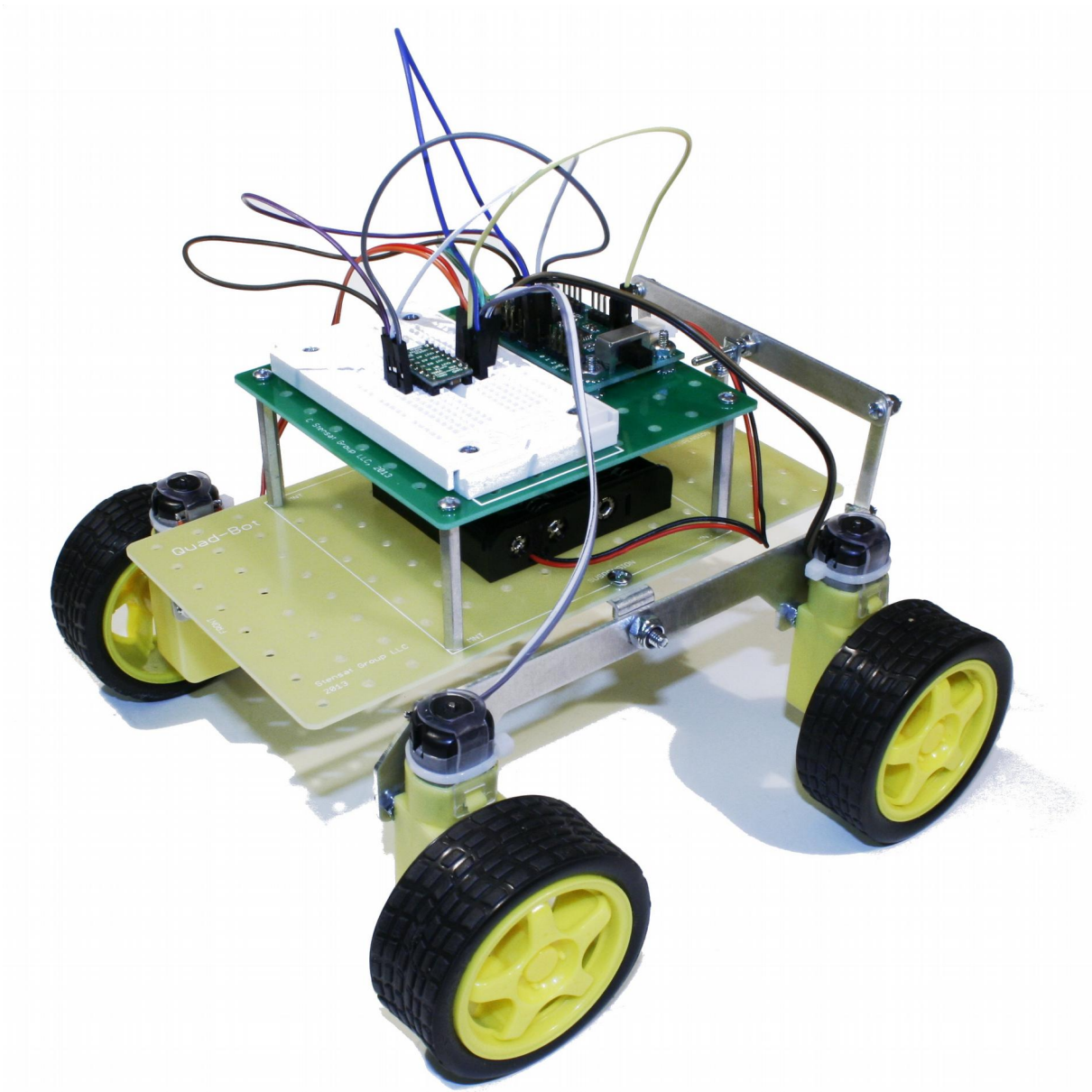
*Sten**BOT*** Rover Kit



Legal Stuff

- Stensat Group LLC assumes no responsibility and/or liability for the use of the kit and documentation.
- There is a 90 day warranty for the Quad-Bot kit against component defects. Damage caused by the user or owner is not covered.
    - Warranty does not cover such things as over tightening nuts on standoffs to the point of breaking off the standoff threads, breaking wires off the motors, causing shorts to damage components, powering the motor driver backwards, plugging the power input into an AC outlet, applying more than 9 volts to the power input, dropping the kit, kicking the kit, throwing the kit in fits of rage, unforeseen damage caused by the user/owner or any other method of destruction.
- If you do cause damage, we can sell you replacement parts or you can get most replacement parts from online hardware distributors.
- This document can be copied and printed and used by individuals who bought the kit, classroom use, summer camp use, and anywhere the kit is used. Stealing and using this document for profit is not allowed.
- If you need to contact us, go to www.stensat.org and click on contact us.

Table of Contents

Overview

Goals

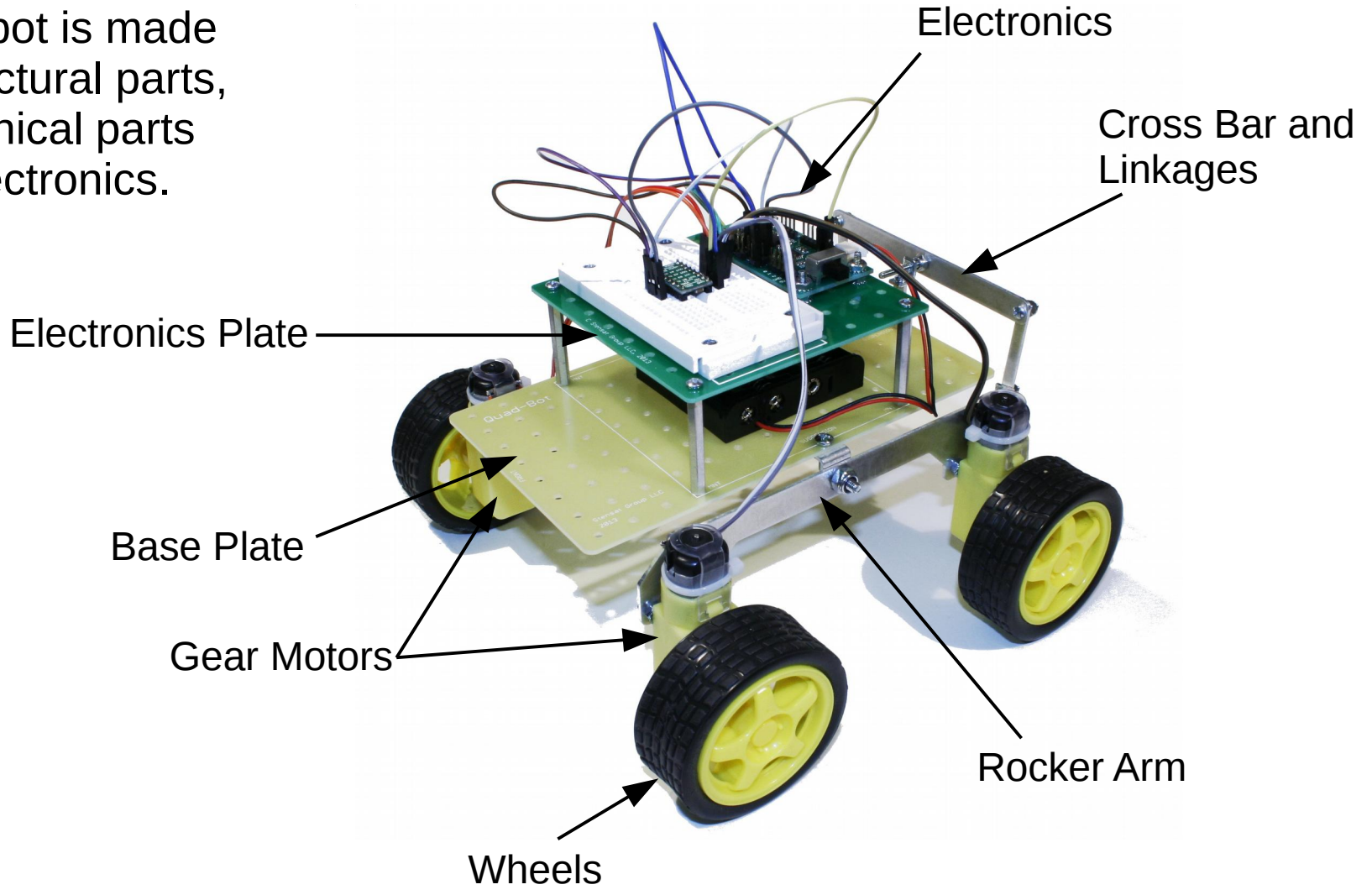
- The goal of this kit is to have a quad-bot platform that can operate on uneven terrain and even in sand autonomously or by remote control.
- You will learn basic autonomous operations using the ultrasonic range sensor.
- For remote operations, you will learn how to configure and program the quad-bot to be controlled by a TV remote control.

Program Overview

- Assemble Kit
- Programming to move
- Learning how to calibrate the motion
- Running the Maze
- Using sensors for collision avoidance
- Running the Maze using collision avoidance
- Remote control

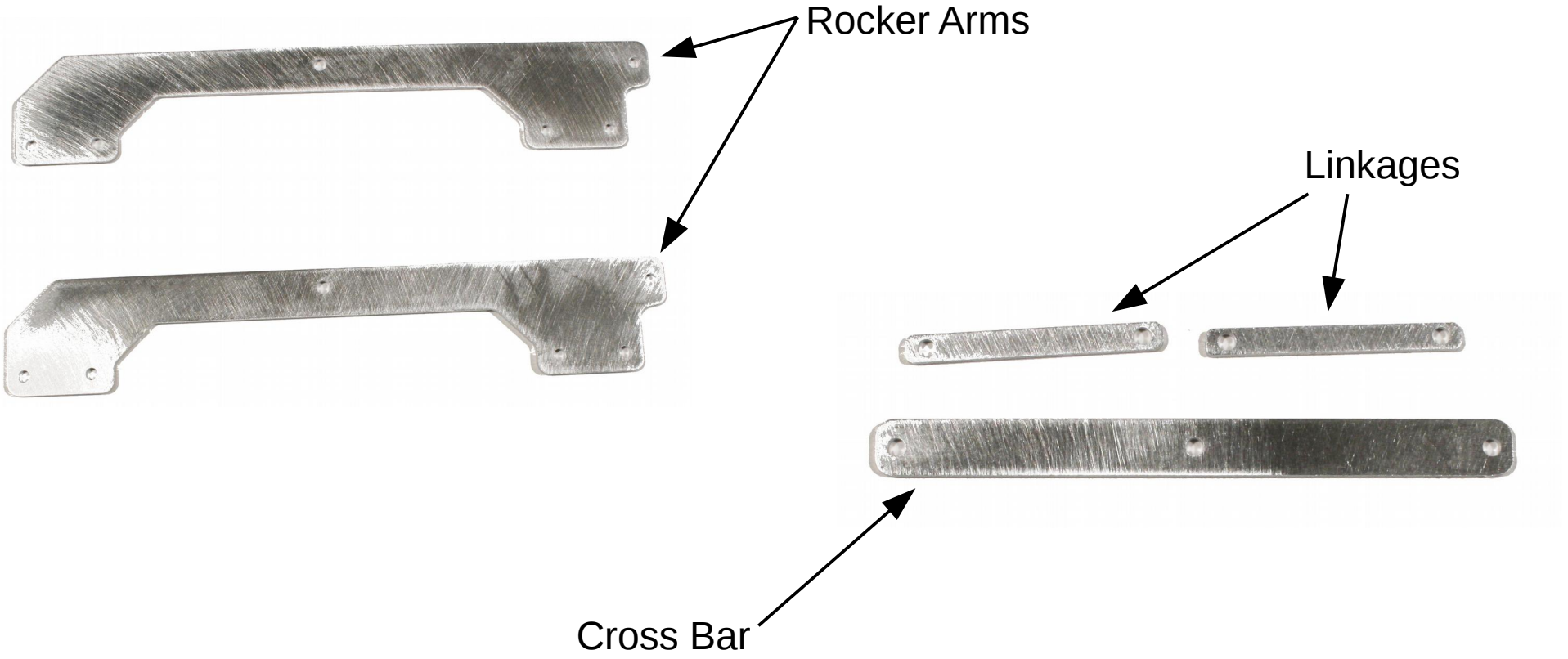
Robot Parts

- The robot is made up structural parts, mechanical parts and electronics.



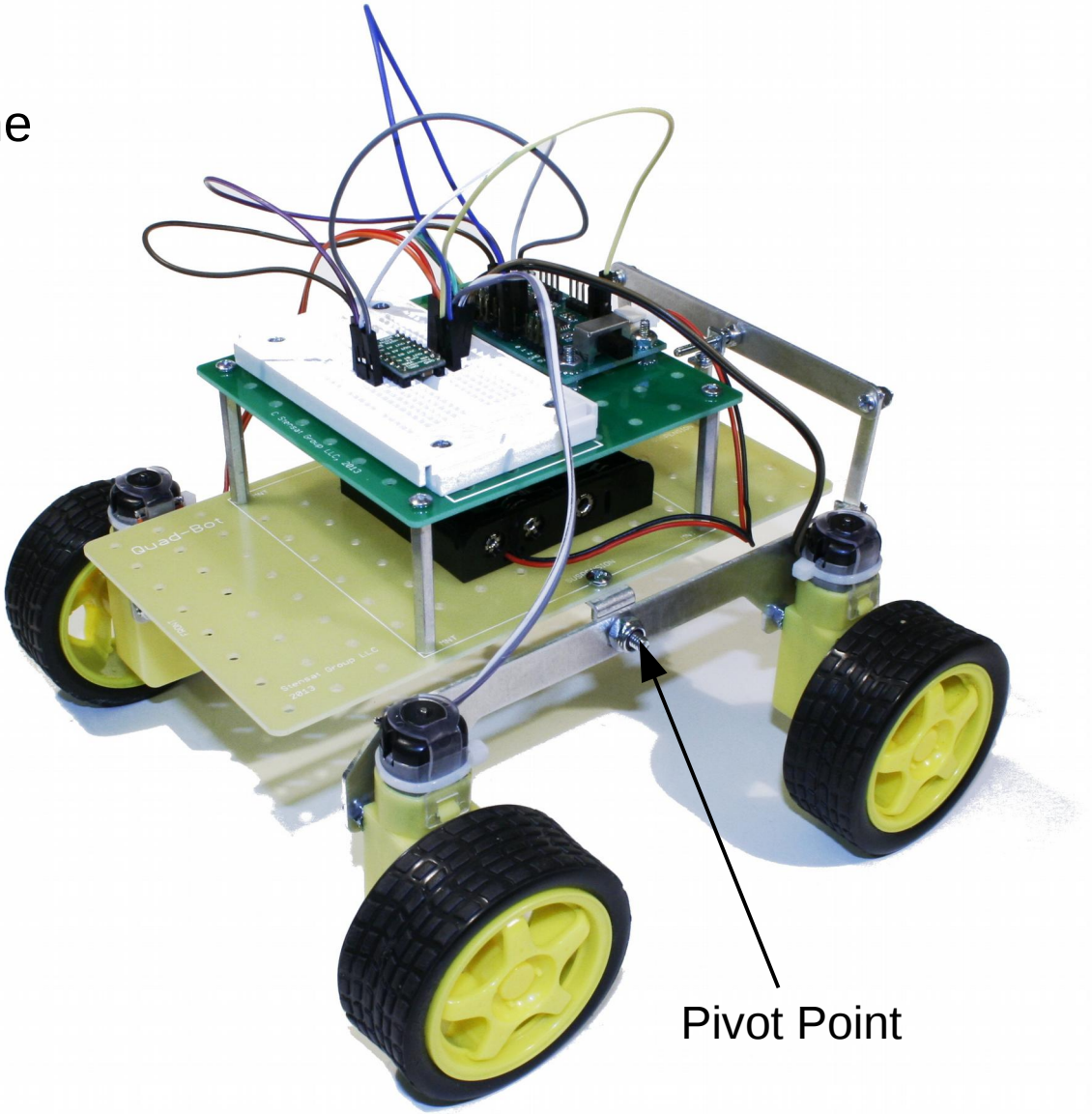
Suspension System

- The rocker arm suspension system is made of a few main components shown below



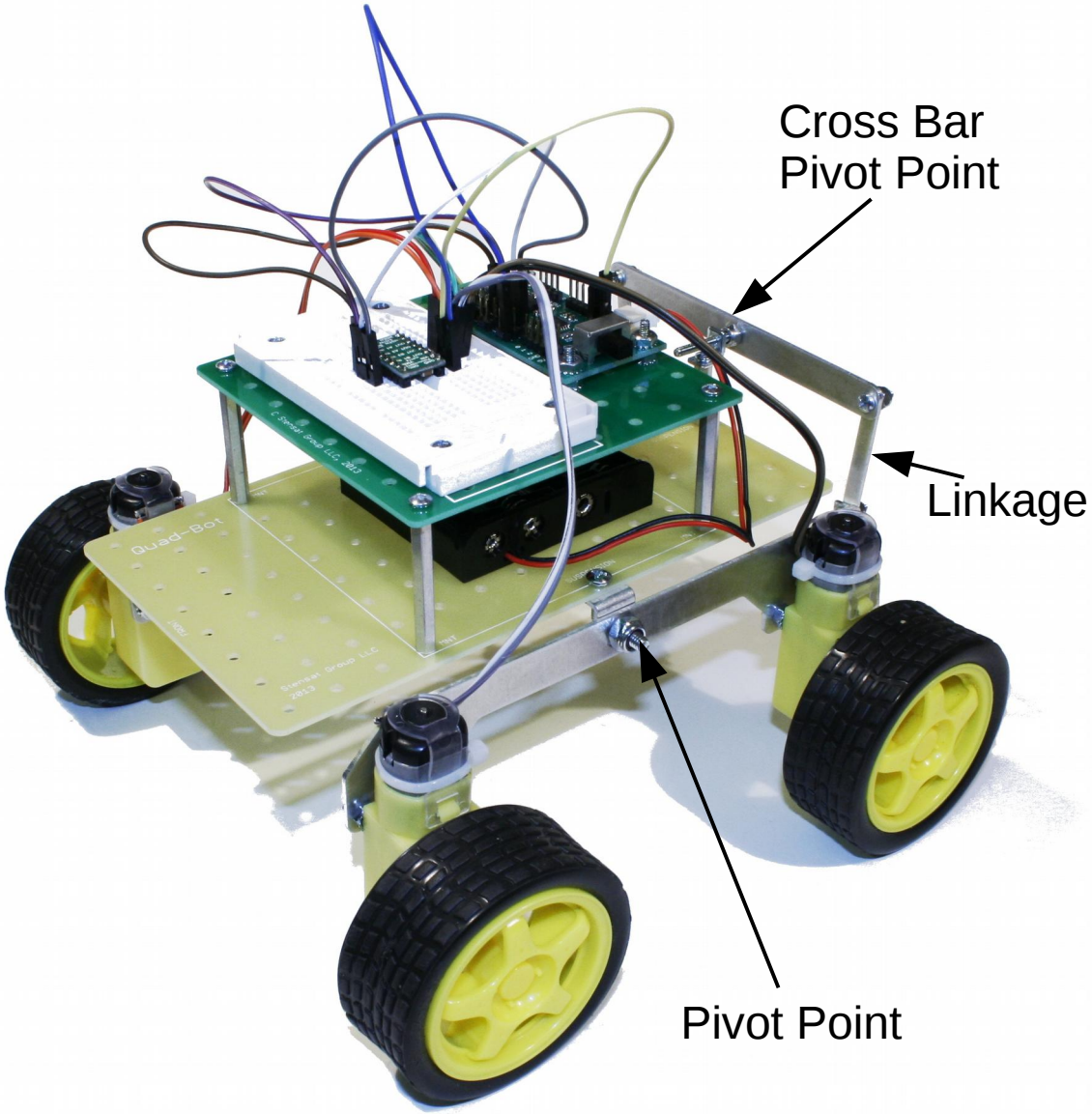
How the Suspension Works

- Two motors are attached to the ends of the rocker arms. The center of the rocker arm is attach to the base plate and pivots. This lets both wheels on the same side keep touching the ground if one drops in a dip or goes up on something.



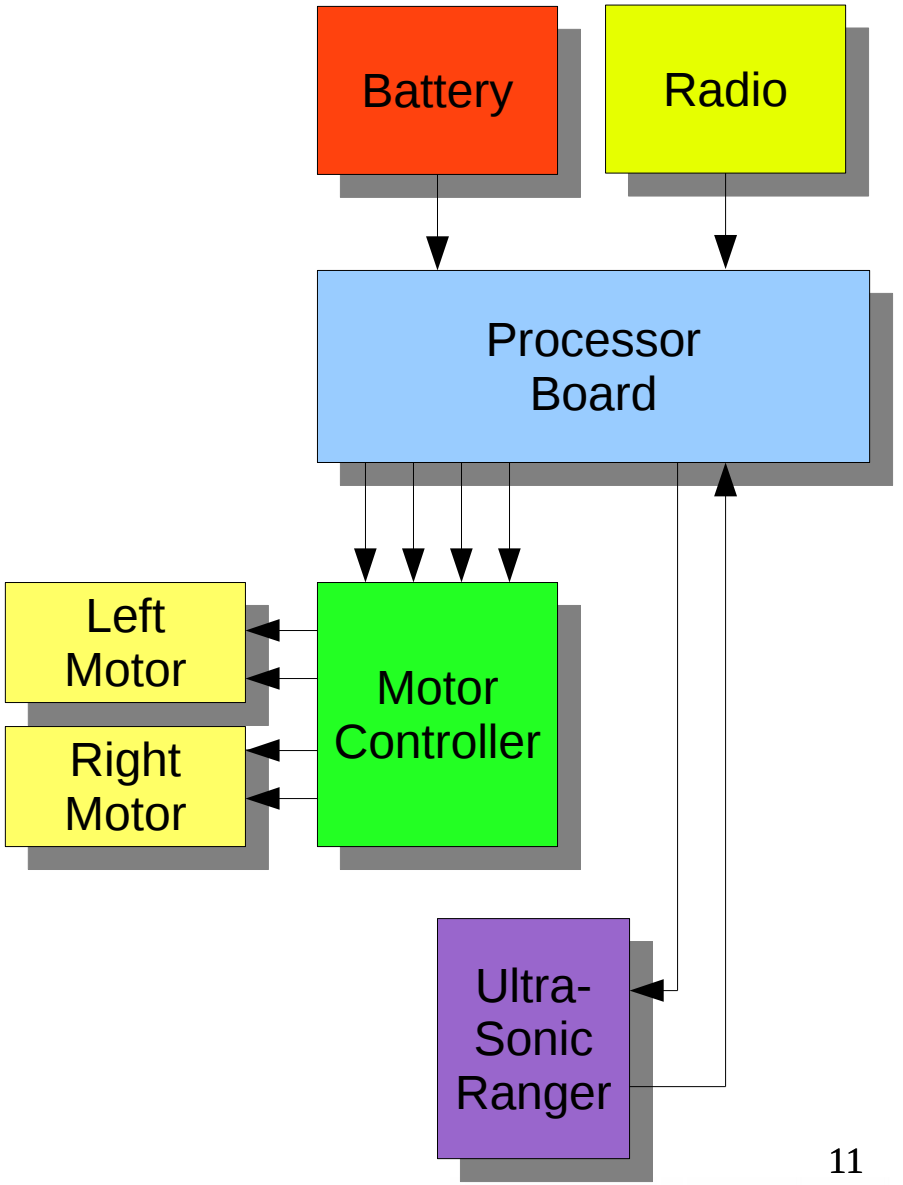
How the Suspension Works

- When the one rocker arm changes its angle, the cross bar is pushed up or down to push the other rocker arm in the opposite direction. This helps keep the wheels on the other rocker arm touching the ground. The linkages attach the rocker arms to the cross bar. The cross bar is attached to the base plate so the base plate changes its angle half way between the two rocker arms.



How the Electronics Works

- The robot electronics controls the operations. It consists of a battery for power, a processor board for the brains, a motor controller to operate the motors and an ultrasonic sensor for detecting obstacles.
- The diagram to the right shows how the electrical components are interconnected.
- The battery powers everything.
- The processor board is the brains and controls everything.
- The motor controller provides the proper interfaces to the motors for the processor board to control the motors.
- The radio provides a method for remote control.

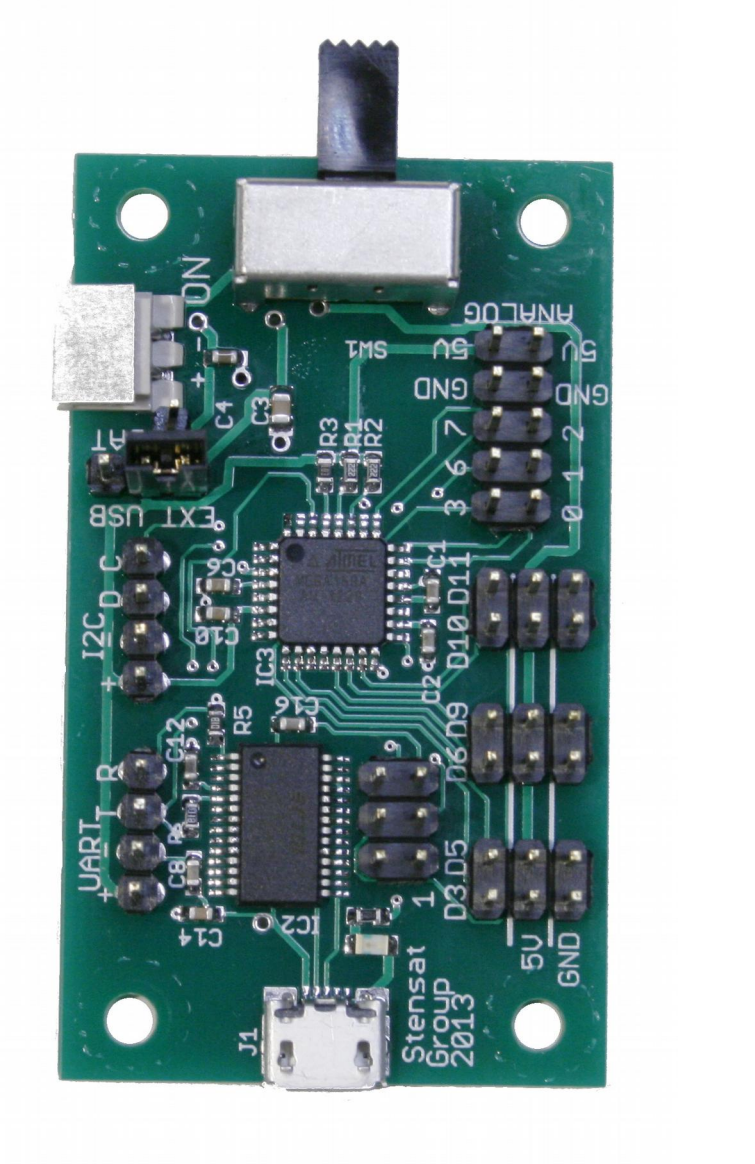


Battery

- The power source for the robot is a battery. The battery is made up of four AA cells.
- Multiple types of AA cells can be used in the robot. The simplest is alkaline cells that cannot be recharged and must be discarded when they are used up. Alkaline cells generate on average 1.5 volts. With four cells, a total of 6 volts is supplied. It will be noticed that over time, the robot will slow down as the batteries drain.
- Rechargeable cells can be used. There is Ni-Cad and nickle metal hydride or Ni-MH. These cells have a slightly lower voltage of 1.25 volts. The benefit is the cells can be recharged and reused many times. Another benefit is these cells can deliver more current without dropping the voltage unlike alkaline cells. These types of batteries also maintain the same voltage longer than Alkaline batteries. This allows the robot to maintain its speed.
- The four cells are connected in series to create a higher voltage.

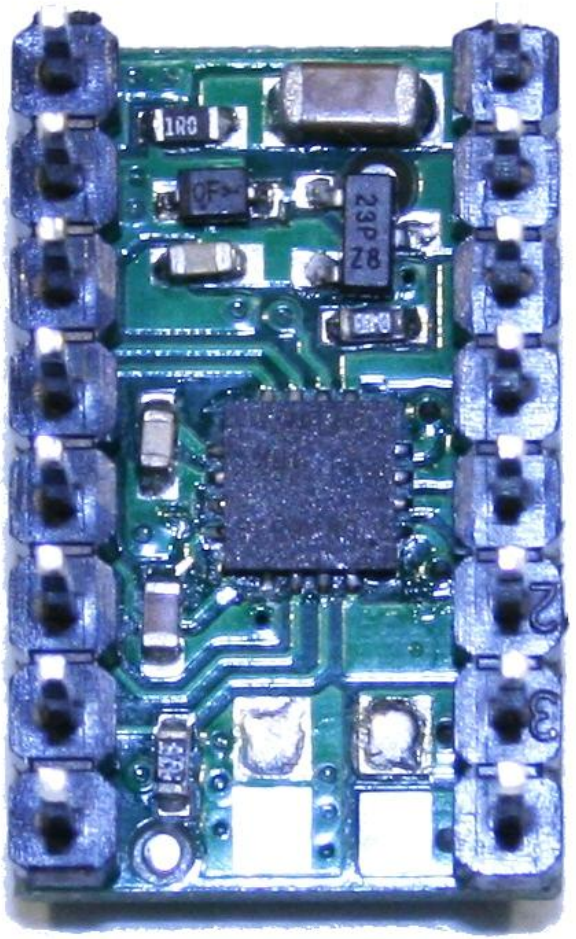
Processor Board

- The processor board is the brains of the robot. It controls everything and connects to all the sensors. This is the part that you program to control the actions of the robot.
- The processor board has multiple interfaces
  - Digital signals
  - Servo motor
  - Communications
  - Analog inputs
- More details will be covered later.



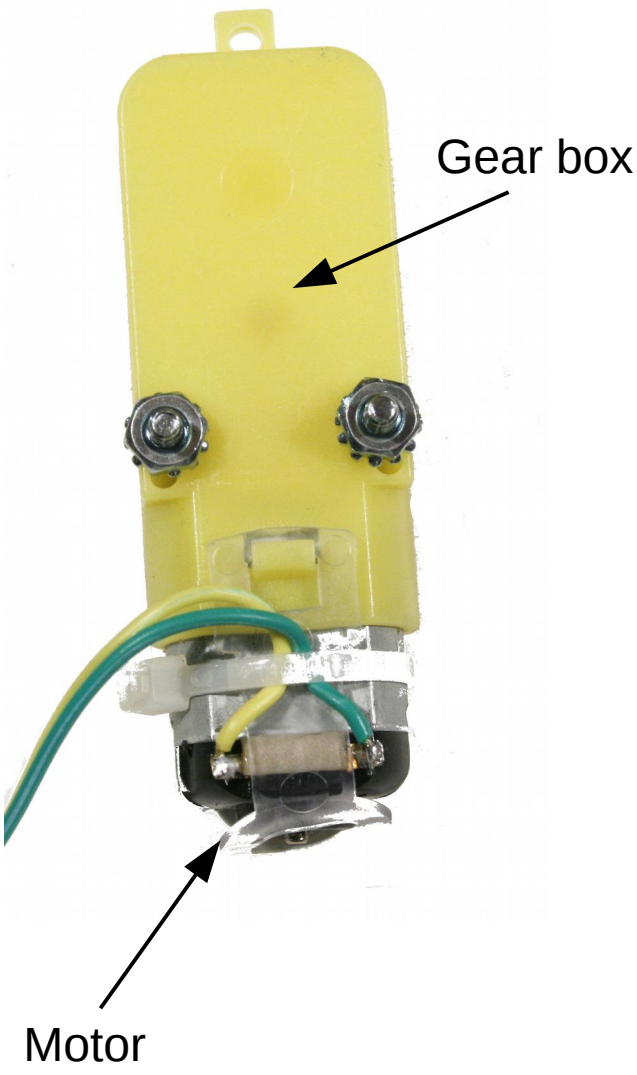
Motor Controller

- The motor controller is the interface between the motors and the processor board. It has circuitry to allow control of the motors and can handle the high currents required to operate the motors. The processor board cannot directly power the motors. The controller is capable of providing the needed current and is used as the interface.
- The black square in the picture to the right is the actual controller. It is an integrated circuit that contains all the circuitry to translate signals from the processor board to the operations of the motors. This controller can operate two sets of motors.
- More details are covered later.



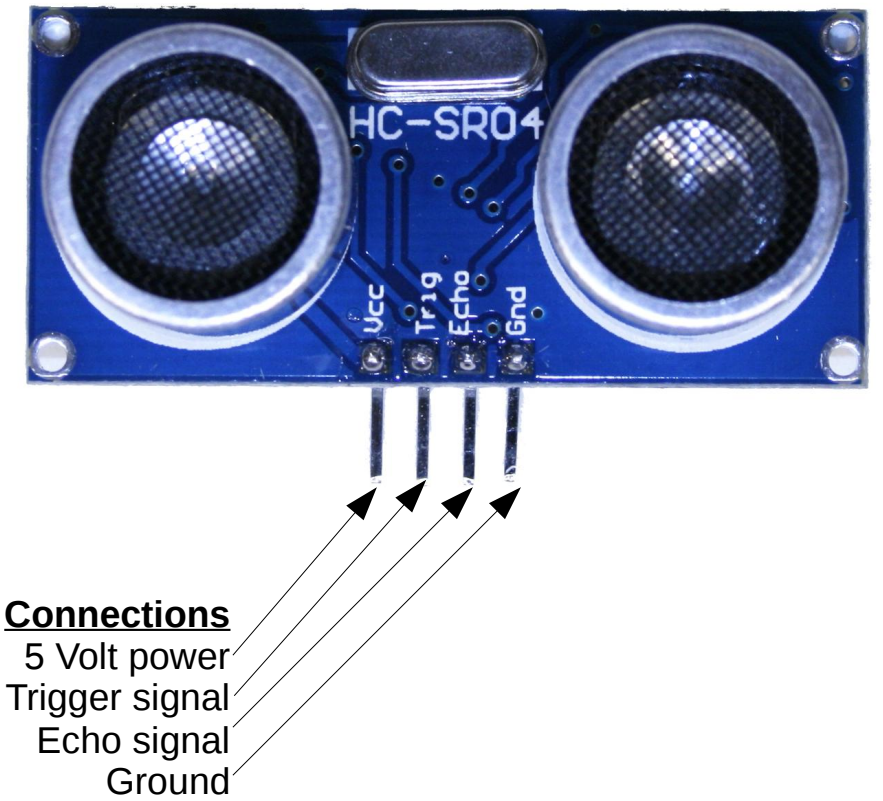
The Motors

- The motors are connected to gears that translate the high speed of the motors to a slower rotations speed. This also increases the power of the motor so the wheels can turn to move the robot.
- The motor is a small DC brush motor shown at the bottom. It is inserted into the yellow gear box.
- At the top side of the picture on the opposite side is a white shaft. The wheel attaches to the shaft.
- The gears convert the high speed rotation of the motor and produces a slower rotating shaft. The torque is also increased at the shaft.



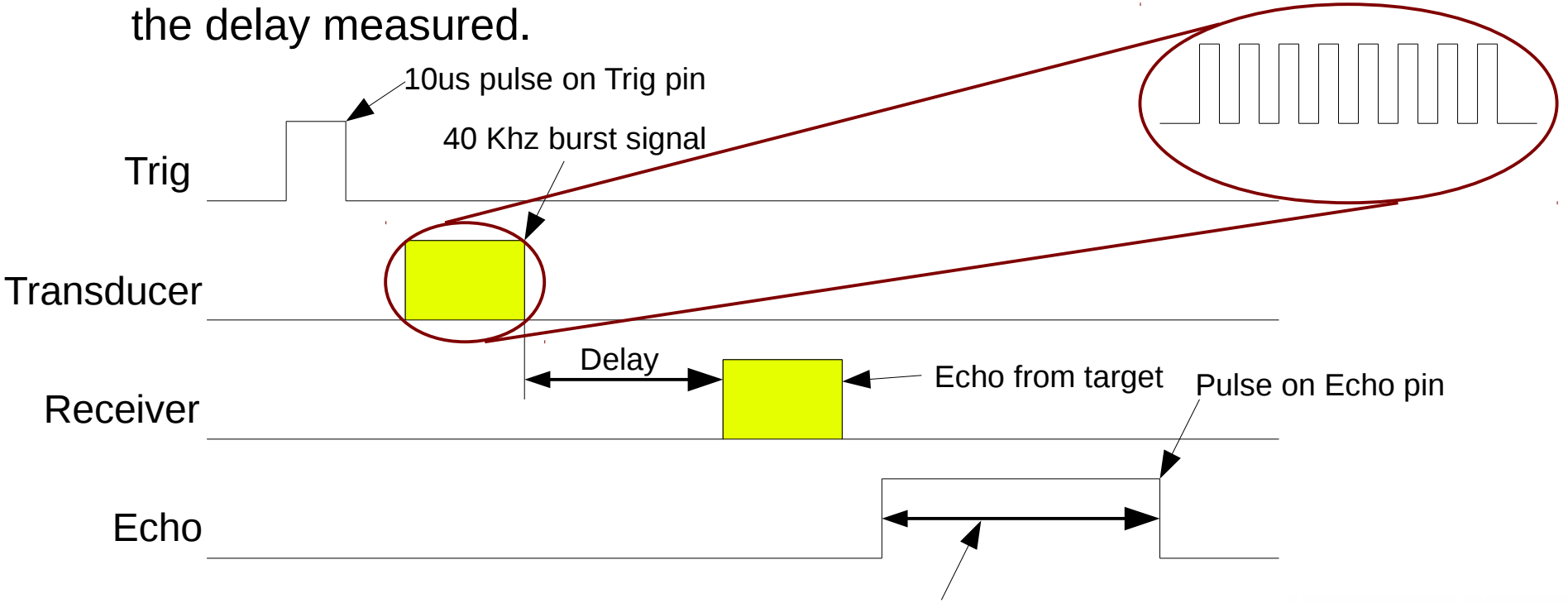
Ultrasonic Range Sensor

- The ultrasonic range sensor is a device that sends a short burst of sound and listens for the echo.
- The processor board starts the measurement by generating a pulse on the Trig pin.
- It them measures the size of the pulse on the Echo pin.
- The processor calculates the distance based on the size of the pulse width.
- Distance is calculated by dividing the pulse width measured in microseconds by 58. Answer is in centimeters.



Ultrasonic Range Sensor Operation

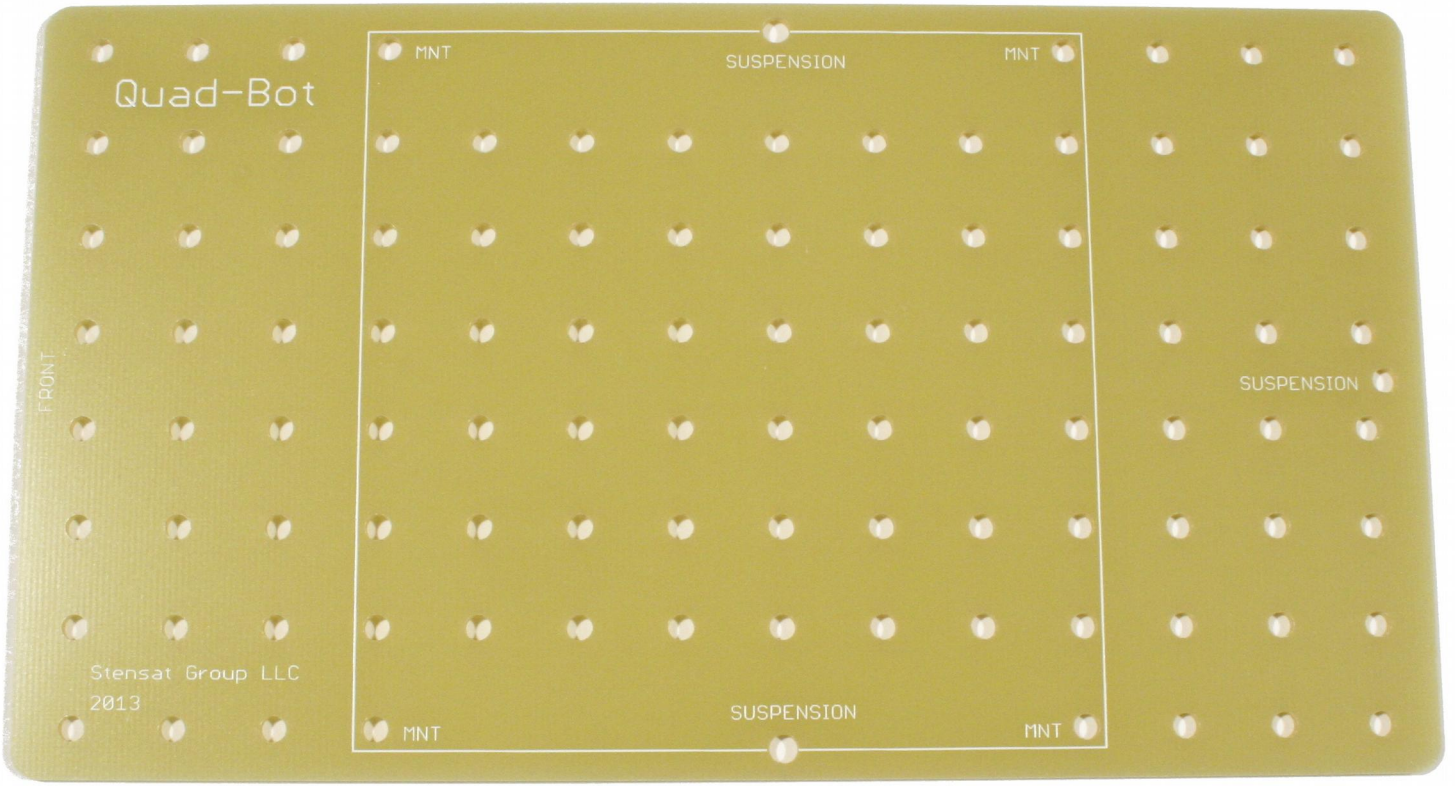
- The ultrasonic range sensor operates in a specific sequence.
- It waits for a trigger signal. The trigger is a 10us pulse. Once the trigger is detected, the sensor generates a short signal at 40 KHz.
- It then waits for an echo and measures the time from sending the short burst to receiving the echo.
- The sensor then generates a pulse on the echo with a length proportional to the delay measured.



distance = pulse width (us) / 58

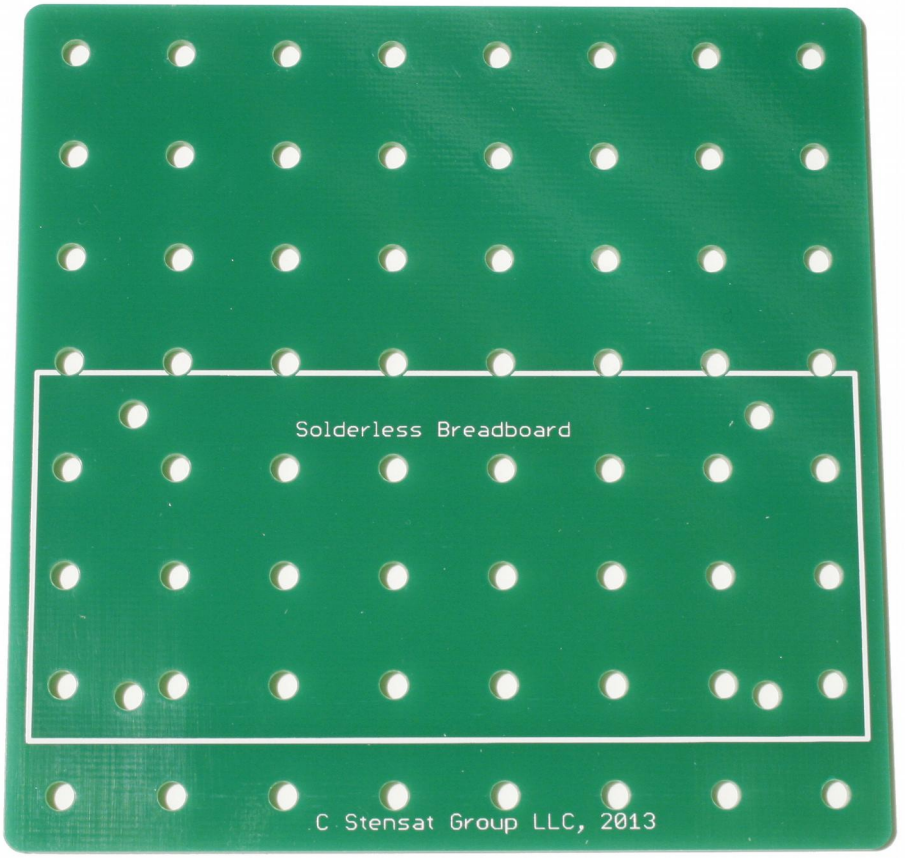
Base Plate

- The base plate is a fiberglass board with holes arranged in a .5 inch grid pattern. There are also other holes to mount specific things such as the suspension systems and the electronics. There are markings on the board identifying specific functions for the holes.



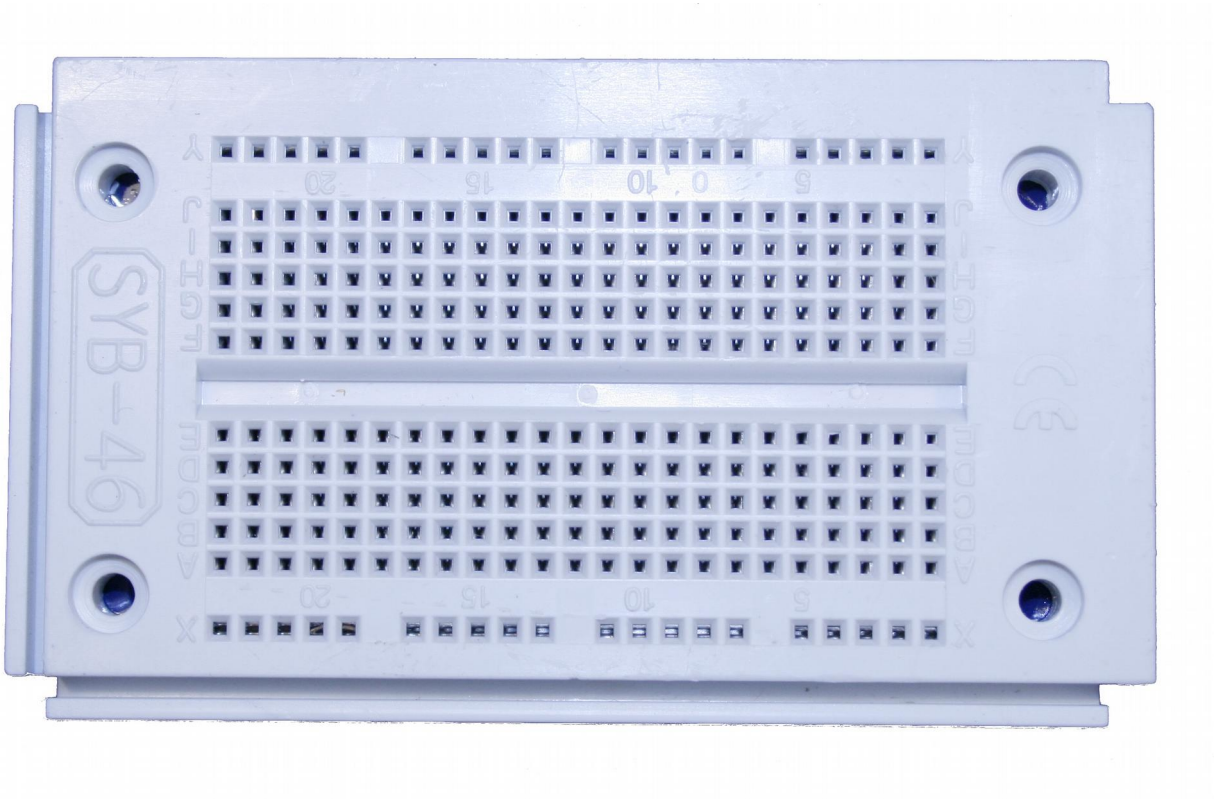
Electronics Plate

- This is another fiberglass board that is used to hold the processor board and solderless bread board.



Solderless Bread Board

- The solderless bread board allows you to wire up circuits. It will be used to connect the motor controller to the motors and the processor board. It is also used for connecting the sensors.



End of Section

- In this section, you learned the parts of the robot and how they all tie together.

Assembly

Robot Assembly

- The robot assembly starts with the wheels, motors and suspension system.
- Once the suspension system with the motors and wheels installed are attached to the base plate, the electronics plate is assembled and installed.

Parts List

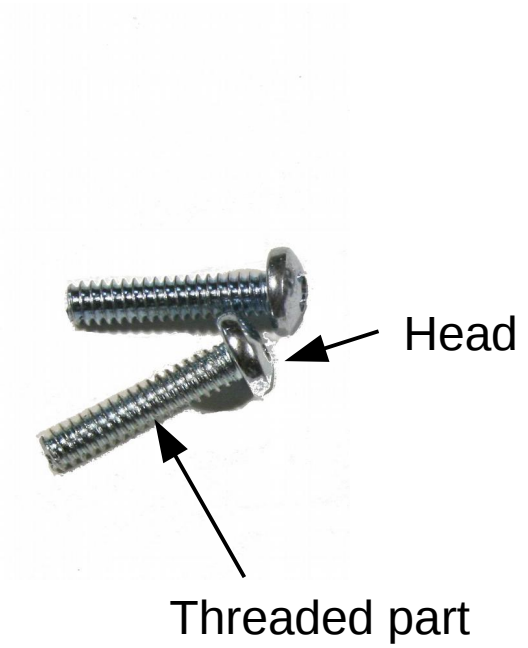
- 6 – 3/8” 4-40 screws
- 5 – 1/4” 4-40 screws
- 8 – 1/2” 4-40 screws
- 9 – 1” 4-40 screws
- 2 – 1/4” 6-32 screws
- 2 – 3/8” 8-32 screws
- 2 – 8-32 nylon lock nuts
- 33 – 4-40 Kep nuts
- 7 – 4-40 nylon lock nuts
- 1 – 3/4” standoff
- 4 – 1.5” standoffs
- 3 – small right angle brackets
- 2 – large right angle brackets
- 1 – Fiberglass base plate
- 1 – Suspension System
- Dual H-Bridge Driver Module
- 4 – geared motors with wheels
- 1 – electronics plate
- 1 – processor board
- 1 – solderless bread board
- 10 – jumpers
- 1 – ultrasonic range sensor
- 1 – LED
- 1 – 270 ohm resistor
- 1 – Photoresistor
- 1 – 4.7K ohm resistor
- 1 – Infrared receiver
- 1 – battery holder
- 1 – USB cable

Tools Needed

- Philips screw driver
- 1/4 inch nut driver
- 11/32 inch nut driver

Definition of Components

- Screw – A cylindrical device with a raised helical thread running around it used to join things together.
  - Sizes
    - 4-40 means it is a #4 size screw with threads that wrap around 40 times per inch length.
    - #4 size is .112 inches diameter
    - #6 is .138 inches diameter
    - #8 is .164 inches diameter
  - Length is how long the threaded part of screw is.



Definition of Components

- Nut – A device that mates to a screw to secure things together. The sizing is specified the same way, ie 4-40 or 6-32.
- Kep nut is a nut with an integrated lock washer.
- A nylon lock nut is a nut with a piece of nylon material inserted to keep the nut from spinning freely. It is used to join things together but let them move against each other.

Nylon Lock Nut

Kep Nut

Definition of Components

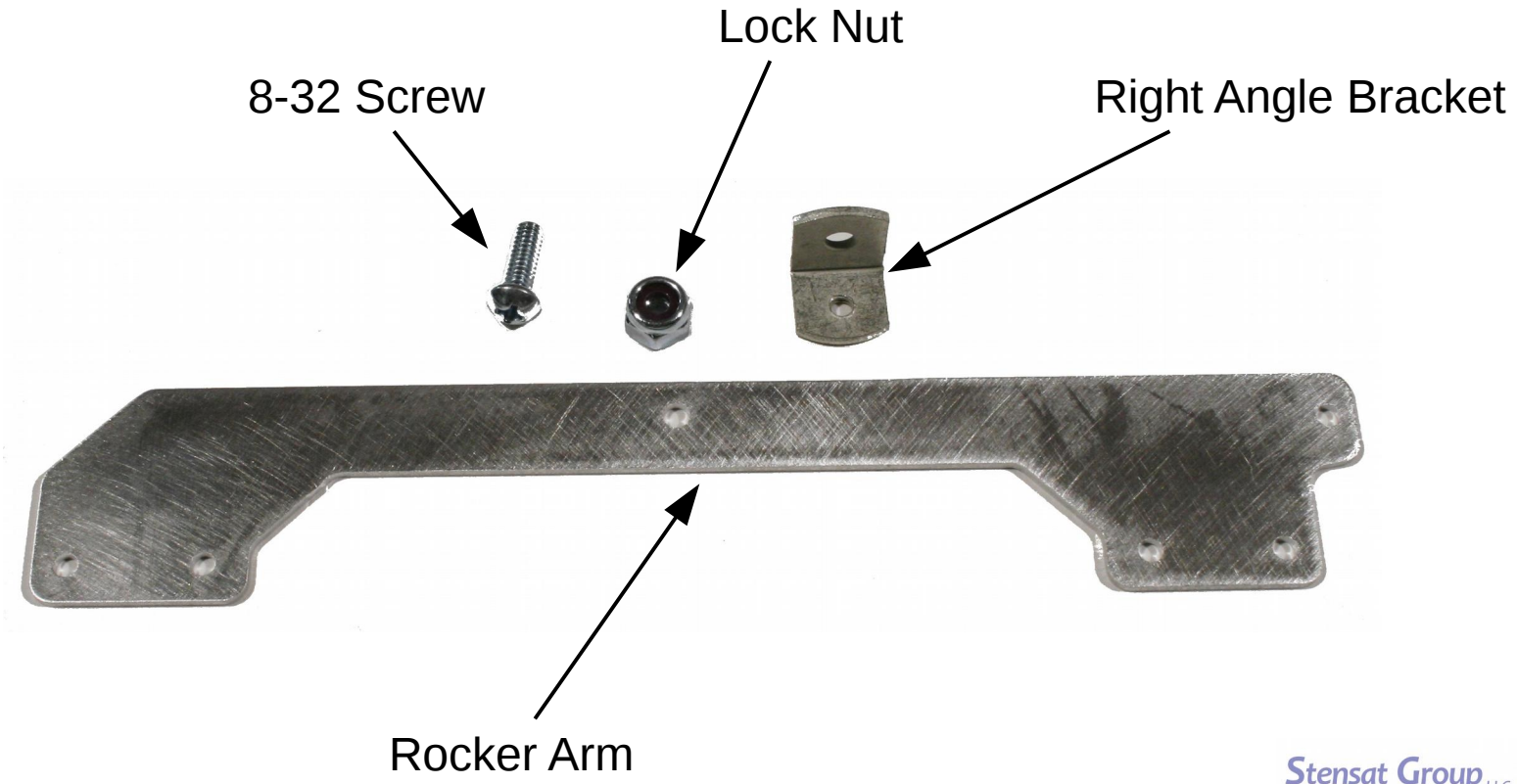
- Right angle bracket – A device that allows two things to be attached at right angles to each other.
- Standoff – A device that allows things to be attached to each other at a distance. Allows stacking. One end can be threaded like a screw and the other hollowed and threaded to be like a nut. They are made in different lengths. The rovers uses a 1.5 inch and ¾ inch long standoffs.

Standoff

Right Angle Bracket

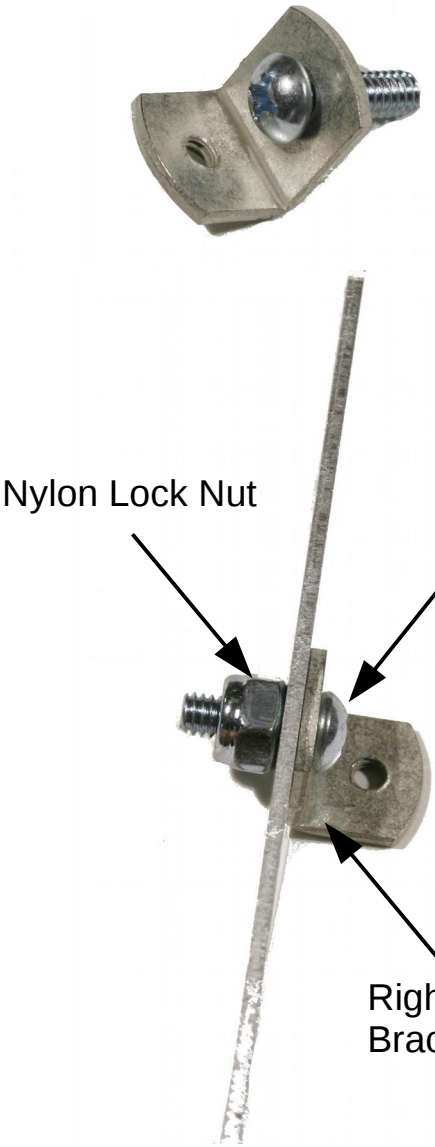
Start of the Assembly

- The assembly will start with the wheels and suspension system.
- Below is a picture of the components. One side is shown. The second side will be assembled as a mirror image.



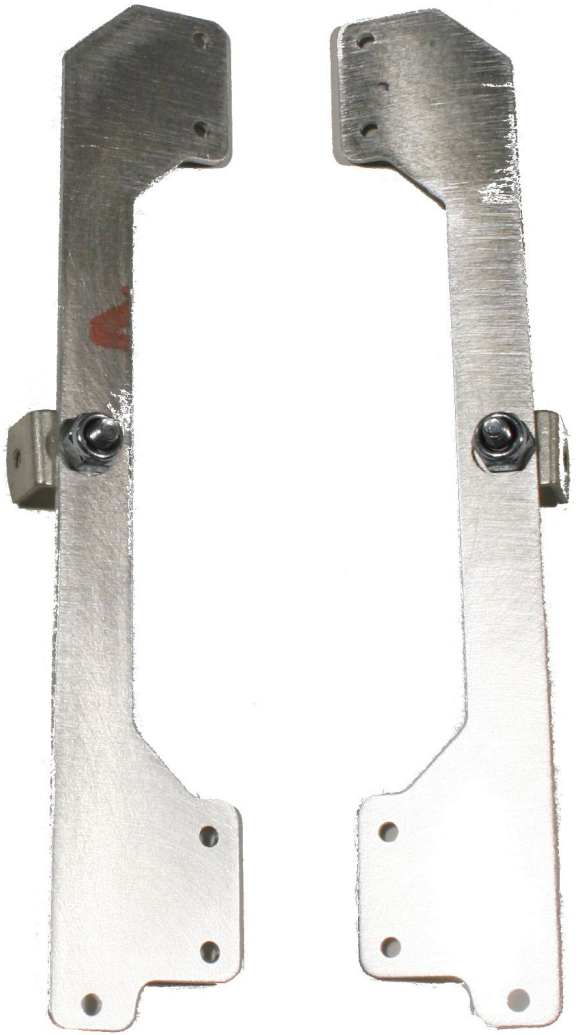
Rocker Arm Assembly

- The rocker arm will mount to the robot base plate using the right angle bracket.
- The 8-32 screw is inserted into the larger hole on the right angle bracket as shown in the top right.
- Insert the screw with the bracket through the hole in the center of the rocker arm.
- Insert the lock nut onto the screw. The lock nut has a nylon insert that makes the nut go on tight. A 11/32 nut driver or pliers will be needed to tighten the nut onto the screw.
- Tighten the nut all the way. Once tight, turn back the nut driver a little bit so the rocker arm can move freely but have little side to side wiggle.



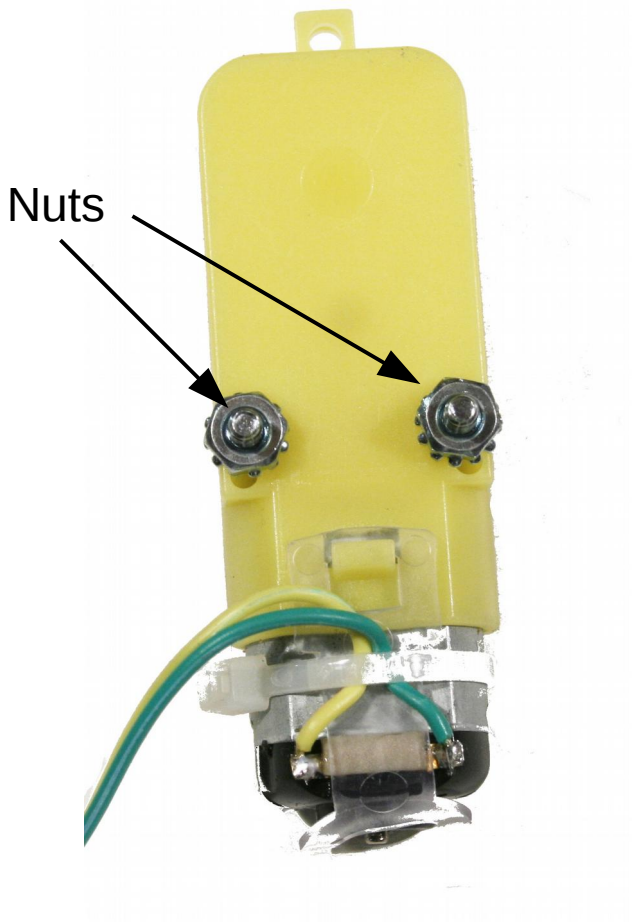
Rocker Arm Assembly

- Assemble the second rocker arm the opposite of the first one.
- The two rocker arms should look like the picture to the right.



Motor Preparations

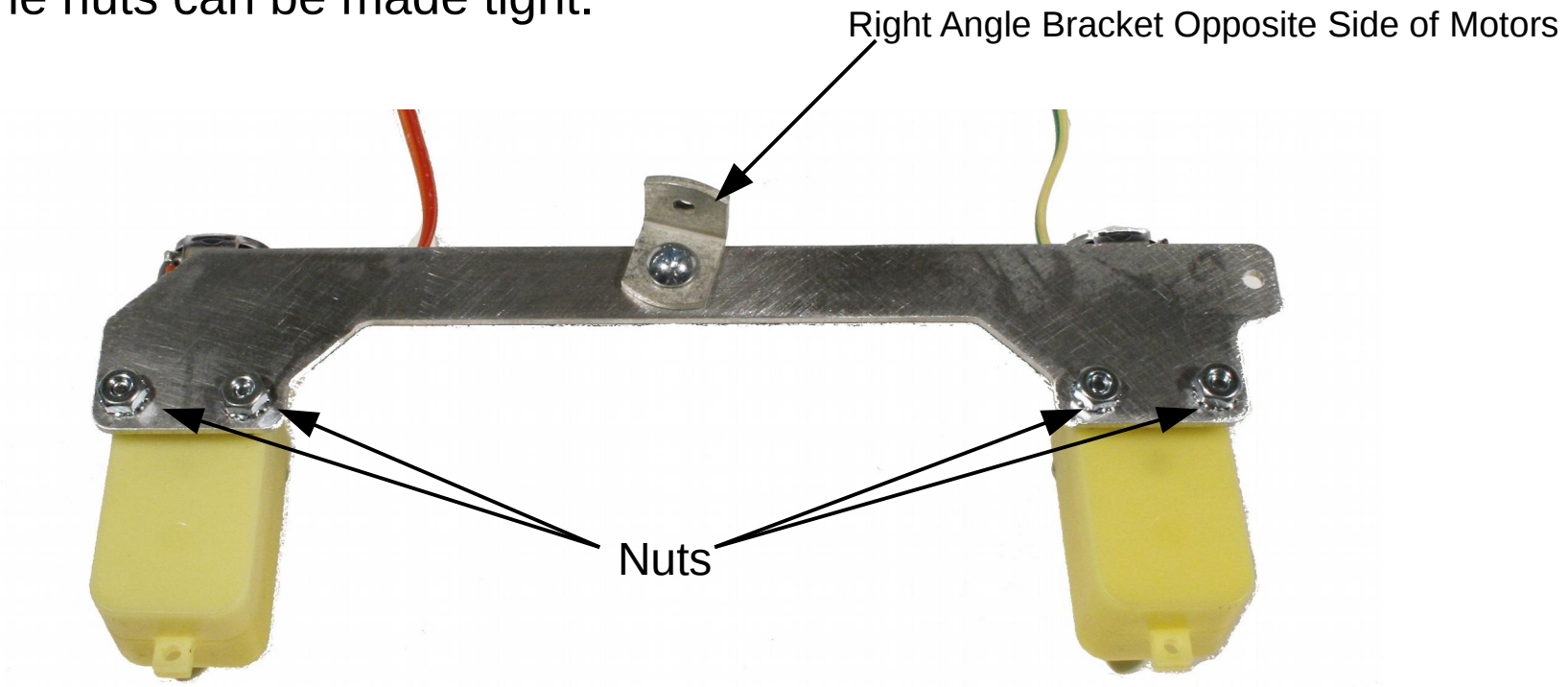
- The four motors are to be prepared the same.
- Take two 1 inch screws and insert them from the white shaft side of the motors.
- Install a nut on each screw and tighten to be snug. Do not over tighten as that can damage the plastic structure of the motors.
- Prepare all four motors the same.
- The nuts are installed to offset the motor from the rocker arm. This allows space for the wire to be between the motor and the rocker arm.
- The tie wrap is used as a strain relief to protect the wire from breaking off the motor connection. This also allows the user to easily make repairs or replace the wiring if desired.



White Wheel Shaft on Opposite Side

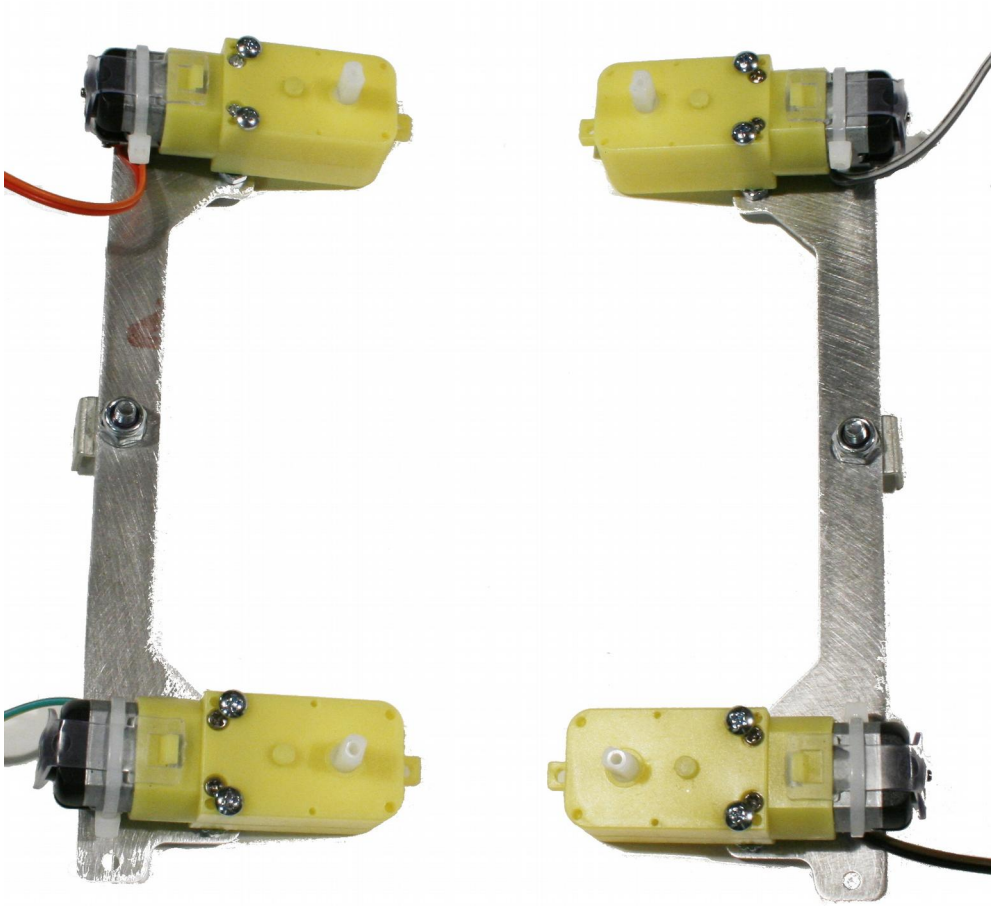
Motor Installation

- The motors are to be installed on the rocker arms.
- The motors mount on the opposite side of the right angle bracket.
- Secure each motor with two nuts.
- The nuts can be made tight.



Mounting Motors

- Install the motors on the second rocker arm.
- They should look like the mirror of each other.
- Make sure the bracket is on the opposite side of the rocker arm from the motors.
- Make sure the wires from the motors are router toward the center of the rocker arms.

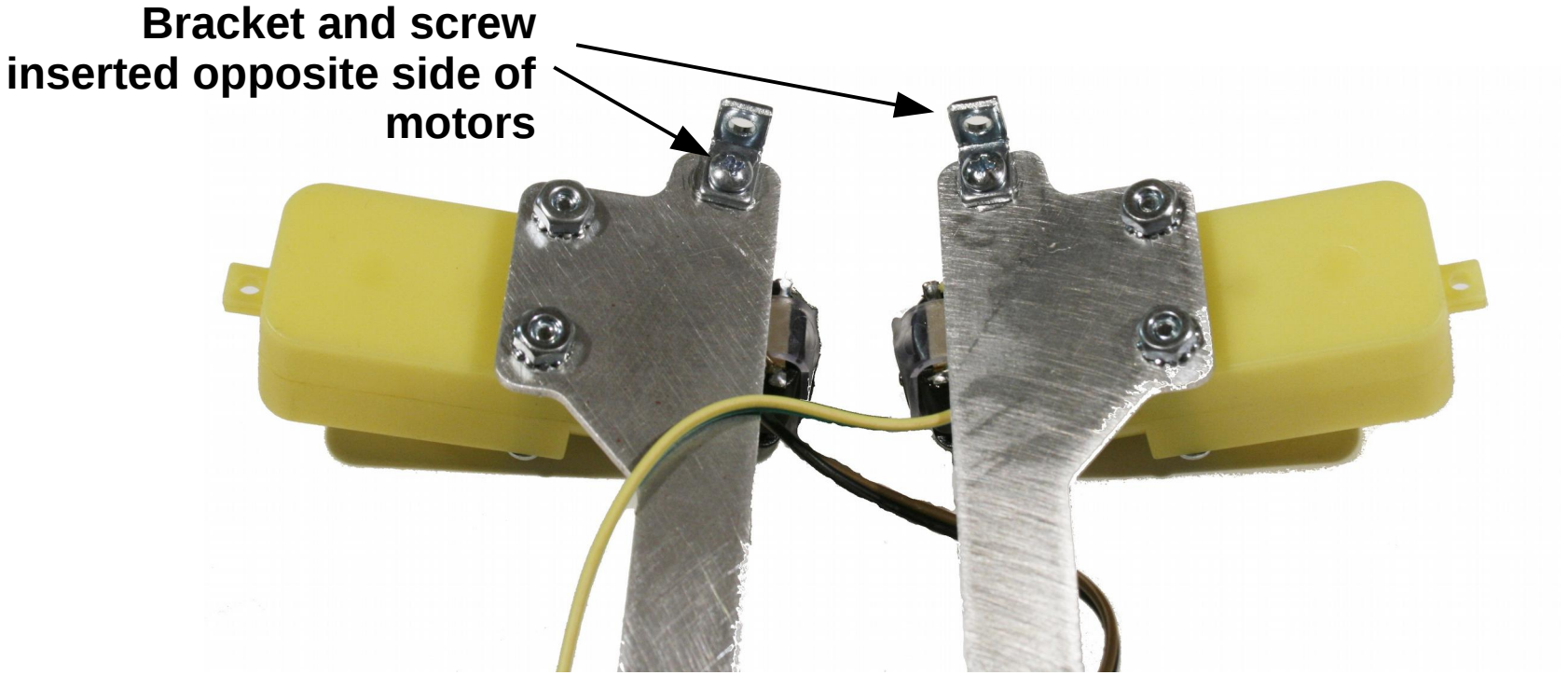


Suspension Linkage Bracket

- Before installing the rocker arms, small right angle brackets for the suspension linkage need to be installed.
- Locate the two small right angle brackets, 4-40 lock nuts, and 3/8 inch 4-40 screws.
- Insert the screws into the smaller hole of the right angle brackets. The hole is threaded so a screw driver is needed. Screw in the screws as shown in the lower right.

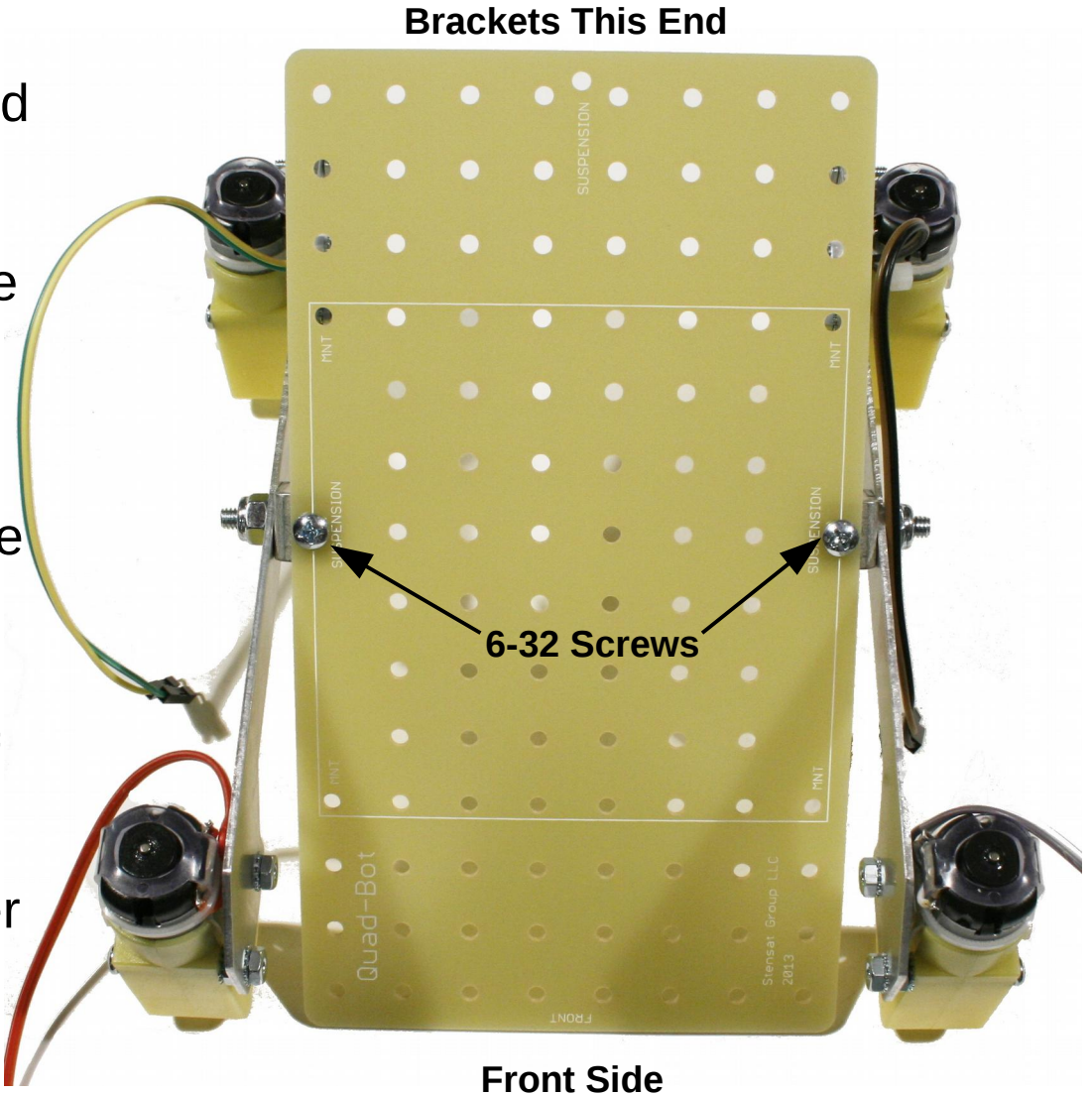
Suspension Linkage Bracket

- Mount the brackets onto the back side of the rocker arms as shown.
- Install the 4-40 lock nut on to the screw from the motor side and tighten with a ¼ inch nut driver.
- Tighten until the nut does not turn any more.
- Loosen the nut one turn. The right angle bracket needs to be loose.



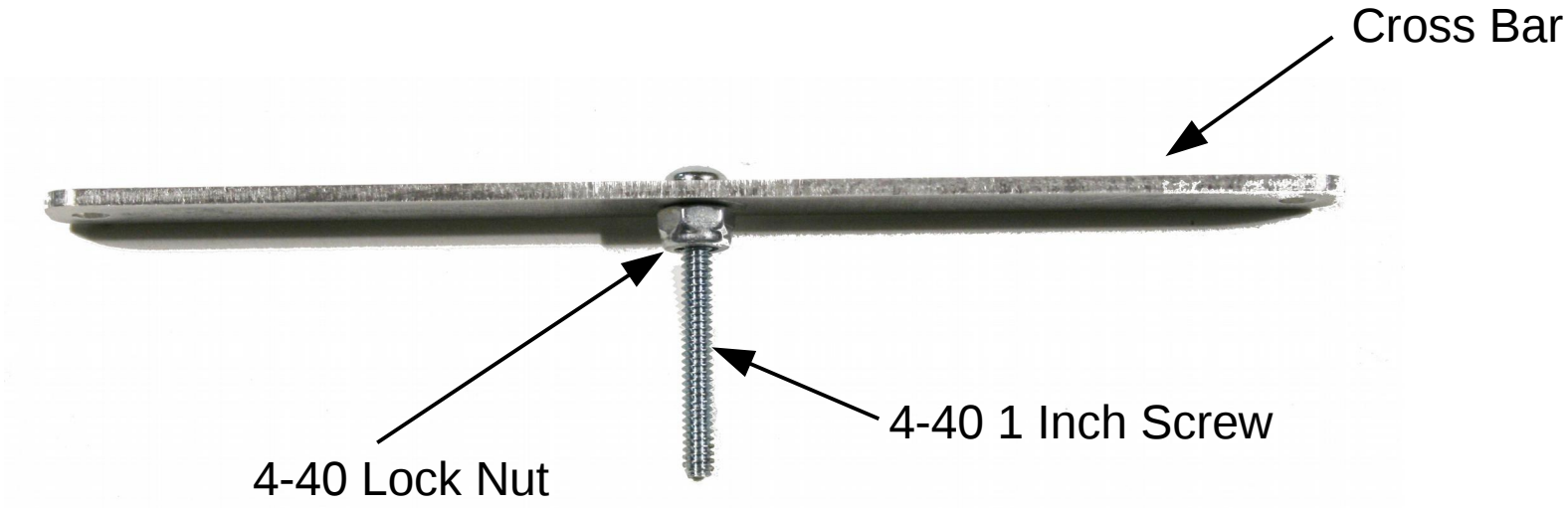
Mounting the Rocker Arms

- The rocker arms are to be mounted to the base plate. The end of the rocker arm with the right angle brackets must be opposite the side marked Front on the base plate.
- Use the ¼ inch 6-32 screws.
- Insert the screws at the center hole along the edges marked “suspension”
- Screw in the right angle bracket of the rocker arm.
- Before tightening, adjust the rocker arms so they are parallel with the edge of the base plate.
- Tighten the screws when the rocker arms are parallel.



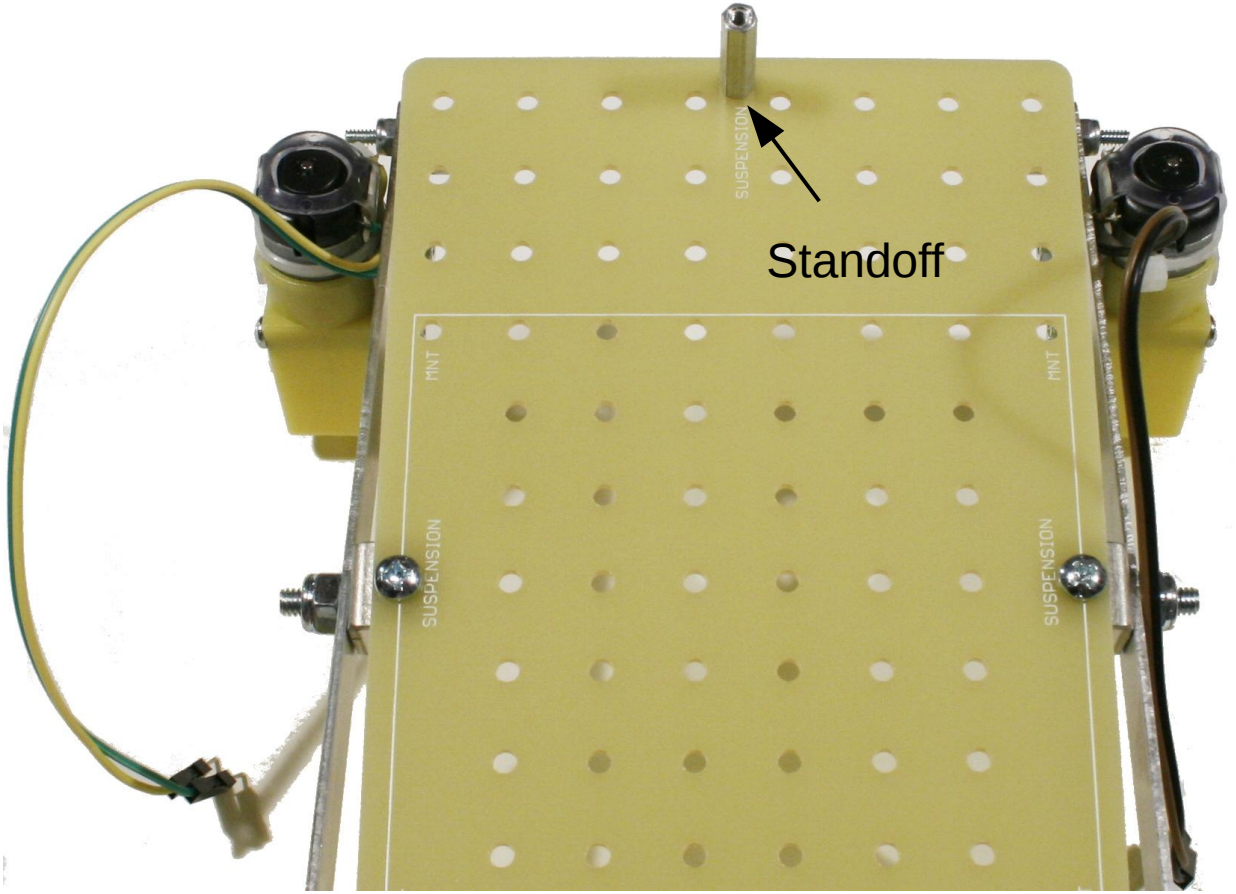
Suspension Linkage

- Now, assemble the cross bar.
- Get a 1 inch 4-40 screw and 4-40 lock nut.
- Insert the screw into the center hole of the cross bar.
- Insert the lock nut and tighten all the way.
- Once tightened, loosen the lock nut about a ½ turn.
- The cross bar should rotate freely on the screw.



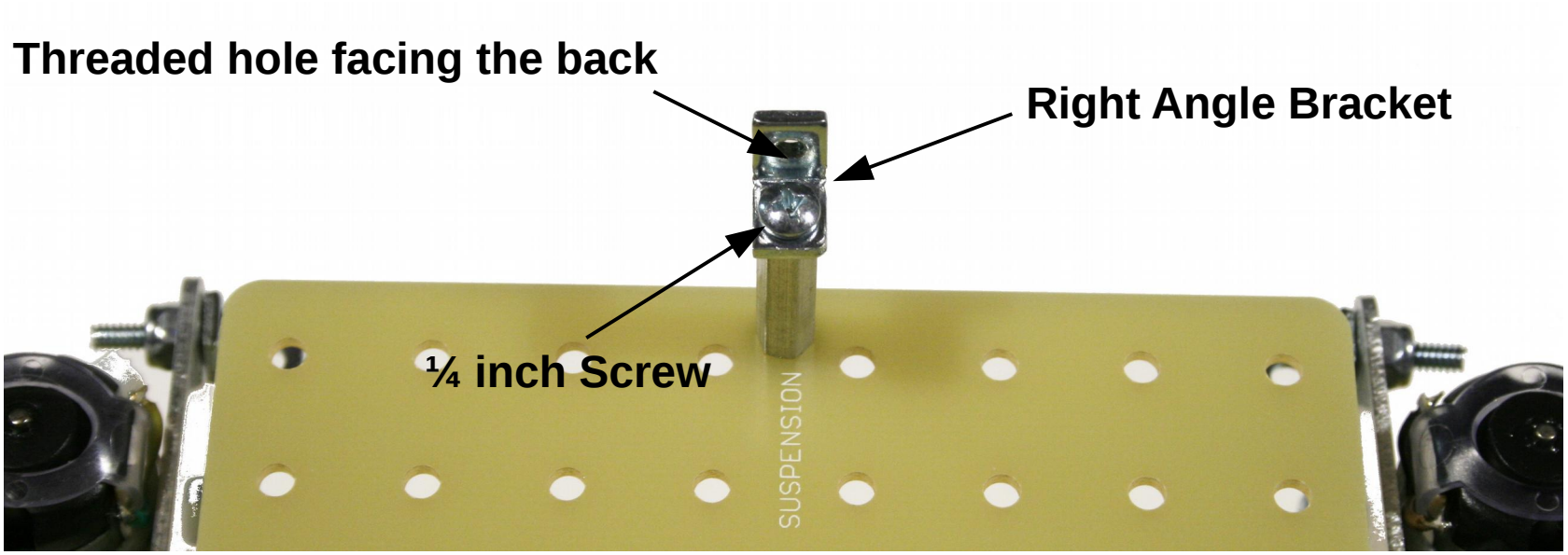
Cross Bar Mount

- Install a ¾ inch standoff in the center back hole of the base plate. The hole is marked “Suspension.” Insert the threaded side of the standoff in the hole and secure with a 4-40 nut with washer from the underside of the base plate. Do not over tighten as you may break the standoff. It is made of aluminum. Once the nut is tightened by hand, **a quarter turn with the nut driver is sufficient. Too tight and you can break the standoff.**



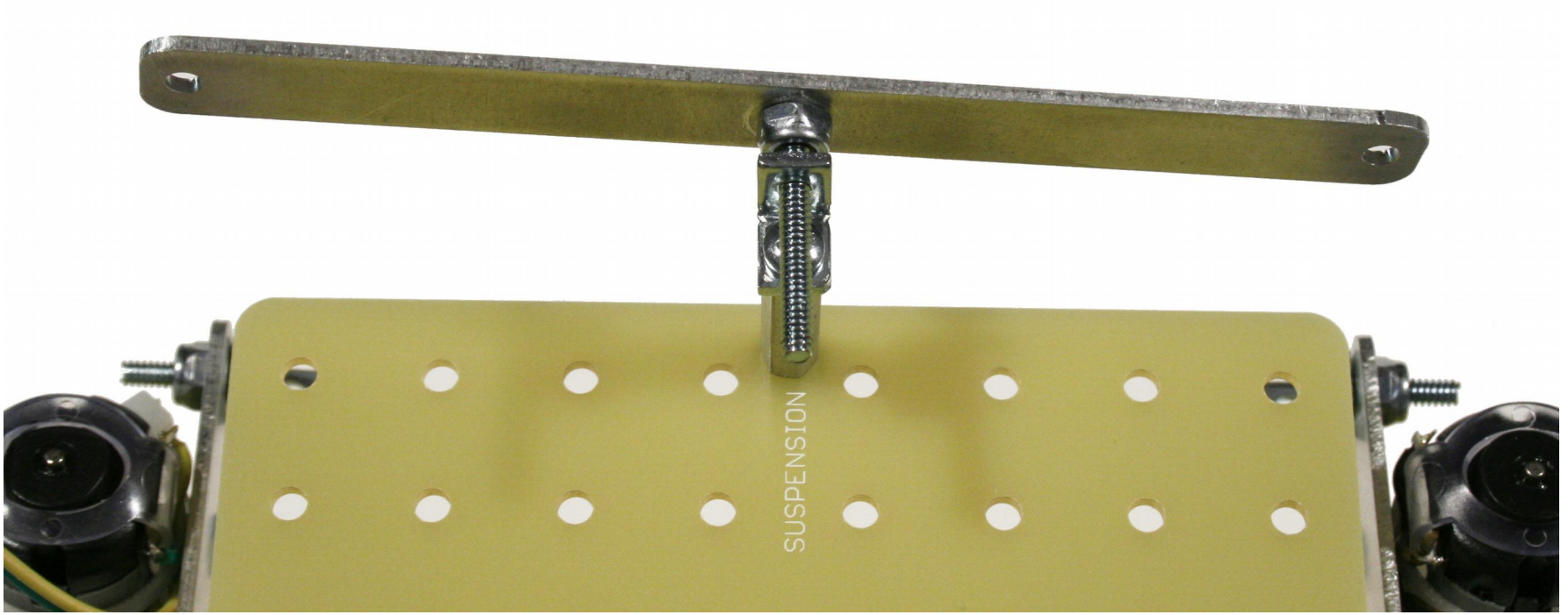
Cross Bar Mounting Bracket

- Install a small right angle bracket on top of the standoff and secure with a ¼ inch 4-40 screw. Use the larger hole of the right angle bracket.
- Align the bracket as shown.



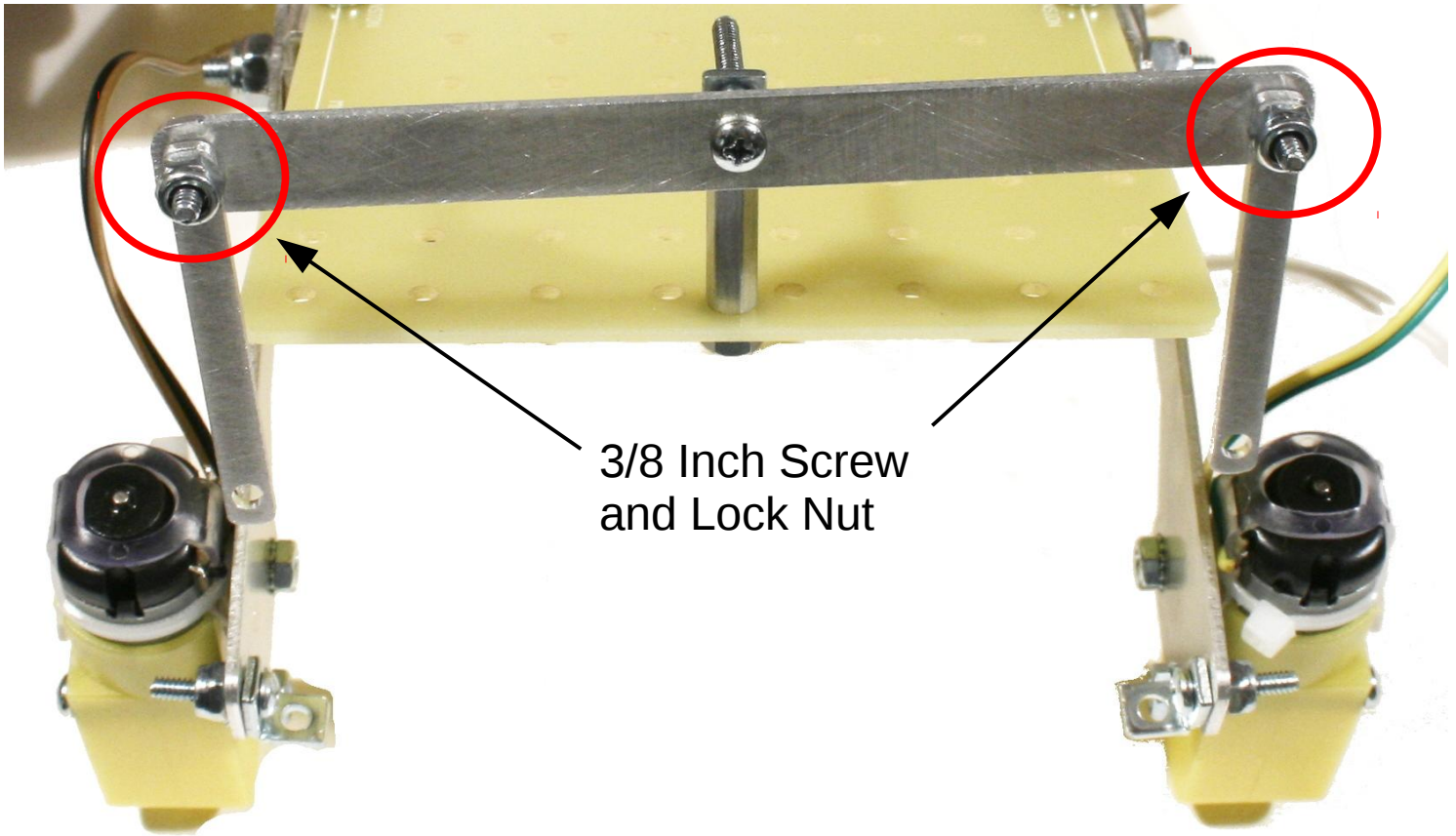
Installing Cross Bar

- Screw the cross bar into the right angle bracket secured on the standoff. Don't screw in all the way. Leave about ¼ inch of threads.



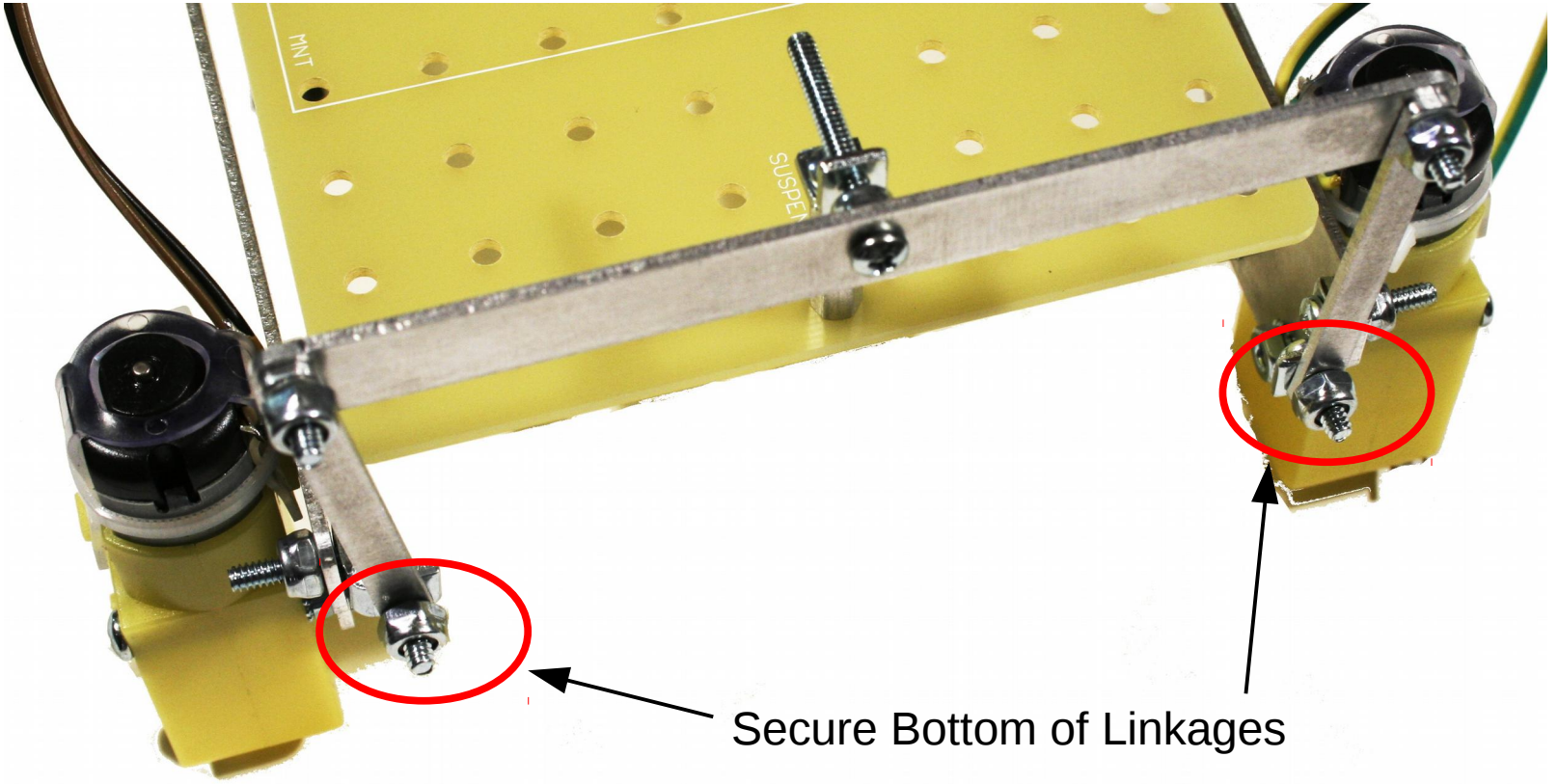
Installing the Linkages

- Install the two linkages on to the cross bar. Use the 3/8 inch long 4-40 screws and lock nuts.
- Tighten the lock nuts and then loosen by a ½ turn of the nut driver.



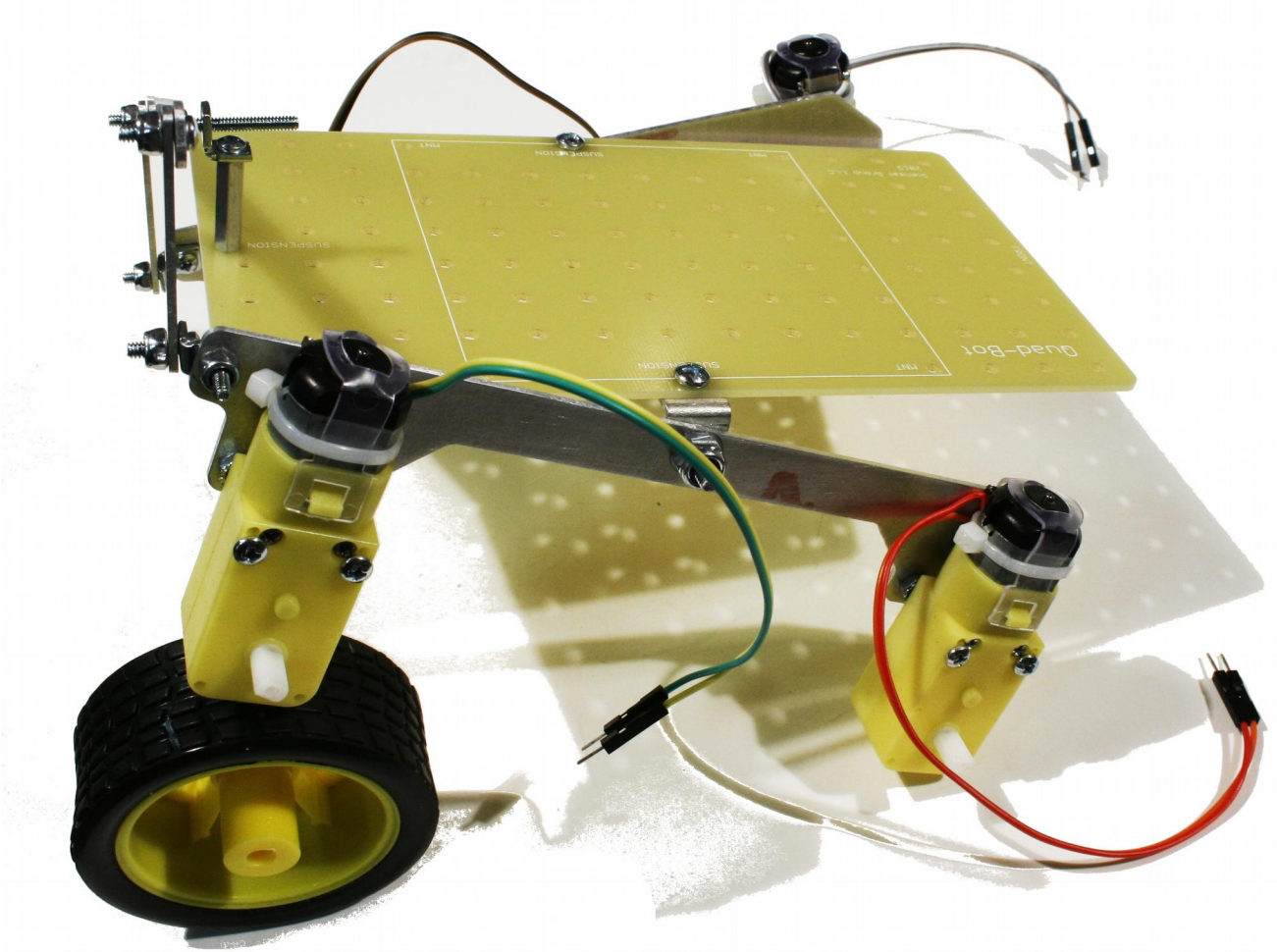
Linking the Linkages

- Use another 3/8 inch 4-40 screw and 4-40 lock nut and secure the bottom end of the linkages to the brackets on the rocket arm.
- Tighten the lock nuts and then loosen with a half turn.



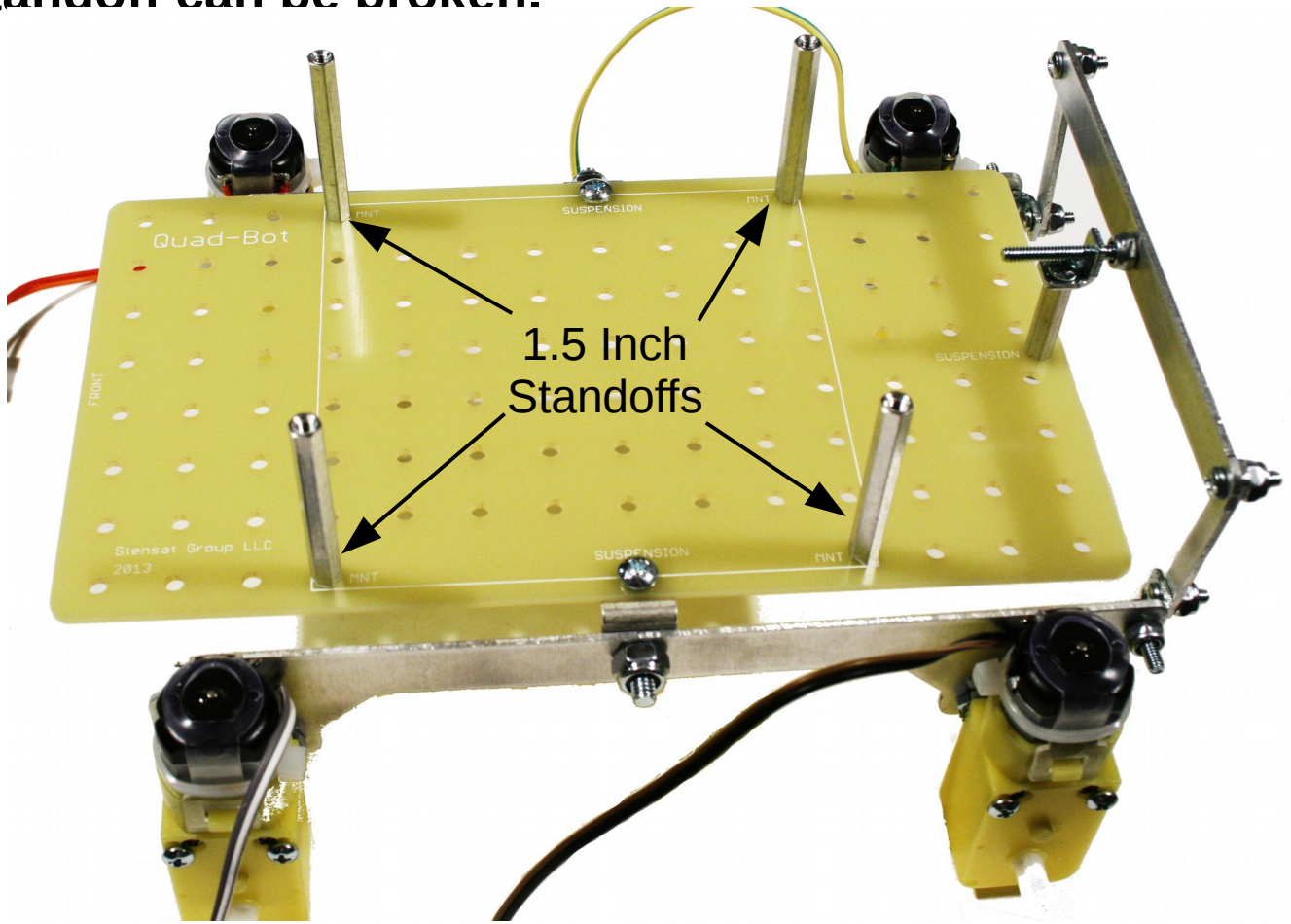
Testing the Suspension

- Make sure all linkage connections are loose. This allows the rocker arms to move freely and keep the base plate stable.
- Lift one motor and make sure all the other three motors stay on the surface. Up two three inches of height should be possible.



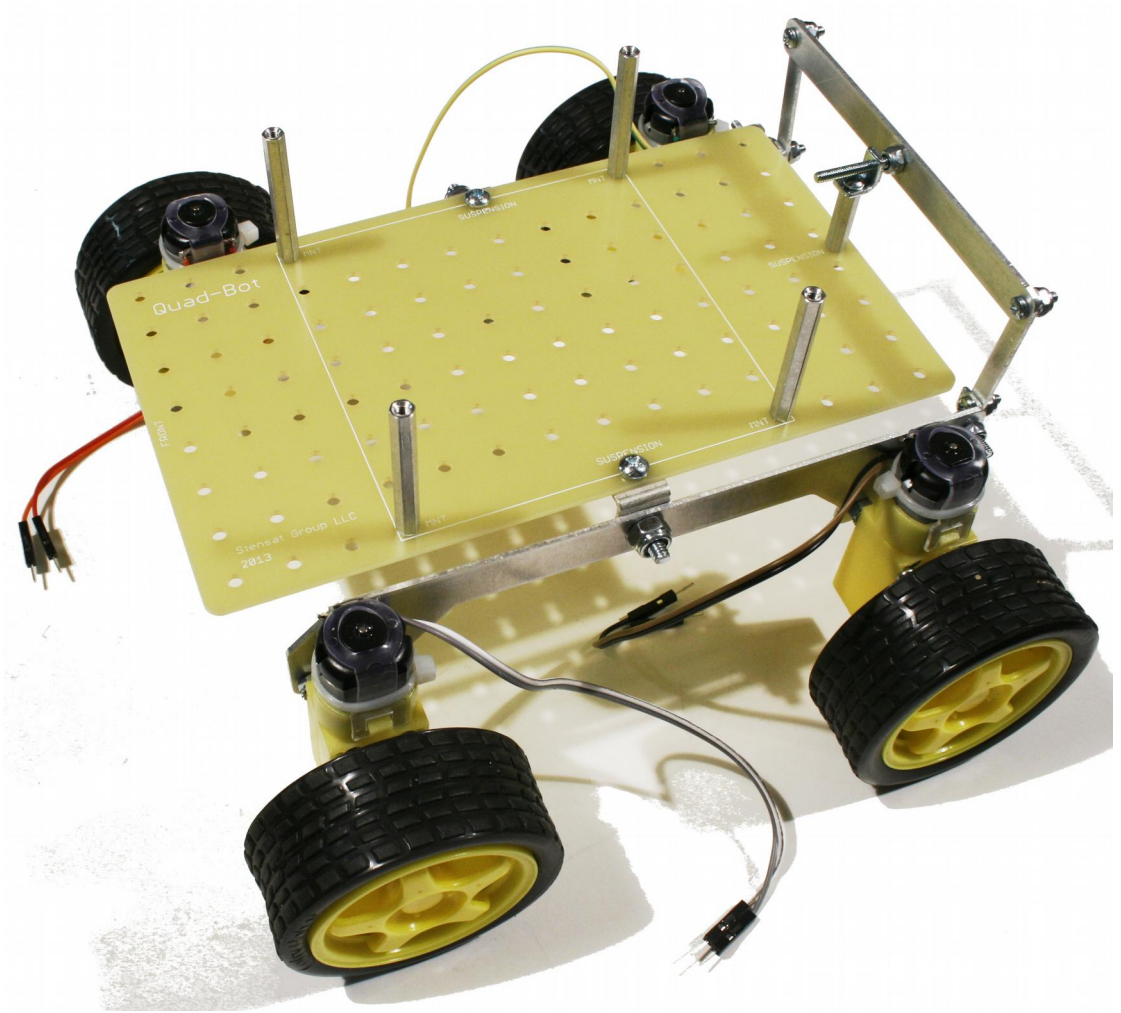
Preparing for Mounting Electronics

- Install the threaded side of the four 1.5 inch standoffs in the holes marked “MNT”.
- Secure the standoffs with the nuts and hand tighten. Use a nut driver to tighten with no more than a quarter turn. **Again, do not over tighten as the standoff can be broken.**



Install the Wheels

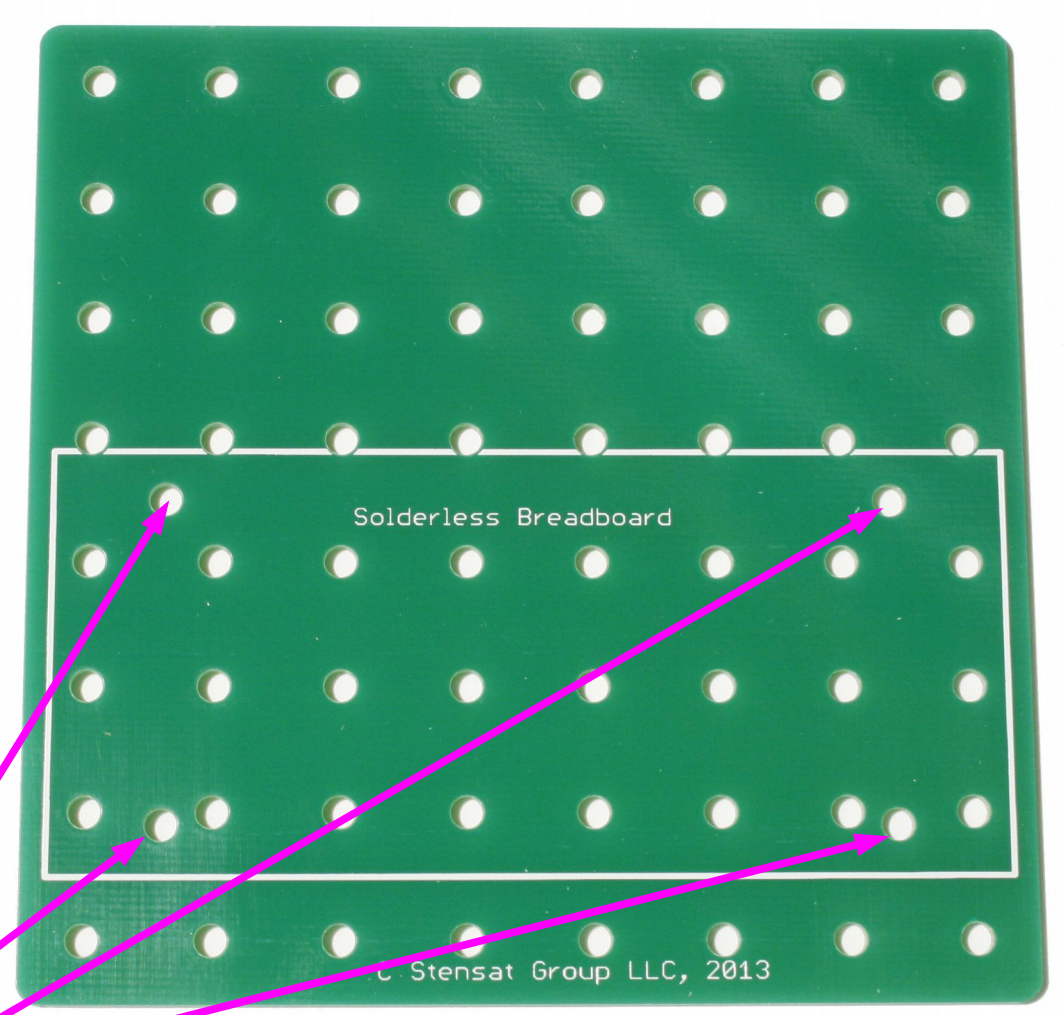
- Next, install the wheels on each motor. The motor shaft is keyed with two flat spots. The motors slide onto the shaft.



Electronics Base Plate

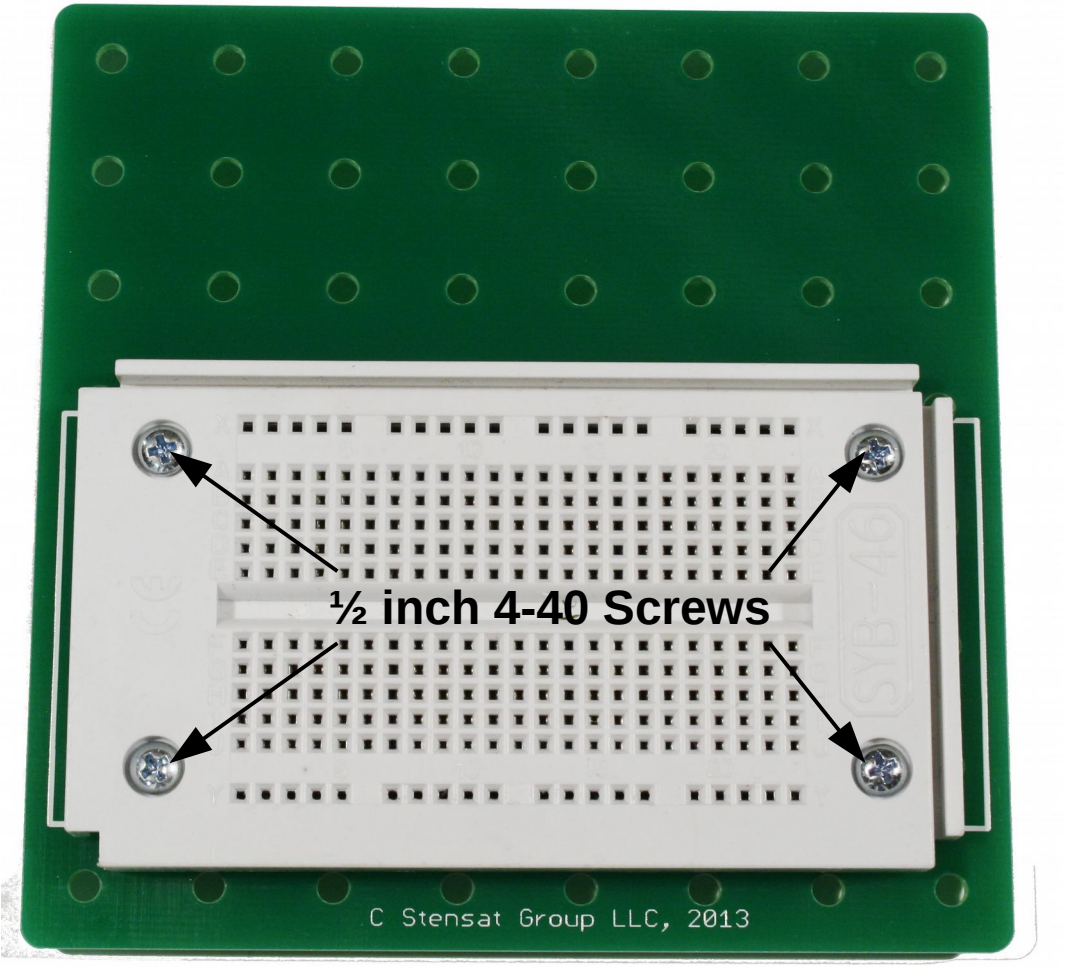
•Base plate is for mounting solderless bread board and processor board
•Solderless bread board is to be mounted in the marked rectangular area .

- Use the 1/2 inch screws and nuts to secure as shown in the next page.
- Insert the screws from the top through the solderless bread board and secure with 4-40 nuts on the back side.
- Make sure the solderless bread board is oriented as shown in the picture on the next page.



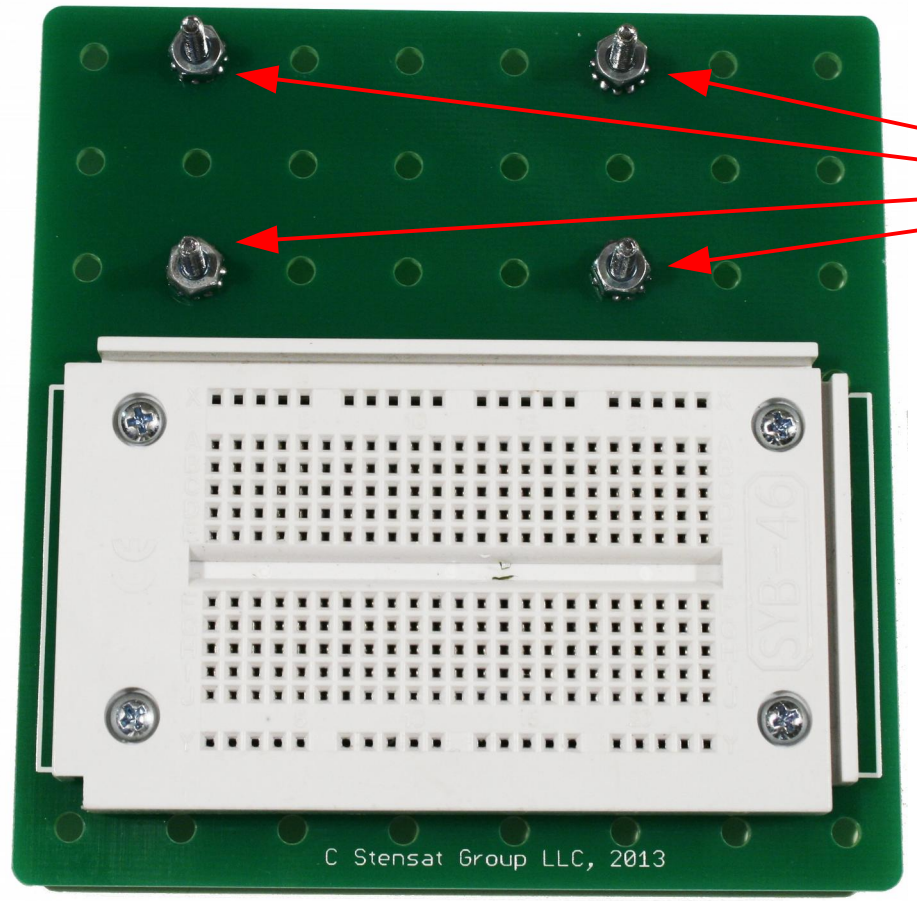
Mount the Solderless Breadboard

- The solderless breadboard is mounted as shown.



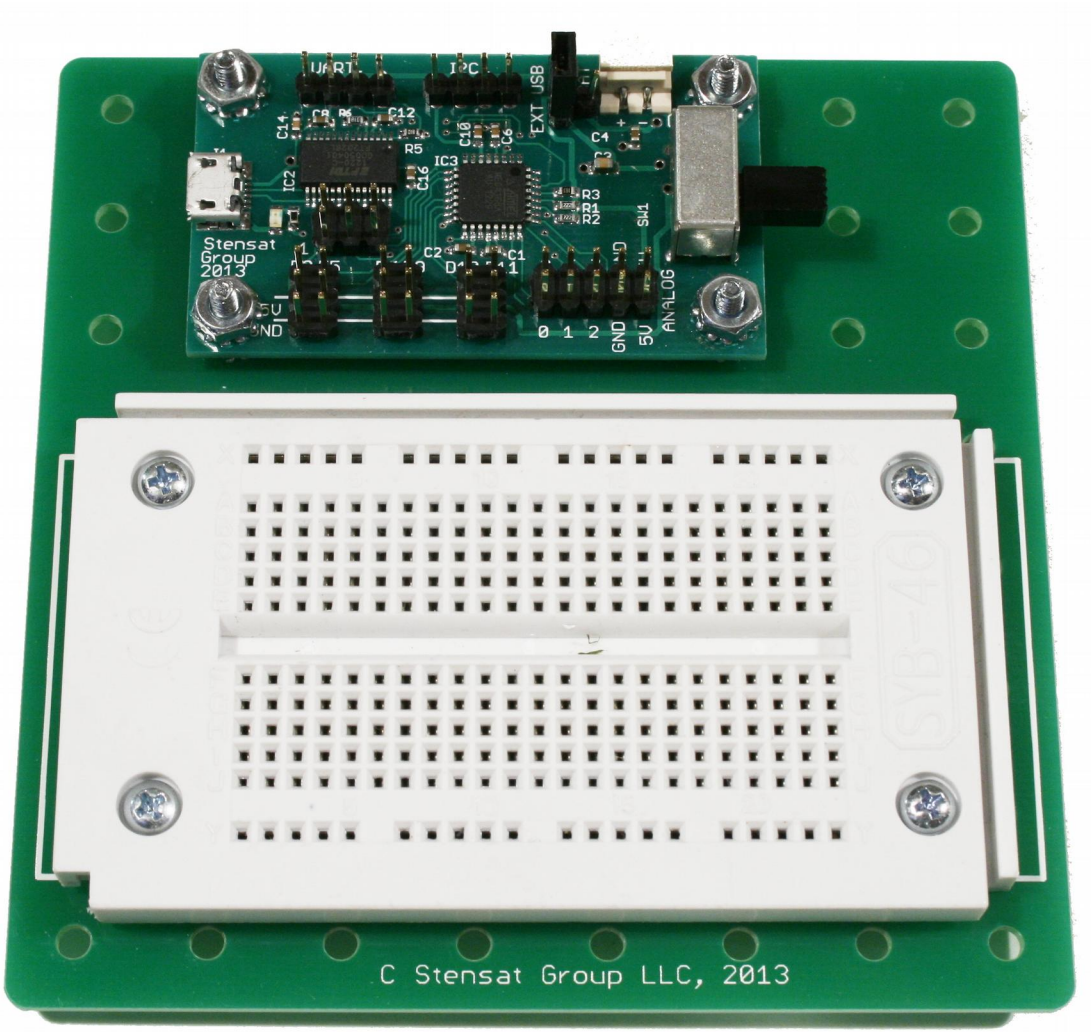
Processor Board Mount

- The process board is mounted differently.
  - Insert screws from the back side and install a nut on each screw. The nuts will serve as standoffs for the processor board. Look at the picture what holes are used.



Electronics Base Plate Assembly

- Place the processor board on top of the nuts.
- Insert another set of four nuts to secure the processor board.



End of Section

- At this point, the robot structure and wheels are assembled along with the suspension system.
- The electronics plate is assembled but not installed on the robot at this time. The next section will focus on learning how the electronics works and how to use it and add circuits.

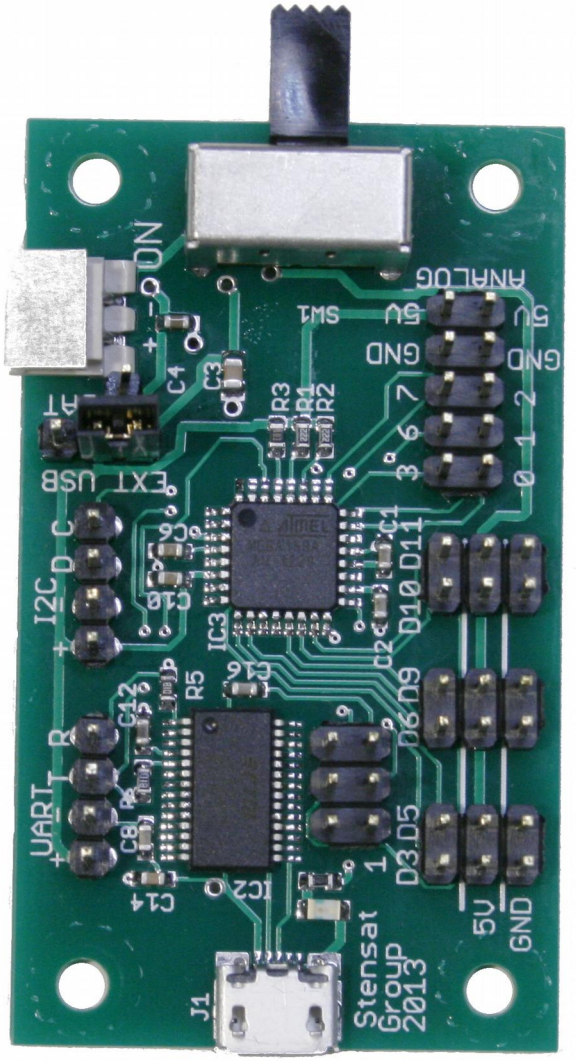
Processor Board
and
Arduino Software

Overview

- In this section, you will be introduced to the processor board electronics and the arduino software.
- At the end of this section, you will be able to write software, control things and sense the environment.

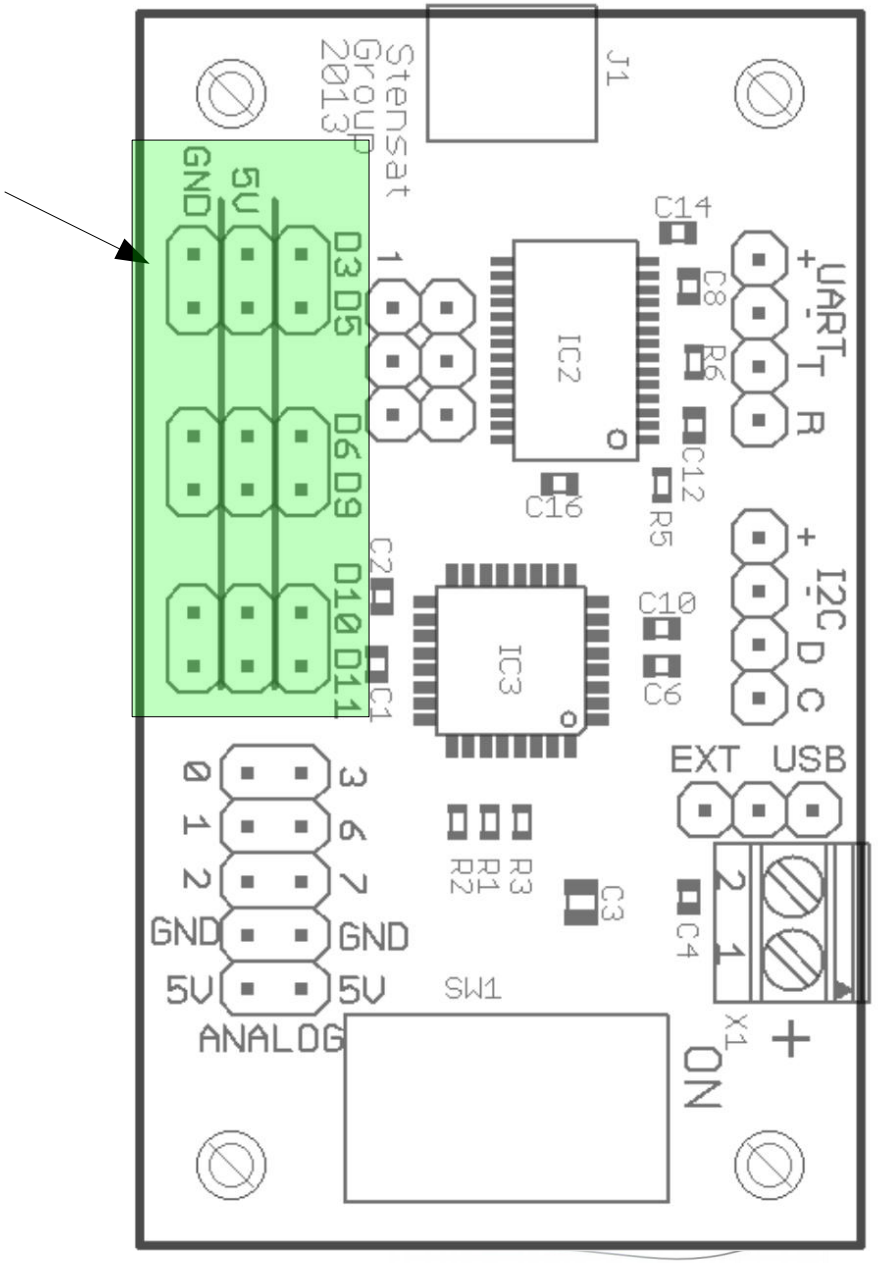
Processor Specifications

- The processor is shown to the right. It is called an embedded computer because it is to be integrated or embedded in something, this case a robot.
- The processor board has connections that allow devices to be interfaced such as lights, motors and sensors.
- There are two types of interfaces that will be used for the robot, digital interfaces and analog.
- The digital interfaces can be configured as an input or output.
  - As an input, the digital interface can detect the state of switches or other signals as being on or off.
  - As an output, the digital interface can turn things on and off such as lights.
- The analog interface is an input only can can measure voltages.



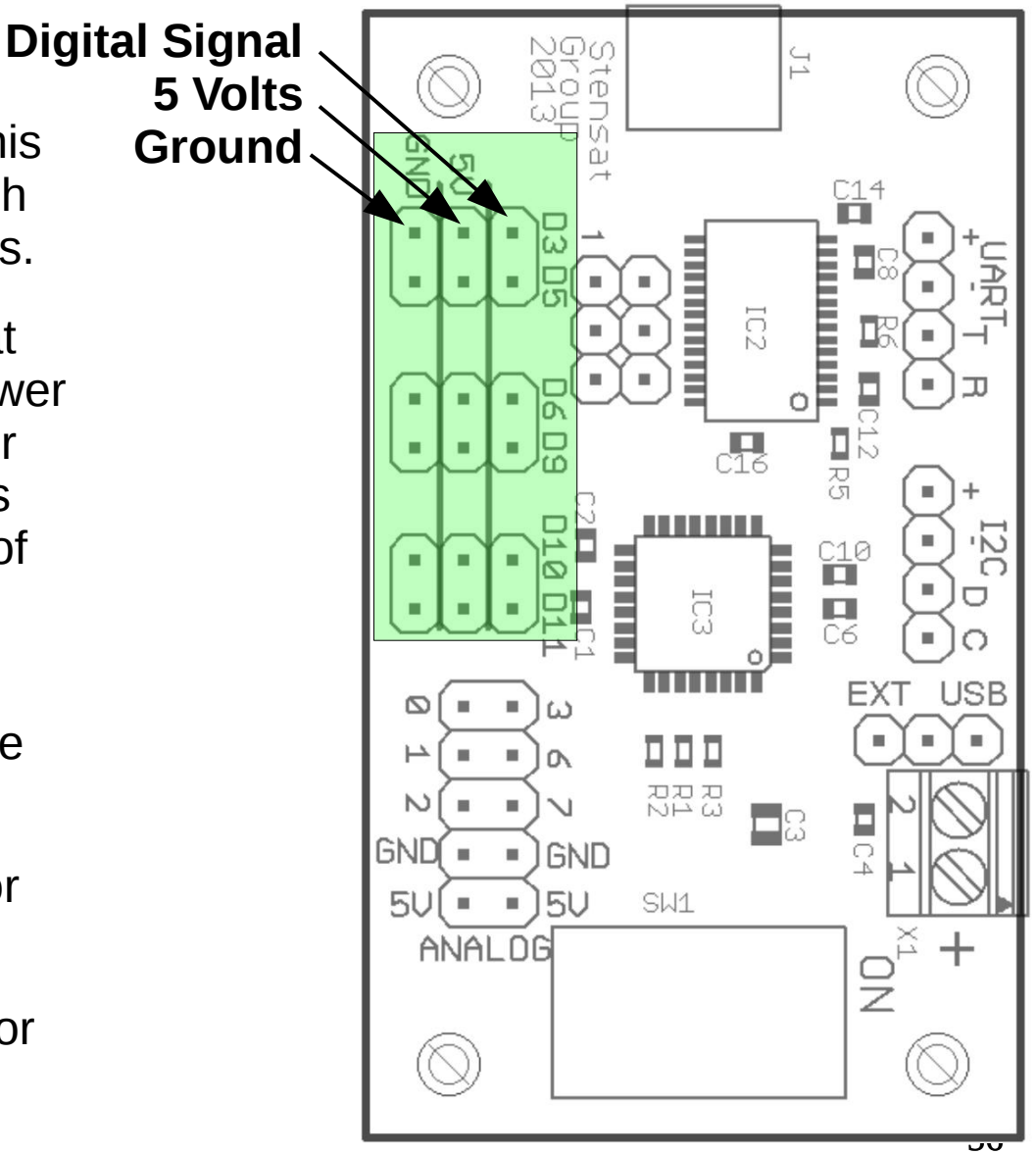
Digital Pins

- The digital pins are highlighted in green.
- There are six digital ports identified as
  - D3, D5, D6, D9, D10, D11
- The digital pins can be used to control things and detect things.
- When the digital pin is set low, the voltage is set to zero volts.
- When the digital pin is set high, the voltage is set to 5 volts.



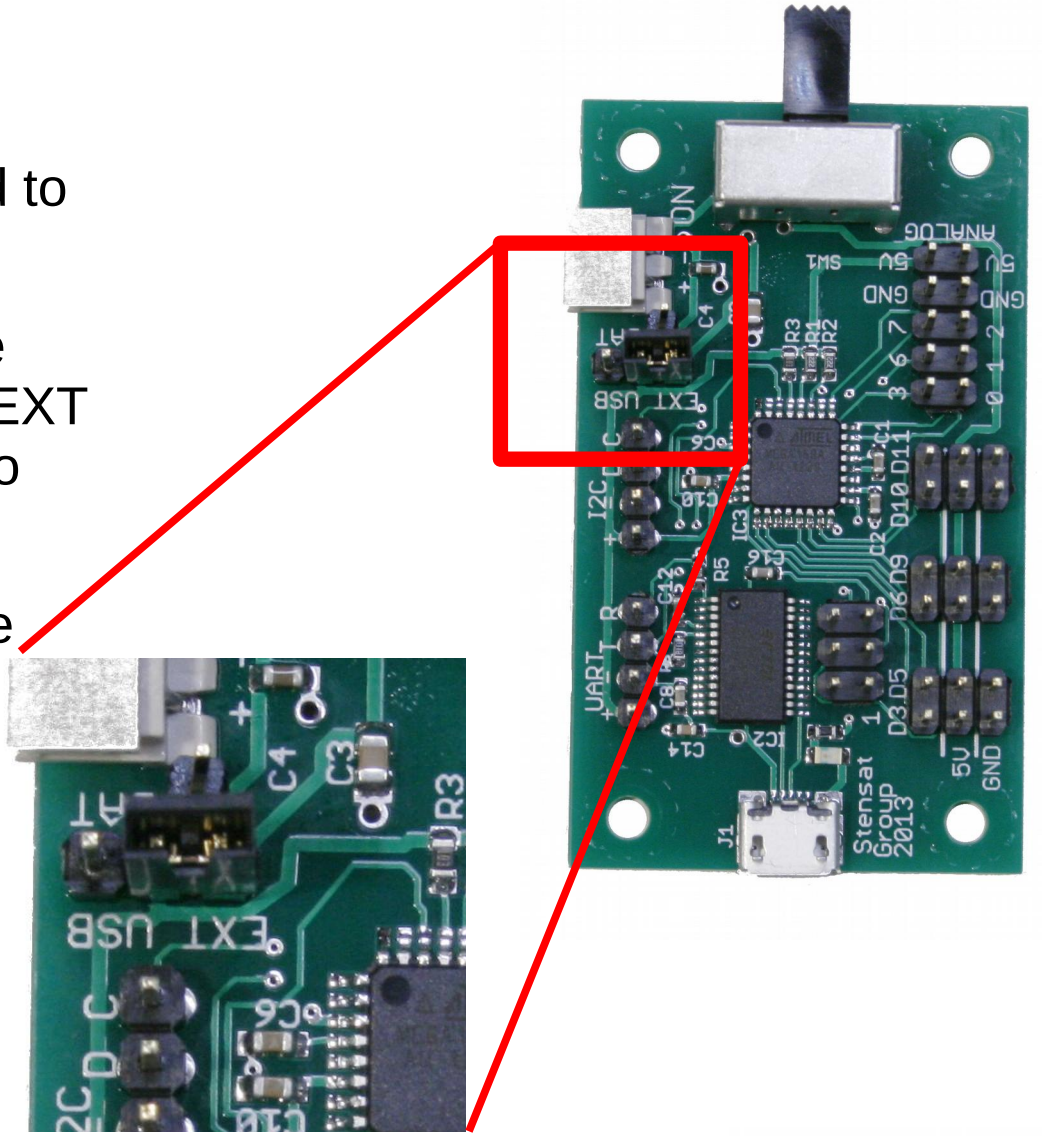
Processor Board Pinout

- The digital pin configuration is shown to the right.
- The digital pin is the most inward pin. This is the digital signal that can be set to high or low, logic level 1 or 0, 5 volts or 0 volts.
- The next pin is 5 volts. This is power that can be used. It is available when the power switch is set to on and an external power source is connected. The USB port does not power this 5 volts for the protection of the host computer.
- The last pin closest to the edge of the board is the ground. This is the reference voltage of zero volts.
- The pins are positioned so a servo motor can be directly plugged in.
- Each pin can be configured as an input or output.



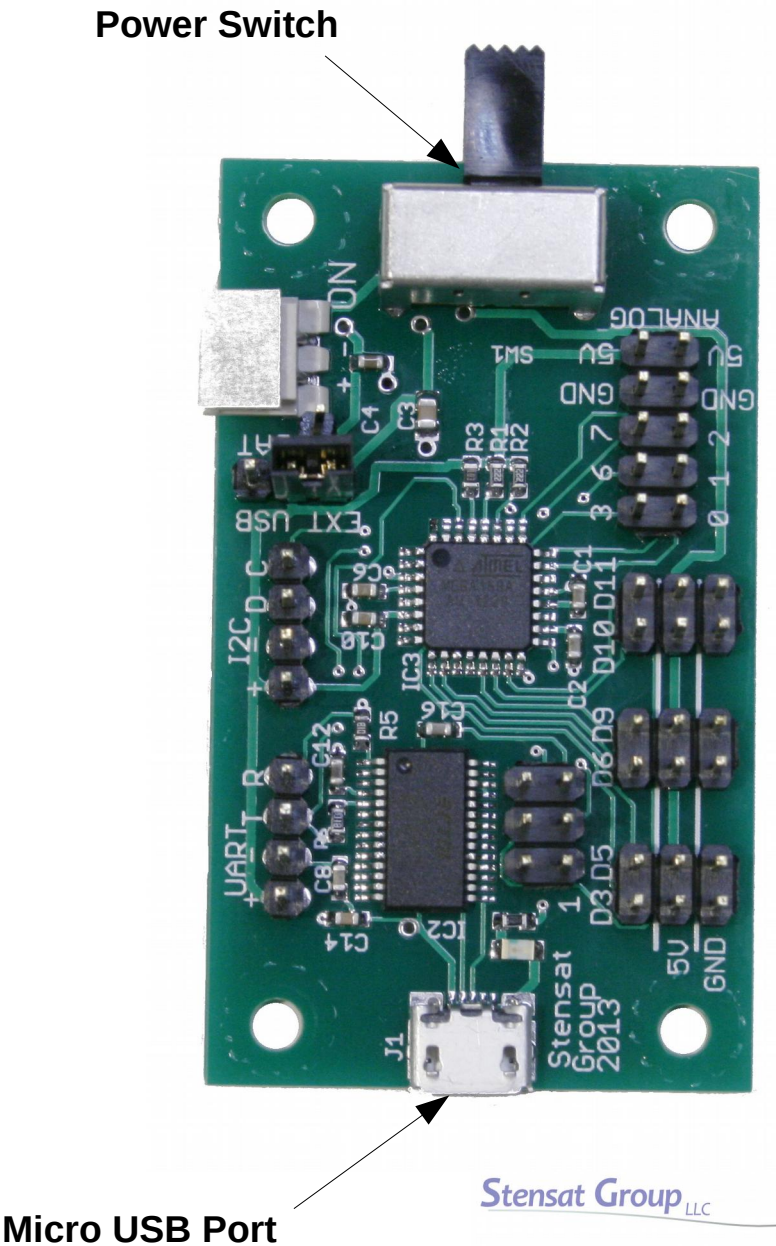
Power Selection

- There is a three pin jumper that lets you select how the processor board is powered.
- A shorting jumper is required to make the selection.
- The picture below shows the shorting jumper connecting EXT which connects the battery to power.
- With the shorting block in the USB position, the USB port would power the processor eliminating the need for batteries.



Other Features

- The power switch allows the processor and anything connected to be turned on and off when the battery is disconnected.
  - The power selection shorting jumper also has to be set to EXT.
- The micro USB port is used for uploading programs and also interacting with the processor.



Software

- Now that the processor features have been covered, it is time to learn about programming it.
- The processor uses the arduino software. This software allows you to write programs, compile them and upload them to the processor. It also allows you to interact with the software running on the processor.
- Only one program can be installed and run at a time. The processor is small and does not have an operating system.
- Embedded computers are designed to perform a specific task and not operate like a desktop computer or laptop.
- More information about the arduino software can be found at
  - www.arduino.cc

Loading and Configuring Arduino Software

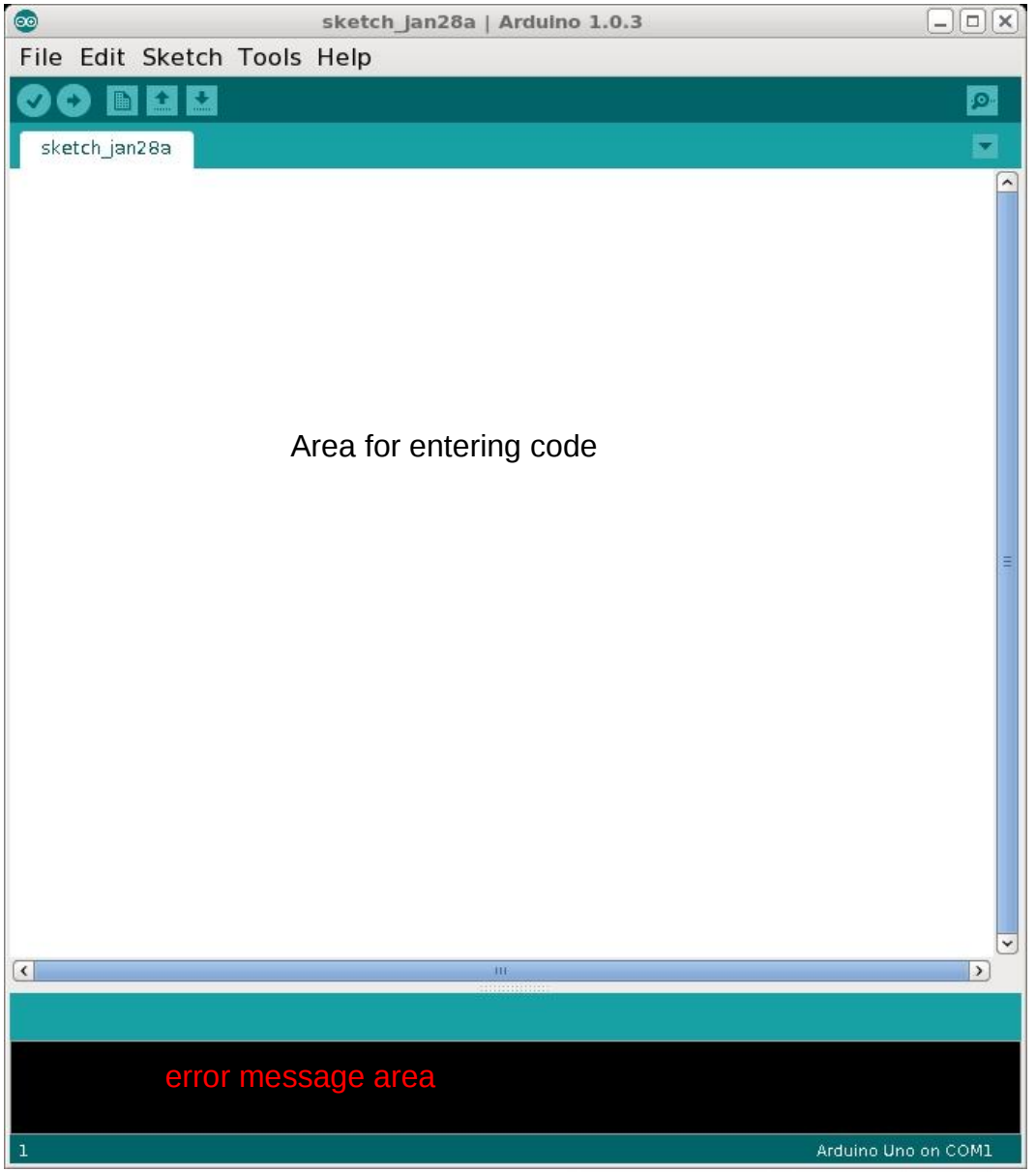
- Copy the arduino-1.6.5 folder from the provided CD to the computer.
- The software can also be downloaded from www.stensat.org/products
  - Just find the robot product and the software will be there.
  - It can be installed anywhere on the computer.
  - Open the folder and double click Arduino.
- The first step is to select the correct processor. Arduino software supports many different variations.
  - In the arduino program select menu “Tools”
  - Select “Board”
  - Select “Arduino Pro or Pro Mini” at or near the top of the menu.
  - Go back and select “Processor” under the “Tools” menu.
  - Select “Atmega328 (5V, 16MHz)”

Configuring Arduino Software

- Plug the processor board into the computer USB port
  - Let the operating system find the drivers. (network connection required)
    - The driver is also included with arduino software
  - In the arduino program select menu “Tools”
  - Select “serial Port”
  - Select the appropriate COM port.
    - If you have a modem built in or existing COM ports, the COM number for the processor will usually be the highest number.

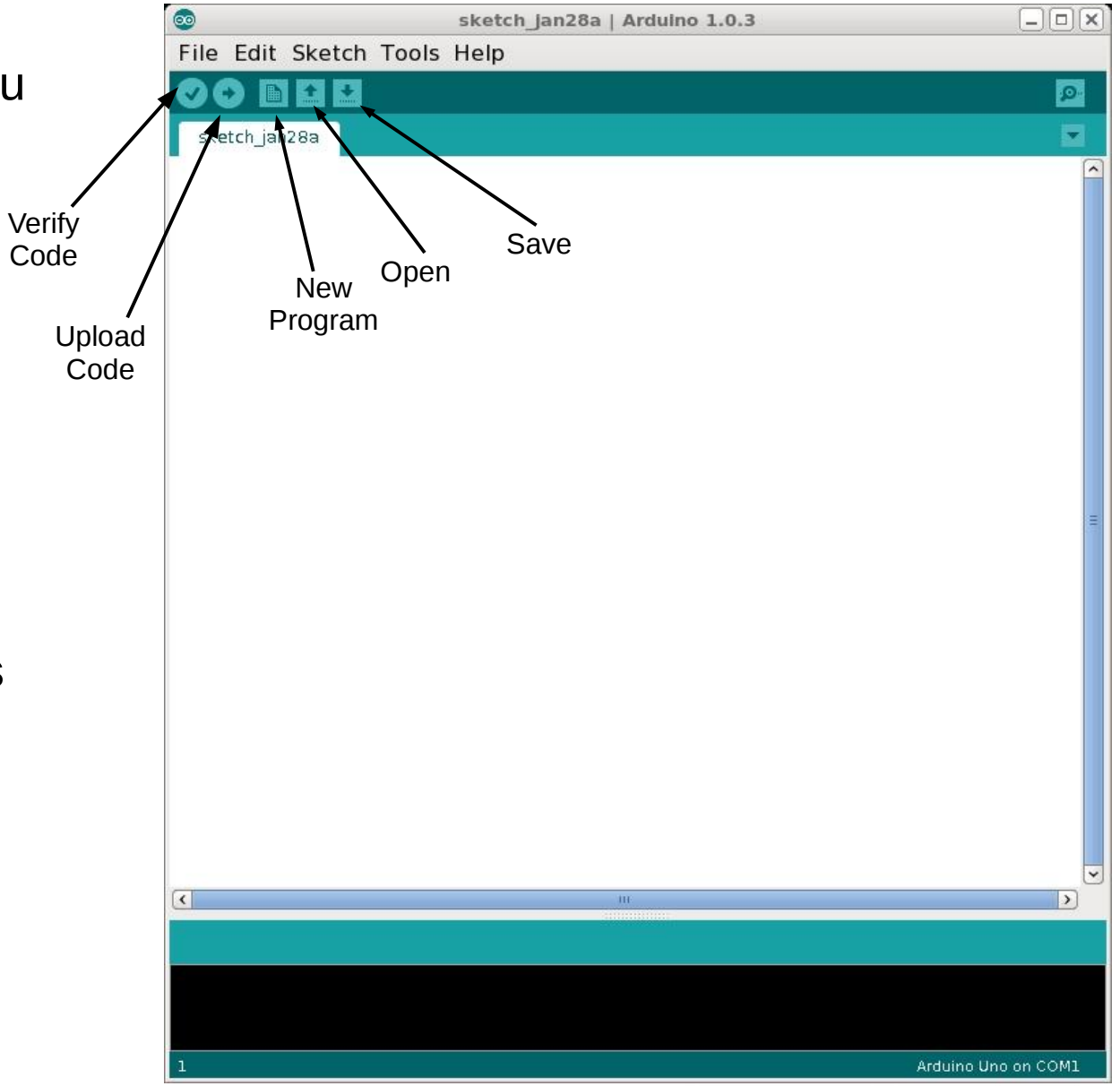
Using Arduino

- This is the arduino software.
- The software will let you enter programs and upload the code to the processor board.
- The large white area is where the code is entered.
- The black area below is where error messages will be displayed such as when there is an error in the code or the software cannot upload code for some reason.



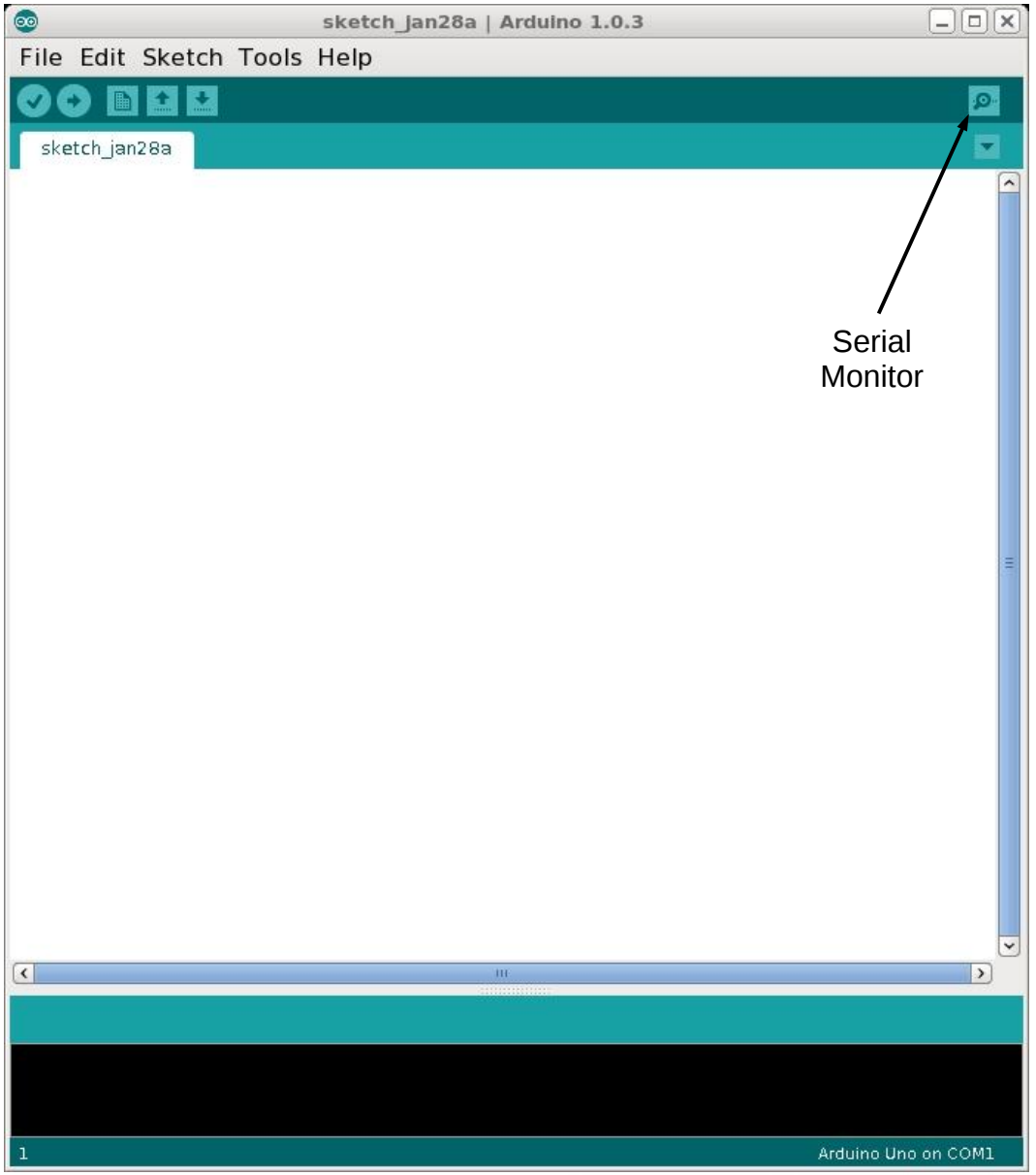
Using Arduino

- The buttons below the menu have different functions.
- The first called Verify Code will compile the code and check for errors but not upload the code.
- The next button will do the same as the first but also upload the code.
- New Program button opens a new copy of the program allowing you to start writing another program.
- Open and Save are for opening and saving the code you have written.



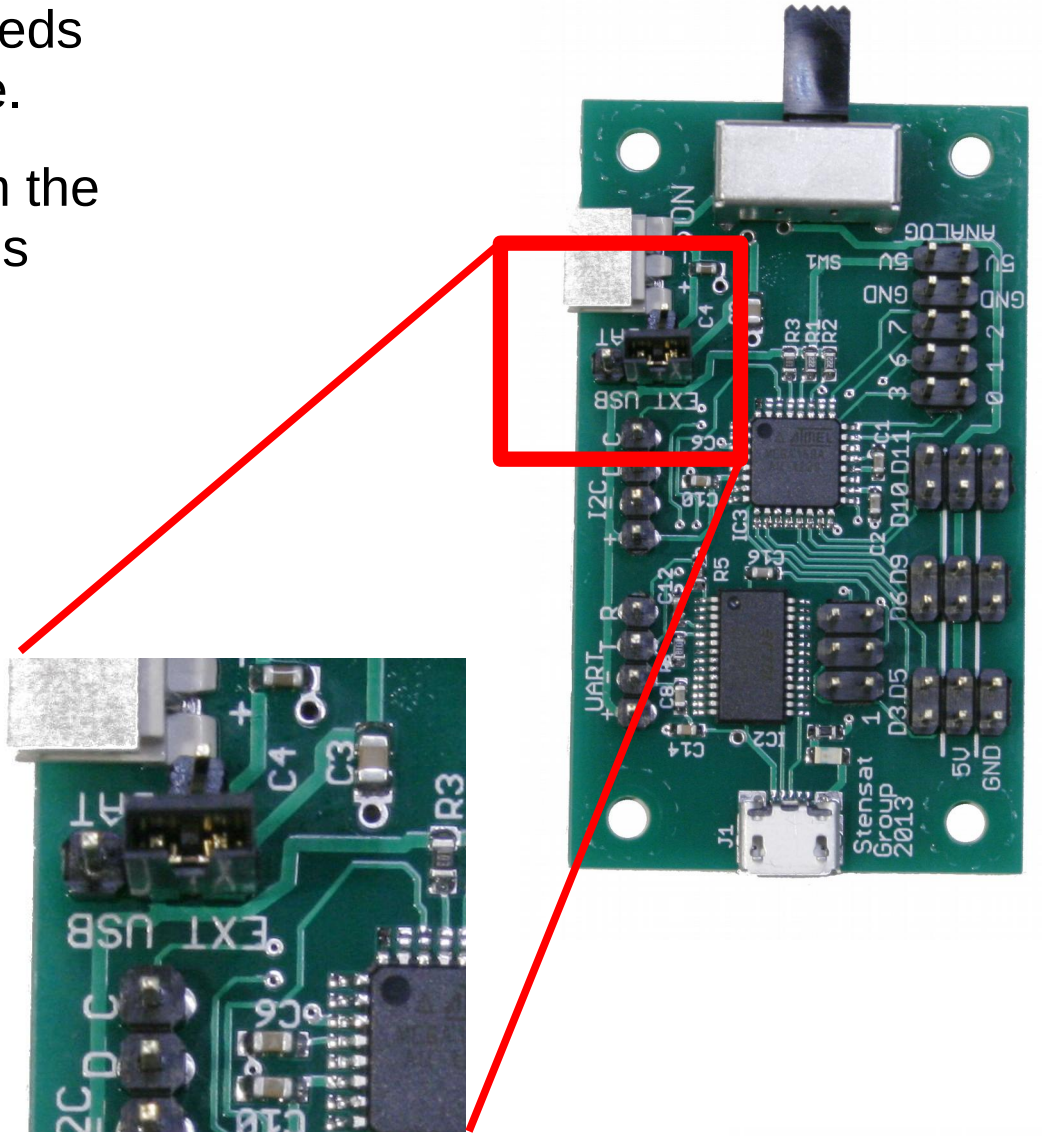
Using Arduino

- Serial Monitor button opens a new window allowing you to interact with the processor.
- The Serial Monitor window allows the processor to display information and you to send information.
- This will be used quite a bit in this section.



Power Selection for Programming

- Before continuing, the power selection shorting jumper needs to be moved to the USB side.
- Pull the shorting jumper from the two pins and insert where it is marked USB.



First Program to Test

- Enter the program in the editor on the right. **Do not copy and paste from the pdf file**. It doesn't work. The compiler is case sensitive so pay attention to capitalized letters.
- Plug the processor board into the USB port.
- Click on the upload Code button to compile and upload the program.
- When the status message at the bottom of the window says done uploading, click on the serial monitor button.
- The Serial Monitor window pops up with the message being displayed.
- Experiment by changing the message.
- Save your program. Pick a file name.

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
}
```

Serial Monitor Window

What are Functions

- A function is basically a set of instructions grouped together. A function is created to perform a specific task.
- The set of instructions for a function are bounded by the curly brackets as seen to the right.
- The **`setup()`** function is used to initialize the processor board, variables, and devices.
- Inside functions, you can call other functions. **`Serial.begin()`** is a function. It is located somewhere else in the arduino software.

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print("Hello World");
}
```

Other Syntax Requirements

- You will notice that some lines end with a semi-colon. This is used to identify the end of an instruction. An instruction can be an equation or function call.
- When you create a function such as **`setup()`**, you do not need a semi-colon.

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.print(“Hello World”);
}
```

Arduino Programming Basics

- The program is made up of two functions.
- **`setup()`** function is run at reset, power up or after code upload only once.
  - It is used to initialize all the needed interfaces and any parameters.
- **`loop()`** function is run after the **`setup()`** function and is repeatedly run hence the name loop.
- This program configures the serial interface to send messages at 9600 bits per second.
- The message is “Hello World” and is repeatedly displayed.

```
void setup()
{
    Serial.begin(9600);
}
```

```
void loop()
{
    Serial.print(“Hello World”);
    delay(500);
}
```

•**`Serial.begin()`** is a function that intializes the serial interface and sets the bit rate.
•**`Serial.println()`** sends the specified message over the serial interface and move the cursor to down one line.
•**`delay(500)`** is a command to stop the program for 500 milliseconds.

What is in the Software

- In the **`setup()`** function, it executes the function **`Serial.begin(9600);`**
  - This function initializes the UART which is connected to the USB port to allow for communications.
- In the **`loop()`** function, it executes the function **`Serial.print("Hello world");`**
  - This function send the text in quotes to the UART. This is displayed in the Serial Monitor window.
- The other function is called **`delay()`**.
  - This function stops the program for a specified period of time. The unit is in milliseconds. The code to the write displays the text every half second.

```
void setup()
{
    Serial.begin(9600);
}
```

```
void loop()
{
    Serial.print("Hello World");
    delay(500);
}
```

What is in the Software

- In the Serial Monitor window, you may have noticed that the text displayed scrolls to the right. That is just how Serial.print() works.
- To have the text displayed on its own line, change the Serial.print() to Serial.println().
- Serial.println() adds a line feed which forces the text in the Serial Monitor to move down one line.
- Make the change, upload the code and open the Serial Monitor window.

```
void setup()
{
    Serial.begin(9600);
}
```

```
void loop()
{
    Serial.println("Hello World");
    delay(500);
}
```

Electronics

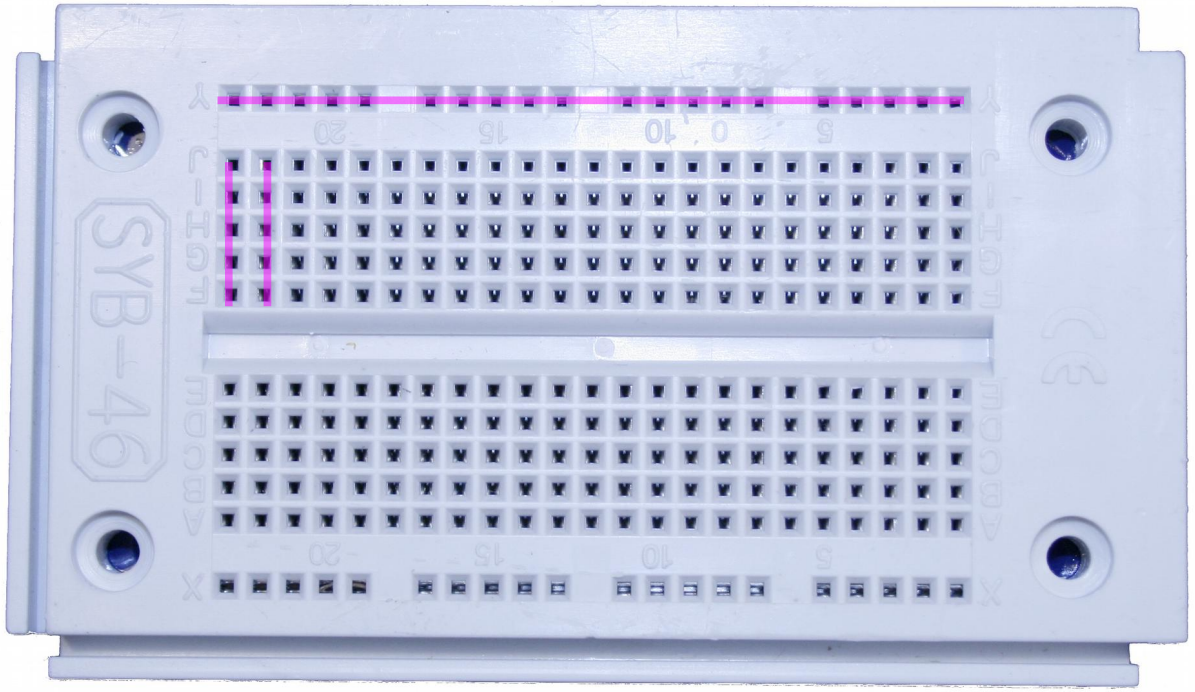
- At this point, you should be able to run the arduino software.
- You should know that the software consists of two functions
  - setup()
  - loop()
- You should know how to initialize the UART and write a program to display text in the Serial Monitor window.
- You should know how to open the Serial Monitor window.
- Next is learning a little about electronics and how to control things.

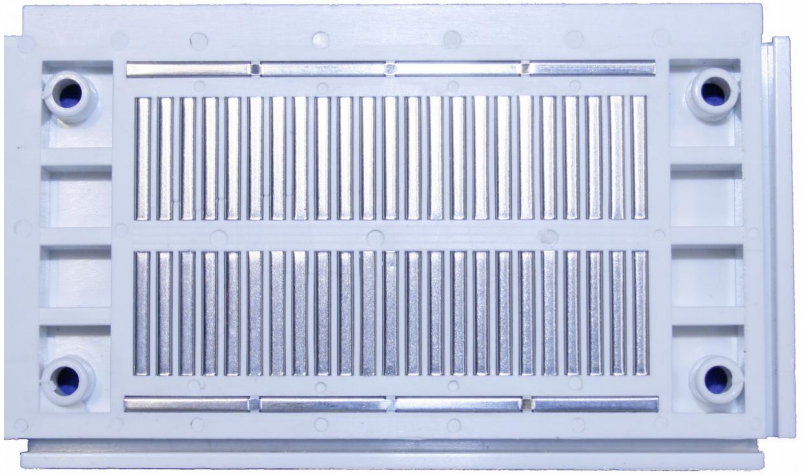
Electrical Circuits

- In this section, you will learn how to connect electrical circuits.
- Electrical circuits is nothing more than connecting wires between devices to allow the flow of electrons. A lamp plugged into an outlet has two wires to make an electrical circuit.

How the Solderless BreadBoard Works

- The solderless bread board allows circuits to be quickly connected.
- Each row of holes that go left to right on the top and bottom are all connected together.
- The columns of 5 holes are all connected together.
- The lines in the picture show the connections.
- Components and wires are inserted in the holes to make connections.

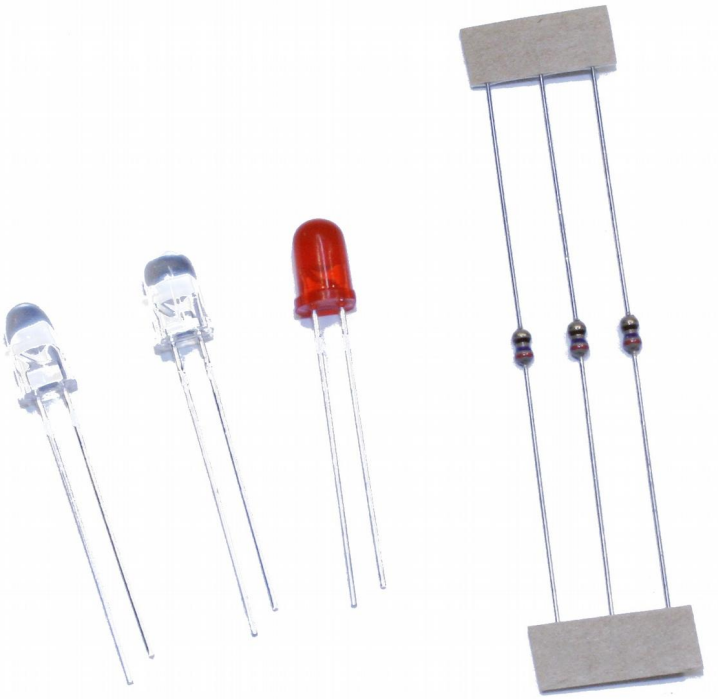




Back side showing how holes are connected

First Circuit

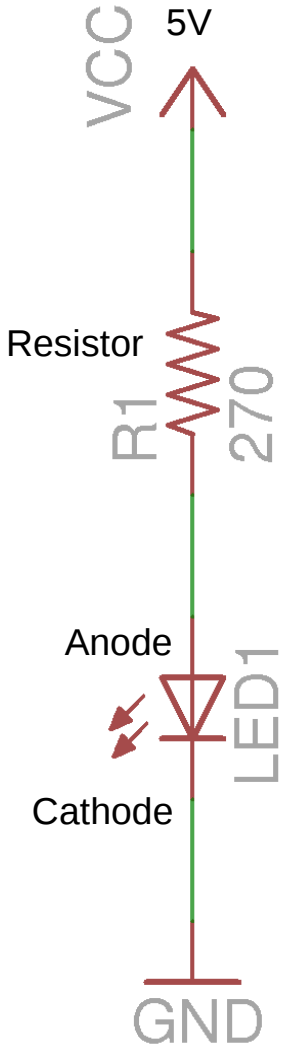
- The first circuit will use a Light Emitting Diode or LED.
- The LED is a polarized device and only works in one direction and gives off light when current flows through it.
- The positive pin on the LED is the longer pin. It is called the anode. The other end is the cathode.
- LEDs need the current to be limited otherwise it will take too much and burn out. A resistor will be used to limit the current flow through the LED.



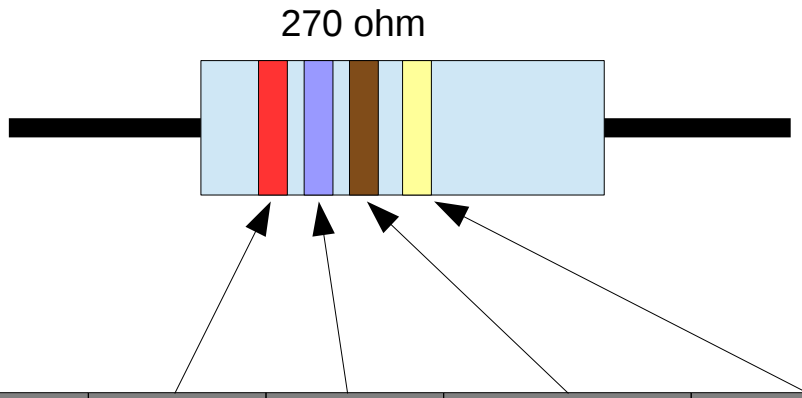
LED Resistors

First Circuit

- The first circuit will connect the LED straight to 5 volts so the LED will always be lit when there is power.
- The schematic for the circuit is shown to the right.
  - The symbol at label R1 is for the resistor.
  - LED1 is next to the symbol for the LED.
- The symbol at the top is the +5V connection. It is called VCC.
- The GND symbol is for ground. This is the zero volt reference.
- The LED has an anode and a cathode. The anode is the long pin.
- When the anode is at a higher voltage than the cathode, the LED will light.



Resistor Code



Color	Name	Digit 1	Digit 2	Multiplier	Tolerance
	Black	0	0	x1	
	Brown	1	1	x10	1%
	Red	2	2	x100	2%
	Orange	3	3	x1,000	3%
	Yellow	4	4	x10,000	4%
	Green	5	5	x100,000	
	Blue	6	6	x1,000,000	
	Violet	7	7		
	Grey	8	8	Gold	5%
	White	9	9	Silver	10%

A resistor is a device that is used to limit the flow of current or reduce the voltage depending on how it is used.

Resistor values are determined by the color bands. The picture to the left shows how to decipher the color bands. The first two bands determine numerical value and the third band is the multiplier. A 270 ohm resistor has a red, violet, and brown band. The last band indicates the how far off the value can be.

The bands start at one end of the resistor.

Wiring Diagram for LED

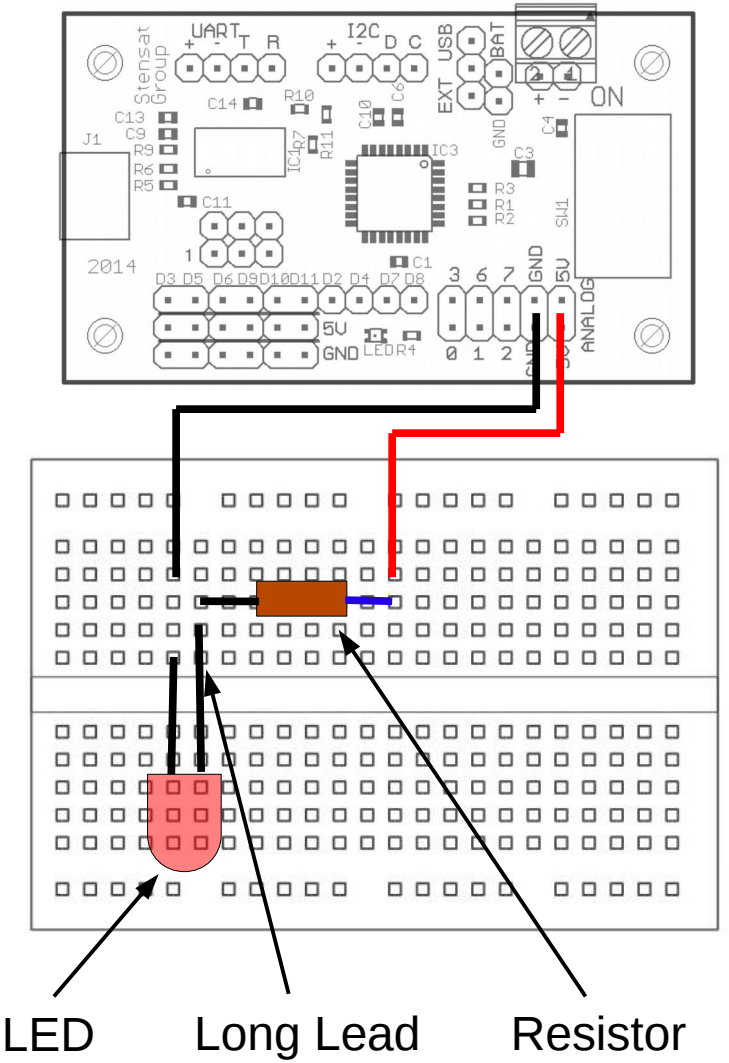
Insert the LED into the bread board as shown. The short lead on the LED should be on the left side.

Insert the 270 ohm resistor as shown. One lead should be installed in the same column as the long lead of the LED. The other resistor lead is inserted in any other column.

Take the black jumper wire and insert the pin into the column of the short LED lead. Plug the other end into the ground pin as shown.

Take a red jumper and connect the pin into the same column as the resistor lead and the other end into 5V pin as shown.

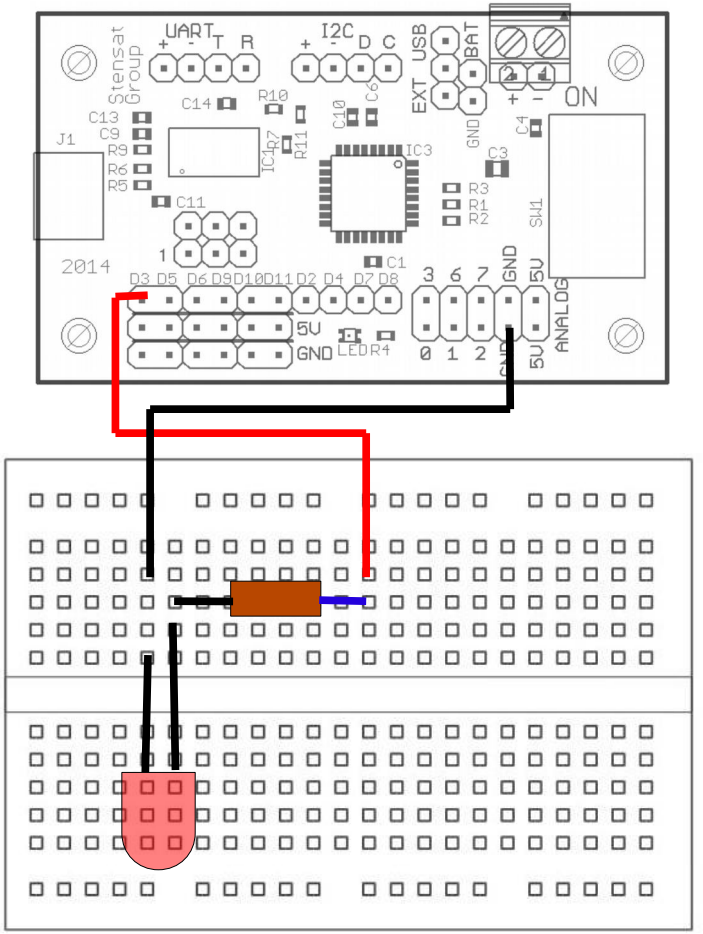
Plug the processor board into the computer USB port. The LED should light up. If not, reinsert the LED in the opposite orientation.



LED Connected to Digital Pin 3

Move the red jumper from the 5V pin on the processor board to the pin marked D3.

Leave everything else as is.



Connecting the LED to a Digital Pin

- The LED is not lit at this time because the digital pin 3 needs to be programmed to generate a voltage.
- The program to the right will cause the LED to blink.
- Create a new program and enter the code.
- Upload the code to the processor board. Make sure the processor boar is plugged into the USB port.
- Save the program and use a new file name such as “blinky”.

```
void setup()
{
   pinMode(3,OUTPUT);
}

void loop()
{
   digitalWrite(3,HIGH);
   delay(500);
   digitalWrite(3,LOW);
   delay(500);
}
```

Connecting the LED to a Digital Pin

- In the **`setup()`** function, digital pin 3 is configured as an output.
  - The function **`pinMode()`** configures digital pin 3 to be an output.
  - **`pinMode()`** takes two arguments separated by a coma.
    - The first argument selects the digital pin.
    - The second argument configures the digital pin as an output.
- in the **`loop()`** function, digital pin 3 is set high which causes the pin to generate 5 volts. The LED turns on.
- The **`delay()`** function halts the program for 500 milliseconds.
- The next **`digitalwrite()`** command sets digital pin 3 to 0 volts turning off the LED.
- Save the program and use a new file name such as “blinky”.

```
void setup()
{
   pinMode(3,OUTPUT);
}

void loop()
{
   digitalWrite(3,HIGH);
   delay(500);
   digitalWrite(3,LOW);
   delay(500);
}
```

digitalWrite()

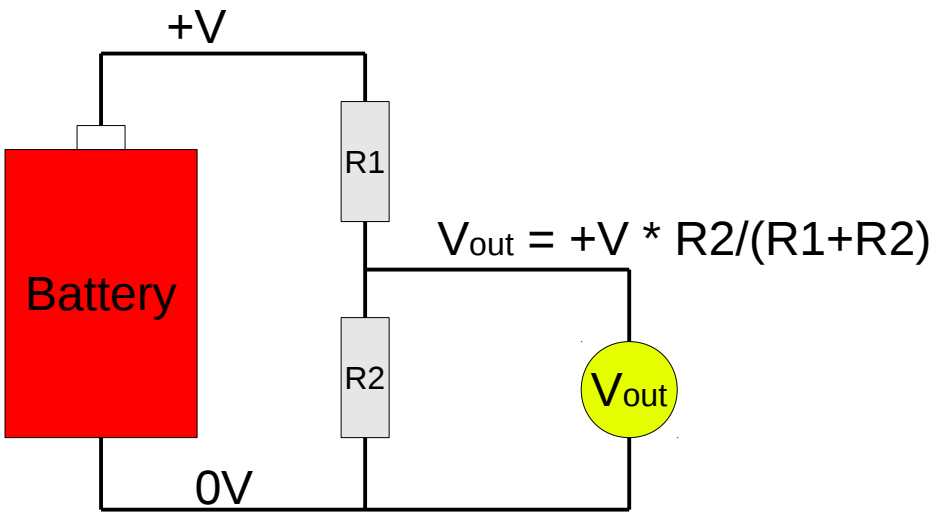
- The digitalWrite() function controls a pin and can set it high or low.
  - The function has two arguments separated by a coma.
    - The first argument selects the digital pin.
    - The second argument sets the digital pin.
      - When set high, the pin is set to 5 volts.
        - In logic terms, a high signal is logic level 1.
      - When set low, the pin is set to 0 volts.
        - In logic terms, a low signal is logic level 0.
  - The function is written as
    - digitalWrite(pin,setting)
      - setting is HIGH or LOW
        - The letters need to be capital.

HIGH = 5 volts = Logic 1
LOW = 0 volts = Logic 0

Next Example

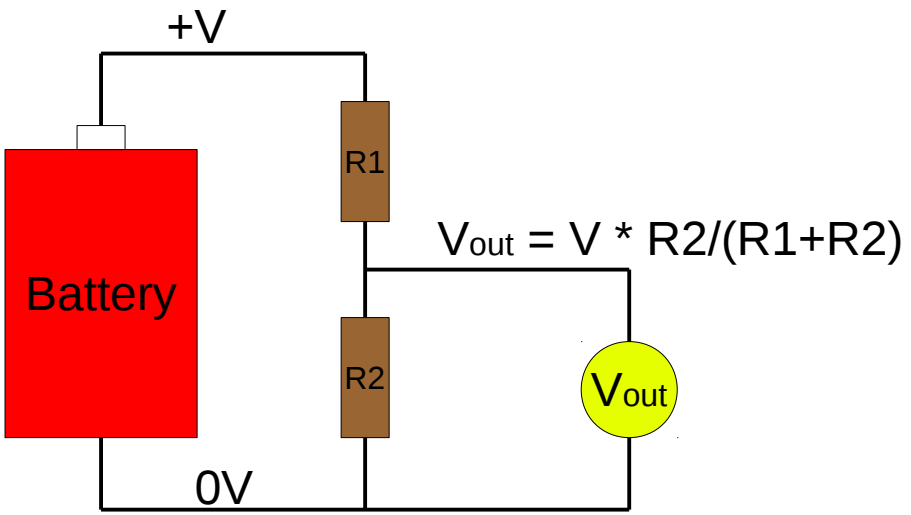
- The next example will be to detect light intensity.
- Remove the LED and switch circuit.

Resistor Voltage Divider



- There is a simple circuit that uses two resistors to divide a voltage from a higher level to a lower level.
- Voltage dividers can be used to reduce a voltage that is too high to a lower voltage that can be handled.
- Another use for this type of circuit is if one of the resistors is a sensor like temperature or light sensor where its resistance changes with what it measures. This circuit makes it simple to connect to an analog-to-digital converter.

Resistor Voltage Divider



- The resistors are connected together in series.
- The voltage source which is the battery in the picture to the left is connected across both resistors.
- The divided voltage is located at the connection between the two resistors.
- The voltage can be calculated by the equation to the left.
  - V is the voltage across both resistors.
  - R1 and R2 are the resistors of some value. They can be any value to get a variety of different voltages.

Analog-to-Digital Converter

- An analog-to-digital converter or ADC is a device that generates a number based on the voltage level it measures.
- The ADC on the processor board can measure a voltage range from zero to 5 volts. If a higher voltage needs to be measured, the voltage divider circuit could be used to reduce the voltage to 5 volts or less.
- The ADC is 10 bits. This gives a numerical range of zero to 1023. It is a linear relationship to the voltage range of 0 to 5 volts.
  - The ADC will generate a value of 0 for 0 volts, 1023 for 5 volts, and 511 for 2.5 volts. The resolution of the ADC is 5/1023 or 0.00489 volts per bit.
  - Voltage the ADC measures can be calculated by the equation

    V = (ADC value/1023) * 5.0

Processor Board Pinout

- The analog ports are highlighted in yellow.
- Six analog inputs are available: 0,1,2,3,6,7.
- The analog ports allow the measurement of voltages from sensors that generate a voltage based on what is being measured.
- Each analog port has 10 bit resolution and an input range of 0 to 5 volts. Signals beyond this range need to be converted to operate within the 0 to 5 volts or damage may occur.
- The processor has an analog to digital converter or ADC to convert the analog voltage to a digital value. The ADC has a 10 bit resolution which converts 0 to 5 volts to a number with a range of 0 to 1023. It is a linear relationship.
- To calculate the voltage
- voltage = ADC/1023.0 * 5.0

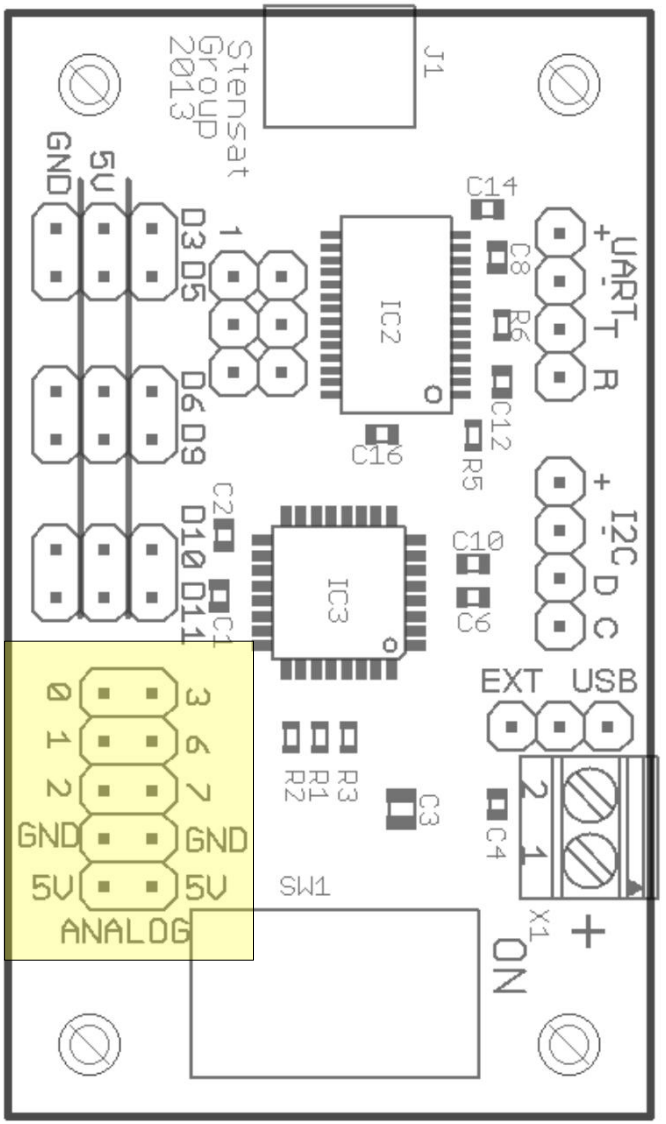
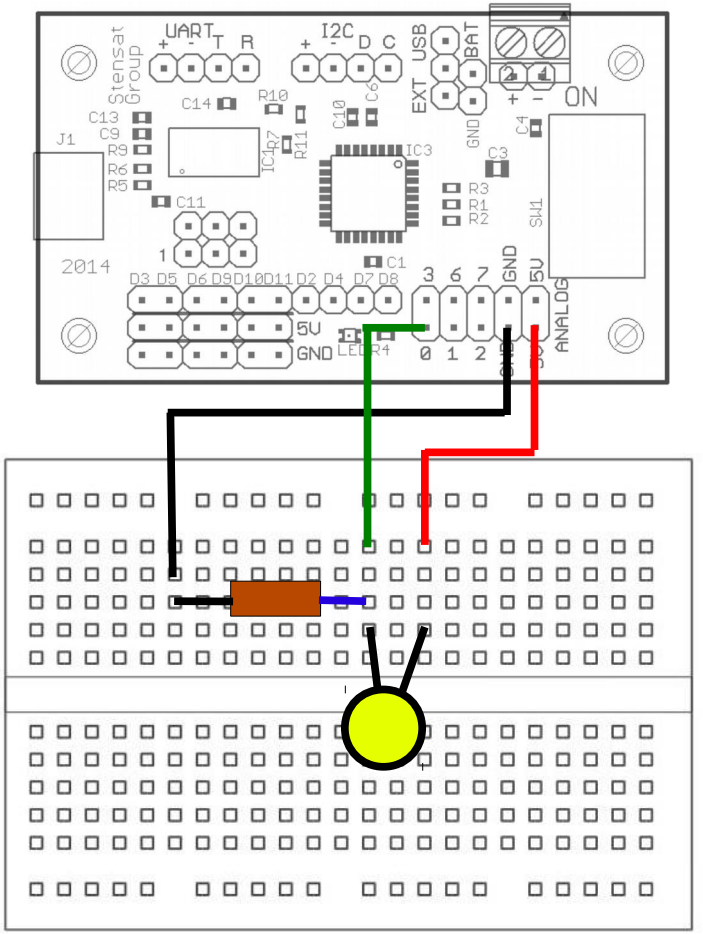


Photo Cell

- The photo cell is a light sensitive device that changes its resistance based on light intensity.
- The photocell can be used in a simple voltage divider circuit with a 4.7Kohm resistor. The color code is yellow, violet and red.
- The photo resistor will have a resistance ranging from 20 Mohm in darkness to 5K ohms in bright light.
- Install the photo cell and 4.7 K resistor on the solderless bread board as shown to the right.
- Connect the free end of the resistor to GND at the analog connector.
- Connect the free end of the photo cell to 5 volts.
- Connect the resistor and photo cell connection to pin 0 of the analog connector.
- To the right is the schematic.



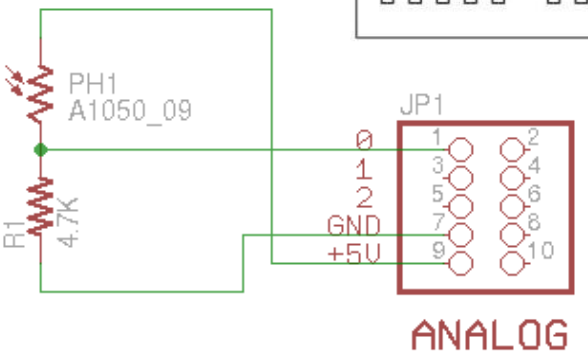


Photo Cell Software

- The program to the right will get an ADC value from analog port 0.
- Create a new program and enter the program to the right.
- To measure the voltage, the function analogRead(port) is used.
- Six ports are available on the processor board.
  - 0,1,2,3,6,7
  - Refer to page 5 for the location.
- Once the ADC value is read, it can be converted to a voltage value. The code to the right shows the equation which can be used for all the analog ports.
- The Serial.println function that displays the volts, includes a numeric argument which specifies the number of decimal places.
- Save the program. Use a new name like photocell.

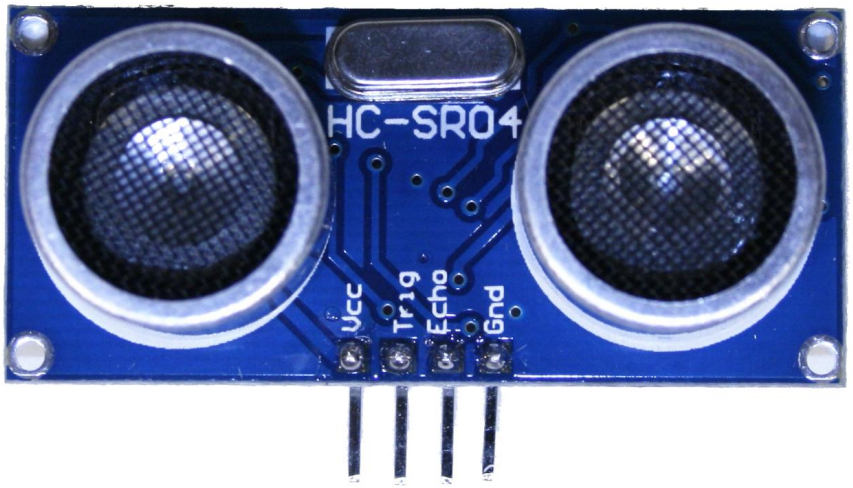
```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0 * 5.0;
  Serial.println(volts,2);
  delay(200);
}
```

Ultrasonic Range Sensor Exercise

- The next exercise will be to learn how to use the ultrasonic range sensor.
- The sensor uses bursts of sound and listens for an echo similar to a bat.
- Remove the photocell circuit.

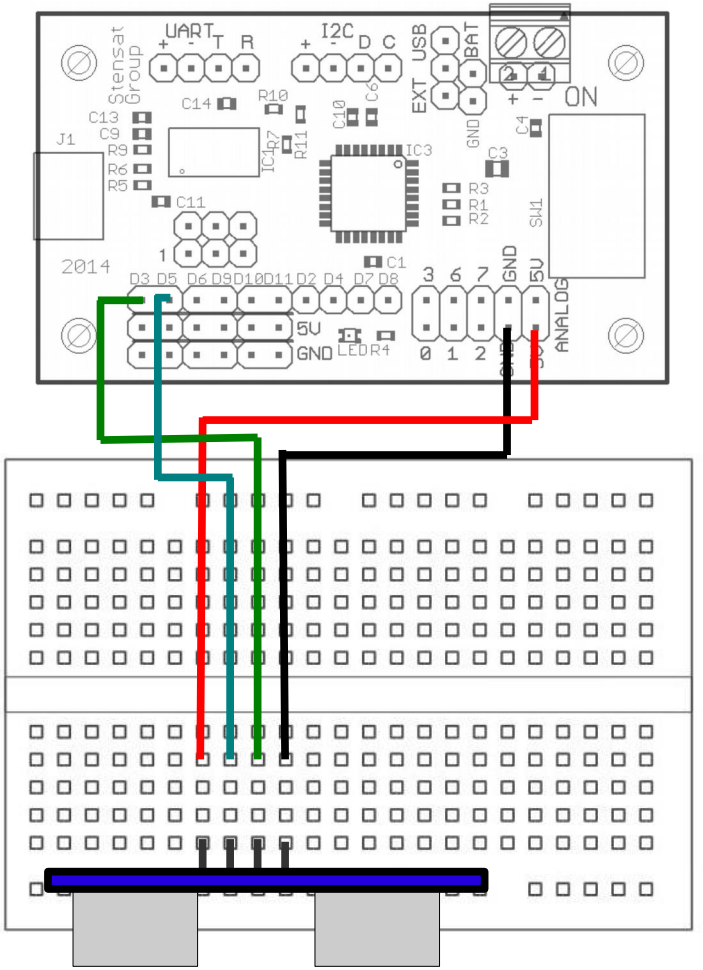
Sensing the Environment

- To detect things in the environment for purpose of collision avoidance, an ultrasonic range sensor will be added to the robot.
- This sensor sends out a burst of audio signal at 40 Khz and detects the echo.
- The processor needs to measure the time it takes for the echo to return.
- This sensor has four pins
  - Ground
  - 5 Volt power input
  - Trigger
  - Echo



Ultrasonic Range Sensor

- Insert the ultrasonic ranger as shown. It should be mounted close to the center of the robot. The pins are inserted at the end of the rows.
- Connect jumpers from the sensor to the processor
  - GND to Analog GND
  - ECHO to pin D3
  - TRIG to pin D5
  - VCC to Analog 5V
- Look on the processor board for the word ANALOG. The power connections are done there to isolate the sensor from the motor power to reduce electrical noise.



Ultrasonic Sensor

- The ultrasonic sensor has two signals, trigger and echo.
- A pulse is sent to the trigger and then the processor is to time when the echo returns.
- This requires two digital pins, one configured as an output and the other as an input. A new command that will be used is called **`pulseIn()`**. This measures the time it takes a pulse to occur in microseconds. Try the program to the right.
- The results are in centimeters.
- Create a new program and enter the code to the right. Save the program and upload it.

```
void setup()
{
    Serial.begin(9600);
    pinMode(3,INPUT);
    pinMode(5,OUTPUT);
}

void loop()
{
    digitalWrite(5,LOW);
    delayMicroseconds(2);
    digitalWrite(5,HIGH);
    delayMicroseconds(10);
    digitalWrite(5,LOW);
    long distance = pulseIn(3,HIGH);
    distance = distance/58;
    Serial.println(distance);
    delay(500);
}
```

Making a Function

- To make this useful for other programs, this program needs to be turned into a function.
- A function is a subroutine or chunk of code that can be called by a name instead of the code being inserted where ever it is needed. This function will return a result.
- The return command specifies which variable is sent back to the calling code.

```
long ultrasonic()
{
    digitalWrite(5,LOW);
    delayMicroseconds(2);
    digitalWrite(5,HIGH);
    delayMicroseconds(10);
    digitalWrite(5,LOW);
    long distance = pulseIn(3,HIGH);
       if(distance == 0) return(1000);
    distance = distance/58;
    return(distance);
}
```

The function **`pulseIn()`** returns the number of microseconds. The result is then divided by 58 to calculate the distance in centimeters.

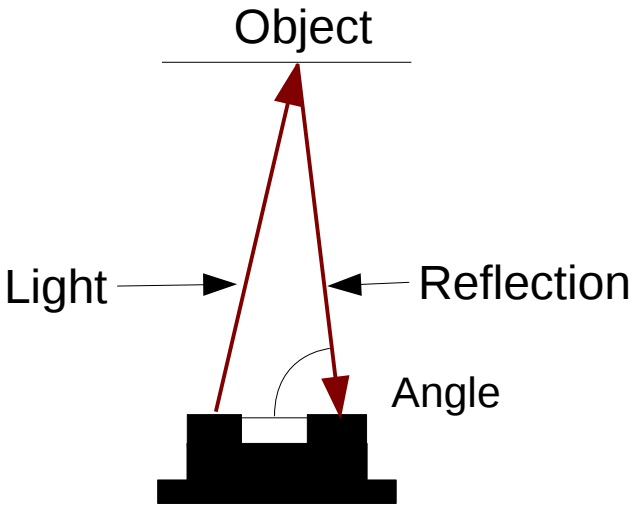
Using the Function

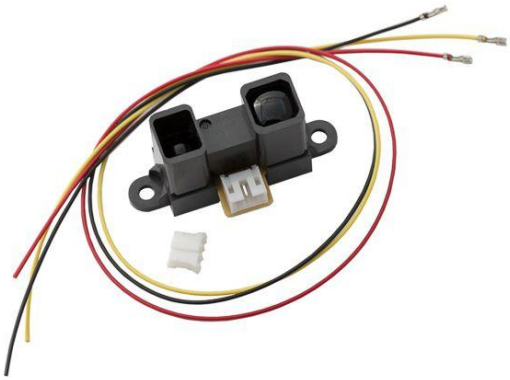
- The program to the right shows how the function is included in the program and where it is located relative to the setup() and loop functions.
- The function has to be located before the code that calls the function.

```
long ultrasonic()
{
    digitalWrite(5,LOW);
    delayMicroseconds(2);
    digitalWrite(5,HIGH);
    delayMicroseconds(10);
    digitalWrite(5,LOW);
    long distance = pulseIn(3,HIGH);
       if(distance == 0) return(1000);
    distance = distance/58;
    return(distance);
}

void set()
{
  Serial.begin(9600);
  pinMode(3,INPUT);
  pinMode(5,OUTPUT);
}

void loop()
{
  long a = ultrasonic();
  Serial.println(a);
}
```

Infrared Proximity Sensor

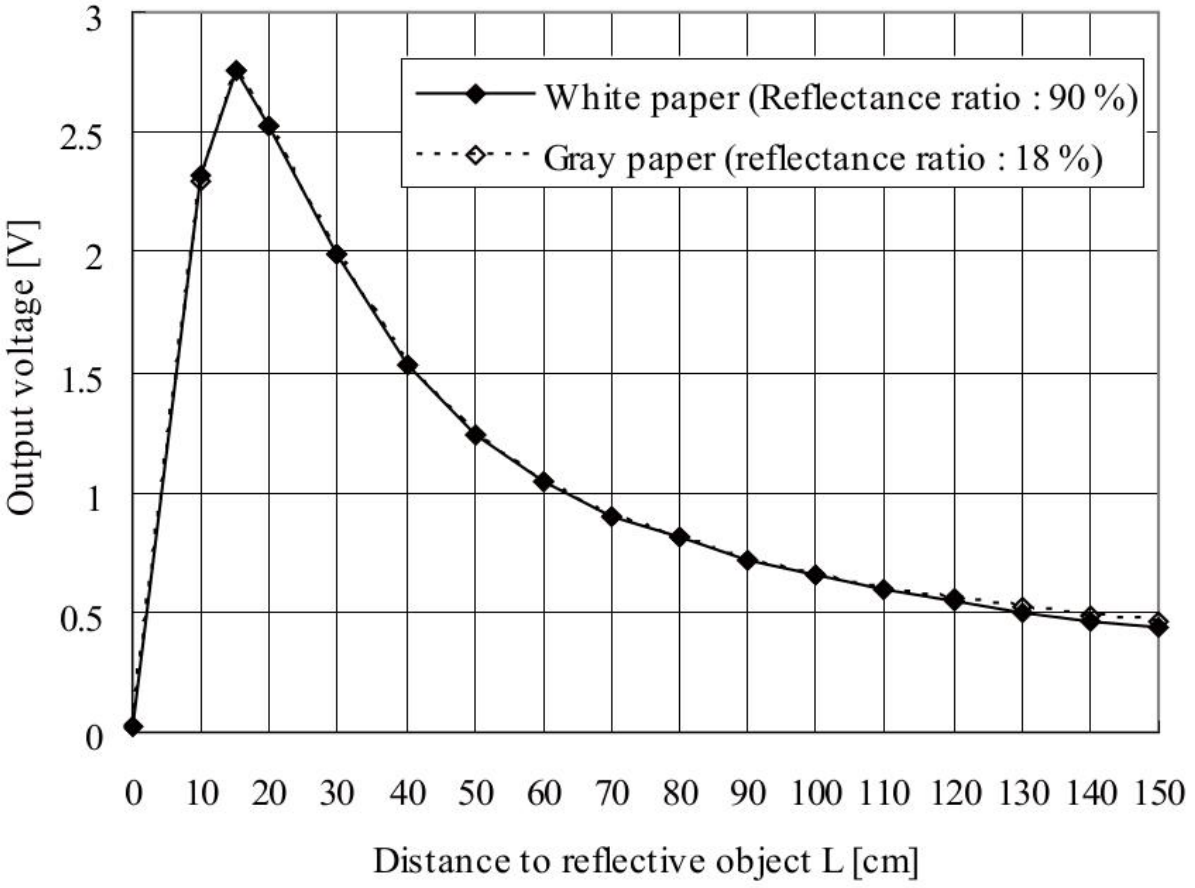
- The Infrared proximity sensor is used to measure the distance of an object from the sensor. The sensor sends out a pulse of light and measures the light beam angle. The wider the angle, the closer the object is. The narrower the angle, the further the object is.





Infrared Proximity Sensor

- The output of the sensor is analog. It generates a voltage based on distance. The plot shows how voltage is correlated with distance. Notice at about 15cm, the voltage starts dropping. This can confuse the rover and is the minimum distance where the sensor is useful. Any objects within 15 cm will not be detected properly.

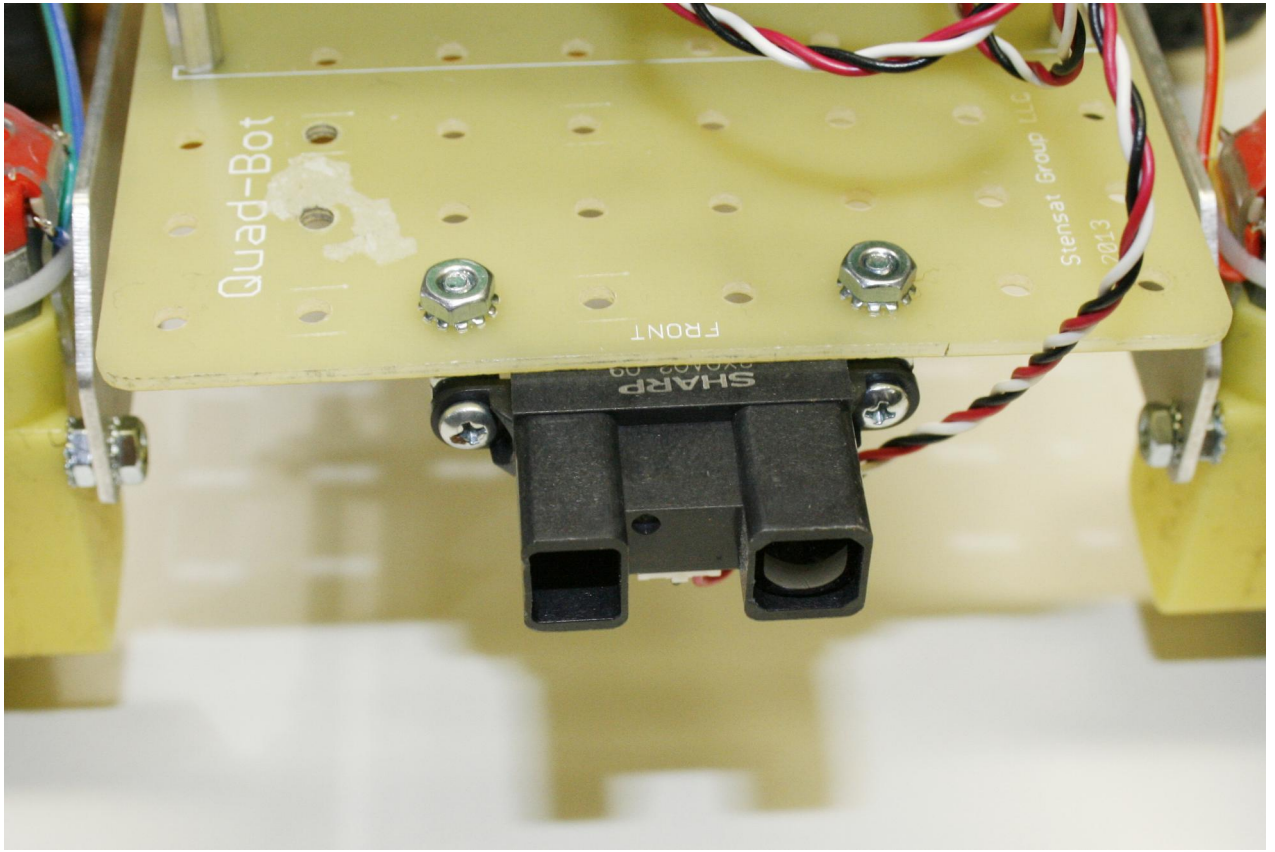


Infrared Proximity Sensor

- The bracket has two holes. One is larger than the other.
- Insert the 3/16 inch long screw into the mounting hole of the sensor and screw it into the small hole of the bracket. The small hole is threaded.
- Align the brackets as shown and tighten the screws until it is snug. Do not overtighten as that may break the plastic.

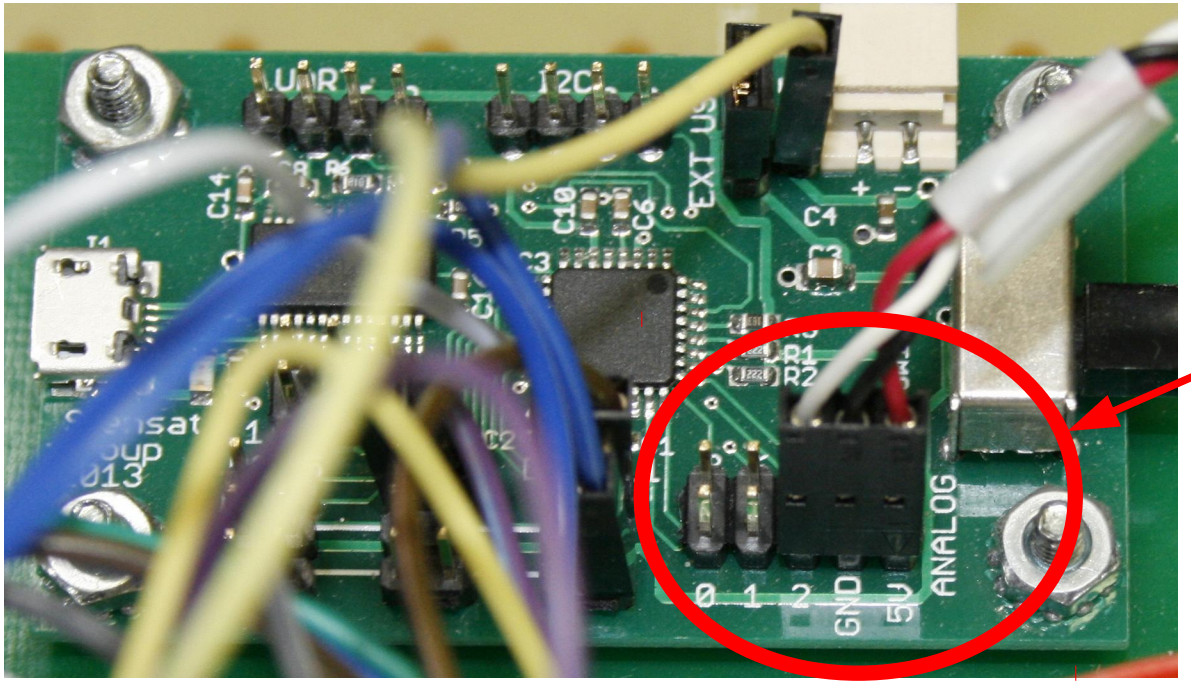
Infrared Proximity Sensor

- Attach the sensor to the rover as shown and use a pair of ¼ inch screws and nuts to secure in place as shown. Insert the screw from below the plate.



Infrared Proximity Sensor

- The sensor comes with a long cable and a connector at the end. The connector has three wires from the cable colored in the order of red, black and white. This allows the sensor to be plugged directly into the analog port connector.
- Orient the connector so that the red wire is connected to the pin marked 5V. The white wire will be aligned to the number 2 which is analog port 2.



IR Proximity Sensor Connector

Infrared Proximity Sensor

- Now it is time to write a program to get a measurement from the proximity sensor.
- Since the measurement is to be viewed, the serial interface is configured.
- Run the program. Use a ruler or yard stick and record the measurement at 3 inch intervals. Start at 3 inches from the sensor and continue for 24 inches. Record the values in the table on the next page.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a = analogRead(2);
  Serial.println(a,DEC);
  delay(200);
}
```

Infrared Proximity Sensor

Distance	ADC Value	Voltage
3		
6		
9		
12		
15		
18		
21		
24		

Infrared Proximity Sensor

- The raw analog converter measurements were recorded.
- Now let's convert the value to a voltage.
- Run the program again and measure the distances again and record the voltage value in the table on the previous page.
- Once completed, compare the values with the plot. You will need to convert inches to centimeters. That is done by multiplying the distance by 2.54.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a = analogRead(2);
  float v = (float)a/1023.0*5.0;
  Serial.println(v,2);
  delay(200);
}
```

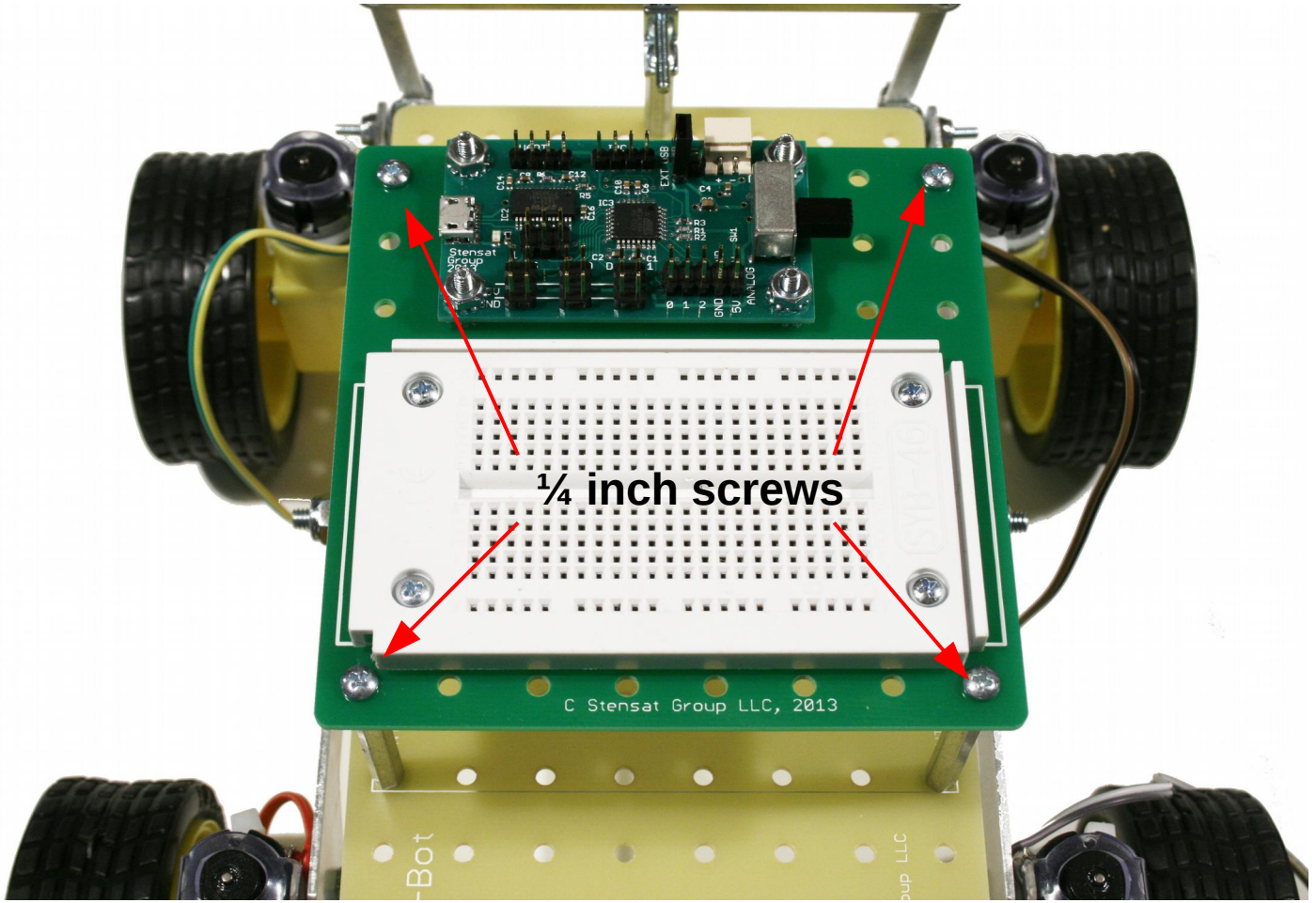
Completing The Robot Motion

Overview

- At this point, you should know how to use the digital pins and analog ports.
- You had an introduction to conditional programming.
- Next is completing the robot and write software to make the robot move.
- First move the power selection shorting jumper to EXT. This will be needed for motor control.
- Remember when uploading code to turn the processor power switch the ON position.

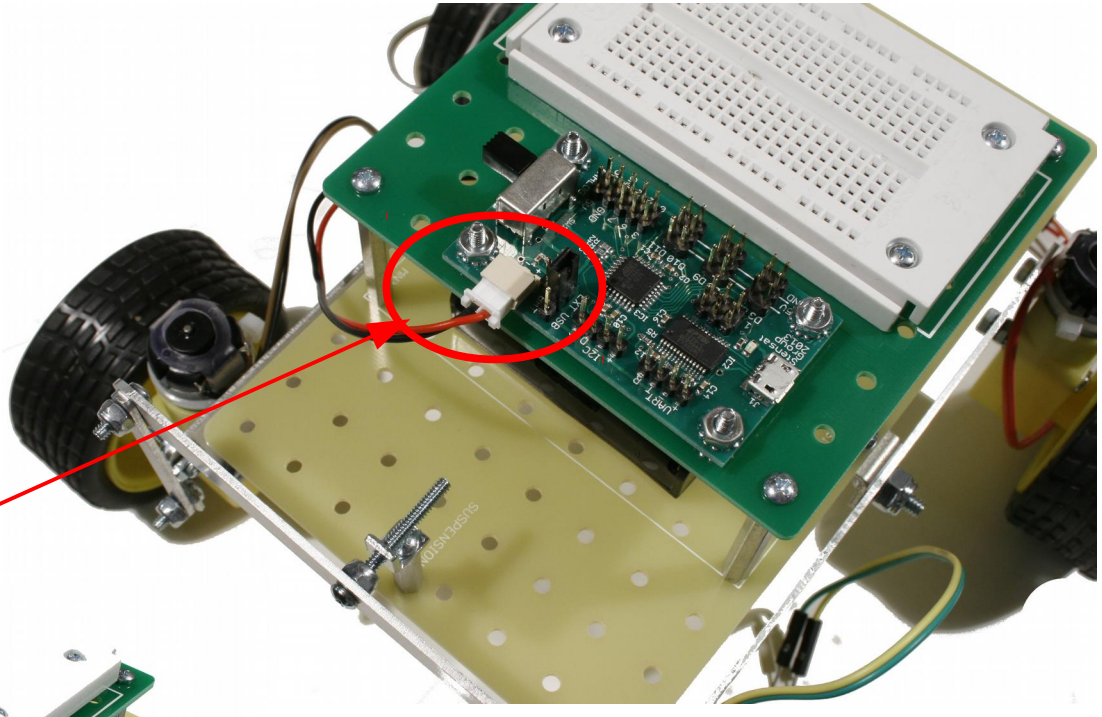
Mounting Electronics Plate

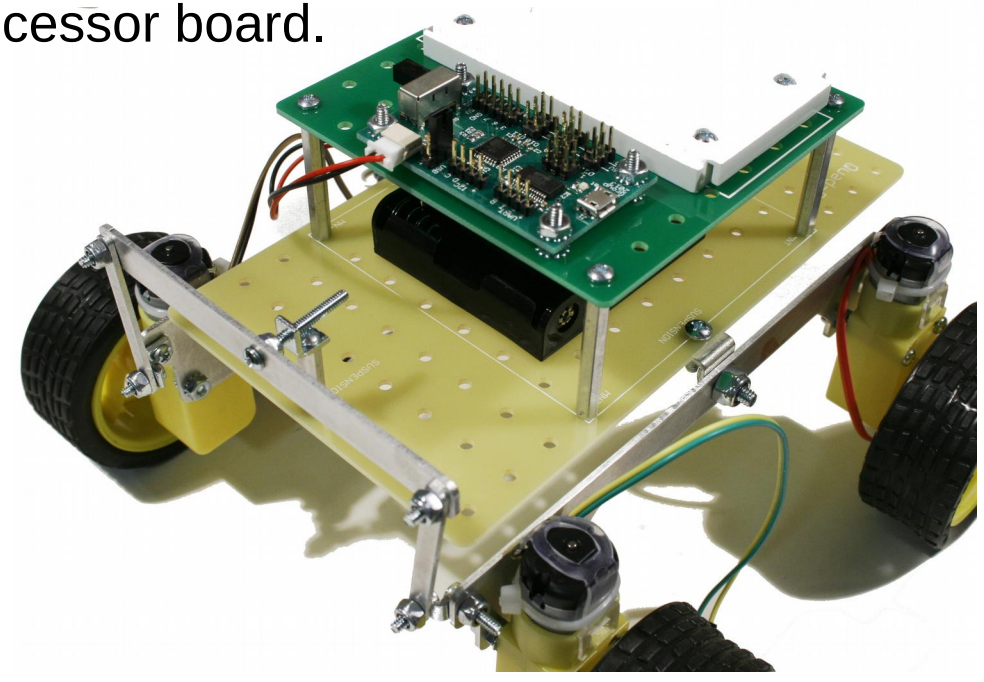
- Place the electronics plate on top of the standoffs. Make sure the solderless breadboard faces the front of the rover.
- Secure the electronics plate with four ¼ inch 4-40 screws at the corners.



Installing the Battery Holder

- To install the battery holder, take a pieces of double sided tape and stick it to the underside of the battery holder.
- Peel the other side and stick it to the base plate under the electronics plate.
- Plug the battery wires into the processor board.



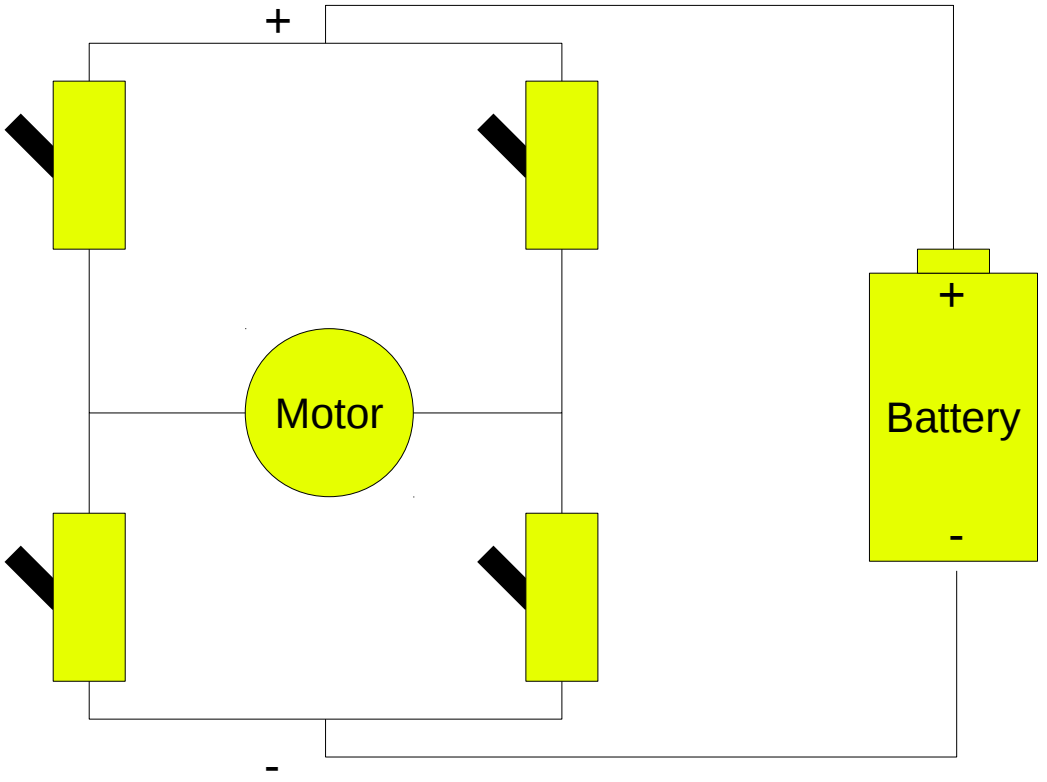


Motor Control

- Controlling the motors is the same as controlling the LED except two signals are needed.
- With two signals, you can control the direction of the motors and turn them on and off.
- The following pages will describe how to hook up the motors.
- A motor driver module is needed. This module allows a computer to control the motors. The motors require more power than the computer signals can provide so the module provides the power.
- The motor driver uses what is called an H-Bridge Driver.

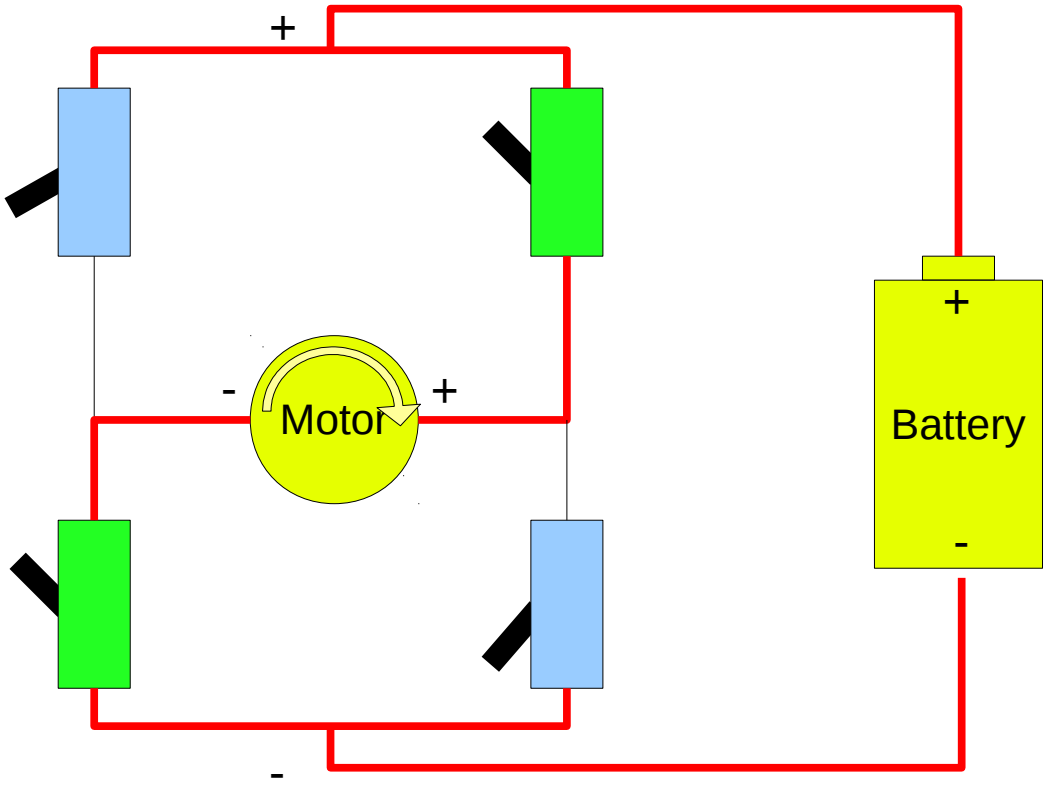
Motor Control

Dual H-Bridge Driver is used to control the motors. It uses four transistors to control the polarity of the voltage supplied to the motor. The transistors are used as switches turning on and off. Below shows the H-bridge driver circuit and the current flows.



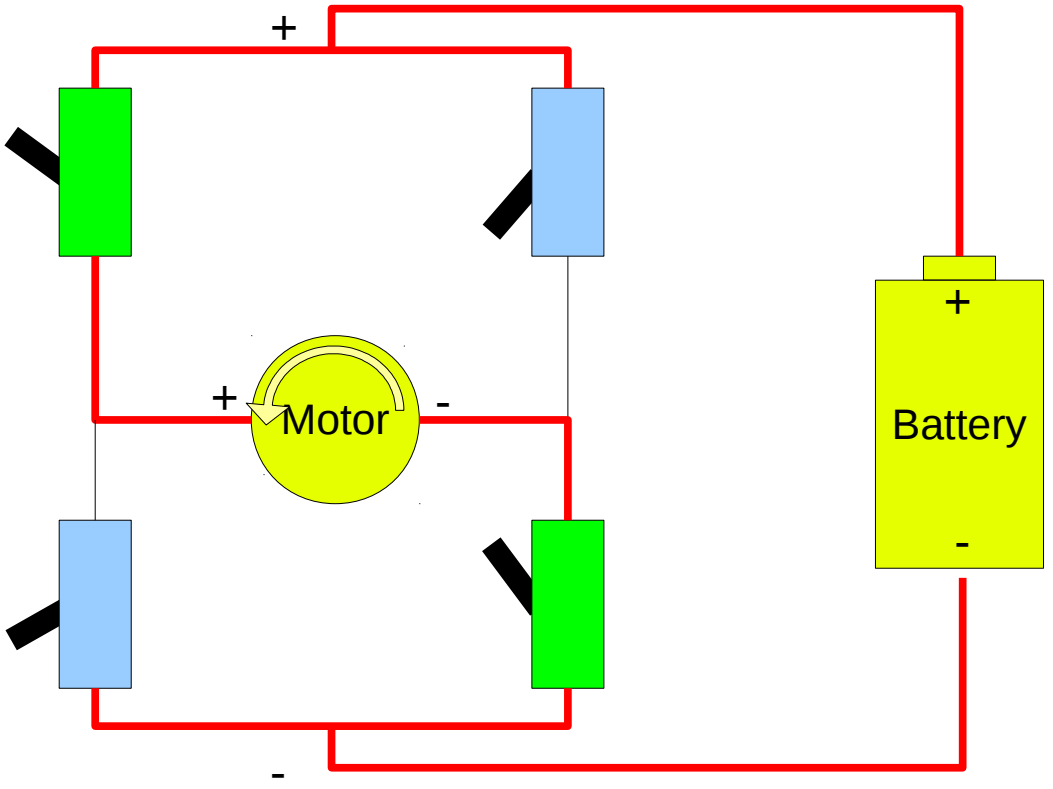
Motor Control

To make the motor turn on one direction, two switches need to be turned on to let power get to the motor. One switch connects the positive side of the battery to to one side of the motor and another switch connects the negative side to the other side of the motor.



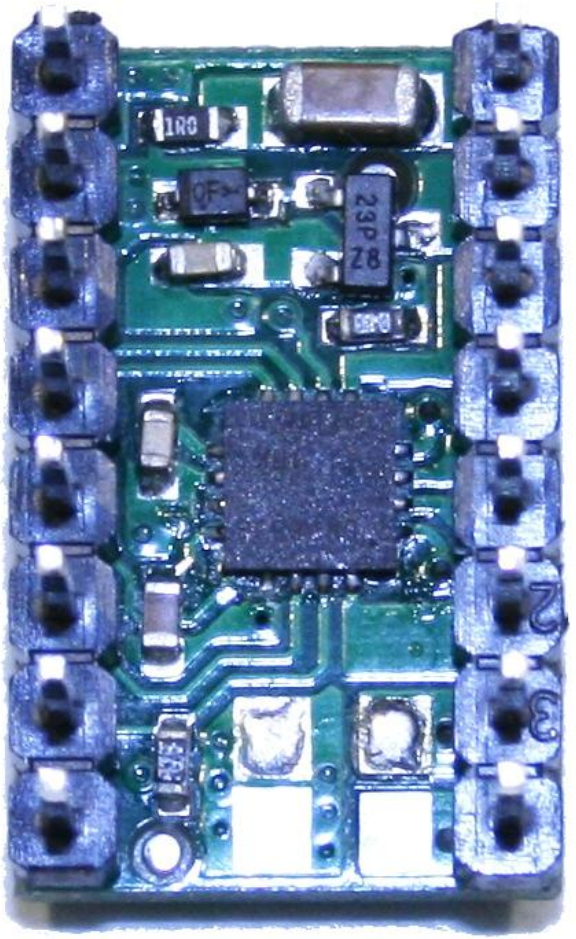
Motor Control

Flip all the switches to the opposite position and the motor turns in reverse. Notice the polarity signs on the motor switched sides.



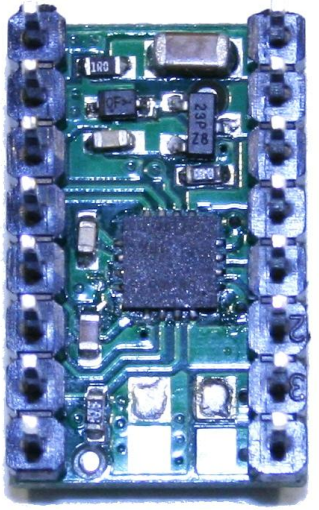
Motor Controller

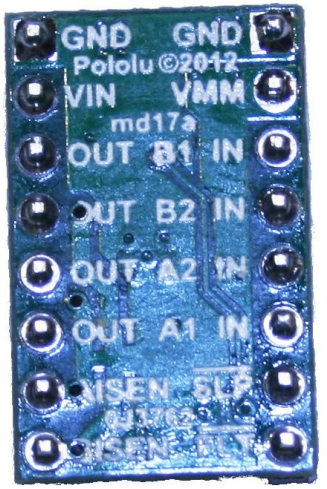
- The motor controller is the interface between the motors and the processor board. It has circuitry to allow control of the motors and can handle the high currents required to operate the motors. The processor board cannot directly power the motors. The controller is capable of providing the needed current and is used as the interface.



H-Bridge Driver

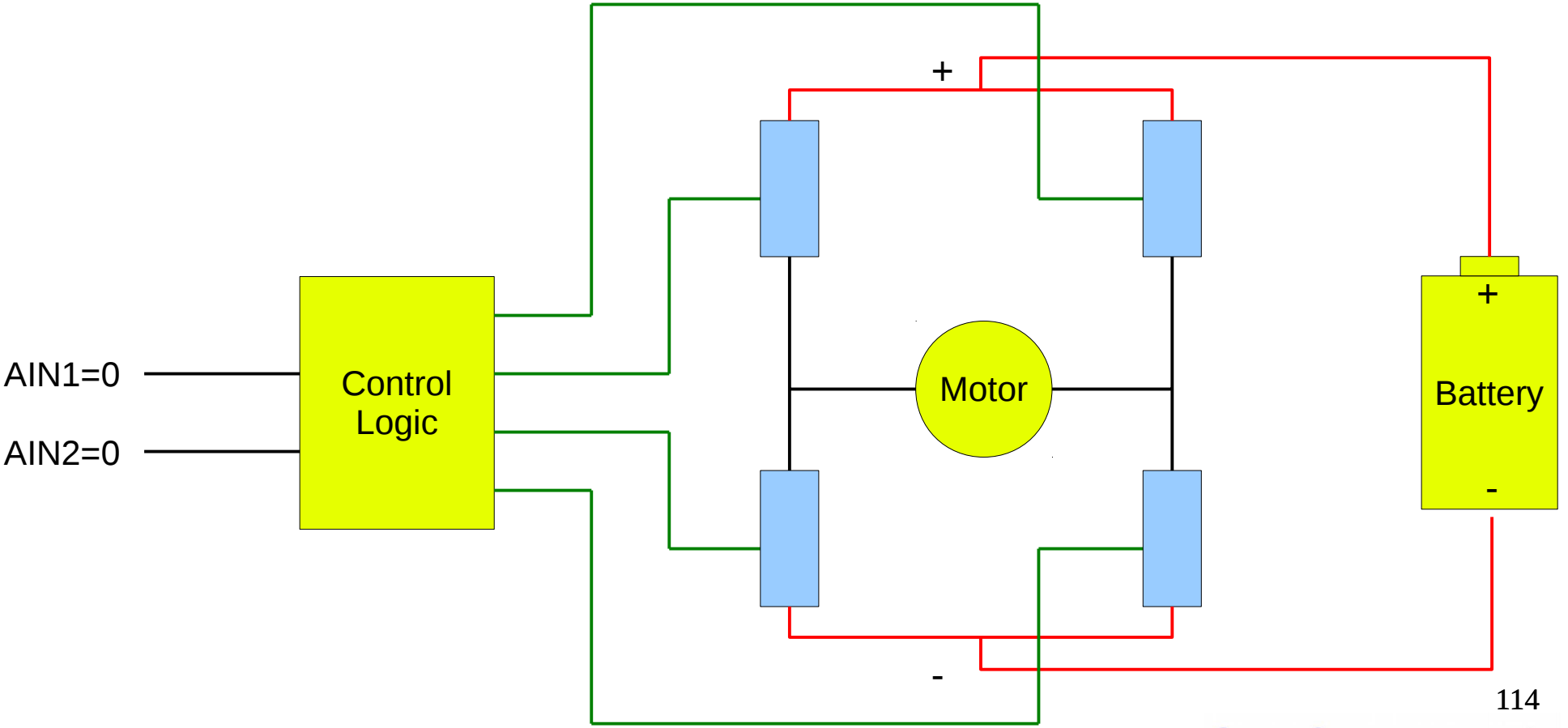
- The motor controller module consists of two H-bridge drivers to control two motors.
- The circuit side is shown at the top right. The square block in the center contains the two motor drivers.
- The bottom picture shows the signal names next to the pins.
- Power is supplied at pins GND and VIN.
- Control signals for each motor is A1 IN, A2 IN, and B1 IN, B2 IN.
- The motors connect to the pins marked OUT.
- The other pins are not used.





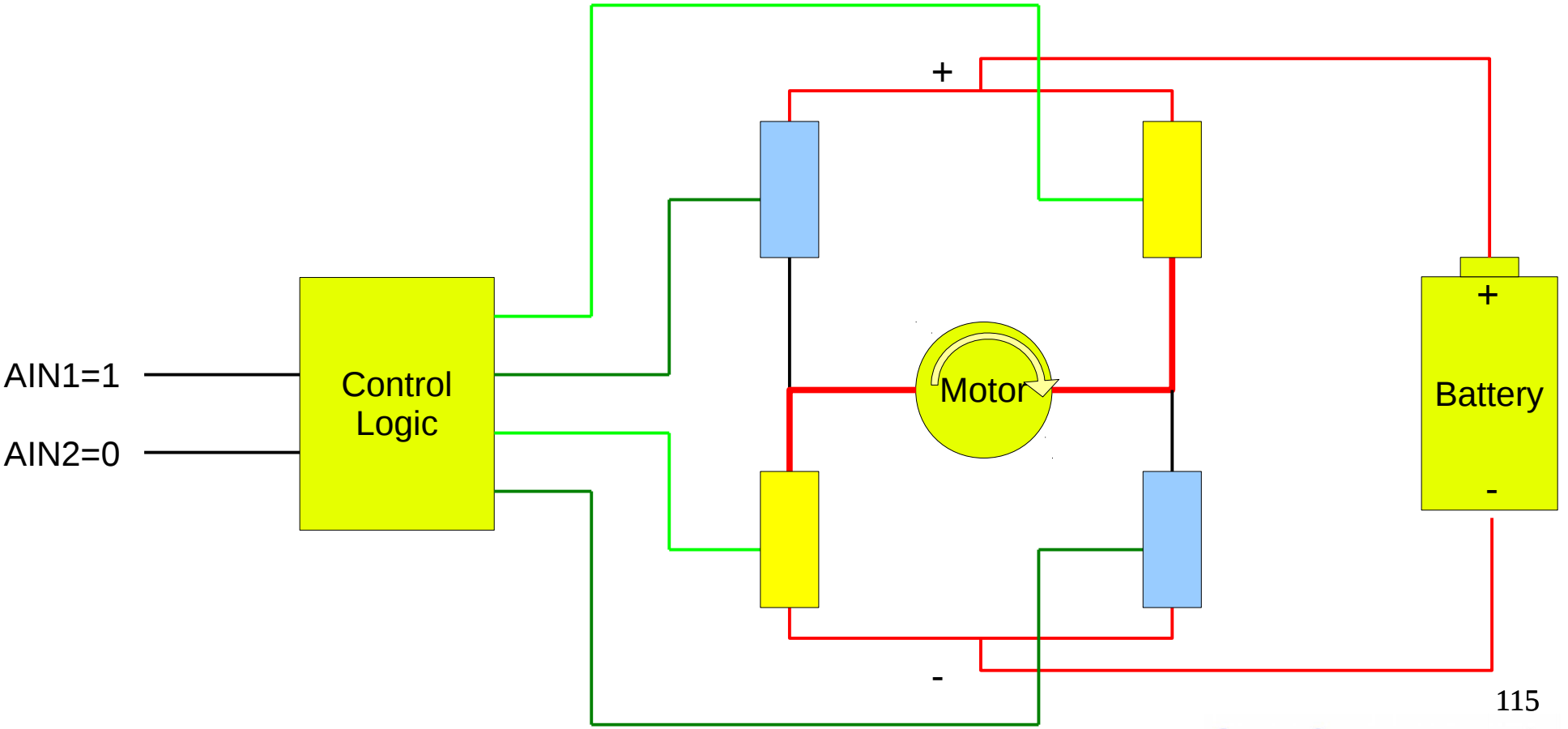
How the H-Bridge Driver Works

- This drawing shows how the H-Bridge driver works. Only one is shown.
- There are two signals that control the direction and operation. Control logic decodes the two signals and turns on the appropriate switches to control the motor. The drawing shows the condition of AIN1 and AIN2 set to logic zero.



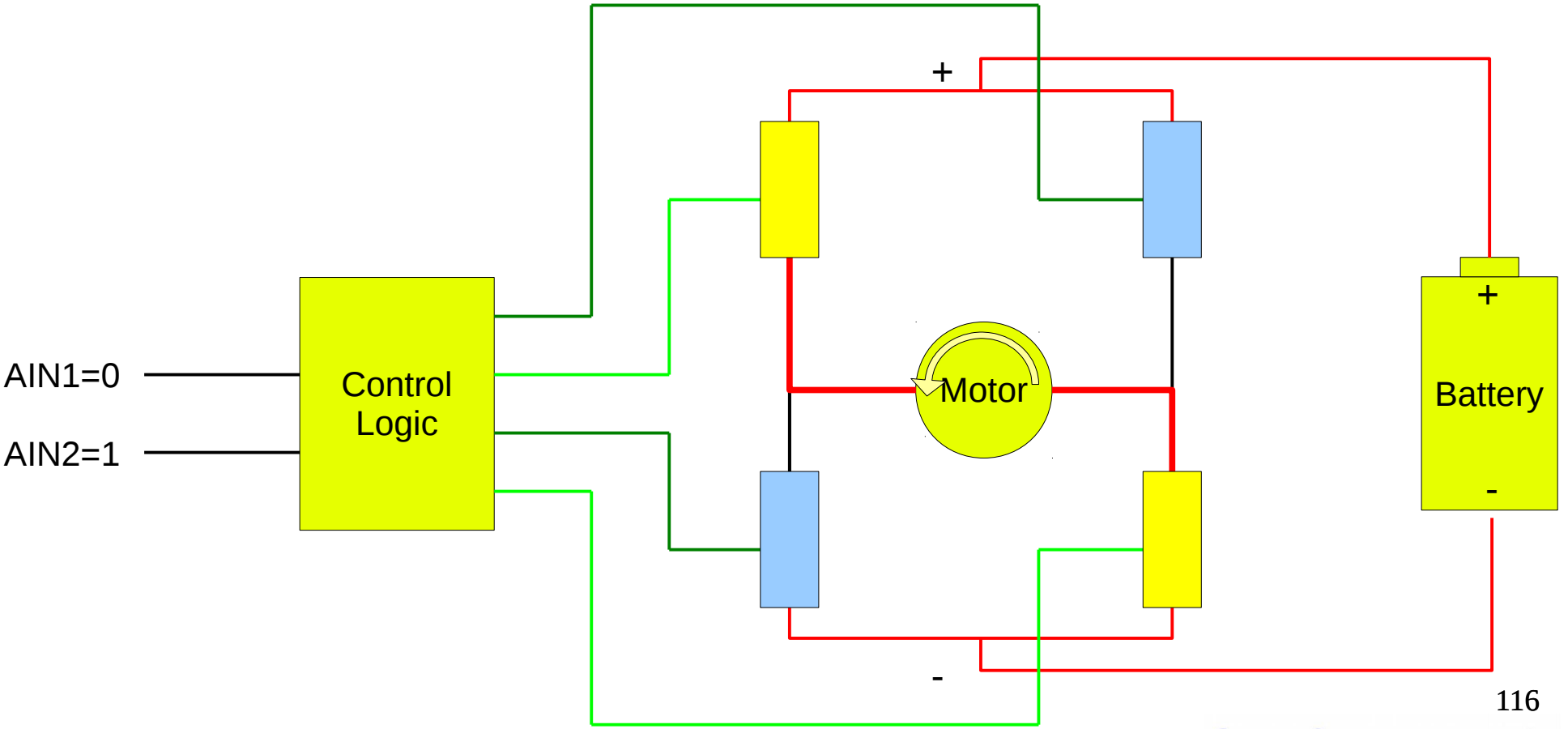
How the H-Bridge Driver Works

- When AIN1 is set to logic 1, the motor drives in the forward direction.
- You will notice that setting AIN1 = 1, and AIN2=0 turns on two signals that turn on the two switches.



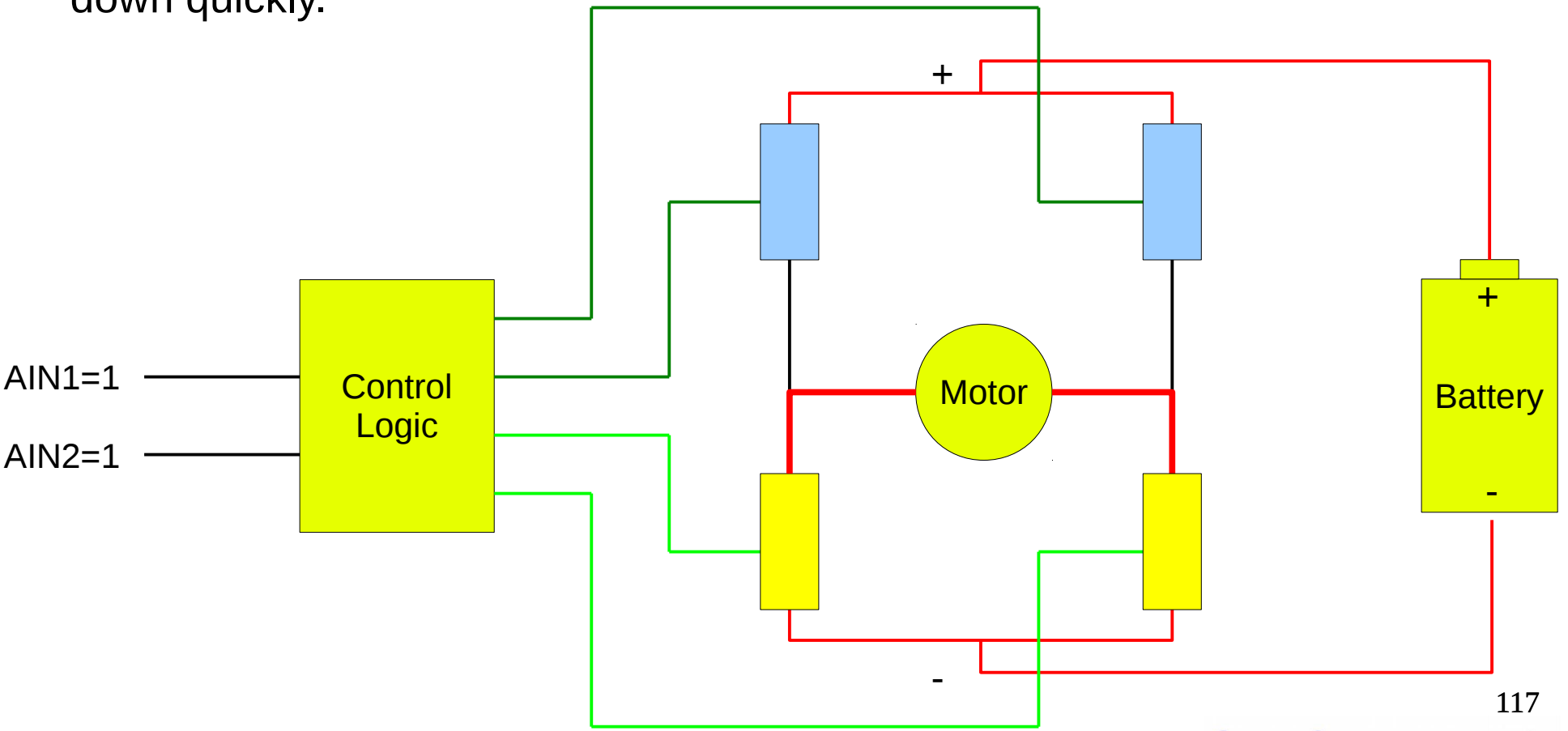
How the H-Bridge Driver Works

- When AIN1 is set to logic 1, the motor drives in the reverse direction.
- You will notice that setting AIN1 = 0, and AIN2=1 turns on two signals that turn on the two switches.



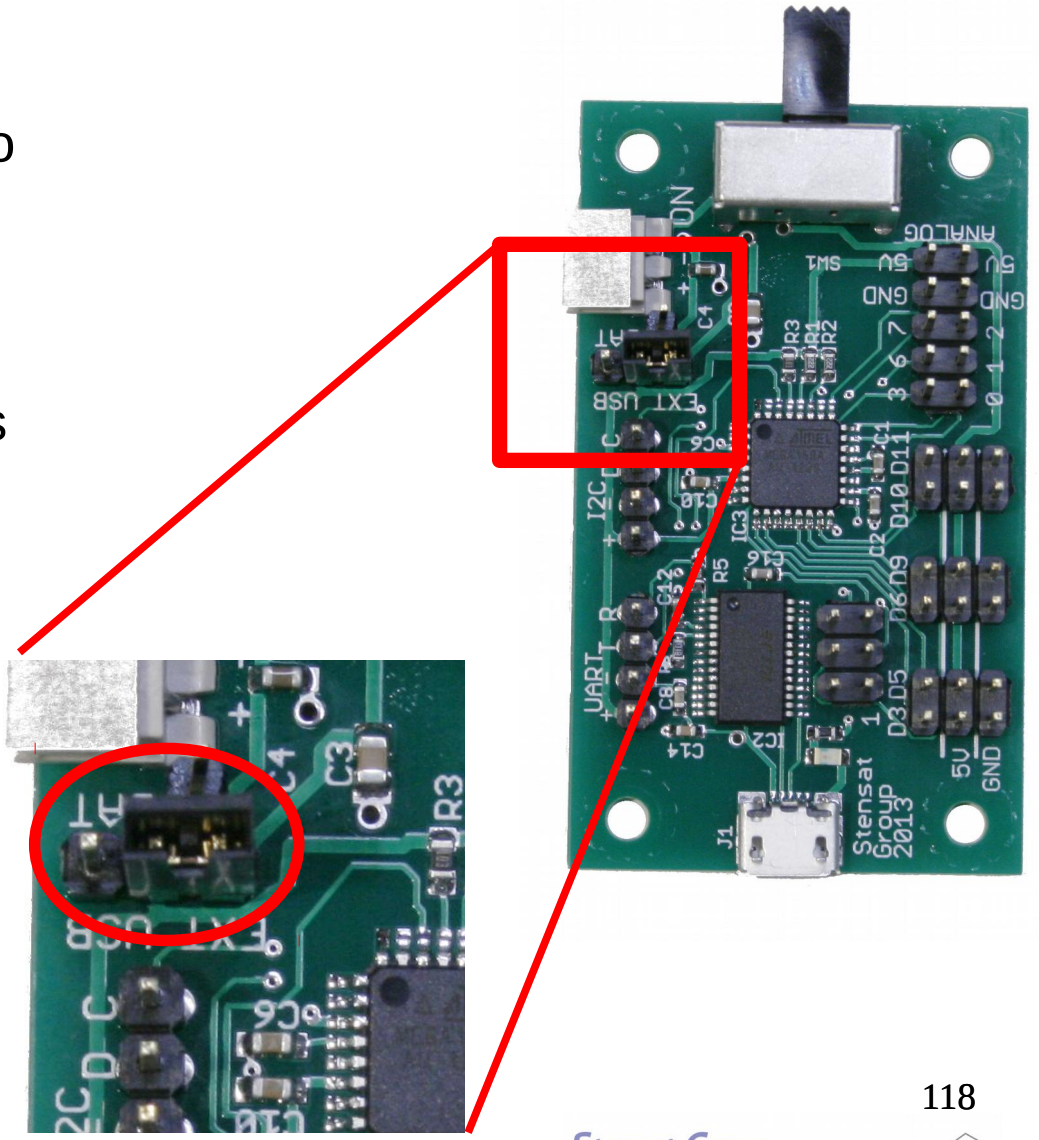
How the H-Bridge Driver Works

- When you set both AIN1 and AIN2 to logic 1, you get a breaking action.
- This turns on the two bottom switches which shorts the motor connections together. The inductance created by the motor turning in one direction will power the motor to turn in the opposite direction. It causes the motor to slow down quickly.



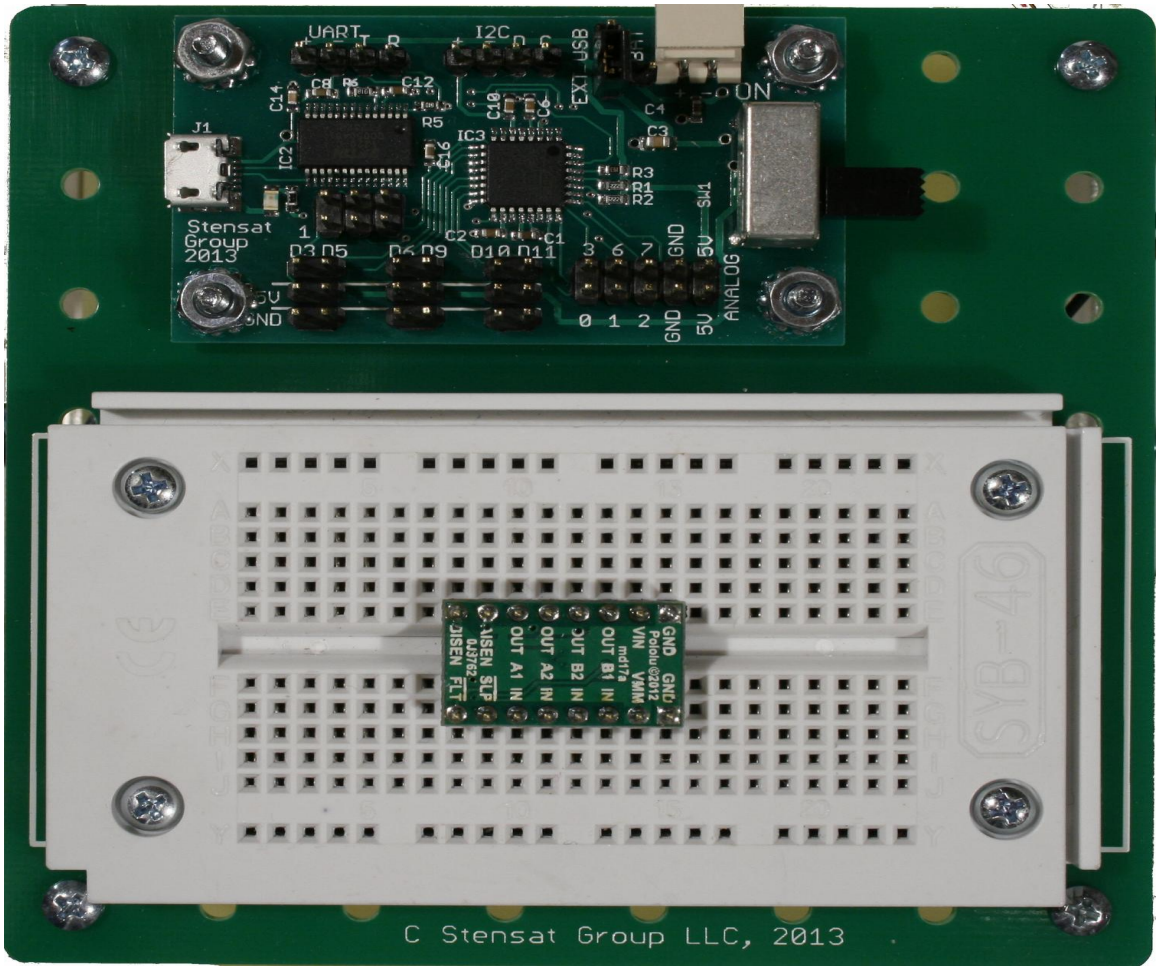
Power Selection for Programming

- Before continuing, the power selection shorting jumper needs to be moved to the EXT side.
- Pull the shorting jumper from the two pins and insert where it is marked EXT.
- Since the USB port cannot provide enough power to operate the motors and should not be connected anyway, the power selection needs to be switched to external. This will allow the batteries to power the processor and the motors.
- Now that external power is selected, the power switch must be in the ON position for programming. Don't let the robot drive off the table during programming.



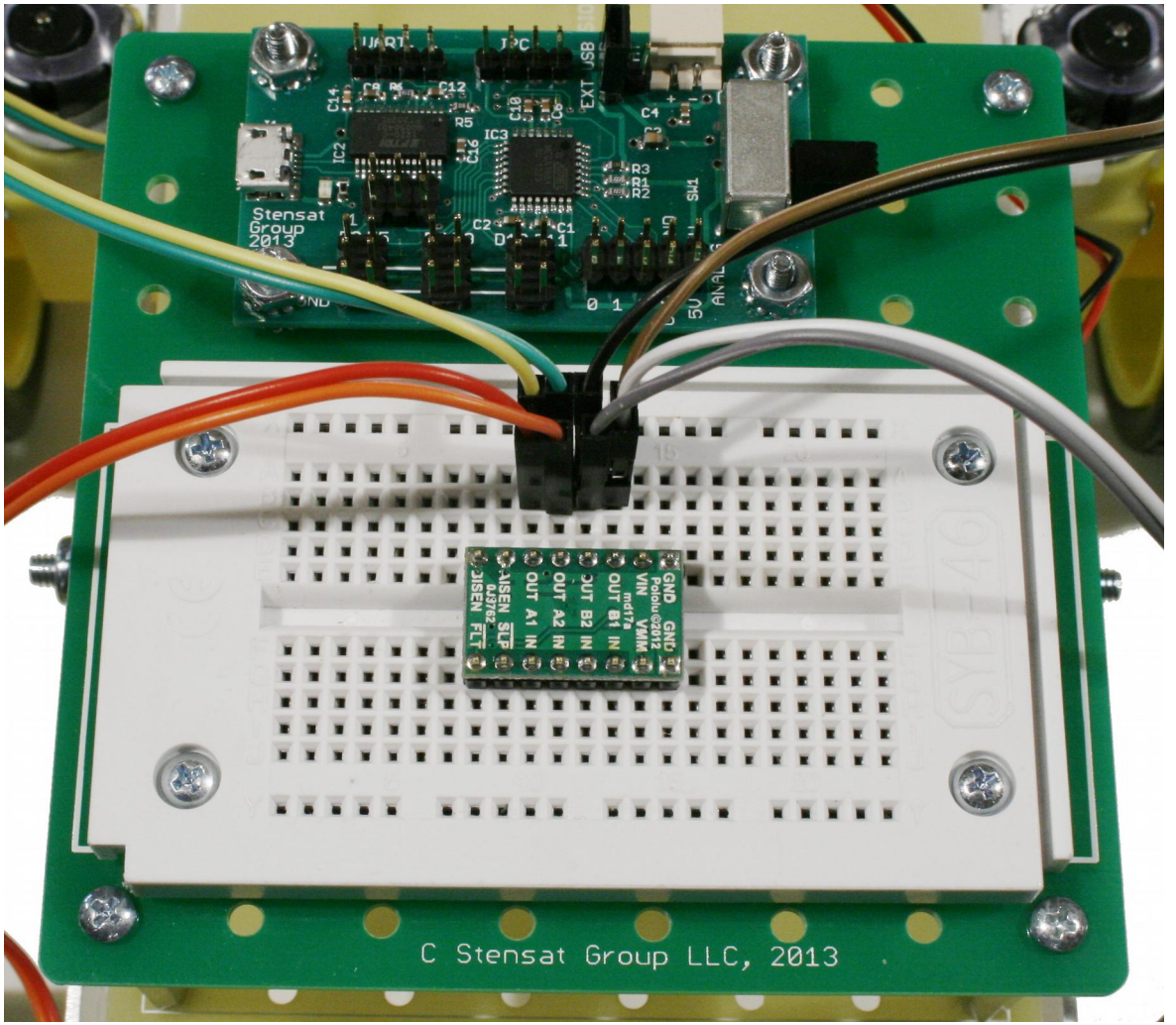
Mount the Motor Controller

- Orient the motor controller module so that the words OUT are toward the processor board. This is so the wires from the motors can reach.
- Insert the motor controller module into the solderless breadboard as shown. The rows of pins need to on either side of the gap down the center of the board.



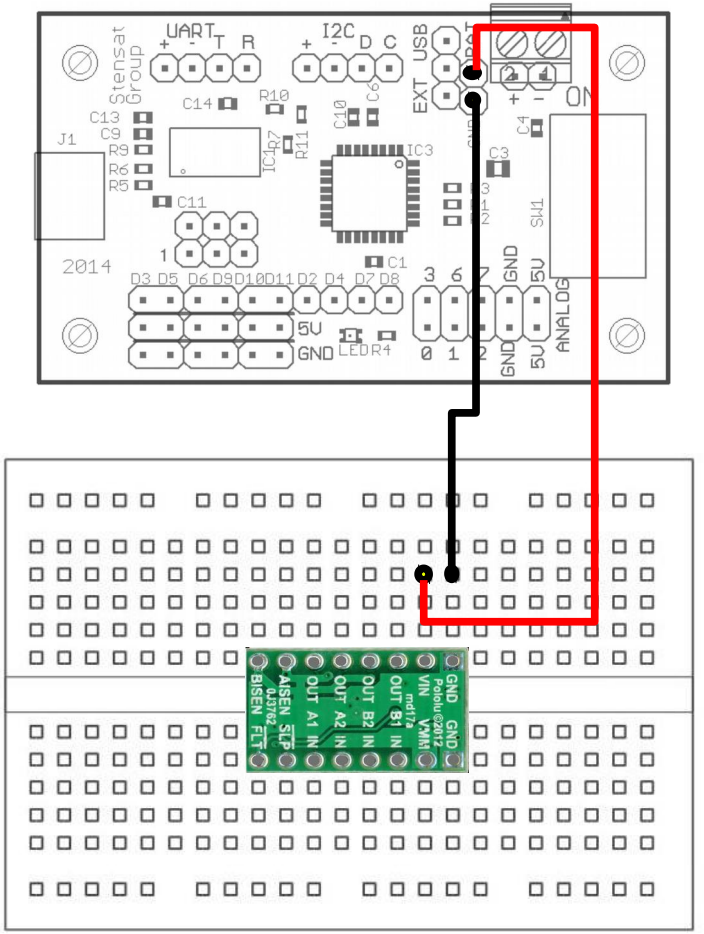
Connect the Motors

- Take the pin connectors from the motors and connect to the motor driver.
- With the rover front facing you, take the left rear motor wires and plug into OUT A1 and OUT A2. It doesn't matter which wire is connected where.
- Do the same for the front left motor.
- Take the wires from the rear right motor and connect to OUT B1 and out B2.
- Do the same for the front right motor.
- Make sure the wires are connected in the correct rows.



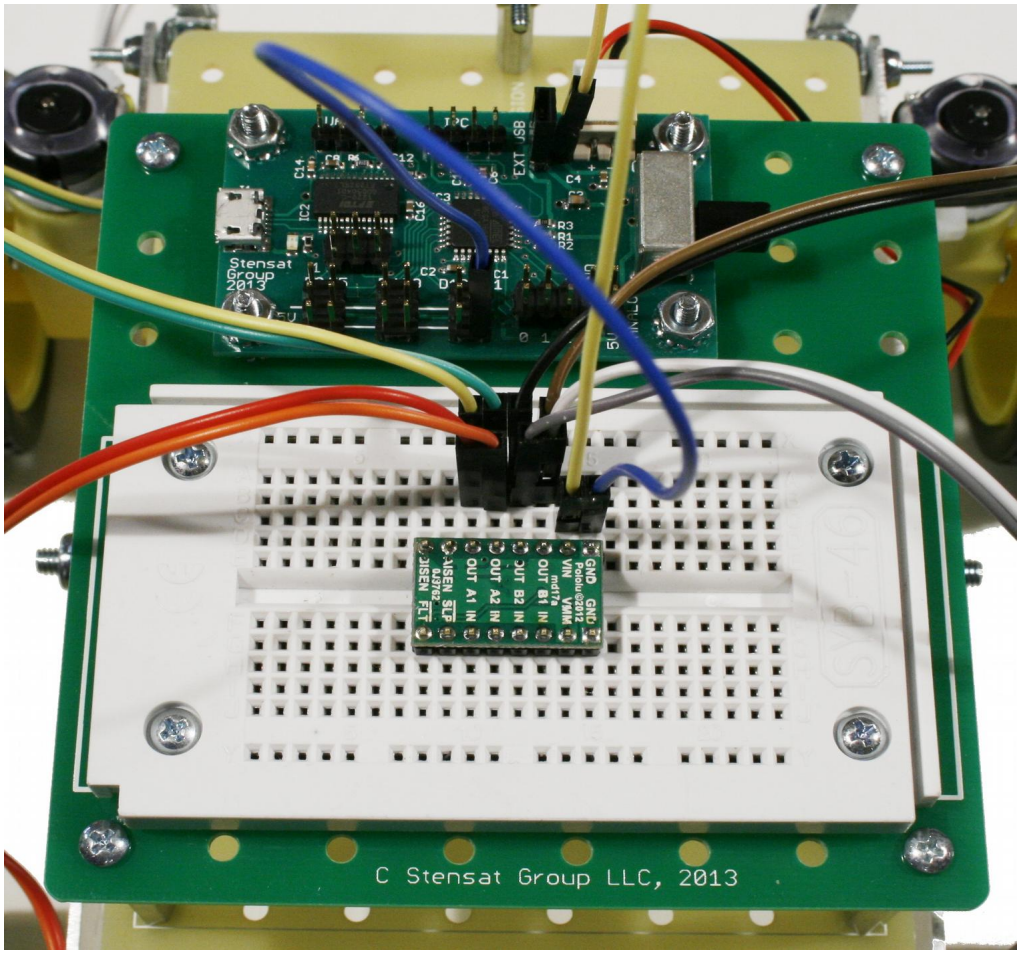
Connecting Power

- Connect a jumper wire from the pin marked BAT to the signal VIN on the motor driver module.
- Connect a jumper wire from the pin next to BAT marked GND to GND on the motor driver module.
- The next page shows the jumpers.



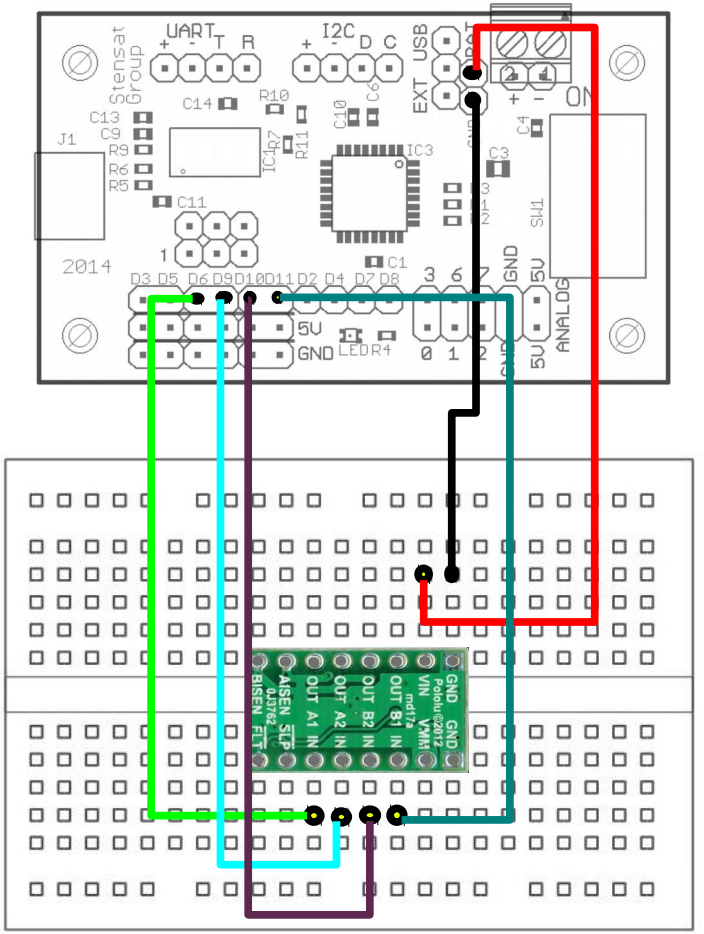
Motor Controller Power Connection

- Use a black wire and connect the pin to the solderless bread board where the motor controller has GND. There are two. Either will work.
- Connect the other end of the black wire to GND on GND below D11 on the processor board.
- Connect the pin of a red wire to VIN on the solderless bread board and the other end to a single pin next to the power connector. This pin connects directly to the battery.

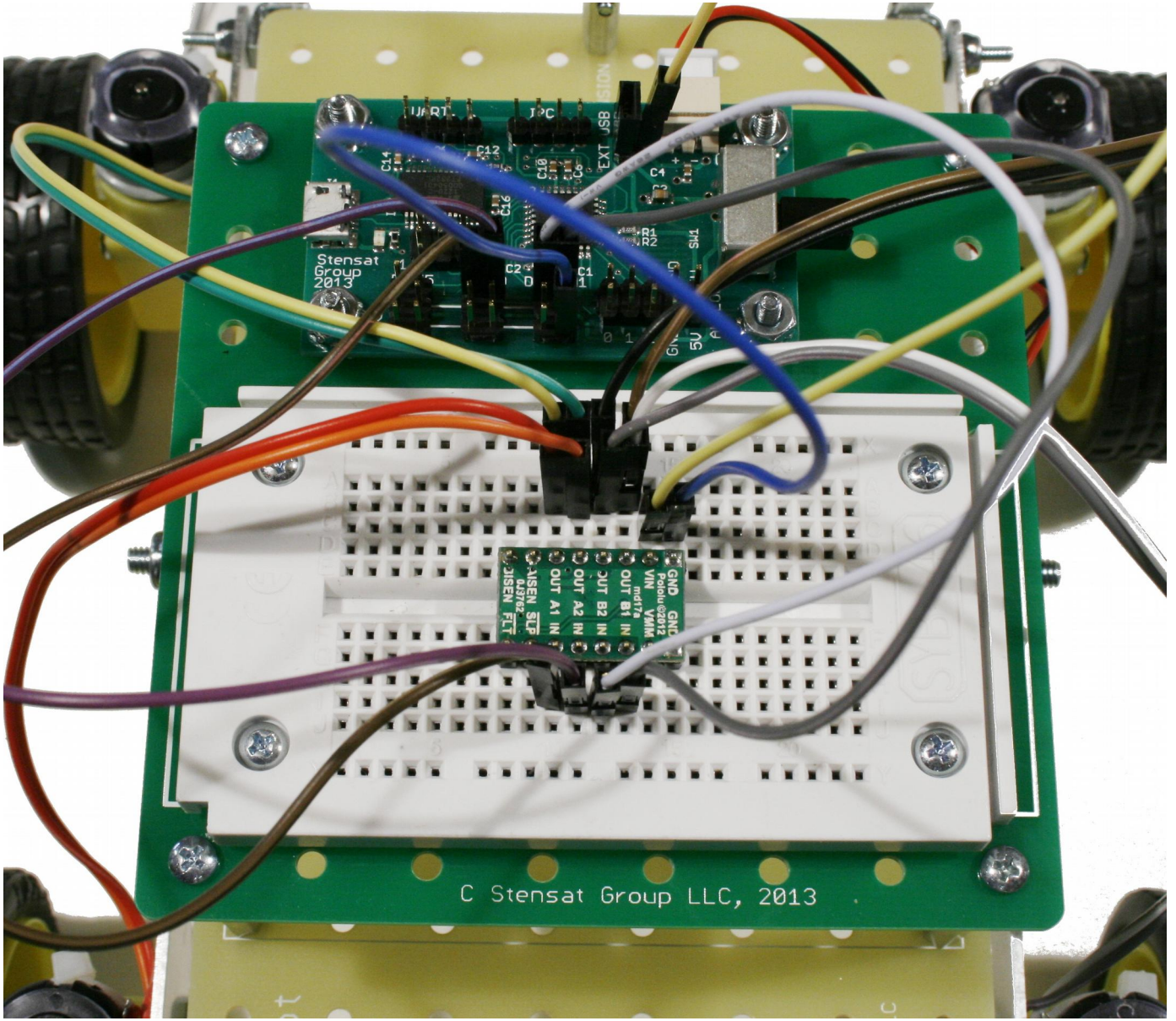


Wiring The Motor Controller

- Use the jumper wires to connect the motor controller.
- Connect A1 IN to D6
- Connect A2 IN to D9
- Connect B1 IN to D10
- Connect B2 IN to D11
- This completes the connections between the motor controller and the processor board.



Digital Signal Connections



Testing the Motors

- To operate the motors, A1 or A2 need to be set high or low.
- Operation is simple if A1 and A2 are set off, the motors do not operate.
- If A1 is set high and A2 is low, the motors will turn one direction.
- If A1 is low and A2 is high, the motors will turn in the opposite direction.
- The same applies for B1 and B2.
- Enter the program on the right to turn the motors on.
- See which way the wheels are turning and swap the motor pins if needed to make the wheels spin forward.
- These pin settings will be used for forward motion.

```
void setup()
{
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}

void loop()
{
    digitalWrite(6,HIGH);
    digitalWrite(9,LOW);
    digitalWrite(10,HIGH);
    digitalWrite(11,LOW);
    delay(5000);
    digitalWrite(6,LOW);
    digitalWrite(10,LOW);
    delay(2000);
}
```

Direction Control

- The digital pins D6 and D9 control the right motors.
  - Setting D6 high and D9 low makes the right wheels spin forward.
  - Setting D6 low and D9 high makes the right wheels spin reverse.
  - Setting D6 low and D9 low turns off the motors.
- The digital pins D10 and D11 control the left motors.
  - Setting D10 high and D11 low makes the left wheels spin forward.
  - Setting D10 low and D11 high makes the left wheels spin reverse.
  - Setting D10 low and D11 low turns off the motors.
- Making the left motors go forward and the right motors go reverse turns the robot right.
- Making the left motors go reverse and the right motors go forward turns the robot left. The next page shows the code for each direction.

Direction Control Code

Forward Motion
```
digitalWrite(6,HIGH);
digitalWrite(9,LOW);
digitalWrite(10,HIGH);
digitalWrite(11,LOW);
```

Reverse Motion
```
digitalWrite(6,LOW);
digitalWrite(9,HIGH);
digitalWrite(10,LOW);
digitalWrite(11,HIGH);
```

Stop
```
digitalWrite(6,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
digitalWrite(11,LOW);
```

Right Turn
```
digitalWrite(6,LOW);
digitalWrite(9,HIGH);
digitalWrite(10,HIGH);
digitalWrite(11,LOW);
```

Left Turn
```
digitalWrite(6,HIGH);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
digitalWrite(11,HIGH);
```

Creating Functions

- To make programming easier, functions will be created to specify the motions of the robot.
- A function is a collection of instructions that are grouped and given a name.
- The format is shown to the right with one of the motions.
- The code in the loop function can call the forward function eliminating the need to rewrite the **`digitalWrite()`** functions every time.
- The motion functions should be inserted at the top of all programs.

```
void forward()
{
  digitalWrite(6,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}
```

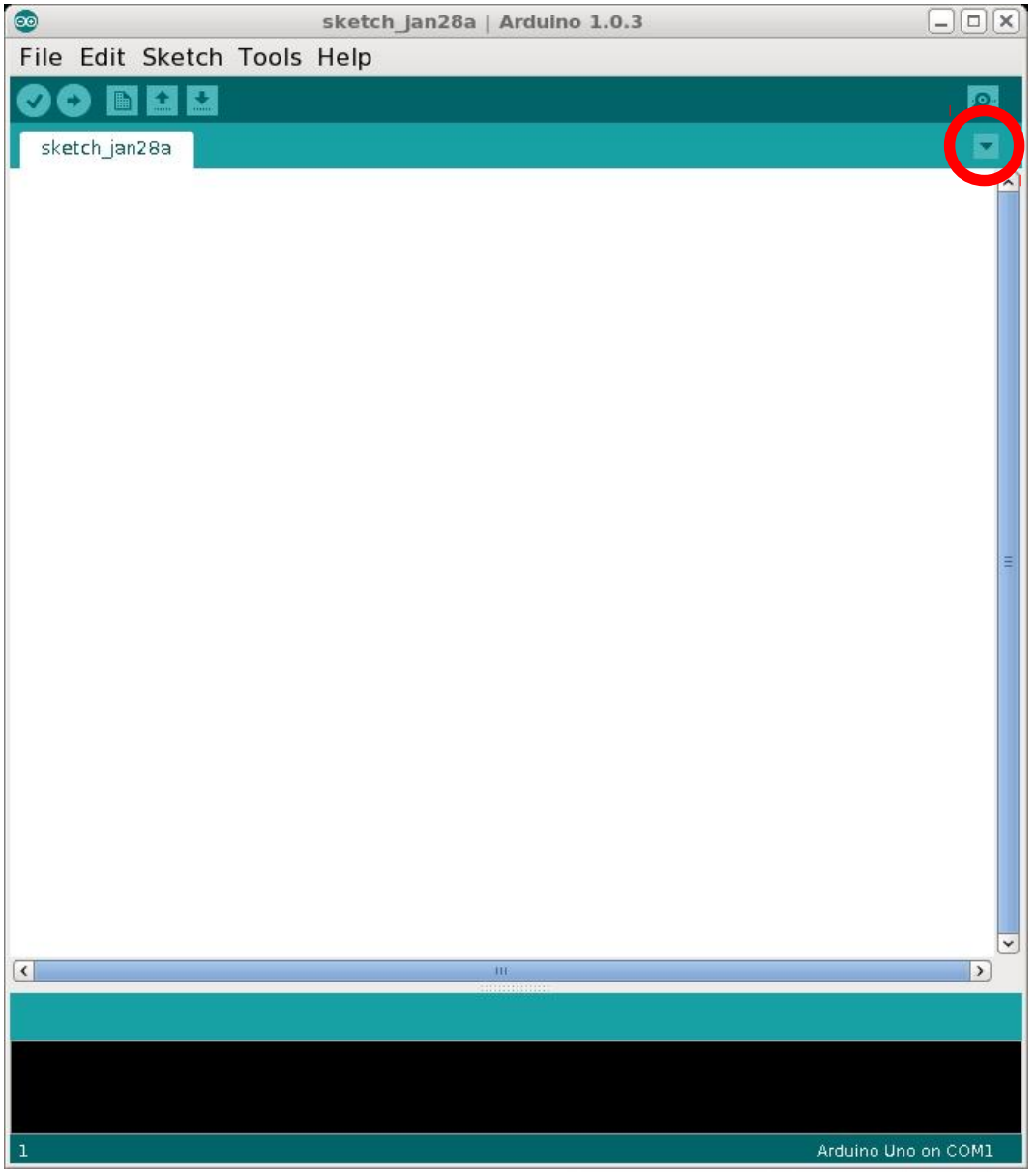
Motion Functions

```
void forward()
{
  digitalWrite(6,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}

void reverse()
{
  digitalWrite(6,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
}
void stop()
{
  digitalWrite(6,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
```

```
void right()
{
  digitalWrite(6,LOW);
  digitalWrite(9,HIGH);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}

void left()
{
  digitalWrite(6,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,HIGH);
}
```


Creating a Separate Function File

- Start a new program with the Arduino program.
- Click on the down arrow to the right where circled in red.
  - A menu will open. Select “New Tab”
  - Below, it will ask for a name. Enter 'move'
    - Do not put .c on the end of the name or the program will not compile properly.
  - Click 'OK'
  - A new tab is created called 'move'
  - You will enter all the movement functions here.
- Enter the functions listed in the previous page.



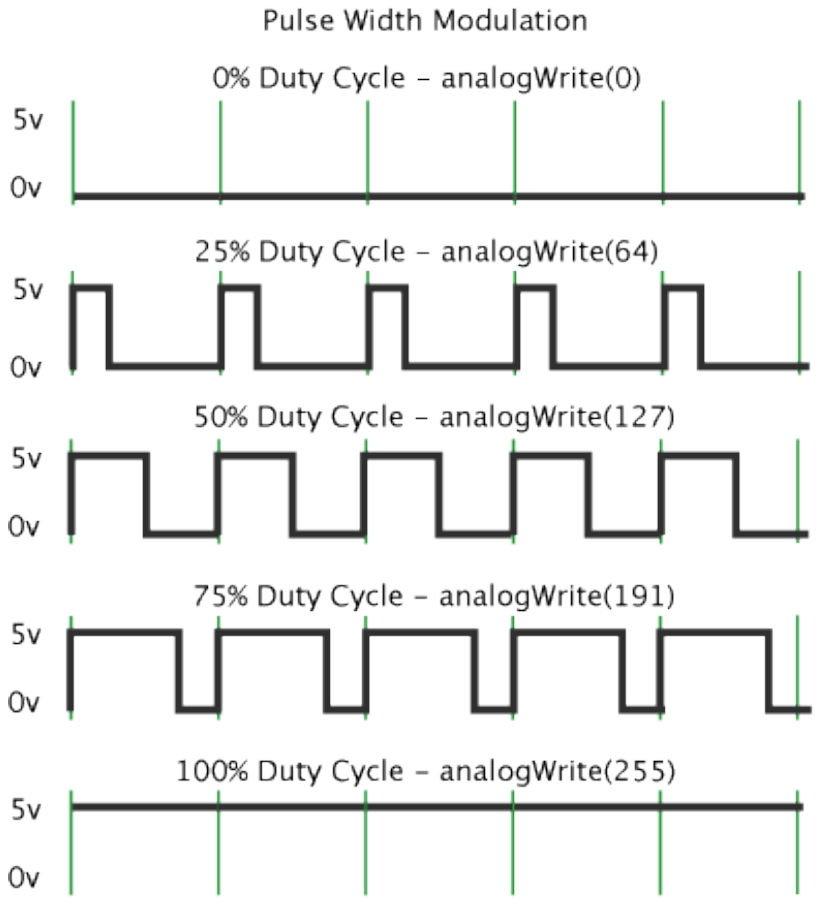
Driving Around

- Click on the tab to the left of 'move' tab. Enter the code to the right.
- The code to the right is a start.
- Notice the **`delay()`** function is included after each motion function. This give the robot time to perform that motion. The value included in the delay function is time in milliseconds.
- Add directions to the program and change up the delays. Come up with a complex set of motions. Always remember to include a delay after the function to move the robot.

```
void setup()
{
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}

void loop()
{
    forward();
    delay(1000);
    left();
    delay(400);
    reverse();
    delay(1000);
}
```

Speed Control

- It may be noticed that the robot may tend to drift to the left or right. This is due to the motors not being equally powerful.
- There is a way to attempt to equalize them by controlling their speed.
- A simple way to control the speed is to pulse power to the motors. This technique is called pulse width modulation.
- On the arduino, the **`analogWrite()`** function performs this. It generates a repeating pulse at about 250 Hz.
- The size of each pulse is the duty cycle. The higher the duty cycle the more power the motor gets.
- Adjusting the duty cycle will adjust the motor speed.



analogWrite()

- The function **`analogWrite()`** function takes two values.
  - First is the pin number.
  - Second is the duty cycle represented as a value from 0 to 255.
    - 0 is 0% duty cycle.
    - 255 is 100% duty cycle.
    - 127 is 50% duty cycle.
  - The function is written as
    - **`analogWrite(pin,duty)`**;

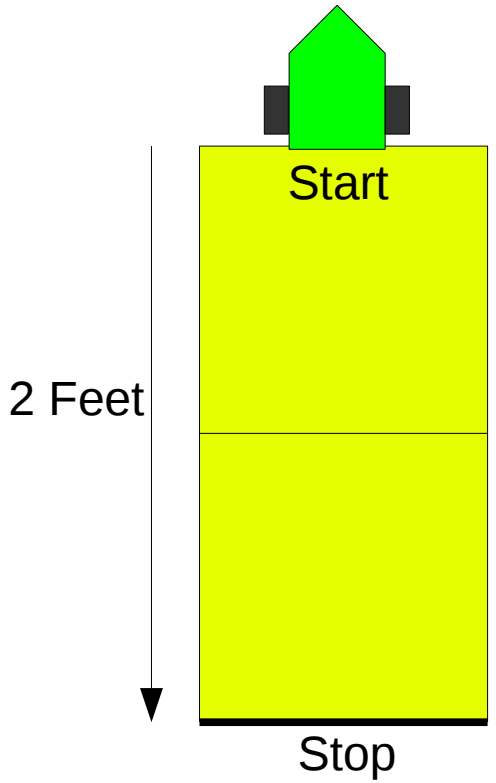
Controlling Motor Speed

- Enter the program to the right. This program generates a PWM signal to the motor. Only one side needs a PWM signal. The other is set to 0 so no PWM signal is present.
- The code sets the PWM signal to 255 which is 100% duty cycle meaning it is on all the time. This is the same as **`digitalWrite()`** function.
- Run the code and see which direction the robot drifts.
- Reduce the value for the opposite direction by 10 and try again. Keep adjusting until the robot drives relatively straight. It won't be perfect.
- The **`analogWrite()`** functions can replace the **`digitalWrite()`** functions in the motion functions.

```
void setup()
{
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}

void loop()
{
    analogWrite(6,255);
    analogWrite(9,0);
    analogWrite(10,255);
    analogWrite(11,0);
    delay(5000);
    analogWrite(6,0);
    analogWrite(10,0);
    delay(2000);
}
```

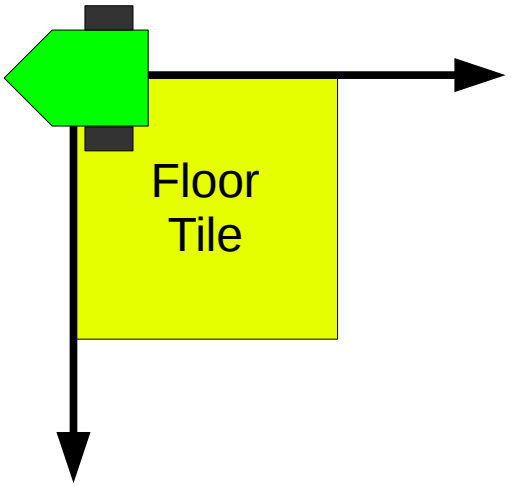
Calibrating Travel Distance

- Since there is no feedback on the motors to detect distance or wheel rotation, time will be used to specify the distance and the amount of turning.
- Mark off two feet on the floor. Floor tile is usually 1 foot square.
- Write a program to move forward two feet and stop. Start with a delay of 1000 ms.
- Adjust the delay until the robot travels two feet. Keep this value.
- If necessary, adjust the PWM values to keep the robot as straight as possible.



Calibrating Turns

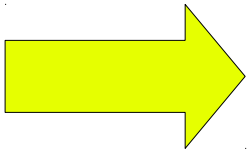
- Now mark on the floor a right angle. If the floor has tiles, use the corner of a tile for your right angle.
- Program the robot to turn right and set the delay to 400 ms and turn off.
- Place the robot on the corner of the right angle facing the left line.
- See how much the robot turns and adjust the delay until it turns 90 degrees.
- Verify the value turning left and adjust if necessary.



New Movement Functions

- The movement functions can be modified using the analogWrite() function. Go back to the move file and replace all the digitalWrite() functions with analogWrite().
- The pin numbers stay the same but the second parameter gets changed. Anywhere there is a 'LOW', replace it with 0. Anywhere there is a 'HIGH', replace it with the number to set the speed.
- Save the program under a new name so the original is not overwritten.
- Example is below:

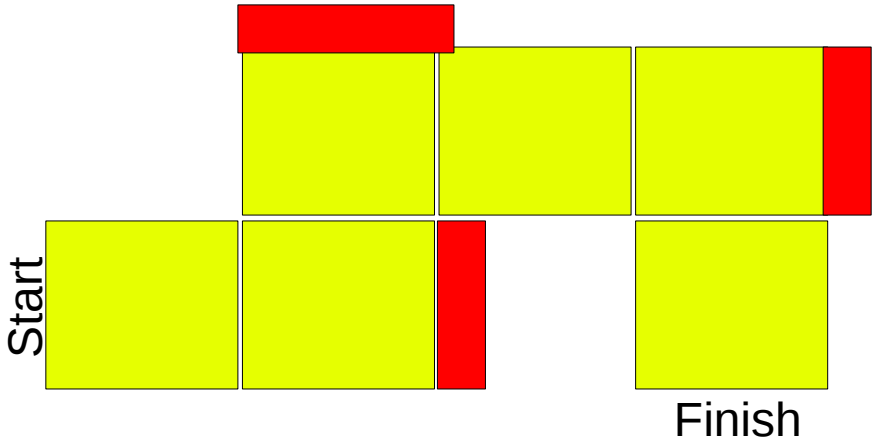
```
void forward()
{
  digitalWrite(6,HIGH);
  digitalWrite(9,LOW);
  digitalWrite(10,HIGH);
  digitalWrite(11,LOW);
}
```



```
void forward()
{
  analogWrite(6,250);
  analogWrite(9,0);
  analogWrite(10,248);
  analogWrite(11,0);
}
```

Obstacle Course Time

- Now for the fun part. Modify and expand the program to go through the obstacle course shown below. The large square represent 2 foot grids.
- This attempt through the course is called dead reckoning. Write a program to go through a sequence of motions to reach the end of the course. Adjust the delays between each motion so the rover travels to the finish line without going on off the squares.
- Change the course and make it more complex if you want.

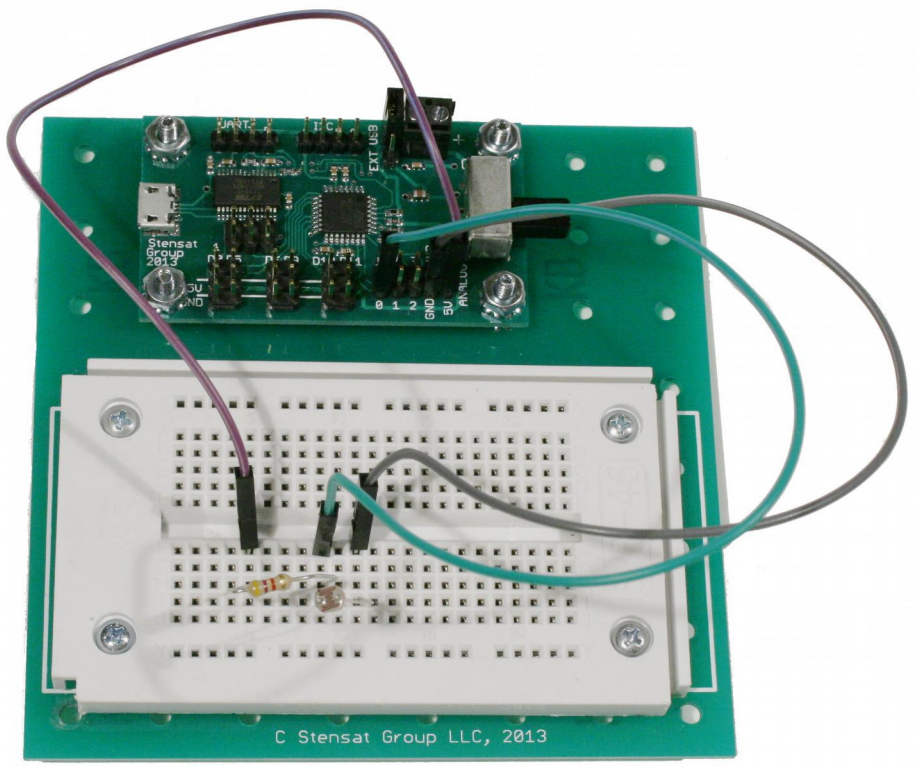


Robot Sensing

- This section, you learn about using sensors to control the robots movements.

Photo Cell

- The photo cell is a light sensitive device that changes its resistance based on light intensity.
- The photocell can be used in a simple voltage divider circuit with another resistor. The resistor is 4.7Kohms.
- The photo resistor will have a resistance ranging from 1 Mohm in darkness to 100 ohms in bright light.
- Install the photo cell and 4.7 K resistor on the solderless bread board. Make sure the photo cell and resistor are connected.
- Connect the free end of the resistor to GND at the analog connector.
- Connect the free end of the photo cell to 5 volts.
- Connect the resistor and photo cell connection to pin 0 of the analog connector.



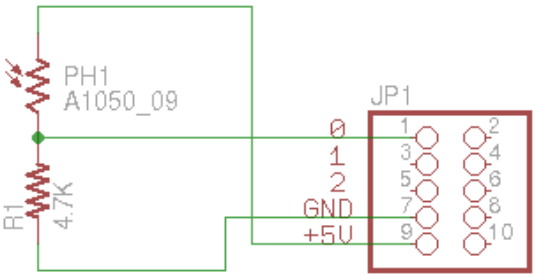


Photo Cell Program

- The program to the right will get an ADC value from analog port 0.
- Create a new program and enter the code.
- To measure the voltage, the function **`analogRead(port)`** is used.
- Six ports are available on the processor board.
  - 0,1,2,3,6,7
  - Refer to page 5 for the location.
- Once the ADC value is read, it can be converted to a voltage value. The code to the right shows the equation which can be used for all the analog ports.
- The **`Serial.println()`** function that displays the volts, includes a numeric argument which specifies the number of decimal places.
- Save the program to a new file.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int a;
  float volts;
  a = analogRead(0);
  Serial.println(a);
  volts = (float)a/1023.0 * 5.0;
  Serial.println(volts,2);
  delay(200);
}
```

Light Seeking Program

- The photo cell can be used to have the robot chase after a light source.
- Get a flash light and run the program in the previous page. Shine the light on the photo cell at some distant and observe the ADC value. The value should increase. Pick a number a little lower. If the value was around 600 then use 500. Adjustments may need to be made.
- The program will have the robot turn in place when the flash light is not detected and drive straight when enough light is detected.

Light Seeking Code

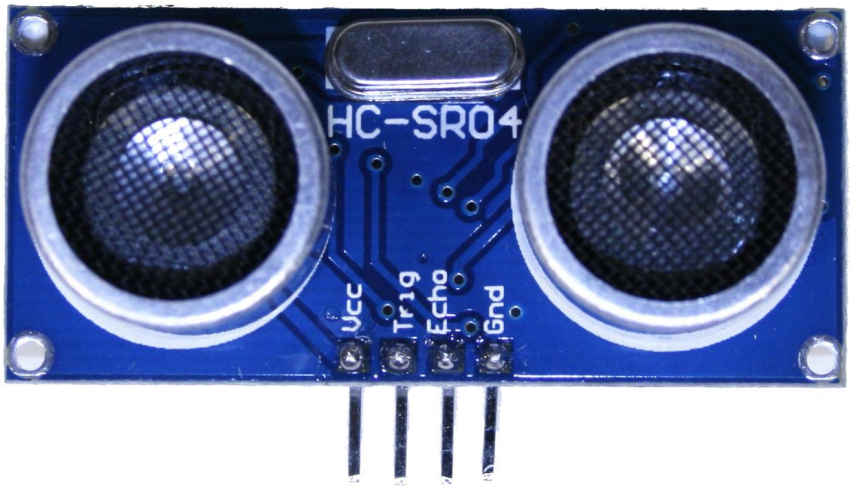
- Don't forget to include the movement functions in a separate tab.
- The code uses a conditional statement called **if**.
- If the photocell circuit detect light, the value from the ADC goes up. If the value is greater than 500, the robot goes forward which should be toward the light source.
- If the photocell does not detect enough light, the robot turns in place. This motion lets the robot scan around until it finds a strong enough light source.
- Depending on the ambient light in the room and the brightness of the flash light, you may need to adjust the number in the **if** statement.
- When the program is loaded see if the robot will chase the flash light.

```
void setup()
{
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  int a = readAnalog(0);
  Serial.println(a);
  if(a > 500) forward();
  else left();
}
```

- Serial.println is included to allow you to see what the ambient light level is and adjust.

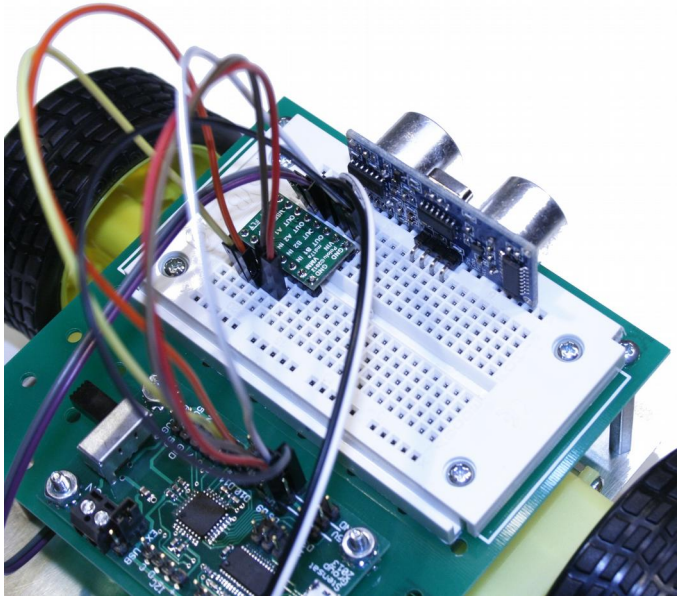
Sensing the Environment

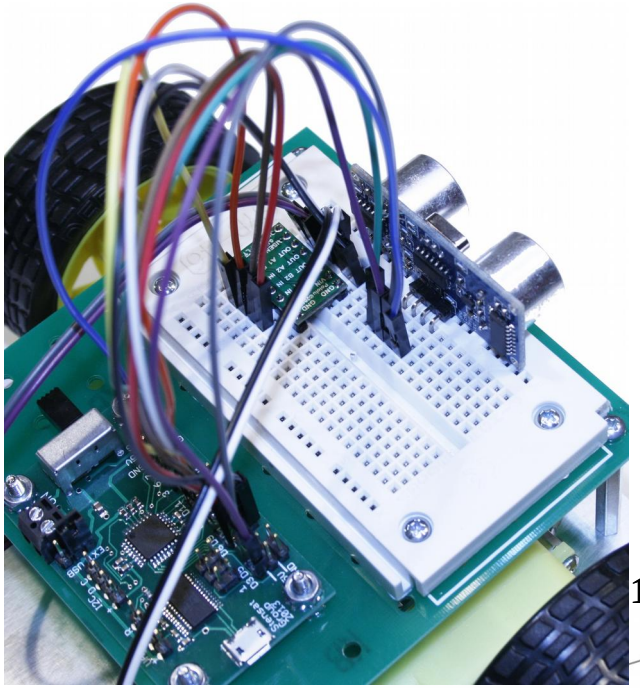
- To detect things in the environment for purpose of collision avoidance, an ultrasonic range sensor will be added to the robot.
- This sensor sends out a burst of audio signal at 40 Khz and detects the echo.
- The processor needs to measure the time it takes for the echo to return.
- This sensor has four pins
  - Ground
  - 5 Volt power input
  - Trigger
  - Echo



Mounting the Ultrasonic Ranger

- Insert the ultrasonic ranger as shown. It should be mounted close to the center of the robot. The pins are inserted at the end of the rows.
- Connect jumpers from the sensor to the processor
  - GND to Analog GND
  - ECHO to pin D3
  - TRIG to pin D5
  - VCC to Analog 5V
- Look on the processor board for the word ANALOG. The power connections are done there to isolate the sensor from the motor power to reduce electrical noise.





Ultrasonic Sensor

- The ultrasonic sensor has two signals, trigger and echo.
- A pulse is sent to the trigger and then the processor is to time when the echo returns.
- This requires two digital pins, one configured as an output and the other as an input. A new command that will be used is called **`pulseIn()`**. This measures the time it takes a pulse to occur in microseconds. Try the program to the right.
- The results are in centimeters.
- Create a new program and enter the code to the right. Save the program and upload it.

```
void setup()
{
   Serial.begin(9600);
   pinMode(3,INPUT);
   pinMode(5,OUTPUT);
}

void loop()
{
   digitalWrite(5,LOW);
   delayMicroseconds(2);
   digitalWrite(5,HIGH);
   delayMicroseconds(10);
   digitalWrite(5,LOW);
   long distance = pulseIn(3,HIGH);
   distance = distance/58;
   Serial.println(distance);
   delay(500);
}
```

Making a Function

- To make this useful for other programs, this program needs to be turned into a function.
- A function is a subroutine or chunk of code that can be called by a name instead of the code being inserted where ever it is needed. This function will return a result.
- The return command specifies which variable is sent back to the calling code.

```
long ultrasonic()
{
    digitalWrite(5,LOW);
    delayMicroseconds(2);
    digitalWrite(5,HIGH);
    delayMicroseconds(10);
    digitalWrite(5,LOW);
    long distance = pulseIn(3,HIGH);
       if(distance == 0) return(1000);
    distance = distance/58;
    return(distance);
}
```

The function **`pulseIn()`** returns the number of microseconds. The result is then divided by 58 to calculate the distance in centimeters.

Conditional Programming

- Now it is time to use the ultrasonic sensor to do collision avoidance.
- The 'if' command will be used to test if the robot will collide with an object.
- The format for the if statement is shown to the right.
- Multiple statements can be inserted between the brackets and will be executed if the condition is true.
- To test for equals, use '=='
- else allows two sets of codes to be executed depending on the condition.

```
if(a < c) {
    execute code here
}

if(a == c) {
    execute this code
}

if(a > c) {
    execute this code
} else {
    otherwise execute this code
}
```

Collision Avoidance Program

- The program on the next page will use the code used to control the motors, the ultrasonic function, and the conditional command.
- Put together, the program will keep the robot from bumping into anything.
- Enter the code on the next page. The code should be written in a single file. The code is split on the next page since it wouldn't fit in a single column.
- Test it and see if you need to tweak the timing for going reverse and turning.
- Don't forget to include the movement functions in a separate tab.
- Save the program and then upload it.
- Change the code to turn a different direction.

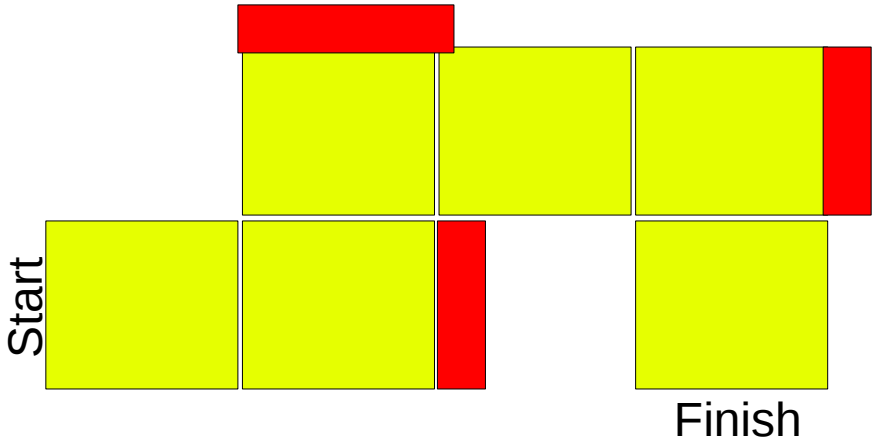
Collision Avoidance Program

```
long ultrasonic()
{
    digitalWrite(5,LOW);
    delayMicroseconds(2);
    digitalWrite(5,HIGH);
    delayMicroseconds(10);
    digitalWrite(5,LOW);
    long distance = pulseIn(3,HIGH);
    if(distance == 0)
        return(1000);
    distance = distance/58;
    return(distance);
}

void setup()
{
    pinMode(3,INPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}
```

```
void loop()
{
long distance;
    forward();
    distance = ultrasonic();
    if(distance < 10) {
        reverse();
        delay(1000);
        left();
        delay(700);
        stop();
    }
}
```

Obstacle Course Time

- Now for the fun part. Modify and expand the program to go through the obstacle course shown below. The large square represent 2 foot grids. The red rectangles represent a barrier that can be detected with the ultrasonic range sensor. Set up some barriers out of any solid material. Card board boxes, poster paper, or other large materials will work.
- Use the ultrasonic range sensor to avoid crashing into the barriers and turns the right direction every time a barrier is detected.
- Hint, use the collision avoidance program and expand it so that it will complete the maze. This requires the rover to back up and turn in specific directions at specific points of the maze.



IR Proximity Sensor for Collision Avoidance

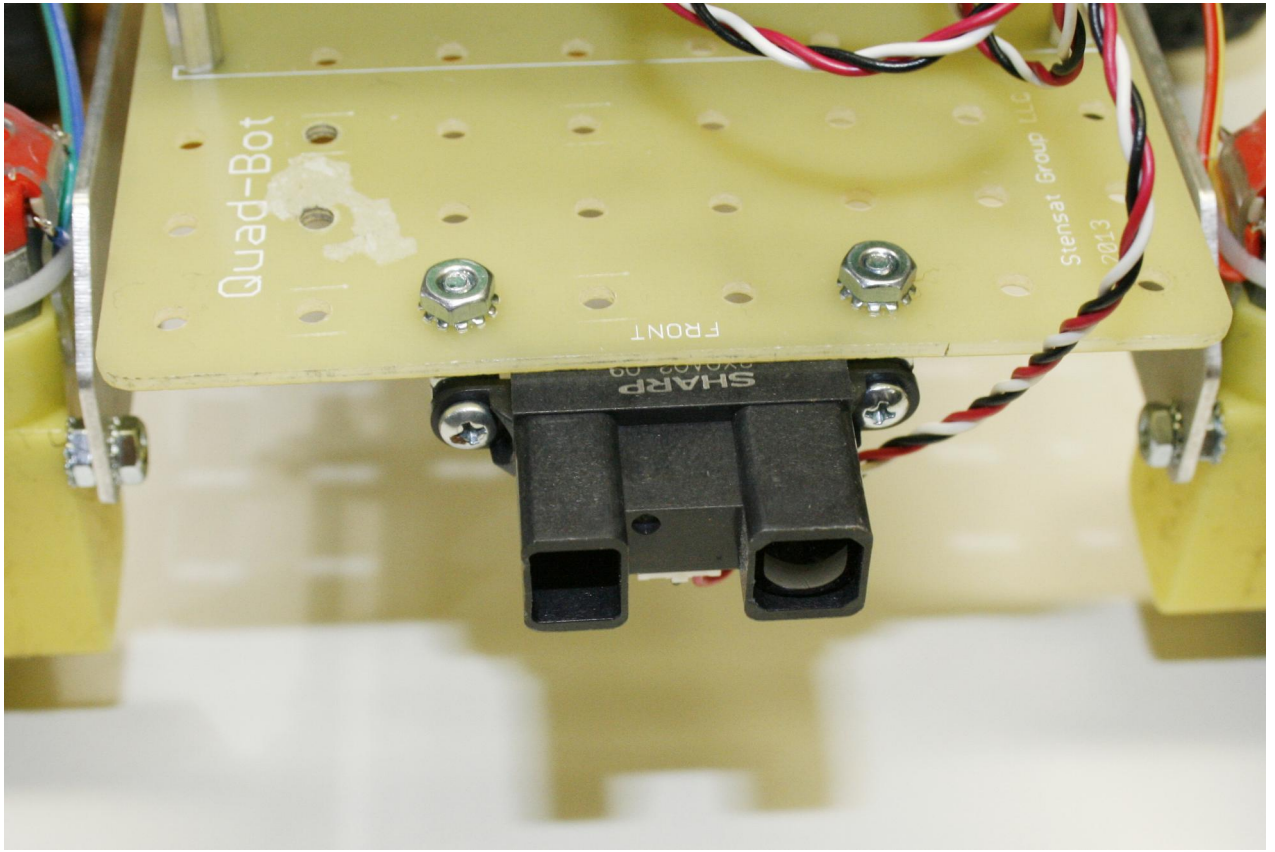
- The same collision avoidance will be attempted with the infrared proximity sensor. Review how the proximity sensor was used.

Installing the Infrared Proximity Sensor

- The bracket has two holes. One is larger than the other.
- Insert the 3/16 inch long screw into the mounting hole of the sensor and screw it into the small hole of the bracket. The small hole is threaded.
- Align the brackets as shown and tighten the screws until it is snug. Do not overtighten as that may break the plastic.

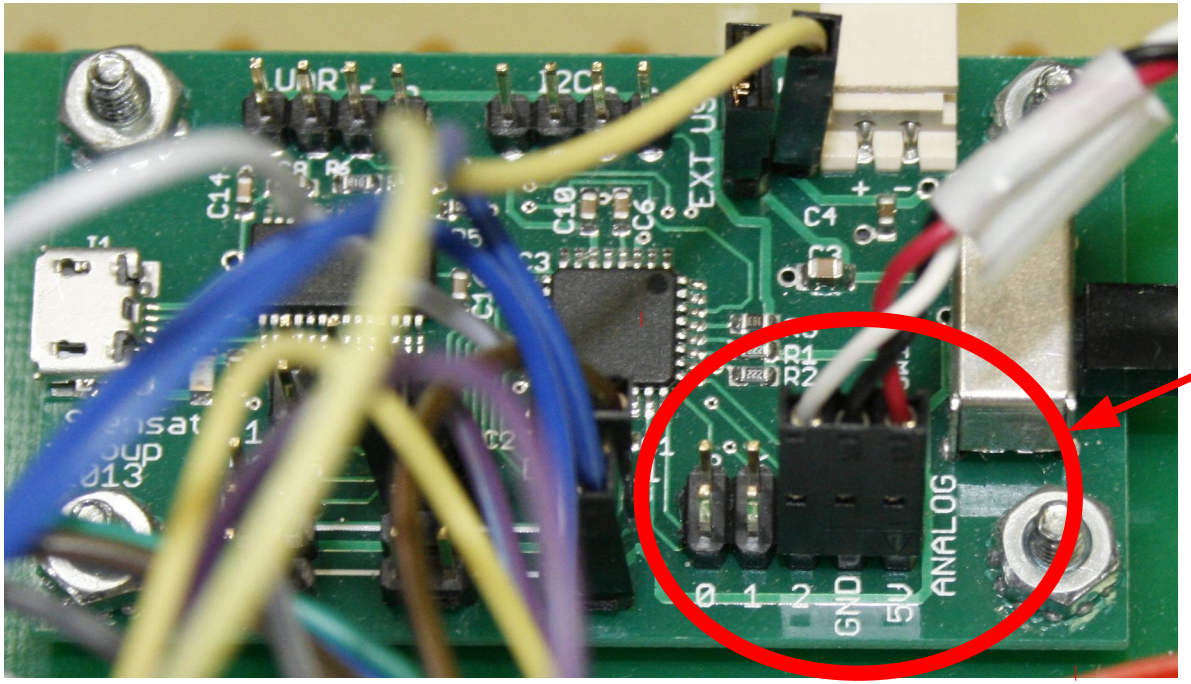
Installing the Infrared Proximity Sensor

- Attach the sensor to the rover as shown and use a pair of ¼ inch screws and nuts to secure in place as shown. Insert the screw from below the plate.



Installing the Infrared Proximity Sensor

- The sensor comes with a long cable and a connector at the end. The connector has three wires from the cable colored in the order of red, black and white. This allows the sensor to be plugged directly into the analog port connector.
- Orient the connector so that the red wire is connected to the pin marked 5V. The white wire will be aligned to the number 2 which is analog port 2.



IR Proximity Sensor Connector

IR Proximity Sensor for Collision Avoidance

- Write a program to perform the same collision avoidance operation.
- Replace the ultrasonic sensor code with the **`analogRead()`** for the infrared proximity sensor.
- Review the table on distance and measured value. Pick a distance for the rover to stop and review and insert the value in place of the underlined number.
- Attempt the same obstacle course with the infrared sensor.

```
void setup()
{
    pinMode(3,INPUT);
    pinMode(5,OUTPUT);
    pinMode(6,OUTPUT);
    pinMode(9,OUTPUT);
    pinMode(10,OUTPUT);
    pinMode(11,OUTPUT);
}

void loop()
{
int distance;
    forward();
    distance = analogRead(2);
    if(distance < 100) {
        reverse();
        delay(1000);
        left();
        delay(700);
        stop();
    }
}
```

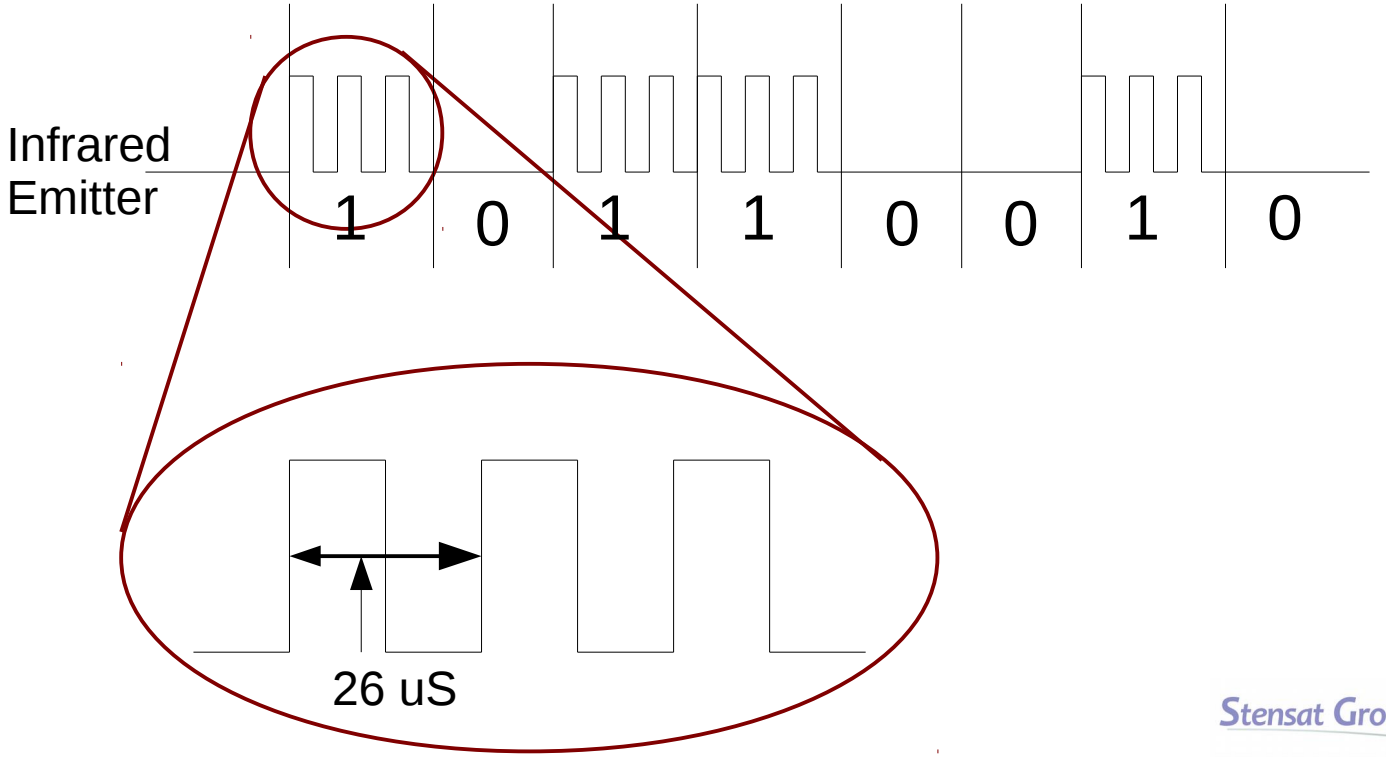
Remote Control

Infrared Remote Control

- Find a TV remote. Chances are very good that it uses an infrared LED to send signals to control the TV. The remote will be used to control the robot.
- First, the control codes need to be captured. The first program will do that.
- There are two ways remotes work. One way is for the remote to resend the code for the key pressed repeatedly until the key is released. The second is to send the code for the key pressed then send a code that equals all '1's.
- The next program will be used to capture the codes to be inserted into the robot program. The program decodes the key code and displays it in hexidecimal.
- First, remove the Ultrasonic sensor from the solderless bread board.

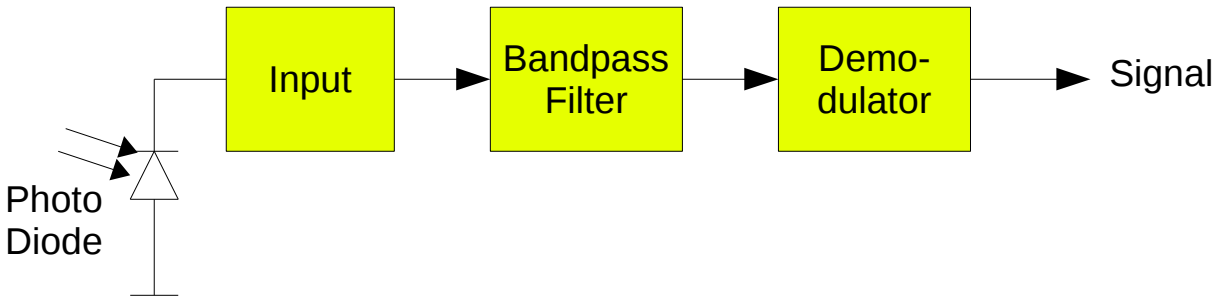
How IR Remotes Work

- The IR remote uses an LED that operates in the infrared range, specifically 990 nanometers. The emitter pulses the infrared light at 38 KHz, 38,000 times per second. The pulsed signal is then turned on and off at a lower rate so that bursts of 38 KHz light is transmitted. The LED is modulated to help the receiver detect the signal from other light in the room including sun light.
- The turning on and off of the modulated light is done in different sequences to generate different codes.



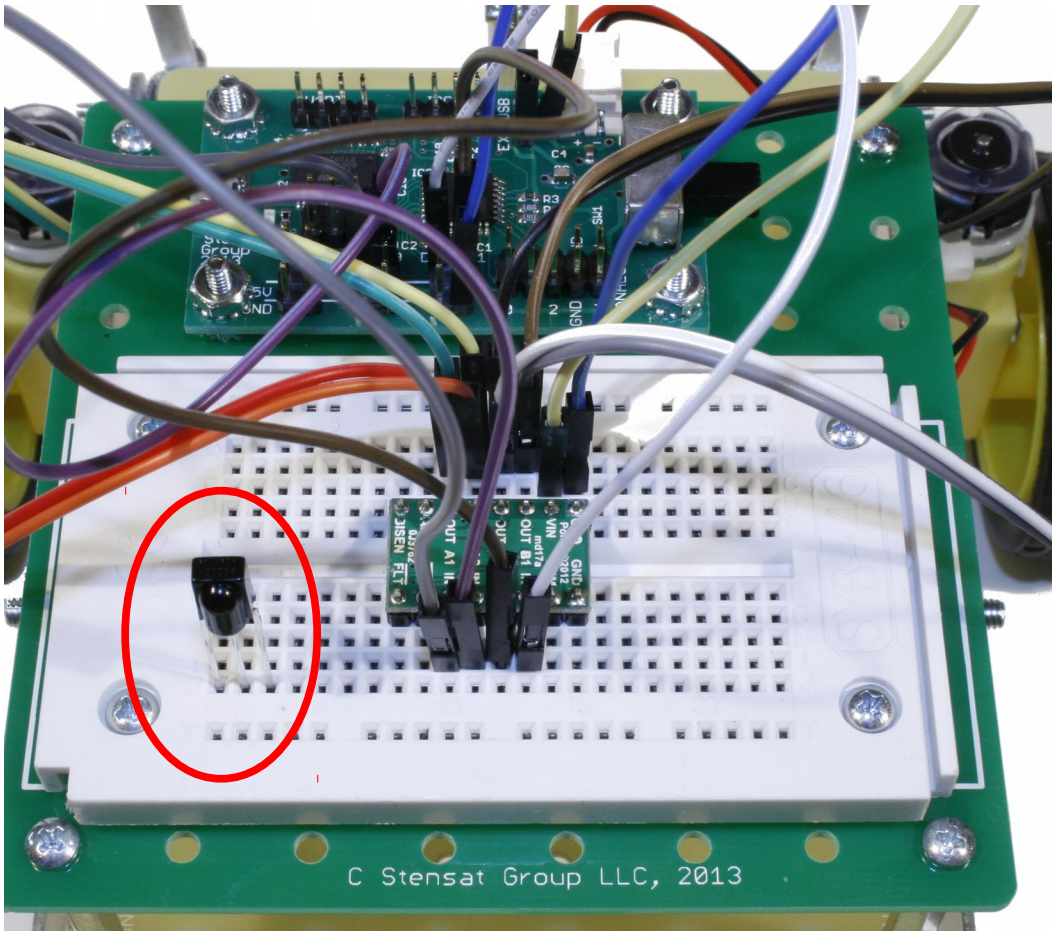
IR Receiver

- The infrared Receiver is a device that includes an IR detector and a circuit to detect IR signals modulated at 38 Khz.
- A photo diode is the sensor and connects to an input circuit that converts the current to a voltage.
- The signal goes through a bandpass filter. This is a filter that only lets a signal of a specific frequency to pass. All other signals cannot pass.
- The filtered signal is then sent to a demodulator that converts the modulated signal to the codes being sent.



IR Receiver

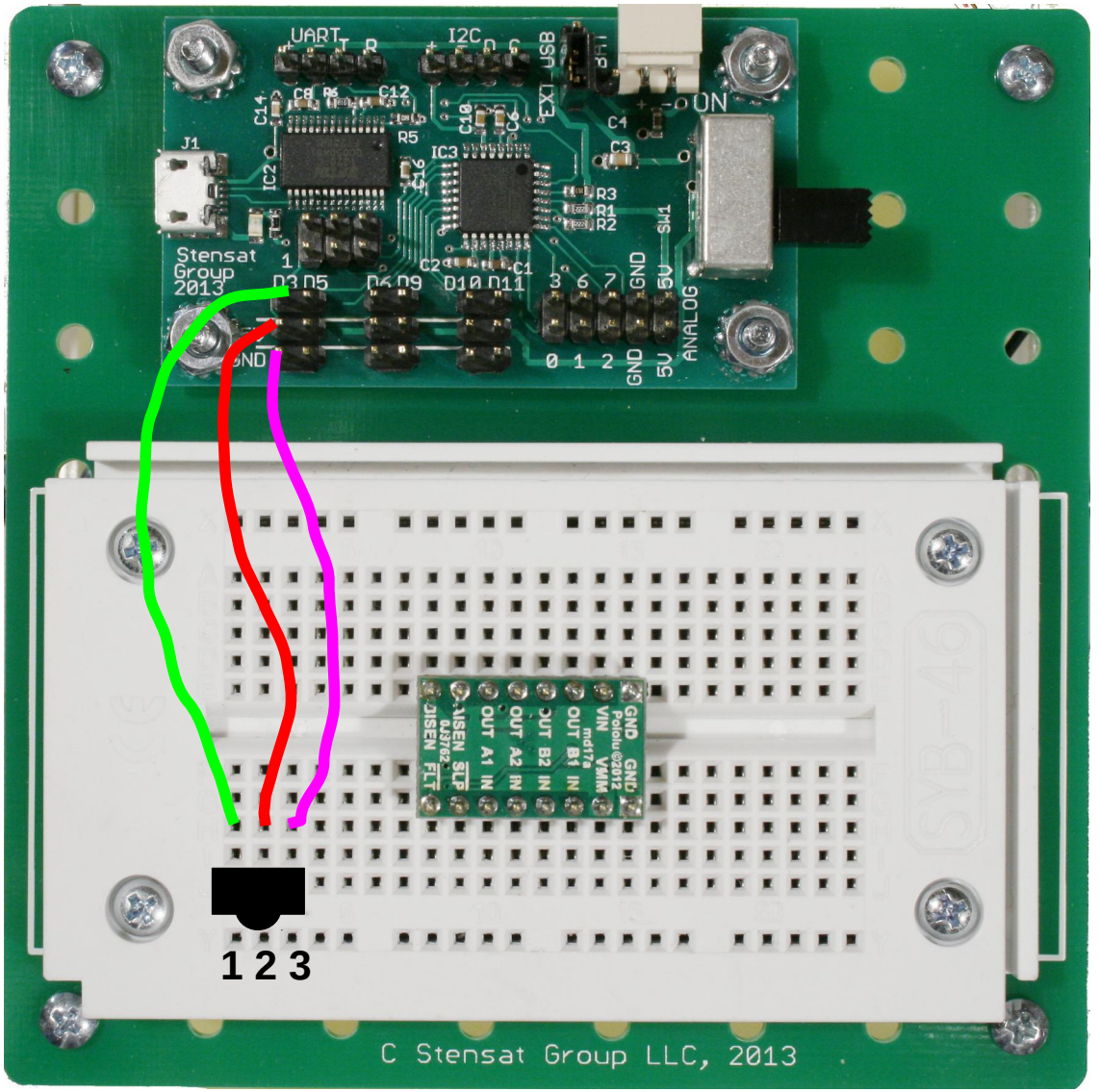
- Add the IR receiver to the solderless breadboard. Note the orientation as the connections need to be made in a specific order or the receiver may be damaged.
- Look at the picture to the right of the IR receiver. The leads on the IR receiver are numbered in a specific orientation.
- When mounting on the solderless bread board, have the rounded side face forward.



Pin 1 – OUT
Pin 2 – 5V
Pin 3 - GND

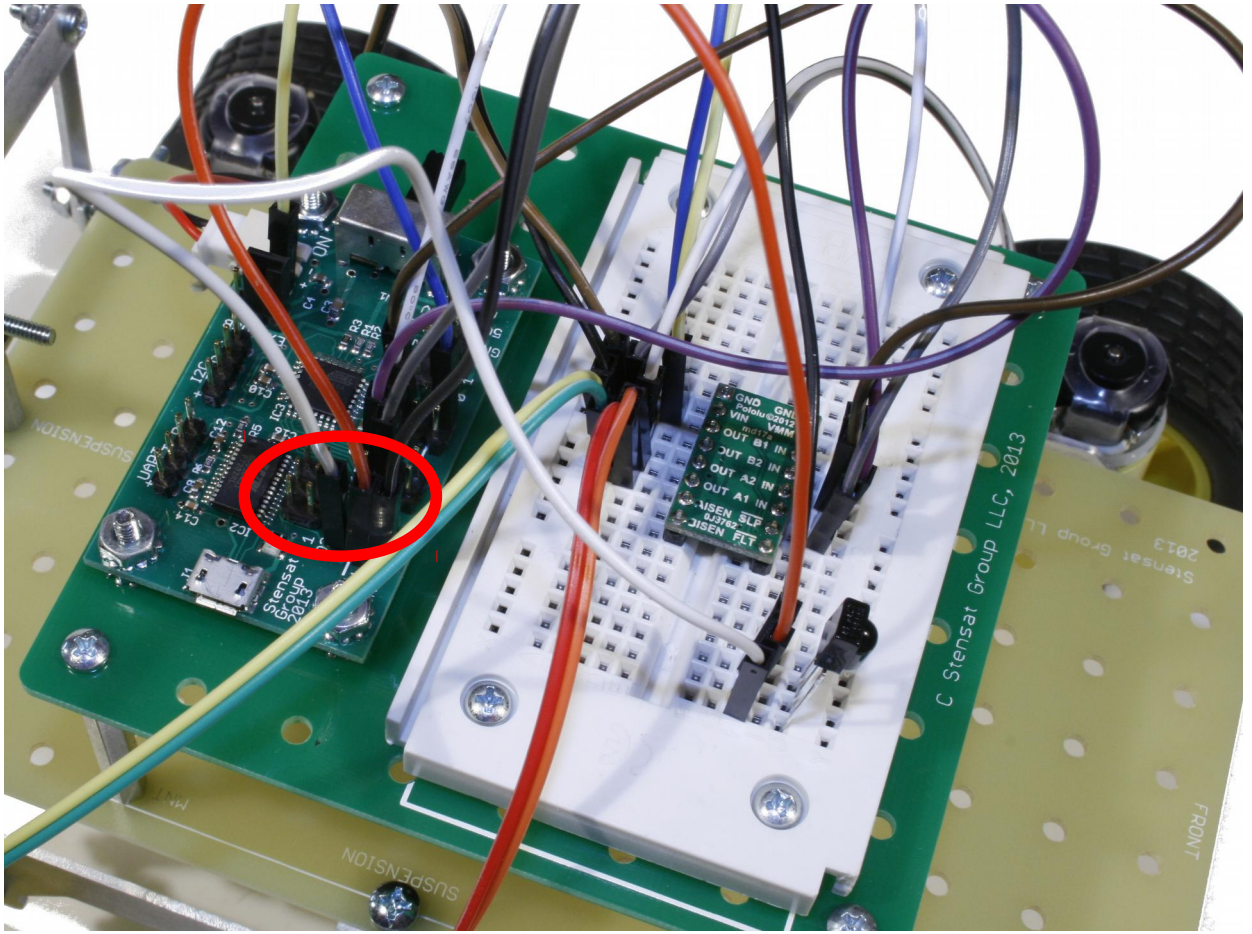
Wiring the IR Sensor

- Connect pin 3 of the IR sensor to GND under Digital pin D3.
- Connect pin 2 of the IR sensor to 5V under Digital pin D3.
- Connect pin 1 of the IR sensor to Digital pin D3.
- Be careful to not connect power backwards or the the IR sensor will be damaged.



IR Receiver Connections

- Connect pin 3 of the IR sensor to GND under Digital pin D3.
- Conenct pin 2 of the IR sensor to 5V under Digital pin D3.
- Connect pin 1 of the IR sensor to Digital pin D3.



Remote Code Capture Program

- In the Arduino software, select File then Examples. Look for Irremote and select IRrecvDemo.
- When the program is loaded, look for the line 'int RECV_PIN = 11;' and change the pin number to 3.
- Upload the program into the robot.
- Open the Serial Monitor.
- Now point the remote control at the IR receiver and press a button.
- Numbers should be displayed in the serial monitor.
- Remotes operate in one of two ways. One type of remote keeps repeating the code for the button pressed as long as the button is pressed. The other type will send the code once and then all logic level 1 after that.
- The codes are generally 32-bit numbers. The program displays the numbers in hexidecimal. There should be 8 digits. There are some remotes that will send smaller codes.

Capturing the Remote Codes

- With the program running, press the button for forward and record the result.
- Do the same for left, right, and reverse.

Forward _____

Left _____

Right _____

Reverse _____

Robot Control Program

- The next program will receive the IR commands and active the motors for the selected operation.
- The sequence is to receive a key code, compare it, then execute the proper code then go back and wait for the key code to be sent again.

IR Remote Code

- Start a new program.
- Under the 'Sketch' menu, select the 'IRremote' library.
- This will insert an include file statement at the top of the program. Two are inserted. Delete the one not shown to the right.
- Enter the program to the right and next page.
- Replace the underlined words with the appropriate code. Since the numbers are in hexidecimal, insert **0x** in front of the number.
- Make sure the move tab is included. Retype it in if it is not.

```
#include <IRremote.h>

unsigned long tt;
IRrecv irrecv(3);
decode_results results;

void setup()
{
  irrecv.enableIRIn();
  pinMode(6,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}
```

IR Remote Code (contintued)

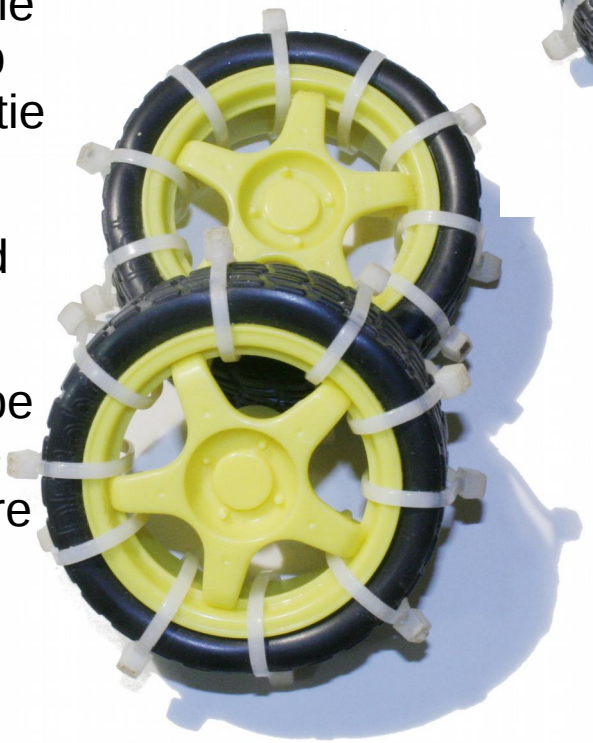
```
void loop()
{
  if(irrecv.decode(&results)) {
    if(results.value == FORWARD) {
      forward();
    } else if(results.value == LEFT) {
      left();
    } else if(results.value == RIGHT) {
      right();
    } else if(results.value == REVERSE) {
      reverse();
    }
    tt = 0;
    irrecv.resume();
  }
  tt++;
  if(tt > 3400) {          // increase value if robot stutters
    stop();
    tt = 0;
  }
}
```

Running the Robot with the Remote

- Make sure the code includes the movement functions from earlier in the lesson material. All those functions should be placed just above the setup() function.
- Now that the code is complete, test the program and verify the robot moves as expected.
- To get better angle, bend the IR receiver so the rounded side points upward. This will allow you to use the remote from wider angles.

Driving on Sand

- The suspension system allows the robot to operate over a variety of terrain.
- Sand is another surface that is a challenge to robots. Try driving the robot in sand and turn. Build a sand mound and try driving up the slope.
- Get a bunch of 4 inch tie wraps.
- To give the wheels more grip, secure tie wraps across the tires as shown. Loop the tie wraps between the spokes. 10 tie wraps per wheel is sufficient.
- Try driving through the sand again and see the difference.
- If the robot doesn't move well, it may be that the motors need more power. The motors are a bit weak. Adding two more AA cells in series will provide plenty of power.



Conclusion

- At this point, you should have a functioning rover that can drive through sand, gravel, and dirt. It can be controlled with a TV remote control and can be configured for collision avoidance.