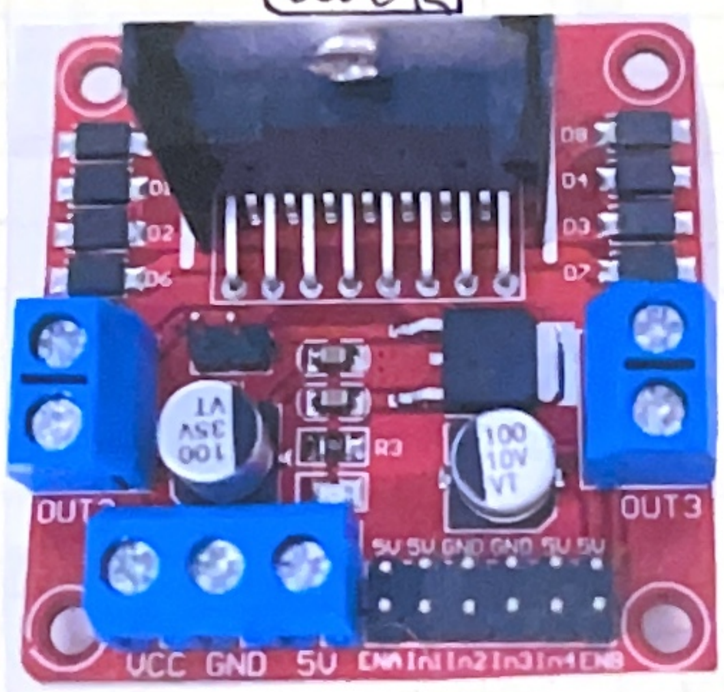
Objective

Make the code easier to understand and maintain/modify.

To explore ways of removing the "arcs" between waypoints, the land rover ~~of~~ version of WIL will be used. I am atempting to write some libraries to ① learn how libraries work and ② make the code easier to read and modify in the future.

back



Front

Figure 1

This a is an L298N h-bridge. It can control two different motors. One on the left and right.

ENA requires a PWM signal and will adjust the speed of the motor connected to Out 2.

IN1 and IN2 will control which way the current flows, causing the motor to spin CW or CCW. These are digital pins.

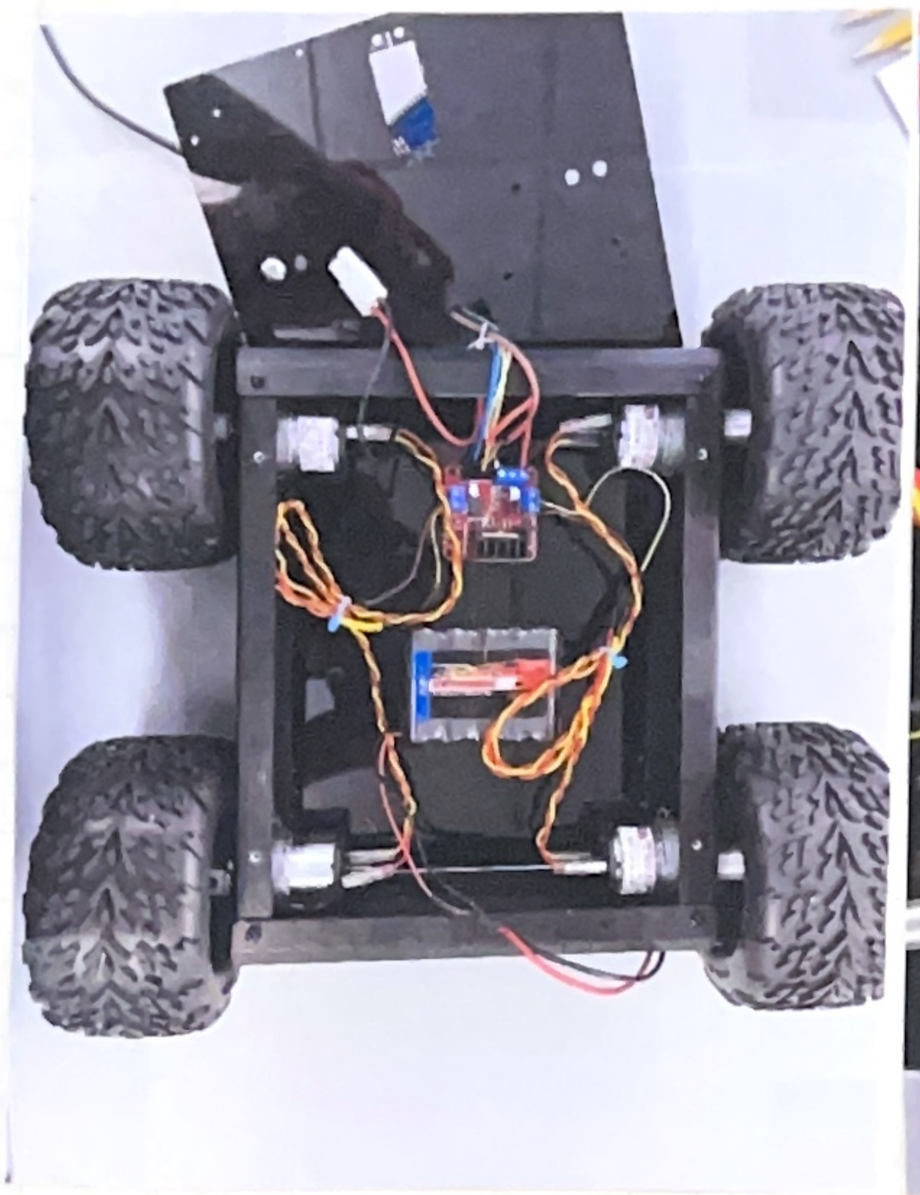
For example:

IN1	IN2	Direction
LOW	LOW	None
LOW	HIGH	CW
HIGH	LOW	CCW
HIGH	HIGH	None

Depending on the order of IN1, IN2, and which way the motors are wired into OUT 3 may affect the direction.

Writing an H-Bridge Library

Dec 2023
Wesley Loeb



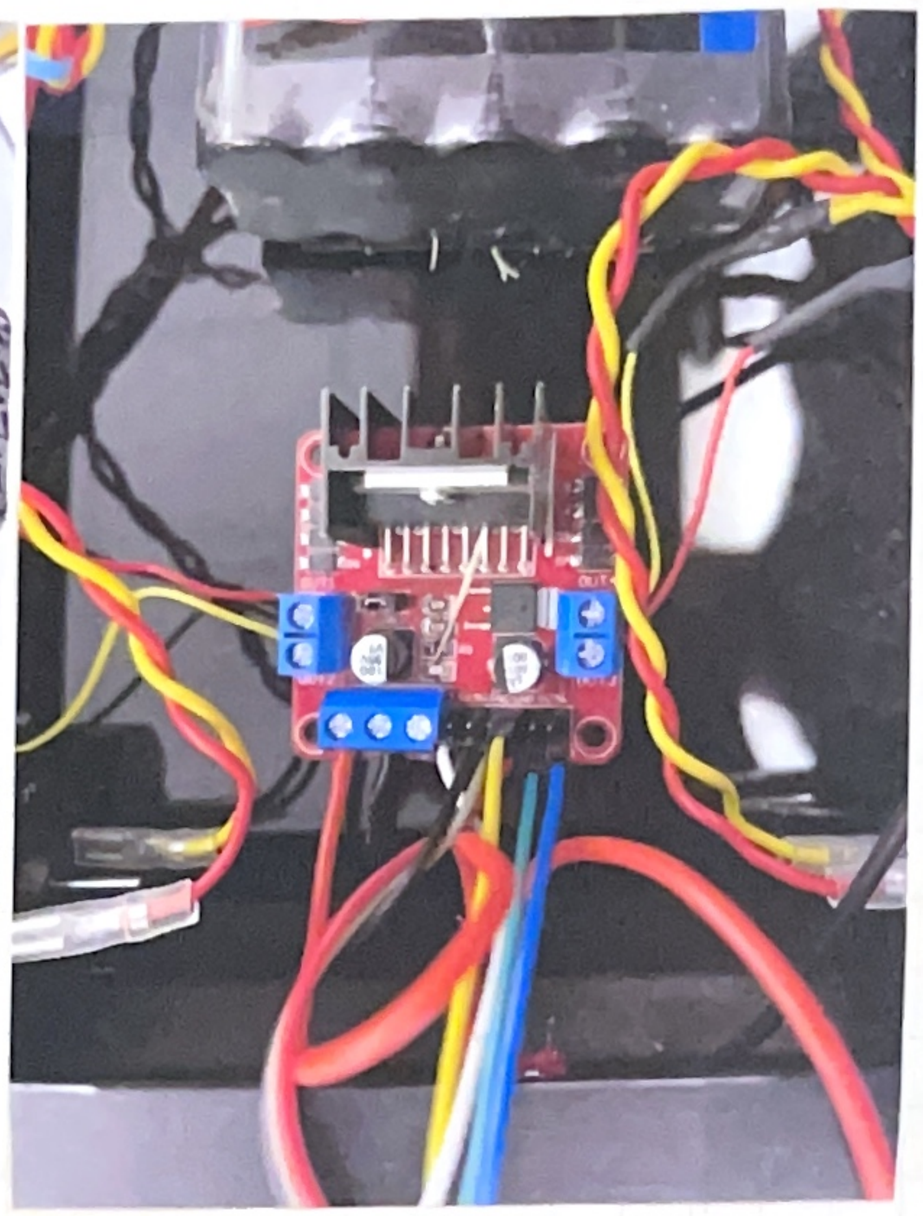
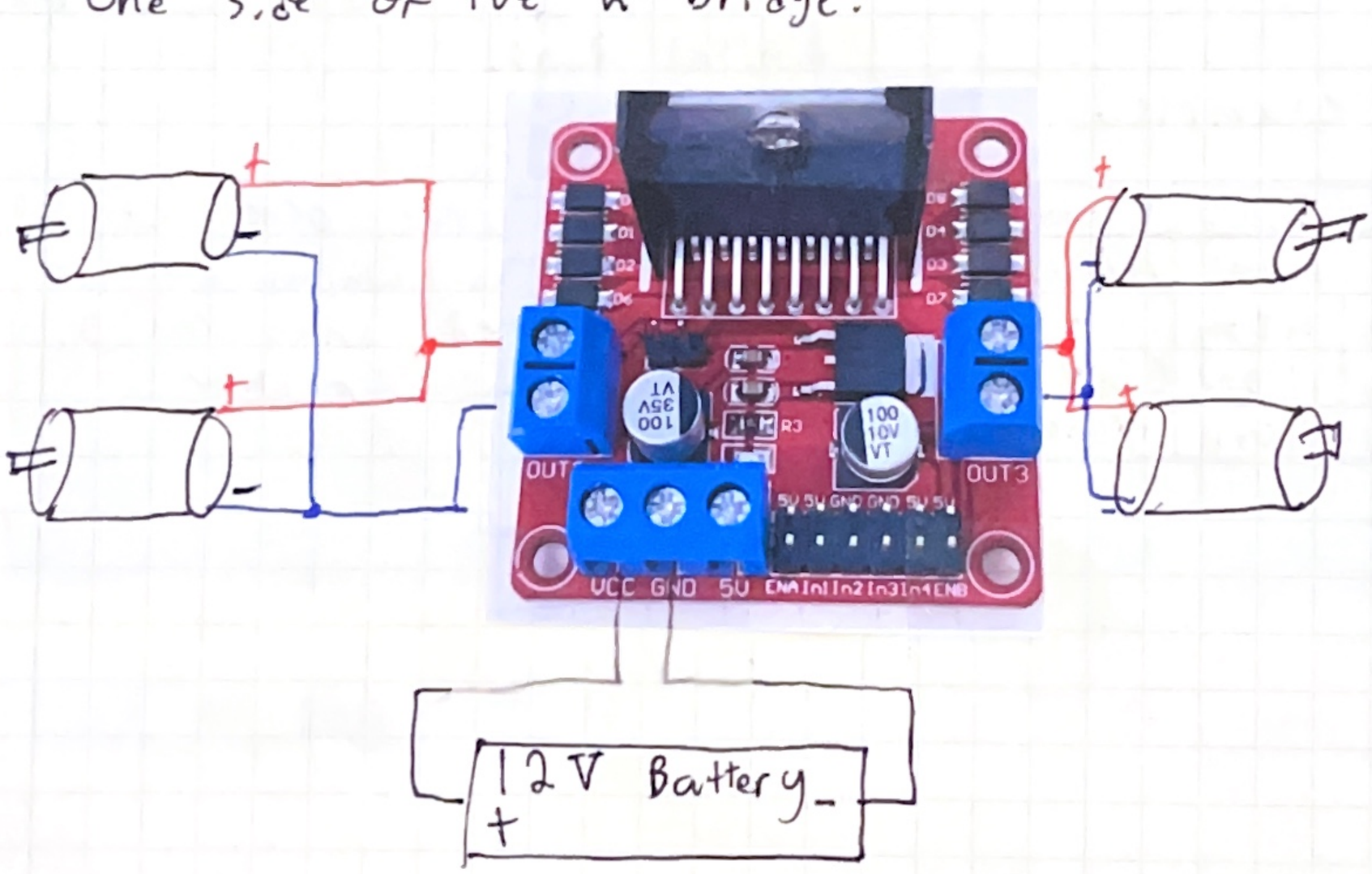


Figure 2

Upgraded WIL Land Chassis.
We got faster motors and bigger wheels.
Notice that we are driving two motors from one side of the h-bridge.



The following connections are made on the giga:

PIN	GIga
ENA	7
ENB	2
IN1	49
IN2	47
IN3	51
IN4	53
GND	GND

Wiring Table

Note that VCC is powered using a 12V battery.

The giga was powered using a rechargable battery.

Writing the Library

The first Step is to write a header file. This will define what we plan to implement in the .CPP file. Place this in a [illegible] folder labeled "src".

MotorControl.h:

```
#ifndef MOTORCONTROL_H
#define MOTORCONTROL_H

#include <Servo.h>
#include <Arduino.h>

const int FORWARD = 1;
const int REVERSE = 0;

class HBridgeControl
{
    public:
        HBridgeControl(int ENA, int ENB, int IN1, int IN2, int IN3, int IN4);
        void SetSpeed(int leftSpeed, int rightSpeed);
        void SetDirection(int leftDir, int rightDir);

        int leftENA;
        int rightENA;
        int leftDir1;
        int leftDir2;
        int rightDir1;
        int rightDir2;
};

#endif
```

Writing an H-Bridge library

Dec 2023
Weekly code

This file is a fancy way of telling arduino that we plan on implementing an object called "HBridge Control". It will have a constructor that requires the pins we are using. We will implement functionality for the "SetSpeed" and "SetDirection" methods.

In a new file called "MotorControl.cpp" we can implement our Constructor.

```cpp
#include "MotorControl.h"

HBridgeControl::HBridgeControl(int ENA, int ENB, int IN1, int IN2, int IN3, int IN4
{
    rightENA = ENA;
    leftENA = ENB;
    rightDir1 = IN1;
    rightDir2 = IN2;
    leftDir1 = IN3;
    leftDir2 = IN4;


    pinMode(leftENA, OUTPUT);
    pinMode(rightENA, OUTPUT);
    pinMode(leftDir1, OUTPUT);
    pinMode(leftDir2, OUTPUT);
    pinMode(rightDir1, OUTPUT);
    pinMode(rightDir2, OUTPUT);
}
```

Notice that we include our header file we just created.

The constructor will always have the same name as the class. Notice that we have to place "HBridgeControl::" before the constructor name. This is to specify that we are implementing a method specifically for the HBridge control class.

We pass in all of our pins and set them as variables. Then we ensure that they are outputs.

Wrong code

Next, we can implement our functionality:

```
20
21  void HBridgeControl::SetSpeed(int leftSpeed, int rightSpeed)
22  {
23      analogWrite(leftENA, leftSpeed);
24      analogWrite(rightENA, rightSpeed);
25  }
26
27  void HBridgeControl::SetDirection(int leftDir, int rightDir)
28  {
29      if (leftDir==FORWARD)
30      {
31          digitalWrite(leftDir1, HIGH);
32          digitalWrite(leftDir2, LOW);
33      }
34      else if (leftDir == REVERSE)
35      {
36          digitalWrite(leftDir2, HIGH);
37          digitalWrite(leftDir1, LOW);
38      }
39
40      if (rightDir == FORWARD)
41      {
42          digitalWrite(rightDir1, HIGH);
43          digitalWrite(rightDir2, LOW);
44      }
45      else if (rightDir == REVERSE)
46      {
47          digitalWrite(rightDir2, HIGH);
48          digitalWrite(rightDir1, LOW);
49      }
50  }
51
52  #endif
```

Notice again the use of `HBridgeControl::SetSpeed`

Lets break it down:

return type

Void HBridgeControl::SetSpeed

HBridgeControl: Class we are Implementing the method of

SetSpeed: Method name we want to implement

Notice how the prototype of the function is inside the header file we created earlier.

122 | Writing an H-Bridge Library

Dec 2023 Wesley Code

Example Driver code:

```
#include "src/MotorControl/MotorControl.h"

// You may have to play around with the order of the IN3, IN4, and IN2, IN1
// to get the right directions, Or you can change the circuit
// I'm using ENA for the Right side motors, and ENB for the left side motors.
//                                                ENA, ENB, IN2, IN1, IN3, IN4
HBridgeControl myHBridge = HBridgeControl(  7,   2,   47,  49,  51,  53);
void setup()
{
  // Turn the motors off.
  myHBridge.SetSpeed(0, 0);// Left, right
  // Set both the motors to forward
  myHBridge.SetDirection(FORWARD, FORWARD);

  //Serial for debugging
  Serial.begin(9600);
  while(!Serial);
  Serial.println("#GetObsessed");
  delay(2000);
}

void loop()
{
  Serial.println("Testing forwards:");
  myHBridge.SetDirection(FORWARD, FORWARD);
  myHBridge.SetSpeed(255, 255);
  delay(2000);

  Serial.println("Testing Reverse:");
  myHBridge.SetDirection(REVERSE, REVERSE);
  myHBridge.SetSpeed(255, 255);
  delay(2000);

  Serial.println("Testing Pivot Right:");
  myHBridge.SetDirection(FORWARD, REVERSE);
  myHBridge.SetSpeed(255, 255);
  delay(2000);

  Serial.println("Testing Pivot Left:");
  myHBridge.SetDirection(REVERSE, FORWARD);
  myHBridge.SetSpeed(255, 255);
  delay(2000);

  Serial.println("Done testing. Freezing. Press reset to test again!");
  myHBridge.SetSpeed(0, 0);
  while(1);
}
```

HBridgeControl = class name; myHBridge = variable name

This creates an "instance" of our class. "constructor"

FORWARD, FORWARD: Left, Right

* Note that the orientation you choose may flip which direction Left/right are.