Piksi Multi ArduPilot Integration Guide

Integrating Piksi Multi into an ArduPilot-based UAV

Caution	*You can irreparably damage Piksi Multi if power is supplied to the wrong pins. Always triple-check and independently verify any custom cabling or advanced electrical integration.*

Caution	*Piksi Multi uses a powerful processor that can generate a significant amount of heat. Provide adequate airflow to Piksi Multi. Use caution when handling the board, as components may reach 140° F (60° C).*

Overview

With the release of Piksi® Multi, Swift Navigation provides a GPS capable of high-availability, high-accuracy positioning for UAVs, with very fast acquisition time thanks to a dual-band L1/L2 architecture. This guide is intended to help you evaluate Piksi Multi in UAV applications. It walks you through integrating Piksi Multi into an ArduPilot and Pixhawk-based UAV. ArduPilot supports Piksi Multi out of the box, and provides an ideal platform to evaluate and familiarize yourself with Piksi Multi's requirements and performance on UAV platforms. On this platform, Piksi Multi can be used in real-time to increase the accuracy of UAV trajectory following and autonomous landing. Additionally, Piksi Multi can be used to accurately position and time-stamp flight data for post-processing and debugging, since it writes all GPS data into ArduCopter's Dataflash logs during flight.

Before You Start

First, work through out Piksi Multi Getting Started Guide, achieve an RTK Lock using our evaluation kit, and read our GPS and GNSS FAQ.

For Advanced Users

We outline advanced options, such as connecting to a remote NTRIP observation source or using MAVProxy as an observation router. We suggest first following our recommended integration, then expanding your setup with our additional instructions.

For System Integrators

Piksi Multi is enabled on Ardupilot by a device driver, which adds support for the Swift Binary Protocol, version 2. If you are attempting to build a custom integration into a different control or estimation system, this infrastructure can serve as a useful template. Additionally, see our Integrating Piksi to Your Application guide.

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Prerequisites and Materials Required

You will need to acquire:

- 1x Piksi Multi Evaluation Kit, which includes 2x Piksi Multi GNSS Modules.
- 1x L1/L2 RTK GPS Antenna appropriate for UAVs. We recommend one of:
  - Maxtenna m1227hct-a2-sma helix, available for purchase here
  - Harxon HX-CH6601A or its predecessor, the Harxon HX-CH4601A
- Cabling and connectors to wire Piksi Multi to a Pixhawk
  - 20-pin ribbon cable (included with Multi Evaluation Kit)
  - DF13 connector or cable
  - Tools to create custom cabling, such as soldering station or crimping tools

Overview

Onboard UAV

- UAV-appropriate Antenna
- Piksi Multi GNSS Module
- Custom interconnect cable
  - Piksi Multi UART1 connected to Pixhawk Serial 4/5
  - Piksi Multi Power connected to 5-15v UAV power supply.
  - (Optional) USB connector or UART connector for debugging
- Standard GPS with onboard magnetometer as secondary GPS.
- Standard UAV MAVLink Telemetry Radio

Ground Station

- Survey Antenna
- Piksi Multi GNSS Module
- Evaluation board
- UART to USB cable; connecting evaluation board to Piksi Multi
- Ground Station Laptop
  - Swift Console, broadcasting base station observations over UDP
  - Mission Planner, injecting base station observations into MAVLink telemetry stream
- Standard UAV MAVLink Telemetry Radio

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

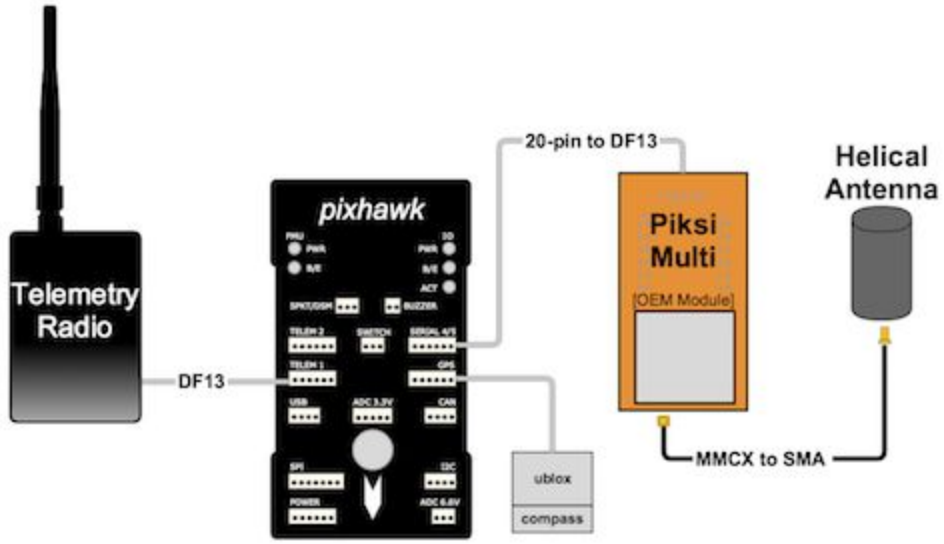
Providing RTK Corrections to Piksi onboard UAV

RTK GPS produces centimeter-accurate positions by correcting its local GPS observations using corrections received from a base station. Piksi supports multiple ways to receive these corrections. In this guide we show how to piggyback observations from a ground station over the standard ArduCopter telemetry radio link, using a feature called **GPS Inject.** Our recommended setup uses the Mission Planner ground station.

For advanced users, the MAVProxy modular command-line interface also supports this feature, and can be used to route messages between any MAVLink ground station. See our notes on MAVProxy in the Advanced Options section of this document.

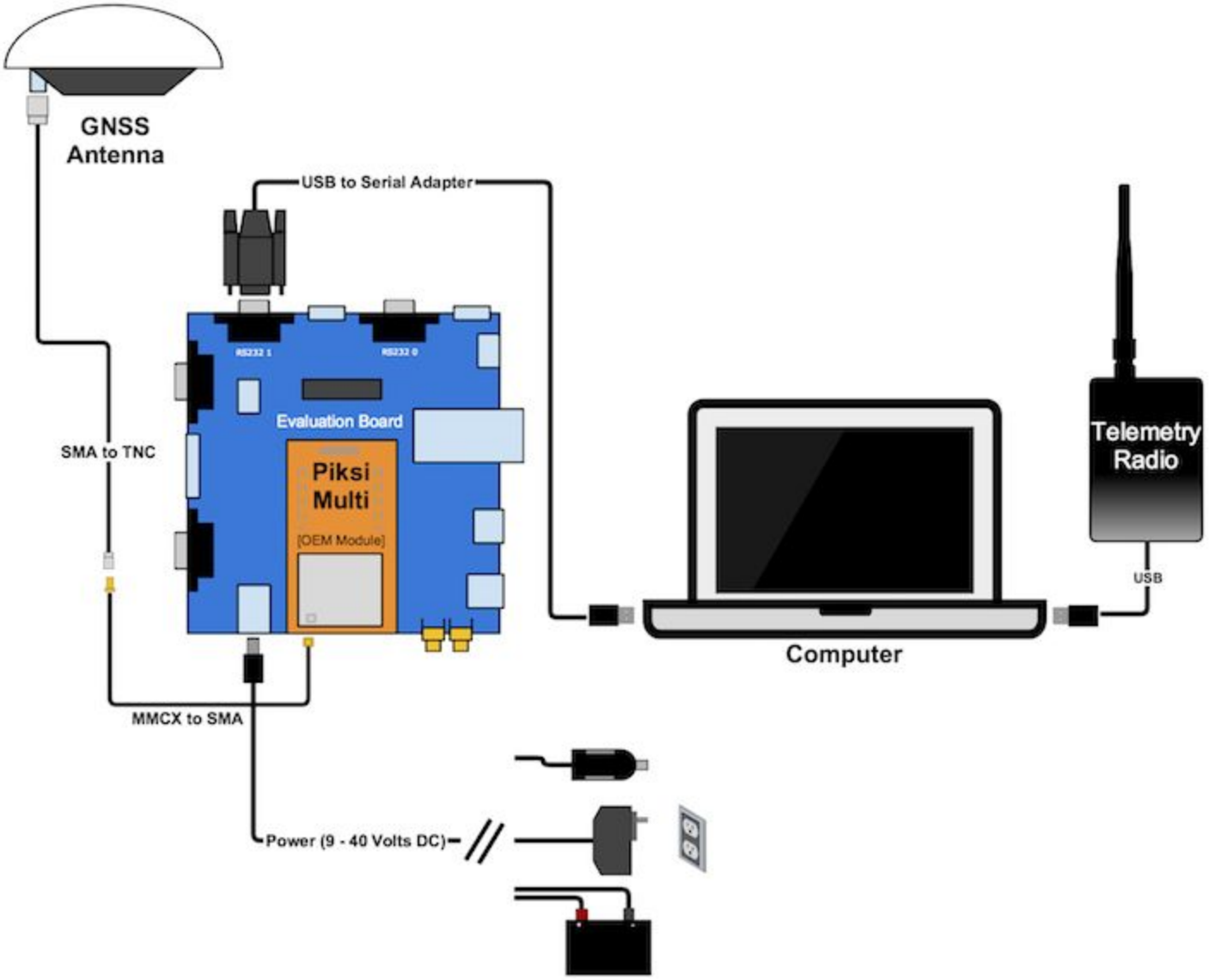
System Diagram

Onboard UAV



©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Ground Station



©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

One-time UAV Setup

We configure Piksi Multi's UART1 to connect to Pixhawk's Serial 4 / 5 connector. Using this connection, Piksi Multi sends both navigation and observation data to Pixhawk, for ArduCopter to perform real-time positioning and record logs for post-processing. Over the same connection, ArduCopter forwards real-time observation data from a ground station to Piksi. We also mount a GPS antenna on your UAV, and physically connect Piksi Multi's 20-pin output header to the Pixhawk.

Configure UAV Piksi

We configure Piksi to send all data to the Pixhawk flight controller, where ArduCopter can perform real-time flight control. Additionally, ArduCopter will log all the emitted SBP messages to its dataflash log for debugging and post-processing. If you want to customize the specific messages logged in the dataflash log, or customize Piksi Multi's behavior beyond this introductory setup, please consult our Knowledge Base at http://support.swiftnav.com/, including our Settings Guide.

1. Plug this Piksi Module into your Evaluation Board
2. Connect using your Swift Console.
3. **Ensure you have the latest firmware. ArduPilot Requires Firmware 1.1.26 or later**
    a. Click the "Firmware Update" tab
    b. If your firmware is not the latest firmware, follow the firmware upgrade steps.
4. **Reset settings to default**
    a. Click the "Settings" tab.
    b. Click "Reset to Defaults"
5. **Set solution settings**
    a. Click the "Settings" tab.
    b. Set the following values:

Section	Setting	Value	Purpose
solution	soln freq	5	Set the solution output rate to 5 Hz
solution	output every n obs	1	Emit observations for every solution

solution	
elevation mask	10
soln freq	5
correction age max	30
output every n obs	1
dgnss solution mode	Low Latency
send heading	False
heading offset	0

Note: The ArduPilot uses only following SBP messages from Piksi Multi: HEARTBEAT (ID 65535), GPS_TIME (258), VEL_NED (526), POS_LLH (522), DOPS (520) and EXT_EVENT (257) so ensure those messages are enabled on

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

UART1 (field enabled_sbp_messages). If any other SBP messages are enabled then they'll be recorded for future postprocessing / troubleshooting.

Configure ArduPilot Parameters to use Piksi Multi as primary GPS.

In this section, we set ArduPilot parameters to use the Piksi Multi GPS appropriately. In this default configuration, Piksi is wired to the SERIAL4/5 port on Pixhawk, and will be detected as **GPS2**, while your stock u-blox GPS module remains wired to the the GPS port on Pixhawk, and will be detected as **GPS**. We configure automatic switch-over so the best possible GPS will be used whenever available.

1. Connect Mission Planner to your UAV.
2. Ensure you are running a compatible ArduPilot version:
    a. **ArduCopter v3.5-rc5 or later**
    b. **ArduPlane v3.8 or later**
3. Set the following parameters:

Param Name	Value	Purpose
GPS_TYPE	1	Auto-detect GPS
GPS_TYPE2	1	Auto-detect GPS2
GPS_SBP_LOGMASK	-1	Log all SBP messages to dataflash. (-1 equates 0xFFFF)
GPS_AUTO_SWITCH	1	Switch to best available GPS for vehicle control, without blending.
SERIAL4_PROTOCOL	5	Set Serial4 to GPS mode
SERIAL4_BAUD	115	Set Serial4 to 115200 baud
EK2_ALT_SOURCE	2	Use GPS as the preferred altitude source for EKF2

Using Piksi Multi ONLY for logging

You can use Piksi Multi only for logging. Set `GPS_AUTO_SWITCH` to `0`, and only the first (conventional) GPS until will be used for navigation.

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

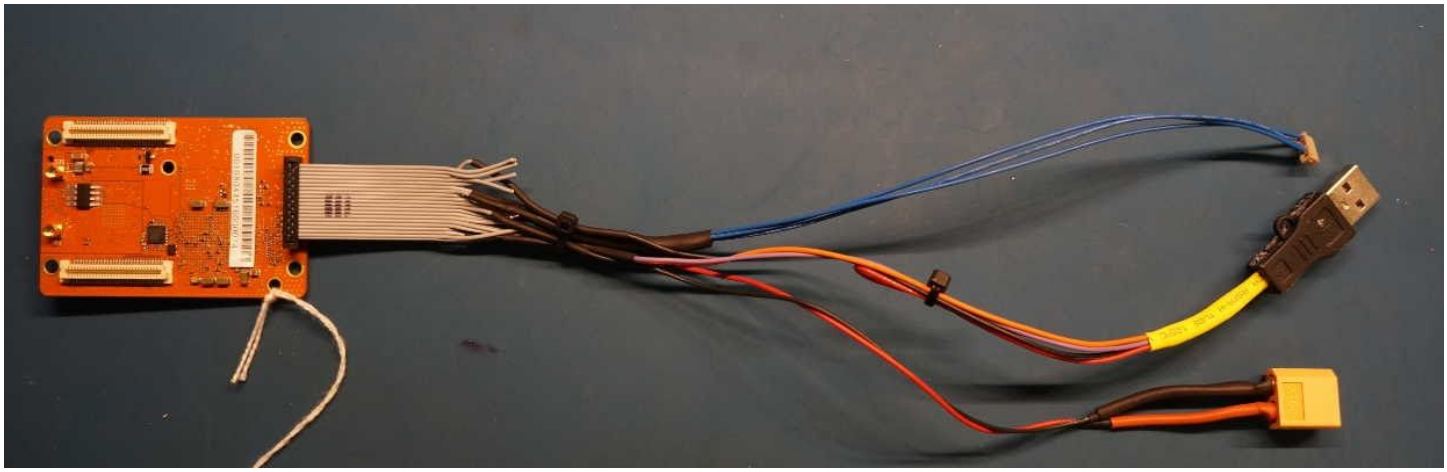
Connect Piksi Multi to Pixhawk

Swift Navigation is currently working on a plug-and-play solution to connect Piksi Multi to Pixhawk. In the meantime, we show how to create a custom interconnect cable using the included 20-pin ribbon cable from your evaluation kit.

Piksi Multi Pinouts: Refer to our Piksi Multi Hardware Specifications for voltage requirements and pinouts.

Pixhawk Pinouts: Refer to Pixhawk's Datasheet for connector pinouts.

Example Interconnect Cable



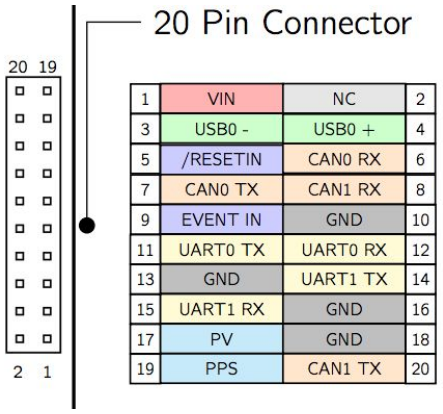
Requirements

- 20-pin ribbon cable (Connector: TLE-110-01-G-DV-A-TR)
- DF13 cable or connector (female), to connect Piksi UART1 to Pixhawk's SERIAL4/5 input
- Connector or direct wiring into a power rail on the UAV.
    - Power either from Pixhawk UART output, or from battery rail.

Cabling Pinout

Piksi Multi 20-pin Connector Pin	Pixhawk SERIAL 4/5 Pin	Purpose
1 (Vin)	1 (VCC +5v)	5V to 15V (3 watts). Can wire to Pixhawk UART 5V output, or directly to 2S/3S battery rail.
10 (GND)	6 (GND)	Common ground
14 (UART1 TX)	3 (#4 RX)	Data output from Piksi Multi to Pixhawk (navigation)
15 (UART1 RX)	2 (#4 TX)	Data input from Pixhawk to Piksi Multi (corrections)

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1



1	VIN	NC	2
3	USB0 -	USB0 +	4
5	/RESETIN	CAN0 RX	6
7	CAN0 TX	CAN1 RX	8
9	EVENT IN	GND	10
11	UART0 TX	UART0 RX	12
13	GND	UART1 TX	14
15	UART1 RX	GND	16
17	PV	GND	18
19	PPS	CAN1 TX	20

Notes on Powering Piksi Multi

Piksi Multi requires ~3 Watts, at 5 to 15 volts. On the low end of the power range (from 5 volts down to 4.75 Volts), the DC bias for the antenna begins to drop, so a 5 V DC power source this is not recommended for antennas that need at least 4.5 Volts DC bias for operation. **Do not power Piksi from a servo rail.**

Caution	***You can irreparably damage Piksi Multi if power is supplied to the wrong pins. Always triple-check and independently verify any custom cabling or advanced electrical integration.***

Mount Antenna on UAV

Your antenna placement is the single most important factor in successfully deploying RTK GPS on a UAV. Your antenna needs to satisfy two important criteria:

1. Antenna needs a clear skyview at all times for all expected aircraft orientations.
2. Antenna needs to avoid any sources of electromagnetic interference, such as radios or CPUs.

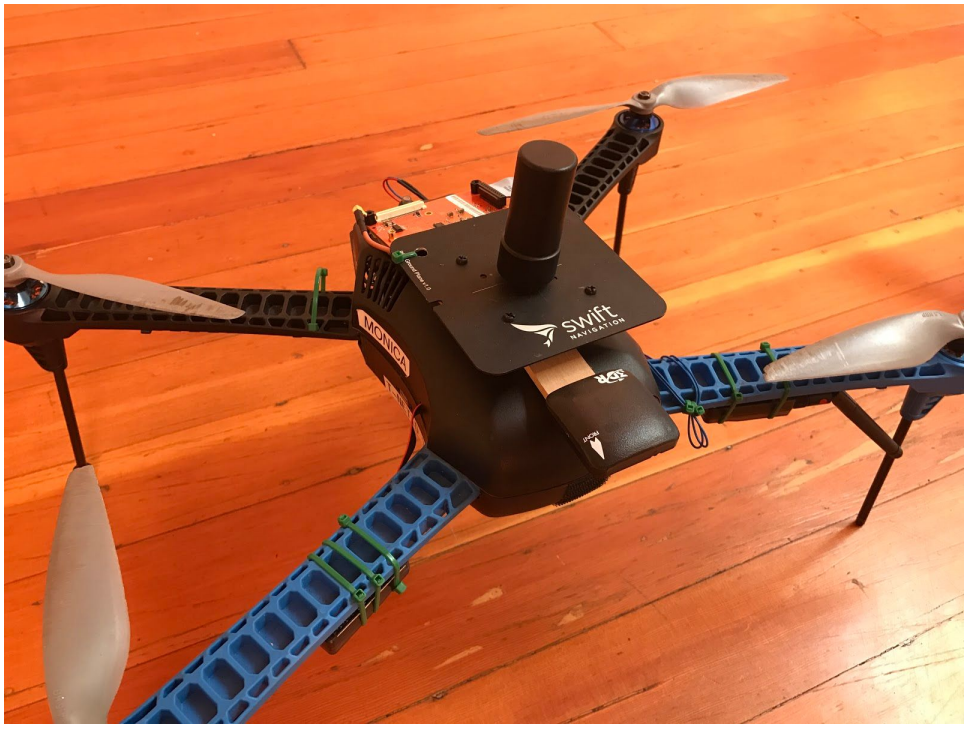
Helical Antennas We have a slight preference for helical antennas, since they provide a very omnidirectional radiation pattern, although we have had success with patch antennas as well.

Ground Planes Most antennas need some form of ground plane.

Avoid Interference We have also found that Pixhawk and the common SiK-based radios such as the 3DR Radio and the RFD900 produces interference that degrade Piksi Multi's performance. Thus we recommend mounting antennas as far from these devices as possible, and shielding antennas using the largest practical ground planes.

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

3DR Solo with Maxtenna m1227hct-a2-sma on custom ground plane. Piksi mounted in accessory bay.



3DR Iris with Harxon HX-CH4601 on custom ground plane. Piksi Multi mounted on rear.

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Configure SiK Telemetry Radios for Optimal RTK Performance

The default settings for SiK-based telemetry radios such as the popular RFD900 or 3DR radios are not optimized for sending RTK corrections. We observe high data loss and high latency using the default configuration, which deteriorates our RTK solution.

Specifically, we change the following settings of the SiK radios:

Radio Setting	Correct Value	Explanation
ECC	OFF	Disables overhead of error correction code
Mavlink	Raw Data	Enables up to 255 bytes per packet

Using Mission Planner

- Power on your UAV. Do NOT connect to it in mission planner.
- Plug your USB cable into your ground station telemetry radio
- Go to Initial Setup, select SiK Radio, click Load Settings, and wait for parameters to load.
- Ensure you have the latest firmware!
    - See this document for firmware upgrade instructions: http://ardupilot.org/copter/docs/common-3dr-radio-advanced-configuration-and-technical-information.html
- Clear the ECC field, and choose Raw Data in Mavlink for both local and remote radios
- Click Save Settings.

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

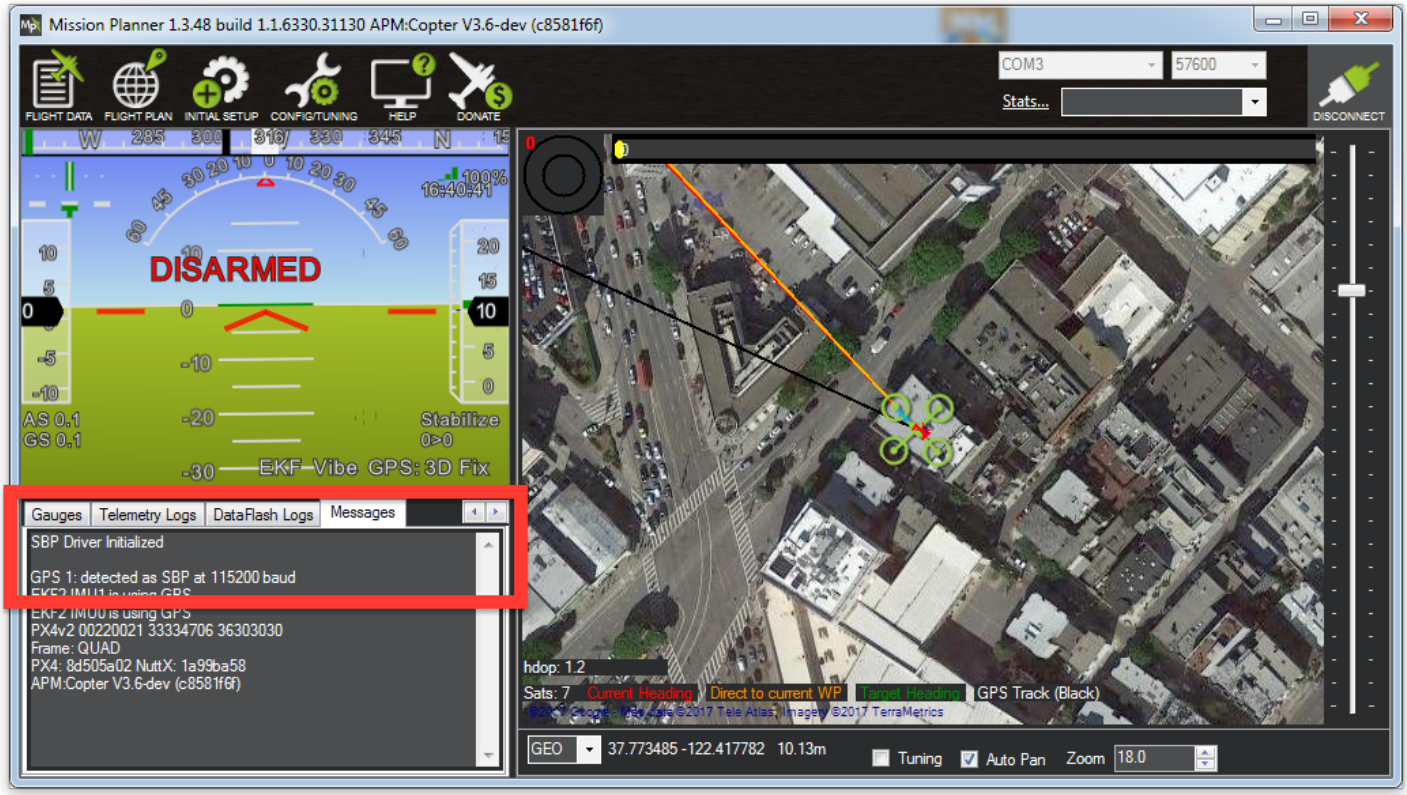
Confirm that ArduPilot detects Piksi Multi

At this stage you are ready to power your UAV. Connect Mission Planner to your Pixhawk, confirm Piksi is powered. You should see two indicators of ArduCopter detecting Piksi Multi successfully: debug output, and status output

Debug Output

You should see the following message appear in your “messages” tab when Piksi is detected. If you don’t see this, try power-cycling Piksi Multi or Pixhawk - this message is only generated when the ArduPilot driver is initialized.

```
APM: GPS 1: detected as SBP at 115200 baud
APM: SBP Driver Initialized
```



Status Output

You should also see both GPS and GPS2 data appear. In Mission Planner, enable “Tuning”, and check both “gpsstatus” and “gpsstatus2”. You should see a nonzero GPS fix status for both GPS units.

Confirm Using Simulation

You can also use Piksi Multi’s built-in simulator to confirm that ArduPilot is correctly receiving data. Enable the simulator as described in section “Testing Integration with Piksi Multi’s built-in simulator”. You should see GPS data appear in your Tuning window, and in your map. *Note: If your conventional GPS has a lock, in conflict with the simulator’s simulated position, ArduPilot will reject the simulated GPS output.*

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Per-Flight Ground Station Setup

Every RTK-powered flight requires a ground station, where a Piksi Multi GPS observes the current GPS constellation and transmits RTK Corrections to the Piksi Multi on the UAV. In this section we discuss configuring your own ground station consisting of:

- GPS antenna
- Piki Multi mounted on evaluation board
- ground station laptop running Mission Planner
- telemetry radio. We transmit corrections over the same radio your UAV normally uses.

We assume you've already acquainted yourself with our Getting Started Guide, installed the Swift Console, and successfully achieved RTK Lock using our Evaluation Kit.

Setup Base Station Piksi

Place Antenna

We recommend using the mini-survey GNSS antenna included with your evaluation kit. Your antenna placement is the single most important factor in successfully deploying RTK GPS! Mount this antenna with a clear view of the sky and far away from sources of electromagnetic interference, such as radios or computers.

Configure Piksi

Use the Swift Console to configure your ground station Piksi Multi, the same way you configured your onboard Piksi Multi.

Section	Setting	Value	Purpose
solution	soln freq	5	Set the solution output rate to 5Hz
solution	output every n obs	5	Emit observations for every solution
surveyed position	broadcast	True	Enable transmitting the surveyed location of the base station. **Required by Pixhawk UAV Driver**

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

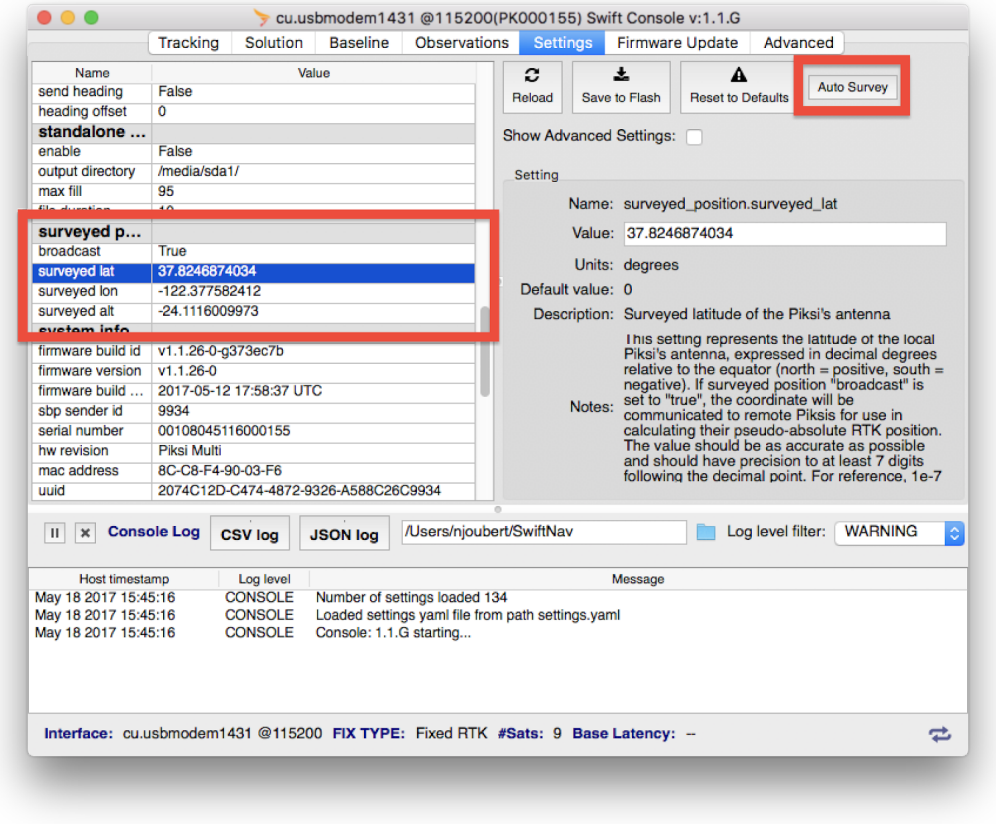
Survey Ground Station Location

The ground station Piksi Multi must transmit its absolute location to the UAV for the UAV to calculate a centimeter-accurate position. This is a current limitation of ArduPilot - even if you are only interested in relative positioning, ArduPilot requires an absolute position.

You need to survey the location of your GPS antenna whenever you move to a new location, and save this location into the settings of your ground station Piksi Multi. **We make this easy through the "Auto Survey" functionality in the Swift Console.**

To survey your current ground station location:

1. Finalize your antenna placement.
2. Open Swift Console, and connect to your ground station Piksi
3. Go to the "Settings" tab
    a. Click on the setting "surveyed lat" in the "surveyed position" category.
    b. An "Auto Survey" button should appear in the top right.
    c. Click on the Auto Survey button.
    d. **Set "Surveyed position", "Broadcast" to "True"**
4. Confirm that the "surveyed lat", "surveyed lon" and "surveyed alt" settings are populated.
5. Confirm that "broadcast" under "surveyed position" is set to true.
6. Save to flash



©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Send RTK corrections to UAV using Swift Console and Mission Planner

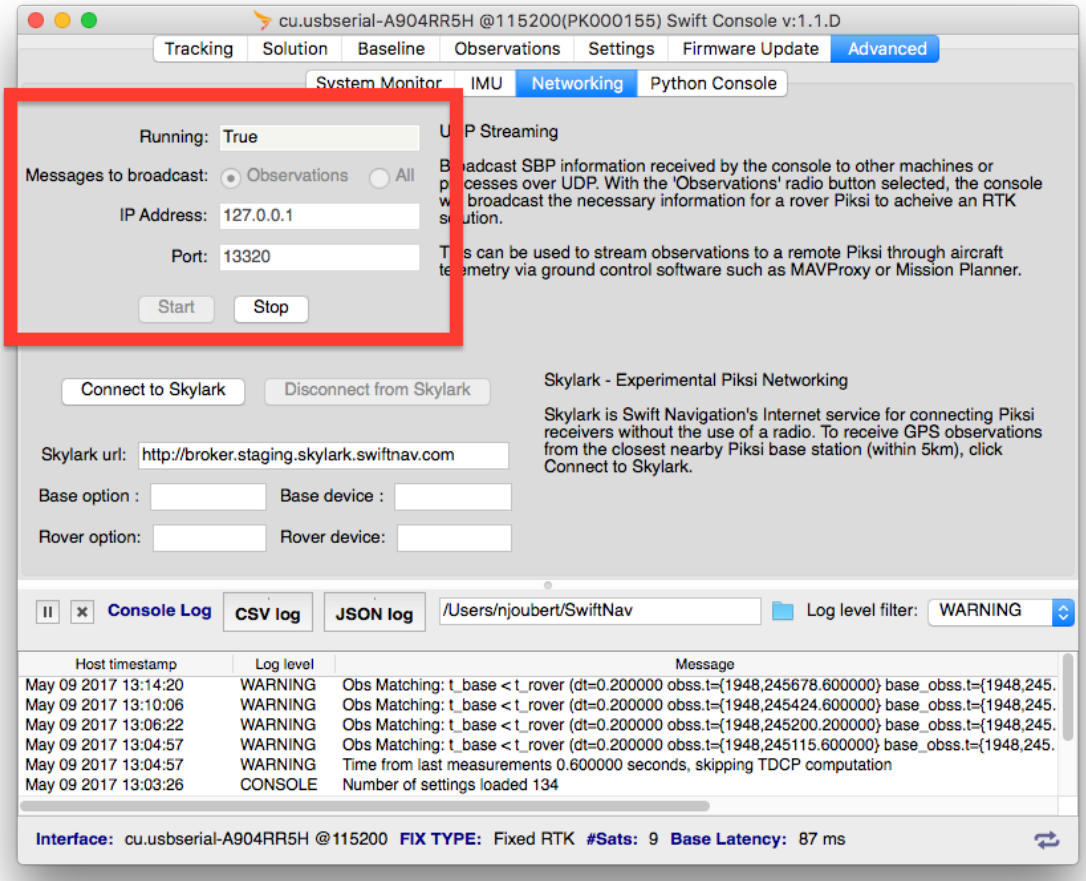
Your ground station Piksi Multi should now be surveyed and observing the GPS constellation. We will now use the Swift Console and Mission Planner to route these observations from the ground station Piksi to your UAV's Piksi.

Use Piksi Console to broadcast SBP Observations over UDP

In Piksi Console:

- Go to the "Advanced", then "Networking" tab.
- Under the "UDP Streaming" section
    - Set the "Messages to broadcast" to "Observations"
    - Set the IP Address to 127.0.0.1, Port 13320 (default values)
- Click "Start". Confirm that "Running: True" appears.
- The console is now forwarding observations over UDP which the ground station can inject onto the main telemetry stream. Keep the console running for the duration of your flight!

Advanced users might be interested in the section "Broadcast SBP over UDP from the command line" of this guide.

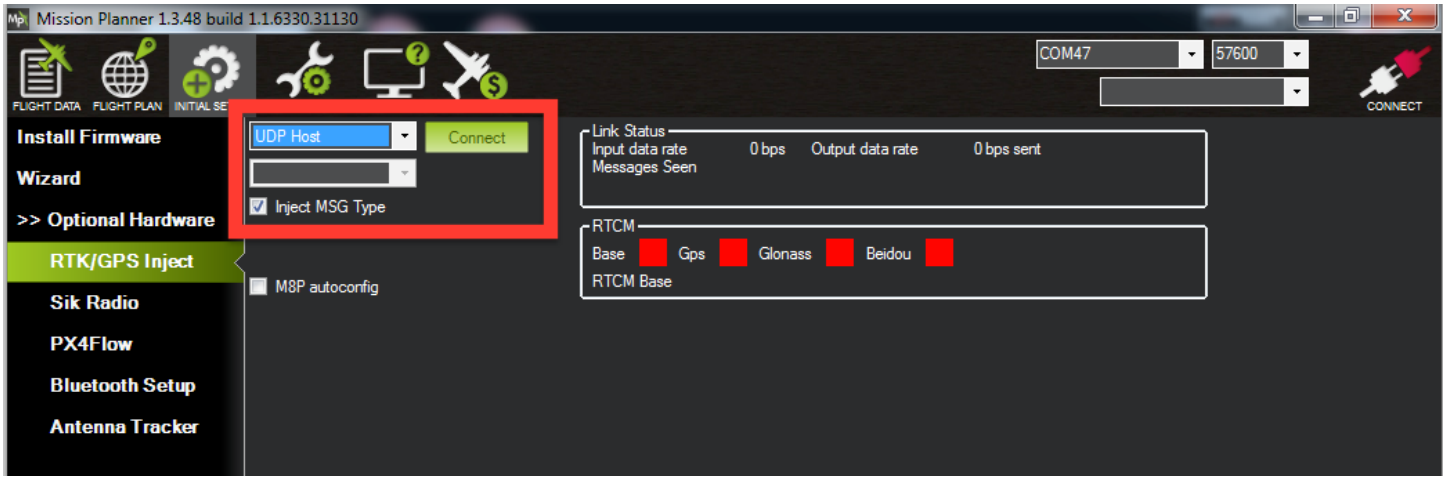


©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Use Mission Planner to inject SBP Observations from UDP into telemetry radio stream

Mission Planner needs to inject the RTK Corrections from the Swift Console for the duration of your flight. Here's how:

- Open Mission Planner. Ensure you have the latest build, >v1.3.48
- Connect to your UAV using your telemetry radio as normal.
- Go to **Initial Setup, Optional Hardware, RTK/GPS Inject** option
- Select **UDP Host** in the drop-down. You can ignore the baudrate dropdown below.
- Ensure **new RTCM msg** is checked
- Click **Connect**
    - **Enter Local Port: 13220**



- Confirm that the "Link Status" box shows corrections being transmitted.

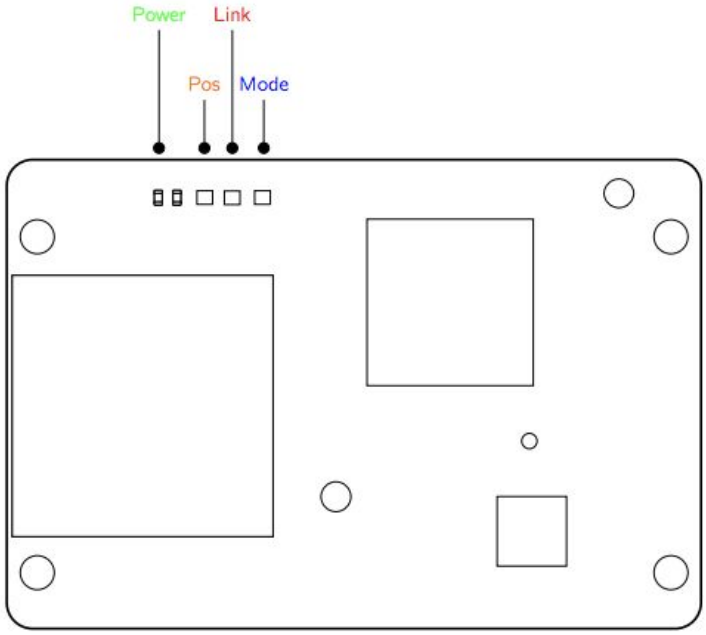
Note: You can also use MAVProxy to inject GPS Observations into your telemetry radio stream. This approach might be more appealing to advanced users. See the section "Use MAVProxy as a command-line ground-station & RTK correction router".

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Expected Behavior

If everything is working properly, we expect:

- UAV Piksi Multi
  - The POS LED is a solid yellow. Indicated position is available
  - The LINK LED blinks red. Indicates corrections being received
  - The MODE LED is a solid blue. Indicated RTK is available in Fixed mode.
- Mission Planner
  - GPS2 status is “6”
  - HUD displays “RTK GPS” as fix level.



LED Name	Color	State	Description
POWER	LED Off	Off	No Power
	Green	Continuously On	Module receiving power
POS	LED Off	Off	No solution, antenna not detected, no GNSS signal received
	Yellow	Slow Blink	No solution, Antenna detected, no GNSS signal received
	Yellow	Fast Blink	No solution, GNSS signal received
	Yellow	Continuously On	GNSS Solution Available (any kind)
LINK	LED Off	Off	No incoming corrections or Internet access
	Red	Flashing	Incoming corrections, no Internet access
	Red	Continuously On	No Incoming corrections, Internet access
	Red	Continuously On and Flashing (occulting)	Internet Access and incoming corrections
MODE	LED Off	Off	No RTK
	Blue	Blinking	Float RTK mode
	Blue	Continuously On	Fixed RTK mode

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Pre-Flight Checklist

Here's a detailed checklist for RTK to function correctly. Run through these steps before every flight!

- Both Piksi Multi units have the same firmware, at least v1.1
- Placement
    - Place and fix base station antenna
    - Place UAV
- Power On
    - Power on base station Piksi
    - Power on UAV.
- Wait for Base station GPS lock (POS LED is solid yellow)
- Open Swift Console
    - Survey Base station GPS position using Swift Console's settings tab. Select one of the "surveyed position" latl/lon/alt setting entries for the auto survey button to become visible.
    - Set Surveyed Position Broadcast to True using Swift Console's settings tab.
    - Check the "Observations" tab, your "local" section should have multiple rows of observations.
    - Start UDP broadcast from Swift Console
- Open Mission Planner
    - Connect to UAV
    - Mission Planner is injecting GPS
- Confirm that UAV Piksi Multi is receiving corrections (LINK LED is flashing red)
- Wait for onboard Piksi to achieve GPS lock (POS LED is solid yellow)
- Confirm that onboard Piksi has RTK lock (MODE LED is solid blue)
- Confirm that Mission Planner reports GPS2 fix of level 6, "RTK"

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Advanced Users

Use Dedicated Telemetry Radio for RTK Corrections

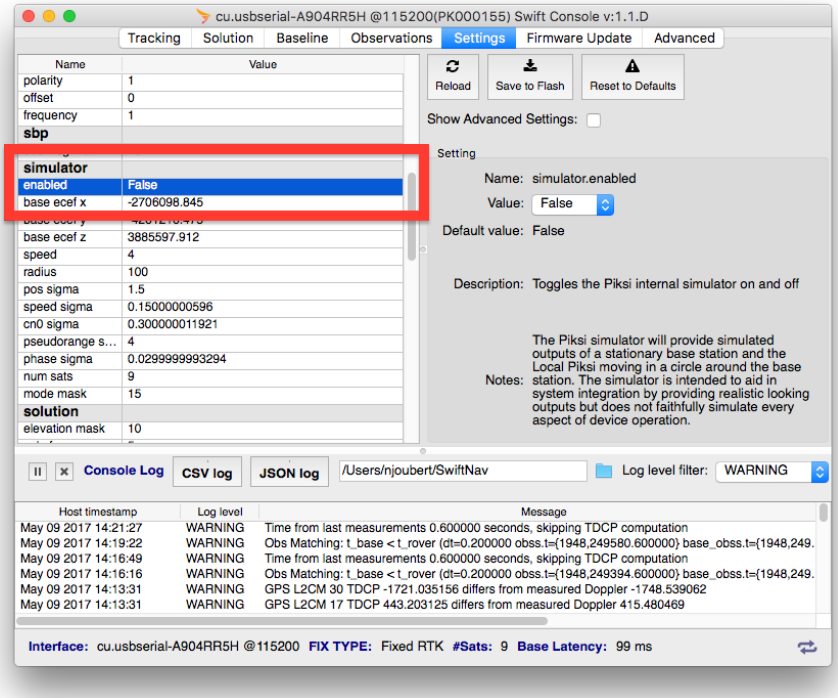
It is also possible to use a dedicated radio link for RTK Corrections, which we recommend if your telemetry radio is particularly flakey. In this case, replicate the setup from our Getting Started Guide. Use the FreeWave radios, connected to UART0, to exchange SBP observations. Connect your onboard Piksi Multi to Pixhawk via UART1. Use the same settings as provided in this guide.

Testing Integration with Piksi Multi's built-in simulator

Piksi Multi provides a built-in simulator. This simulator produces realistic and reproducible solution and observation data from Piksi, without requiring an antenna. The simulator is a powerful tool: you can test your end-to-end system without having an antenna with a skyview connected to Piksi, and you can confirm that your system is correctly parsing the Swift Binary Protocol.

The Simulator is enabled through the settings dialog. See the figure below. Once this setting is flipped to True, navigate through the tracking, solutions, baseline and observations tab. You will see simulated data populating the console.

Note: In the settings tab you can "Save to Flash", and Piksi Multi will boot into simulation mode until you change the setting back to False..



©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Broadcast SBP over UDP from the command line rather than Swift Console

If you don't want to use the Swift Console to broadcast observations, there exists an equivalent command line tool. The UDP_Bridge tool is part of the piksi_tools github repository (https://github.com/swift-nav/piksi_tools), and broadcasts SBP over UDP. The specific file is `piksi_tools/ardupilot/udp_bridge.py`, available here on Github.

Use MAVProxy as a command-line ground station & RTK correction router

MAVProxy is a full-featured, modular ground station and MAVLink router that runs on Linux, macOS, and Windows. See the MAVProxy documentation for installation instructions and more details. In the context of this document, we use it for two purposes:

1. As primary ground station, monitoring flight, sending commands to UAV
2. As mavlink router, combining multiple telemetry streams, injecting GPS observations, and proxying mavlink to a front-end ground station.

We internally use MAVProxy to inject RTK corrections into our telemetry stream in a headless manner rather than relying on Mission Planner.

Injecting RTK Observations using DGPS module

- Use Piksi Console to broadcast SBP Observations over UDP
- Start Mavproxy, and run the following:

```
$ mavproxy.py --master=/dev/tty.usbmodem1 --mav10
STABILIZE> module load DGPS
DGPS: Listening for RTCM packets on UDP://127.0.0.1:13320
Loaded module DGPS
```

- MAVProxy should now be routing GPS Observations from the local Swift Console (broadcasting UDP on post 13320), through the vehicle MAVLink connection, to the onboard Piksi itself.

Use MAVProxy to forward mavlink from telemetry radio to local tcp port, and to a ground station of your choice

MAVProxy can forward mavlink packets from your telemetry radio to a remote device (serial, USB or network address/port). This functionality is useful if using multiple ground stations, using a ground station that doesn't support GPS Inject, or relaying the stream through an intermediate node.

```
$ mavproxy.py --master=/dev/tty.usbmodem1 --mav10 --out=udp:127.0.0.1:14550
STABILIZE> module load DGPS
```

Now connect your ground station to `127.0.0.1:14550` over **UDP.**

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Extracting SBP Log from Pixhawk Dataflash Log

The ArduPilot driver automatically logs all SBP communication to the internal Dataflash logs, stored on Pixhawk's SD card. We are making our ***experimental, internal tools*** for extracting SBP data from these logs available to expert users.

We are currently working with ArduPilot to get this functionality ready. Coming soon!

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Logging hardware events (such as camera shutters) using EVENT_IN

Piksi Multi supports Event markers, which can accurately synchronize such events as a camera shutter with GNSS time. Here is how these event markers are exposed to users. Piksi Multi logs the occurrence of a digital pulse on its EVENT_IN pin. Whenever a transition from 0 to 3.3V occurs on this pin, Piksi Multi emits a MSG_EXT_EVENT. This message is timestamped with 10 nanosecond GPS time accuracy, allowing you to log events with extreme time precision. You can wire this pin to such things as a camera shutter to log camera events.

The Ardupilot driver can log **all** SBP messages to Pixhawk's internal dataflash log system. Ensure that ArduCopter's GPS_SBP_LOGMASK is set to -1. All hardware events can now be recovered by extracting SBP from Pixhawk's dataflash logs. Additionally, Piksi can log all SBP messages to a USB thumbdrive or SD card.

System Integrators can use this functionality to potentially increase the accuracy of mapping, by time-stamping camera events with high time and position accuracy.

For details on the EVENT_IN pin, exposed as pin 9 on the 20-pin header, see the Piksi Multi Hardware Specification.

For details on MSG_EXT_EVENT, see the SBPv2 Protocol Specification.

Caution	*Piksi Multi supports a 3.3V signal input on EVENT_IN. Do not exceed this voltage, as permanent damage can occur to your Piksi Multi, and such damage is not covered under the warranty.*

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Use a NTRIP client and MAVProxy to connect to a CORS reference station

It is possible to connect your UAV Piksi Multi to a CORS reference station rather than configure your own base station. **This functionality is currently EXPERIMENTAL and UNSUPPORTED.** If you are not comfortable working with haskell and python, installing experimental code from githuv repositories, and resolving dependency issues, we suggest running your own base station.

OK, now that we have the disclaimer out of the way! We demonstrate how to route RTCMv3 data, received over the NTRIP protocol from a CORS station, to our ground station, converting RTCMv3 to SBP along the way. We assume you are using a *NIX-based system such as Linux or macOS, or you are comfortable using *NIX-style command line tools on Windows.

Data pipeline overview

- Use `curl` to open an RTCMv3 stream from a CORS station
- Use our `rtcm32sbp` script to convert RTCMv3 to SBP
- Use `netcat` to broadcast converted SBP over UDP
- Use `MAVProxy` or Mission Planner to inject SBP into telemetry stream

Preparation: Installing Haskell Stack and rtcm32sbp

1. Ensure you have haskell stack installed.
    a. Follow the instructions at https://docs.haskellstack.org/en/stable/README/
        i. This step might direct you to **update your PATH appropriately**
        ii. This step might direct you to **install xcode command line tools**

2. Download the gnss-converters github repository from https://github.com/swift-nav/gnss-converters

```
$ git clone https://github.com/swift-nav/gnss-converters.git
```

3. Install gnss-converters. *This step can take up to dozens of minutes.*

```
$ cd gnss-converters/haskell
$ stack install --resolver lts-6 --install-ghc
```

4. Confirm existence of rtcm32sbp binary

```
$ which rtcm32sbp
/Users/njoubert/.local/bin/rtcm32sbp
```


©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

Prerequisite: NTRIP access to CORS station

We assume you have:

- internet access
- the URL to a local (<50km away) CORS station that produces **RTCMv3** (RTCMv2 is not supported)
- credentials to access this CORS station in the form of a username and password

Running Pipeline

In one terminal window, we will connect to a CORS station over NTRIP, convert the RTCMv3 stream to a SBP stream, and emit the resulting SBP stream over UDP. **Replace $USERNAME, $PASSWORD, and $URL with the corresponding values for your CORS station.**

```
$ curl -s -N -u $USERNAME:$PASSWORD -H "Ntrip-Version: Ntrip/2.0" -A "NTRIP
curl/7.43.0" $URL | rtcm32sbp | nc -u 127.0.0.1 13320
```

You can now point MAVProxy or Mission Planner to your local UDP port 13320, and inject the SBP stream from this port into your mavlink telemetry stream. For example:

```
$ mavproxy.py --master=/dev/tty.usbmodem1
STABILIZE> module load DGPS
DGPS: Listening for RTCM packets on UDP://127.0.0.1:13320
Loaded module DGPS
```

Confirm Pipeline is running: On your UAV-mounted Piksi Multi, you should see the LINK LED flashing red as corrections arrive. Keep this pipeline running during flight!

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1

System Integrators

Swift Navigation System Integration Guide

Read here

Routing RTK corrections using the MAVLink RTCM_DATA message

This driver, and the majority of RTK GPS drivers in ArduPilot, receives corrections from a base station GPS using MAVLink's RTCM_DATA message.

See the ArduPilot SBP2 driver source code here:
https://github.com/ArduPilot/ardupilot/blob/master/libraries/AP_GPS/AP_GPS_SBP2.h

Specification of the RTCM_DATA message is available here:
http://mavlink.org/messages/common#GPS_RTCM_DATA

Payload Note: The RTCM_DATA message is a slight misnomer - it does not have to carry RTCM data. In our case, we use this message to carry raw SBP data. RTCM_DATA overcomes a specific limitation in mavlink: that messages can only be 255 bytes long. RTCM_DATA supports multi-part messages, carrying up to 720 bytes of payload to the UAV.

The easiest way to route an SBP data stream through your system is using UDP in your proprietary system, then using mavproxy to chunk SBP packets into RTCM_DATA messages. If you want to roll your own system to inject SBP messages into a mavlink stream, a good template is MAVProxy's DGPS module, source code available in the MAVPRoxy github repository:

MAVProxy source code:
https://github.com/ArduPilot/MAVProxy

module_DGPS source code, responsible for converting a UDP stream to a RTCM_DATA stream:
https://github.com/ArduPilot/MAVProxy/blob/master/MAVProxy/modules/mavproxy_DGPS.py

©2017-2018 Swift Navigation, Inc. All rights reserved | Version 2.1