

Xaxxon Project – Oculus Robot Documentation – User Manual

Version 1.0

August 11, 2017

Research Co-op for Dr. Shahram Payandeh

Written by Lestley A. Gabo

Contents

Inventory	3
Xaxxon Oculus Robot	3
3D Printed Platform	4
MALG board	4
Power LiPo3S.....	5
Servo Motors with Encoders SPG30 - 200K and 60K Gear Ratio	5
Limit Switches	5
Connecting the Batteries	6
Turning on the robot.....	6
Turning on the platform.....	7
Plugging in the MALG board	8
MALG board plugged in before turning on the robot.....	8
MALG board not plugged in before turning on the robot	8
Reconnecting the node server if MALG was not plugged in when robot turned on	9
Attaching the tablet	13
Opening the web application.....	14
References	17
Demonstration	17
Codes used	17

Inventory

Xaxxon Oculus Robot



Figure 1: Xaxxon Oculus Robot

3D Printed Platform

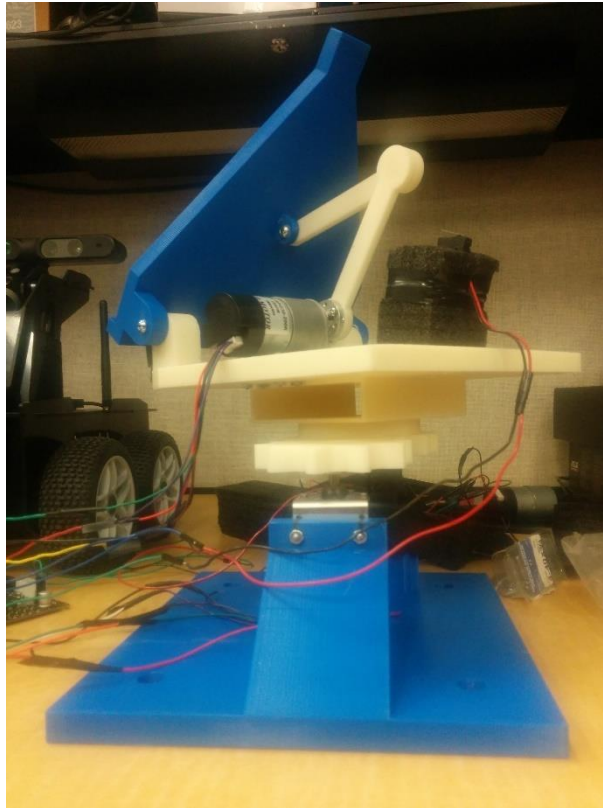


Figure 2: 3D Printed Platform

MALG board

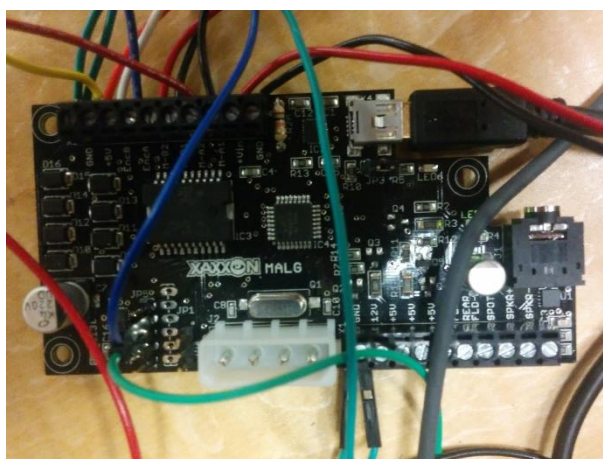


Figure 3: MALG board

Power LiPo3S

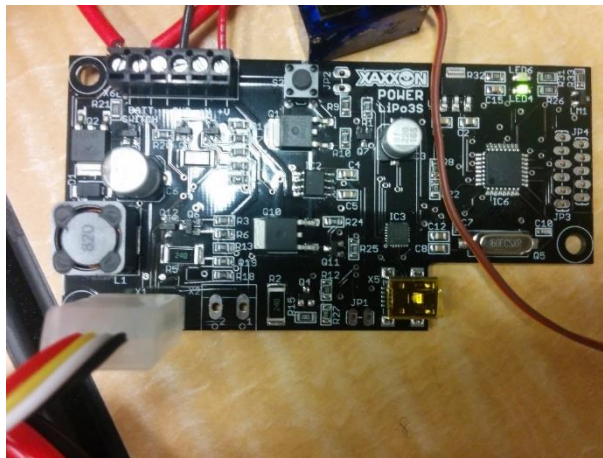


Figure 4: Power LiPo3S

Servo Motors with Encoders SPG30 - 200K and 60K Gear Ratio



Figure 5: Servo Motors with Encoders

Limit Switches

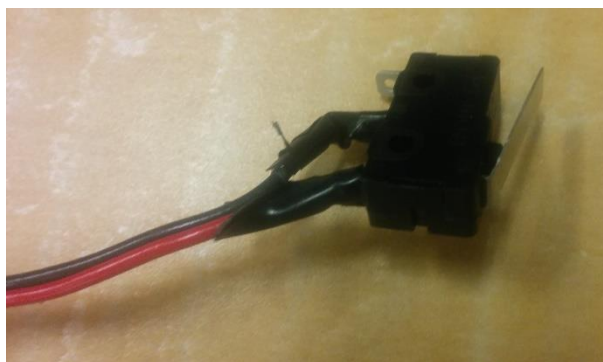


Figure 6: Limit Switch

Connecting the Batteries

Turning on the robot

To turn on the Xaxxon Oculus Prime robot (this will be called robot from now on), flip the robot on its front and look at its bottom.

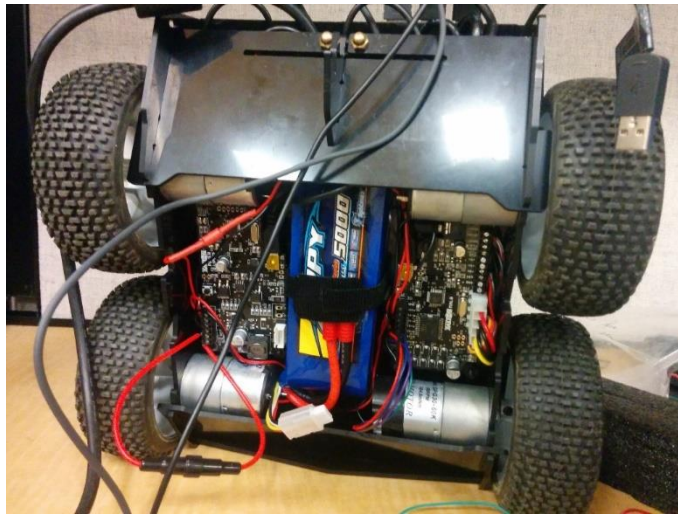


Figure 7: Powering on the robot

The blue bar in the middle is the battery. You need to connect this battery to the Xaxxon Power LiPo3S as seen on the left side of the battery in Figure 7.

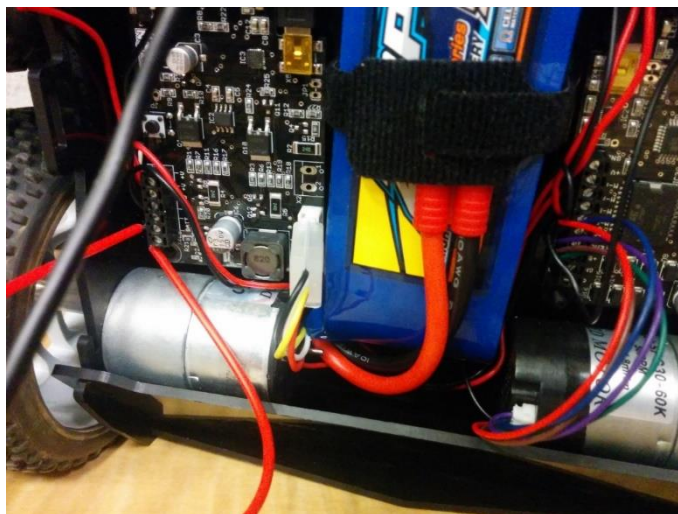


Figure 8: When plugging in the battery, BLACK IS BACK

When plugging in the battery to the robot, remember to make sure that BLACK IS BACK. This means the black wire is facing towards the back of robot. There are notches on the plug to guide the wires in, but it is possible to force plug the wires in the opposite way.

Turning on the platform

The MALG board by itself cannot power the servo motors therefore it needs an external power supply. This power is supplied by the Power LiPo3S which is connected to a 3 cell 11.1 V battery.



Figure 9: Power LiPo3S connected to battery

Plugging in the MALG board

MALG board plugged in before turning on the robot

It would be great if the MALG board is plugged in before you turn on the robot.



Figure 10: MALG board USB on the very right plugged in to the USB hub

MALG board not plugged in before turning on the robot

We want the MALG to be connected to the USB hub before turning on the robot because the node server checks if the port path at `/dev/arduino` is open. If the robot is turned on while the MALG is unplugged then the node server cannot open that port path so the web application Platform Controls won't be able to control the platform.

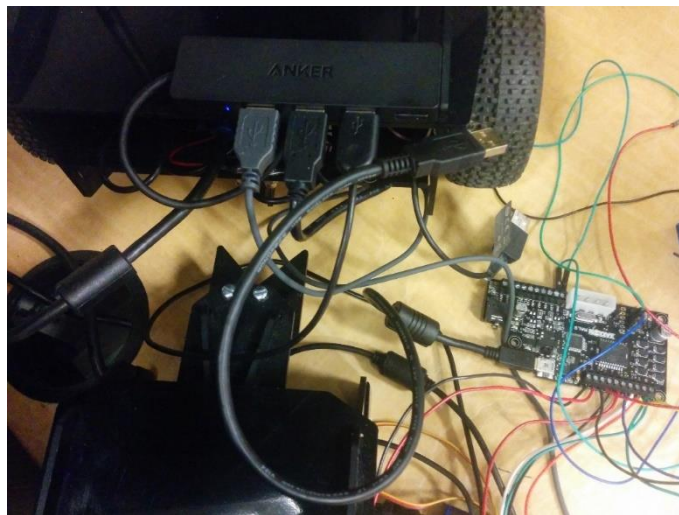


Figure 11: Unplugged MALG board


```
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/arduino
[STREAMING] Now streaming realtime logs for [server] process
```

Figure 12: Unplugged MALG board, no port path

Reconnecting the node server if MALG was not plugged in when robot turned on
We just need to reset the node server to make it find the MALG board.

Open the terminal and type in

`pm2 restart server`

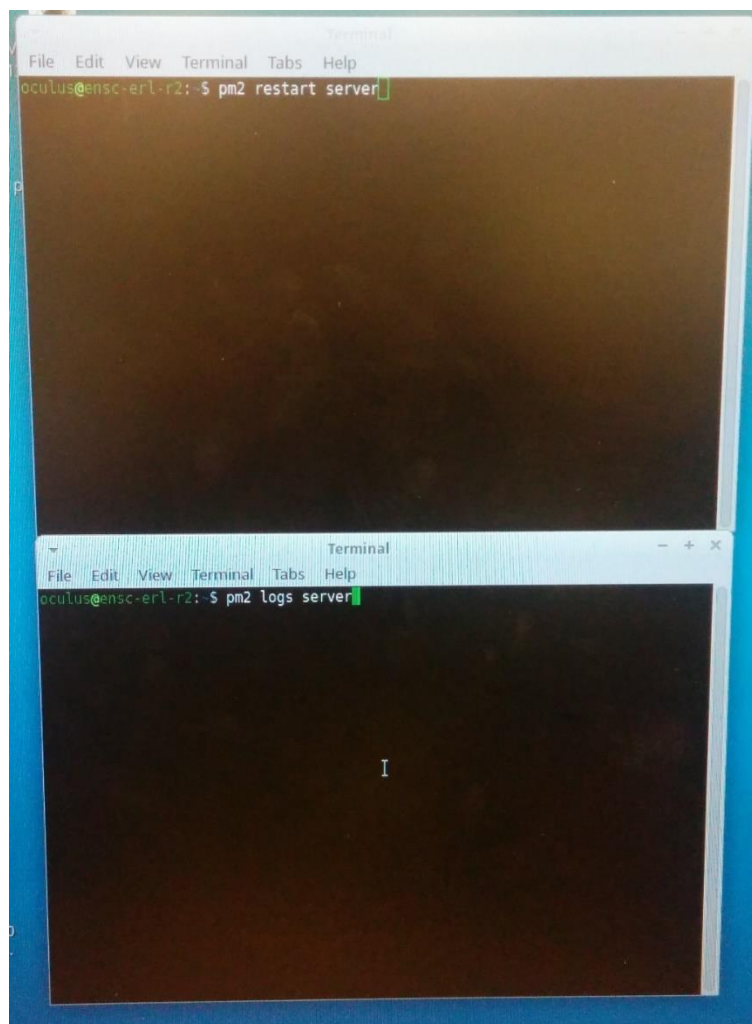
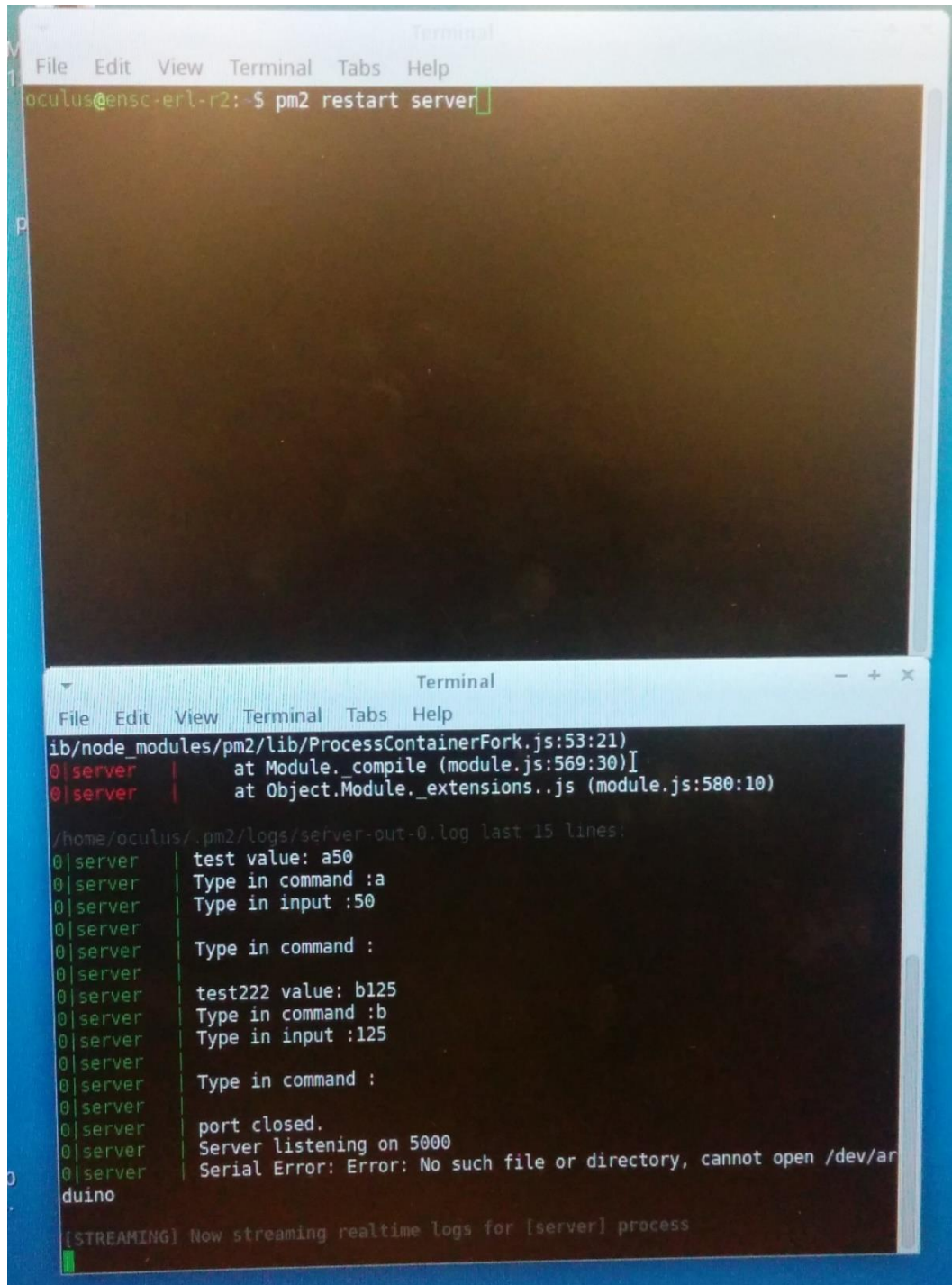


Figure 13: Restarting the node server

To check the console of the node server

Open the terminal and type in

`pm2 logs server`



The image shows two terminal windows. The top window is titled 'Terminal' and shows the command `pm2 restart server` being entered at the prompt `oculus@ensc-erl-r2:~$`. The bottom window is also titled 'Terminal' and shows the output of `pm2 logs server`. It displays error messages from the PM2 process manager and the last 15 lines of the server's output log.

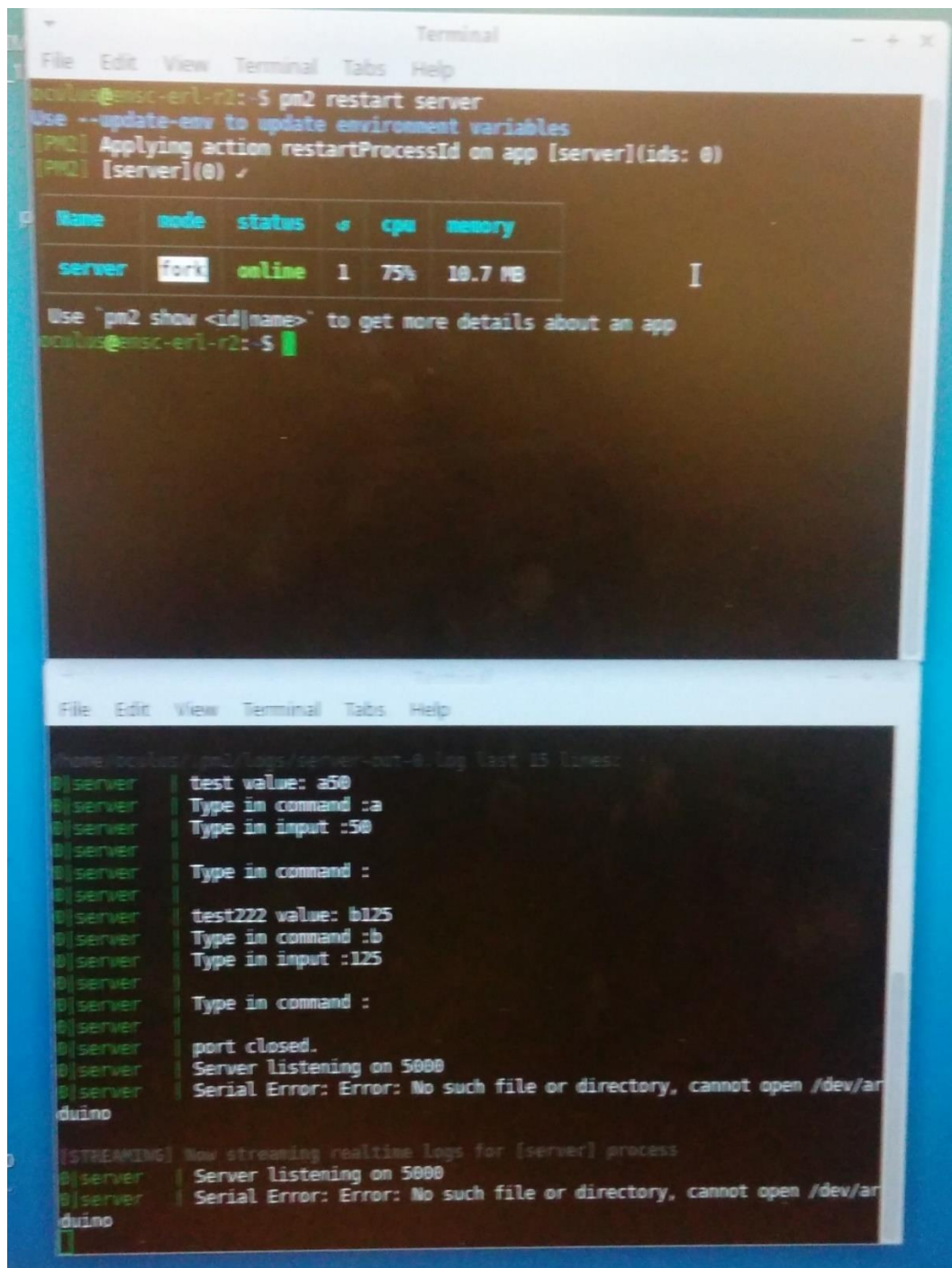
```
ib/node_modules/pm2/lib/ProcessContainerFork.js:53:21)
0|server | at Module._compile (module.js:569:30)
0|server | at Object.Module._extensions..js (module.js:580:10)

/home/oculus/.pm2/logs/server-out-0.log last 15 lines:
0|server | test value: a50
0|server | Type in command :a
0|server | Type in input :50
0|server |
0|server | Type in command :
0|server |
0|server | test222 value: b125
0|server | Type in command :b
0|server | Type in input :125
0|server |
0|server | Type in command :
0|server |
0|server | port closed.
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/ar
duino

[STREAMING] Now streaming realtime logs for [server] process
```

Figure 14: Checking the console log of the node server

If you restart the server without plugging in the MALG board, you will see the same port closed error.



The image consists of two terminal window screenshots. The top screenshot shows a terminal with the command `pn2 restart server` being executed. It displays a table with columns: Name, mode, status, id, cpu, and memory. The table contains one entry: 'server' with mode 'fort', status 'online', id '1', cpu '75%', and memory '10.7 MB'. Below the table, it says 'Use `pn2 show <id|name>` to get more details about an app' and shows the prompt `oculus@msc-erl-r2: $`. The bottom screenshot shows the last 15 lines of a log file. The log entries are: 'test value: a50', 'Type in command :a', 'Type in input :50', 'Type in command :', 'test222 value: b125', 'Type in command :b', 'Type in input :125', 'Type in command :', 'port closed.', 'Server listening on 5000', and 'Serial Error: Error: No such file or directory, cannot open /dev/arduino'. It also shows a '[STREAMING] Now streaming realtime logs for [server] process' message and repeats the 'Server listening on 5000' and 'Serial Error' messages.

```
oculus@msc-erl-r2: $ pn2 restart server
Use --update-env to update environment variables
[PMQ] Applying action restartProcessId on app [server](ids: 0)
[PMQ] [server](0) ✓



| Name   | mode | status | id | cpu | memory  |
|--------|------|--------|----|-----|---------|
| server | fort | online | 1  | 75% | 10.7 MB |

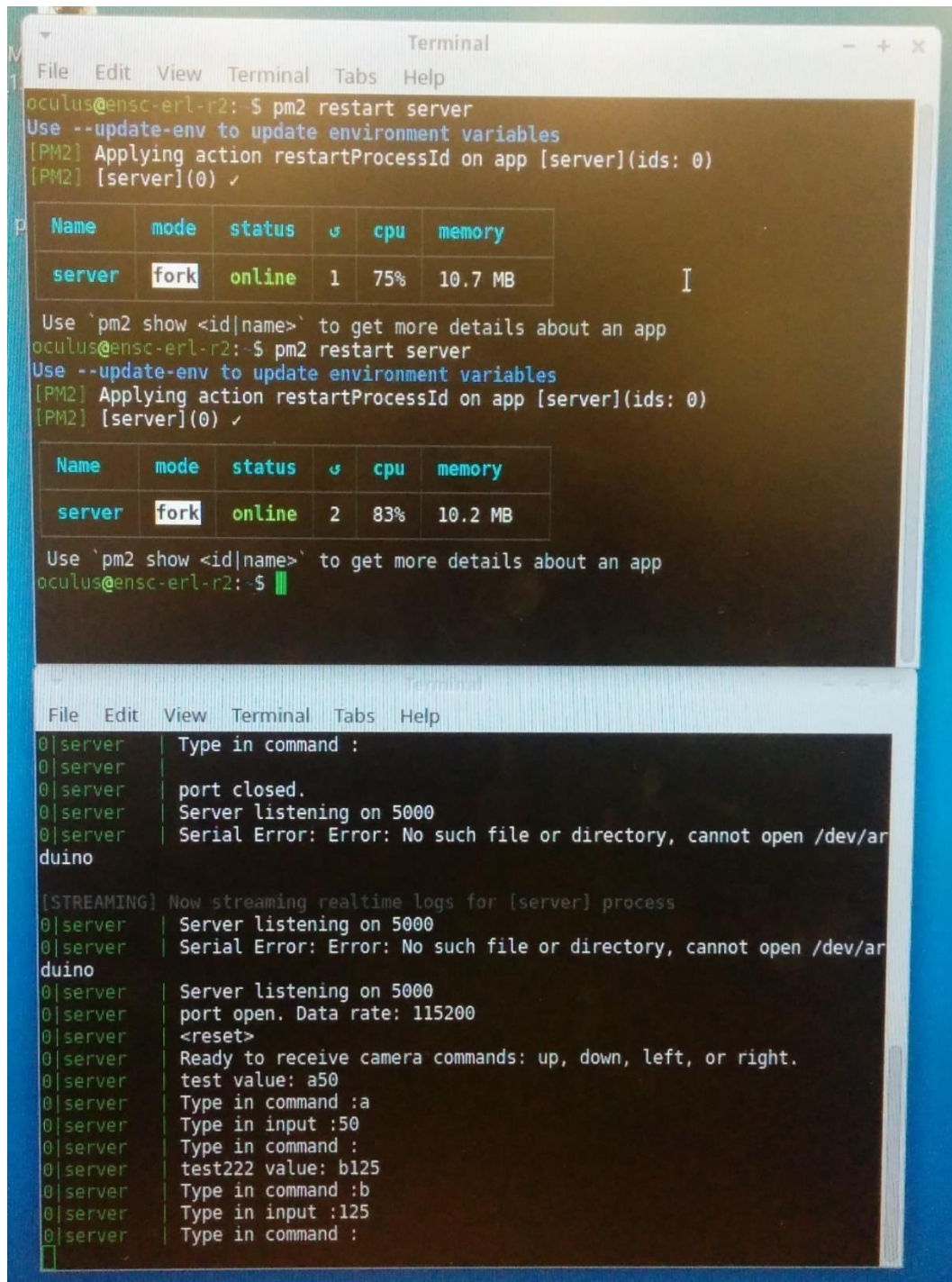


Use `pn2 show <id|name>` to get more details about an app
oculus@msc-erl-r2: $

/home/oculus/.pn2/logs/server-out-0.log last 15 lines:
0|server | test value: a50
0|server | Type in command :a
0|server | Type in input :50
0|server |
0|server | Type in command :
0|server |
0|server | test222 value: b125
0|server | Type in command :b
0|server | Type in input :125
0|server |
0|server | Type in command :
0|server |
0|server | port closed.
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/arduino
[STREAMING] Now streaming realtime logs for [server] process
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/arduino
0|server |
```

Figure 15: Restarting the server with MALG board unplugged

After plugging in the MALG board and restarting the server, the logs should show the port being open and the serial commands to the MALG being accepted.



The image consists of two terminal window screenshots. The top screenshot shows the command 'pm2 restart server' being executed. It displays PM2 status for the 'server' app, which is in 'fork' mode and 'online' status, using 1 CPU and 10.7 MB of memory. The bottom screenshot shows the server logs after the restart. The logs indicate that the server is listening on port 5000 and that the serial port is open with a data rate of 115200. The logs also show the server receiving camera commands and test values.

```
oculus@ensc-erl-r2:~$ pm2 restart server
Use --update-env to update environment variables
[PM2] Applying action restartProcessId on app [server](ids: 0)
[PM2] [server](0) ✓

Name      mode  status  ↕   cpu  memory
server    fork  online   1   75%  10.7 MB

Use `pm2 show <id|name>` to get more details about an app
oculus@ensc-erl-r2:~$ pm2 restart server
Use --update-env to update environment variables
[PM2] Applying action restartProcessId on app [server](ids: 0)
[PM2] [server](0) ✓

Name      mode  status  ↕   cpu  memory
server    fork  online   2   83%  10.2 MB

Use `pm2 show <id|name>` to get more details about an app
oculus@ensc-erl-r2:~$
```

```
0|server | Type in command :
0|server |
0|server | port closed.
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/ar
duino
[STREAMING] Now streaming realtime logs for [server] process
0|server | Server listening on 5000
0|server | Serial Error: Error: No such file or directory, cannot open /dev/ar
duino
0|server | Server listening on 5000
0|server | port open. Data rate: 115200
0|server | <reset>
0|server | Ready to receive camera commands: up, down, left, or right.
0|server | test value: a50
0|server | Type in command :a
0|server | Type in input :50
0|server | Type in command :
0|server | test222 value: b125
0|server | Type in command :b
0|server | Type in input :125
0|server | Type in command :
```

Figure 16: Restarting the server with MALG board plugged

Attaching the tablet

The servo motors are compensated with more power than necessary because it expects to turn with the weight of the tablet attached to the platform. If the platform is controlled without the tablet then the platform will turn up, down, left, or right very aggressively.

To secure the tablet to the platform we are using Velcro straps. This is just a temporary solution.



Figure 17: Tablet attached to the platform using Velcro

The tablet we are using is a Samsung Galaxy Tab which has a screen size of 10 inches, but any tablet will fit the platform. A smaller and lighter tablet would improve the performance of the servo motors.

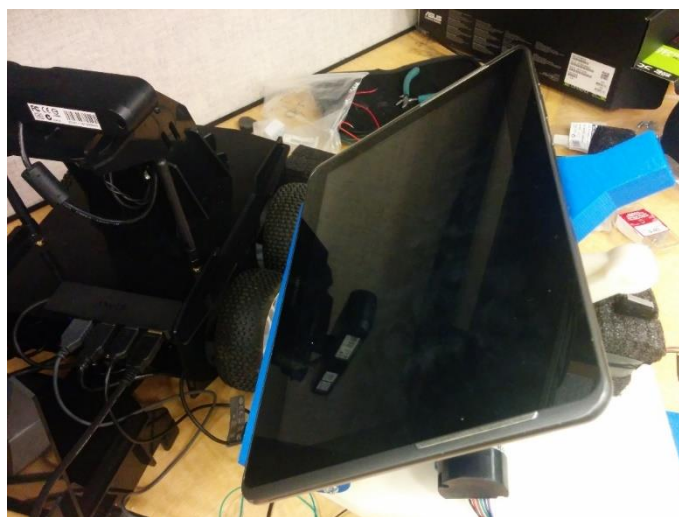


Figure 18: Tablet attached to the platform

Opening the web application

Now that the robot is on and the node server is connected to the MALG board, we open the Xaxxon web application to access our node server application. The robot's server automatically turns on when the robot turns on. We know that the server turns on when Chrome browser opens and shows the Xaxxon Oculus server browser page.

To access our own node server web application click launch controls when in the server page

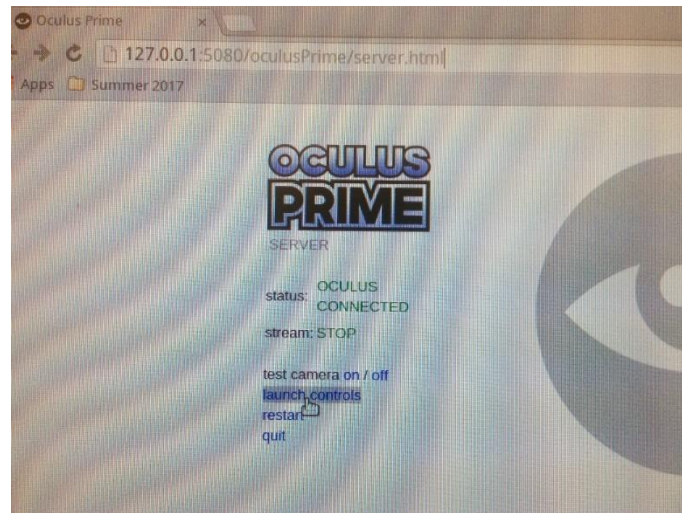


Figure 19: Xaxxon Oculus server web application, clicking launch controls

Then click on the wrench icon

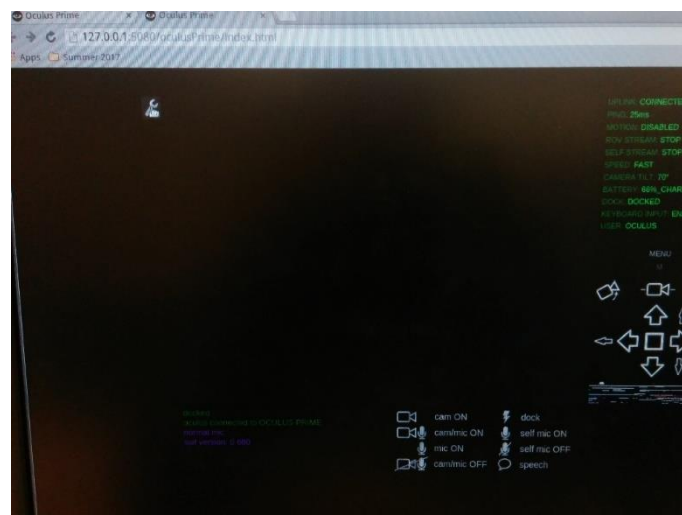


Figure 20: Xaxxon Oculus web application, clicking wrench menu icon

Then click test

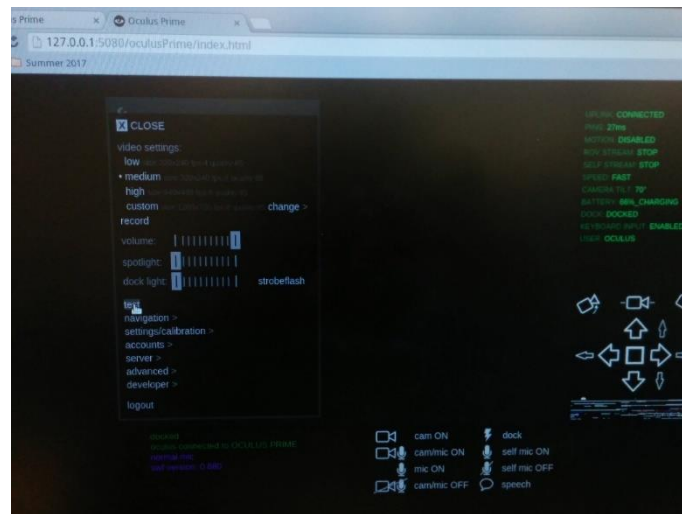


Figure 21: Xaxxon Oculus web application, clicking test

After clicking test, an iframe of our node server application will open.

Note: the iframe set up is not optimized, there is no close button even. One way to get out of the iframe is to click the top part of the iframe then drag it down and click the wrench menu icon. The best way to get out of the iframe is to refresh the page.

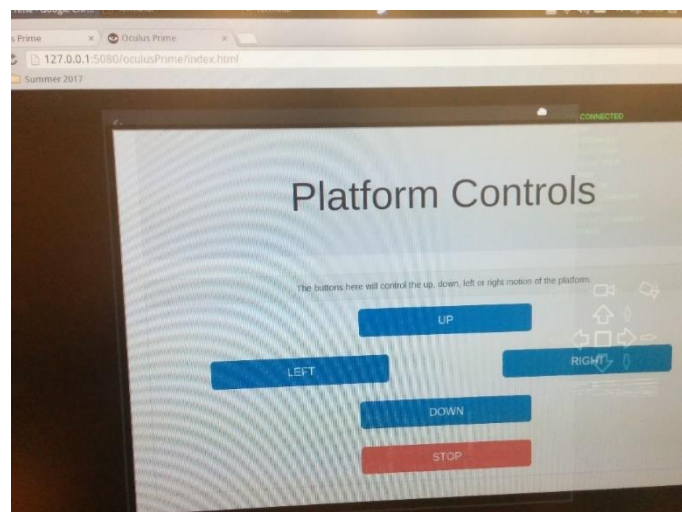


Figure 22: Xaxxon Oculus web application, with opened iframe of node server application

Another way to access the node server application is to open another tab/browser and go to URL address 'localhost:5000'. The localhost can be replaced with the current IP address of the robot. So, in another device, to access the Platform Controls and assuming we know the IP address, open a tab/browser and go to '207.23.183.201:5000'. The 207.23.183.201 is the static IP address of the robot.

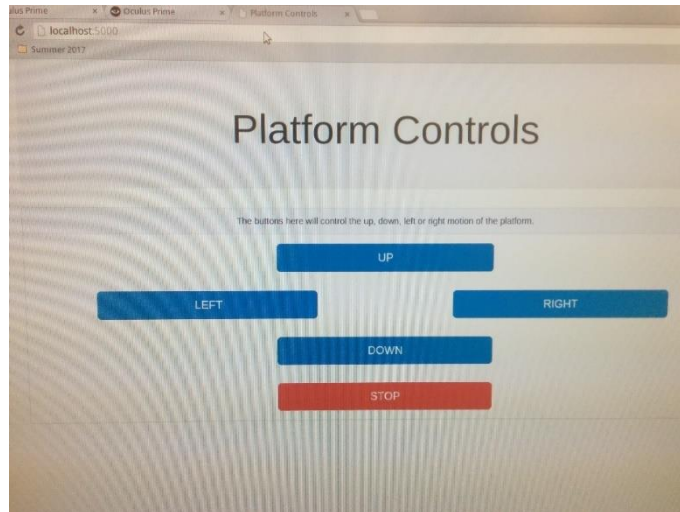


Figure 23: Platform Controls, our own node server application found at localhost:5000

References

Demonstration

Check out this short video for the demonstration of the project

<https://www.youtube.com/watch?v=ZmU-hsg7x38>

Codes used

All the code used in this project are in the GitHub account 'elderlyrobotics'

<https://github.com/elderlyrobotics>