Alexis Diaz
CPE 301-1001
Lab 04
October 06, 2017

Assignment Description:

In this lab, the objective was to utilize resistors, a MOSFET, to activate a light emitting diode (LED) and understand the purpose of a MOSFET transistor, and then use an ATmega 2560 programmed with the Arduino IDE to activate and manipulate an LED.

Problems Encountered:

There were no major problems during this lab. A minor point in the lab where my partner and I were stuck was trying to figure out how to change an LED to light up from active high to active low by changing whether the anode or the cathode was connected to ground.

Lessons Learned:

In this lab, I learned how to make an LED light up when the ground is connected to the LED's cathode or anode. I also learned that the purpose of transistors, specifically a MOSFET, is to save the state of the LED until its state is changed.

Description of Completed Lab:

Procedure:

1. The first part of the lab says to download the Arduino IDE, but I had it installed on my computer prior to the lab.

2. Second, we constructed the first circuit. The first circuit consisted of simply powering an LED with constant current. After assembling the circuit, we answered the questions in the PDF. The questions for the circuit will be answered at the end of the report. A diagram/schematic of the first circuit is shown below on the left, while a photograph of the first circuit is shown below on the right.

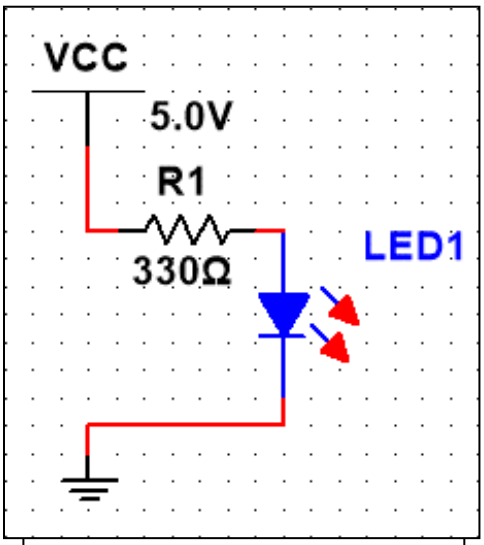

Figure 01 – Circuit 01 Diagram

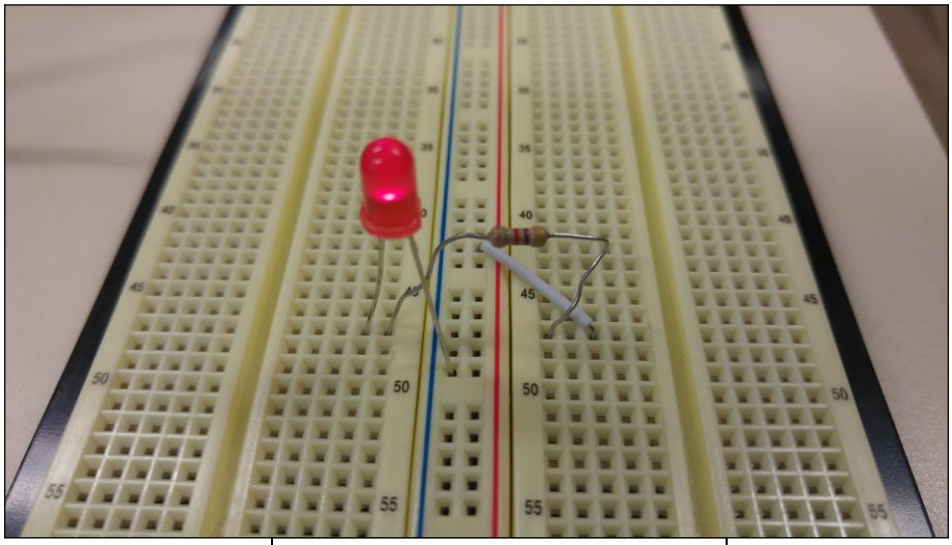

Figure 02 – Circuit 01 Photograph

3. Third, we used the digital oscilloscope to measure the voltage of the circuit before and after the current passed through the 330Ω resistor. As you can see in the images below of the oscilloscope display, the measurements were: Voltage before 330Ω resistor = 5.20V; Voltage after 330Ω resistor = 1.70V

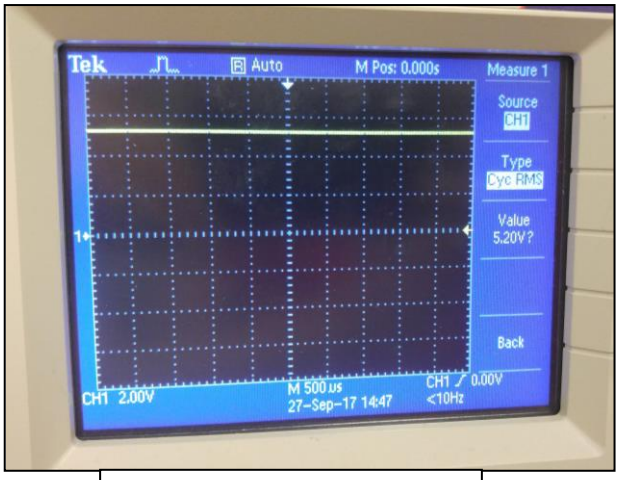

Figure 03 – Circuit 01
Voltage before 330Ω

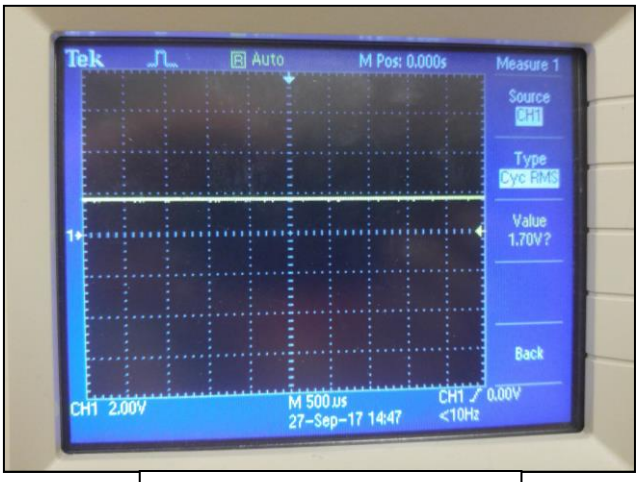

Figure 04 – Circuit 01
Voltage after 330Ω resistor

4. Next, we created our second circuit. It is similar to the first circuit, but now there is a switch between the LED cathode and ground. We tested this circuit, by pressing the button and found that the LED lit up while the button was pressed. A diagram/schematic of the second circuit is shown below on the left, while a two photographs of the first circuit are shown below.

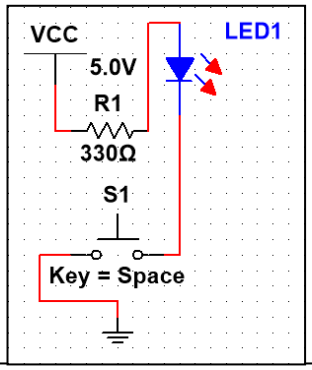

Figure 05 – Circuit 02 Diagram

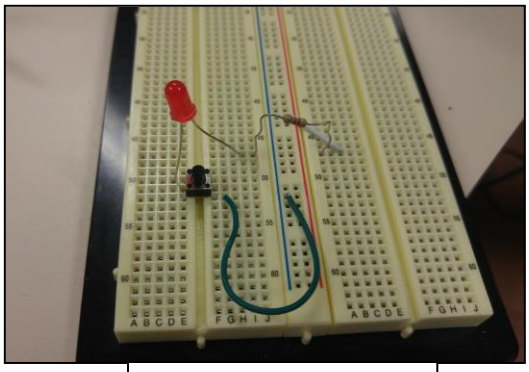

Figure 06 – Circuit 02
Button not pressed

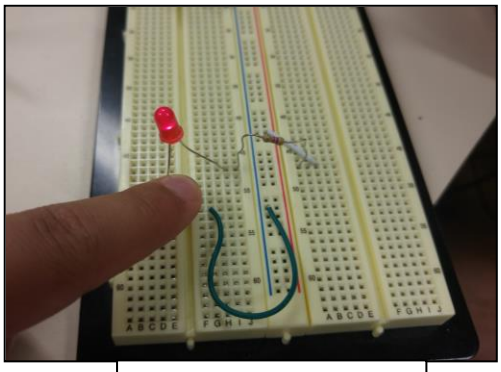

Figure 07 – Circuit 02
Button pressed

5. After that, we built the third circuit, which again, was similar to the circuit before it, but the difference with this one was that the LED cathode was connected to a MOSFET instead of ground. Below are four photographs with the different states that the MOSFET could have read into the LED.

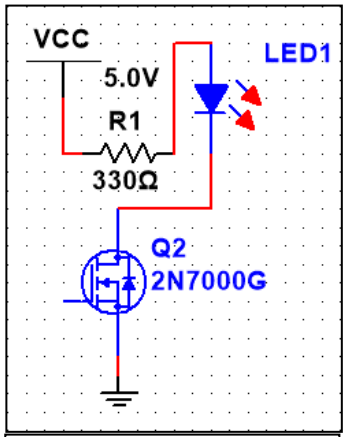

Figure 08 – Circuit 03 Diagram

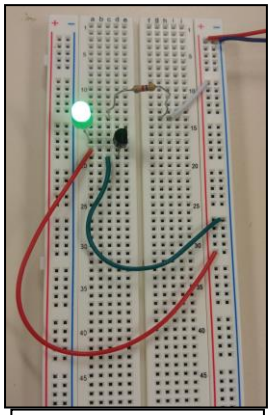

Figure 09 – Circuit 03
LED:ON, POWER:YES

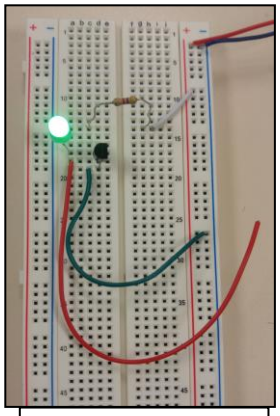

Figure 10 – Circuit 03
LED:ON, POWER:NO

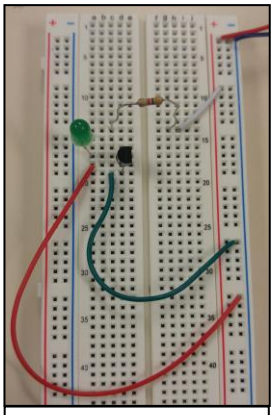

Figure 11 – Circuit 03
LED:OFF, GROUND:YES

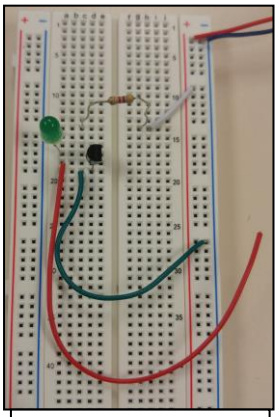

Figure 12– Circuit 03
LED:OFF, GROUND:NO

6. After testing the third circuit, we rearranged some wires to create the fourth circuit. The fourth circuit used the same components as the third circuit, but this time the source was connected to the circuit after the resistor, the gate was varied between active high and active low, and the drain was connected to GROUND. Another change was that the cathode of the LED was now grounded.

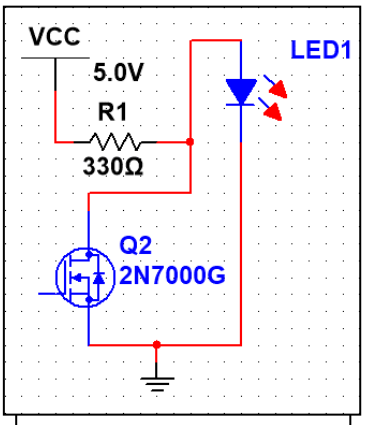

Figure 13 – Circuit 04 Diagram

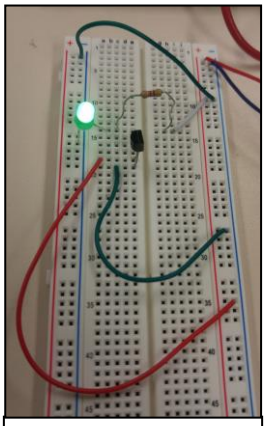

Figure 14 – Circuit 04 LED:ON, GROUND:YES

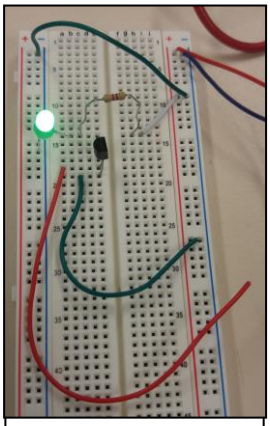

Figure 15 – Circuit 04 LED:ON, GROUND:NO

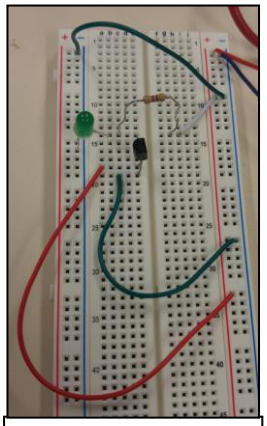

Figure 16 – Circuit 04 LED:OFF, POWER:YES

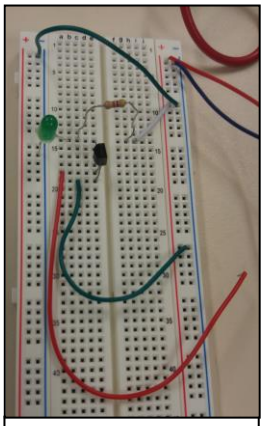

Figure 17 – Circuit 04 LED:OFF, POWER:NO

7. Then, after we made sure the LED lit up on active low instead of active high in the fourth circuit, we moved onto working with the Mega 2560. We set up the Mega 2560 by connecting it via USB to a computer, selecting the correct board(Mega 2560), and set the correct port number. Then, we ran the "Blink" Example program included in the Arduino IDE:

Figure 18 – "Blink" Example Sketch

8. Then, we ran the same program with a few minor adjustments. We changed the speed of the blinking my modifying the delay from 1000ms to 200ms both after writing both HIGH and LOW to the LED.

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(200);                        // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(200);                        // wait for a second
}
```

Figure 19 – "Blink" Faster

9. After that, I wrote a program so that the Arduino would light an external LED (with a resistor). The following is a photograph of the Mega 2560 running the program at an instance where the Mega 2560 had the LED lit:

```
PART_II_-_Q3__Blink_External_

// CPE 301: Lab 04
// Used in Questions 3 and 4
// September 27, 2017

int ledPin = 53;

// Initialize pins as outputs.
void setup()
{
  pinMode(ledPin, OUTPUT);
}

// Loops: Turns LED on, waits, Turns LED off, waits
void loop()
{
  // Change delay for faster blinking
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}

Done compiling.

Sketch uses 1470 bytes (0%) of program storage space
Global variables use 9 bytes (0%) of dynamic memory,

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM5
```

Figure 20 – "Blink" w/ external LED (Program)

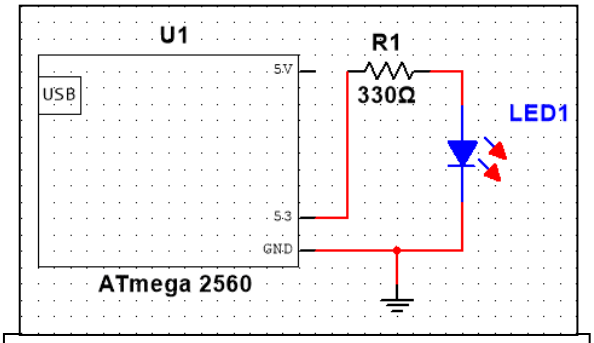

Figure 21 – "Blink" w/ external LED (Diagram)

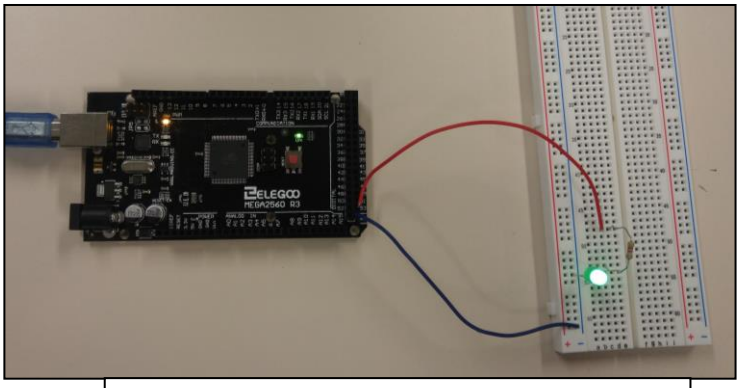

Figure 22 – "Blink" w/ external LED (Photo)

10. For the circuit after that one, which I will call Circuit 6, we modified the circuitry to use a MOSFET to light the LED, but we still used the same Arduino program.

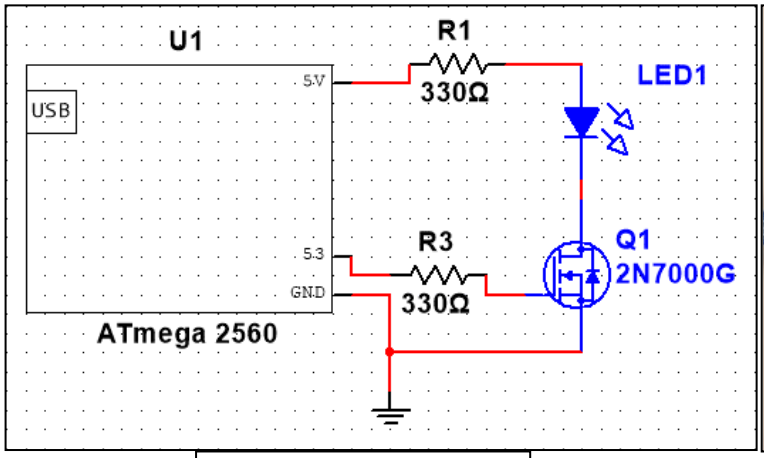

Figure 23 – Circuit 06 Diagram

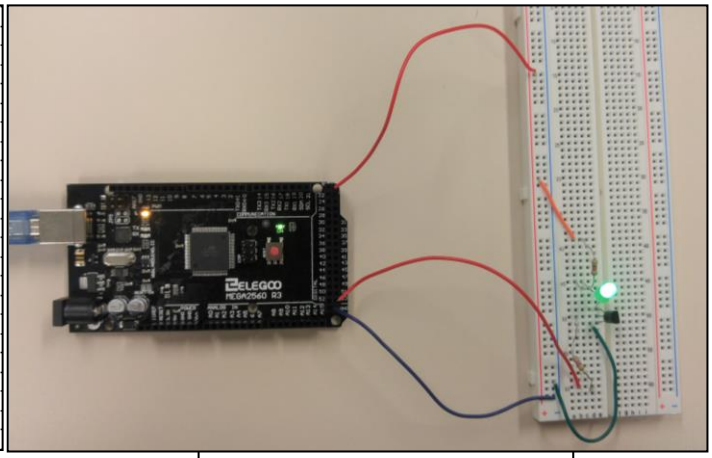

Figure 24 – Circuit 06 Photograph

11. Finally, for Circuit 07, we did something similar that was done to go from Circuit 3 to Circuit 4. We moved the circuit around for the LED to turn on on active low instead of active high.

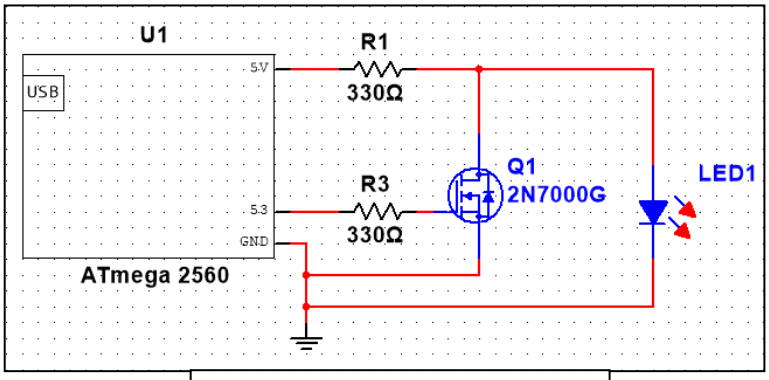

Figure 25 – Circuit 07 Diagram

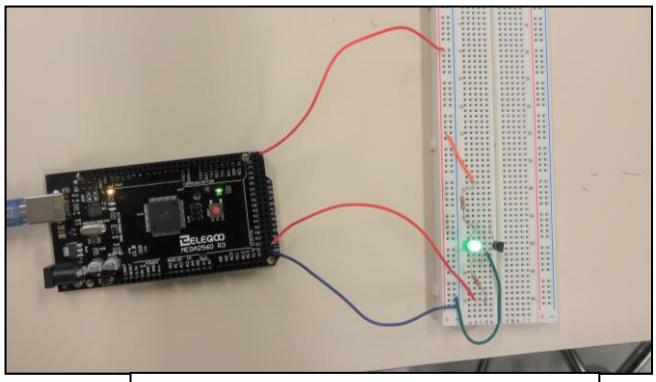

Figure 26 – Circuit 07 Photograph

Questions:

- Part I: Q1a
  As you can see in Figures 3 and 4, the initial voltage was 5.2V and the voltage after the resistor was 1.7V. Therefore, the voltage drop across the resistor was 5.2V-1.7V = 3.5V
- Part I: Q1b
  If you were to increase the resistance to 1000Ω, the LED would be dimmer.
- Part I: Q1c
  If you flip the LED around (exchange anode and cathode), the LED will not power on.

- Part I: Q3a
  A transistor saves the state of power (Active HIGH or Active LOW) until it is changed.

- Part I: Q4a
  If you connect the MOSFET output to the anode instead of cathode but keeping the LED polarity the same, then the LED now powers on on active LOW instead of active HIGH.

- Part II: Q3a
  The Arduino's current limit per I/O pin is 40mA. Assuming I'm using a Red LED (Which uses about 2.2 forward Voltage), 5V power supply; 5V – 2.2V = 2.8V/70Ω = 40mA. The smallest resistor I would dare use is, anything greater than 70Ω, but the recommended current per I/O pin is 20mA, so I would use a 140Ω resistor minimum.
- Part II: Q3b
  If you blink the LED too fast, the blinking becomes so fast that the human eye can't easily see when the LED turns off that it looks like it's constantly on.

- Part II: Q4a
  The thing that is limiting the current that you can supply to the LED is the 300Ω resistor.
- Part II: Q4b
  The advantages of using this circuit over the previous one is that we can now easily change the state of the LED thanks to the MOSFET transistor keeping the on or off state.

- Part III: Q1a
  The advantage of using preprocessor definitions like HIGH and LOW are that it allows the programmer to use the words "HIGH" and "LOW", which are more "human-friendly" to read and write than "0x01" and "0x00".

- Part III: Q1b

  Again, an advantage to using macros instead of the definition of a macro is much easier to use because it allows the programmer to write more concise code because all they have to do is write: `bitRead(value, bit)` instead of `(((value) >> (bit)) & 0x01)` which is more "human-friendly" language.

- Part III:Q1c

  Here is an explanation of the pinMode function line-by-line:

```
void pinMode(uint8_t pin, uint8_t mode)
{
        uint8_t bit = digitalPinToBitMask(pin);  // Assigns the pin of the microcontroller to a bit
        uint8_t port = digitalPinToPort(pin);    // Assigns the port
        volatile uint8_t *reg, *out;             //

        if (port == NOT_A_PIN) return;           //

        // JWS: can I let the optimizer do this?
        reg = portModeRegister(port);            //
        out = portOutputRegister(port);          //

        if (mode == INPUT) {                     //
                uint8_t oldSREG = SREG;          //
                cli();                           //
                *reg &= ~bit;                    //
                *out &= ~bit;                    //
                SREG = oldSREG;                  //
        } else if (mode == INPUT_PULLUP) {       //
                uint8_t oldSREG = SREG;          //
                cli();                           //
                *reg &= ~bit;                    //
                *out |= bit;                     //
                SREG = oldSREG;                  //
        } else {                                 //
                uint8_t oldSREG = SREG;          //
                cli();                           //
                *reg |= bit;                     //
                SREG = oldSREG;                  //
        }
}
```

- Part III:Q1d

  Here is an explanation of the digitalWrite function line-by-line:

```
void digitalWrite(uint8_t pin, uint8_t val)
{
        uint8_t timer = digitalPinToTimer(pin);                         //
        uint8_t bit = digitalPinToBitMask(pin);                         //
        uint8_t port = digitalPinToPort(pin);                           //
        volatile uint8_t *out;
        //

        if (port == NOT_A_PIN) return;                                  //

        // If the pin that support PWM output, we need to turn it off   //
        // before doing a digital write.                                //
        if (timer != NOT_ON_TIMER) turnOffPWM(timer);                   //

        out = portOutputRegister(port);                                 //

        uint8_t oldSREG = SREG;                                         //
        cli();                                                          //

        if (val == LOW) {                                               //
                *out &= ~bit;                                           //
        } else {                                                        //
                *out |= bit;                                            //
        }

        SREG = oldSREG;                                                 //
}
```

- Part III: Q1e

  Here is an explanation of the digitalRead function line-by-line:

```
int digitalRead(uint8_t pin)
{
        uint8_t timer = digitalPinToTimer(pin);                         //
        uint8_t bit = digitalPinToBitMask(pin);                         //
        uint8_t port = digitalPinToPort(pin);                           //

        if (port == NOT_A_PIN) return LOW;                              //

        // If the pin that support PWM output, we need to turn it off
        // before getting a digital reading.
        if (timer != NOT_ON_TIMER) turnOffPWM(timer);                   //

        if (*portInputRegister(port) & bit) return HIGH;                //
        return LOW;                                                     //
}
```

- Part III: Q2a

  The setup function in main gets called once. The loop function is called by a for loop with no initialization expression, no test/conditional expression, and no update expression, which means it will loop indefinitely. Inside the for loop it also checks for a serialEventRun and runs the function if there is a serialEvent.

- Part III: Q3

  The TA said that we will not be using any of the Arduino Library functions for the rest of the semester unless specified to do so. Instead, we will have to create the library functions ourselves using ANSI C. Any assignment or test answers that contain Arduino Library functions will be marked wrong.