Lab Assignment, Module 1

Introduction

The purpose of this lab is to introduce, or reintroduce, the student to the Arduino nano microcontroller, the Arduino IDE and how to read and write to pins on the microcontroller to toggle an LED.

Upon conclusion

- What did you learn from this lab?
- How has this lab expanded your knowledge on how a microcontroller works?
- How has this lab expended your knowledge on developing a C program?
- Did you experience any issues in connecting and configuring the external LED

Lab Equipment & Supplies

Table 1 displays the Lab components and equipment that we will be using as part of the first few lab modules. The datasheets, reference manuals, and programming manuals for this equipment is listed under the Course Orientation module in Canvas.

Table 1: Module 1 Lab Equipment and Supplies

Lab Equipment	File(s) / Document(s)	Description
Tektronix TBS2000B	Tektronix-TBS2000B-Datasheet.pdf Tektronix-TBS2000B-User-Manual.pdf Tektronix-TBS2000B-Programmer-Manual.pdf	Digital Oscilloscope
Tektronix AFG1022	Tektronix-AFG1022-Datasheet.pdf Tektronix-AFG1022-Programmer-Manual.pdf Tektronix-AFG1022-User-Manual.pdf	Arbitrary Function Generator
Keithley 2231A-30	Keithley-2231A-30-Datasheet.pdf Keithley-2231A-30-Reference-Manual.pdf	Triple Channel DC Power Supply
Keithley 2110	Keithley-2110-Datasheet.pdf Keithley-2110-Reference-Manual.pdf	Digital Multimeter
Lab Component		Description
Arduino Nano	Arduino-Nano-datasheet.pdf ATmega328P-DataSheet.pdf	Microcontroller Development Board
Arduino nano data logger		Data Logging Module
USB 2.0 A/B Cable		Connection cable for connecting computer
SD Flash card		Memory card for storing data
Software		Description
Arduino IDE		

Arduino Equipment

For this class, we will be using the Arduino Nano microcontroller development board. We will also be using the Arduino data logger for the Arduino Nano

Lab Assignment, Module 1

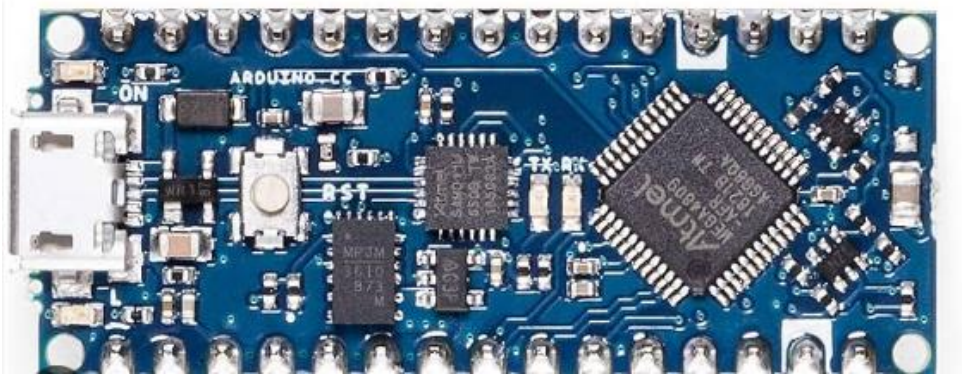


Figure 1: Arduino Nano

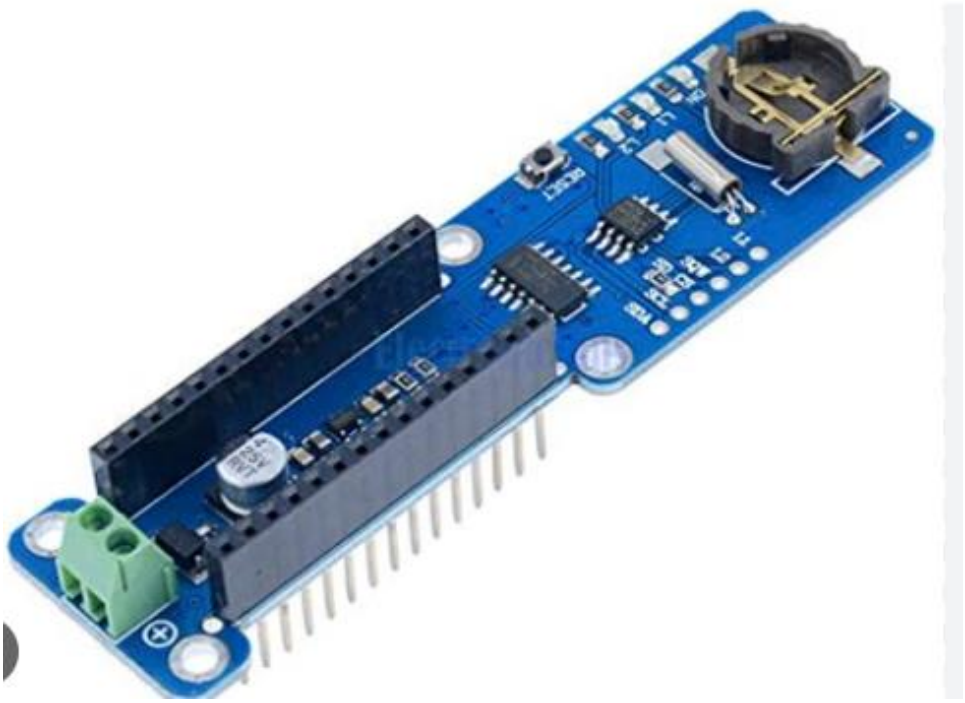


Figure 2: Arduino Nano Data Logger

Lab Assignment, Module 1

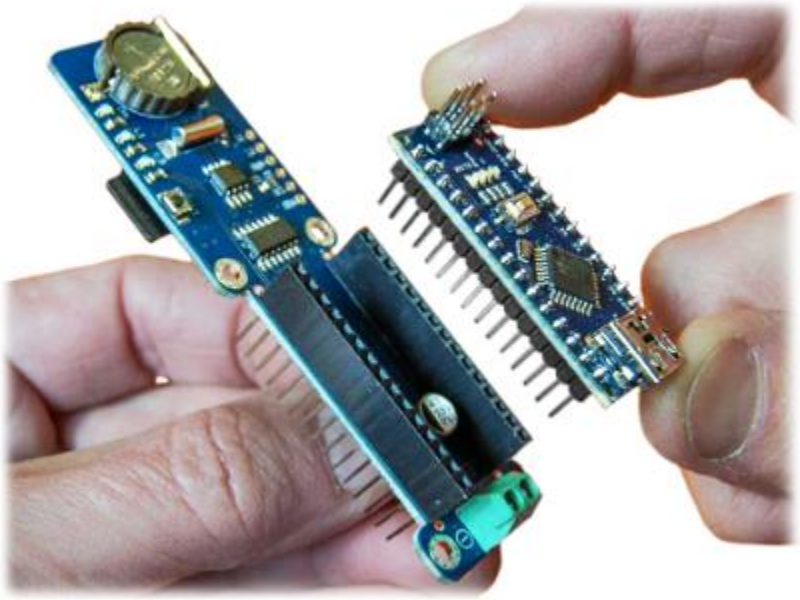


Figure 3: Arduino Nano with Data Logger

Note: Do Not connect the board to your computer yet.

Learning about Arduino Nano

The reason that we are using the Arduino IDE is that it can support C/C++ and shows basic coding principles.

As shown, the body of the application looks like:

```
#include <avr/io.h>

int main(void)
{
    while (1)
    {

    }
}
```

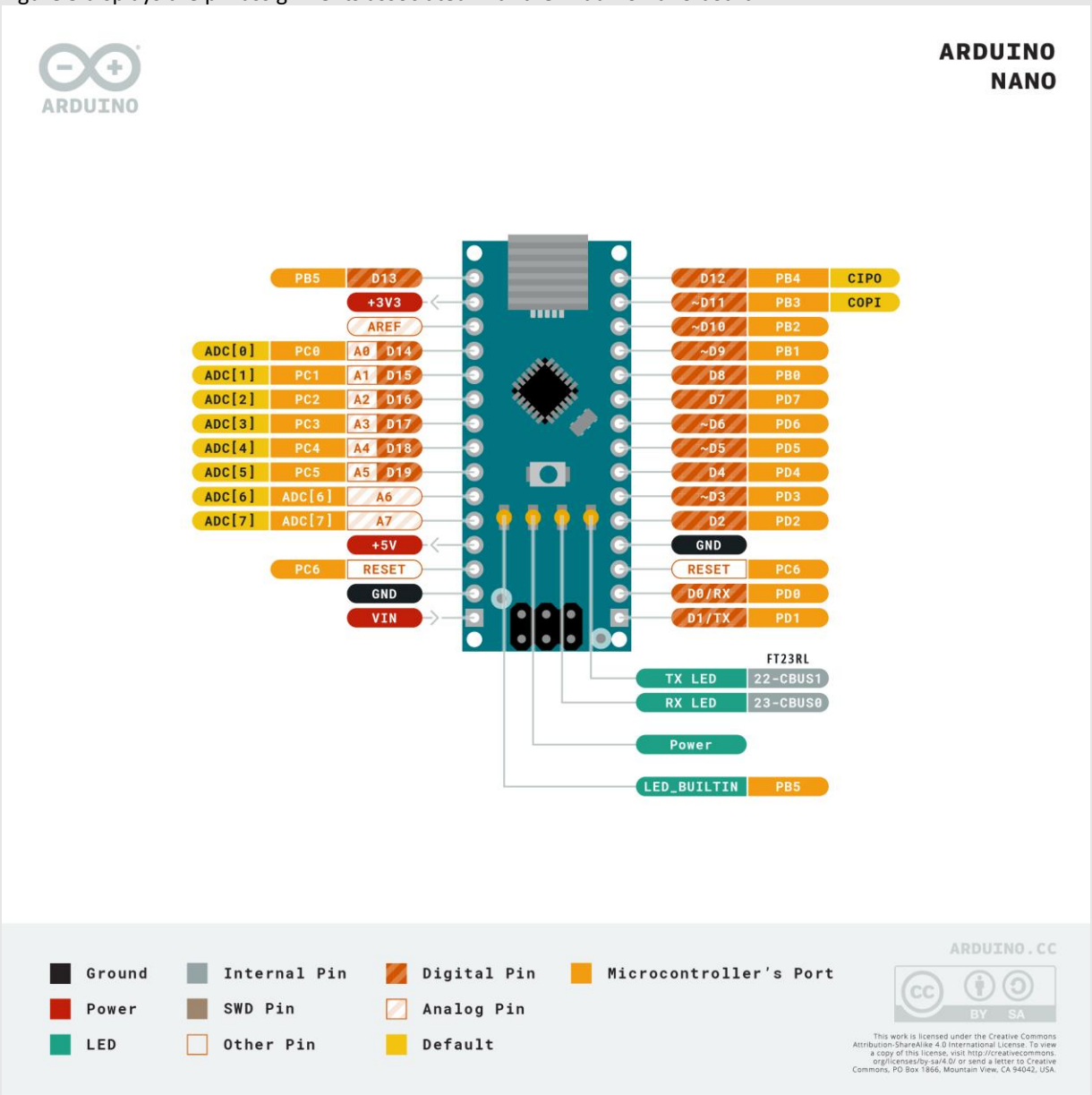
Starting with the curly brackets after main(), the commands are executed within these curly brackets. We use a while(1) loop, so that the application loops through and never exists.

Figure 4 shows the topology of the Arduino Nano board.

Lab Assignment, Module 1

Figure 4: Arduino Nano Board Topology

Figure 5 displays the pin assignments associated with the Arduino nano board.



Lab Assignment, Module 1

Figure 5: Arduino Nano Pin Definitions

For this first part of the lab, we will be simply blinking the LED on the Nano board. If you look at Figure 5, you can see a reference to LED_BUILDIN. Next to this, we can see a label called PB5. This means that the LED is connected to Pin 5 of Port B.

Big Idea (background)

In order to turn on the LED, we have to configure Pin 5 of Port B as an output Pin. By looking at the ATmega328P (same as Nano) data sheet, section 14.4.3, we see the DDRB (Port B Data Direction Register). The Arduino is an 8-bit microcontroller so each of the registers internal to the microcontroller are 8 bits wide. Bit 0 is associated with Pin 0, Bit 1 is associated with Pin 1, etc.

14.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Note on register: For Pin 5 to be an output pin, we must write a binary 1 to Bit 5 of the DDRB register, such that the 8-bit register has a value of 00100000b (0x20).

The tricky part of configuring these registers, is that if we write 0x20 to the DDRB register, it will configure pin 5 as an output pin and the rest of the pins on port B as inputs. So, we must configure pin 5 as an output without disturbing any of the other pin configurations. We do this by OR'ing the existing value of the DDRB register with 0010000b. Thus, pin 5 of the DDRB register will be set but the other pins of the DDRB registers will remain as they were.

```
Port B Direction Register, DDRB
Bit                    7 6 5 4 3 2 1 0
                       - - - - - - - -
Existing DDRB Value    1 0 0 0 0 1 0 0 -> 0x84
OR                     0 0 1 0 0 0 0 0 -> 0x20
                       ===============
Resulting DDRB Value   1 0 1 0 0 1 0 0 -> 0xA4
```

If we desired to clear a bit, we must do something similar except that we have to AND the pin we want reset with a 0. In the example below, Pins 2, 5, and 7 are configured as Outputs. We want to configure Pin 2 as an Input so we must AND the DDRB register with 0xFB. This will leave all pins as they were previously configured but it will change Pin 2 as an output to an input.

```
Port B Direction Register, DDRB
Bit                    7 6 5 4 3 2 1 0
                       - - - - - - - -
Existing DDRB Value    1 0 1 0 0 1 0 0 -> 0xA4
AND                    1 1 1 1 1 0 1 1 -> 0xFB
                       ===============
Resulting DDRB Value   1 0 1 0 0 0 0 0 -> 0xA0
```

Lab Assignment, Module 1

If we were configuring this application from scratch, we would have to dig into the ATmega328P datasheet further and make lots of definitions. Since we are using Arduino IDE Studio, all of this has been done for us.

There is a header file contained in the Studio compiler directories called iom328p.h. This header file contains all of the pin definitions for the ATmega328P microcontroller. Open this file from Canvas and take a look.

Creating Your First Project, Toggling the On-Board LED

In our application, in our initialization code, we want to configure Pin 5 of Port B as an output.

This is done by:

```
DDRB |= 0x20;

alternatively,
DDRB |= ( 1 << DDB5 );
```

DDRB is defined in iom328p.h. If we don't configure the pin as an output, the microcontroller will ignore our command to toggle the LED.
To toggle the LED, we will alternatively write 0 and 1 to pin 5 of Port B. The same pin concept presented above works for PORTB, just as it did for DDRB.

Our resulting application looks like the following:

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

int main(void)
{
   DDRB |= 0x20;          // (1 << DDB5)

   while (1)
   {
      PORTB ^= 0x20;             // (1<<PORTB5)
      _delay_ms(1000);
   }
}
```

F_CPU is the frequency of the Arduino board and is needed to run the clock and accurately delay for the specified milliseconds. This is found by looking at the Arduino_Nano_datasheet again.

- **ATMega328P** Processor
  - **Memory**
    - AVR CPU at up to 16 MHz
    - 32KB Flash
    - 2KB SRAM
    - 1KB EEPROM

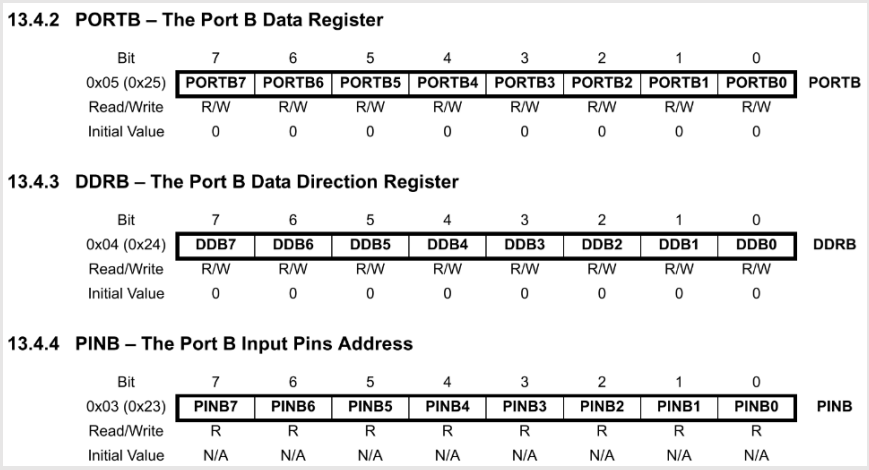
Lab Assignment, Module 1

The source code for this project can be found on the following git repository.
https://github.com/tpvolkman/EME483_Lab_01_Blilnk_LED.git

Creating Your Second Project, Toggling the On-Board LED

For this part of the project, you can either create a new project or modify the first part of this. We are again going to turn the on-board LED but now we are going to do it by reading the status of the LED, and then changing it. If the LED is On, we will turn it Off. If the LED is Off, we will turn it On.

To recall, we used the DDRB register to configure the LED port as output and the PORTB register to turn the LED On and Off. In this second part of the lab, we will use the PINB register to read the status of Port B.



13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

When we read PORTB to determine if the LED is On or Off, we use the PINB register. We can create a variable of 8 bits to store the value of the PORTB register.

```
unsigned port;
```

We can then store the value of the PORT B Pins by assigning it to this variable.

```
port = PINB;
```

To determine if the LED is ON, we check the specific bit to determine if it is a 0 or a 1. To ensure that we are only checking on that specific pin, we must make a comparison. Since the LED is on Pin 5, we check this bit by the following:

```
if (( port & 0x20 ) == 0x20 )
{
}
```

The source code for this project can be found on the following git repository.

Lab Assignment, Module 1

https://github.com/tpvolkman/EME483_Lab_01_Toggle_LED.git

Creating Your Third Project, Toggling an External LED

For this last part of the lab assignment, you will be hooking up an external LED to any port on the Nano that is not on Port B and not on Pin 5. Refer to Figure 5 to see which pins are available on the Nano directories, there are a number of pins available for digital Input.

An LED must be connected so it will properly illuminate. As show in Figure 6, the two legs of the LED has two legs. The longer leg of the LED is called the Anode and will be connected to the Port Pin on the microcontroller that will control the voltage to it. The shorter leg of the LED is called the Cathode and must be connected to ground.

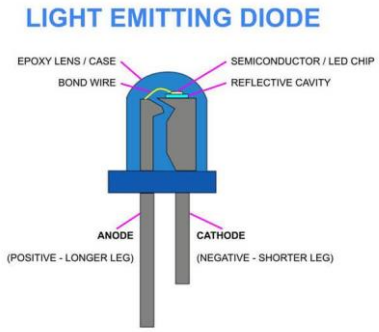


Figure 6: External LED

Configure the pin that you select to be an output pin. Then write software that will toggle the Nano onboard LED and the external LED at alternate intervals. When the Nano onboard LED is On, the external LED shall be Off. When the Nano onboard LED is Off, the external LED shall be On.

Grading Rubric

See "EME483_LabReportFormat.pdf" on Canvas