Intro Week Robot Football Competition

Wednesday 21 September 2016

Dr Siobhán North

Department of Computer Science, University of Sheffield

Aim

The aim of this exercise is to build a working Lego robot that can compete in a robot football competition. You are provided with a partially completed Java program that, when completed, will allow the robot to be moved by pressing keys on the computer keyboard. Towards the end of the session the robots will attempt to play football.

Learning outcomes

Having completed these exercises you will:

- Understand how to compile and run a Java program;
- Understand how to edit a Java program using jEdit;
- Have gained some experience of team working;
- Understand how to control a Lego robot with a Java program.

If you have any problems put your hand up for one of the demonstrators, who are wearing blue T shirts, to come and help you.

Planning

- Remember that this is a team activity. Plan the robot before you attempt to build it, and think carefully about how to organize the team members so that you make best use of the available time.
- Start with the simplest approach possible and make that work before making things complicated for yourselves.
- Only run your robot program with the robot on the floor so that it doesn't dive off the desk and when it is on the floor make sure it doesn't scuttle off on its own.
- Consider how your robot is going to kick the ball. The most obvious way is to move a "kicker" with a motor, but there are many alternatives. Think about novel ways of using the components that you've got.
- Most of you, even the ones nearest the arena, won't be able to see your robot once the football starts. Practice controlling the robot in the arena using a team member as spotter.
- Think of ways to make your robot distinctive so that you can recognise it in the arena. Ideally you should be able to spot it via the display from the camera over the arena.

Setting up your robot

Each team has been given a named EV3 micro-computer and a box of Lego components that can be made into a working robot. The main elements are:

- The named EV3 micro-computer that controls the rest of the components;
- A chassis with two wheels controlled by large motors and a third small wheel;
- Some connecting cables;
- A medium motor;
- An ultrasonic (distance) sensor;
- Two touch sensors;
- A colour sensor;
- A gyroscopic sensor;
- Seven cables;
- Wheels, gears, axles and assorted building blocks.

Inside the box is a basic chassis. The microcomputer clips on top of this via four plastic pins at its four corners. The pins might be in the chassis or the microcomputer or both but the whole thing should slot together quite easily. Position the microcomputer so that its yellow stick on name label is over the two big wheels of the chassis. That is the front.

Next connect the motors which drive the wheels. Motors are driven by **motor ports** on the micro-computer which are labelled A, B, C and D. To drive a motor, you must connect its input port to one of the motor ports using a black cable. The program you are going to use assumes the left motor is connected to port B and the right motor to port C so that is how you must connect them.

Now you need to turn your robot on by pressing the dark-grey button on the front of the EV3 micro-computer until the light comes on. It takes a couple of minutes to sort itself out. When it is ready to use it will play a three notes and a menu will have appeared on its screen.

Setting up your Computer to Control the Robot

The general principle is that a Java program running on a desktop PC will act as a "remote control" for the robot. The PC communicates with the robot via a wireless (Bluetooth) connection, and your task is to modify the program we have given you to make the robot move in the right way when certain keys are pressed on the PC keyboard. The program responds to the arrow keys (forwards, left, right and backwards) and the space bar (for kicking).

We have provided you with the skeleton of a program to control the robot. It doesn't actually do anything except make things appear on the screen but it is a good idea to run it anyway to deal with any set up problems before you start programming yourself to make sure everything is working. The instructions below tell you what to do.

1. One of you must log in to a computer. Only one computer can control each robot so no one else need log on but you should all cooperate to write the program.
2. First download the skeleton program and all the associated software.
   a. Open a web browser and go to

http://staffwww.dcs.shef.ac.uk/people/S.North/campus_only/robots/index.html

b. Click on the link to the football software. This is a zipped directory which you should download and unzip. The link is called “The zipped directory to download at the start of the session”.

c. Open the directory **U:\ManWin** by picking
Computer → your username → ManWin
and drag the unzipped directory, which is called **Football**, into it.

3. Once the robot is switched on and ready, connect it to the computer by Bluetooth following the directions below. This is probably not the way you normally make a Bluetooth connection but follow these directions anyway because nothing else will work. You only need to do all this once per session.
    a. Right click on the Bluetooth icon at the bottom right of the PC screen and select **Show Bluetooth Devices**.
    b. Look for your robot’s name in the list of devices. Its name is on a yellow sticker on the robot and is repeated at the top of the screen on the EV3 micro-computer. It will be a single capital letter followed by a single digit.
    c. If your robot is present in the list already skip to step (f) below.
    d. If not, right click on the Bluetooth icon or on the Bluetooth device window and select **Add a device**
    e. When asked whether a number appears on the device select **Yes** and **Next**, even though no number appears. It should now be in the list of Bluetooth devices.
    f. Right click on your robot in the list of Bluetooth devices and select **Connect Using** and then **Access Point**
4. Now you can test the connection by running the skeleton program.
    a. Click on the **Start** button at the bottom left of the PC screen in the Windows menu bar then select **All Programs → Java SDK** and then click on the picture of a little black window. A new screen will open. It is called the **Command** window.
    b. Type **U:** followed by return into that window and the prompt will change to
    ```
    U:\>
    ```
    c. Type
    ```
    cd ManWin
    ```
    followed by
    ```
    cd Football
    ```
    The prompt will now be to
    ```
    U:\ ManWin\Football>
    ```
    d. Type
    ```
    setupconsole
    ```
    as one word followed by return.
    You only need to do these four steps (a)-(d) once per session.
    e. Run your program by typing
    ```
    java EV3football
    ```
    with exactly that capitalization.
    f. If all is well, a window will appear. It will say **Stop**. Do not let this put you off. Click anywhere on the window to activate it. Now if you press, or press and hold, an arrow key the corresponding direction is displayed in the window. If

you press the space bar then **Kick** is shown in the window. When you've had enough, select **Quit** at the top left of the window and the program will terminate. If the window doesn't appear there is a connection problem so put your hand up and ask for help.

g. Leave the black Command window open because you will need it again

Now you have dealt with all the initialization issues and you need to edit the program so that the correct motors are moved when you hit the appropriate key. If you're more adventurous, you can also modify the program to read sensors and take appropriate action but keep things simple at first.

Changing the Program

You will need to edit the skeleton program using **jEdit** which is basically a text editor with extras for Java. Every time it has been edited you will need to save it and **compile** it (which turns it into something a computer can obey) before running it again. This is what to do.

1. Open the program in the jEdit editor

   a. Click on the Windows button at the bottom left of the PC again. This time select **All Programs → jEdit → jEdit**

   b. The partially completed Java program is called **EV3football.java** and is in the directory **Football** you downloaded earlier. Open this program in jEdit by clicking
   File → Open and navigating to **U:\ManWin\Football\EV3football.java**

   c. If all is well the program will load and you'll see it on the screen.

   d. You can leave this jEdit window open until the end of the session too so you don't need to do the steps above again.

2. The program looks quite long, but don't panic - you don't need to understand very much of this to make the robot work. Scroll down to the **run** method which is towards the end of the program at line 121 (if you can't see any line numbers pick **View → Toggle line numbers**). The first thing to do is to make the robot go forward and stop on command (see below). Start with stop which is on line 133.

3. **Every time** you edit your program you need to do the following to save and test it.

   a) In **jEdit** save your program by picking **File** then **Save** but do not change its name.

   b) Go back to the black Command window and compile it by typing
   ```
   javac EV3football.java
   ```

   c) If all goes well the
   ```
   U:\ ManWin\Football>
   ```
   prompt will appear again and you can run the new improved program by typing
   ```
   java EV3football
   ```
   again. If there is something wrong with your program you will get an error message and you will need to go back to editing your program to correct the problem. If you don't understand the error message put up your hand and ask what it means.

Controlling the wheel motors

The Java program already contains definitions of the two big motors that control the wheels; they are called **leftMotor** and **rightMotor** with that capitalization. The things you can do with them are make them go forwards, backwards or stop and you can set their speed. Setting the speed is important because you have to do that before the motor can do anything but stop. The range of speed available is 0 to about 700 degrees per second but the maximum speed depends on the battery's charge.

The sort of commands you can use are

```
leftMotor.stop();
rightMotor.forward();
leftMotor.backward();
rightMotor.setSpeed(200);
```

Once the robot has started moving it will keep going until you tell it to stop, that is why I suggested writing the **STOP** instructions first.

You can tell the robot to turn by either stopping one wheel and not the other or by setting one wheel to go forwards and the other backward. You will need to do some experiments to get this reasonably controllable.

Adding a Kicker

You don't need to have your kicker as a moving part but it tends to be more flexible if you do. In order to make this work you need to build something out of Lego, attach it to your robot, give it a motor plugged in to one of the spare motor ports (A or D) and finally change the program to make it kick. There are some pictures of the various components in the box to help you recognise the spare (medium) motor.

In order to change your program to make the robot kick you will need to add a definition of the motor that controls the kicker near the start of the **run** method you have been editing all along and then add some commands to make it move in the **KICK** section of the **run** method.

The definition of the kicker motor could look something like this

```
Motor foot = myRobot.getMediumMotor(Motor.Port.D);
```

This is slightly different to the definitions of **leftMotor** and **rightMotor** because they are both large, not medium, motors and the motor port is different. The motor port obviously depends on the one you have plugged the motor into on the real robot.

Once you have told the program about your kicker you can the same sort of commands you used for the big motors. So you can use things like

```
foot.setSpeed(100);
```

You don't have to call your kicker "foot". If you want to call it "boot", "kicker" or after your favourite footballer that is fine but you have to use the same name everywhere you refer to it. Whatever name you choose must be all one word and start with a lower case letter.

Making a sound

The robot is able to make simple sounds through an internal speaker. This can be useful for testing sensors (e.g., make the robot beep if a touch sensor is pressed – see below) or just for fun. As always you need to define it at the start of the run method like this

```
Speaker speaker = myRobot.getSpeaker();
```

Then you can use its **playTone()** method like this

```
speaker.playTone(1000,200);
```

Which play a 1000 Hz tone with a duration of 200 ms

Using Sensors

The Lego box contains quite a few sensors but the only one that is likely to be useful to you is **the touch sensor**. There is a picture of it in the box.

To use a touch sensor you must fix it securely on to your robot in a position where it is most likely to be useful and connect it to one of the **Sensor ports** using a black cable. Be careful not to mix sensor ports and motor ports up. The sensor ports are at the other end of the microcomputer to the motor ports and are labelled 1, 2, 3 and 4.

To program the robot to use the sensor you must define it at the start of the run method like everything else. The definition could look like this.

```
        TouchSensor touch = myRobot.getTouchSensor(Sensor.Port.S1);
```

Although only if it was plugged into sensor port 1.

To use it you have to test to see if it has been pressed. This will make the robot bleep if it bumps into something

```
        if (  touch.isPressed()  )  {
              Sound.playTone(1000,200);
        }
```

although replacing the instructions to bleep by something to make it go backwards might be more practical.

An alternative to the touch sensor is the ultrasonic distance sensor but with the arena as crowded as it is likely to be when the football competition starts this will probably be less useful. Nevertheless if you want to use it it can be defined by

```
        UltrasonicSensor bat =
              myRobot.getUltrasonicSensor (Sensor.Port.S2);
```

and used to test the distance (in metres) from the nearest object like this

```
        if (  bat.getDistance() < 0.3  ) {
              Sound.playTone(2000,50);
        }
```

although you don't have to call it "bat", or the touch sensor "touch" if you can think of better names.

At the end of the football match

Please

- Turn the robot off by pressing the light-grey button at the bottom left of the screen, navigating to the **Yes** option on the screen using the right button, and confirming with the central dark-grey button.
- Remove any sensors, motors etc. you have attached to the robot and put them back in the box.
- Unplug all the cables and put them back in the box
- Unclip your microcomputer from the top of its chassis, put the chassis in the box and the microcomputer on top of the box
- Hand the box and microcomputer to one of the demonstrators or technicians to put away.