PACMULE

(Payload Acquisition Carrier Multi-Utility Location Estimator)

Initial Design Review

GPS-Denied Navigation Platform

The PACMULE Team

David Amanor
- Telecommunications
- Hardware Programming (VHDL/FPGA)
- Signal Processing
- Software – MATLAB

Tavin Clary
- Telecommunications
- Signal Processing
- Software – MATLAB

Joshua Eddy
- Autonomous Vehicle Design
- Aircraft Stability and Control
- Microcontrollers
- Software – MATLAB, Python, Java

Niti Madhugiri
- Astronomy
- Image Processing
- Robotic Motion Planning / AI
- Software – MATLAB, Python, C

Nicole Ogden
- Hardware Programming (VHDL/FPGA)
- Microcontrollers – Arduino, Raspberry Pi, NETduino, BeagleBone Black
- Software – MATLAB, C/C++, Android, Python

Robert Ryan
- Visualization
- Mechatronics
- Software – Java, Android, Python, C/C++

GPS-Denied Navigation

Conditions:

- Flying Donkey without Cargo Hold (i.e., no payload)
- GPS disabled
- Low-cost embedded solution (< $500 for navigation components)
- Must carry organizer-supplied flight logger (60 g)

Criteria to win the sub-challenge:

- Successfully complete a 1 km route, in both directions, at 8 a.m., noon and 5 p.m. (i.e. 6 km in total)
- Out at 300m above ground level
- Back at 50m above ground level
- No damage to the aerial vehicle
- Stay within the Flight Corridor

The winner will have the smallest average deviation from the Flight Corridor's center line

A Three-Pronged System

The PACMULE will employ three navigation modules:

Inertial Measurement Unit

GSM Chipset

Computer Vision System

Why these three?

The PACMULE, at a bare minimum, should be able to determine orientation, position, and velocity. Ideally, PACMULE should be designed for **maximal redundancy** in the case of subsystem failure.

The three subsystems chosen for the PACMULE offer maximal sensory overlap to preclude the possibility of total sensory failure at any point.

Inertial Navigation

An IMU employs a combination of:

- Accelerometer
- Gyroscope
- Compass
- Barometer
- Thermometer

By measuring accelerations and angular velocities, the PACMULE can make displacement estimates.

GSM Chipset

The GSM chipset communicates with nearby cell towers to ascertain:

- Mobile Country Code
- Mobile Network Code
- Location Area Code
- Cell ID
- Signal Strength

This data can be employed to estimate the PACMULE's position by trilateration.

Computer Vision System

Optical Flow Analysis compares two consecutively taken images to determine the flow of pixels, enabling estimations of velocity.

Horizon-Line Determination locates the horizon, allowing for orientation estimates.

Inertial Measurement Unit

1. IMU to Microcontroller
   - Output – Data Values
2. Smartphone Sensors
   - Output – Lat/Long
   - Output – Data Values

GSM Trilateration

1. GSM Chipset
   - Output – MCC, MNC, LAC, Cell ID, Signal Strength
2. Smartphone
   - Output – Lat/Long

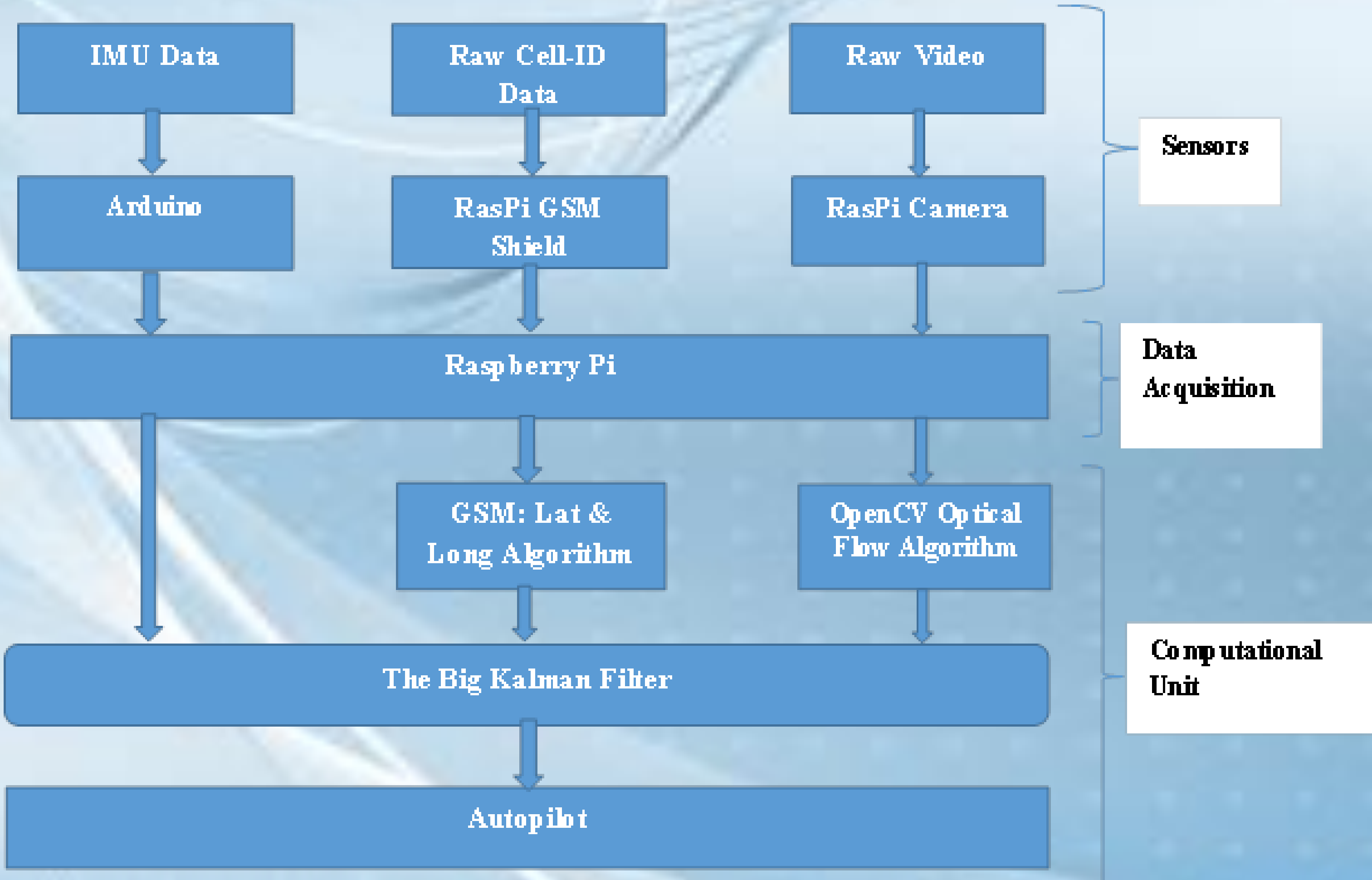
Computer Vision

1. Night Vision Camera, into Raspberry Pi
   - Output – Velocity Vector
2. External Camera, interfaced to Smartphone
   - Output – Velocity Vector
3. Onboard Smartphone Camera
   - Output – Lat/Long
   - Output – Velocity Vector

Package One

Languages: Python, C/C++, AT Commands

Adafruit IMU, GSM Shield, Night-Vision Camera into Raspberry Pi



Package One – Pros and Cons

Pros:

- Modularity
- Individual components are replaceable

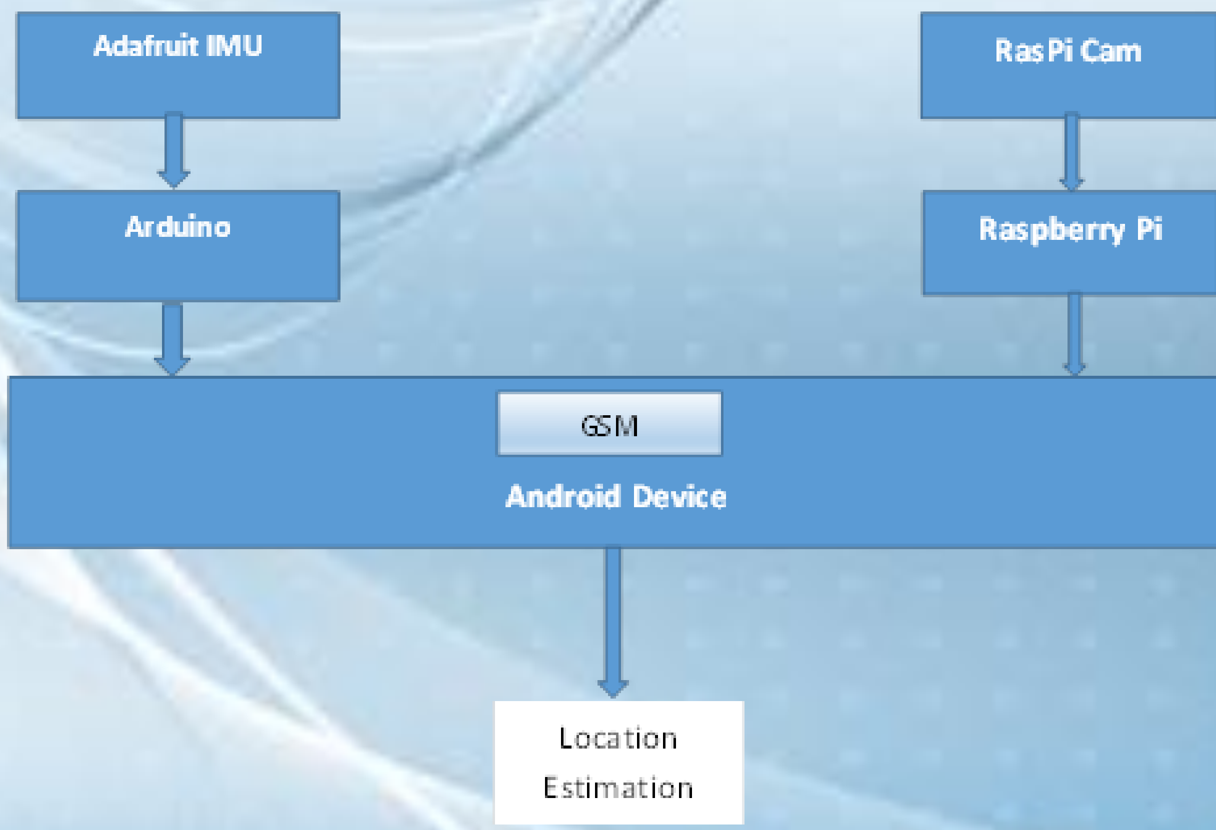
Cons:

- Complicated
- Small Processor
- Possible interfacing communication issues

Package Two

Languages: Python, C/C++, Java *or* Python and Java

Either Adafruit IMU or Android IMU, Android Cell Phone, Night-Vision Camera into Raspberry Pi



Package Two – Pros and Cons

Pros:

- Greater processing power
- Less hardware interfacing

Cons:

- Only one accessible hardware connection
- Less customizable

Going from *Simulation* to *Simulator*

Simulation:

- Representation of the Flying Donkey in an environment
- Each component of the PACMULE represented within model

Simulator:

- Adding Hardware-in-the-Loop (HIL) functionality

Simulation

- Select 3D software capable of modeling both the PACMULE and the environment
- PACMULE System-Level Model:
  - IMU Module
    - IMU
    - **Arduino** (HIL capable)
  - Cellular Tracking Module
    - **Smart Phone** (HIL capable) or GSM
  - Image Processing Module
    - Camera
    - **Raspberry Pi** (HIL capable)
  - Vehicle
  - Terrain (ground, trees, rocks, buildings, gophers)
  - Environment (wind, lighting, gravity, cell signal, etc.)

Project Timeline

Week	Device Tasks				Simulation Tasks				
4	Modules				Coarse Models				
5									
6		Packages				Fine Models			
7							Sim		
8									
9			PACMULE					True Models	
10				Trials					HIL
11									
12									

“Opportunities to Excel”

The Learning Curve

- Hardware Interfacing
- Programming Languages
- Visualization

Delivery Concerns

- Subsystem Integration
- Testing
- Debugging / Unforeseen Issues
- Budget
- Vehicle interoperability

A Few Hanging Questions

What are NASA's expectations?

For the Flying Donkey organizers:

- Floor/ceiling altitudes? Required altitudes?
- Exact specification of flight corridor?
- Confirmation of given GPS landing zones?
- Foreknowledge of waypoints? Competition timing?
- Competitor-provided infrastructure?
- Competition data logger?

Moving Forward

- Select hardware and algorithms for each module
  - Hardware choices will likely dictate programming needs
  - Possible choices: Arduino UNO, Raspberry Pi, Samsung Galaxy S1
  - Possible Computer Vision algorithms: Lucas Kanade, Horn-Schunck
- Select IDE’s based on programming needs
  - Possible choices: IDLE, Eclipse
- Select visualization software
  - Possible choices: jMonkeyEngine, Microsoft Robotics Developer Studio, Panda3D