Prep C
Player Stage activities

Dr. Madhav Rao

Revision Date: July 22, 2016

Introduction

In this tutorial, you will write a robot simulation program, which will move random positions and coincide with the wall. The random positions are detected with an optical encoder, which determines the position moved with respect to robots initial position. Your task is to print the trace positions both in local (robot) coordinates and global coordinate system.

Task 1: Robot local frame coordinates

Download the *move.tgz* file from LMS or dropbox in *clab* directory. The tgz file is a tar-gunzip compressed file. You need to untar the file by running the following command.

```
tar -xvzf move.tgz
```

Save the following code in *move.c* file.

```
// Include header files
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <libplayerc/playerc.h>
//#include <playerc.h>


// Random numbers generated for random walk  movement by robots
double
randomInt(int low, int high)
{
return low + (high - low) * (rand()/(RAND_MAX * 1.0 ));
}


// Main function for the program
int
main(int argc, const char **argv)
{
playerc_client_t *client;
playerc_position2d_t *position2d;
int cycle, index=0;
double dist,angle,fidAngle = 0,lineAngle=0, fidDist=0, prevYaw=0,posAngle=0;

// Create a client and connect it to the server.
client = playerc_client_create(NULL, "localhost", 6665);
if (0 != playerc_client_connect(client))
return -1;

// Create and subscribe to a position2d device.
position2d = playerc_position2d_create(client, 0);
if (playerc_position2d_subscribe(position2d, PLAYER_OPEN_MODE))
return -1;

// Initiating random walk movement
if (0 != playerc_position2d_set_cmd_vel(position2d, randomInt(0.1,1)
,randomInt(0.1,1),DTOR(randomInt(-20,20)) ,1))
return -1;

fprintf(stdout, "robot random positions \n");

//looping in random positions
while(1)
{

// Wait for new data from server
playerc_client_read(client);

fprintf(stdout, "X: %3.2f, Y: %3.2f, Yaw: %3.2f \n",
position2d->px, position2d->py, position2d->pa);

// Random walk is continued till finding first marker
if (0 != playerc_position2d_set_cmd_vel(position2d, randomInt(0.1,1)
,randomInt(0.1,1),DTOR(randomInt(-20,20)) ,1))
return -1;

usleep(1000);
}

playerc_position2d_unsubscribe(position2d);
playerc_position2d_destroy(position2d);
playerc_client_disconnect(client);
playerc_client_destroy(client);
return 0;
}
```

Now put the following lines in your *makefile* in the current activity folder.

```
INC_PATH = -I/usr/local/include/player-3.0

LIB_PATH = -L/usr/local/lib64 -lplayerc -lm -lz -lplayerinterface -lplayerwkb -lplayercommon

move: move.c
    gcc -o move  $(INC_PATH) move.c $(LIB_PATH)
```

```
clean:
    rm -rf move
```

For 32 bit system users, you need to edit lib64 to lib in the makefile. Try to compile the program using the following command:

```
make all
```

and run your code on player-stage model. The player stage model works by two process: (process no. 1 communicating to process no. 2). Hence you have to open two terminals. In one terminal run the following command:

```
player move.cfg
```

In other terminal run the following command:

```
./move
```

Check the movement of the robot in your simulation GUI. Understand the above code and make sure that you can explain to the instructor while giving a demo. Once understood, your next activity starts.

Task 2: Global frame coordinates

Note that you get positions (x, y and yaw) with respect to robot starting position (-6.0, -6.0 and 0). Your task is to print the movement in global coordinates. Transforming from robot-frame-coordinates to global-frame-coordinates can be done using trignometry calculations. Check the geometry in your rough paper and modify the program in *move.c* file.

Task 3: iRobot Create

Follow the instructions to get your iRobot started.

- Attach BAM module to iCreate and turn ON. Note that there are four characters written in marker on the BAM module. Please write these characters down for use in the next step. Turn the iCreate on by pressing the leftmost button. The light next to the button should blink. Then a red light on the BAM module should blink.
- Connect BT usb to desktop and test

  ```
  [madhav@localhost ~]$ /usr/sbin/hciconfig
  hci0: Type: USB
  BD Address: 00:50:B6:80:31:82 ACL MTU: 1017:8 SCO MTU: 64:0
  UP RUNNING PSCAN ISCAN
  RX bytes:90050 acl:3014 sco:0 events:1469 errors:0
  TX bytes:34231 acl:2249 sco:0 commands:82 errors:0


  [madhav@localhost ~]$ hcitool scan
  Scanning ...
  00:1A:6B:79:05:24 DAVIDPETTIETTE
  00:1C:A4:0C:92:64 Josh
  00:12:D2:03:4B:33 T-Mobile Dash
  ```

```
00:1C:62:5B:29:5A Jewel 1906
00:17:D5:79:92:8D PLS-M5
00:0A:3A:26:48:7F Element Serial
00:02:72:07:FC:92 EE121a
00:16:B8:3D:CC:73 Z525
```

Find your MAC address for the next step. Run hcitool scan and find the address that ends in the 4 characters marked on your BAM module.

- Start and verify serial/BT connection between desktop and iCreate. Verify no current rfcomm device

  ```
  [madhav@localhost ~]$ sudo rfcomm
  ```

- start connection using the MAC address of *your* BAM module

  ```
  [madhav@localhost ~]$ sudo rfcomm bind 0 00:0a:3a:26:48:7f

  [madhav@localhost ~]$ sudo rfcomm
  rfcomm0: 00:0A:3A:26:48:7F channel 1 clean
  ```

- Test your robot with playerjoy application In one window..

  ```
  [madhav@localhost ~]$ player roomba.cfg
  * Part of the Player/Stage/Gazebo Project [http://playerstage.sourceforge.net].
  * Copyright (C) 2000 - 2006 Brian Gerkey, Richard Vaughan, Andrew Howard,
  * Nate Koenig, and contributors. Released under the GNU General Public License.
  * Player comes with ABSOLUTELY NO WARRANTY. This is free software, and you
  * are welcome to redistribute it under certain conditions; see COPYING
  * for details.
  Listening on ports: 6665
  ```

  In another window..

  ```
  [madhav@localhost ~]$ playerjoy
  Connecting to Player at localhost:6665 - calling connect
  done
  playerc error : got NACK from request
  Success
  Failed to open joystick: No such file or directory
  Falling back on keyboard controlReading from keyboard
  ---------------------------
  Moving around:
  u i o
  j k l
  m , .
  q/z : increase/decrease max speeds by 10%
  w/x : increase/decrease only linear speed by 10%
  e/c : increase/decrease only angular speed by 10%
  anything else : stop
  ```

  New message appear in player window when playerjoy is started

```
accepted client 0 on port 6665, fd 5
Opening connection to Roomba on /dev/rfcomm0...Done.
warning : Unhandled message for driver device=16777343:6665:4:0 type=3
subtype=2 len=1
```

- To quit playerjoy, preess Ctl-C

  ```
  [player window] closing connection to client 0 on port 6665
  ```

  Note: once rfcomm device has been used, rfcomm command shows this

  ```
  [madhav@localhost ~]$ rfcomm
  rfcomm0: 00:0A:3A:26:48:7F channel 1 closed
  ```

- to delete and restart

  ```
  [madhav@localhost ~]$ sudo rfcomm release 0
  [madhav@localhost ~]$ sudo rfcomm
  rfcomm0: 00:0A:3A:26:48:7F channel 1 clean
  ```

Task4: Client program to run roomba

Create roomba.cfg with the following lines as shown below:

```
driver
(
  name "roomba"
  provides ["position2d:0" "power:0" "bumper:0"]
  port "/dev/rfcomm0"
  safe 1
)
```

Use the same *move.c* file which you used for simulation. You may need to add *usleep(1000)* function in the loop after you set the velocity.

Demonstration

Make sure your are in the *move* directory and do 'ls'.