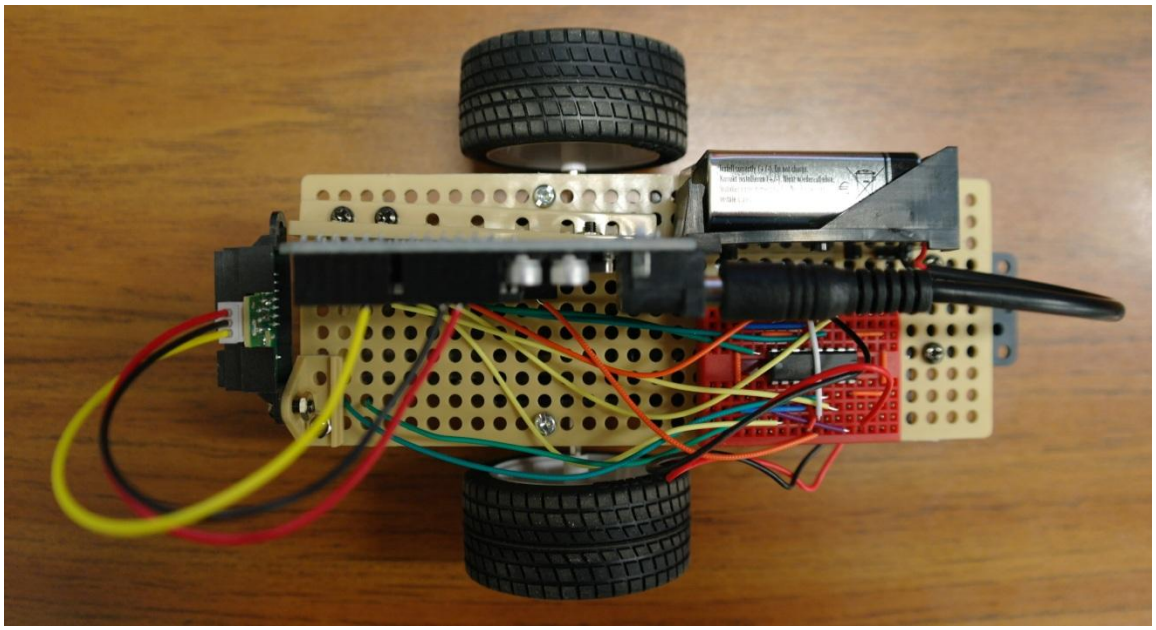
Introduction to Mechatronics Laboratory Manual

Mendenhall Innovation Program
Howard R. Hughes College of Engineering
University of Nevada Las Vegas



Latest Revision: April 22, 2012

Mendenhall Innovation Program

The Howard R. Hughes College of Engineering couples entrepreneurship and design in the Mendenhall Innovation Program, a track that integrates students' engineering background and business savvy.

The Mendenhall Innovation Program, with its minor focused on the early stages of product conception and development, offers all engineering majors the opportunity to learn the basics of product commercialization from faculty and industry practitioners.

For motivated students, it provides the opportunity to collaborate with other students to develop and present a potentially winning idea in the Governor's Cup Business Plan Competition — the capstone of the technology commercialization minor.

Mission Statement

Improve hands-on experience and skill set of engineering students in order to increase their effectiveness as practicing engineers in the commercialization of new technologies.

Dr. Robert Mendenhall

Dr. Robert Mendenhall, founder of Las Vegas Paving Corporation, has done much to inspire our students, and has committed his private support to start the program. With a natural drive for entrepreneurship, he launched America's first recycled highway in the 1970's, and has developed numerous other inventions that continue to benefit the construction industry and conserve natural resources.

Mendenhall Innovation Program Lecture Series

The Howard R. Hughes College of Engineering co-sponsors a lecture series directed at students who are aspiring to entrepreneurship within the engineering community.

Laboratory

Students have a laboratory dedicated to providing them the space needed to create, invent, and design prototypes and or competition apparatus in the Mendenhall Innovation & Design Laboratory (MIDL). The lab is available for all disciplines in the Howard R. Hughes College of Engineering.

Hands-On Learning Modules

The following self-paced laboratory modules are available at any time in the MIDL. These modules were created to augment traditional engineering classes and to provide a resource for independent learning.

- Introduction to Mechatronics
- Interactive Learning Module for Planar Network (Available Fall 2012)
- Embedded Systems Design using Arduino and Processing (Available Fall 2012)
- Dynamic Systems Learning Aid Modules (Available Fall 2012)

Contact Information

Director Mendenhall Innovation Program	Associate Director Mendenhall Innovation Program
Brendan O'Toole, Ph.D. Associate Professor of Mechanical Engineering Email: brendan.otoole@unlv.edu Office Phone: (702) 895 – 3885	Pushkin Kachroo, Ph.D., P.E. Professor of Electrical Engineering Director, Transportation Research Center Email: pushkin.kachroo@unlv.edu

Table of Contents

List of Figures

Introduction to Mechatronics

Mechatronics is the synergistic combination of mechanical engineering, electronic engineering, control engineering, systems design engineering, and computer engineering to create useful products. One of the purposes of this interdisciplinary engineering field is the study of automata from an engineering perspective and serves the purposes of controlling advanced hybrid electro-mechanical systems. The word itself is a combination of 'Mechanics' and 'Electronics'. The importance of multi-disciplinary engineering projects has increased along with the accelerated rate of technological advancement industry-wide. Because all areas of technology are advancing, effective design of a single product often requires close integration of a wide range of disciplines. Microprocessors and sensors have become pervasive within many engineering products, and mechanical engineers will often interface with electrical engineers or select and implement electronic components themselves.

Lab Module Overview

The purpose of this manual is to help participants build a Mechatronics robotic car. This project involves use of breadboards, DC motors, microcontrollers, and programming in order to control a Mechatronics car. This manual provides step-by-step instructions as described below.

- Assembly of a drive system for a small car (gear box, motors, and wheels)
- Assembly of electronic components on an electronic breadboard
- Integration and control of these components using software on an Arduino controller.
- A complete set of parts is shown in the figure below.

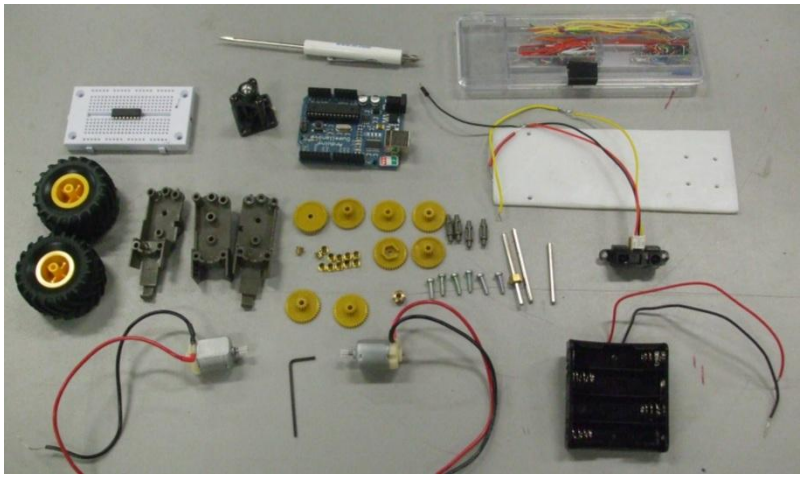


Figure 1 – Complete Kit of Parts for Mechatronics Lab Module

Gearbox Build

Building the Gearbox

The gearbox is probably the most time consuming part of the whole project but it's a vital component of the Mechatronics Kit. The gearbox is assembled so that it supports 2 DC motors connected to wheels through a series of gears. Most DC motors spin too fast for smooth wheel control and provide a very low torque for driving power. The gears serve two functions; they slow down the wheel rotation and increase the torque so the vehicle will have more power.

1. Make sure that all of the required pieces are in the kit, as they will all be used to correctly build the gearbox. (Notice gold bushings are already placed.)

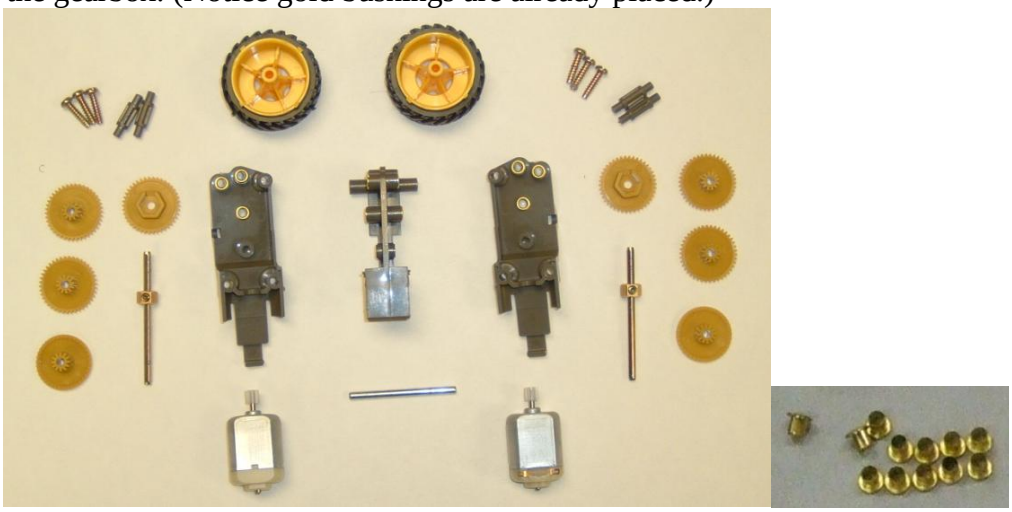


Figure 2 – Gearbox Parts Layout

2. Start by taking the gearbox (2 sides and 1 middle piece) and placing the gold bushings in the 3 spots shown in the picture below on each side of the gearbox as well as the corresponding places in the center piece.

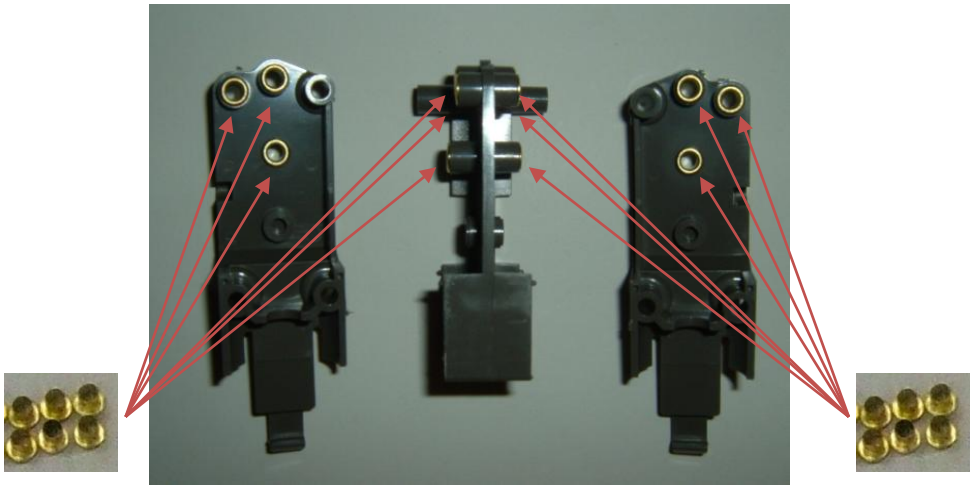


Figure 3 – 12 Gold Bushings Inserted in Gearbox Pieces (You will assemble one half at a time so only place bushings on one side of the middle bracket initially.)

3. Gears will be placed in position during the next steps. The different types of gears are shown below. There are 2 sets of 4 gears (8 total).

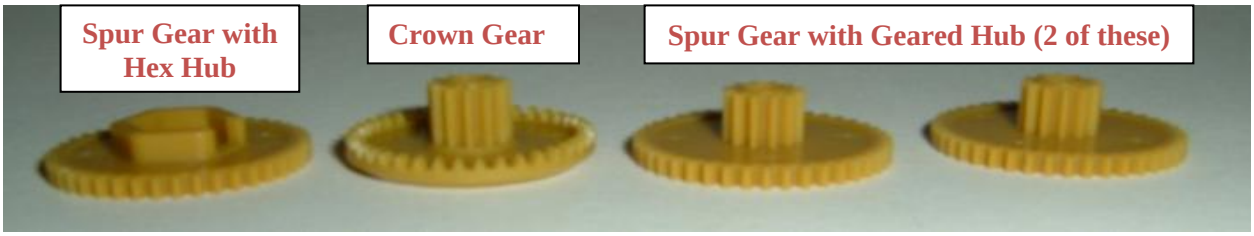


Figure 4 - Hexagonal Gear and 2nd Gear

4. Place the crown gear on one side of the middle bracket as shown below. The crown gear has teeth pointing out to one side only (Figure 5a).
5. Take a hexagonal shaped shaft that has the hexagonal nut attached and place the spur gear with the hexagonal shaped slot on first. Put it on the shorter side of the shaft. Then place a spur gear with the geared hub next to it as shown (Figure 5b).

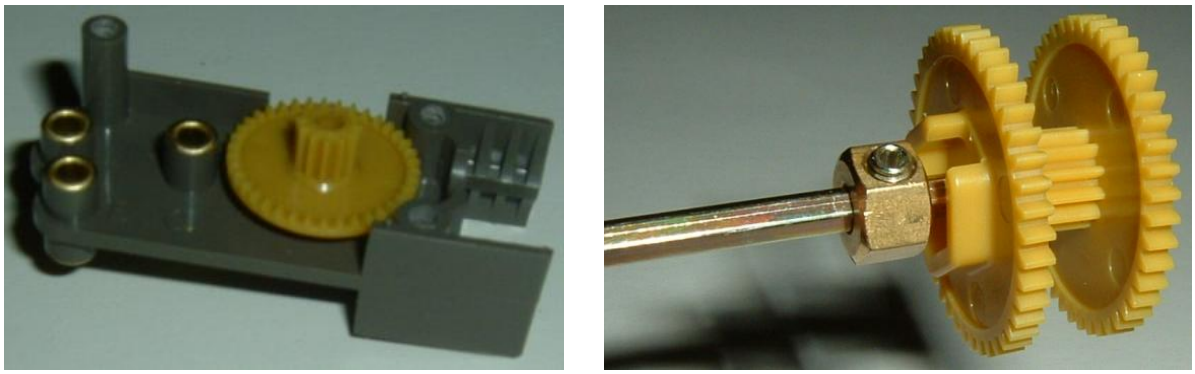


Figure 5 – a) Crown Gear on Middle Bracket and b) 2 Gears on Shaft

6. Place the end of the hexagonal shaft, with the gears mounted on, through the hole with a gold bushing located in the center of the gearbox center piece.

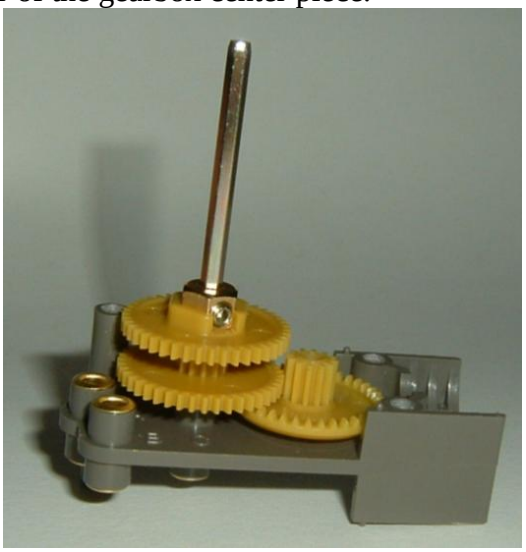


Figure 6 - First 3 Gears Mounted on One Side of Center Bracket of Gearbox

7. Place the final gear on top of the crown gear so that the teeth on its' hub mate with the spur gear closest to the hexagonal nut. Also the outer teeth of the final gear mate with the hub teeth of the bottom gear on the hexagonal shaft.
8. Now that the gears are placed correctly you can insert the smooth shaft through the two gears that are lined together. Then place two of the gray pegs into the two open gold bushings.

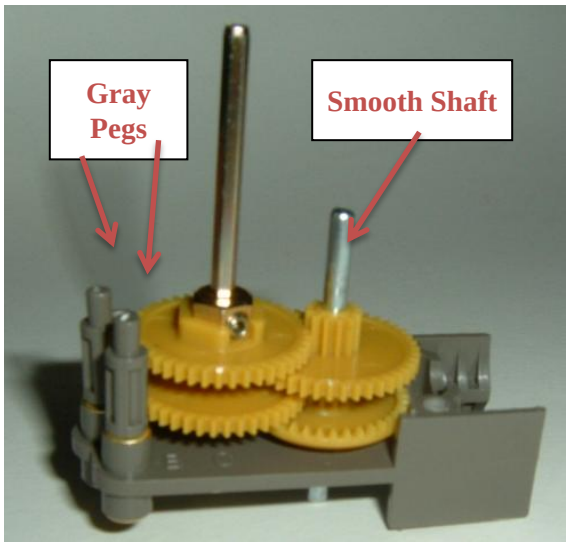


Figure 7 - All Gears, Pegs, and Shafts Placed

9. Next place the DC motor shaft in between the two gears mounted on the smooth shaft and let the motor sit in its holster. ****WARNING** The wire connections to the motor are very brittle. DO NOT flex the motor wire connections or they will break.** The teeth on the motor shaft gear should mate with the crown gear teeth. Finally, being careful not to drop the gold bushings, attach the side of the gear box over the shafts and hold the center piece and the side piece together (Figure 8a)
10. Lastly, insert and tighten (hand tight; do not over tighten) three screws on the side of the gearbox (Figure 8b).

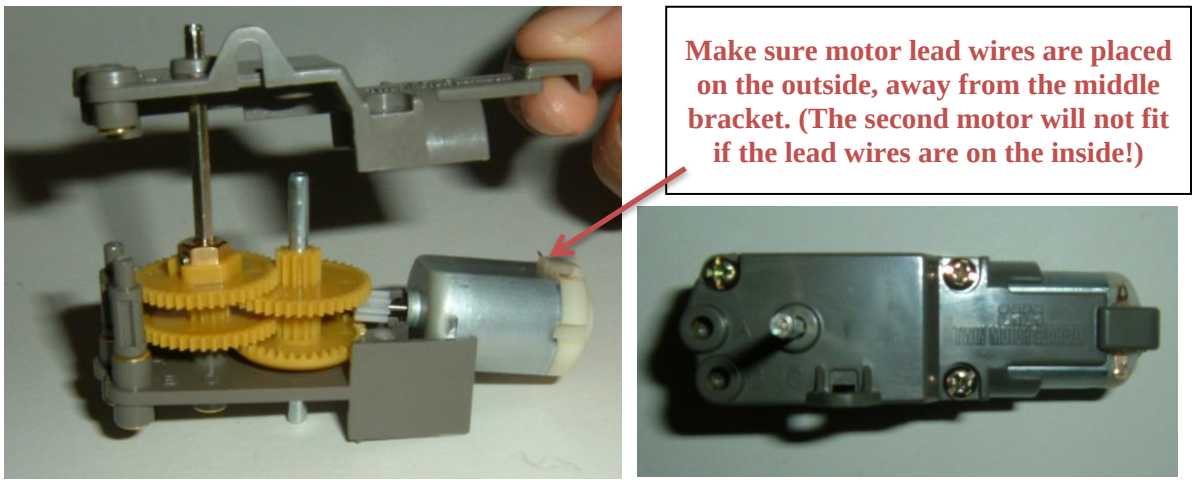


Figure 8 – a) Motor Placement with Gearbox Side Mounting and b) Screw Placement

11. The other side of the gearbox is mirrored with the center of the middle bracket of the gearbox. Repeat Steps 3) – 10) for the other half of the gearbox. Building from the center to the outside will make things the easiest.

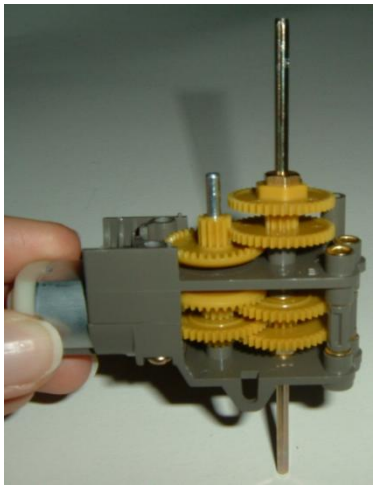


Figure 9 - Building from Finished side of Gearbox

12. The final gearbox assembly is shown below. Tires slide directly on to the hexagonal shafts.

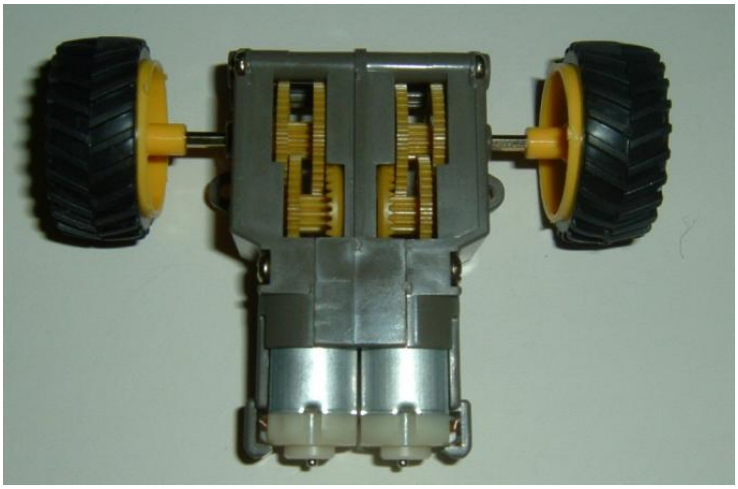


Figure 10 - Final Gearbox with Tires Added to Shaft Ends

13. Put the gearbox aside for now. The next phase of the lab will be to assemble the electronic control system.

Electronics Assembly

This section of the manual will describe how to use the Arduino microcontroller in conjunction with the H-bridge and breadboard to control the Mechatronics car. An H-bridge is an electronic component used to control DC motors.

The first thing that needs to be done is to establish the difference between the digital and analog pins on the Arduino. In reference to the Arduino board being shown in Figure 11, the side that is directly above the Arduino Duemilanove brand are the digital pins. The ones that are below the Arduino Duemilanove on the bottom of the board are the analog pins. If this is confusing they are labeled on the actual board right next to the pins.

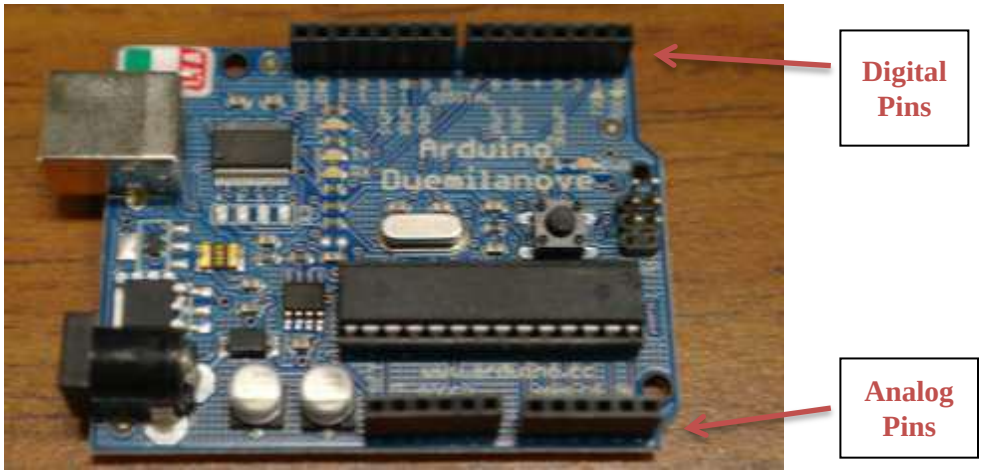


Figure 11 - Arduino Microcontroller

Before you begin wiring take a minute to understand how a breadboard works. As you can see from the figure below, metal strips connect anything in line with their respective positions. Notice that the two middle sections are not connected to each other so in order to connect the column of 5 pins on the top with the 5 pins on the bottom you would need to bridge the two columns with a wire. In our use of breadboards we will reserve one side strip for power and one side strip for ground. The figure below shows a standard breadboard. However, our circuit is much smaller thanks to the H-Bridge so a smaller breadboard is used. It has one strip on each side of the board and overall shorter but the same wiring configuration applies to the connections.

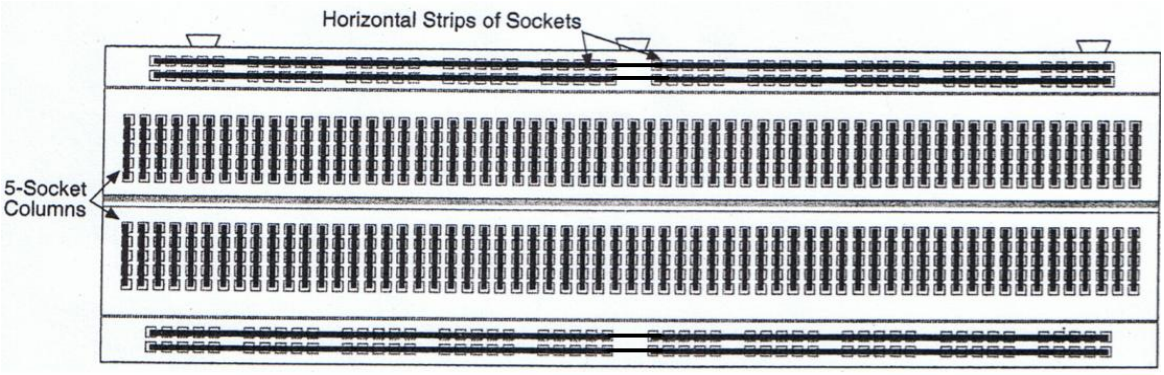


Figure 12 - Example of Breadboard Connections

Circuit Building

Now the circuit needed for the Mechatronics car can be built following these sequential steps.

14. Remember to choose a side for power and a side for ground but do not connect power (batteries) yet. You will not connect anything to power or ground until step 22.

15. An H-Bridge motor controller IC chip has already been mounted to the breadboard. The H-Bridge has 8 connections on either side that are mounted directly into the breadboard. Each of these connectors has a different purpose and they are numbered according to the figure below. There is a U-shaped notch on one end of the H-bridge that is there for reference purpose. This is the top of the chip and the pins are numbered from the immediate left of that from 1 – 8 then at the bottom cross directly over and they are numbered 9 through 16. Take time to examine the chip and breadboard and get familiar with this layout.

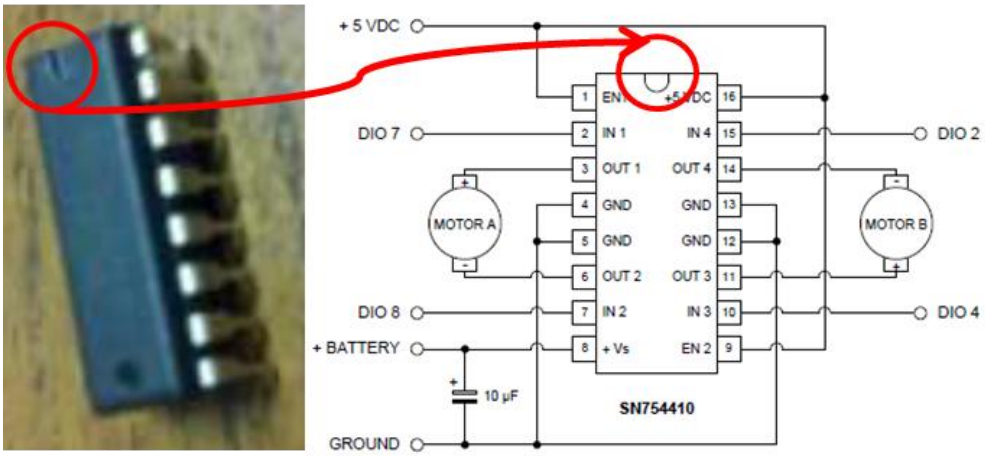


Figure 13 – H-Bridge IC Chip Pin Numbering System

16. Connect Pin 2 of the H-Bridge to the Digital Pin 8 on the Arduino (Top red wire in figure below).
17. Connect Pin 7 of the H-Bridge to the Digital Pin 7 on the Arduino (Bottom red wire in figure below).

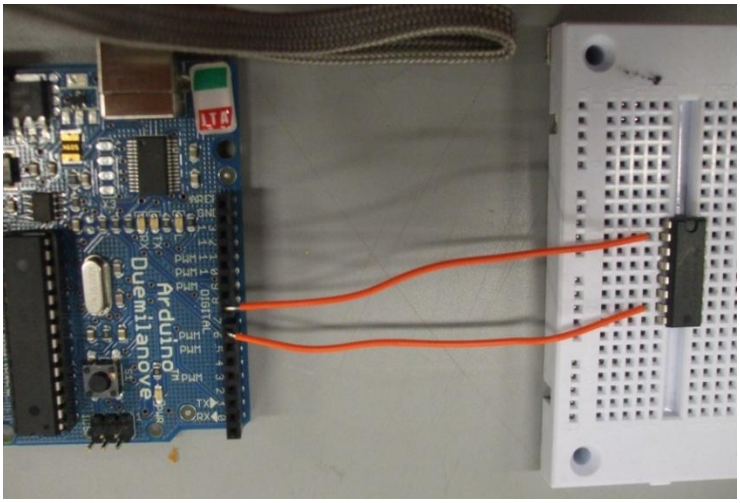


Figure 14 - Steps 16 and 17 Logic Pin Connections

18. Connect Pin 10 of the H-Bridge to the Digital Pin 4 on the Arduino (Yellow wire below).
19. Connect Pin 15 of the H-Bridge to the Digital Pin 2 on the Arduino (Yellow wire below).

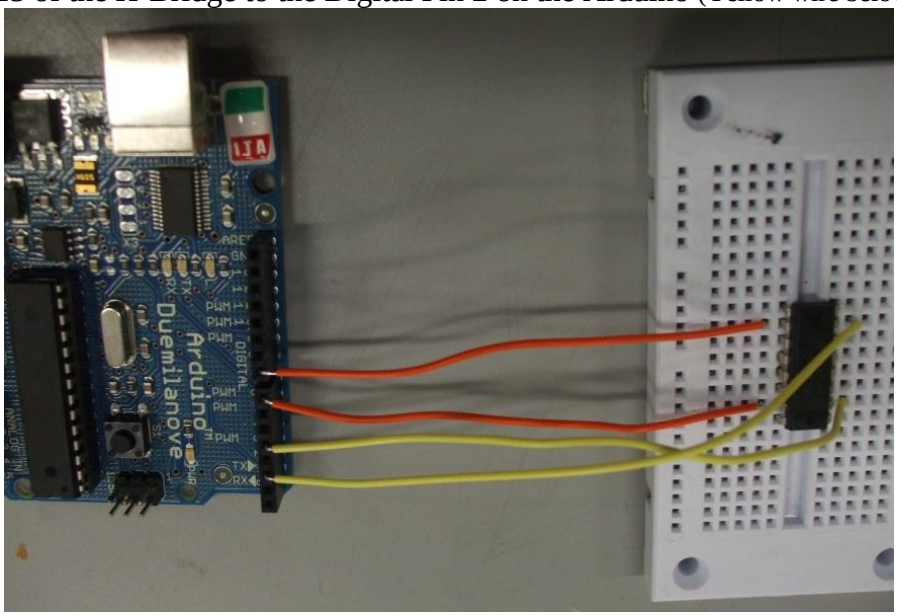


Figure 15 - Steps 18 and 19 w/ Step 16 and 17 Logic Pins

20. Connect Motor A's wires to Pins 3 and 6 on the H-Bridge. (If the motor runs the opposite direction of what is expected simply switch the two wires). **Warning Note: Please bend the wires at their ends and not the leads to the motor.**
21. Connect Motor B's wires to Pins 14 and 11 on the H-Bridge. (If the motor runs the opposite direction of what is expected simply switch the two wires). **Warning Note: Please bend the wires at their ends and not the leads to the motor.**

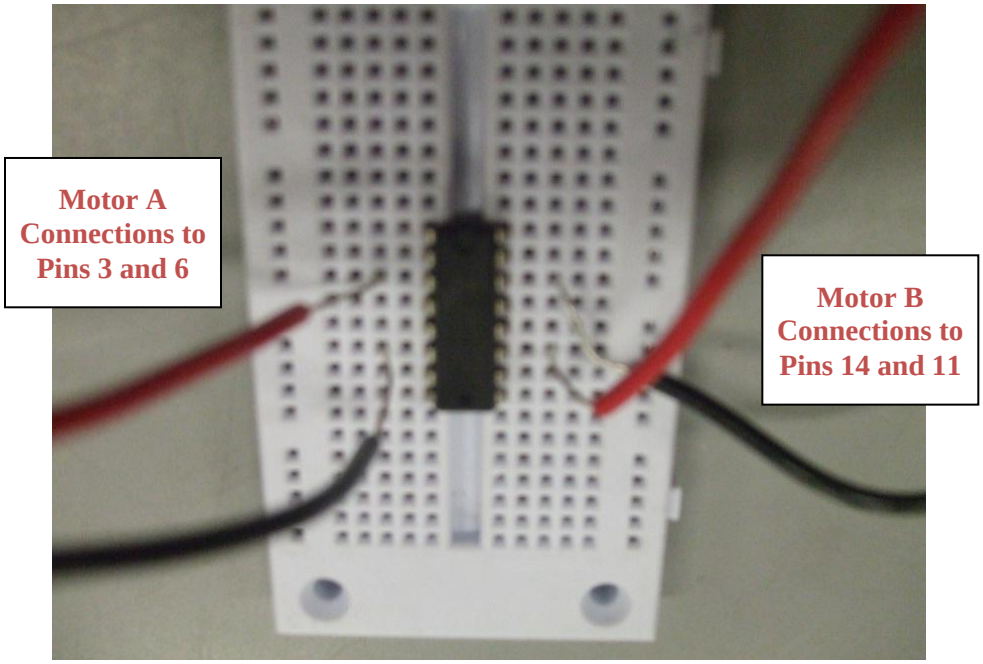


Figure 16 - Steps 20 and 21, Motor Connections (The connections from the previous steps were removed so that these connection locations are easier to see. DO NOT remove your previous connections.)

22. Connect Pins 1, 8, 9, and 16 on the H-Bridge to the power side of the Breadboard.

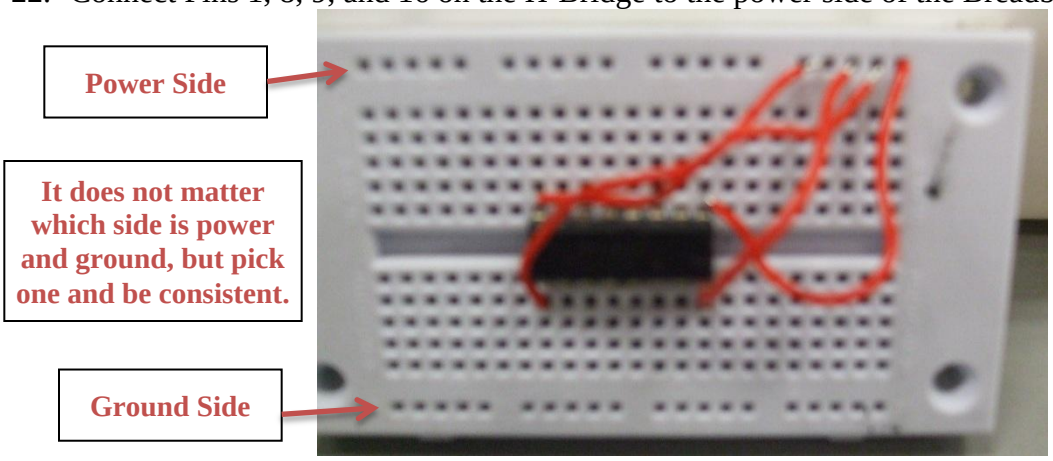


Figure 17 - Step 22 Power Connections (Previous connections not shown)

23. Connect Pins 4, 5, 12, and 13 on the H-Bridge to the ground side of the Breadboard.

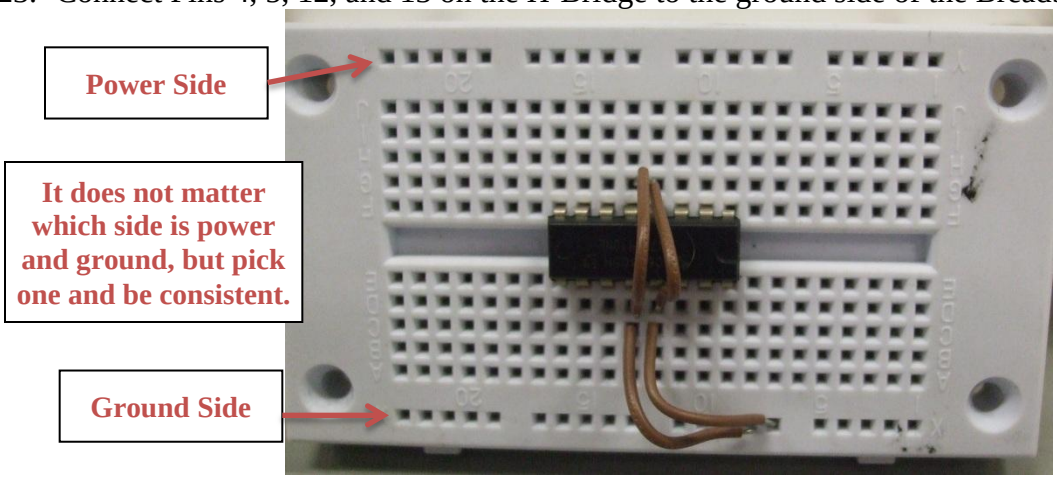


Figure 18 - Step 23 Ground Connections (Previous connections not shown)

24. **The capacitors we use are sensitive to the direction of current flow so it is important to connect the capacitor correctly**. Connect the positive lead of the capacitor to the same strip as pin 8 in between the wire from the battery and the H-Bridge.
25. Connect the negative side of the capacitor to the ground strip.

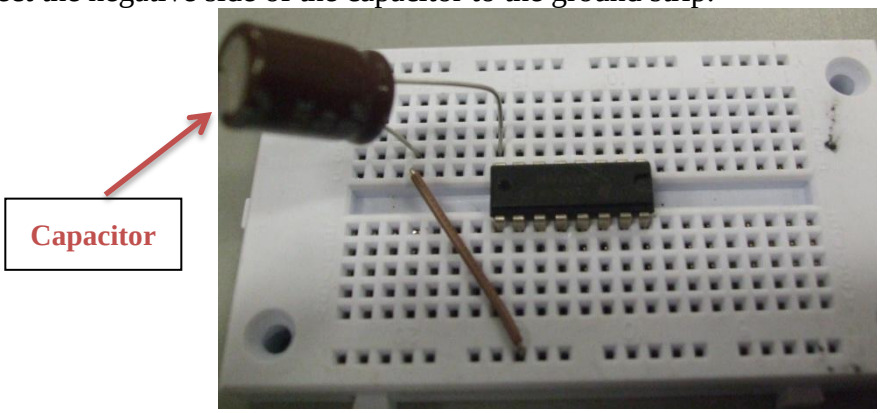


Figure 19 - Steps 24 and 25 Capacitor Connections (Previous connections not shown)

26. Connect the IR sensor's red wire to the power side of the breadboard.
27. Connect the IR sensor's black wire to the ground side of the breadboard.
28. Connect the IR sensor's yellow wire to the Analog Pin A1 on the Arduino.

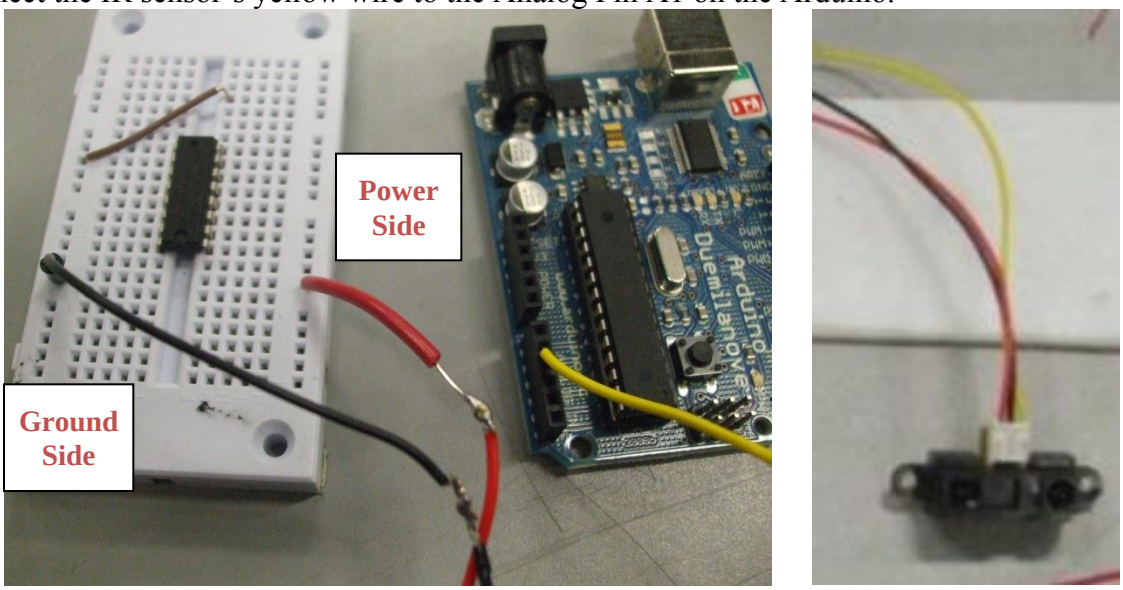


Figure 20 - Steps 26, 27, and 28 IR Sensor Connections

This completes the electronics subassembly.

Final Assembly

The gearbox, breadboard, Arduino, battery pack, ball caster and IR sensor can now be mounted onto a board for the final assembly of the car.

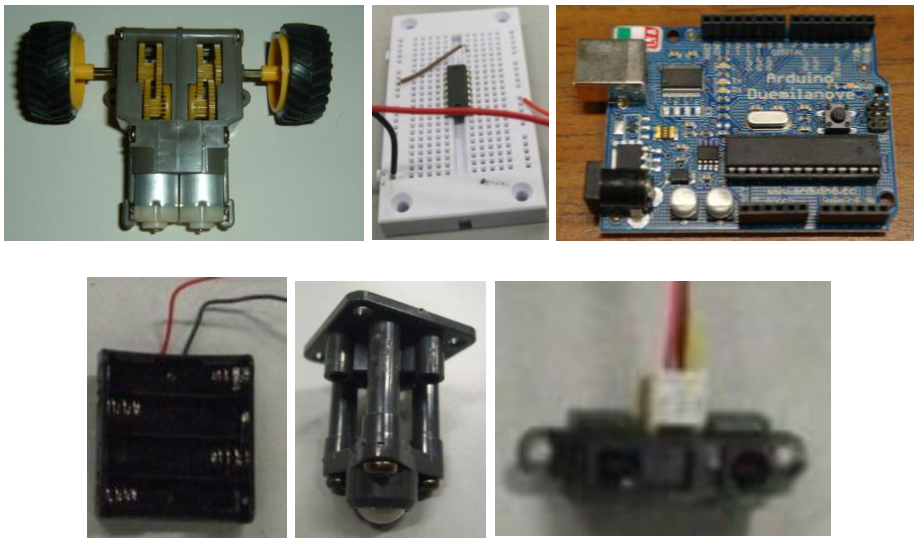


Figure 21 – Gearbox, Breadboard, Arduino, Battery Pack, Ball Caster, and IR Sensor

29. Mount your gearbox, breadboard, Arduino, and sensor to a board in any fashion that keeps everything connected and safe. **Note: A board will be provided if this is a short term project mounting using electrical tape is easiest in order to make changes to the car.**

30. Insert the 4 AA batteries into the battery pack and connect the Red wire to the V_{in} pin on the Arduino.
31. Connect the Black wire to the ground side of the breadboard.
32. Connect the 5V pin to the power side of the breadboard.

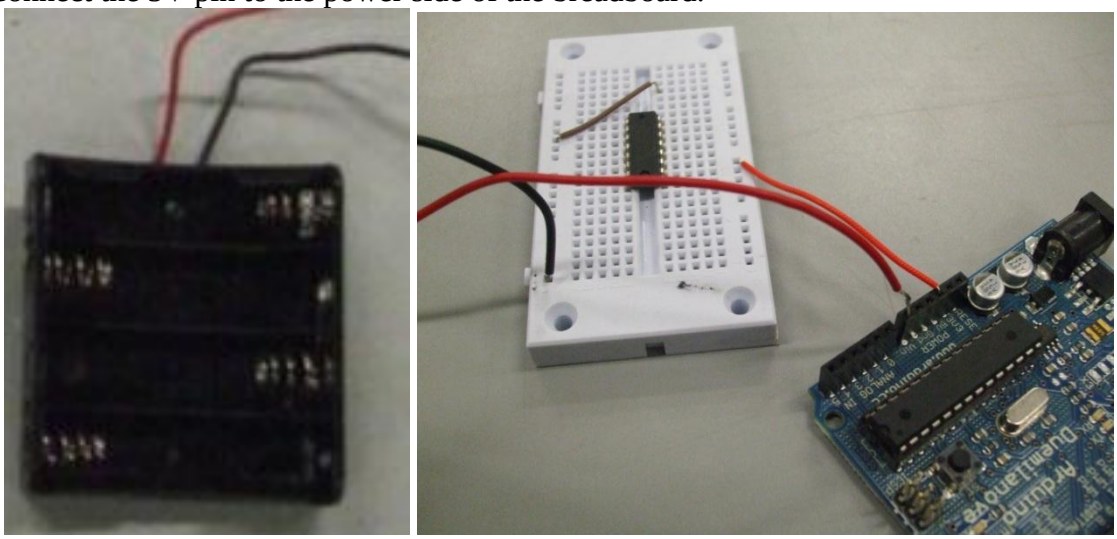


Figure 22 - Steps 30, 31, and 32 Power Supply and Ground

One example of the final assembly is shown below. Some of the subcomponents may be different than the ones used in your kit.

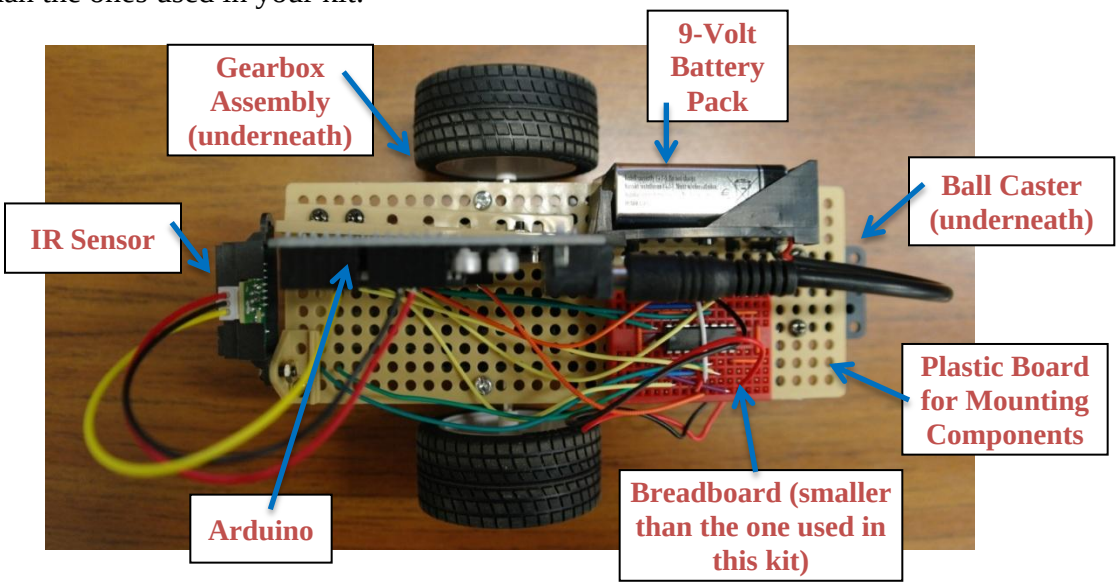
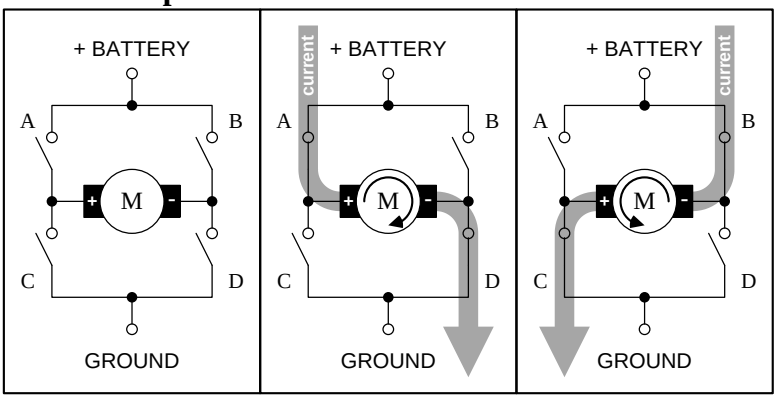


Figure 23 - Steps 30, 31, and 32 Power Supply and Ground

Control Program

33. The Arduino already has an obstacle avoidance program uploaded so just press the reset button and the code will begin. It is a simple obstacle avoidance code so when the Arduino is close to an obstacle, the sensor will see it and the motors will stop and turn. The program code is listed at the end of this manual.

Below are two tables to help in the circuit building and to help understand the motor/current relationship.



(i) OFF **(ii) FORWARD** **(iii) REVERSE**

IN 1 or 3	IN 2 or 4	A	B	C	D	ROTATION
LOW	LOW	OFF	OFF	OFF	OFF	OFF
HIGH	LOW	ON	OFF	OFF	ON	FORWARD
LOW	HIGH	OFF	ON	ON	OFF	REVERSE

Figure 24 - Motor Current Table

Preloaded Obstacle Avoidance Program

```
/*
 * Simple Obstacle Avoidance
 *
 * Copyright (c) 2011
 * Mendenhall Inovation and Design Laboratory
 * Howard R. Hughes College of Engineering
 * University of Nevada, Las Vegas
 * http://engineering.unlv.edu/
 *
 * This code demonstrates Obstacle avoidance
 * using an analog proximity sensor to provide
 * feedback from which motor direction is determined
 *
 * Motor Control IC: Texas Instruments SN754410
 * Proximity Sensor: Sharp GP2Y0A21YK
 *
 */
```

```
// Define Global Variables
int leftPin1 = 8;   // Digital pin 8 connected to IN 2 of SN754410
int leftPin2 = 7;   // Digital pin 7 connected to IN 1 of SN754410
int rightPin1 = 4; // Digital pin 4 connected to IN 3 of SN754410
int rightPin2 = 2; // Digital pin 2 connected to IN 4 of SN754410
int sensor = 1;              // Analog pin 1 connected to Vo of GP2Y0A21YK
int val = 0;                 // Initializing variable to store sensor reading

void setup()
{
  // Configure select Digital pins to behave as logic outputs.

  pinMode(rightPin1, OUTPUT);
  pinMode(rightPin2, OUTPUT);
  pinMode(leftPin1, OUTPUT);
  pinMode(leftPin2, OUTPUT);

  // Configure analog pin 1 as an input for the proximity
  // sensor's voltage output.

  pinMode(sensor, INPUT);

}

void loop()
{

  // The "analogRead" function is used to take a reading
  // from the proximity sensor and store it for future
  // use.  The analog sensor reading is converted to a
  // 10-bit value ranging from 0 to 1023.

  val = analogRead(sensor);

  // 0.1 second delay

  delay(100);

  // Use a "while" loop to control the Forward direction
  // of both motors.  The goal is to move forward until an
  // object is detected at a distance of 15 cm or less from
  // the front of the proximity sensor.
  //
  // The 10-bit value equal to 15 cm is 343.
  //
  // Move forward when "val" is less than 343.
  //
  // Move in reverse when "val" is greater than or
  // equal to 343.

  while (val < 343)
```