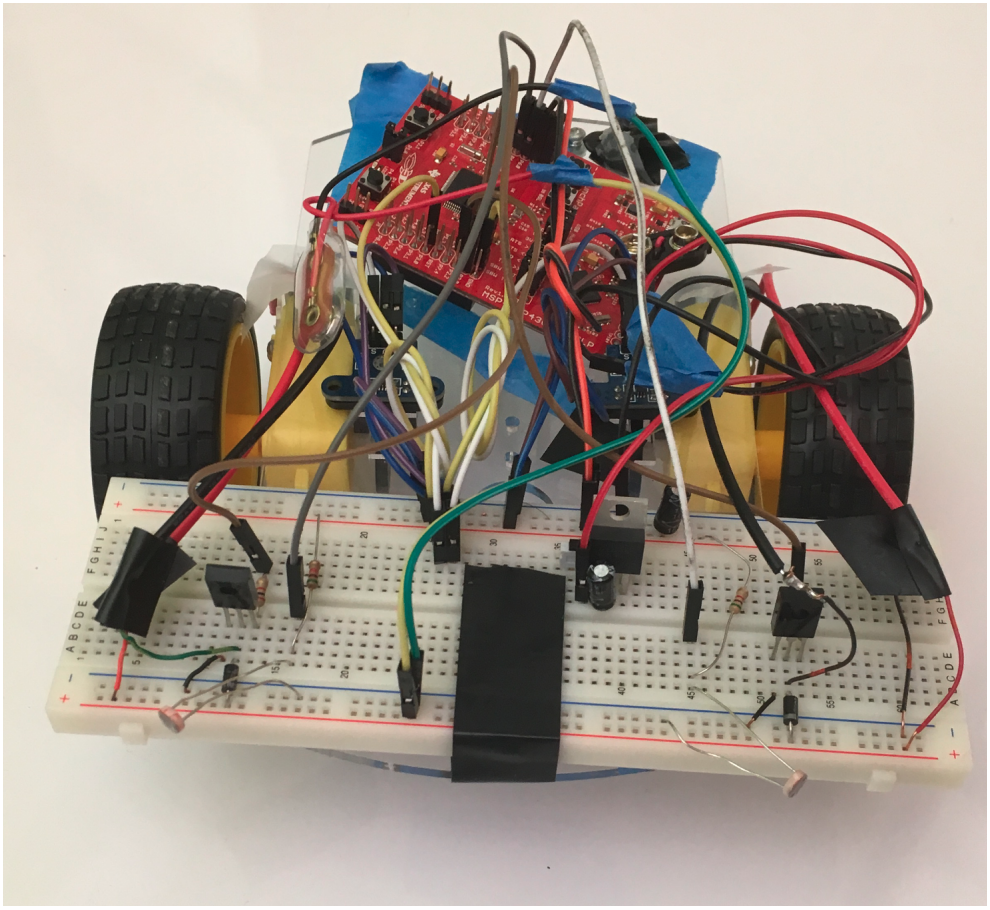
Light Seeking Robots!

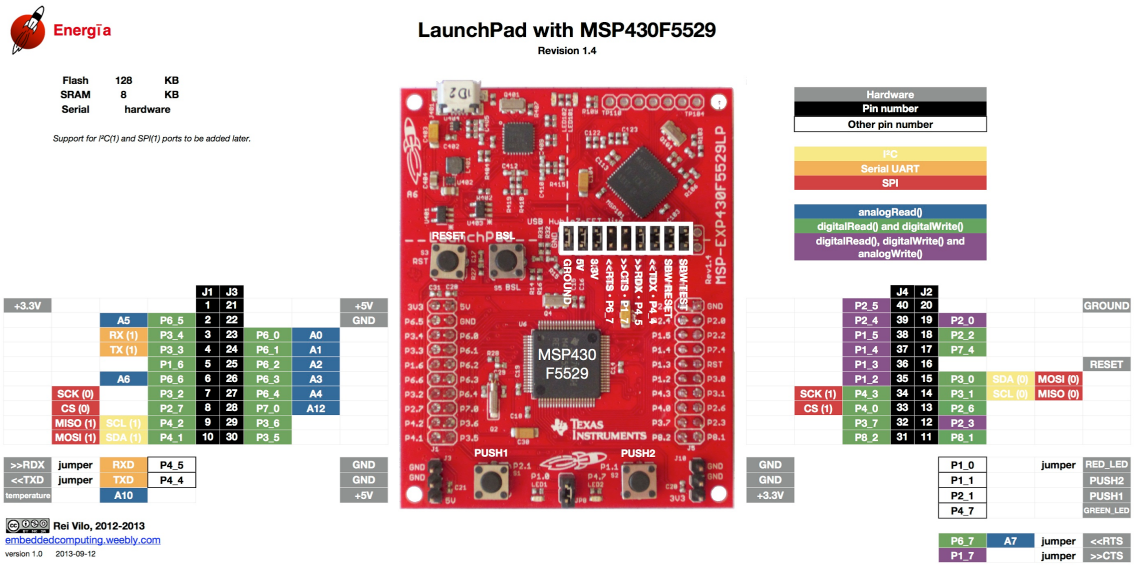


The Challenge:
We will be building robots that seek light. Each robot will begin facing away from the light source, and must guide itself to the light source. These robots will be autonomous, so human interaction is not allowed. Once started, the robot's must act on their own. The goal of this challenge is to learn more about embedded code, controls, and some aspects of circuits. Some programming experience is recommended, but not required. Teams can be of any size.

Resources

Along with your mentors, below is a list of some resources:

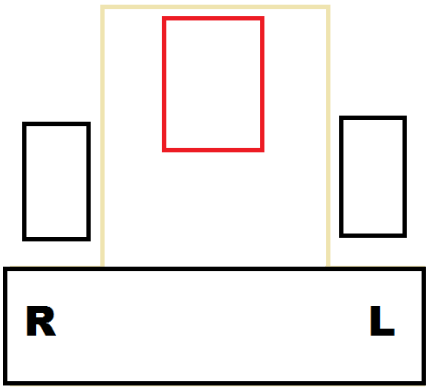
Energia Website: http://energia.nu/reference/



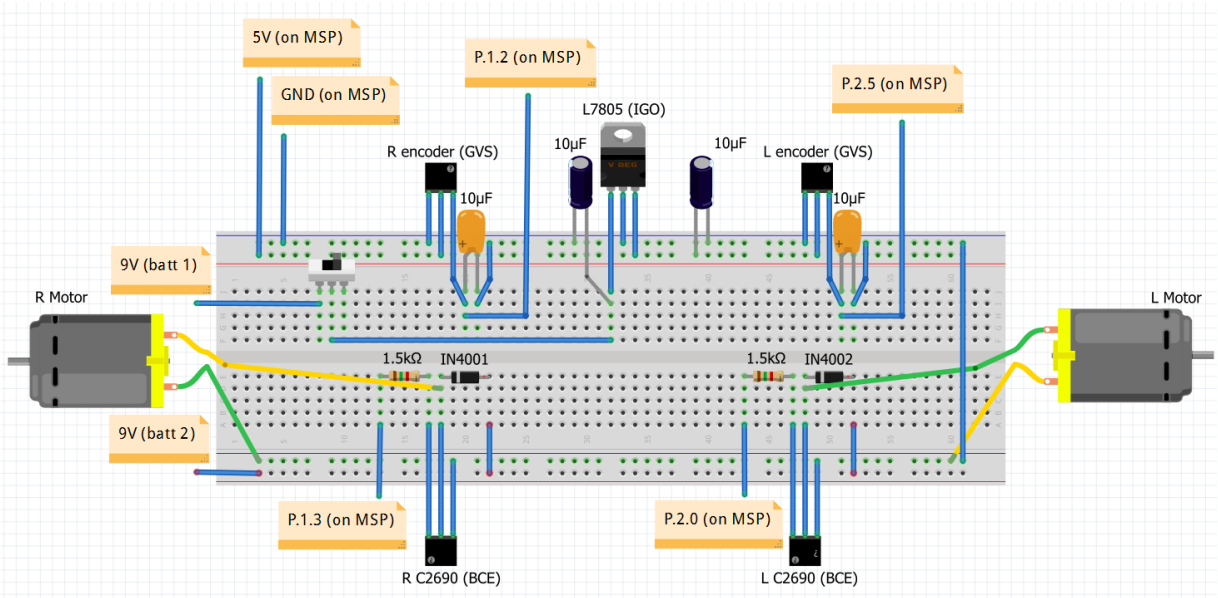
The Given Driving Circuit (for if you unplug something by accident):

Parts List:

- 1 switch
- 2 1.5kΩ resistors
- 4 10μF capacitors
- 2 IN4001 diodes
- 1 L7805 voltage regulator
- 2 C2690 transistors
- 2 9V batteries (- terminal connected to ground [not shown])
- 2 encoders
- 2 motors
- 1 MSP430 TI Launchpad microcontroller



* NOTE: we've noted the terminal symbols/ names for the regulator, transistors, and encoders from left to right in the picture below. *



My first Energia Program:

If you are an absolute beginner and would like to start with a small simple example of how programming an MSP430 works, we recommend you work through the tutorial given below: http://energia.nu/guide/tutorial_blink/

Moving around:

This program demonstrates how to move the robot. In order to move the robot, we write PWM signals to the motor terminals. A PWM signal turns the motor on and off very fast, and by adjusting how long the motor is on compared to off, we can adjust the speed of the motors. Learn more about PWM here: https://www.arduino.cc/en/Tutorial/PWM.

The code provides you with a function called drive(). Feel free to add this function to your own code. The format of the function is *drive(leftSpeed, rightSpeed);* leftSpeed and rightSpeed must be *integers* between 0 and 255.

Reading Analog Inputs (File->Examples->03.Analog->AnalogInput):

This program demonstrates how to read the input of a sensor. We will read voltage from the photocell, using the command analogRead(*sensorpin*). This returns us an integer between 0 and 4095. Run the code to see the output of the sensor on your computer. You will need to open the serial monitor to see the output of the MSP. Select tools->Serial Monitor in the Energia window.

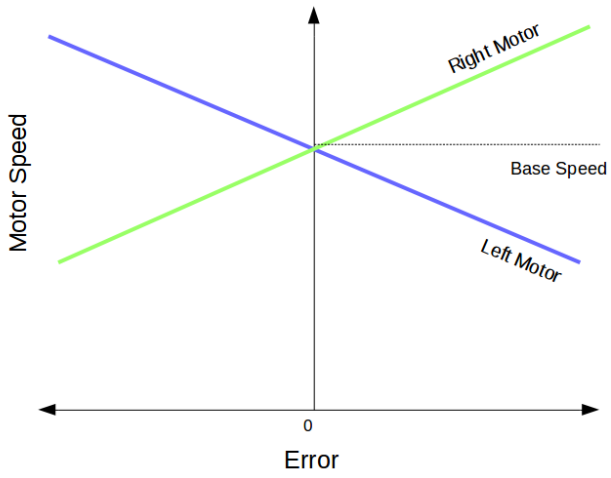
Driving Straight/ Basic Controls

The goal of this tutorial is to show you how to drive the robot straight with controls. Feedback controls is very large and complicated topic and the subject of many PhDs. However, the simplest form of feedback is just proportional feedback, and in many cases, this is adequate. This program shows you how to use proportional feedback to make your robot drive straight. It will be up to you to figure out how to use proportional feedback to make your robot find the light.

What is feedback acontrols? Let's say you wanted to drive your robot straight. You could set the motor speeds to be equal. Unfortunately, this does not mean your robot will actually drive in a straight line. In order to make sure we actually drive in a straight line, we need to try to measure how much each motor has moved, and change their speeds so they move an equal amount.

To measure how much each motor has moved, we use an encoder that counts the number of 30 degree movements of each motor. We will refer to this as a tick. For example, if a wheel spun 360 degrees, the encoder would count 12 ticks. The code provided keeps track of the number of ticks of both motors. If the robot were driving straight, the number of ticks on both motors would be relatively equal. We can also take the difference between the values to get a measure of how far from straight we are. We will call this the *error*.

How do we use this error? Let's first consider the binary case. Suppose error was the difference between the left ticks and right ticks. If error is positive, we know left ticks > right ticks, or the left motor has moved too much. Thus we need to slow down the left motor, and speed up the right motor. By how much do we speed up or slow down? We probably should scale it by the error. A large error means we're really far from being straight, and thus we need to make a large change in speed to fix this error. The plot below visualizes this:



What is base speed? Notice that when error equals zero, we want the robot to drive be driving. So we will set the motor to drive at equal speeds at some base speed. What about the slopes? The slopes of the lines are actually a parameter you will have to set and play around with. A greater slope means a small error will create a greater difference in speed, but this could overshoot, where we overcompensate for the error. Hopefully, this gave you a basic understanding of feedback control, feel free to ask questions if you are confused. Run the code provided to have you robot drive straight!