

CISC 3415, Project-1 (6-pt)

1 Description

This project involves writing control programs for a simple robot, in fact for the Create, the education/research version of the Roomba robot vacuum cleaner.




All of the projects that we will do with this course will use the Player system. This provides an API for talking to robots — it reads data from sensors and writes commands to the robot.

This term with a completely online setup, we will exclusively use the companion package Stage, which provides a simulated Create robot and an environment for the robots to run in. That is, we will not be using the physical robots, and you have to download and make use of the virtual machine with Player/Stage preinstalled (see class website for detail)

Today we will concentrate on learning about Player, using it to get the Create to do some simple things.

2 Login, get ready.

1. Login to your computer/virtual machine. The user name is student, the password is student.
2. The virtual machine runs Ubuntu, a flavor of Linux. Some of the things you have to do should be familiar from CIS 15/CISC 3110 where you should have used Unix.
3. If you didn't use Unix, you'll quickly get the hang of the few simple things we use it for.
4. The first thing to do is to open a terminal window.

- Just click on the  icon on the menu bar at the left of the screen.
- You can also get hold of the terminal via:
Applications > Accessories > Terminal
- In fact, open two of these windows (or open another tab, on Windows, you can press Ctrl+Alt+T to open the first Terminal and Ctrl+Shift+T to add a new tab).

3 Play with Player and Stage

1. Most of our interaction with Ubuntu will be through the terminal windows. If you don't find the project1 folder on your Desktop, download and copy the project1.tar file onto the Desktop, then in one of the terminal window, you type the following to extract the tar file (a new project1 folder will be created. In general, you might want to do this for all projects to start everything anew):

```
cd ~/Desktop/  
tar xvf project1.tar
```

2. Now start by getting to the right place in the file system. To do that type:

```
cd ~/Desktop/project1
```

in both terminal windows.

3. In one terminal, type:

```
player world.cfg
```

This should pop up a square window labelled **Player/Stage: ./world.world** which contains a blank world with a grey dot at the center. This is the simulated world in which your robot will operate.

Note: You can grab and drag the grey dot (the simulated robot) around with your mouse to move it to any new position. Also you can go to the main menu and toggle on or off [View | Show trails] to display/clear the traces of your robot's movement. Select [File | Reset] to reset the world (robot returns to the origin), so you don't have to relaunch the stage for each test run.

4. Now, in the second terminal, type:

```
./build roomba-roam
```

This compiles the program roomba-roam.cc which is listed later in these notes.

5. If you type ./build roomba-roam.cc you'll get an error.

6. Once the program is compiled, run it using:

```
./roomba-roam
```

A stream of text should invade the window, and the robot should start moving, though it doesn't go far. That's the behavior of the sample code which is the starting point of your project-1.

4 The programming challenge

1. Your first task for today is easy to state — make the Create drive in a square, 3 meters on a side, turning *clockwise* at each corner.
2. To do that you have to write a C++ program which makes function calls that talk to the robot hardware. The functions are provided by Player.
3. To get you started, you have the program roomba-roam.cc, which you can also find at the end of these instructions.

4. To edit the program, type:

```
gedit roomba-roam.cc &
```

into one of your terminal windows. This starts up an editor which you should find pretty easy to use — you can do all you need to do from the menu bar at the top of the editor window.

Of course, if you have a preferred Unix editor, you can use it.

5. The & at the end of the command makes the program run in the background so that you don't

need to close the editor to continue to use the terminal.

6. This robot has a *differential drive*, which means that you can make it go forwards and backwards, and you can also make it turn.
7. To compile the program, as before you type:

```
./build roomba-roam
```

into one of your terminal windows. This will create a controller called roomba-roam.

8. To run your controller, you go back and do exactly what you did in Section 3.
9. When the robot runs in a square, save the program as **(your-names)-proj1.cc**.

The **(your-names)** should be the family names of the students in the group (in an alphabetic order). So, if I was working in a group with Reid and Hart, my project would be called **hart-reid-xiang-proj1-part1.cc**. *Alternatively*, you can name your file with your group #, and include the names of your group members who contributed to the project as comments in the front of the source code.

10. Take a copy of the program with you and email it to me. (**Important:** For each project, please send all your programs in **one** mailing. You could choose to send one zip file)

5 Some hints

1. Note that a positive value of `turnrate` will make the robot turn to its left. A negative value of speed will make the robot go backwards.
2. You might want to start by making the robot drive three meters in a straight line.
3. Then make the robot turn through 90 degrees.

6 The analysis challenge

This is the part 2 of this project. In writing, answer the following questions:

1. If you run the same program multiple times, would the robot follow exactly the same trajectories without error?
2. If you modify the program only in the turning direction so that it turns *counter clockwise* around the square. Suppose all speeds and durations remain the same, would the robot still make a perfect square?
3. Now, imagine this program is run on a real Create robot, would you achieve similar trajectories? What about the counter-clockwise squares? Give some possible reasons to your statement.
4. You may include this part of the answer in the body of your email.
5. When you have had enough, close the square window to stop the simulation.

```

/*
 * roomba-roam.cc
 *
 * Sample code for a robot that is suitable for use with the Roomba
 * and Create.
 *
 * Based on an example provided by Monica Anderson and Jeff Forbes,
 * via Carlos Jaramillo, and changed to (hopefully) make it easier to
 * understand.
 *
 * Modified:    Simon Parsons
 * Date:        15th June 2009
 * Last change: 20th September 2011
 */

#include <iostream>
#include <cstdlib>
#include <libplayerc++/playerc++.h>

int main(int argc, char *argv[])
{
    using namespace PlayerCc;

    // Set up proxy. Proxies are the data structures that Player uses to
    // talk to the simulator and the real robot.

    PlayerClient robot("localhost");
    Position2dProxy pp(&robot,0);          // The 2D proxy is used to send
                                           // motion commands.

    int timer = 0;                        // A crude way to time what we do
                                           // is to count.

    // Allow the program to take charge of the motors (take care now)
    pp.SetMotorEnable(true);

    // Control loop
    while (true)
    {
        double turnrate, speed;

        // Increment the counter.

        timer++;

        // Read from the proxies.
        robot.Read();

        // First make the robot go straight ahead, then make it turn, and
        // finally make it stop.
    }
}

```

```

    If (timer < 30) {
        speed = 0.1;
        turnrate = 0;
    }
    else
    if ((timer >= 30) && (timer < 60)) {
        speed = 0;
        turnrate = 0.1;
    }
    else{
        speed = 0;
        turnrate = 0;
    }

    // Print out what we decided to do?
    std::cout << "Speed: " << speed << std::endl;
    std::cout << "Turn rate: " << turnrate << std::endl << std::endl;

    // Send the motion commands that we decided on to the robot.
    pp.SetSpeed(speed, turnrate);
}
}

```