CanberraUAV Workshop Flight Controllers

Feb 2017

Flight Controllers

- Also known as Autopilot(s)
- Take in information from sensors
- Calculate the current state of the UAV
- Compare this to where it’s supposed to be
- Output that action to the engines and control surfaces

Flight Controllers

- Can have different levels of automation
  - Stabilisation
  - Waypoint-based navigation
  - Full decision-making capability
- May have failsafes for recovery from emergency situations

Popular Flight Controllers

- Open-source
  - Ardupilot
  - PX4
  - Paparazzi
  - Cleanflight
  - KKMulticopter
  - MultiWii
  - Naze32

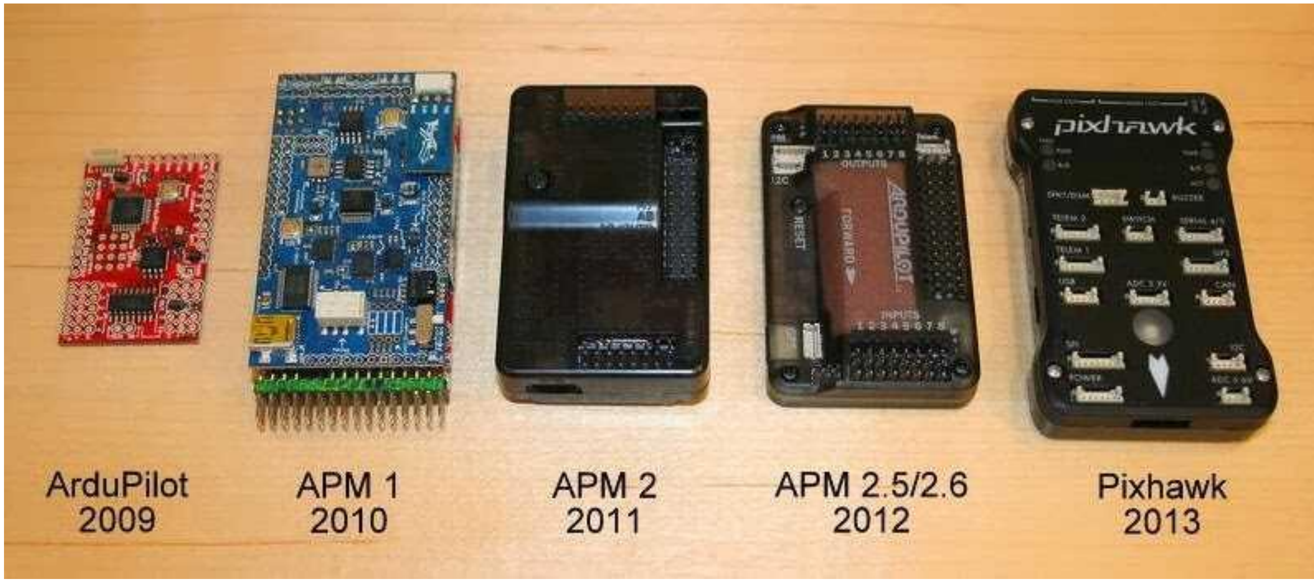
APM History – Ardupilot

- Also known as
  - Arducopter/Arduplane/Ardurover
  - APM
- Capable of controlling many different types of vehicles
  - Planes, Multicopters, Helicopters, Rovers, Boats, Submarines
- Waypoint-based navigation
- Advanced failsafe system
- Highly configurable via (many!) parameters

APM History – Ardupilot

- Started in 2009 by Jordi Munoz, Doug Weibel, and Jose Julio
- Designed to run on an Arduino board
- Jordi Munoz and Chris Anderson went on to found 3D Robotics
- Open source (GPL V3) project

APM History – Hardware



Pixhawk 2(.1) (2016)

Arduino

ARM

APM Basics – Ardupilot Organisation

- Was previously funded by 3D Robotics
    - 3DR heavily used the Ardupilot software in their UAV's
    - Also sold many DIY parts
- Was part of the Dronecode foundation
- In early 2016, moved to the ardupilot.org non profit organisation

Before we go any further…

- All information if based on the current Arduplane release (3.5.2)
- Some settings/parameters may be different for Arducopter/rover
- Some settings/parameters may change in future releases of Arduplane

APM Basics – Maintainers

- Andrew Tridgell
  - *Vehicle*: Plane, AntennaTracker
  - *Board*: APM1, APM2, Pixhawk, Pixhawk2, PixRacer
- Randy Mackay
  - *Vehicle*: Copter, AntennaTracker
- Robert Lefebvre
  - *Vehicle*: TradHeli
- Grant Morphett:
  - *Vehicle*: Rover
- Tom Pittenger
  - *Vehicle*: Plane
- Paul Riseborough
  - *Subsystem*: AP_NavEKF2
- Lucas De Marchi
  - *Subsystem*: Linux
- Peter Barker
  - *Subsystem*: DataFlash
  - *Subsystem*: Tools
- Michael du Breuil
  - *Subsystem*: uBlox GPS
- Francisco Ferreira
  - *Bug Master*
- Matthias Badaire
  - *Subsystem*: FRSky
- Víctor Mayoral Vilches
  - *Board*: PXF, Erle-Brain 2, PXFmini
- Mirko Denecke
  - *Board*: BBBmini
- Georgii Staroselskii
  - *Board*: NavIO
- Emile Castelnuovo
  - *Board*: VRBrain
- Julien BERAUD
  - *Board*: Bebop & Bebop 2
- Pritam Ghanghas
  - *Board*: Raspilot
- Jonathan Challinger
  - *Vehicle*: 3DRobotics Solo ArduPilot maintainer
- Gustavo José de Sousa
  - *Subsystem*: Build system
- Craig Elder
  - *Administration*: ArduPilot Technical Community Manager

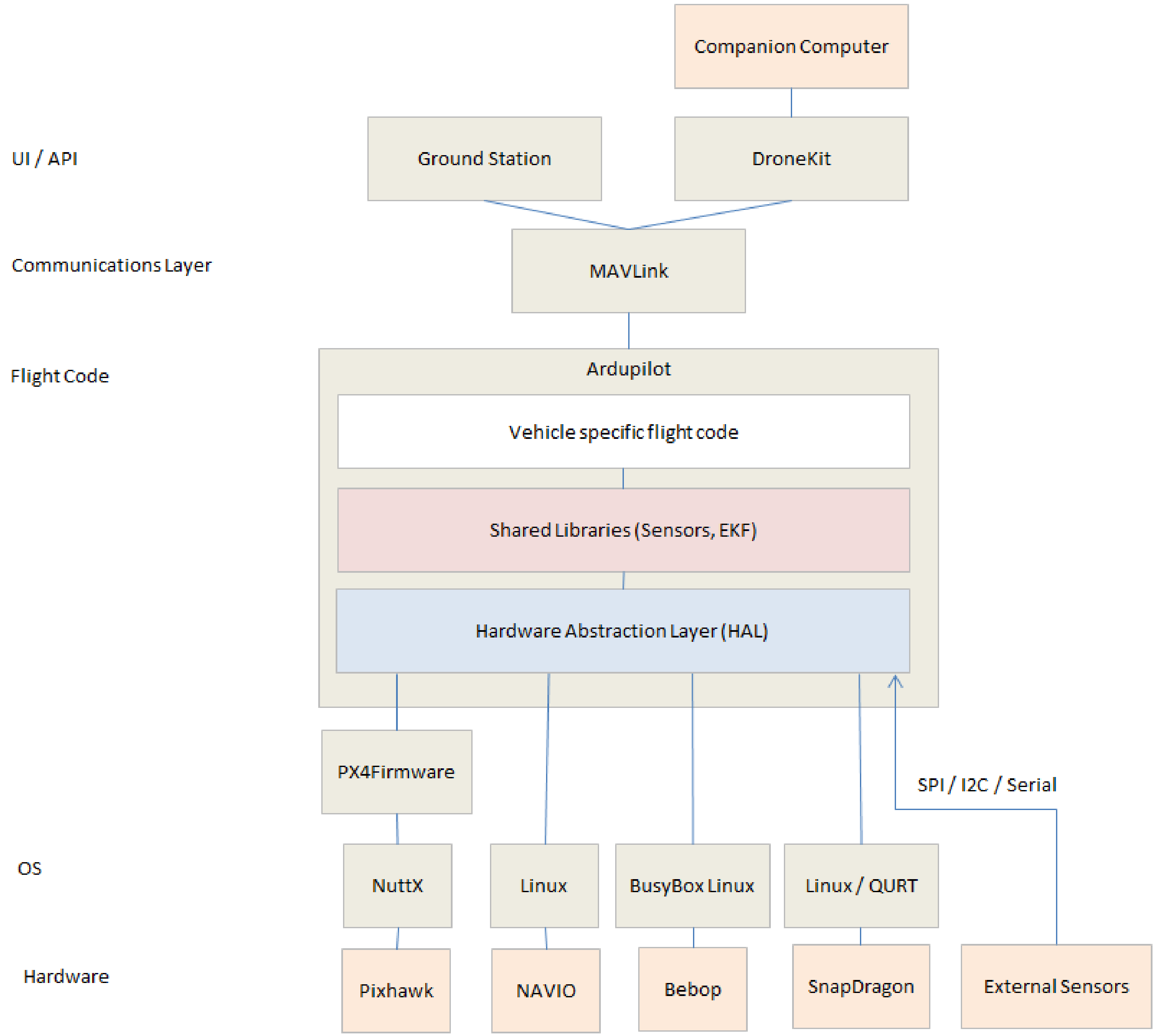
Ardupilot – HW vs SW

- Ardupilot refers to the software
- It can run on many different platforms
  - Pixhawk
  - BeagleBoneBlack
  - Raspberry Pi
  - X86
  - Many different ARM-based boards
  - (Arduino support has been dropped in recent versions)

Ardupilot – HW vs SW

- New hardware boards are being added regularly
- Many variants of the Pixhawk platform in particular

Ardupilot – Architecture



Ardupilot – Architecture

- Core Libraries
  - **AP_AHRS** – attitude estimation using DCM or EKF
  - **AP_Common** – core includes required by all sketches and libraries
  - **AP_Math** – various math functions especially useful for vector manipulation
  - **AC_PID** – PID controller library
  - **AP_InertialNav** – inertial navigation library for blending accelerometer inputs with gps and baro data
  - **AC_AttitudeControl** –
  - **AP_WPNav** – waypoint navigation library
  - **AP_Motors** – multicopter and traditional helicopter motor mixing
  - **RC_Channel** – a library to more convert pwm input/output from APM_RC into internal units such as angles
  - **AP_HAL, AP_HAL_AVR, AP_HAL_PX4** – libraries to implement the "Hardware abstraction layer" which presents an identical interface to the high level code so that it can more easily be ported to different boards.

Ardupilot – Architecture

- Multithreaded (where supported) for low-level IO work and sensor drivers
- Uses the AP_Scheduler library in the main vehicle thread

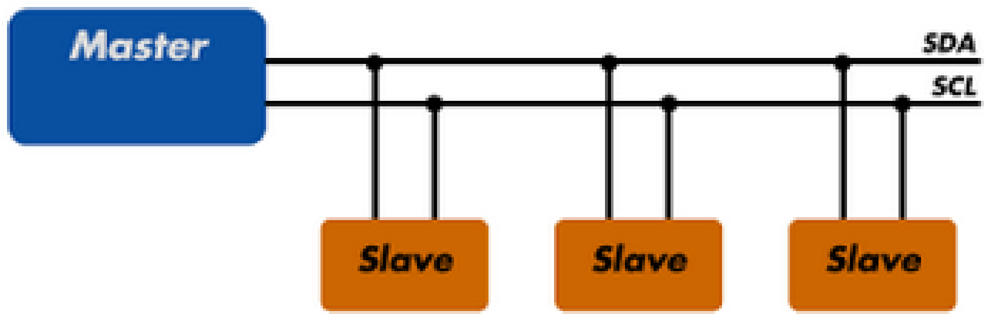
```
/*
  scheduler table - all regular tasks are listed here, along with how
  often they should be called (in Hz) and the maximum time
  they are expected to take (in microseconds)
 */
const AP_Scheduler::Task Plane::scheduler_tasks[] = {
                           // Units:   Hz      us
    SCHED_TASK(ahrs_update,           400,    400),
    SCHED_TASK(read_radio,             50,    100),
    SCHED_TASK(check_short_failsafe,   50,    100),
    SCHED_TASK(update_speed_height,    50,    200),
    SCHED_TASK(update_flight_mode,    400,    100),
    SCHED_TASK(stabilize,             400,    100),
    SCHED_TASK(set_servos,            400,    100),
    SCHED_TASK(read_control_switch,     7,    100),
    SCHED_TASK(gcs_retry_deferred,     50,    500),
    SCHED_TASK(update_GPS_50Hz,        50,    300),
    SCHED_TASK(update_GPS_10Hz,        10,    400),
```

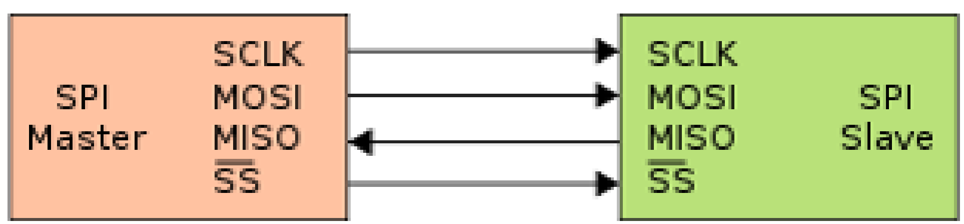
Ardupilot – Architecture

- 2 Persistent Storage areas
  - StorageManager
    - Parameters
    - Waypoints
    - Geofence points
    - Rally points
  - DataFlash
    - System log
    - Are the *.bin files on the Pixhawk SD card

Ardupilot – Sensors

- Sensors provide information about the current state of the UAV
- Different communications buses are supported
  - I2C
  - SPI
  - UART
  - CAN





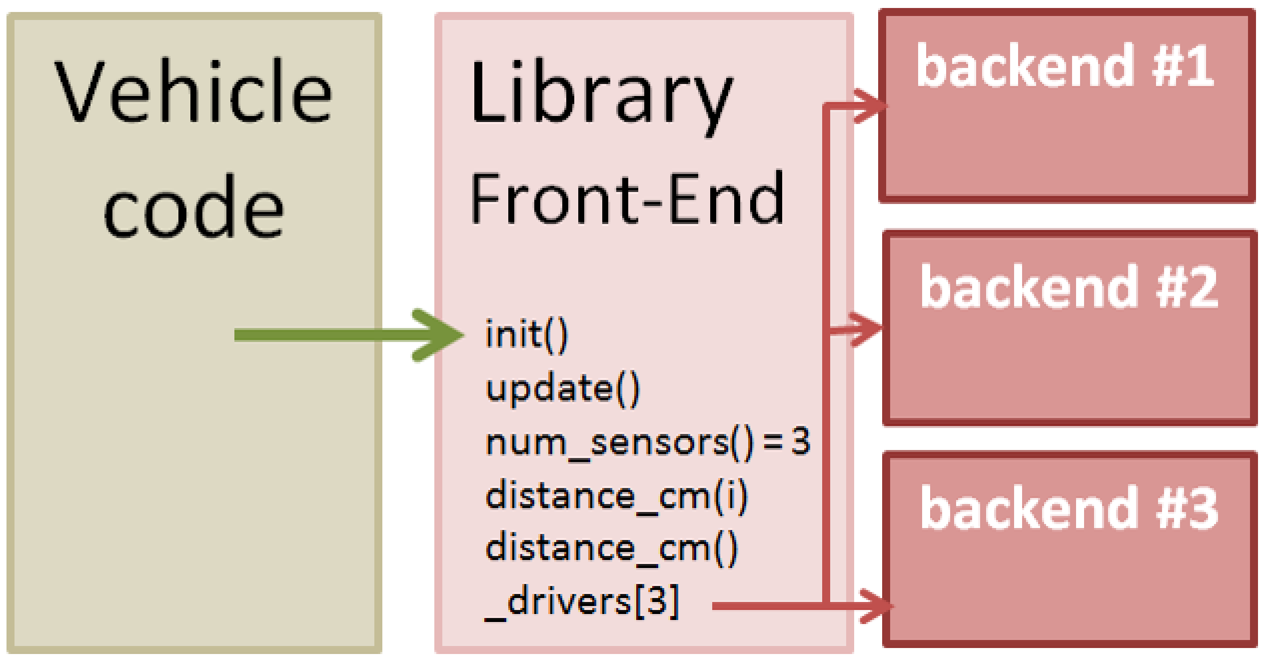
Ardupilot – Sensors

- Gyro (I2C/SPI)
- Accelerometer (SPI)
- Magnetometer (I2C)
- GPS (UART)
- Power Sensor (I2C)
- Barometer (I2C)
- Pitot (I2C)
- Laser Rangefinder (UART)
- And more…

Support redundant sensors

Ardupilot – Sensors

- Sensors are auto-detected on startup
- Hardware Abstraction Layer (HAL) separates the front and back end



Ardupilot – Preflight Checks

- System performs a check before arming
  - Barometer
  - Inertial (Gyro/Accel)
  - Attitude solution (AHRS)
  - Compass
  - GPS
  - Battery
  - Airspeed
  - Logging
  - RC Control
  - Safety switch
- Will not arm if a check fails
  - Checks can be disabled. Not recommended!

Ardupilot – Compiling

- Building the code into a single binary file
- Different compilers needed for each hardware target
  - G++ for Linux/Windows
  - GCC-ARM (non-eabi) 4.9.7 or 5.9.3 for Pixhawk

Ardupilot – Compiling

- Ardupilot uses the “waf” make system
  - `waf --board=navio2 --targets =bin/arduplane`
- `--board` is the hardware target (sitl, px4-v1, etc)
- `--target` is the airframe type (coax heli hexa octa octa-quad single tri y6)
- `waf --help` to get documentation

Ardupilot – Compiling

- Useful waf commands
- `waf list`
  - Lists all vehicle types (and other test programs)
- `waf list-boards`
  - Lists all board targets
- `waf clean`
  - Delete all files created during build
- Also useful to add a –jx, where x is the number of threads to use in build

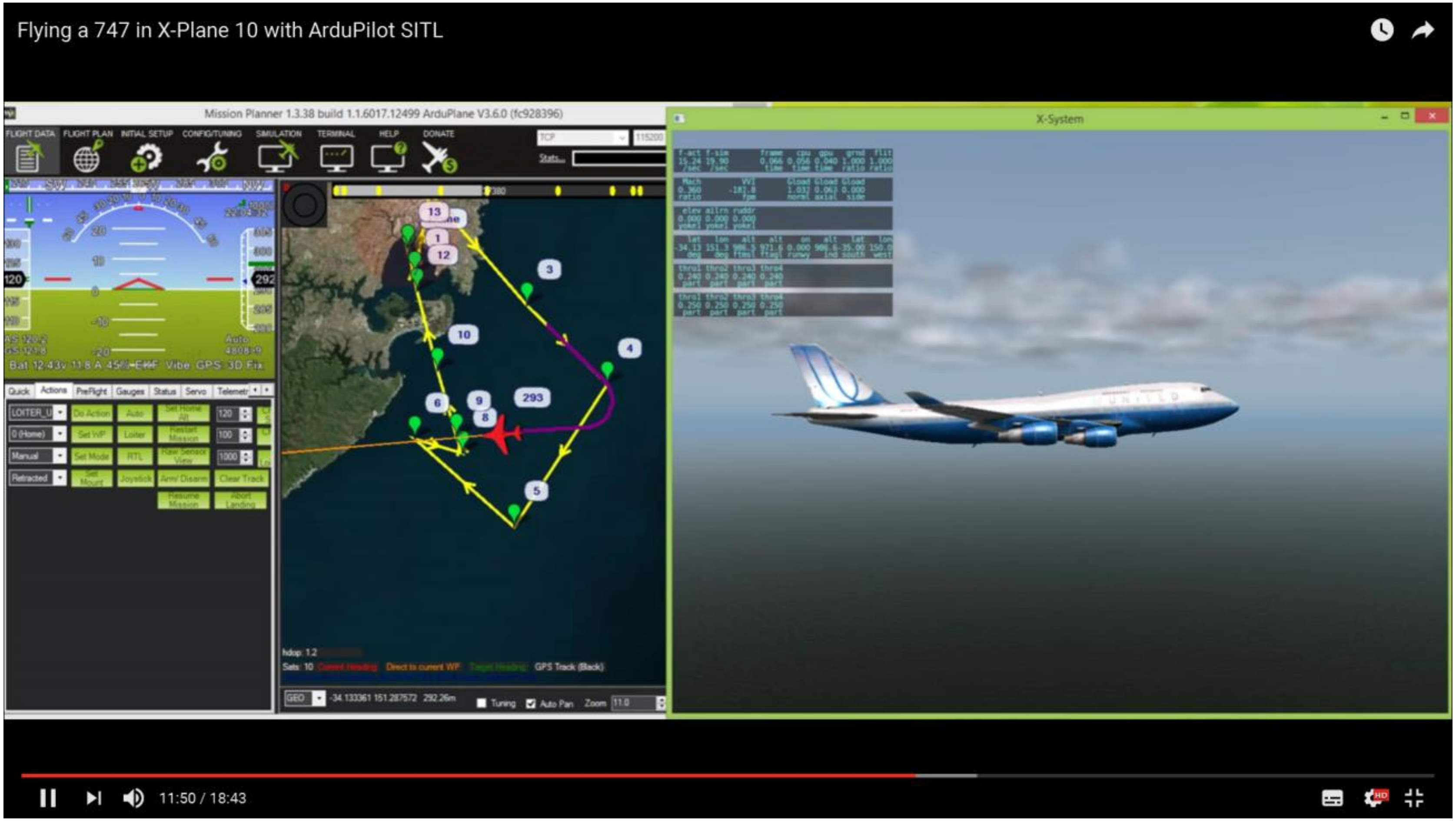
Ardupilot – Uploading

- To upload to a Pixhawk
  - `waf --upload /bin/arduplane`
- Mostly for working with the Pixhawk. Most other platforms (such as a Raspberry Pi) are a simple copy and paste

Ardupilot – SITL

- Software In The Loop
- Runs Ardupilot attached to a flight simulator
  - Jsbsim for Plane
  - Custom simulators for copter, rover
  - Can be attached to other simulators (Gazebo, Crrcsim, X-plane, etc)
- Very useful for testing!

Ardupilot – SITL



Ardupilot – SITL

- Ardupilot has a single script to build and run a SITL environment
  - `cd ardupilot/ArduPlane`
  - `../Tools/autotest/sim_vehicle.py`

Ardupilot – SITL

- sim_vehicle options:
  - `-w` Wipe and reset EEPROM to defaults
  - `-L <location>` Start at a specific location (CMAC, Kingaroy, QMAC). Full list in ./Tools/autotest/locations.txt
  - `--console` Use the MAVProxy console
  - `--map` Enable to moving map
  - `-f <frame>` Use a specific frame (+, X, quad or octa for Arducopter)

Practical Session 1 (20min)

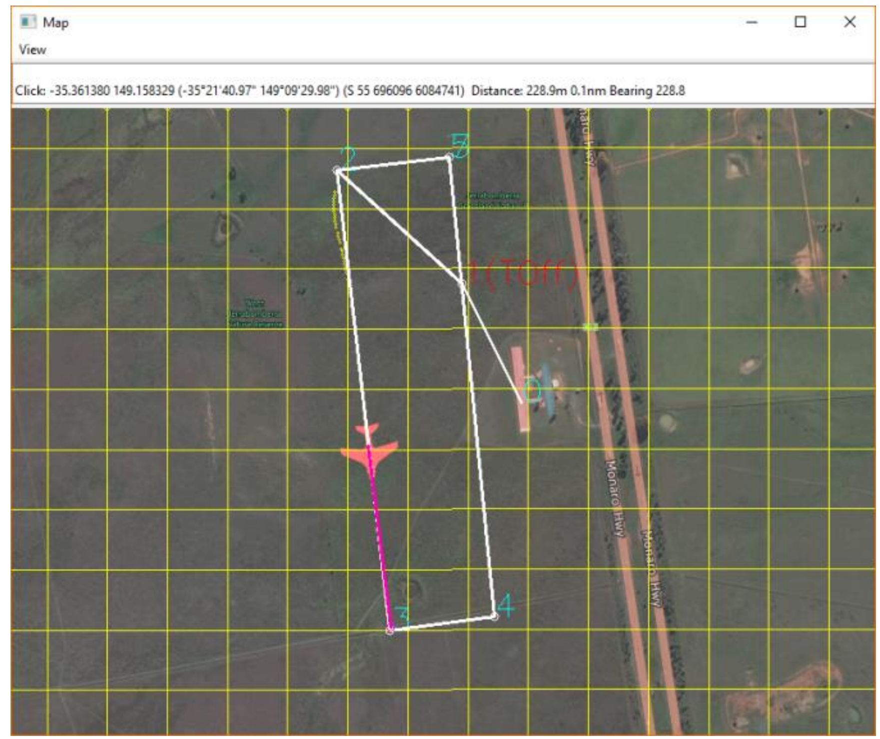
- Configure WAF and build Arduplane
  - `cd ./ardupilot`
  - `alias waf="$PWD/modules/waf/waf-light"`
  - `waf configure --board=sitl`
  - `waf --board=sitl --targets=bin/arduplane`
- Try building Arducopter
  - `waf --board=sitl --targets=bin/arducopter-quad`

Ardupilot – Mission

- A mission is a set of waypoints that will be flown in auto mode
- Missions are quite simple
  - Go here, do that
- No conditional statements or branching
  - But can do loops
  - Some exceptions (we'll see later)

Ardupilot – Waypoints

- Simple text file
- Each line is one waypoint
- Can be
  - Navigation commands
  - Do auxiliary function
  - Condition commands



Ardupilot – Waypoints

- File starts with line `QGC WPL 110`
- Next line is the home location
- Each line thereafter is a series of 12 tab-separated values
  - Wp index number
  - Current wp
  - Coordinate frame
  - The waypoint type
  - Next 7 columns are the waypoint options
  - Last column is autocontinue

Ardupilot – Waypoints

Frame
0 = absolute altitude
3 = relative altitude

CMAC-toff-loop.txt - Notepad

File Edit Format View Help

```
QGC WPL 110
0	1	0	16	0.000000	0.000000	0.000000	0.000000	-35.362938	149.165085	584.409973	1
1	0	3	22	15.000000	0.000000	0.000000	0.000000	-35.361164	149.163986	28.110001	1
2	0	3	16	0.000000	0.000000	0.000000	0.000000	-35.359467	149.161697	99.800003	1
3	0	3	16	0.000000	0.000000	0.000000	0.000000	-35.366333	149.162659	100.730003	1
4	0	3	16	0.000000	0.000000	0.000000	0.000000	-35.366131	149.164581	100.000000	1
5	0	3	16	0.000000	0.000000	0.000000	0.000000	-35.359272	149.163757	100.000000	1
6	0	3	177	2.000000	-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1
7	0	3	16	0.000000	0.000000	0.000000	0.000000	-35.359272	149.163757	100.000000	1
```

X,Y,Z coords

Waypoint Type
16=Navigate to WP
22=Autotakeoff
177=Do Loop

Ardupilot – Waypoints

- Popular waypoints
  - `MAV_CMD_NAV_WAYPOINT` (Navigate to the specified position
  - `MAV_CMD_NAV_LOITER_TIME` (Loiter at the specified location for a set time)
  - `MAV_CMD_NAV_RETURN_TO_LAUNCH` (Return to the home location or the nearest Rally Point)
  - `MAV_CMD_DO_JUMP` (Jump to the specified command in the mission list)

Ardupilot – Waypoints

- `MAV_CMD_DO_SET_RELAY` (Set a Relay pin's voltage high (on) or low (off))
- `MAV_CMD_DO_SET_SERVO` (Set a given servo pin output to a specific PWM value)
- `MAV_CMD_CONDITION_DISTANCE` (Delay next DO_ command until less than `x` metres from next waypoint)
- Plus many more…

Ardupilot – Flight Modes

- Many flight modes that give different mixes of user and computer controlled output
  - `MANUAL` Complete manual control
  - `FLY BY WIRE_A (FBWA)` Will hold roll and pitch
  - `AUTO` Will run the mission stored in memory
  - `Return To Launch (RTL)` Will return straight to home point
  - `LOITER` Circle around current location
  - … plus more modes

Ardupilot – Failsafes

- Failsafes are systems that take over control of the UAV if there is a perceived emergency
- User configurable
- Ensure you know which failsafes are active and:
  - Under what condition they will activate
  - Ardupilot’s resulting action(s)
  - How to regain control

Ardupilot – Failsafes

- Short Failsafe
  - Default 1.5 sec
  - Choice of either Continue or circle
- Long Failsafe
  - Default 5 sec
  - Choice of either Continue or RTL

Ardupilot – Failsafes

- RC Failsafe
  - Activates at loss of signal from RC transmitter
  - Requires RC TX/RX to be set up first, so it can signal Ardupilot on loss of signal
  - Most recievers have a failsafe mode. Need to set this to output a low throttle value (<950 PWM)
- GCS Failsafe
  - Activates at loss of heartbeat packets from GCS

Ardupilot – Failsafes

- Battery voltage Failsafe
  - Activates when battery reaches low voltage
- Battery remaining Failsafe
  - Activates when remaining battery charge (mAh) is reached

Ardupilot – Failsafes

- GPS
  - There is not a GPS failsafe in Arduplane
  - Plane will warn the user and attempt to dead-reckon

Ardupilot – Failsafes

- Geofence
  - A set of points that define a closed polygon around the UAV
  - Can have altitude limits
  - Ardupilot Response can be:
    - Ignore
    - Report
    - Take over control and return
  - Disable for takeoff and landing!

Ardupilot – Failsafes

- There is an Advanced Failsafe System (AFS)
  - Designed to comply to the rules of the UAV Challenge

Practical Session 2 (20min)

Type this in Cygwin console

- Run sim_vehicle for a plane at CMAC, with default parameters, console and map
  - `cd ./ArduPlane`
  - `../Tools/autotest/sim_vehicle.py -w --console --map`
- Load a mission
  - `wp load ../Tools/autotest/ArduPlane-Missions/CMAC-toff-loop.txt`
- Run the mission in AUTO mode
  - `arm throttle`
  - `auto`

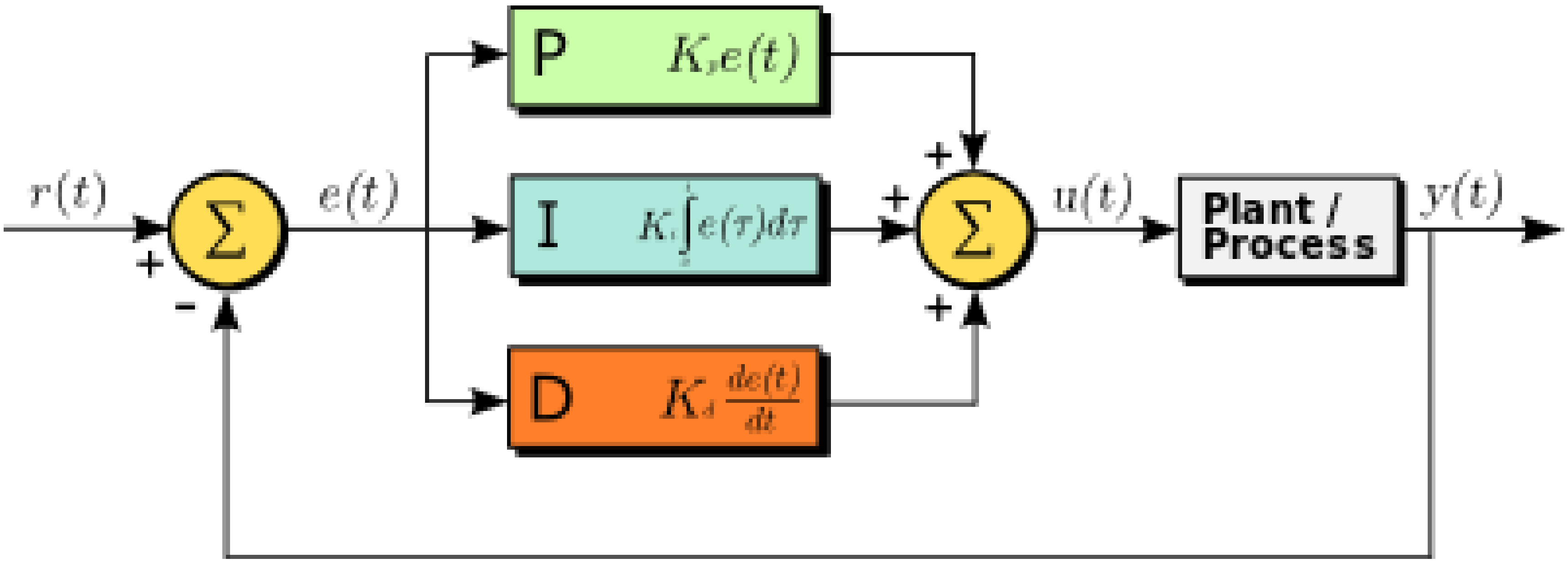
Type this in MAVProxy console

Ardupilot – Tuning

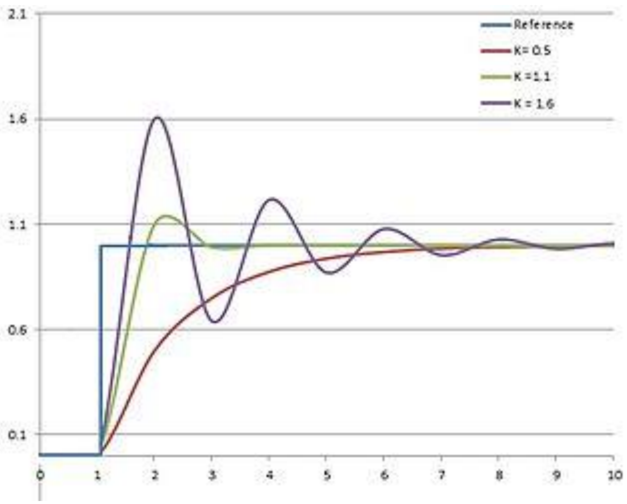
- Each airframe has different responses to movements in it's flight control surfaces
- Ardupilot needs to take account of these responses for precise control
- 3 Controllers that require tuning
  - PID (roll, pitch and yaw)
  - L1 (horizontal navigation)
  - TECS (height controller)
- PID is the most important
  - L1 and TECS defaults will cover most circumstances

Ardupilot – Tuning

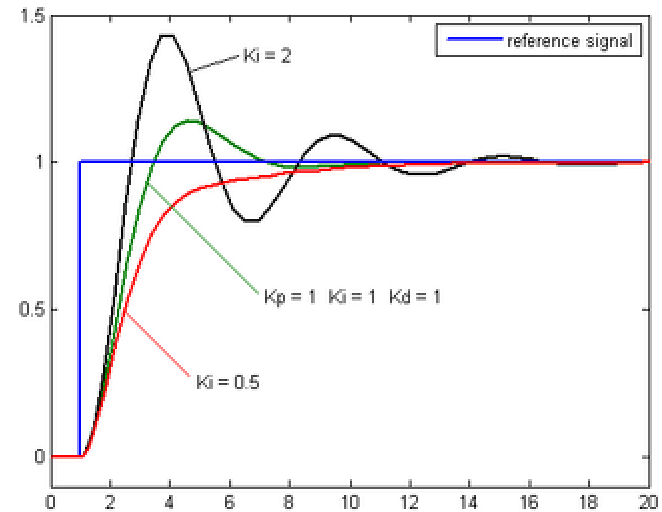
- PID Controllers



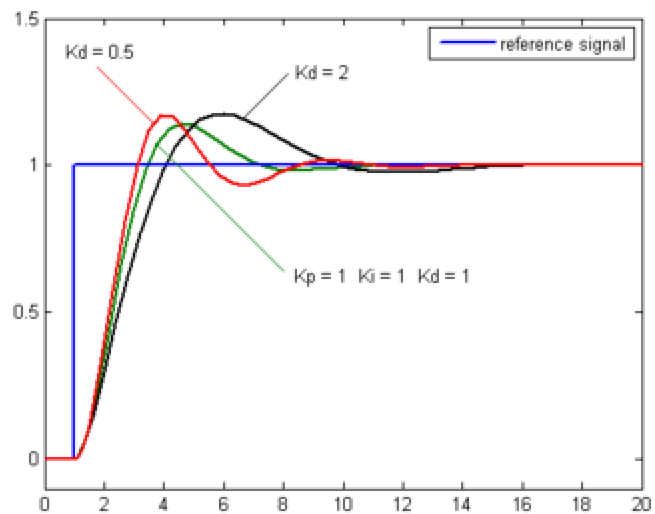
Ardupilot – Tuning



P



I



D

Ardupilot – Tuning

- PID tuning can be done manually or via autotune
  - Manual: One person flies the UAV whilst the GCS operator monitors the roll/pitch response and changes the PID values
  - Autotune: As above, but Ardupilot automatically measures the roll/pitch reponse and changes the PID values.

Ardupilot – Tuning

- Total Energy Control System (TECS)
  - Coordinates throttle and pitch angle demands to control the aircraft's height and airspeed
  - Trading off demanded speed and demanded climb rate
  - Complex tuning method

Ardupilot – Tuning

- L1 controller
  - Controls horizontal turns both for waypoints and loiter
  - Tuning the navigation controller usually involves adjusting one key parameter, called NAVL1_PERIOD
  - Small value = sharp turns
  - Large value = gentle turns

Ardupilot – EKF

- Extended Kalman Filter
  - Algorithm to estimate vehicle position, velocity and angular orientation
  - Take in measurements from all sensors (except rangefinder and pitot)
  - "Fuses" the readings from the sensors together for an accurate solution
  - Can reject readings with large errors
    - Single sensor failure can be handled
  - Does require a powerful CPU (>Arduino)

Practical Session 3 (20min)

- Start up Arduplane SITL
- Load and run the same mission as last time
- Vary the L1 controller in the MAVProxy console
  - `param set NAVL1_PERIOD n`
  - Where `n` is between 5 and 40 (default 20)
  - Watch the effect on the turns
- Vary the roll and pitch PID controllers in the MAVProxy console
  - `param set RLL2SRV_P n`
  - `param set PTCH2SRV_P n`
  - Where `n` is between 0.1 and 4 (default 2.5)
  - Watch the effect on the turns

The End!

- Ardupilot History
- Ardupilot Architecture
  - Sensors
  - Libraries
- Compiling and SITL
- Controller tuning
- Useful links
  - http://ardupilot.org/plane/index.html
  - http://ardupilot.org/dev/index.html
  - http://discuss.ardupilot.org/