ACS - Installer

The Aerial Combat Swarms system includes many software subsystems. A series of bash scripts was created that are intended to automate some of the installation process. This document outlines what is installed and gives some example installations scenarios.

Installing the installer

To run the installer on Ubuntu 14.04, first ensure that the git package has been installed.

    sudo apt-get update
    sudo apt-get install git

It is also highly recommended that the system be brought up to date before running the installer:

    sudo apt-get update
    sudo apt-get upgrade

The remote repositories for all of ACS, including the acs-installer repository reside on server using a U.S. DoD SSL certificate. Either ensure that your system can recognize these certificates (not covered here), or, as a workaround, disable SSL cert validation in git:

    git config --global http.sslVerify "false"

Then clone the installer repository into the desired location on your system:

    git clone https://gitlab.nps.edu/acs/acs-installer.git

Running the installer

A prerequisite to running or installing the installer is an internet connection that can access gitlab.nps.edu

Change into the directory where the acs-installer repository has been cloned, e.g.

    cd acs-installer

To get a sense of what the installer provides type:

    ./acs-devbox-installer.bash

Which results in a usage print together with an overview of the command line options:

Usage: ./acs-devbox-exec-installer.bash [options]
Options:
-N     No apt-get update/upgrade called.
-s     Install SITL
-o     Install GCS
-U     Update existing repos
-R     Install SITK radio firmware and tools
-s     Install git repos from ssh instead of https
-c     Use repoConfig.txt file to set git url, fork, and branch
-t     Test installer (for devst, skips update to installer step)
-d DIRECTORY Installation directory (defaults to home folder)

Typical installation

    ./acs-devbox-installer.bash -SGR

will install all ACS software systems. There will be a warning for developers:

This installer will wipe out local changes to any pre-existing repos!!
Developers should commit or stash any local edits before proceeding.

Do you wish to run the installer? (y/N)

most users can safely enter "y", indicating that it is OK to proceed with the install.

When the installer completes, be sure to log out and log back in to ensure all environment changes completely take affect.

Post-install test run

First, initialize some SITL instances. At a command prompt type:

    init_n_sitls.bash 10

The instances are created in your $HOME/flying directory.

#####Single Plane Test

To test single-plane simulation, change to one of the sitl directories:

    cd $HOME/flying/sitl

and use the ardupilot sim_vehicle.sh utility to start a single instance:
sim.vehicle.sh -v ArduPlane -L McMillan --map --console
after the simulator starts up you should see a red plane on McMillan airfield. To get the plane to takeoff, type:

```
arm throttle
AUTO
```
at the prompt (this is the MAVProxy GCS prompt).

If you wish to see the mission the plane is flying type:

```
wp list
rally list
fence list
```
at the prompt.

Further MAVProxy documentation is not provided here, but a good starting place is typing `help` at the prompt, as well as:

```
```
To end the single-plane test type `exit` or `:C:` at the prompt.

```
#### Multi-Plane Test
```
To clean up screen real estate during multi-plane simulations, you may set the MAVProxy map and console windows to no longer appear at startup. Otherwise, you end up with many maps and consoles for each plane.

Edit the file `~/.mavinit.acr` in your editor of choice.

Near the bottom of the file find the `module load console` line and comment it out by placing a `#` in front of it. Do the same for the `module load map` line.

To start 3 instances of the single-plane SITL, enter the following at the terminal:

```
multi-sitl-start.sh ~ 3
```
This should pop up several xterm windows. After they've started, select the original terminal and type:

```
acs_health_mon.py -d sitl_bridge_1
```
This will pop up two windows of the ACS Health monitor application. You should see 3 aircraft, though in the map they are all at exactly the same position. Note you can drag the map around with the mouse and zoom in and out with the mouse wheel.

To get plane(s) to takeoff, select them in the spreadsheet-like table in the ACS Health Monitor, the click the "Arm" button, then click the "Mode" select box and select "AUTO."

When you are finished testing you can clean up by closing both ACS Health Monitor windows and then at the original terminal type:

```
multi-sitl-cleanup.sh ~B
```
Further details on single and multi-plane simulation operations are covered in a separate document.

Detailed descriptions of each installation option:

**-N**
This option suppresses a default behavior in the installer. Typically the installer will run:

```
apt-get update
apt-get upgrade
```
to ensure the underlying Ubuntu system is up to date before attempting to install dependencies. If the -N option is used, the installer does not perform this preparation.

It is recommended that most users do not use this option.

**-S**
This option specifies that we wish to install all software required to run a Software In The Loop simulator for swarming planes.

**-G**
This is a subset of the -S option above. Installs all Ground Control Station (GCS) software.

**-U**
Update existing repos. Use this option if you have performed a previous install and wish to update your installing using updates that have appeared at the ACS gitlab remote server.

Do not use this option for first time installs.

**-R**
Install the SIK radio firmware and the atcommander tool.
The installer defaults to using SSL for downloading repositories. Use this option to use a private ssh key instead. This option is largely for developers to remove the onerous need for multiple authentication tokens when pushing up code updates.

Most users do not need to use this option.

-c
Use a config file, repoConfig.txt (located at the root of the acs-installer repository) to specify where to find the git repositories, rather than using the default locations.

Most users do not need to use this option.

-t
This option is for developers testing the acs-installer itself. When the installer runs, the first step is to check the acs-installer git repository for updates -- if so the installer updates itself. Use this option to bypass that step and suppress updating the installer before running the installer.

Most users do not need to use this option.

-d
This option allows a user to specify the directory into which they would like to install the ACS systems. The default location is $HOME/ACS and after the install is complete the environment variable $ACS_ROOT maintains this value.

Map imagery
The installer installs map tiles with satellite imagery for use with our sample SITL (Software In The Loop) simulator scenarios. To prevent an excessive amount of download time, the acs-installer only includes imagery of the area around McMillan Airfield in Camp Roberts, CA. If you need to download imagery for your flying site use the mp_tile.py utility, which should have been installed when MAVProxy was installed by acs-installer.

```
mp_tile.py --help
```
Will give a listing of the options available to the tool. The acs-installer will install tiles as follows:

```
mp_tile.py --service OviHybrid --lat 35.736 --lon -120.791 --width 3000 --max-zoom 17
```
which installs tiles around a roughly 3km square area at the given lat/lon. The maximum zoom is 20, but the installer only goes to 17 to avoid a prohibitively long install time to download map tiles.

To install map tiles for your location you can use this tool at the approprite lat/lon. Alternatively, you can open up any ACS Ground Control Station (GCS, examples are MAVProxy, SwarmCommander and ACS Health Monitor) with an active Internet connection and browse on the map until you are at your location. The mp_tile.py tool is useful when you want to ensure you have pre-downloaded all tiles before traveling to a flying site lacking an Internet connection.

Services available in addition to "OviHybrid" at the time of this writing included:

- GoogleSat
- GoogleMap
- GoogleTer
- MicrosoftBingMap
- MicrosoftHyb
- MicrosoftSat
- MicrosoftMap
- MicrosoftTer
- OviSat
- OpenStreetMap
- OSMARender
- OpenAerialMap
- OpenCycleMap

Compatibility notes
Much of the ACS software has been targeted to a Ubuntu 14.04 environment. The installer is intended to work in that environment.

Manual install and porting options
Installation to other Linux environments other than Ubuntu 14.04 would likely be fairly straightforward, but require some porting.

The ardupilot SITL (including ArduPlane, JSBSim, MAVLink, and MAVProxy) has been ported to Windows:


so it should be possible to port a single-plane SITL to Windows. The ROS packages (where the swarming software lives) are untested in a Windows environment, but since they are implemented in Python it should be possible to port a multi-plane SITL for swarming to Windows, though that has not been done at the time of this writing.
The Swarm Commander, Health Monitor, and FTI ground systems are written in Qt and Python so they should be portable to Windows but that has not been done at the time of this writing.

A bash based install system on Windows would only be possible using something such as Cygwin. This would be a heavy dependency, so it may be worthwhile to have a different install system (as an alternative or supplement to acs-installer) for Windows in the future.

Core ACS software systems

Following is a description of each ACS software system installed by acs-installer. Note that all branch configurations described should be automatically done by the installer.

SiK

The ACS fork of the SiK repository. Firmware and tools for the 3DR radio used for single-plane (not swarm) communications with an aircraft.

ACS Repo:
https://gitlab.nps.edu/acs/SiK

Use the mc_ate_unstick branch in the ACS Repo. The master branch in the ACS Repo is an unmodified older version from the original repo.

Original Repo:
https://github.com/Dronecode/SiK

acs_lib

Python libraries shared between ACS systems.

https://gitlab.nps.edu/acs/acs_lib

acs-env

Scripts and data used by ACS systems.

https://gitlab.nps.edu/acs/acs-env

ardupilot

The ACS fork of the ardupilot repository. The ArduPlane system lives here.

ACS Repo:
https://gitlab.nps.edu/acs/ardupilot

Use the dev branch of the ACS Repo. The master branch in the ACS repo is an unmodified older version from the original repo.

Original Repo:
https://github.com/diydrones/ardupilot

mavlink

The ACS fork of the MAVLink repository. This is a communication protocol used by the single-plane ground control system and ArduPlane.

ACS Repo:
https://gitlab.nps.edu/mavlink/acs-env

Use the dev branch of the ACS Repo. The master branch in the ACS repo is an unmodified older version from the original repo.

Original Repo:
https://github.com/mavlink/mavlink

mavproxy

The ACS fork of the MAVProxy repository. A single-plane (not multi-plane or swarm) ground control station.

ACS Repo:
https://github.com/mavlink/mavproxy

Use the dev branch of the ACS Repo. The master branch in the ACS repo is an unmodified older version from the original repo.
swarmcommander
Swarm Commander ground control station. Used to perform swarm operations with groups of aircraft.

https://gitlab.nps.edu/acs/swarmcommander

acs_dashboards
ACS GUI dashboards. The FTI (flight tech interface) used prior to and at takeoff resides here. The ACS Health Monitor used to monitor groups of aircraft during flight also resides here.

https://gitlab.nps.edu/acs/acs_dashboards

autonomy-payload
Software that resides aboard the companion computer on operational aircraft. It is provided here because it is needed for multi-plane STTL operations.

https://gitlab.nps.edu/acs/autonomy-payload

autopilot_bridge
Software that provides communications between the autopilot (ArduPlane) and the companion computer (autonomy-payload).

https://gitlab.nps.edu/acs/autopilot_bridge

arbiter
Application that is used when two swarm teams are competing. Provides virtual sensing and hit detection.

https://gitlab.nps.edu/acs/arbiter.git

jsbsim
Flight dynamics engine behind the ArduPlane simulator. There is no ACS fork of this, though we use an ArduPlane developer's fork:

https://github.com/tridge/jsbsim

Dependency notes
The installer itself uses these Ubuntu packages:

- bash
- git
- Ubuntu/Debian apt-get tool

ArduPlane / PX4 builds require these Ubuntu packages:

- g++
- make
- autoconf
- libtool
- Ubuntu build-essential metapackage
- ARM compiler
- python-argparse
- openmocd
- flex
- bison
- libxml2-dev
- texinfo
- libftdi-dev
- zlib-dev
- zip
- gromofs
- libc6:i386 libgcc1:i386 gec-4.6-base:i386 libstdc++:++5:i386 libstdc++:++6:i386

ArduPlane / PX4 builds recommend this package:

- cache
ArduPlane Tools require this package:

- xterm

Python ground control systems (MAVProxy, SwarmCommander, Health Monitor, FTI) use:

- python-setuptools
- python-pip
- python-serial
- python-openscm
- python-espeak
- python3-espeak
- python3-serial
- python3-urllib3
- python3-ntplib
- python3-setuptools
- python3-pip
- python3-pyqt5
- python3-pil
- python3-pyside

Custom ACS tools and scripts require:

- python-pexpect
- python3-pexpect
- screen
- expect-dev

System modifications

These functions are automatically performed by the installer, but documented here for completeness.

The Ubuntu modemmanager package must be removed from the system and all ModernManager processes killed.

The current user is added to the dialout system group.

GNU tools for ARM embedded processors (for building on Pixhawk) are downloaded and installed. As of this writing the URL for obtaining these is:


http://wiki.ros.org/Indigo/Installation

Packages installed are:

- ros-indigo-desktop-full
- ros-indigo-sensor-maga
- ros-indigo-robot-pose-ckf

The installed ROS files reside in the /opt/ros/indigo directory and the /opt/ros/indigo/setup.bash file is sourced in the $HOME/.bashrc file.

An ACS directory is created where all git repositories will be stored. The default location is $HOME/ACS but the default can be changed with the -d option when the installer is executed.

An environment variable, $ACS_ROOT, is created that stores the location where the ACS directory is located.

Many scripts and Python libraries are placed in $HOME/.local/bin and $HOME/.local/lib and path environment variables are updated as appropriate in the $HOME/.bashrc file.

A default .mavinit.scr file is placed in the $HOME folder. MAVProxy uses this file for startup options.

A $HOME/flying directory is created where the SITL data files live.

Modifications are made to the current user’s $HOME/.bashrc to update the $PATH and other environment variables to facilitate command-line usage of ACS, ROS, and ardupilot tools. Changes are at the bottom of the file.

An initial set of map satellite imagery tiles is downloaded via the mp_tile.py tool and placed in the $HOME/.tilecache directory.

A log file (acs_installer_log.txt) of the install process is placed in the same folder the installer was run from using the Linux tee utility.