**Using the robot:**

To use the robot, first connect the computer/laptop to its wifi ‘RPiAdHocNetwork’. Then SSH into the Raspberry Pi with Putty. It sometimes fails a few times but should work on the third or so attempt. Once connected it should ask you for a username and password. Username is pi and password is raspberry. Then change to the correct directory ‘cd Initio-PiRoCon/Python’. Run the script ‘sudo python pirover.py’. It should say ‘Waiting for connection…’. Now start the Java program on the computer/laptop, the Raspberry Pi console should say ‘Connection Successful!’. You can now send and receive commands from the Java UI, what is sent can be seen in the ‘Sent Messages’ tab in Java, and whenever the Raspberry Pi receives a message it prints it to the console. When multiple commands are sent at once they are put into an array, which the Raspberry Pi prints for you to see.

**Common Problems:**

**Commands piling up on the Raspberry Pi** e.g. forward02.0reverse0.3, the Raspberry Pi will still put them in an array, but the robot will just do nothing so no problem. Just send the commands again. Caused by very short term connection issues.

**Raspberry Pi console unresponsive/ bad connection** - Wait a few seconds, but if this happens for a while Putty may say ‘Connection Caused Software Abort’ or some other error, just restart Putty, log in and run the script again and it should work. If it doesn’t, look at the WiFi dongle on the robot, if the blue light is on the WiFi is fine so just try logging in again. If there is no blue light this is probably due to the batteries needing charging as the dongle is very sensitive to current so is the first thing to go if the batteries start running out.

**Socket already in use** – This may happen if the program stops due to an error or is forcibly stopped by ctrl + Z while it says ‘Waiting for connection’ so the socket hasn’t closed. Only reliable solution is to restart the Raspberry Pi ‘sudo reboot’. To prevent this only forcibly stop the program when it doesn’t say ‘Waiting for connection’ (i.e. when the Java program is running).

**The robot’s motors won’t stop** – This may happen if there’s an error in pirover.py while the robot’s motors are running. First stop the pirover.py script with ctrl + Z then run stop.py in the console ‘sudo python stop.py’.

**Remotely accessing Raspberry Pi**

Can SSH into command line by using Putty with IP 192.168.1.1. Can use almost any port (I used 22).

Can access GUI by using TightVNC, information on how to do that here: http://www.raspberrypi.org/documentation/remote-access/vnc/.

**Important Python scripts for running robot**

These are in /home/pi/Initio-PiRoCon/Python:

(Note some came preinstalled on the SD card for the initio robot specifically. They can be seen here: https://github.com/4tronix/PiRoCon)

Initio.py – Came preinstalled on SD card and sets up GPIO pins for PiRoCon and defines some useful methods for running the robot’s motors and checking sensor input e.g. forward(speed), spinLeft(speed). I had to change the names of some of the methods as initially the robot didn’t do the correct things (for example, spinLeft(speed) made the robot turn right and vice versa), this is detailed in comments in the code.

Linesensor.py – sets up line following script. If both line sensors are on when this is run, then the robot must be on a white line, so follows a white line. If both are off then the robot will follow a dark line. If only one is activated then nothing will happen. There’s also some commented out code for following the edge of a line (one line sensor activated, one not) but I’m not sure how well this works. Also sometimes I have replaced spinLeft(100) with turnForward(5,100), the only reason for this was that quite often I was testing it on carpet and the robot can’t turn on the spot on a rough surface.

Servo.py – this sets up methods for moving the servos. A short pulse of about 0.5-3ms to the servo pins dictates where the servo should move to, this pulse needs to be repeated about every 20ms until the servo is at the correct position. There is a servoinit method which returns both servos to the centre which should be done when the java program is started up to make it work more reliably. The servo movement is somewhat unreliable due to the very short pulses involved (they may be inaccurate) but I’m not sure how this could be improved. Also the numbers used for prevPan and prevTilt are just what I found moved the servos to the centre through trial and error.

Pirover.py – creates socket and waits for a connection from the java program. Once it has done this it receives strings from the java program and makes the robot perform actions depending on what is received.

Stop.py – is an emergency stop. If an error has occurred while the motors are running so they keep going, run this in the console to make it stop.

**Improvements that could be made (known problems):**

* If more than 1 command is sent at once with the condition box ticked, when the sensors are activated and the robot does the condition action sometimes it doesn’t do it for long enough e.g. ‘Go forwards for 2 seconds, then turn left for 3 seconds, but if the obstacle sensors are activated reverse for 2 seconds’ sometimes the robot will only reverse for 1 second or less. Initio.stop() must be getting called from another thread before the robot has had a chance to reverse for the full 2 seconds.
* If the condition box is ticked but the sensors are already activated when the Go is pressed, the robot will do the condition action, but will then proceed to do the other commands too, instead of just stopping afterwards.
* If you put a large number of seconds in the command box (e.g. go forwards for 20 seconds) then press stop while it is running, then send another command quickly, the robot won’t do the next command for 20 seconds because the thread is still sleeping on the previous 20 seconds. Maybe find a way to wake the thread up when the stop button is pressed.
* The stop button doesn’t work when the follow line button has been pressed, because the followLine() method is still executing and can’t be interrupted. Find a way to interrupt and stop it safely.
* Very rarely when you tell the robot to do something like go forwards, it will go forwards but seem to struggle and move in a jerky way. Usually if the command is sent again it works fine. This has happened when the batteries are fully charged, so doesn’t seem to be a power issue. I don’t know what causes it.
* Add a feature to be able to control the speed of the robot from the Java UI.

**Setting up Raspberry Pi**

**Setting up ad-hoc network with Wi-Fi dongle**

Followed instructions here: <http://spin.atomicobject.com/2013/04/22/raspberry-pi-wireless-communication/>

Summary:

Opened network file for editing:

sudo nano /etc/network/interfaces

Altered file so it contains:

auto lo

iface lo inet loopback

iface eth0 inet dhcp

auto wlan0

iface wlan0 inet static

address 192.168.1.1

netmask 255.255.255.0

wireless-channel 1

wireless-essid RPiAdHocNetwork

wireless-mode ad-hoc

Restart wlan0 with:

sudo ifdown wlan0

sudo ifup wlan0

Raspberry Pi is now broadcasting a network and has IP 192.168.1.1. Now need to configure the Raspberry Pi as a DHCP server so it assigns the laptop (or whatever you’re using to connect with the Raspberry Pi) an IP within a certain range so they are on the same subnet and can communicate with each other.

Need to connect to the internet (Ethernet cable?) and download DHCP server package:

sudo apt-get update

sudo apt-get install dhcp3-server

Access DHCP config file:

sudo nano /etc/dhcp/dhcpd.conf

Alter it to contain:

ddns-update-style interim;

default-lease-time 600;

max-lease-time 7200;

authoritative;

log-facility local7;

subnet 192.168.1.0 netmask 255.255.255.0 {

range 192.168.1.5 192.168.1.150;

}