

Hexacopter Server Software

Generated by Doxygen 1.8.9.1

Sat May 16 2015 21:14:14

Contents

1	Todo List	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	picopter::gpio Namespace Reference	11
6.1.1	Detailed Description	11
6.1.2	Function Documentation	11
6.1.2.1	Init	11
6.1.2.2	IsAutoMode	11
6.1.2.3	SetBuzzer	11
6.1.3	Variable Documentation	12
6.1.3.1	BUZZER_PIN	12
6.1.3.2	MODE_PIN	12
7	Class Documentation	13
7.1	picopter::Buzzer Class Reference	13
7.1.1	Constructor & Destructor Documentation	13
7.1.1.1	Buzzer	13
7.1.1.2	~Buzzer	13
7.1.2	Member Function Documentation	13
7.1.2.1	Play	13
7.1.2.2	PlayWait	13
7.1.2.3	Stop	14

7.2	BuzzerTest Class Reference	14
7.3	CamComponent Struct Reference	14
7.4	picopter::CameraStream Class Reference	14
7.4.1	Member Function Documentation	15
7.4.1.1	SetArrow	15
7.5	picopter::CamWindow Struct Reference	15
7.6	picopter::navigation::Coord2D Struct Reference	16
7.6.1	Detailed Description	16
7.6.2	Member Data Documentation	16
7.6.2.1	lat	16
7.6.2.2	lon	16
7.7	picopter::navigation::Coord3D Struct Reference	16
7.7.1	Detailed Description	16
7.7.2	Member Function Documentation	16
7.7.2.1	operator Coord2D	16
7.7.3	Member Data Documentation	17
7.7.3.1	alt	17
7.7.3.2	lat	17
7.7.3.3	lon	17
7.8	picopter::DataLog Class Reference	17
7.8.1	Detailed Description	17
7.8.2	Constructor & Destructor Documentation	17
7.8.2.1	DataLog	17
7.8.2.2	~DataLog	17
7.8.3	Member Function Documentation	18
7.8.3.1	PlainWrite	18
7.8.3.2	Write	18
7.8.3.3	Write	18
7.9	picopter::navigation::EulerAngle Struct Reference	18
7.9.1	Detailed Description	18
7.9.2	Member Data Documentation	18
7.9.2.1	pitch	18
7.9.2.2	roll	19
7.9.2.3	yaw	19
7.10	gps_data_t::fix Struct Reference	19
7.11	picopter::FlightBoard Class Reference	19
7.11.1	Detailed Description	19
7.11.2	Constructor & Destructor Documentation	20
7.11.2.1	FlightBoard	20
7.11.2.2	FlightBoard	20

7.11.2.3	~FlightBoard	20
7.11.3	Member Function Documentation	20
7.11.3.1	GetData	20
7.11.3.2	SetAileron	20
7.11.3.3	SetData	20
7.11.3.4	SetElevator	20
7.11.3.5	SetGimbal	21
7.11.3.6	SetRudder	21
7.11.3.7	Stop	21
7.12	picopter::FlightController Class Reference	21
7.12.1	Detailed Description	22
7.12.2	Constructor & Destructor Documentation	22
7.12.2.1	FlightController	22
7.12.2.2	FlightController	22
7.12.2.3	~FlightController	22
7.12.3	Member Function Documentation	22
7.12.3.1	CheckForStop	22
7.12.3.2	GetCurrentState	22
7.12.3.3	GetCurrentTaskId	22
7.12.3.4	InferBearing	23
7.12.3.5	RunTask	24
7.12.3.6	Sleep	24
7.12.3.7	Stop	24
7.12.3.8	WaitForAuth	24
7.12.4	Friends And Related Function Documentation	24
7.12.4.1	operator<<	24
7.12.5	Member Data Documentation	25
7.12.5.1	buzzer	25
7.12.5.2	cam	25
7.12.5.3	fb	25
7.12.5.4	gps	25
7.12.5.5	imu	25
7.13	picopter::FlightData Struct Reference	25
7.13.1	Detailed Description	25
7.13.2	Member Data Documentation	25
7.13.2.1	aileron	25
7.13.2.2	elevator	25
7.13.2.3	gimbal	26
7.13.2.4	rudder	26
7.14	picopter::FlightTask Class Reference	26

7.14.1 Detailed Description	26
7.14.2 Constructor & Destructor Documentation	26
7.14.2.1 ~FlightTask	26
7.14.3 Member Function Documentation	26
7.14.3.1 Run	26
7.14.3.2 SetCurrentState	27
7.15 picopter::GPS Class Reference	27
7.15.1 Detailed Description	28
7.15.2 Constructor & Destructor Documentation	28
7.15.2.1 GPS	28
7.15.2.2 GPS	28
7.15.2.3 ~GPS	28
7.15.3 Member Function Documentation	28
7.15.3.1 GetLatest	28
7.15.3.2 HasFix	28
7.15.3.3 TimeSinceLastFix	28
7.15.3.4 WaitForFix	28
7.15.4 Member Data Documentation	29
7.15.4.1 FIX_TIMEOUT_DEFAULT	29
7.15.4.2 WAIT_PERIOD	29
7.16 gps_data_t Struct Reference	29
7.17 picopter::GPSData Struct Reference	29
7.17.1 Detailed Description	30
7.17.2 Member Data Documentation	30
7.17.2.1 err	30
7.17.2.2 fix	30
7.17.2.3 timestamp	30
7.18 picopter::GPSFix Struct Reference	30
7.18.1 Member Data Documentation	30
7.18.1.1 alt	30
7.18.1.2 bearing	30
7.18.1.3 heading	30
7.18.1.4 lat	31
7.18.1.5 lon	31
7.18.1.6 speed	31
7.19 picopter::GPSGPSD Class Reference	31
7.19.1 Detailed Description	31
7.19.2 Constructor & Destructor Documentation	31
7.19.2.1 GPSGPSD	31
7.19.2.2 GPSGPSD	31

7.19.2.3	~GPSGPSD	32
7.20	gpsmm Class Reference	32
7.20.1	Member Data Documentation	32
7.20.1.1	LATLON_SET	32
7.21	picopter::GPSNaza Class Reference	32
7.21.1	Detailed Description	33
7.21.2	Constructor & Destructor Documentation	33
7.21.2.1	GPSNaza	33
7.21.2.2	GPSNaza	33
7.21.2.3	~GPSNaza	33
7.22	picopter::IMU Class Reference	33
7.22.1	Detailed Description	33
7.22.2	Constructor & Destructor Documentation	33
7.22.2.1	IMU	33
7.22.2.2	IMU	34
7.22.2.3	~IMU	34
7.22.3	Member Function Documentation	34
7.22.3.1	GetLatest	34
7.23	picopter::Lawnmower Class Reference	34
7.23.1	Detailed Description	34
7.24	NavigationTest Class Reference	34
7.25	NazaDecoderLib Class Reference	35
7.26	picopter::ObjectTracker Class Reference	36
7.26.1	Detailed Description	36
7.26.2	Member Function Documentation	36
7.26.2.1	Run	36
7.27	picopter::Options Class Reference	36
7.27.1	Detailed Description	37
7.27.2	Constructor & Destructor Documentation	37
7.27.2.1	Options	37
7.27.2.2	Options	37
7.27.2.3	~Options	37
7.27.3	Member Function Documentation	37
7.27.3.1	GetBool	37
7.27.3.2	GetInt	38
7.27.3.3	GetReal	38
7.27.3.4	GetString	38
7.27.3.5	Remove	38
7.27.3.6	Save	39
7.27.3.7	Save	39

7.27.3.8	Save	39
7.27.3.9	Set	39
7.27.3.10	Set	39
7.27.3.11	Set	39
7.27.3.12	Set	39
7.27.3.13	SetFamily	40
7.28	OptionsTest Class Reference	40
7.29	picopter::PID Class Reference	40
7.29.1	Detailed Description	41
7.29.2	Constructor & Destructor Documentation	41
7.29.2.1	PID	41
7.29.3	Member Function Documentation	41
7.29.3.1	Compute	41
7.29.3.2	Reset	41
7.29.3.3	SetBias	41
7.29.3.4	SetInputLimits	41
7.29.3.5	SetInterval	42
7.29.3.6	SetMode	42
7.29.3.7	SetOutputLimits	42
7.29.3.8	SetProcessValue	42
7.29.3.9	SetSetPoint	42
7.29.3.10	SetTunings	42
7.30	picopter::navigation::Point2D Struct Reference	43
7.30.1	Detailed Description	43
7.30.2	Member Function Documentation	43
7.30.2.1	magnitude	43
7.30.3	Member Data Documentation	43
7.30.3.1	x	43
7.30.3.2	y	43
7.31	picopter::navigation::Point3D Struct Reference	43
7.31.1	Detailed Description	44
7.31.2	Member Function Documentation	44
7.31.2.1	magnitude	44
7.31.2.2	operator Point2D	44
7.31.3	Member Data Documentation	44
7.31.3.1	x	44
7.31.3.2	y	44
7.31.3.3	z	44
7.32	vec2 Struct Reference	44
7.33	picopter::Waypoints Class Reference	45

7.33.1 Detailed Description	45
7.33.2 Member Function Documentation	45
7.33.2.1 Run	45
7.34 webInterfaceHandler Class Reference	45
8 File Documentation	47
8.1 emulation/gps-emu/libgpsmm.h File Reference	47
8.1.1 Detailed Description	48
8.2 emulation/wiringPi.h File Reference	48
8.2.1 Detailed Description	48
8.3 emulation/wiringSerial.h File Reference	48
8.3.1 Detailed Description	49
8.4 include/buzzer.h File Reference	49
8.4.1 Detailed Description	49
8.5 include/camera_stream.h File Reference	49
8.5.1 Detailed Description	49
8.6 include/common.h File Reference	50
8.6.1 Detailed Description	50
8.7 include/config.h File Reference	50
8.7.1 Detailed Description	50
8.8 include/datalog.h File Reference	51
8.8.1 Detailed Description	51
8.9 include/flightboard.h File Reference	51
8.9.1 Detailed Description	51
8.10 include/flightcontroller.h File Reference	51
8.10.1 Detailed Description	52
8.11 include/gpio.h File Reference	52
8.11.1 Detailed Description	52
8.12 include/gps_feed.h File Reference	52
8.12.1 Detailed Description	53
8.13 include/gps_gpsd.h File Reference	53
8.13.1 Detailed Description	53
8.14 include/gps_naza.h File Reference	53
8.14.1 Detailed Description	53
8.15 include/imu_feed.h File Reference	54
8.15.1 Detailed Description	54
8.16 include/lawnmower.h File Reference	54
8.16.1 Detailed Description	54
8.17 include/log.h File Reference	54
8.17.1 Detailed Description	55

8.17.2	Function Documentation	55
8.17.2.1	FatalEx	55
8.17.2.2	LogEx	55
8.17.2.3	LogInit	55
8.17.2.4	LogSimple	55
8.18	include/navigation.h File Reference	56
8.18.1	Detailed Description	57
8.18.2	Macro Definition Documentation	57
8.18.2.1	RADIUS_OF_EARTH	57
8.19	include/object_tracker.h File Reference	57
8.19.1	Detailed Description	57
8.20	include/opts.h File Reference	57
8.20.1	Detailed Description	57
8.21	include/picopter.h File Reference	57
8.21.1	Detailed Description	58
8.22	include/waypoints.h File Reference	58
8.22.1	Detailed Description	58
8.23	src/apps/naza_decoder.cpp File Reference	58
8.23.1	Detailed Description	59
8.23.2	Function Documentation	59
8.23.2.1	decodeMessage	59
8.23.2.2	main	59
8.24	src/apps/speedcal.cpp File Reference	59
8.24.1	Detailed Description	59
8.25	src/base/buzzer.cpp File Reference	59
8.25.1	Detailed Description	60
8.26	src/base/camera_stream.cpp File Reference	60
8.26.1	Detailed Description	60
8.27	src/base/datalog.cpp File Reference	60
8.27.1	Detailed Description	61
8.28	src/base/flightboard-private.h File Reference	61
8.28.1	Detailed Description	61
8.28.2	Macro Definition Documentation	61
8.28.2.1	AILERON_CHANNEL	61
8.28.2.2	AILERON_HIGH	62
8.28.2.3	AILERON_LOW	62
8.28.2.4	AILERON_PIN_PHYSICAL	62
8.28.2.5	AILERON_SCALE	62
8.28.2.6	ELEVATOR_CHANNEL	62
8.28.2.7	ELEVATOR_HIGH	62

8.28.2.8	ELEVATOR_LOW	62
8.28.2.9	ELEVATOR_PIN_PHYSICAL	62
8.28.2.10	ELEVATOR_SCALE	62
8.28.2.11	GIMBAL_CHANNEL	62
8.28.2.12	GIMBAL_HIGH	62
8.28.2.13	GIMBAL_LOW	62
8.28.2.14	GIMBAL_PIN_PHYSICAL	63
8.28.2.15	GIMBAL_SCALE	63
8.28.2.16	LINEAR_SCALE	63
8.28.2.17	RUDDER_CHANNEL	63
8.28.2.18	RUDDER_HIGH	63
8.28.2.19	RUDDER_LOW	63
8.28.2.20	RUDDER_PIN_PHYSICAL	63
8.28.2.21	RUDDER_SCALE	63
8.28.2.22	SPEED_SCALE	63
8.29	src/base/flightboard.cpp File Reference	63
8.29.1	Detailed Description	64
8.30	src/base/flightcontroller.cpp File Reference	64
8.30.1	Detailed Description	64
8.30.2	Function Documentation	64
8.30.2.1	InitialiseItem	64
8.31	src/base/gpio.cpp File Reference	65
8.31.1	Detailed Description	65
8.32	src/base/gps.cpp File Reference	65
8.32.1	Detailed Description	65
8.33	src/base/gps_gpsd.cpp File Reference	65
8.33.1	Detailed Description	66
8.34	src/base/gps_naza.cpp File Reference	66
8.34.1	Detailed Description	66
8.35	src/base/imu.cpp File Reference	66
8.35.1	Detailed Description	66
8.36	src/base/log.cpp File Reference	66
8.36.1	Detailed Description	67
8.36.2	Function Documentation	67
8.36.2.1	FatalEx	67
8.36.2.2	LogEx	67
8.36.2.3	LogInit	67
8.36.2.4	LogSimple	67
8.37	src/base/opts.cpp File Reference	68
8.37.1	Detailed Description	68

8.38	src/modules/object_tracker.cpp File Reference	68
8.38.1	Detailed Description	68
8.39	src/modules/waypoints.cpp File Reference	68
8.39.1	Detailed Description	68
8.40	src/server/picopter.cpp File Reference	69
8.40.1	Detailed Description	69
8.40.2	Function Documentation	69
8.40.2.1	terminate	69
Index		71

Chapter 1

Todo List

File [flightcontroller.cpp](#)

Add in camera things.

Class [picopter::GPSData](#)

Do we need the uncertainty in the timestamp too? 2D info enough?

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

picopter::gpio	11
--	----

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

picopter::Buzzer	13
CamComponent	14
picopter::CameraStream	14
picopter::CamWindow	15
picopter::navigation::Coord2D	16
picopter::navigation::Coord3D	16
picopter::DataLog	17
picopter::navigation::EulerAngle	18
gps_data_t::fix	19
picopter::FlightBoard	19
picopter::FlightController	21
picopter::FlightData	25
picopter::FlightTask	26
picopter::ObjectTracker	36
picopter::Waypoints	45
picopter::GPS	27
picopter::GPSGPSD	31
picopter::GPSNaza	32
gps_data_t	29
picopter::GPSData	29
picopter::GPSFix	30
gpsmm	32
picopter::IMU	33
picopter::Lawnmower	34
NazaDecoderLib	35
picopter::Options	36
picopter::PID	40
picopter::navigation::Point2D	43
picopter::navigation::Point3D	43
Test	
BuzzerTest	14
NavigationTest	34
OptionsTest	40
vec2	44
webInterfaceIf	
webInterfaceHandler	45

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

picopter::Buzzer	13
BuzzerTest	14
CamComponent	14
picopter::CameraStream	14
picopter::CamWindow	15
picopter::navigation::Coord2D	16
picopter::navigation::Coord3D	16
picopter::DataLog	17
picopter::navigation::EulerAngle	18
gps_data_t::fix	19
picopter::FlightBoard	19
picopter::FlightController	21
picopter::FlightData	25
picopter::FlightTask	26
picopter::GPS	27
gps_data_t	29
picopter::GPSData	29
picopter::GPSFix	30
picopter::GPSGPSD	31
gpsmm	32
picopter::GPSNaza	32
picopter::IMU	33
picopter::Lawnmower	34
NavigationTest	34
NazaDecoderLib	35
picopter::ObjectTracker	36
picopter::Options	36
OptionsTest	40
picopter::PID	40
picopter::navigation::Point2D	43
picopter::navigation::Point3D	43
vec2	44
picopter::Waypoints	45
webInterfaceHandler	45

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

emulation/ wiringPi.h	48
Simple stub file to get 'wiringPi' when not compiling on the RPi	
emulation/ wiringSerial.h	48
Simple stub file to get 'wiringSerial' when not compiling on the RPi	
emulation/gps-emu/ libgpsmm.h	47
A <i>very thin</i> emulation of libgpsmm. Will always return that there is data, but with all values zeroed. Data will randomly be marked as set or unset	
include/ buzzer.h	49
Methods to control the buzzer	
include/ camera_stream.h	49
include/ common.h	50
Commonly included headers	
include/ config.h	50
System-specific and compile-time specific configuration parameters	
include/ datalog.h	51
Declaration of functions for data logging	
include/ flightboard.h	51
Defines the FlightBoard class	
include/ flightcontroller.h	51
Defines the FlightController class	
include/ gpio.h	52
Gpio class defines	
include/ gps_feed.h	52
Base GPS header	
include/ gps_gpsd.h	53
GPS header for the gpsd implementation	
include/ gps_naza.h	53
GPS header for the NAZA implementation	
include/ imu_feed.h	54
Defines the IMU class	
include/ lawnmower.h	54
Defines the lawnmower controls	
include/ log.h	54
Declaration of functions for printing log messages and/or terminating program after a fatal error	
include/ navigation.h	56
General navigation code	
include/ NazaDecoderLib.h	??

include/object_tracker.h	
The header file for the object tracking code	57
include/opts.h	
Header for handling options and persistent configurations	57
include/picopter.h	
The main include file	57
include/PID.h	??
include/waypoints.h	
The header file for the waypoints code	58
src/apps/naza_decoder.cpp	
Sample application to decode and print out NAZA gps data	58
src/apps/speedcal.cpp	
Application to determine relation between input and speed	59
src/base/buzzer.cpp	
Buzzer manipulation code. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM	59
src/base/camera_stream.cpp	60
src/base/datalog.cpp	
Implement data logging functions	60
src/base/flightboard-private.h	
Contains calibration data and pin mappings for the FlightBoard class. Note: The unit 'step' is ServoBlaster specific. Usually 1 step is 10us	61
src/base/flightboard.cpp	
Controls the output to the flight board. Calculates the correct PWM pulse width to be sent to the PWM drivers for the aileron, elevator and rudder, as well as camera gimbal. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM	63
src/base/flightcontroller.cpp	
Base controller. Ties in all the equipment (sensors and actuators)	64
src/base/gpio.cpp	
GPIO handling code. Uses wiringPi to control the GPIO pins on the RPi	65
src/base/gps.cpp	
Base GPS interaction code	65
src/base/gps_gpsd.cpp	
GPS interaction code. Uses gpsd to interact with the GPS	65
src/base/gps_naza.cpp	
GPS interaction code. Uses the NAZA decoder to interact with the NAZA GPS	66
src/base/imu.cpp	
IMU interaction code	66
src/base/log.cpp	
Implement logging and error handling functions	66
src/base/opts.cpp	
Options and persistent configurations handler	68
src/modules/object_tracker.cpp	
The object tracking code	68
src/modules/waypoints.cpp	
Contains the waypoints handling code	68
src/server/picopter.cpp	
The main entry point to the server	69

Chapter 6

Namespace Documentation

6.1 picopter::gpio Namespace Reference

Functions

- void `Init` ()
- bool `IsAutoMode` ()
- void `SetBuzzer` (bool value)

Variables

- const int `MODE_PIN` = 5
- const int `BUZZER_PIN` = 2

6.1.1 Detailed Description

Controls the GPIO pins on the RPi, including PWM functionality.

6.1.2 Function Documentation

6.1.2.1 void picopter::gpio::Init ()

Initialises the GPIO pins, if necessary.

6.1.2.2 bool picopter::gpio::IsAutoMode ()

Determines if autonomous mode has been enabled by the user. Assumes that `gpio::init` has already been called. Probably not thread-safe.

Returns

true iff the user has enabled the switch for autonomous mode

6.1.2.3 void picopter::gpio::SetBuzzer (bool value)

Turns the buzzer on or off. Assumes that `gpio::init` has already been called. Probably not thread-safe.

Parameters

<i>value</i>	Indicates whether the buzzer should be on (true) or off (false).
--------------	--

6.1.3 Variable Documentation

6.1.3.1 `const int picopter::gpio::BUZZER_PIN = 2`

The GPIO pin of the buzzer (wiringPi numbering)

6.1.3.2 `const int picopter::gpio::MODE_PIN = 5`

The GPIO pin of the switch for auto mode (wiringPi numbering)

Chapter 7

Class Documentation

7.1 picopter::Buzzer Class Reference

Public Member Functions

- [Buzzer](#) ()
- virtual [~Buzzer](#) ()
- void [Play](#) (int duration, int frequency, int volume)
- void [PlayWait](#) (int duration, int frequency, int volume)
- void [Stop](#) ()

7.1.1 Constructor & Destructor Documentation

7.1.1.1 [Buzzer::Buzzer](#) ()

Creates a new [Buzzer](#) instance. Starts a worker thread that will play the sounds

7.1.1.2 [Buzzer::~~Buzzer](#) () [virtual]

Destructor method. Stops the worker thread.

7.1.2 Member Function Documentation

7.1.2.1 void [Buzzer::Play](#) (int *duration*, int *frequency*, int *volume*)

Plays a tone, non-blocking. If a tone is already being played, then that is stopped and the new tone is played instead.

Parameters

<i>duration</i>	The length of time to play the sound, in ms
<i>frequency</i>	The frequency of the sound, in Hz (10-5000Hz).
<i>volume</i>	The loudness of the sound, as a percentage (0 - 100%)

7.1.2.2 void [Buzzer::PlayWait](#) (int *duration*, int *frequency*, int *volume*)

Plays a tone, blocking until it is complete. If a tone is already being played, then that is stopped and the new tone is played instead.

Parameters

<i>duration</i>	The length of time to play the sound, in ms
<i>frequency</i>	The frequency of the sound, in Hz (10-5000Hz).
<i>volume</i>	The loudness of the sound, as a percentage (0 - 100%)

7.1.2.3 void Buzzer::Stop ()

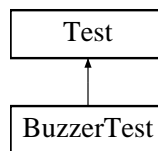
Signals the buzzer to stop if it is running

The documentation for this class was generated from the following files:

- [include/buzzer.h](#)
- [src/base/buzzer.cpp](#)

7.2 BuzzerTest Class Reference

Inheritance diagram for BuzzerTest:

**Protected Attributes**

- [Buzzer b](#)

The documentation for this class was generated from the following file:

- [test/test_buzzer.cpp](#)

7.3 CamComponent Struct Reference**Public Attributes**

- int **M00**
- int **M01**
- int **M10**

The documentation for this struct was generated from the following file:

- [src/base/camera_stream.cpp](#)

7.4 picopter::CameraStream Class Reference

Public Types

- enum **CameraMode** {
MODE_NO_PROCESSING = 0, **MODE_COM** = 1, **MODE_CAMSHIFT** = 2, **MODE_CONNECTED_COM** ←
PONENTS = 3,
MODE_LEARN_COLOUR = 999 }

Public Member Functions

- **CameraStream** ([Options](#) *opts)
- bool **Start** (void)
- void **Stop** (void)
- int **GetInputWidth** ()
- int **GetInputHeight** ()
- CameraMode **GetMode** (void)
- void **SetMode** (CameraMode mode)
- void **SetLearningSize** (bool decrease)
- void **ShowLearningThreshold** (bool show)
- int **GetLearningHue** ()
- void **DoAutoLearning** (std::map< std::string, int32_t > *ret)
- void **DoManualLearning** (const std::map< std::string, int32_t > &values, std::map< std::string, int32_t > *ret)
- void **GetDetectedObjects** (std::vector< [navigation::Point2D](#) > *)
- void **SetArrow** ([navigation::Point2D](#) vec)
- double **GetFramerate** (void)
- void **TakePhoto** (std::string)

7.4.1 Member Function Documentation

7.4.1.1 void CameraStream::SetArrow ([navigation::Point2D](#) vec)

Set an arrow to be displayed from the centre of the image.

Parameters

vec	The arrow vector, as an integer percentage. E.g. 100% vec.x draws an arrow from centre to the right side of the image.
-----	--

The documentation for this class was generated from the following files:

- include/[camera_stream.h](#)
- src/base/[camera_stream.cpp](#)

7.5 picopter::CamWindow Struct Reference

Public Attributes

- int **x**
- int **y**
- int **l**
- int **w**

The documentation for this struct was generated from the following file:

- include/[camera_stream.h](#)

7.6 `picopter::navigation::Coord2D` Struct Reference

```
#include <navigation.h>
```

Public Attributes

- double [lat](#)
- double [lon](#)

7.6.1 Detailed Description

Holds 2-dimensional geographical coordinate position information.

7.6.2 Member Data Documentation

7.6.2.1 `double picopter::navigation::Coord2D::lat`

Latitude

7.6.2.2 `double picopter::navigation::Coord2D::lon`

Longitud

The documentation for this struct was generated from the following file:

- include/[navigation.h](#)

7.7 `picopter::navigation::Coord3D` Struct Reference

```
#include <navigation.h>
```

Public Member Functions

- [operator Coord2D](#) ()

Public Attributes

- double [lat](#)
- double [lon](#)
- double [alt](#)

7.7.1 Detailed Description

Holds 3-dimensional geographical coordinate position information.

7.7.2 Member Function Documentation

7.7.2.1 `picopter::navigation::Coord3D::operator Coord2D ()` [`inline`]

Implicit conversion to 2D coordinate

7.7.3 Member Data Documentation

7.7.3.1 double picopter::navigation::Coord3D::alt

Altitude

7.7.3.2 double picopter::navigation::Coord3D::lat

Latitude

7.7.3.3 double picopter::navigation::Coord3D::lon

Longitude

The documentation for this struct was generated from the following file:

- include/[navigation.h](#)

7.8 picopter::DataLog Class Reference

```
#include <datalog.h>
```

Public Member Functions

- [DataLog](#) (const char *name, bool log_startstop=true, const char *location=PICOPTER_LOG_LOCATION)
- virtual [~DataLog](#) ()
- void [Write](#) (size_t sz, const char *buf)
- void [Write](#) (const char *fmt,...)
- void [PlainWrite](#) (const char *fmt,...)

7.8.1 Detailed Description

Class to log data in a flexible manner.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 DataLog::DataLog (const char * file, bool log_startstop = true, const char * location = PICOPTER_LOG_LOCATION)

Creates a log file for logging *data*. The filename will be of the form 'file-*timestamp*.txt'. The *timestamp* is that from FileTimestamp. If the file exists, it will be overwritten.

Parameters

<i>file</i>	The name of the file, excluding any extension.
<i>log_startstop</i>	Whether or not to log the start and stop times of the log. Defaults to true.
<i>location</i>	The folder where the file should be stored. Defaults to PICOPTER_LOG_LOCATION, which is set in config.h . This should be the home folder of the user.

7.8.2.2 DataLog::~DataLog () [virtual]

Destructor. Closes the file pointer.

7.8.3 Member Function Documentation

7.8.3.1 void DataLog::PlainWrite (const char * *fmt*, ...)

Writes a line of text to the file (no timestamp).

Parameters

<i>fmt</i>	A format string.
...	Arguments to be printed according to the format string.

7.8.3.2 void DataLog::Write (size_t *sz*, const char * *buf*)

Writes data to the log file, verbatim.

Parameters

<i>sz</i>	The size of the buffer to be written.
<i>buf</i>	The pointer to the buffer.

7.8.3.3 void DataLog::Write (const char * *fmt*, ...)

Writes a line of text to the file, prepending a timestamp.

Parameters

<i>fmt</i>	A format string.
...	Arguments to be printed according to the format string.

The documentation for this class was generated from the following files:

- include/[datalog.h](#)
- src/base/[datalog.cpp](#)

7.9 picopter::navigation::EulerAngle Struct Reference

```
#include <navigation.h>
```

Public Attributes

- double [roll](#)
- double [pitch](#)
- double [yaw](#)

7.9.1 Detailed Description

Holds a set of Euler angles.

7.9.2 Member Data Documentation

7.9.2.1 double picopter::navigation::EulerAngle::pitch

Pitch of the [IMU](#), -pi to pi

7.9.2.2 double picopter::navigation::EulerAngle::roll

Roll of the [IMU](#), -pi to pi

7.9.2.3 double picopter::navigation::EulerAngle::yaw

Yaw of the [IMU](#), -pi to pi

The documentation for this struct was generated from the following file:

- include/[navigation.h](#)

7.10 gps_data_t::fix Struct Reference

Public Attributes

- double **time**
- double **latitude**
- double **epy**
- double **longitude**
- double **epx**
- double **speed**
- double **eps**
- double **track**
- double **epd**

The documentation for this struct was generated from the following file:

- emulation/gps-emu/[libgpsmm.h](#)

7.11 picopter::FlightBoard Class Reference

```
#include <flightboard.h>
```

Public Member Functions

- [FlightBoard](#) ()
- [FlightBoard](#) ([Options](#) *opts)
- virtual [~FlightBoard](#) ()
- void [Stop](#) ()
- void [GetData](#) ([FlightData](#) *d)
- void [SetData](#) ([FlightData](#) *d)
- void [SetAileron](#) (int speed)
- void [SetElevator](#) (int speed)
- void [SetRudder](#) (int speed)
- void [SetGimbal](#) (int pos)

7.11.1 Detailed Description

Controls the actuation of the hexacopter. This class is not thread-safe.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 `FlightBoard::FlightBoard ()`

Constructor. Constructs a new flight board with default settings.

7.11.2.2 `FlightBoard::FlightBoard (Options * opts)`

Constructor; initiates a connection to ServoBlaster. Assumes that ServoBlaster has already been started and initialised.

Parameters

<i>opts</i>	A pointer to options, if any (NULL for defaults)
-------------	--

Exceptions

<code>std::invalid_argument</code>	if it can't connect to ServoBlaster.
------------------------------------	--------------------------------------

7.11.2.3 `FlightBoard::~~FlightBoard ()` [virtual]

Destructor. Closes the connection to ServoBlaster.

7.11.3 Member Function Documentation

7.11.3.1 `void FlightBoard::GetData (FlightData * d)`

Returns a copy of the current flight data.

Parameters

<i>d</i>	A pointer to the output location.
----------	-----------------------------------

7.11.3.2 `void FlightBoard::SetAileron (int speed)`

Sets the aileron speed.

Parameters

<i>speed</i>	The aileron speed, as a percentage (-100% to 100%)
--------------	--

7.11.3.3 `void FlightBoard::SetData (FlightData * d)`

Sets the flight data and actuates the hexacopter.

Parameters

<i>d</i>	Specifies the flight data for how the hexacopter should be actuated.
----------	--

7.11.3.4 `void FlightBoard::SetElevator (int speed)`

Sets the elevator speed.

Parameters

<i>speed</i>	The elevator speed, as a percentage (-100% to 100%)
--------------	---

7.11.3.5 void FlightBoard::SetGimbal (int *pos*)

Sets the gimbal angle.

Parameters

<i>pos</i>	The gimbal angle, in degrees (0 to 90)
------------	--

7.11.3.6 void FlightBoard::SetRudder (int *speed*)

Sets the rudder speed.

Parameters

<i>speed</i>	The rudder speed, as a percentage (-100% to 100%)
--------------	---

7.11.3.7 void FlightBoard::Stop (void)

Stops the hexacopter. Note: Stopping refers to making it hold its current position. Sets all speeds and the gimbal angle to 0.

The documentation for this class was generated from the following files:

- include/[flightboard.h](#)
- src/base/[flightboard.cpp](#)

7.12 picopter::FlightController Class Reference

```
#include <flightcontroller.h>
```

Public Member Functions

- [FlightController](#) ()
- [FlightController](#) ([Options](#) *opts)
- virtual ~[FlightController](#) ()
- ControllerState [GetCurrentState](#) ()
- TaskIdentifier [GetCurrentTaskId](#) ()
- void [Stop](#) ()
- bool [CheckForStop](#) ()
- bool [Sleep](#) (int ms)
- bool [WaitForAuth](#) ()
- bool [RunTask](#) (TaskIdentifier tid, [FlightTask](#) *task, void *opts)
- bool [InferBearing](#) (double *ret, int move_time=5000)

Public Attributes

- [FlightBoard](#) *const & [fb](#)
- [IMU](#) *const & [imu](#)

- [GPS](#) *const & [gps](#)
- [Buzzer](#) *const & [buzzer](#)
- [CameraStream](#) *const & [cam](#)

Friends

- `std::ostream & operator<< (std::ostream &stream, FlightController &fc)`

7.12.1 Detailed Description

The base controller for the hexacopter. It ties in all the actuators and sensors for access from a central point.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 `FlightController::FlightController ()`

Constructor. Constructs a new flight controller with default settings.

7.12.2.2 `FlightController::FlightController (Options * opts)`

The flight controller constructor. Initialises all members as necessary.

Exceptions

<code>std::invalid_argument</code>	if a required component fails to initialise.
------------------------------------	--

7.12.2.3 `FlightController::~~FlightController ()` [`virtual`]

Destructor. Stops/cleans up the base components (depends on RAI)

7.12.3 Member Function Documentation

7.12.3.1 `bool FlightController::CheckForStop ()`

Check whether a stop should occur or not.

7.12.3.2 `ControllerState FlightController::GetCurrentState ()`

Retrieves the current state of the flight controller.

Returns

The current state.

7.12.3.3 `TaskIdentifier FlightController::GetCurrentTaskId ()`

Retrieves the current ID of the task being run.

Returns

The current task ID.

7.12.3.4 `bool FlightController::InferBearing (double * ret, int move_time = 5000)`

Infer the current bearing by moving forwards and using the [GPS](#) heading.

Parameters

<i>ret</i>	The return location (bearing in radians).
<i>move_time</i>	The time spent moving forwards in ms (default is 5000ms).

Returns

true iff the bearing could be inferred.

7.12.3.5 bool FlightController::RunTask (TaskIdentifier *tid*, FlightTask * *task*, void * *opts*)

Runs a given task, if no task is currently being run.

Parameters

<i>tid</i>	The task identifier of the task to be run.
<i>task</i>	The task instance to be run.
<i>opts</i>	The task-specific options to be passed to its handler.

Returns

true iff the task was started.

7.12.3.6 bool FlightController::Sleep (int *ms*)

Perform a checked sleep which can be interrupted by the stop signal.

Parameters

<i>ms</i>	The sleep time, in milliseconds.
-----------	----------------------------------

Returns

true iff the sleep completed normally (i.e. not interrupted).

7.12.3.7 void FlightController::Stop (void)

Send an indication that the flight controller should stop the running task.

7.12.3.8 bool FlightController::WaitForAuth ()

Wait for the user to give authorisation for autonomous mode.

Returns

true iff authorisation was given. Will return false when the 'all stop' signal is returned.

7.12.4 Friends And Related Function Documentation

7.12.4.1 std::ostream& operator<< (std::ostream & *stream*, FlightController & *fc*) [friend]

Stream operator of the Flight Controller.

Returns

The current state of the flight controller (description).

7.12.5 Member Data Documentation

7.12.5.1 `Buzzer* const& picopter::FlightController::buzzer`

A pointer to the [Buzzer](#) instance.

7.12.5.2 `CameraStream* const& picopter::FlightController::cam`

A pointer to the Camera stream instance.

7.12.5.3 `FlightBoard* const& picopter::FlightController::fb`

A pointer to the flight board controller instance.

7.12.5.4 `GPS* const& picopter::FlightController::gps`

A pointer to the [GPS](#) instance.

7.12.5.5 `IMU* const& picopter::FlightController::imu`

A pointer to the [IMU](#) instance, or nullptr if not present.

The documentation for this class was generated from the following files:

- [include/flightcontroller.h](#)
- [src/base/flightcontroller.cpp](#)

7.13 `picopter::FlightData` Struct Reference

```
#include <flightboard.h>
```

Public Attributes

- `int` [aileron](#)
- `int` [elevator](#)
- `int` [rudder](#)
- `int` [gimbal](#)

7.13.1 Detailed Description

Contains information about the actuation of the hexacopter

7.13.2 Member Data Documentation

7.13.2.1 `int picopter::FlightData::aileron`

Aileron speed, -100 to 100

7.13.2.2 `int picopter::FlightData::elevators`

Elevator speed, -100 to 100

7.13.2.3 int picopter::FlightData::gimbal

Gimbal angle, 0 to 90

7.13.2.4 int picopter::FlightData::rudder

Rudder speed, -100 to 100

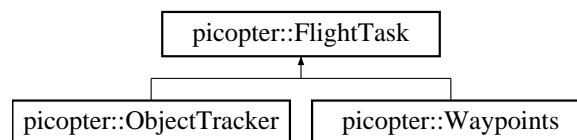
The documentation for this struct was generated from the following file:

- include/[flightboard.h](#)

7.14 picopter::FlightTask Class Reference

```
#include <flightcontroller.h>
```

Inheritance diagram for picopter::FlightTask:



Public Member Functions

- virtual [~FlightTask](#) ()
- virtual void [Run](#) ([FlightController](#) *fc, void *opts)=0

Static Protected Member Functions

- static ControllerState [SetCurrentState](#) ([FlightController](#) *fc, ControllerState state)

7.14.1 Detailed Description

The base class for all flight tasks. All flight tasks must inherit from this class and implement its methods.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 virtual picopter::FlightTask::~~FlightTask () [inline],[virtual]

The destructor. Will be called immediately after [Run\(\)](#) exits.

7.14.3 Member Function Documentation

7.14.3.1 virtual void picopter::FlightTask::Run ([FlightController](#) * fc, void * opts) [pure virtual]

The method that will be called by the flight controller to perform the task.

Parameters

<i>fc</i>	The pointer to the calling flight controller
<i>opts</i>	Task-specific options.

Implemented in `picopter::ObjectTracker`, and `picopter::Waypoints`.

7.14.3.2 `static ControllerState picopter::FlightTask::SetCurrentState (FlightController * fc, ControllerState state)`
`[inline], [static], [protected]`

Sets the current state of the parent flight controller.

Returns

The previous controller state.

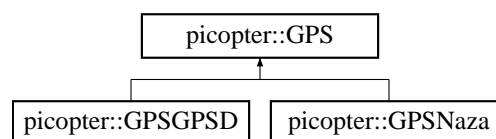
The documentation for this class was generated from the following file:

- `include/flightcontroller.h`

7.15 `picopter::GPS` Class Reference

```
#include <gps_feed.h>
```

Inheritance diagram for `picopter::GPS`:



Public Member Functions

- `GPS ()`
- `GPS (Options *opts)`
- `virtual ~GPS ()`
- `virtual void GetLatest (GPSData *d)`
- `int TimeSinceLastFix ()`
- `bool HasFix ()`
- `bool WaitForFix (int timeout=-1)`

Protected Attributes

- `int m_fix_timeout`
- `std::atomic< GPSData > m_data`
- `std::atomic< int > m_last_fix`
- `std::atomic< bool > m_quit`

Static Protected Attributes

- `static const int FIX_TIMEOUT_DEFAULT = 2`
- `static const int WAIT_PERIOD = 200`

7.15.1 Detailed Description

Class that interacts with the [GPS](#).

7.15.2 Constructor & Destructor Documentation

7.15.2.1 `GPS::GPS ()`

Constructor. Constructs a new [GPS](#) with default settings.

7.15.2.2 `GPS::GPS (Options * opts)`

Constructor. Initialises default stuff.

7.15.2.3 `GPS::~GPS () [virtual]`

Destructor. Sets `m_quit` to true.

7.15.3 Member Function Documentation

7.15.3.1 `void GPS::GetLatest (GPSTData * d) [virtual]`

Returns the latest [GPS](#) fix, if any.

Parameters

<i>d</i>	A pointer to the output location. If no value is present for that parameter, then that value is filled with NaN.
----------	--

7.15.3.2 `bool GPS::HasFix ()`

Determines if there is a current [GPS](#) fix or not

Returns

true iff there is a [GPS](#) fix.

7.15.3.3 `int GPS::TimeSinceLastFix ()`

Returns the amount of time since the last [GPS](#) fix was acquired.

Returns

The time (in seconds) since the last [GPS](#) fix

7.15.3.4 `bool GPS::WaitForFix (int timeout = -1)`

Waits for a [GPS](#) fix.

Parameters

<i>timeout</i>	The timeout (in ms) before this method returns. Default is no timeout (-1).
----------------	---

Returns

true iff a [GPS](#) fix was acquired.

7.15.4 Member Data Documentation

7.15.4.1 `const int picopter::GPS::FIX_TIMEOUT_DEFAULT = 2` `[static]`, `[protected]`

The [GPS](#) fix timeout (in s)

7.15.4.2 `const int GPS::WAIT_PERIOD = 200` `[static]`, `[protected]`

The time checks waits when waiting for a fix (in ms)

The documentation for this class was generated from the following files:

- [include/gps_feed.h](#)
- [src/base/gps.cpp](#)

7.16 `gps_data_t` Struct Reference

Classes

- struct [fix](#)

Public Attributes

- struct [gps_data_t::fix](#) **fix**
- int **set**

The documentation for this struct was generated from the following file:

- [emulation/gps-emu/libgpsmm.h](#)

7.17 `picopter::GPSData` Struct Reference

```
#include <gps_feed.h>
```

Public Member Functions

- `operator navigation::Coord2D ()`

Public Attributes

- [GPSFix](#) **fix**
- [Uncertainty](#) **err**
- double **timestamp**

7.17.1 Detailed Description

Stores information about the current [GPS](#) fix.

Todo Do we need the uncertainty in the timestamp too? 2D info enough?

7.17.2 Member Data Documentation

7.17.2.1 Uncertainty `picopter::GPSData::err`

The uncertainty in the current fix

7.17.2.2 `GPSFix` `picopter::GPSData::fix`

The coordinates of the current [GPS](#) fix

7.17.2.3 `double` `picopter::GPSData::timestamp`

The timestamp of the fix (Unix epoch in seconds w/ fractional)

The documentation for this struct was generated from the following file:

- `include/gps_feed.h`

7.18 `picopter::GPSFix` Struct Reference

Public Attributes

- `double` `lat`
- `double` `lon`
- `double` `alt`
- `double` `speed`
- `double` `heading`
- `double` `bearing`

7.18.1 Member Data Documentation

7.18.1.1 `double` `picopter::GPSFix::alt`

The [GPS](#) altitude, in metres. Uncertainty in metres.

7.18.1.2 `double` `picopter::GPSFix::bearing`

The magnetic bearing, in radians, if available.

7.18.1.3 `double` `picopter::GPSFix::heading`

The heading (track angle), in radians. Uncertainty in radians.

7.18.1.4 double picopter::GPSFix::lat

The latitude, in radians. Uncertainty in metres.

7.18.1.5 double picopter::GPSFix::lon

The longitude, in radians. Uncertainty in metres.

7.18.1.6 double picopter::GPSFix::speed

The speed, in m/s. Uncertainty in m/s.

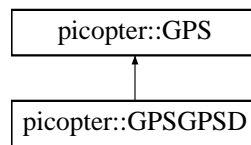
The documentation for this struct was generated from the following file:

- [include/gps_feed.h](#)

7.19 picopter::GPSGPSD Class Reference

```
#include <gps_gpsd.h>
```

Inheritance diagram for picopter::GPSGPSD:



Public Member Functions

- [GPSGPSD \(\)](#)
- [GPSGPSD \(Options *opts\)](#)
- virtual [~GPSGPSD \(\)](#) override

Additional Inherited Members

7.19.1 Detailed Description

Class that interacts with the [GPS](#).

7.19.2 Constructor & Destructor Documentation

7.19.2.1 GPSGPSD::GPSGPSD ()

Constructor. Constructs a new [GPS](#) with default settings.

7.19.2.2 GPSGPSD::GPSGPSD (Options * opts)

Constructor. Establishes a connection to gpsd, assuming it is running on the default gpsd port. Starts the worker thread to receive [GPS](#) data.

Parameters

<i>opts</i>	A pointer to options, if any (NULL for defaults)
-------------	--

Exceptions

<i>std::invalid_argument</i>	if a connection to gpssd cannot be established.
------------------------------	---

7.19.2.3 GPSSGSPD::~GPSSGSPD () [override],[virtual]

Destructor. Stops the worker thread and waits for it to exit.

The documentation for this class was generated from the following files:

- [include/gps_gpsd.h](#)
- [src/base/gps_gpsd.cpp](#)

7.20 gpssmm Class Reference

Public Member Functions

- **gpssmm** (const char *host, const char *port)
- void * **stream** (int a)
- bool **waiting** (int timeout)
- struct [gps_data_t](#) * **read** ()

Public Attributes

- **LATLON_SET**

7.20.1 Member Data Documentation

7.20.1.1 gpssmm::LATLON_SET

Initial value:

```
{
    srand(time(NULL))
```

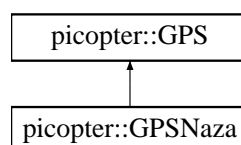
The documentation for this class was generated from the following file:

- [emulation/gps-emu/libgpssmm.h](#)

7.21 picopter::GPSNaza Class Reference

```
#include <gps_naza.h>
```

Inheritance diagram for picopter::GPSNaza:



Public Member Functions

- [GPSNaza](#) ()
- [GPSNaza](#) ([Options](#) *opts)
- virtual [~GPSNaza](#) () override

Additional Inherited Members

7.21.1 Detailed Description

Class that interacts with the [GPS](#).

7.21.2 Constructor & Destructor Documentation

7.21.2.1 GPSNaza::GPSNaza ()

Constructor. Constructs a new [GPS](#) with default settings.

7.21.2.2 GPSNaza::GPSNaza ([Options](#) * opts)

Constructor. Opens the RPi's serial port and reads off data.

7.21.2.3 GPSNaza::~~GPSNaza () [override],[virtual]

Destructor. Stops the worker thread and waits for it to exit.

The documentation for this class was generated from the following files:

- include/[gps_naza.h](#)
- src/base/[gps_naza.cpp](#)

7.22 picopter::IMU Class Reference

```
#include <imu_feed.h>
```

Public Member Functions

- [IMU](#) ()
- [IMU](#) ([Options](#) *opts)
- virtual [~IMU](#) ()
- void [GetLatest](#) ([IMUData](#) *d)

7.22.1 Detailed Description

Reads data from the [IMU](#)

7.22.2 Constructor & Destructor Documentation

7.22.2.1 IMU::IMU ()

Constructor. Constructs [IMU](#) with default options.

7.22.2.2 IMU::IMU (Options * opts)

Constructor. Uses the XSens library to establish a connection to the [IMU](#). Once established, starts the worker thread to receive data from the [IMU](#).

Parameters

<i>opts</i>	A pointer to options, if any (NULL for defaults)
-------------	--

Exceptions

<i>std::invalid_argument</i>	if IMU initialisation fails (e.g. disconnected)
------------------------------	---

7.22.2.3 IMU::~IMU () [virtual]

Destructor. Stops the worker thread and closes the connection to the [IMU](#).

7.22.3 Member Function Documentation

7.22.3.1 void IMU::GetLatest (IMUData * d)

Get the latest [IMU](#) data, if available. Unavailable values are indicated with NaN.

The documentation for this class was generated from the following files:

- [include/imu_feed.h](#)
- [src/base/imu.cpp](#)

7.23 picopter::Lawnmower Class Reference

```
#include <lawnmower.h>
```

Public Member Functions

- **Lawnmower** ([FlightController](#) &fc, [Options](#) *opts)
- void **Run** (Coordinates p1, Coordinates p2)

7.23.1 Detailed Description

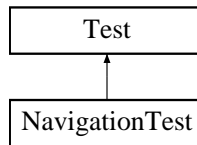
Class to fly the hexacopter in a lawnmower pattern.

The documentation for this class was generated from the following file:

- [include/lawnmower.h](#)

7.24 NavigationTest Class Reference

Inheritance diagram for NavigationTest:



The documentation for this class was generated from the following file:

- test/test_navigation.cpp

7.25 NazaDecoderLib Class Reference

Public Types

- enum **gps_fix_t** { **NO_FIX** = 0, **FIX_2D** = 2, **FIX_3D** = 3, **FIX_DGPS** = 4 }

Public Member Functions

- uint8_t **decode** (int input)
- double **getLat** ()
- double **getLon** ()
- double **getGpsAlt** ()
- double **getSpeed** ()
- gps_fix_t **getFixType** ()
- uint8_t **getNumSat** ()
- double **getHeadingNc** ()
- double **getCog** ()
- double **getGpsVsi** ()
- double **getHdop** ()
- double **getVdop** ()
- uint8_t **getYear** ()
- uint8_t **getMonth** ()
- uint8_t **getDay** ()
- uint8_t **getHour** ()
- uint8_t **getMinute** ()
- uint8_t **getSecond** ()
- int16_t **getMagXRaw** ()
- int16_t **getMagYRaw** ()
- int16_t **getMagZRaw** ()
- double **getMagXVal** ()
- double **getMagYVal** ()
- double **getMagZVal** ()

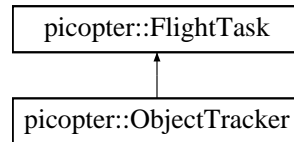
The documentation for this class was generated from the following files:

- include/NazaDecoderLib.h
- src/base/NazaDecoderLib.cpp

7.26 picopter::ObjectTracker Class Reference

```
#include <object_tracker.h>
```

Inheritance diagram for picopter::ObjectTracker:



Public Types

- enum **TrackMethod** { **TRACK_STRAFE**, **TRACK_ROTATE** }

Public Member Functions

- ObjectTracker** (int camwidth, int camheight, TrackMethod method=TRACK_STRAFE)
- ObjectTracker** ([Options](#) *opts, int camwidth, int camheight, TrackMethod method=TRACK_STRAFE)
- TrackMethod **GetTrackMethod** ()
- void **SetTrackMethod** (TrackMethod method)
- void [Run](#) ([FlightController](#) *fc, void *opts) override

Additional Inherited Members

7.26.1 Detailed Description

Class for moving the hexacopter through waypoints.

7.26.2 Member Function Documentation

7.26.2.1 void **ObjectTracker::Run** ([FlightController](#) * *fc*, void * *opts*) [override], [virtual]

The method that will be called by the flight controller to perform the task.

Parameters

<i>fc</i>	The pointer to the calling flight controller
<i>opts</i>	Task-specific options.

Implements [picopter::FlightTask](#).

The documentation for this class was generated from the following files:

- include/[object_tracker.h](#)
- src/modules/[object_tracker.cpp](#)

7.27 picopter::Options Class Reference

```
#include <opts.h>
```


Public Member Functions

- [Options](#) ()
- [Options](#) (const char *file)
- virtual [~Options](#) ()
- void [SetFamily](#) (const char *family)
- int [GetInt](#) (const char *key, int otherwise=0)
- bool [GetBool](#) (const char *key, bool otherwise=false)
- const char * [GetString](#) (const char *key, const char *otherwise="")
- double [GetReal](#) (const char *key, double otherwise=0.0f)
- void [Set](#) (const char *key, int val)
- void [Set](#) (const char *key, bool val)
- void [Set](#) (const char *key, const char *val)
- void [Set](#) (const char *key, double val)
- bool [Remove](#) (const char *key)
- void [Save](#) ()
- void [Save](#) (const char *file)
- void [Save](#) (FILE *fp)

7.27.1 Detailed Description

Provides methods to persistently store and retrieve options. This class is not thread safe.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 Options::Options ()

Constructs an options class with no initial file. This is the same as calling Options::Options(NULL).

7.27.2.2 Options::Options (const char * file)

Constructor. Initialised empty, optionally loading from a file.

Parameters

<i>file</i>	The location of a file containing options, or NULL if not present.
-------------	--

7.27.2.3 Options::~Options () [virtual]

Destructor. Deletes the rapidjson instance.

7.27.3 Member Function Documentation

7.27.3.1 bool Options::GetBool (const char * key, bool otherwise = false)

Retrieves the Boolean value associated with a key.

Parameters

<i>key</i>	The key to the value.
------------	-----------------------

<i>otherwise</i>	The value to return if it does not exist.
------------------	---

Returns

The retrieved value, or *otherwise* if it does not exist.

7.27.3.2 `int Options::GetInt (const char * key, int otherwise = 0)`

Retrieves the integer value associated with a key.

Parameters

<i>key</i>	The key to the value.
<i>otherwise</i>	The value to return if it does not exist.

Returns

The retrieved value, or *otherwise* if it does not exist.

7.27.3.3 `double Options::GetReal (const char * key, double otherwise = 0.0f)`

Retrieves the Real (double precision) value associated with a key.

Parameters

<i>key</i>	The key to the value.
<i>otherwise</i>	The value to return if it does not exist.

Returns

The retrieved value, or *otherwise* if it does not exist.

7.27.3.4 `const char * Options::GetString (const char * key, const char * otherwise = " ")`

Retrieves the string value associated with a key.

Parameters

<i>key</i>	The key to the value.
<i>otherwise</i>	The value to return if it does not exist.

Returns

The retrieved value, or *otherwise* if it does not exist. This value must not be modified. It will only be valid until this parameter is modified (via Set or Remove), or in the case of *otherwise* being returned, until that value is either freed or goes out of scope.

7.27.3.5 `bool Options::Remove (const char * key)`

Removes a value.

Parameters

<i>key</i>	The key to the value.
<i>val</i>	The value to be stored.

7.27.3.6 void Options::Save ()

Saves the current settings to a file. Will save to the file that was specified on construction.

Exceptions

<i>std::invalid_argument</i>	If no file has been set previously.
------------------------------	-------------------------------------

7.27.3.7 void Options::Save (const char * *file*)

Saves the current settings to the specified file. Will keep a copy of the specified path for use with [Save\(\)](#).

7.27.3.8 void Options::Save (FILE * *fp*)

Saves the current settings to the specified stream.

7.27.3.9 void Options::Set (const char * *key*, int *val*)

Stores an integer value.

Parameters

<i>key</i>	The key to the value.
<i>val</i>	The value to be stored.

7.27.3.10 void Options::Set (const char * *key*, bool *val*)

Stores an Boolean value.

Parameters

<i>key</i>	The key to the value.
<i>val</i>	The value to be stored.

7.27.3.11 void Options::Set (const char * *key*, const char * *val*)

Specialisation of Options::SetImpl to store strings. This specialisation is necessary because the value must be copied. rapidjson requires that (a.) it explicitly be copied, or that (b.) it will retain a reference that is guaranteed to be valid for longer than itself.

Parameters

<i>key</i>	The key to the value.
<i>val</i>	The value to be stored.

7.27.3.12 void Options::Set (const char * *key*, double *val*)

Stores an Real value.

Parameters

<i>key</i>	The key to the value.
<i>val</i>	The value to be stored.

7.27.3.13 void Options::SetFamily (const char * *family*)

Sets the family under which to store and retrieve settings from.

Parameters

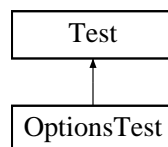
<i>family</i>	The name of the family. If NULL, it will default to Options::FAMILY_DEFAULT.
---------------	--

The documentation for this class was generated from the following files:

- [include/opts.h](#)
- [src/base/opts.cpp](#)

7.28 OptionsTest Class Reference

Inheritance diagram for OptionsTest:



The documentation for this class was generated from the following file:

- [test/test_opts.cpp](#)

7.29 picopter::PID Class Reference

```
#include <PID.h>
```

Public Member Functions

- [PID](#) (float Kc, float tauI, float tauD, float interval)
- void [SetInputLimits](#) (float inMin, float inMax)
- void [SetOutputLimits](#) (float outMin, float outMax)
- void [SetTunings](#) (float Kc, float tauI, float tauD)
- void [Reset](#) (void)
- void [SetMode](#) (int mode)
- void [SetInterval](#) (float interval)
- void [SetSetPoint](#) (float sp)
- void [SetProcessValue](#) (float pv)
- void [SetBias](#) (float bias)
- float [Compute](#) (void)
- float [GetInMin](#) ()
- float [GetInMax](#) ()
- float [GetOutMin](#) ()

- float **GetOutMax** ()
- float **GetInterval** ()
- float **GetPPParam** ()
- float **GetIParam** ()
- float **GetDParam** ()

7.29.1 Detailed Description

Proportional-integral-derivative controller.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 PID::PID (float *Kc*, float *taul*, float *tauD*, float *interval*)

Constructor.

Sets default limits [0-3.3V], calculates tuning parameters, and sets manual mode with no bias.

Parameters

<i>Kc</i>	- Tuning parameter
<i>taul</i>	- Tuning parameter
<i>tauD</i>	- Tuning parameter
<i>interval</i>	PID calculation performed every interval seconds.

7.29.3 Member Function Documentation

7.29.3.1 float PID::Compute (void)

[PID](#) calculation.

Returns

The controller output as a float between outMin and outMax.

7.29.3.2 void PID::Reset (void)

Reinitializes controller internals. Automatically called on a manual to auto transition.

7.29.3.3 void PID::SetBias (float *bias*)

Set the bias.

Parameters

<i>bias</i>	The bias for the controller output.
-------------	-------------------------------------

7.29.3.4 void PID::SetInputLimits (float *inMin*, float *inMax*)

Scale from inputs to 0-100%.

Parameters

<i>InMin</i>	The real world value corresponding to 0%.
<i>InMax</i>	The real world value corresponding to 100%.

7.29.3.5 void PID::SetInterval (float *interval*)

Set how fast the PID loop is run.

Parameters

<i>interval</i>	PID calculation performed every interval seconds.
-----------------	---

7.29.3.6 void PID::SetMode (int *mode*)

Set PID to manual or auto mode.

Parameters

<i>mode</i>	0 -> Manual Non-zero -> Auto
-------------	------------------------------

7.29.3.7 void PID::SetOutputLimits (float *outMin*, float *outMax*)

Scale from outputs to 0-100%.

Parameters

<i>outMin</i>	The real world value corresponding to 0%.
<i>outMax</i>	The real world value corresponding to 100%.

7.29.3.8 void PID::SetProcessValue (float *pv*)

Set the process value.

Parameters

<i>pv</i>	The process value as a real world value.
-----------	--

7.29.3.9 void PID::SetSetPoint (float *sp*)

Set the set point.

Parameters

<i>sp</i>	The set point as a real world value.
-----------	--------------------------------------

7.29.3.10 void PID::SetTunings (float *Kc*, float *taul*, float *tauD*)

Calculate PID constants.

Allows parameters to be changed on the fly without ruining calculations.

Parameters

<i>Kc</i>	- Tuning parameter
<i>tauI</i>	- Tuning parameter
<i>tauD</i>	- Tuning parameter

The documentation for this class was generated from the following files:

- `include/PID.h`
- `src/base/PID.cpp`

7.30 `picopter::navigation::Point2D` Struct Reference

```
#include <navigation.h>
```

Public Member Functions

- `double magnitude ()`

Public Attributes

- `double x`
- `double y`

7.30.1 Detailed Description

Holds a 2-dimensional position in Cartesian space.

7.30.2 Member Function Documentation

7.30.2.1 `double picopter::navigation::Point2D::magnitude ()` `[inline]`

Returns the vector magnitude.

7.30.3 Member Data Documentation

7.30.3.1 `double picopter::navigation::Point2D::x`

x-coordinate.

7.30.3.2 `double picopter::navigation::Point2D::y`

y-coordinate.

The documentation for this struct was generated from the following file:

- `include/navigation.h`

7.31 `picopter::navigation::Point3D` Struct Reference

```
#include <navigation.h>
```

Public Member Functions

- [operator Point2D](#) ()
- double [magnitude](#) ()

Public Attributes

- double [x](#)
- double [y](#)
- double [z](#)

7.31.1 Detailed Description

Holds a 3-dimensional position in Cartesian space.

7.31.2 Member Function Documentation

7.31.2.1 `double picopter::navigation::Point3D::magnitude ()` `[inline]`

Returns the vector magnitude.

7.31.2.2 `picopter::navigation::Point3D::operator Point2D ()` `[inline]`

Implicit conversion to 2D coordinate.

7.31.3 Member Data Documentation

7.31.3.1 `double picopter::navigation::Point3D::x`

x-coordinate.

7.31.3.2 `double picopter::navigation::Point3D::y`

y-coordinate.

7.31.3.3 `double picopter::navigation::Point3D::z`

z-coordinate.

The documentation for this struct was generated from the following file:

- include/[navigation.h](#)

7.32 vec2 Struct Reference

Public Attributes

- int [x](#)
- int [y](#)

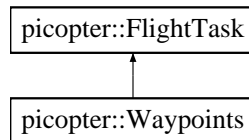
The documentation for this struct was generated from the following file:

- [src/base/camera_stream.cpp](#)

7.33 picopter::Waypoints Class Reference

```
#include <waypoints.h>
```

Inheritance diagram for picopter::Waypoints:



Public Member Functions

- **Waypoints** (std::deque< [navigation::Coord2D](#) > pts, WaypointMethod method)
- **Waypoints** ([Options](#) *opts, std::deque< [navigation::Coord2D](#) > pts, WaypointMethod method)
- void **Run** ([FlightController](#) *fc, void *opts) override

Additional Inherited Members

7.33.1 Detailed Description

Class for moving the hexacopter through waypoints.

7.33.2 Member Function Documentation

7.33.2.1 void Waypoints::Run ([FlightController](#) * *fc*, void * *opts*) [override],[virtual]

The method that will be called by the flight controller to perform the task.

Parameters

<i>fc</i>	The pointer to the calling flight controller
<i>opts</i>	Task-specific options.

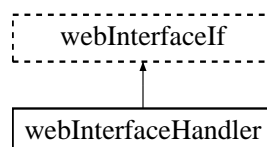
Implements [picopter::FlightTask](#).

The documentation for this class was generated from the following files:

- [include/waypoints.h](#)
- [src/modules/waypoints.cpp](#)

7.34 webInterfaceHandler Class Reference

Inheritance diagram for webInterfaceHandler:



Public Member Functions

- **webInterfaceHandler** (Options *opts, std::unique_ptr< [picopter::FlightController](#) > &fc)
- bool **beginWaypointsThread** ()
- bool **beginLawnmowerThread** ()
- bool **beginUserTrackingThread** ()
- bool **beginObjectTrackingThread** (const int32_t method)
- int32_t **setCameraMode** (int32_t mode)
- int32_t **requestCameraMode** ()
- bool **setCameraLearningSize** (bool decrease)
- bool **showLearningThreshold** (bool show)
- void **doCameraAutoLearning** (std::map< std::string, int32_t > &_return)
- void **setCameraLearningValues** (std::map< std::string, int32_t > &_return, const std::map< std::string, int32_t > &values)
- int32_t **requestLearningHue** ()
- bool **allStop** ()
- void **requestStatus** (std::string &_return)
- void **requestCoords** (coordDeg &_return)
- double **requestBearing** ()
- void **requestNextWaypoint** (coordDeg &_return)
- bool **updateUserPosition** (const coordDeg &wpt)
- bool **updateWaypoints** (const std::vector< coordDeg > &wpts)
- bool **resetWaypoints** ()

The documentation for this class was generated from the following file:

- [src/server/picopter.cpp](#)

Chapter 8

File Documentation

8.1 emulation/gps-emu/libgpsmm.h File Reference

A *very thin* emulation of libgpsmm. Will always return that there is data, but with all values zeroed. Data will randomly be marked as set or unset.

```
#include <config.h>
#include <thread>
#include <chrono>
#include <cstdlib>
```

Classes

- struct [gps_data_t](#)
- struct [gps_data_t::fix](#)
- class [gpsmm](#)

Macros

- #define **DEFAULT_GPSD_PORT** ""
- #define **WATCH_ENABLE** 1
- #define **WATCH_JSON** 1
- #define **ONLINE_SET** (1llu<<1)
- #define **TIME_SET** (1llu<<2)
- #define **TIMERR_SET** (1llu<<3)
- #define **LATLON_SET** (1llu<<4)
- #define **ALTITUDE_SET** (1llu<<5)
- #define **SPEED_SET** (1llu<<6)
- #define **TRACK_SET** (1llu<<7)
- #define **CLIMB_SET** (1llu<<8)
- #define **STATUS_SET** (1llu<<9)
- #define **MODE_SET** (1llu<<10)
- #define **DOP_SET** (1llu<<11)
- #define **HERR_SET** (1llu<<12)
- #define **VERR_SET** (1llu<<13)
- #define **ATTITUDE_SET** (1llu<<14)
- #define **SATELLITE_SET** (1llu<<15)
- #define **SPEEDERR_SET** (1llu<<16)
- #define **TRACKERR_SET** (1llu<<17)

- `#define CLIMBERR_SET (1llu<<18)`
- `#define DEVICE_SET (1llu<<19)`
- `#define DEVICELIST_SET (1llu<<20)`
- `#define DEVICEID_SET (1llu<<21)`
- `#define RTCM2_SET (1llu<<22)`
- `#define RTCM3_SET (1llu<<23)`
- `#define AIS_SET (1llu<<24)`
- `#define PACKET_SET (1llu<<25)`
- `#define SUBFRAME_SET (1llu<<26)`
- `#define GST_SET (1llu<<27)`
- `#define VERSION_SET (1llu<<28)`
- `#define POLICY_SET (1llu<<29)`
- `#define LOGMESSAGE_SET (1llu<<30)`
- `#define ERROR_SET (1llu<<31)`
- `#define TIMEDRIFT_SET (1llu<<32)`
- `#define EOF_SET (1llu<<33)`

8.1.1 Detailed Description

A *very thin* emulation of libgpsmm. Will always return that there is data, but with all values zeroed. Data will randomly be marked as set or unset.

8.2 emulation/wiringPi.h File Reference

Simple stub file to get 'wiringPi' when not compiling on the RPi.

Macros

- `#define wiringPiSetup()`
- `#define pinMode(pin, mode)`
- `#define digitalWrite(pin) 1`
- `#define digitalWrite(pin, value)`
- `#define HIGH 1`
- `#define LOW 0`
- `#define delayMicroseconds(x) (std::this_thread::sleep_for(std::chrono::microseconds(x)))`

8.2.1 Detailed Description

Simple stub file to get 'wiringPi' when not compiling on the RPi.

8.3 emulation/wiringSerial.h File Reference

Simple stub file to get 'wiringSerial' when not compiling on the RPi.

Macros

- `#define serialOpen(A, B) 0`
- `#define serialDataAvail(fd) 0`
- `#define serialGetchar(fd) fd`
- `#define serialClose(fd)`

8.3.1 Detailed Description

Simple stub file to get 'wiringSerial' when not compiling on the RPi.

8.4 include/buzzer.h File Reference

Methods to control the buzzer.

Classes

- class [picopter::Buzzer](#)

8.4.1 Detailed Description

Methods to control the buzzer.

8.5 include/camera_stream.h File Reference

```
#include "common.h"
#include "navigation.h"
#include <opencv2/opencv.hpp>
```

Classes

- struct [picopter::CamWindow](#)
- class [picopter::CameraStream](#)

Macros

- #define **STREAM_FILE** "/mnt/ramdisk/out.jpg"
- #define **LOOKUP_SIZE** 8
- #define **CHAR_SIZE** 256
- #define **CAMERA_OK** 0

8.5.1 Detailed Description

Author

Michael Baxter 20503664@student.uwa.edu.au
Jeremy Tan 20933708@student.uwa.edu.au

Class used to start camera stream.

Wonderful omni-function camera streaming action fun!

8.6 include/common.h File Reference

Commonly included headers.

```
#include "config.h"
#include "log.h"
#include "datalog.h"
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <iostream>
#include <string>
#include <vector>
#include <deque>
#include <map>
#include <stdint.h>
#include "opts.h"
#include <thread>
#include <future>
#include <mutex>
#include <condition_variable>
#include <atomic>
```

Functions

- `template<typename T >`
`T picopter::clamp` (const T &n, const T &lower, const T &upper)

8.6.1 Detailed Description

Commonly included headers.

8.7 include/config.h File Reference

System-specific and compile-time specific configuration parameters.

Macros

- `#define PICOPTER_VERSION` "eddba9ba81c0e3b3b822e310bb862e9c397b9b39"
- `#define PICOPTER_DATE` "2015-05-15 17:17:34 +0800"
- `#define PICOPTER_HOME_LOCATION` "/home/jeremy"
- `#define PICOPTER_LOG_LOCATION` "/home/jeremy/logs"
- `#define USE_SYSLOG`
- `#define IS_REAL_LIBGPS`

8.7.1 Detailed Description

System-specific and compile-time specific configuration parameters.

8.8 include/datalog.h File Reference

Declaration of functions for data logging.

```
#include "config.h"
#include <cstdio>
```

Classes

- class [picopter::DataLog](#)

8.8.1 Detailed Description

Declaration of functions for data logging.

8.9 include/flightboard.h File Reference

Defines the FlightBoard class.

Classes

- struct [picopter::FlightData](#)
- class [picopter::FlightBoard](#)

Typedefs

- typedef struct [picopter::FlightData](#) [picopter::FlightData](#)

8.9.1 Detailed Description

Defines the FlightBoard class.

8.10 include/flightcontroller.h File Reference

Defines the FlightController class.

```
#include "buzzer.h"
#include "gps_gpsd.h"
#include "gps_naza.h"
#include "imu_feed.h"
#include "flightboard.h"
#include "camera_stream.h"
```

Classes

- class [picopter::FlightController](#)
- class [picopter::FlightTask](#)

Typedefs

- typedef enum picopter::ControllerState **picopter::ControllerState**
- typedef enum picopter::TaskIdentifier **picopter::TaskIdentifier**

Enumerations

- enum **ControllerState** {
picopter::STATE_STOPPED, **picopter::STATE_GPS_WAIT_FOR_FIX**, **picopter::STATE_AWAITING_↵**
AUTH, **picopter::STATE_INFER_BEARING**,
picopter::STATE_WAYPOINTS_MOVING, **picopter::STATE_WAYPOINTS_IDLING**, **picopter::STATE_↵**
_WAYPOINTS_FINISHED, **picopter::STATE_TRACKING_SEARCHING**,
picopter::STATE_TRACKING_LOCKED }
- enum **TaskIdentifier** {
picopter::TASK_NONE, **picopter::TASK_WAYPOINTS**, **picopter::TASK_LAWNMOWER**, **picopter::T_↵**
ASK_OBJECT_TRACKING,
picopter::TASK_USER_TRACKING, **picopter::TASK_SPIRAL_SEARCH** }

8.10.1 Detailed Description

Defines the FlightController class.

8.11 include/gpio.h File Reference

gpio class defines

Namespaces

- [picopter::gpio](#)

Functions

- void [picopter::gpio::Init](#) ()
- bool [picopter::gpio::IsAutoMode](#) ()
- void [picopter::gpio::SetBuzzer](#) (bool value)

Variables

- const int [picopter::gpio::MODE_PIN](#) = 5
- const int [picopter::gpio::BUZZER_PIN](#) = 2

8.11.1 Detailed Description

gpio class defines

8.12 include/gps_feed.h File Reference

Base GPS header.

```
#include "navigation.h"
```


Classes

- struct [picopter::GPSFix](#)
- struct [picopter::GPSData](#)
- class [picopter::GPS](#)

Typedefs

- typedef struct [picopter::GPSFix](#) **picopter::GPSFix**
- typedef GPSFix **picopter::Uncertainty**
- typedef struct [picopter::GPSData](#) **picopter::GPSData**

8.12.1 Detailed Description

Base GPS header.

8.13 include/gps_gpsd.h File Reference

GPS header for the gpsd implementation.

```
#include "gps_feed.h"
```

Classes

- class [picopter::GPSGPSD](#)

8.13.1 Detailed Description

GPS header for the gpsd implementation.

8.14 include/gps_naza.h File Reference

GPS header for the NAZA implementation.

```
#include "gps_feed.h"
```

Classes

- class [picopter::GPSNaza](#)

8.14.1 Detailed Description

GPS header for the NAZA implementation.

8.15 include/imu_feed.h File Reference

Defines the IMU class.

```
#include "navigation.h"
```

Classes

- class [picopter::IMU](#)

Typedefs

- typedef navigation::EulerAngle **picopter::IMUData**

8.15.1 Detailed Description

Defines the IMU class.

8.16 include/lawnmower.h File Reference

Defines the lawnmower controls.

Classes

- class [picopter::Lawnmower](#)

8.16.1 Detailed Description

Defines the lawnmower controls.

8.17 include/log.h File Reference

Declaration of functions for printing log messages and/or terminating program after a fatal error.

```
#include "config.h"  
#include <syslog.h>
```

Macros

- #define **Log**(level, ...) [LogEx](#)(level, __PRETTY_FUNCTION__, __FILENAME__, __LINE__, __VA_ARGS__,
__)
- #define **Fatal**(...) [FatalEx](#)(__PRETTY_FUNCTION__, __FILENAME__, __LINE__, __VA_ARGS__)

Functions

- void [LogInit](#) ()
- void [LogSimple](#) (int level, const char *fmt,...)

- void [LogEx](#) (int level, const char *funct, const char *file, int line,...)
- void [FatalEx](#) (const char *funct, const char *file, int line,...)

8.17.1 Detailed Description

Declaration of functions for printing log messages and/or terminating program after a fatal error.

8.17.2 Function Documentation

8.17.2.1 void FatalEx (const char * *funct*, const char * *file*, int *line*, ...)

Handle a Fatal error in the program by printing a message and exiting the program CALLING THIS FUNCTION WILL CAUSE THE PROGRAM TO EXIT

Parameters

<i>funct</i>	- Name of the calling function
<i>file</i>	- Name of the source file containing the calling function
<i>line</i>	- Line in the source file at which Fatal is called
<i>fmt</i>	- A format string
...	- Arguments to be printed according to the format string

8.17.2.2 void LogEx (int *level*, const char * *funct*, const char * *file*, int *line*, ...)

Print a message to stderr and log it via syslog. The message must be less than BUFSIZ characters long, or it will be truncated.

Parameters

<i>level</i>	Specify how severe the message is. If level is higher (less urgent) than the program's verbosity (see options.h) no message will be printed.
<i>funct</i>	String indicating the function name from which this function was called. If this is NULL, Log will show the unspecified _funct string instead.
<i>file</i>	Source file containing the function
<i>line</i>	Line in the source file at which Log is called
<i>fmt</i>	A format string
...	Arguments to be printed according to the format string

8.17.2.3 void LogInit ()

Initialises the logger. Should be called at the start of a program.

8.17.2.4 void LogSimple (int *level*, const char * *fmt*, ...)

Print a message to stderr and log it via syslog. The message must be less than BUFSIZ characters long, or it will be truncated. This is a simple version that does not print the line number/file from which the call was made.

Parameters

<i>level</i>	Specify how severe the message is. If level is higher (less urgent) than the program's verbosity (see options.h) no message will be printed.
--------------	--

<i>fmt</i>	A format string
...	Arguments to be printed according to the format string

8.18 include/navigation.h File Reference

General navigation code.

```
#include <cmath>
```

Classes

- struct [picopter::navigation::Coord2D](#)
- struct [picopter::navigation::Coord3D](#)
- struct [picopter::navigation::Point2D](#)
- struct [picopter::navigation::Point3D](#)
- struct [picopter::navigation::EulerAngle](#)

Macros

- #define **_USE_MATH_DEFINES**
- #define [RADIUS_OF_EARTH](#) 6364.963
- #define **RAD2DEG**(x) ((x) * (180.0 / M_PI))
- #define **DEG2RAD**(x) ((x) * (M_PI / 180.0))
- #define **sin2**(x) (sin(x) * (sin(x)))

Typedefs

- typedef struct [picopter::navigation::Coord2D](#) **picopter::navigation::Coord2D**
- typedef struct [picopter::navigation::Coord3D](#) **picopter::navigation::Coord3D**
- typedef struct [picopter::navigation::Point2D](#) **picopter::navigation::Point2D**
- typedef struct [picopter::navigation::Point3D](#) **picopter::navigation::Point3D**
- typedef struct [picopter::navigation::EulerAngle](#) **picopter::navigation::EulerAngle**

Functions

- template<typename Coord1 , typename Coord2 , typename Coord3 >
bool **picopter::navigation::CoordInBounds** (Coord1 here, Coord2 bl, Coord3 tr)
- template<typename Coord >
void **picopter::navigation::CoordInRadians** (Coord &a)
- template<typename Coord >
void **picopter::navigation::CoordInDegrees** (Coord &a)
- template<typename Coord1 , typename Coord2 >
double **picopter::navigation::CoordDistance** (Coord1 from, Coord2 to)
- template<typename Coord1 , typename Coord2 >
double **picopter::navigation::CoordBearing** (Coord1 from, Coord2 to)

Variables

- const Coord2D **picopter::navigation::PERTH_BL** = {-33, 115}
- const Coord2D **picopter::navigation::PERTH_TR** = {-31, 117}

8.18.1 Detailed Description

General navigation code.

8.18.2 Macro Definition Documentation

8.18.2.1 #define RADIUS_OF_EARTH 6364.963

Radius of the earth (Australian tuned; in km)

8.19 include/object_tracker.h File Reference

The header file for the object tracking code.

```
#include "flightcontroller.h"
#include "navigation.h"
#include "PID.h"
```

Classes

- class [picopter::ObjectTracker](#)

8.19.1 Detailed Description

The header file for the object tracking code.

8.20 include/opts.h File Reference

Header for handling options and persistent configurations.

Classes

- class [picopter::Options](#)

8.20.1 Detailed Description

Header for handling options and persistent configurations.

8.21 include/picopter.h File Reference

The main include file.

```
#include "common.h"
#include "gpio.h"
#include "buzzer.h"
#include "navigation.h"
#include "gps_feed.h"
#include "imu_feed.h"
#include "flightboard.h"
#include "flightcontroller.h"
#include "waypoints.h"
#include "object_tracker.h"
```

8.21.1 Detailed Description

The main include file.

8.22 include/waypoints.h File Reference

The header file for the waypoints code.

```
#include "flightcontroller.h"
#include "PID.h"
```

Classes

- class [picopter::Waypoints](#)

Typedefs

- typedef enum picopter::WaypointMethod **picopter::WaypointMethod**

Enumerations

- enum **WaypointMethod** { **WAYPOINT_SIMPLE** }

8.22.1 Detailed Description

The header file for the waypoints code.

8.23 src/apps/naza_decoder.cpp File Reference

Sample application to decode and print out NAZA gps data.

```
#include "common.h"
#include "navigation.h"
#include "NazaDecoderLib.h"
#include <cmath>
#include <wiringPi.h>
#include <wiringSerial.h>
```

Functions

- void `decodeMessage` (`NazaDecoderLib` &`decoder`, `uint8_t` `buf`)
- int `main` (int `argc`, char *`argv`[])

8.23.1 Detailed Description

Sample application to decode and print out NAZA gps data.

8.23.2 Function Documentation

8.23.2.1 void `decodeMessage` (`NazaDecoderLib` & `decoder`, `uint8_t` `buf`)

Decode the bytestream and print the decoded message when available.

Parameters

<i>decoder</i>	The NAZA GPS decoder
<i>buf</i>	The current byte in the raw bytestream

8.23.2.2 int `main` (int `argc`, char * `argv`[])

Main entry point.

8.24 src/apps/speedcal.cpp File Reference

Application to determine relation between input and speed.

```
#include "picopter.h"
#include <signal.h>
```

Functions

- int `main` (int `argc`, char *`argv`[])

8.24.1 Detailed Description

Application to determine relation between input and speed.

8.25 src/base/buzzer.cpp File Reference

Buzzer manipulation code. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM.

```
#include "common.h"
#include "gpio.h"
#include "buzzer.h"
#include <wiringPi.h>
```

Typedefs

- using **hrc** = std::chrono::high_resolution_clock

8.25.1 Detailed Description

Buzzer manipulation code. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM.

8.26 src/base/camera_stream.cpp File Reference

```
#include "common.h"
#include "camera_stream.h"
#include <queue>
```

Classes

- struct [CamComponent](#)
- struct [vec2](#)

Macros

- #define **BLACK** 0
- #define **WHITE** 255

Functions

- template<typename T1 , typename T2 , typename T3 >
T3 **GetFromMap** (T1 &m, T2 val, T3 otherwise)

8.26.1 Detailed Description

Author

Michael Baxter 20503664@student.uwa.edu.au
Jeremy Tan 20933708@student.uwa.edu.au

Camera functions

8.27 src/base/datalog.cpp File Reference

Implement data logging functions.

```
#include "common.h"
#include "datalog.h"
#include <ctime>
#include <cstdarg>
```


8.27.1 Detailed Description

Implement data logging functions.

8.28 src/base/flightboard-private.h File Reference

Contains calibration data and pin mappings for the FlightBoard class. Note: The unit 'step' is ServoBlaster specific. Usually 1 step is 10us.

Macros

- #define [LINEAR_SCALE](#)(x, xl, xh, yl, yh) $((yl) + (((yh) - (yl)) * ((x) - (xl))) / ((xh) - (xl)))$
- #define [SPEED_SCALE](#)(speed, yl, yh) [LINEAR_SCALE](#)(speed, -100, 100, yl, yh)
- #define [INV_SPEED_SCALE](#)(val, yl, yh) [LINEAR_SCALE](#)(val, yl, yh, -100, 100)
- #define [AILERON_CHANNEL](#) 0
- #define [AILERON_PIN_PHYSICAL](#) 11
- #define [AILERON_LOW](#) 111
- #define [AILERON_HIGH](#) 193
- #define [AILERON_SCALE](#)(x) [SPEED_SCALE](#)(x, [AILERON_LOW](#), [AILERON_HIGH](#))
- #define [INV_AILERON_SCALE](#)(x) [INV_SPEED_SCALE](#)(x, [AILERON_LOW](#), [AILERON_HIGH](#))
- #define [ELEVATOR_CHANNEL](#) 1
- #define [ELEVATOR_PIN_PHYSICAL](#) 12
- #define [ELEVATOR_LOW](#) 111
- #define [ELEVATOR_HIGH](#) 195
- #define [ELEVATOR_SCALE](#)(x) [SPEED_SCALE](#)(x, [ELEVATOR_LOW](#), [ELEVATOR_HIGH](#))
- #define [INV_ELEVATOR_SCALE](#)(x) [INV_SPEED_SCALE](#)(x, [ELEVATOR_LOW](#), [ELEVATOR_HIGH](#))
- #define [RUDDER_CHANNEL](#) 2
- #define [RUDDER_PIN_PHYSICAL](#) 15
- #define [RUDDER_LOW](#) 110
- #define [RUDDER_HIGH](#) 194
- #define [RUDDER_SCALE](#)(x) [SPEED_SCALE](#)(x, [RUDDER_LOW](#), [RUDDER_HIGH](#))
- #define [INV_RUDDER_SCALE](#)(x) [INV_SPEED_SCALE](#)(x, [RUDDER_LOW](#), [RUDDER_HIGH](#))
- #define [GIMBAL_CHANNEL](#) 3
- #define [GIMBAL_PIN_PHYSICAL](#) 16
- #define [GIMBAL_LOW](#) 95
- #define [GIMBAL_HIGH](#) 210
- #define [GIMBAL_SCALE](#)(x) [LINEAR_SCALE](#)(x, 0, 90, [GIMBAL_LOW](#), [GIMBAL_HIGH](#))
- #define [INV_GIMBAL_SCALE](#)(x) [LINEAR_SCALE](#)(x, [GIMBAL_LOW](#), [GIMBAL_HIGH](#), 0, 90)

8.28.1 Detailed Description

Contains calibration data and pin mappings for the FlightBoard class. Note: The unit 'step' is ServoBlaster specific. Usually 1 step is 10us.

8.28.2 Macro Definition Documentation

8.28.2.1 #define AILERON_CHANNEL 0

The ServoBlaster channel for the aileron

8.28.2.2 #define AILERON_HIGH 193

The upper bound for the pulse width (in steps) of the aileron

8.28.2.3 #define AILERON_LOW 111

The lower bound for the pulse width (in steps) of the aileron

8.28.2.4 #define AILERON_PIN_PHYSICAL 11

The actual (physical) pin number for the aileron

8.28.2.5 #define AILERON_SCALE(x) SPEED_SCALE(x, AILERON_LOW, AILERON_HIGH)

Scales the aileron speed to the corresponding PWM range

8.28.2.6 #define ELEVATOR_CHANNEL 1

The ServoBlaster channel for the elevator

8.28.2.7 #define ELEVATOR_HIGH 195

The upper bound for the pulse width (in steps) of the elevator

8.28.2.8 #define ELEVATOR_LOW 111

The lower bound for the pulse width (in steps) of the elevator

8.28.2.9 #define ELEVATOR_PIN_PHYSICAL 12

The actual (physical) pin number for the elevator

8.28.2.10 #define ELEVATOR_SCALE(x) SPEED_SCALE(x, ELEVATOR_LOW, ELEVATOR_HIGH)

Scales the elevator speed to the corresponding PWM range

8.28.2.11 #define GIMBAL_CHANNEL 3

The ServoBlaster channel for the gimbal

8.28.2.12 #define GIMBAL_HIGH 210

The upper bound for the pulse width (in steps) of the gimbal

8.28.2.13 #define GIMBAL_LOW 95

The lower bound for the pulse width (in steps) of the gimbal

8.28.2.14 #define GIMBAL_PIN_PHYSICAL 16

The actual (physical) pin number for the gimbal

8.28.2.15 #define GIMBAL_SCALE(x) LINEAR_SCALE(x, 0, 90, GIMBAL_LOW, GIMBAL_HIGH)

Scales the gimbal angle (0 - 90) to the corresponding PWM range

8.28.2.16 #define LINEAR_SCALE(x, xl, xh, yl, yh) ((yl) + (((yh) - (yl)) * ((x) - (xl))) / ((xh) - (xl)))

Linearly scales from one range to another range.

Parameters

x	The input value (assumed that $x_l < x < x_h$)
x_l	The lower bound on the input scale
x_h	The upper bound on the input scale
y_l	The lower bound on the output scale
y_h	The upper bound on the output scale

8.28.2.17 #define RUDDER_CHANNEL 2

The ServoBlaster channel for the rudder

8.28.2.18 #define RUDDER_HIGH 194

The upper bound for the pulse width (in steps) of the rudder

8.28.2.19 #define RUDDER_LOW 110

The lower bound for the pulse width (in steps) of the rudder

8.28.2.20 #define RUDDER_PIN_PHYSICAL 15

The actual (physical) pin number for the rudder

8.28.2.21 #define RUDDER_SCALE(x) SPEED_SCALE(x, RUDDER_LOW, RUDDER_HIGH)

Scales the rudder speed to the corresponding PWM range

8.28.2.22 #define SPEED_SCALE(speed, yl, yh) LINEAR_SCALE(speed, -100, 100, yl, yh)

Helper to scale a speed (-100 to 100) value

8.29 src/base/flightboard.cpp File Reference

Controls the output to the flight board. Calculates the correct PWM pulse width to be sent to the PWM drivers for the aileron, elevator and rudder, as well as camera gimbal. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM.

```
#include "common.h"
#include "flightboard.h"
#include "flightboard-private.h"
```

8.29.1 Detailed Description

Controls the output to the flight board. Calculates the correct PWM pulse width to be sent to the PWM drivers for the aileron, elevator and rudder, as well as camera gimbal. Uses wiringPi and a GPIO pin for driving the buzzer with software based PWM.

8.30 src/base/flightcontroller.cpp File Reference

Base controller. Ties in all the equipment (sensors and actuators).

```
#include "common.h"
#include "flightcontroller.h"
#include "gpio.h"
```

Typedefs

- using **steady_clock** = std::chrono::steady_clock

Functions

- template<typename Item >
void [InitialiseItem](#) (const char *what, Item *&pt, [Options](#) *opts, [Buzzer](#) *b, bool required, int tries=-1)
- std::ostream & **picopter::operator**<< (std::ostream &stream, [FlightController](#) &fc)

8.30.1 Detailed Description

Base controller. Ties in all the equipment (sensors and actuators).

Todo Add in camera things.

8.30.2 Function Documentation

8.30.2.1 template<typename Item > void [InitialiseItem](#) (const char * *what*, Item *& *pt*, [Options](#) * *opts*, [Buzzer](#) * *b*, bool *required*, int *tries* = -1)

Helper method to initialise a base module.

Template Parameters

<i>Item</i>	The class of the base module to be initialised.
-------------	---

Parameters

<i>what</i>	A description of what the base module is.
<i>pt</i>	A reference to the pointer where the result will be stored.
<i>opts</i>	A pointer to the options instance, or NULL.
<i>b</i>	The buzzer.
<i>required</i>	Indicated if this module is required.
<i>tries</i>	The number of tries to make.

8.31 src/base/gpio.cpp File Reference

GPIO handling code. Uses wiringPi to control the GPIO pins on the RPi.

```
#include "common.h"
#include "gpio.h"
#include <wiringPi.h>
```

8.31.1 Detailed Description

GPIO handling code. Uses wiringPi to control the GPIO pins on the RPi.

8.32 src/base/gps.cpp File Reference

Base GPS interaction code.

```
#include "common.h"
#include "gps_feed.h"
```

Typedefs

- using **steady_clock** = std::chrono::steady_clock

Functions

- **m_last_fix** (999)
- **m_quit** (false)

8.32.1 Detailed Description

Base GPS interaction code.

8.33 src/base/gps_gpsd.cpp File Reference

GPS interaction code. Uses gpsd to interact with the GPS.

```
#include "common.h"
#include "gps_gpsd.h"
#include "libgpsmm.h"
```

Typedefs

- using **steady_clock** = std::chrono::steady_clock

8.33.1 Detailed Description

GPS interaction code. Uses gpsd to interact with the GPS.

8.34 src/base/gps_naza.cpp File Reference

GPS interaction code. Uses the NAZA decoder to interact with the NAZA GPS.

```
#include "common.h"
#include "gps_naza.h"
#include "NazaDecoderLib.h"
#include <wiringSerial.h>
```

Typedefs

- using **steady_clock** = std::chrono::steady_clock

8.34.1 Detailed Description

GPS interaction code. Uses the NAZA decoder to interact with the NAZA GPS.

8.35 src/base/imu.cpp File Reference

IMU interaction code.

```
#include "common.h"
#include "imu_feed.h"
#include "cmt3.h"
```

8.35.1 Detailed Description

IMU interaction code.

8.36 src/base/log.cpp File Reference

Implement logging and error handling functions.

```
#include "common.h"
#include "log.h"
#include <cstdarg>
#include <unistd.h>
```

Functions

- void [LogInit](#) ()
- void [LogEx](#) (int level, const char *funct, const char *file, int line,...)
- void [LogSimple](#) (int level, const char *fmt,...)
- void [FatalEx](#) (const char *funct, const char *file, int line,...)

8.36.1 Detailed Description

Implement logging and error handling functions.

8.36.2 Function Documentation

8.36.2.1 void FatalEx (const char * *funct*, const char * *file*, int *line*, ...)

Handle a Fatal error in the program by printing a message and exiting the program CALLING THIS FUNCTION WILL CAUSE THE PROGRAM TO EXIT

Parameters

<i>funct</i>	- Name of the calling function
<i>file</i>	- Name of the source file containing the calling function
<i>line</i>	- Line in the source file at which Fatal is called
<i>fmt</i>	- A format string
...	- Arguments to be printed according to the format string

8.36.2.2 void LogEx (int *level*, const char * *funct*, const char * *file*, int *line*, ...)

Print a message to stderr and log it via syslog. The message must be less than BUFSIZ characters long, or it will be truncated.

Parameters

<i>level</i>	Specify how severe the message is. If level is higher (less urgent) than the program's verbosity (see options.h) no message will be printed.
<i>funct</i>	String indicating the function name from which this function was called. If this is NULL, Log will show the unspecified_ <i>funct</i> string instead.
<i>file</i>	Source file containing the function
<i>line</i>	Line in the source file at which Log is called
<i>fmt</i>	A format string
...	Arguments to be printed according to the format string

8.36.2.3 void LogInit ()

Initialises the logger. Should be called at the start of a program.

8.36.2.4 void LogSimple (int *level*, const char * *fmt*, ...)

Print a message to stderr and log it via syslog. The message must be less than BUFSIZ characters long, or it will be truncated. This is a simple version that does not print the line number/file from which the call was made.

Parameters

<i>level</i>	Specify how severe the message is. If level is higher (less urgent) than the program's verbosity (see options.h) no message will be printed.
<i>fmt</i>	A format string
...	Arguments to be printed according to the format string

8.37 src/base/opts.cpp File Reference

Options and persistent configurations handler.

```
#include "common.h"
#include "opts.h"
#include <rapidjson/document.h>
#include <rapidjson/filereadstream.h>
#include <rapidjson/filewritestream.h>
#include <rapidjson/prettywriter.h>
```

8.37.1 Detailed Description

Options and persistent configurations handler.

8.38 src/modules/object_tracker.cpp File Reference

The object tracking code.

```
#include "common.h"
#include "object_tracker.h"
```

8.38.1 Detailed Description

The object tracking code.

8.39 src/modules/waypoints.cpp File Reference

Contains the waypoints handling code.

```
#include "common.h"
#include "waypoints.h"
#include <cmath>
```

8.39.1 Detailed Description

Contains the waypoints handling code.

8.40 src/server/picopter.cpp File Reference

The main entry point to the server.

```
#include "picopter.h"
#include "webInterface.h"
#include <arpa/inet.h>
#include <thrift/concurrency/ThreadManager.h>
#include <thrift/concurrency/PosixThreadFactory.h>
#include <thrift/protocol/TBinaryProtocol.h>
#include <thrift/server/TThreadPoolServer.h>
#include <thrift/transport/TServerSocket.h>
#include <thrift/transport/TBufferTransports.h>
#include <sstream>
#include <csignal>
```

Classes

- class [webInterfaceHandler](#)

Functions

- std::unique_ptr< TThreadPoolServer > **g_server** (nullptr)
- std::unique_ptr< [picopter::FlightController](#) > **g_fc** (nullptr)
- void [terminate](#) (int signum)
- int **main** (int argc, char **argv)

8.40.1 Detailed Description

The main entry point to the server.

8.40.2 Function Documentation

8.40.2.1 void terminate (int *signum*)

SIGINT/SIGTERM interrupt handler

Index

- ~Buzzer
 - picopter::Buzzer, [13](#)
- ~DataLog
 - picopter::DataLog, [17](#)
- ~FlightBoard
 - picopter::FlightBoard, [20](#)
- ~FlightController
 - picopter::FlightController, [22](#)
- ~FlightTask
 - picopter::FlightTask, [26](#)
- ~GPS
 - picopter::GPS, [28](#)
- ~GPSGPSD
 - picopter::GPSGPSD, [32](#)
- ~GPSNaza
 - picopter::GPSNaza, [33](#)
- ~IMU
 - picopter::IMU, [34](#)
- ~Options
 - picopter::Options, [37](#)
- AILERON_CHANNEL
 - flightboard-private.h, [61](#)
- AILERON_HIGH
 - flightboard-private.h, [61](#)
- AILERON_LOW
 - flightboard-private.h, [62](#)
- AILERON_PIN_PHYSICAL
 - flightboard-private.h, [62](#)
- AILERON_SCALE
 - flightboard-private.h, [62](#)
- aileron
 - picopter::FlightData, [25](#)
- alt
 - picopter::GPSFix, [30](#)
 - picopter::navigation::Coord3D, [17](#)
- BUZZER_PIN
 - picopter::gpio, [12](#)
- bearing
 - picopter::GPSFix, [30](#)
- Buzzer
 - picopter::Buzzer, [13](#)
- buzzer
 - picopter::FlightController, [25](#)
- BuzzerTest, [14](#)
- cam
 - picopter::FlightController, [25](#)
- CamComponent, [14](#)

- CheckForStop
 - picopter::FlightController, [22](#)
- Compute
 - picopter::PID, [41](#)
- DataLog
 - picopter::DataLog, [17](#)
- decodeMessage
 - naza_decoder.cpp, [59](#)
- ELEVATOR_CHANNEL
 - flightboard-private.h, [62](#)
- ELEVATOR_HIGH
 - flightboard-private.h, [62](#)
- ELEVATOR_LOW
 - flightboard-private.h, [62](#)
- ELEVATOR_PIN_PHYSICAL
 - flightboard-private.h, [62](#)
- ELEVATOR_SCALE
 - flightboard-private.h, [62](#)
- elevator
 - picopter::FlightData, [25](#)
- emulation/gps-emu/libgpsmm.h, [47](#)
- emulation/wiringPi.h, [48](#)
- emulation/wiringSerial.h, [48](#)
- err
 - picopter::GPSData, [30](#)
- FIX_TIMEOUT_DEFAULT
 - picopter::GPS, [29](#)
- FatalEx
 - log.cpp, [67](#)
 - log.h, [55](#)
- fb
 - picopter::FlightController, [25](#)
- fix
 - picopter::GPSData, [30](#)
- FlightBoard
 - picopter::FlightBoard, [20](#)
- FlightController
 - picopter::FlightController, [22](#)
- flightboard-private.h
 - AILERON_CHANNEL, [61](#)
 - AILERON_HIGH, [61](#)
 - AILERON_LOW, [62](#)
 - AILERON_PIN_PHYSICAL, [62](#)
 - AILERON_SCALE, [62](#)
 - ELEVATOR_CHANNEL, [62](#)
 - ELEVATOR_HIGH, [62](#)
 - ELEVATOR_LOW, [62](#)

- ELEVATOR_PIN_PHYSICAL, [62](#)
- ELEVATOR_SCALE, [62](#)
- GIMBAL_CHANNEL, [62](#)
- GIMBAL_HIGH, [62](#)
- GIMBAL_LOW, [62](#)
- GIMBAL_PIN_PHYSICAL, [62](#)
- GIMBAL_SCALE, [63](#)
- LINEAR_SCALE, [63](#)
- RUDDER_CHANNEL, [63](#)
- RUDDER_HIGH, [63](#)
- RUDDER_LOW, [63](#)
- RUDDER_PIN_PHYSICAL, [63](#)
- RUDDER_SCALE, [63](#)
- SPEED_SCALE, [63](#)
- flightcontroller.cpp
 - InitialiseItem, [64](#)
- GIMBAL_CHANNEL
 - flightboard-private.h, [62](#)
- GIMBAL_HIGH
 - flightboard-private.h, [62](#)
- GIMBAL_LOW
 - flightboard-private.h, [62](#)
- GIMBAL_PIN_PHYSICAL
 - flightboard-private.h, [62](#)
- GIMBAL_SCALE
 - flightboard-private.h, [63](#)
- GPS
 - picopter::GPS, [28](#)
- GPSPSD
 - picopter::GPSPSD, [31](#)
- GPSNaza
 - picopter::GPSNaza, [33](#)
- GetBool
 - picopter::Options, [37](#)
- GetCurrentState
 - picopter::FlightController, [22](#)
- GetCurrentTaskId
 - picopter::FlightController, [22](#)
- GetData
 - picopter::FlightBoard, [20](#)
- GetInt
 - picopter::Options, [38](#)
- GetLatest
 - picopter::GPS, [28](#)
 - picopter::IMU, [34](#)
- GetReal
 - picopter::Options, [38](#)
- GetString
 - picopter::Options, [38](#)
- gimbal
 - picopter::FlightData, [25](#)
- gps
 - picopter::FlightController, [25](#)
- gps_data_t, [29](#)
- gps_data_t::fix, [19](#)
- gpsmm, [32](#)
 - LATLON_SET, [32](#)
- HasFix
 - picopter::GPS, [28](#)
- heading
 - picopter::GPSFix, [30](#)
- IMU
 - picopter::IMU, [33](#)
- imu
 - picopter::FlightController, [25](#)
- include/buzzer.h, [49](#)
- include/camera_stream.h, [49](#)
- include/common.h, [50](#)
- include/config.h, [50](#)
- include/datalog.h, [51](#)
- include/flightboard.h, [51](#)
- include/flightcontroller.h, [51](#)
- include/gpio.h, [52](#)
- include/gps_feed.h, [52](#)
- include/gps_gpsd.h, [53](#)
- include/gps_naza.h, [53](#)
- include/imu_feed.h, [54](#)
- include/lawnmower.h, [54](#)
- include/log.h, [54](#)
- include/navigation.h, [56](#)
- include/object_tracker.h, [57](#)
- include/opts.h, [57](#)
- include/picopter.h, [57](#)
- include/waypoints.h, [58](#)
- InferBearing
 - picopter::FlightController, [22](#)
- Init
 - picopter::gpio, [11](#)
- InitialiseItem
 - flightcontroller.cpp, [64](#)
- IsAutoMode
 - picopter::gpio, [11](#)
- LATLON_SET
 - gpsmm, [32](#)
- LINEAR_SCALE
 - flightboard-private.h, [63](#)
- lat
 - picopter::GPSFix, [30](#)
 - picopter::navigation::Coord2D, [16](#)
 - picopter::navigation::Coord3D, [17](#)
- log.cpp
 - FatalEx, [67](#)
 - LogEx, [67](#)
 - LogInit, [67](#)
 - LogSimple, [67](#)
- log.h
 - FatalEx, [55](#)
 - LogEx, [55](#)
 - LogInit, [55](#)
 - LogSimple, [55](#)
- LogEx
 - log.cpp, [67](#)
 - log.h, [55](#)
- LogInit

- log.cpp, 67
- log.h, 55
- LogSimple
 - log.cpp, 67
 - log.h, 55
- lon
 - picopter::GPSFix, 31
 - picopter::navigation::Coord2D, 16
 - picopter::navigation::Coord3D, 17
- MODE_PIN
 - picopter::gpio, 12
- magnitude
 - picopter::navigation::Point2D, 43
 - picopter::navigation::Point3D, 44
- main
 - naza_decoder.cpp, 59
- navigation.h
 - RADIUS_OF_EARTH, 57
- NavigationTest, 34
- naza_decoder.cpp
 - decodeMessage, 59
 - main, 59
- NazaDecoderLib, 35
- operator Coord2D
 - picopter::navigation::Coord3D, 16
- operator Point2D
 - picopter::navigation::Point3D, 44
- operator<<
 - picopter::FlightController, 24
- Options
 - picopter::Options, 37
- OptionsTest, 40
- PID
 - picopter::PID, 41
- picopter.cpp
 - terminate, 69
- picopter::Buzzer, 13
 - ~Buzzer, 13
 - Buzzer, 13
 - Play, 13
 - PlayWait, 13
 - Stop, 14
- picopter::CamWindow, 15
- picopter::CameraStream, 14
 - SetArrow, 15
- picopter::DataLog, 17
 - ~DataLog, 17
 - DataLog, 17
 - PlainWrite, 18
 - Write, 18
- picopter::FlightBoard, 19
 - ~FlightBoard, 20
 - FlightBoard, 20
 - GetData, 20
 - SetAileron, 20
 - SetData, 20
 - SetElevator, 20
 - SetGimbal, 21
 - SetRudder, 21
 - Stop, 21
- picopter::FlightController, 21
 - ~FlightController, 22
 - buzzer, 25
 - cam, 25
 - CheckForStop, 22
 - fb, 25
 - FlightController, 22
 - GetCurrentState, 22
 - GetCurrentTaskId, 22
 - gps, 25
 - imu, 25
 - InferBearing, 22
 - operator<<, 24
 - RunTask, 24
 - Sleep, 24
 - Stop, 24
 - WaitForAuth, 24
- picopter::FlightData, 25
 - aileron, 25
 - elevator, 25
 - gimbal, 25
 - rudder, 26
- picopter::FlightTask, 26
 - ~FlightTask, 26
 - Run, 26
 - SetCurrentState, 27
- picopter::GPS, 27
 - ~GPS, 28
 - FIX_TIMEOUT_DEFAULT, 29
 - GPS, 28
 - GetLatest, 28
 - HasFix, 28
 - TimeSinceLastFix, 28
 - WAIT_PERIOD, 29
 - WaitForFix, 28
- picopter::GPSData, 29
 - err, 30
 - fix, 30
 - timestamp, 30
- picopter::GPSFix, 30
 - alt, 30
 - bearing, 30
 - heading, 30
 - lat, 30
 - lon, 31
 - speed, 31
- picopter::GPSGPSD, 31
 - ~GPSGPSD, 32
 - GPSGPSD, 31
- picopter::GPSNaza, 32
 - ~GPSNaza, 33
 - GPSNaza, 33
- picopter::IMU, 33

- ~IMU, 34
- GetLatest, 34
- IMU, 33
- picopter::Lawnmower, 34
- picopter::ObjectTracker, 36
 - Run, 36
- picopter::Options, 36
 - ~Options, 37
 - GetBool, 37
 - GetInt, 38
 - GetReal, 38
 - GetString, 38
 - Options, 37
 - Remove, 38
 - Save, 39
 - Set, 39
 - SetFamily, 40
- picopter::PID, 40
 - Compute, 41
 - PID, 41
 - Reset, 41
 - SetBias, 41
 - SetInputLimits, 41
 - SetInterval, 42
 - SetMode, 42
 - SetOutputLimits, 42
 - SetProcessValue, 42
 - SetSetPoint, 42
 - SetTunings, 42
- picopter::Waypoints, 45
 - Run, 45
- picopter::gpio, 11
 - BUZZER_PIN, 12
 - Init, 11
 - IsAutoMode, 11
 - MODE_PIN, 12
 - SetBuzzer, 11
- picopter::navigation::Coord2D, 16
 - lat, 16
 - lon, 16
- picopter::navigation::Coord3D, 16
 - alt, 17
 - lat, 17
 - lon, 17
 - operator Coord2D, 16
- picopter::navigation::EulerAngle, 18
 - pitch, 18
 - roll, 18
 - yaw, 19
- picopter::navigation::Point2D, 43
 - magnitude, 43
 - x, 43
 - y, 43
- picopter::navigation::Point3D, 43
 - magnitude, 44
 - operator Point2D, 44
 - x, 44
 - y, 44
 - z, 44
- pitch
 - picopter::navigation::EulerAngle, 18
- PlainWrite
 - picopter::DataLog, 18
- Play
 - picopter::Buzzer, 13
- PlayWait
 - picopter::Buzzer, 13
- RADIUS_OF_EARTH
 - navigation.h, 57
- RUDDER_CHANNEL
 - flightboard-private.h, 63
- RUDDER_HIGH
 - flightboard-private.h, 63
- RUDDER_LOW
 - flightboard-private.h, 63
- RUDDER_PIN_PHYSICAL
 - flightboard-private.h, 63
- RUDDER_SCALE
 - flightboard-private.h, 63
- Remove
 - picopter::Options, 38
- Reset
 - picopter::PID, 41
- roll
 - picopter::navigation::EulerAngle, 18
- rudder
 - picopter::FlightData, 26
- Run
 - picopter::FlightTask, 26
 - picopter::ObjectTracker, 36
 - picopter::Waypoints, 45
- RunTask
 - picopter::FlightController, 24
- SPEED_SCALE
 - flightboard-private.h, 63
- Save
 - picopter::Options, 39
- Set
 - picopter::Options, 39
- SetAileron
 - picopter::FlightBoard, 20
- SetArrow
 - picopter::CameraStream, 15
- SetBias
 - picopter::PID, 41
- SetBuzzer
 - picopter::gpio, 11
- SetCurrentState
 - picopter::FlightTask, 27
- SetData
 - picopter::FlightBoard, 20
- SetElevator
 - picopter::FlightBoard, 20
- SetFamily
 - picopter::Options, 40

- SetGimbal
 - picopter::FlightBoard, [21](#)
- SetInputLimits
 - picopter::PID, [41](#)
- SetInterval
 - picopter::PID, [42](#)
- SetMode
 - picopter::PID, [42](#)
- SetOutputLimits
 - picopter::PID, [42](#)
- SetProcessValue
 - picopter::PID, [42](#)
- SetRudder
 - picopter::FlightBoard, [21](#)
- SetSetPoint
 - picopter::PID, [42](#)
- SetTunings
 - picopter::PID, [42](#)
- Sleep
 - picopter::FlightController, [24](#)
- speed
 - picopter::GPSFix, [31](#)
- src/apps/naza_decoder.cpp, [58](#)
- src/apps/speedcal.cpp, [59](#)
- src/base/buzzer.cpp, [59](#)
- src/base/camera_stream.cpp, [60](#)
- src/base/datalog.cpp, [60](#)
- src/base/flightboard-private.h, [61](#)
- src/base/flightboard.cpp, [63](#)
- src/base/flightcontroller.cpp, [64](#)
- src/base/gpio.cpp, [65](#)
- src/base/gps.cpp, [65](#)
- src/base/gps_gpsd.cpp, [65](#)
- src/base/gps_naza.cpp, [66](#)
- src/base/imu.cpp, [66](#)
- src/base/log.cpp, [66](#)
- src/base/opts.cpp, [68](#)
- src/modules/object_tracker.cpp, [68](#)
- src/modules/waypoints.cpp, [68](#)
- src/server/picopter.cpp, [69](#)
- Stop
 - picopter::Buzzer, [14](#)
 - picopter::FlightBoard, [21](#)
 - picopter::FlightController, [24](#)
- terminate
 - picopter.cpp, [69](#)
- TimeSinceLastFix
 - picopter::GPS, [28](#)
- timestamp
 - picopter::GPSData, [30](#)
- vec2, [44](#)
- WAIT_PERIOD
 - picopter::GPS, [29](#)
- WaitForAuth
 - picopter::FlightController, [24](#)
- WaitForFix
 - picopter::GPS, [28](#)
- webInterfaceHandler, [45](#)
- Write
 - picopter::DataLog, [18](#)
- x
 - picopter::navigation::Point2D, [43](#)
 - picopter::navigation::Point3D, [44](#)
- y
 - picopter::navigation::Point2D, [43](#)
 - picopter::navigation::Point3D, [44](#)
- yaw
 - picopter::navigation::EulerAngle, [19](#)
- z
 - picopter::navigation::Point3D, [44](#)