

## ME4020: Machine Dynamics Lab

# Using Arduino to Control Machines

January 16, 2018

**CAUTION!!!!** This machine has a powerful motor and pinch points. Do not reach into the apparatus. Be sure to unplug the device while not in use.

## 1 Introduction

In this lab, a machine that uses a brushed electric motor actuates a window regulator mechanism from a Toyota Corolla. The regulator uses a worm gear connected to a crank. The crank then has a slider that acts like a scissor mechanism. To change the direction of the motor, an H-Bridge can be used (See [https://en.wikipedia.org/wiki/H\\_bridge](https://en.wikipedia.org/wiki/H_bridge)).

The lab apparatus has two H-Bridge Circuits where each bridge is selected by a toggle switch.

**Mechanical:** A single Double Pole - Double Throw (DPDT) momentary switch is used to reverse the current flowing through the motor.

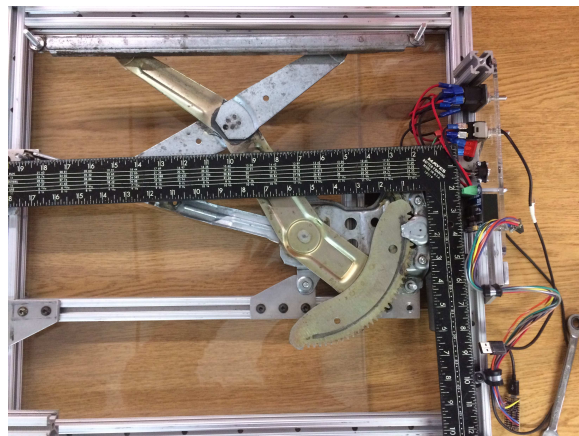


Figure 1: Underneath of Lab Apparatus

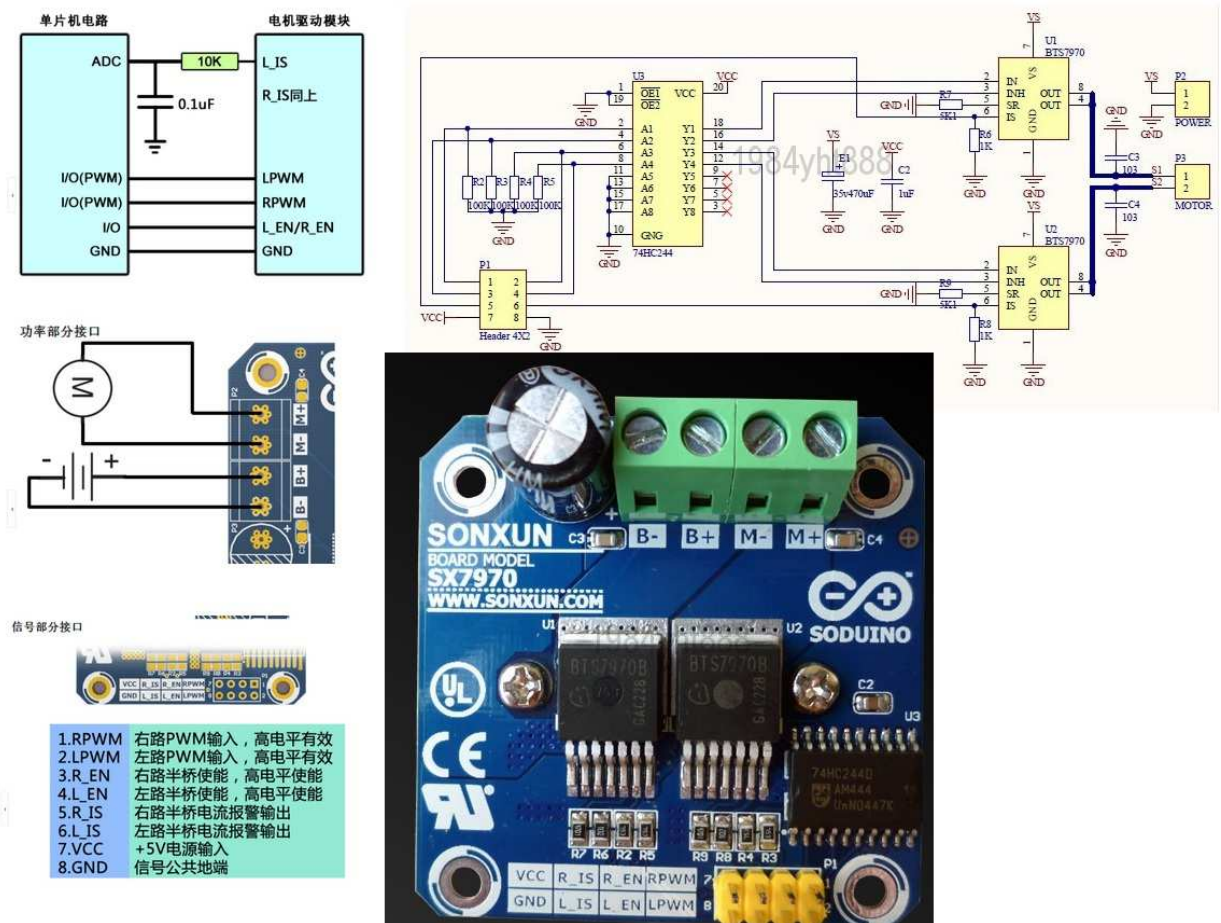


Figure 2: Documentation for the electronic H-Bridge

**Electronic:** Two Infineon BTS7970 Half bridge Integrated Circuits are used in an H-Bridge module as shown in Fig. 2.

## 1.1 Hardware

The wires for the lab are connected according to Table 1

The small circuit board is a Teensy 3.2 USB development board that supports the Arduino programming environment. It is a powerful 32-bit ARM Cortex processor.

The wiring has already been assembled and soldered. If creating a new design, you would be responsible for laying out the circuit.

# Welcome to Teensy 3.2

32 Bit Arduino-Compatible Microcontroller

To begin using Teensy, please visit the website & click [Getting Started](#).

[www.pjrc.com/teensy](http://www.pjrc.com/teensy)

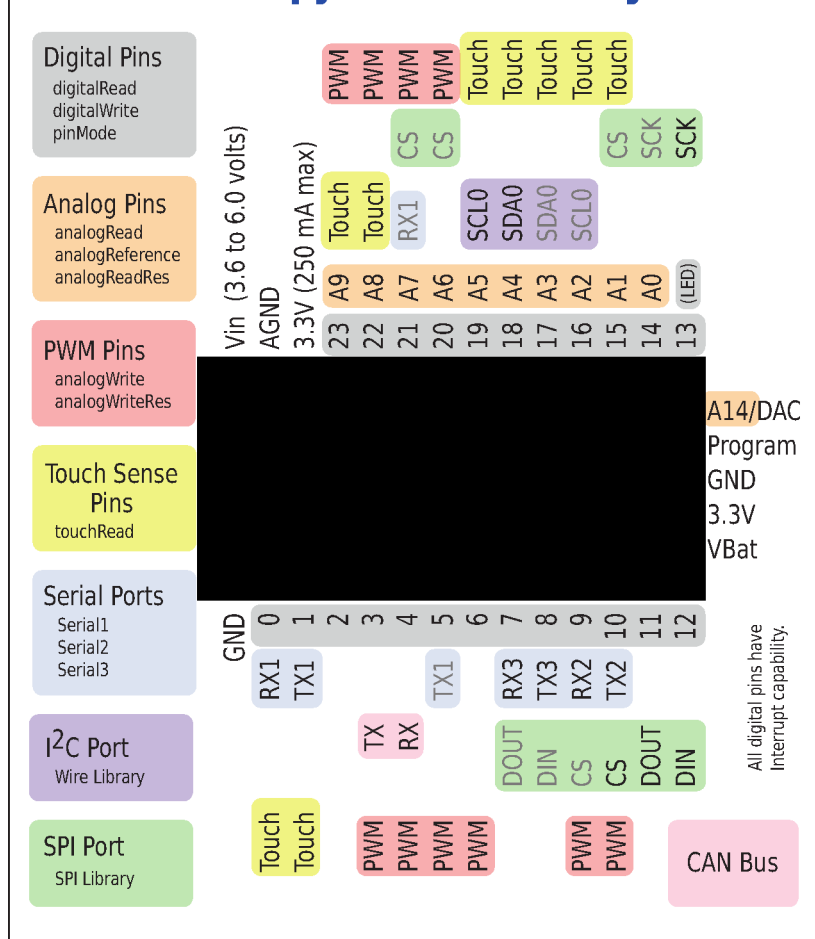
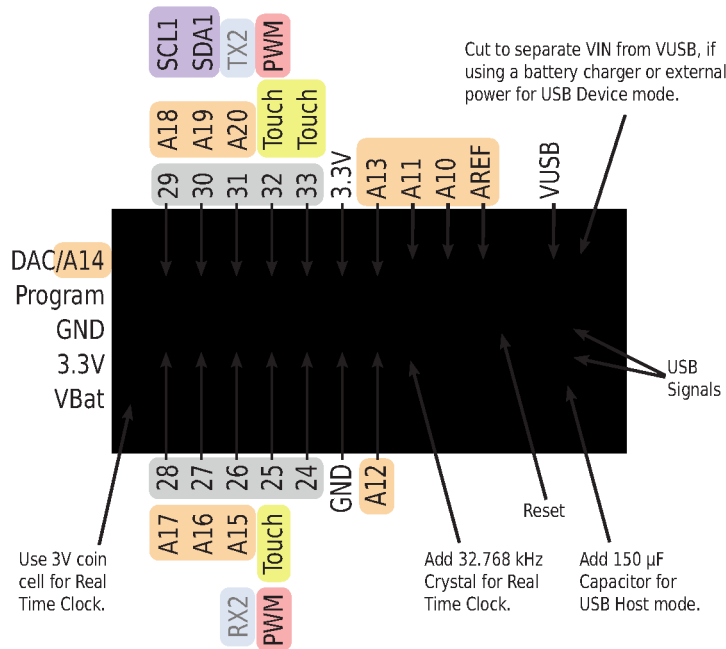


Figure 3: Teensy 3.2 Pinout - Front Side

# Teensy 3.2 Back Side

Additional pins and features available on the back side



For solutions to the most common issues and technical support, please visit:

[www.pjrc.com/help](http://www.pjrc.com/help)

Teensy 3.2 System Requirements:  
 PC computer with Windows 7, 8, 10 or later  
 or Ubuntu Linux 12.04 or later  
 or Macintosh OS-X 10.7 or later  
 USB Micro-B Cable

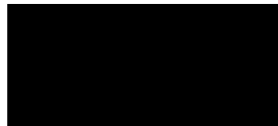


Figure 4: Teensy 3.2 Pinout - Front Side

Number	H Bridge Cir	Description	Teensy 3.2	Color
1	R_PWM	Right Pulse Width Modulated Signal	10	Red
2	L_PWM	Left Pulse Width Modulated Signal	20	Gray
3	R_EN	Right Bridge Enable	23	Tan
4	L_EN	Left Bridge Enable	9	Blue/White
5	R_IS	Current Sense	22	Purple
6	L_IS	Current Sense	21	Green
7	Vcc	Power	3.3V	Orange
8	Ground			Yellow
		Up Limit Switch	14	Blue
		Down Limit Switch	16	Brown
		Push Button Switch	18	Black
		Switch Common	GND	Blue

Table 1: Wiring Harness Description

## 1.2 Arduino Programming Environment

The Teensy 3.2 can be programmed using the Arduino platform with the Teensyduino add-on. To check the software, open the Arduino program. Under Tools, select Boards and select the Teensy 3.2 as shown in Figure 5. If this check fails, be sure to install the Teensyduino software by following the instructions at <https://www.pjrc.com/teensy/teensyduino.html>.

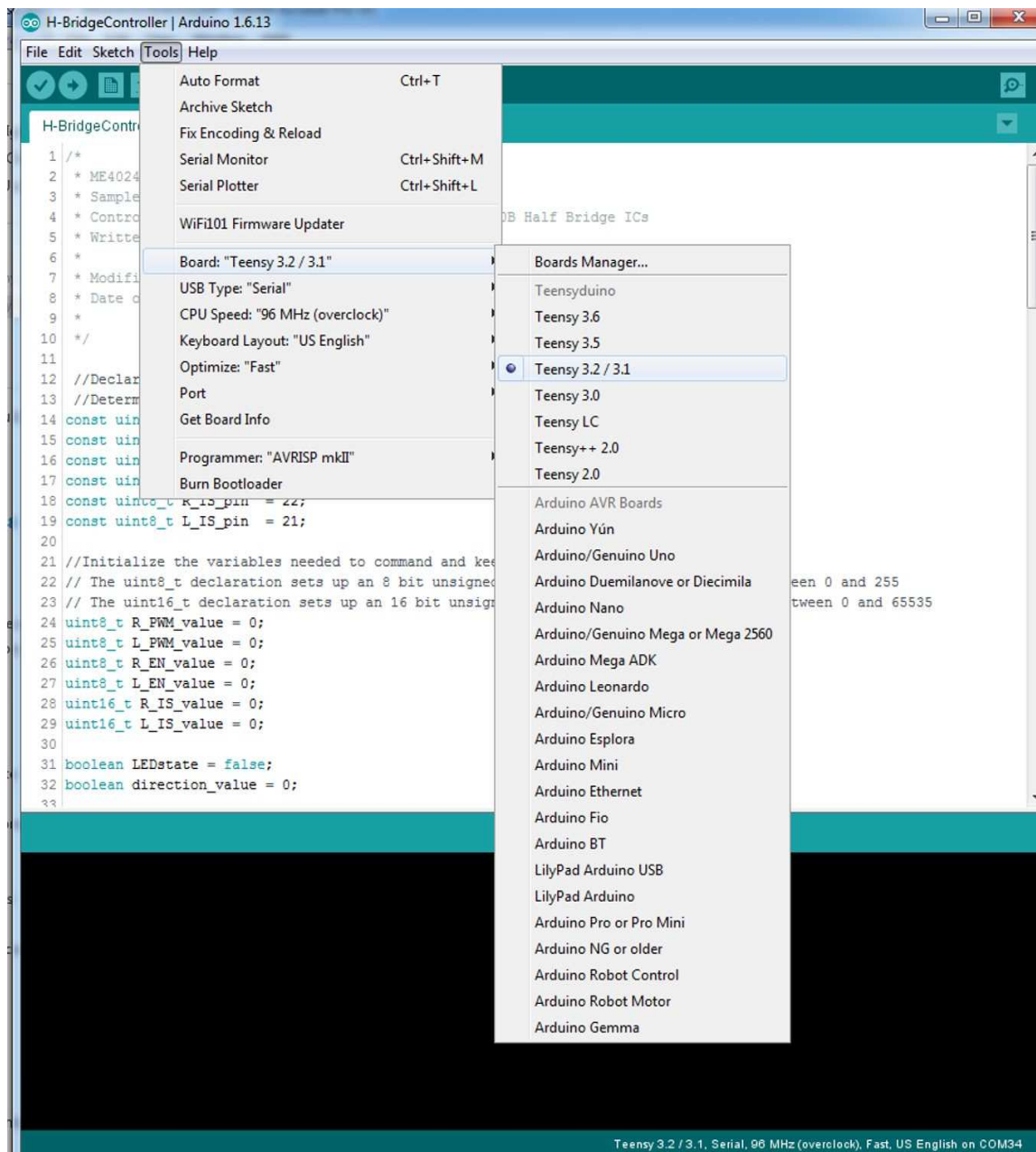


Figure 5: Checking for Teensy 3.2 Support.

You can turn on line numbers by selecting File -> Preferences.

### 1.3 Structure of an Arduino Sketch

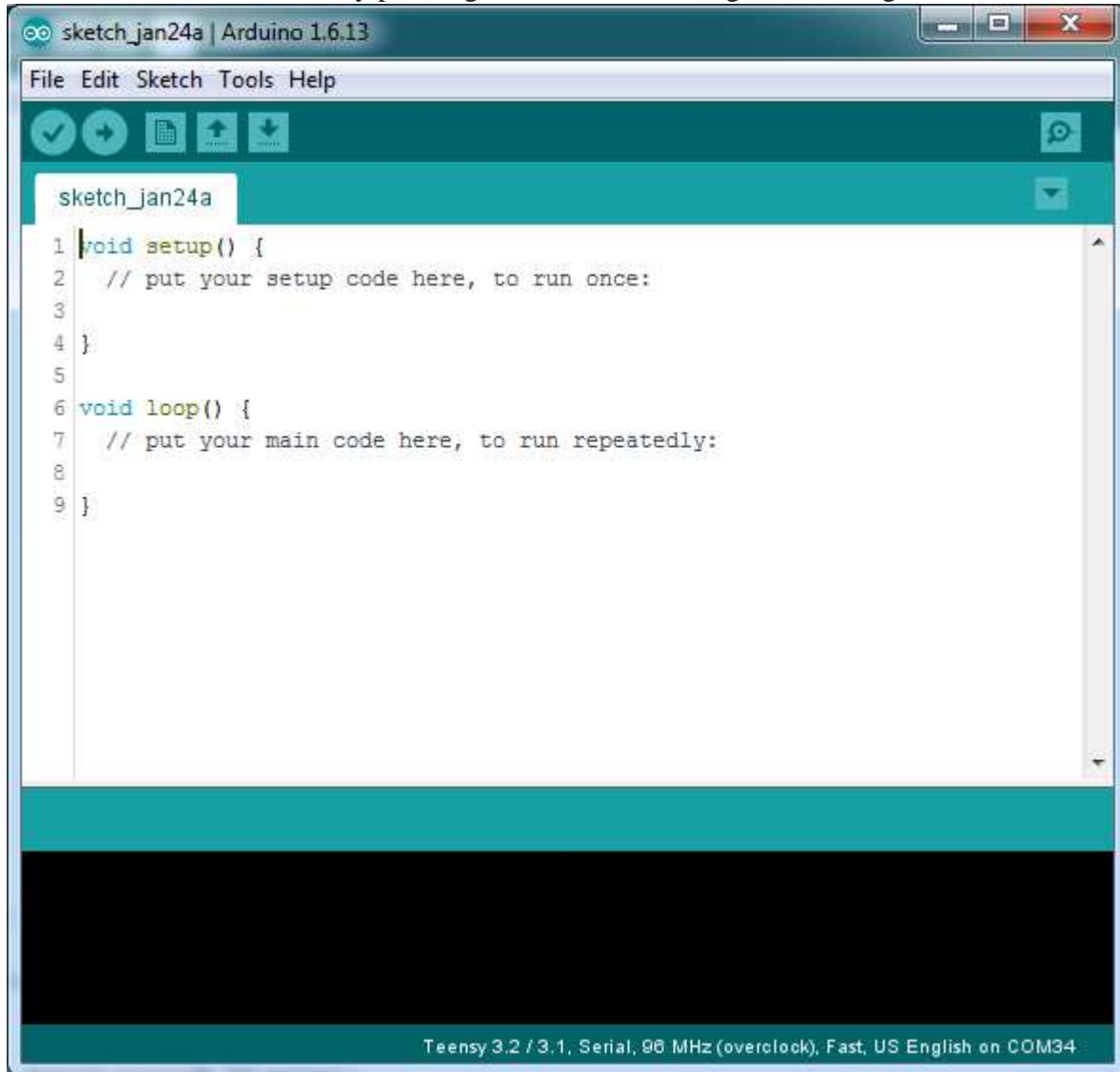
The program written for an Arduino usually has the following parts:

1. Header comment block explaining the program and requirements
2. Library headers to include if expanded support is needed. These can be libraries to interface with peripheral like a CAN bus, SPI sensor, or real-time clock.
3. Declarations of variables. In C and Arduino, the variable name and type must be declared before using it. We commonly use 8, 16, and 32-bit numbers as well as characters and boolean (1 bit) variables.
4. A `setup()` function that runs once. This function sets modes and initializes systems.
5. A `loop()` function that is the main execution loop that repeats over and over.
6. Other functions can be declared and called from another function.

Once a sketch is written, it can be compiled and uploaded to the Teensy by pressing Ctrl-U or clicking on the right arrow.

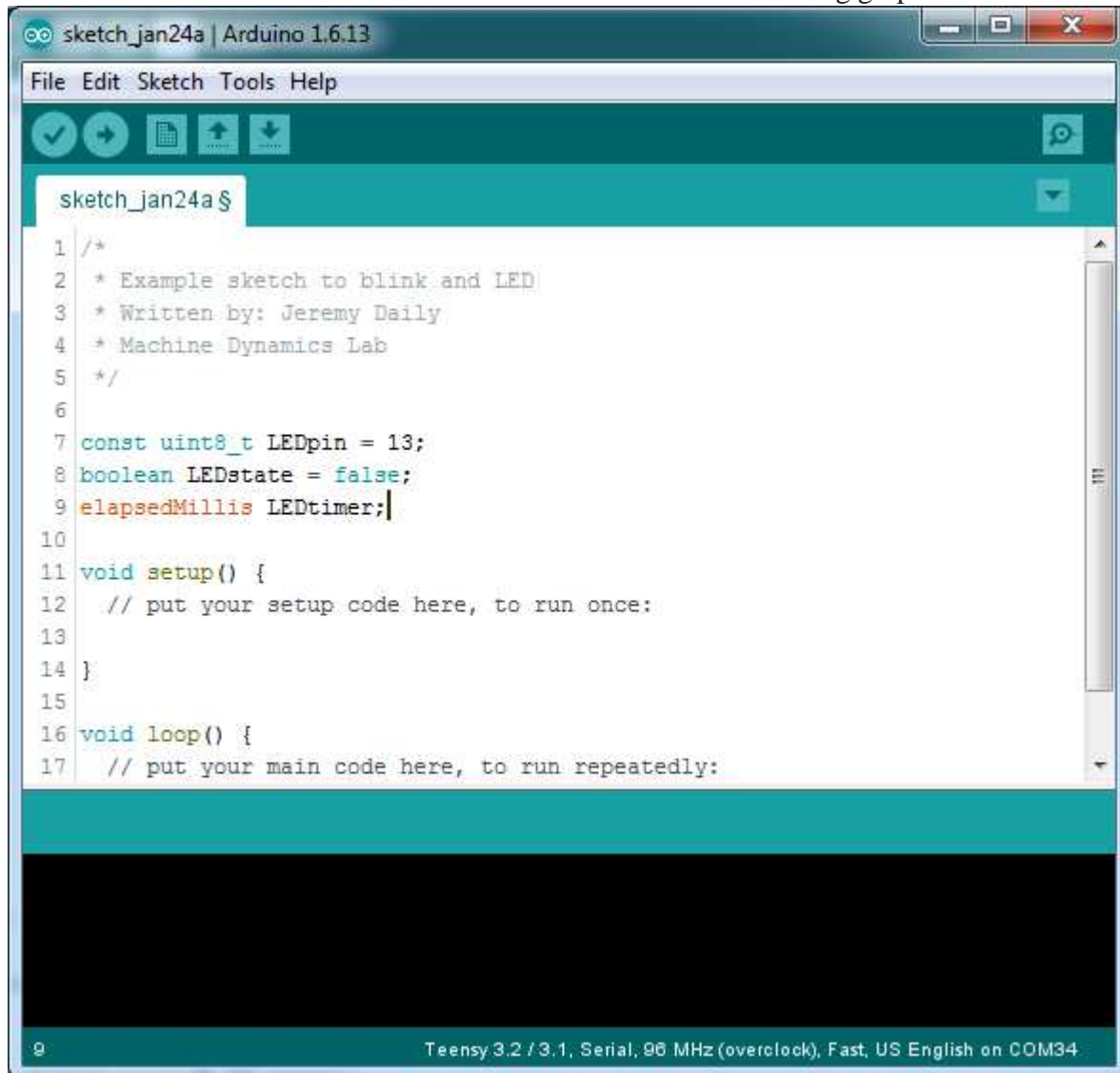
## 1.4 Example Sketch

Create a new Arduino Sketch by pressing Ctrl-N. You should get something that looks like this:

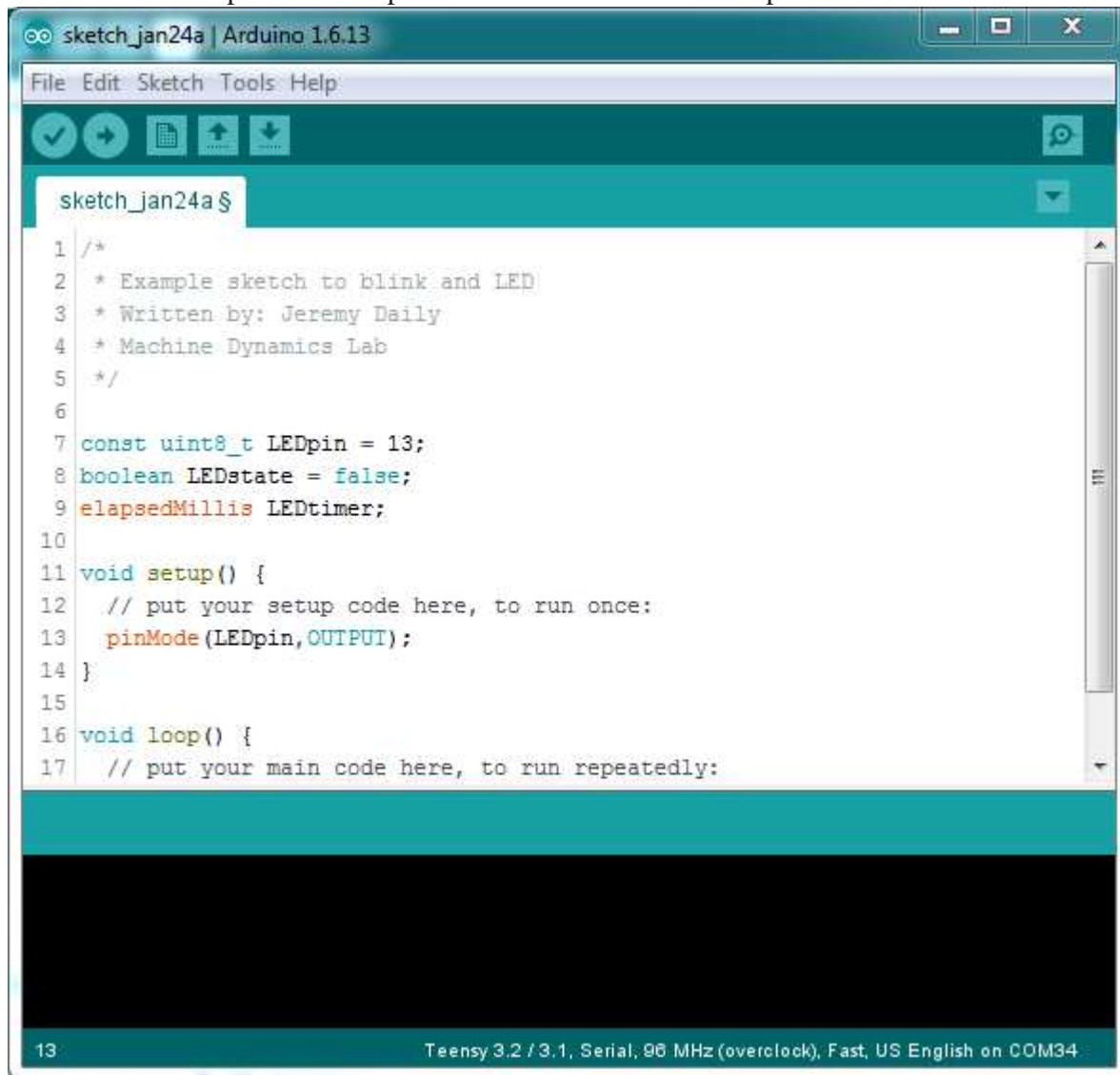




Let's make the LED blink every tenth of a second. So we need to start by creating a timer and declaring the LED pin. Based on the pinouts for the Teensy 3.2 shown in Figure 3, the LED is pin 13. Let's add a header and declare the pin number, setup a variable to keep track of the LED state, and establish a timer. These new elements are shown in the following graphic:

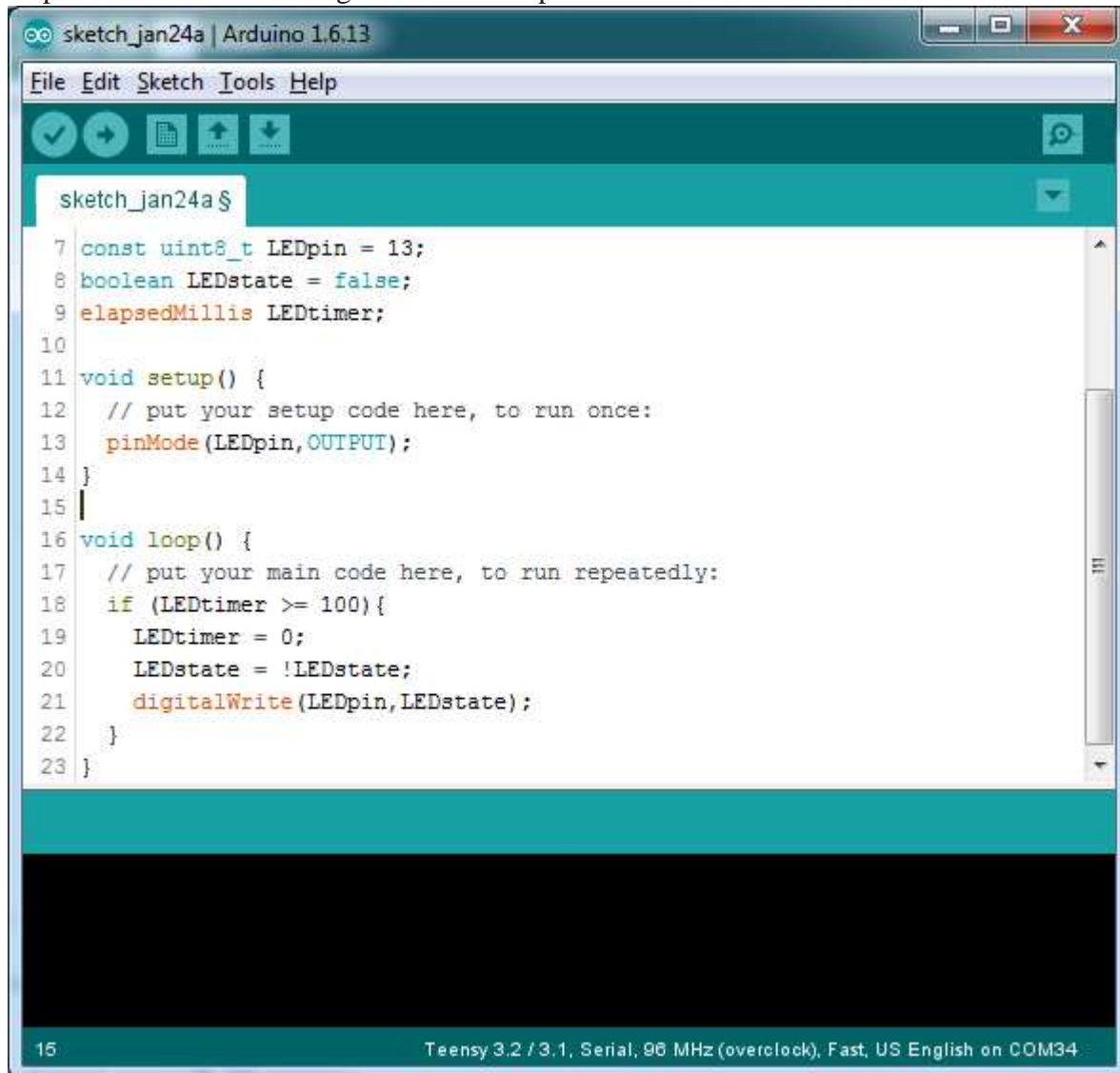


Keep in mind that all commands are terminated by a semicolon. Next we have to set the pin mode of the LED pin to let the processor know that it is an output.

A screenshot of the Arduino IDE interface. The window title is "sketch\_jan24a | Arduino 1.6.13". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, and uploading. The main text area shows a C++ sketch for blinking an LED. The sketch includes comments, variable declarations for LEDpin, LEDstate, and LEDtimer, and functions for setup and loop. The status bar at the bottom indicates "13" and "Teensy 3.2 / 3.1, Serial, 96 MHz (overclock), Fast, US English on COM34".

```
1 /*
2  * Example sketch to blink and LED
3  * Written by: Jeremy Daily
4  * Machine Dynamics Lab
5  */
6
7 const uint8_t LEDpin = 13;
8 boolean LEDstate = false;
9 elapsedMillis LEDtimer;
10
11 void setup() {
12   // put your setup code here, to run once:
13   pinMode(LEDpin, OUTPUT);
14 }
15
16 void loop() {
17   // put your main code here, to run repeatedly:
```

Now that the variables are declared and the LED pin is set up to be an output, we need to repeatedly check the timer and see if it has passed the limit. The elapsed millis timer keeps a running count of the number of milliseconds using a 32-bit register. This means the timer will rollover (go back to zero) after  $2^{32}$  micro seconds, or 4294967.296 seconds, which is about 1193 hours. However, to repeatedly use the timer, we will reset it to zero after each time period has elapsed. Enter the following lines in the loop function:

The image is a screenshot of the Arduino IDE interface. The title bar at the top reads 'sketch\_jan24a | Arduino 1.6.13'. Below the title bar is a menu bar with 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Underneath the menu bar is a toolbar with icons for checking, running, saving, uploading, and downloading. The main text area contains the following C++ code:

```
sketch_jan24a $
7  const uint8_t LEDpin = 13;
8  boolean LEDstate = false;
9  elapsedMillis LEDtimer;
10
11 void setup() {
12   // put your setup code here, to run once:
13   pinMode(LEDpin, OUTPUT);
14 }
15
16 void loop() {
17   // put your main code here, to run repeatedly:
18   if (LEDtimer >= 100){
19     LEDtimer = 0;
20     LEDstate = !LEDstate;
21     digitalWrite(LEDpin, LEDstate);
22   }
23 }
```

The status bar at the bottom of the window displays '15' on the left and 'Teensy 3.2 / 3.1, Serial, 96 MHz (overclock), Fast, US English on COM34' on the right.

This code checks to see if the timer has overrun. It is a good idea to use an inequality in case a command in the loop takes longer than 1 millisecond. The parentheses are used to distinguish the conditional statement and the curly braces are used to tell the program what to run for that condition. Additional else if and else statements can be used. The unary operator of ! is used to say “not” so the statement on line 20 switches the state of the boolean variable. The updated state is pushed to

as the pin value in line 21.

This program is ready to save and compile. Press Ctrl-U to automatically save and upload. Observe the blinking LED.

## 2 Lab Procedure

1. Draw a schematic of an H-Bridge using a DPDT switch. Understand how it reverses current through a motor.
2. Ensure the test apparatus is plugged in using a 12VDC power supply. The power supply should provide 8 amps or more.
3. Place the selector switch to the Mechanical setting. This routes the current through the mechanical H-Bridge for motor direction control.
4. Test the motor out to see the mechanism work. Identify the two screw holes (Near and Far) on the moving cross member. These threaded holes used to accept screws to attach a window on the driver's door of a Toyota Corolla. If it doesn't work. Try tapping the motor with a hard object.
5. For every turn of the motor, estimate the (x,y) coordinates of each screw hole. Tabulate these estimates on a separate sheet of paper.
6. Switch the selector to Electronic to engage the Teensy controlled motor controller.
7. Download and run the example sketch from Harvey.
8. While the program is running, open the Serial Plotter by pressing Ctrl-Shift-L. Take a screen shot of the output. Notice the PWM signal going to the motor ramps up.
9. Change the code to enable the square wave input by changing the lines around 122 to say (Swap comments to enable speed square wave):

```
// Set the speed value each loop
// speed_value = speedTimer*0.2 * ( -1 + 2*direction_value ); //Ramp soft
speed_value = 180 * ( -1 + 2*direction_value ); // Full up or full down (
// speed_value = 180 * sin(0.001*millis()); // Harmonic
speed_value = constrain( speed_value , -180,180);
```

Est. X, Near	Est. Y, Near	Est. X, Far	Est. Y, Far
	-3		
	-2		
	1		
	0		
	1		
	2		
	3		
	4		
	5		
	6		
	7		
	8		

Table 2: Estimated position of threaded holes.

Description	X Location	Y location
Location of the gear motor axis		
Location of the base arm main pivot (O)		
Location of the bolt at the end of the fixed slider		

Table 3: Estimated locations of fixed points.

10. Again, while the program is running, open the Serial Plotter by pressing Ctrl-Shift-L. Take a screen shot of the output. Notice the high current draw at the beginning of the motion.
11. Change the source code to have the motor move up and down in a harmonic sense. That is use the function  $255 \cdot \sin(\text{freq} \cdot \text{millis}() \cdot 0.001)$  to set the speed to change at a frequency of 1 Hz.

```
// Set the speed value each loop
// speed_value = speedTimer*0.2 * ( -1 + 2*direction_value ); // Ramp soft
// speed_value = 180 * ( -1 + 2*direction_value ); // Full up or full down
speed_value = 180 * sin(0.001*millis()); // Harmonic
speed_value = constrain(speed_value, -180, 180);
```

12. Again, while the program is running, open the Serial Plotter by pressing Ctrl-Shift-L. Take a screen shot of the output. Notice the smooth current draw

Description	Estimated Value	Units
Number of teeth on the spur gear driven by the worm		
Center to center distance from the gear motor to the main pivot (O)		
Distance from Point D to the near hole (N)		
Distance from Point D to the far hole (F)		
Distance from O to A		
Distance from A to C		
Angle between OA and AC		
Distance from A to B		
Distance from A to D		
Angle of O to P		
Radius of the gear sector attached to the crank (link OAC)		

Table 4: Estimated Values for the mechanism description.

## 3 Mechanism Analysis

### 3.1 Theory of Mechanisms

Analyzing the mechanism utilizes the so-called loop closure equations. There are 2 loops in this scissor mechanism: loop OAB and loop ACD. Let's define the following vectors:

$\vec{r}_1$  is from O to B and points in the direction of  $\theta_1$

$\vec{r}_2$  is from B to A and points in the direction of  $\theta_2$

$\vec{r}_3$  is from O to A and points in the direction of  $\theta_3$

Similarly for the top loop.

$\vec{r}_4$  is from A to D and points in the direction of  $\theta_4$

$\vec{r}_5$  is from A to C and points in the direction of  $\theta_5$

$\vec{r}_6$  is from D to C and points in the direction of  $\theta_6$

Introduce two more vectors that describe the position of N and F with respect to point D:

$\vec{r}_{N/D}$  is from D to the near side bolt and has a direction of  $\theta_6 + 180^\circ$

$\vec{r}_{F/D}$  is from D to the far side bolt and has a direction of  $\theta_6$

Now, let's write the loop closure equations:

$$\vec{r}_1 + \vec{r}_2 = \vec{r}_3$$

and

$$\vec{r}_5 + \vec{r}_6 = \vec{r}_4$$

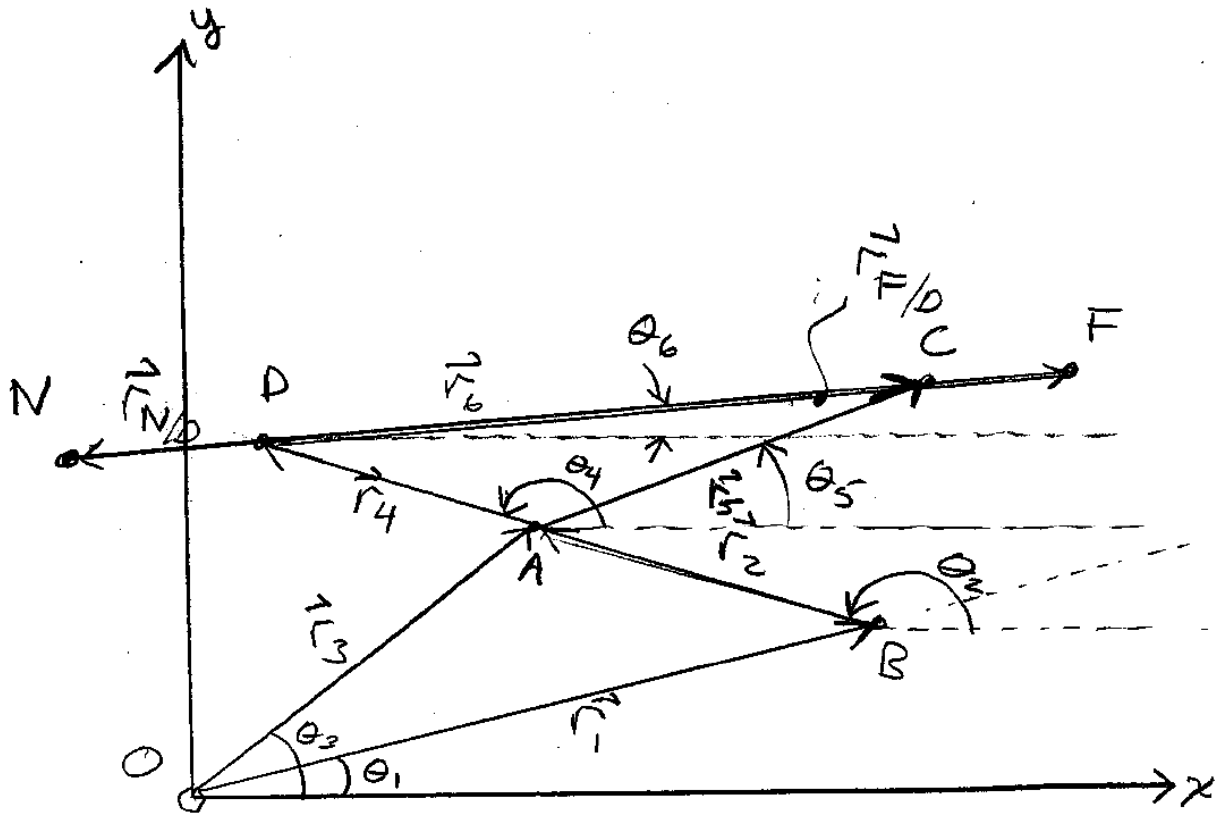


Figure 6: Skeleton diagram of the window regulator mechanism.

These are vector equations in the X-Y system and can be broken into components:

$$r_1 \cos \theta_1 + r_2 \cos \theta_2 = r_3 \cos \theta_3 \quad (1)$$

$$r_1 \sin \theta_1 + r_2 \sin \theta_2 = r_3 \sin \theta_3 \quad (2)$$

$$r_4 \cos \theta_4 + r_6 \cos \theta_6 = r_5 \cos \theta_5 \quad (3)$$

$$r_4 \sin \theta_4 + r_6 \sin \theta_6 = r_5 \sin \theta_5 \quad (4)$$

For the bottom loop, the input angle is  $\theta_3$  because it is driven by the sector gear. The input for the top loop is  $\theta_5$ , which is equal to  $\theta_3$  plus a constant offset. Note: the offset may be a negative number and needs to be determined by assessing the mechanism.

If we square equation 1, square equation 2 and add them together after rearranging, we get

$$(r_1 \cos \theta_1 - r_3 \cos \theta_3)^2 + (r_1 \sin \theta_1 - r_3 \sin \theta_3)^2 = (r_2 \cos \theta_2)^2 + (r_2 \sin \theta_2)^2$$

which eliminates  $\theta_2$  using the Pythagorean trigonometric identity. Now you can solve for  $r_1$ .

We can also set up the equations to solve for  $\theta_3$  by rearranging and dividing equation 2 by equation 1:

$$\frac{r_2 \sin \theta_2}{r_2 \cos \theta_2} = \frac{r_3 \sin \theta_3 - r_1 \sin \theta_1}{r_3 \cos \theta_3 - r_1 \cos \theta_1}$$

$$\theta_2 = \text{atan2} \left( \frac{r_3 \sin \theta_3 - r_1 \sin \theta_1}{r_3 \cos \theta_3 - r_1 \cos \theta_1} \right)$$

Be sure to use the two argument arc tangent to capture solutions that are between 90 degrees and 270 degrees.

You will have to set up similar equations for the top loop and solve for the unknown angles and distances.

Once the loop closure equations are solved, you can define a vector to each point of interest. For example

$$\vec{r}_N = \vec{r}_3 + \vec{r}_4 + \vec{r}_{N/D}$$

$$\vec{r}_F = \vec{r}_3 + \vec{r}_4 + \vec{r}_{F/D}$$

You will have to break these vectors into components to arrive at a set of points that you can plot.



## 3.2 Analysis Procedure

1. Estimate the gear ratio between the motor and crank. There are 2 gear interfaces: 1) worm gear and 2) the crank gear.
  - a) Count teeth on the spur gear following the worm to determine its ratio.
  - b) Determine the crank gear ratio. Since the small gear is hidden, you have to estimate its radius by subtracting the pitch radius (radius to about the center of the teeth) from the center to center distance. Photographs are available for measurements.
2. Plot the positions of the mechanism on a graph using discrete points.
  - a) The graph must have an equal scale for both the X and Y axis.
  - b) The location of the mechanism pivot point must be plotted and identified.
  - c) The near side threaded hole estimates should be plotted with an open circle.
  - d) The far side threaded hole estimates should be plotted with a triangle.
3. Write a computer program to determine the (x,y) locations of each point (A, B, C, D, N, and F) through the range of travel.
  - a) Plot the locus of points for the near end hole (N) as a red line on the same graph as the estimated points.
  - b) Plot the locus of points for the far end hole (F) as a blue line on the same graph as the estimated points.

## 4 Reporting Requirements

1. Hand drawn schematic of an H-Bridge with a DPDT switch.
2. Table 2 with hand written values.
3. Table 3 with hand written values.
4. Table 4 with hand written values.
5. A hand written gear ratio estimation. Be sure to include the worm gear ratio and the sector gear ratio.

6. Graph of the estimated/measured and predicted positions of the near and far holes on the same graph.
  - a) Plot the estimated positions based on Table 2 as discrete points.
  - b) Plot the calculated positions as smooth curves based on the loop closure analysis.
7. Screenshot of the Serial Plotter for the ramped speed command.
8. Screenshot of the Serial Plotter for the square speed command.
9. Screenshot of the Serial Plotter for the harmonic speed command.

Be sure to include a cover sheet with your name and the date you completed the lab. Staple all sheets together.