ECED3204 – Lab #8

STUDENT NAME(s): _____ .

STUDENT NUMBER(s): B00 _____ .

Pre-Lab Information

It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab, as you will be required to write some simple C code during this lab!

Objective

- Use the Analog to Digital Converter (ADC) to read a potentiometer.

Required Materials

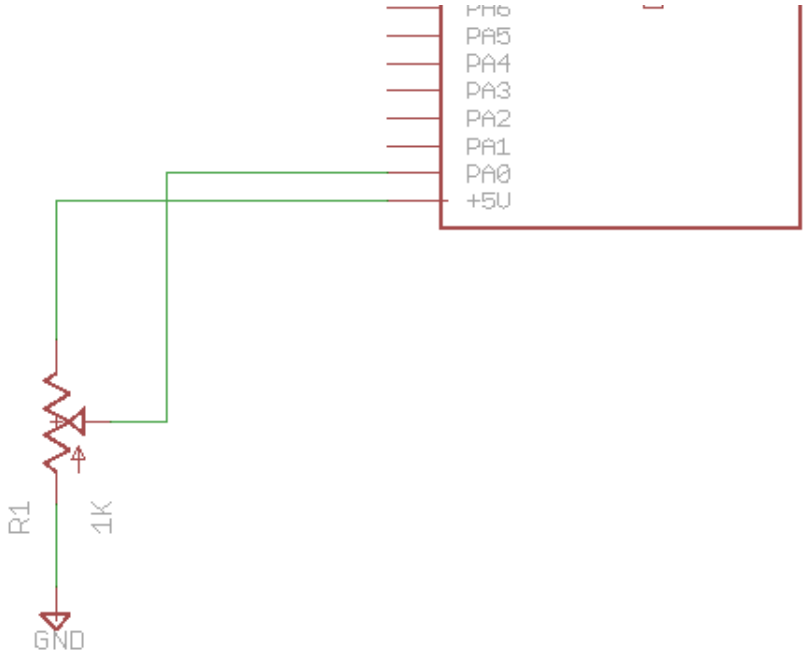
- Microprocessor Module with Programmer
- Breadboard
- USB Cable
- Power Supply
- Computer with Atmel Studio 6.2 and Programmer Utility installed
- Variable Resistor (potentiometer)

Background

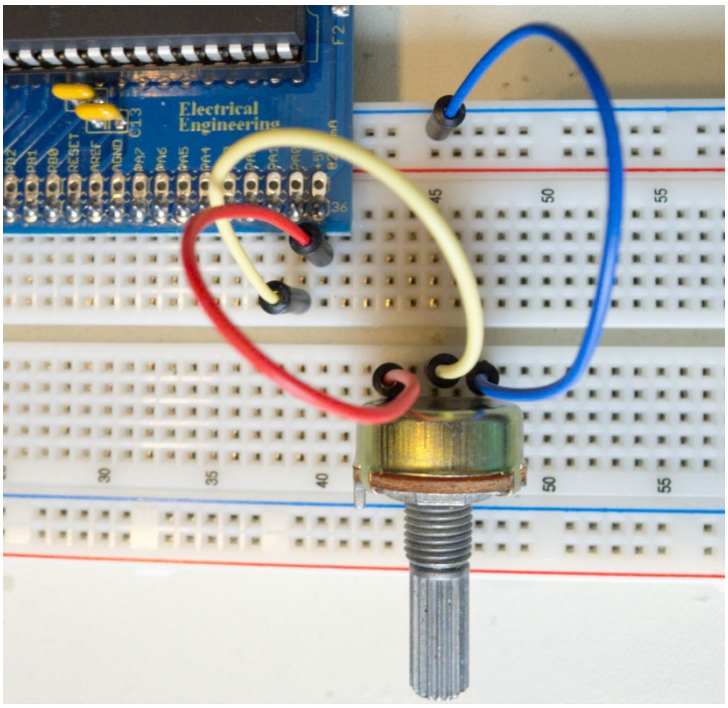
See the course textbook for information –the Analog to Digital Converter chapter details many of the required conversions.

Procedure

1. Build the following circuit:



Which might look like this, note your variable resistor (potentiometer or 'pot') might look different:



2. Start a new C/C++ project (see Lab #1 for details), write the following code into it, see the course textbook for details:

```
#include <stdio.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#define F_CPU 14745600UL
#include <util/delay.h>

static int uart_putchar(char c, FILE *stream);
static int uart_getchar(FILE *stream);
FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

static int uart_putchar(char c, FILE *stream)
{
   loop_until_bit_is_set(UCSR0A, UDRE0);
   UDR0 = c;
   return 0;
}

static int uart_getchar(FILE *stream)
{
   loop_until_bit_is_set(UCSR0A, RXC0); /* Wait until data exists. */
   return UDR0;
}

void init_uart(void)
{
   UCSR0B = (1<<RXEN0) | (1<<TXEN0);
   UBRR0 = 7;
   stdout = &mystdout;
   stdin = &mystdin;
}


int main(void)
{
   init_uart();
   printf_P(PSTR("System Booted, built %s on %s\n"), __TIME__, __DATE__);

   ADCSRA = 0x87;
   ADMUX = 0x40;
   ADCSRB = 0;
   DIDR0 = 0xFE;
   DIDR1 = 0xFF;

   int mv_reading;

   while(1){
          ADCSRA |= 1<<ADSC;
          loop_until_bit_is_set(ADCSRA, ADIF);
          ADCSRA |= 1<<ADIF;

          mv_reading = ((float)ADC / 1024.0) * 5.0 * 1000.0;

          printf("Reading = %d mV\n", mv_reading);
          _delay_ms(500);
   }
}
```

3. Confirm that as you move the potentiometer, the printed voltage varies. Check the code size – this code is using floating point so it will be very large! Consider the following code instead:

```
mv_reading = (ADC / 1024) * 5 * 1000;
```

Implement the code – what does this not work as-is?

4. As a fix to the previous code, the following should avoid the problems, but it still won't work as expected:

```
mv_reading = ((ADC * 1000 * 5) / 1024;
```

5. Fix the above code such that it works – as a hint you will have to cast many variables to (uint32_t) in order to fix the problem.

6. What happens to code size for your working code that does not use floating point?

Lab Questions

The procedure had several questions, be sure to answer:

1) Why the code in Step 3 does not work.

2) Why the code as written in Step 4 does not work.

3) The fixed version of the code (Step 5)

4) The size difference in FLASH (Program Memory) and SRAM (Data Memory) comparing the code from Step 2 and Step 5.