**Goal:**

Implement a feedback loop using the IR sensor signal so that a Redbot can follow a wall at a constant distance.

**Goals:**

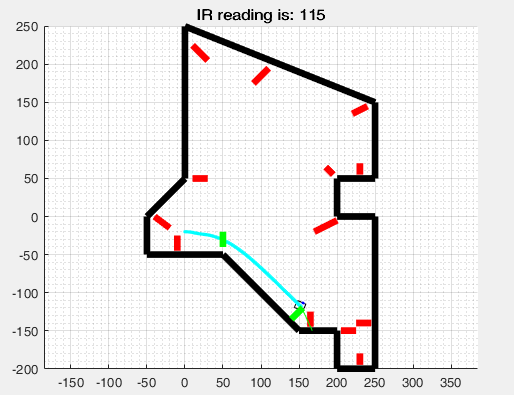
1. Implementing a feedback controller on the MATLAB.
2. Tuning the controller settings and gains to adjust Redbot steering performance.
3. Compete in the wall following performance tests against other students.

**Materials:**

* MATLAB

**Assignment**

* Download the .m and .p MATLAB code from CANVAS, and put them in the same folder. Make sure you can run script\_simRobotWallFollowing\_TEMPLATE.m script successfully.
* Design your torque command algorithm in the script\_simRobotWallFollowing\_TEMPLATE.m script so that the two-wheeled robot can move following a wall and **pass through all the red check markers**. The output torque range of each robot motor is [-1,1]. The red marker turns to green once the robot passes through it, shown in Figure 1. The same IR distance sensor in remote-lab 02 is used to measure the distance between robot and wall. You can change its position and orientation in the Matlab script.

****

Check maker

Figure Wall following lab scenario

* Once the robot passes through all the check markers successfully, Matlab will prompt that “Competition complete! Your **simulated completion time** is: XXX.XX seconds. Wall time simulation duration was: XXX.XX seconds.” in command window. **The team with the smallest “simulated completion time” will win the competition.**

**Procedure (these are just some suggestions for doing the tasks and is not a sequential set of instructions for doing the four tests):**

* Adjust IR sensor position and orientation so that it better suits your control strategy.
* Determine the reference distance you want to keep from the wall. You should consider the measurement range of the sensor and the check markers position.
* Using the distance signal, you can implement a control strategy for your robot:
  + Set a baseline forward speed for the robot.
  + Use your sensors to determine the distance to the wall, and determine the error from your reference distance.
  + Implement any combination of PID control to implement the tasks, **but you have to use at least one of them.**
  + Implement the turning command by adding it to the speed of one motor, and subtracting it from the other, which will make the robot turn.
* You can also use conditionals to make the robot change strategies:
  + For example, if the IR sensor’s signal suddenly changes to a very small value (because it can’t detect a wall anymore, like in the 90° exterior corner), make a turn to find the wall again.
  + What happens in the small inner corner.

**Report:**

Explain your strategy for navigating the maze and report your results for the tests.

1. Explain your overall control strategy, and show how your code implemented it.
2. How did you determine the desired distance from the wall?
3. How did you mount the sensors on the robot (give your sensorx, sensory, and sensor\_angle\_in\_degrees parameter values) and how is your configuration connected to the control strategy?
4. How did you choose the baseline forward speed?
5. How did you tune the gains for your PID controller?
6. What trade-offs did you encounter during the tuning process?
7. Submit the code that you have used for achieving the tasks
8. Optional: show plots of distance-to-the-wall vs. time for tests you conducted yourself.