**Jacqueline Luo (26938949)**

*I have also included the cpp files, for easier viewing of the code.*

# Question 1. Teensy PWM

**1a. Write a program to control the frequency and intensity of flashing of two LEDs by commands received from the console. For this question, use bare metal to generate the PWM according to the instructions in the ATMEGA 328 data sheet (manual) using timer1, output compare, and, if you wish, toggle mode on the OC1A and OC1B pins) You should show clearly the setting of the appropriate register values in your program. Control the frequency from 100 msec to 2 seconds.**

#include<avr/io.h>

#define F\_CPU 1000000UL //1 MHz

#include <util/delay.h>

#include <string.h>

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define USART\_BAUDRATE 4800

#define BAUD\_PRESCALE (((F\_CPU/(USART\_BAUDRATE\*16UL)))-1)

bool lightOn = false;

int intensityA = 0;

int intensityB = 0;

//Initialization for LEDs

void initLED()

{

DDRD |= (1 << PD5) | (1 << PD6); //set PD5 and PD6 to output

}

//Initialization for PWM

void initPWM()

{

TCCR0A |= (1 << COM0A1) | (1 << COM0B1) | (1 << WGM01) | (1<< WGM00); //8 bit clock

TCCR0B |= (1 << CS00); //No prescaler, start the clock

TCNT0 = 0; //start timer

OCR0A = OCR0B = 0;

}

//initialization for analog read's clock

void initBlinkTimer()

{

TCCR1A = 0;

TCCR1B = 0;

TCCR1B |= (1 << WGM12)|(1 << CS12); // set up timer with prescaler = 256 and CTC mode

TCNT1H = 0;

TCNT1L = 0; // initialize counter

OCR1A = 0;

TIMSK1 = (1 << OCIE1A); //enable interrupt for compare

}

void stopTimer()

{

TIMSK1 = (0 << OCIE1A);

OCR1A = OCR1B = 0;

OCR0A = OCR0B = 0;

TCNT0 = TCNT1H = TCNT1L = 0;

}

//Initialization for serial communication

void initUsart()

{

UCSR0B |= (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0); //Enable RX, TX

UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01); //This sets data size to 8 bit

UBRR0H = (BAUD\_PRESCALE >> 8);

UBRR0L = BAUD\_PRESCALE;

}

//Returns a char that was typed/ received

char receiveByte()

{

while( !( UCSR0A & ( 1 << RXC0 ) ) ){};

return UDR0;

}

//Returns a char that was transmitted

void transmitByte(char data)

{

while( !( UCSR0A & ( 1 << UDRE0 ) ) ){};

UDR0 = data;

}

//Method to print text, it is done by transmitting each char of a string

void printStr(char const\*s)

{

while(\*s)

{

transmitByte(\*s);

s++;

}

}

//Method that processes what user has typed and returns it as an int

int getInput()

{

int bufferIndex = 0; //tracks the current index

char givenByte; //whatever the user typed

char myBuffer[32]; //the string that we are building

int input; //the number that will be returned

while(1)

{

givenByte = receiveByte(); //call method receive byte to get the typed byte

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0) //if the buffer is empty

continue; //do nothing and restart the loop

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(isdigit(givenByte)) //if a number was sent

{

myBuffer[bufferIndex] = givenByte;//add it to the buffer

bufferIndex++; //increment the index

}

else //if something other than a number or enter was sent

{

printStr("Please enter a number.\n\r"); //message the user and ask them to enter a number

}

}

}

//similar to get input, but just for the LED

int getLED()

{

int bufferIndex = 0;

char givenByte;

char myBuffer[32];

int input;

while(1)

{

givenByte = receiveByte();

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0)

continue;

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(givenByte == '1' | givenByte == '2') //if 1 or 2 was typed

{

if(bufferIndex < 1) //if there is nothing in the buffer

{

myBuffer[bufferIndex] = givenByte;

bufferIndex++;

}

else//if there is already something in the buffer

{

printStr("Press Enter!\n\r");//it should already be a 1 or 2, so we ask the user to press enter

}

}

else

{

printStr("Please enter 1 or 2.\n\r");//ask user to enter 1 or 2

}

}

}

//method that blinks the LEDs according to how the user specified.

void blink(int dca, int dcb)

{

if(lightOn == false)

{

OCR0A = dca; //LED1 will get duty cycle a

OCR0B = dcb; //LED2 will get duty cycle b

lightOn = true;

}

else

{

OCR0A = 0; //turn off both LEDs

OCR0B = 0;

lightOn = false;

}

}

int main(void)

{

int currentLED = 0;

int givenDelay = 0;

//initializers

initLED();

initUsart();

initBlinkTimer();

initPWM();

//set all the outputs to high

PORTD |= (1 << PD5) | (1 << PD6);

printStr("Hello, welcome to LED blinker!\n\r");

printStr("Please press 1 or 2 to pick an LED.\n\r");

currentLED = getLED();

printStr("Please enter a number from between 0 to 255 for the intensity.\n\r");

if(currentLED == 1)

intensityA = getInput();

else if (currentLED == 2)

intensityB = getInput();

printStr("Please enter a number from between 0 to 255 for the intensity of the other LED.\n\r");

if(currentLED == 1)

intensityB = getInput();

else if (currentLED == 2)

intensityA = getInput();

printStr("Please enter a number from between 100 to 2000 for a delay in milliseconds.\n\r");

givenDelay = getInput();

OCR1A = givenDelay \* 4.2; //4200 is roughly 1 sec, I multiplied by 4200 and then divided by 1000, which gives 4.2

sei();

TCNT1H = 0;

TCNT1L = 0;

while(1) //main loop

{

}

return 0;

}

ISR(TIMER1\_COMPA\_vect) // timer1 compare interrupt

{

blink(intensityA,intensityB);

}

**1b. Indicate clearly which pins are connected to the LEDs, and any hardware you may need, State clearly the design of the codes you send for selection of the LED, the intensity, and the frequency of flashing.**

The pins PD5 and PD6 are connected to the LEDs because they correspond to OC0A and OC0B, the outputs of timer0. The hardware I used was the Atmega328 microcontroller, a bunch of wires, a resistor for the Atmega328, a capacitor for the Atmega, an LED pack, a resistor for the LED pack and a Sparkfun FTDI basic.

**This is the long explanation for my program, the short version is in part 1c.**

I have a method called getInput that deals with what the user types/ what is received by the Atmega. getInput is a continuous loop that checks if a byte that was received was an 0x0D (hexadecimal for enter), a digit or something else. If it is a digit, it will get put into a buffer. This is because in this program, only numbers should be entered. If what was entered was something else, the method will ask the user to enter a number. If what is received is an enter, the method will then check if the buffer is empty or not. If it is, the method shouldn’t do anything. If there are numbers in the buffer, the method will add ‘\n’ which is newline, ‘\r’ which is carriage return and a 0 (not the digit but ASCII 0) which turns my array of chars into a null-terminated string. This string will then be converted into an int which will be transmitted back to the console, so that the user can see what they typed, and the int will also be returned at the end of the method.

At the beginning of my program, after the initializations, it transmits a message to the user, telling them to type a number, 1 or 2, in order to select an LED to control. A variable called currentLED will contain the getInput, the number that the user has typed. After, the program will transmit a message, asking them to select a number from between 0 to 255 to set the intensity of the LED that they picked. My program will check what currentLED is. If it is 1, variable intensityA (the duty cycle of OC0A) will be set to what the user has inputted. If currentLED is 2, variable intensityB (duty cycle of OC0B) will be set to what the user has inputted. After that is done, the program will transmit a message asking them to enter another number from between 0 to 255 to set the intensity of the other LED. Now, the variable intensityA or intensityB that wasn’t set in the previous part will be set. Finally, the program will transmit a message asking for a number from between 100 and 2000 as this will set the delay in milliseconds. Variable givenDelay will contain the entered number. After taking in all these inputs, the program progresses to the endless loop in main. The blinking is done by an interrupt, generated when timer1’s counter is equal to what was set in OCR1A, which in this case is the given delay multiplied by 4.2.

**1c. Explain your code briefly, within code documentation and a few external lines**

In short, the program will ask the user to pick an LED, set its intensity, set the intensity of the second LED and then set the delay (which will be applied to both LEDs). The intensities of both LEDs will be passed into a method which sets OCR0A and OCR0B to their respective intensities. OCR0A is LED1 and OCR0B is LED2. The intensity is controlled by timer0’s pwm value, the frequency of the blinking is controlled by using CTC and interrupts for when the timer hits OCR1A. When it reaches that, the interrupt vector is called and it turns the LED on or off.

**1d. Describe the software support and the procedures you need to execute. You do not need to demonstrate. This is entirely a paper exercise.**

We need to use the command line, avr-gcc and avrdude in order to compile the code and flash it onto the Atmega328. First we need to have a c or cpp file and make it into an .o file. After that, we turn the .o into a .hex file. Finally, we flash the .hex file, using avrdude. This is all done using Windows’ cmd.exe

# Question 2. Teensy Analog Read

**2a. In the same program, read an analog value from an appropriate pin at 1 second intervals, and send this value to the console. Use bare metal code. Read using the ADMUX, ADCSRA, ADCSRB, and ADC Data registers.**

#include<avr/io.h>

#define F\_CPU 1000000UL //1 MHz

#include <util/delay.h>

#include <string.h>

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define USART\_BAUDRATE 4800

#define BAUD\_PRESCALE (((F\_CPU/(USART\_BAUDRATE\*16UL)))-1)

bool lightOn = false;

int intensityA = 0;

int intensityB = 0;

int ADCCounter = 0;

int maxCount = 0;

//Initialization for LEDs

void initLED()

{

DDRD |= (1 << PD5) | (1 << PD6); //set PD5 and PD6 to output

}

//Initialization for the ADC

void initADC()

{

ADMUX |= (1 << REFS0); //Voltage reference is AVcc, multiplexer is set to read pin PC0

ADCSRA |= (1 << ADEN); //This enables the ADC

}

//Method which reads from an analog source and returns an int

int readADC()

{

ADCSRA |= (1 << ADSC); //Start read

while(!(ADCSRA & (1<<ADIF))); //Do nothing while data is being read

ADCSRA |= (1 << ADIF); //reset interrupt flag

return (ADC); //return the read value

}

//Initialization for PWM

void initPWM()

{

TCCR0A |= (1 << COM0A1) | (1 << COM0B1) | (1 << WGM01) | (1<< WGM00); //8 bit clock

TCCR0B |= (1 << CS00); //No prescaler, start the clock

TCNT0 = 0; //start timer

OCR0A = OCR0B = 0;

}

//initialization for analog read's clock

void initBlinkTimer()

{

TCCR1A = 0;

TCCR1B = 0;

TCCR1B |= (1 << WGM12)|(1 << CS12); // set up timer with prescaler = 256 and CTC mode

TCNT1H = 0;

TCNT1L = 0; // initialize counter

OCR1A = 0;

TIMSK1 = (1 << OCIE1A); //enable interrupt for compare

}

void initADCTimer()

{

TCCR2A = 0;

TCCR2B = 0;

TCCR2A |= (1 << WGM21); //set timer to CTC

TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20);//set prescaler to 1024

TCNT2 = 0;

OCR2A = 98; //this is roughly 0.1 seconds

TIMSK2 = (1 << OCIE2A);

}

void stopTimer()

{

TIMSK1 = (0 << OCIE1A);

OCR1A = OCR1B = 0;

OCR0A = OCR0B = 0;

TCNT0 = TCNT1H = TCNT1L = 0;

}

//Initialization for serial communication

void initUsart()

{

UCSR0B |= (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0); //Enable RX, TX

UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01); //This sets data size to 8 bit

UBRR0H = (BAUD\_PRESCALE >> 8);

UBRR0L = BAUD\_PRESCALE;

}

//Returns a char that was typed/ received

char receiveByte()

{

while( !( UCSR0A & ( 1 << RXC0 ) ) ){};

return UDR0;

}

//Returns a char that was transmitted

void transmitByte(char data)

{

while( !( UCSR0A & ( 1 << UDRE0 ) ) ){};

UDR0 = data;

}

//Method to print text, it is done by transmitting each char of a string

void printStr(char const\*s)

{

while(\*s)

{

transmitByte(\*s);

s++;

}

}

//Method that processes what user has typed and returns it as an int

int getInput()

{

int bufferIndex = 0; //tracks the current index

char givenByte; //whatever the user typed

char myBuffer[128]; //the string that we are building

int input; //the number that will be returned

while(1)

{

givenByte = receiveByte(); //call method receive byte to get the typed byte

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0) //if the buffer is empty

continue; //do nothing and restart the loop

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(isdigit(givenByte)) //if a number was sent

{

myBuffer[bufferIndex] = givenByte;//add it to the buffer

bufferIndex++; //increment the index

}

else //if something other than a number or enter was sent

{

printStr("Please enter a number.\n\r"); //message the user and ask them to enter a number

}

}

}

//similar to get input, but just for the LED

int getLED()

{

int bufferIndex = 0;

char givenByte;

char myBuffer[128];

int input;

while(1)

{

givenByte = receiveByte();

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0)

continue;

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(givenByte == '1' | givenByte == '2') //if 1 or 2 was typed

{

if(bufferIndex < 1) //if there is nothing in the buffer

{

myBuffer[bufferIndex] = givenByte;

bufferIndex++;

}

else//if there is already something in the buffer

{

printStr("Press Enter!\n\r");//it should already be a 1 or 2, so we ask the user to press enter

}

}

else

{

printStr("Please enter 1 or 2.\n\r");//ask user to enter 1 or 2

}

}

}

//method that blinks the LEDs according to how the user specified.

void blink(int dca, int dcb)

{

if(lightOn == false)

{

OCR0A = dca; //LED1 will get duty cycle a

OCR0B = dcb; //LED2 will get duty cycle b

lightOn = true;

}

else

{

OCR0A = 0; //turn off both LEDs

OCR0B = 0;

lightOn = false;

}

}

void getReads()

{

char tempbuff[128];

int tempIndex = 0;

int holder = 0;

holder = readADC();

sprintf(tempbuff, "%d\n\r", holder);

printStr(tempbuff);

}

int main(void)

{

int currentLED = 0;

int givenDelay = 0;

//initializers

initADC();

initADCTimer();

initLED();

initUsart();

initBlinkTimer();

initPWM();

//set all the outputs to high

PORTD |= (1 << PD5) | (1 << PD6);

printStr("Hello, welcome to LED blinker!\n\r");

printStr("Please press 1 or 2 to pick an LED.\n\r");

currentLED = getLED();

printStr("Please enter a number from between 0 to 255 for the intensity.\n\r");

if(currentLED == 1)

intensityA = getInput();

else if (currentLED == 2)

intensityB = getInput();

printStr("Please enter a number from between 0 to 255 for the intensity of the other LED.\n\r");

if(currentLED == 1)

intensityB = getInput();

else if (currentLED == 2)

intensityA = getInput();

printStr("Please enter a number from between 100 to 2000 for a delay in milliseconds.\n\r");

givenDelay = getInput();

OCR1A = givenDelay \* 4.2; //4200 is roughly 1 sec, I multiplied by 4200 and then divided by 1000, which gives 4.2

maxCount = 10;

sei();

TCNT1H = 0;

TCNT1L = 0;

while(1) //main loop

{

}

return 0;

}

ISR(TIMER1\_COMPA\_vect) // timer1 compare interrupt

{

cli();

blink(intensityA,intensityB);

sei();

}

ISR(TIMER2\_COMPA\_vect) // timer2 compare interrupt

{

ADCCounter++;

if(ADCCounter >= maxCount)

{

getReads();

ADCCounter = 0;

}

}

**2b. Explain your code briefly, within code documentation and a few external .lines**

In this part, I used a third timer so that there wouldn’t be issues with PWM generation and reading ADC values and the other interrupt counter. The user input all the information of part 1, and when all that is done and the LED is blinking, the SHARP starts reading. Whenever the timer reaches the set compare value, an interrupt will be called and this will cause a read to happen. Because timer2 is an 8bit timer, I set it so that a compare happens every 0.1 seconds. After, there is a counter for how many times the compare happened. I use this to calculate 1 or more seconds.

**2c. Vary the interval from your console**

#include<avr/io.h>

#define F\_CPU 1000000UL //1 MHz

#include <util/delay.h>

#include <string.h>

#include <avr/interrupt.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#define USART\_BAUDRATE 4800

#define BAUD\_PRESCALE (((F\_CPU/(USART\_BAUDRATE\*16UL)))-1)

bool lightOn = false;

int intensityA = 0;

int intensityB = 0;

int ADCCounter = 0;

int maxCount = 0;

//Initialization for LEDs

void initLED()

{

DDRD |= (1 << PD5) | (1 << PD6); //set PD5 and PD6 to output

}

//Initialization for the ADC

void initADC()

{

ADMUX |= (1 << REFS0); //Voltage reference is AVcc, multiplexer is set to read pin PC0

ADCSRA |= (1 << ADEN); //This enables the ADC

}

//Method which reads from an analog source and returns an int

int readADC()

{

ADCSRA |= (1 << ADSC); //Start read

while(!(ADCSRA & (1<<ADIF))); //Do nothing while data is being read

ADCSRA |= (1 << ADIF); //reset interrupt flag

return (ADC); //return the read value

}

//Initialization for PWM

void initPWM()

{

TCCR0A |= (1 << COM0A1) | (1 << COM0B1) | (1 << WGM01) | (1<< WGM00); //8 bit clock

TCCR0B |= (1 << CS00); //No prescaler, start the clock

TCNT0 = 0; //start timer

OCR0A = OCR0B = 0;

}

//initialization for analog read's clock

void initBlinkTimer()

{

TCCR1A = 0;

TCCR1B = 0;

TCCR1B |= (1 << WGM12)|(1 << CS12); // set up timer with prescaler = 256 and CTC mode

TCNT1H = 0;

TCNT1L = 0; // initialize counter

OCR1A = 0;

TIMSK1 = (1 << OCIE1A); //enable interrupt for compare

}

void initADCTimer()

{

TCCR2A = 0;

TCCR2B = 0;

TCCR2A |= (1 << WGM21); //set timer to CTC

TCCR2B |= (1 << CS22) | (1 << CS21) | (1 << CS20);//set prescaler to 1024

TCNT2 = 0;

OCR2A = 98; //this is roughly 0.1 seconds

TIMSK2 = (1 << OCIE2A);

}

void stopTimer()

{

TIMSK1 = (0 << OCIE1A);

OCR1A = OCR1B = 0;

OCR0A = OCR0B = 0;

TCNT0 = TCNT1H = TCNT1L = 0;

}

//Initialization for serial communication

void initUsart()

{

UCSR0B |= (1 << RXCIE0) | (1 << RXEN0) | (1 << TXEN0); //Enable RX, TX

UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01); //This sets data size to 8 bit

UBRR0H = (BAUD\_PRESCALE >> 8);

UBRR0L = BAUD\_PRESCALE;

}

//Returns a char that was typed/ received

char receiveByte()

{

while( !( UCSR0A & ( 1 << RXC0 ) ) ){};

return UDR0;

}

//Returns a char that was transmitted

void transmitByte(char data)

{

while( !( UCSR0A & ( 1 << UDRE0 ) ) ){};

UDR0 = data;

}

//Method to print text, it is done by transmitting each char of a string

void printStr(char const\*s)

{

while(\*s)

{

transmitByte(\*s);

s++;

}

}

//Method that processes what user has typed and returns it as an int

int getInput()

{

int bufferIndex = 0; //tracks the current index

char givenByte; //whatever the user typed

char myBuffer[128]; //the string that we are building

int input; //the number that will be returned

while(1)

{

givenByte = receiveByte(); //call method receive byte to get the typed byte

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0) //if the buffer is empty

continue; //do nothing and restart the loop

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(isdigit(givenByte)) //if a number was sent

{

myBuffer[bufferIndex] = givenByte;//add it to the buffer

bufferIndex++; //increment the index

}

else //if something other than a number or enter was sent

{

printStr("Please enter a number.\n\r"); //message the user and ask them to enter a number

}

}

}

//similar to get input, but just for the LED

int getLED()

{

int bufferIndex = 0;

char givenByte;

char myBuffer[128];

int input;

while(1)

{

givenByte = receiveByte();

if(givenByte == 0x0D) //if enter was sent

{

if(bufferIndex == 0)

continue;

myBuffer[bufferIndex++] = '\n'; //newline

myBuffer[bufferIndex++] = '\r'; //carriage return

myBuffer[bufferIndex] = 0; //null-terminate the chars, this turns it into a string

input = atoi(myBuffer); //turn the string into an int

printStr(myBuffer); //print what was in the buffer so user can see what they entered

return input;

}

else if(givenByte == '1' | givenByte == '2') //if 1 or 2 was typed

{

if(bufferIndex < 1) //if there is nothing in the buffer

{

myBuffer[bufferIndex] = givenByte;

bufferIndex++;

}

else//if there is already something in the buffer

{

printStr("Press Enter!\n\r");//it should already be a 1 or 2, so we ask the user to press enter

}

}

else

{

printStr("Please enter 1 or 2.\n\r");//ask user to enter 1 or 2

}

}

}

//method that blinks the LEDs according to how the user specified.

void blink(int dca, int dcb)

{

if(lightOn == false)

{

OCR0A = dca; //LED1 will get duty cycle a

OCR0B = dcb; //LED2 will get duty cycle b

lightOn = true;

}

else

{

OCR0A = 0; //turn off both LEDs

OCR0B = 0;

lightOn = false;

}

}

void getReads()

{

char tempbuff[128];

int tempIndex = 0;

int holder = 0;

holder = readADC();

sprintf(tempbuff, "%d\n\r", holder);

printStr(tempbuff);

}

int main(void)

{

int currentLED = 0;

int givenDelay = 0;

int readDelay = 0;

//initializers

initADC();

initADCTimer();

initLED();

initUsart();

initBlinkTimer();

initPWM();

//set all the outputs to high

PORTD |= (1 << PD5) | (1 << PD6);

printStr("Hello, welcome to LED blinker!\n\r");

printStr("Please press 1 or 2 to pick an LED.\n\r");

currentLED = getLED();

printStr("Please enter a number from between 0 to 255 for the intensity.\n\r");

if(currentLED == 1)

intensityA = getInput();

else if (currentLED == 2)

intensityB = getInput();

printStr("Please enter a number from between 0 to 255 for the intensity of the other LED.\n\r");

if(currentLED == 1)

intensityB = getInput();

else if (currentLED == 2)

intensityA = getInput();

printStr("Please enter a number from between 100 to 2000 for a delay in milliseconds.\n\r");

givenDelay = getInput();

printStr("Please enter a number from 0 to 15 for the interval of reads.\n\r");

readDelay = getInput();

OCR1A = givenDelay \* 4.2; //4200 is roughly 1 sec, I multiplied by 4200 and then divided by 1000, which gives 4.2

maxCount = readDelay \* 10;

sei();

TCNT1H = 0;

TCNT1L = 0;

while(1) //main loop

{

}

return 0;

}

ISR(TIMER1\_COMPA\_vect) // timer1 compare interrupt

{

cli();

blink(intensityA,intensityB);

sei();

}

ISR(TIMER2\_COMPA\_vect) // timer2 compare interrupt

{

ADCCounter++;

if(ADCCounter >= maxCount)

{

getReads();

ADCCounter = 0;

}

}