# Introduction

Raspberry Pi 3 + PS3 robots, which passes data to/from a central server over WiFi. The Raspberry Pi 3 controller must not be dependent on the server for basic operation. PS3 controller will bind directly to the Raspberry Pi 3. Non-blocking message/data passing is necessary to prevent lagging or unresponsiveness.

## Methods

Use socket communication (TCP/IP) between the individual robots and the server. The server will perform actions such as keeping score, directing individual/global commands, gathering telemetry data, and make decisions to control gameplay.

There’s lots of resources on socket programming:

<http://www.linuxhowtos.org/C_C++/socket.htm>

<http://www.thegeekstuff.com/2011/12/c-socket-programming/>

Try to avoid hard coding IP’s. Use fully resolved DNS names instead (roboticscomp.ad.unb.ca for example).

## Server/Client process

* Upon powerup, a Raspberry Pi robot will tell the server that it is online, and inform the server of its IP and any other relevant information. Update: client IP doesn’t matter, socket will determine.
* Server will keep track of which robots are online.
* May want a heartbeat (check if robot is still online) every so often. Master system could run client, connect to small socket (server) on Raspberry Pi for “alive” message or debug info.

This code may be just what we need.

<http://cs.smith.edu/dftwiki/index.php/Tutorial:_Client/Server_on_the_Raspberry_Pi#Server_code>

modified myClient to: Point to raspberryPi/Linux server IP. End transaction with (-1) instead of (-2), which causes myServer to exit().

Single socket program. Setup myClient code to push variables needed to server, and read any data required from server. Terminate by sending ‘-1’ from myClient. myServer then waits for new data. myClient will need to check if port is already in use. If so, wait a few ms (random?), then retry (loop).

## Interprocess Communication

Evaluate:

1. Sockets
2. Pipe
3. Message queue
4. Semaphores
5. Shared memory (requires synchronization)
6. Mpi4py (message passing)
7. 0MQ (zeroMQ)

0MQ looks promising.

## Apr 27, 2016 zeroMQ (0MQ) messaging library

Setting up 0MQ: <http://www.pihomeserver.fr/en/2014/01/15/raspberry-pi-home-server-le-message-broker-zeromq-pour-lier-vos-machines/>

<http://conoroneill.net/running-zeromq-with-node-js-on-raspberry-pi/>

Newest version is zeromq-4.1.4.tar.gz

Mkdir ~/zeromq

wget http://Download.zeromq.org/zeromq-4.1.4.tar.gz

Looks like libsodium 1.0.6 had issues, so these old examples used 1.0.3. Since 1.0.10 is now available, I’m going to try to use that version.

**OK, did the install but python could not find zmq. It appears that there’s already a zmq package:** [**http://www.anites.com/2013/12/zmq-messaging-system-lala-pi.html**](http://www.anites.com/2013/12/zmq-messaging-system-lala-pi.html)

Installed and working (tested pairclient and pairserver).

Great website with 0mq examples: <https://learning-0mq-with-pyzmq.readthedocs.org/en/latest/pyzmq/pyzmq.html>

## Network Credentials (resolved, requires update to interfaces and mpa\_supplicant.conf file)

UNB requires authentication to login to the wireless network.

UNB uses WPA2-Enterprise with PEAP security:

<http://www.unb.ca/its/students/smartphone-tablet/blackberry/wireless/index.html>

## Data Structure (Proposed)

### Server Introduction

Server will listen for information from Clients (robot detects a RFID tag, for example). It will process and track data (Robot fuel level, for example), and both automatically (out of fuel) and manually (ex: pause all robots) send data to robot(s). Manual entry may be via a GUI or text based input.

### Client (Robot) Introduction

Robot must not be blocked by sending or receiving data. Priority is for uninterrupted user control. If the server fails, or fails to respond quickly, the robots must not be effected.

Robot Commands

|  |  |
| --- | --- |
| **Command** | **Action** |
| 00 | Pause (disable all motors) |
| 01 | Resume (enable all motors) |
| 02 | Disable Drive motors |
| 03 | Enable Drive motors |
| 99 | NOP |

### Bluetooth and PS3 Controller

## Apr 26, 3016

Continuing on using the instructions for Raspberry Pi 2 as a starting point.

Default Jessie (Debian) install includes Bluetooth.

pi@raspberrypi:~ $ sudo service bluetooth status

● bluetooth.service - Bluetooth service

Loaded: loaded (/lib/systemd/system/bluetooth.service; enabled)

Active: active (running) since Tue 2016-04-26 18:24:16 UTC; 29min ago

Docs: man:bluetoothd(8)

Main PID: 650 (bluetoothd)

Status: "Running"

CGroup: /system.slice/bluetooth.service

└─650 /usr/lib/bluetooth/bluetoothd

Apr 26 18:24:16 raspberrypi bluetoothd[650]: Bluetooth daemon 5.23

Apr 26 18:24:16 raspberrypi systemd[1]: Started Bluetooth service.

Apr 26 18:24:16 raspberrypi bluetoothd[650]: Starting SDP server

Apr 26 18:24:16 raspberrypi bluetoothd[650]: Bluetooth management interface ...d

Apr 26 18:24:16 raspberrypi bluetoothd[650]: Sap driver initialization failed.

Apr 26 18:24:16 raspberrypi bluetoothd[650]: sap-server: Operation not permi...)

Apr 26 18:39:36 raspberrypi systemd[1]: Started Bluetooth service.

Hint: Some lines were ellipsized, use -l to show in full.

And the Bluetooth address can be displayed with ‘sudo bluetoothctl’

pi@raspberrypi:~ $ sudo bluetoothctl

[NEW] Controller B8:27:EB:89:3E:90 raspberrypi [default]

[bluetooth]#

And you can scan for BT devices:

pi@raspberrypi:~ $ sudo hcitool scan

Scanning ...

This may be useful:

<https://www.piborg.org/rpi-ps3-help>

*Actually it works perfect*. Only difference is that the ‘connect mac\_address’ step wouldn’t work, so skip and go straight to **trust**. Reboot and the PS3 controller will now connect, and jstest output works.

Also found a good example of Python socket programming: <http://www.binarytides.com/python-socket-programming-tutorial/>. This may be useful, otherwise may need to port the existing Raspberry Pi Demo Robot code to C/C++.

Default python code is causing the RPi 3 to reboot upon start. Need to debug.

## Apr 27, 2016

Running script causes reboot:

pi@raspberrypi:~/Demo\_Robot $ sudo python PS3\_robot1\_reverse\_buttons3.py

Attempting to pair PS3 controller via USB...

hci0: Type: BR/EDR Bus: UART

BD Address: B8:27:EB:89:3E:90 ACL MTU: 1021:8 SCO MTU: 64:1

UP RUNNING

RX bytes:1163171 acl:19703 sco:0 events:61 errors:0

TX bytes:1903 acl:13 sco:0 commands:48 errors:0

sudo: ./sixpair: command not found

This is likely caused because I did not have the interface board connected.

Connected interface board, script now runs fine, no reboot. Can press PS3 button after reboot (slow LED blink on interface board), pairs and starts OK.

### Apr 28, 2016

Hmm, this is complicated.

Could use the stream socket examples (myClient/myServer), where myClient is called as a python subprocess or os.system call from the main robot python script. Output of myClient (received from Server) is saved to variable, then processed.

This will be the basis:

<http://cs.smith.edu/dftwiki/index.php/Tutorial:_Client/Server_on_the_Raspberry_Pi>

But the client is blocking. If the port is busy it will wait. If the port isn’t open, it will immediately exit (that’s ok). But what to do if the port is busy? Crap, can’t have the robot script wait for a reply, EVER.

Another option is to have the comm/client write output commands to a file, and have the python (robot) script check if the file exists. Lock, read, delete. Comm client will lock, write.

Robot sends command

C++/G++ file I/O

<http://www.raspberry-projects.com/pi/programming-in-c/file-input-and-output/working-with-files>

Setting up RAM drive: <https://www.domoticz.com/wiki/Setting_up_a_RAM_drive_on_Raspberry_Pi>

## 0MQ Review

Installation: <http://www.anites.com/2013/12/zmq-messaging-system-lala-pi.html>

<https://library.oreilly.com/book/0636920026136/zeromq/toc>

1. Can’t use REQ/REP, blocking (synchronous)?
2. PUB/SUB is one direction. If the robot or server isn’t listening, the info is lost. Ensuring that connections are not made in rapid succession, and that calculations/database accesses are not time consuming, will improve performance and minimize data loss. For performance, multiple Publishers and Subscribers can be used.

# Robot Data Comm Structure (Robot to Server / Server to Robot)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IP | DATA | | | | EOT | Description |
| Robot/  Source IP | Data 1 (int) | Data 2 (int) | … | Data 8 (int) | <EOT> = -1 |  |
|  |  |  |  |  |  |  |

**Examples**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| IP | DATA | | | |  |  |  |  | EOT | Description |
| 131.202.12.117 | 00 | 00 | 00 | 00 |  |  |  |  | -1 |  |

# Database

Should we use a local or global database for sharing data?

<http://raspberrywebserver.com/sql-databases/using-mysql-on-a-raspberry-pi.html>

### May 2, 2016

Installing MySQL.

**Root password is “energySHOULDERreally03”**

### May 5, 2016

Create user account:

mysql> CREATE USER 'ps3robot'@'localhost' IDENTIFIED BY 'RaspPS3Robot';

mysql> GRANT ALL PRIVILEGES ON temps.\* TO 'ps3robot'@'localhost';

mysql> FLUSH PRIVILEGES;

mysql> CREATE DATABASE PS3Robot;

mysql> USE PS3Robot;

mysql> quit;

mysql -u PSrobot –p (enter password)

CREATE TABLE command (name TEXT, cdate DATE, ctime TIME, enmotors TINYINT, pwml TINYINT, pwmr TINYINT, pwma1 TINYINT, pwma2 TINYINT, report TINYINT);

mysql> describe command;

+----------+------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+------------+------+-----+---------+-------+

| name | text | YES | | NULL | |

| cdate | date | YES | | NULL | |

| ctime | time | YES | | NULL | |

| enmotors | tinyint(4) | YES | | NULL | |

| pwml | tinyint(4) | YES | | NULL | |

| pwmr | tinyint(4) | YES | | NULL | |

| pwma1 | tinyint(4) | YES | | NULL | |

| pwma2 | tinyint(4) | YES | | NULL | |

| report | tinyint(4) | YES | | NULL | |

+----------+------------+------+-----+---------+-------+

Granting access to the database by the entire 131.202.12 subnet:

<http://stackoverflow.com/questions/8348506/grant-remote-access-of-mysql-database-from-any-ip-address>

mysql> GRANT ALL ON PS3Robot.\* TO ps3robot@'131.202.12.%' IDENTIFIED BY 'RaspPS3Robot';

## May 5, 2016

Figured out we can perform remote table insertion in MySQL, so may not need a command + PUB/SUB to upload commands to the robots.

# Example Table Structure (Command)

What should be included in the table?

* Name (Text)
  + Text field to store PS3Robot “name”. May be blank.
* Date (DATE)
  + Log date command was sent.
* Date (TIME)
  + Log time command was sent.
* Enable/Disable motors (integer)
  + 0 : Disable all motors
  + 1 : Disable drive motors
  + 2 : Disable Aux motors
  + 3-99 (anything else) – Enable All motors
* Set PWM limit Drive motor Left (integer)
  + Value from 0-100. Values outside this range are ignored.
* Set PWM limit Drive motor Right (integer)
  + Value from 0-100. Values outside this range are ignored.
* Set PWM limit Aux motor # 1 (integer)
  + Value from 0-100. Values outside this range are ignored.
* Set PWM limit Aux motor # 2 (integer)
  + Value from 0-100. Values outside this range are ignored.
* Report Status (integer). Ask robot to report current status.
  + 0 : Do not Report Status
  + 1 : Report Status

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Date | Enable/Disable  motors | PWM  Left | PWM  Right | PWM  Aux#1 | PWM  Aux#2 | Report  Status |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

# Starting a subprocess from a Python script

Note, this can cause issues with sequential access, and is slower than running an “os.” command.

a=str(random.randint(1,10000))

subprocess.Popen(['/home/pi/test/comm1',a])

# Implementation Steps

1. Create python publisher script with argument input for port and data. Spawn using subprocess command. Exit after publish. Use second RPi as subscriber to ensure it works.
2. Ignore!
3. PUB directly from Robot.

## May 4, 2016

Finally realized that I could run a PUB directly from the PS3\_robot python script, and it works quite well.

At a 50Hz loop rate, memory usage on the SUB is very high (84%). Issue was the default sub\_client script used the “for nbr loop”. Removed, resolved high mem usage. CPU usage directly correlates to # messages/sec sent. At 1000 messages/sec, script takes <20%. This will never be needed in actual use, more like 1 or 2 messages/sec expected. Ran for about 15h (overnight), 49M messages published, no errors.

At a ~2Hz loop rate, CPU usage = 1%.

## May 24, 2016

Status update in prep for May 25th meeting:

Copied current test code to the “current code” directory”.

**PS3\_client4.py** - Drives robot paired to PS3 controller, writes out a PUB (zMQ) message approximately 2x per second (adjustable).

**insert2.py** - Local database setup, working. Local access working

**remote\_insert3.py** - Remote access not working (security/permission setting adjustment required).

**sub\_client4.py** - Listener (SUB) working on remote host.

# Security Recommendations

Change default Pi password to prevent unauthorized access.