E 190Q – Autonomous Robot Navigation

Lab 1

Introduction to the Jaguar Lite Robot

INTRODUCTION

The purpose of this lab is to introduce students to the Jaguar Lite Autonomous Vehicle platform and the associated C# base code that will be programmed within Microsoft Visual Studio (MVS). By the end of the lab, students should understand the system architecture, and how to program the robots to conduct future lab experiments. This lab should be done in pairs.

BACKGROUND

The Jaguar uses a host computer to command and control the robot. Command signals are accepted from a host computer application then sent over wireless communication to an embedded microcontroller on the robot. Sensor signals are sent from the embedded microcontroller back through the wireless communication to the host computer application. See Figure 1 below.

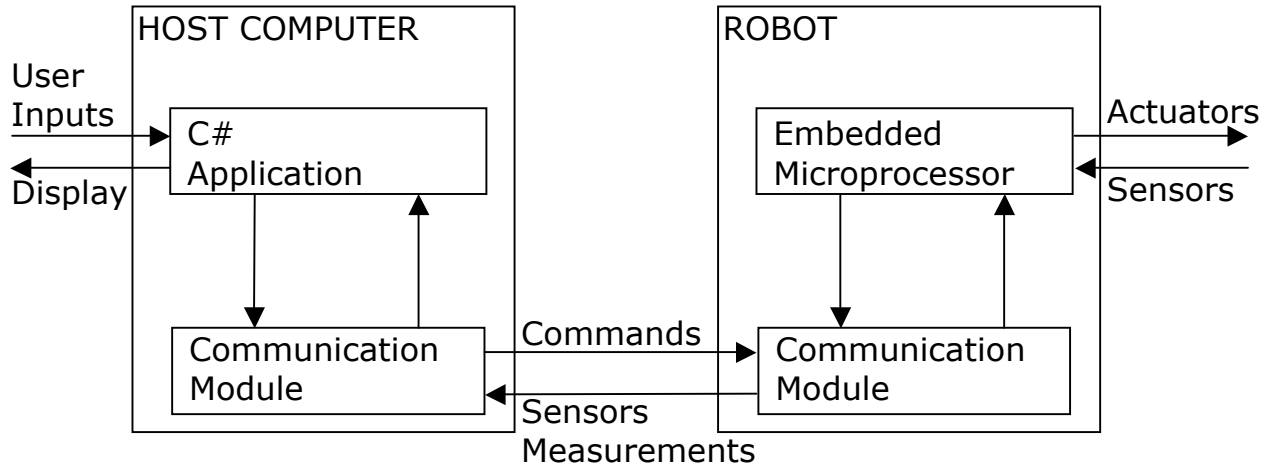


Figure 1: System Architecture

In this course, we will only be concerned with programming the C# Application on the host computer. The communication modules, embedded microprocessor, actuators and sensors will not be reprogrammed or reconfigured.

For detailed description of the Jaguar Lite, see the manual that can be downloaded from the course website's lab page.

EXPERIMENTS

1. Locate the parts

Familiarize yourself with the robot. Be careful when lifting the robot off the ground. Find the following components:

- Laser Range Finder
- Head Lights
- GPS/IMU
- Handle Bar
- Video camera
- Recharging sockets
- Wireless Ethernet antenna
- Power switch

Turn the robot on.

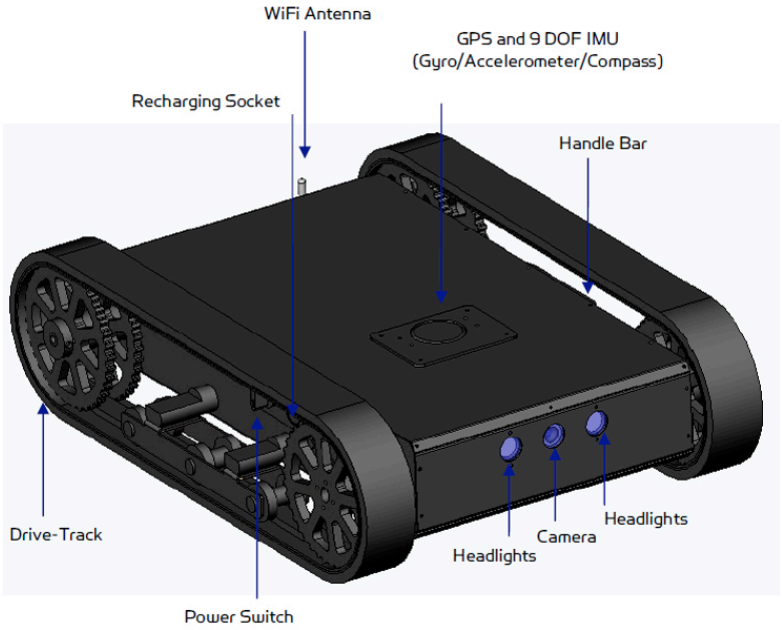


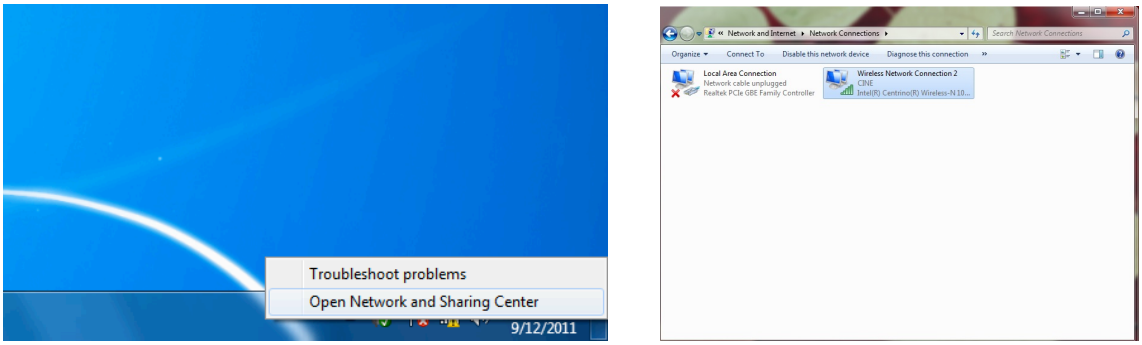
Figure 2: Jaguar Lite Components.

2. Setup the Wireless Connection

To make a wireless communication connection with the robot, you must first establish a connection between your host PC and the wireless router in the robot. Each robot has a different wireless router, so make sure you connect to the correct router, (i.e. robot).

a. **Set the Host PC's IP Address** – Open the *Network and Sharing Center* by clicking on the wireless icon of the bottom right section of the windows tray. Then select *Change Adapter Settings*. Double click on the Wireless Connection Settings (Fig. 2b). Select *Properties* (Fig 2c). Then double click the *Internet Protocol Version 4 (TCP/IPv4)* connection, (Fig. 2d). Now, set the IP address of the PC to be 192.168.0. XXX, (Fig. 2e). Note, XXX can be anything but the

addresses already used by the robot's router. See Table 1 for a list of robot router IPs not available for your host PC. Be sure to set your subnet mask to 255.255.255.0.

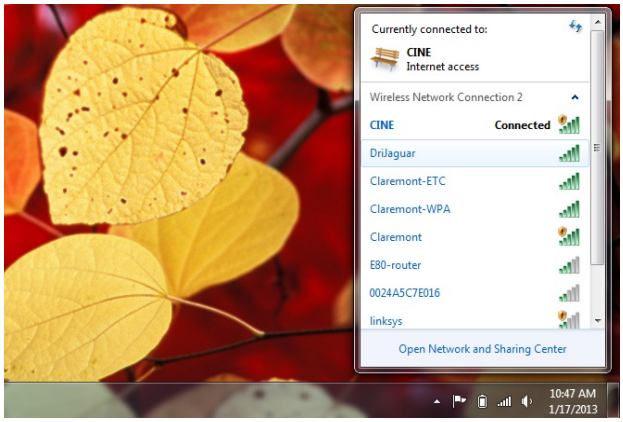


(a) (b)

(c) (d) (e)

Figure 2: Setting the Host PC IP address

b. **Connect to the Robot Router** - Left-click on the wireless connection icon located on the bottom right corner of you system tray, (see Fig. 3a). Disconnect from CLAREMONT-WPA, CINE, etc. if your computer is connected to it.



(a) (b)

Figure 3: Connecting to the wireless router

Select the robot's router SSID that is specific to your robot as indicated in Table 1, by double-clicking the SSID. Each robot has its own router and you must select the one corresponding to your robot (see table below). Enter the routers Web key: drrobotdrrobot. Note you will not be able to get internet access and be connected to the robot simultaneously. The internet needs the Host PC IP to be automatically configured with DHCP, while connecting to the robot requires the Host PC IP to be hardcoded as indicated below.

.

Robot	Gateway Robot ID	Robot Router IP	Robot Motion Board IP	Host PC IP	Router SSID
Lopez	DrRobot	192.168.0.245	192.168.0.60	192.168.0.XXX	DrJaguar
Yogi	DrRobot	192.168.0.244	192.168.0.70	192.168.0.XXX	DrJaguarY
Zed	DrRobot	192.168.0.243	192.168.0.60	192.168.0.XXX	DrJaguarZ
Alice	DrRobot	192.168.0.242	192.168.0.90	192.168.0.XXX	DrJaguarA

Table 1: Network Configuration

c. **Setting the Robot IP address** – On your computer, you must set the IP addresses for your robots various components. The easiest way to do this is double clicking the *Jaguar Control* icon. This will launch the *Robot Login* window, (see Fig. 4). You must modify the 1st digit on the 4th number of each of the three IP addresses. This digit is specific to your robot, (see table 1 Robot Motion Board IP column). Once you select "Connect Robot" your settings will be saved to a text file and you won't have to run this program again unless you switch robots.

Note, once you finish your IP changes, the rest of the Jaguar Control program will be launched. You can explore the program, but you won't be needing it for class.

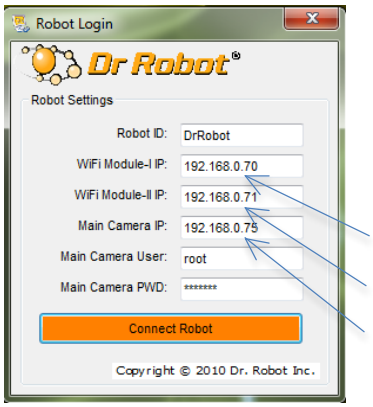
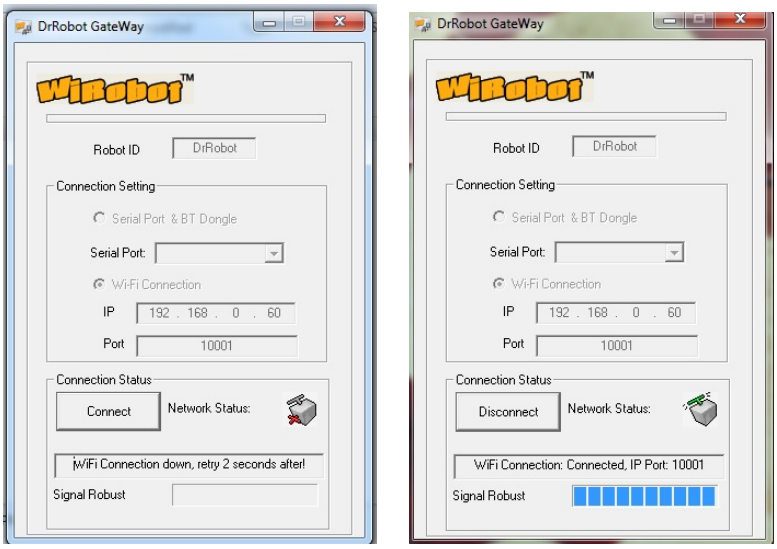


Figure 4: Changing IP addresses for each robot with the Robot Login utility. The blue arrows point to the three digits that are specific to the robot.

d. **Setup WiRobot GateWay** – There is a utility program created by Dr. Robot that enables the code we write to connect (through the wireless connection) with the robot hardware. This utility program, called the *WiRobot Gateway*, must be run for the robot to work. To run this program, double-click on the *WiRobotGateWay* shortcut located on your desktop. A window should pop up that looks like:



(a) (b)

Figure 5: WiFi gateway utility window. No connection has been made in (a), but a successful connection has been made in (b).

If the connection is made, the WiFi gateway window will minimize itself. You can double check that the connection is made by selecting it from the windows tray at the bottom. Make sure there is a green check on the power switch icon (see Figure 3b).

Ensure that the *Robot ID* is set to “DrRobot” as shown in Fig. 5. Note ALL robots in class have this same *Robot ID* regardless of IP. This will map your C# code to the hardware. If you want to use a different name other than DrRobot, you must also change this in your C# code.

Ensure the Wi-Fi Connection IP of the robot’s main controller is set to 192.168.0.XX, where XX is dependent on your robot (see Table 1). Note, all other IPs for that robot should have the same first digit in the last field of the IP. For example, if the robot motion board IP is 192.168.0.60, then the vision board should be 192.168.0.61. Likewise, if the robot motion board IP is 192.168.0.70, then the vision board should be 192.168.0.71. Enter the *Port* as “10001” as shown in Fig. 4.

Lucky for you, the instructor has written the C# base code so that it will immediately run the WiRobot Gateway utility upon application startup. The appropriate settings are saved in a file so you can enter them if you change robots. The settings are located in the file “C:\Dr RobotAppFile\OutdoorRobotConfig.xml.

3. Locate the code

In this class, we will not be touching the embedded code on the robot's microprocessor. Instead, we will program everything on the host PC. There is a C# code base that will run on the PC, using the WiFi connection to get closed loop control of the robot.

First, let's get the code base on to the PC. From the website lab page, download the zipped folder "E190Q-Lab01-BaseCode.zip" and extract it to your *Desktop*. You may want to rename the extracted folder "E190Q-Lab01-BaseCode_X", where X is your group name. There may be several groups using this same computer.

Open the folder and double click on the file "JaguarControl.sln". A visual studio window should open, with the base code solution, (see Fig. 6).

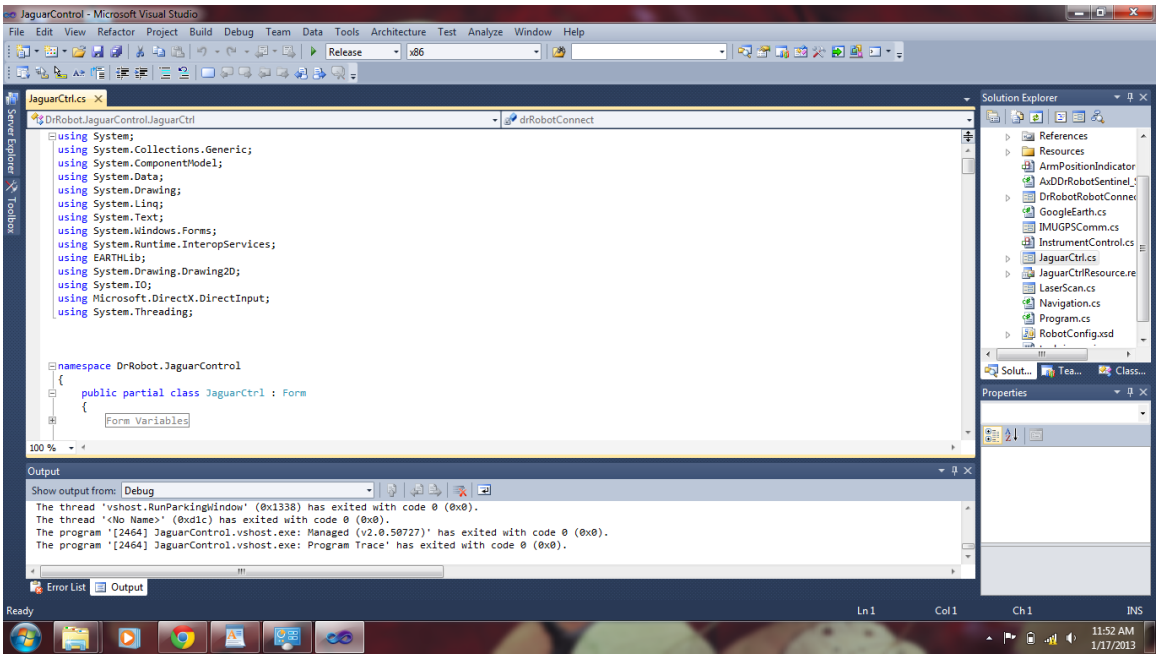


Figure 6: Jaguar Control Solution opened in Microsoft Visual Studio

Now you can use the list of files in the *Solution Explorer* (left or right pane) to navigate the code. You can right-click a file and select "View Code" to open them for viewing or editing. Make sure you can find the files listed below which you will edit in subsequent steps of this lab.

- JaguarCtrl.cs
- Navigation.cs

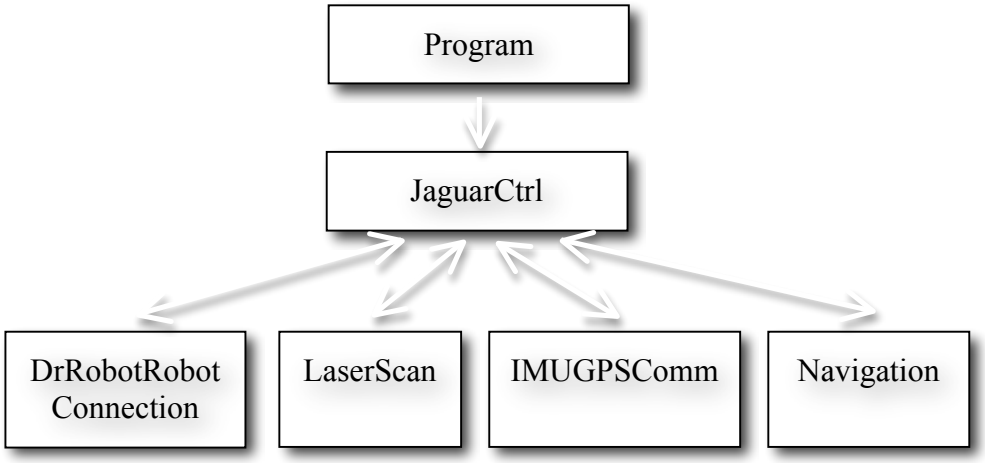


Figure 7: Overview of Class structure.

The overall code architecture is shown in Fig. 7. Note that the application file is *Program*, which instantiates a dialogue window: *JaguarCtrl*. Within *JaguarCtrl*, an instantiation of *Navigation* is created. This is where most of the programming is done for localization and control of the robot.

The *DrRobotRobotConnection* class is used to call the *WiRobotGateWay* utility application interface with the real robot through WiFi. The *LaserScan* and *IMUGPSComm* classes are used to access the respective sensors.

4. Build and Run

From the *Build* drop down menu on Visual Studio, select *Build Solution*. From the *Debugging* drop down menu, select *Start Debugging*. A dialog box should appear as shown in Fig. 8.

Orient yourself with the GUI. The left panel displays a 2D top down view of the robot within a Cartesian coordinate frame. The track bars allow forward/reverse actuation, and CW or CCW rotation actuation. Various sensor measurements are displayed. The bottom right panels have control buttons and settings.

Of importance is the *Hardware Mode* check box. This allows users to switch between control of a simulated robot agent, or the actual robot itself. Students should conduct all debugging in *Simulator* mode.

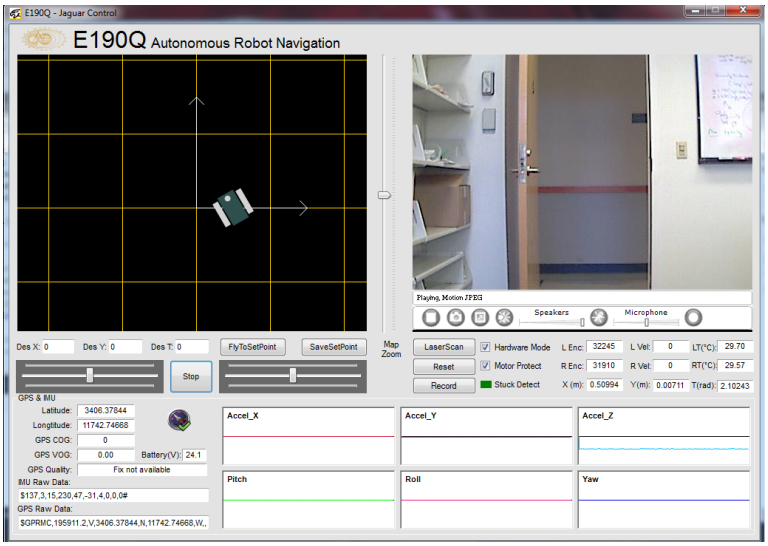


Figure 8: Jaguar Control Dialogue Window

5. Control the Robot Manually

Beach the robot on something so that it cannot move when the motors are actuated. Select the *Hardware Mode* check box. Experiment with the *Track bars* and *Stop* buttons.

Repeat using the *Simulator*. Note at this point the robot will not move around in your display, we will code that up in the next lab!

6. Reverse directions of robot Motion

In this part of the lab, you will edit the JaguarCtrl.cs file to make the motion control buttons work in reverse. The purpose of this step is to get acquainted with the code in two ways:

- Where command functions from the dialogue window are located.
- How to send commands to the robot

Find the function `trackBarForwardPower_ValueChanged()`. This function is called when the leftmost trackbar is changed. The code in this function sets the value of `forwardVel` based on the value of the trackBar.

The `DCMotorVelocityNonTimeCtrAll()` function sends the desired speeds to each motor on the robot. There are two calls with this function, depending on whether the desired motor velocities should be sent to the real robot or a simulator. Note the negative signs on the fifth argument being sent to the function, why are they present?

Now that you are somewhat familiar with the code, you can make some simple changes. See if you can edit the code to make the robot move backwards instead of forward when the leftmost track bar is moved to the left. Don't forget to change this back for later.

7. Modify the Robot Position

Find the function `LocalizeRealWithOdometry()` in the Navigation.cs file. This is called at every iteration of the control loop. In this function, let `x=x+0.005` (i.e. at every iteration of the control loop the robot's position will be increased 5 mm in the positive x direction.)

Compile and run the code. Your GUI robot should move forward on the screen.

8. Log Data

Try logging data using the *Record* button. Make sure you can find the file where the data is logging. Track within the code, how a button click results in file creation and data being saved. Determine what data is being written to the file. Modify the code so that a value of 9999 will be written to the log file in place of `y` at every iteration of the control loop. Have the instructor or proctor check this.

9. Closed loop control

BEFORE YOU START THIS SECTION, BEACH YOUR ROBOT. DON'T ACTUATE THE MOTORS ON THE GROUND UNTIL THE PROCTOR HAS GIVEN YOU AN OK.

The goal of this part of the lab is to make the robot drive to a wall, and position itself 1.000 m from the wall. The robot should start facing the wall, and be located somewhere between 0.50 m and 3.00 m from the wall.

The closed loop control should work as follows. If the robot is farther than 1.00 m from the wall, it should move forward and stop at a distance of 1.00 m from wall. If the robot is closer than 1.00 m to the wall, it should move backwards and stop at a distance of 1.00 m from the wall. If you are clever you can use a "Proportional Feedback Control" system. The laser range finder can provide the necessary sensing information.

The function to be edited is `WallPositioning()`, located in `Navigation.cs`. Note how the function is called from `RunControlLoop()`, the main thread called from the Navigation constructor. (Try to find these calls).

Within `RunControlLoop`, most of your future labs will be based. It is from here that different localization and control algorithms will be called.

To make this happen, you must set the values for `motorSignalR` and `motorSignalL`. Note, the central laser range measurement has been setup for you to use and is called `centralLaserRange`.

Test your code with the robot beached. Ask a proctor to check that the code is working before you put the robot on the ground for testing.

10. Report

Write a 1-2 page report documenting your closed-loop control design. Be sure to include the following sections: abstract, introduction, problem definition, control design, results, conclusion. Be sure that your results section includes plots of distance to the wall as a function of time.

Note, all lab documents in this class will follow the template found at:
http://www.ieee.org/conferences_events/conferences/publishing/templates.html

Please submit your lab 1 report by 9am, Friday January 30th, 2015.