MIRO Robot Manual for Running Code On-Board

Written by Ellie Sona in April 2019

# Introduction

This manual extends and modifies the MIRO operating manual written by James Zhu and modified by Sidharth Babu available at <https://github.com/ellieasona/RASL-MIRO_Sensor_React/blob/master/README.pdf> . The preparation and startup described in that manual are still accurate for running the code off-board, but this manual describes the preparation and how to run the code on-board. The package configuration and system for writing code described in that manual is still accurate, though the code has been modified and is available at <https://github.com/ellieasona/RASL-MIRO_Sensor_React>.

# Robot Information

The RASL lab has 2 robots – one has a collar and one does not. When connected to the vuDevice network, the robot with the collar has an IP address of 10.68.1.92, and the robot without the collar has an IP address of 10.68.1.94. the vuDevice password used with both robots is pbPv)-X3aGOMB. The MiRo app should be used to connect to Wi-Fi, though the robots will automatically connect to the vuDevice network. With the new, reflashed SD cards (currently installed on the robot without a collar), the root password for the robot is raslMIRO. With the old SD cards (currently installed on the robot with a collar), the root password is !amMIRO which is the default password for the robots. With both, root should be used as the SSH username. The robot without a collar is currently setup to be run default behavior on-board.

# Running code on-board

To run the code on-board, the on-board option should be selected in the ~/.profile file. The preparation described in the original operating manual is not necessary when running the code on-board. This is already done on both robots.

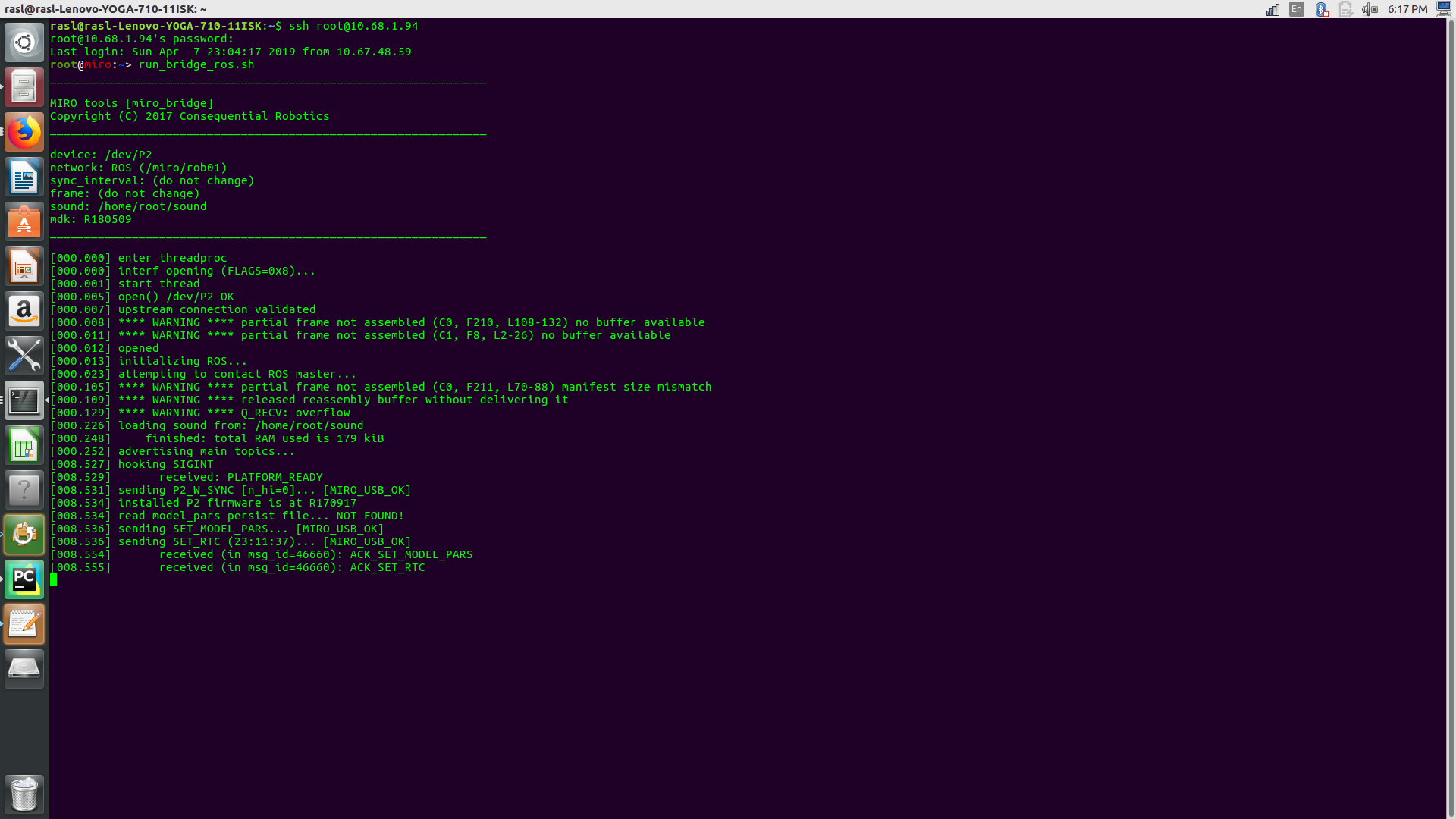
To automatically run roscore on startup, add the roscore command to the ~/bin/user\_ready.sh file where indicated. An appropriate user\_ready.sh file is available at <https://github.com/ellieasona/RASL-MIRO_Sensor_React/tree/master/bot%20files>. To run your start\_user.sh file when the bridge is turned on (done via the app with the Run/Stop button), add the command to run your start\_user.sh file under # start bridge, and be sure to send output to a log file. To mimic the behavior of tapping Run/Start on the app, run the command bridge\_control.sh toggle\_running. Note that the environment for SSH is very different from the Bluetooth daemon environment used for the app, so many times the code will work when you run the bridge\_control.sh toggle\_running command but not when you do essentially the same thing from the app.

This environment problem is partially solved by sourcing the appropriate files and setting the necessary environment variables in the start\_user.sh file, so be careful not to remove those from the file. However, this will not solve all issues. A log file is created from the output of start\_user.sh, and start\_user.sh creates a log file of the output from running the python script, so use those logs to debug any issues that arise. The start\_user.sh file currently on the robot is available at <https://github.com/ellieasona/RASL-MIRO_Sensor_React/tree/master/bot%20files>.

Place your code to run your python scripts at the bottom of the start\_user.sh file. Likely, you will replace the call to Initiator.py with whatever code you want to run. Be sure to keep the part of the command that directs output to a log file to aid with any debugging.

# Lessons learned from email support

Because I spent so much time communicating with Consequential Robotics over email about getting the code to run on-board, I learned a lot of information that isn’t readily available in MiRo documentation about how the robot works internally that could be helpful with future debugging that is catalogued below.

1. If there are errors when running the bridge, there is a log file available at /tmp/log/miro\_bridge.ros. Also check ~/.profile to make sure ROS master is set for the code to run on-board if there are bridge problems. More information is available at <https://consequential.bitbucket.io/Developer_Profiles_Off-board.html>
2. For debugging bridge problems, you can also take the shell off and look at the small red LEDs for guidance. When you start the bridge manually, you'll see LED2 starts to flash. If the bridge manages to contact the ROS master, LED2 will go solid after a few seconds, then the bridge is running. If it fails, LED2 will go out, and MIROapp with then show the bridge is not running when you refresh. More information on the LED indicators is available at <https://consequential.bitbucket.io/Demonstrator_Commissioning_Boot_And_Shutdown.html>.
3. To reset the robots to their factory settings and update the software, you can reflash the SD card. This will wipe any data, so be sure to transfer any necessary files to the laptop before you do this. Instructions to reflash the SD card are available at <https://consequential.bitbucket.io/Technical_Processors_Maintenance.html>. Note that the P2 processor must be updated when the P3 processor is, so if you switch between the old and new SD cards for the robots you must reprogram the P2. Instructions for that are also available at the above link.
4. ~/bin/user\_starting.sh starts the Wi-Fi, so check that file if Wi-Fi is not connecting properly
5. ~/bin/user\_ready.sh should have a call to start\_bridge.sh with or without a comment mark “#” before it. Adding and removing that comment mark is what happens when you press toggle autostart in the app.
6. You can manually start the bridge and see its output by running “run\_bridge\_ros.sh” in the terminal. This can be helpful when running the bridge the first time after changing settings to allow you to see all output of starting the bridge. If you get the message “USB INTERFACE IN USE”, this means that the bridge is already running. When the bridge has started properly, the console output looks like the image below. Note that this does not return, which is why the bridge is normally started in the background. The start\_user.sh script allows you to run the bridge manually in the background. 
7. There is a log file for what happens via the app available at /tmp/log/daemon.
8. The Bluetooth daemon does not operate properly when there is output to the console. A good way to fix this is sending output to a log file.
9. The version of the app must match the version of the MDK installed on the robot, though I have found that newer versions of the app tend to work better than older versions of the app regardless of MDK version. The MDK version is available at ~/mdk/mdk.release on the robots.
10. If the robot is moving around, whimpering, and not running your code, it is likely in demo mode. There are a few ways that the robot can get in and out of demo mode. You may have to try multiple methods to get it back to normal mode which is what is needed to run code
    1. Dip switches – there are white dip switches under the shell that should be set to binary zero for normal mode and 12-15 for demo mode. A full description of the dip switches and list of options for them is available at <https://consequential.bitbucket.io/Technical_Platform_Modes.html>
    2. App – the mode can be set in the app, though I have not had much success with this.
    3. SSH – You can change the mode by setting line 52 in ~/bin/start\_bridge.sh to say export MODE=demo or export MODE=normal according to your desired mode.
11. Sound files need to be configured properly in the correct folder. They should be in the ~/sound folder, and there should be sound files labeled sound.1, sound.2, and so on, a sound.ini file listing the sound files in the following format:

sound.1

sound.2

sound.3

for each of the sound files. Be sure that you have the same number of files listed in your .ini files as you do in the directory. There should also be a README and a license file. If there is an issue with your sound directory, when you run run\_bridge\_ros.sh, it will stall indefinitely after printing “loading sound from: /home/root/sound”.

1. The Bluetooth daemon operates from the / directory, so include the full path for any files you call and use /home/root/ instead of ~/.
2. The SIGTERM signal is sent to all processes when the robot shuts down.
3. If you have issues not addressed here, the support team at [support@consequentialrobotics.com](mailto:support@consequentialrobotics.com) should be able to help. Ben Mitchinson is especially helpful.