# OPEN-ROBOT Experiments

## Introduction:

This document will explain several experiments that can be executed using OPEN-ROBOT. "Out of the box" experiments will be covered first. After that we discuss experiments that are possible when additional hardware is purchased and added to OPEN-ROBOT.

## Obstacle Avoidance:

OPEN-ROBOT can avoid various environmental obstacles by using the two frontward facing GP2D120 infrared range sensors. A built-in "Wander" mode is preprogrammed into OPEN-ROBOT. However, you can create your own custom obstacle avoidance program using a high-level programming language that supports TCP/IP Socket communication. We recommend using Microsoft's® freely available C# Express since we've created a class library that encapsulates all of OPEN-ROBOT's functionality. Of course you can choose to use Java, C, C++, Python, or any other suitable programming language. The 8-bit measured output of each GP2D120 sensor can be correlated to an obstacle detection distance in inches, centimeters or another unit of length.

## Light Tracking/Avoidance:

Light tracking or avoidance can be accomplished by using the two frontward facing cadmium sulfide light sensors. As the amount of incident light increases, the returned 8-bit value also increases. Likewise, as the amount of incident light decreases so does the 8-bit reading. Using these two sensors you can create light tracking or light avoidance behavior. For light tracking you might to have your robot follow a flashlight or search out the brightest spot in a room. In plants this behavior is called Positive Phototropism. For light avoidance you could program your robot to avoid bright light. This light avoidance in the plant world is called Negative Phototropism.

## Maze Navigation:

You can perform maze navigation by using OPEN-ROBOT's built-in RFID reader. Simply program each RFID tag with a special code and then place these throughout your maze. Now try to create artificial intelligence for your robot so that it can successfully navigate the maze.

## Survival of the Fittest:

A swarm of OPEN-ROBOT's can compete for survival of the fittest when RFID tags are programmed with virtual food and then distributed throughout the robot environment. You can specify a set of rules that control how many pieces of food can be eaten from a tag or even how many pieces of food a robot can carry. Additionally, you can prescribe

how frequent and how much food a robot must consume to stay alive. After programming your robots with food foraging intelligence, let them run free and see which robot lives the longest.

**Vector Navigation (*Requires WW02 Wheel Encoders*):**

When WW02 Wheel Encoders are added to OPEN-ROBOT you can perform closed-loop velocity and position control. This means that you can command OPEN-ROBOT to drive to a specific position in encoder ticks. By specifying the number of encoder ticks for each wheel, you can command OPEN-ROBOT to drive in a straight line for a specified distance or rotate to a prescribed angle. There are 128 encoder ticks per wheel rotation. You will need to measure the diameter of OPEN-ROBOT's wheels and the distance in-between the two wheel's centerlines. With this information you can use algebra to determine how many encoder ticks are required to command OPEN-ROBOT along a specific vector heading or rotate to a specific angle. This is a great way to introduce the concept of a vector.

**Open-Loop Versus Closed-Loop Control (*Requires WW02 Wheel Encoders*):**

An introduction to basic control theory can be covered when the WW02 Wheel Encoders are added. Run OPEN-ROBOT using open-loop velocity control and then using closed-loop velocity control. Do you notice any differences? You should notice that OPEN-ROBOT is able to drive straighter when closed-loop control is utilized. This is because the WW02 encoders provide feedback regarding the wheel rotation rate for each wheel. With this information, the PIC18F4520 microcontroller can adjust the rotation rate of each wheel to achieve the desired velocity. From here more advanced topics like Proportional-Integral-Derivative position and velocity control can be discussed. How does the PIC18F4520 calculate the Integral term? Technically speaking the PIC18F4520 would be required to perform integration, but there is a trick. When the control loops are updated on a fixed frequency there is no need to perform integration. The same is true for the derivative term.

**Intro to Image Processing (*Requires SRV-1 Blackfin Camera*):**

Adding the SRV-1 Blackfin Camera will open up a whole new world of possibilities. The SRV-1 Camera Board is based upon the BF537 Blackfin DSP, which runs at 500MHz and retrieves images from the attached 1.3 Mega Pixel camera. With this great piece of hardware you can decide to perform image processing at the Blackfin level or retrieve the images across the WiFi connection and process them using a desktop or laptop computer.