IEEE VTS

Connecting the Mobile World

WEBINAR SERIES ON ADVANCED MOBILITY

Acknowledgement

The presenter wishes to acknowledge the IEEE Vehicular Technology Society for their sponsorship of the Webinar Series on Advanced Air Mobility.

IEEE VTS Connecting the Mobile World

AERPAW

https://aerpaw.org/

PAWR Project Office

AERPAW Vehicles: Hardware and Software Design

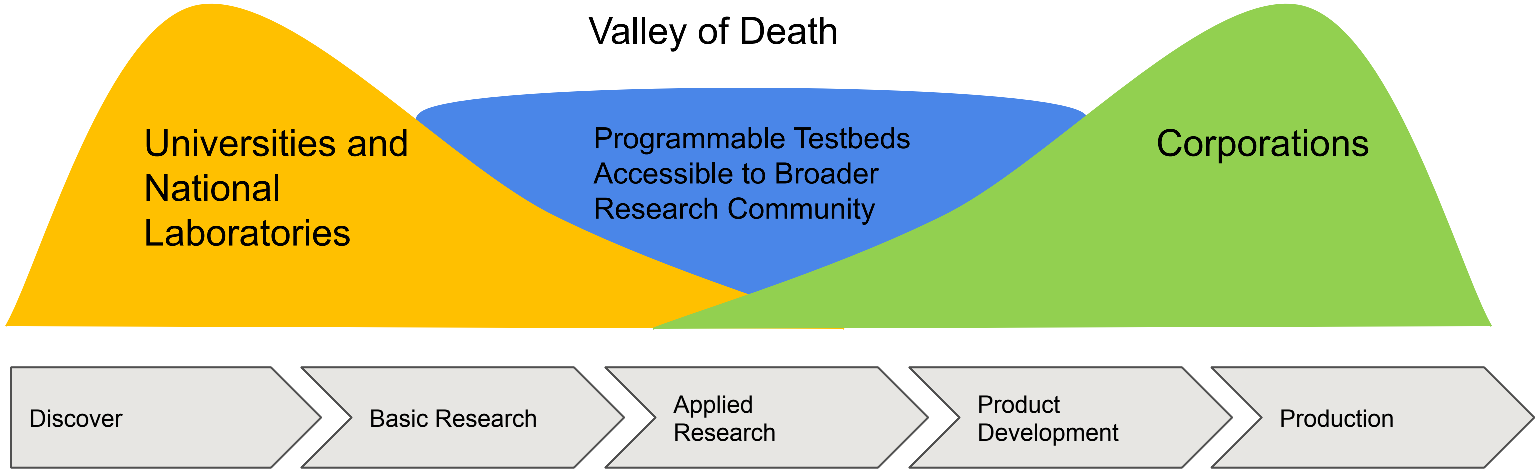
Mihail L. Sichitiu, Professor
Dept. of ECE
NC State University

October 2nd, 2023

Outline

- Why?
  - PAWR Program
  - AERPAW
- Hardware
  - Requirements
  - Options
  - Solutions: LAM, SAM, Rover
- Software
  - Emulation vs. Testbed Modes
  - Software Stack
  - AERPAWlib
  - Sample Applications

Valley of Death for Wireless Research



NSF Platforms for Advanced Wireless Research (PAWR)

POWDER
Salt Lake City, UT

Software Defined Networks and Massive MIMO

COSMOS
West Harlem, NY

Millimeter wave and backhaul research

AERPAW
Raleigh, NC

Unmanned Aerial Vehicles and mobility

ARA
Ames, IA

Rural Broadband

AERPAW Project Team and Partners


Ismail Guvenc
PI, NC State (SDRs, 4G/5G standards, PHY/MAC)


Rudra Dutta
NC State (SDN, architecture, CentMesh)


Mihail Sichitiu
NC State (drones, architecture, CentMesh)


Brian Floyd
NC State (mmW circuits, arrays)


Tom Zajkowski
NC State (UAS operations, FAA permitting)


Lavanya Sridharan
NC State (Project Coordinator)


Ed Rogers
NC State (construction permits)


Ozgur Ozdemir NC State (SDRs, Keysight, Facebook TG)


Vuk Marojevic
MSU (security, SDRs, waveforms, CORNET)


Gerard Hayes
NC State, WRC (wireless and testing)


Yufeng Xin
RENCI, UNC-CH (data models, software architecture control framework)


David W. Matolak
USC (aerial propagation, waveforms)


David Love
Purdue (MIMO, SDRs, agriculture)


Magreth Mushi
NC State (Network Arch. & Platform Ops.)


Mike Barts
WRC (RF, Towers, Antennas, Front Ends)


Asokan Ram
WRC (4G/5G Ericsson deployment)


Alphan Sahin, USC
mmWave Development

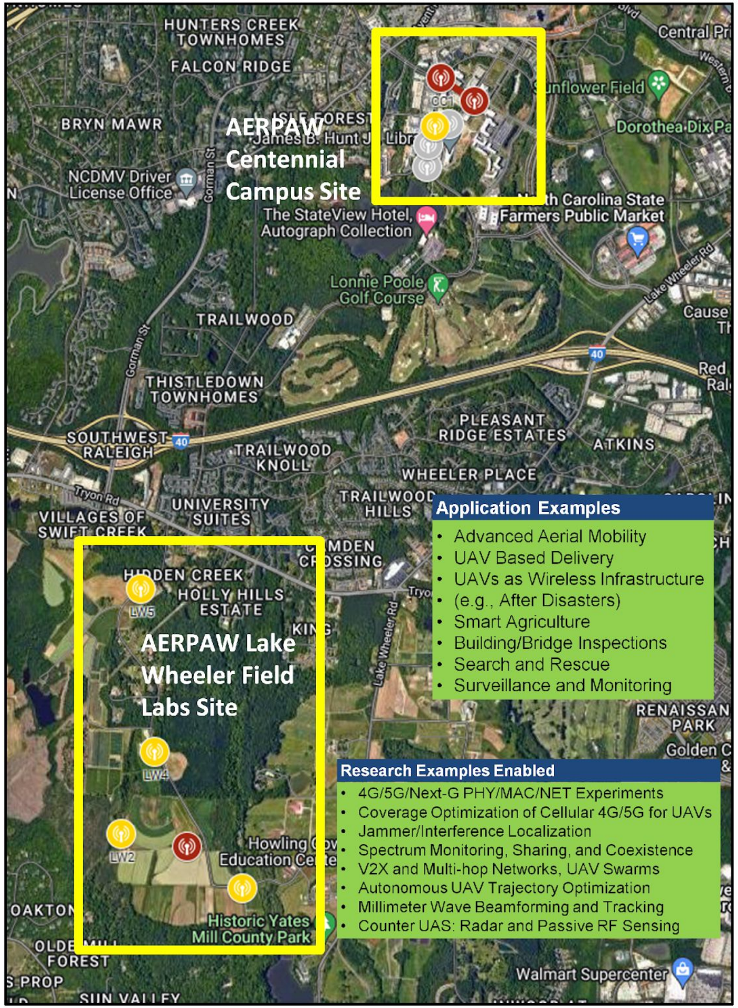

Andrew Balmos, Purdue
LoRa Development


Mark Funderburk, NC State
UAV/UGV Development

Other Personnel:

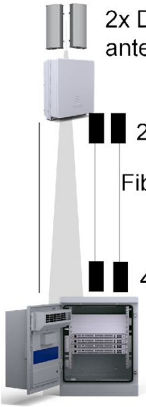
- **Postdoctoral Scholars:** Talha F. Rahman, Sung Joon Maeng, Simran Singh
- **PhD Students:** Anil Gurses, Keith Powell, Ashwini Ganesh, Mrugen Deshmukh, Moin Chowdhury
- **MS Students:** Vishwas Gowda, Sergio Vargas Villar, Jake Meckley, Dylan Snarr, Byron Qi
- **Other WRC and RENCI Personnel:** Thomas Hoover, Michael Stealey, Erica Fu, Erik Scott
- **ITRE Aviation Personnel/Pilots:** Evan Arnolds, Shawn Deardorff, Michael Picinich
- **Undergrad Students:** John Kessler, Keshav Sridhar, Byron Qi, Joshua Moore, Jack Sabine, Ethan Lake, Christian Hoimes, Vaibhav Patil, Aman Anas

Deployment Map for AERPAW Fixed Nodes



AERPAW Fixed Node Equipment

NI Software Defined Radios (SDRs) (Every Fixed Node)



Ericsson 4G/5G Base Station (Lake Wheeler)

Keysight RF Sensors

Facebook Terragraph Radios

LoRa Gateways

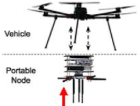
AERPAW Vehicles

LAM UAV

SAM UAV

UGV

Helikite



AERPAW Portable Node Equipment

NI Software Defined Radios

4G/5G Mobile Phones

LoRa Sensors

Compute: Intel NUC10

Project Website: https://aerpaw.org/ **User Manual:** https://sites.google.com/ncsu.edu/aerpaw-wiki

Outline

- Why?
  - PAWR Program
  - AERPAW
- Hardware
  - Requirements
  - Options
  - Solutions: LAM, SAM, Rover
- Software
  - Emulation vs. Testbed Modes
  - Software Stack
  - AERPAWlib
  - Sample Applications

UAV Requirements

Hardware

- Payload: 3-10kg
- Endurance: >30 minutes (at 3kg)
- Stay a small UAS (AUW<25kg)
- Interchangeable payloads
- Easy to exchange batteries
- Reliable
  - Few failure points
  - Redundancy
- Open design desired
- Commodity hardware

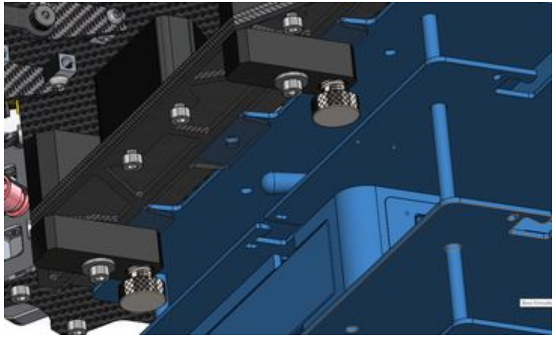
Software

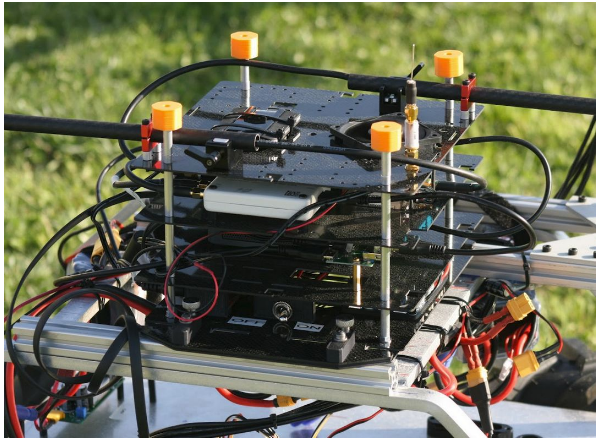
- Open autopilot software
- Open communication software
- Open control software
- Standard Ground Control Station (GCS)
- Software in the loop (SITL) support
- Heterogeneous programming language support
- Different levels of Experimenter access to the software

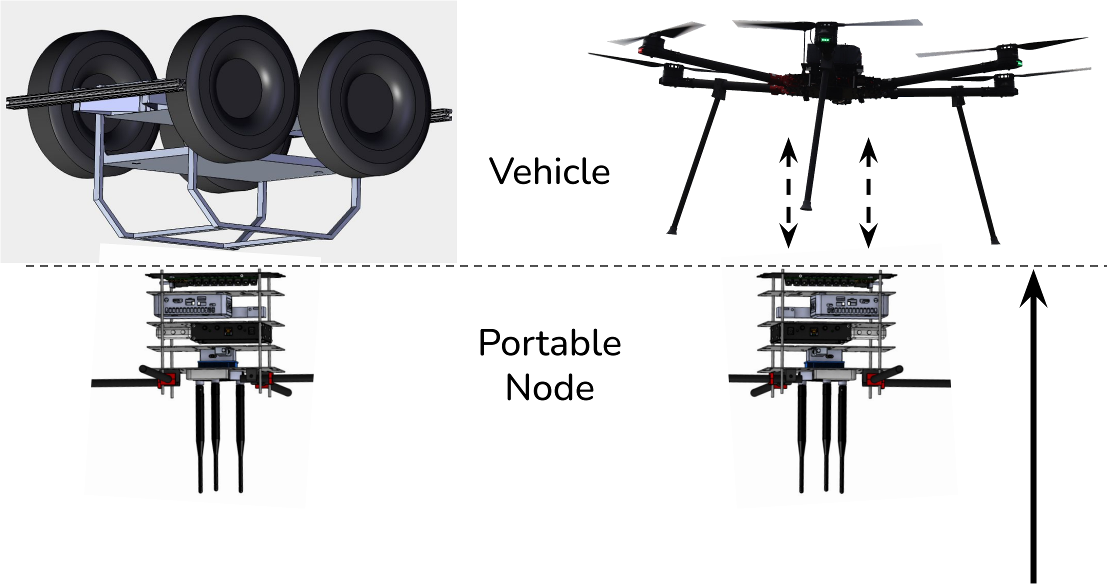
At the heart of AERPAW: programmable radios on programmable vehicles.

How?

Portable Node + Vehicle = Mobile Node

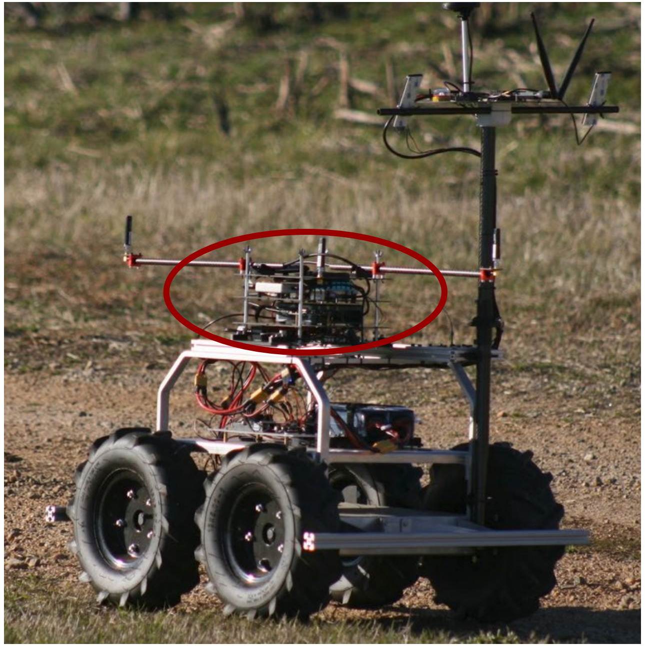




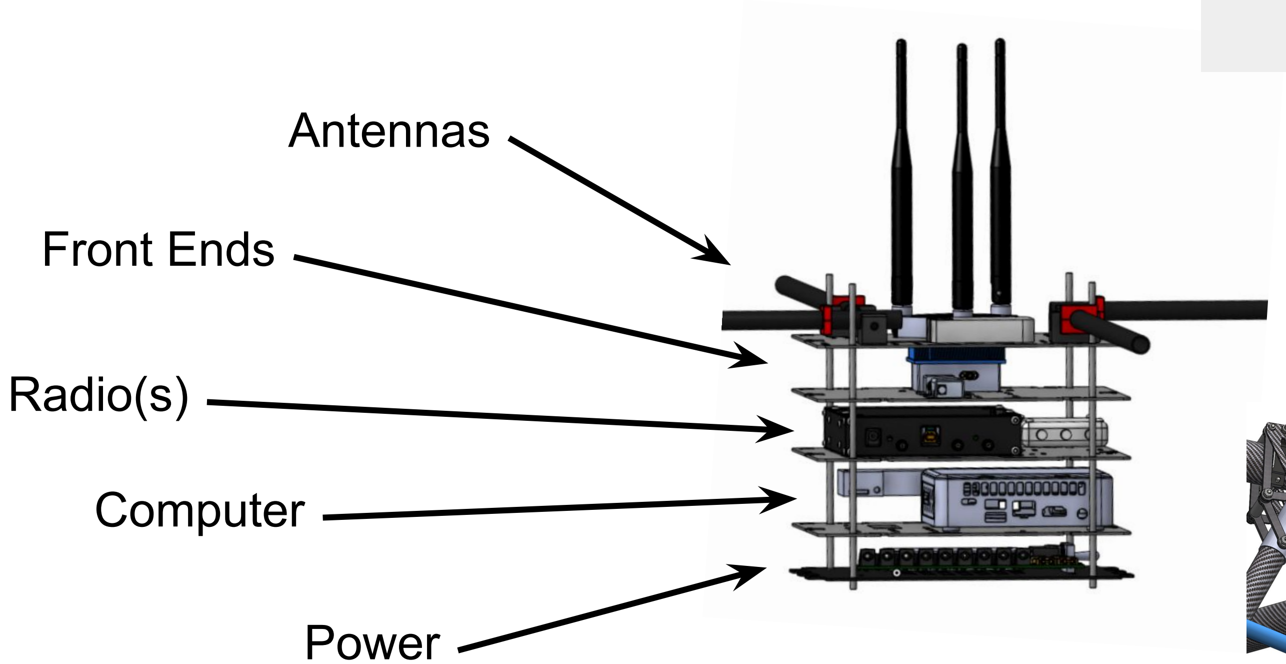


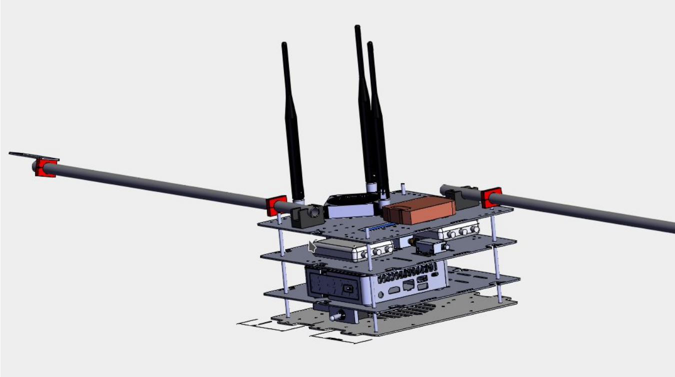
Mechanical, Electrical and Control Interface

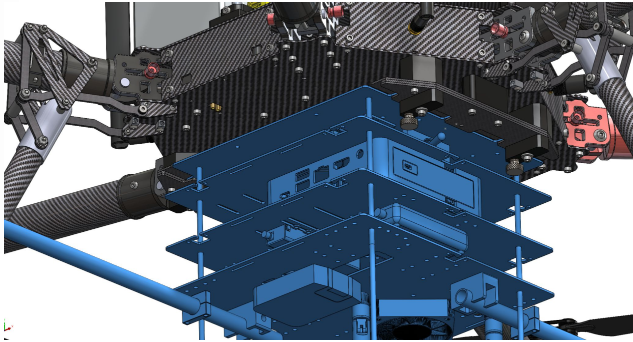





(Large) Portable Nodes







Drone Hardware Choices
DJI Frame + DJI Autopilot

Hardware

- Payload: 3-10kg (6kg)
- Endurance: >30 minutes (at 3kg) (35 minutes without payload, 16 minutes with 6kg)
- Stay a small UAS (AUW<25kg)
- Interchangeable payloads
- Easy to exchange batteries
- Reliable
    - Few failure points
    - Redundancy
- Open design desired
- Commodity hardware

Software

- Open autopilot software
- Open communication software
- Open control software
- Standard Ground Control Station (GCS)
- Software in the loop (SITL) support
- Heterogeneous programming language support
- Different levels of Experimenter access to the software

Drone Hardware Choices
DJI Frame + Open Source AutoPilot

Hardware

- Payload: 3-10kg (6kg)
- Endurance: >30 minutes (at 3kg) (35 minutes without payload, 16 minutes with 6kg)
- Stay a small UAS (AUW<25kg)
- Interchangeable payloads
- Easy to exchange batteries
- Reliable
  - Few failure points
  - Redundancy
- Open design desired
- Commodity hardware

Software

- Open autopilot software
- Open communication software
- Open control software
- Standard Ground Control Station (GCS)
- Software in the loop (SITL) support
- Heterogeneous programming language support
- Different levels of Experimenter access to the software

Difficult Integration

Drone Hardware Choices
COTS Frame + Open Source AutoPilot

Hardware

- Payload: 3-10kg
- Endurance: >30 minutes (at 3kg)
- Stay a small UAS (AUW<25kg)
- Interchangeable payloads
- Easy to exchange batteries
- Reliable
  - Few failure points
  - Redundancy
- Open design desired
- Commodity hardware

Software

- Open autopilot software
- Open communication software
- Open control software
- Standard Ground Control Station (GCS)
- Software in the loop (SITL) support
- Heterogeneous programming language support
- Different levels of Experimenter access to the software

COTS Drone Frame - Lessons Learned

Tarot T18 octocopter

Large AERPAW Multicopter (LAM)

Max Takeoff Weight: 30kg (25kg - Part 107)

“Standard” Battery: 2x 25Ah 6S Li-HV (1140 Wh)

No Payload Flight Time: 47min

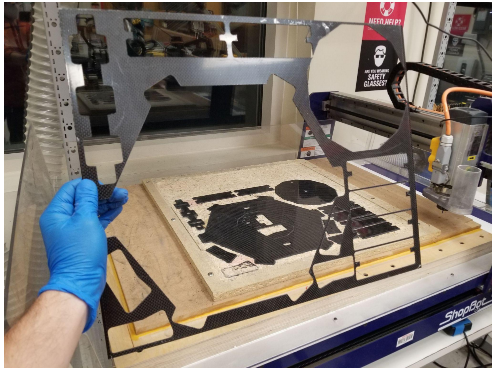
Flight Time w/ 3kg Payload: >35min

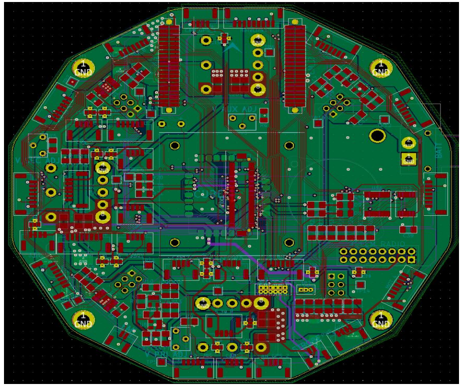
Payload w/ Standard Battery: 16kg (11 kg - Part 107)

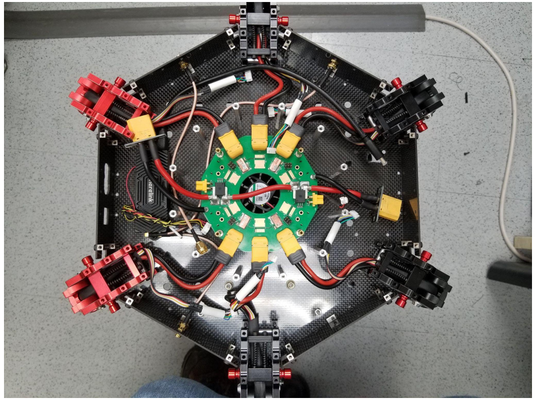
Propellers: 6x 23”

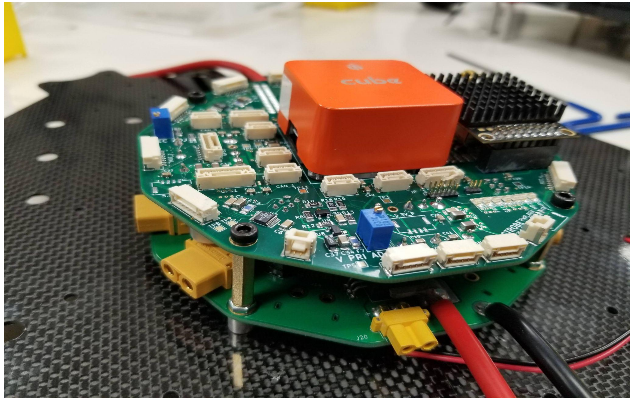
- Triple Redundant Heated IMUs
- Dual GPS With RTK Support
- Redundant Flight Controller Power Supplies
- Quick Attach System for Payload and Batteries
- 20 Km Standard Control / Telem Range
- Independent 900 MHz and 2.4Ghz Telemetry Systems
- 10 - 28 VDC Power Supply for Payloads (up to 300W)
- Capable of Fully Autonomous Operation

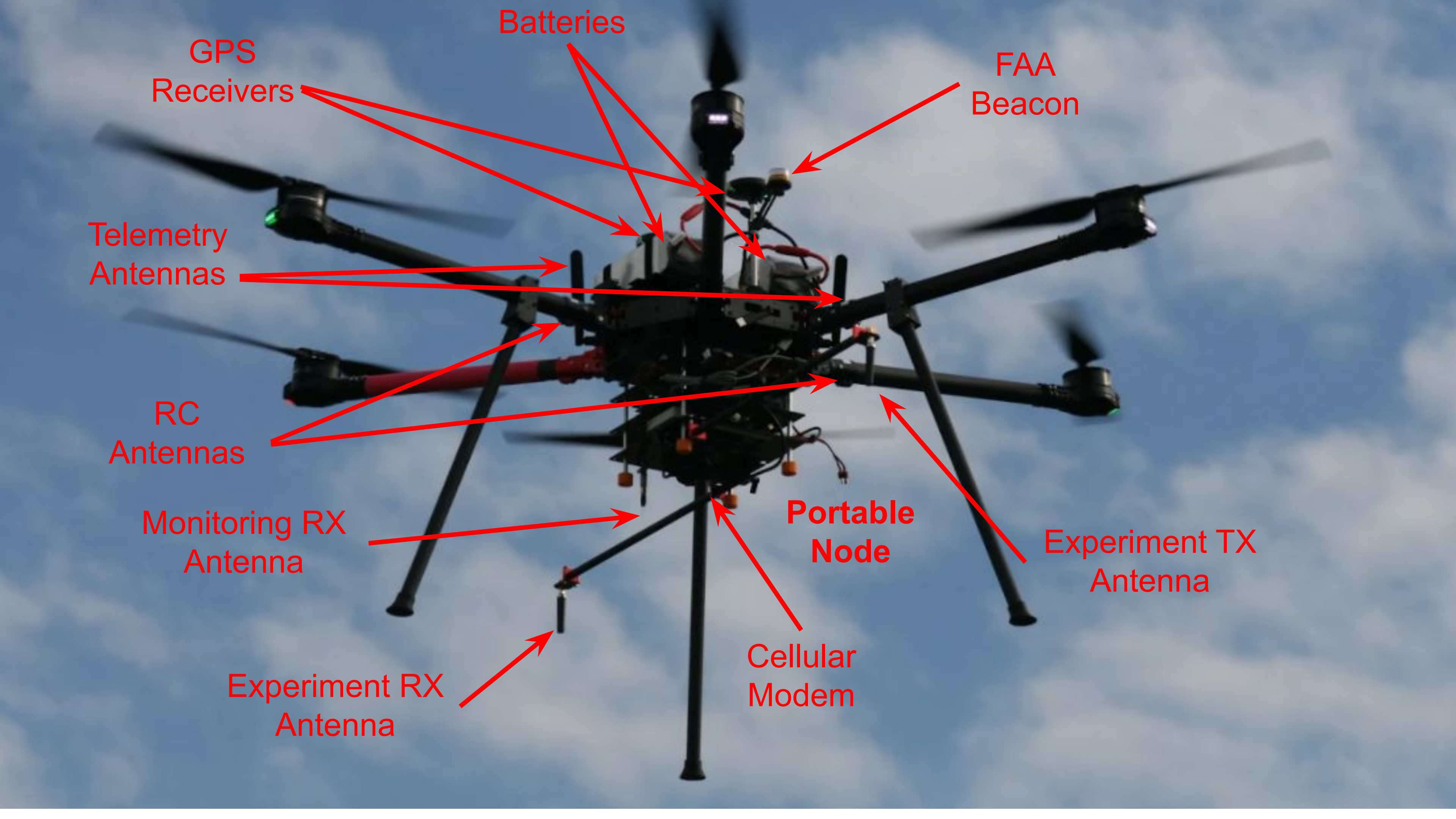
AERPAW LAM Frame and Electronics











Batteries
GPS Receivers
FAA Beacon
Telemetry Antennas
RC Antennas
Monitoring RX Antenna
Portable Node
Experiment TX Antenna
Experiment RX Antenna
Cellular Modem

LAM Test Stand - 4 Degrees of Freedom

Small AERPAW Multicopter (SAM)

Max Takeoff Weight: 6.5kg

“Standard” Battery: 2x 7Ah 6S LiPo (310 Wh)

Payload With Standard Battery: 3kg

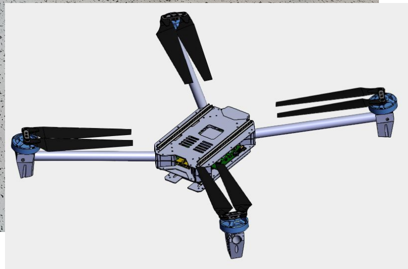
No Payload Flight Time: 60min (Std battery)

Flight Time With 500g Payload: 50min

Propellers: 4x 18”

Interface: Small Portable Node

- 82 min flight time with 30Ah Semi-Solid Li-ion battery and LattePanda Computer + LTE Modem (300g)
- Same flight controller as LAM6



Size Comparison

LAM6 vs SAM4

3 x LAM6 vs DJI Phantom

Rover

Weight: 30kg

“Standard” Battery: 25 Ah 6S LiPo

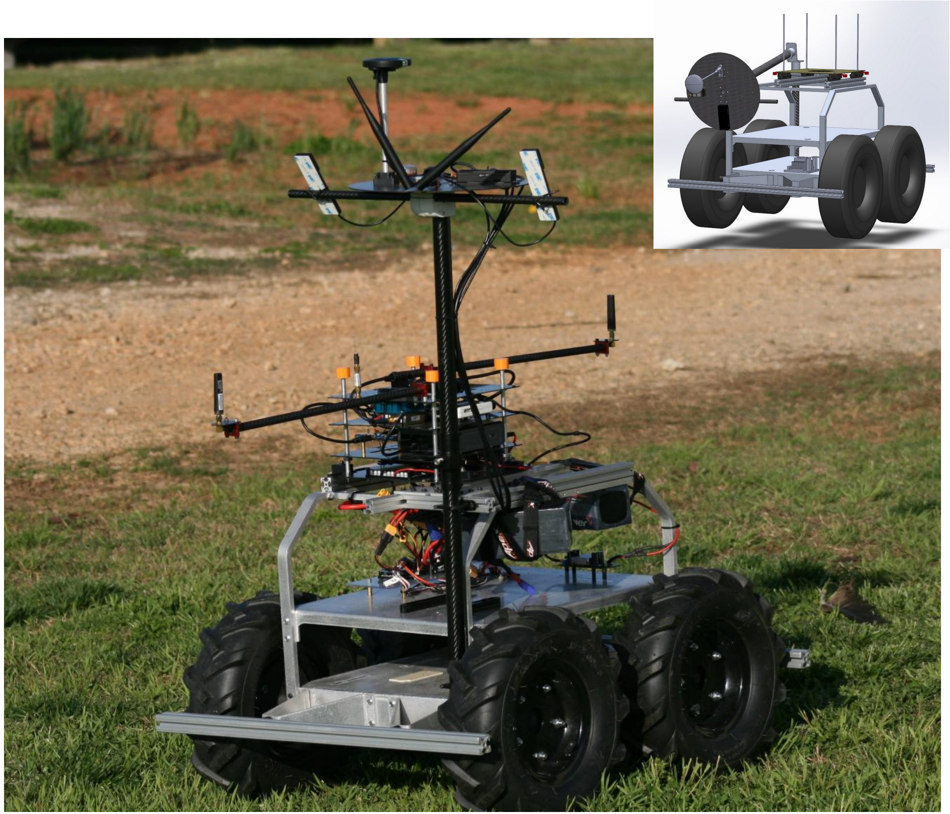
Endurance: 120min

Wheels: 4x 13”, skid steer

Max Payload: 15kg

Interface: Large Portable Node

Suspension: None :(




Portable
Node

Outline

- Why?
  - PAWR Program
  - AERPAW
- Hardware
  - Requirements
  - Options
  - Solutions: LAM, SAM, Rover
- Software
  - Emulation vs. Testbed Modes
  - Software Stack
  - AERPAWlib
  - Sample Applications

The Main Problem

At the heart of AERPAW: programmable radios on programmable vehicles.

How?

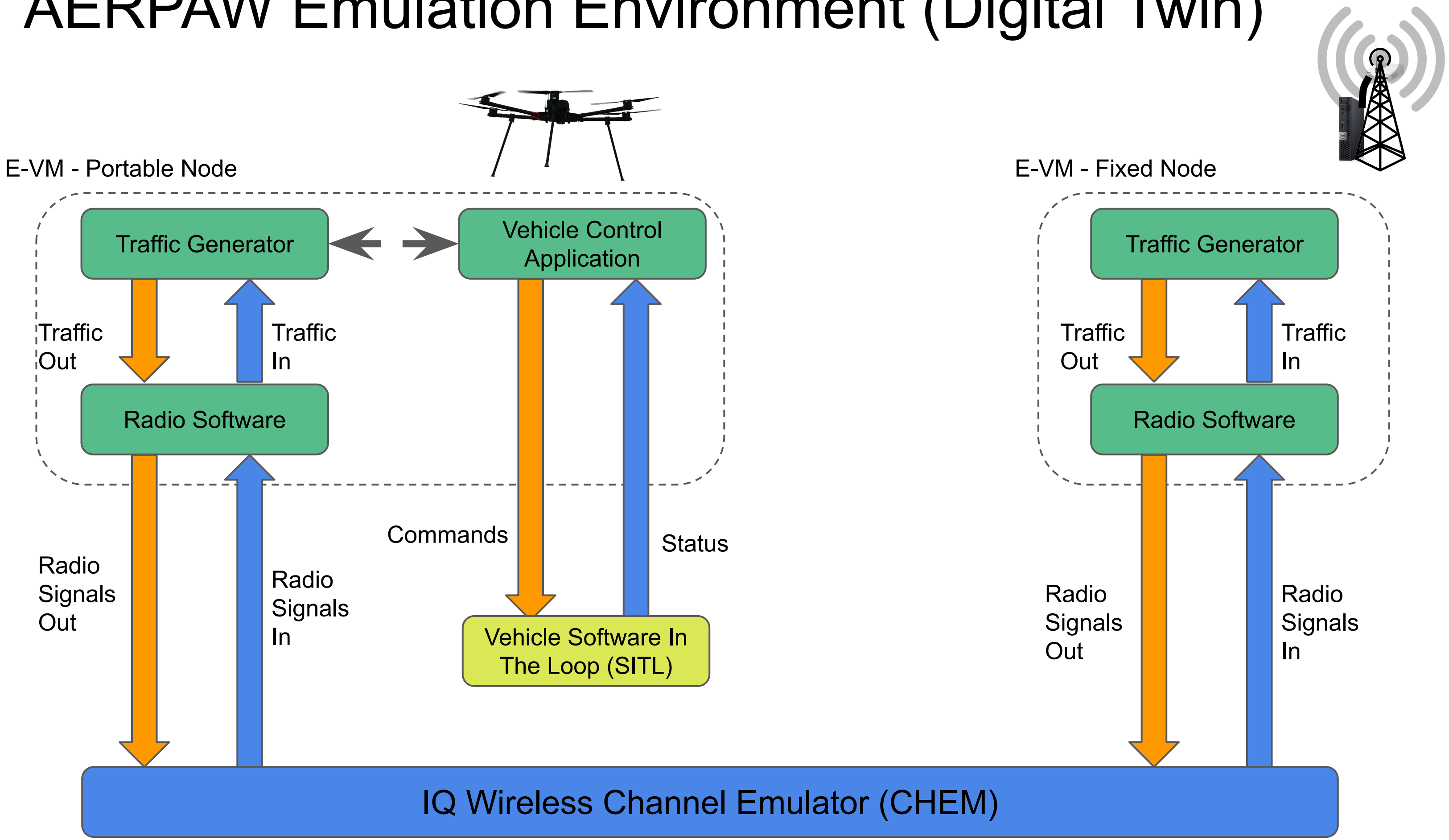
- How can an Experimenter to program the drones?
- How can the Operators make sure that the programming is safe?
- All the other testbeds give the Experimenters live access to the testbed. Can we do the same?

Experimenter Code

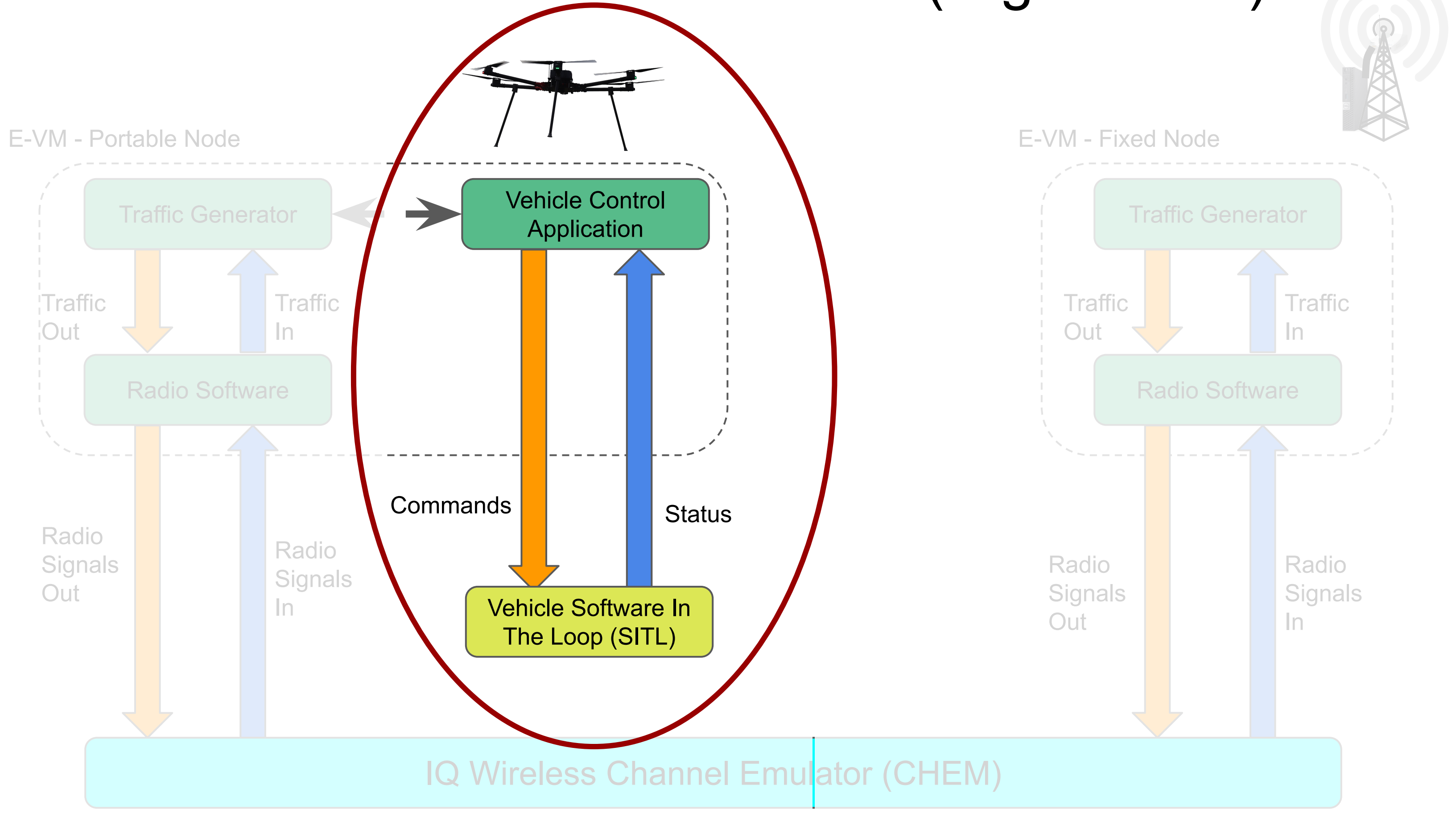
Existing Code

AERPAW Code

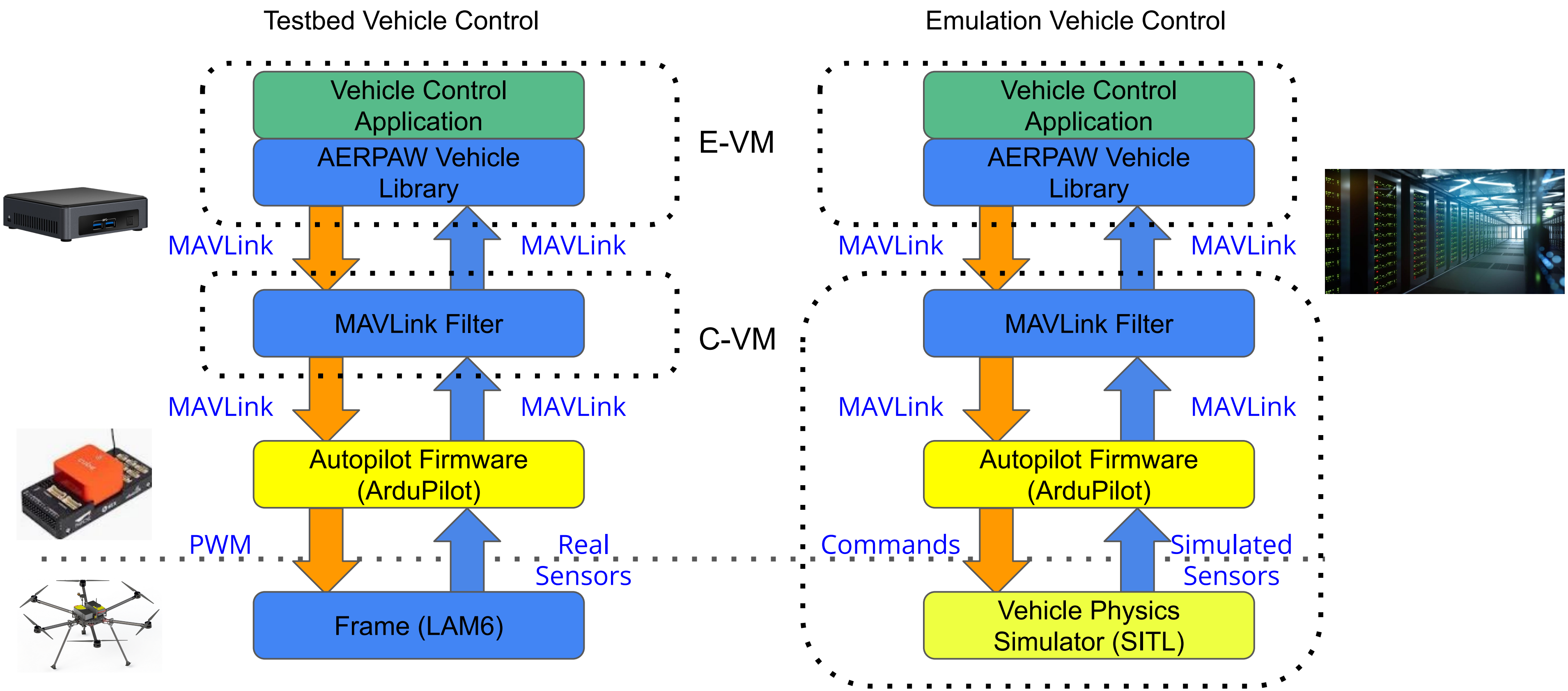
AERPAW Emulation Environment (Digital Twin)



AERPAW Emulation Environment (Digital Twin)



Vehicle Control in Testbed vs. Emulation



MAVLink Filter

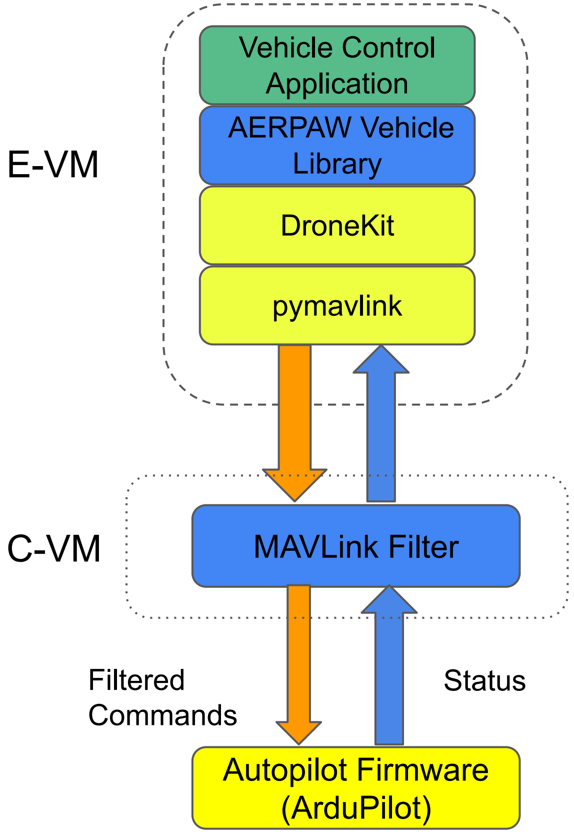
Drones and Rovers are allowed to:

1. Navigate to a specified waypoint (latitude, longitude, altitude)
2. Change the yaw angle (specified in degrees, clockwise, with respect to true North).
3. Takeoff at a specified altitude. The specified altitude has to be between the minimum and maximum altitudes. The altitude is specified in meters with respect to ground level (AGL).
4. Change the drone mode to GUIDED, or LAND

Each movement from a Drone or a Rover has to:

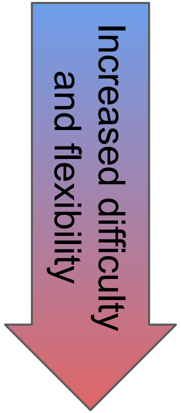
1. Stay entirely in the geofenced area.
2. Stay entirely out of a no-go zone.
3. Stay at an altitude higher than MIN_ALT (20M) and lower than MAX_ALT (100m)

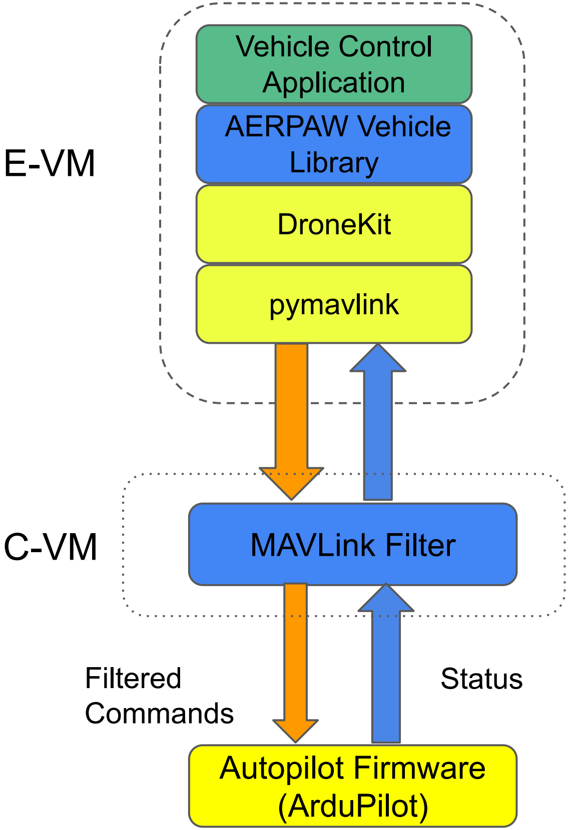
No other MAVLink commands pass through (including arm).



Writing Vehicle Software

- Use “Preplanned Trajectory” and edit the mission plan file
- Modify “Preplanned Trajectory” (e.g., to change parameters in the traffic or radio software, log different vehicle parameters, etc.)
- Write a different application based on AERPAW Vehicle Library (see “SquareOff” example with relative movement)
- Write a different application based on DroneKit
- Write a different application based on pymavlink
- Write a different application based on MAVLink (any programming language)





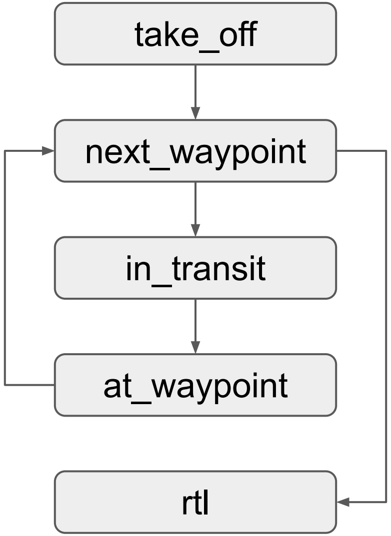
AERPAWLib

Native support for state machine based experiments in Python syntax

Treats decorated functions as states and handles transitioning

Offers asynchronous control of vehicle to solve scheduling problem

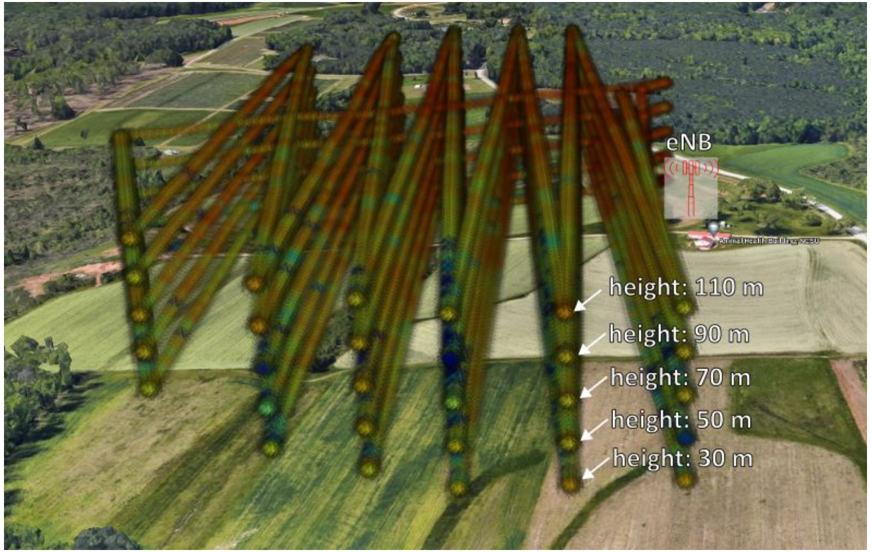
Support constructs such as GPS coordinates or waypoint files




```python
@state(name="take_off", first=True)
async def take_off(self, drone: Drone):
    takeoff_alt = self._waypoints[self._current_waypoint]["pos"][2]
    await drone.takeoff(takeoff_alt)
    return "next_waypoint"

@state(name="next_waypoint")
async def next_waypoint(self, drone: Drone):
    self._current_waypoint += 1
    waypoint = self._waypoints[self._current_waypoint]
    if waypoint["command"] == 20:
        return "rtl"

    coords = Coordinate(*waypoint["pos"])
    in_background(drone.goto_coordinates(coords))
    return "in_transit"

@state(name="in_transit")
async def in_transit(self, drone: Drone):
    avg_ping_latency = in_background(self._ping_latency("127.0.0.1", 5))
    print(f"Average ping latency: {avg_ping_latency}ms")

    await drone.await_ready_to_move()
    return "at_waypoint"

@timed_state(name="at_waypoint", duration=3)
async def at_waypoint(self, _):
    return "next_waypoint"

@state(name="rtl")
async def rtl(self, drone: Drone):
    await drone.goto_coordinates(drone.home_coords)
    await drone.land()
```

AERPAWLib (continued)

- Provides support for coordinated multi-vehicle movement
- Provides support for autonomous drone movement
- Is the basis for all the vehicle sample applications we support.
- Handles all AERPAW vehicles (LAMs, SAMs, Rovers, Helikite, and more)



AERPAW Sample Vehicle Applications

- Preplanned Trajectory
  - The “workhorse”
  - Supports GUI-based definitions of plan files
  - Supports pauses at waypoints, changes in speed, and changes in yaw
  - Provides software hooks for “in transit” and “at waypoint” states
- GPS Logger
  - For “read-only” of the state of the vehicle (no control)
- Multi-Vehicle Control
  - The Tracer and the Orbiter
- Autonomous Vehicle Movement
  - AERPAW Find A Rover (AFAR) Challenge

QGroundControl Integration

- Fully featured ground control station
- Support for MAVLink
- Support for multiple vehicles
- Used by Experimenters during development and Operators in Testbed

Layered Approach to Safety

Hardware

- Redundancy
  - 6 propellers (Thrust/Weight>4)
  - 3 IMUs
  - 2 GNSSs
  - 2 EKFs
  - 2 (different!) yaw sources
  - Cellular + RC C2 Links
  - 2 autopilot power sources
  - 2 x 2 C2 antennas
- Reduction in points of failure
  - Custom PCBs reduce wiring
- All arm checks enabled
- All failsafes enabled

Software

- Emulator allows for safe experiment development and testing
- MAVLink Filter restricts movement of individual drones (geofence, no-go zones, speed, altitude, mode, landing site, etc.)
- OEO System for coordinating multiple drones
- Safety-checker API - allows Experimenters to check if a command is safe.
- aerpawlib reduces the availability of commands
- ADS-B Alert System

Safety Pilots

- Fly under Part 107 rules
- Pre-flight checklists
- Manual override control
- Use of visual observers
- Chooses RTL or S-RTL
- Arm the drones

Questions?

Join IEEE VTS at
www.vtsociety.org

Follow IEEE VTS on social media

Website
www.vtsociety.org

Facebook
facebook.com/IEEEVTS

Twitter
@IEEE_VTS

LinkedIn
www.linkedin.com/company/ieee-vehicular-technology-society