University of Mary

Hamm School of Engineering

EME 483. Mechanical Measurements Lab

LABORATORY REPORT A

Intro to Arduino Nano

Student Name(s):

1. Isaac Anderson
2. Emmanuel Lopez

Submission Date : September 26, 2024

Contents

1 Introduction

- This lab was an introduction to Arduino Nano registers. There were three parts to the lab. The first part was to get the internal LED to turn on. The second part was to get the internal LED to turn on and off. The last part was to add an external LED and have the internal LED off when the external LED is on and vice versa.

- The primary goal of this lab is to become familiar with the Arduino Nano microcontroller and learn how to configure and manipulate its digital I/O pins to control external devices, such as LEDs by manipulating the bits on an Arduino register with logic gates.

- The Goal of this lab report is to show an understanding of what the different types of Arduino Nano register do the different registers do and that the student can use this knowledge to control the Arduino using these registers in C.

Figure 1: Arduino Nano Micro Controller

Figure 2: LED

2 Theory

- The theoretical foundation of this lab involves understanding basic microcontroller operations, specifically how an Arduino Nano microcontroller interfaces with peripheral devices. The primary theory centers around digital logic and how to use registers (like the Data Direction Register (DDRB) and Port Register (PORTB)) to control the flow of current to specific pins, allowing for the toggling of an LED. This involves concepts like bit-wise operations to modify specific bits without affecting others and the use of C programming to control hardware. The purpose of this lab is to introduce students to the Arduino Nano microcontroller and its associated development environment (Arduino IDE). It familiarizes students with basic operations such as reading from and writing to the pins of the microcontroller. This is accomplished through the simple task of toggling the onboard LED and later, an external LED, using C programming techniques to manipulate digital inputs and outputs. The lab serves as a foundational exercise to develop a practical understanding of microcontrollers and how software can control hardware components.

- **Arduino Nano Microcontroller:** The Arduino Nano is a versatile, small-sized microcontroller board built around the ATmega328P chip. It offers several digital I/O pins, making it ideal for basic control applications, such as toggling LEDs. The lab utilizes its GPIO (general-purpose input/output) capabilities to teach students how to read and write digital signals, essential for embedded systems development.

- **Arduino IDE:** The Arduino Integrated Development Environment (IDE) allows for easy programming and uploading of code to the microcontroller. It supports C/C++ and includes built-in libraries that simplify hardware control tasks. This lab uses the Arduino IDE to write programs that control the state of LEDs connected to the Arduino Nano.

- **External LED:** An external LED is used to expand the lab's scope, allowing students to apply their knowledge beyond the onboard LED. Proper connection of the LED to the correct pins (including ground and a resistor if necessary) is critical for successful operation.

- **USB Cable:** Used to connect the Arduino Nano to the computer, the USB cable provides both power to the board and a communication link for uploading programs and receiving data.

3 Methodology

3.1 Items Used

- Arduino, Breadboard, USB data cable,

- A Green LED, A Jumper Wire, A laptop with Arduino IDE

3.2 Part 1

- The setup for part 1 was to plug the Arduino into a breadboard. Then use the USB cable to plug into the Arduino. How this looks is shown in Figure 3.

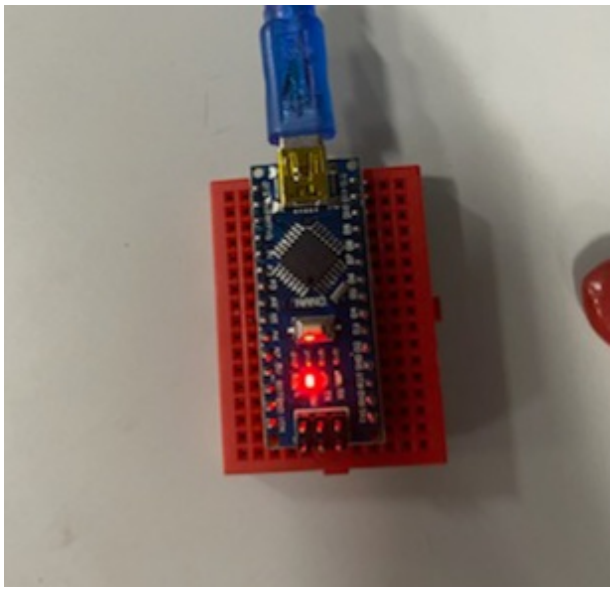

Figure 3: How the Arduino was set up for parts 1 and 2

- How the code in part 1 works is that DDRB is set to have pin 5 on PORT B or the internal LED as an output.
- Next a never-ending while is put in place to keep the light on.
- Then PORTB pin5 is set to on using the bit-wise operator XOR and the HEX value 0x20.

Code For Part 1

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void){
  DDRB |= 0x20;
   while (1){
     PORTB ^= 0x20;
      _delay_ms(1000);
   }
}
```

3.3 Part 2

- For part 2 the the way the Arduino is set up is reused as shown in Figure 3.
- The code for part 2 is similar to part 1 with some parts added.
- Before the while loop an unsigned variable named port is made
- After the while statement whatever PINB is saved by port
- this if statement will check if the status of PORTB pin 5 is on. If it is on PORTB will be changed using an AND Bit-Wise operator and the HEX value 0xDF. This will turn off the internal LED.
- if the if condition is false then PORTB will be sent the command to turn on.
- A delay of 1000 milliseconds is added to the end of the loop so the blinking of the internal LED can be Seen.

Code For Part 2

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void){
  DDRB |= 0x20;
   unsigned port;
   while (1){
     port = PINB;
     if ((port & 0x20) == 0x20 ){
       PORTB &= 0xDF;
     }
     else{
      PORTB ^= 0X20;
     }
     _delay_ms(1000);
   }
}
```

3.4 Part 3

- For part 3 the Arduino setup was modified. An external LED the pin D11 on the Arduino and the other end of the LED connected to ground. How the LED is connected is shown in Figure 4.
- The code for part 3 is modified from the code from part 2.
- Before the while loop DDRB is set so both pin 5 and pin 3 on PORTB are set as outputs.
- After this PORTB is sent the command to turn on pin 5 before the loop starts.
- If the if is met for pin 5 being on then pin 5 is set to turn off then pin 3 is turned on.
- If pin 5 is not seen as on pin 3 will be turned off then pin 5 will turn on.
- A delay is added at the end so the two LEDs can be seen cycling between on and off can be seen

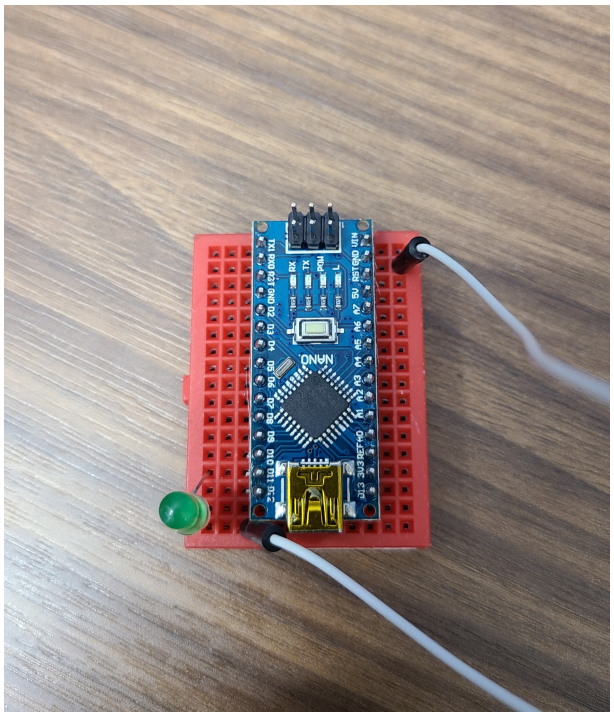

Figure 4: How he Arduino was set up for part 3

Code For Part 2

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void){
  DDRB |= 0x28;
   unsigned port;
  PORTB ^= 0X20;
   while (1){
     port = PINB;
     if ((port & 0x20) == 0x20 ){
       PORTB &= 0xDF;
       PORTB ^= 0x08;
     }
     else{
      PORTB &= 0XF7;
      PORTB ^= 0X20;
     }
     _delay_ms(1000);
   }
}
```

4 Results

4.0.1 Part 1

- As can be seen in Figure 5 the light was able to be turned on using code.

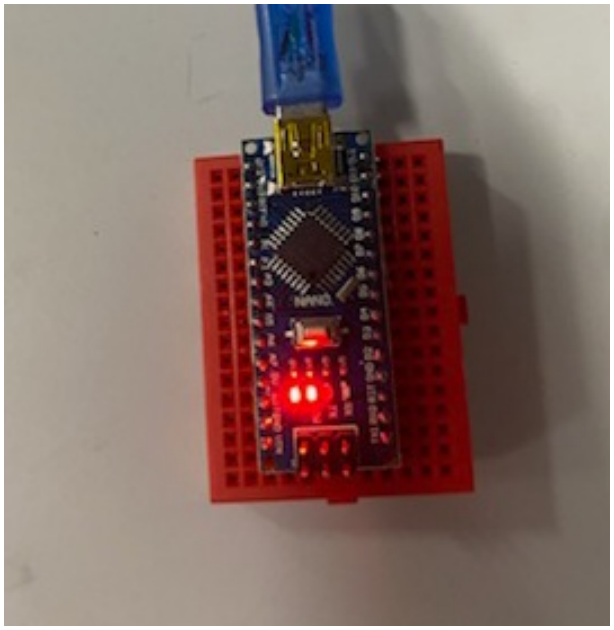

Figure 5: Picture of internal LED on

4.0.2 Part 2

- The internal LED was blinking just as intended. This can be seen in the pictures in Figure 6.

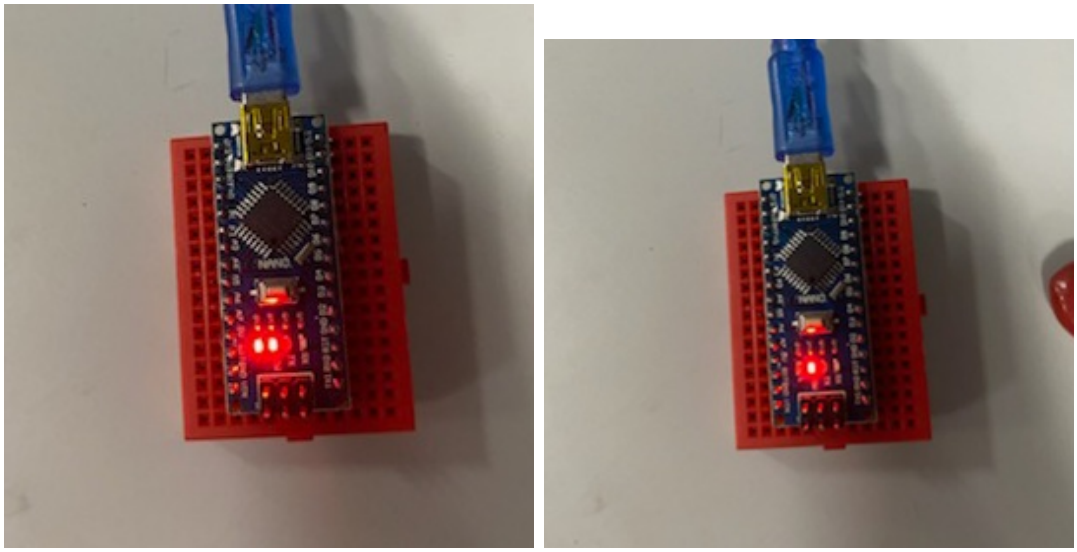

Figure 6: Pictures of internal LED blinking on and off

4.0.3 Part 3

- The external LED and the internal LED are cycling between on and off just as intended. Pictures of this are shown in Figure 7.

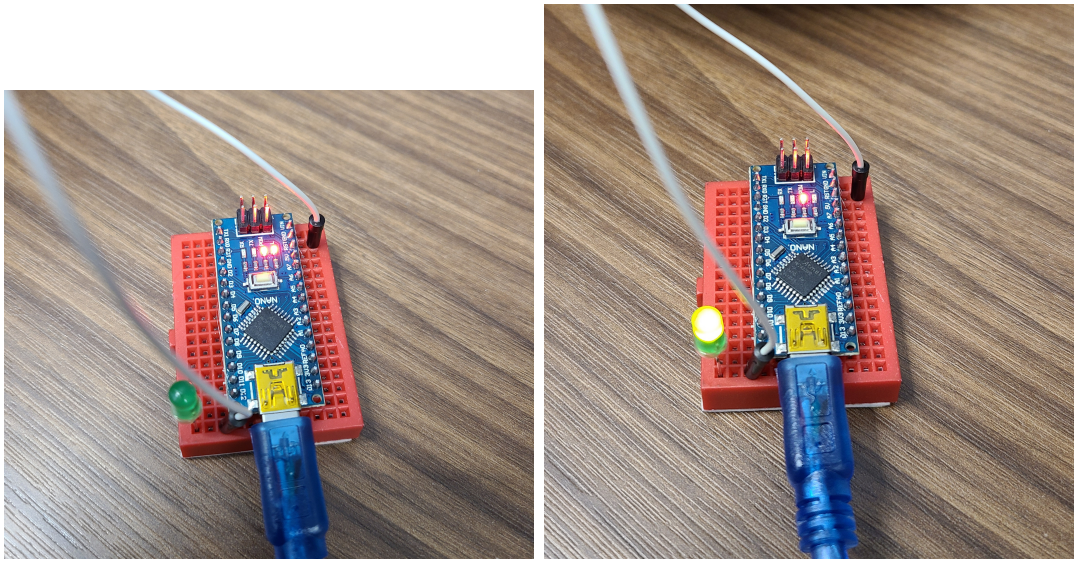

Figure 7: Pictures of external LED turning on when internal LED is off and vise versa

5 Discussion and Conclusion

In this lab, the goal was to introduce or reintroduce the student to the Arduino Nano microcontroller, the Arduino IDE, and the process of reading from and writing to pins on the microcontroller. Specifically, we focused on toggling an onboard LED, followed by controlling an external LED, demonstrating how basic microcontroller programming can interface with external components.

The experiment successfully guided us through configuring the Arduino Nano to control digital pins using bitwise operations, including AND and OR gates, to manipulate registers without disturbing other pin configurations. By writing a simple program, we were able to toggle the onboard and external LEDs, showcasing the practical application of pin control through software.

Results showed that we could toggle the LEDs effectively, confirming that the microcontroller, when properly configured, can handle multiple outputs simultaneously. Additionally, we learned how to connect external components, such as resistors and LEDs, and safely interact with them through the Arduino's I/O pins.

In conclusion, this lab expanded our understanding of how microcontrollers, such as the Arduino Nano, interface with hardware and provided practical experience in writing and deploying C code for basic electronic control. This foundational skill set is crucial for future labs and projects, where more complex sensor and actuator interactions will be required.