Converting iRobot Create to Raspberry Pi and RealSense

Alec Newport (acn55) & Zhilong Li (zl242)

PART I: PREPARING THE RASPBERRY PI

1. Pull Needed Files from GitHub to Desktop Computer
    a. Repository located at https://github.com/acn55/iRobotRealSense
2. Power Raspberry Pi
    a. Acquire a micro USB power cord such as https://www.digikey.com/product-detail/en/raspberry-pi/T5875DV/1690-1021-ND/6674284
        i. This particular cord is 18 AWG which is too large for the Molex connectors on the power board. Need to splice with 22 AWG wire to fit into connectors.
        ii. On this cord, the wire with the black stripes is GND and that with the writing is VCC.
    b. Cut wire to desired length and attach 2-socket Molex connector
        i. Looking down from wire side and with notch on connector facing up, VCC is the left pin and GND is the right
    c. Plug Molex connector into 5V supply on I/O board
    d. Solder three wires to DB-25 connector: one to pin 10 (VCC), one to pin 11 (VCC), and one to pin 21 (GND)
    e. Splice the wires from pins 10 and 11 to a single VCC wire
        i. Each pin can source up to 1.5A and Raspberry Pi may occasionally draw up to 2.5A
    f. Cut wire to desired length and attach 2-socket Molex connector
        i. Looking down from wire side and with notch on connector facing up, VCC is the left pin and GND is the right
    g. Plug this Molex connector into VIN on I/O board
    h. Plug DB-25 into the cargo bay connector on the Create and the micro-USB into the Raspberry Pi
    i. Turn on the Create, and the red power LED should illuminate on the Raspberry Pi to indicate everything is connected correctly
3. Flash Ubuntu MATE to Micro SD Card
    a. Download balenaEtcher found here: https://www.balena.io/etcher/ to PC
    b. Run balenaEtcher
    c. Click Select Image and navigate to file named ubuntu-mate-16.04.2-desktop-armhf-raspberry-pi.img in GitHub repository folder
        i. File downloaded from here: https://ubuntu-mate.org/download/
    d. Ensure that micro SD card is selected under Select Drive

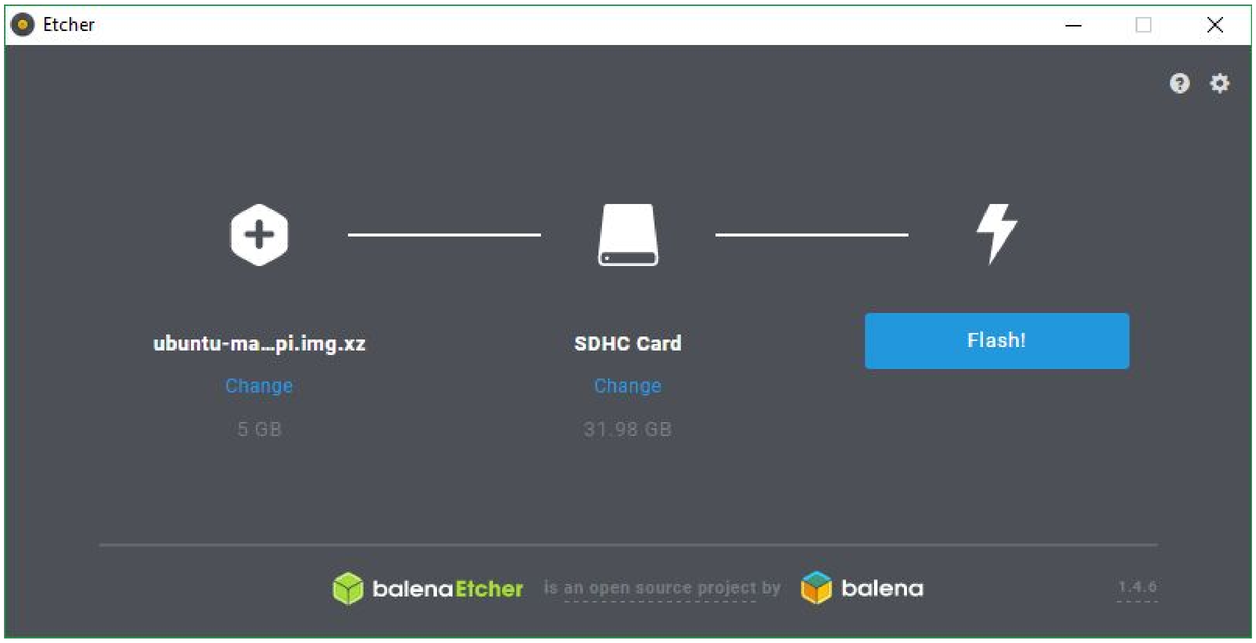


Figure 1: Etcher with Ubuntu MATE File and Micro SD Card Selected

e. Click Flash!
f. If prompted, allow app to make changes to your device
g. Wait for balenaEtcher to finish

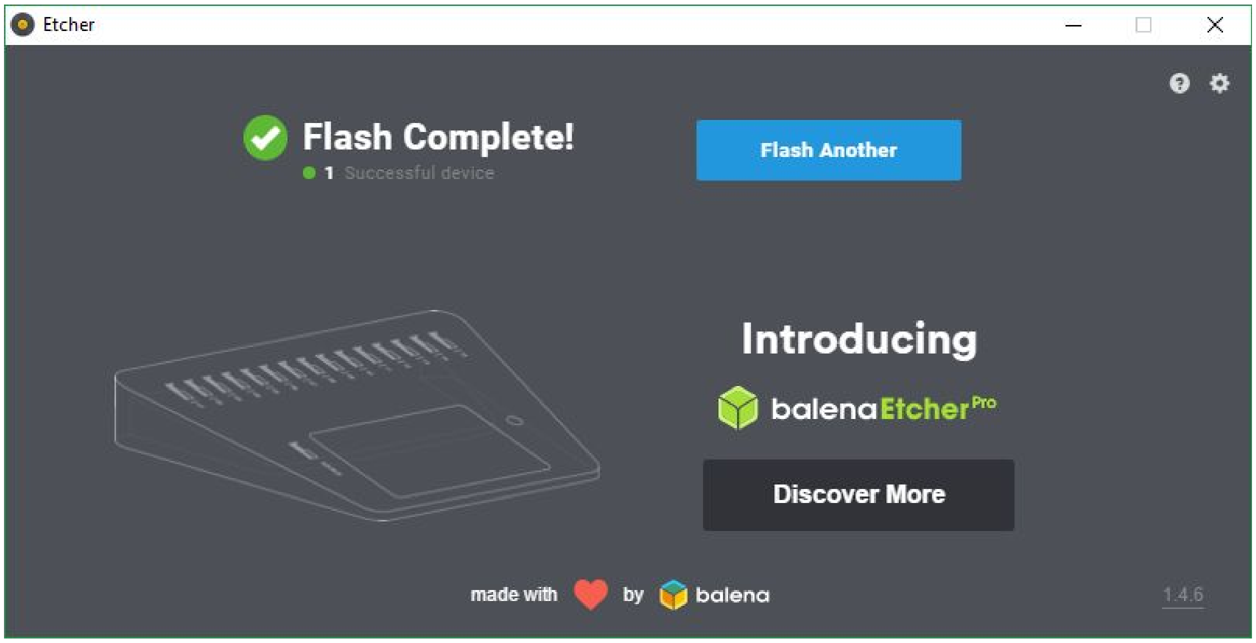


Figure 2: balenaEtcher Flash Complete

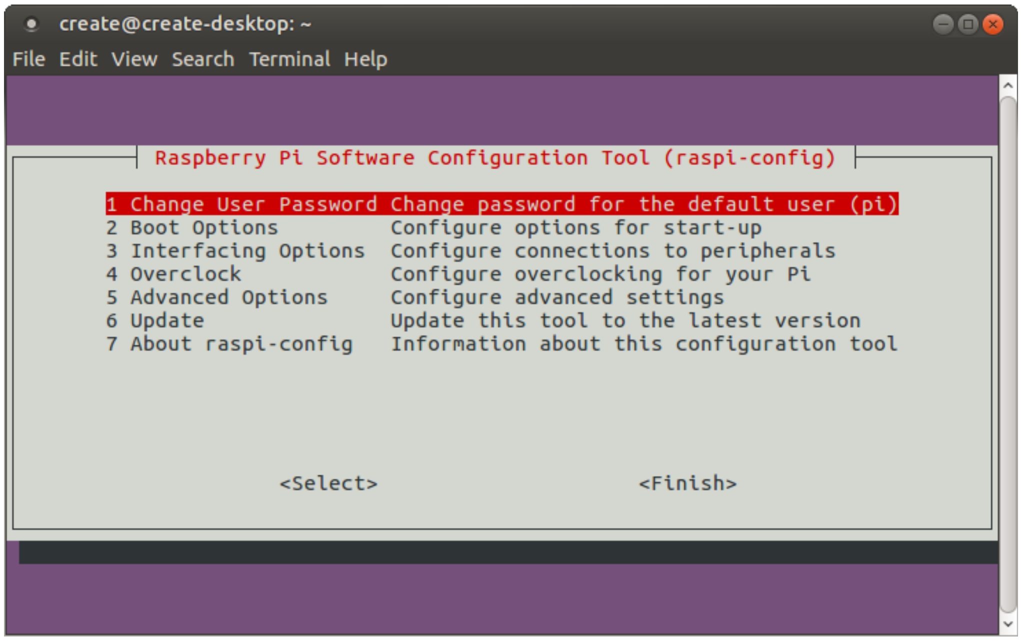
4. Set up Ubuntu MATE
   a. Plug monitor into HDMI port and mouse and keyboard into USB ports on Raspberry Pi
   b. Plug in power cable to Raspberry Pi

        i. Power must be connected after monitor or desktop will not load
    c. Wait for system to boot. Once finished, System Configuration window will pop up
    d. On language screen, select English and click Continue
    e. On Wireless screen, select desired network, enter password if necessary, and click Continue
        i. To set up a new WiFi connection later, there is a well-written guide by user Luis Alavarado here: https://askubuntu.com/questions/16584/how-to-connect-and-disconnect-to-a-network-manually-in-terminal There are guides both for the terminal (which can be done over SSH) and the desktop
    f. When "Connection Established" notification appears, click Continue again
    g. On Where are you? screen, if not already filled in, type "New York" into the text box and click Continue
    h. On Keyboard layout screen, if not already selected, click English (US) in both lists and click Continue
    i. On Who are you? screen, fill in all text boxes with appropriate information, select whether the system should log in automatically or require the password, and click Continue
        i. For the two micro SD cards we set up, the username is "create" and the password is "temppwd"
    j. Wait for system to finish initializing. It will restart.
    k. May need to restart system again manually for WiFi to connect
        i. WiFi will connect automatically on startup from now on

5. Setting up SSH
    a. Open a terminal
        i. In the upper left corner of the desktop, click Applications > System Tools > MATE Terminal
    b. Run the command:

```
sudo raspi-config
```

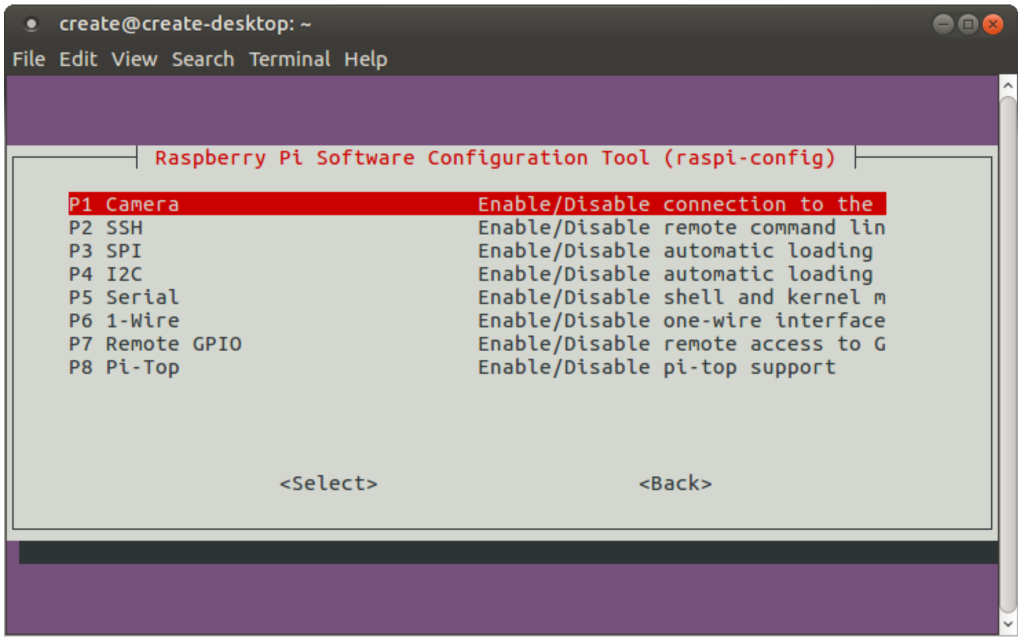
    c. Enter password if prompted
    d. You should see the window shown in Figure 3. Use the arrow keys to navigate to Interfacing Options and press Enter



```
create@create-desktop: ~
File  Edit  View  Search  Terminal  Help

            Raspberry Pi Software Configuration Tool (raspi-config)

        1 Change User Password Change password for the default user (pi)
        2 Boot Options         Configure options for start-up
        3 Interfacing Options  Configure connections to peripherals
        4 Overclock            Configure overclocking for your Pi
        5 Advanced Options     Configure advanced settings
        6 Update               Update this tool to the latest version
        7 About raspi-config   Information about this configuration tool

                      <Select>                     <Finish>
```

Figure 3: Raspberry Pi Configuration

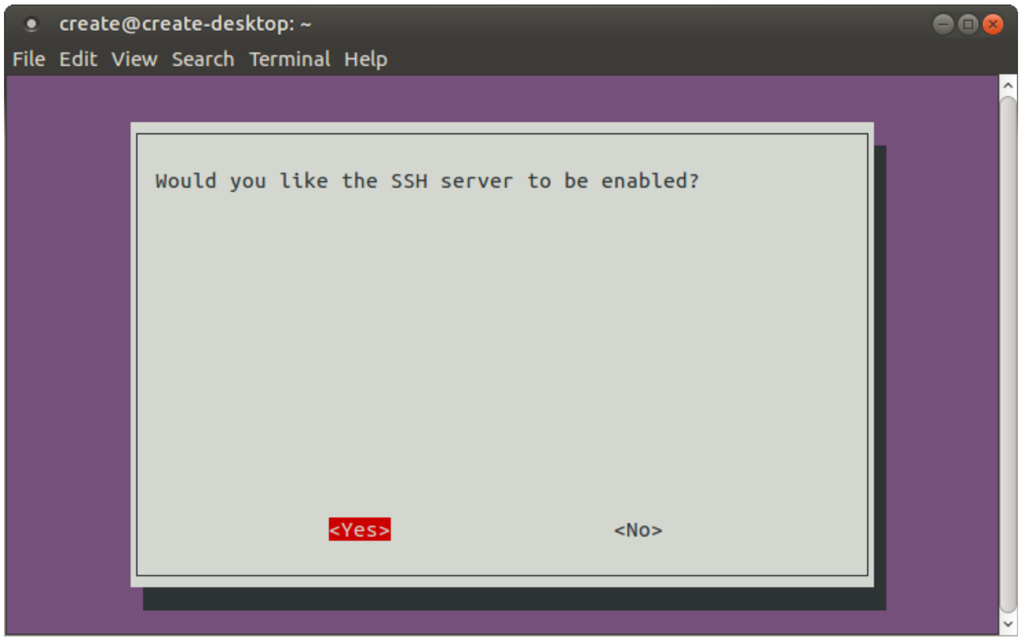
e. This will show the screen in Figure 4. Use the arrow keys to navigate to SSH and press Enter



```
create@create-desktop: ~
File  Edit  View  Search  Terminal  Help

            Raspberry Pi Software Configuration Tool (raspi-config)

     P1 Camera                         Enable/Disable connection to the
     P2 SSH                            Enable/Disable remote command lin
     P3 SPI                            Enable/Disable automatic loading
     P4 I2C                            Enable/Disable automatic loading
     P5 Serial                         Enable/Disable shell and kernel m
     P6 1-Wire                         Enable/Disable one-wire interface
     P7 Remote GPIO                    Enable/Disable remote access to G
     P8 Pi-Top                         Enable/Disable pi-top support

                      <Select>                     <Back>
```

Figure 4: Interfacing Options

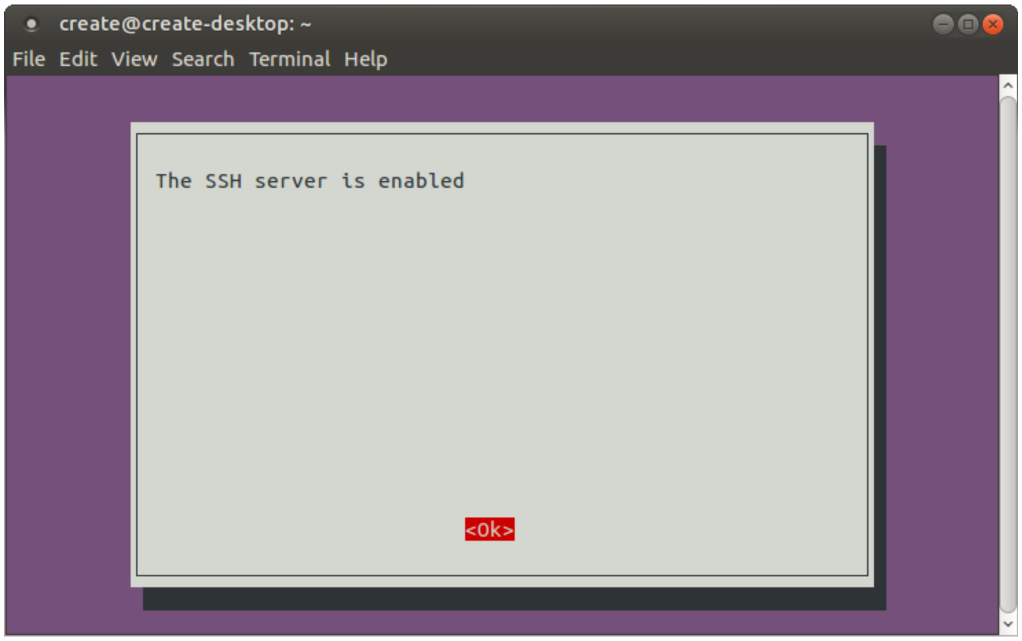
f. This shows the screen in Figure 5. Use the arrow keys to navigate to <Yes> and press Enter



```
Would you like the SSH server to be enabled?

<Yes>                    <No>
```

Figure 5: Enable SSH Server

g. Wait for the command to run. When the screen in Figure 6 appears, press Enter to select <Ok>



```
The SSH server is enabled

<Ok>
```

Figure 6: SSH Enabled

h. Use the right arrow key to select <Finish> and press Enter
i. Test the connection using PuTTy. If PuTTy receives the prompt “login as:”, everything is working correctly
j. You may now login with your username and password

k. Everything from now on can be done from the SSH connection, no desktop setup required

6. Prepare Ubuntu to make the RealSense SDK
   a. SSH into the Raspberry Pi from PuTTy and login
   b. Update Ubuntu by running the following command:
   ```
   sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade
   ```
      i. When asked if you want to continue, type ‘y’ and press Enter
      ii. If prompted to update /etc/gnome/defaults.list press Enter to keep current version (we do not use Gnome)
   c. Install Git by running the following command:
   ```
   sudo apt install git
   ```
      i. When asked if you want to continue, type ‘y’ and press Enter
   d. Clone the RealSense SDK Git repository by running the command:
   ```
   git clone https://github.com/IntelRealSense/librealsense
   ```
   e. Change directories to the new librealsense directory:
   ```
   cd librealsense
   ```
   f. Ensure that the RealSense is not plugged in to the Raspberry Pi
   g. Install the core directories necessary for building librealsense using the following command:
   ```
   sudo apt-get install git libssl-dev libusb-1.0-0-dev pkg-config libgtk-3-dev
   ```
      i. When asked if you want to continue, type ‘y’ and press Enter
   h. Then run the command:
   ```
   sudo apt-get install libglfw3-dev
   ```
      i. When asked if you want to continue, type ‘y’ and press Enter
   i. Install the permission scripts with the following two commands:
   ```
   sudo cp config/99-realsense-libusb.rules /etc/udev/rules.d/
   sudo udevadm control --reload-rules && udevadm trigger
   ```
   j. Install cmake using the command:
   ```
   sudo apt install cmake
   ```
      i. When asked if you want to continue, type ‘y’ and press Enter

7. Create a 1GB swap file
   a. Ensure that there are no existing swap files by making sure the return value from the following command is an empty list:
   ```
   sudo swapon -s
   ```
   b. Create the swap file using the following command:
   ```
   sudo dd if=/dev/zero of=/swapfile bs=1024 count=1024k
   ```
      i. This is a large file, so it will take a minute to complete this command
   c. Designate the new file as a swap file by running the command:
   ```
   sudo mkswap /swapfile
   ```

    d. Enable the swap file using the command:
    ```
    sudo swapon /swapfile
    ```
    e. Set the permissions on the swap file using:
    ```
    sudo chmod 0600 /swapfile
    ```
    f. Ensure the file was created successfully by ensuring it appears in the list returned by the following command:
    ```
    sudo swapon -s
    ```
    g. This swapfile is deleted on restart, so if the Raspberry Pi is rebooted before completely building the SDK, perform step 7 again

8. Build the SDK
    a. If not already in the librealsense directory, navigate there
    b. Create a build directory and navigate into it using the following command:
    ```
    mkdir build && cd build
    ```
    c. Run cmake using the command:
    ```
    cmake ../
    ```
    d. Recompile and install using the following command:
    ```
    sudo make uninstall && make clean && make && sudo make install
    ```
        i. This is the step requiring the large swapfile and a lot of time. It took us about 50 minutes to complete

9. Install Python libraries
   Overview: to run the realsense SDK on the Raspberry Pi, we would need a Python wrapper called pyrealsense2. To recognize and process apriltags in the field of view, we also need to install the apriltag package for Python. Numpy is required to carry out the necessary calculations. Note that in this project we are using **Python3.**
    a. Navigate to the home directory using:
    ```
    cd ~
    ```
    b. Clone the apriltag GitHub repository with the command:
    ```
    git clone https://github.com/swatbotics/apriltag
    ```
    Follow instructions in the link to build the Python apriltag package from source
    c. Install numpy with:
    ```
    pip install numpy
    ```
    d. Then install the pyrealsense2 wrapper for the RealSense SDK. Follow the instructions here:
    https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python
    Again, the Pi will run out of memory during normal installation, so a swap file has to be created beforehand, follow instructions above. This installation will take 30-45 minutes to complete.

10. Now that all the dependencies are installed, ensure that all the files are in the right directory so that when importing packages Python doesn't run into any problems. The robot_controller.py code is in the /apriltag/python folder.
    a. Move the pyrealsense compiled files (the 5 files with .so extension) from the build folder to the folder containing the robot_controller code, refer to image below.

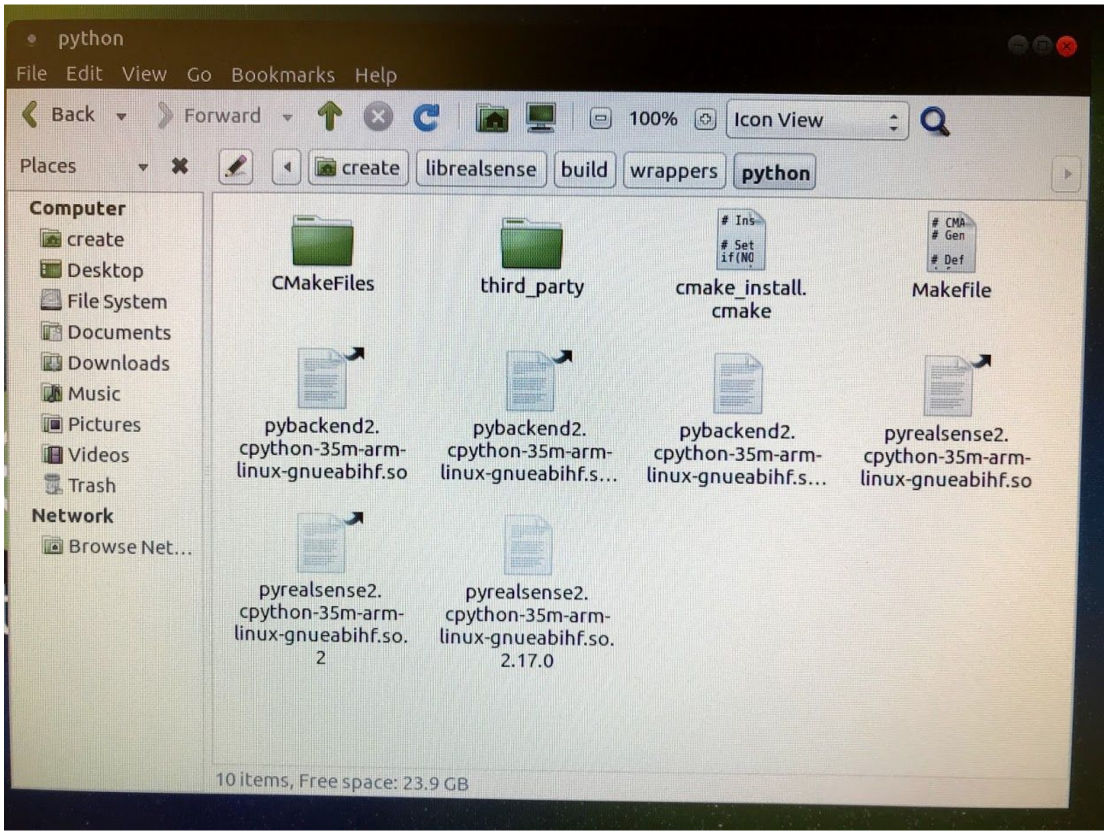

python
File Edit View Go Bookmarks Help
Back
Forward
100%
Icon View
Places
create
librealsense
build
wrappers
python
Computer
create
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Browse Net...
CMakeFiles
third_party
cmake_install.cmake
Makefile
pybackend2.cpython-35m-arm-linux-gnueabihf.so
pybackend2.cpython-35m-arm-linux-gnueabihf.s...
pybackend2.cpython-35m-arm-linux-gnueabihf.s...
pyrealsense2.cpython-35m-arm-linux-gnueabihf.so
pyrealsense2.cpython-35m-arm-linux-gnueabihf.so.2
pyrealsense2.cpython-35m-arm-linux-gnueabihf.so.2.17.0
10 items, Free space: 23.9 GB

Figure 7: .so Files

    b. Move the .so files from the apriltag build folder to the python folder containing robot_controller.py. Finally, the folder with robot_controller.py code should contain all of the files in Figure 8 below:

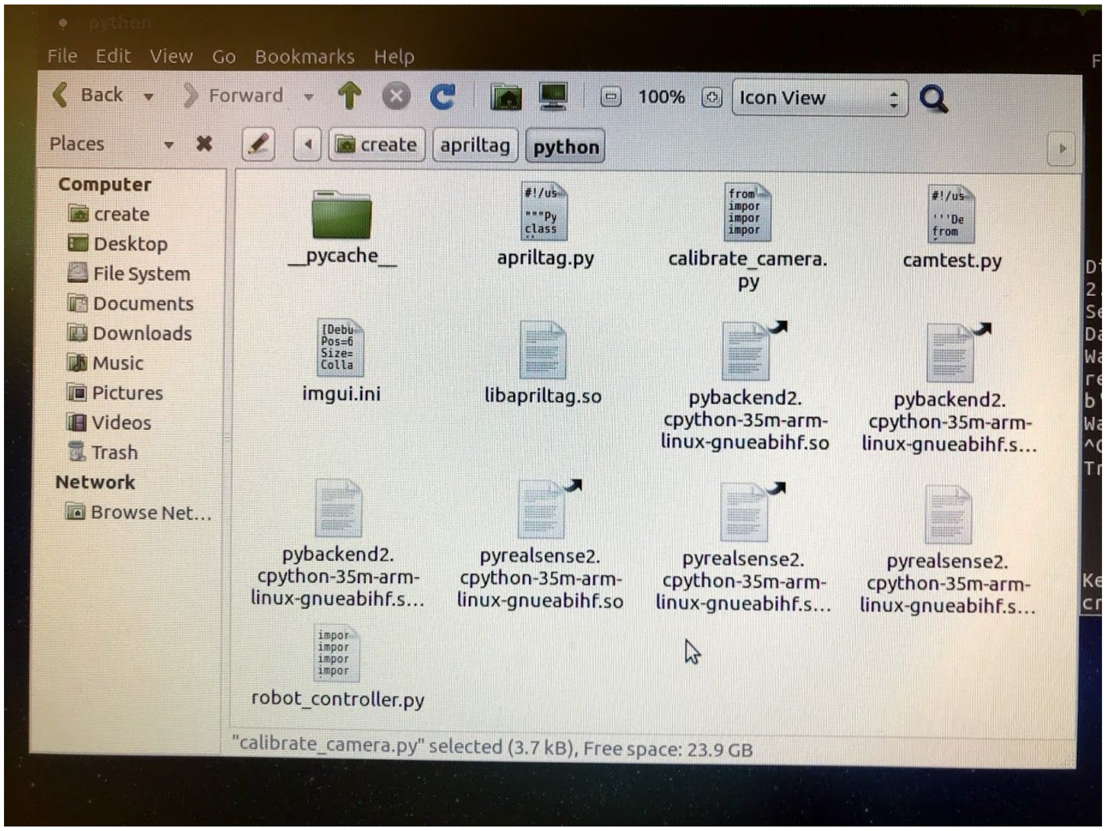

File Edit View Go Bookmarks Help
Back
Forward
100%
Icon View
Places
create
apriltag
python
Computer
create
Desktop
File System
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Browse Net...
__pycache__
apriltag.py
calibrate_camera.py
camtest.py
imgui.ini
libapriltag.so
pybackend2.cpython-35m-arm-linux-gnueabihf.so
pybackend2.cpython-35m-arm-linux-gnueabihf.s...
pybackend2.cpython-35m-arm-linux-gnueabihf.s...
pyrealsense2.cpython-35m-arm-linux-gnueabihf.so
pyrealsense2.cpython-35m-arm-linux-gnueabihf.s...
pyrealsense2.cpython-35m-arm-linux-gnueabihf.s...
robot_controller.py
"calibrate_camera.py" selected (3.7 kB), Free space: 23.9 GB

Figure 8: All Files in ~/apriltag/python Directory

11. Test that the Python code robot_controller.py now can run without error. If there is a "Permission Denied" error for serial connection, type `sudo usermod -a -G dialout $USER` into the terminal and reboot.

PART II: THE MATLAB TOOLBOX

Most of the functions are unchanged from the previous version of the iRobot Create MATLAB toolbox. Functions that control the robot and reading the onboard sensors can be used in exactly the same way as before (except CreateBeagleInit has been renamed to CreatePiInit). Since the sonars and the RGB camera are removed, the original functions ReadBeacon, ReadSonar, and PlotCreateVideo are no longer used. Instead, three new functions are added to interact with the RealSense camera.

All of the functions, with the exception of RealSenseImage, should terminate and return within a second, otherwise check the putty SSH terminal to see if the Python controller code failed with an error.

1. RealSenseDist
   Returns an array of floats. The top element in the array represents the distance (in meters) at the leftmost point of the depth image at the specified height, and the bottom element represents the rightmost point of the depth image. The other points are interpolated between the left and right points. The code enforces the precondition that the number of points sampled is between 2 and 9, and the height in degrees is between 1 and 40. The effective range of the RealSense camera is between 0.15 and 10 meters. Any returned reading of less than 0.15 meters should be treated as erroneous data.
2. RealSenseTag
   Returns a 2D array representing the apriltags seen in the RGB image. If no tags are detected, then an empty array is returned. Each row of the returned array represents a tag in this order: [id z x yaw].
   - "id" is the apriltag ID
   - "z" is the distance between the center of the tag and the RealSense camera, in the camera frame along the axis perpendicular to the camera
   - "x" is the distance between the center of the tag and the center of the image, in the camera frame along the horizontal axis across the image.
   - "yaw" is the angle of rotation of the tag about the z axis of the camera frame (axis perpendicular to the camera), in radians.

   The distance data x and z are calculated using the Python apriltag package, which returns the pose of the tags as a homogeneous transform matrix. The calculations are dependent on the size of the apriltag which is entered as an input, giving the side length in meters, into the detect_pose function in the Python code. This parameter is set and can be changed at line 125 of robot_controller.py code.
   Note: if the apriltag is only partially in the image, then it would not be detected.
3. RealSenseImage
   This function is for debugging purposes, it displays the current video image seen by the RealSense camera as a grayscale image in a MATLAB figure. [height] represents the

angle at which to display a white line on the image; this angle corresponds to the [height] input into the RealSenseDist function, demarcating the region from which the distance points are sampled in that function.

Because of the image size and the TCPIP connection, it takes about 5 seconds for all the data to transferred to the host computer. Sometimes not the entire image gets transferred over, where the bottom of the displayed image will be truncated and a warning will be displayed in the MATLAB command window. Most of the time this truncation is insignificant and will not affect the regions of interest, but if it does, simply close the displayed figure and try again.

A sample output figure is shown below, where the [height] is set at 10 degrees:

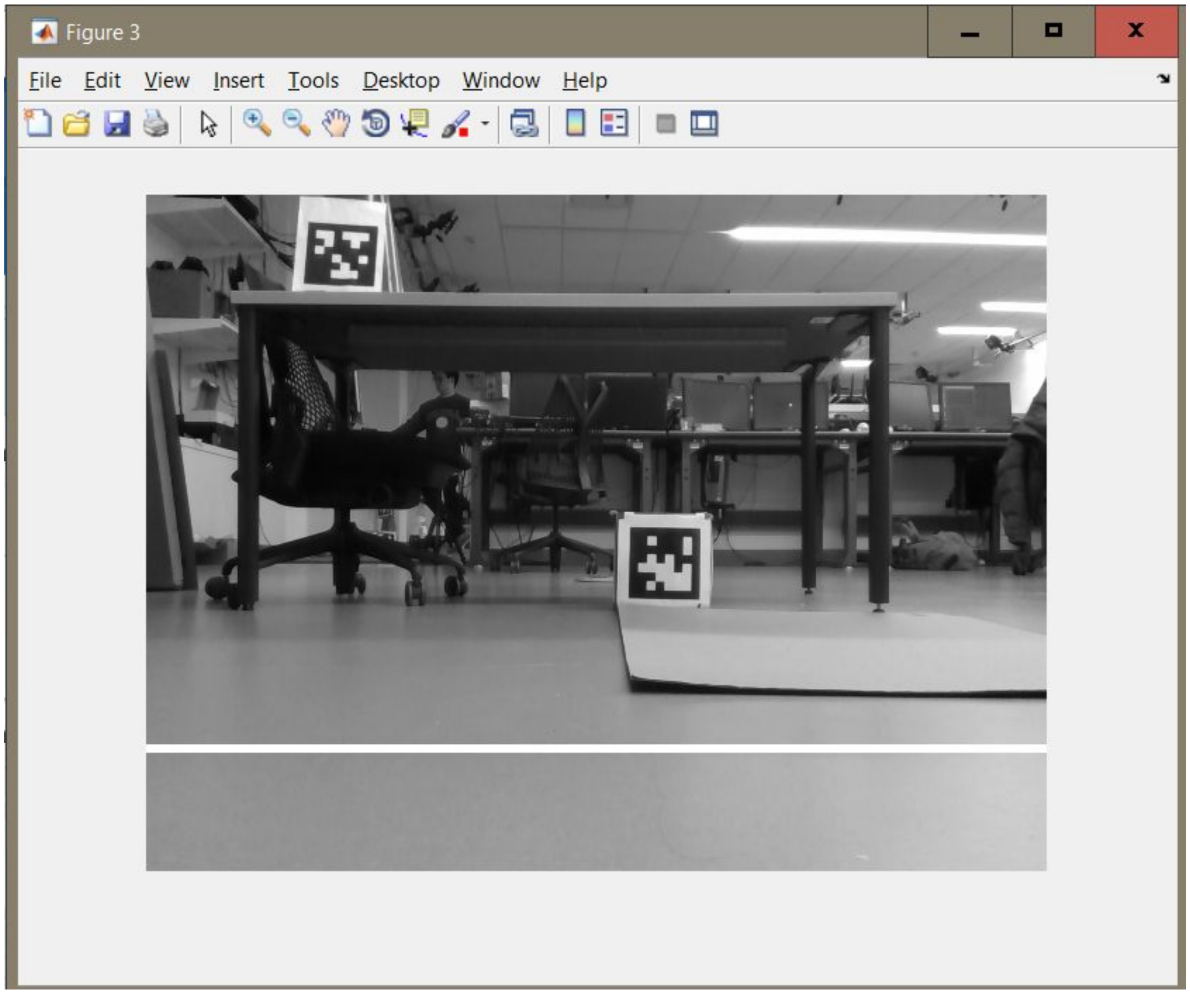


Figure 9: Sample RealSenseImage Output

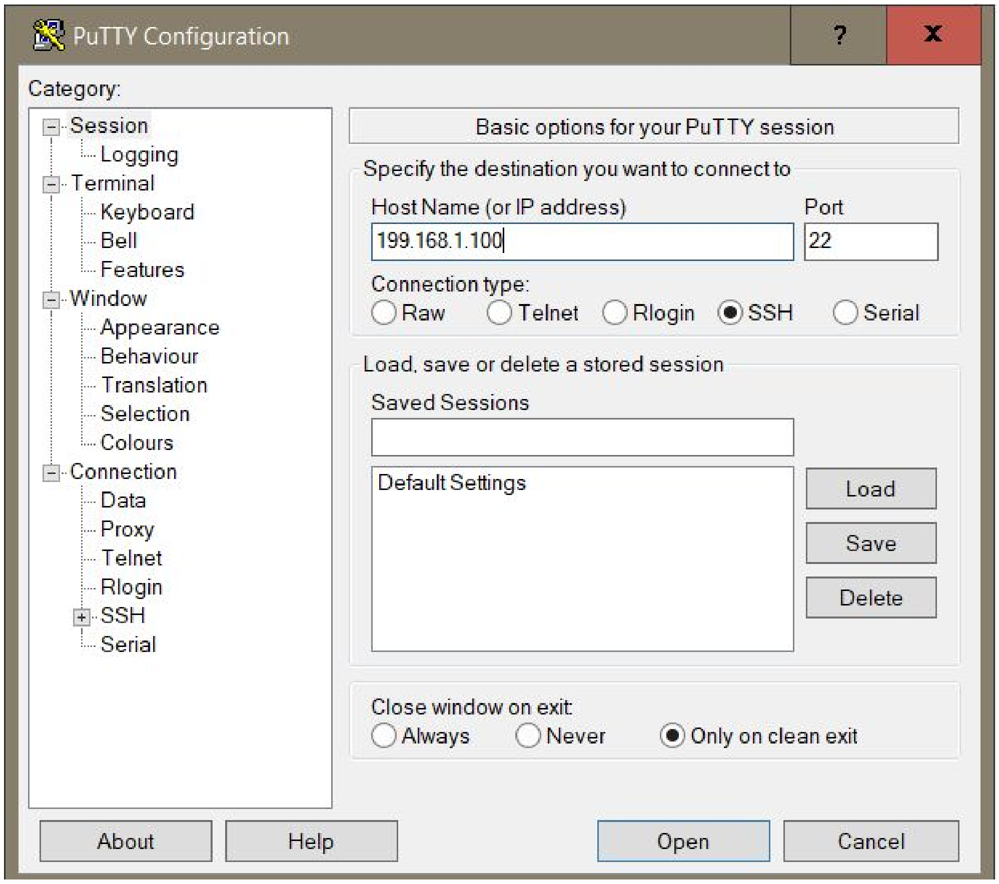
PART III: THE PYTHON CODE

All the code responsible for communicating with the robot and the host computer is housed in "robot_controller.py" in the /create/apriltag/python directory. This program can be started either with "./robot" command in the top level directory of the Raspberry Pi, or by running "python3 robot_controller.py" from the apriltag/python directory. Below are the main components of the code, refer to the python script for detailed specifications.

1. main(). Starts the TCPIP connection, sets the serial port for communicating with the robot, starts the RealSense camera streaming pipeline, and listens for commands from the host computer. The resolution for both the depth and the color image frames are set to be 640*480 pixels. The depth frame is also aligned to the color frame, and the effective field of view for both frames is approximately 52*40 degrees.
2. send_message(data, ser_port, seriallock). This is a thread-safe function to send the binary data to the robot through the serial port [ser_port]. Any commands from the host computer other than the RealSense commands is sent to the robot via this function.
3. get_distance(depth_frame, num_points, height). This function processes the depth frame from the camera and gets the specified number of points at the specified height. At each of the points, the code takes the average of a region that is 4 pixels x 4 pixels and returns that distance.
4. get_tag(color_frame, depth_frame, params). This functions finds all the apriltags seen in the color image and returns their id and pose. [params] is the camera intrinsic parameters found in the main() function, necessary for calculating the pose of the apriltags. To set or change the size of apriltags in use, change the last argument in the detect_pose() function in line 125.

PART IV: ESTABLISHING CONNECTION

1. To start the Raspberry Pi, power on the iRobot, open Putty in host computer and SSH into the Pi through its specific static IP address



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 199.168.1.100

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

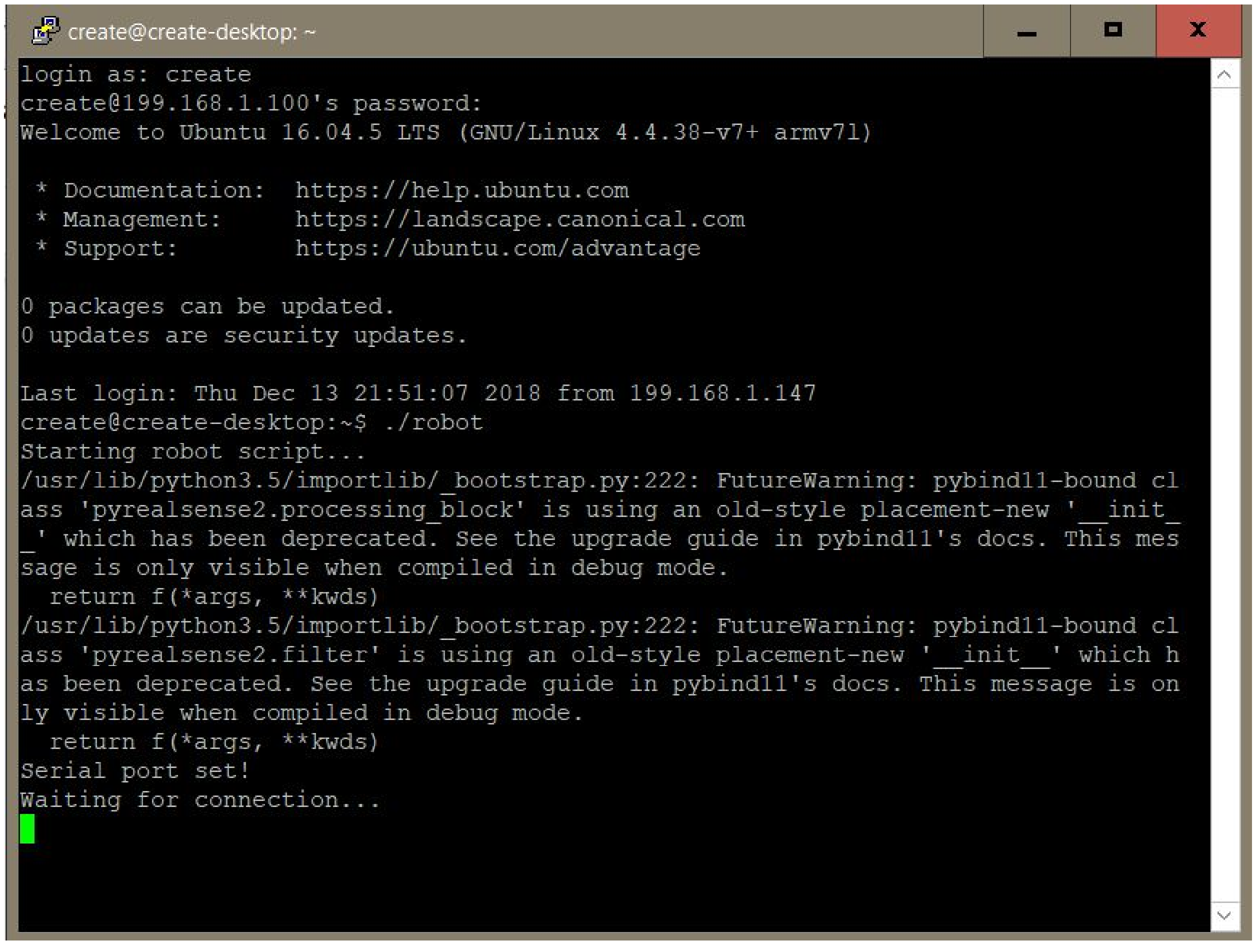
Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Figure 10: Opening PuTTY SSH Connection

2. In the next screen, when prompted, key in “create” as the username and “temppwd” as the password
3. Run the robot controller Python file directly by typing “./robot”
    a. The terminal should display “waiting for connection”, indicating that the Python code has successfully started and is waiting for incoming TCPIP connection from the host computer
4. Run “<port name> = CreatePiInit(<IP address of the Raspberry Pi>)” on MATLAB to start the connection between the Pi and the host computer
    a. The iRobot will beep if the connection is successful, otherwise, an error will be displayed in both MATLAB and Python terminal indicating an exception. In this case, repeat this step (step 3 only) from the beginning.



create@create-desktop: ~

```
login as: create
create@199.168.1.100's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.38-v7+ armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

Last login: Thu Dec 13 21:51:07 2018 from 199.168.1.147
create@create-desktop:~$ ./robot
Starting robot script...
/usr/lib/python3.5/importlib/_bootstrap.py:222: FutureWarning: pybind11-bound cl
ass 'pyrealsense2.processing_block' is using an old-style placement-new '__init_
_' which has been deprecated. See the upgrade guide in pybind11's docs. This mes
sage is only visible when compiled in debug mode.
  return f(*args, **kwds)
/usr/lib/python3.5/importlib/_bootstrap.py:222: FutureWarning: pybind11-bound cl
ass 'pyrealsense2.filter' is using an old-style placement-new '__init__' which h
as been deprecated. See the upgrade guide in pybind11's docs. This message is on
ly visible when compiled in debug mode.
  return f(*args, **kwds)
Serial port set!
Waiting for connection...
```

Figure 11: Create Waiting for Connection from MATLAB

PART V: KNOWN ISSUES AND DEBUGGING

1. If all the procedures above as followed, then the system should work as described in the documentations and as shown in the demos. Below are some of the issues that we have encountered during development process.
2. Installation of software and packages are Raspberry Pi: during installation of large software, the Pi can be very slow, taking up to an hour for some packages. At times, it seemed that the Pi has frozen. It never did freeze during our installation, so we recommend leaving it to install, eventually it will finish. It would be best if the entire SD card can be cloned and reproduced on other cards to avoid going through the lengthy installation process.
3. Occasionally, the RealSense camera pipeline fails to shutdown correctly on program exit of the robot_controller.py, this will manifest itself the next time the user tries to establish connection between the host computer and the Raspberry Pi, where the Python code would fail with a "resource busy" error. If this happens, simply repeat step 3 of PART III of this document.
4. Noise in the depth output of the camera: we noticed two instances where the depth output performs poorly; namely when an object is placed close to the left side of the camera or when measurements are taken off a smooth surface at an oblique angle.
    a. Objects near the left of the camera casts a "shadow" further to the left. The image below shows the shadow to the left of the piece of paper.

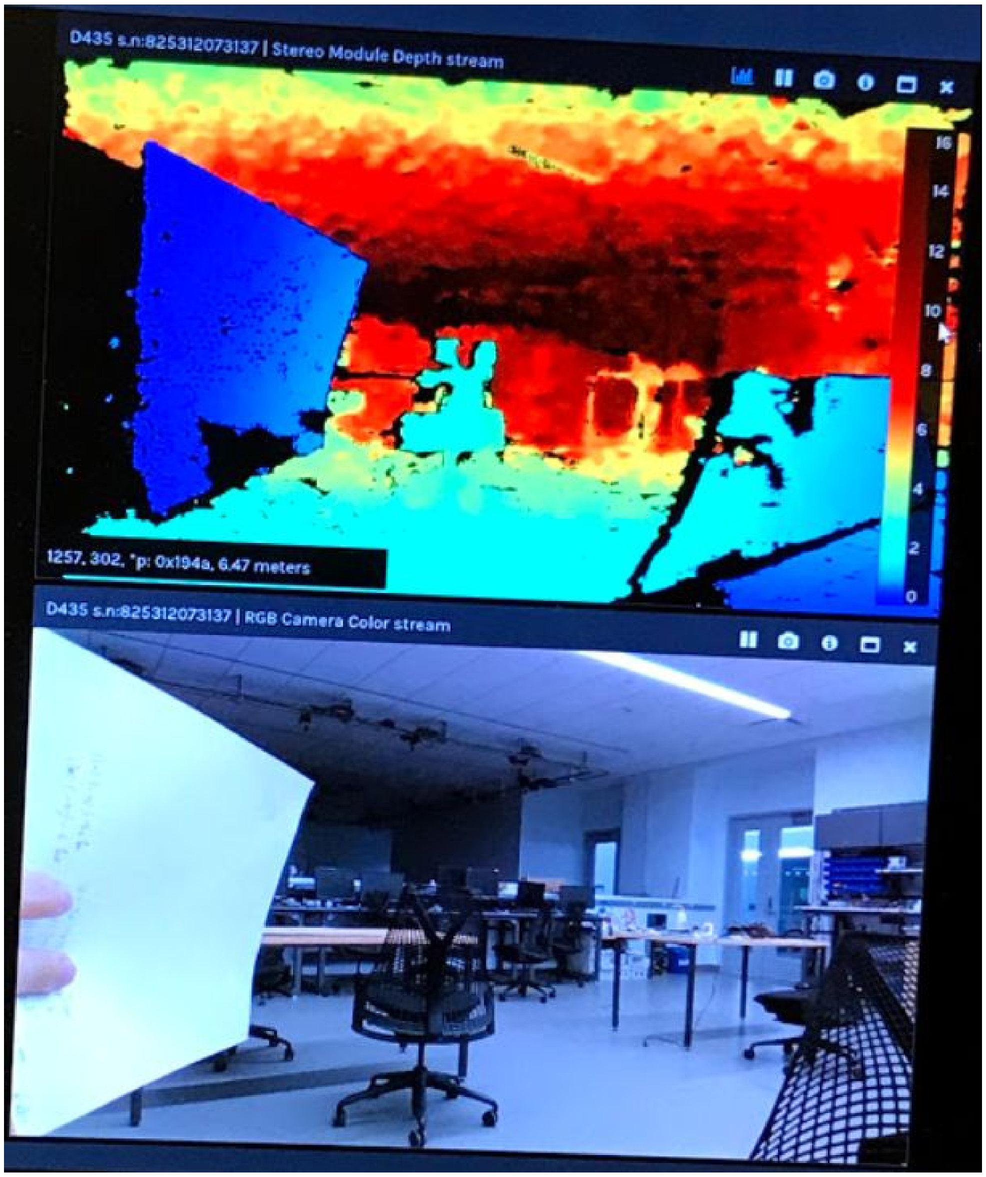

D435 s.n:825312073137 | Stereo Module Depth stream
16
14
12
10
8
6
4
2
0
1257, 302, *p: 0x194a, 6.47 meters
D435 s.n:825312073137 | RGB Camera Color stream

Figure 12: Distance Sensor Shadow

b. The depth readings on a smooth surface tends to be patchy, as seen from the example below where the computer monitor that is at a very oblique angle (near 90 degrees) to the camera shows regions of zero depth readings.

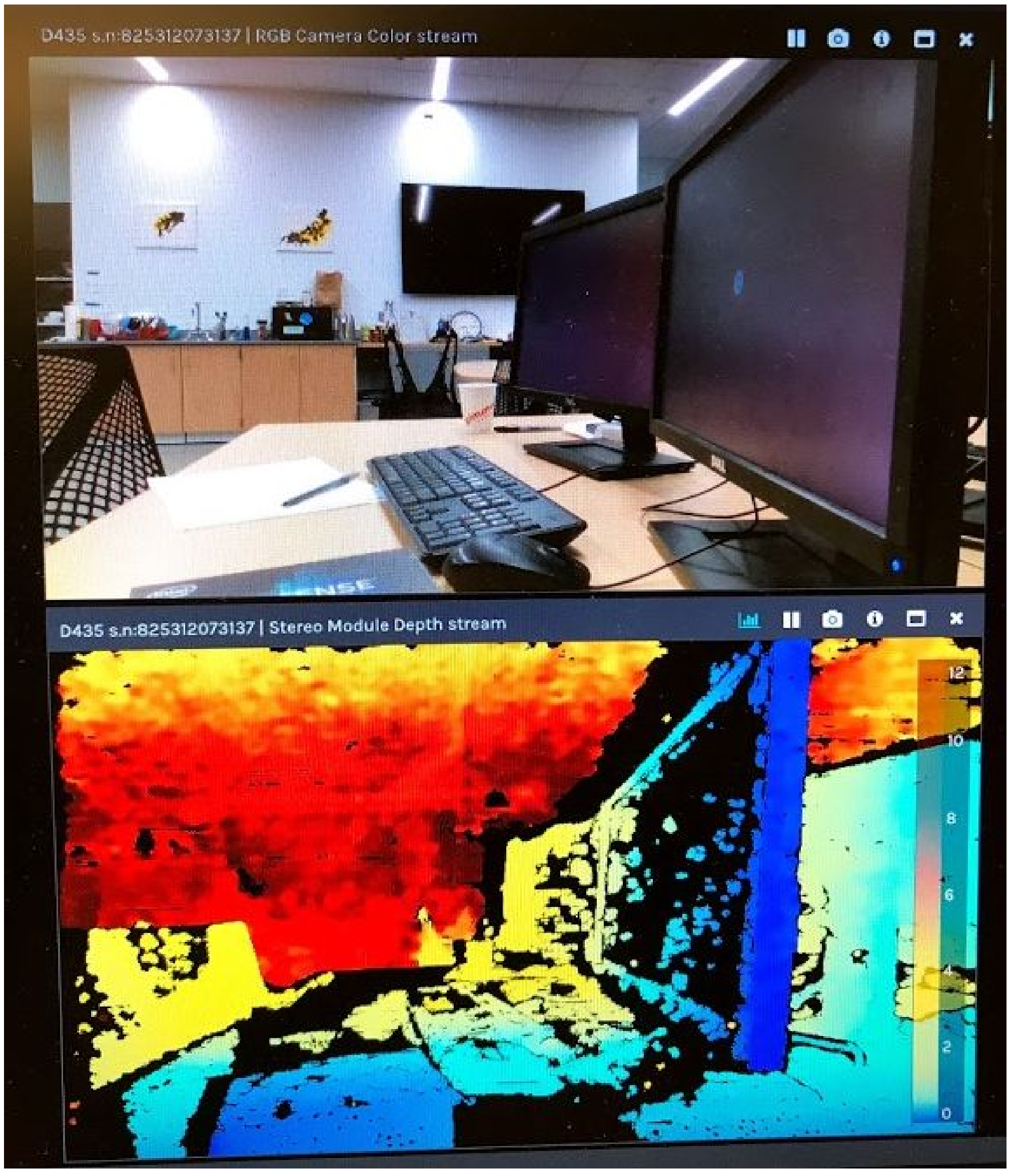

D435 s.n:825312073137 | RGB Camera Color stream
D435 s.n:825312073137 | Stereo Module Depth stream
12
10
8
6
4
2
0

Figure 13: Patchy Distance Reading

From our tests the camera's depth readings are accurate as long as the objects are within the effective range and do not fall into the above two categories. If the readings seem very off, the best way to figure out the source of the problem is to connect the camera to a computer with the RealSense Viewer program installed and look at both the RGB and depth outputs. Note again that the resolution of both the depth and RGB frames used in robot_controller.py is 640*480, and that the field of view of the depth frame is aligned with that of the RGB frame.