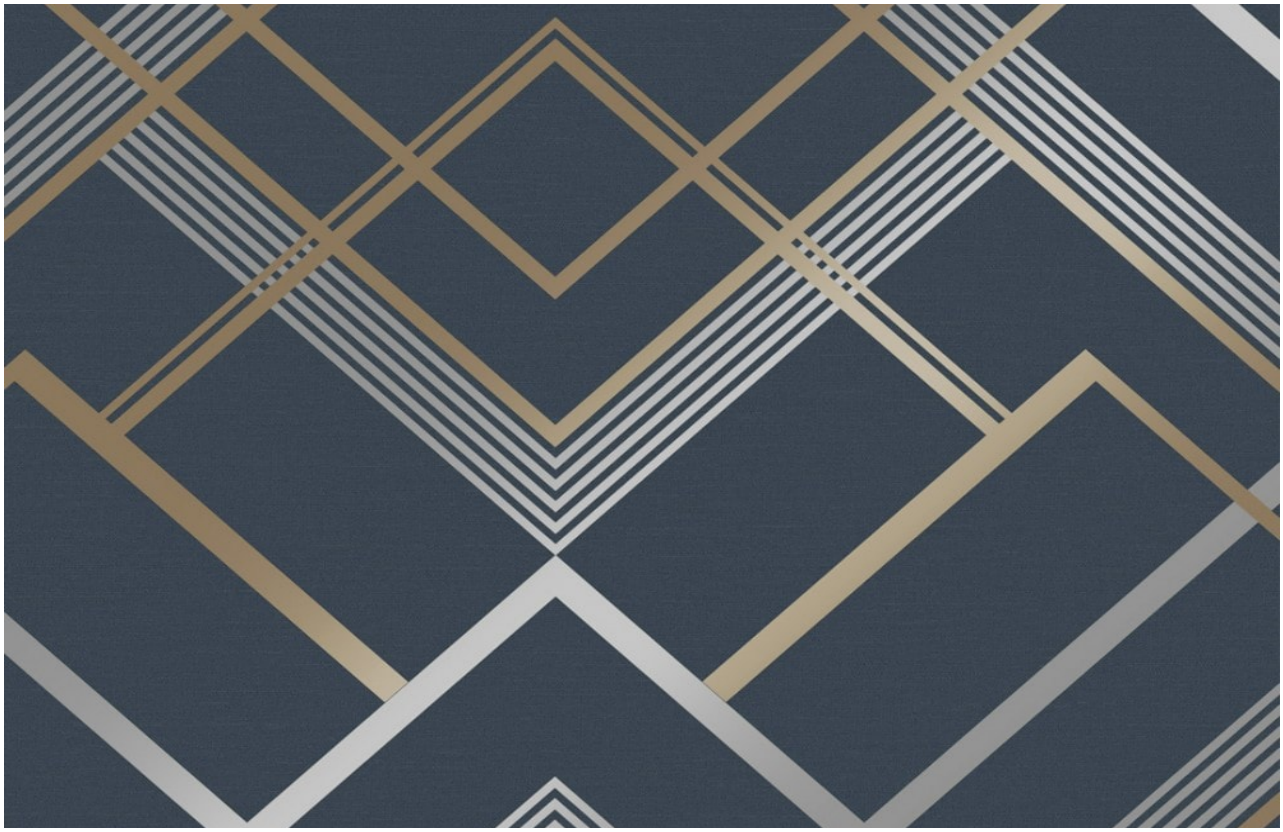


Rosmaster X3 Documentation

Rostmaster Setup

12 . 15 . 2023

Ryan Woodward

Grand Canyon University

SWE 452

Prof. Bill Hughes

GRAND CANYON UNIVERSITY

Table of Contents

Introduction

The purpose of this document is to detail the setup and execution of a variety of functions and projects developed for the *Yahboom ROSMaster x3* robot. Hereafter this document will refer to the robot as ROSM. It is important to note that this document does not detail how to construct the robot or how to set up the ROS (Robot Operating System) on the Raspberry PI (RPI). For information on that see the appendices of this document for resources.

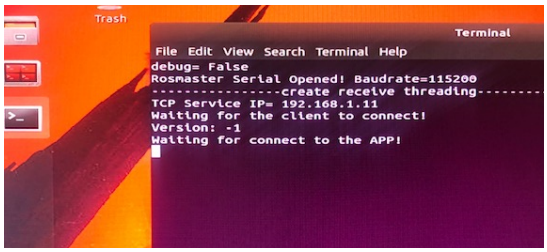
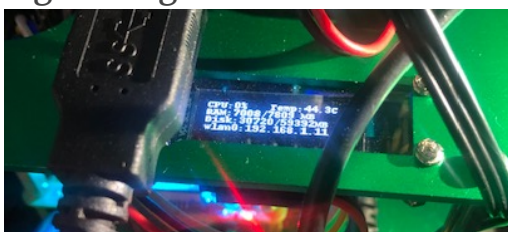
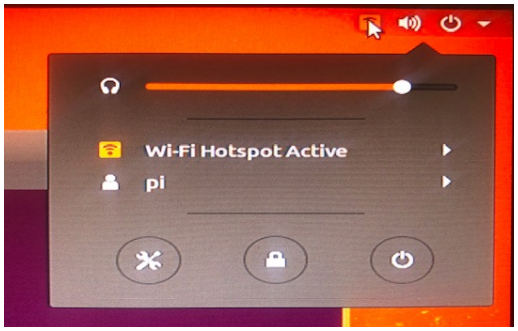
Controlling the Robot

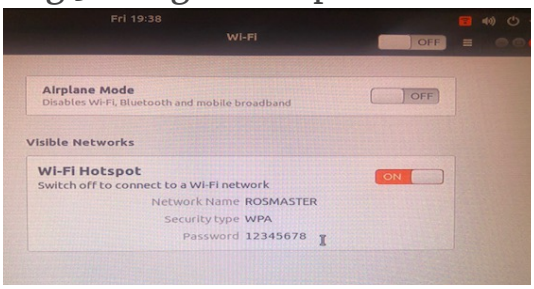
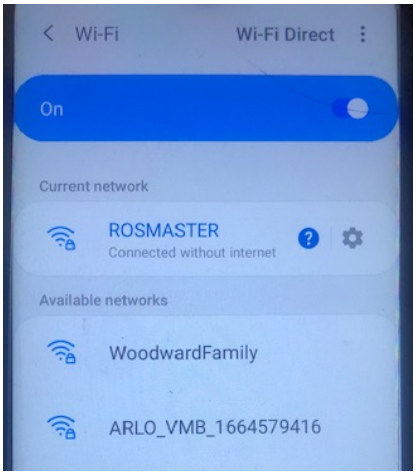
Controlling the Robot via Smartphone

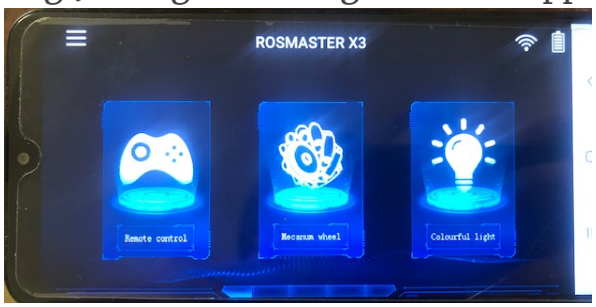
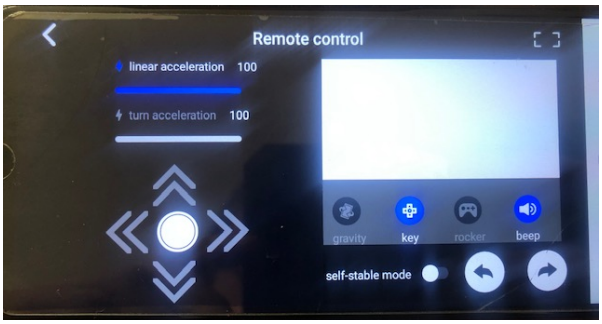
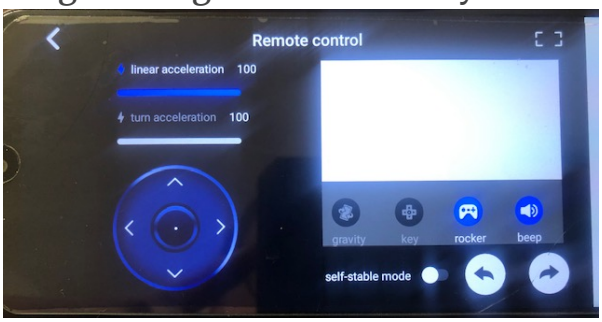
Notes:

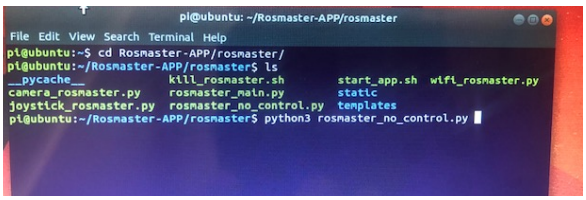
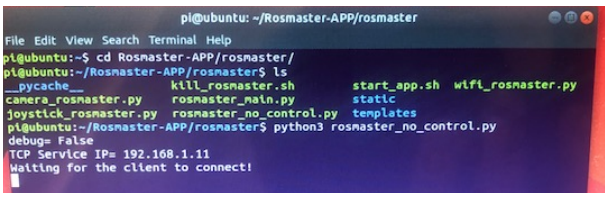
- After booting up the **ROSM** a terminal window is automatically opened. This terminal is running the '**rosmaster_no_control.py**' Script found in the '**Rosmaster-APP/rosmaster**' directory If the terminal window is closed the script's execution is terminated but can be started by running moving to the directory from '**~/**' and running the command '**python3 rosmaster_no_control.py**' (See **Step A,** at the bottom of the table below for more information).

Controlling ROSM from a Smartphone			
#	Name	Steps	Screenshot
Required Materials		• USB Keyboard • USB Mouse • Android Smartphone	
1	Install **Maker Control** App	1. Open the google play store. 2. Search for **Maker Control** 3. Select the application that looks like the image on the right 4. Download the application to your phone.	 Fig 1. Image of PlayStore App Icon

			Scan the QR code to Download App
2	Determine the IP address of your **ROSM**	1. Boot up your **ROSM** a. Let the Boot sequence complete . b. The Ubuntu GUI should become visible with a terminal instance. 2. After a successful boot the IP address should be displayed in two places. a. OLED Screen b. Terminal on the main display	 Fig 2. Image of Terminal with IP  Fig 3. Image of OLED with IP
3	Determine the Password & Network Name of **ROSM** Hotspot	1. Connect a Mouse directly to the **RPI** 2. Click the Connection Icon in the Top Right corner 3. Click the **Wi-Fi Hotspot Active** dropdown 4. Click **Wi-Fi Settings** 5. Listed should be:	 Fig 4. Image of Connections Menu

		a. Network Name *b. Security Type* *c. Pasword*	 Fig 5. Image of Dropdown menu  Fig 6. Image of Hotspot Details
4	Connect Phone to the **ROSM Hotspot**	1. Open Wi-Fi Settings on your phone 2. Find and connect to the **ROSMASTER** hotspot by using the password determined in **step 3**	 Fig 7. Image of ROSMASTER Hotspot
5	Connect Phone to **ROSM** using **Maker Control** App	1. Open the **Maker Control** App 2. Select **ROSMASTER X3** 3. You will be directed to a **Wi-Fi Configuration** screen 4. Here you must enter following details:	 Fig 8. Image of App Home Menu

		a. IP Address (found in **Step 2**) b. Port (Typically **6000**) c. Video (Actually baudrate, should be **6500**) 5. Once all details are entered, tap **connect** 6. You will be navigated to a new screen (see **Fig 10** to the left)	 Fig 9. Image of Config Screen in App  Fig 10. Image of Control Home
6	Driving the Robot	1. By Selecting the **Remote Control** option users can move the robot. a. Turning in place is achieved with the arrows in the bottom right corner of the menu 2. The white windows on **figures 11 and 12** should show the camera feed from the **ROSM**	 Fig 11. Image Of Control: Key Mode  Fig 12. Image of Control with Rocker Mode

A	Executing the Script	1. If you closed the terminal that opens on Boot of the **ROSM** 2. You can execute it again by opening a terminal and running these commands: 3. **cd Rosmaster-APP/rosmaster** 4. **python3 rosmaster_no_control.py** 5. After running that python script you can follow the steps to connect and control the **ROSM** from a smartphone.	 Fig 13. Navigating to the Script in the terminal  Fig 14. Executing the Script
B	**Demo**	https://youtu.be/W7TLQYEMvag	

Controlling the Robot via Gaming Remote

Notes:

- In order for this function to work the **rosmaster_no_control.py** script must be terminated. This script is executed upon boot. To stop its execution simply close the terminal that appears on successful boot or type **ctrl+c**.
- Similarly for using the Smartphone to connect to the **ROSM** you must terminate the execution of the **joystick_rosmaster.py** before you can control the **ROSM** with your smartphone.

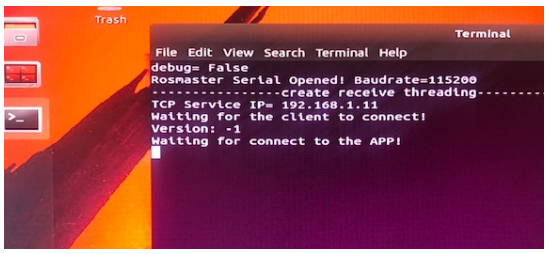
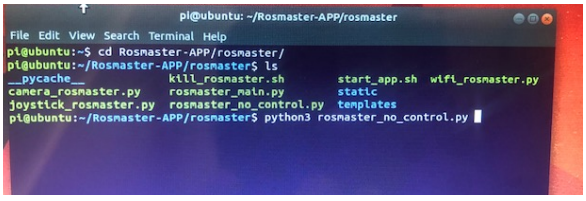
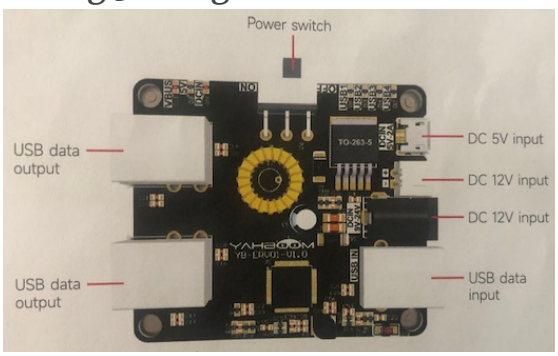
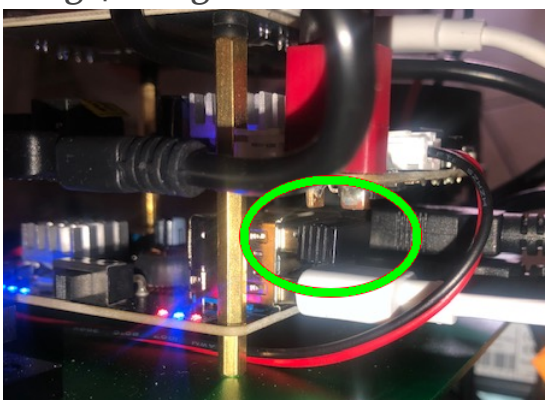
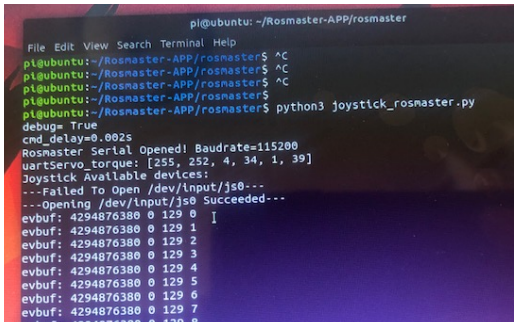
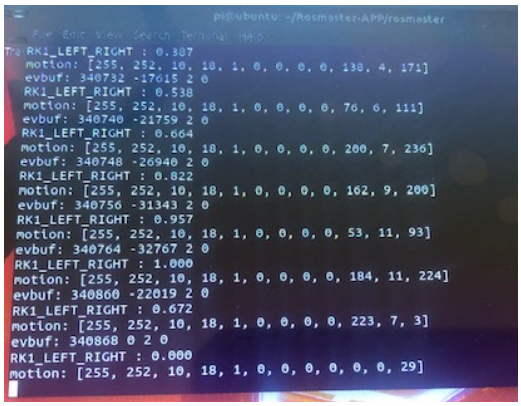
Controlling ROSM from a Gaming Remote			
#	Name	Steps	
Required Materials		• USB Keyboard • USB Mouse • Gaming Remote (Included with ROSM Kit)	
1	Boot up the **ROSM**	1. Boot up your **ROSM** a. Let the Boot sequence complete . b. The Ubuntu GUI should become visible with a terminal instance. 2. Type **ctrl+c** to terminate the default script	 Fig 1. Image of Successful Boot
2	Navigate to the python script	1. In the terminal run the command: a. **cd Rosmaster-APP/rosmater** b. **ls** 2. You should now see a script called **joystick_rosmaster.py** listed in the terminal. (See	 Fig 2. Navigate and List the directory contents

		Fig. 2)	
3	Connect USB Receiver to **ROSM**	1. Included in the **ROSM** kit is a Gaming Remote. It resembles a PS2 remote. 2. There should be a small USB Receiver and Two USB adapters in the remotes package. 3. Locate the Receiver that looks like the one depicted in **Fig 3** 4. Plug the USB receiver into the **USB HUB Board** (See **Fig 4** and **Fig 5**)	 Fig 3. Image of USB Receiver  Fig 4. Image of USB Hub Board  Fig 5. Receiver Connected to USB Hub
4	Run the script	1. Continuing from **Step 1** 2. Run the command: 3. **python3 joystick_rostmaster.py** 4. If the Receiver is not connected you will see an error like the one in **Fig 6** 5. Just plug in the Receiver (**Step 3**) and it will connect (See **Fig 6**)	 Fig 6. Failed to Find USB receiver

			 Fig 7. Connected to USB Receiver
5	Connect the Gaming Remote to the **ROSM**	1. Press and Hold the **Mode Button** until the green light is blinking. 2. Press the **Start button** to confirm the mode. The Green light should now stay solid green (**See Fig 9**.) 3. You should begin to see contents like **Fig 10** populate the terminal 4. The LED bar may also be changing colors. 5. You now should be able to control the **ROSM** with the gaming remote	 Fig 8. Image of the Gaming Remote Fig 9. Image of the Colors on the Remote  Fig 10. Image of the terminal being populated
A	Demo	https://youtu.be/fIy1VjNlL3M	

Astra Depth Camera

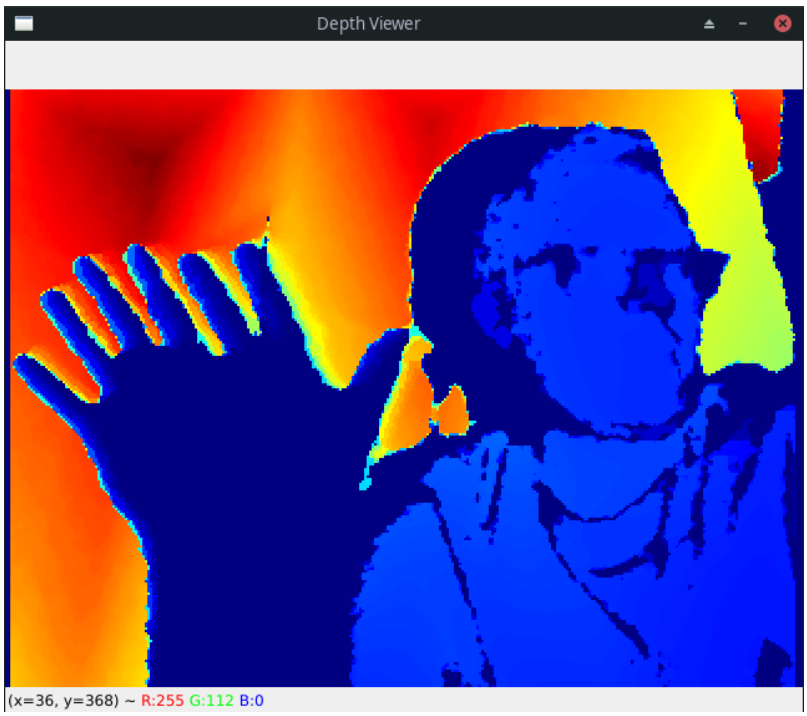
Notes:

- The guide below walks through the setup of the libraries for using the Astra Depth Camera.
- This guide requires the Raspberry PI of the **ROSM** to be connected to the internet.
- Much of the steps below come from the **ReadMe.md** in the python SDK wrapper Repo from Orbbec. (See Appendices).

Astra Depth Camera Setup		
#	Name	Steps
Required Materials		• Sample Python Scripts • USB Keyboard • USB Mouse

1	Install Depend-encies	• Run Command to Install Dependencies
		  ○ *sudo apt-get install python3-dev python3-venv python3-pip python3-opencv*
		  ○ *Pip install pybind11*
		• Install **CMake** Version **3.15** This can Take some time!
		  ○ Download the CMake 3.15 Source Code
		  ■ *wget https://cmake.org/files/v3.15/cmake-3.15.7.tar.gz*
		  ○ Decompress the archive file
		  ■ *tar -xzvf cmake-3.15.7.tar.gz*
		  ○ Navigate Decompressed directory
		  ■ *cd cmake-3.15.7*
		  ○ Build CMake
		  ■ *./bootstrap*
		  ○ Make the Application Binary
		  ■ *make*
		  ○ Install CMake
		  ■ *sudo make install*

		○ cmake --version
2	Clone the Repo	• Run the command: (Doing this, from Desktop is convenient, but can be done anywhere) *git clone https://github.com/orbbec/pyorbbecsdk.git*
3	Build the SDK	• Run the Commands: ○ *cd pyorbbecsdk* ○ *python3 -m venv ./venv* ○ *source venv/bin/activate* ○ *pip3 install -r requirements.txt* ○ *mkdir build* ○ *cd build* ○ *cmake -Dpybind11_DIR=`pybind11-config --cmakedir` ..* ○ *make -j4* ○ *make install*
4	Test a Sample Application	• Run these commands ○ *cd pyorbbecsdk* ○ *export PYTHONPATH=$PYTHONPATH:$(pwd)/install/lib/* ○ *sudo bash ./scripts/install_udev_rules.sh* ○ *sudo udevadm control --reload-rules && sudo udevadm trigger* ○ *python3 examples/depth_viewer.py* • *Result:*

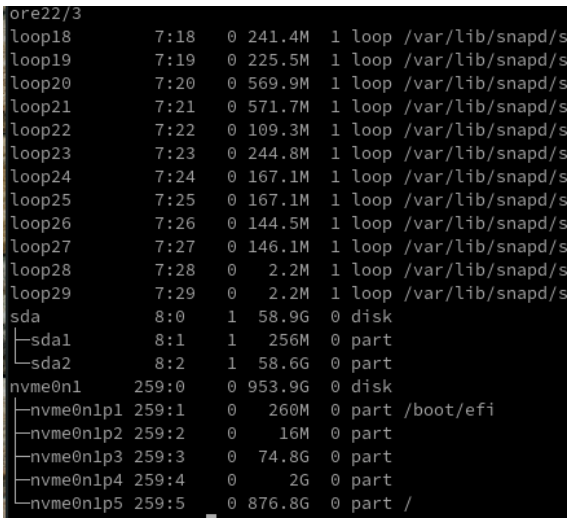
		 Depth Viewer (x=36, y=368) ~ R:255 G:112 B:0

Cloning a Linux SD Card

Notes:

- Because Setup of the Astra Camera required unusual things such as internet connection I provided this portion to quickly demo how to clone a Linux Image on an SD Card and Flash that Image to another SD Card. That way all the tedious setup and requirements are avoided.
- The guide below is for Linux Operating Systems

Astra Depth Camera Setup		
#	Name	Steps
Required Materials		• Etcher Application (dd, Balena, Rufus) • SD Cards
1	Plug the Linux Image SD into your machine	• Plug the SD card, that has the **ROSM** Kernel that would be used to boot up the robot, into your machine
2	Determine Source Disk information	• Enter the command to show the connected disks. ○ *lsblk*

		 • The card showed up as **/dev/sda** for me but often it will be **/dev/sdc**

3	Clone the Image to your machine	• Enter the command to clone to SD Card image to your machine:
		○ *sudo dd if=/dev/sda of=/home/ryan/Desktop/linux_image.img bs=4M status=progress*
		○ **NOTE:** The first path is the device node of the SD card determined in **step 2** the second path is wherever you want the image to be saved from the cloning. The command above will leave a **.img.gz** file on my desktop.
		`[ryan@rampart1 ~]$ sudo dd if=/dev/sda of=/home/ryan/Desktop/linux_image.img bs=4M status=progress`
		`9743368192 bytes (9.7 GB, 9.1 GiB) copied, 528 s, 18.5 MB/s`

4	Flash the Image	• The image can now be used to flash other SD cards, and it will have all the dependencies for the Astra Depth Camera already installed • Repo with Image file: https://github.com/Ryanjwoodward/gcu-rosmaster

Appendices

Resources

Rosmaster X3 Repository: http://www.yahboom.net/study/ROSMASTER-X3

Astra Camera SDK Home: https://www.orbbec.com/developers/orbbec-sdk/

Astra Camera SDK Python Wrapper: https://github.com/orbbec/pyorbbecsdk

Link to This Document in Google Docs:
https://docs.google.com/document/d/1v0PGlXtVrsT-hZDE15aiRbunsW5qmOCxTHBIC0iK36s/edit?usp=sharing

GitHub Repo: https://github.com/Ryanjwoodward/gcu-rosmaster