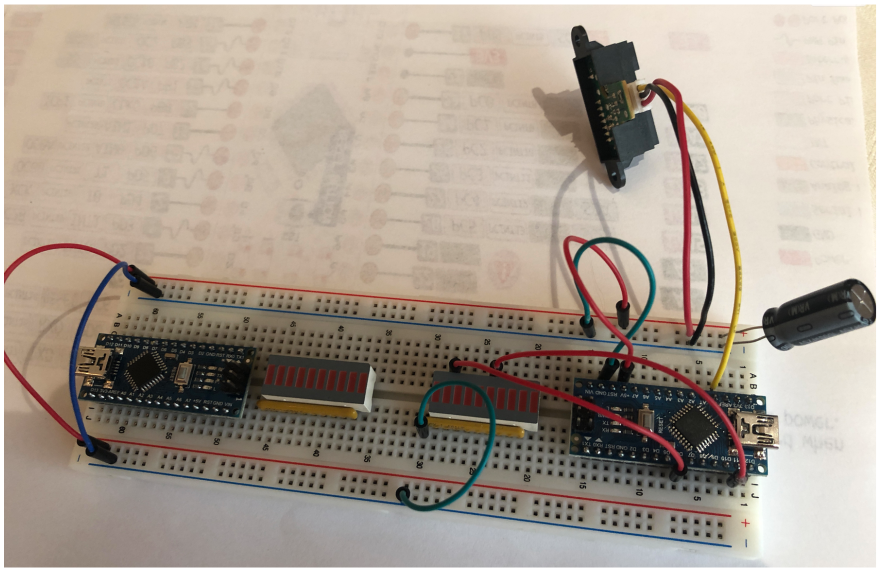
Maximilien Malderle
ID: 26562906
SOEN 422
LAB 1B

Introduction:

The purpose of this lab is to connect a computer source to the Arduino Nano board, to then program the array of lights, while setting up communication links between the board and the computer. Two lights are programmed, one from a regular digital output with only two states, HIGH meaning on and LOW meaning off. One portion of the program tests to see if this light is on, if this is the case the light is switched off. Due to the nature of the program, running in a loop, in the next iteration this off light is switched on again. This portion creates a blinking light. The second light is also set to a digital output pin, but it is configured to a pulse width modulation (PWM), so that different brightness's can be achieved through varying voltages. The next step is to set up an infra-red sensor, by connecting it to a analog input pin. This sensor will be handled through a interrupt event which is set up through the TimerOne library. The rest of the lab is dedicated to pulling information from the board, through Serial print functions and pushing information to the board through Serial read functions.

Physical Resources:

To complete this lab, an Arduino Nano board was used. Power is provided by the USB port which can be connected to a computer. The 5V from the chip is connected to the positive line on the outside of the board, which is then connected to the other positive line on the other side, the same is done with the ground from the chip to the negative line of the board. This makes sure that power is available on both sides of the board for peripherals or to ground the LED array. Between the positive and negative on the chip connected side, there is a capacitor to even out the wave, to get better readings off the infra-red sensor. As for the digital pins used to connect the chip to the LED, PIN 6 (PD6 or D6) was used as the PWM connection, PIN 11 (PB3 or D11) was used for the blinking LED. The analog PIN 14 (PC0 or A0) was used as the input pin for the infra-red sensor, while the red wire is plugged into the positive line and the black wire is plugged into the negative line, providing the sensor with power. PIN 6 has PWM capabilities using timer 0, the necessary registers are configured to achieve this result (see Program snippets).



Software Resources:

To complete this lab, the Arduino IDE was used, many functions and operations are done in a “Bare Metal” fashion, meaning bits are manipulated using C functions and libraries. Due to the use of Timer1 the program could not be run in a main() meaning that it is still considered an Arduino program. The C compiler is used to handle all the bitwise operations, which are necessary to configure the ports.

Program Snippets:

In this section snippets of code will be presented to indicate the manner in which the suggested problems of the lab were solved.

Problem 2:

To turn on the light attached to PIN 11 the DDR register for port B associated PB2 is set to Output. The PORT itself is then turned to HIGH, so that the associate LED is turned on.

```
//turn on light at PB2
DDRB |= (1<<flashLED);
PORTB |= (1<<flashLED);
```

The if statement tests to see if PIN 11 is set to on, in the true case the associated PORT is set to zero turning the light off, otherwise if the light is off then the PORT is set to one turning the light on.

```
void blink()
{
  //test if PIN B2 is set to High
  if(PINB & (1<<flashLED))
  {
    //close PIN B2
    PORTB &= ~(1<<flashLED);
  }
  else
  {
    //open PIN B2
    PORTB |= (1<<flashLED);
  }

};
```

Problem 3:

To set PIN 6 to PWM using Timer0 the TCCR0A registers bits COM0A1, WGM01 and WGM00 are set to one to achieve a Fast/non-Inverted PWM. TCCR0B register bits CS01 and CS00 are set to 1 so that the prescaler is set to 64. OCR0A is set to 0 so that the initial value held in the register is 0. Afterwards PIN 6 is turned on just as in problem 2.

```
//PWM timer0
//Fast and Non-Inverted PWM
TCCR0A |= (1<<COM0A1) | (1<<WGM01) | (1<<WGM00);
//Prescaler set to clkIO/64
TCCR0B |= (1<<CS01) | (1<<CS00);
TCNT0 = 0;

//compare register value set to 0
OCR0A = 0;

//turn on PD6
DDRD |= (1<<varyLED);
PORTD |= (1<<varyLED);
```

Here OCR0A is incremented by 64 to create 4 different levels of brightness. The Serial.print is used to display the brightness information in the computers monitor (THIS is problem 4.

```
void toggle_LED()
{
  //Increments the value of the register by a quarter of 255
  OCR0A += 64;
  _delay_ms(1000);

 //prints information to monitor
 Serial.print("Brightness: ");
 Serial.println(OCR0A,DEC);
}
```

Problem 5:

The first code snippet is to configure the PIN to work with an analog input. Firstly PIN 14 is set to be an Input. Afterwards channel 0 is stored in a variable. The ADMUX registers is set to the appropriate channel and the REFS0 bit is set to one, setting the reference voltage to 5v. The ADCSRA register bits ADEN is set to one, to enable ADC and the rest are set to one to achieve a prescaler of 128, so that the frequency lies between 50Hz and 200Hz.

```
//settup for IR_Sensor
//Setting PIN 14 to Input
DDRC |= (0<<IR);

//choosing channel one
uint8_t ch=ch&0b00000001;

//setting channel and setting reference voltage to 5v
ADMUX |= ch | (1<<REFS0);

//ADEN enables EDC
//Others set to one sets the Prescaeler to 128 to achieve ADC in range of 50kHz - 200 kHz
ADCSRA |= (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
```

This portion is used to for the actual read and display of the IR_Sensor data. The ADSCRA register bit ADSC is set to one, setting up the conversion. The while loop is used to wait for the conversion to complete, then the ADIF in register ADCRA is set to one to signal that the conversion is complete. The Serial.print is used to print the information to the computer Monitor.

```
void IR_Sensor()
{
  //ADC setting up conversion
  ADCSRA|=(1<<ADSC);

  //waiting for conversion to complete
  while(!(ADCSRA & (1<<ADIF))){};

  //conversion is complete
  ADCSRA|=(1<<ADIF);

  //print information to Monitor
  Serial.print("IR: ");
  Serial.println(ADC);
}
```

Problem 6:

This is the code for the interrupt handler, the initialize function sets the length of the interval between interrupts. The second associates a function to the interrupt routine, in this case it handles the IR_Sensor code.

```
//Creates timer1 interrupt with interval and routine
Timer1.initialize(interval);
Timer1.attachInterrupt(IR_Sensor);
```

Problem 7:

Serial.available() is used to inspect the command line of the computer monitor, if there is something pending to read, then the if results in true and them Serial.read() is used to store this value into a String. This string is then processed to extract the valuable brightness and interval information, which is then inserted into the OCR0A register and Timer1.setPeriod() respectively to change the relative conditions.

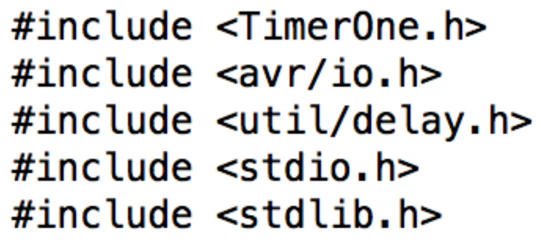
```
if(Serial.available())
{
  String hold;
  String tmp;
  int pos1;
  int pos2;
  bool loop = true;

  //reads Serial String
  hold = Serial.readString();
```

```
//Brightness of the toggle LED is changed
OCR0A = bright;

//Interrupt interval is changed
Timer1.setPeriod(interval);
```

Reference Code Common:



```
#include <TimerOne.h>
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
```

TimerOne.h is an Arduino library which allows the user to use Timer1 to create an interrupt service routine.

Avr/io.h is used so that the C language can set all the register bits.

Stdio.h and stdlib.h are used as standard C libraries, allowing the user to use predefined C functions.

Util/delay.h is included to give the user the ablity to call _*delay*_ms() functions, creating a waiting condition in the program.

Conclusion/Discussion:

This lab focused on the fundamental structure of an Arduino Nano program. Concepts such as the setup and loop function are introduced. It also familiarized us with the concepts of bare metal programming, which requires the programmer to understand the structure and configuration of the hardware. Achieving the necessary results requires that the programmer understand the different configurations of the chip on the board. Necessary hardware modifications through the C language are required to run either a digital PWM signal to an LED or an analog input signal to read an IR_Sensor. Serial communication and interrupts are introduced, but as seen in class there is a Bare Metal solution available, which removes Arduino specific functions.