Firing the Solenoid Valve Using the Arduino

For the last part of the lab you should work on trying to control the pneumatic piston using the Arduino. In order to do so, you can follow these steps:

1. You should start with the pneumatic setup we worked on the last hour of the previous lab (see Figure 1). Inflate your tire to 30 psi using the bike pump. **Make sure the system does not have any leaks! Tighten up the push-to-connect fittings and the Schrader valve adapter using an adjustable wrench.**
2. The electronic connections schematic is given in Figure 2. In order to access the solenoid terminals, you will have to remove the transparent plastic cap. Note how the red wire (output 2B on the shield) is connected to the terminal where the LED is also connected (Figure 3).
3. Upload the code you will find at the end of this text to the Arduino.
4. The solenoid should now open and close, making the piston extend and retract. Feel free to change the parameters in the code in order to change how often the solenoid opens and closes.

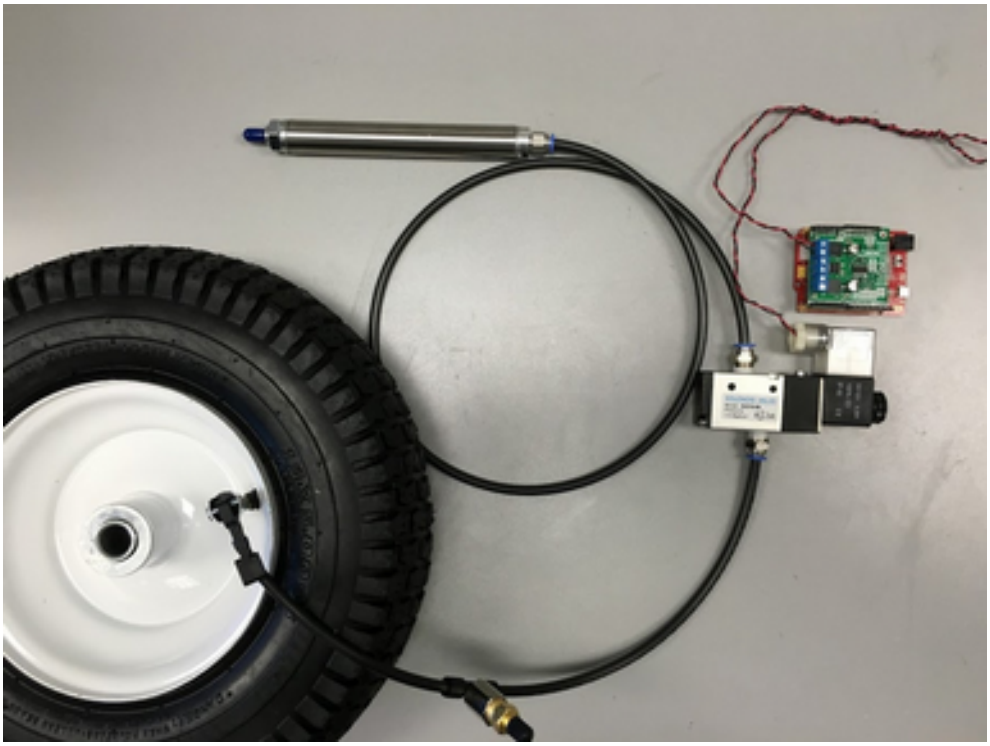

Figure 1: Pneumatic connections between: tire, T-connector, solenoid valve, and pneumatic piston. Electric connections between: solenoid valve and Arduino.

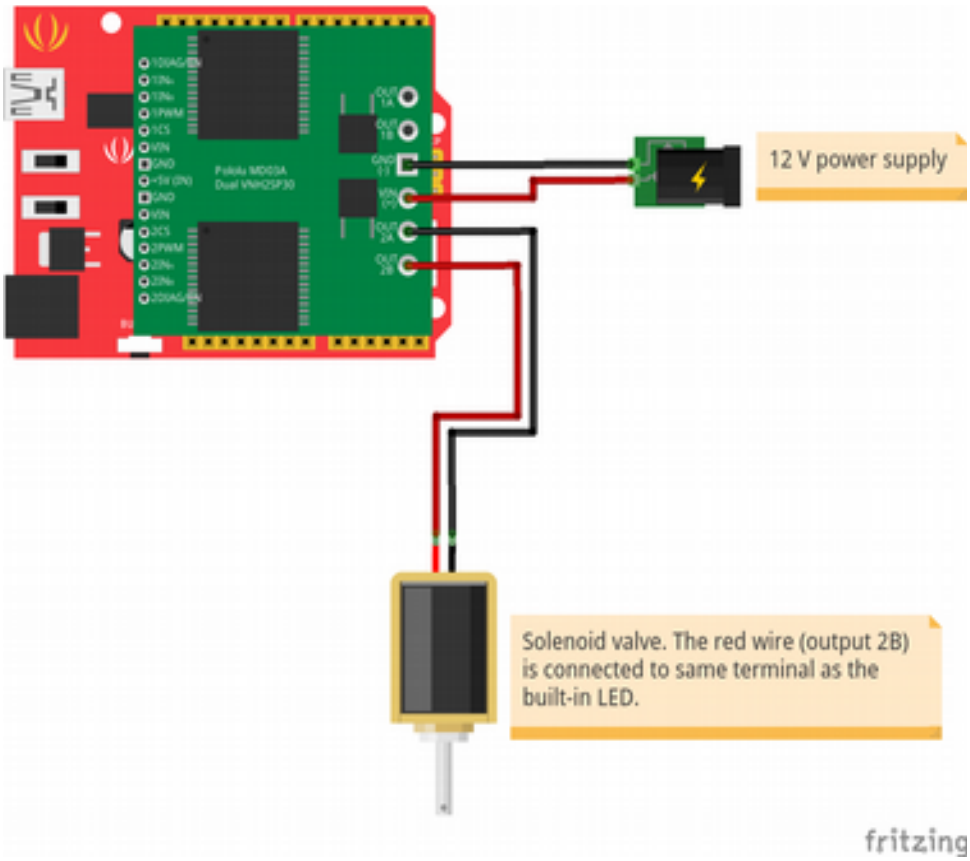

Figure 2: Connections between: power supply, Arduino and solenoid valve.

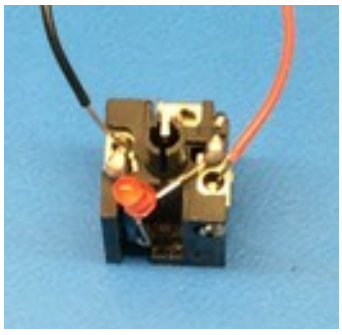

Figure 3: Solenoid terminals. Note how the red wire (output 2B on the shield) is connected to the terminal where the LED is connected.

Arduino Code

```
//-------------------------------------------------------------------------

/*
 * Code to trigger a solenoid using a Pololu Dual MC33926 motor shield. It
 * use motor port #2 on the shield.
 *
 * Author:  Joan Aguilar Mayans + Brendan Smith
 * Some parts are adapted from the BlinkWithoutDelay Arduino example created
 * by David A. Mellis and modified by Paul Stoffregen.
 * Date:   4/23/16
 */

//-------------------------------------------------------------------------
//----------------- Parameters (Change These) -----------------------------
//-------------------------------------------------------------------------

long onTime = 1000;  // Amount of time the piston is extended (milliseconds)
long offTime = 1000;  // Amount of time the piston is retracted (milliseconds)
int solenoidPolarity = HIGH;  // Either HIGH or LOW


//-------------------------------------------------------------------------
//-------------Initialize variable names (Don't Change These)--------------
//-------------------------------------------------------------------------


// Shield variables
const int D2 = 4;  // D2 pin, disables the shield if low
const int M2DIR = 8;  // Motor port #2 polarity
const int M2PWM = 10;  // Motor port #2 input (PWM)

// Time variables
long currentMillis;  // time in milliseconds
long previousMillis; // will store last time the solenoid state changed

// Solenoid state
int solenoidState = LOW;


//-------------------------------------------------------------------------
//-------------Setup() runs one time when the Arduino turns on---------------
//-------------------------------------------------------------------------

void setup() {

  // Enable shield
  pinMode(D2, OUTPUT);
  digitalWrite(D2, HIGH);

  // Choose solenoid polarity
  pinMode(M2DIR, OUTPUT);
  digitalWrite(M2DIR, solenoidPolarity);
```

```
  // Set the solenoid pin as output
  pinMode(M2PWM, OUTPUT);

  // Initialize time
  currentMillis = millis();
}


//-------------------------------------------------------------------------
//------After setup runs, this loop runs over and over forever--------------
//-------------------------------------------------------------------------

void loop() {

  // here is where you'd put code that needs to be running all the time.

  // Update time
  long currentMillis = millis();

  // check to see if it's time to trigger the solenoid, that is: check if the
  // piston has been extended for an amount of time longer than onTime or
  // retracted for an amount of time longer than offTime
  if ((currentMillis - previousMillis > onTime && solenoidState == HIGH) ||
(currentMillis - previousMillis > offTime && solenoidState == LOW)) {

    // save the last time the solenoid changed state
    previousMillis = currentMillis;

    // if the solenoid is closed, open it, and vice-versa:
    if (solenoidState == LOW) {
      solenoidState = HIGH;
    }
    else {
      solenoidState = LOW;
    }

    // Trigger the solenoid
    digitalWrite(M2PWM, solenoidState);
  }
}
```