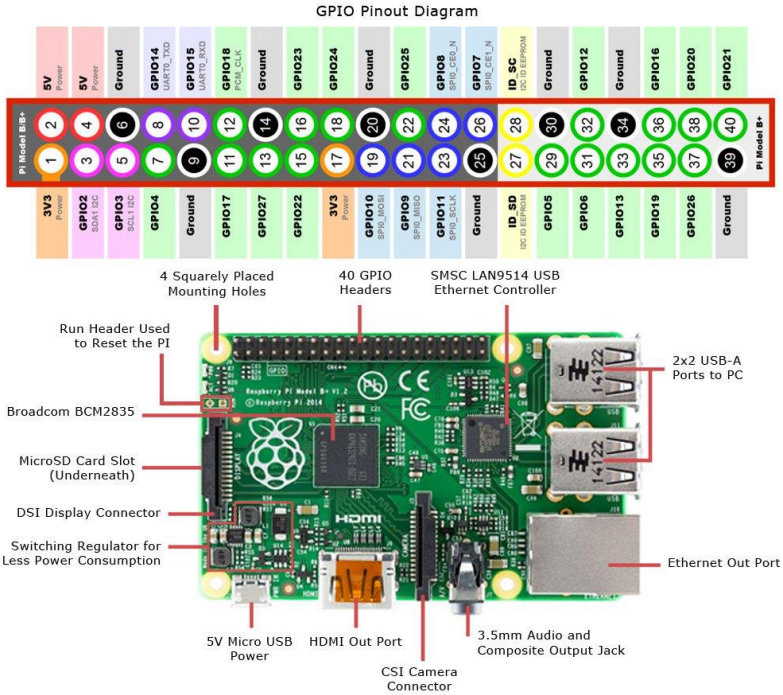
SE498 Introduction to Autonomous Vehicle System
Laboratory Assignment 3: Wall Following

Goals for this Lab Assignment:

1. Learn to interface with robot vehicle.
2. Explore all the sensors feedback and understand what they mean.
3. Control the vehicle to follow left/right wall using available sensor data.

Exercise 1 – Setup the interface system

1. Download the driver package zip file
2. Extract it to your workspace source folder.
3. Use catkin_make at your root workspace directory to compile all files.
4. Connect your PC to the mps432 via Serial port (RX/TX to TX/RX).
    a. On Raspberry Pi, TX port is pin 8 and RX port is pin 10
    b. Connect ground to msp432 ground



    c. On FTD1232 the pin layout is

5. Now, we go ahead and test which serial port we are using on our PC
6. Open terminal 1
    a. $ cd ~/catkin_NETID
    b. $ catkin_make

c. $ source devel/setup.bash
d. $ roslaunch cardriver enumeratePorts.launch

7. This will give you a list of ports and look for a port that is currently connected.
    a. If you are connected to PC, the port's name is usually /dev/ttyUSB0
    b. If you are connected to Pi, the port's name is usually /dev/ttyS0
8. Close the enumeratePort.launch.
9. Go to cardriver package, find yaml folder and setting.yaml file to update the port name
    a. $ roscd cardriver
    b. $ gedit yaml/setting.yaml
    c. Find the line port: '/dev/ttyS0' and make change accordingly
    d. To comment a line in yaml file, use '#'
10. The port name has been set correctly, launch the communication driver for the vehicle
    a. $ roslaunch cardriver COMM.launch
11. The launch file will automatically run roscore if it's not currently running so we do not need to run roscore separately.
    a. IF exception of permission has been thrown when launching the port, add the current user to dialout group and login / logout to enable access to serial ports.
    b. For example, in lab's PC, do
    c. $ sudo adduser ros dialout
    d. Then logout and log back in
    e. Launch the COMM launch file as in 10.a, the exception should be gone.

Exercise 2 – Test all sensors

1. Download lab3 skeleton code and extract to your workspace /src folder
2. Compile the code
    a. $ cd ~/catkin_NETID && catkin_make
    b. $ source devel/setup.bash
3. Find lab3 package, open /src folder
4. Lab3.cpp will contain the main() function, however the core implementation will be defined in WallFollow class implemented in wallfollow.cpp and wallfollow.hpp
5. The constructor for WallFollow class will initialize all publisher and subscribers.
6. The callback functions are all member functions of the WallFollow class.
7. BEFORE GOING FUTHER, please briefly read wallfollow.ccp code.
8. First, turn on your vehicle check if you see a LED blink at 1 Hz, make sure serial port is connected
    a. If comm driver is not launched, launch it using
    b. $roslaunch cardriver COMM.launch
9. Test the IR sensor using lab3code
    a. Uncomment #define TEST_IR from the wallfollow.hpp
    b. Compile the code
    c. Run following in another terminal
    d. $cd ~/catkin_NETID
    e. $catkin_make
    f. $cd source devel/setup.bash
    g. $rosrun lab3 lab3node
10. Question: What is the max and minimum range for each IR sensor? Write it down as a comment in your code.
11. Test the bumber switches
    a. Uncomment #define TEST_SW from the wallfollow.hpp, comment out the rest #define

b. $cd ~/catkin_NETID
c. $catkin_make
d. $cd source devel/setup.bash
e. $rosrun lab3 lab3node

12. Question: What is the format of the switch sensor data coming from the /ti/switches topic? Which bit of data represent which switch? What is the value of individual bit when it is triggered? Write it down as a comment in your code.

Exercise 3 – Follow a wall!

1. Now you have all the sensor information you need to write an algorithm to follow a wall.
2. See wallfollow.cpp comment to find where you need to implement your code
3. Requirements:
    a. The vehicle should be able to follow wall on the left or right side depend on user's input.
    b. The vehicle should be able to detect low profile obstacles with switches,
    c. And able to maneuver around the said obstacle.
    d. If no wall is detected the vehicle will travel forward until a wall is found.
4. DEMO is worth 80% of the lab grade
    a. The vehicle should not hit any of the wall
    b. The vehicle should can hit low profile obstacles and maneuver around it
5. REPORT is worth 20% of the lab grade
6. Your report should document the design decisions for the wall following algorithm. It should also discuss the challenges you face and what improvements could be made to the algorithm.

Reference: http://wiki.ros.org/turtlesim