***Ed’s First Robot***

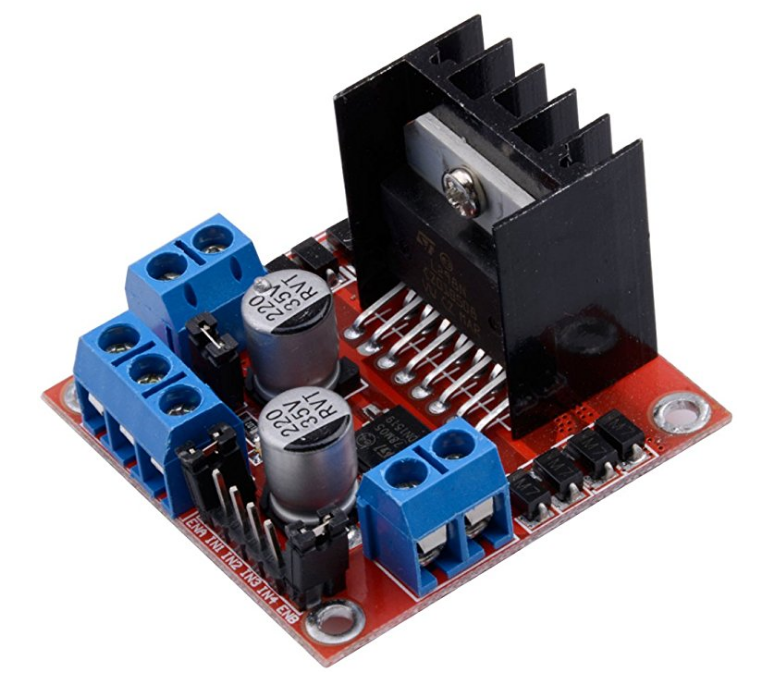
I picked up an Actobotics, Runt Rovers “Whippersnapper” 4WD robot kit.

The kit basically snaps together in a few minutes and includes a bracket to mount an Arduino.

I also picked a couple of dual motor controllers - thinking I would need to control all four motors.







Motor Controller

I wired-up and tested the motor controller following the directions at **tronixstuff**:

<http://tronixstuff.com/2014/11/25/tutorial-l298n-dual-motor-controller-modules-and-arduino/>

Next, I connected both left wheels to one set of outputs and both right wheels to the other set.

Worked fine, I only needed the one controller for this project.

Collision Avoidance

After running into several chair legs and walls I added collision avoidance.

I had one “forward” ultrasonic sensor, and plan to add another later for driving in reverse.

The best solution I found for integrating this was at <http://forum.arduino.cc/index.php?topic=106043.0>

The “NewPing” Library Project is awesome, but it seems the latest libraries were posted prematurely.

Rolling back an Arduino library is a bit convoluted, so I just downloaded the previous release and cut and pasted the version 8 library source, header and examples over the version 9.

Wow – the test code displays the distance in centimeters to any object within range!

*(When coding for this, note that if an object is not in range, the code returns “0” cm.)*





Power

Both the Arduino and the motor controller allow for a range of voltages.

The controller provides two motor supply options. Leave the power jumper in place to supply 5-12 volts for the motors. Remove the jumper to connect 12-35v for the motors. The controller also provides a regulated 5v out, to power the Arduino board. The wheel motor spec suggests 4.5v and suggests (4) “AA” batteries but does not employ a motor controller. I happened to have a couple of 11.1v 500mAh LiPo batteries, the size of a 9v battery. The measured voltage to the motors was 5.6v. I had planned to run the 11.1v thru a 7805 voltage regulator for a 5v output and a diode to protect the boards from reverse polarity accidents. While I still plan to use a diode, the regulator is not necessary – and would probably not power the motors with the voltage drop of the controller!

Code Development

The first step was to cut and paste both the motor controller and ultrasonic sensor code into one script. Then modularize and parameterize it so a control script would be easy to create and maintain. I first created basic movement scripts: driveForward, driveReverse, spinLeft, spinRight.

These routines all accept two parameters: duration and speed. The duration is entered in milliseconds and tells the routine how long to execute. The speed param allows 0-255 with zero being off and 255 being full speed. My left wheels are weaker than the right, resulting in a slow left turn during forward motion and less of a deviation in reverse. I tweaked the forward and reverse sub-routine duration loops accordingly.

After that I added an varySpeed routine that accepts a starting speed and ending speed. This routine accelerates or decelerates the robot. This was added to both the forward and reverse routines.

A subroutine was then created for the ultrasonic sensor called “pingDelay”. A Delay function exists in the Arduino libraries that waits for the duration time, entered in milliseconds. The motor functions all set motor controller params, activate the motors then call a Delay for the specified number of milliseconds. For collision avoidance, a time loop that also incorporated checking for objects was needed. I replaced the Delay call with the pingDelay and hard-coded a trigger distance. I set this to 50 cm based on the stopping time required at full speed. With the sensor feed it would be easy to determine whether the robot was closing in on an object or an object was pulling away. This could be useful if multiple robots are employed.

The second ultrasonic sensor was added for running in reverse. To conserve and I/O pin, I tied the ping lines of both the ultrasonic cards together. This sends a ping out from both the front and rear but only listens for the echo in the travelling direction.

I ordered some line sensors thru Amazon. Those and a few switches - will allow me to complete the project.





The Project

Here’s the idea. I want a line-following robot to navigate a path in either direction.

There will be two push-buttons on the front and rear of the robot. One will be toward the left side, the other to the right. When the left button is pushed, the robot will move in the opposite direction and follow all left turns on its adventure – if it does not “see” a signal prior to an intersection. The same with the right button, taking all right turns. Here’s the proposed path for the robot to follow:



The loading point is down the aisle to our right, off-screen. The unloading point is straight ahead, at the end of the aisle. The function of the left branch is to rotate the robot 180 degrees for loads that need to be off-loaded, at the unload point, facing a specific side of the warehouse:

After placing a load onto the robot at the loading point (to our right), if the load is facing left and needs to be reversed and face right (or vice-versa) at the unload point, kick the left button. It will follow the left path, in this case choosing the straight line to the left rather than turning right. The markings at the end of the left line tell the robot to reverse. The robot reverses, moving from the far left towards the center, then again chooses the left path and proceeds to the unload point – which is marked to stop the robot. The robot can be sent back by kicking either button. Here, if desired, the right button will reverse the robot direction by taking all right turns. Which button to use should be intuitive just by looking ahead at the path. Kick the left button to send the robot directly to the loading point or the right to turn it around before going for another load.

At the loading point, with a load that is facing the correct direction for unloading - or if no directionality is involved - you would just hit the right button to send the robot directly to the unload point.

Multiple robots could be employed and staged at the loading point using the collision avoidance sensors.

Next Steps

*(aside from building a second unit, painting them blue, moving the Arduino and battery inside the frame - and making small wooden pallets!)*

Circular patterns are often employed, and directions can be controlled using combinations of left and right signals. Marks could be added to create an unload point at each aisle. At the loading point a destination aisle could be preset with a push-button and a display. The robot could just count the marks and stop at the appropriate one. Maybe set the end of the line to reverse the robot for returning to the loading point.

In that scenario you could send Robot1 from the loading point to the first aisle. Load and send Robot2 to aisle 5. When Robot1 is empty, send it home by sending it to the end of the line. Robot2 moves with it as far as aisle5 and Robot2 gets unloaded. If Robot1 returns before Robot2 is finished, it waits. Just send Robot2 towards the unload point and both return.

Anything is doable and communicating between the robots could be done with Bluetooth, a robot with a load has higher priority… Doing a short video with models is more fun than using a simulator and less costly than crashing robots but both methods could be employed to demonstrate ideas to customers.

Parts

$24 Actobotics, Runt Rovers “Whippersnapper” 4WD robot kit

$10 Arduino Uno R3

$10 OSEPP MTD-1 Motor Driver for controlling two DC motors

$12 (2) HC-SR05 Ultrasonic Sensor

$11 (3, or 6) 3-Line Sensors (pkg of 10!)

$$$ LiPo Batts / charger

My robot script:

This is the script I used to test the sensor scripts. Test video at <https://youtu.be/x2TURrRXOxw>

delay(10000); // gives me 10 seconds to position robot

driveForward(4000, 100); // drive forward for 4 seconds; 100/255 speed

// unless it sees an object within 50cm

delay(1000); // wait a second

spinRight(750, 150); // left wheels forward; right wheels reverse

// ¾ second, at 250/255 speed

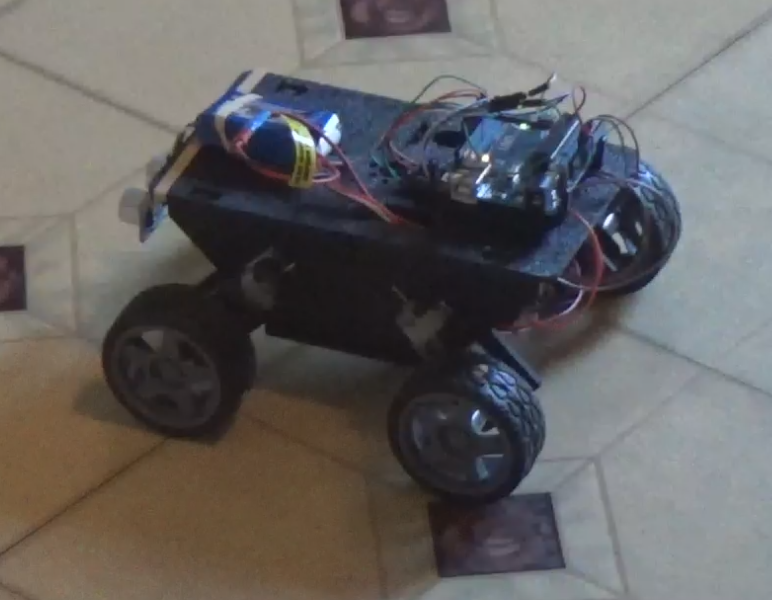
delay(500); // wait half a second

spinRight(750, 150); // left wheels reverse; right wheels forward

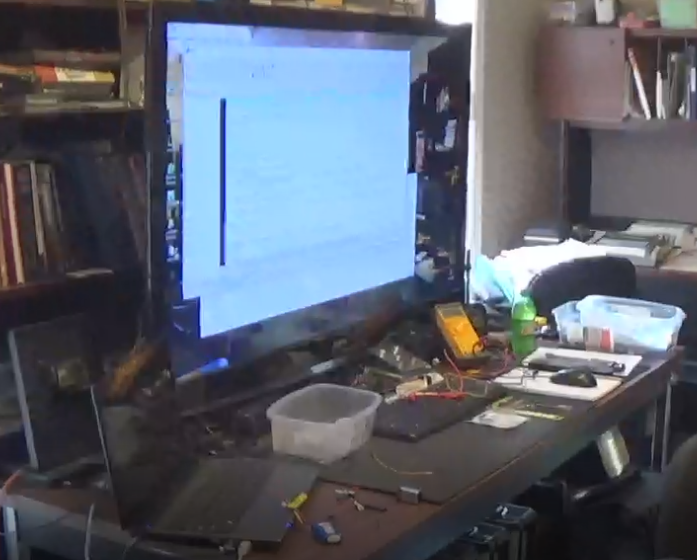
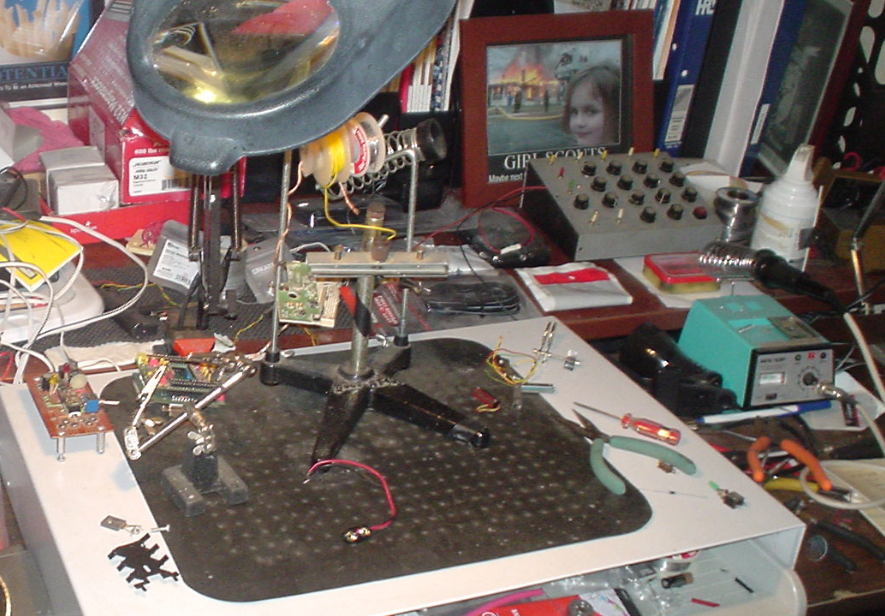
// ¾ second, at 250/255 speed

delay(1000); // wait a second

driveReverse(2000, 100); // drive backward for 2 seconds; 100/255 speed



“Robot” Lab

Laptop Dock & Robot Lab Soldering Station