (unsigned char *) 0x7B;
volatile unsigned char *myADMUX =  (unsigned char *) 0x7C;
volatile unsigned char *myDIDR0 =  (unsigned char *) 0x7E;
volatile unsigned int *myADCData = (unsigned int *) 0x78; //0x78 is the low
byte

void U0init (int rate);
unsigned char U0kbhit ();
unsigned char U0getchar ();
void U0putchar (unsigned char U0pdata);
void InitializeADC ();

void setup()
{
  // initialize the serial port on USART0:
  U0init(9600);

}

void loop()
{
    InitializeADC ();
    PrintInt(MyAnalogRead());

    U0putchar('\r');

}

// Fast int to ASCII conversion, used for serial output
void PrintInt(int i)
{
    static const char asciiDigits[10] = { '0', '1', '2', '3', '4', '5',
'6', '7', '8', '9' };
    int n;
    int print = 0;

    if(i < 0)
    {
        U0putchar('-');
        i = -i;

    }

    if(i >= 10000)
```

```
    {
        n = i/10000;
        U0putchar(asciiDigits[n]);
        i = i % 10000;
        print = 1;

    }

    if(i >= 1000 || print)
    {
        n = i / 1000;
        U0putchar(asciiDigits[n]);
        i = i % 1000;
        print = 1;

    }

    if(i >= 100 || print)
    {
        n = i / 100;
        U0putchar(asciiDigits[n]);
        i = i % 100;
        print = 1;

    }

    if(i >= 10 || print)
    {
        n = i / 10;
        U0putchar(asciiDigits[n]);
        i = i % 10;

    }

    U0putchar(asciiDigits[i]);

    return;

}

//
// function to initialize USART0 to "int" Baud, 8 data bits,
// no parity, and one stop bit. Assume FCPU = 16MHz.
//
void U0init(int U0baud)
{
  unsigned long FCPU = 16000000;
```

```
  unsigned int tbaud;
  tbaud = (FCPU / 16 / U0baud - 1);
  *myUCSR0A = 0x20;
  *myUCSR0B = 0x18;
  *myUCSR0C = 0x06;
  *myUBRR0  = tbaud;

}

//
// Wait for USART0 TBE to be set then write character to
// transmit buffer
//
void U0putchar(unsigned char U0pdata)
{
  while ((*myUCSR0A & 0x20) != 0x20); //Wait for data register to be empty

  *myUDR0 = U0pdata;

  return;

}

//Has the ADC record a value
void InitializeADC ()
{
  *myADCSRA = (unsigned char) 0x84;
  *myADCSRB = (unsigned char) 0x00;
  *myADMUX = (unsigned char) 0x40;
  *myDIDR0 = (unsigned char) 0x00;

  *myADCSRA |= 0x40; //Start conversion

  while ((*myADCSRA & 0x40) == 0x40); //Wait for conversion to finish

  return;

}

unsigned int MyAnalogRead ()
{
  return *myADCData;

}
```

ADCSerialPrint Compile Lines:


Done compiling.
Sketch uses 1082 bytes (0%) of program storage space. Maximum is 253952 bytes.
Global variables use 19 bytes (0%) of dynamic memory, leaving 8173 bytes for local variables. Maximum is 8192 bytes.