PX4 and Pixhawk Case Study

Jordan Taranto
CS456
Midterm Exam

The name/use of the application, website, or software (+2)

PX4 | Pixhawk

Created by Lorenz Meier in 2008 while working on his masters at ETH Zurich. He wanted to develop algorithms to be used in drones. At the time drone computing power and technology dd not exist. So he recruited 14 students to create the drone, flight controller unit (FCU), and flight controller software. After 9 months of work the team won the European Micro Air Vehicle competition in the indoor autonomy category in 2009, with a custom flight controller and flight controller software written from scratch. The teams name was Pixhawk and that is how it was born.

- Pixhawk makes the process of creating drones easier than ever by tying everything into one simple box
- Fully open source flight controller software PX4
- Flight controller for purchase Pixhawk

What you think needs to be changed about it (+2)

Installation:

The installation process is extremely challenging and is not user friendly in any way. Depending on what software you want to install also effects what operating system you can use. They are dependent on one another. For this example we will be using ROS and Ubuntu Linux

Requirements:

- Linux Ubuntu 20.04
- Robot Operating System Noetic (ROS)
- MavLink
- PX4
- Catkin
- Cmake

Steps:

1. Install on a new SSD or partition Linux Ubuntu 20.04
    1. Make sure the correct kernel is installed
    2. Make sure the correct GPU drivers are installed
2. Install CMake
3. Install ROS
    1. Build ROS
    2. Verify ROS runs
    3. Verify Catkin builds, cleans, and runs
4. Install PX4
    1. Before installing PX4 you must install PX4 toolchain so it will integrate with ROS
    2. Verify the Toolchain works
    3. Build PX4
        1. Make sure you build it for your specific flight controller
    4. Run PX4
5. Install MavLink
    1. Build MavLink
    2. Verify Mavlink
6. Install QGroundControl (QGC)
   This is a very intensive process making sure everything is the correct version otherwise it

will not work. (I run a business focused solely on installing PX4 and ROS for companies and academics, because it is a very challenging task)
This needs to be fixed, if it was easier to install much more people would be interested in PX4.

Reliability:

If their is an update of PX4, ROS, or Linux, it can break your current installation which leads to 'dependency hell' and you have to start from scratch. The reasons listed above is a domino effect for various other issues that can arise.

Who the users are that need these changes (+2)

Hobbyists:

Hobbyists who are interested in building their own drone do not have an easy way of installing this. It requires a high level of knowledge to be able to install these correctly and safely. This prevents a lot of hobbyists from being able to install and utilize this software.

Military:

In the United States when you want to install a software, driver, and/or package it must be vetted by the governments security team before installation on a government system. Since there are updates for the operating system, software, and/or drivers that can break this it must be vetted which takes a long time. It also opens up security risks, if you don't keep your software up to date that leaves bugs and vulnerabilities which can be taken advantage of by adversaries.

What problems the users need to solve (+2)

Installation:

- Installation of all the components that go into PX4

Learning curve:

- The learning curve to run the software let alone install it

What characteristics make up a good solution (+2)

Scalability:

- Able to scale a software on multiple systems

Open standards:

- Standards for software so it is consistent, PX4 being open source sometimes standards are not follow leading to things breaking

User-friendly:

- This ties into Scalability, if it is not user friendly that prevents the software to scale so it is important to have a user-friendly experience.

Potential interview questions you might ask of a user population (+2)

- **How do you currently integrate advanced drone technology into your operations or products?**
- **What challenges do you face with current drone technology platforms?**
- **What features are most important to you in a drone technology solution?**
- **How do you prioritize security and privacy in your drone operations?**

Either a verbal description or a drawing/picture of what your suggested interface would look like (+4)

Current interface(s):

Their are three parts to simulating the drone.

1. PX4 (required) - The flight controller which handles all of the sensors/ hardware aspects and the flight control laws and algorithms for the drone
2. QGround Control(required) - This is the ground control station which gives you a birds eye view of the drone(s) and where they are, relative to you
3. Gazebo (not required) - This allows you to visualize the drone in a virtual environment. It is not required but it gives us humans a better way to understand what the drone is doing versus reading numbers out of a terminal

1 - PX4

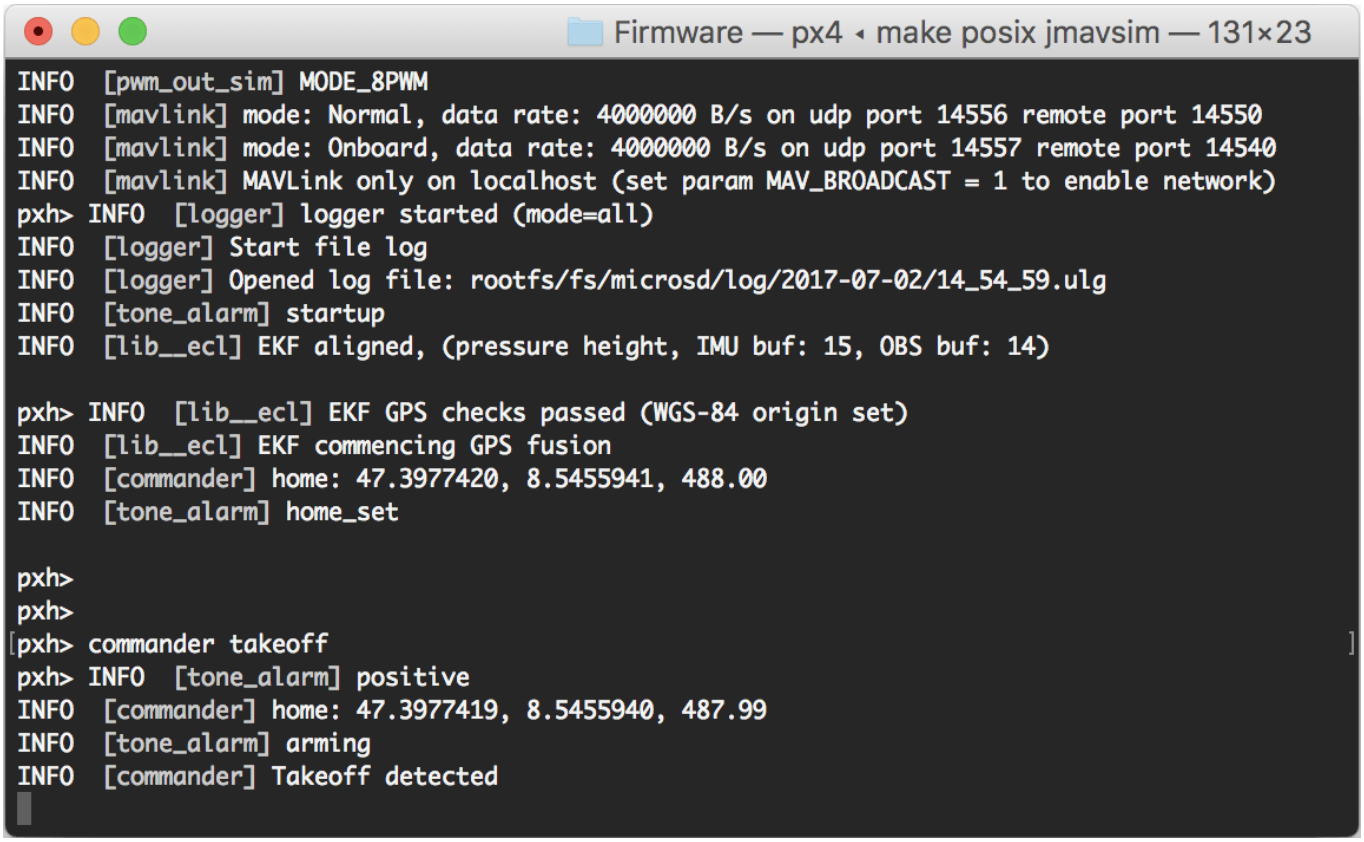
- The user can send commands to the drone through the Command Line Interface (CLI)

```
_____  __   __    ___
| ___ \ \ \ / /   /   |
| |_/ /  \ V /   / /| |
|  __/   /   \  / /_| |
| |     / /^\ \ \___  |
\_|     \/   \/     |_/

px4 starting.

INFO  [px4] Calling startup script: /bin/sh etc/init.d-posix/rcS 0
INFO  [param] selected parameter default file eeprom/parameters_10016
[param] Loaded: eeprom/parameters_10016
INFO  [dataman] Unknown restart, data manager file './dataman' size is
11798680 bytes
INFO  [simulator] Waiting for simulator to connect on TCP port 4560
Gazebo multi-robot simulator, version 9.0.0
Copyright (C) 2012 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazebosim.org
```

```
...
INFO  [ecl/EKF] 5188000: commencing GPS fusion
```

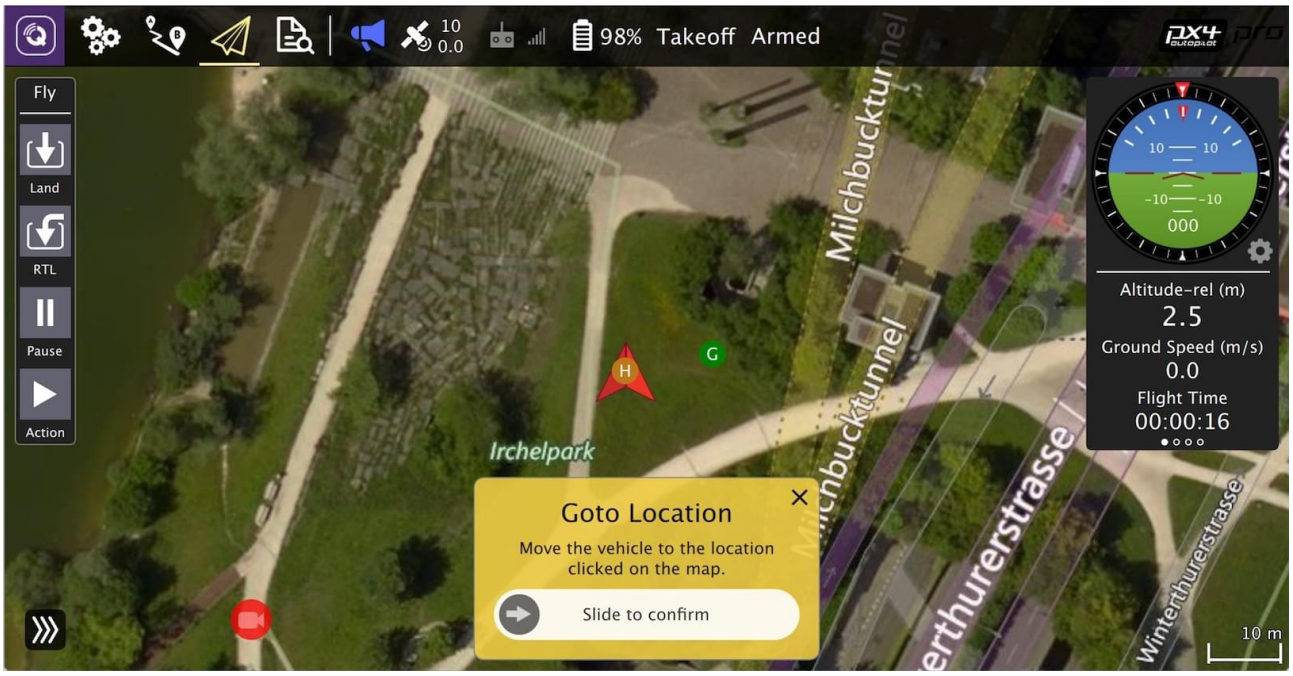


- To have the drone take flight the user can input the 'takeoff' command into the CLI

```
pxh> commander takeoff
```

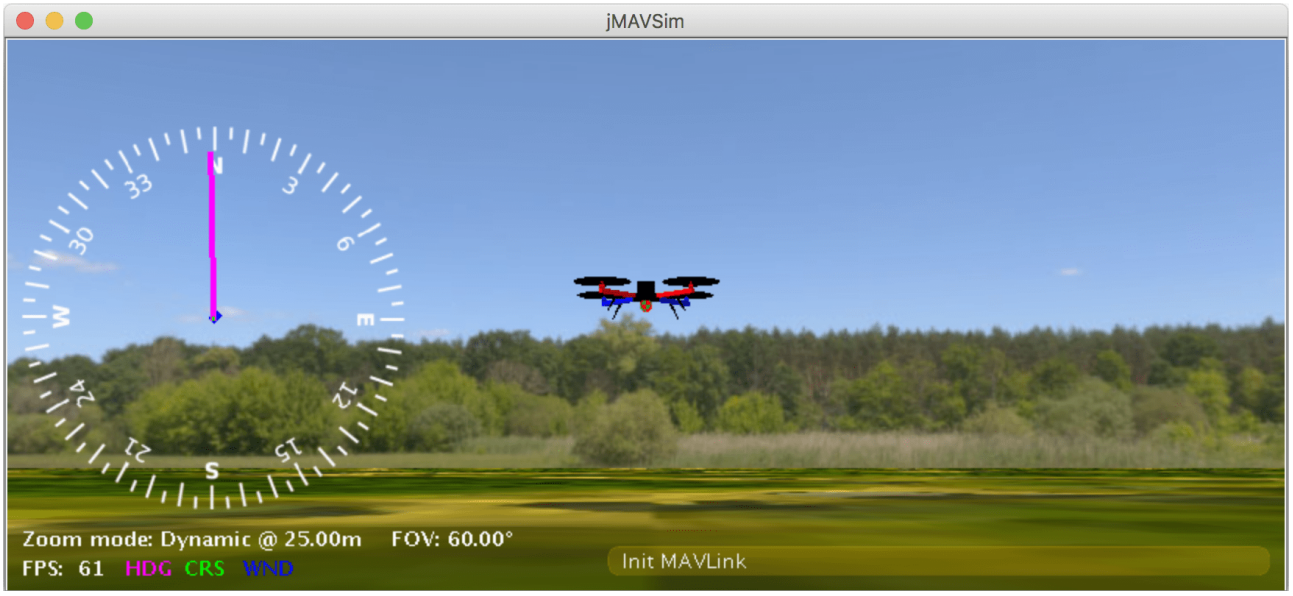
2 - QGC

- The user can see the current state and location of the drone through a friendly User Interface (UI)



3 - Gazebo simulation

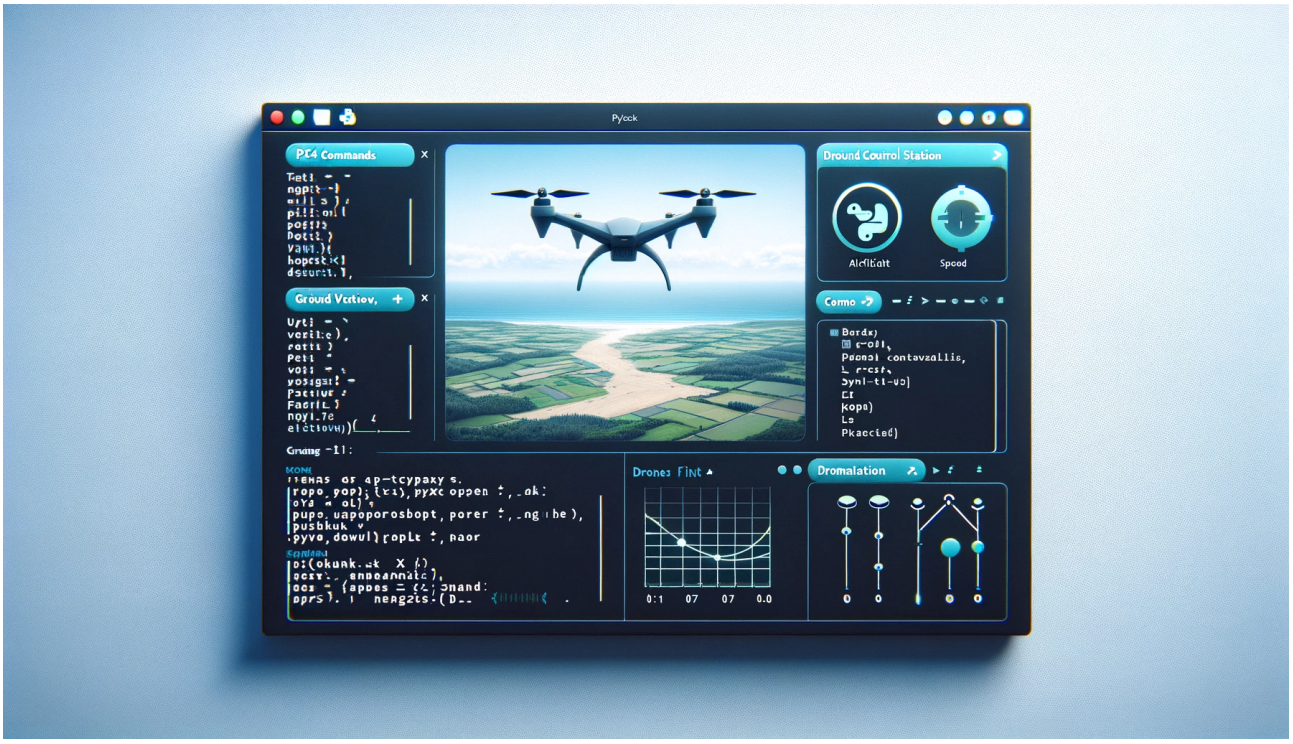
- The user can visualize the drone in real time through



Suggested interface:

(Used Fooocus an open source stable diffusion model to create this image)
A single window containing:

- Command terminal
- Ground control station
- Simulated environment



A telemetry plan (+2)

- Collecting data on system performance
- Mission success rates
- User interaction and what issues they face to inform continuous improvement
- Insure anonymity when using the software, especially true for military applications

The security/privacy plan (+2)

- Privacy-by-design approach ensuring user
- Robust encryption for data transmission and storage
    - ex. A drone crashes in an adversaries territory data cannot be retrieved
    - ex. Commands being sent to the drone in real time, the telemetry data cannot be seen by an adversary
- A system breach for system that are currently operational
- Security audits and updates that won't break the system or software to address emerging threats