Friendly Flying Robots With Ruby!

Jim Weirich / @jimweirich / jim@neo.com
Chief Scientist

neo

Time for Geeky Stuff

Time for Geeky Stuff

- Ruby Code

Time for Geeky Stuff

- Ruby Code
- Technical Specifications

Time for Geeky Stuff

- Ruby Code
- Technical Specifications
- Hardware Interfacing

Time for Geeky Stuff

- Ruby Code
- Technical Specifications
- Hardware Interfacing
- UML Diagram (only 1)

Time for Geeky Stuff

- Ruby Code
- Technical Specifications
- Hardware Interfacing
- UML Diagram (only 1)
- Flying Robots

Safety First!

How Not to Catch A Flying Robot

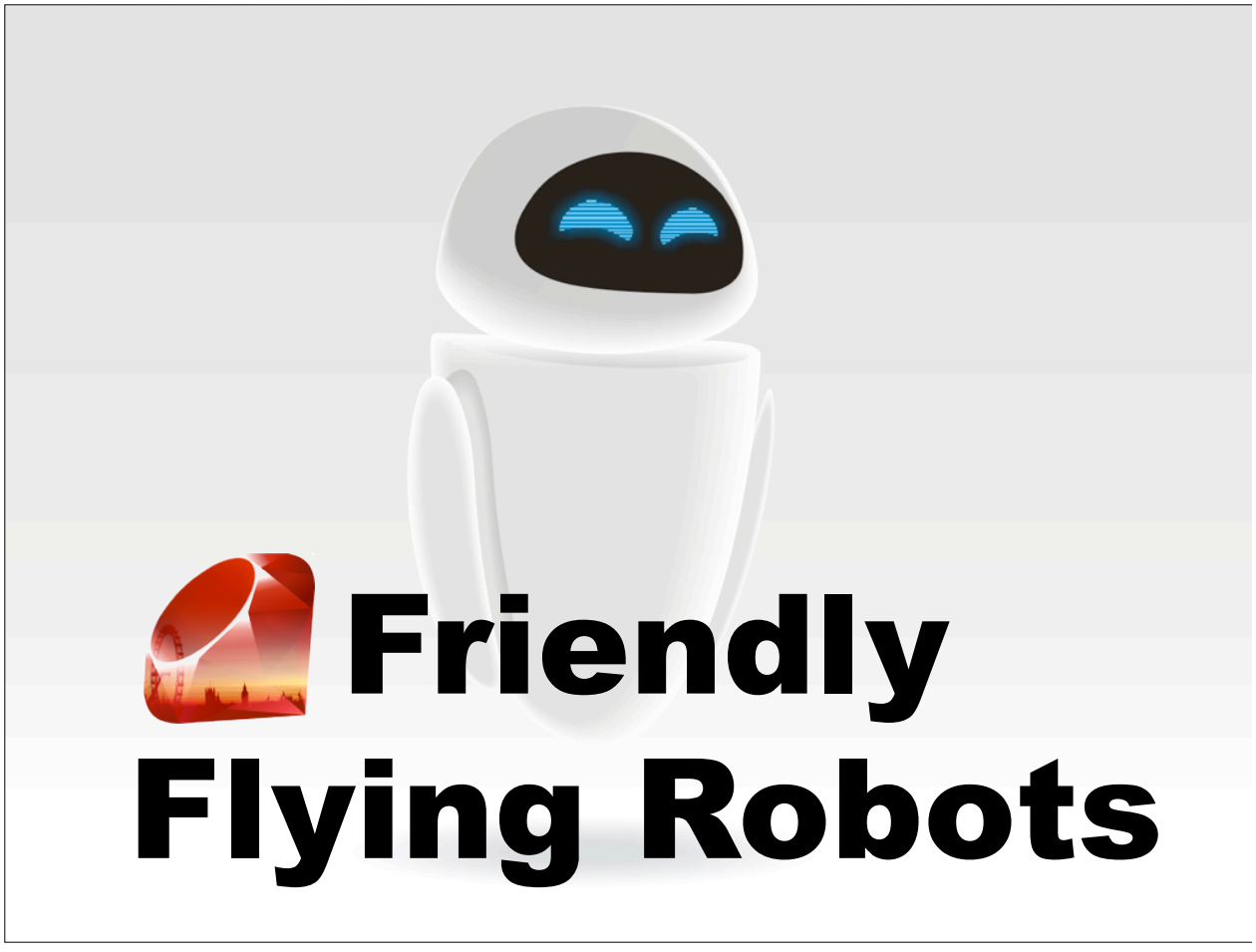
<< movie >>

How to Properly Catch A Flying Robot

<< movie >>

Terminology


NY
Drones

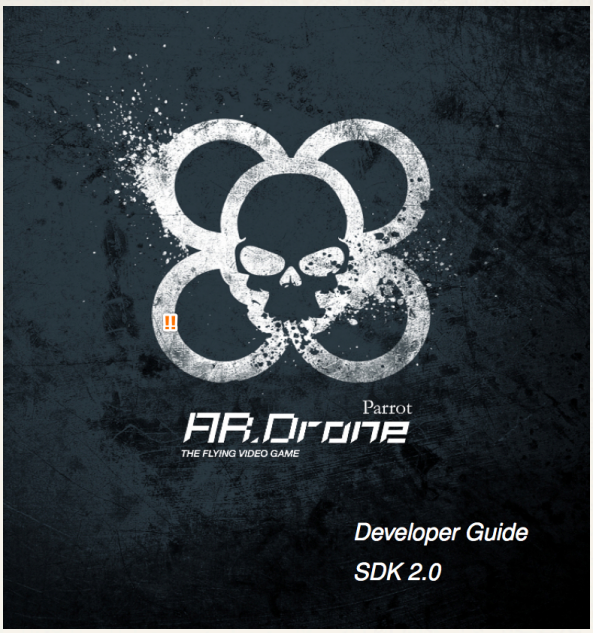

Friendly
Flying Robots

Hardware

AR Parrot Drone

- 2 Cameras (one forward, one down)
- Video recording onboard or remote
- 3 Axis Gyroscope / Accelerometer / Magnetometer
- Ultrasound sensors for height
- Pressure Sensor (outdoor height)
- 1 GHz 32 Arm processor running Linux
- Wifi
- iOS or Android Controllers

Software



Drone Developer Guide

https://projects.ardrone.org/wiki/ardrone-api/Developer_Guide

Command Stream

Navigation Data

Video Stream

Command Stream

```
AT*REF=seq,flags

AT*PCMD=seq,flags,roll,pitch,gaz,yaw

AT*CONFIG=seq,string
```

Command Stream

Sequence Number

```
AT*REF=seq,flags

AT*PCMD=seq,flags,roll,pitch,gaz,yaw

AT*CONFIG=seq,string
```

In order to avoid the drone from processing old commands, a sequence number is associated to each sent AT command, and is stored as the first number after the "equal" sign. The drone will not execute any command whose sequence number is less than the last valid received AT-Command sequence number. This sequence number is reset to 1 inside the drone every time a client disconnects from the AT-Command UDP port (currently this disconnection is done by not sending any command during more than 2 seconds), and when a command is received with a sequence number set to 1.

Command Stream

Note : According to tests, a satisfying control of the AR.Drone 2.0 is reached by sending the AT-commands every 30 ms for smooth drone movements. To prevent the drone from considering the WIFI connection as lost, two consecutive commands must be sent within less than 2 seconds.

Command Stream

Take off
Emergency Landing

AT*REF=*seq*,*flags*

Description :
Send this command to control the basic behaviour of the drone. With SDK version 1.5, only bits 8 and 9 are used in the control bit-field. Bits 18, 20, 22, 24 and 28 should be set to 1. Other bits should be set to 0.

Bits	31 .. 10	9	8	7 .. 0
Usage	Do not use	Takeoff/Land (aka. "start bit")	Emergency (aka. "select bit")	Do not use

Command Stream

Movement
Flags

AT*PCMD=*seq*,*flags*,*roll*,*pitch*,*gaz*,*yaw*

Movement Flags

Always set the flag (argument 2) bit zero to one to make the drone consider the other arguments. Setting it to zero makes the drone enter *hovering* mode (staying on top of the same point on the ground).

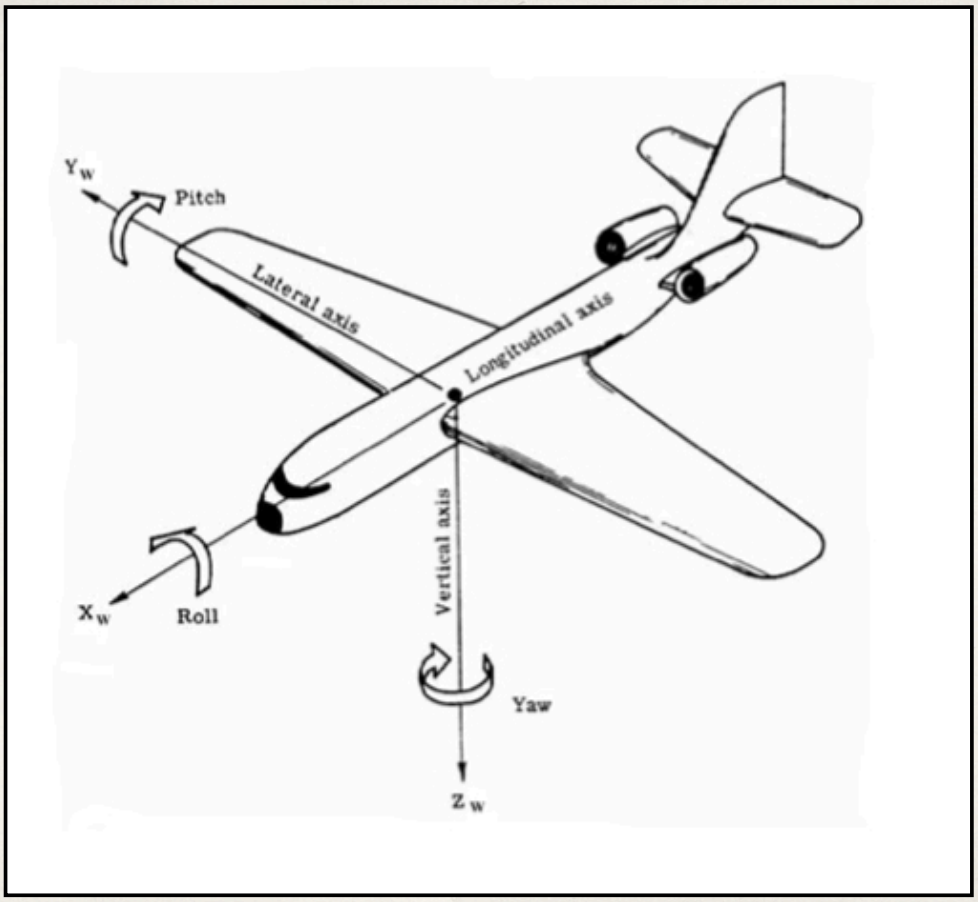
(hover mode!)

Bits	31 .. 3	2	1	0
Usage	Do not use	Absolute Control enable	Combined yaw enable	Progressive commands enable

Command Stream

Movement Control

```
AT*PCMD=seq,flags,roll,pitch,gaz,yaw
```



Floating Point Data

Let's see an example of using a *float* argument and consider that a progressive command is to be sent with an argument of -0.8 for the pitch. The number -0.8 is stored in memory as a 32-bit word whose value is $BF4CCCCD_{(16)}$, according to the IEEE-754 format. This 32-bit word can be considered as holding the 32-bit integer value $-1085485875_{(10)}$. So the command to send will be **AT***PCMD_MAG=xx,xx,−1085485875,xx,xx,xx,xx.

floating point	IEEE 32-bit Representation	Treated as a Signed Integer
-0.8 →	0xBF4CCCCD →	-1085485875

Floating Point Data

```
[-0.8].pack('g').unpack("l>").first
# => -1085485875
```

floating point	IEEE 32-bit Representation	Treated as a Signed Integer
-0.8 →	0xBF4CCCCD →	-1085485875

Floating Point Data

```
[-0.8].pack('g').unpack("l>").first
# => -1085485875
```

```
AT*PCMD=480,1,-1085485875,0,0,0
```

Command Stream

AT*CONFIG=*seq*,*key*,*value*

Light Configuration
Video Targeting

```
AT*CONFIG=831,"detect:enemy_colors","2"
AT*CONFIG=832,"detect:detect_type","10"
AT*CONFIG=833,"detect:detections_select_h","32"
AT*CONFIG=834,"leds:leds_anim",3,1073741824,5
```

```ruby
require 'argus'

drone = Argus::Drone.new
drone.start

drone.take_off
sleep 5
drone.turn_right(1.0)
sleep 2
drone.turn_left(1.0)
sleep 2
drone.hover
sleep 1
drone.land
sleep 2
drone.stop
```

<< movie >>

```ruby
require 'argus'

drone = Argus::Drone.new
drone.start

drone.take_off
sleep 5
2.times do
  drone.left(0.3)
  sleep 1
  drone.right(0.3)
  sleep 1
end
drone.hover
sleep 1
drone.land
sleep 5
drone.stop
```

<< movie >>

Command Stream

```
AT*REF=477,290718208
AT*PCMD=478,0,0,0,0,0
AT*REF=479,290718208
AT*PCMD=480,1,-1097229926,0,0,0
AT*REF=481,290718208
AT*PCMD=482,1,-1097229926,0,0,0
...
AT*REF=575,290718208
AT*PCMD=576,1,1050253722,0,0,0
```

Command Stream

Hover Mode

```
AT*REF=477,290718208
AT*PCMD=478,0,0,0,0,0
AT*REF=479,290718208
AT*PCMD=480,1,-1097229926,0,0,0
AT*REF=481,290718208
AT*PCMD=482,1,-1097229926,0,0,0
...
AT*REF=575,290718208
AT*PCMD=576,1,1050253722,0,0,0
```

Non-Hover Mode

Roll Control (go right 0.3)

Roll Control (go left 0.3)

Navigation Data

Header (FL)

NavData (VL)

Option (VL)

Option (VL)

Vision Detect (VL)

Checksum (VL)

Navigation Data

Header (FL)

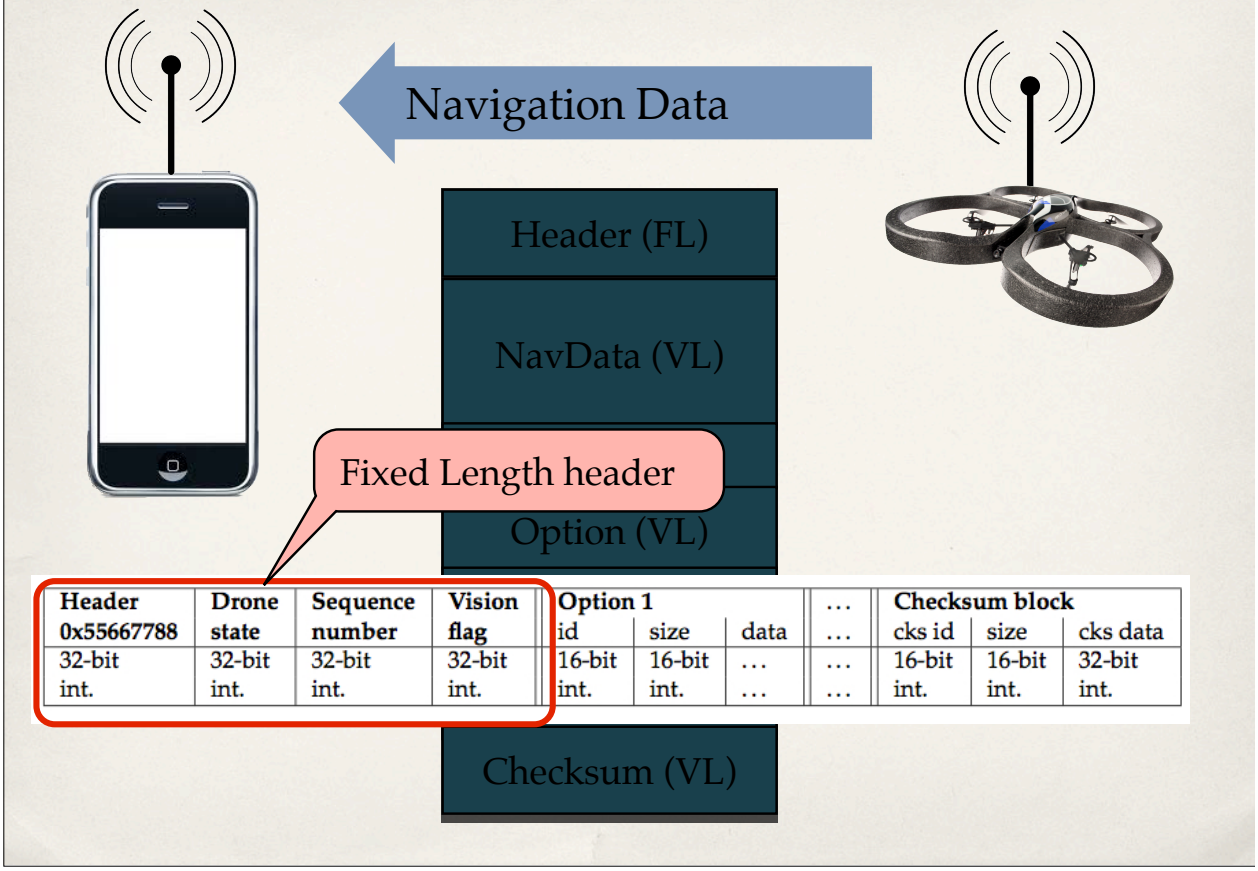
NavData (VL)

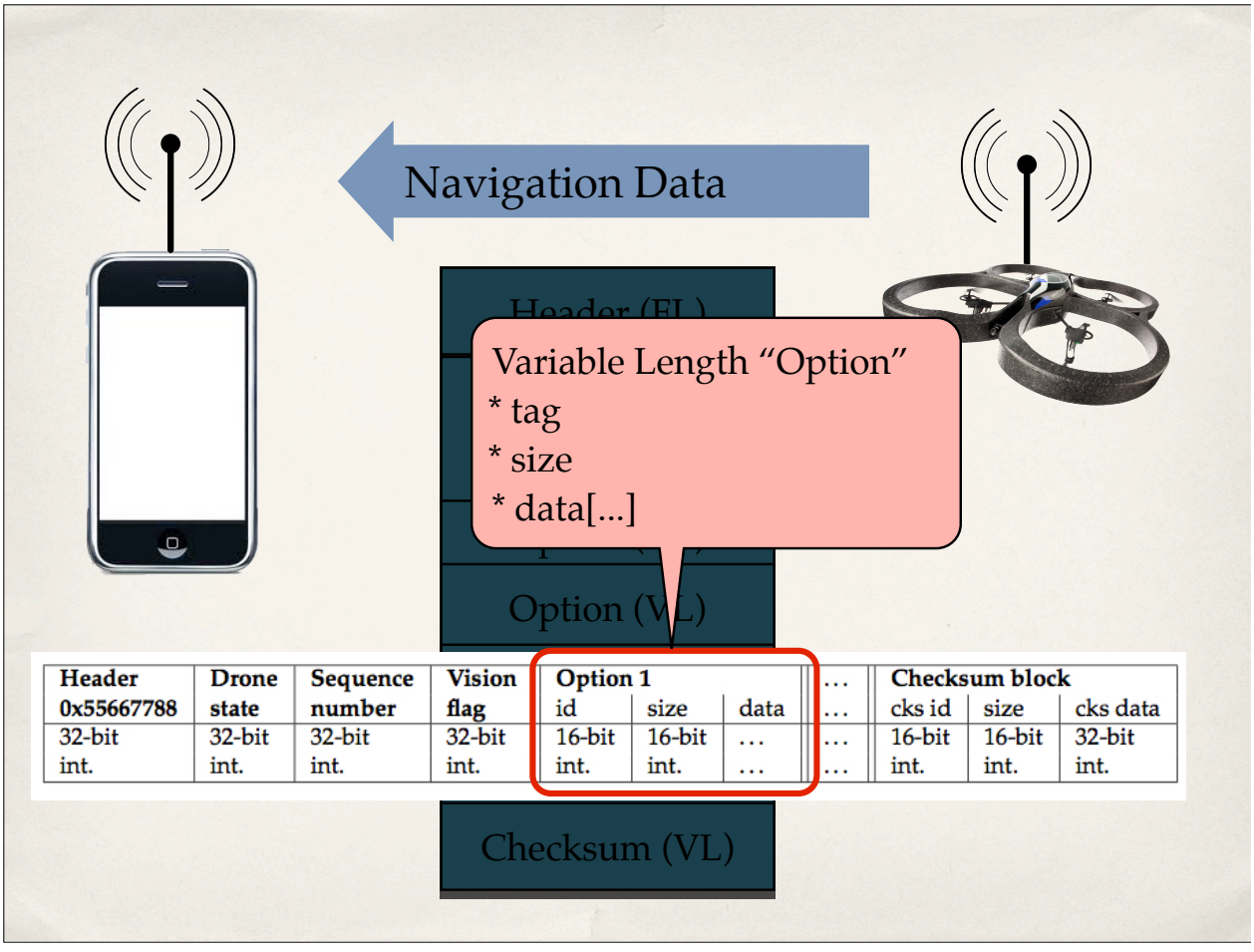
Option (VL)

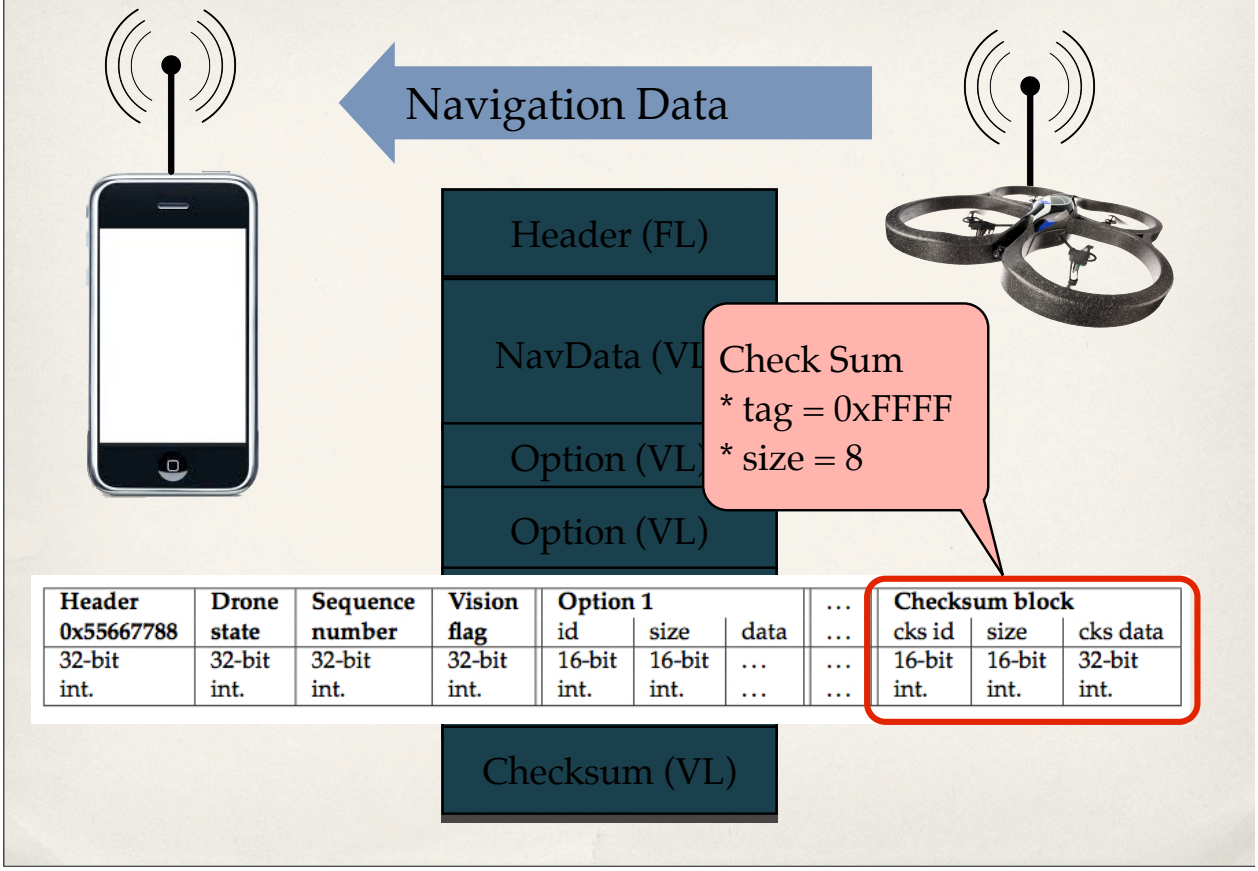
Option (VL)

Header 0x55667788	**Drone state**	**Sequence number**	**Vision flag**	**Option 1**			…	**Checksum block**		
				id	size	data	…	cks id	size	cks data
32-bit int.	32-bit int.	32-bit int.	32-bit int.	16-bit int.	16-bit int.	…	…	16-bit int.	16-bit int.	32-bit int.

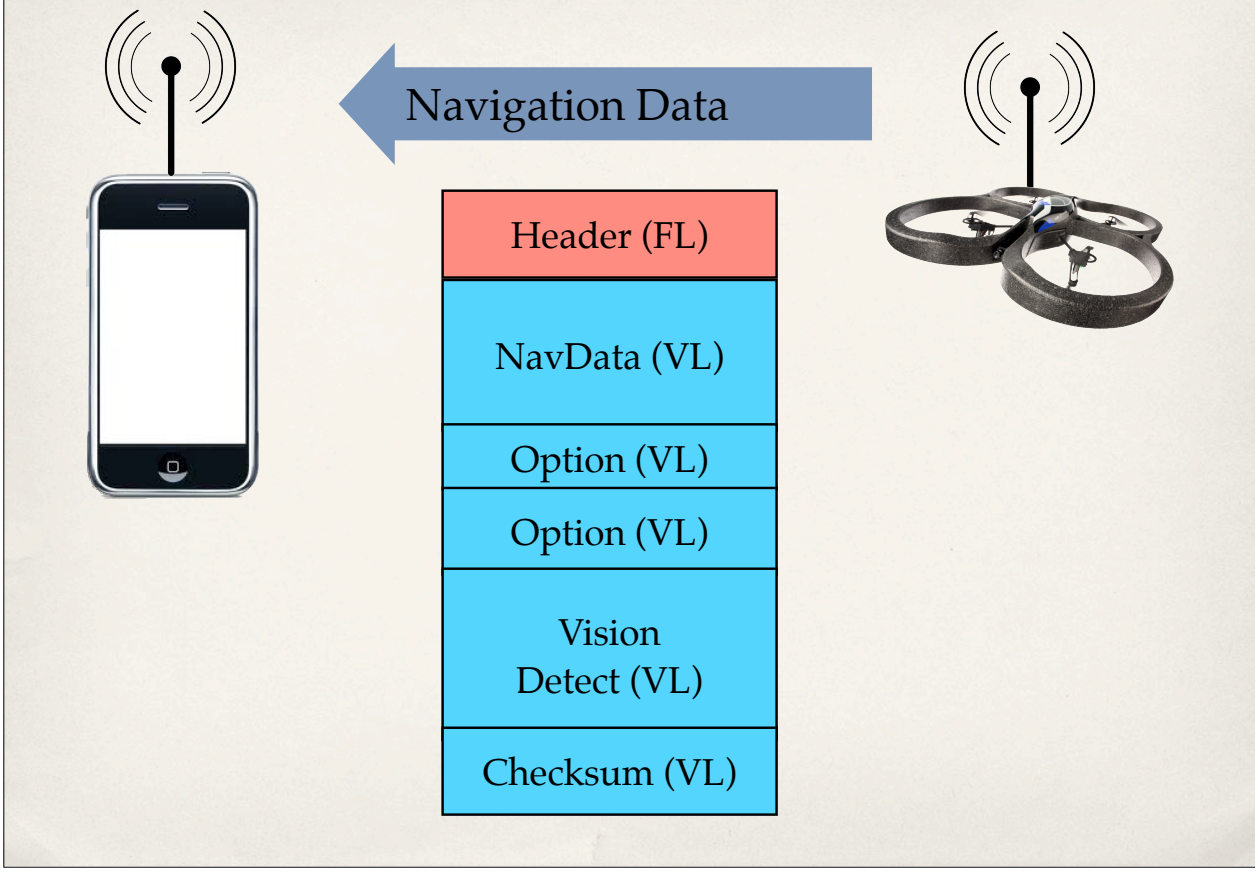
Checksum (VL)


Navigation Data
Header (FL)
NavData (VL)
Option (VL)
Checksum (VL)
Fixed Length header
Header 0x55667788
Drone state
Sequence number
Vision flag
Option 1
id
size
data
...
Checksum block
cks id
size
cks data
32-bit int.
16-bit int.


Navigation Data
Variable Length "Option"
* tag
* size
* data[...]
Option (VL)
Checksum (VL)
Header 0x55667788
Drone state
Sequence number
Vision flag
Option 1
id
size
data
Checksum block
cks id
size
cks data
32-bit int.
16-bit int.



Header 0x55667788	Drone state	Sequence number	Vision flag	Option 1			…	Checksum block		
				id	size	data	…	cks id	size	cks data
32-bit int.	32-bit int.	32-bit int.	32-bit int.	16-bit int.	16-bit int.	…	…	16-bit int.	16-bit int.	32-bit int.



Fixed Length Header

```
typedef struct _navdata_t {
  uint32_t    header;           /* 0x55667788 */
  uint32_t    ardrone_state;
  uint32_t    sequence;
  bool_t      vision_defined;

  navdata_option_t  options[1];
}_ATTRIBUTE_PACKED_ navdata_t;
```

Navigation Data

Header (FL)
NavData (VL)
Option (VL)
Option (VL)
Vision Detect (VL)
Checksum (VL)

```
typedef struct _navdata_demo_t {
  uint16_t    tag;                /*!< Navdata block ('
  uint16_t    size;               /*!< set this to the

  uint32_t    ctrl_state;                 /*!< Flying
  uint32_t    vbat_flying_percentage; /*!< batter

  float32_t   theta;                      /*!< UAV's
  float32_t   phi;                        's
  float32_t   psi;                        's

  int32_t     altitude;                   /*!< UAV's

  /* ... */
}_ATTRIBUTE_PACKED_ navdata_demo_t;
```

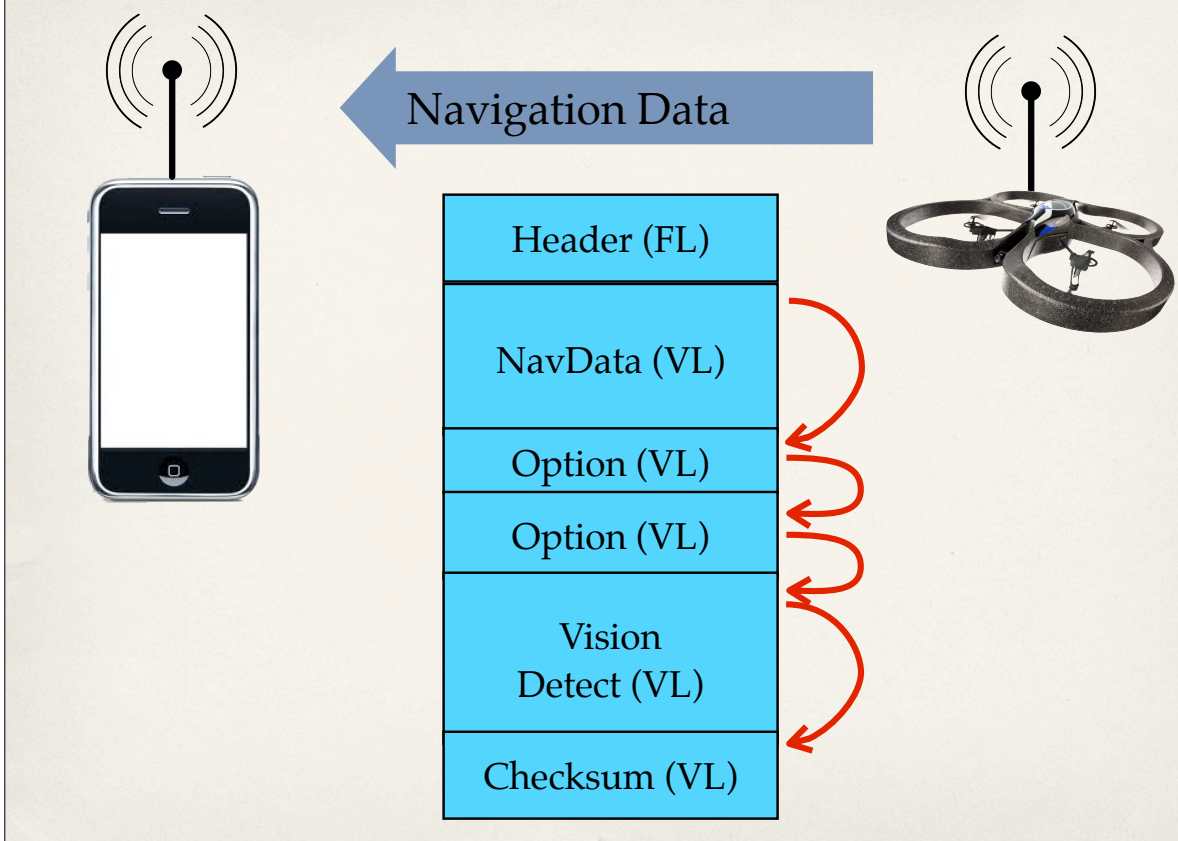
EVERY variable length section begins with tag and size.

```
typedef struct _navdata_demo_t {
  uint16_t    tag;                /*!< Navdata block ('
  uint16_t    size;               /*!< set this to the

  uint32_t    ctrl_state;                 /*!< Flying
  uint32_t    vbat_flying_percentage; /*!< batter

  float32_t   theta;                      /*!< UAV's
  float32_t   phi;                        /*!< UAV's
  float32_t   psi;                        /*!< UAV's

  int32_t     altitude                    's

  /* ... */
}_ATTRIBUTE_PACKED_ navdata_demo_t;
```

tag identifies type of "option". (DEMO=0, VISION_DETECT=16)

```
typedef struct _navdata_demo_t {
  uint16_t    tag;           /*!< Navdata block ('
  uint16_t    size;          /*!< set this to the

  uint32_t    ctrl_state;              /*!< Flying
  uint32_t    vbat_flying_percentage;  /*!< batter

  float32_t   thet;                    /*!< UAV's
  float32_t   phi;                     /*!< UAV's
  float32_t   psi;                     /*!< UAV's

  int32_t     altit                    s

  /* ... */
}_ATTRIBUTE_PACKED_ navdata_demo_t;
```

size is the number of bytes in the option, so you can find the next option.



```
typedef struct _navdata_demo_t {
  uint16_t    tag;           /*!< Navdata block ('
  uint16_t    size;          /*!< set this to the

  uint32_t    ctrl_state;             /*!< Flying
  uint32_t    vbat_flying_percentage; /*!< batter

  float32_t   theta;                  /*!< UAV's
  float32_t   phi;                    /*!< UAV's
  float32_t   psi;                    /*!< UAV's

  int32_t     altitude;               /*!< UAV's

  /* ... */
}_ATTRIBUTE_PACKED_ navdata_demo_t;
```

```
typedef struct _navdata_demo_t {
  uint16_t    tag;
  uint16_t    size;

  uint32_t    ctrl_state;             /*!< Flying
  uint32_t    vbat_flying_percentage; /*!< batter

  float32_t   theta;                  /*!< UAV's
  float32_t   phi;                    /*!< UAV's
  float32_t   psi;                    /*!< UAV's

  int32_t     altitude;               /*!< UAV's

  /* ... */
}_ATTRIBUTE_PACKED_ navdata_demo_t;
```

Start with a C declaration

```
class NavOptionDemo < NavOption
  include CFields

  uint32_t    ctrl_state;
  uint32_t    vbat_flying_pe

  float32_t   theta;
  float32_t   phi;                    /*!< UAV's
  float32_t   psi;                    /*!< UAV's

  int32_t     altitude;               /*!< UAV's

  /* ... */
end
```

Turn into a Ruby Class
(leave off tag and size)

```
class NavOptionDemo < NavOpti
  include CFields

  uint32_t    :ctrl_state             /*!< Flying
  uint32_t    :vbat_flying_percentage /*!< battery

  float32_t   :theta                  /*!< UAV's
  float32_t   :phi                    /*!< UAV's
  float32_t   :psi                    /*!< UAV's

  int32_t     :altitude               /*!< UAV's

  /* ... */
end
```

Turn fields into symbols

```
class NavOptionDemo < NavOption
  include CFields

  uint32_t     :ctrl_state              # Flying st
  uint32_t     :vbat_flying_percentage  # battery v

  float32_t    :theta                   # UAV's pit
  float32_t    :phi                     # UAV's rol
  float32_t    :psi                     # UAV's yaw

  int32_t      :altitude                # UAV's alt

  # ...
end
```

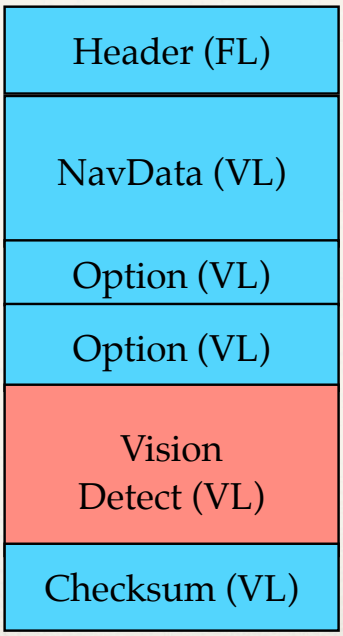
Switch to Ruby Comments

```
class NavOptionDemo < NavOption
  include CFields

  uint32_t     :ctrl_state              # Flying st
  alias :control_state :ctrl_state
  uint32_t     :vbat_flying_percentage  # battery v
  alias :battery_level :vbat_flying_percentage

  float32_t    :theta                   # UAV's pit
  float32_t    :phi                     # UAV's rol
  float32_t    :psi                     # UAV's yaw

  int32_t      :altitude                # UAV's alt

  # ...
end
```

Add convenient aliases.

```
class NavOptionDemo < NavOption
    include CFields

    uint32_t :ctrl_state                # Flying state (landed, fl
    alias :control_state :ctrl_state
                                        # hovering, etc.) defined
                                        # CTRL_STATES enum.
    uint32_t :vbat_flying_percentage    # battery voltage filtered
    alias :battery_level :vbat_flying_percentage

    float32_t :theta                    # UAV's pitch in milli-deg
    float32_t :phi                      # UAV's roll  in milli-deg
    float32_t :psi                      # UAV's yaw   in milli-deg

    int32_t :altitude                   # UAV's altitude in centi

    # ...
end
```

Navigation Data



```
typedef struct _navdata_vision
  uint16_t   tag;
  uint16_t   size;

  uint32_t   nb_detected;
  uint32_t   type[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   xc[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   yc[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   width[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   height[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   dist[NB_NAVDATA_DETECTION_RESULTS];
  float32_t  orientation_angle[NB_NAVDATA_DETECTION_RESULTS];
  matrix33_t rotation[NB_NAVDATA_DETECTION_RESULTS];
  vector31_t translation[NB_NAVDATA_DETECTION_RESULTS];
  uint32_t   camera_source[NB_NAVDATA_DETECTION_RESULTS];
}_ATTRIBUTE_PACKED_ navdata_vision_detect_t;
```

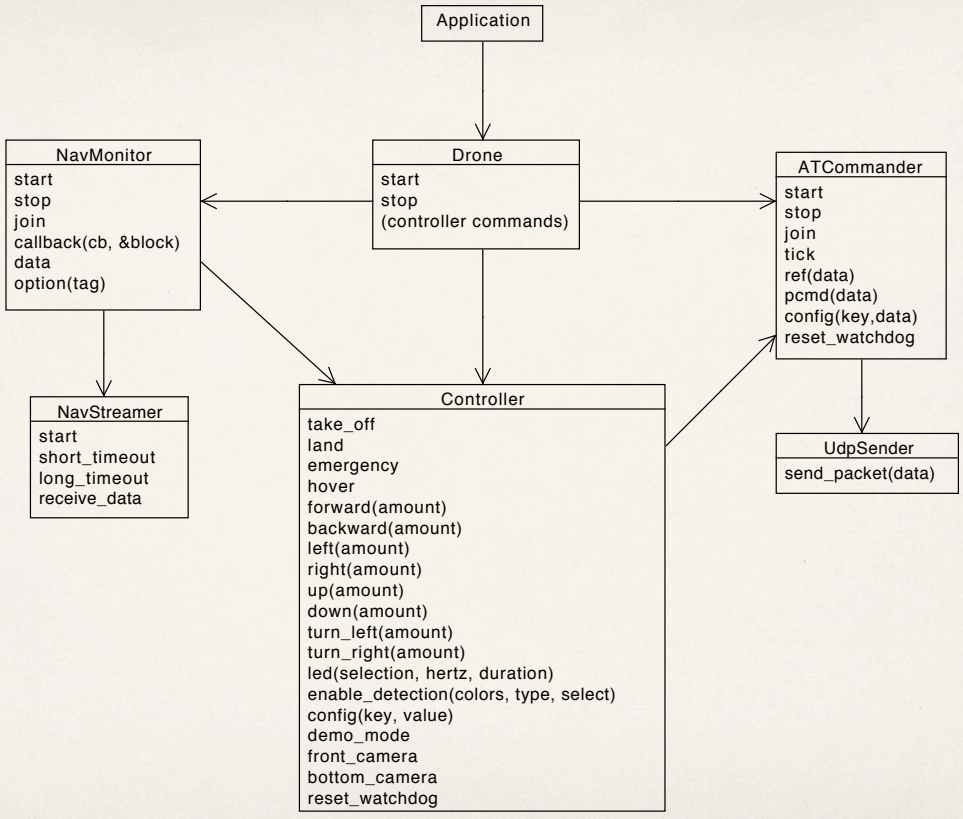
Visual Target Detection!
X/Y position in camera field.

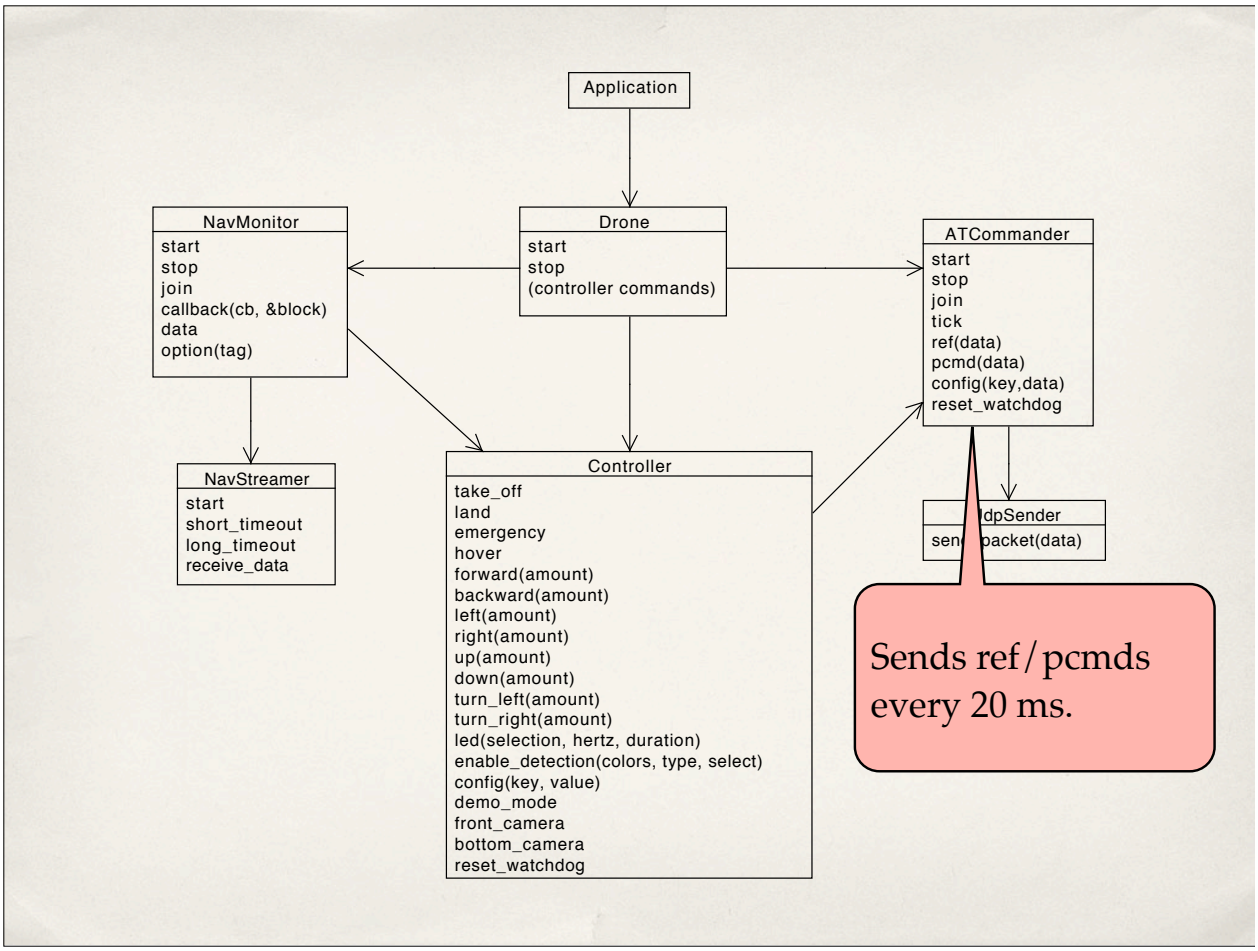
Navigation Data

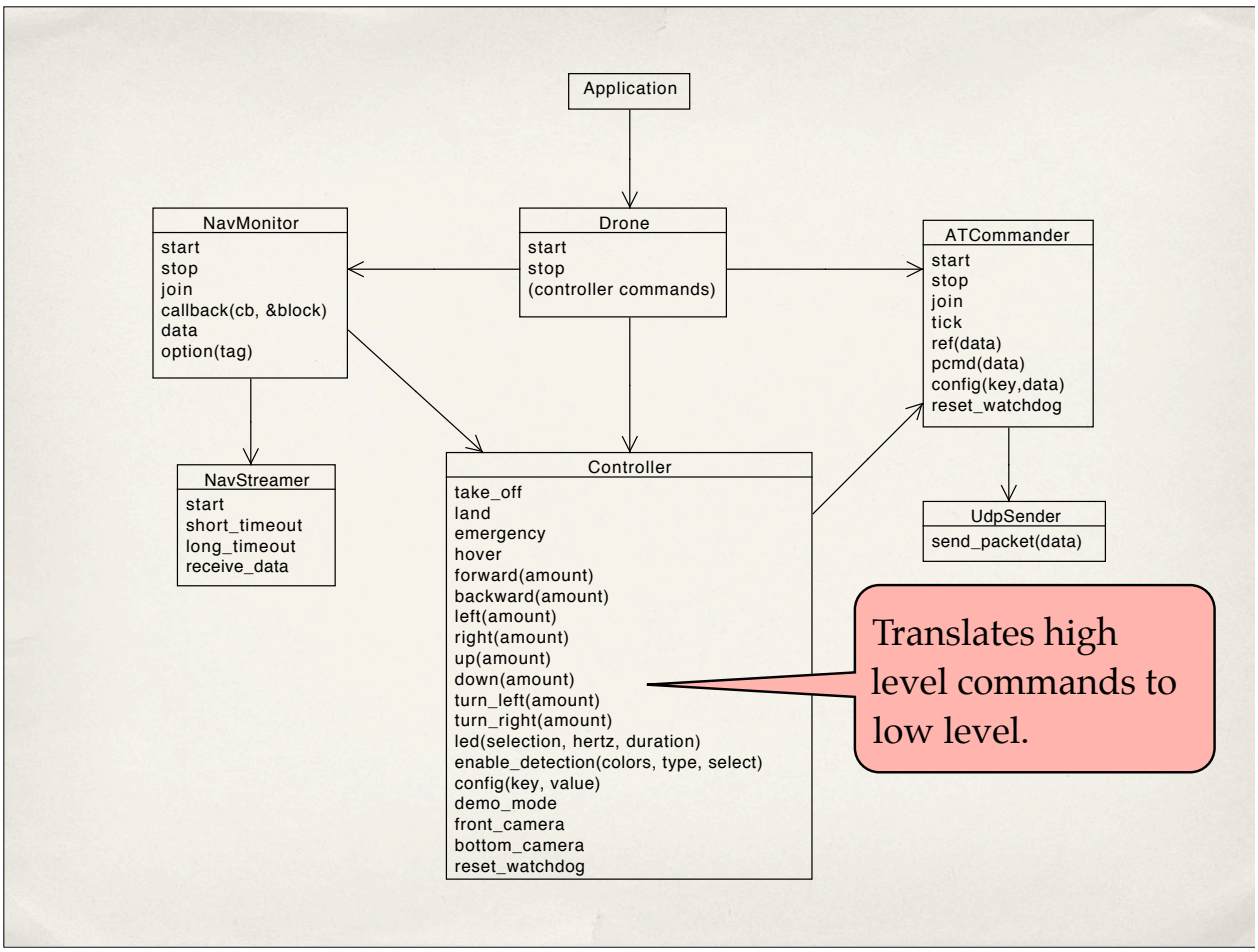
Header (FL)
NavData (VL)
Option (VL)
Option (VL)
Vision Detect (VL)
Checksum (VL)

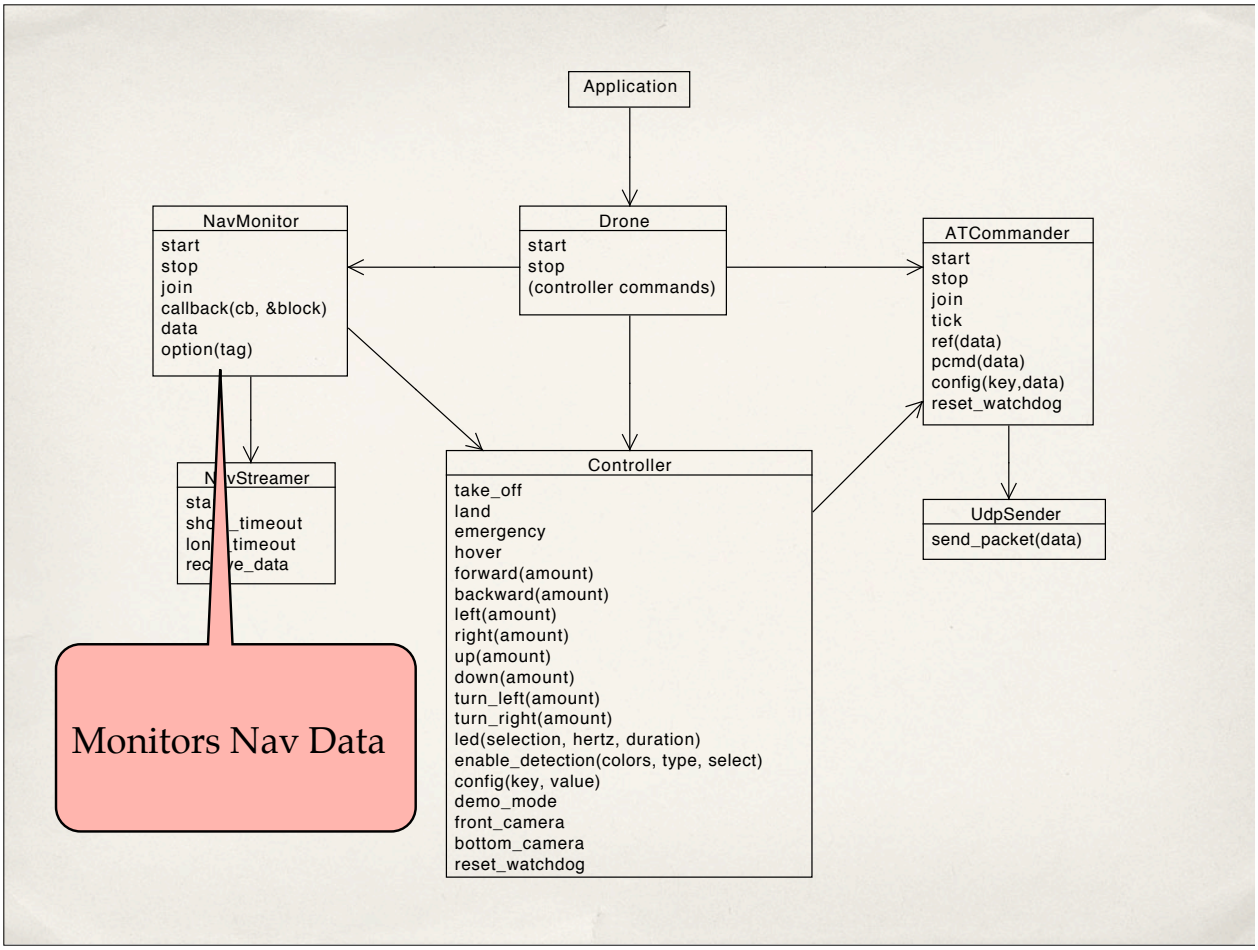
```
typedef struct _navdata_cks_t {
  uint16_t  tag;
  uint16_t  size;

  // Checksum for all navdatas (including options)
  uint32_t  cks;
}_ATTRIBUTE_PACKED_ navdata_cks_t;
```

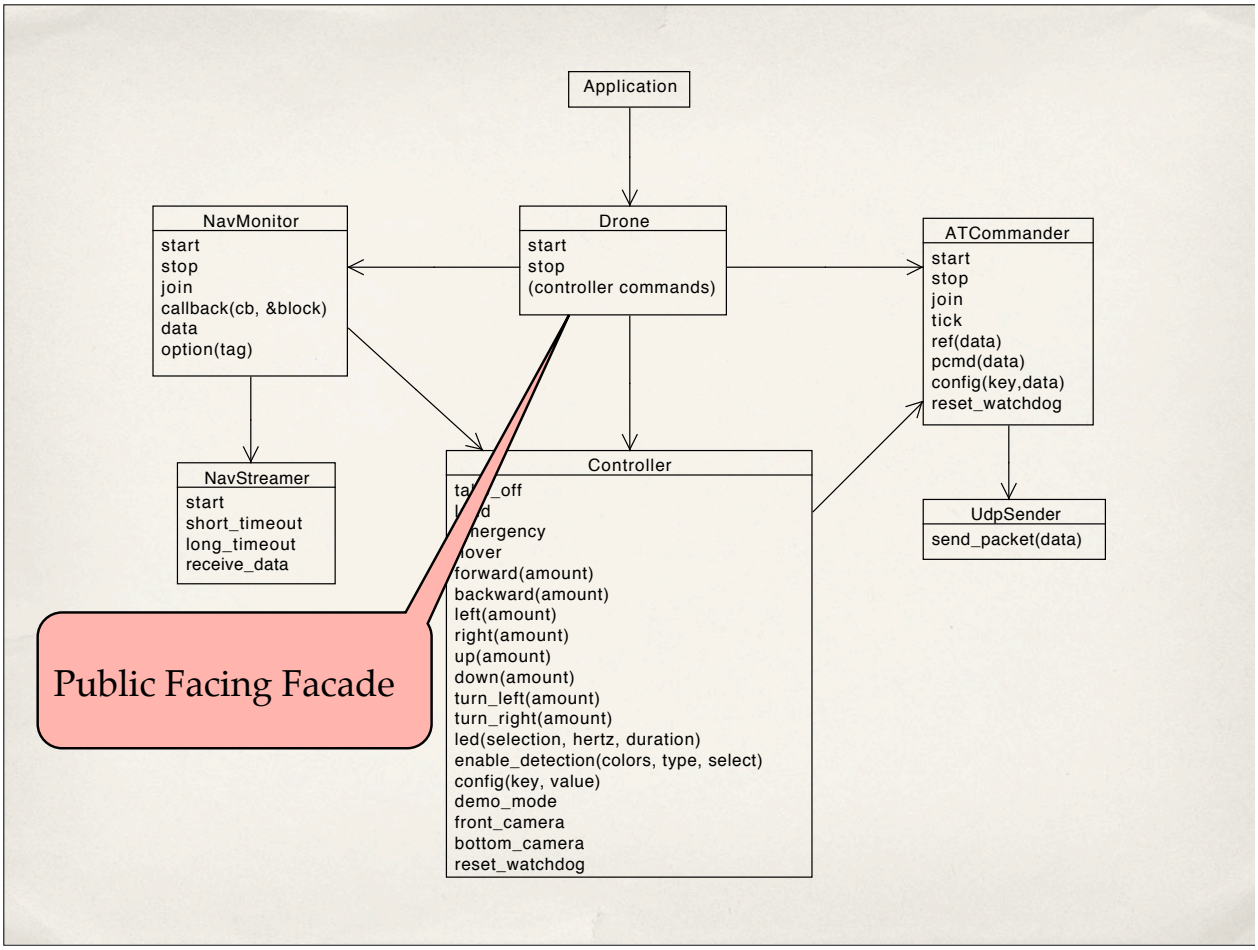
Check sum for entire packet




Application
Drone
start
stop
(controller commands)
NavMonitor
start
stop
join
callback(cb, &block)
data
option(tag)
ATCommander
start
stop
join
tick
ref(data)
pcmd(data)
config(key,data)
reset_watchdog
NavStreamer
start
short_timeout
long_timeout
receive_data
Controller
take_off
land
emergency
hover
forward(amount)
backward(amount)
left(amount)
right(amount)
up(amount)
down(amount)
turn_left(amount)
turn_right(amount)
led(selection, hertz, duration)
enable_detection(colors, type, select)
config(key, value)
demo_mode
front_camera
bottom_camera
reset_watchdog
dpSender
packet(data)
Sends ref/pcmds every 20 ms.


Application
Drone
start
stop
(controller commands)
NavMonitor
start
stop
join
callback(cb, &block)
data
option(tag)
ATCommander
start
stop
join
tick
ref(data)
pcmd(data)
config(key,data)
reset_watchdog
NavStreamer
start
short_timeout
long_timeout
receive_data
Controller
take_off
land
emergency
hover
forward(amount)
backward(amount)
left(amount)
right(amount)
up(amount)
down(amount)
turn_left(amount)
turn_right(amount)
led(selection, hertz, duration)
enable_detection(colors, type, select)
config(key, value)
demo_mode
front_camera
bottom_camera
reset_watchdog
UdpSender
send_packet(data)
Translates high level commands to low level.


Application
NavMonitor
start
stop
join
callback(cb, &block)
data
option(tag)
Drone
start
stop
(controller commands)
ATCommander
start
stop
join
tick
ref(data)
pcmd(data)
config(key,data)
reset_watchdog
NavStreamer
Controller
take_off
land
emergency
hover
forward(amount)
backward(amount)
left(amount)
right(amount)
up(amount)
down(amount)
turn_left(amount)
turn_right(amount)
led(selection, hertz, duration)
enable_detection(colors, type, select)
config(key, value)
demo_mode
front_camera
bottom_camera
reset_watchdog
UdpSender
send_packet(data)
Monitors Nav Data


Application
NavMonitor
start
stop
join
callback(cb, &block)
data
option(tag)
Drone
start
stop
(controller commands)
ATCommander
start
stop
join
tick
ref(data)
pcmd(data)
config(key,data)
reset_watchdog
NavStreamer
start
short_timeout
long_timeout
receive_data
Controller
forward(amount)
backward(amount)
left(amount)
right(amount)
up(amount)
down(amount)
turn_left(amount)
turn_right(amount)
led(selection, hertz, duration)
enable_detection(colors, type, select)
config(key, value)
demo_mode
front_camera
bottom_camera
reset_watchdog
UdpSender
send_packet(data)
Public Facing Facade

```ruby
def call(data)
   return if @done
   data.options.each do |opt|
     if opt.is_a?(Argus::NavOptionVisionDetect)
       if opt.detected_count == 0
         drone.hover
       elsif opt.detected_count > 0
         d = opt.detections.first
         if d.x < 400
           drone.turn_left(0.2)
         elsif d.x > 600
           drone.turn_right(0.2)
         else
           drone.hover
         end
       end
     end
   end
 end
```

<< movie >>

Challenges

Follow Me

Spacial Positioning

Follow the Laser

Resources

Where to get Drones

http://ardrone2.parrot.com/

http://amazon.com
(search for “Parrot AR Drone”)

http://diydrones.com/

Drone Software

https://github.com/jimweirich/argus

https://artoo.io

https://github.com/gigasquid/clj-drone

https://github.com/felixge/godrone

ETH Zurich

Flying Machine Arena

http://www.flyingmachinearena.org/videos/

http://cincycopter.org

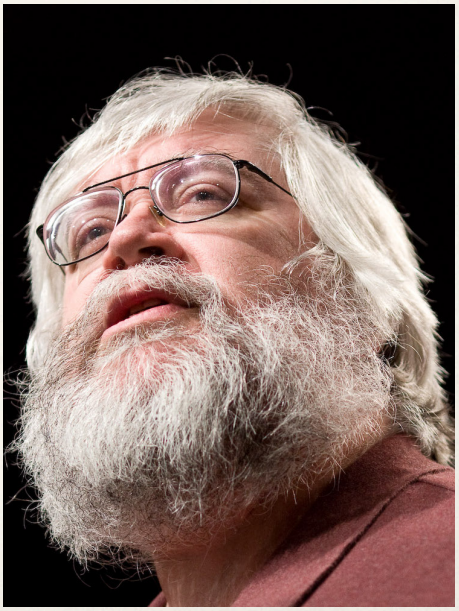
Sold Out
(in 3 hours)

NODECOPTER CINCINNATI
NODECOPTER — SAT SEPT 7TH

Questions?

Thank You

Jim Weirich
Chief Scientist



jim@neo.com
@jimweirich