PiQuad™

April 4th 2018

Robotics In Flight, LLC

Outline

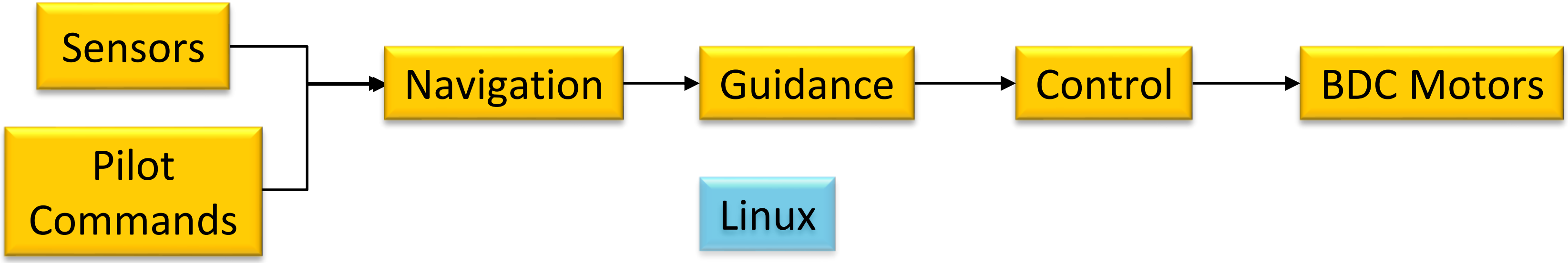
- PiQuad™ Overview
- Functional Block Diagrams
- Datasheet
- Radio Controller
  - RC Command Layout
  - Message Structure
- Open Architecture
  - SW Arch / Interfaces
  - Modes / State Machine
  - Expansion
  - Interface Control Document
  - Telemetry
- Control Design
- Next Steps

PiQuad™ Overview

PiQuad™ Overview

- ***PythonPilot™ (flight controller) developed entirely in the USA***
- A turnkey, foundational platform well suited for:
  - education (mechatronics labs, capstone projects, etc.)
  - product development
  - commercial & military R&D

- Open architecture, open source
  - coded in Python - running on Linux
  - open architecture and interfaces
  - user configurable
    - SW functions and/or HW devices can be altered or replaced
  - modular and scalable processing modules



PiQuad™ Overview

PiQuad™ is a quadcopter with a flight controller coded in Python and running on the Raspberry Pi, a single board computer realizing key functions needed for controlled flight – guidance, navigation, control, sensor reception, motor commands, and a remote pilot interface

Platform INCLUDES

-- Raspberry Pi A+ or Pi 3

-- Turnigy 9x 2.4 GHz radio controller

-- Ultimate GPS receiver

-- SJCAM 12 MP 1080p camera & stabilized mount

-- 5000 mAHr LiPo battery and charger

-- WiFi

-- Tether clips for constrained flight testing

FEATURES

-- Coded in Python and running on Linux

-- Open architecture, open interfaces

-- Flight data logging and in-flight plotting

-- 8 Ch Remote pilot command interface

-- Inertial Navigation:

- **IMU—angular rates, linear accelerations**
- **magnetometer**
- **baro-altimeter**

-- Lost RC link return home

Target Projects & Labs

-- Plant parameter measurement

-- IMU calibration

-- PID & LQG/LQR attitude control design

-- Swarm & multi-node formation flying

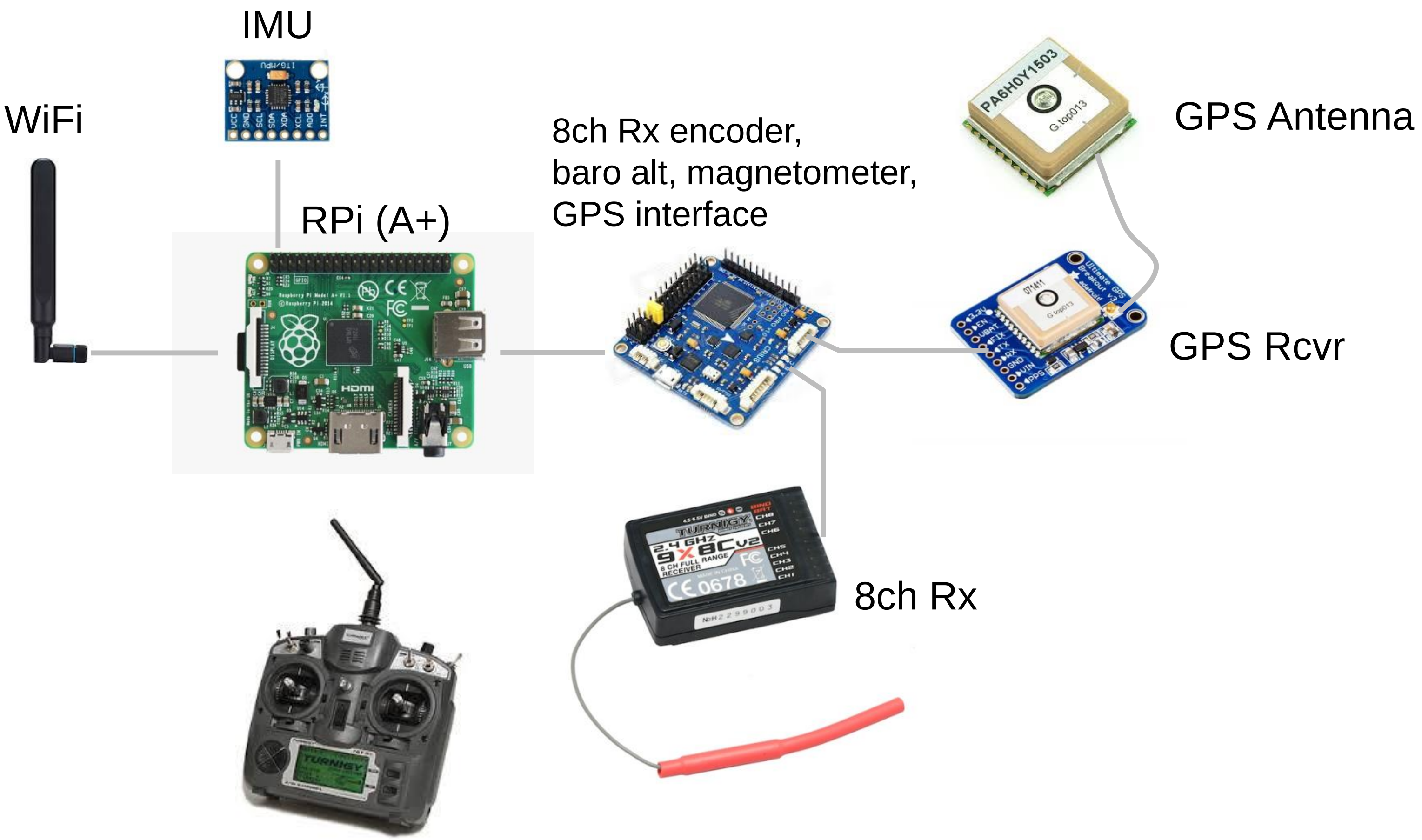
-- Differential GPS navigation

-- Vision-aided navigation, image recognition

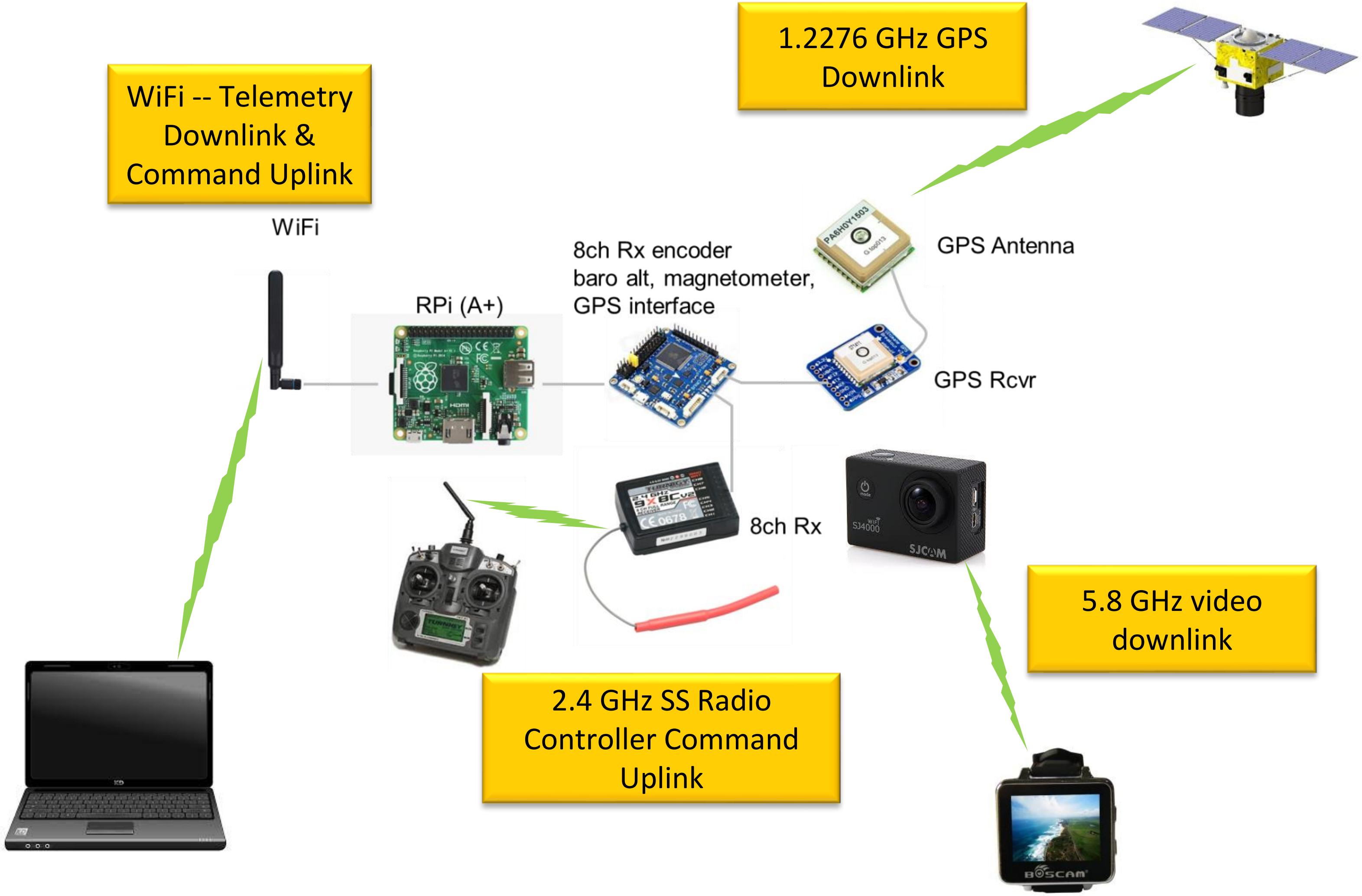
PiQuad™ – Datasheet

- **SWAP:**
  - Size: 28” diameter at rotor tip x 9” high
  - Weight: 2.8 lb; with camera – 3.5 lb
  - Power: 5000 mAhr LiPo
- **Lift Capacity –** 6.8 lb (3 kg)
  - 4 lb (1.4 kg) in excess of its own weight
- **Flight time** – 15-20 minutes
- **Vertical altitude hold accuracy:** 2 ft 1σ
- **Max. altitude –** 400 ft AGL (limited by FAA UAS rules, settable parameter)
- **Max. vertical velocity –** 2 m/s (default setting)
- **Max. pitch/roll setpoint command** – 15 deg (default setting) to 30 deg (max.)
- **Max. horizontal velocity –** 5 m/s (depending on max. tilt angle setting)
- **Data Links:** Bi-directional WiFi providing realtime telemetry including GPS location tracking and plotting, 2.4GHz SS Command uplink, 5.8 GHz video downlink

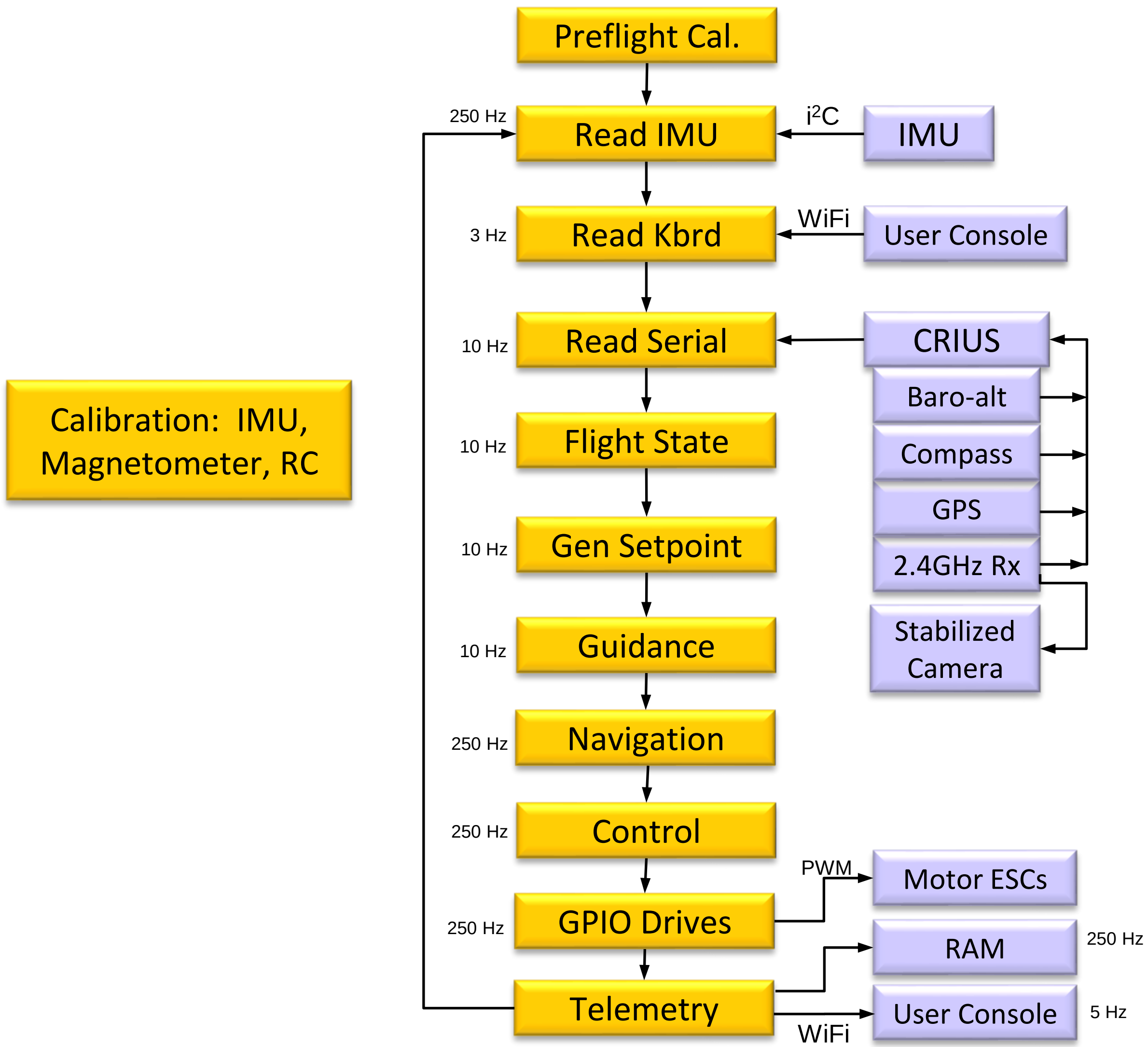
HW Interconnect Diagram



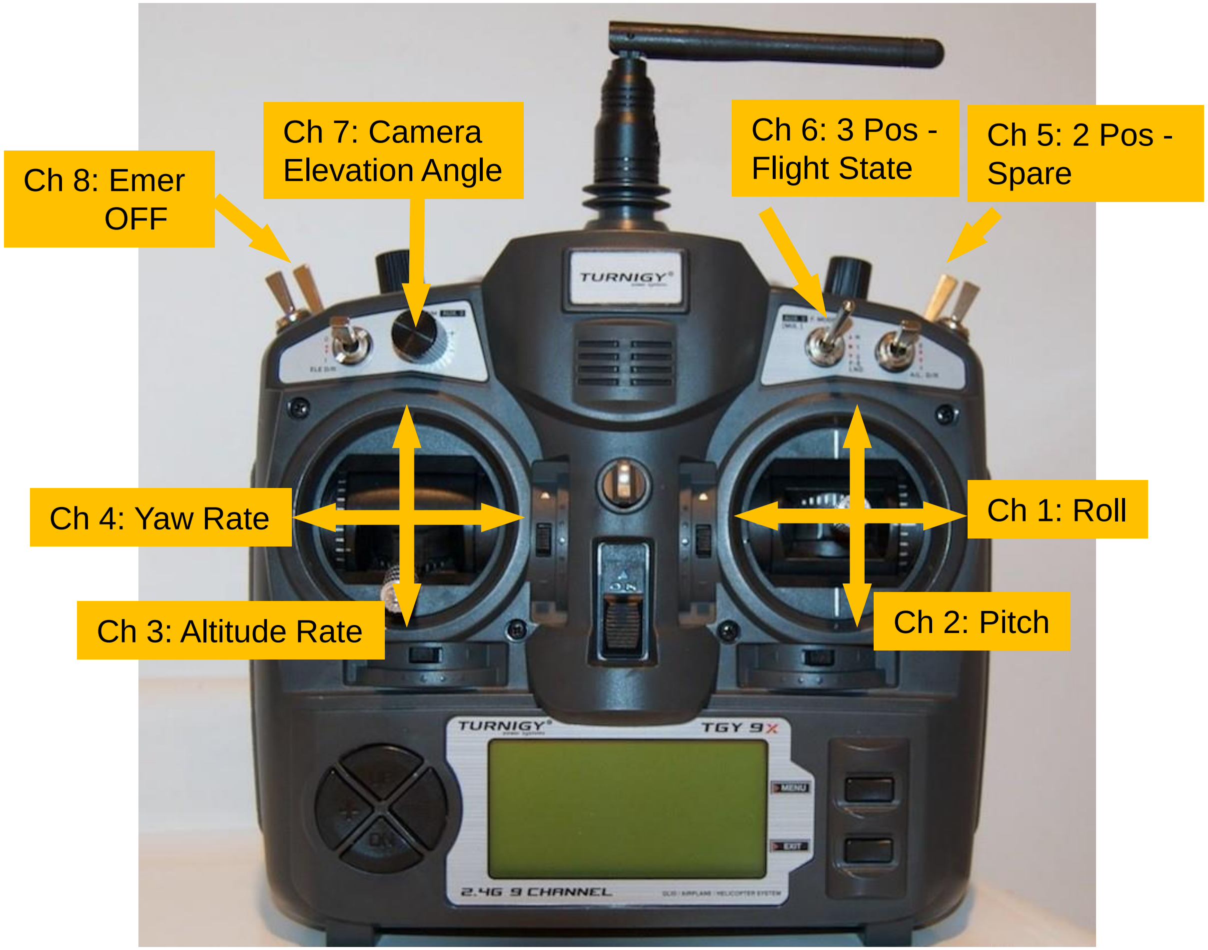
Functional Block Diagram – Data Links



Functional Block Diagram – SW Flow



Radio Controller Command Interface



- 8 channels received as PWM signals at RC Receiver and available as digital data on PiQuad
- PiQuad currently using 7 leaving 1 spare available

Radio Controller Command Message Structure

```python
# ----------------------------------------------------------------------------------
# Radio Controller inputs
# ----------------------------------------------------------------------------------
new_stick = 0
if ((key_cnt-15)%30 == 0):  # Read RC transmit at ~8 Hz
    new_stick = 1
    [stk_r, stk_p, stk_v, stk_y, rx_alt, magx, magy, magz, lat, lon, gps_alt, ch5, ch6, ch7, ch8] = \
        rccmd(rc_port, serial_port_index, alt_zero_data, prnt_flag)
    # Catch an serial read glitch and set magnetometer values to previously measured outputs
    if magx == []:
        magx = magxp
        magy = magyp
        magz = magzp
        stk_r = stk_rp
        stk_p = stk_pp
        stk_v = stk_vp
        stk_y = stk_yp
        #print "CAUGHT GLITCH"
    else:
        magxp = magx
        magyp = magy
        magzp = magz
        stk_rp = stk_r
        stk_pp = stk_p
        stk_vp = stk_v
        stk_yp = stk_y
```

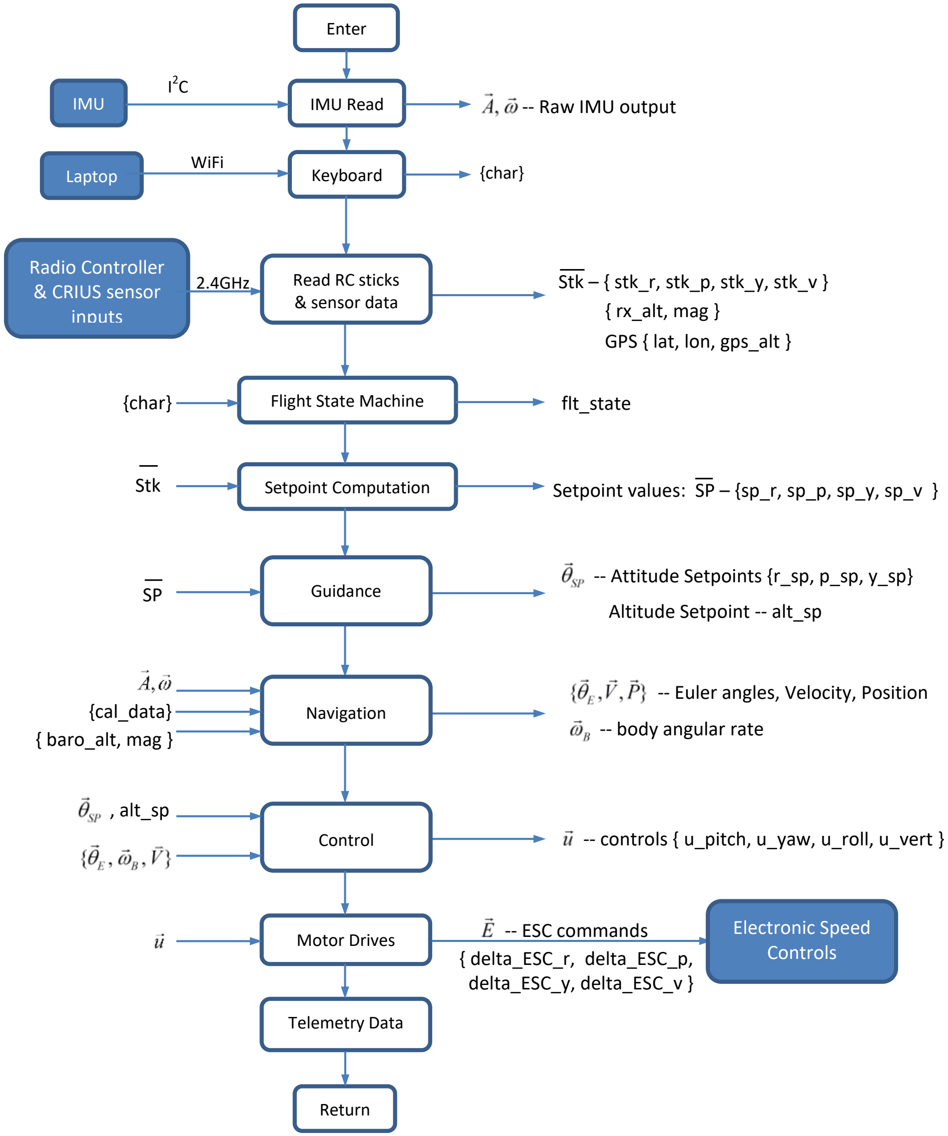
Python file   length : 21040   lines : 526   Ln : 315   Col : 27   Sel : 0 | 0   Dos\Windows   UTF-8 w/o BOM   INS

Read RC and other data

Table 3.4-2: rccmd() outputs

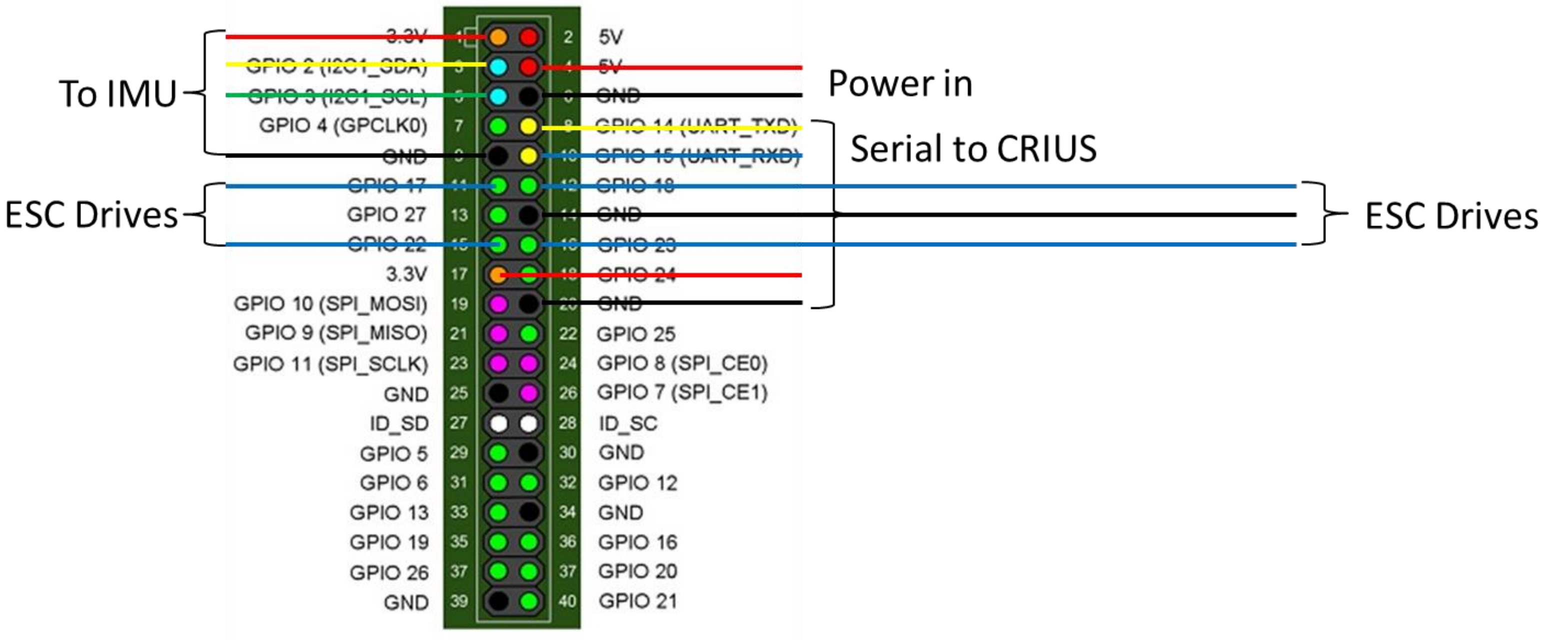
Variable Name	Description	Units
stk_r, stk_p, stk_v, stk_y	Raw RC input stick positions read directly from RC	counts
rx_alt	Received baro-altitude relative to starting altitude (alt_zero_data)	meters
magx, magy, magz	Raw magnetometer readings	counts
lat, lon	GPS latitude and longitude	degrees
gps_alt	GPS altitude	m
ch5, ch6, ch7, ch8	Raw RC input stick positions channels 5 – 8	counts

Python-Pilot™ – an open-source autopilot



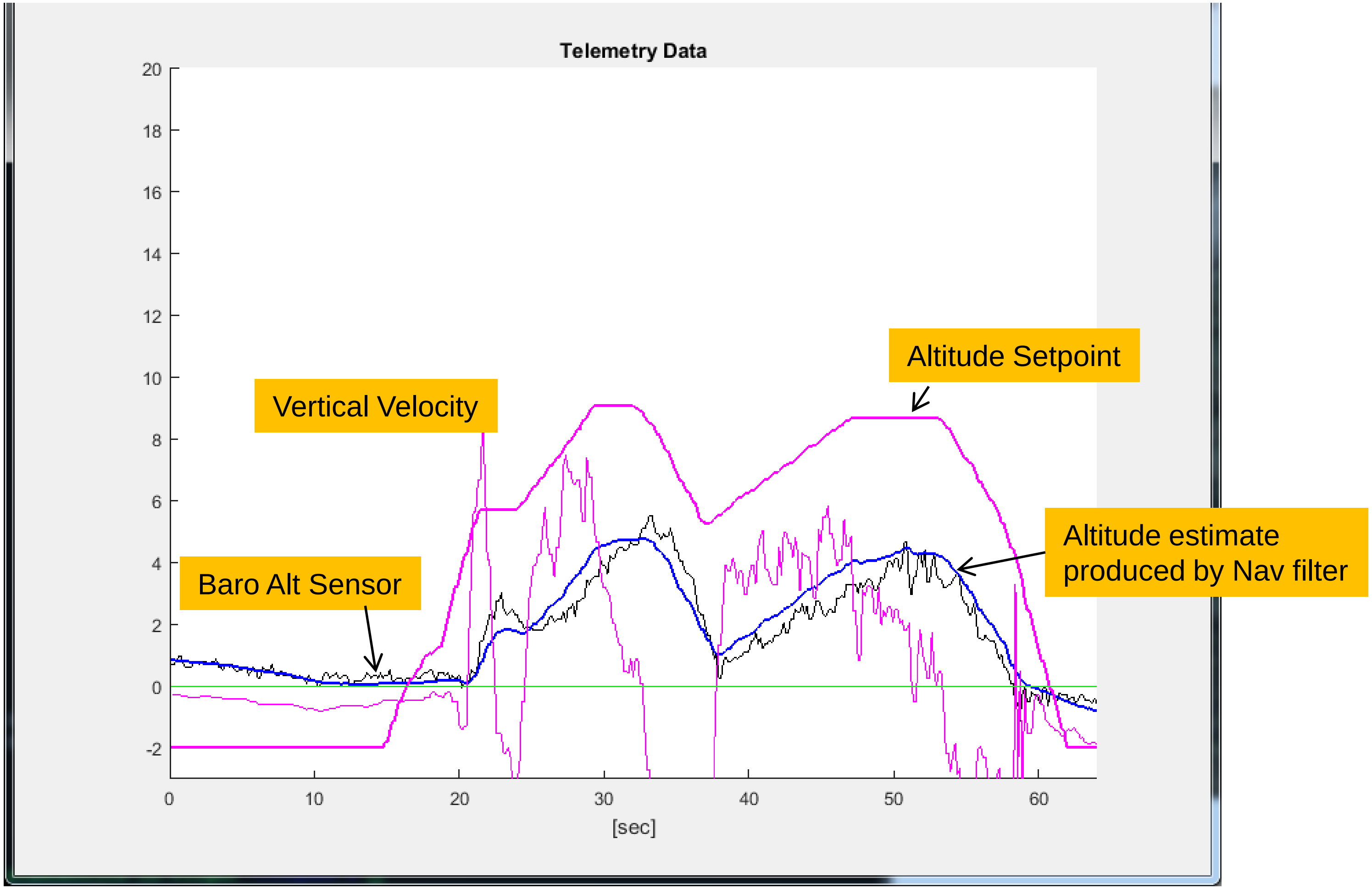
Open Architecture – HW Expansion

- Interfaces available on the Pi
  - 17 unused GPIO shown below
    - SPI bus
    - ID_SD/SC EPROMM boot pins
  - PiCam video in
  - HDMI
  - audio



Open Architecture -- Telemetry

- Telemetry content is open and adjustable



Open Architecture – ICD

PIQUAD™ FLIGHT CONTROL SOFTWARE

INTERFACE CONTROL

Version *NJIT*

2017 May

State Machine and Flight States

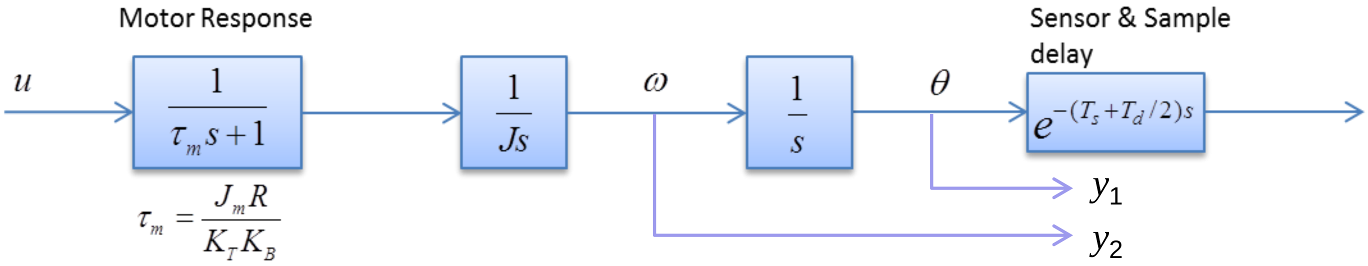
Table 3.5-1: statemachine() inputs

Variable Name	Description	Units
flt_state	Current flight state index -1 – Preflight Auto-calibration 0 – Idle 1 – Flight 2 – Emergency Off 3 – User Commanded Calibration Mode	-
on_time	Time elapsed since the program's execution start time	sec
char	Input character	
sp_r, sp_p, sp_y, sp_v,	Setpoints: roll, pitch, yaw, and vertical	-
umag	Control magnitude	counts
statics_rc	RC input statics	
mpu6050	MPU6050 class instantiation identifier	
serial_port_index	Serial port index	
cal_data	IMU calibration data	
RC_cal_data	RC calibration data	
heading_bias	Heading calibration data (raw heading angle reading when oriented with x axis pointing true north)	rad
telem_flag	Telemetry storage ON/OFF flag (1/0)	
telm_idx	Telemetry storage group index	
cv	Control variable	
em_off	Emergency Off index (1 – Prop commands set to 0, Emergency Off Mode)	
idle_off	Idle Off index (1 – Prop commands set to 0, Idle Mode)	

Flight Controller Design

- 6 Control Loops:
  - 3 Attitude (roll/pitch/yaw)
  - 1 Altitude Location
  - 2 Horizontal Location

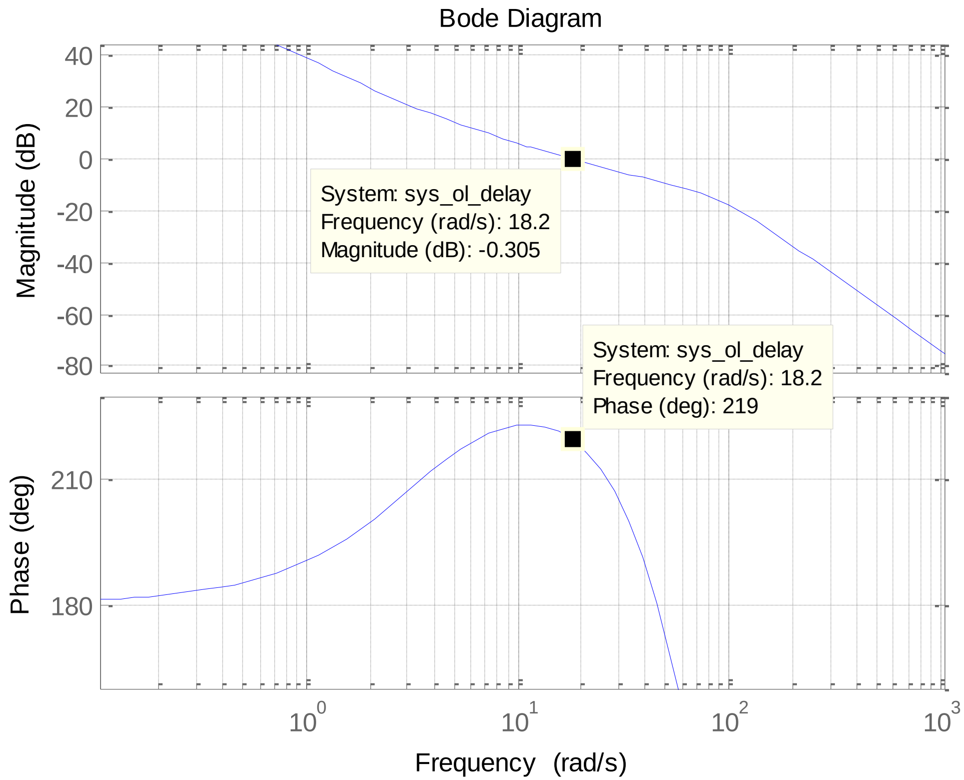
- Example – the roll/pitch attitude stabilization design is an LQG/LQR controller based on a single axis design model having 1 input, 2 outputs
  - Truth Model



  - includes the pure delay associated with the S/H and the IMU
    - T_s = .008 and T_d = .003 seconds
    - 8 msec latency (IMU) and sampling time delay of about 3 msec (sample frequency ~ 330 Hz)

Flight Controller Design

- The open loop frequency response of the LQG/LQR controlled plant including the sample and IMU delay



Phase margin – 39 deg
Bandwidth – 2.8 Hz

Next Steps

- ***Robotics In Flight*** *is pursuing collaborative academic or government sponsored research opportunities that will showcase the PiQuad^{TM} and Python-Pilot^{TM} flight controller*