Terry Stillone
Software Architecture
terry@originware.com

Avatar Interface

Speech Recognition

Speech Synthesis

The Dronenaut App Talk

Scalable Notification Fabric

SNF Software Design

Originware Technology

Sydney CocoaHeads August 2017

What is the talk about ?

About the Dronenaut App and associated tech .

Avatar UI

Software Architecture

Speech Recog

SNF Design

Speech Synth

Software Visualisation

Drones

Drone Flying

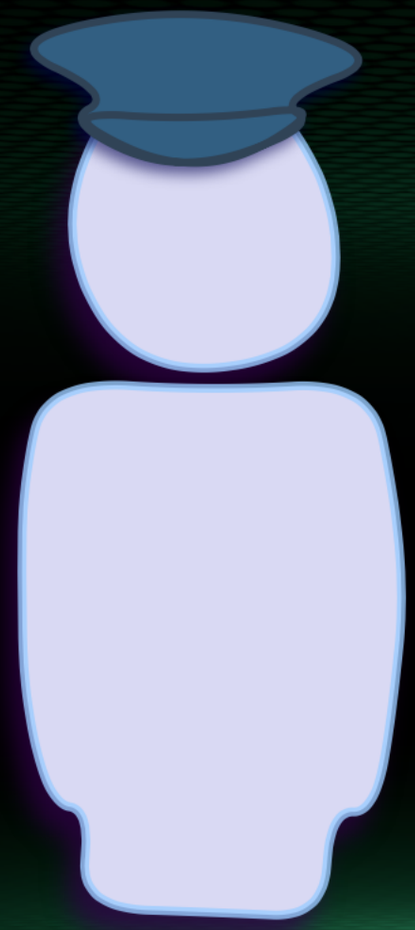
Small amount on the social aspects of Drones

Dronenaut is a Demonstrator for SNF Design and associated SDK

Dronenaut

(for Parrot™ Minidrone)

0.25



Drone

Drone name: Mambo_325346

Model	*Mambo*
Status	*Ready*
Battery	*60%*
Signal	*-37db*
Serial	*PI040408AA6J325346*
Software Version	*1.0.17*
Motion Calibration Parameters	*(1.0, 1.0, 1.0, 1.0)* ▷

Battery: 60%

Drone | Flight Log | Settings | Explain | Help | Purchases | About

Dronenaut

(for Parrot™ Minidrone)

0.25

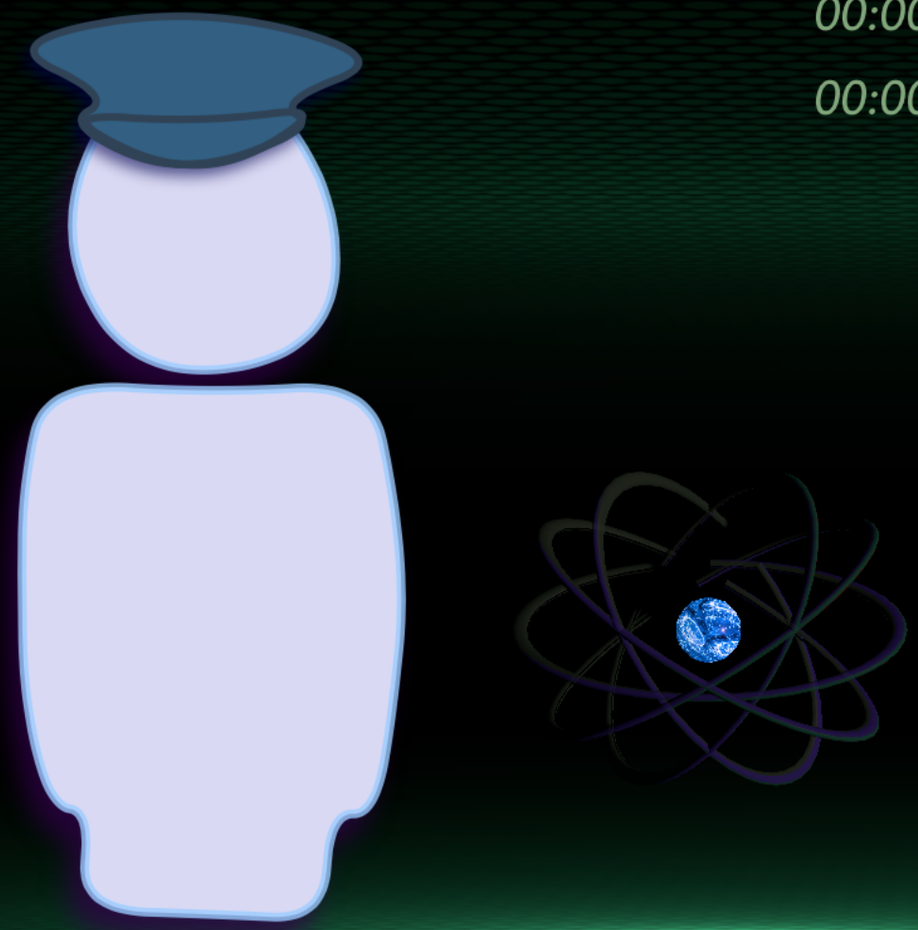
Flight Log

00:00:01 *Flight control ready*

00:00:04 *Scanning for drones*

00:00:17 *Auto connecting with:Mambo drone*

00:00:19 *Battery: 65%*



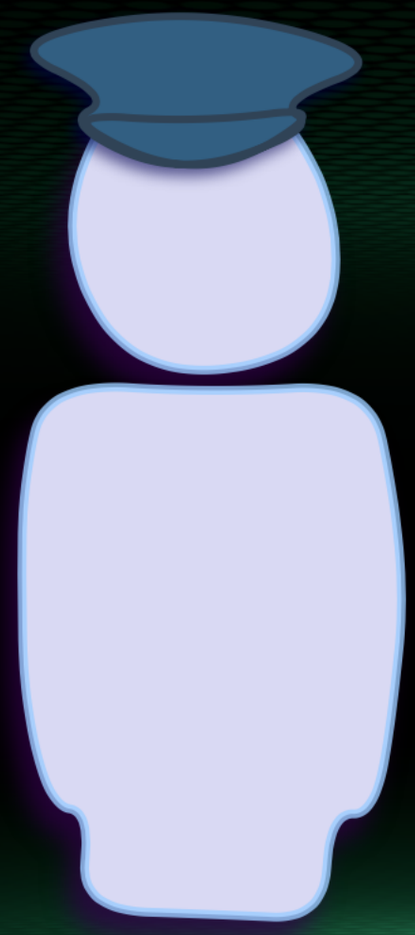
Battery: 65%

Drone | Flight Log | Settings | Explain | Help | Purchases | About

Dronenaut

(for Parrot™ Minidrone)

0.25



Battery: 65%

Settings

Audio IO	iPad Microphone ⋯› Speaker
Speech Rate	0.4
Vocal Feedback Level	medium
Max Listen Duration	6.0 sec
Speaking Voice	en-GB (Daniel)
Recognition Language	en-AU (Australian)
Drone Auto Connect	Enabled
Drone Motion Responsiveness	max (1.0)
Max Altitude	2.0 m
Max Speed	0.5 m/s

Google Speech Recognition Time

Remaining Allowance	3480 sec
Session Elapsed Recog Time	0 sec

Drone | Flight Log | Settings | Explain | Help | Purchases | About

Components of the App

Speech Recog: Google Speech API

Apple Speech Synth Voices

Design in SNF

3D in SceneKit

Dronenaut figure modelled in Apple Motion
and then converted to Quad Beziers.

Custom Minidrone protocol stack in SNF

Dronenaut animation performed by line morphing.

Language: Swift

Parrot Minidrone Section

The Parrot Minidrone / Specifications

Specs

Models
Airborne Night (includes flashing leds)
Airborne Cargo (can carry objects)
Rolling Spider
Hydrofoil (with hydrofoil full)
Mambo (with catcher and canon)

Sensors:
IMU (gyro, accelerometer)
Barometer
Ultrasonic ground sensor

Battery
550 mAh LiPo
2.5A max current

OS
Cut down version of Linux called Delos

Vertical Camera
60FPS 640x480

CPU
ARM A9-800Mhz
1GB of DDR Ram
Flash Drive

Network
Bluetooth LE4.0
Range: 10 - 20 m

The Parrot Minidrone / Comms Protocol

Comms Arranged by Function, Represented by Channel

Channel	Function	Channel Direction
Status:	Drone Status	Drone -> Controller
Sender:	Commands	Controller -> Drone
Receiver:	Command Results	Drone -> Controller
PCMD:	PCMD	Drone -> Controller
FTP21:	Bulk Data Transfer (Photo data)	Drone <-> Controller
FTP51:	Bulk Data Transfer (Photo data)	Drone <-> Controller

The Parrot Minidrone / Channel Packet Protocol

Packet Types

* Data Packet
* Ack Packet

Packet Data Format

<Byte: Packet Type>
<Byte: Sequence number>
<Byte Sequence: Data>

The Parrot Minidrone / Comms Protocol (Events)

Events: (Status/Receiver channels) Drone -> Controller

Alerts:	Battery low, Motor cut out.
Battery:	Battery level
Charging State:	Charge rate.
Flight Status:	Landed, Taking off, Hovering, Landing, Rolling, Auto Takeoff.
Media Events:	Picture taken, Media changes.

The Parrot Minidrone / Comms Protocol (Cmds)

Commands: (Sender Channel) Controller -> Drone

Config: Set config

Flight Mode: Trim, Take off, Land, Auto Take off

Settings: Set light intensity

Picture: Take picture, transfer photo.

Manoeuver: Flip (back, forward, left, right)

Lights: Set light pattern (flash, oscillate)

Misc: Disconnect, Reboot, Debug Modes.

Config

Max Rotation Speed
Max Horizontal Speed
Max Vertical Speed
Auto Cutout
Country
Date/Time
Accessories

The Parrot Minidrone / Comms Protocol (Cmds)

Command Replies: Drone -> Controller

(Receiver/Status channel)

Config: Config settings

Lights: Headlight intensity change.

Sensor: Sensor list and availability.

Config

Max Rotation Speed
Max Horizontal Speed
Max Vertical Speed
Auto Cutout
Country
Date/Time
Accessories
Software version

The Parrot Minidrone / Comms Protocol (Motion)

Motion Command: (on PCMD Channel)
Controller -> Drone

Single PCMD command:

Set Pitch Angle (-100 - 100)
Roll Angle (-100, 100),
Yaw Angle(-100, 100)
Gaz (-100, 100)

Sets Motor Controls

Can be pulsed to with in 20 - 50 ms.

Max speed in the order of 3m/s

Also used as a keep alive event by the Drone.

INDEPENDENT

Man arrested for landing 'radioactive' drone on Japanese Prime Minister's roof

A man has been arrested in Fukui, western Japan, for landing a drone that was carrying a small amount of radioactive sand on the roof of the Japanese Prime Minister's home.

40-year old Yasuo Yamamoto turned himself in to police late on Friday night, claiming he landed the drone on Prime Minister Shinzo Abe's roof in protest against the Japanese government's nuclear energy policy.

Fukui is home to around a quarter of Japan's 48 nuclear reactors - all are offline after the 2011 Fukushima disaster, but Mr Abe's government wants to restart as many of the reactors as possible.

Officials cover the DJI Phantom II drone before a proper containment crew arrives. Dated: April 2015

Drone Flying Section

Flying Demonstration

Turning demo by o’clock/degrees

Taking Picture

Flipping

Steer Square

Auto takeoff

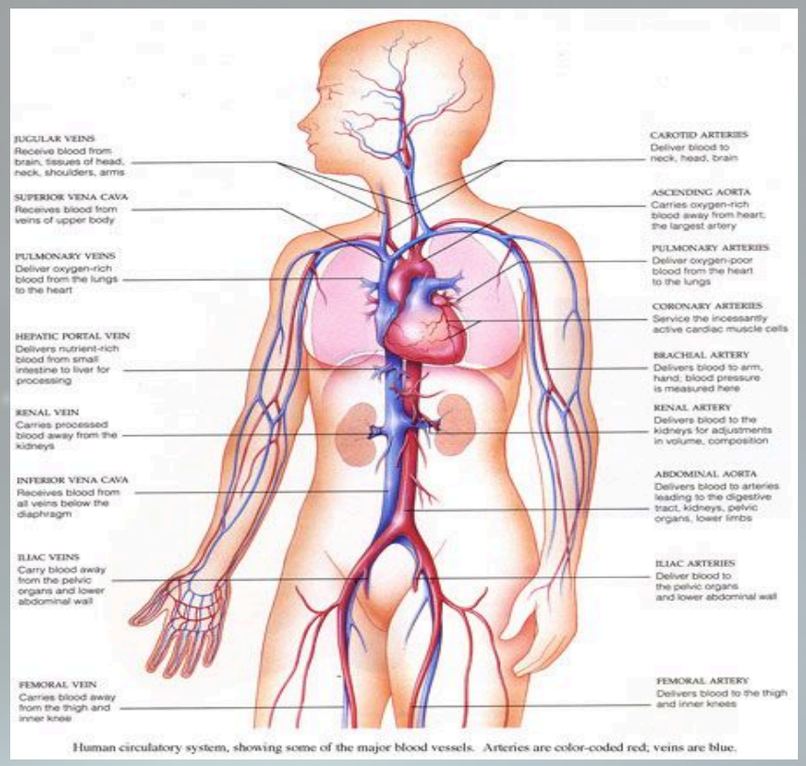
Manual flight

- lateral mode
- steering mode
- turning mode

Introduction to the Scalable Notification Fabric Design Method

The Design Principle - Example

Example: The human body (Structural Level Of Detail)



Human circulatory system, showing some of the major blood vessels. Arteries are color-coded red; veins are blue.

Arm Muscle Control System

Supervision: Brain

Arm

Arm Muscles

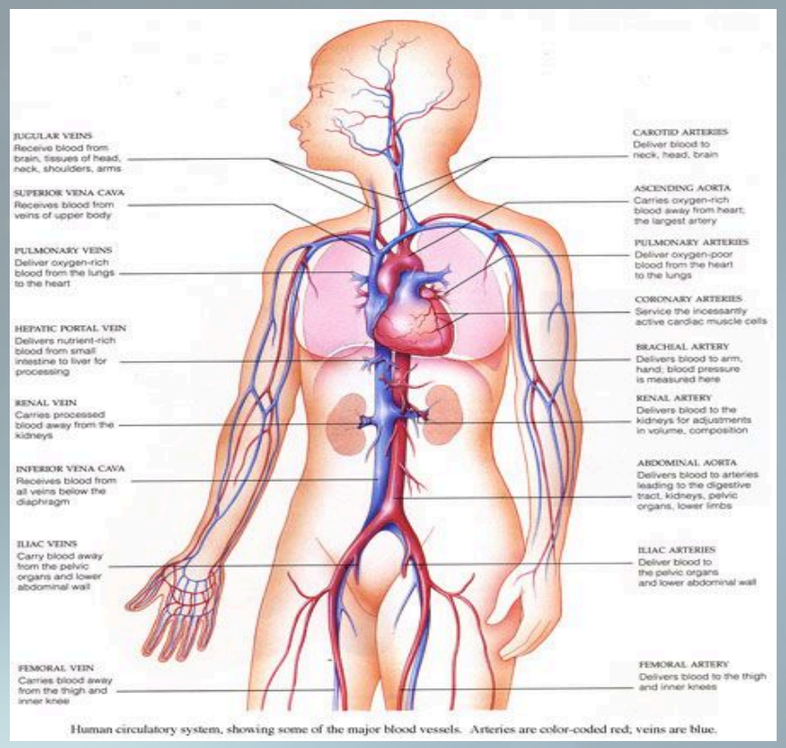
Arm Sense Perceptors

The Nervous System is the transport mechanism for forward signalling

Back signalling is performed by sense preceptors and transported back through the nervous system.

The Design Principle - Example

Example: The human body (Organ and Extra-Cell Level Of Detail)



Human circulatory system, showing some of the major blood vessels. Arteries are color-coded red; veins are blue.

Pineal Gland signals by growth hormone notification.

Pineal Gland

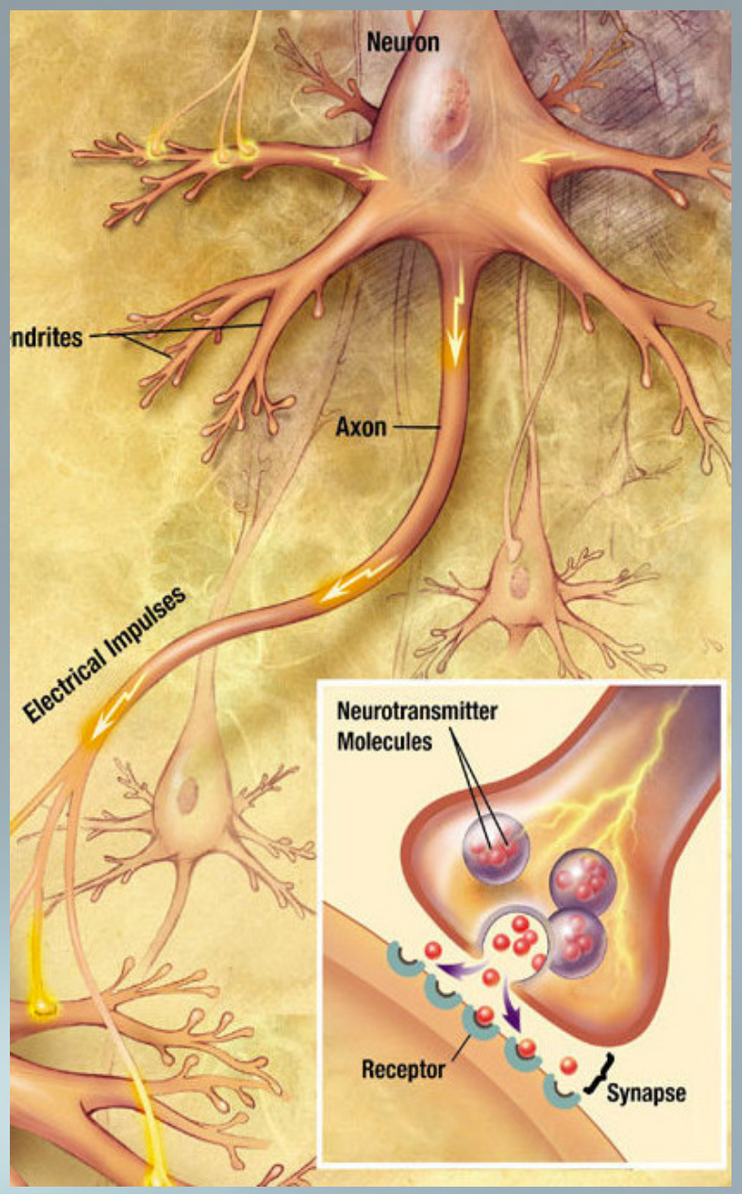
Organ/Cell System

The Arterial System is the transport mechanism for forward signalling growth hormones

Back-signalling hormones regulate the gland activity

The Design Principle - Example

Example: The human body (Synapse Level Of Detail)



Synapse signalling by
neuro transmitter notification.

Source Synapse Axion

Target Synapse Dentrite

The
Synaptic Cleft
is the transport
pathway
for the
signalling
neuro transmitter

The SNF Design Principle Part 1/2

1. Defines the structural design levels of detail (LOD)

Example Behaviour LODs:

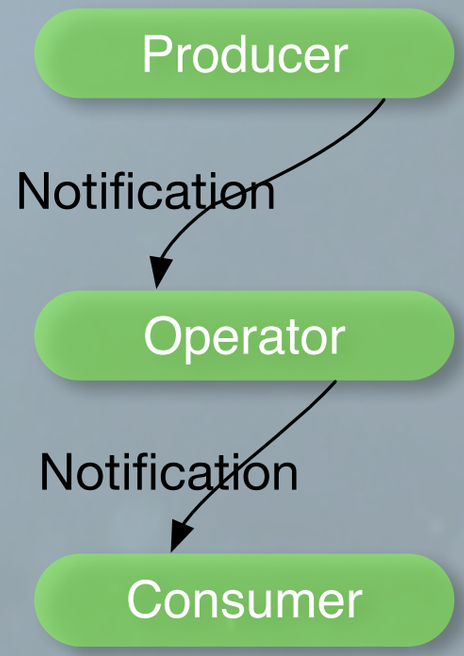
Macro Behaviour LOD
Micro Behaviour LOD

Example Embedded LODs:

Supervision LOD
Device Controller LOD

2. Defines the processing elements.

Processing elements message other elements with Notifications



3. Defines operational scopes for individual processing elements.

Example Network Scopes:

1. Device Discovery
2. Peripheral Session
3. Peripheral Service

Example App Scopes:

1. Initialisation Layer
2. Service Discovery Layer
3. Presentation Layer
4. Application Layer

The SNF Design Principle Part 2/2

4. Defines notification transport pathways and the interaction level for the pathway.

5. Defines the Notification enumerations for the pathway.

Interaction Levels:

>> Single way
>> Bidirectional

The transport mechanism is normally some type of notification channel/stream.

Example Enumeration

```
>> beginScope
    >> doSomething(params)
    >> discoverDevices(replyBackClosure)
    >> beginSessionScope(discoveredDevice)
>> endScope
```

FPV Drone Racing

FVP (First Person View) quadcopter flying with realtime video feeds

Started in France 2014, when a small group of enthusiasts got together for race through forest and posted the feeds on the net.

2016: Dubai hosted a Drone Racing match with a net purse of $US1M

In the US drone racing is being prompted in the same way NFL is.

Speech Recognition Model Design Scenario

(To demonstrate the SNF Design Process)

Speech Recognition Services with Available SDKs:

- Google Speech API
- Apple Siri
- Amazon Alexa
- Microsoft Cortanta
- Nuance (Dragon)
- Viv (Bought by Samsung)

Speech Recognition

Problems:

- Latency
  - Systems online.
  - Use continuous audio feed.
- Background noise
- Background chatter
  - Use directional audio signal processing.
- Identify false positives
  - Adaptive learning.
- Content context
  - Supply expected words.

Google Speech API

Service URL: googleapis.l.google.com.

URL: located in MountainView CA, ping time ~16mSec

Recog Latency: 3 - 4sec (generally)

Audio Feed: 16Kbps mono 16 bit channel.

Performs continuous recognition from the time of feed commencement.

Replies with running possible matches and then final match result.

Cost: approx $AUS 1.16 per hour of recog feed time.

Google Speech API SDK

Uses Google Protocol Buffers (gRPC)

Uses SSL security

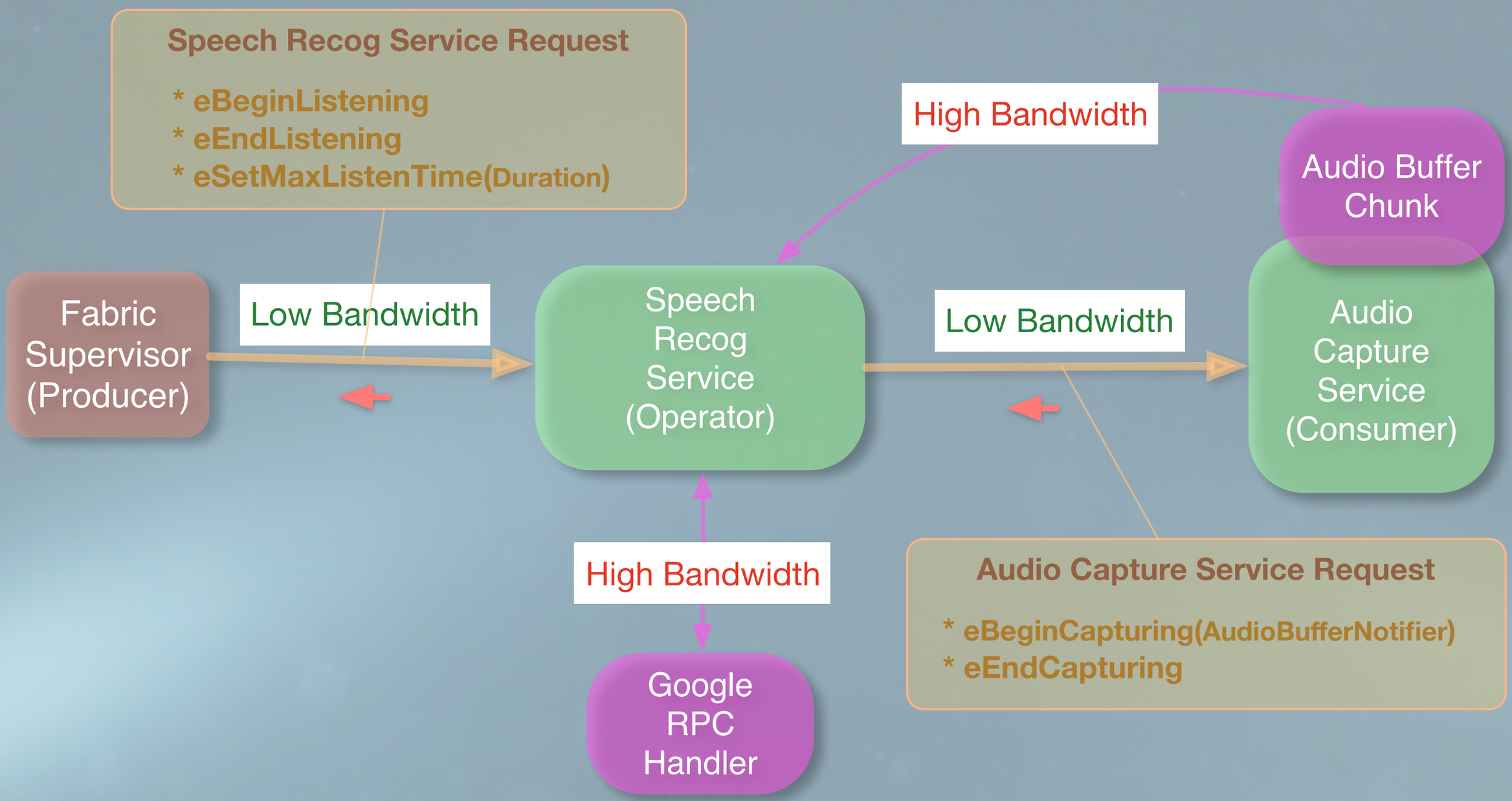
SDK Size: 440K lines of C code and 40K lines objc

Swift version of SDK is from Xcode 8 Beta

Bad design in the sense to use it you have to incorporate many headers

I ended up writing my own interface in objc and then exposed that.

Speech Recog Design (First Pass Concept Design)



Speech Recog Design (First Implementation Design)

Speech Recog Service Request

* eBeginScope(ReplyAction)
  * eBeginListening
  * eEndListening
  * eSetMaxListenTime(Duration)
* eEndScope

Audio Capture Service Request

* eBeginScope(ReplyAction)
  * eBeginCapturing(AudioBufferNotifier)
  * eEndCapturing
* eEndScope

Fabric Supervisor (Producer)

Speech Recog Service (Operator)

Google RPC Handler

Notifier

Audio Capture Service (Consumer)

Audio Buffer Chunk Notifier in separate thread

Speech Recog Service Reply

* eDidBeginScope
  * eDidRecognise(String)
  * eDidNotRecognise
  * eReportError(Error)
* eDidEndScope

Audio Capture Service Reply

* eDidBeginScope
  * eDidBeginCapture
  * eDidEndCapture
  * eReportError(Error)
* eDidEndScope

Speech Recog Design (Second Implementation Design)

Speech Recog Service Request

* eBeginScope(ReplyAction)
  * ePushIntent([String], IntentReplyAction)
  * ePopIntent
  * eBeginListening
  * eEndListening
  * eSimulateVocalCmd(String)
  * eSetMaxListenTime(Duration)
  * eSetLanguage(String)
* eEndScope

Audio Capture Service Request

* eBeginScope(ReplyAction)
  * eSetAudioDataNotifier(AudioBufferNotifier)
  * eBeginCaptureMode(Mode)
  * eEndCaptureMode(Mode)
  * eEndScope

Notifier

Fabric Supervisor (Producer)

Speech Intent Model (Operator)

Speech Recog Service (Operator)

Google RPC Handler

Audio Capture Service (Consumer)

Audio Buffer Chunk Notifier in separate thread

Speech Intent Reply

Speech Recog Reply

Audio Capture Reply

Final Speech Recog Design (Micro LOD)

Speech Intent Request

* eBeginScope(ReplyAction)
  * ePushIntent([AppCmd])
  * ePopIntent
  * eBeginListening
  * eEndListening
  * eCancelListening
  * eSimulateVocalCmd(String)
  * eSimulateSequence(Seq)
  * eSetLanguage(Language)
  * eSetMaxListenTime(Duration)
* eEndScope

Speech Recog Service Request

* eBeginScope(ReplyAction)
  * eBeginListening
  * eEndListening
  * eCancelListening
  * eSetLanguage(Language)
  * eSetMaxListenTime(Duration)
* eEndScope

Audio Service Capture Request

* eBeginScope(ReplyAction)
  * eSetAudioDataNotifier (AudioBufferNotifier)
  * eEnableCaptureMode(Mode)
  * eDisableCaptureMode(Mode)
  * eEnableAudio
  * eDisableAudio
* eEndScope

Google RPC Handler

Audio Buffer Chunk Notifier in separate thread

Direct call

Fabric Supervisor (Producer)

Speech Intent Reply

Speech Intent Model (Operator)

Intent Stack

Speech Recog Reply

Speech Recog Service (Operator)

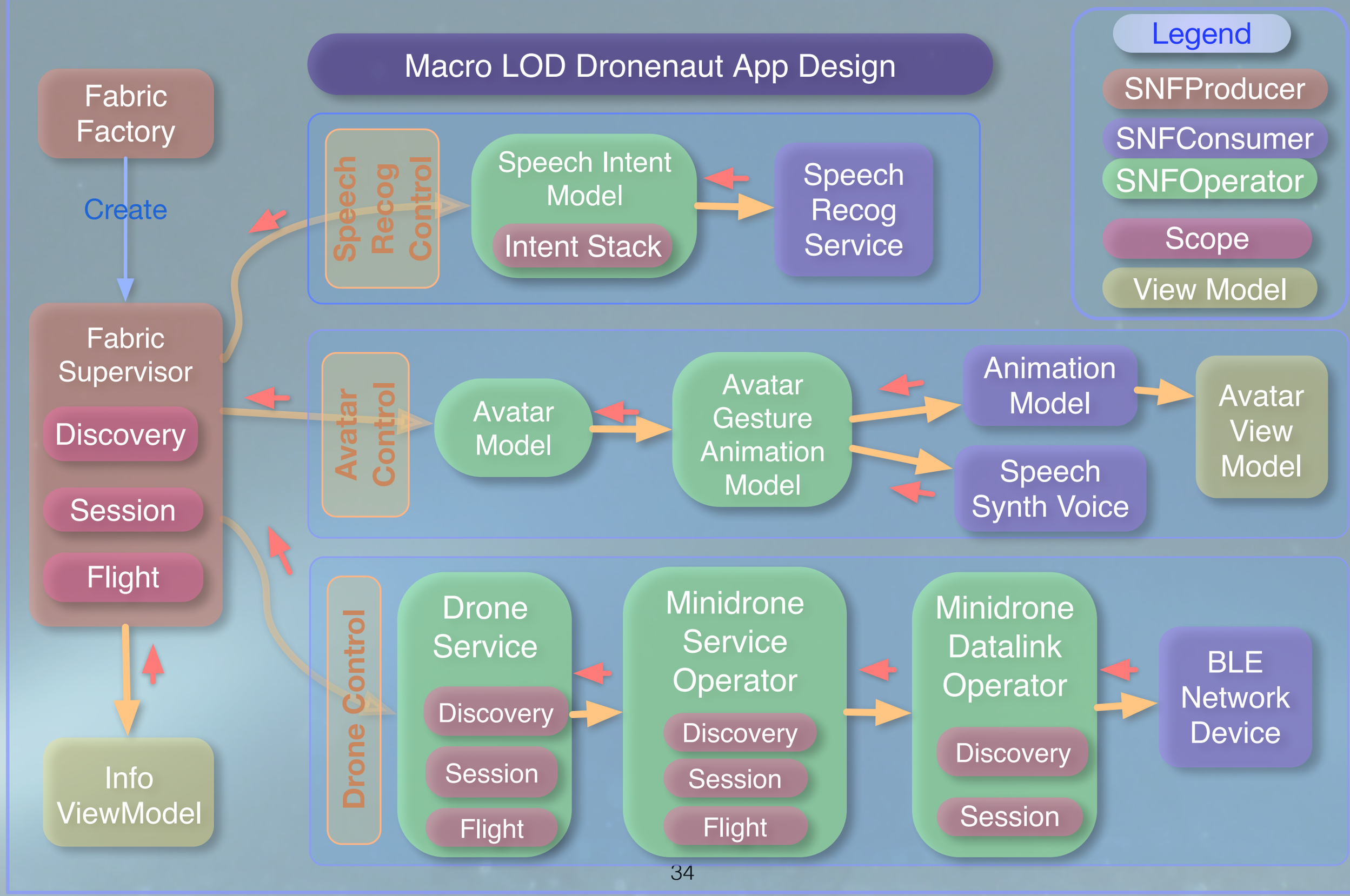
Audio Capture Reply

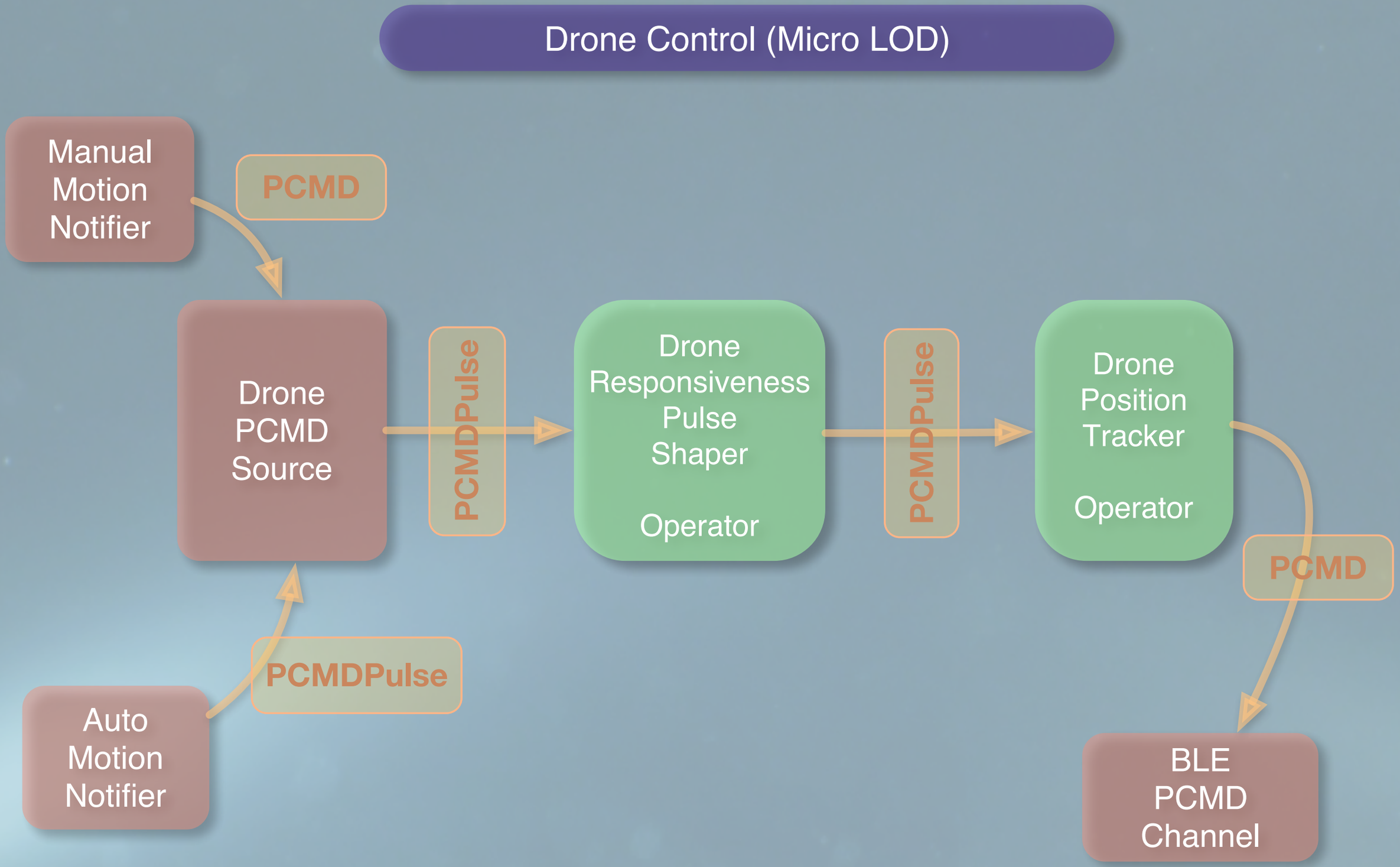
Audio Service (Consumer)

Configure

Capture

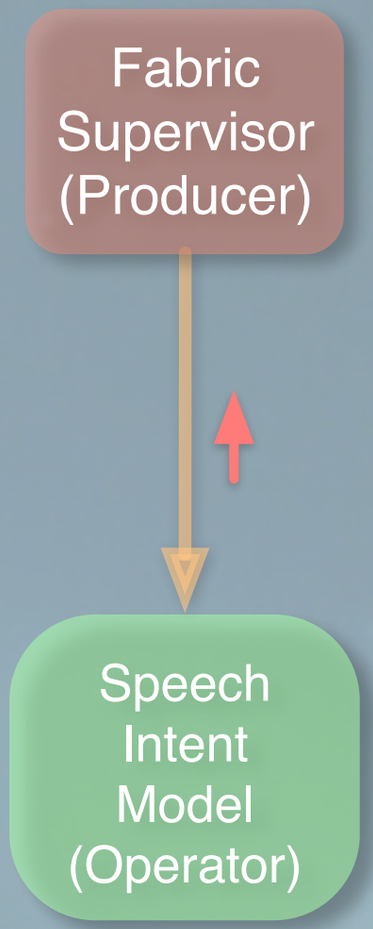
Macro LOD Dronenaut App Design




Drone Control (Micro LOD)
Manual Motion Notifier
PCMD
Drone PCMD Source
PCMDPulse
Drone Responsiveness Pulse Shaper Operator
PCMDPulse
Drone Position Tracker Operator
PCMD
PCMDPulse
Auto Motion Notifier
BLE PCMD Channel

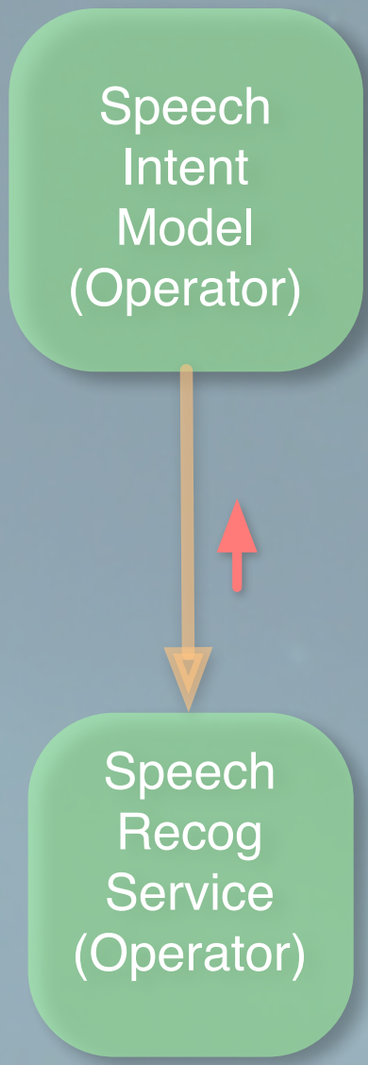
Speech Recog Design (Speech Intent Request/Reply)

```
enum eRequest: ISpeechIntentModelRequest
{
    case eBeginScope((eReply) -> Void)
    case ePushIntent([AppCmd], (eIntentReply) -> Void)
    case ePopIntent
    case eBeginListening(() -> Void)
    case eEndListening
    case eCancelListening
    case eCaseListening(onListening: (() -> Void)?, onNotListening: (() -> Void)?)
    case eSimulateVocalCommand(String)
    case eSimulateTokenSeq(Tokenizer.Seq)
    case eSetLanguage(String)
    case eSetMaxListenTime(TimeInterval)
    case eEndScope
}

enum eReply
{
    case eDidBeginScope
    case eDidStartListening
    case eDidEndListening(TimeInterval)
    case eDidHaveAudioInputLow
    case eReportError(eServiceError)
    case eDidEndScope
}

enum eIntentReply
{
    case eDidSetIntents([String])
    case eDidRecognize([eAppCmd])
    case eDidNotRecognize(String?)
    case eDidHaveAudioInputLow
    case eReportError(eServiceError)
}
```



Speech Recog Design (Speech Recog Service Request/Reply)

```
enum eRequest: ISpeechRecogRequest
{
    case eBeginScope((eReply) -> Void)
    case eBeginListening(() -> Void)
    case eEndListening
    case eCancelListening
    case eCaseListening(onListening: (() -> Void)?, onNotListening: (() -> Void)?)
    case eSetLanguage(String)
    case eSetMaxListenTime(TimeInterval)
    case eEndScope
}

enum eReply
{
    case eDidBeginScope
    case eDidStartListening
    case eDidEndListening(TimeInterval)
    case eDidRecognize(String)
    case eDidNotRecognize(String?)
    case eDidHaveAudioInputLow
    case eReportError(eServiceError)
    case eDidEndScope
}
```



```
/// The handler for the Active Scope requests and replies.
fileprivate final class Channel: SpeechIntentModelRequestHandler<eRequest, eReply>
{
    private var m_replyNotify: ReplyNotify!

    private var m_tokenMap = Tokenizer()
    private var m_intentStack = IntentStack()

    fileprivate init(container: Container)
    {
        super.init(name: "/SpeechIntentModel/Scope/Active/Channel", container: container)

        routeOutput(toNotifyFunc: onRequest)
    }

    /// The handler for ApplicationScope requests.
    /// - Parameter context: The executional context for the request.
    /// - Parameter channelRequest: The request to control the Speech Intent Service.
    func onRequest(_ context: IRxDevContext, _ channelRequest: ISpeechIntentModelRequest)
    {
        func reply(_ withReply: eReply)                                 { m_replyNotify(withReply) }
        func replyFailure(_ error : eServiceError)                      { m_replyNotify(eReply.eReportError(error)) }

        // Cast the request to this scope.
        guard let request = channelRequest as? eRequest
                else
        {
            replyFailure(.eRequestFailure(.ePresentation, "Invalid request for Speech Intent ActiveScope Scope: \(channelRequest)"));
            return
        }

        let output = m_container.m_scopeStack.output

        switch request
        {
            case .eBeginScope(let replyNotify):

                m_replyNotify = replyNotify

                output.issueRequest(context, SpeechRecogService.eRequest.eBeginScope({ [weak self] (outputReply) in

                    guard let strongSelf = self else { return }

                    switch outputReply
                    {
                        case .eDidBeginScope:
                            reply(.eDidBeginScope)

                        case .eDidStartListening:
                            reply(.eDidStartListening)

                        case .eDidEndListening(let duration):
                            reply(.eDidEndListening(duration))
```

```
/// The handler for the Active Scope requests and replies.
fileprivate final class Channel: SpeechIntentModelRequestHandler<eRequest, eReply>
{
    private var m_replyNotify: ReplyNotify!

    private var m_tokenMap = Tokenizer()
    private var m_intentStack = IntentStack()

    fileprivate init(container: Container)
    {
        super.init(name: "/SpeechIntentModel/Scope/Active/Channel", container: container)

        routeOutput(toNotifyFunc: onRequest)
    }

    /// The handler for ApplicationScope requests.
    /// - Parameter context: The executional context for the request.
    /// - Parameter channelRequest: The request to control the Speech Intent Service.
    func onRequest(_ context: IRxDevContext, _ channelRequest: ISpeechIntentModelRequest)
    {
        func reply(_ withReply: eReply)                                 { m_replyNotify(withReply) }
        func replyFailure(_ error : eServiceError)                      { m_replyNotify(eReply.eReportError(error)) }

        // Cast the request to this scope.
        guard let request = channelRequest as? eRequest
                else
        {
            replyFailure(.eRequestFailure(.ePresentation, "Invalid request for Speech Intent ActiveScope Scope: \(channelRequest)"));
            return
        }

        let output = m_container.m_scopeStack.output

        switch request
        {
            case .eBeginScope(let replyNotify):

                m_replyNotify = replyNotify

                output.issueRequest(context, SpeechRecogService.eRequest.eBeginScope({ [weak self] (outputReply) in

                    guard let strongSelf = self else { return }

                    switch outputReply
                    {
                        case .eDidBeginScope:
                            reply(.eDidBeginScope)

                        case .eDidStartListening:
                            reply(.eDidStartListening)

                        case .eDidEndListening(let duration):
                            reply(.eDidEndListening(duration))
```

```
/// The handler for ApplicationScope requests.
/// - Parameter context: The executional context for the request.
/// - Parameter channelRequest: The request to control the Speech Intent Service.
func onRequest(_ context: IRxDevContext, _ channelRequest: ISpeechIntentModelRequest)
{
    func reply(_ withReply: eReply)                                 { m_replyNotify(withReply) }
    func replyFailure(_ error : eServiceError)                      { m_replyNotify(eReply.eReportError(error)) }

    // Cast the request to this scope.
    guard let request = channelRequest as? eRequest
            else
    {
        replyFailure(.eRequestFailure(.ePresentation, "Invalid request for Speech Intent ActiveScope Scope: \(channelRequest)"));
        return
    }

    let output = m_container.m_scopeStack.output

    switch request
    {
        case .eBeginScope(let replyNotify):

            m_replyNotify = replyNotify

            output.issueRequest(context, SpeechRecogService.eRequest.eBeginScope({ [weak self] (outputReply) in

                guard let strongSelf = self else { return }

                switch outputReply
                {
                    case .eDidBeginScope:
                        reply(.eDidBeginScope)

                    case .eDidStartListening:
                        reply(.eDidStartListening)

                    case .eDidEndListening(let duration):
                        reply(.eDidEndListening(duration))

                    case .eDidRecognize(let text):

                        if let tokenSequence = strongSelf.m_tokenMap.tokenize(text: text),
                           let appCmds = strongSelf.m_intentStack.match(tokenSequence: tokenSequence)
                        {
                            strongSelf.m_intentStack.reply(.eDidRecognize(appCmds))
                        }
                        else
                        {
                            strongSelf.m_intentStack.reply(.eDidNotRecognize(text))
                        }

                    case .eDidNotRecognize(let text):
                        strongSelf.m_intentStack.reply(.eDidNotRecognize(text))
```

```
    }

    let output = m_container.m_scopeStack.output

    switch request
    {
        case .eBeginScope(let replyNotify):

            m_replyNotify = replyNotify

            output.issueRequest(context, SpeechRecogService.eRequest.eBeginScope({ [weak self] (outputReply) in

                guard let strongSelf = self else { return }

                switch outputReply
                {
                    case .eDidBeginScope:
                        reply(.eDidBeginScope)

                    case .eDidStartListening:
                        reply(.eDidStartListening)

                    case .eDidEndListening(let duration):
                        reply(.eDidEndListening(duration))

                    case .eDidRecognize(let text):

                        if let tokenSequence = strongSelf.m_tokenMap.tokenize(text: text),
                           let appCmds = strongSelf.m_intentStack.match(tokenSequence: tokenSequence)
                        {
                            strongSelf.m_intentStack.reply(.eDidRecognize(appCmds))
                        }
                        else
                        {
                            strongSelf.m_intentStack.reply(.eDidNotRecognize(text))
                        }

                    case .eDidNotRecognize(let text):
                        strongSelf.m_intentStack.reply(.eDidNotRecognize(text))

                    case .eDidHaveAudioInputLow:
                        reply(.eDidHaveAudioInputLow)

                    case .eReportError(let error):
                        replyFailure(error)

                    case .eDidEndScope:
                        reply(.eDidEndScope)
                }
            }))

        case .ePushIntent(let (appCmds, intentReplyNotify)):

            var appCmdMap : [String : AppCmd] = [:]
```

```
                else
                {
                    strongSelf.m_intentStack.reply(.eDidNotRecognize(text))
                }

            case .eDidNotRecognize(let text):
                strongSelf.m_intentStack.reply(.eDidNotRecognize(text))

            case .eDidHaveAudioInputLow:
                reply(.eDidHaveAudioInputLow)

            case .eReportError(let error):
                replyFailure(error)

            case .eDidEndScope:
                reply(.eDidEndScope)
        }
    }))

case .ePushIntent(let (appCmds, intentReplyNotify)):

    var appCmdMap : [String : AppCmd] = [:]

    for appCmd in appCmds
    {
        assert(appCmdMap[appCmd.anchor] == nil, "Expected unique entries in appCmdMap found duplicate: \(appCmd.anchor)")

        appCmdMap[appCmd.anchor] = appCmd
    }

    let entry = IntentStackEntry(name: name, appCmdMap: appCmdMap, intentReplyNotify: intentReplyNotify)

    m_intentStack.push(entry)

case .ePopIntent:

    m_intentStack.pop()

case .eBeginListening(let completionAction):

    output.issueRequest(context, SpeechRecogService.eRequest.eBeginListening(completionAction))

case .eEndListening:

    output.issueRequest(context, SpeechRecogService.eRequest.eEndListening)

case .eCancelListening:

    output.issueRequest(context, SpeechRecogService.eRequest.eCancelListening)

case .eCaseListening(let (onListen, onNotListening)):

    output.issueRequest(context, SpeechRecogService.eRequest.eCaseListening(onListening: onListen, onNotListening:
onNotListening))
```

```
                output.issueRequest(context, SpeechRecogService.eRequest.eBeginListening(completionAction))

            case .eEndListening:

                output.issueRequest(context, SpeechRecogService.eRequest.eEndListening)

            case .eCancelListening:

                output.issueRequest(context, SpeechRecogService.eRequest.eCancelListening)

            case .eCaseListening(let (onListen, onNotListening)):

                output.issueRequest(context, SpeechRecogService.eRequest.eCaseListening(onListening: onListen, onNotListening:
onNotListening))

            case .eSimulateVocalCommand(let text):

                if let sequence = m_tokenMap.tokenize(text: text), let appCmds = m_intentStack.match(tokenSequence: sequence)
                {
                    m_intentStack.reply(.eDidRecognize(appCmds))
                }
                else
                {
                    m_intentStack.reply(.eDidNotRecognize(text))
                }

            case .eSimulateTokenSeq(let sequence):

                if let appCmds = m_intentStack.match(tokenSequence: sequence)
                {
                    m_intentStack.reply(.eDidRecognize(appCmds))
                }
                else
                {
                    m_intentStack.reply(.eDidNotRecognize("sequence \(sequence)"))
                }

            case .eSetLanguage(let language):

                output.issueRequest(context, SpeechRecogService.eRequest.eSetLanguage(language))

            case .eSetMaxListenTime(let duration):

                output.issueRequest(context, SpeechRecogService.eRequest.eSetMaxListenTime(duration))

            case .eEndScope:

                m_intentStack.popAll()

                reply(.eDidEndScope)
                self.m_container.m_scopeStack.popScope(context)
        }
    }
}
```

A Short Gripe

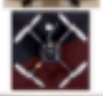
How Fast Do Drones Go ?

Current Guinness Record: 179.6 Mph

These are the fastest FPV Racing Drones of 2017

August 13, 2017 By James — 17 Comments

FASTEST FPV RACING DRONES 2017

	DESIGN	SPEED (MPH)	VERIFIED	THRUST:WEIGHT	THRUST (G)	WEIGHT (G)	MOTORS	BATTERY	TYPE	RTF PRICE	ASSEMBLY	
VXR-190		166	GPS	15	7400	479	2450kv	5S	FPV	$738	MATERIALS +PARTS	BUILD LIST
RACER X		164	TIMER	>9	>7550	800	2500kv	10S	FPV	>$1000	3D-PRINT +PARTS	AVAILABLE PARTS
VX1		152	GPS	15	7400	485	2450kv	5S	FPV	$720	MATERIALS +PARTS	BUILD LIST
SPEEDY GONZALAS		145	GPS	12	6800	570	2750kv	6S	FPV	$544	PARTS	BUILD LIST
STIGG 195		128	RADAR	10	5400	560	2500kv	4S	FPV	$738	PARTS	BUILD LIST
MORPHEUSX 195		125	RADAR	13	7100	557	2400kv	4S	FPV	$568	PARTS	BUILD LIST
GT2 CRUSADER		89	RADAR	10	5200	504	2300kv	4S	FPV	$459	ARF	LINK
WARPQUAD		80	EST.	15	6100	410	2600kv	4S	LOS	$413	PARTS	BUILD LIST
LIZARD 95		76	RADAR	6	648	111	6000kv	3S	FPV	$223	BNF	LINK
ARC 200		75	EST.	9	4000	458	2600kv	4S	FPV	$434	PARTS	BUILD LIST
LISAM 210		75	EST.	9	3800	416	2300kv	4S	FPV	$233	PARTS	BUILD LIST
TEAL DRONE		70	CLAIM	5	3500	730	-	-	FPV	$1300	RTF	LINK
WIZARD X220		68	RADAR	6	3100	535	2300kv	3S	FPV	$276	RTF	LINK

Speech Recognition
Tokenisation Section

```
class TokenizerInitializer
{
    var currentContext                = Context()
    var keywordMatches: [TextMatch]    = []
    var patternMatches: [PatternMatch] = []

    func initialize()
    {
        // Set initial root context.
        self <<- Context()

          // Exact match for <computer>
          <<- TextMatch(["computer"], Seq("<computer>"))

          // Exact match for <engage>
          <<- TextMatch(["engage", "enable", "engaged"], Seq("<engage>"))

          // Exact match for <disengage>
          <<- TextMatch(["disengage", "disable", "disengaged"], Seq("<disengage>"))

        // Set engage/disengage context at root level.
        self <<- Context(["<engage>", "<disengage>"])

            // Exact match for <engage>:<tokeniser> and <disengage>:<tokeniser>
            <<- TextMatch(["tokeniser", "tokenizer", "tokenization", "tokenisation"], Seq("<tokeniser>"))

        // Set computer context at root level.
        self <<- Context([“<computer>"])

          // Regexp match for <computer>:<attack>
          <<- PatternMatch("(?:(?:attack|attacks|attacked|a text|text) (?:sequence|plan|secret|thickness|second|seconds) (.*))",
                  Seq("<attack>",
                  mapFunc: { (matchString: String, template: String, regexp: NSRegularExpression, match: NSTextCheckingResult) -> Any? in

                let matchedText = regexp.replacementString(for: match, in: matchString, offset: 0, template: template)

                return !matchedText.isEmpty ? matchedText : nil
          }))
    }
}

@discardableResult fileprivate func <<-(lhs: TokenizerInitializer, rhs: TokenizerInitializer.Context) -> TokenizerInitializer
{
    lhs.currentContext = rhs

    return lhs
}
```

App Explain and Demonstration Modes

Bluetooth LE Section

Bluetooth LE (Smart) 4.0 - 4.2

Spread Spectrum Transmission: 40 channels separated at 2 Mhz intervals over 2.4 - 2.4835 GHz.

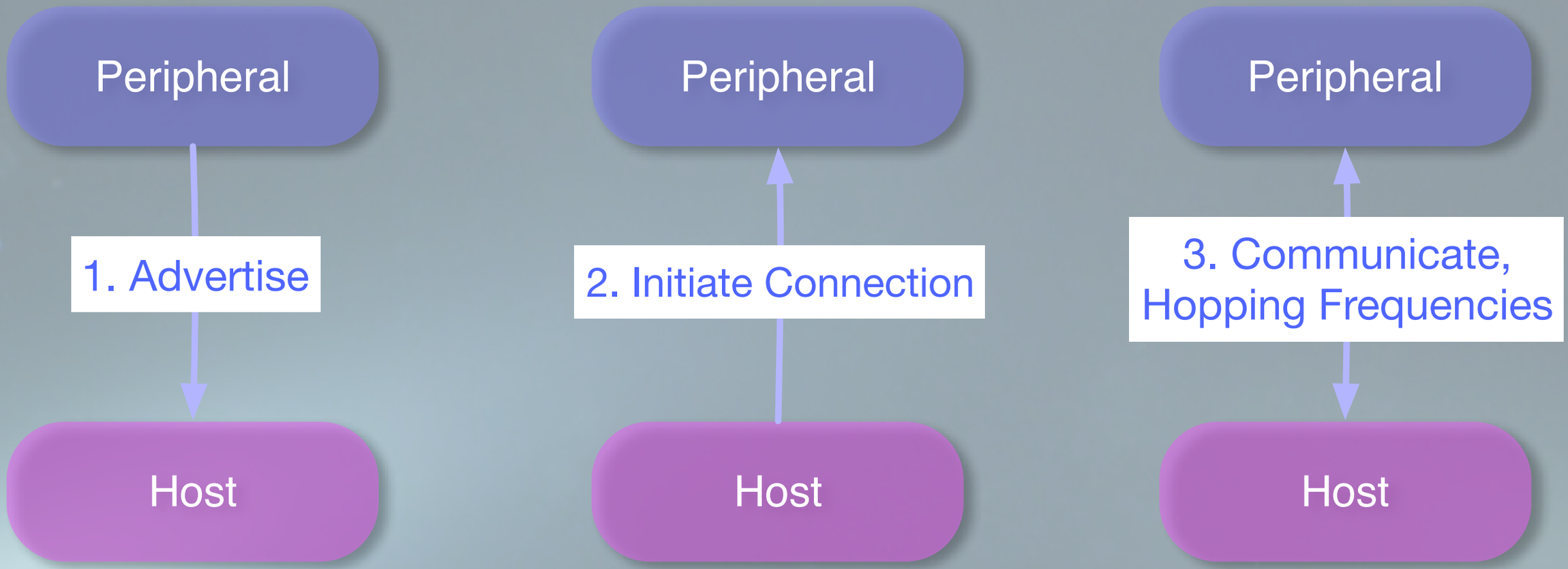
Distance Range given by spec: Class 1: 100m, Class 2: 10m, Class 3: 1m

Approx Realistic Data Rate: ~ 260Kbs (BLE 4.0), ~ 650 Kbs (BLE 4.2)

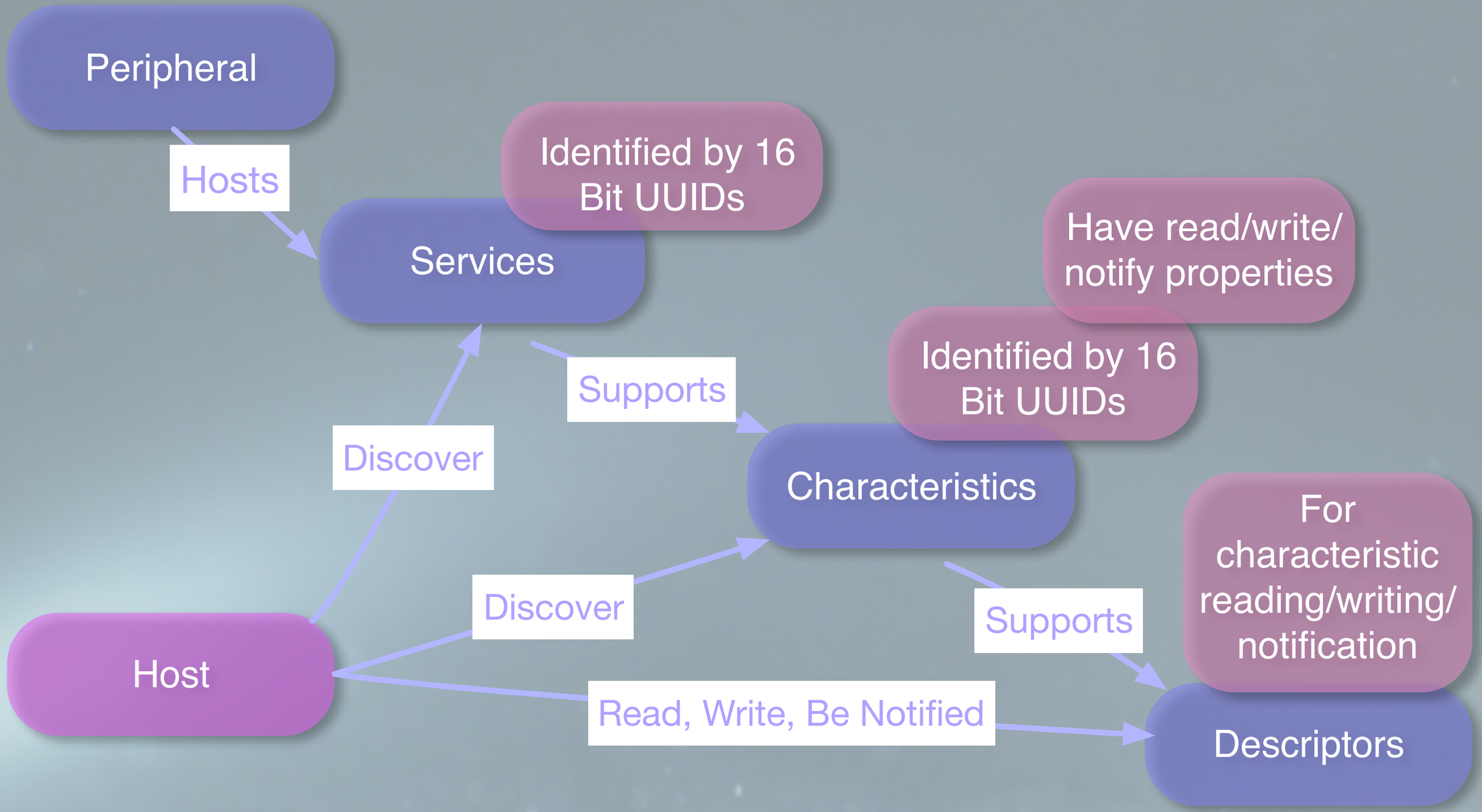
Peak Current consumption: < 15mA

Note: The most recent standard is 5.0, released June 2016, devices should start appearing this year.

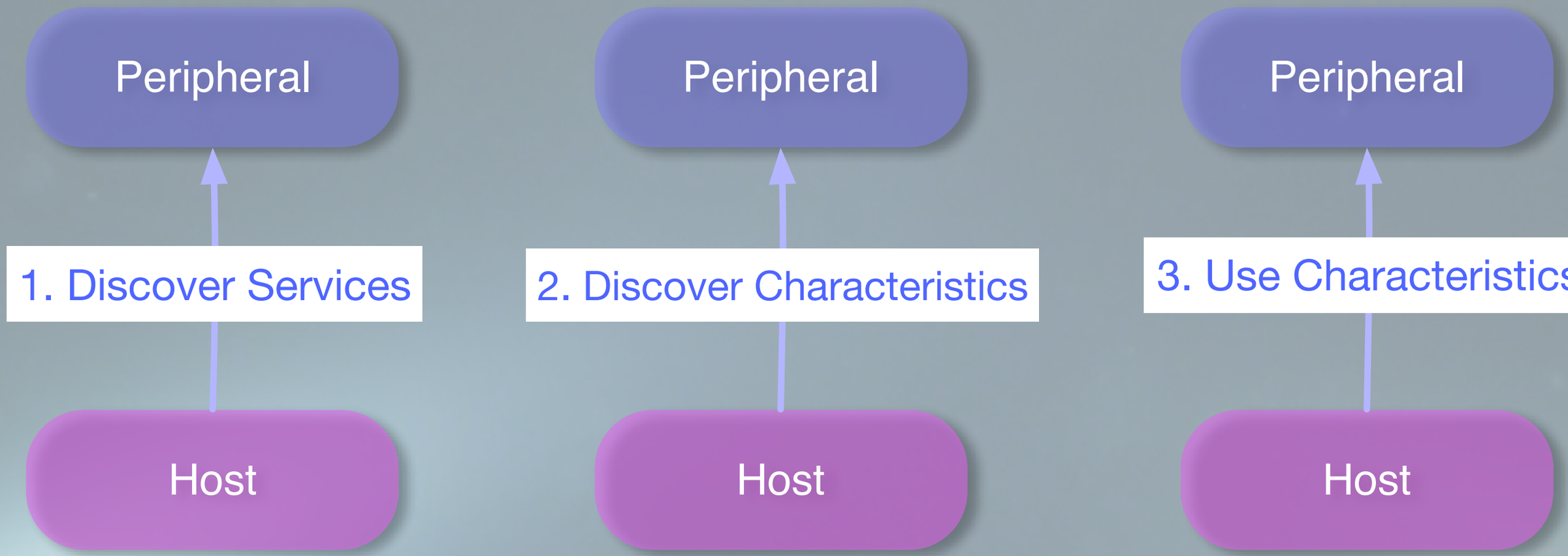
Bluetooth LE - Connection Protocol



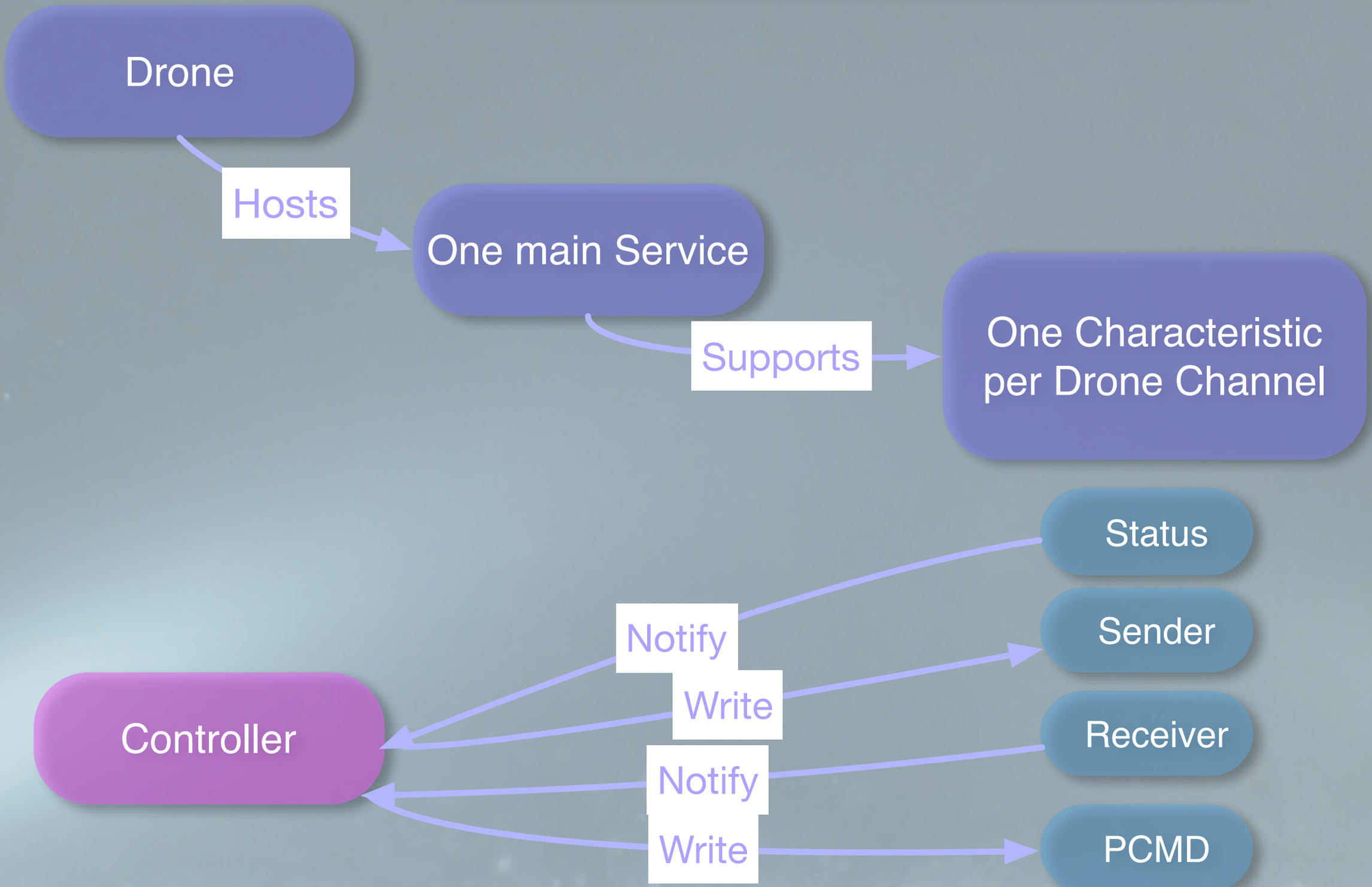
Bluetooth LE - Services



Bluetooth LE - Service Discovery Protocol



The Parrot Minidrone BLE Services



Virginia Woman Says She Shot Down Drone Near Actor Robert Duvall's Home

"I had my .20-gauge there, so I put two 71/2 birdshot shells in it," Jennifer Youngman said

A Virginia woman says she shot down a drone after she spotted the device flying over her famous next door neighbor Robert Duvall's house and it veered onto her land.
Jennifer Youngman said she was cleaning two guns on her front porch in Fauquier County when she saw two men park in front of the actor's home.

The men set up a table and began operating the drone over "The Godfather" star's property.The device buzzed about 75 feet in the air and disturbed his cows, Youngman said.

But when the men appeared to have lost control of the drone, Youngman took action.

“They were going a little too fast and they went over my airspace," she told "I had my .20-gauge there, so I put two 71/2 birdshot shells in it, and there you are.”

Source: Virginia Woman Says She Shot Down Drone Near Actor Robert Duvall's Home | NBC4 Washington http://www.nbcwashington.com/news/local/Virginia-Woman-Shoots-Down-Drone-Near-Actor-Robert-Duvalls-Home-391423411.html#ixzz4Z7SMIjh6

www.originware.com

Originware

Evaluation Kits | Doc | Contact | About

- Reactive Extensions Technology.
- iPhone/iPad App Development.
- Consulting: Software Design and Architecture.
- Agile Project Management.
- Contract Software Development.

Looking for technology partners.

Looking for app developers that want to incorporate the technology.

Please tweet about the Dronenaut app, if you like it.

Technology Software Development.

Originware Blog

Follow @originwaredev