*Gesture controlled iRobot with Intel Galileo*

*Qiukai Lin*

This project aims to program iRobots to move according gesture inputs from webcams. After webcams receive image data, we translate the image into corresponding iRobots commands. These commands will be processed and transmitted wirelessly to Intel Galileo, which linked to iRobots. Intel Galileo sends signals to control the motion of iRobots.

***Communication between iRobot and Computer***

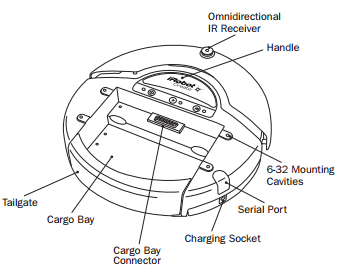
Before using Galileo to control iRobot, we can first use serial cable to control iRobot directly from laptop by using Open Interface commands of iRobot. This will help you become familiar with serial codes used in iRobot.

1. Read [Owner’s guide](http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Manual_Final.pdf) and [Open Interface](http://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf) to learn OI commands in iRobot

Owner’s guide provides some general introduction of iRobots and have some sample serial codes, while Open Interface serves as a reference of all serial codes and ports. You can also look at this [tutorial](http://projectsfromtech.blogspot.com/2013/11/irobot-create-getting-started.html).

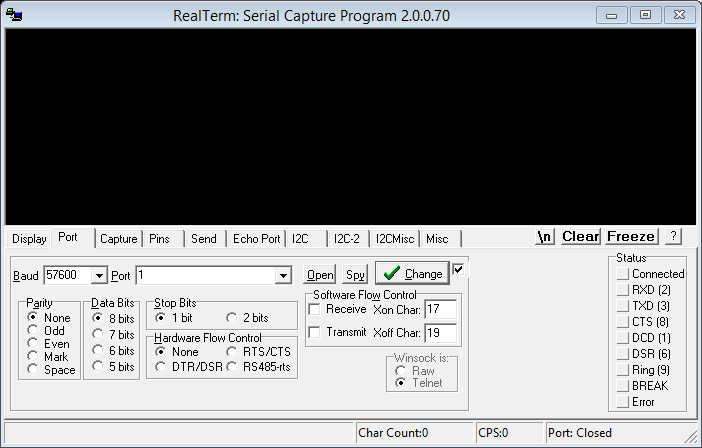
In order to connect iRobot with computer, we need to use the serial cable that comes with iRobot. Since my laptop can only connect to an USB port, I need to connect the serial cable with another USB to RS-232 DB9 Serial Converter. I connect the USB port with the Serial Port of iRobot, which is shown at the top view of iRobot.

Here is a top view of iRobot:



1. Download [Realterm](http://sourceforge.net/projects/realterm/files/) for serial control as shown below

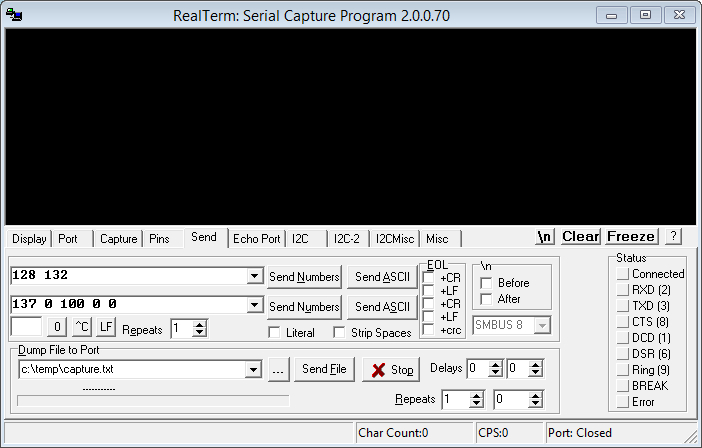
Realterm is a serial terminal used for debugging and development.



1. Select “Port” tab and select port
2. Configure the serial settings to 57600 baud, 8 data bits, 1 stop bit and no flow control

Note: 57600 baud is the default baud rate for iRobot.

1. Select “Send” tab to enter OI commands as shown below:



1. Click “Send Numbers” to send commands to iRobot

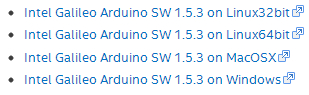
You can always refer to Open Interface for complete explanations of all serial codes and can use their examples to test iRobot if you wish.

***Communication between Arduino IDE and Intel Galileo***

Now we have to install necessary software. You can find more detailed steps in this [Guide](https://communities.intel.com/docs/DOC-21838). The following instruction is based on Windows. Basic steps are:

1. Download and install [Arduino IDE and Linux image](https://communities.intel.com/docs/DOC-22226)

Arduino IDE has different versions for various boards. Currently there is a specific version of Arduino IDE for Intel Galileo, which can be linked from the link above. When you go to that website, you can see as following:



You can download any version that works in your operating system.

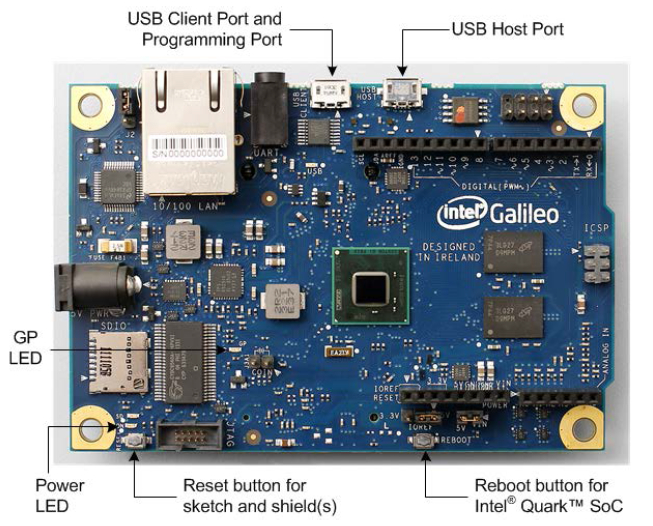
Also, you will see Linux image in Additional Downloads:

C:\Users\ENQLIN\AppData\Roaming\Tencent\Users\418788819\QQ\WinTemp\RichOle\(EEI2L]0G}XCR(UCM(4JYAH.jpg

Download it as you will need it in WiFi part of this project.

To install Arduino IDE, we need to unzip all files to C:\ directory. Make sure to preserve the folder structure. This for Windows, and you can see instructions of other OS in the Started Guide in the website of Intel, which is at the link before the first step.

1. Connect power and USB Client port (closer next to Ethernet) to your computer through serial cable, as shown below. Remember to wait until USB LED lights up to connect the cable, otherwise you may damage the board!



1. Install Drivers and software

In your computer, follow **Start>Control Panel>System>Device Maneger**

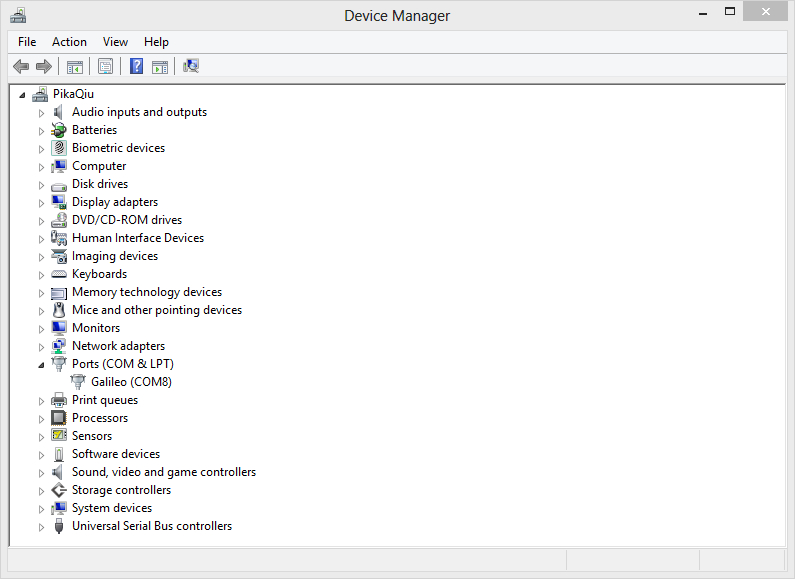
Go down and find **Port**. If not, go to **Other Devices**.

You should see **Gadget Serial V2.4**. Right click and select **Update Driver Software**.

Choose **Browse my computer for Driver software**

Navigate to **hardware/arduino/x86/tools** directory

Once it is installed properly, you should see something similar to the following screenshot:



1. Update firmware

Launch Arduino IDE. It is under folder **Arduino-1.5.3**, double click **arduino.exe**

Under **Help** tab, select **Firmware Update**.

It will take few minutes to update firmware on board, and you should have power connected all the time. The board itself usually comes with an older version, which may have unexpected outcomes in the future if you don’t update it.

Note: Do not have SD care plugged in at this time.

1. Upload programs

You can select **Board** and **Serial Port** under **Tools**.

You can go to **Files>Examples** for numerous sample codes already written with helpful comments. You can also upload them to Intel Galileo in order to see how things work if you are not familiar with Arduino IDE yet. It is a great way to introduce yourself to Arduino.

You can now click **Upload** button to upload. Programs will automatically compile.

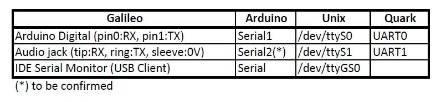
Note: If you want to add libraries to Arduino IDE, you can just download libraries in a zip file and add libraries by **Sketch>Import Library>Add Library**. You can use either zip file or folders. If you need more detailed information about how to add additional libraries, you can refer go [here](http://arduino.cc/en/Guide/Libraries).

Note: Intel Galileo has more than one Serial ports.

**Serial** refers to USB Client port. It is responsible to send programs from other devices to Intel Galileo board, and Serial.write() will print out results in Serial Monitor, which can be found in the upper right corner of Arduino IDE.

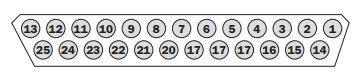
**Serial1** refers to Pin 0 and Pin 1 in Intel Galileo digital pins. Therefore, when you are programming iRobot using wires connecting Pin 0 and Pin 1 for TX and RX, you have to use **Serial1** to talk to iRobot, while you need **Serial** to enable communication between computer and Intel Galileo.

This can be better seen from the chart below, which is from [here](https://communities.intel.com/thread/48048):

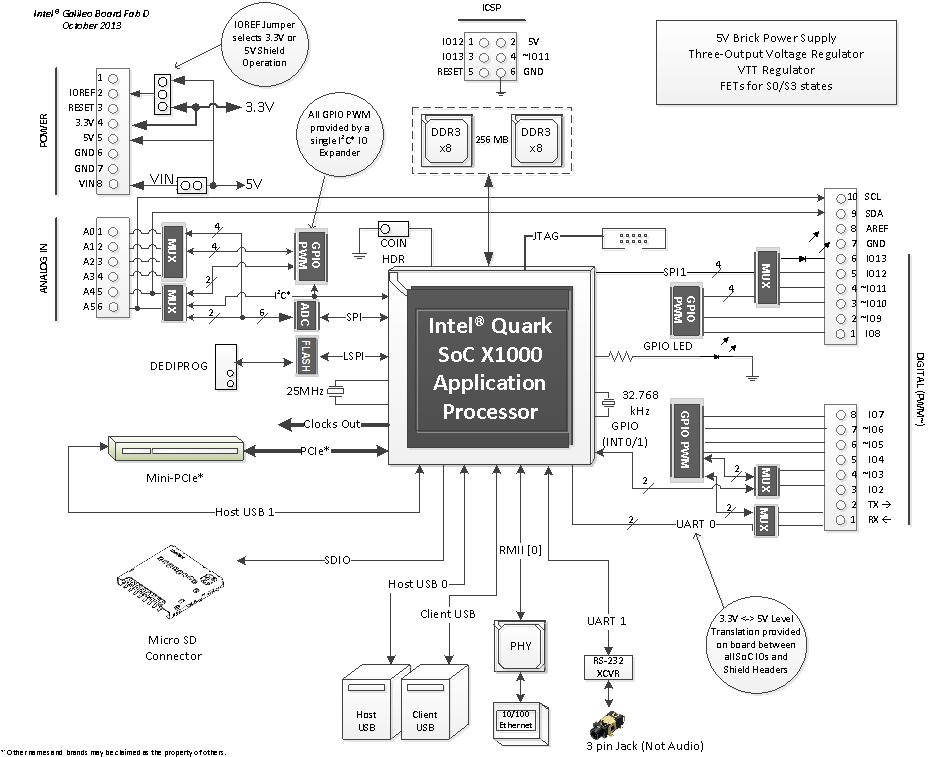


***From Intel Galileo to iRobot***

We are now ready to connect power and serial communication interface between Intel Galileo and iRobot. Make sure that you have power on before making any connections. Otherwise you may damage the board. There are 25 pins on iRobot as shown below. It is fine to just hook up a wire to connect Galileo pins and iRobot pins.



If you refer to the picture in the first section, Pin 1 (RXD) and Pin 2 (TXD) are located in DIGITAL section, while the ground power pin is located Pin 6 or Pin 7 of the Power section. You can see these from the datasheet attached below, and you can also go [here](http://arduino.cc/en/ArduinoCertified/IntelGalileo#.UyyQH42IOa8) for all hardware information of Intel Galileo.



1. Connect Pin 1 (RXD) on Galileo to Pin 2 (TXD) on iRobot

Pin 0 on Galileo receives data, while Pin 2 on iRobot sends data

1. Connect Pin 2 (TXD) on Galileo to Pin 1 (RXD) on iRobot

Pin 1 on Galileo sends data, while Pin 2 on iRobot receives data

1. Connect GND pin on Galileo to Pin 14 (GND) on iRobot

Note: Pins on iRobot start with Pin 1 until Pin 25, while digital pins on Galileo start with Pin 0.

There are more than 1 GND pins on iRobot. Both of them work.

The wiring is all set.

***WiFi Setup and connection***

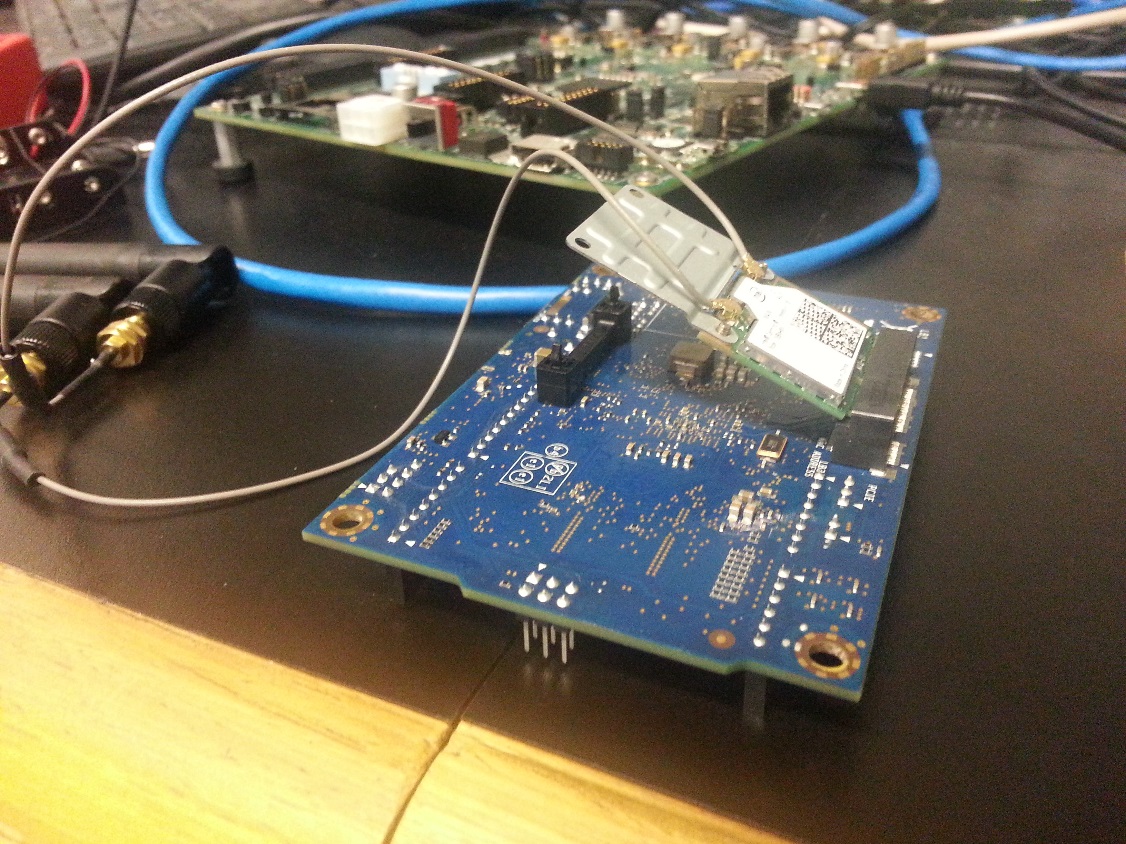
When everything is working as expected, you can get started with WiFi set up. We first focus on hardware set up. Here is an [instruction](http://ionospherics.com/intel-galileo-setting-up-wifi/) with graphs.

1. Insert two screws to put together *Half to Full Height Mini PCIE Bracket Adapter* with Intel WiFi card N-135

Note: WiFi card N-135 and N-6205 already has their drivers included in the Linux image provided. If you are using other WiFi cards, you have to download drivers separately from [here](http://wireless.kernel.org/en/users/Drivers/iwlwifi).

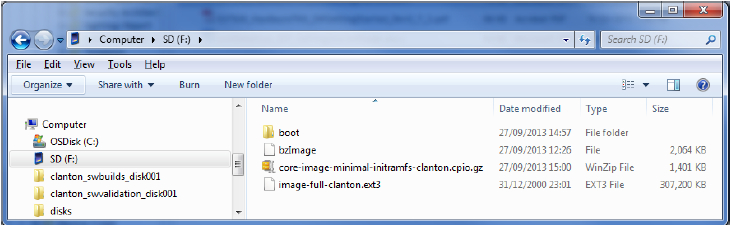
When your download is complete, Unpack with “tar –xvzf iwlwifi-XXX.tgz”, then copy files from your PC to the SD card.

1. Attach antennas to the WiFi card



1. Insert card into PCIE connector slot 45 degrees up and then pull all the way down
2. Unzip Linux image and transfer everything to SD card

Plug in the SD card into your computer. Unzip the files and upload to the SD card directly. Again, preserve the folder structure. Following is a sample from Intel:



After booting SD card, we can plug it into Intel Galileo, and then we can power on the board through 5V power cable.

When you connect your computer with Intel Galileo wirelessly, you can set up a new WPA network on your computer and make Intel Galileo connect to it. In order to do that, you have to set up network and upload programs through serial port to make Intel Galileo ready for WiFi.

1. In Windows, search cmd and run as administrator. You will see a command window

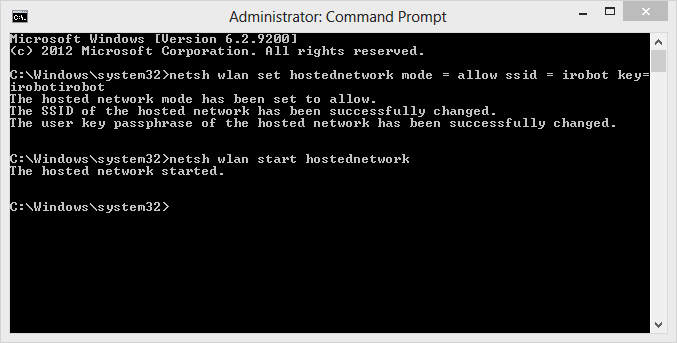
If you do not run as an administrator, you will not be able to set up a new network.

1. Enter: netsh wlan set hostednetwork mode=allow ssid=yourssid key=yourkey

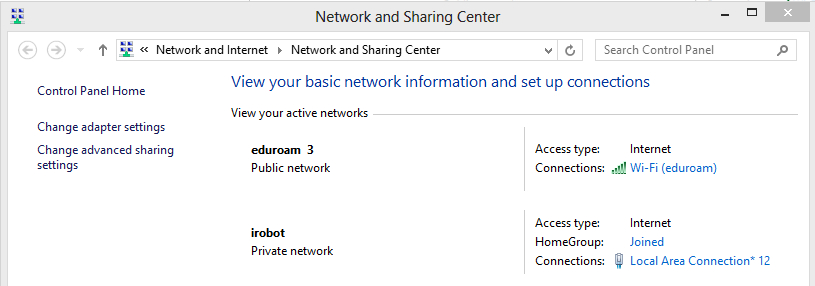
Change yourssid to the name of the network. You can make whatever you want

Change yourkey to the password. It should be at least 8 characters

1. Enter: netsh wlan start hostednetwork. I set ssid as irobot and key as irobotirobot. After you complete these steps, you should see something similar to this:



1. You can now check status of network in **Network and Sharing Center**



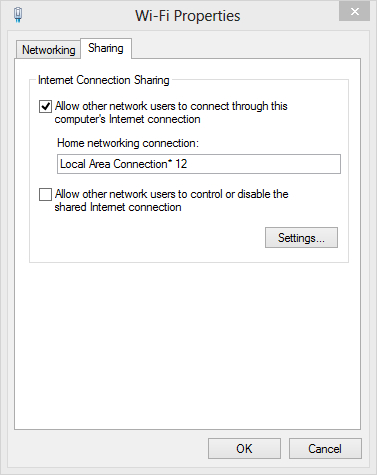
1. Share WiFi with the network that you just create

On the left hand side, choose **Change adapter settings**

Right click on your current. It is eduroam WiFi in my case. Choose **Properties**

Go to tab **Sharing** and share the network with the network that you create

If you find problems, you can refer to [this](http://www.talkofweb.com/creating-wi-fi-hotspot-in-windows-8-share-laptop-internet-connection/) site for instructions and trouble shooting



1. Open Arduino IDE. Go to **File>Examples>WiFi>ConnectWithWPA**

In char ssid[], change “yourNetwork” to your ssid

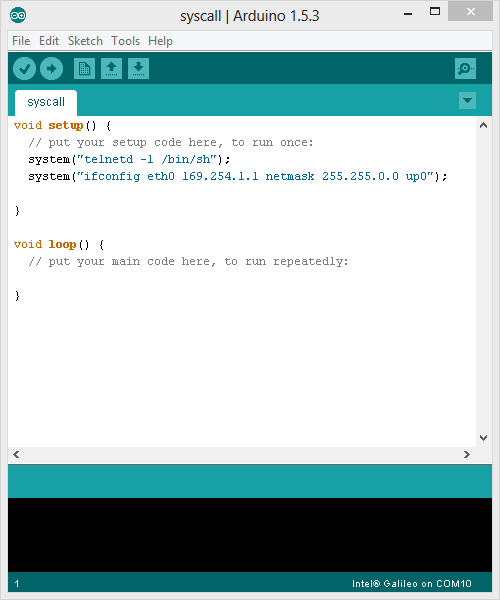
In char pass[], change “secretPassword” to your password of network

1. Upload the program to Intel Galileo. Then click **Serial Monitor** to check status of connection
2. You can find IP Address in Serial Monitor. It will be used later.

Note: In Step 5, after you set up network once at your computer, you can then just enter the last line to activate the network that you have previously set up.

As you may notice, the IP address that Intel Galileo obtain varies every time. You probably want to make it static and therefore easier to use. In order to do that, you have to download PuTTY or any SSH client that you prefer. These instructions of WiFi static address setup are from [here](https://communities.intel.com/thread/49348).

1. Download PuTTY from [here](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html) and install it
2. Upload this program to Intel Galileo

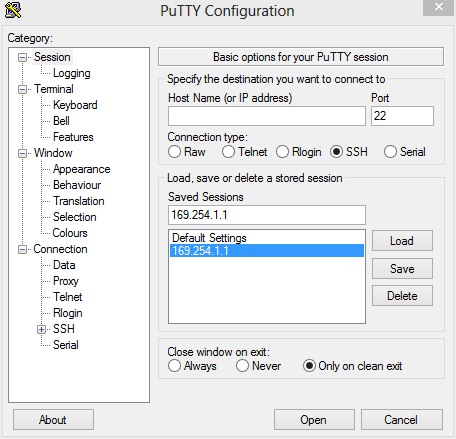


1. Then we can use PuTTY for communication between computer and linux image on Intel Galileo

We have to first connect Ethernet cable from computer to Intel Galileo

Next we will open PuTTY to connect to Intel Galileo. At this time only Ethernet cable is needed.

Here is a screenshot of my PuTTY configuration:



1. create a backup of your /etc/network/interfaces file

You can log in as “root”

cp /etc/network/interfaces  /etc/network/interfaces.backup

1. vi /etc/network/interfaces

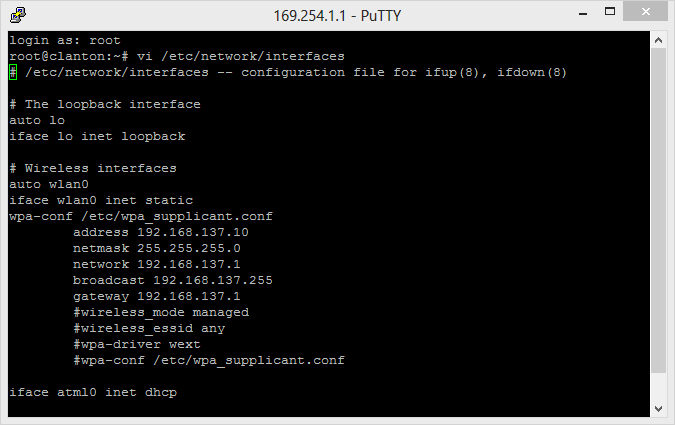
Press “i” to enter edition mode

Change dhcp in the "iface wlan0 inet dhcp" line to static

Comment out all the lines below and add the following lines.

In my case, I set the static IP address to be 192.168.137.10

This is how my setting look like:



1. Save the file, and restart the wlan0

You can type “:wq” to save and exit vi

ifdown wlan0

ifup wlan0

***Use Processing with OpenCV library***

Next part will be gesture recognition. You may use Processing with OpenCV to achieve this goal. I use Processing because it is very similar to Arduino IDE. The GUI is almost the same, and therefore Processing will be very easy to use if you already know about Arduino IDE. You can refer to [this](https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/introduction) site to familiarize yourself with the communication between Arduino IDE and Processing.

Let’s first setup Processing. The tutorial can be found [here](http://robottini.altervista.org/tag/javacvpro).

1. Download and Install [Processing](https://www.processing.org/download/?processing)



Processing is a programming language that is built by MIT media lab for visual design purposes.

The installation is same as Arduino IDE. Simply unzip all the files to C:\ directory

1. Install both runtime components for Microsoft Visual Studio [x86](http://www.microsoft.com/en-us/download/details.aspx?id=5555) and [x64](http://www.microsoft.com/en-us/download/details.aspx?id=14632)
2. Download [OpenCV](http://opencv.org/downloads.html) and extract to C:\ directory



1. Change Environment Variables

To change Environment Variables directly is not very convenient. You could download [RapidEE](http://www.rapidee.com/en/download) to modify Environment Variables.

Again, you have to run as an administrator if you wish to change Environment Variables.

1. Find register key “Path” and right click on it. Choose **Add value**

**C:\opencv\build\x64\vc10\bin\**

**C:\opencv\build\x64\vc10\lib\**

**C:\opencv\build\common\tbb\intel64\vc10\**

1. Use JavaCVPro library

Download from [here](http://www.mon-club-elec.fr/pmwiki_reference_lib_javacvPro/pmwiki.php)

Extract and copy all in the Processing library directory: **Processing /Mode/java/libraries**

1. Use JavaCV library

Download from [here](http://code.google.com/p/javacv/downloads/list?can=1&q=)

Extract and copy all in the Processing library directory: **Processing /Mode/java/libraries**

Create a directory called **library**inside the**javacv-bin** directory and put all the file (not the sample directory) inside this new **library** directory

Rename the library**javacv-bin** extracted with this name:**javacv**

1. Use Javacpp library

Download from [here](http://code.google.com/p/javacpp/downloads/list?can=1&q=)

Extract and copy all in the Processing library directory: **Processing /Mode/java/libraries**

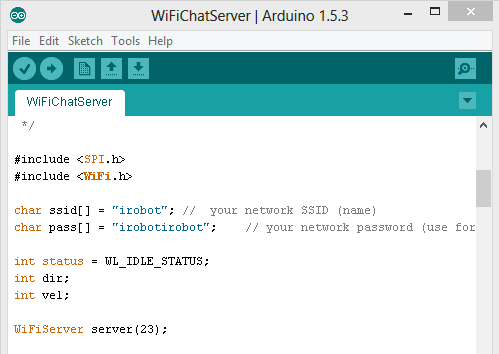
Create a directory called **library**inside the**javacv-bin** directory and put all the file (not the sample directory) inside this new **library** directory

Rename the library**javacpp-bin** extracted with this name:**javacpp**

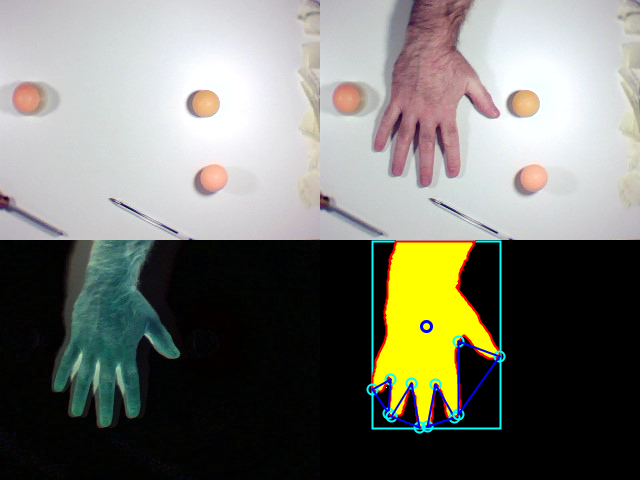
After all these steps, the chain of Processing —> JavaCvPro —> javacv —> javacpp- –> OpenCVis built, and therefore we can use Processing to access OpenCV and do some gesture recognition.

In order to make Processing to send information to Arduino, I set Processing to be client and Arduino IDE to be server. In order to complete the codes, you can refer to **WiFiChatServer** in Arduino IDE, under **WiFi** libray, and **HTTPClient** in Processing under **Net** library. After codes are complete, you can upload Arduino code to Intel Galileo, then unplug wires and start Processing codes.

For the Arduino codes, you can basically look at how **WiFiChatServer** is used and make some easy modifications so that it can receive commands from Processing. Remember to use your own ssid and key. Use port **23** for WiFi communication. Also, you can refer to [this](http://arduino.cc/en/Reference/WiFi) site for detailed usage of WiFi class in Arduino.



For the Processing codes that recognize gestures, I use examples from JavaCVPro. You can go to **Processing> modes> java> libraries> javacvPro> examples> javacvpro\_exemple\_detectFinger**. If you need reference for **net** library in Processing, you can refer to [this](http://processing.org/reference/libraries/net/) site. When you run this example, you will see two static pictures and how they are processed as below:



You can make modifications to this program to make it detect video from the webcam.

The modifications I made primarily include the following:

1. In Processing code, I move the steps that extract hands from background from setup() to loop, so it will continuously run, and also the static images are replaced by captures from the webcam. I first take a picture of the background, and compared it to new images captured by the webcam to detect hands.
2. I use net library, and set Processing as a WiFi client. I mirror the video before the images are processed, so that it will be more straight forward for manual control.
3. I change the minimum size that a blob will be detected so that the program will be more sensitive for smaller blobs, while some minor noises can still be filtered. I pick the point in blobs with lowest y position(highest point in the video) every time when a movement is detected. At the same time, I save previous positions every time when a new position is recorded, so that two points could be compared and calculated the direction and speed of movement.
4. I save the data from movement in a byte. The byte will be two digit number, while the first digit stands for direction of movement, and the second digit stands for the speed of movement.
5. In the Arduino code, I set Arduino as a server, and it will read data sent from the client. The Arduino code will first enable control of iRobot, and it will also connect to WiFi. It will continuously read data from the client if there is any. The code separates the byte sent from the client into two integers, one for speed and one for direction. According to different speeds and directions, various commands will be sent to iRobot.
6. The code is designed in the way that every time a new movement of up or down is detected, it will keep moving up or down. However, if the last movement detected is to turn left or to turn right, iRobot will keep turning for certain amount of time, with minimum degree to be 30 degrees. After turning is completed, iRobot will stop and wait for next command. If there is no movement is detected, the current movement will be continued, so that people do not have to keep waving hands if they wish to let iRobot run continuously in one direction. If there is not blob detected, iRobot will then stop all movements.
7. There are four lines across the screen. If you pass hands through the upper line, iRobot will go up with a certain speed corresponding to the speed of movement detected. Same thing happens when hands move left, right or down.

I attached my codes. In these specific codes, Processing will discover the direction that the hands are moving, and respond accordingly. For example, when the hand is waved to the left, the iRobot will turn left with specific speed that is detected from movement of hands. Also, it will keep moving until next command is received. Therefore, you do not have to keep waving hands when you want iRobot to move in the same pattern continuously. When no blob is detected, iRobot will stop moving.

***Helpful Links:***

[Serial monitor](http://arduinobasics.blogspot.com/search/label/Serial%20Monitor)

[Arduino MIDI](http://arduino.cc/en/Tutorial/Midi?from=Tutorial.MIDI)

[Intel Software Academic Program](http://www.intel-software-academic-program.com/pages/courses#diy)

[iRobot and Arduino](http://web.ics.purdue.edu/~fwinkler/AD61600_S14/AD61600_Arduino_iRobot.pdf)

[Create WiFi hotspot](http://www.talkofweb.com/creating-wi-fi-hotspot-in-windows-8-share-laptop-internet-connection/)

[Processing Network Library](http://processing.org/reference/libraries/net/)

[Connecting Arduino to Processing](https://learn.sparkfun.com/tutorials/connecting-arduino-to-processing/from-arduino)

[Wireless card tutorial](https://communities.intel.com/docs/DOC-22470)

[Intel Galileo Meets Wireless](http://www.malinov.com/Home/sergey-s-blog/intelgalileo-addingwifi)

opencv\_calib3d249d.lib

opencv\_contrib249d.lib

opencv\_core249d.lib

opencv\_features2d249d.lib

opencv\_flann249d.lib

opencv\_gpu249d.lib

opencv\_highgui249d.lib

opencv\_imgproc249d.lib

opencv\_legacy249d.lib

opencv\_ml249d.lib

opencv\_nonfree249d.lib

opencv\_objdetect249d.lib

opencv\_ocl249d.lib

opencv\_photo249d.lib

opencv\_stitching249d.lib

opencv\_superres249d.lib

opencv\_ts249d.lib

opencv\_video249d.lib

opencv\_videostab249d.lib

opencv\_calib3d249.lib

opencv\_contrib249.lib

opencv\_core249.lib

opencv\_features2d249.lib

opencv\_flann249.lib

opencv\_gpu249.lib

opencv\_highgui249.lib

opencv\_imgproc249.lib

opencv\_legacy249.lib

opencv\_ml249.lib

opencv\_nonfree249.lib

opencv\_objdetect249.lib

opencv\_ocl249.lib

opencv\_photo249.lib

opencv\_stitching249.lib

opencv\_superres249.lib

opencv\_ts249.lib

opencv\_video249.lib

opencv\_videostab249.lib



style="font-family: Handlee; font-weight: bold; font-size: 16px; width: 209px; min-height: 35px; padding: 0px 2px; border: none; color: rgb(255, 255, 255); box-shadow: rgb(63, 155, 155) 0px 3px; top: auto; background: rgb(0, 190, 152);"