# **3rd Person Reality System**

**Luke Bird, Matthew Stradling, and Jason Andersen**

**ECEN 361**

**Winter Semester 2016**

# 

# 

# **Overview**

The 3rd Person Reality System starts with a quadcopter base controlled by a PixHawk Flight Controller. The PixHawk receives commands sent from a Raspberry Pi Zero attached to a headset worn by the user of the system. A GPS connected to the Pi tracks the changes in movement of the user. Then the Pi transforms those changes into flight commands that are sent to the PixHawk to cause the quadcopter to follow the user. A camera is mounted on the copter capturing video of the back of the user. The camera feed is streamed to the headset of the user providing a 3rd Person experience.

# 

# 

# **Resources**

**Sources**

1. Mission Planner

<http://ardupilot.org/planner/docs/mission-planner-overview.html>

1. First Time Setup

<http://ardupilot.org/copter/docs/initial-setup.html>

1. First Flight with the Copter

<http://ardupilot.org/copter/docs/flying-arducopter.html>

1. Communicating with Raspberry Pi via MavLink

<http://ardupilot.org/dev/docs/raspberry-pi-via-mavlink.html>

1. Making a Mavlink WiFi bridge using the Raspberry Pi

<http://ardupilot.org/dev/docs/making-a-mavlink-wifi-bridge-using-the-raspberry-pi.html>

6. Setting up the GPS

<https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/setting-everything-up>

7. Using UART instead of USB for GPS

<https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/using-uart-instead-of-usb>

8. MavProxy.py script

<https://github.com/ArduPilot/MAVProxy/blob/master/MAVProxy/mavproxy.py>

9. Using your GPS

<https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/using-your-gps>

10. Github repository with source code

<https://github.com/mstradling/3PR-System-Main>

**Materials**

1. Quadcopter

<https://www.amazon.com/QuadCopter-MultiCopter-Brushless-Propeller-Landing/dp/B015N2U1S0/ref=sr_1_3?ie=UTF8&qid=1480982384&sr=8-3&keywords=quadcopter+motor+kit>

1. PixHawk AutoPilot

<https://pixhawk.org/modules/pixhawk>

1. Raspberry Pi Zero

<https://www.amazon.com/CanaKit-Raspberry-Zero-Starter-Kit/dp/B01N3XNPAM/ref=sr_1_1/161-3440764-7148646?ie=UTF8&qid=1480982227&sr=8-1-spons&keywords=raspberry+pi+zero&psc=1>

1. Stackable USB hub for Raspberry Pi Zero

<http://makerspot.com/stackable-usb-hub-for-raspberry-pi-zero/>

1. Raspberry Pi WiFi dongle

<https://www.amazon.com/CanaKit-Raspberry-Wireless-Adapter-Dongle/dp/B00GFAN498/ref=sr_1_5?ie=UTF8&qid=1480982311&sr=8-5&keywords=raspberry+pi+wifi+dongle>

1. Ultimate GPS Breakout Board

<http://www.digikey.com/catalog/en/partgroup/ultimate-gps-breakout-board/54859?WT.srch=1&gclid=CjwKEAiAvZTCBRDvnoOaoa2j3xISJABxPjN94DDISAwJ6f2M8NUpjhNQM5XzJWUju7tJlCEVLhuKXhoCEQ_w_wcB>

1. Fat Shark Teleporter V3 Video Goggles

<http://www.getfpv.com/fatshark-teleporter-v3-video-goggles.html>

1. 3DR Radio Telemetry Kit 915 MHz

<https://www.amazon.com/YKS-Telemetry-915Mhz-Pixhawk-Quadcopter/dp/B0196LF6PW/ref=sr_1_1?ie=UTF8&qid=1481065229&sr=8-1&keywords=Radio+Telemetry+USB+antenna>

# 

# 

# **Procedures**

1. **Set up Mission Planner.**
   1. Mission Planner is a full-featured ground station application for the ArduPilot open source autopilot project. To set up mission planner please follow the link found under **Sources 1 and 2**. These 2 sources give a small explanation about what Mission Planner is, how to install it, and how to run it for the first time.
   2. Along with installation of mission planner there are very important links in the “First Time Setup” page. [Loading Firmware](http://ardupilot.org/copter/docs/common-loading-firmware-onto-pixhawk.html) is important to update the pixhawk, [Connect Mission Planner to AutoPilot](http://ardupilot.org/copter/docs/common-connect-mission-planner-autopilot.html) explains how to connect mission planner to the pixhawk, and [Configuring Hardware](http://ardupilot.org/copter/docs/configuring-hardware.html) will explain how to setup the configurations in mission planner. Make sure to follow all of the directions in the links to get an overall understanding of how mission planner works.
2. **First flight with only the transmitter remote.**
   1. Follow the link found under **Source 3.** This will direct you to a tab in ardupilot called “First Flight.” Under this tab there are 10 sub categories. Read through the page and follow what they say to do. This page will explain the basics about how to set up Mission Planner, how to arm your copter, and the basic concepts behind flying the copter for the first time and in general.
3. **Connecting the Raspberry Pi to the pixhawk.**
   1. **Sources 4 and 5**  explain how to connect the Raspberry Pi to the Pixhawk.
   2. This can be a little confusing because **Source 4** “Communicating with the Raspberry Pi via MAVLink” shows how to connect them using wires from 4 pins on the raspberry pi to the Telem2 port on the Pixhawk. This is not the only way to do this. This same connection can be made with a usb cable from the Raspberry Pi to the side of the Pixhawk or with the usb telemetry module. Both of these methods require the devices to be close to each other and connected via wires. If this is all you are trying to do then you only need to read through **Source 4.**
   3. We used a telemetry module found under **Materials 8.** If you also want to do this you have to use information from **Source 4 and Source 5** “Making a Mavlink WiFi Bridge using the Raspberry Pi.” This page expects you to set up your raspberry pi as an access point, you don’t need to do this so skip this part and go down to Installing and configuring MavProxy.
   4. There is a command that is used in both **Source 4** and **Source 5** to connect the Pi to the Pixhawk. “mavproxy**.**py **--**master**=/**dev**/**ttyUSB0 **--**baudrate 57600 **--**aircraft MyCopter” It is important to know that this command needs to change slightly depending on how to are trying to connect the Pi to the Pixhawk.
      1. If you are using a serial connection, shown in images 1 and 2, you change *ttyUSB0* to *ttyAMA0.*
      2. If you are using a usb cord directly or a telemetry module connected to the Pi via usb then use the *ttyUSB0*.
4. **Flying the copter with the Raspberry Pi.** 
   1. Once MavProxy is installed follow the instructions on the second half of the webpage of **Source 4**. In this source you will find a brief introduction to MavLink/MavProxy and your initial start with the program, connecting to the PixHawk on the quadcopter, and sending your first commands. **Image 3** shows an example of running the MavProxy code and making a connection.
   2. You may run into issues sending commands to the PixHawk as we did. (Refer to **Major Points To Remember** section) If you do then be sure to update the firmware on the PixHawk (through mission planner there is a button to update firmware when the copter is connected) and make sure that the GPS on the copter has a fix.
   3. The first flight command in MavProxy is “takeoff”. You must type the command with the distance, in meters, of how high we want the copter to rise on takeoff. In order for the PixHawk to accept this command from MavProxy in the Raspberry Pi, make sure that the copter is in GUIDED mode.
5. **Connecting the Teleporter video system to the copter.**
   1. The teleporter system is comprised of 7 parts (battery pack, headset, antenna x2, camera, power board, and 5G8 transmitter board). Connect the battery pack to the right side of the headset to power the headset. Attach one of the antennas to the headset. Secure the Raspberry Pi and the GPS module onto the headset (we used velcro but building some kind of secure holder onto it would be better). You also need to secure an additional battery pack onto the headset that will power the Raspberry Pi.
   2. Using AV cables connect the camera and the power board to the transmitter board. Then connect the second antenna to the transmitter board.
   3. Using tape, velcro, or a custom holder connect the materials in step b (camera and transmitter) to the bottom of the copter with the camera facing forward. (There is an arrow on the PixHawk indicating the forward direction.) We used velcro.
   4. Plug the XT60 plug of the copter’s battery to the PixHawk and the 4-pin white plug to the power board of the camera system to power it.
6. **Write a script to use GPS data to calculate Mavlink commands and send them to the Pixhawk.** 
   1. Become familiar with the GPS and get it set up. This can be done by following **Source 6 and 7** setting up the GPS and using UART for connecting with the Raspberry Pi.
   2. Copy the code found in mavproxy.py found at **Source 8** into a new file called mavproxy3pr.py.
   3. Follow the instructions on **Source 9** on using the GPS in code. Use this code to gather GPS speed and heading values.
   4. Based on the difference of those values send a specific command through the MavProxy script. For example to set the speed and direction of the copter from the GPS we would use a line of code similar to: setvelocity(gps.xspeed, gps.yspeed, gps.zspeed) (not actual syntax). When the host turns their head (the GPS turns) you update your current heading by subtracting the initial heading calculated at the startup of the system. Then you call a function similar to: setyaw(heading) (not actual syntax).
   5. To see the actual code used to transform GPS values to MavProxy commands please refer to **Source 10**. This is our github account for the project.
   6. This script will be used instead of the original mavproxy.py to send commands to the PixHawk.
7. **Configure Raspberry Pi to begin Mavlink on Boot up.**
   1. Refer back to **Source 4**. If this step is not already done, follow the directions under the **Configure MavProxy To Always Run** section. This will enable your MavProxy script to start on startup of the Raspberry Pi.
8. **Test system and correct for errors.** 
   1. At this point we would just test the system and debug to fix any errors.

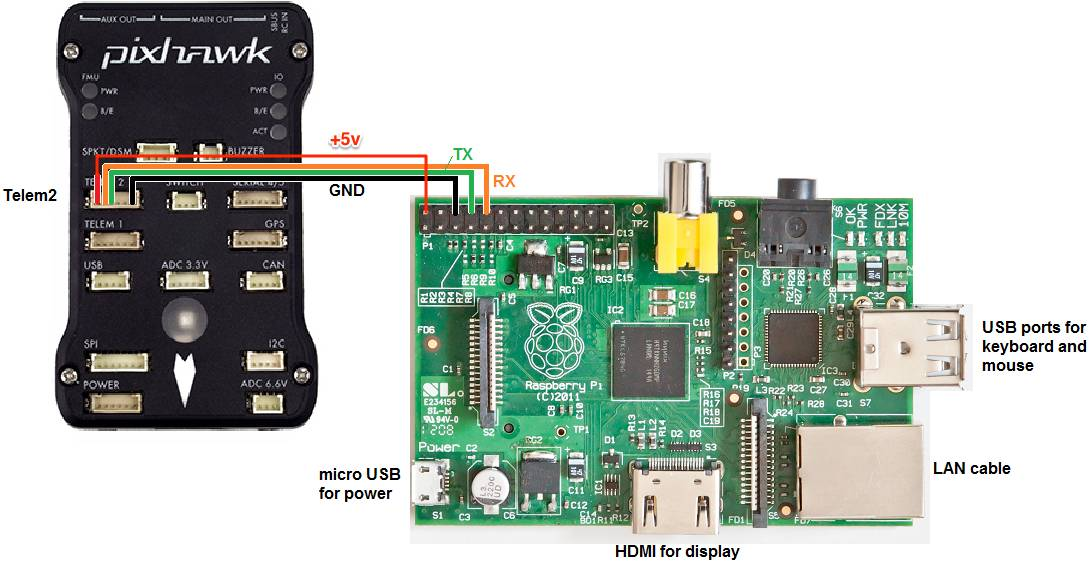
**How Far We Got**

By the end of the semester we got to step 6 of the procedures. If you refer to the **Major Points to Remember** section you’ll read through the many major problems that we encountered and overcome. We also had to learn how to quadcopter works from the ground up. We had a steep learning curve ahead of us. We did make some great progress and finished the semester by wrapping up the GPS/MavProxy code. We succeeded in executing flight commands from the Raspberry Pi and were able to perform a successful takeoff of the quadcopter. We were able to mount all the parts and finish the code. The only thing remaining was to fix the kinks out of the code and flight of the system for a more bug-free experience.

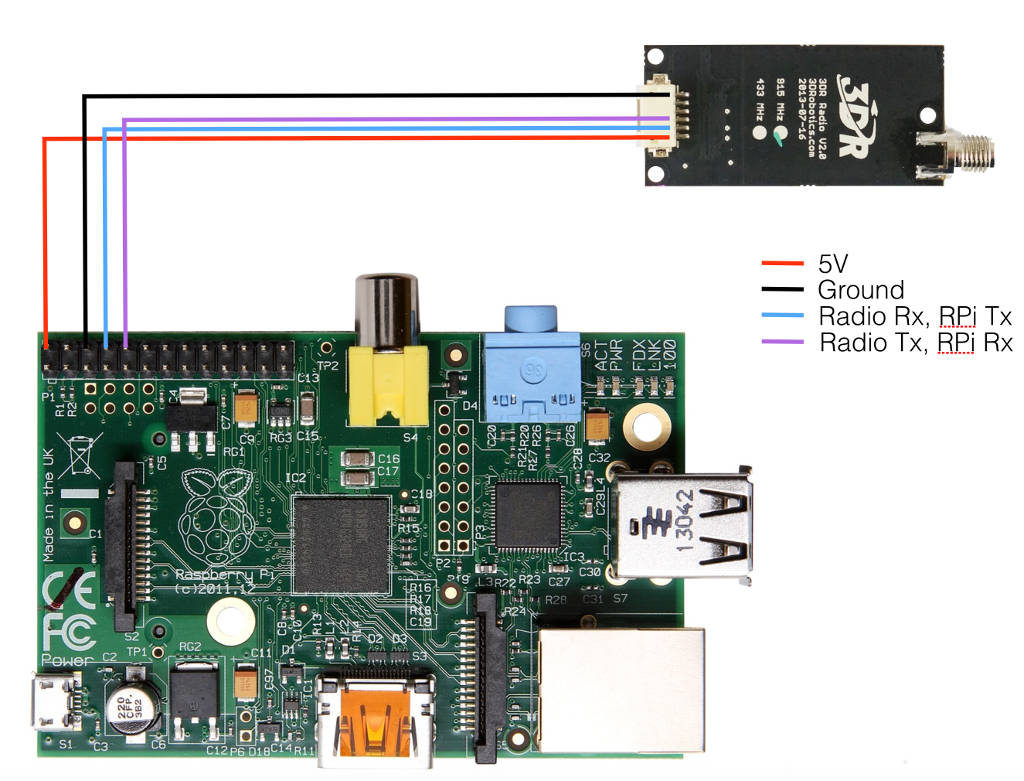
# 

**Images**

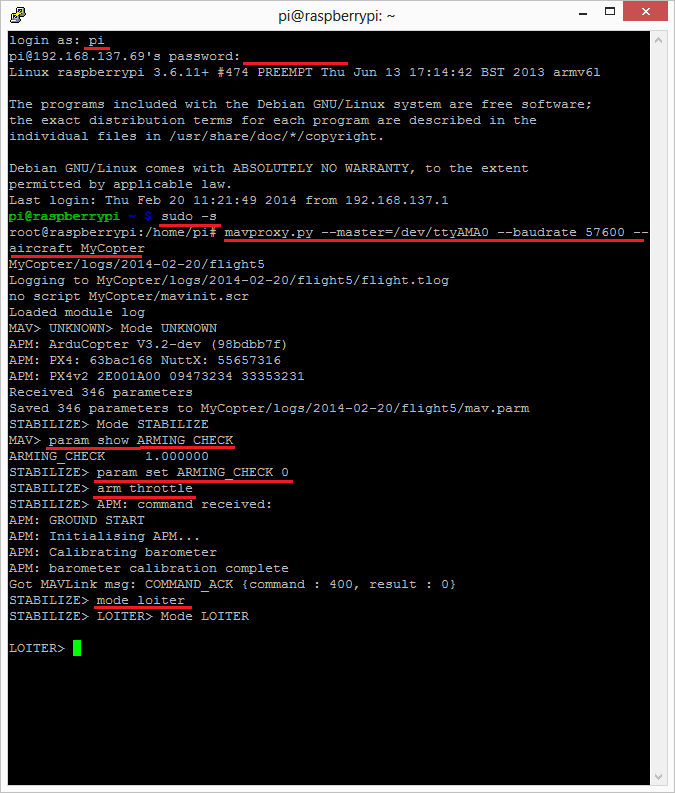
**Image 1 “Pi to Pixhawk Connection 1”**



**Image 2 “Pi to Pixhawk Connection 2”**



**Image 3**



# 

# **Major Points to Remember**

While working on this project we encountered several major problems that took us a long time to fix. Here is a list of the top problems we encountered and how to correct them.

1. Copter quirks. When we first started working with the quadcopter we didn’t understand certain things it was doing or how it worked in general. We had a lot to learn from the ground up.
   1. Example: when we would start the motors of the copter while the copter remained on the ground, 2 of the motors would stop spinning and the others would start spinning faster. We thought we needed to calibrate the motors or something else must have been wrong. After talking with someone who knows copters very well we found out that it this was a normal occurrence. The copter was not level, while standing on the ground and the copter was trying to compensate the angles that the copter was residing with. It was doing this by taking power from a motor or two to give that power to the other motors. The solution to this issue was to quickly get the copter off the ground. We tested that and sure enough the copter flew just fine.
   2. Lesson to learn is that if something is happening with the quadcopter itself try and talk to someone who has more experience to see if it is a normal problem with an easy fix before spending a lot of time trying to fix it.
2. When initially connecting the Raspberry Pi to the pixhawk through Mavlink/MavProxy we were able to receive information but could not send any commands.
   1. The solution was that the pixhawk firmware was out of date and it simply needed to be updated. Make sure to get the latest software for the pixhawk. Once this was done the pixhawk recognized when we sent it commands from the Raspberry Pi.
3. Once we were able to send commands none of our commands were working. We looked up tutorial after tutorial trying to figure out how to send a command to the pixhawk successfully. We looked into DroneKit and several other sites with MavLink commands and none of them were working.
   1. First solution: Instead of trying to use commands found in tutorials simply type “help” into the command line when you are running the MavProxy.py script and a list of commands will come up. The “velocity” command is very useful for specifying directional movements.
   2. Second: When we sent the correct commands from the Raspberry Pi to the copter, the copter would acknowledge that it received the commands but would not execute them. We discovered that in order for the copter to execute any command sent to it from MavLink in the Raspberry Pi, the copter had to have a functioning and connected GPS system on the copter (with a fix on the current location) and the copter had to be in GUIDED mode. Once the GPS had a fix on its location and the copter mode was switched to GUIDED mode a command sent from the Raspberry Pi would execute successfully.