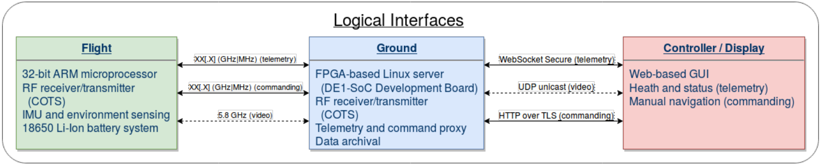
Fault-Tolerant Quadcopter

ECE 453 Project Proposal (Fall 2018)
University of Wisconsin-Madison

Vaughn Kottler, Mayank Katwal, Cooper Green

Overview



Logical Interfaces

Flight

32-bit ARM microprocessor
RF receiver/transmitter
(COTS)
IMU and environment sensing
18650 Li-Ion battery system

XX[.X] (GHz|MHz) (telemetry)
XX[.X] (GHz|MHz) (commanding)
5.8 GHz (video)

Ground

FPGA-based Linux server
(DE1-SoC Development Board)
RF receiver/transmitter
(COTS)
Telemetry and command proxy
Data archival

WebSocket Secure (telemetry)
UDP unicast (video)
HTTP over TLS (commanding)

Controller / Display

Web-based GUI
Heath and status (telemetry)
Manual navigation (commanding)

Quadcopter – Battery-powered, four-motor flying machine

Ground Station – Linux server managing the quadcopter's radio endpoint, hosts wired-network services (i.e. telemetry)

Web-based UI – A modern dashboard for visualizing data and manually commanding the vehicle

Learning Objectives

Exposure to:

- Radio-frequency communication
- Control theory
- System-level engineering
- SoC platform(s)

Also:

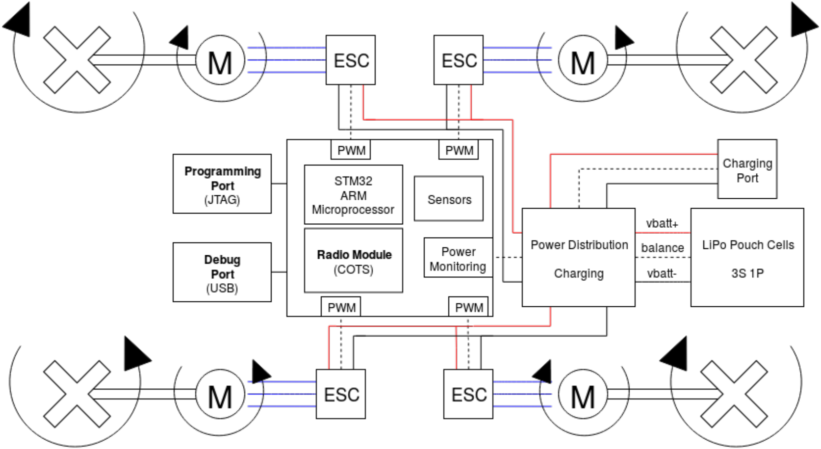
- Data pipelining in the aerospace/avionics problem space
- Modern user-interface design and implementation

Features

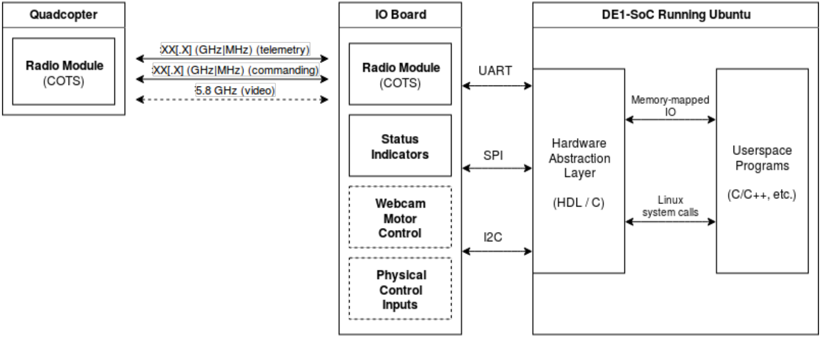
Single-Fault Tolerant – Land safely in the event of communication “heartbeat timeout”

Telemetry Archival – Implement long-term telemetry storage for post-flight data analysis

Quadcopter


M
ESC
ESC
M
PWM
PWM
Programming Port (JTAG)
STM32 ARM Microprocessor
Sensors
Charging Port
Radio Module (COTS)
Power Monitoring
Power Distribution
Charging
vbatt+
balance
vbatt-
LiPo Pouch Cells
3S 1P
Debug Port (USB)
PWM
PWM
M
ESC
ESC
M

Ground Station



User Interface

API Commands (A)

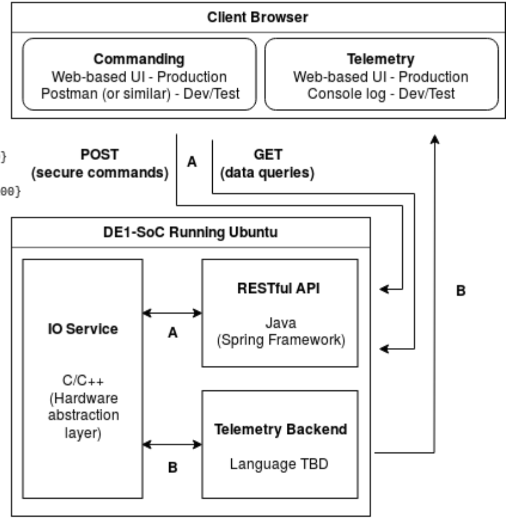
HTTP over TLS

```
https://host/move/up/{0 - 100}
https://host/move/down/{0 - 100}
https://host/move/left/{0 - 100}
https://host/move/right/{0 - 100}
https://host/move/forward/{0 - 100}
https://host/move/back/{0 - 100}
https://host/move/rotate/{-100 - 100}
...
```

Telemetry Data (B)

Secure WebSocket

```
telemetry_packet {
  timestamp: 1536646557,
  age: 15,
  type: "sensors",
  data: [
    temperature: 22,
    pressure: 101325,
    gyro: {
      rate_xy: -1,
      rate_xz:  2,
      rate_yz: -3
    }
    ...
  ]
}
```



Inter-process communication over local-loopback socket streams (TCP)

Milestones

1) Establish wireless communication between the vehicle and ground station

2) Establish percentage-based throttle control over each motor

3) Establish manual-commanding capability to the vehicle from a web-based user interface

4) View live telemetry from a web-based user interface

5) Sense angular velocity via gyroscope and force experienced via inertial measurement unit

Milestones cont.

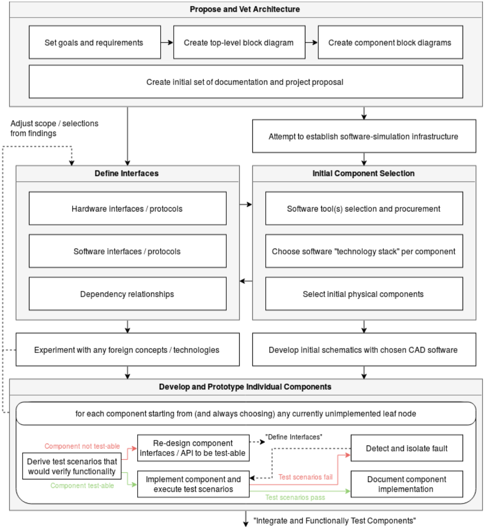
6) Develop a control algorithm to fly in a stable hover or holding pattern

7) Extend control algorithm to control for velocity in three axes to achieve controlled motion

8) Sense relative altitude

9) Extend control algorithm to control for a specific *delta-y* (perpendicular to ground plane)

Initial Development and Prototyping



Propose and Vet Architecture

Set goals and requirements

Create top-level block diagram

Create component block diagrams

Create initial set of documentation and project proposal

Adjust scope / selections from findings

Attempt to establish software-simulation infrastructure

Define Interfaces

Hardware interfaces / protocols

Software interfaces / protocols

Dependency relationships

Initial Component Selection

Software tool(s) selection and procurement

Choose software "technology stack" per component

Select initial physical components

Experiment with any foreign concepts / technologies

Develop initial schematics with chosen CAD software

Develop and Prototype Individual Components

for each component starting from (and always choosing) any currently unimplemented leaf node

Derive test scenarios that would verify functionality

Component not test-able

Component test-able

Re-design component interfaces / API to be test-able

"Define Interfaces"

Implement component and execute test scenarios

Test scenarios fail

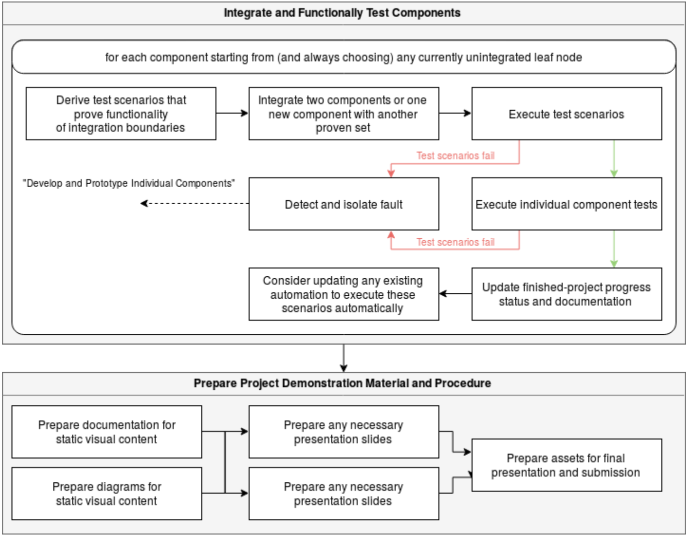
Test scenarios pass

Detect and isolate fault

Document component implementation

"Integrate and Functionally Test Components"

Final Stages



Integrate and Functionally Test Components

for each component starting from (and always choosing) any currently unintegrated leaf node

Derive test scenarios that prove functionality of integration boundaries

Integrate two components or one new component with another proven set

Execute test scenarios

Test scenarios fail

"Develop and Prototype Individual Components"

Detect and isolate fault

Execute individual component tests

Test scenarios fail

Consider updating any existing automation to execute these scenarios automatically

Update finished-project progress status and documentation

Prepare Project Demonstration Material and Procedure

Prepare documentation for static visual content

Prepare any necessary presentation slides

Prepare diagrams for static visual content

Prepare any necessary presentation slides

Prepare assets for final presentation and submission