Introduction Flight Firmware: rc_pilot

Yevhenii Kovryzhenko

Advisor: Dr. Ehsan Taheri

Department of Aerospace Engineering

Auburn University

AUBURN UNIVERSITY

Aerospace Engineering

ACE LAB

Outline

- **Introductory Concepts**
  - Naming conventions and nomenclature
  - Real time computing and data availability issue
  - Clock, update frequency, threads and sequential processes

- **Flight firmware overview: rc_pilot**
  - High level overview of the algorithm
    - Overall flow of information
  - Mid-level overview:
    - Guidance
    - Feedback control
    - Communication
  - Low level overview:
    - Hardware interfaces
    - Communication protocols

- **Conclusion and Future Work**

Introduction to Embedded Systems Designed to Work in Fast-Paced Environments

❑ **Embedded System** is a combination of software and hardware designed to execute a specific task. Embedded systems designed for fast paced environment must execute their task in sufficiently short period of time. In the context of this presentation, a flight control system is expected to operate with update frequency of **100+ Hz.** Nominal mode of operation for rc_pilot has a set update frequency of 200Hz.

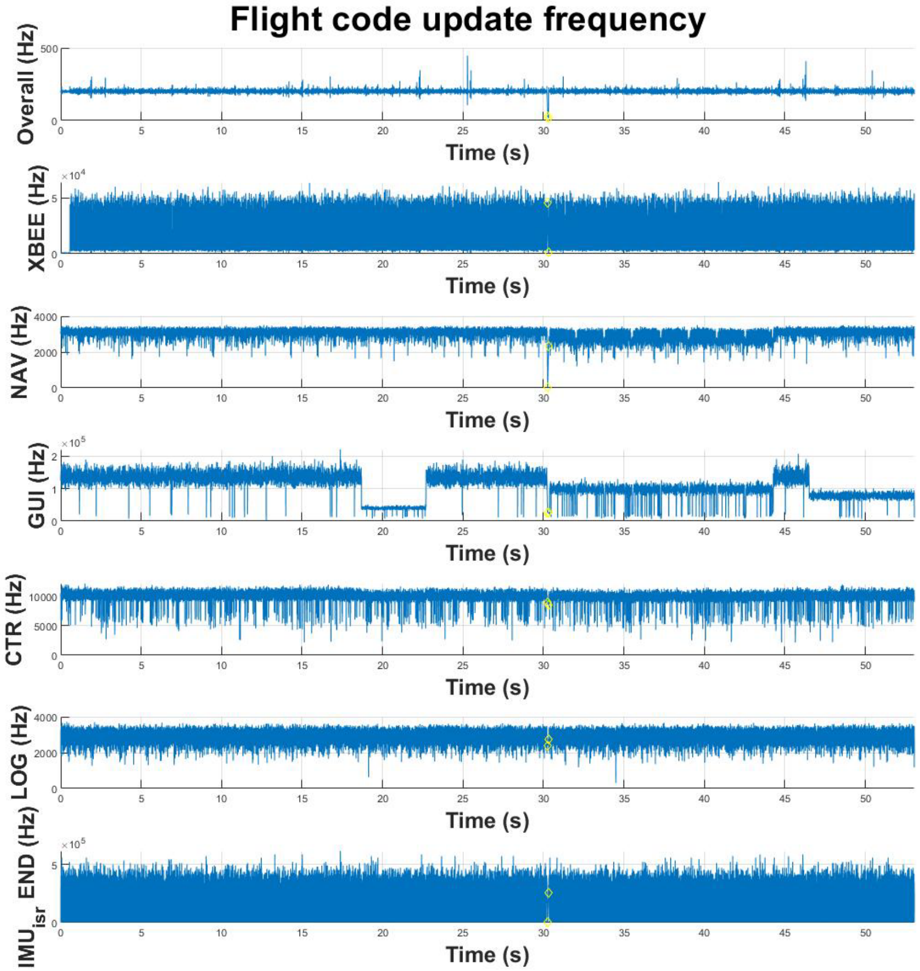
❑ **Update frequency** in the context of an embedded system defines 1/Time it takes to evaluate one complete operational loop under nominal mode of operations.

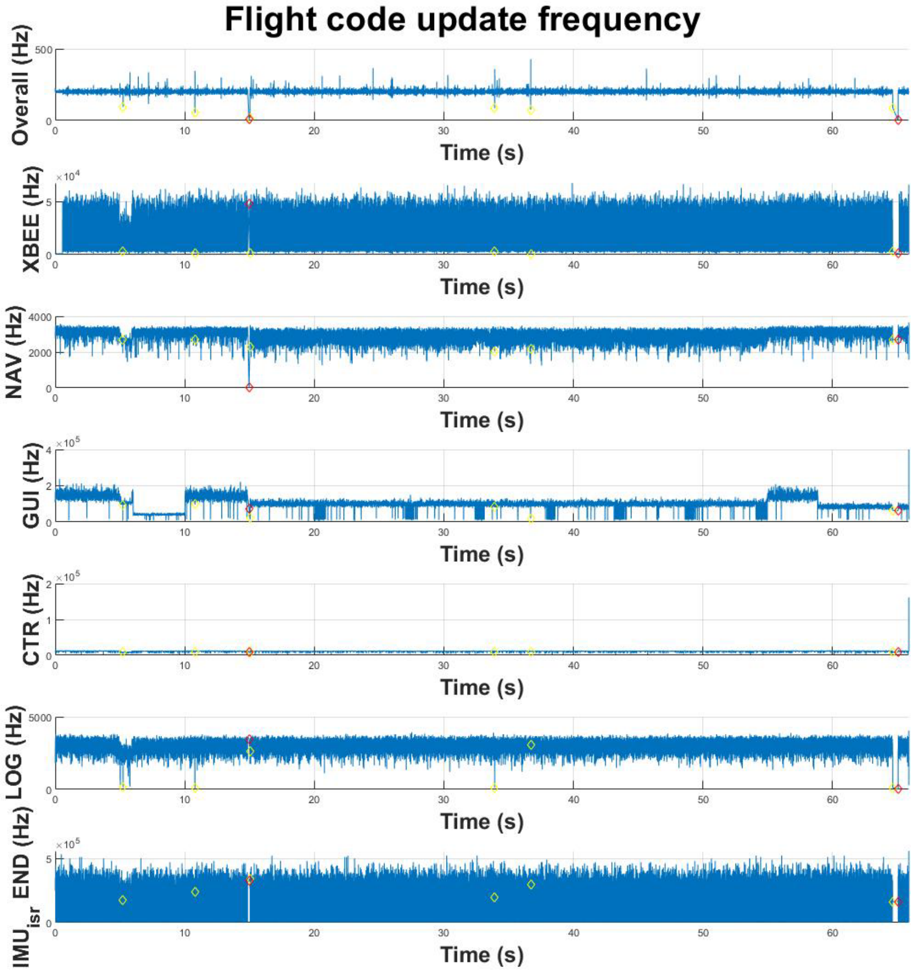
❑ **Operational loop** or main set of tasks defines all the required subroutines, evaluations, function calls, etc. embedded system must execute to complete one complete cycle of nominal mode of operations.

❑ **Nominal mode of operations** or expected/standard set of routines the system must undergo to fulfill its task. This excludes startup routines, termination routines, emergency or any “off-nominal” events.

❑ **Thread** in the context of computer program, is usually defined as a small set of instructions designed to be scheduled and executed by CPU independently of the parent process. This allows asynchronous execution of separate tasks irrespective of the other process.

Update Frequency Example: rc_pilot log

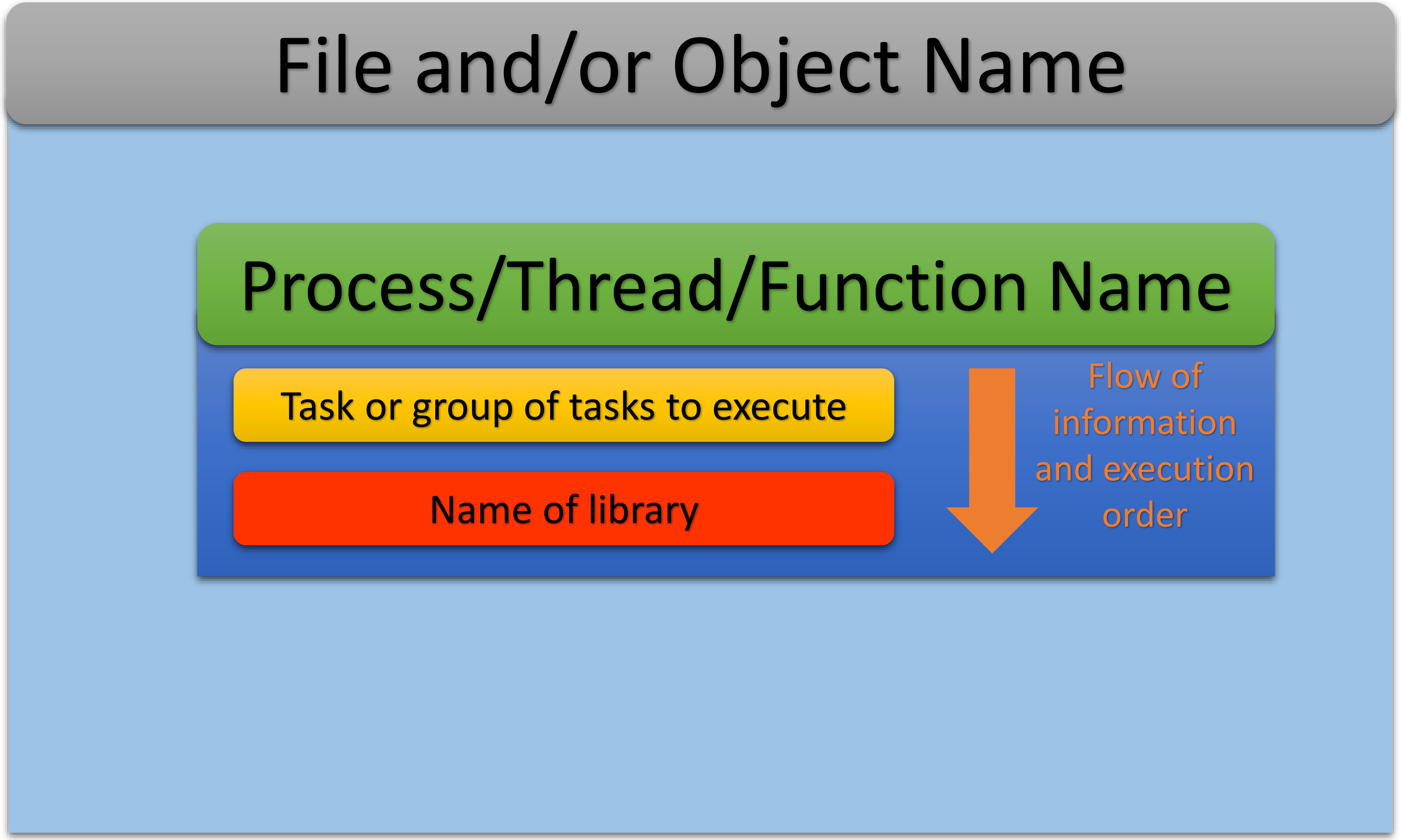




rc_pilot

- ❑Designed to work on BeagleBone Blue Linux Computer with:
  - ❑AM335x 1GHz ARM Cortex-A8 processor
  - ❑512MB DDR3 RAM
  - ❑4GB 8-bit eMMC flash storage

Color Code and Notations



File and/or Object Name

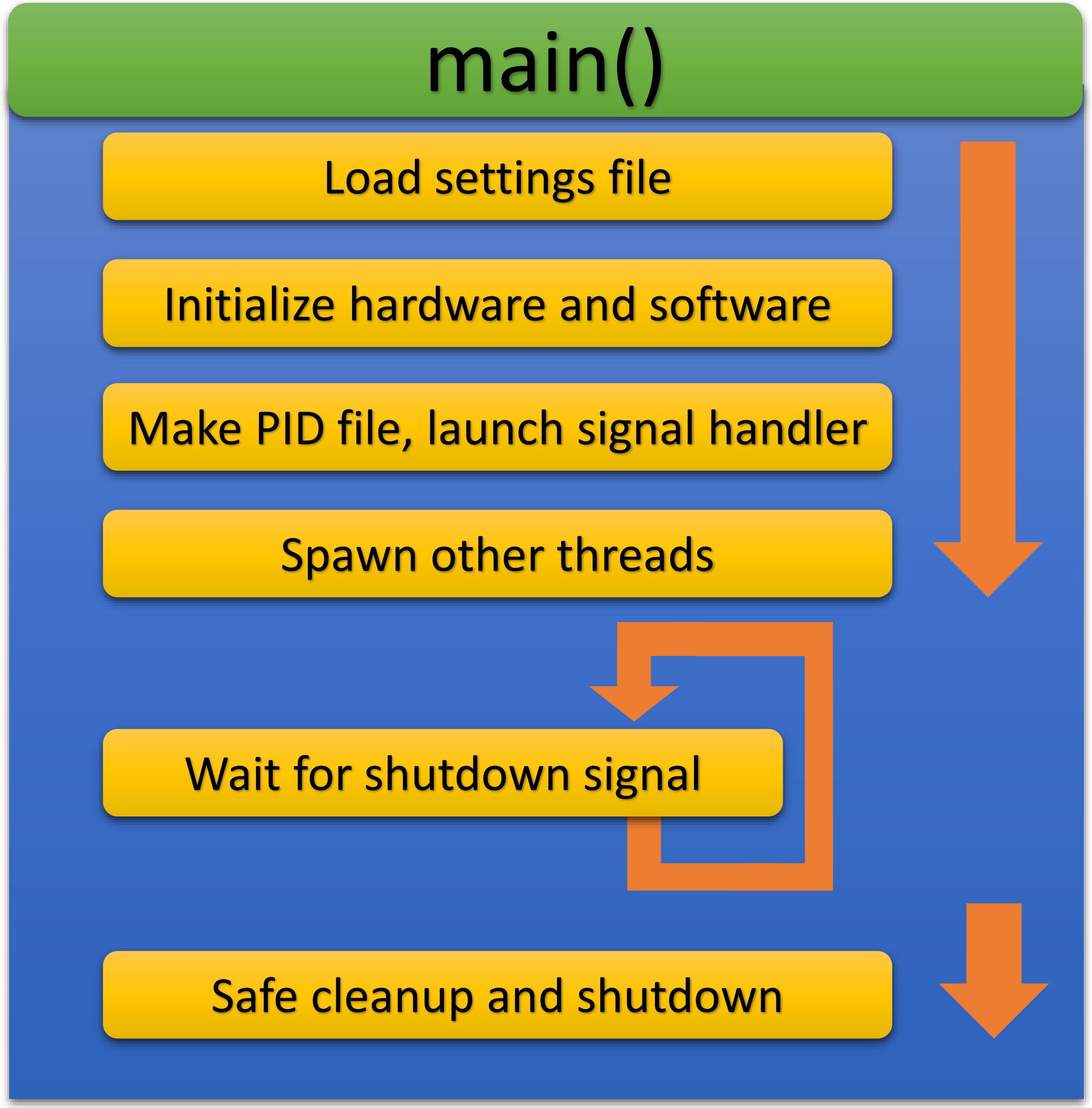
Process/Thread/Function Name

Task or group of tasks to execute

Name of library

Flow of information and execution order

High Level Overview: main thread


main()
Load settings file
Initialize hardware and software
Make PID file, launch signal handler
Spawn other threads
Wait for shutdown signal
Safe cleanup and shutdown

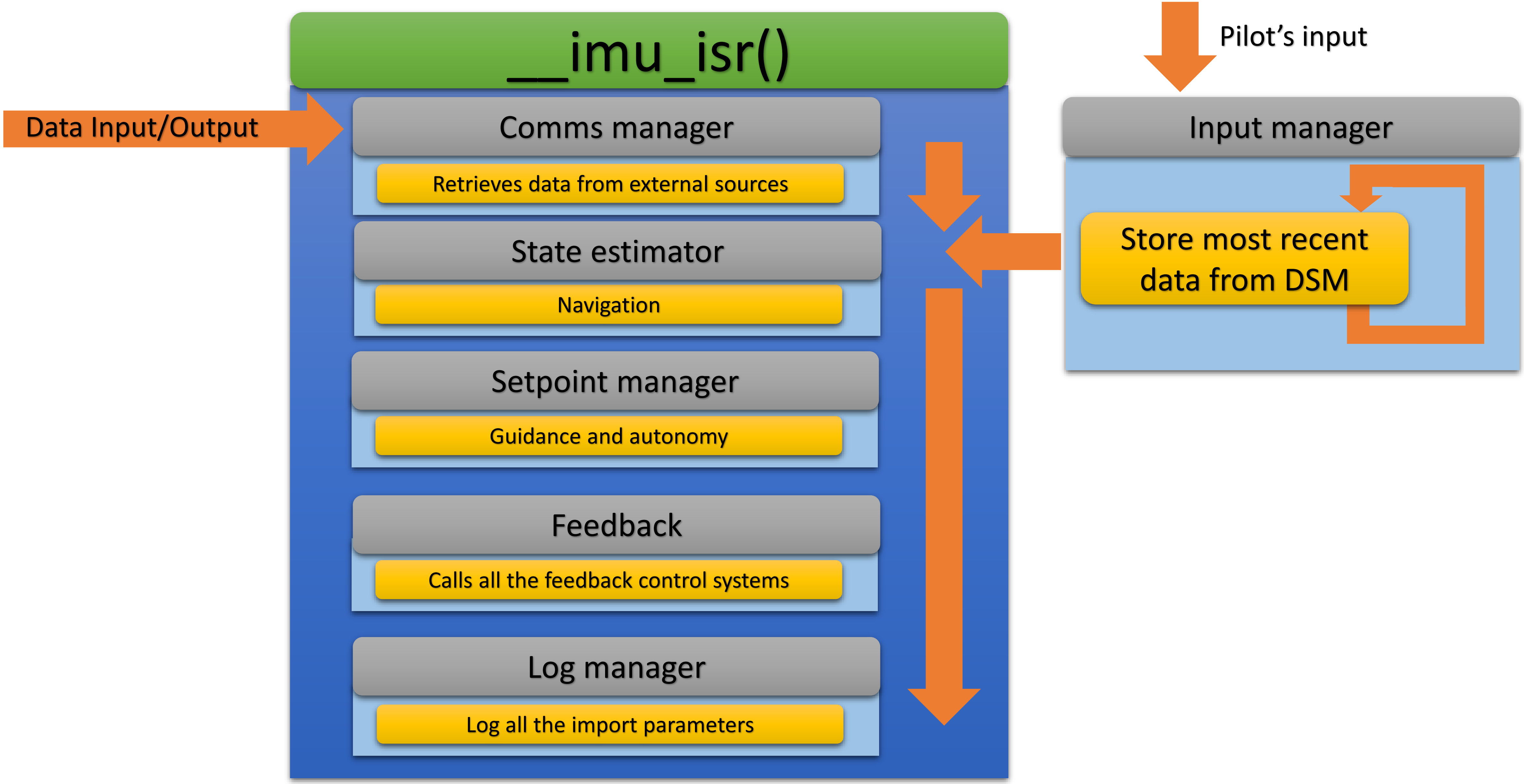
❑Settings file – contains all user-specified not set **on-compile time**

❑Initialization refers to memory allocation and single-run functions

❑PID file (process ID file) and signal handler is required for proper functioning and termination of threads

❑Safe cleanup and shutdown routines free up the allocated memory, disengage all open communication lines and hardware interfaces, etc.

High Level Overview: IMU interrupt routine thread



__imu_isr()

Data Input/Output

Comms manager

Retrieves data from external sources

State estimator

Navigation

Setpoint manager

Guidance and autonomy

Feedback

Calls all the feedback control systems

Log manager

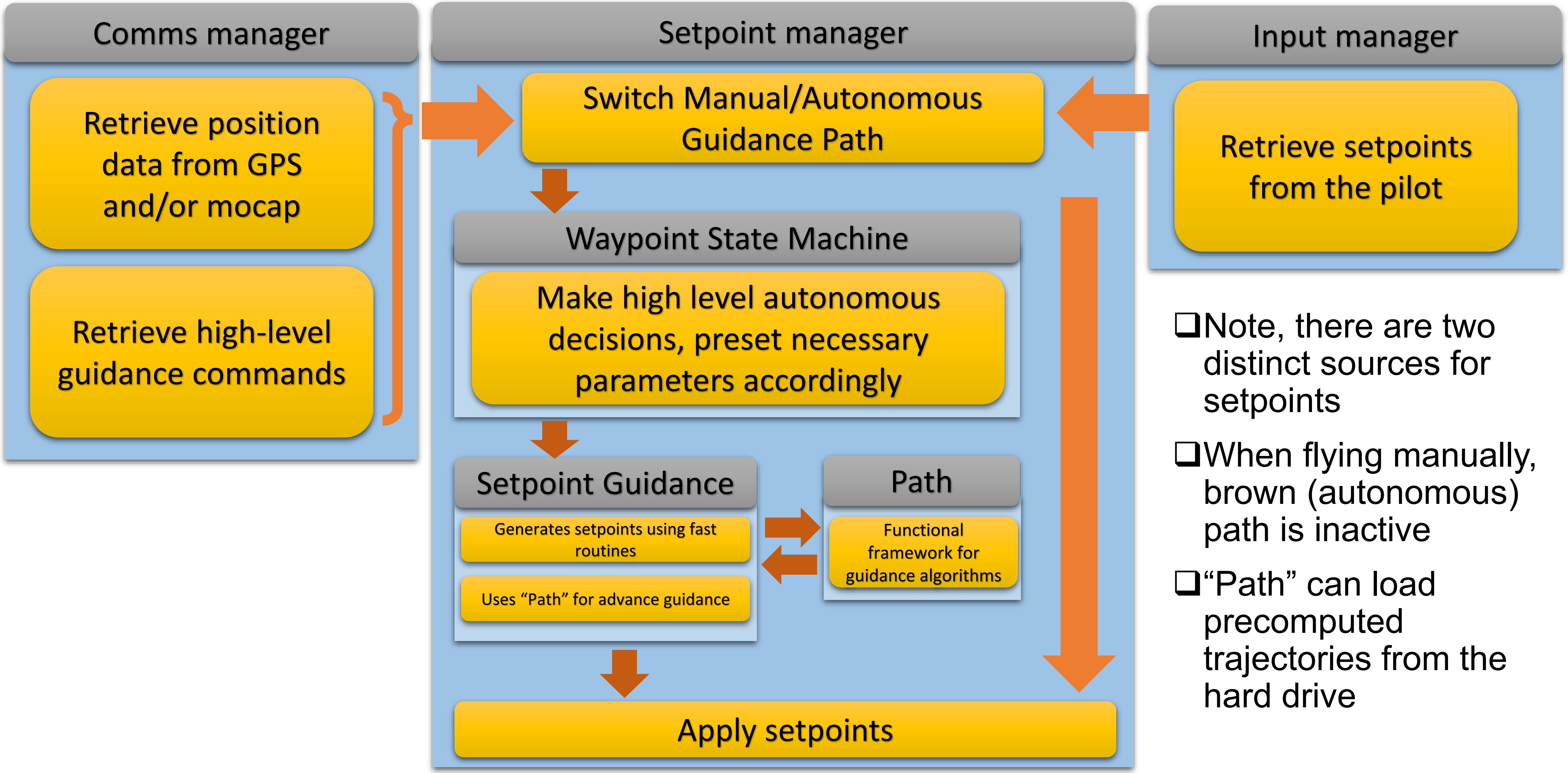
Log all the import parameters

Pilot’s input

Input manager

Store most recent data from DSM

Mid-Level Overview: Guidance



Comms manager

Retrieve position data from GPS and/or mocap

Retrieve high-level guidance commands

Setpoint manager

Switch Manual/Autonomous Guidance Path

Waypoint State Machine

Make high level autonomous decisions, preset necessary parameters accordingly

Setpoint Guidance

Generates setpoints using fast routines

Uses “Path” for advance guidance

Path

Functional framework for guidance algorithms

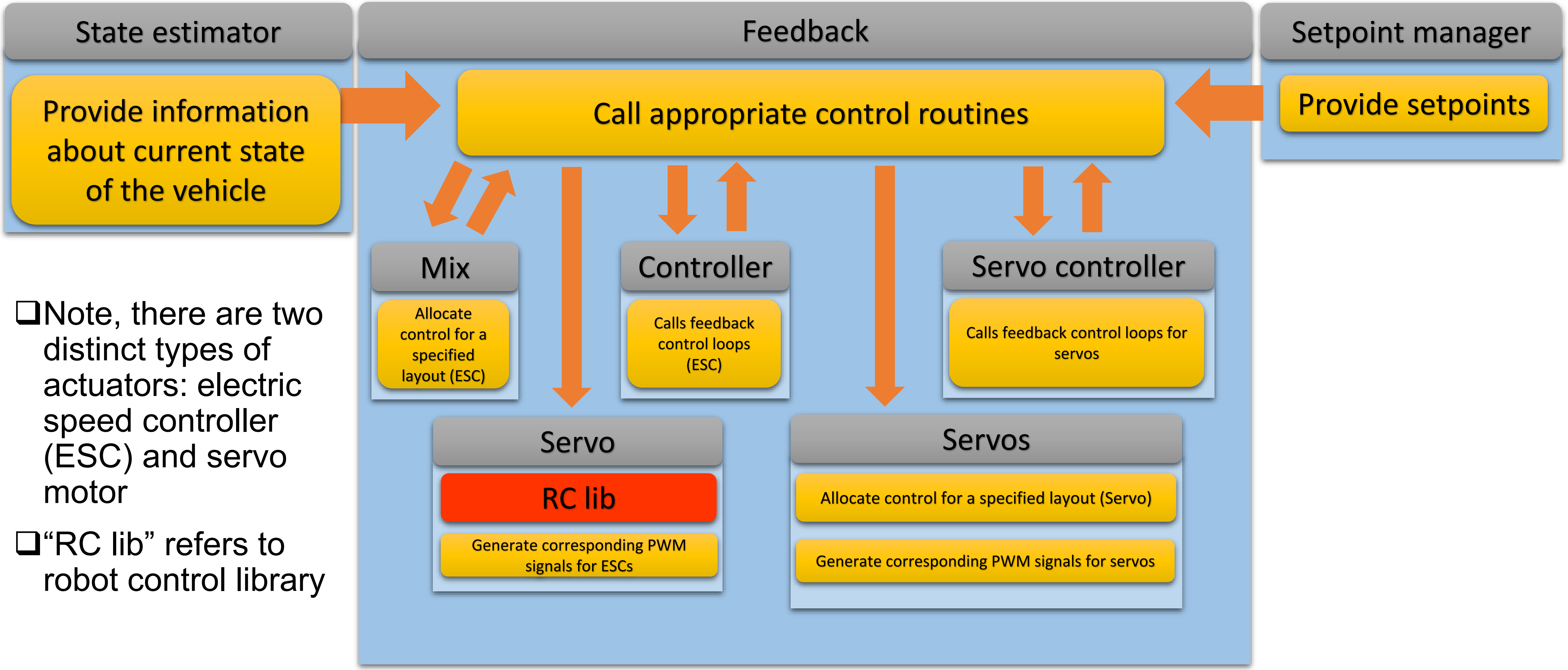
Apply setpoints

Input manager

Retrieve setpoints from the pilot

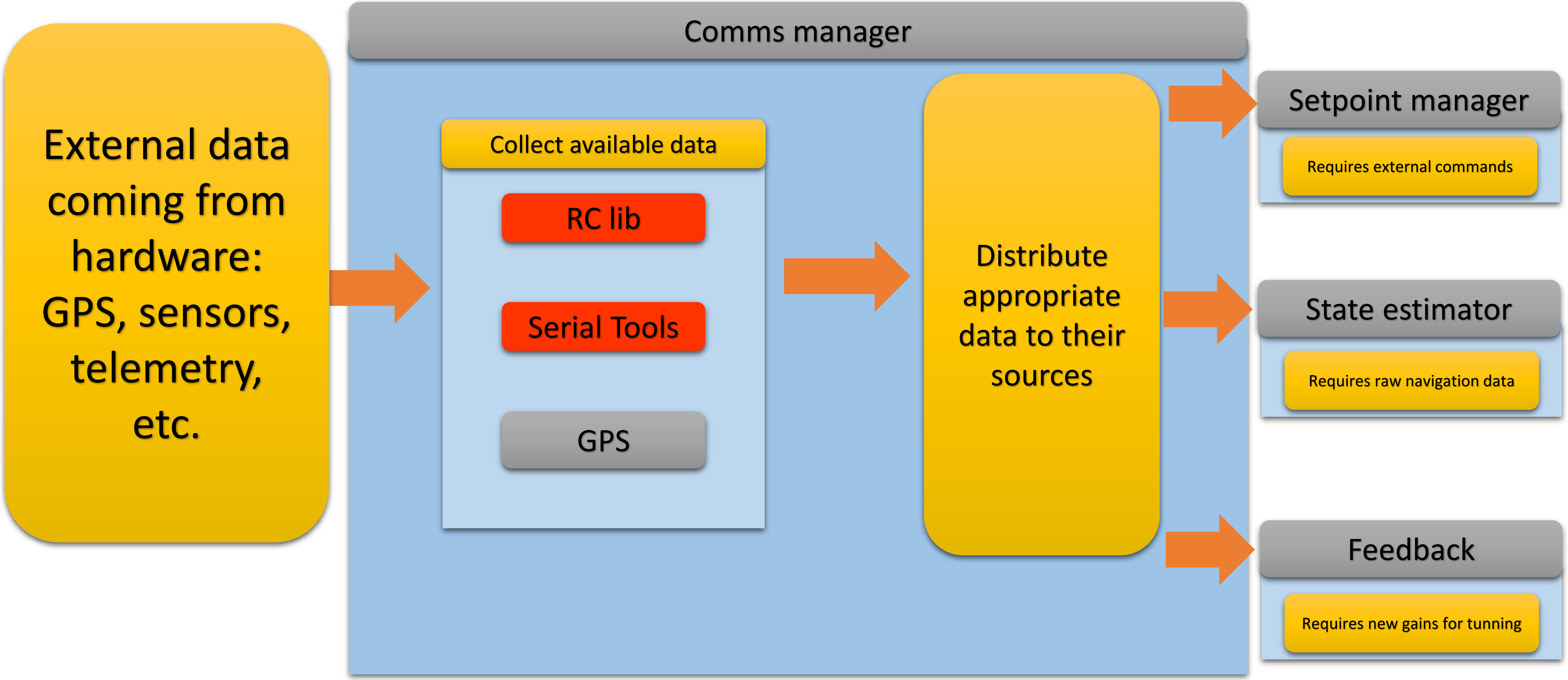
- ❑Note, there are two distinct sources for setpoints
- ❑When flying manually, brown (autonomous) path is inactive
- ❑“Path” can load precomputed trajectories from the hard drive

Mid-Level Overview: Feedback control


State estimator
Provide information about current state of the vehicle
Feedback
Call appropriate control routines
Setpoint manager
Provide setpoints
Mix
Allocate control for a specified layout (ESC)
Controller
Calls feedback control loops (ESC)
Servo controller
Calls feedback control loops for servos
Servo
RC lib
Generate corresponding PWM signals for ESCs
Servos
Allocate control for a specified layout (Servo)
Generate corresponding PWM signals for servos

- ❑ Note, there are two distinct types of actuators: electric speed controller (ESC) and servo motor
- ❑ “RC lib” refers to robot control library

Mid-Level Overview: Comms manager



External data coming from hardware: GPS, sensors, telemetry, etc.

Comms manager

Collect available data

RC lib

Serial Tools

GPS

Distribute appropriate data to their sources

Setpoint manager

Requires external commands

State estimator

Requires raw navigation data

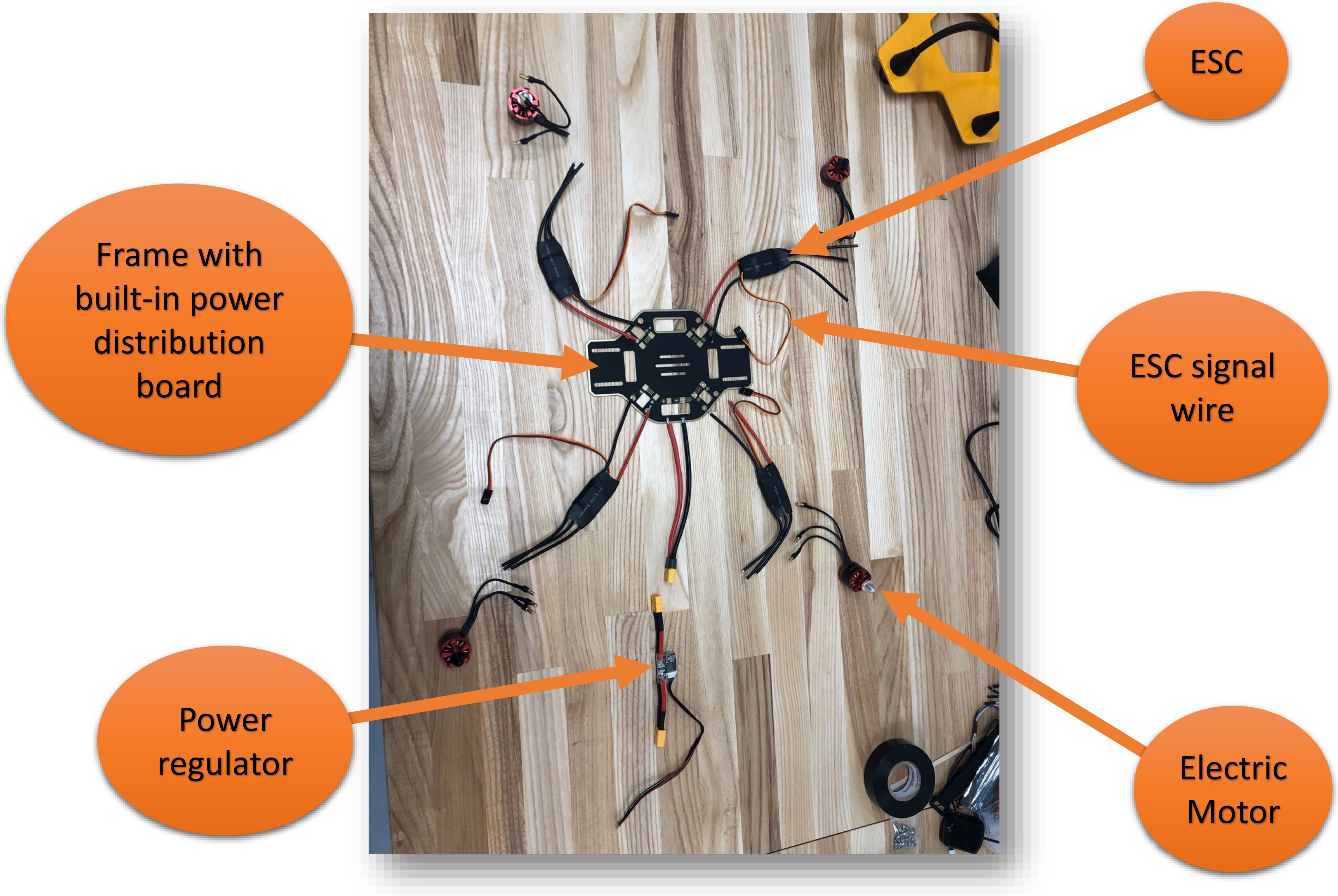
Feedback

Requires new gains for tunning

Low Level Overview: Common communication protocols

- ❑Two-wire protocols:
  - ❑**I2C** – Inter-Integrated Circuit
  - ❑**USART** or **UART** – Universal Synchronous/Asynchronous Receiver/Transmitter
  - ❑CAN – Control Area Network
- ❑Four-wire protocols:
  - ❑SPI – Serial Peripheral Interface
- ❑Others:
  - ❑USB – Universal Serial Bus
  - ❑ETHERNET

Quadrotor Hardware Elements: power and actuation



ESC

Frame with built-in power distribution board

ESC signal wire

Power regulator

Electric Motor

Conclusion & Future Works

- ❑ rc_pilot is a lightweight drone control system capable of operating vehicles having ESCs, servos or both types of actuators.

- ❑ Main advantage: offers a simplified introduction to embedded system implementation for fast-paced environments.

Future work

- ❑ Complete development of critical firmware elements: state estimation, two-way communication with remote gain-tunning and implementation of separate threads for communication and data logging.
- ❑ Increase robustness of the algorithm and improve performance of the control system for the in-door setting (with mocap)
- ❑ Implement advance state estimation and navigation capabilities and enable outdoor flight (addition of GPS, camera vision, extra sensors, etc.)
- ❑ Investigate real-time trajectory planning tools and advanced control systems

Thank You!