Patrick Austin
CPE 301 - 1104
Assignment # 4
10/6/2016

Assignment Description:

In this lab we experimented with various ways of wiring an LED, using simple breadboarding and then bringing in the Arduino to drive enabling and then blinking of the LED. We experimented with using the Arduino, MOSFET transistors, and physical switches in the process of the wiring. We answered various questions about these circuits and diagrammed them. Then we investigated some of the built-in library functions of the Arduino IDE as well as the design of the main program.

Problems Encountered:

Difficulties were largely conceptual, relating to understanding the workings of the MOSFET, diagramming of the circuits, and reading the code for the Arduino library functions. Other than wrapping my head around these concepts and how to word/draw my answers to the given questions the lab went smoothly.

Lessons Learned:

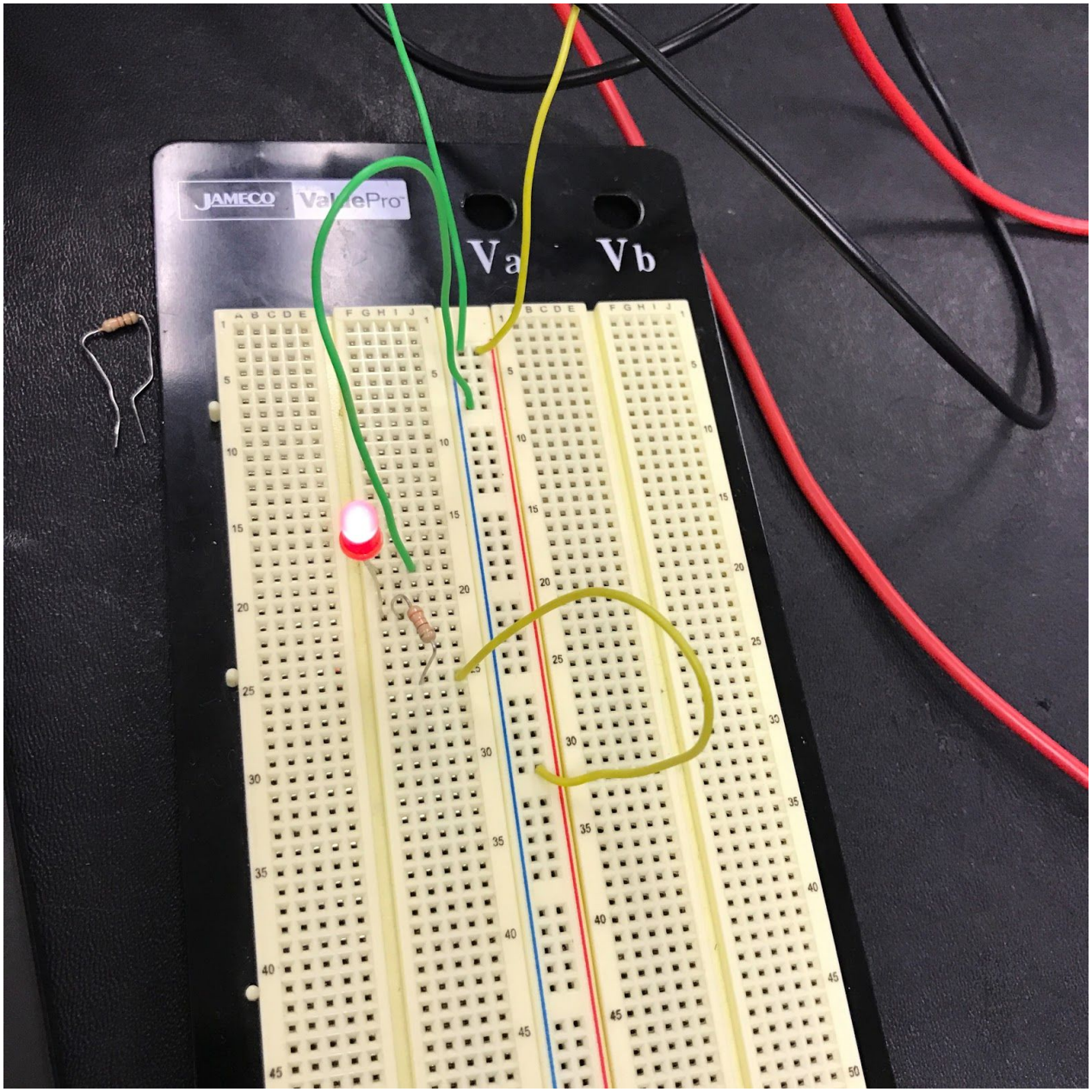
Got a minor crash course on transistors and their potential uses in a circuit via the MOSFET, and got some initial experience in actual physical work with the Arduino out to a breadboard. This was neato and reinforced what we've been building up to in lecture. One thing to read about it, another thing to make it happen on your own board and all that. Also benefitted from some review on basic circuits concepts.

Description of Completed Lab:

Part I

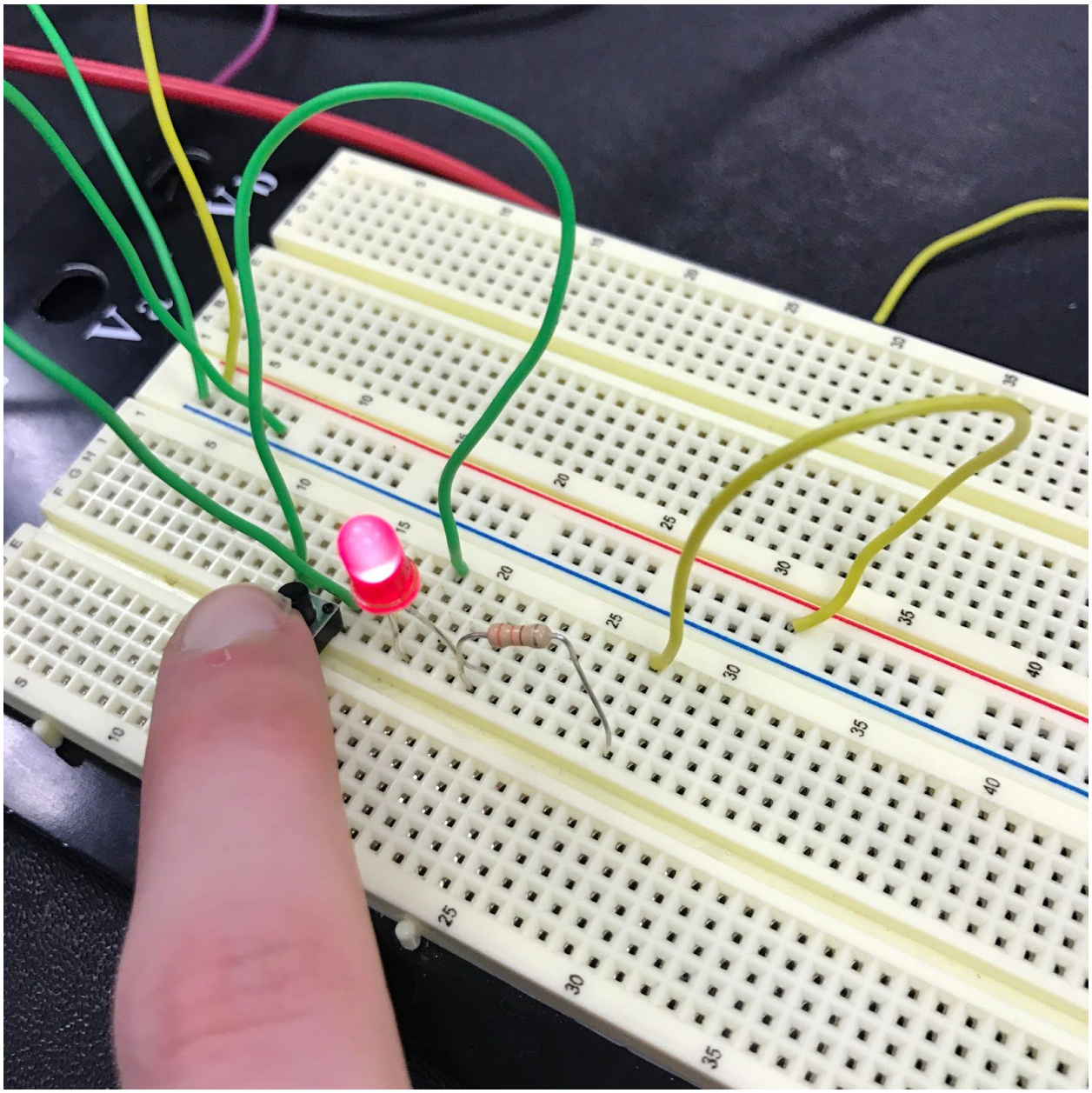
1. a. Power supply, 330 ohm resistor, LED, breadboard & wires
   b. Attached.
   c. a. 5V is dropped across the resistor.
      b. The LED is dimmer when the 1000 ohm resistor is used.
      c. When anode and cathode are flipped the LED does not light up.

   Photo:



2. a. Power supply, 330 ohm resistor, LED, switch, breadboard & wires
b. Attached.

Photo:

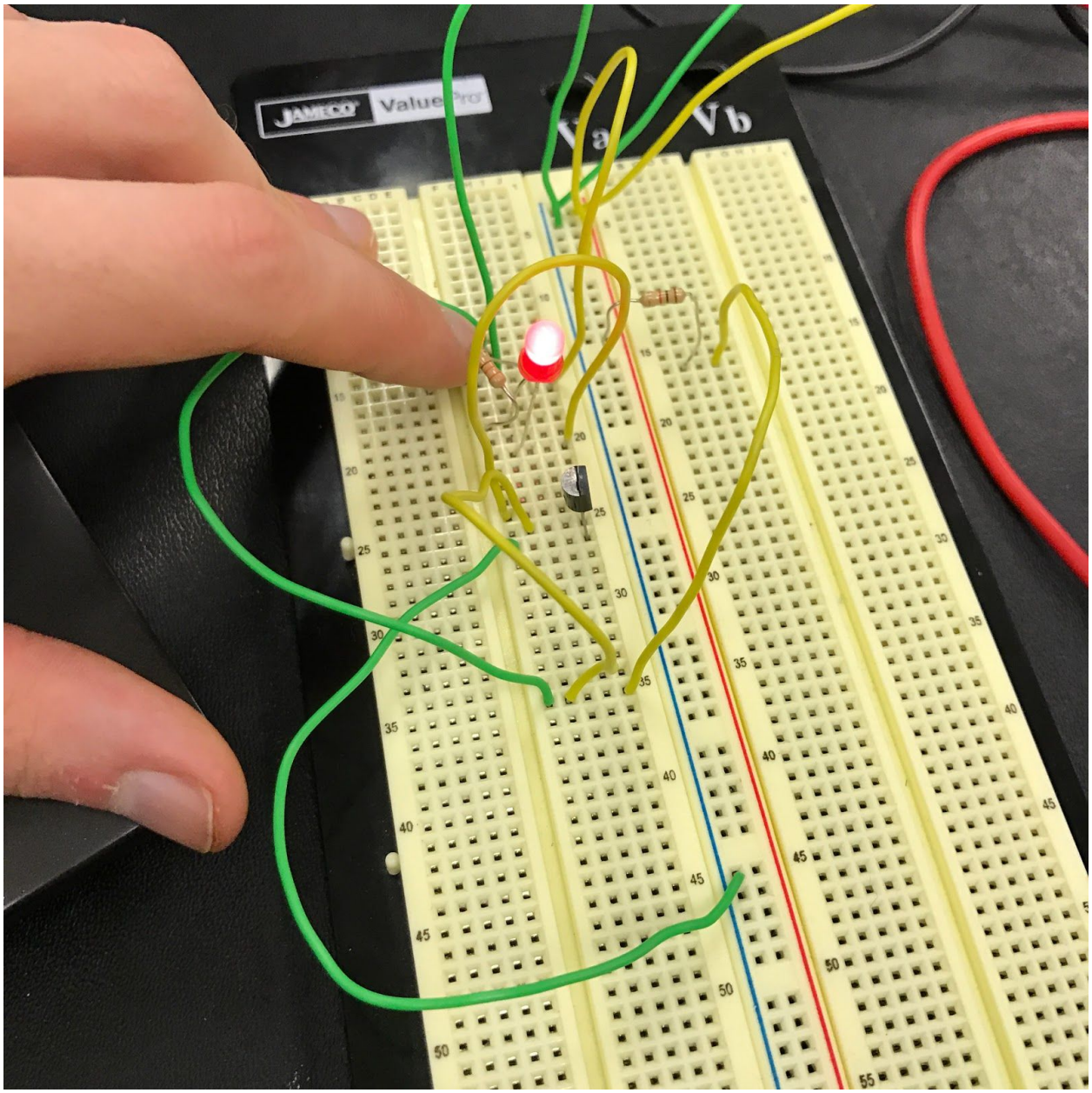


3. a. Power supply, 330 ohm resistor, LED, switch, MOSFET, breadboard & wires

b. Attached.

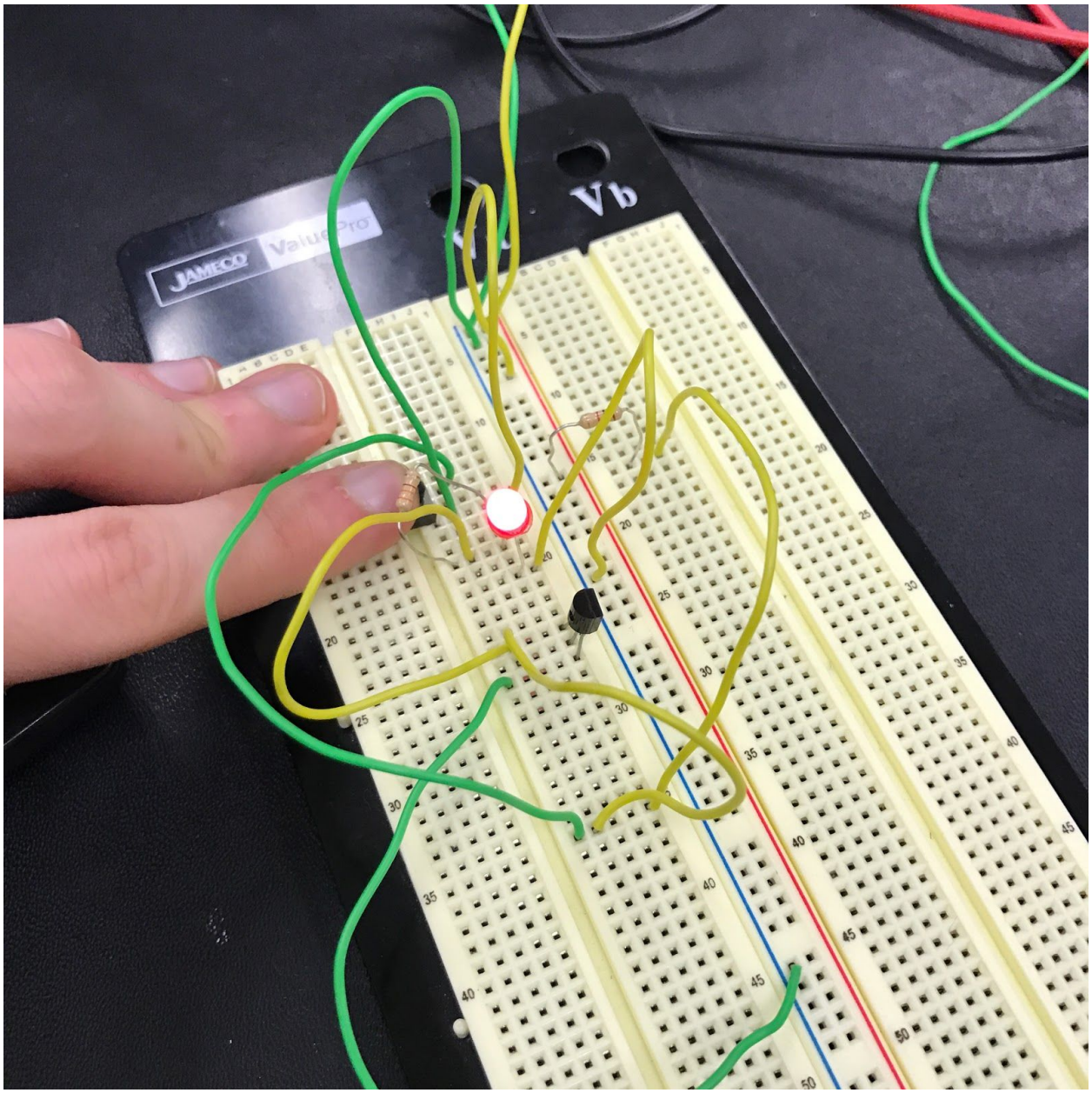
c. a. The transistor completes the circuit and enables the LED to come on based on the switch input.

Photo:



4. a. Power supply, 2 330 ohm resistors, LED, switch, MOSFET, breadboard & wires
b. Attached.
c. a. Whether the LED was active on button press or deactivated on button press changed.

Photo:



Part 2

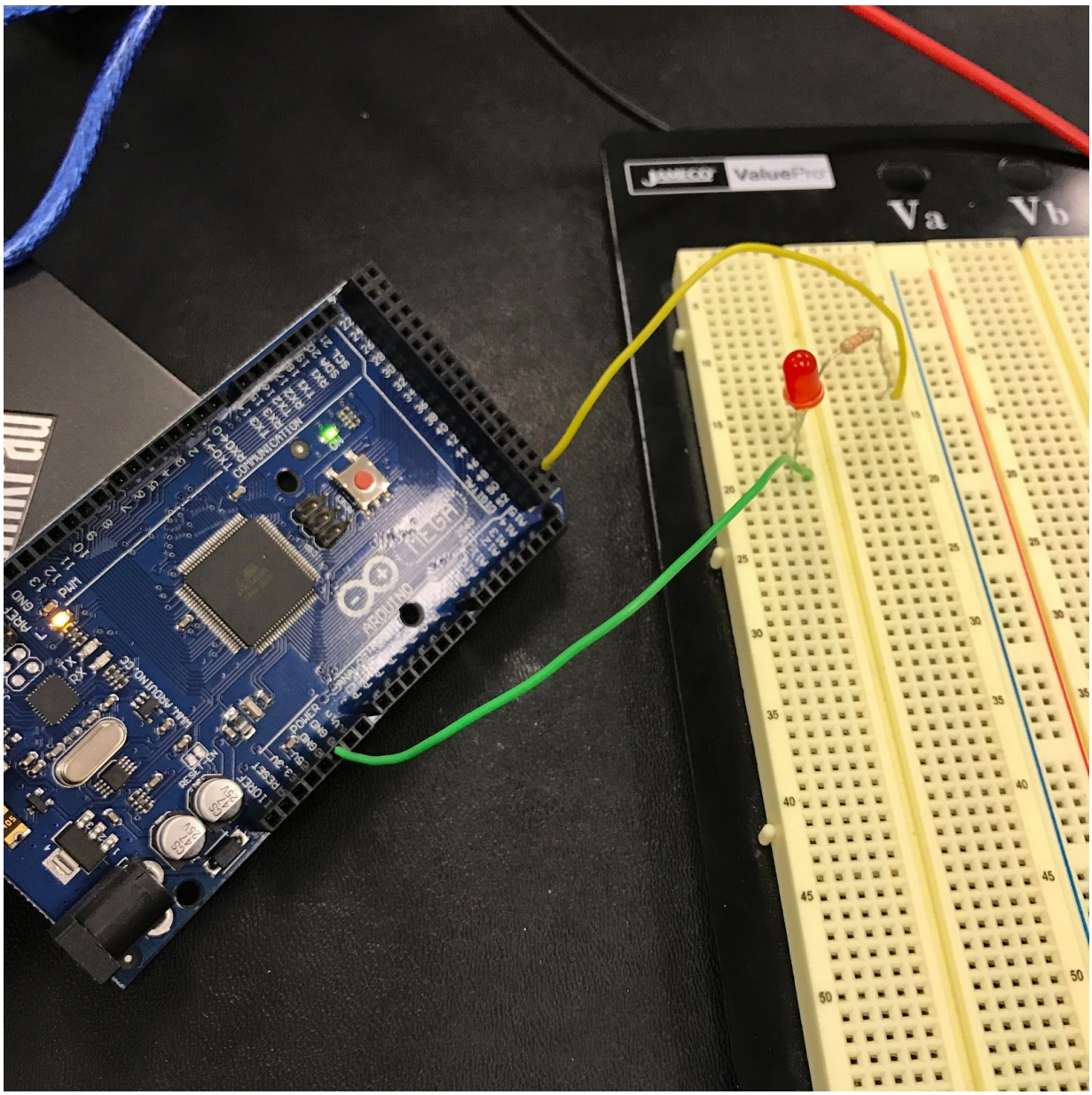
3. a. Arduino, resistor, LED, laptop, breadboard & wires

b. Attached.

c. a. As specified on the pinout, the absolute max current per pin is 40mA and the recommended value is 20mA. So the absolute smallest resistor one would dare to use is given by V = IR where V = 5 and I = .04A. So a 125 ohm resistor would be the smallest possible and a 250 ohm resistor (as above, where I = .02A) would be recommended.

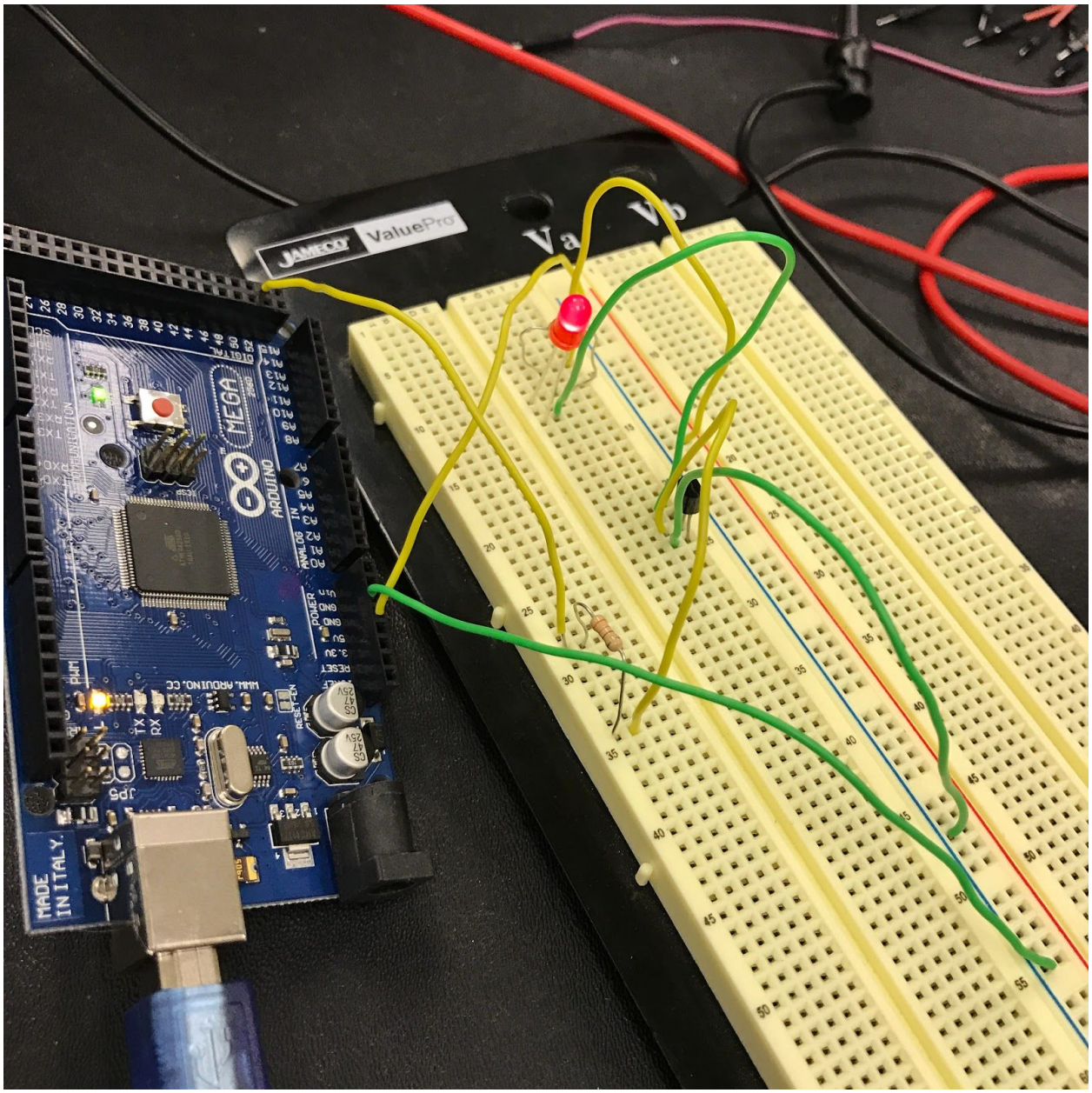
b. Once the blinking is fast enough it is indistinguishable from the light simply being on. We observed this at 1000 hZ.

Photo:



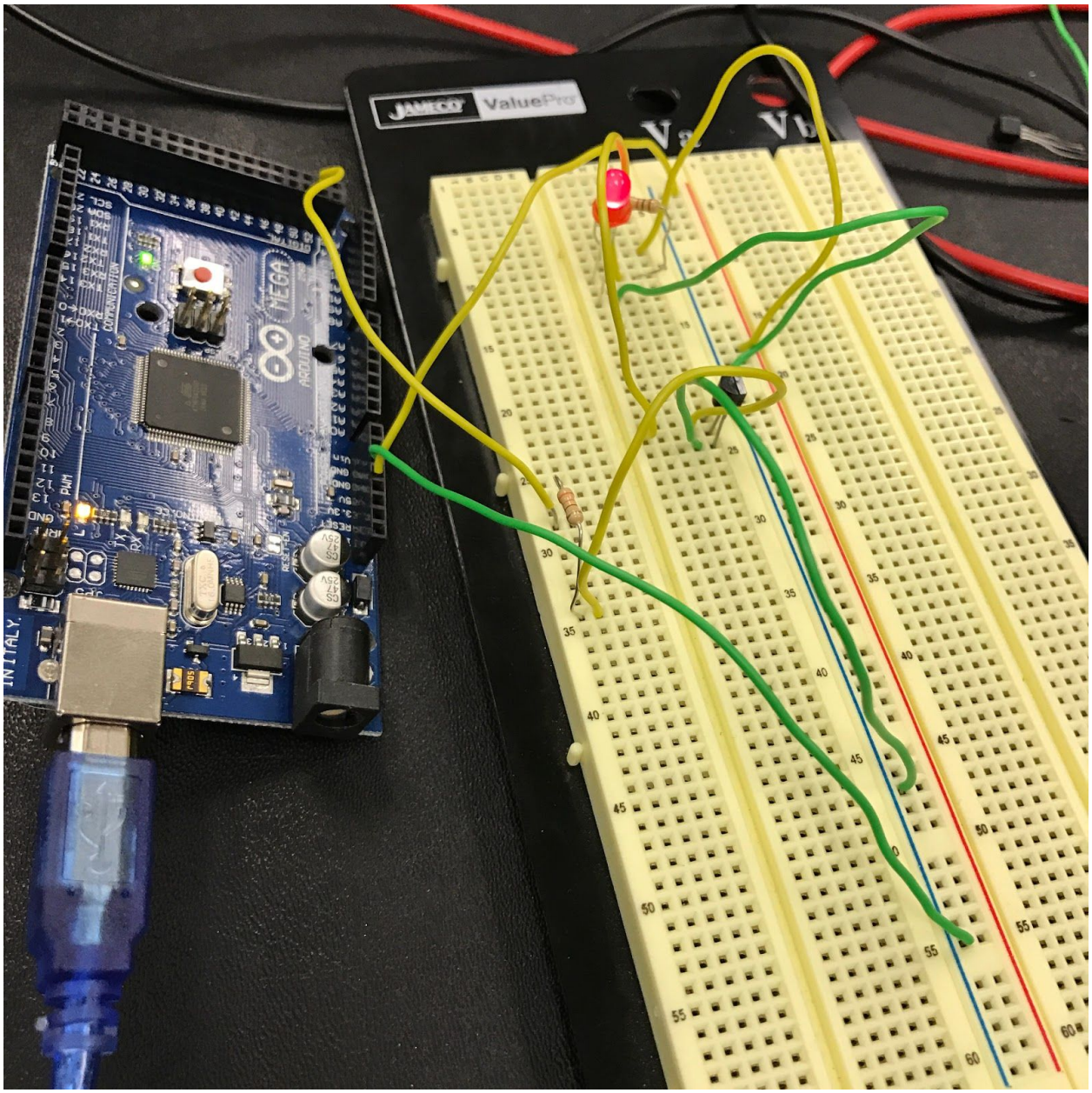
4. a. Arduino, 2 330 ohm resistors, LED, laptop, breadboard & wires, MOSFET
b. Attached.
c. a. The MOSFET is limiting the current supplied to the LED.
B. MOSFET switching is more efficient and provides another layer of protection for the board from shorting.

Photo:



5. a. Arduino, 2 resistors, LED, laptop, breadboard & wires, MOSFET
b. Attached.

Photo:



Part 3

1. a. Preprocessor definitions have the advantage of making clearer code that is not as reliant on magical numbers.

b. Macros avoid the overhead of making a function call, since they are just inline text substitution.

c.

```
void pinMode(uint8_t pin, uint8_t mode)
{
   uint8_t bit = digitalPinToBitMask(pin);      //specifies the mask to be used to select the bit
   uint8_t port = digitalPinToPort(pin);                //specifies the port where the pin is located
   volatile uint8_t *reg, *out;                         //declare variables

   if (port == NOT_A_PIN) return;               //if the port is not a pin nothing can be done, exit

   // JWS: can I let the optimizer do this?
   reg = portModeRegister(port);                //get the address of the DDR for the port
   out = portOutputRegister(port);              //get the address of the output register for the port

   if (mode == INPUT) {                         //if specifying as input pin...
            uint8_t oldSREG = SREG;             //save value of sreg
           cli();                               //swap interrupt flag
            *reg &= ~bit;                       //mask reg with complement of bit (&)
            *out &= ~bit;                       //mask out with complement of bit (&)
            SREG = oldSREG;                     //restore value of sreg
   } else if (mode == INPUT_PULLUP) {           //if specifying pullup resistor...
            uint8_t oldSREG = SREG;             //save value of sreg
           cli();                               //swap interrupt flag
            *reg &= ~bit;                       //mask reg with complement of bit (&)
            *out |= bit;                        //mask out with bit (|)
            SREG = oldSREG;                     restore value of sreg
   } else {                                     //otherwise...
            uint8_t oldSREG = SREG;             //save value of sreg
           cli();                               //swap interrupt flag
            *reg |= bit;                        //mask reg with bit (|)
            SREG = oldSREG;                     //restore value of sreg
   }
}
```

d.

```
void digitalWrite(uint8_t pin, uint8_t val)
{
   uint8_t timer = digitalPinToTimer(pin);      //get timer value for pin
   uint8_t bit = digitalPinToBitMask(pin);      //specify mask for bit selecting at pin
   uint8_t port = digitalPinToPort(pin);                //specify port where pin is located
   volatile uint8_t *out;                       //declare variable

   if (port == NOT_A_PIN) return;               //if not a pin, just exit

   // 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);    //as stated

   out = portOutputRegister(port);              //get output register for port

   uint8_t oldSREG = SREG;                      //save value of sreg
   cli();                                       //toggle interrupt flag

   if (val == LOW) {                            //if low, mask & with complement of bit
           *out &= ~bit;
   } else {                                     //otherwise, mask | with bit
           *out |= bit;
   }

   SREG = oldSREG;                              //restore value of sreg
}
```

e.

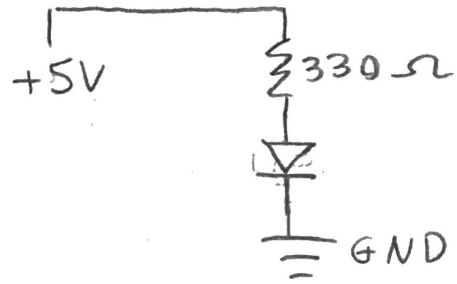
```
int digitalRead(uint8_t pin)
{
   uint8_t timer = digitalPinToTimer(pin);      //get timer value for pin
   uint8_t bit = digitalPinToBitMask(pin);      //specify mask for bit selecting at pin
   uint8_t port = digitalPinToPort(pin);                //specify port where pin is located

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

   if (*portInputRegister(port) & bit) return HIGH;         //read masked value to determine high...
   return LOW;                                              //or low
}
```

2. a. Setup gets called in main once after the program has been read in from the USB in the init process. Loop gets called afterwards inside a for loop that has no end condition, so it will repeat indefinitely.

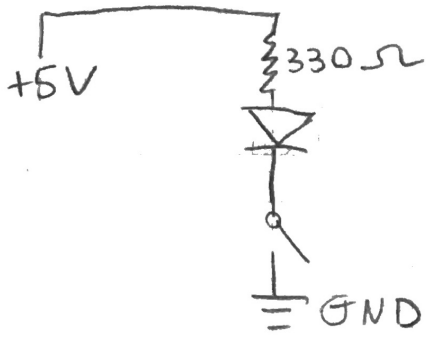
3. Arduino library functions are only to be used where explicitly specified. Assignments or exams containing Arduino library functions receive an automatic zero.

CPE 301 Lab 4 Circuits

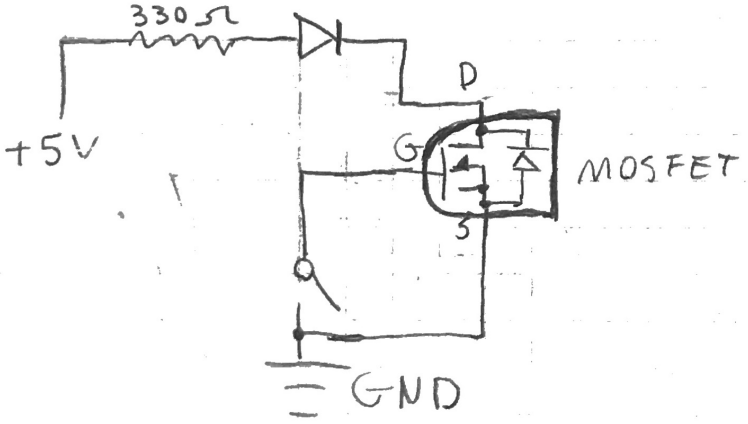
1.1



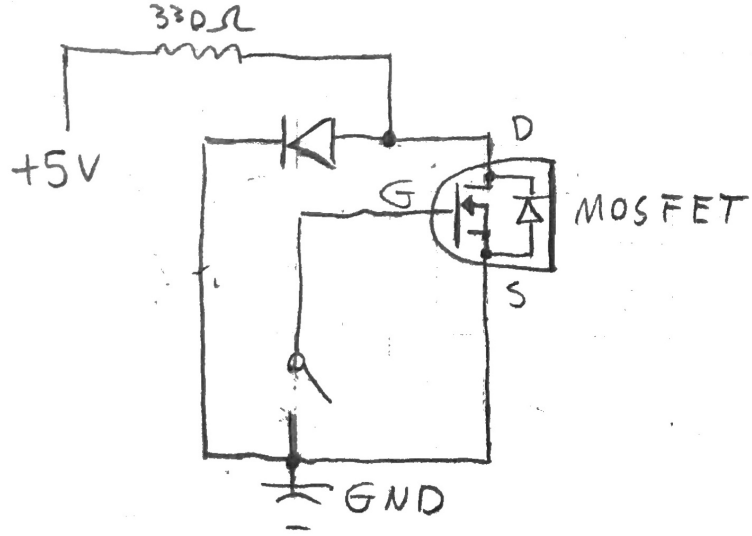
1.2



1.3

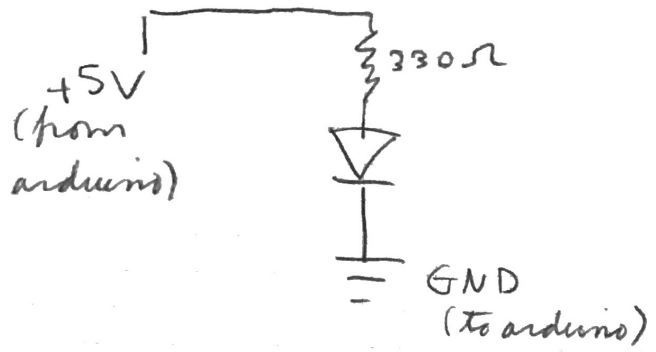


1.4

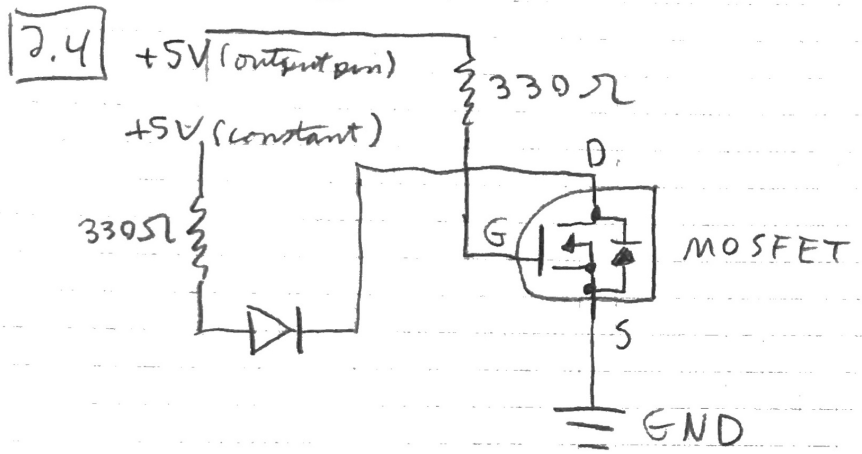


CPE 301 Lab 4 Exercises Cont

2.3



2.4



2.5