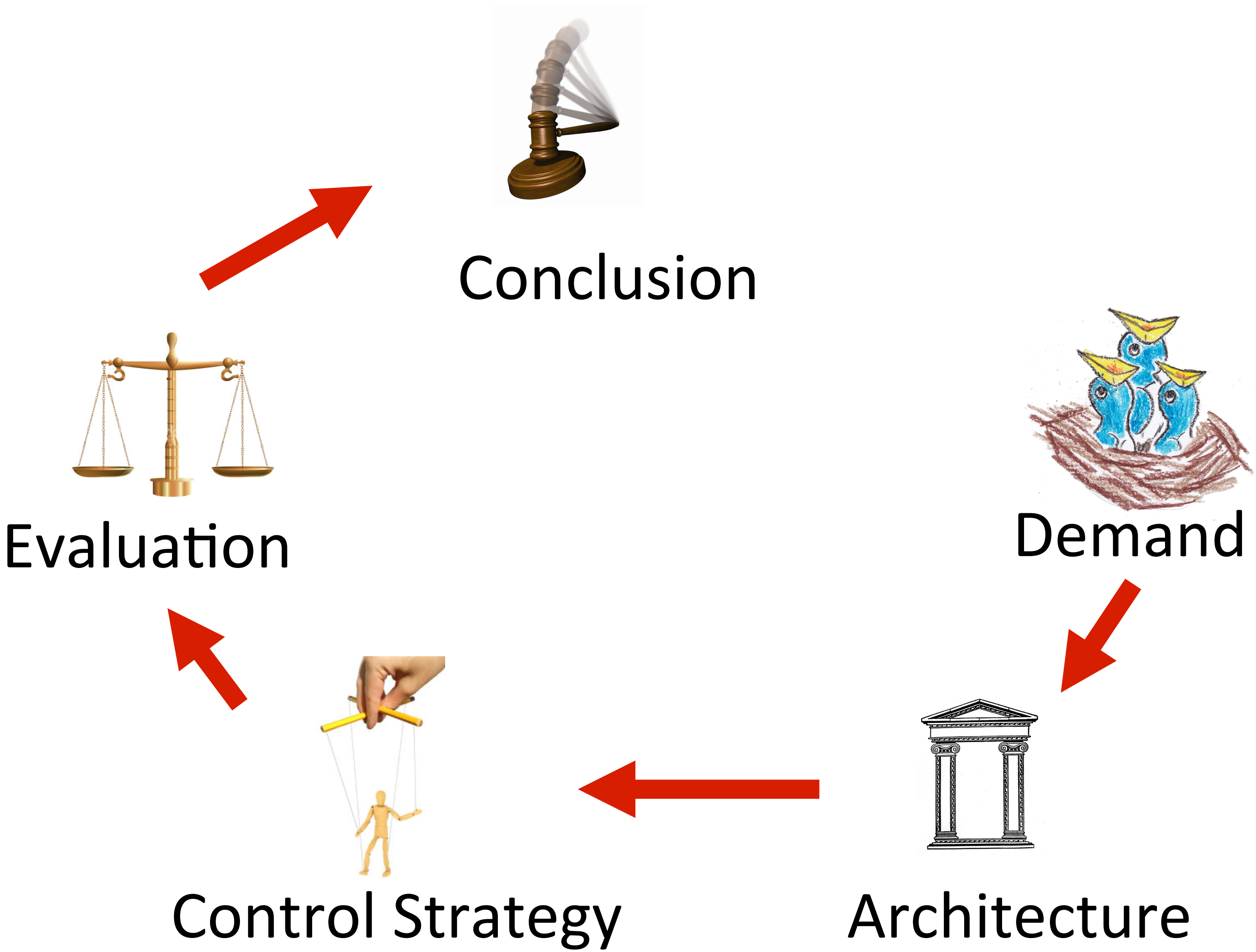
Ard-μ-copter: A Simple Open Source Quadcopter Platform

Zhijian He, Yanming Chen, Zhaoyan Shen, Enyan Huang, Shuai Li, Zili Shao, Qixin Wang
The Embedded Systems and CPS Lab, Dept. of Computing
The Hong Kong Polytechnic Univ.

December, 2015


Conclusion
Demand
Architecture
Control Strategy
Evaluation

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Movie industry

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Movie industry

Surveillance

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Movie industry

Transportation

Surveillance

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Construction site monitoring

Movie industry

Transportation

Surveillance

Demand: increasing usage of UAV (quadcopter) in various domains, and as perfect CPS testbed

Photo/video taking

Virtual reality gaming

Building inspection

Construction site monitoring

Movie industry

Transportation

Surveillance

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Aerodynamics

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Aerodynamics

Mechanics

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Aerodynamics

Mechanics

Robotics/SLAM/Pathplanning

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Aerodynamics

Computer vision

Mechanics

Robotics/SLAM/Pathplanning

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Control theory

Sensing and surveillance

Aerodynamics

Computer vision

Mechanics

Robotics/SLAM/Pathplanning

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Real-time systems

Wireless communication

Control theory

Sensing and surveillance

Aerodynamics

Computer vision

Mechanics

Robotics/SLAM/Pathplanning

Demand: UAV (quadcopter) is a potential perfect CPS testbed

Embedded systems

Mobile ad-hoc networks

Real-time systems

Wireless communication

Control theory

Sensing and surveillance

Aerodynamics

Computer vision

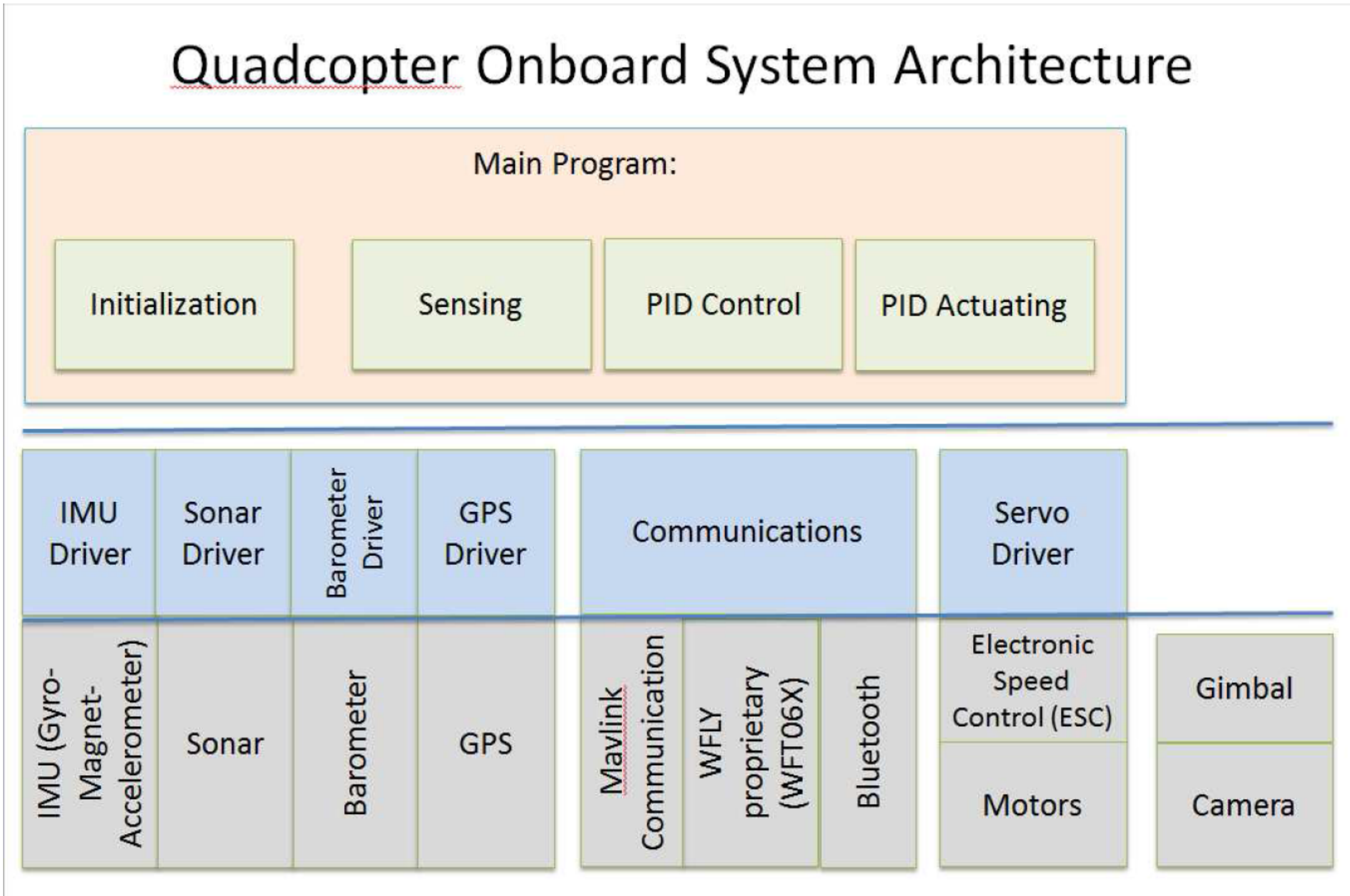
Mechanics

Robotics/SLAM/Pathplanning

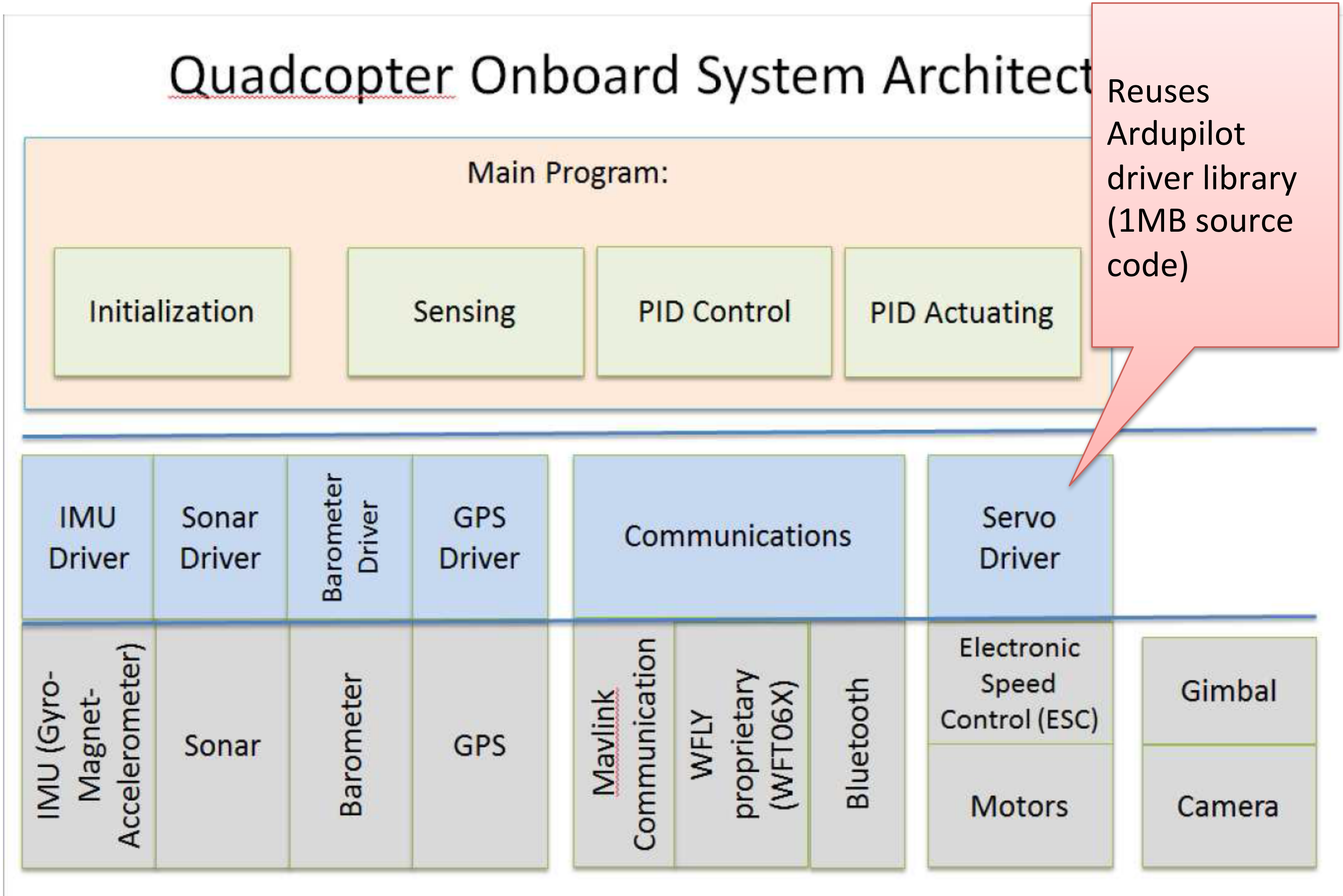
From potential to actual perfect CPS test bed: open source, light weight, good documentation

Product	DJI	MicroPilot	Ardupilot	Ard-µ-copter
Open upper application layer (coarse grain control)	Yes	Yes	Yes	Yes
Open source lower application layer (fine grain control, attitude control)	No	No	Yes (20MB source code)	Yes (135KB source code)
Open source OS/driver layer	No	No	Yes (1MB source code)	Yes (1MB source code)
Open source hardware	No	No	Yes	Yes

Ard-μ-copter architecture

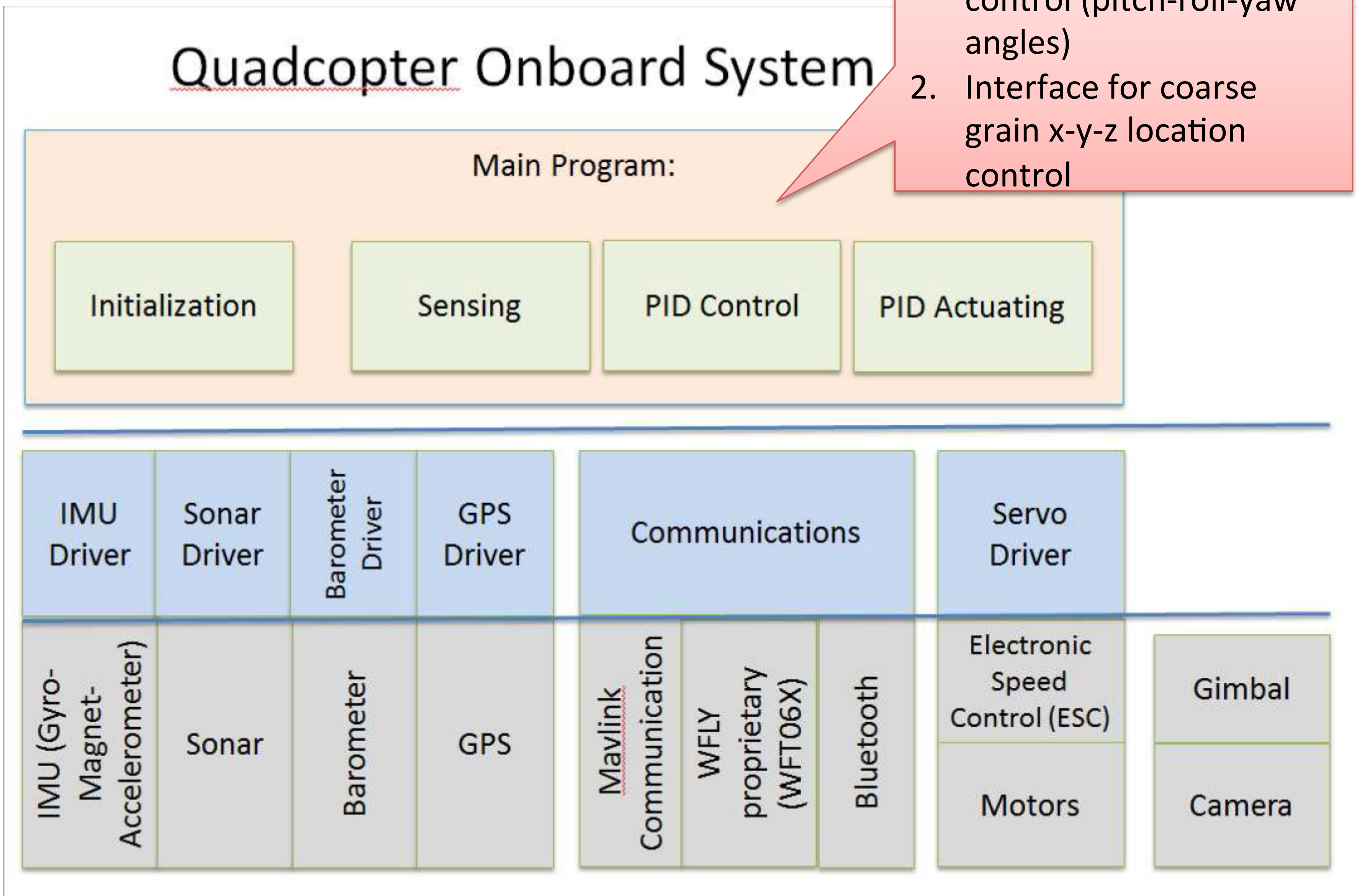


Ard-µ-copter architecture



Ard-µ-copter architect

Supports:
1. Fine grain attitude control (pitch-roll-yaw angles)
2. Interface for coarse grain x-y-z location control



Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

AVR architecture,
8bit, 16MHz,
86 input/output pins,
AD/DA,
UART/USART, SPI,
I2C, EBI/EMI
256MB Flash,
4KB EEPROM, 8K RAM,

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

3-axis gyro,
3-axis accelerometer,
16bit data,
up to 400Hz sampling rate,
SPI comm interface

Additional Digital Motion Processing (DMP) engine to synthesize angle readings

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

Range: 1.3~8 Gauss
Package size: 1.8x1.3cm
IIC comm interface

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

Package size: 15x12x2cm
Weight: 57g
Serial/I2C comm interface

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

24bit AD converter
SPI comm. interface

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

Max range: 7.65m

Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

Usmile Mini OSD MAVLink Flight Communications Module

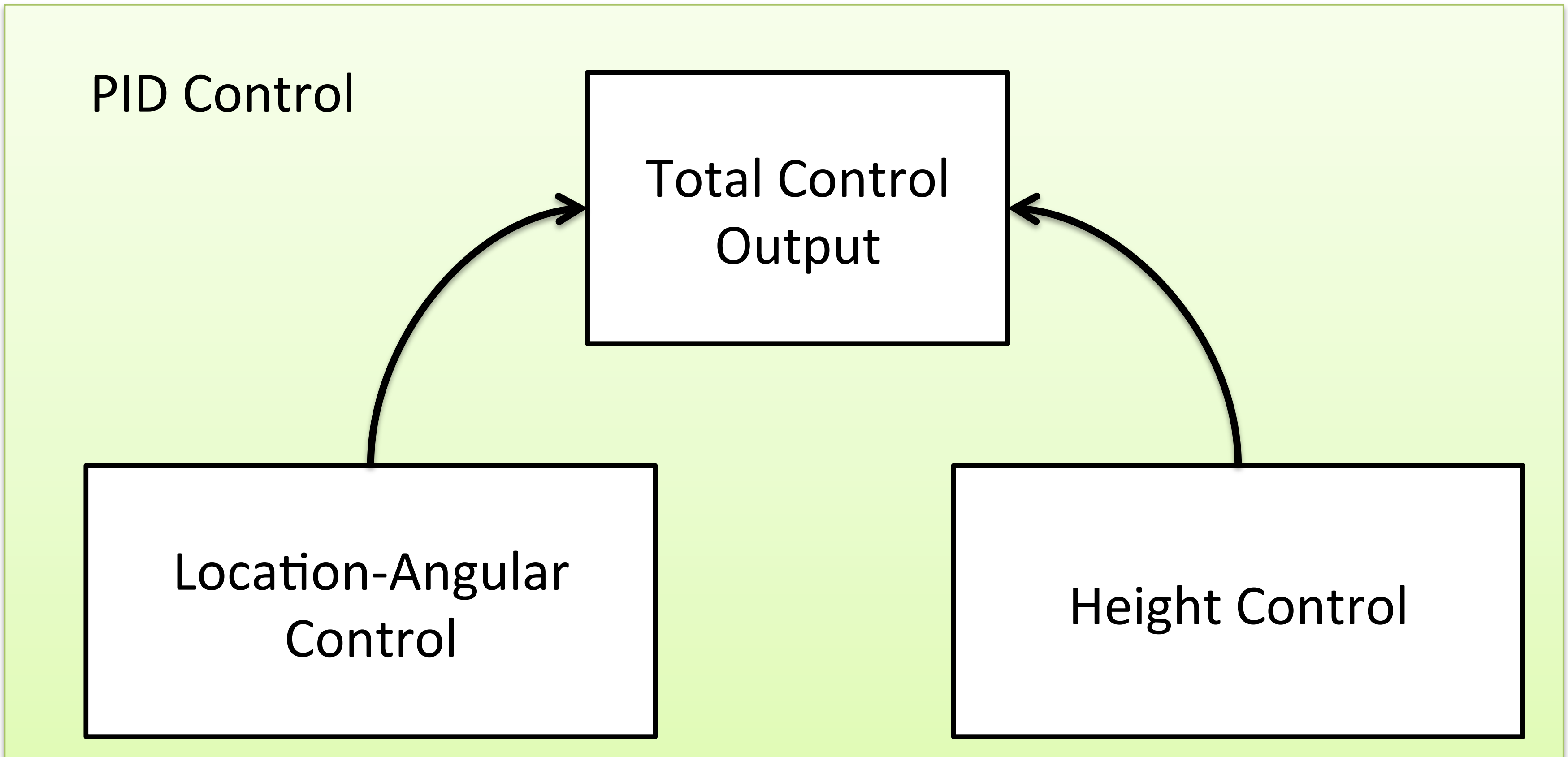
Platform architecture: hardware layer uses the open source ArduPilot Mega 2.5.2 (aka APM 2.5.2) board

MCU	ATmega 2560
IMU	MPU-6050
Magnetometer	HMC5883
GPS	UBLOX-6H
Barometer	MS5611
Sonar	XL-MaxSonar EZ4
Wireless	Usmile Mini 433M

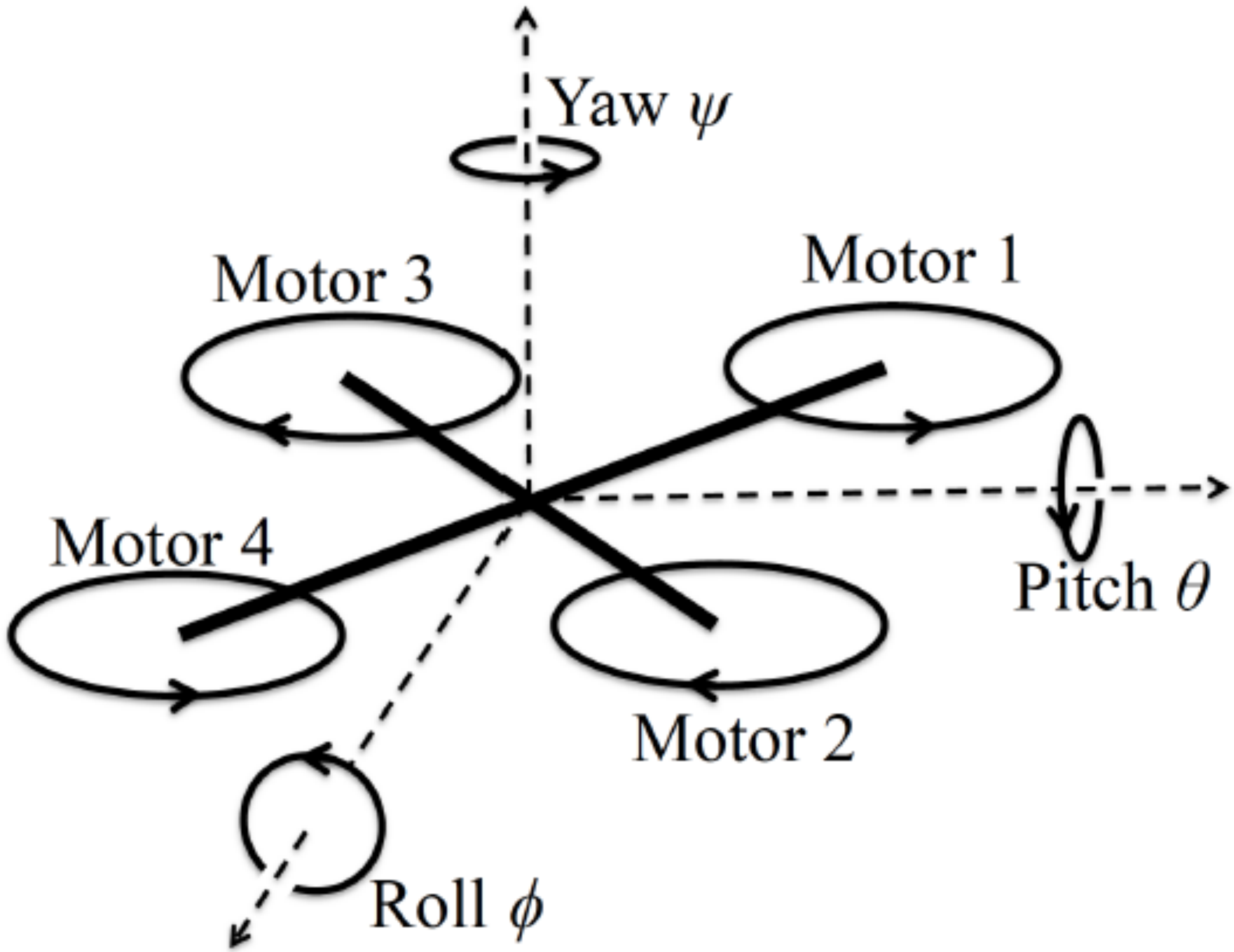
Actuator:

Pulse Width Modulation (PWM) driven motors, duty cycle: 2.5msec (i.e. 400Hz)

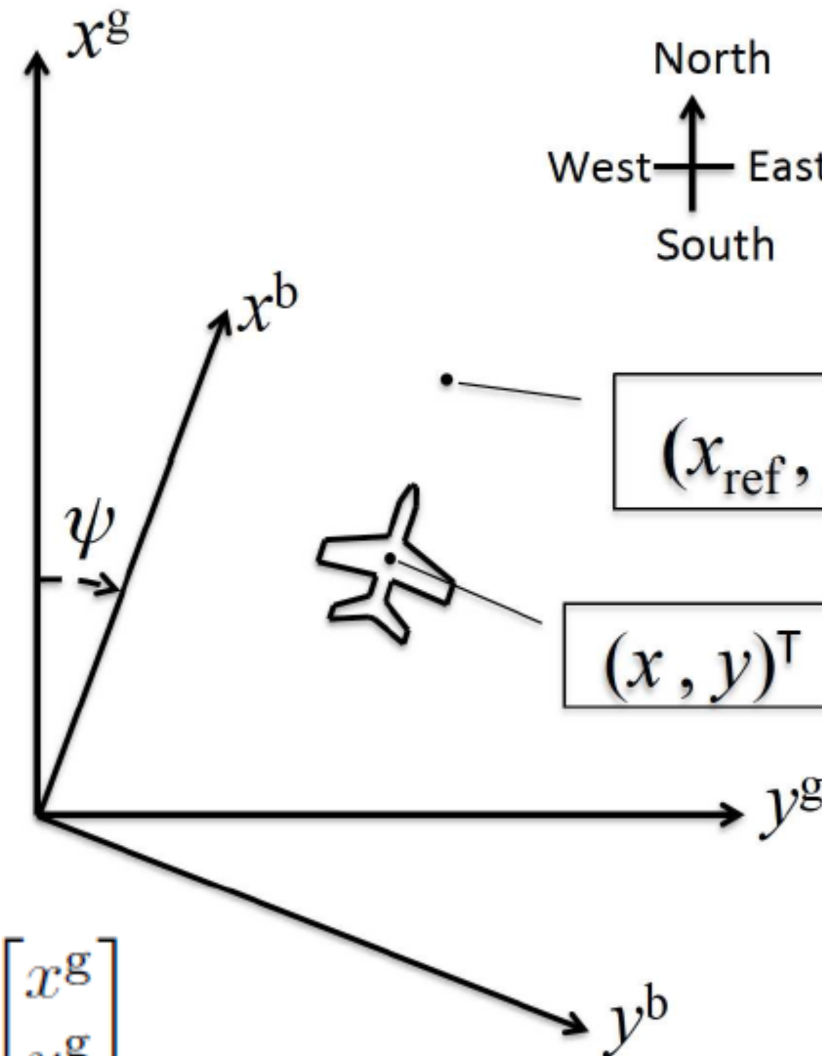
Platform architecture: application layer implementation



Control Strategy: the quadcopter plant and states

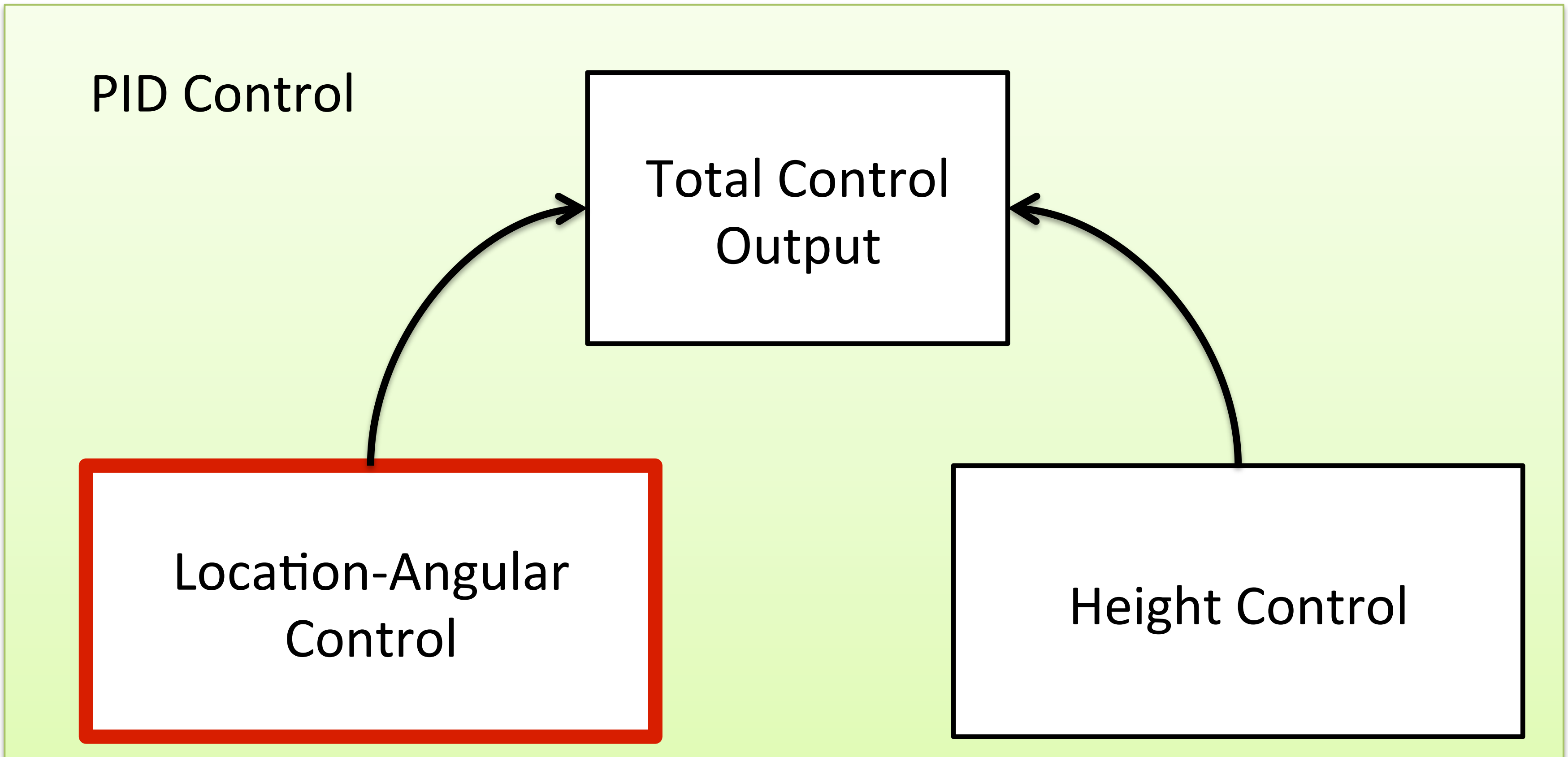


Control Strategy: ground coordinates and body-oriented coordinates



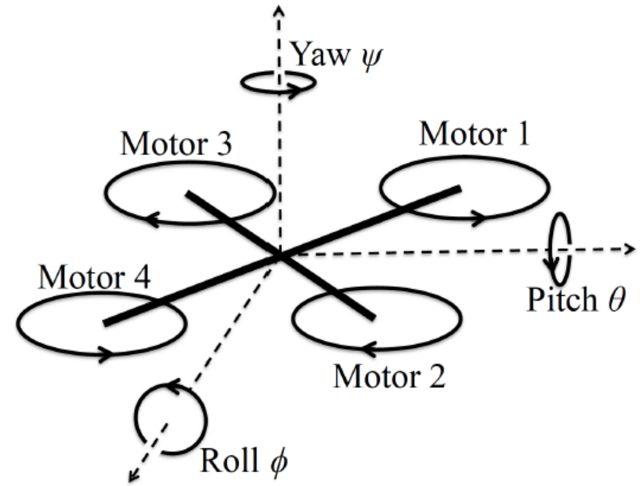
$$\begin{bmatrix} x^{b} \\ y^{b} \end{bmatrix} = \begin{bmatrix} sin\psi & cos\psi \\ cos\psi & -sin\psi \end{bmatrix} \begin{bmatrix} x^{g} \\ y^{g} \end{bmatrix}$$

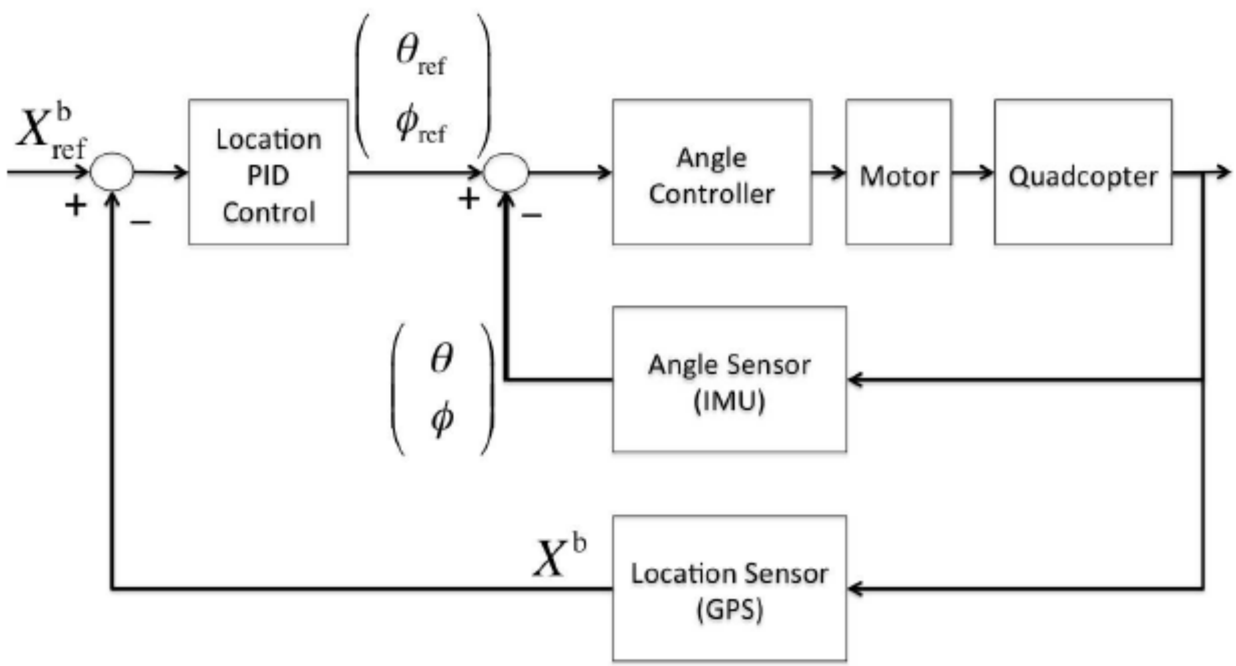
Platform architecture: application layer implementation



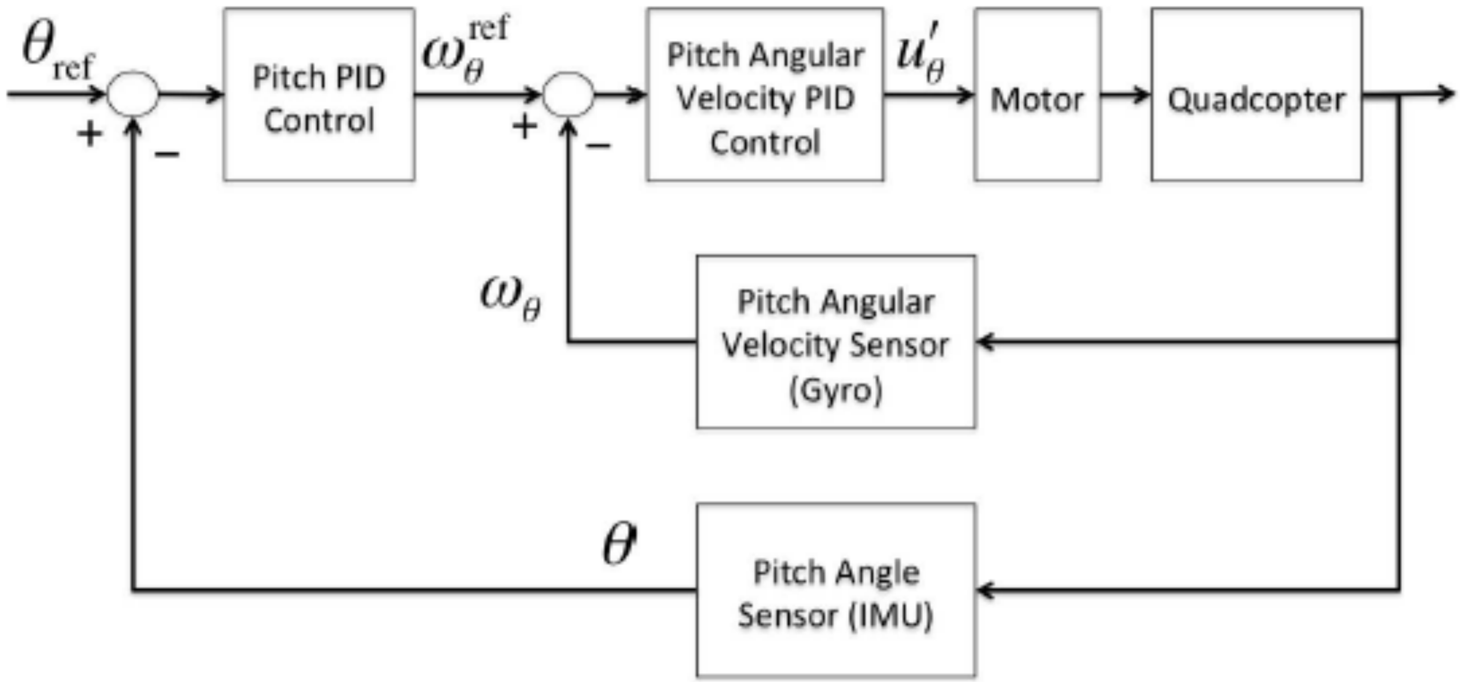
location-angular control takes a nested outer-inner control loop form

$$
\begin{aligned}
\theta_{\text{ref}} &= k_x^{\text{p}}(x_{\text{ref}}^{\text{b}} - x^{\text{b}}) + k_x^{\text{d}}(\dot{x}_{\text{ref}}^{\text{b}} - \dot{x}^{\text{b}}) \\
&\quad + k_x^{\text{i}} \int_0^t (x_{\text{ref}}^{\text{b}}(\tau) - x^{\text{b}}(\tau))\text{d}\tau, \\
\phi_{\text{ref}} &= k_y^{\text{p}}(y_{\text{ref}}^{\text{b}} - y^{\text{b}}) + k_y^{\text{d}}(\dot{y}_{\text{ref}}^{\text{b}} - \dot{y}^{\text{b}}) \\
&\quad + k_y^{\text{i}} \int_0^t (y_{\text{ref}}^{\text{b}}(\tau) - y^{\text{b}}(\tau))\text{d}\tau,
\end{aligned}
$$



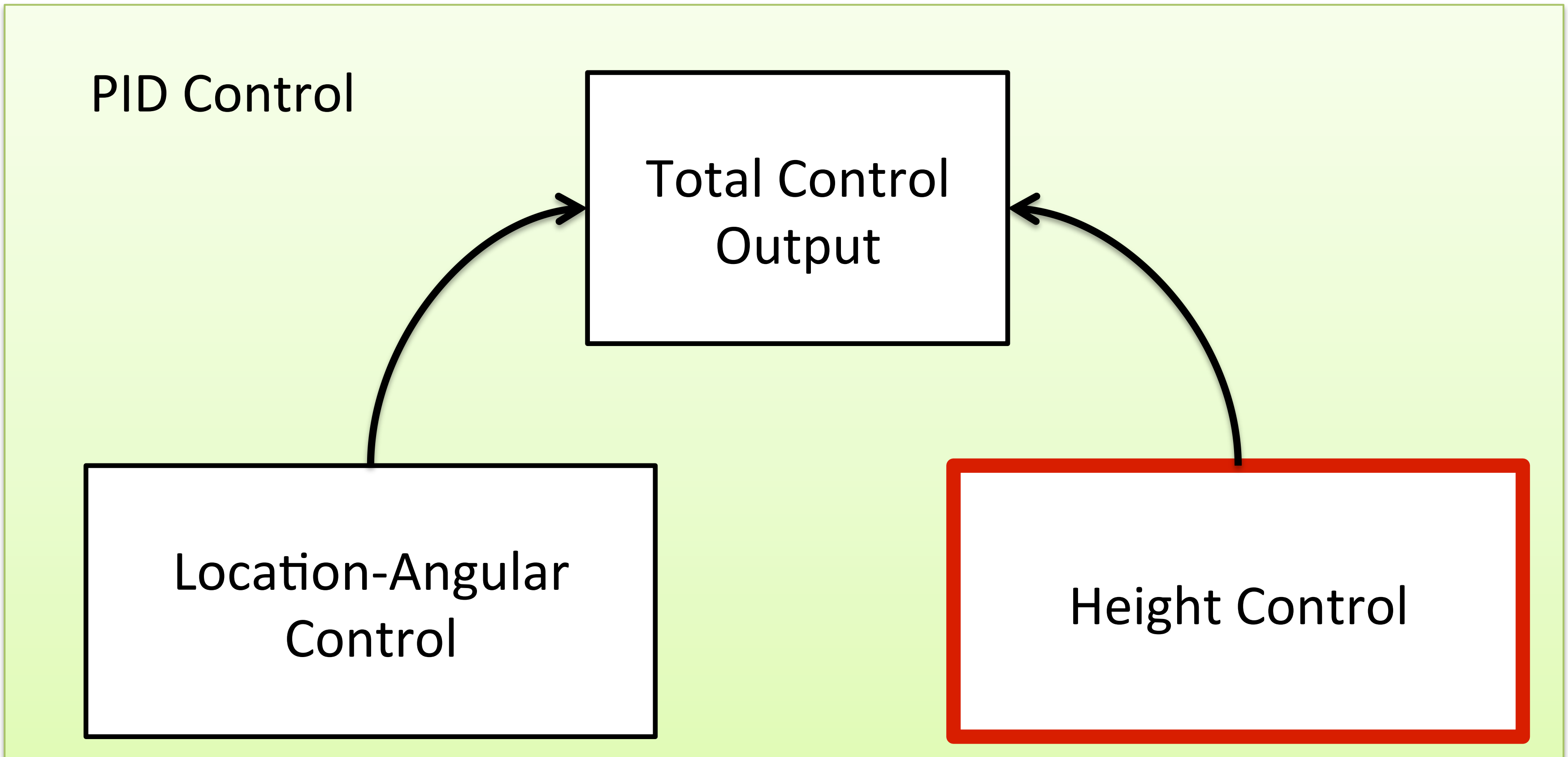


Inner loop: accurate angular velocity readings, inaccurate angle readings

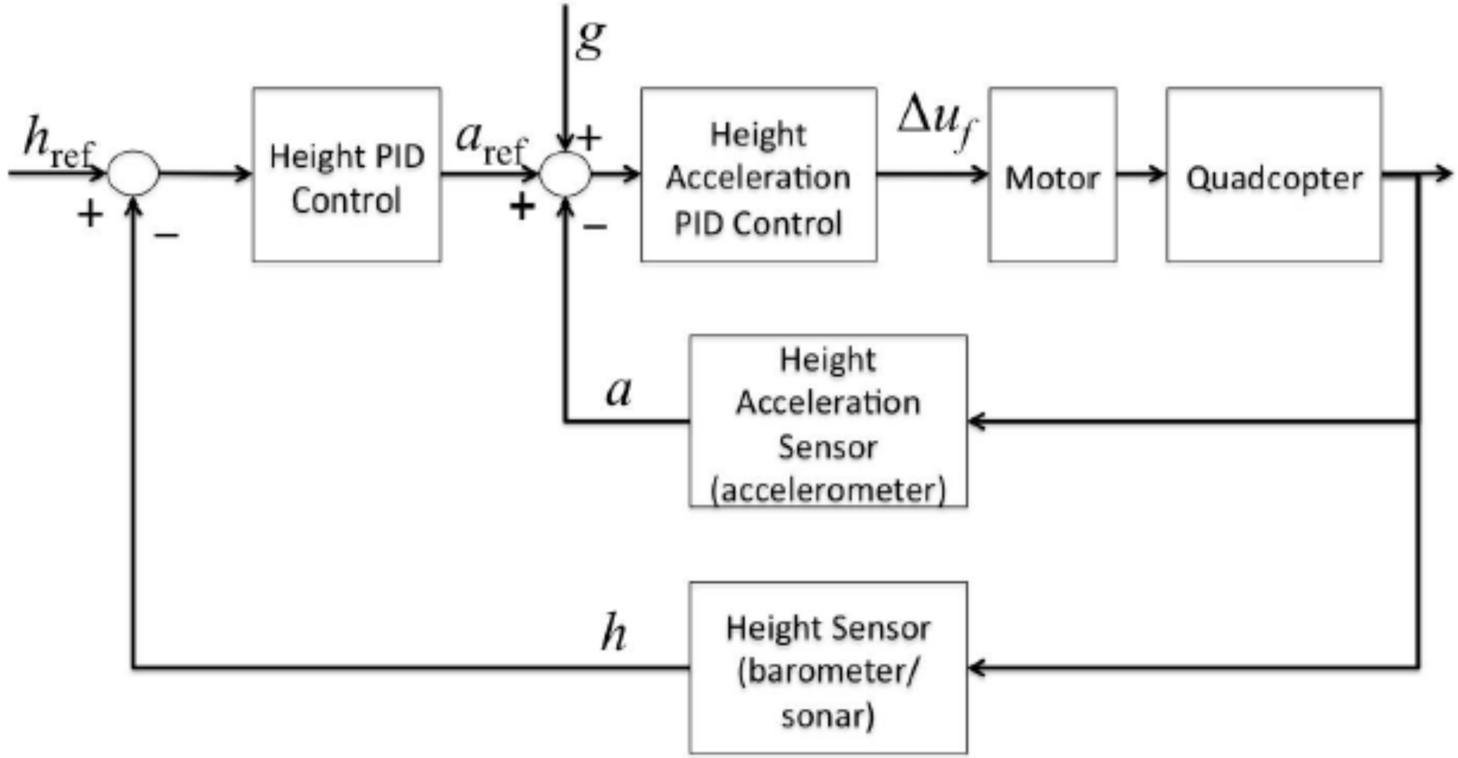


$$
\begin{aligned}
\omega_\theta^{\text{ref}} &= k_\theta^{\text{p}}(\theta_{\text{ref}} - \theta) + k_\theta^{\text{d}}(\dot{\theta}_{\text{ref}} - \dot{\theta}) \\
&\quad + k_\theta^{\text{i}} \int_0^t (\theta_{\text{ref}}(\tau) - \theta(\tau))\mathrm{d}\tau, \\
u'_\theta &= k_{\omega_\theta}^{\text{p}}(\omega_\theta^{\text{ref}} - \omega_\theta) + k_{\omega_\theta}^{\text{d}}(\dot{\omega}_\theta^{\text{ref}} - \dot{\omega}_\theta) \\
&\quad + k_{\omega\theta}^{\text{i}} \int_0^t (\omega_\theta^{\text{ref}}(\tau) - \omega_\theta(\tau))\mathrm{d}\tau,
\end{aligned}
$$

Platform architecture: application layer implementation

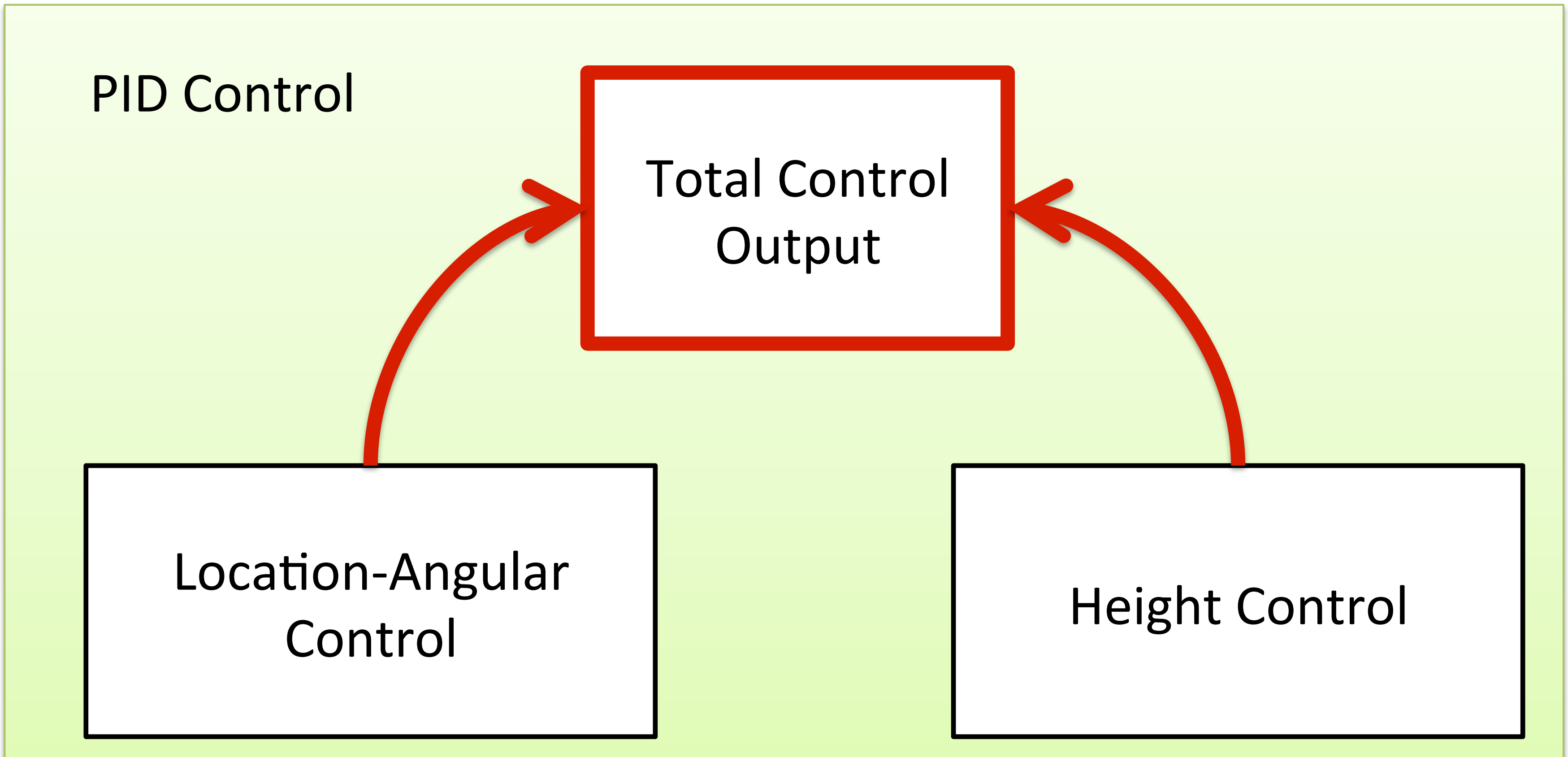


Height control: accurate vertical acceleration readings, inaccurate height readings



$$
\begin{aligned}
a_{\text{ref}} &= k_h^{\text{p}}(h_{\text{ref}} - h) + k_h^{\text{d}}(\dot{h}_{\text{ref}} - \dot{h}) \\
&\quad + k_h^{\text{i}} \int_0^t (h_{\text{ref}}(\tau) - h(\tau))\mathrm{d}\tau, \\
\Delta u_f &= k_a^{\text{p}}(a_{\text{ref}} + g - a) + k_a^{\text{d}}(\dot{a}_{\text{ref}} + \dot{g} - \dot{a}) \\
&\quad + k_a^{\text{i}} \int_0^t (a_{\text{ref}}(\tau) + g(\tau) - a(\tau))\mathrm{d}\tau,
\end{aligned}
$$

Platform architecture: application layer implementation



All above control outputs sum up to become the total control output to propeller motors

$$u_f(t + \mathrm{d}t) = u_f(t) + \Delta u_f(t)$$

Updated throttle component

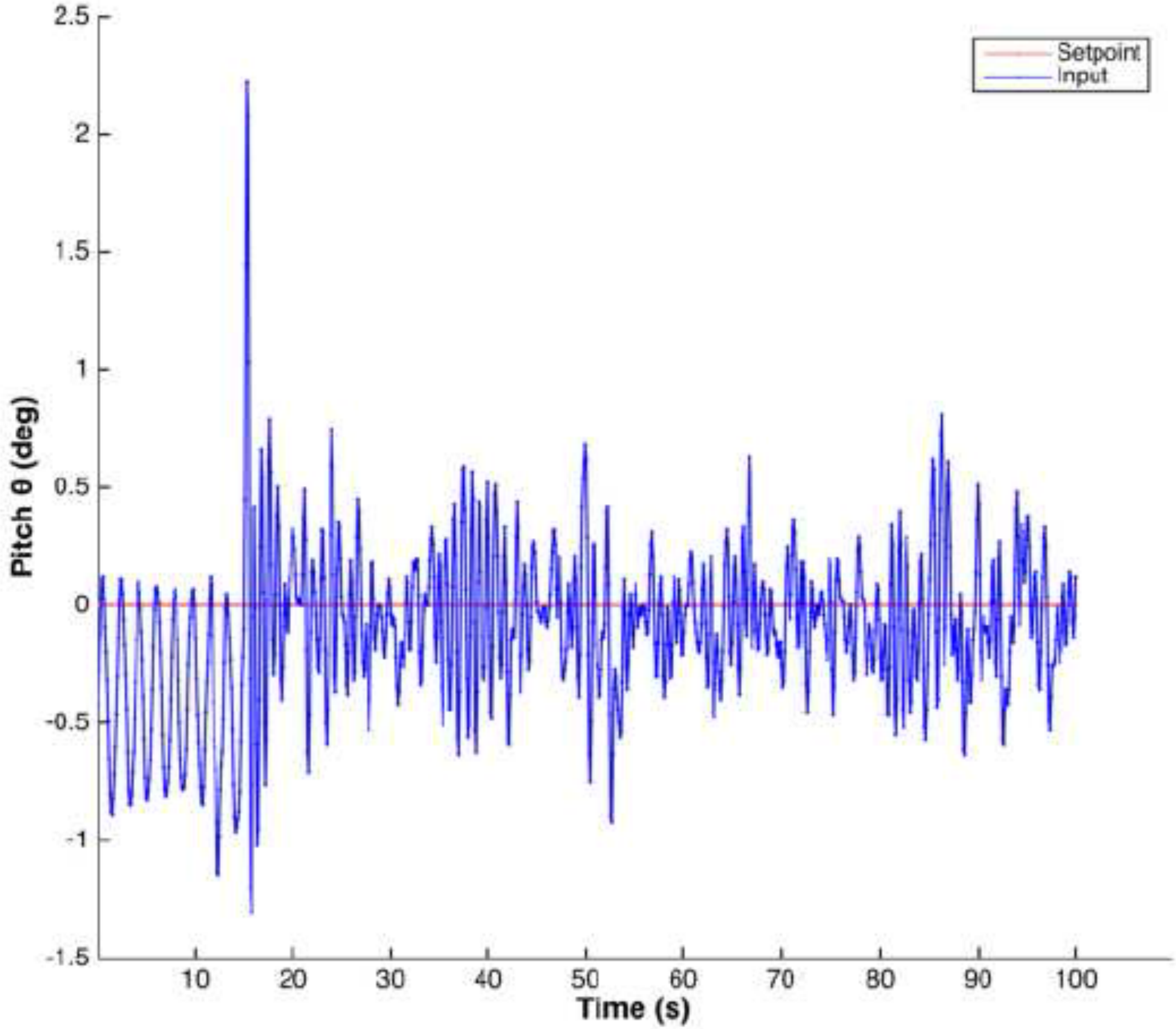
Height control output

All above control outputs sum up to become the total control output to propeller motors

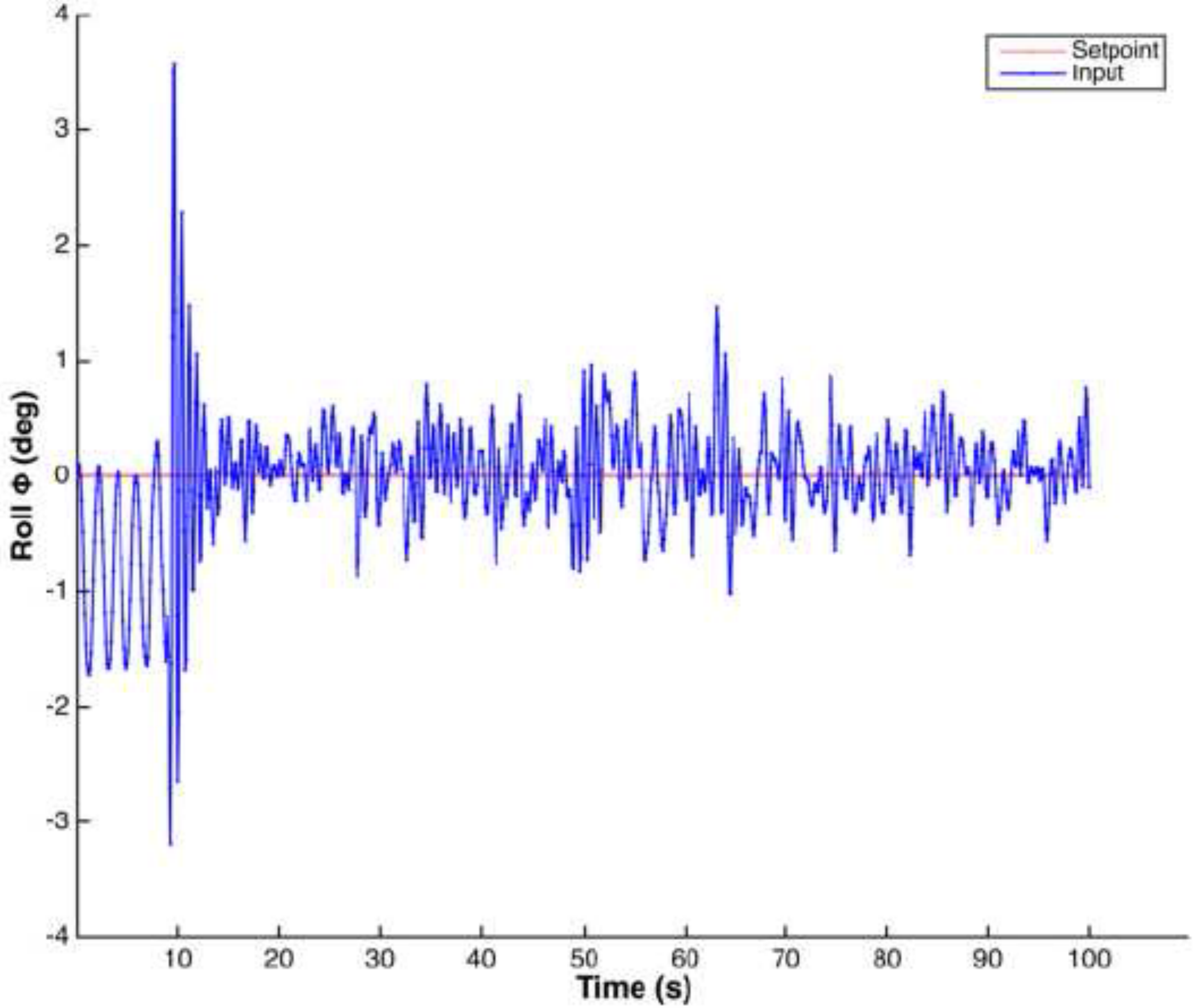
$$u_f(t+\mathrm{d}t) = u_f(t) + \Delta u_f(t)$$

$$U_1(t+\mathrm{d}t) = u_f(t+\mathrm{d}t) + u'_\theta(t) + u'_\phi(t) - u'_\psi(t),$$
$$U_2(t+\mathrm{d}t) = u_f(t+\mathrm{d}t) - u'_\theta(t) + u'_\phi(t) + u'_\psi(t),$$
$$U_3(t+\mathrm{d}t) = u_f(t+\mathrm{d}t) + u'_\theta(t) - u'_\phi(t) + u'_\psi(t),$$
$$U_4(t+\mathrm{d}t) = u_f(t+\mathrm{d}t) - u'_\theta(t) - u'_\phi(t) - u'_\psi(t).$$

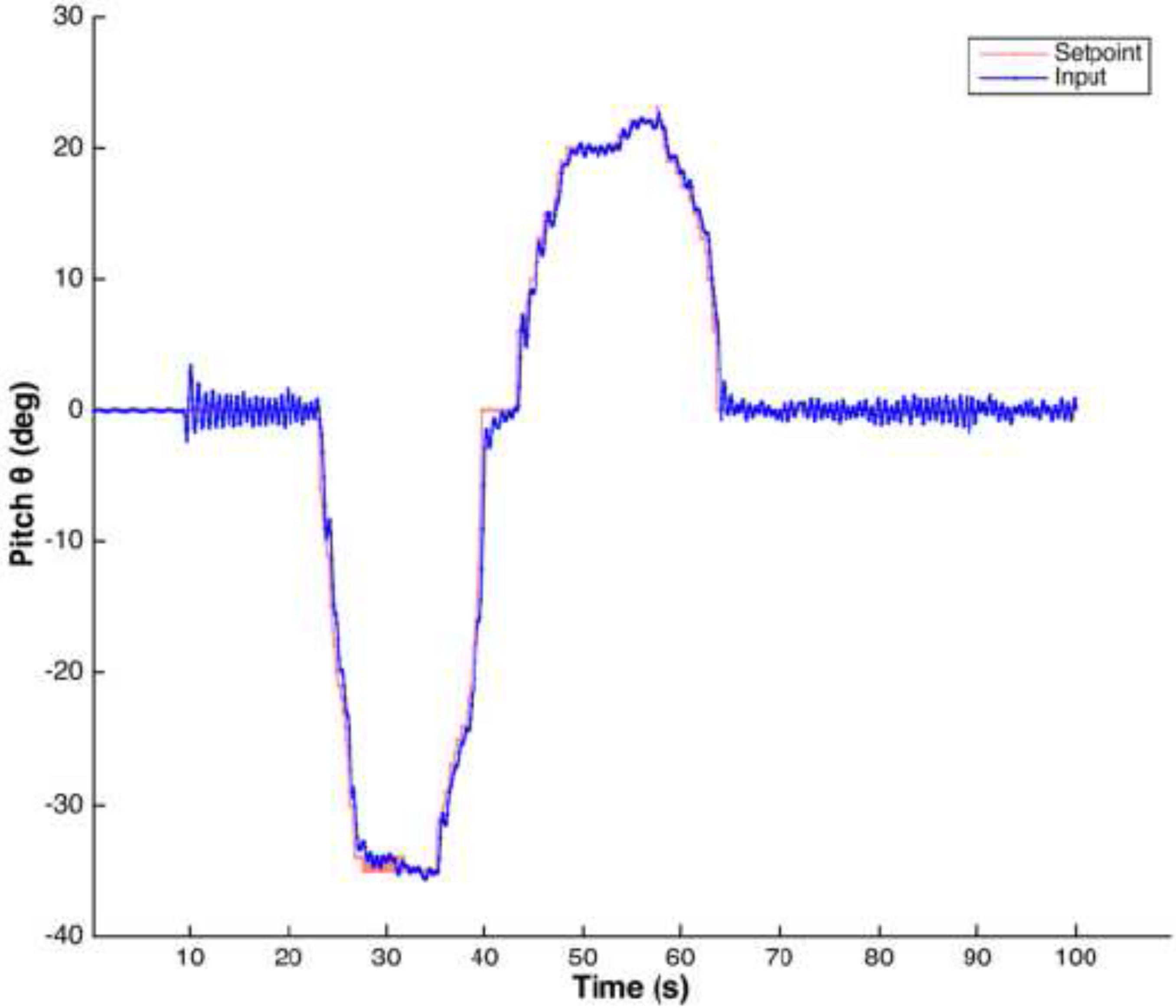
Pitch Angle Horizontal Stabilization (IMU starts at t = 0; motors start at around t = 15 sec; "Setpoint": the desired pitch angle)



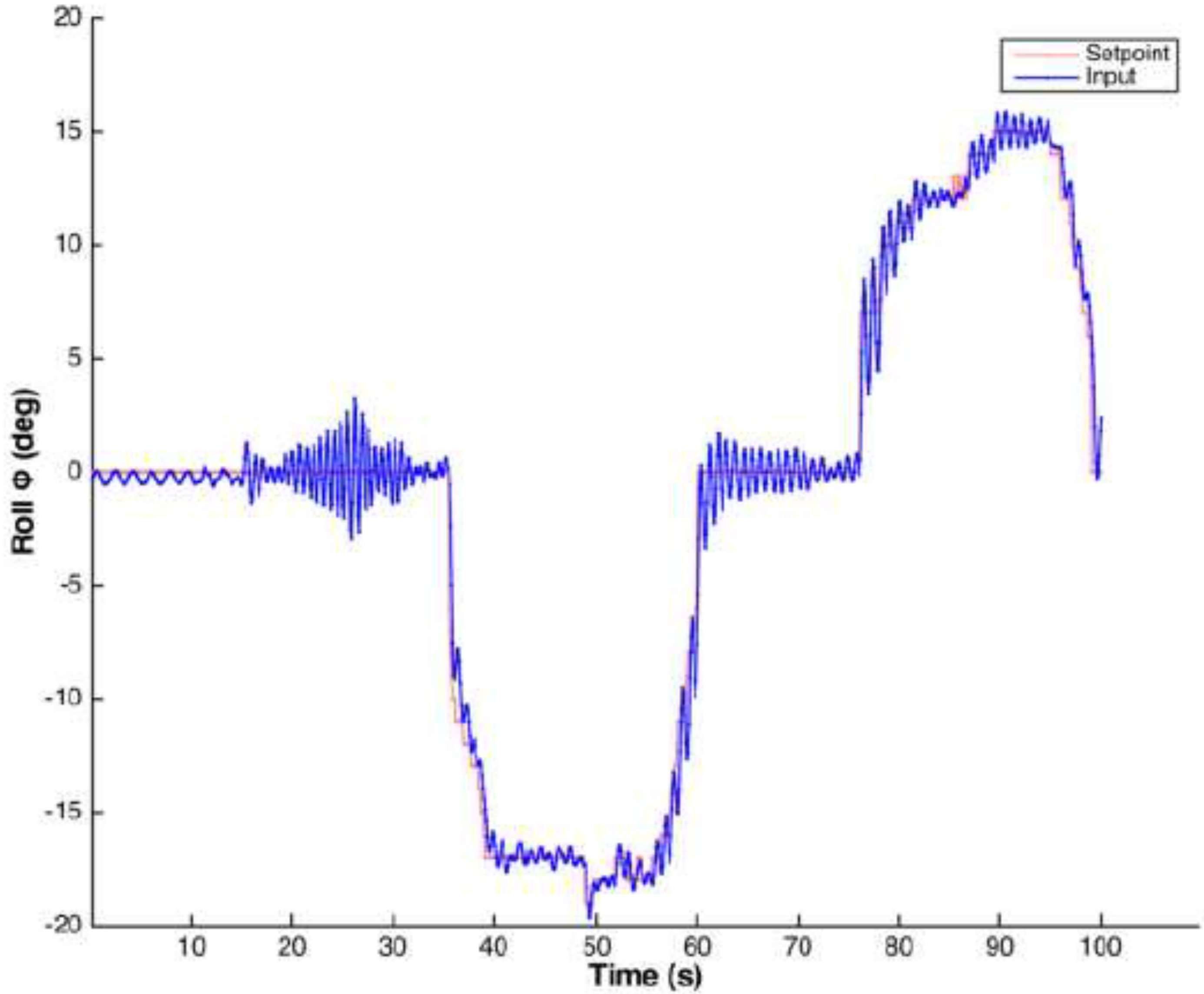
Roll Angle Horizontal Stabilization (IMU starts at t = 0; motors start at around t = 9 sec; “Setpoint” means the desired roll angle)



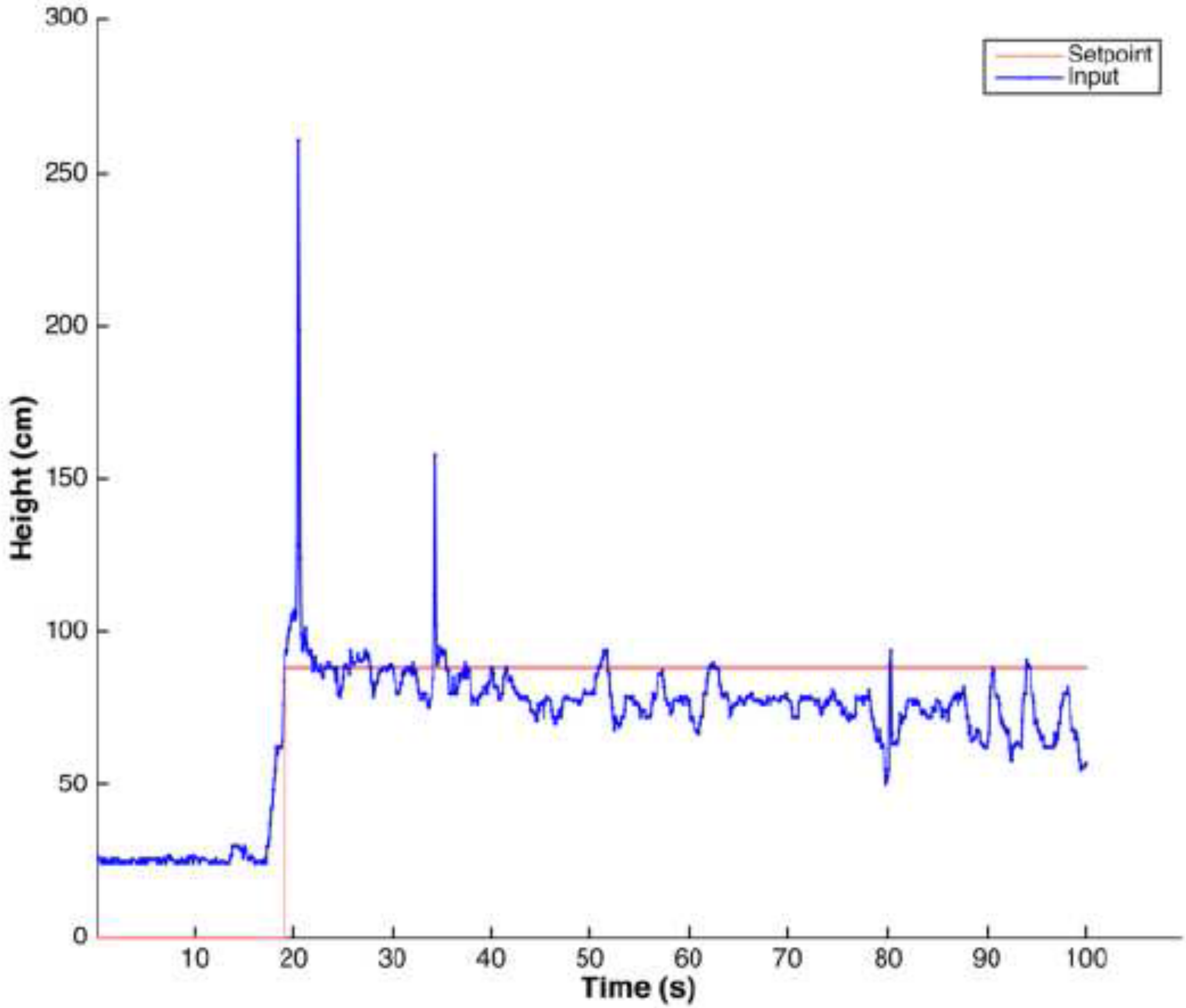
Pitch Tracking (IMU starts at t = 0; motors start at around t = 10 second; "Setpoint" means the desired pitch angle)



Roll Tracking (IMU starts at t = 0; motors start at around t = 15 sec; “Setpoint” means the desired roll angle)



Height Control Trace (IMU and sonar start at t = 0; motors start at around t = 17 sec; “Setpoint” means the desired height)



Conclusion

We build an open source quadcopter platform.

Much simpler source code compared to Ardupilot (135KB application layer source code)

Good documentation

Thank you!

We build an open source quadcopter platform.

Much simpler source code compared to Ardupilot (135KB application layer source code)

Good documentation

Thank you!

Demo:

http://www.comp.polyu.edu.hk/~csqwang/coolstuff

Source Code URL:

https://github.com/Ard-mu-copter/