Student Manual for
IVR Robotics Assignment

Introduction

The aim of this practical is to build and to control a robot such that it interacts with its environment. The drill assignment tasks are described in the next section below. We will use the EV3 Lego Mindstorms robot. Robots will be given out in the first robots practical session.

You will work in groups of two or three and must submit one joint report, i.e. only one student of each pair needs to submit the report. The report must contain a brief explanation of how the work was shared and how you think the marks should be distributed (e.g. equally or in some other proportion if you and your partner did not contribute equally to the work). Please contact maurice.fallon@ed.ac.uk immediately if you do not have a group.

Group List:
http://www.inf.ed.ac.uk/teaching/courses/ivr/assignment2/2016-17/ivr_third_yr_assignment_pairings_robotics.pdf

The assignment is estimated to take about 40 hours of work in total, i.e. your share will be about 20 hours. This time includes writing the report, but excludes revision of any material from the lectures or tutorials. Try to distribute the work evenly between you and your partner and in such a way that you can contribute what you are good at.

Responsibility for the Equipment

When handing out the equipment, both students are required to sign to show they have received the EV3 kit. A £5 deposit will also be taken for a key to one of the lockers. It is the responsibility of the students to take care of the equipment and to return a complete kit. The contents of the kit will be checked on return.

Using your own laptop

Setting up the EV3 for this class is time consuming and there are about 40 groups. For these reasons, **we will only provide help to students connecting to the EV3 using a DICE computer**. Instructions are provided below for using your own laptop instead, but we will not help with troubleshooting. In particular, avoid switching between DICE and your laptop, network settings are likely to break. We can always reset the SD card.

Typical Usage of the EV3 robot

The EV3 has been formatted with a version of Linux called EV3DEV (installed on the SD memory card). It can read from the sensors and can command the motors. We will use a python programming interface to control the robot. The following instructions describe 1) how to connect to the robot (aka the brick) 2) copy your program to the robot and run it 3) some drill tasks to become familiar with the robot.

You can find useful details about ev3dev on its project webpage:

- http://www.ev3dev.org/

Starting Up the robot:

1. Plug USB into computer and brick
2. Press square button on ev3. It boots through to the ev3dev penguin screen and the brickman screen to the main menu (showing file browser, device browser etc)
   On DICE computer:
   a. When the ev3 brick boots, the DICE network manager will create the ev3dev network interface with a static ip address
   b. Ssh onto the computer: ssh robot@ev3dev. Password: maker
3. You are now connected to the robot
4. (When you are finished shut down by pressing the upper right button a few times)

As a final note, do not remove the SD card from the robot as this will reset some of the settings and will require you to reconfigure them.

Typical Development Cycle:

You will develop your code (in python) in a text editor of your choice on your computer. When you want to test it on the robot, you will copy your entire code directory to the robot. This way you can keep your latest files on your computer at all times. (**Tip: use (git) version control as your code WILL break as it gets more complex**)

1. Write code on your computer.
   a. Make sure your scripts are executable (chmod +x test.py)
2. Copy the code onto the ev3 to test: **scp -r ivr_directory ev3:/home/robot/**
3. When you have copied your programs to the ev3 brick you can use its UI screen to run the program - by selecting it through 'file browser'. Unplug the USB cable and press the square button on the EV3 to run the program.
   a. You can also execute it directly from a terminal over usb to avoid having to reconnect or to easily see terminal printing ($ python test.py)
4. It's ALIVE! The robot will operate autonomously
   a. To quit a program press the upper right button for about 3 seconds

Robot Drill Lab 1:

First download the code samples from this location:

- https://github.com/robotperception/ivr_robotics

This contains the drill test scripts. Please uncompress it to ~/
You should add to this directory what you write during the drills, and then for the assignment. Please build reusable classes and functions. Do not implement everything in a single sequence of predetermined functions - it will eventually break.

1. Explore Basic Operation of Sensors and Motors

1. Build the simple wheeled vehicle illustrated below. Make sure to connect the switches to numbers and motors to letters, as shown
2. Connect to the robot as described in 'Typical Usage'
3. Progressively work through the tutorials in main.py. You can run main.py from either a terminal or through the ev3's navigation UI.
    a. Open Loop driving
    b. Turn an LED on and off with a switch
    c. Drive backwards and forwards using one switch to command each direction
    d. Build your first python class to keep 'state' of robot
    e. Read a set of measurements from the Ultrasound sensor and then write them to a text file

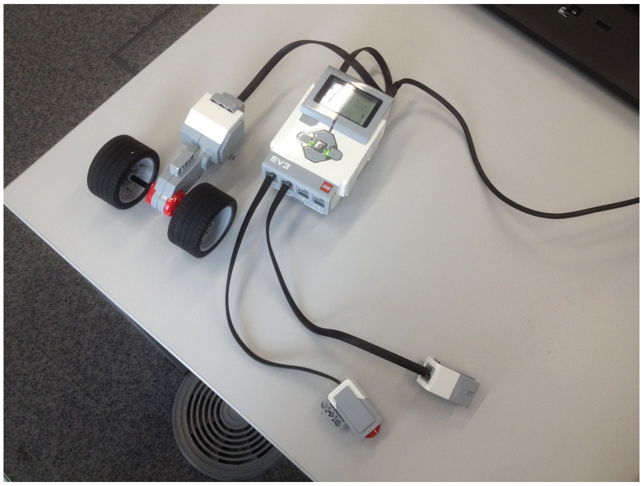

Fig: A Very Basic Robot - just wheels, the caster, a single motor and two switches

2. Build the reference robot

Use steps 1-40 of the EV3 manual to build a robot with two motored wheels and a castor. Place a color sensor facing downward (at the front) and a gyro on top (with the recycling symbol facing upward). You will later add the ultrasound sensor.

The robot will look like this: https://www.youtube.com/watch?v=u9SYv3LLMek
Make sure to have the robot ready before Drill Lab 2.

Robot Drill Lab 2:

A. Obstacle position estimator

1. Attach the MediumMotor and the sonar servo to the top of robot so that the servo can rotate the sonar to point in any direction to sense obstacles
2. Write a detector algorithm: it should servo the sensor over and back and to find an obstacle (such as a large box) placed in front of the sensor
3. Explore what the servo can and can't see e.g. how wide is the beam that it detects
    a. Determine the range from the Ultrasound reading e.g. what units does the sensor report? Metres, cm or mm?
4. After detecting the obstacle, have the robot to turn and move towards the obstacle (to within a few centimetres) and then stop. (HINT: You will need to implement a PID controller to control the rotation of the robot)

Snippet of code to print the ultrasonic sensor reading. It returns millimetres:

```
us = ev3.UltrasonicSensor()
while True:
   print us.value()
```

IMPORTANT: make sure that no other objects are in range of the sensor when testing. Otherwise it will get confused. It would be useful for you to do experiments to see what the sensor can detect e.g. at what maximum range can it pick up a large object like a box? How precisely can you detect where the object is? Properly described experiments would be very useful in your report later.

B. Wheel Odometry (plus gyro):

1. Develop an understanding of how the robot moves when commanded to turn its wheels for certain amounts of time and power. (Keep details of the experiments you do will be useful later on)
    a. for the following commands, determine where the robot ended up and what the motor encoder and gyro readings were before and after?
        i. 2 seconds forward at 25% power
        ii. 4 seconds forward at 50% power
        iii. 2 seconds turning at 25% power ...
    b. Repeat the experiment for one of the tests. How repeatable is each run?
    c. Create plots or experimental tables showing the relationship between the duration the wheels are commanded to turn versus the gyro reading change you

sensed. Can you learn a mapping between these two values? (You can use the plots in your report)

2. Develop a python class to command the robot to go to a position relative to where it is, open loop. This will use what you learned above.
    a. Given a goal relative to where the robot is [x,y, yaw]:
        i. Rotate towards the goal (turn until the gyro reads the desired heading)
        ii. Drive forward to the goal and stop when the encoder reading matches what you would expect.
        iii. Turn to face in the required final yaw (correcting for the first turn)
    b. How repeatable is this? I.e. how accurately could you do this for about 5 runs starting in the same place.

Snippet of code to print the gyro reading. It returns a number in degrees that the sensor has turned:

```
g = ev3.GyroSensor()
g.connected
g.mode = 'GYRO-ANG'
while True:
   print g.value()
```

Snippet of code to operate the motor and read its encoder value (which is the count of the number of counts it turns)

```
m = ev3.LargeMotor('outA')
m.connected
print m.position
m.run_timed(time_sp=3000, duty_cycle_sp=75)
print m.position
```

Outcomes:
- Explore what the sensor can see
- Develop obstacle detector which can be reused later
- Learn about gyros and odometry
- Build a navigation module which could be reused later

--------------------------- REFERENCE MATERIAL (NOT MANDATORY) --------------------------------

Installing ev3dev on SD Card to work with your own Laptop

These instructions were only tested with Ubuntu 14.04. This is required for distance students. **For local students we do NOT provide any support for you using your own laptop**.

Setup time was about 40 mins is downloading and installing ev3dev on an SD card

1. Install Ev3Dev on SD card (plugged into your computer). Then boot the ev3 with SD card installed:
   http://www.ev3dev.org/docs/getting-started/
   - Set up connection to the internet with "Ethernet over USB". When its functioning you should be able to "ping google.com"
   - restart the ev3 after doing all these steps as sensors might not work until after the first reboot. (so don't do the "do something awesome" step)
   - username: robot /// password: maker

   To set up a new network connection:
   a. Edit connections:
      i. Add ethernet connection
      ii. Connection name: ev3dev
      iii. Device MAC address: 12:16:53:xx:xx:xx
      iv. iPv4 Settings: manual, address: 192.168.xx.xx , netmask: 255.255.255.0
   b. Ssh onto the computer: ssh robot@ev3dev. Password: maker
2. Install the python bindings for the ev3dev:
   https://github.com/rhempel/ev3dev-lang-python#python2x-and-python3x-compatibility

Setting up a Development Environment [Advanced Users]

Setting up password-less login and aliases to use one command to send your files to the robot. **This is not essential but will make copying files over and back easier.**

SSH Login without needing to enter a password
Purpose: be able to log into ev3 computer as 'robot' from YourComputer without entering the password each time. From: http://www.linuxproblem.org/art_9.html

1. First on your computer, generate a pair of authentication keys. Do not enter a passphrase.

```
yourname@YourComputer:~> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/yourname/.ssh/id_rsa):
Created directory '/home/yourname/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/yourname/.ssh/id_rsa.
Your public key has been saved in /home/yourname/.ssh/id_rsa.pub.
The key fingerprint is:
##:##:##:##:##:#:##:##:##:##:##:#: yourname@YourComputer
```

2. Now use ssh to create a directory ~/.ssh as robot on ev3dev (The directory may already exist):

```
yourname@YourComputer:~> ssh robot@ev3dev mkdir -p .ssh
robot@ev3dev's password:
```

3. Append your public key to robot@ev3dev:.ssh/authorized_keys and enter robot's password one last time:

```
yourname@YourComputer:~> cat .ssh/id_rsa.pub | ssh robot@ev3dev 'cat >> .ssh/authorized_keys'
robot@ev3dev's password:
```

4. You should now be able to ssh to ev3 without using a password
yourname@YourComputer:~> ssh robot@ev3dev

Creating short SSH aliases

Purpose: To be able to make a short alias of the ssh command. On your computer, add these contents to .ssh/config:

```
Host ev3
    User robot
    HostName ev3dev
```

Final Result

- To ssh onto the robot: **ssh ev3**
- To scp a file onto the robot: **scp test.py ev3:/home/robot/**
- To scp an entire directory onto the robot: **scp -r ivr_directory ev3:/home/robot/**