Robots and Line Following Activity Walkthrough

Commands to execute on the console are shown in ***bold and italic*** and the outputs you're likely to see are shown as *just italics*.

Fire up the Raspberry PI, and launch a console terminal window by clicking on Start > Accessories > Terminal as demonstrated by the facilitator. The Start button is an oblong button in the bottom left of the screen.

Type the following inside the console window and hit the enter key, to scan for available bluetooth modules

hcitool scan

You should see an entry like the following which matches the label on your robot...

$ hcitool scan
Scanning ...
07:12:11:23:70:57 IronGiant

Note the MAC address of your robot (the numbers separated by colons).

Run the following command, so you can refer to your address information just as MACADDRESS for the future.

MACADDRESS=07:12:11:23:70:57

To save time typing you can select the text by dragging your mouse in the console and use the menu within the Console window to Copy and Paste the text.

This line tells the system what the PIN of the device is. All the bluetooth modules have the same PIN...

echo "1234" | bluez-simple-agent hci0 $MACADDRESS

This line connects to the device and pairs with it(using the pin from the last command)...

sudo bash -c "hcitool cc $MACADDRESS && hcitool auth $MACADDRESS"

This line makes the remote bluetooth module on the robot available as a serial device at /dev/rfcomm0, which we will refer to in our python scripts later on. Note you may have to run *rfcomm release rfcomm0* before running this, if there was a previous active connection.

rfcomm bind rfcomm0 $MACADDRESS

Open a console window and change directory to the appropriate location

cd ~/Desktop/scripts

At this stage, you can put all the batteries into the robot, and it should power up and the Bluetooth module (with a name sticker on it) should start flashing.

Launch an interactive python session by running the following in a console

python

Within the python console execute the following line, which should create a connection to the robot (you'll see the blinking light on the bluetooth module go solid).

from speed_control import *

Let's prove that the Robot is connected by making it flash the light which is attached to it...
led.write(HIGH)
...and turning it off...
led.write(LOW)

If the robot doesn't respond, then press the small reset button on the Shrimp board and try again until it does. Once it's connected properly we can progress to later stages.

One of the things now available (imported from the speed_control.py file) is the setup() function, which configures the robot. Run it within the python console. The steps carried out in setup() will make it possible to control the motors and read the sensors of the robot.
setup()

Type the following to make the robot's right motor go to full speed
setSpeed(RIGHT,1)
...and stop it by entering...
stop()

then wait 0.1 of a second before turning off the motor again.
setSpeed(RIGHT,1),sleep(0.1),setSpeed(RIGHT,0)

Now try reading a sensor, by running this at the console, with the reflectance sensor over something white...
sense(reflectSensors)
...move the robot so that the reflectance sensor is over the black tape and run the same command again...

See the values coming from the LDR light sensors by running this step instead...
sense(lightSensors)

The list below shows additional routines (groups of steps known as functions) which have already been written for you.

They turn on and off the correct pins to carry out the following behaviours...

goForward() # runs both motors forward briefly
goBackward() # runs both motors forward briefly
turnLeft() # turns the robot left on the spot, with motors in opposite directions
turnRight() # turns the robot right on the spot, with motors in opposite directions
bearLeft() # moves just the right wheel forward, with the left wheel still
bearRight() # moves just the left wheel forward, with the right wheel still
stop() # stops the robot

...try to navigate the line by watching the robot and running different commands.

A few more advanced functions are available to you...

end() # disconnects from the robot and stops the python interpreter
navigateLine() # tries to use the reflectance sensors to navigate a line
followLight() # uses the light sensors to steer toward a torch or window

You can see the full source code of the speed_control module after ending your python session by typing the following at the console, to see how it is put together...

geany speed_control.py