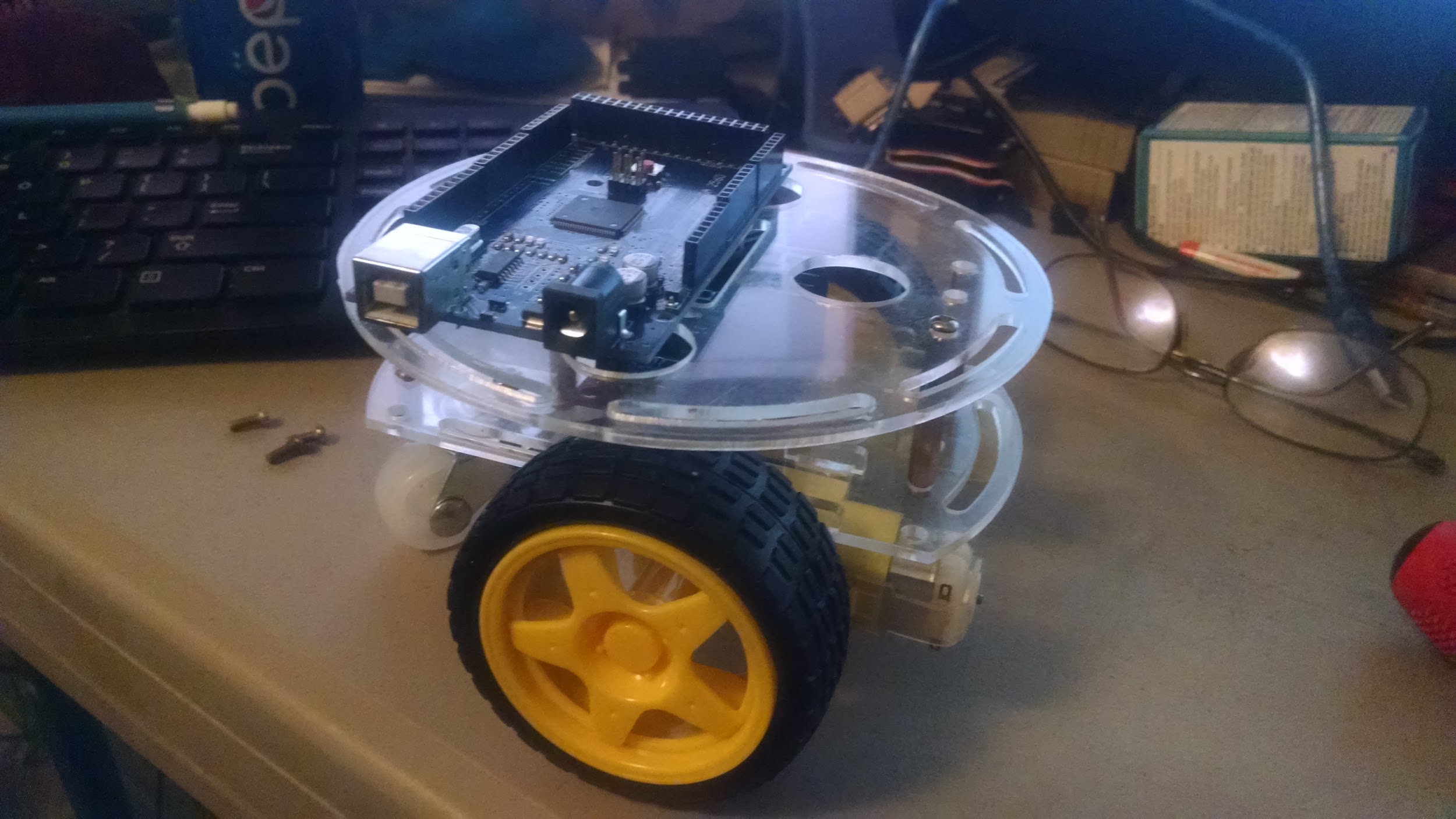
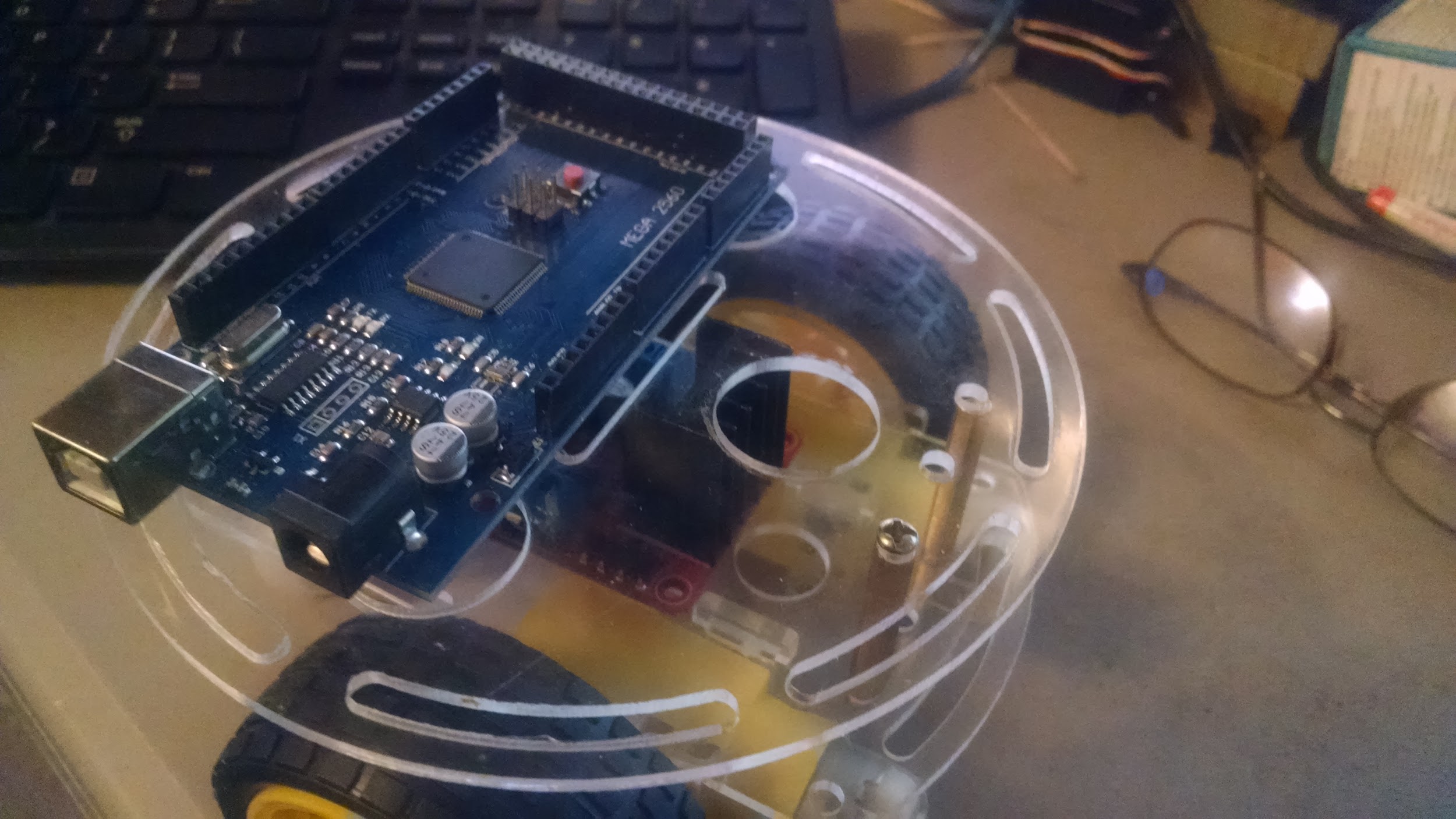
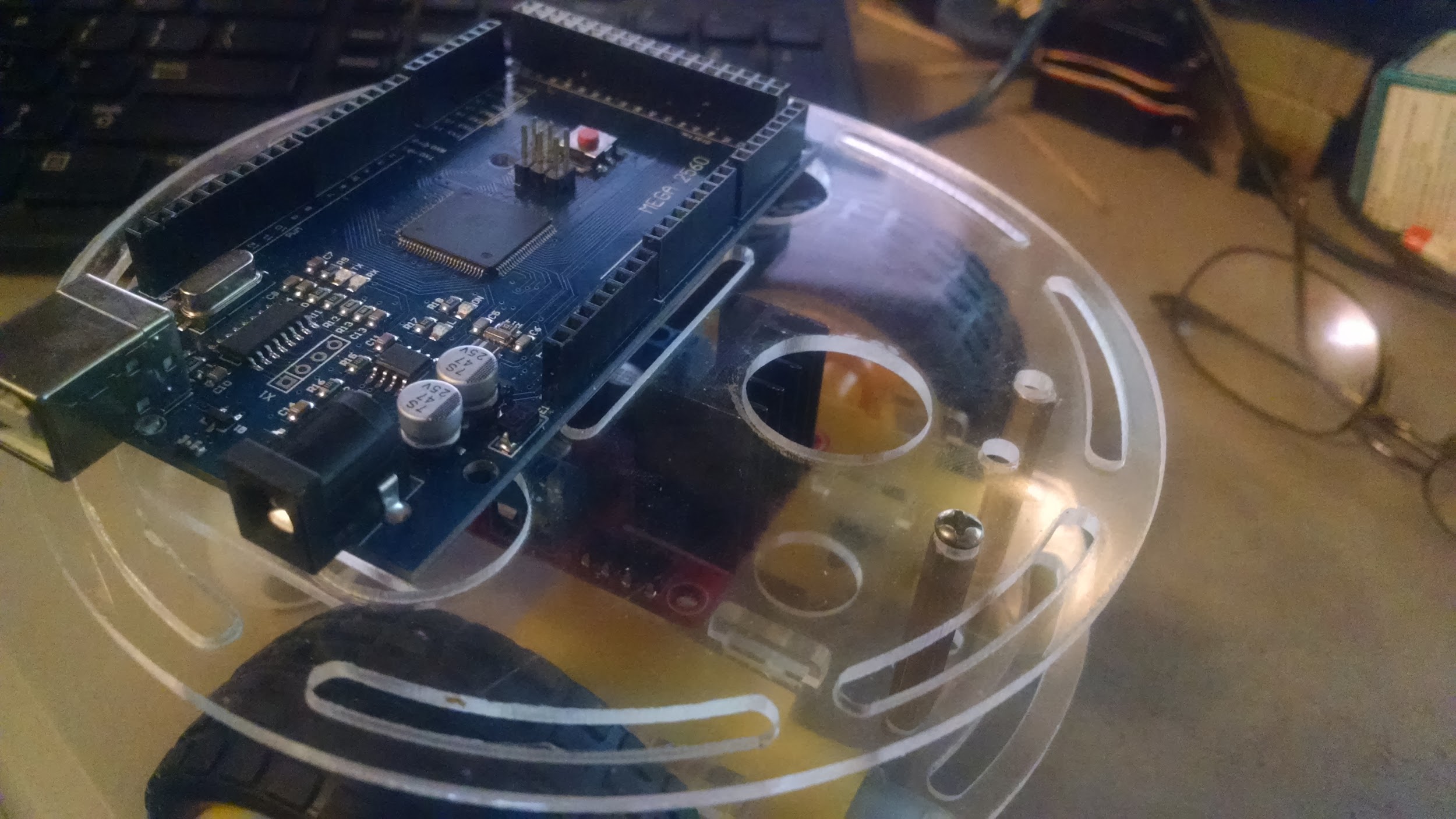
Sept 8, 2015 - 1st just a quick picture of the “alternative” chassis, with a microcontroller and driver board on them just to show size comparison.





And now:

Remember all code can now be found on github:

<https://github.com/automation-technology-club/Bitty_Bot>

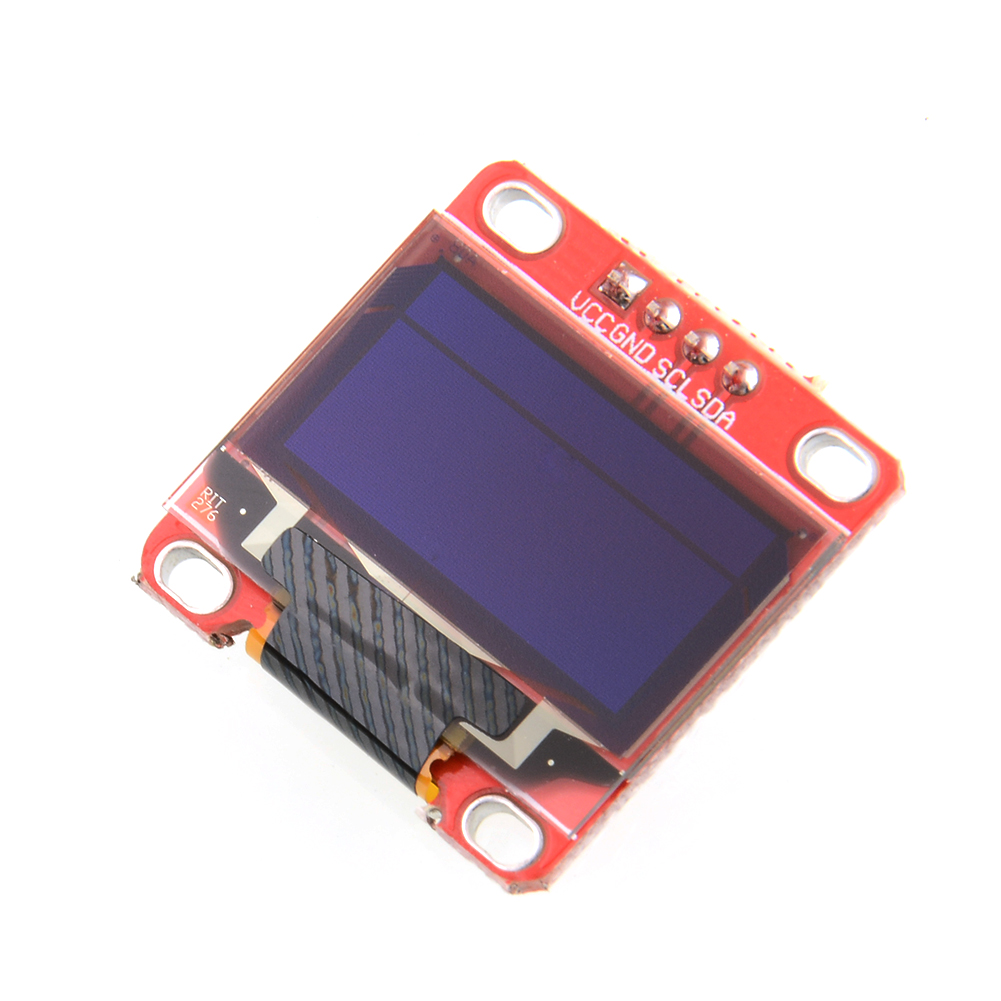
The Master Branch has been updated to include code for a voltage sensor. And a OLED screen.

(rest of the code will be updated to include these changes but may take a little time)

Voltage sensor I am using:

<http://www.ebay.com/itm/161758492882?_trksid=p2057872.m2749.l2649&ssPageName=STRK%3AMEBIDX%3AIT>

The OLED Screen (I don’t remember paying this much for them, this isn’t the one I’m using but I’m sure it’s the same thing, couldn’t find the one I have (?))

<http://www.ebay.com/itm/0-96-Serial-128X64-OLED-LCD-Yellow-and-Blue-LED-Display-Screen-for-Arduino-AVR-/121733393322?hash=item1c57e02baa>

Voltage Sensor is hooked up to the battery directly, and 5 volt from the Arduino, it’s also connected to Analog 1 Pin. The OLED is hooked up to 5v and I2C Pins, it uses the Adafruit libraries.

Ultrasonic demo code appears to be working correctly, and no changes were made this week.

IR Drop Sensor - the code appears to be working correctly, the repository will be updated from testing to stable soon.

I added 3 new Branches to github this week:

ESP8266WIFI, Bitty Bot IR drive test, and Library-testing

ESP8266WIFI has 2 modes,

1st it has a command mode - this lets a website send commands to the Bitty Bot

It’s down with the WIFI modules IP address, like this:

192.168.4.1/?drive=XX where XX is a number that tells the robot what to do.

10 - Tight Left Turn

11 - Tight Right Turn

12 - Left Turn

13 - Right Turn

14 - Backward Left

15 - Backward Right

16 - Forward

17 - Backward

18 - All Stop

In this mode the Bitty Bot turns into an AP (access point) but does not serve a webpage. The Webpage must be hosted somewhere else, and you need to be connected to the Bitty Bot AP

2nd mode is a status type mode, - in this mode the Bitty Bot sets up as an AP, and serves a webpage that shows the status of the battery (yes, I only did the battery for now) As more sensors get added, other status updates can be placed on the website.

The IP address for both modes is 192.168.4.1

The ESP8266 is a 3.3v device, it is also a serial device, which makes it easy to talk to it.

I am using the Arduino Mega Serial 1. Because it is a 3v device, and requires a little extra power, I made a carrier board for it that has a 3.3v regulator, and a 2 channel level shifter (the level shift lets 3v devices “safely” talk to 5v devices)

The Code for the Arduino/ESP8266 is based on code found here:

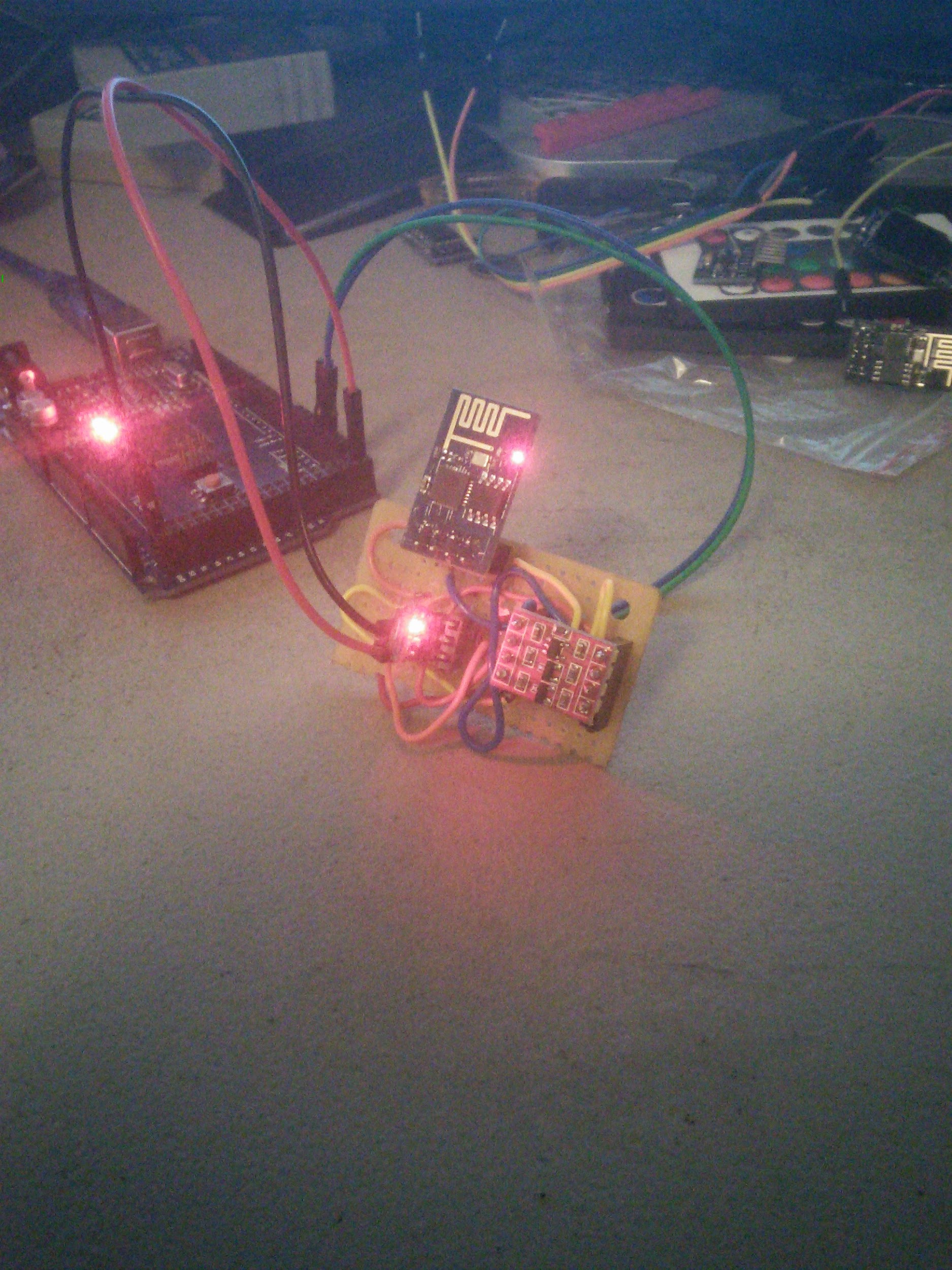
<http://allaboutee.com/2014/12/30/esp8266-and-arduino-webserver/>

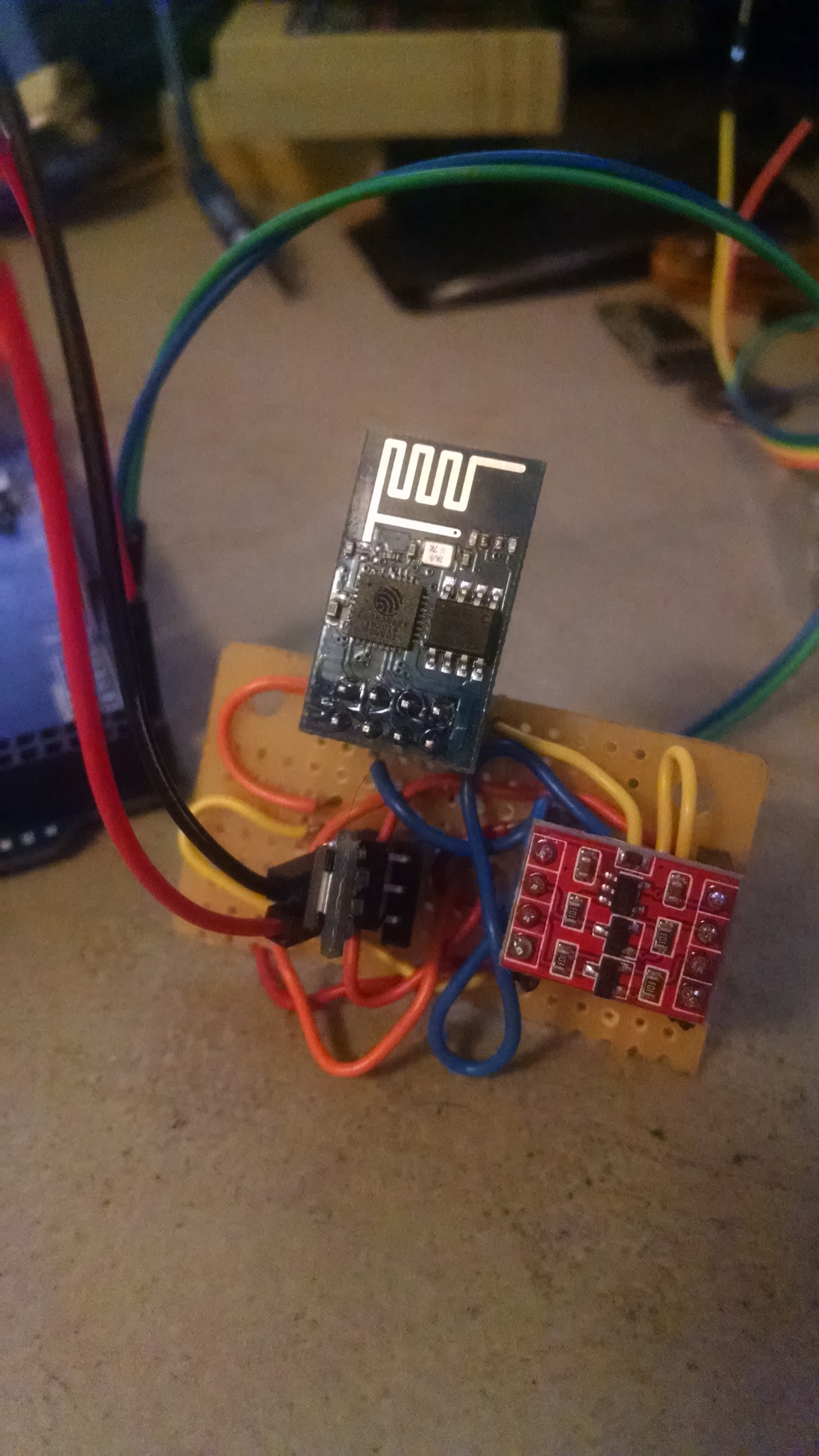
<http://allaboutee.com/2015/01/02/esp8266-arduino-led-control-from-webpage/>

The Android App is very loosely based on this (the idea anyway, not the code).

<http://allaboutee.com/2015/01/20/esp8266-android-application-for-arduino-pin-control/>

The Android app was made using MIT App Inventor - It’s a simple app, that probably could stand to be made a little prettier.





Bitty Bot IR Drive Control:

For this I used a simple IR receiving unit



The one I used came from this kit, but I didn’t use the remote, as it’s not that great of a remote (really cheap). This whole “kit” cost right at $1.50

<http://www.ebay.com/itm/251982913002>

The IR receiver is connected to 5v, Ground, and PIN 24 on the Mega.

I decided to use this remote to control it (something I picked up from Radio Shack when they were closing stores, but they are pretty reasonable on eBay, around $5 or $6 bucks.)

<http://www.ebay.com/itm/MAKE-IT-ROBOTICS-REMOTE-CONTROL-/281770353941?hash=item419ad26515>



So, each button the remote has a “code” that is just for it, some buttons (on some remotes) may be multifunctional and have multiple codes.

You need to know what the codes for each button are, this sketch can help with that:

<https://codebender.cc/sketch:148841>

Pretty much what you do is hook up the receiver, open the serial monitor, and hit buttons on your remote, it should be noted, that you probably should hit each button more than once (10 or 15 times) just to make sure it always gives the same code - or better than 90% of the time.

Some of these codes can be quite long, so converting them to HEX is probably easier to deal with them when it comes time to use them.

The easiest way to deal with all the codes is to setup a switch/case routine in your sketch.

So for now, my IR remote only does 5 things, I have the codes for the other 3 buttons, but don’t know what to make those do yet. I also know that this remote has additional codes if more than one button is pressed at a time. I don't have those codes.

I have Forward, Backward, Left Tight turn, Right Tight turn, and all stop

Lastly but not least: I attempted to make a Library.

This is my very 1st time making a library, it was not as bad as I thought it would be, but it wasn’t that easy either. Lots of little things needed to be done to make it work.

The Library has all of the current drive functions in it, with the exception of

circleleft()

circleright()

You use this like you would use most libraries, there were some changes that had to happen, but for the most part this is it:

#include "BittyBot.h"

create a BittyBot object -

BittyBot BittyBot(PWMLPin, PWMRPin, L1Pin, L2Pin, L3Pin, L4Pin);

BittyBot.lefttight(speed); //turn in place to the left BittyBot.righttight(speed); //turn in place to the right BittyBot.left(speed); //turn left going forward BittyBot.leftb(speed); //turn left going backward BittyBot.allstop(); //stop all motors and pwm, small delay for everything to catch up BittyBot.right(speed); //turn right going forward BittyBot.rightb(speed); //turn right going backward BittyBot.forward(speed); //move the robot forward BittyBot.backward(speed); // move the robot backward

The biggest difference is you now have to send the speed parameter with each call.

I’ve done very limited testing for the library I think it will work, but use it your own risk.

So What is in the works for my bitty bot:

Better Power Control (Better Batteries might just fix that problem)

The WIFI code still needs some work done to it, mostly I want to combine both ideas into one complete package. This will be a work in progress.

IR Line Follower, I want to get this done sometime this week before Thrs, would like to demo this along side Franks line follower.

Must DO something that will correct for Driving forward (mine tends to drift in one direction if it’s goes for a long period of time) - Correcting for this drift is a MUST DO.

Accelerometer

Ideas:

Sound Sensor - If it heres a sound it backs up and goes the other direction - if it hits something it backs up and goes the other direction.

Find more to do with the Buzzer

Using a Keypad matrix - program the robot and let it go.

Work in progress (still): (All of it really, but….)

Ultrasonic with Servo