PYTHON PILOT™ FLIGHT CONTROL SOFTWARE

INTERFACE DESCRIPTION DOCUMENT

Version -

2020 July

VERSION HISTORY

This is the initial draft Interface Control Document prepared to describe the open interfaces and operation of the PiQuad Flight Control Software, Rev. -.

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
-	D. Haessig	28 Jul 2020	na		Initial release

TABLE OF CONTENTS

1. INTRODUCTION

This Interface Control Document (ICD) captures the necessary information required to effectively define the PythonPilot and PiQuad's software interfaces. The intended audience is all users of the PythonPilot™ flight control software and the PiQuad™ system.

1.1 SAFETY WARNING

Users of the PiQuad should use good judgement and extra caution when flying or ground testing the PiQuad. Because this product is flexible and modifiable, be aware of the potential for introducing or exciting a flaw (a bug) in the flight software or hardware that could affect or degrade performance, potentially creating a dangerous condition. Use motion limiting restraints – tethers – whenever testing new features or code modifications, and only fly untethered after a long enough period of constrained and issue-free flight testing. Obey all FAA rules (https://www.faa.gov/uas/getting_started/fly_for_work_business/). See Appendix B for additional warnings.

2. REFERENCE DOCUMENTS AND INFORMATION

2.1 PYTHON AND LINUX

The flight control software described in this document is written in Python and runs within the Linux operating system. For information and documentation pertaining to Python and Linux we refer the reader to:

- https://www.python.org/
- https://www.linuxfoundation.org/

3. INTERFACE CONTROL INFORMATION

3.1 TOP LEVEL FLOW DIAGRAM

The flight control top level diagram given in Figure 3-1 shows the order of execution of the functions present in the code. Hardware interfaces are those that connect to the solid blue blocks. The key outputs from each block are listed to the right of each software function block. Key inputs to software blocks are shown entering from the left. Not all IO data are listed on this figure. A complete listing of the input and output data are provided in the individual sections given below, one for each function.

3.2 IMU (MPU6050) READ

Calling function: C = mpu6050.ReadData()

The array C contains the IMU sensed body accelerations and angular rates:
C = [Ax, Ay, Az, Gx, Gy, Gz]

- Ax, Ay, Az - IMU accelerations (G's)
- Gx, Gy, Gz - IMU angular velocities (deg/sec)

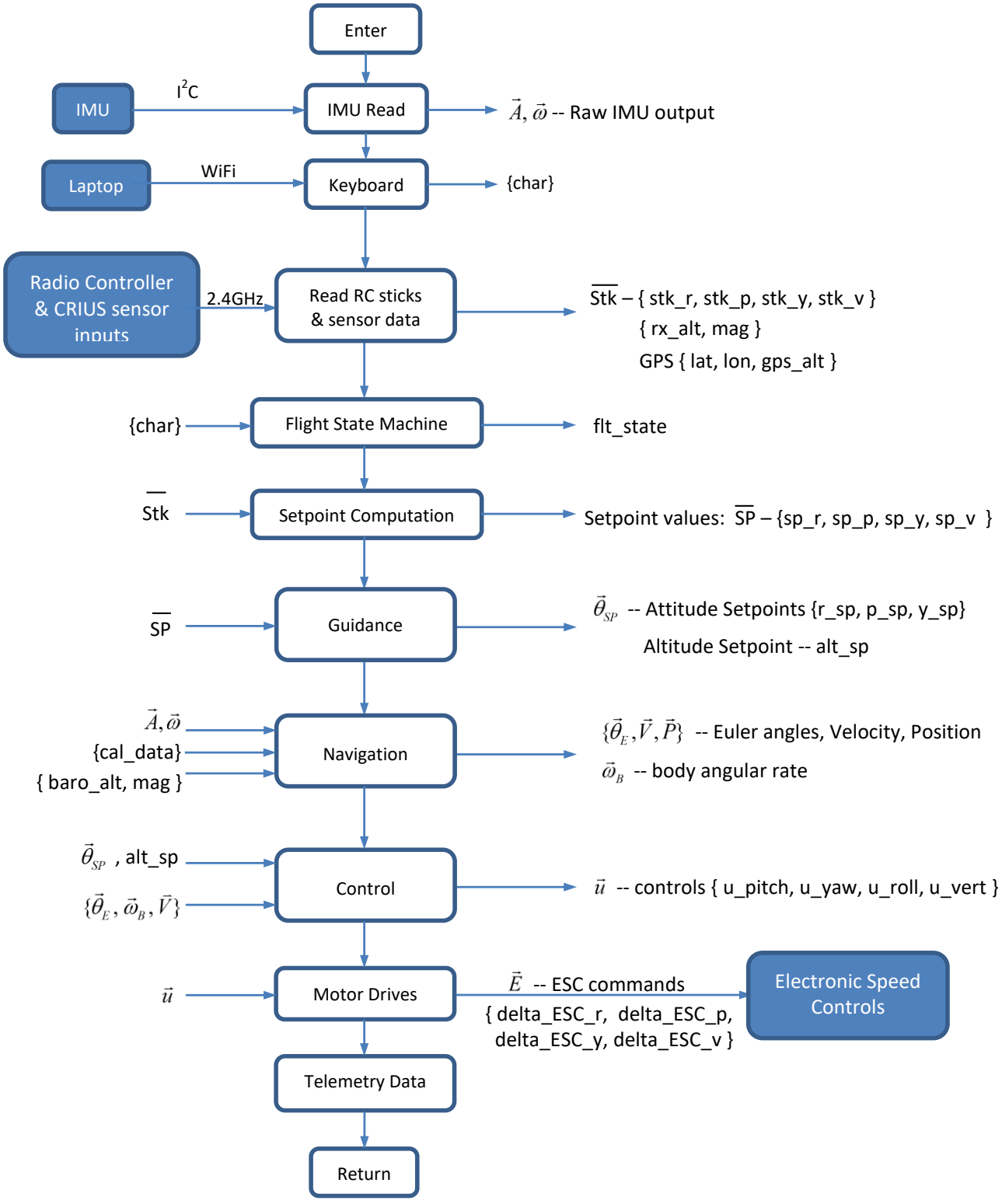

Figure 3-1 – Flight Control Software Flow Diagram

3.3 KEYBOARD INPUTS

Calling function: char = keyPressed()

The variable 'char' is a scalar character, an ascii. Keyboard inputs are passed from the Putty console through the WiFi link to Python running on the PiQuad. Input command definitions are as follows:

Character (char)	Associated command
'c'	Switch to Calibration Mode
0	Switch to Idle Mode
1	Switch to Flight Mode
2, 3, 4	Set control variable index command: cv = int(char) – 2
	cv = 0: Thrust and yaw rates set directly from commanded stick derived setpoints
	cv = 1: Guidance function produces altitude and heading setpoints
	cv = 2: unused
5	Stop writing telemetry to console
6	Write telemetry group 1 to console
7	Write telemetry group 2 to console
8	Write telemetry group 3 to console
's'	'Stop updating telemetry data and write content of telemetry data memory to file upon shutdown

3.4 READ RC STICKS AND SENSOR DATA

Calling function:

[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)

Table 3.4-1: rccmd() inputs

Variable Name	Description	Units
rc_port	Serial port index variable	-
serial_port_index	Serial port input flag for future modifications to produce alternative return data. Currently fixed to 'T'	-
alt_zero_data	Initial altitude measured with baro-altimeter at starting location	meters
prnt_flag	Flag to cause printing of RC raw input 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

3.5 FLIGHT STATE MACHINE

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)	

Table 3.5-2: statemachine() outputs

Symbol	Description	Units
flt_state	Same as above	
statics_rc	Same as above	
telem_flag	Same as above	
telm_idx	Same as above	-
cv	Same as above	-
RC_cal_data	Same as above	-
em_off	Same as above	-
idle_off	Same as above	-

Table3.5-3: IMU Rough Calibration data

Symbol	Description	Units
[b_{ax}, b_{ay}, b_{az}]	Accelerometer output biases	G
[SF_{ax}, SF_{ay}, SF_{az}]	Accelerometer output scale factors	-
[b_{wx}, b_{wy}, b_{wz}]	Gyro bias	deg/sec
[SF_{wx}, SF_{wy}, SF_{wz}]	Gyro output scale factors	-
[$alpha_x$, $alpha_y$, $alpha_z$]	IMU Coordinate frame misalignment angles relative to fixed body frame	rad

cal_data = [b_{ax}, b_{ay}, b_{az}, SF_{ax}, SF_{ay}, SF_{az}, b_{wx}, b_{wy}, b_{wz}, SF_{wx}, SF_{wy}, SF_{wz}, $alpha_x$, $alpha_y$, $alpha_z$]

Table3.5-3: Radio Controller Calibration data

Symbol	Description	Units
rc_zero_data[0..3]	RC nominally zero location (roll, pitch, yaw, vertical)	counts
rc_range_data[0..3]	RC range -- maximum output minus minimum output (roll, pitch, yaw, vertical)	counts

RC_cal_data = [rc_zero_data, rc_range_data]

3.6 SETPOINT

Table 3.6-1: setpoint() inputs

Variable Name	Description	Units
cv	Control variable	
stk_r, stk_p, stk_v, stk_y	Raw RC input stick positions read directly from RC (uncompensated)	counts
statics_sp	Setpoint static variables	
RC_cal_data		-

Table 3.6-2: setpoint() outputs

Symbol	Description	Units
sp_r, sp_p, sp_y, sp_v	Setpoint outputs (setpoint range: -1 to1)	na
statics_sp	Same as above	

3.7 GUIDANCE

Table 3.7-1: guidance() inputs

Variable Name	Description	Units
cv	Control variable	
sp_r, sp_p, sp_y, sp_v	Setpoint outputs (setpoint range: -1 to 1)	na
statics_guid	Guidance static variables	
statics_nav[2]	Body frame yaw angle (theta_yaw)	na
flt_state	Flight state	na

Table 3.7-2: guidance() outputs

Symbol	Description	Units
[thet_roll_sp, thet_pitch_sp, thet_yaw_sp]	Roll, pitch, and yaw angle setpoints	rad
omeg_yaw_sp	Yaw angular rate setpoint	rad/sec
vthrust_sp	Vertical thrust setpoint (for raw cv=0 control mode)	Counts
alt_sp	Altitude setpoint	m
statics_guid	Same as above	

3.8 NAVIGATION

Table 3.8-1: nav() inputs

Variable Name	Description	Units
[Ax, Ay, Az]	Accelerometer data from IMU transformed onto NED coordinate frame	G
[Gx, Gy, Gz]	Gyro rate data from IMU transformed onto NED coordinate frame	deg/sec
mpu6050	MPU6050 class instantiation identifier	
delta_time	Elasped time between this and prior call to nav()	seconds
statics_nav	Navigation statics	
cal_data	IMU cal data (defined above)	
rx_alt	Received baro-altitude relative to starting altitude (alt_zero_data)	meters
magx, magy, magz	Raw magnetometer readings in PiQuad Body frame	counts
heading_bias	Heading calibration data (raw heading angle reading when oriented with x axis pointing true north)	rad
statics_sensor	Sensor statics	
cv	Control variable	cv

Table 3.8-2: nav() outputs

Symbol	Description	Units
[omega_r, omega_p, omega_y]	Filtered body rates (roll, pitch, yaw)	rad/sec
[vx_kp1, vy_kp1, vz_kp1]	Filtered vehicle velocity along NED coordinates	m/sec
statics_nav	Navigation statics	
statics_sensor	Sensor statics	

3.9 CONTROL

Table 3.9-1: control() inputs

Variable Name	Description	Units
cv	Control variable	
[thet_pitch_sp, thet_roll_sp, omeg_yaw_sp, thet_yaw_sp, alt_sp]	Control setpoints (same definitions as above)	rad, rad/sec, m
[xr_kp1, xp_kp1, xy_kp1]	Filtered roll, pitch and yaw angles	rad
[omega_r, omega_p, omega_y]	Filtered roll, pitch and yaw angular rates	rad/sec
[vx_kp1, vy_kp1, vz_kp1]	Filtered vehicle velocity along NED coordinates	m/sec
pz_k	Filtered altitude (z location along NED coordinates)	m
statics_cntr	Control statics	

Table 3.9-2: control() outputs

Symbol	Description	Units
[u_pitch, u_yaw, u_roll]	Control torque signal about pitch, yaw, and roll axes	N-m
u_vert	Control force signal along body z-axis (nominally vertical direction)	N

3.10 STATIC VARIABLES

Static variables are created using arrays at the top level code and passing these into functions and returning them back to the top level code for the purpose of retaining their values.

3.3.1 Navigation function statics -- statics_nav[]

Static variables used in navigation function realization.

statics_nav = [ϕ_r , θ_p , ψ_y , V_x , V_y , V_z , A_x , A_y , A_z , P_x , P_y , P_z , b_{ax} , b_{ay} , b_{az} , $b_{\omega x}$, $b_{\omega y}$, $b_{\omega z}$, S_{ax} , S_{ay} , S_{ay}]

Name	Sym	Description	Units
Roll Euler angle	ϕ_r	Body frame roll angle	rad
Pitch Euler angle	θ_p	Body frame pitch angle	rad
Yaw Euler angle	ψ_y	Body frame yaw angle	rad
Velocity x	V_x	x-axis velocity in NED local level frame	m/s
Velocity y	V_y	y-axis velocity in NED local level frame	m/s
Velocity z	V_z	z-axis velocity in NED local level frame	m/s
Acceleration x	A_x	x-axis acceleration in NED local level frame	m/s^2
Acceleration y	A_y	y-axis acceleration in NED local level frame	m/s^2
Acceleration z	A_z	z-axis acceleration in NED local level frame	m/s^2
Position x	P_x	x-axis location in NED local level frame	m
Position y	P_y	y-axis location in NED local level frame	m
Position z	P_z	z-axis location in NED local level frame	m
Bias acceleration x	b_{ax}	x-axis acceleration in NED local level frame	m/s^2
Bias acceleration y	b_{ay}	y-axis acceleration in NED local level frame	m/s^2
Bias acceleration z	b_{az}	z-axis acceleration in NED local level frame	m/s^2
Bias angular rate x	$b_{\omega x}$	x-axis acceleration in NED local level frame	rad/s
Bias angular rate y	$b_{\omega y}$	y-axis acceleration in NED local level frame	rad/s
Bias angular rate z	$b_{\omega z}$	z-axis acceleration in NED local level frame	rad/s
Scalefactor accel x	S_{ax}	x-axis acceleration scalefactor	-
Scalefactor accel y	S_{ay}	y-axis acceleration in scalefactor	-
Scalefactor accel z	S_{ay}	z-axis acceleration in scalefactor	-

NED – North East Down coordinate frame

3.3.2 Control function statics -- statics_cntr[]

Static variables used in pitch and control law implementation.

```
statics_cntr = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Name	Sym	Description	Units
State 1 pitch	x_{1p}	Pitch controller state vector element 1 – angle	rad
State 2 pitch	x_{2p}	Pitch controller state vector element 2 – angular rate	rad/sec
State 3 pitch	x_{3p}	Pitch controller state vector element 3 – torque	N-m
State 4 pitch		spare	
State 5 pitch		spare	
State 1 roll	x_{1p}	Roll controller state vector element 1 – angle	rad
State 2 roll	x_{2p}	Roll controller state vector element 2 – angular rate	rad/sec
State 3 roll	x_{3p}	Roll controller state vector element 3 – torque	N-m
State 4 roll		spare	
State 5 roll		spare	

3.3.3 Setpoint function statics -- statics_sp[]

Static variables used in setpoint generation.

```
statics_sp = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Name	Sym	Description	Units
Roll cmd state 0	x_{R0}	Roll setpoint filter state 0	-
Roll cmd state 1	x_{R1}	Roll setpoint filter state 1	-
Roll cmd state 2	x_{R2}	Roll setpoint filter state 2	-
Roll cmd state 3	x_{R3}	Roll setpoint filter state 3	-
Pitch cmd state 0	x_{P0}	Pitch setpoint filter state 0	-
Pitch cmd state 1	x_{P1}	Pitch setpoint filter state 1	-
Pitch cmd state 2	x_{P2}	Pitch setpoint filter state 2	-
Pitch cmd state 3	x_{P3}	Pitch setpoint filter state 3	-
Yaw cmd state 0	x_{Y0}	Yaw setpoint filter state 0	-
Yaw cmd state 1	x_{Y1}	Yaw setpoint filter state 1	-
Yaw cmd state 2	x_{Y2}	Yaw setpoint filter state 2	-
Yaw cmd state 3	x_{Y3}	Yaw setpoint filter state 3	-

3.3.4 Guidance function statics -- statics_guid[]

Static variables used in sensor signal generation.

```
statics_guid = [0, 0]
```

Name	Sym	Description	Units
Yaw setpoint	θ_{SP_y}	Setpoint angle yaw (sp_y)	rad
Altitude setpoint	Alt_{SP}	Altitude setpoint (alt_sp)	m

3.3.5 Guidance function statics -- statics_rc[]

Static variables used in processing of RC command setpoint inputs in 'statemachine'.

```
statics_rc = [0, 0, 0]
```

Name	Sym	Description	Units
statics_rc[0]	-	-	counts
statics_rc[1]	-	-	counts
statics_rc[2]	-	-	counts

APPENDIX A: Acronyms

The following table provides definitions for terms relevant to this document.

Acronym	Definition
IO	**Input-Output**
IMU	**Inertial Measurement Unit**
RPAS	**Remotely Piloted Air System**
UAS	**Unmanned Air System**
UAV	**Unmanned Air Vehicle**
FAA	**Federal Aviation Administration**
RIF	**Robotics In Flight™**

APPENDIX B: Disclaimer

If you're in doubt of your skills as a UAV, Drone, multi-copter or RPAS pilot, or are unsure of your equipment or piloting skills, do not fly.

Multicopters are inherently dangerous and should not be used by anyone without professional training and experience. It is the responsibility of the user to decide if the equipment is suitable for your intended use.

You (purchaser/user) are responsible for following all federal, state and local laws which may regulate the use of Remotely Piloted Air Systems operated in your area.

Purchaser/user assumes all liability for damages to property and persons from the equipment.

Purchaser/user confirms that they will operate UAV's, Drones or RPAS in accordance with FAA rules

Purchaser/user assumes all liability for improper use of the equipment.

Purchaser/user is responsible for proper configuration and maintenance of the equipment.

Users of this document and/or of the PiQuad should have proper liability insurance covering their operation of the platform. RC aircraft are not covered under standard liability insurance. Please check with your insurer regarding the correct insurance coverage.

Robotics In Flight LLC shall not responsible for any warranty claim due to misuse, crashes, damage, incorrect setup/configuration, or other negligent behavior.

In no event shall Robotics In Flight, LLC be liable for special, indirect, incidental or consequential damages. Indemnity: users of the PiQuad multicopter or this document shall indemnify and hold harmless Robotics In Flight, LLC, its officers, directors, agents, representatives and employees from any and all claims, liabilities, damages, and expenses (including attorneys' fees actually incurred) on account of death or injury to any person or damage to any property arising from or in connection with any goods supplied. This indemnity shall apply without regard to whether the claim, damage, liability or expense is based on breach of contract, breach of warranty, negligence, strict liability, or other.

APPENDIX C: Licenses

Name	Description	Ver	Copyright holder	License
piquad_main.py	Top level module (initialization, static variables, main loop)	1	RIF, 2017	GPL v3
flt_st.py	Flight state machine	1	RIF, 2017	GPL v3
kb_cmd.py	Console interface	1	RIF, 2017	GPL v3
esc.py	Electronic speed control driver control	1	RIF, 2017	GPL v3
imu.py	Inertial measurement unit sensor	na	Jeff Rowberg, 2012	MIT
rif_gnc.py	Guidance, navigation, and control functions	1	RIF, 2017	GPL v3
rif_cmd.py	Sensor interface library module (RC, GPS, baro-alt, magnetometer)	1	RIF, 2017	RIF Proprietary
ATMega2560_4chRC_GPS_Alt_Mag_BaroFix3.ino	CRIUS All-in-One Pro sensor interface executable module	1	RIF, 2017	RIF Proprietary

RIF – Robotics In Flight LLC.™