Caleb Hubbell
5/2/2018
Team 9

Movement Sensing / Motor Comm Block Artifacts

Order:

Schematic
Block Diagram
Interfaces and Properties
Bill of Materials
Mechanical Drawings
Testing Procedure
Testing Results
Code

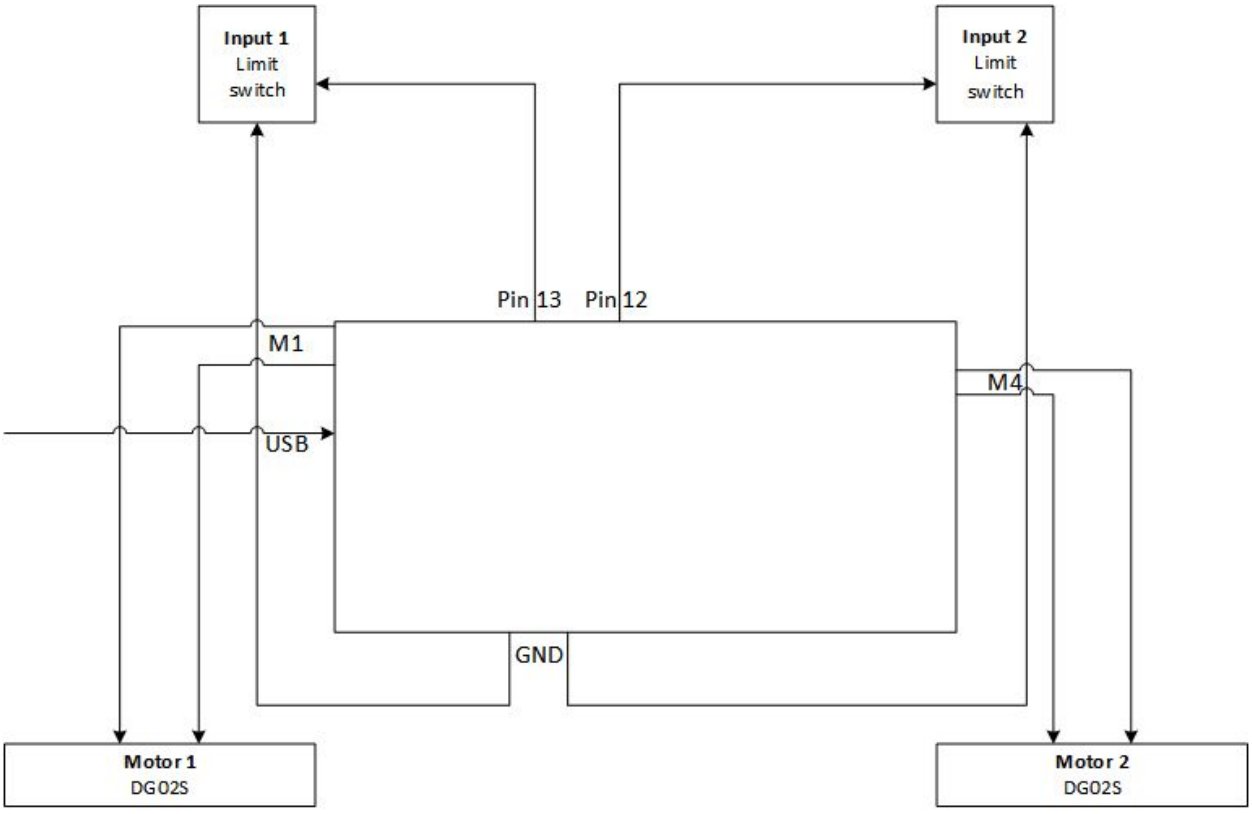

Figure 1: Wiring Schematic

Caleb Hubbell
5/2/2018
Team 9

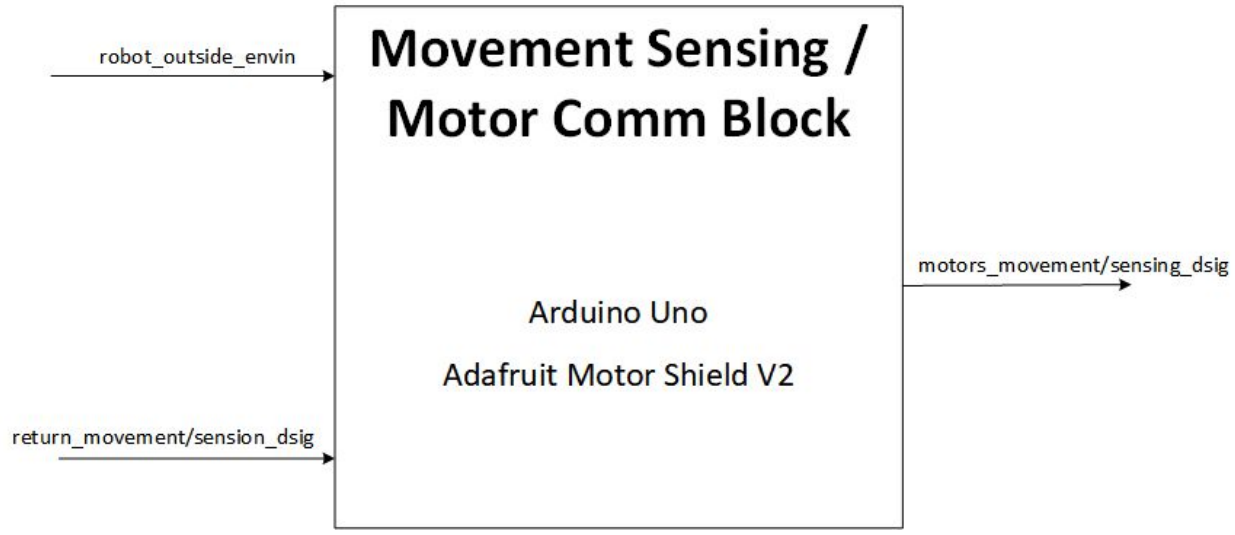

Figure 2: Block Diagram

Interface	Properties
robot_outside_envin	• Surfaces include: tile, short carpet, concrete • Obstacles include: walls, chair legs, cords, clothing
return_movement/sension_dsig	• Function calls which demand motor movement in all 4 directions
motors_movement/sensing_dsig	• PWM DC signal powering 2 motors • PWM frequency of 1.6KHz. • Draw of 0 - 0.45A • Duty cycle of 0-40% • 2 active high inputs; 5V high, 0V low

Table 1: Block Interfaces and Properties

Item	Cost	Quantity	Link
2 DAGU DG02S	$9.90	1	https://www.ebay.com/i/280921404028?chn=ps
Limit Switch	$2.16	2	https://www.mouser.com/ProductDetail/Omron/SS-5GL2/?qs=0w99tykdtPLFgpdchJ6bXg%3D%3D&gclid=EAIaIQobChMIyJna2N_n2gIVAf5kCh1n7gSHEAQYASABEgLstPD_BwE
Arduino Uno	$22.00	1	https://store.arduino.cc/usa/arduino-uno-rev3
Adafruit Motor Shield	$19.95	1	https://www.adafruit.com/product/1438

Caleb Hubbell
5/2/2018
Team 9

Table 2: Bill of Materials

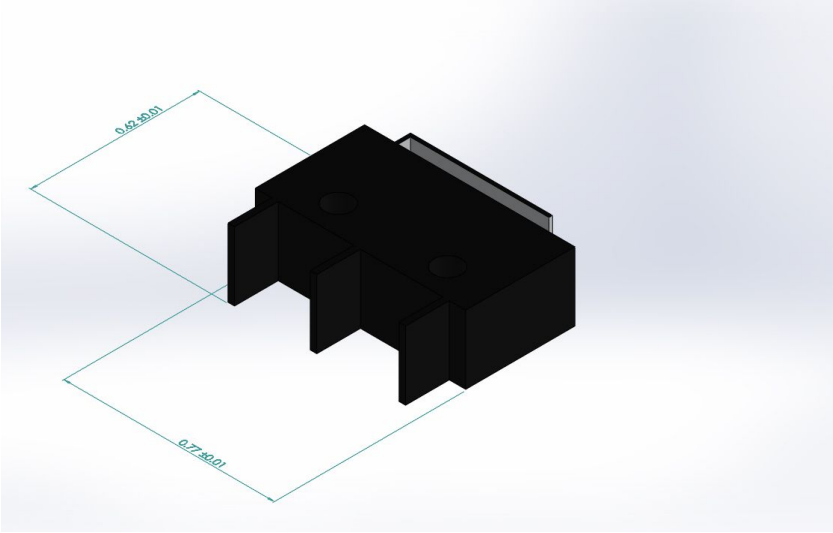

Figure 3: Limit Switch

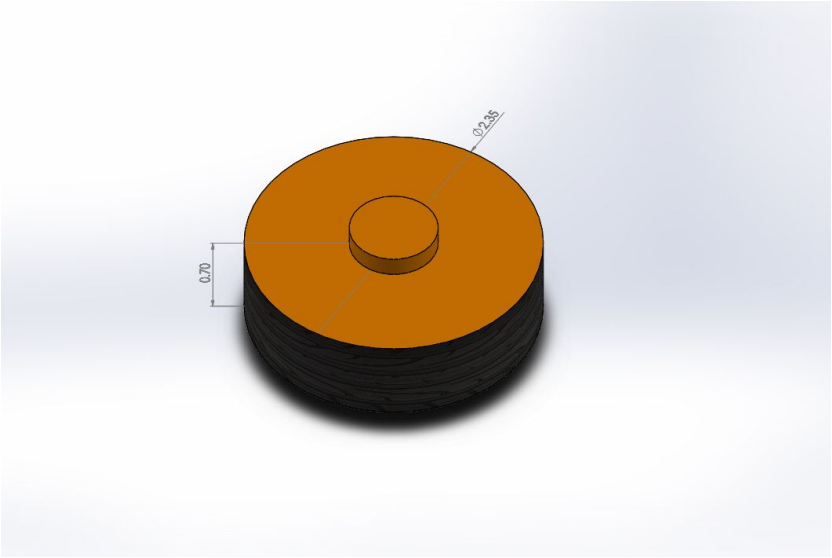

Figure 4: Wheel

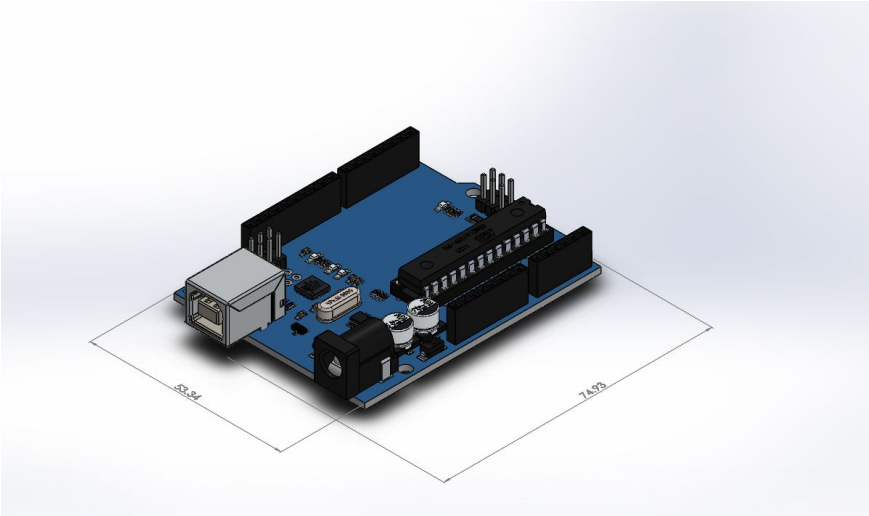

Figure 5: Arduino Uno (NOTE: This drawing is not mine, it was found online)

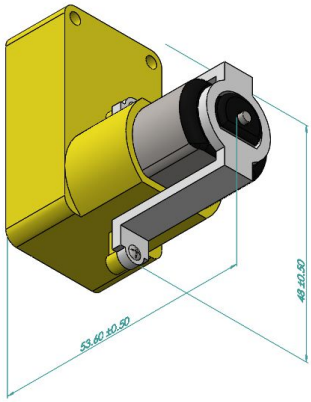

Figure 6: Motor (NOTE: This drawing is not mine, it was found online)

Testing Procedure:

The following testing procedure should be followed to verify that the motor control and object avoidance code works correctly.

A. Setup

All four of the below tests require the same setup. Because of such, we will go through the setup process once now, so that it does not get covered four different times.

a. Upload file “Motor_Control” to the arduino Uno.

b. Make sure the motor shield is connected to the arduino.
c. Connect the left motor to “M1” on the shield. (Red wire towards the button, black wire towards “GND”).
d. Connect the right motor to “M4” on the shield. (Red wire towards “GND”, black wire towards the pins).
e. Connect one wire off of the left limit switch to Pin 12, and the other to GND.
f. Connect one wire off of the right limit switch to Pin 13, and the other to GND.
g. Launch the serial monitor.
h. Leave the USB plugged in.

B. Forward movement test

Because the block must meet the criterion of “Function calls which demand motor movement in all 4 directions”, this test will see if the function call to move forward works correctly

a. Make sure step A has been successfully completed.
b. Look at the serial monitor, nothing should be displayed.
c. Look at the wheels, they should turn as if a ball could be fed between them.

PASS: This test passes if both b and c are true.

C. Left object test

Because the block must meet the criterion of “Obstacles include: walls, chair legs, cords, clothing”, this test will see if the robot can detect, and maneuver around, an object on the left.

a. Make sure step A has been successfully completed.
b. Press the left limit switch once.
c. Look at the serial monitor, “Left” should be displayed.
d. Look at the wheels, they should turn backwards for a little while, and then the left one should turn forward independently.
e. The wheels should then resume forward motion.

PASS: This test passes if c, d, and e are true.

D. Right object test

Because the block must meet the criterion of “Obstacles include: walls, chair legs, cords, clothing”, this test will see if the robot can detect, and maneuver around, an object on the right.

a. Make sure step A has been successfully completed.
b. Press the right limit switch once.
c. Look at the serial monitor, “Right” should be displayed.

d. Look at the wheels, they should turn backwards for a little while, and then the right one should turn forward independently.
e. The wheels should then resume forward motion.

PASS: This test passes if c, d, and e are true.

E. Front object test

Because the block must meet the criterion of "Obstacles include: walls, chair legs, cords, clothing", this test will see if the robot can detect, and maneuver around, an object directly in front of it.

a. Make sure step A has been successfully completed.
b. Press both limit switches simultaneously.
c. Look at the serial monitor, "Both" should be displayed. (if it is not, continue to hold both switches until it appears).
d. Look at the wheels, they should turn backwards for a little while, and then the left one should turn forward independently.
e. The wheels should then resume forward motion.

PASS: This test passes if c, d, and e are true.

If the block passes all of the listed tests, all interface properties have been verified and the block is ready for inclusion into the system.

Testing Results

B. Pass.
C. Pass.
D. Pass.
E. Pass.

Arduino Code:

```
// Written by Caleb Hubbell for ECE 342

//============================================================
====
//     Declare Global Variables & Header Files
//============================================================
====
// Including Arduino's I2C library, as well as the motorshield,
and LCD libraries.
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include <LiquidCrystal.h>
```

Caleb Hubbell
5/2/2018
Team 9

```
// Constants are defined for the physical elements connecting
to the Arduino
const int LEFT_LIMIT = 12; // these are pulled up, so other end
goes to ground
const int RIGHT_LIMIT = 13;

// Constants used numerically in the algorithms
const int AMOUNT_OF_PULSES = 12; // maybe need, for tony?

// declare states
int DEFAULT_STATE = 1;
int HIT_RIGHT_STATE = 2;
int HIT_LEFT_STATE = 3;
int HIT_STATE = 4;

// declare variables
int _currentState;

// adjust to adjust the speed of the robot. Could be tuned to
make it drive straight too
int LSpeed = 120;
int RSpeed = 120;
int Back_Delay = 2000; // should allow bot to get about 6
inches from wall
int Turn_Delay = 1000; // should allow bot to turn ~90 degrees

unsigned long refreshTime = 0;
unsigned long stateTime = 0;

// Create the motor shield object with the default I2C address
Adafruit_MotorShield motorShield = Adafruit_MotorShield();

// Select which 'port' for each motor.
Adafruit_DCMotor *leftMotor = motorShield.getMotor(1);
Adafruit_DCMotor *rightMotor = motorShield.getMotor(4);

// Initialize the LCD with the numbers of the interface pins
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);


void setup()
{
```

```
  // The motor shield and the serial monitor are initialized
  motorShield.begin();
  Serial.begin(9600);

  // Motors started
  leftMotor->run(RELEASE);
  rightMotor->run(RELEASE);

  // Limit inputs enabled
  pinMode(LEFT_LIMIT, INPUT_PULLUP);
  pinMode(RIGHT_LIMIT, INPUT_PULLUP);

  // go to first state
  _currentState = DEFAULT_STATE;
}

// the loop routine runs over and over again forever:
void loop()
{
  // Every quarter second clear the LCD and print to the serial
monitor.
  if ((millis() - refreshTime > 250))
  {
    refreshTime = millis();
    // Prints the current state to the serial monitor. States
are printed
    // to the screen for troubleshooting.
    //Serial.println(currentState);
    // Clears the LCD screen.
    lcd.clear();
  }

  if(_currentState == DEFAULT_STATE) // Just go forward
  {

    // go forward
    forward();

    // show state
    lcd.setCursor(0, 0);
    lcd.print("Forward");
```

```
    // check the limit switches
    if ((digitalRead(RIGHT_LIMIT) == LOW) &&
(digitalRead(LEFT_LIMIT) == LOW))
    {
      // Center hit
      _currentState = HIT_STATE;
      Serial.println("Both");
    }
    else if (digitalRead(LEFT_LIMIT) == LOW)
    {
      // Left hit
      _currentState = HIT_LEFT_STATE;
      Serial.println("Left");
    }
    else if (digitalRead(RIGHT_LIMIT) == LOW)
    {
      // Right hit
      _currentState = HIT_RIGHT_STATE;
      Serial.println("Right");
    }
  }

  if(_currentState == HIT_RIGHT_STATE) // Backup and turn left
  {
    // backup
    backward();
    delay(Back_Delay);

    // turn left
    left();
    delay(Turn_Delay);

    // go back to main loop
    _currentState = DEFAULT_STATE;
  }

  if(_currentState == HIT_LEFT_STATE) // Backup and turn right
  {
    // backup
    backward();
    delay(Back_Delay);

    // turn right
```

```
      right();
      delay(Turn_Delay);

      // go back to main loop
      _currentState = DEFAULT_STATE;
    }

    if(_currentState == HIT_STATE) // Backup and turn around
    {
      // backup
      backward();
      delay(Back_Delay);

      // turn right
      right();
      delay(Turn_Delay * 2);

      // go back to main loop
      _currentState = DEFAULT_STATE;
    }
  }



  // FUNCTIONS
  void forward()
  {
    leftMotor->run(FORWARD);
    leftMotor->setSpeed(LSpeed);
    rightMotor->run(FORWARD);
    rightMotor->setSpeed(RSpeed);
  }

  void backward()
  {
    leftMotor->run(BACKWARD);
    leftMotor->setSpeed(LSpeed);
    rightMotor->run(BACKWARD);
    rightMotor->setSpeed(RSpeed);
  }

  void left()
  {
```

```
  leftMotor->run(FORWARD);
  leftMotor->setSpeed(5);
  rightMotor->run(FORWARD);
  rightMotor->setSpeed(RSpeed);
}

void right()
{
  leftMotor->run(FORWARD);
  leftMotor->setSpeed(LSpeed);
  rightMotor->run(FORWARD);
  rightMotor->setSpeed(5);
}
```