Solex/Desktop Manual

This is the manual for Solex/Desktop. At some point I'll edit this text so it sucks less than it does right now. Currently I'm just knocking the stupid thing out and I have a lot of crap to do. So get off my ass.

Table of Contents

Getting Started

Solex is ready to use right away, but there are optional tools you can install to add functionality. The following gives details on those components.

GStreamer

GStreamer is a cross-platform streaming framework that can be used to consume video streams from any source and transcode them to the MPEG-TS format used by Solex.

To install on **Windows**, install the binaries for GStreamer from this link.

To install on Linux, you need the following packages:

- libgstreamer1.0-dev
- libgstreamer-plugins-bad1.0-dev
- libgstreamer-plugins-base1.0-dev
- libgstreamer-plugins-good1.0-dev
- libgstreamer-plugins-ugly1.0-dev

On Ubuntu, you can install all of these with this command:

apt-get install libgstreamer1.0-dev libgstreamer-plugins-bad1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-good1.0-dev libgstreamer-plugins-ugly1.0-dev

This will give you all possible plugins for transcoding from any supported format to MPEG-TS. Use the "External Stream" video type for your vehicle. Worst-case, you might have to craft your own gstreamer pipeline in order to get video, but the good news is that you can.

SITL

When creating missions, it's super useful to be able to "fly" them without using a real vehicle. To do this, you can install SITL on your computer, and use it from within Solex.

- **SITL on Windows** Follow the instructions here to get SITL running on your Windows machine.
- **SITL on Linux** Follow the instructions here to get SITL running on your Linux machine.

Control Mappings on Linux

If you want to use the control mappings in Solex with a game controller on Linux, it's necessary to create a udev rule so your user account has permissions to use the USB controller. Here are the steps:

1. Copy this text and paste it into /etc/udev/rules.d/50-steam-deck.rules:

   SUBSYSTEM=="input", GROUP="input", MODE="0666"
   SUBSYSTEM=="usb", ATTRS{idVendor}=="28de",
   ATTRS{idProduct}=="1205", MODE:="666", GROUP="plugdev"
   KERNEL=="hidraw*", ATTRS{idVendor}=="28de",
   ATTRS{idProduct}=="1205", MODE="0666", GROUP="plugdev"

   ...and save the file.

2. Run this command in a terminal, or reboot:

   sudo udevadm control --reload-rules

Start Page

The start page is where you first connect to your vehicle.

Connection types

If you have a device plugged into the USB port, you'll see a list of USB devices in the drop-down list next to the **Connect** button. The device will most likely have a more-descriptive name in the list than /dev/ttyS5 or something, and will most likely have a name identifying its manufacturer. Select the desired item from the list, and click the **Connect** button, and the app will connect to the vehicle. If you plug something in and want to see it in the list, click the **Refresh** button.

If you want to connect on UDP, just select **UDP** from the list. Any vehicle visible on the network that is sending data on the UDP port specified in Settings (14550 by default) will appear as a connected vehicle.

Flight Page

The Flight Page is where you'll spend most of your time, flying your vehicle around.

This page will display at least a map view. If you've configured a video input from your vehicle and Solex can see data from it, you'll see a selector in the bottom right area of the screen where you can switch to a video view.

The **Telemetry** area at the top shows various fields of data being reported by the vehicle. The **Mode** selector lets you select the vehicle mode. If you do this and the mode change is successful, you'll see the field immediately to the right of the selector change to the name of the current mode.

On the right is a field that tells whether the vehicle is armed or disarmed. You can click on this and arm or disarm the vehicle.

Panel buttons

- **Map Type selector** - selects the type of map to display.
- **Open Mission** - Open a mission file to view.
- **Download Mission** - Download the current mission from the vehicle.
- **Send Mission** - Send the mission loaded from "Open Mission" to the vehicle.
- **Geofence** - Options for creating, loading, saving, or clearing a geofence.
- **ROI** - Click this button to drop a ROI point on the map.
- **Clear ROI** - Clear a previously-set ROI.

ROI Point

When you click the "ROI" button, you can click on the map to drop a ROI point. The vehicle will point at the ROI location if it can (if it's a multirotor or copter). You can drag the ROI marker around to change the ROI point, or right-click the ROI marker to delete and clear the ROI on the vehicle.

Missions Page

The Missions page is where you create and edit missions.

To create a new mission, click the "add" icon, select a vehicle type for the mission, and select the type of item you want to add. Then click the map. If the item is a "spatial" item (something with a location which appears on the map), you'll see a marker appear for it. If not, the item will appear in the item list, which will open automatically. Examples of spatial items are Waypoints, Splines, Land items, etc. Examples of non-spatial items are Change Speed commands, Take Picture commands, etc.

To change the position of a dropped item, just click and drag it.

To change the order in the mission where an item appears, open the item list (click the "Click for details" view) and drag the item up or down in the list, or click the up or down arrows.

To edit a specific item, you can click it on the map or in the item list, and the details panel will appear on the right side of the screen. The attributes of the item will be displayed in the panel, and you can edit them to suit what you're trying to do.

To **delete** an item from the mission, you can either click the trash-can icon and click the item on the map, or click the "x" button in the item list.

Undo undoes a previous operation. You can undo a maximum of 10 operations.

Import imports mission files from external sources. Click "Import" from the top bar and select the type of mission you want to import. You can import recorded flight logs, and QGC mission files. You can also make your own importer by putting a FileImporter subclass in the mission_import directory under the Solex base directory.

Params Page

The Params page lets you view and edit vehicle parameters. The parameters for the vehicle are loaded when the page loads.

To edit a parameter, double-click on it in the table, and specify a new value for it. Do this for as many rows as you need to. When you change a parameter, the **Send Params** button appears. Click this button to send the parameter changes to the vehicle.

If the parameter you've selected has additional data associated with it to let you pick from a set of acceptable values, you'll see those at the bottom of the page.

App Settings

The Settings page is where general settings for the app are shown.

The **Display Type** setting is for showing measurements in imperial or metric units. Where possible, edit fields also honor this setting (for example a waypoint altitude in the mission editor can be either feet or meters depending on this setting).

Default altitude is for setting the default altitude of new waypoints in the mission editor.

Offline Maps is useful for caching map tiles locally so you can use the map when you're in a location without internet access. Actually, "useful" is an understatement. "Necessary" is more like it. Just turn this on.

Video Source is how you set Solex to make use of a video stream put out by your vehicle. Currently supported modes are **None** (pretty obvious), **TCP** and **UDP**. Find the IP and port your vehicle is emitting UDP packets on, and ensure they're encoded as MPEG-TS, and they will display in the Video part of the Flight page. (Also, you won't *have* a video panel if this is not set.)

If you have an active video source, you'll see the video stream in an inset panel on the flight page. Clicking that panel will make it go full screen. Clicking the "map" icon on the green mode selector in the bottom right will switch back to the map view with the video inset. You can also press "M" and "V" on the keyboard to toggle between the modes.

When deciding between TCP and UDP for video, the first concern is which mode your vehicle supports. Either of them can be implemented on the vehicle. The main difference comes down to the "visual effects" you'll see when the network connection has any sort of problem. Because TCP ensures all video packets arrive in order at the expense of some overhead in the connection, the effect of network issues on TCP will be increased latency, followed by a possible sequence of very fast motion, with the feed eventually catching up to about 80ms behind real time. UDP will have better latency (since packets either arrive on time or they don't arrive at all), at the expense of crappy picture quality.

Vehicle Settings

Any time you connect a vehicle to Solex, it creates a "UserVehicle" object and stores it for later use. A UserVehicle contains the various settings a vehicle might make use of such as name, battery chemistry, video source, camera definition, camera interface, and any control mapping set you want to use with it. The **Vehicle Settings** page is where you can see these settings and change them.

- **Name:** Pretty darn obvious what this is for. Name your vehicle so it's easy to pick out of a list when you have multiple vehicles connected.
- **Battery type:** This has an influence on how remaining flight time is calculated in the flight screen.
- **Solex CC IP/Port:** Ignore this for now, not many people are using Solex CC.
- **Video Source**: This is a big one. When you connect a vehicle, you need to know its IP address and the port it sends video data on. Once you know this, you can put these values into the Video Source field once you select a type, and see video coming from your vehicle.
- **Camera interface**: This has a bearing on the controls you'll see when interacting with your camera. There are several built-in camera interface types, and you can also specify your own types (installed through plugins).
- **Camera Definition:** There is a large list of pre-defined camera definitions in Solex, with (of course) the ability to add custom ones. These specify camera focal length, sensor size, etc and influence overlap and sidelap on survey missions when calculating the path they'll take.
- **Button mapping set:** Any control mapping sets you've created in the "Buttons" page (described elsewhere in the help) appear in this list. Select the one you want to use. When you connect to the vehicle (or re-connect after setting this value), the UserVehicle will load the button mappings in the set you specify, and the vehicle will respond to the movements of sticks, button presses, and so on.

Vehicle Console

Solex has a vehicle console in the Flight screen. Click the little "TV" icon in the lower right (or press ctrl+t), and a console will pop up. From there, you can type help to get a list of all of the supported commands. You can start and stop missions, set and clear ROIs, change speed, view and set parameters, and other stuff. You can add your own commands via console extensions in a plugin.

Note for mavproxy users:

Although this console works lot like mavproxy is used in the SITL simulator, it doesn't contain anything from mavproxy and isn't based on it. Some of the commands (for example, param show or param set) have similar syntax to that of mavproxy (but not the same syntax).

Terrain Following

Terrain following can be accomplished in a few different ways. A vehicle can be fitted with a sensor that lets the autopilot correct altitude based on distance from the actual ground, for example.

For cases where you don't have such a sensor on your vehicle, you can use Solex's terrain following to retrieve elevations for the locations where your waypoints are stored, and apply corrections at the point where mission items are sent to the vehicle. This approach differs from the Android version of Solex, where corrections are applied directly to the mission items themselves. The reason for this difference is that this approach allows you to move waypoints around freely in a mission during editing, and apply elevations to them repeatedly.

Terrain Sets

In the Mission Editor, there's an icon at the bottom that looks like mountains in a folder. Click that, and you can make a terrain set. Click **New Set** and then draw a polygon on the map by tapping points here and there. Drag them around to encompass the area where you'll be flying. When done, press **Done** and answer "OK" to the prompt asking if you want to download elevations. Once you've done this, you'll be prompted to save the elevations to a file. Give it a name like "my_field" or something and save it.

To apply a Terrain set, open the flight screen and click "Terrain Set" on the menu and select the terrain set you created in the previous step. Press OK and it will be loaded. Open a mission in the area, and click the little icon along the bottom with the wavy lines entitled "Apply elevations to mission". If you have waypoints in the mission, you'll notice that afterward, their altitudes will be adjusted to account for the terrain elevation. Items such as Grids and Surveys don't show this change, since they wait until you actually upload a mission to the vehicle to apply their elevations.
Upload the mission to the vehicle, and fly it. You should see its altitude change to match the contour of the ground, with 20-meter resolution.

A way to double-check the altitudes of waypoints is to do the following:

1. Apply elevations to your waypoints.
2. Send the mission to the vehicle.
3. Download the mission from the vehicle into the mission editor.
4. Click any waypoint to see its altitude in the detail panel. It will show the corrected altitude.

Caution

The way elevations are applied is as follows: The first item in a mission that has an altitude is used as the base altitude to work from. The elevation at that point is used as the baseline elevation. Subsequent item altitudes are calculated from these points. **Bear in mind** that if you park your vehicle at the bottom of a 50-foot hill and set the first waypoint of the mission at the top of that hill with an altitude of 20 feet (for example), no correction will be applied to that waypoint. If you launch a mission from the bottom of the hill, the vehicle will head toward the first waypoint at the un-corrected altitude, and you will crash. You must set the first waypoint at a location as close as possible to the same elevation as the vehicle launch point, so the corrections can be applied safely.

Multiple Vehicles

Solex supports the connection to multiple vehicles at a time, and provides various ways of coordinating their activities.

Unique sysids. They're important

In Solex, all vehicles connected at the same time must have unique sysids. Otherwise two different vehicles will appear to Solex as the same vehicle, and yield weird-looking results. If you see weird things like the vehicle mode rapidly toggling between two flight modes or showing up on the map in seemingly 2 locations at the same time, you're almost sure to have 2 (or more) vehicles connected to Solex with the same sysid.

If you have multiple vehicles with the same sysid that you intend to use together on the same UDP port, connect to them one at a time (that is, turn one of them on and leave the others turned off). Once connected, go into the Flight screen and open the terminal. Run the following command:

param set SYSID_THISMAV unique_id

...where unique_id is the sysid for that vehicle. Then, power off the vehicle and power on the next one.

UDP Connections

Given a single UDP port, Solex will receive messages from all vehicles on the network which are broadcasting on that port. Most vehicles send UDP on port 14550. Once you connect to UDP, any vehicle broadcasting on the specified port will appear as a vehicle in the vehicle selector.

Serial

You can connect to a vehicle on a serial port, using something like a SiK radio plugged into a USB port on your computer. Plug the device into a USB port and select the "Refresh" button on the Home screen. The drop-down will display any devices it recognizes as candidates for a serial connection. Select the appropriate port, and click Connect. The vehicle will connect and appear as an entry in the vehicle selector in the upper-right area of the home screen.

To connect another vehicle, select the appropriate serial port (or UDP) from the drop-down box next to "Connect Another", and click "Connect Another". The additional vehicle will appear as a vehicle in the vehicle selector.

Disconnecting a vehicle

Select the vehicle you want to disconnect from, and click the little "x" button to the right of the selector. The vehicle will be disconnected. For UDP vehicles, this means messages from the vehicle will be ignored. Restarting Solex and connecting to UDP will restore the connection to any previously-disconnected vehicles.

Focused vehicle

To keep the user interface straightforward, Solex's UI treats the currently-selected vehicle as "the vehicle". That is, the telemetry display, command line, various widgets, mission upload/download commands, etc all act on the currently-selected vehicle. However, selecting a new vehicle is easy, as described in the next section.

Selecting a vehicle

The obvious way to select a vehicle is from the drop-down selector in the upper-right area of the Solex window. In the map view, you can also click on a vehicle's icon to select it. From the terminal, you can run vehicle use id to select a vehicle. Type vehicle show to see a list of visible vehicles and their ids and names.

Multi-vehicle interaction

This is evolving, but the most current documentation for this can be found at the Location Sources page on Github.

Video Sources

There's a large variety of video formats that vehicle cameras can provide. To accommodate this, Solex includes several types of video sources:

- TCP (TCP packets from a specified IP/port)
- UDP (UDP packets from a specified IP/port)
- External (launch external command and display video in a separate view)
- External stream (launch external command and capture output to display on internal view)
- Any custom video sources you define as extensions

Currently, Solex's internal video decoder handles only MPEG-TS (although that will change at some point soon). Thus TCP and UDP video sources above have to provide MPEG-TS video frames, or video won't work. The "External stream" type gets around this by capturing output from an external program, typically in the form of a gstreamer pipeline that gets data from the source (e.g. UDP) and transcodes to MPEG-TS. If you leave the command blank, Solex provides a sensible default that works.

Location Sources

Location Sources are entities within Solex that can emit location data. A connected vehicle, if it has a GPS on it, is an obvious example. It has a location which changes as it moves. This stream of locations can be observed by location listeners.

Location Listeners are enties within Solex that can subscribe to a location stream from a location source. A connected vehicle can do this. This is useful if, for example, you want vehicle_1 to be aware of vehicle_2's location. The question is, how does a vehicle know what to *do* with a location stream?

Commands, that's how. There are a few built into Solex, such as roi and follow. Using these is a matter of:

1. Identifying the location source to use
2. Issuing a command to use it

To get a list of location sources, type location sources in the terminal in the flight screen.

Following a location source You can follow a location source by running follow (source id) in the terminal. Suppose you have 2 vehicles (vehicle_1 and vehicle_2) connected, and vehicle_1 is flying a mission. You can run vehicle use vehicle_2 to focus vehicle_2 and then run follow vehicle_1 behind 10 with vehicle_2 in Guided mode, and it will follow the other vehicle behind at a distance of 10 meters. follow vehicle_1 left 10 will fly to the left of it. Obviously you need to take care to avoid having vehicle_2 crashing into vehicle_1. The follow vehicle_1 path 10 command will tell vehicle_2 to follow the same *path* as vehicle_1 at a distance of 10m. If vehicle_1 turns to head in a different direction, vehicle_2 will turn at the same location, lessening the chance of the two vehicles meeting in the air.

You can use the smooth option to make the copter fly smoothly when following something moving at a low speed, such as a person walking. In this case, the copter will match the target's speed. Something like follow source_id behind 10 smooth look will follow behind a target at a distance of 10 meters, point its camera directly at it, and match its speed.

Run follow stop to stop following. The following vehicle will hover in place until you switch to a different mode and fly elsewhere.

You can use the roi follow (source id) command to have a vehicle "look at" a location source. Use roi follow stop to stop following.

Custom Location Sources With all of that being said, you might ask "what's the point of all of this?" That's a pretty good question, especially if you don't have multiple vehicles in the air. Well... You can also make your own location sources, so that's something. Suppose you have a GPS device you can plug into your computer's USB port. You can write a javascript module that reads location data from it and emits location updates to `LocationSource` in Solex, give it a name, and have your vehicle follow it around by issuing follow (whatever you call your source) behind 10. Or suppose for some reason you end up with a huge list of GPS locations in a CSV file or something. You could write something to read that file and emit locations every few seconds and have your vehicle follow that around. It's one of those things that if you don't need it, there is no point. But if you do, here it is.

Monitoring a location source If you're interested in seeing what locations are coming from a given location source, you can see them on the map in the flight page. In the terminal, run these commands (for example)

1. location start source_id
2. location show source_id

A marker will appear on the map whenever the source_id source emits a location. Use location hide source_id to hide the location source.

Control Mappings

A **Button Mapping** is a connection between a button, switch or slider on a game controller and an action that is triggered or updated when that control is moved. This is useful if you want to use a game controller to control either your entire vehicle, or a single feature on it (pan/zoom/yaw a camera on a gimbal, for instance).

The **Buttons** page in Solex is where you create and edit Button mapping sets, which are sets of button mappings saved in files. You can create any number of button mapping sets, and then assign them to a vehicle you connect to. Once you've done this, the various buttons on the controller can be used to trigger these actions.

There's a fairly large assortment of built-in actions. Here's a partial list:

- RTL
- Arm
- Disarm
- Toggle Arm
- Clear ROI
- Mavlink command_long
- Set Relay (button)
- Set Relay (latching/switch)
- Set Servo (button)
- Set Servo (latching/switch)
- Takeoff

You can also make custom actions as part of a plugin.

Network Presence

When making a mission or doing other detailed work, it's nice to have a desktop computer to do it on, or a laptop. Something with a mouse and keyboard.

But a desktop computer with a mouse and keyboard isn't exactly ideal for taking to the field to fly a mission. For that, you want something smaller, like a controller with a smaller screen. Which of course isn't great for creating missions or doing detailed work.

Ideally, you'd be able to make missions in Solex on the big computer and transfer them to the smaller one. Solex makes this easy. Just start Solex on both machines connected to the same local network in your office or whatever. Any missions or terrain sets you created on one machine can be seen by the other.

In the Mission Editor, you'll see a **Local Server** button in the top bar. Click that, and you'll get a panel showing Solex host machines running on the local network. Pick the one you want and then pick the type of data you're interested in seeing. Click one of the items in the list that appears, and you can download it from the other machine to the one you're using.

Note that if the host machine has custom mission-item types (which it only would if you had made or installed them on the host machine yourself), attempts to import the mission on the client machine will fail unless the client machine also has the same custom mission-item types installed.

Voice Alerts

Solex can be set to use a TTS voice on Linux, Mac and Windows. The underlying TTS engine is different depending on the platform. For Linux and Mac, you can install the espeak package. For example, on Linux, run this command:

```
apt-get install espeak
```

At that point, you can enable Voice alerts in Solex and pick a voice you like. They sound "ok", and the various voices are all basically the same voice with different accents.

On Windows, the TTS speech engine should already be installed, but overall the system works differently. The speech is performed by external scripts that are installed to your Solex home directory on Windows the first time the app is launched. These are run whenever speech is needed in the app.