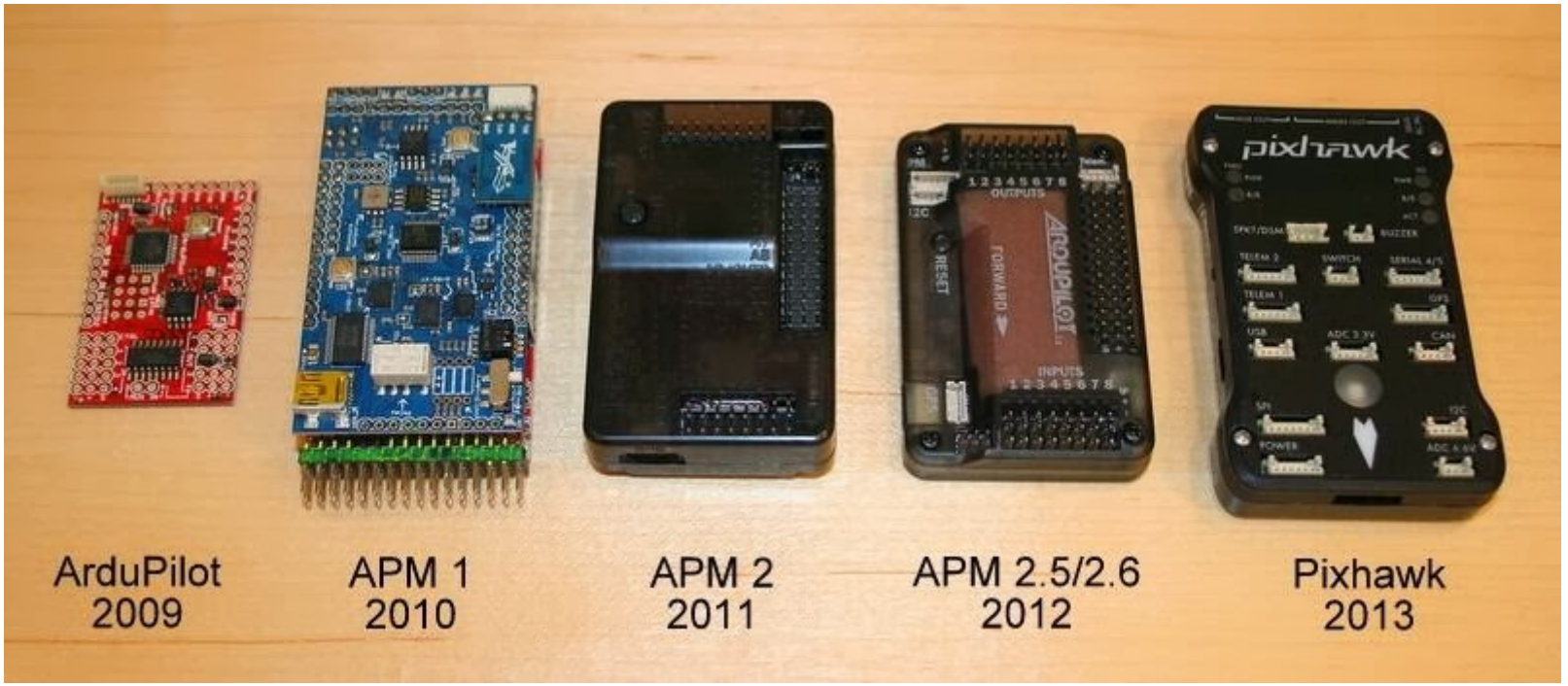
# APM on Linux

Porting an 8 bit autopilot to Linux

Andrew Tridgell

# APM autopilots over time



# History of ArduPilot

- Originally for 8 bit AVR micros
  - 'arduino' sketch
  - 8k ram
  - 16 MHz
  - extensive used of arduino 'wire' libraries
- SITL port
  - port to desktop OSs (eg. Linux), hacked on top of wire emulation layer

# Introduction of AP_HAL

- AP_HAL added for PX4
  - creates hardware abstraction layer
  - work begun by Pat Hickey
  - allows porting to many OSes and CPU architectures
  - avoids arduino wire library
  - keeps 'sketch' structure to allow building with arduino GUI

# AP_HAL ports

- Current ports
  - AP_HAL_AVR (8 bit AVR2560)
  - AP_HAL_AVR_SITL (SITL simulator)
  - AP_HAL_PX4 (PX4 based autopilots)
  - AP_HAL_Flymaple (low cost ARM autopilot)
  - AP_HAL_VRBrain (ARM32 autopilot)
  - AP_HAL_Linux (embedded Linux port)
  - AP_HAL_Empty (very useful!)

# Why AP_HAL_Linux?

- Complex UAVs
  - many complex UAVs have a embedded Linux box on board for imaging and other complex tasks
  - having a separate board makes UAV larger and wiring complex
  - Linux port enables WiFi, cameras and other features
  - allows for self-hosted autopilot
  - should be a fun hack!

# Difficulties of AP_HAL_Linux

- Predictable timing
  - autopilots need predictable timing
  - difficult to achieve on Linux
- Bus access and latency
  - autopilots need good SPI and I2C drivers
  - these are unusual on Linux and usually low rate

# What latencies are needed?

- 100nsec:
  - SPI bus transitions, almost certainly done in hardware
- 1usec:
  - PWM transitions, probably done in PRUs
  - PPMSUM input and SBUS, maybe in PRUs, maybe in timer capture?
- 1msec
  - sensor input (gyros, accels). Possibly helped by FIFOs
- 20msec
  - barometer, compass, airspeed, sonar (I2C, SPI and analog)
- 200msec
  - GPS

# Linux Boards

- Two boards investigated
  - RaspberryPi
  - BeagleBone Black
- BeagleBone Black looks more interesting
  - more GPIO pins
  - has PWM support
  - has two PRUs for realtime operations

# Distros and Setup

- Angstrom
  - Started with Angstrom on BBB
  - found it difficult for general development
  - switched to Debian – much happier!
- Setup as NFS root and NFS kernel
  - makes development faster at home
  - switched back to MMC for this talk

# Autopilot Sensors

- An autopilot needs:
  - 3D gyroscope
  - 3D accelerometer
  - 3D magnetometer
  - barometer
  - GPS
  - airspeed
  - telemetry ports

# Cheap Sensor

- Started with cheap “10 Dof” ebay sensor
  - 3D accel ADXL345
  - 3D gyro L3G4200D
  - HMC5883 magnetometer
  - barometer bmp085
- Added normal additional devices
  - MS4525DO I2C differential pressure sensor (airspeed)
  - uBlox GPS

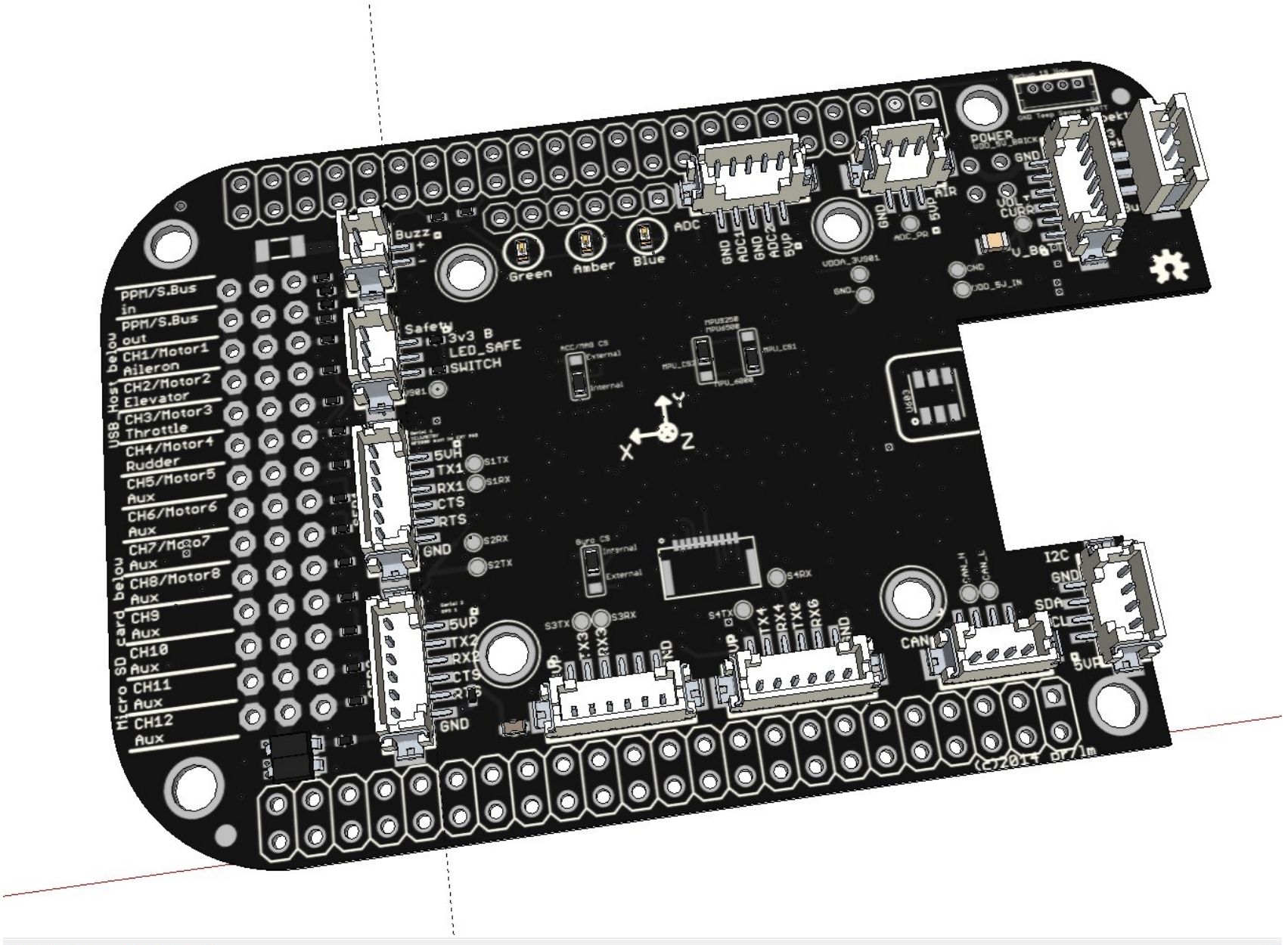
# Linux I2C interface

- Started with I2C drivers
  - latency was very high for default API
  - switched to ioctl API and better, but not good
  - now using batched ioctl, and a bit better
  - need fast SPI!
  - probably will end up with kernel driver

# Scheduling issues

- Can Linux scheduler reliably enough?
  - difficult with default kernel!
  - lots of scheduling misses
  - switch to RT/Preempt kernel from Ingo
  - much better!
  - also needed to disable freq scaling
  - needed to lock down memory and prefault stack
  - needed to force 400kHz I2C bus

# PixHawk Fire Cape



# Interesting issues

- Issues hit during port
  - how to measure time? (montonic vs wallclock)
  - how to delay for a time?
  - coping with tight loops
  - scheduling priorities
  - uart and IO threads

# What's missing?

- Still to be done
  - PixHawk Fire cape drivers
    - PRU drivers
  - PWM drivers
  - SPI drivers for cape sensors
  - RC input
  - power handling
  - analog sensing
  - flight tests!
  - web based development environment