Φ PhiInnovations

Android built for Radio over IP (RoIP) system implementation

Henrique Kuehne – HW Engineer
contact@phiinnovations.com

Project overview

- Develop a proof-of-concept (PoC) of using an Android system to implement an embedded system solution in telecommunication domain
- Implement a Radio over IP solution using standard components and off-the-shelf devices
- Control software and communication protocol implemented using Android Operating System
  - The idea is to implement an Android solution which in general would be implemented in an embedded Linux environment

Project requirements

- Development of a stand-alone embedded system
- (Optional) implementation of a Graphical User Interface
- Embedded system should activate a radio using PTT
- Embedded system should receive and send audio to the radio
- Communication between the embedded systems must be using TCP/IP
  - Ethernet
  - Wifi
  - Blutetooth

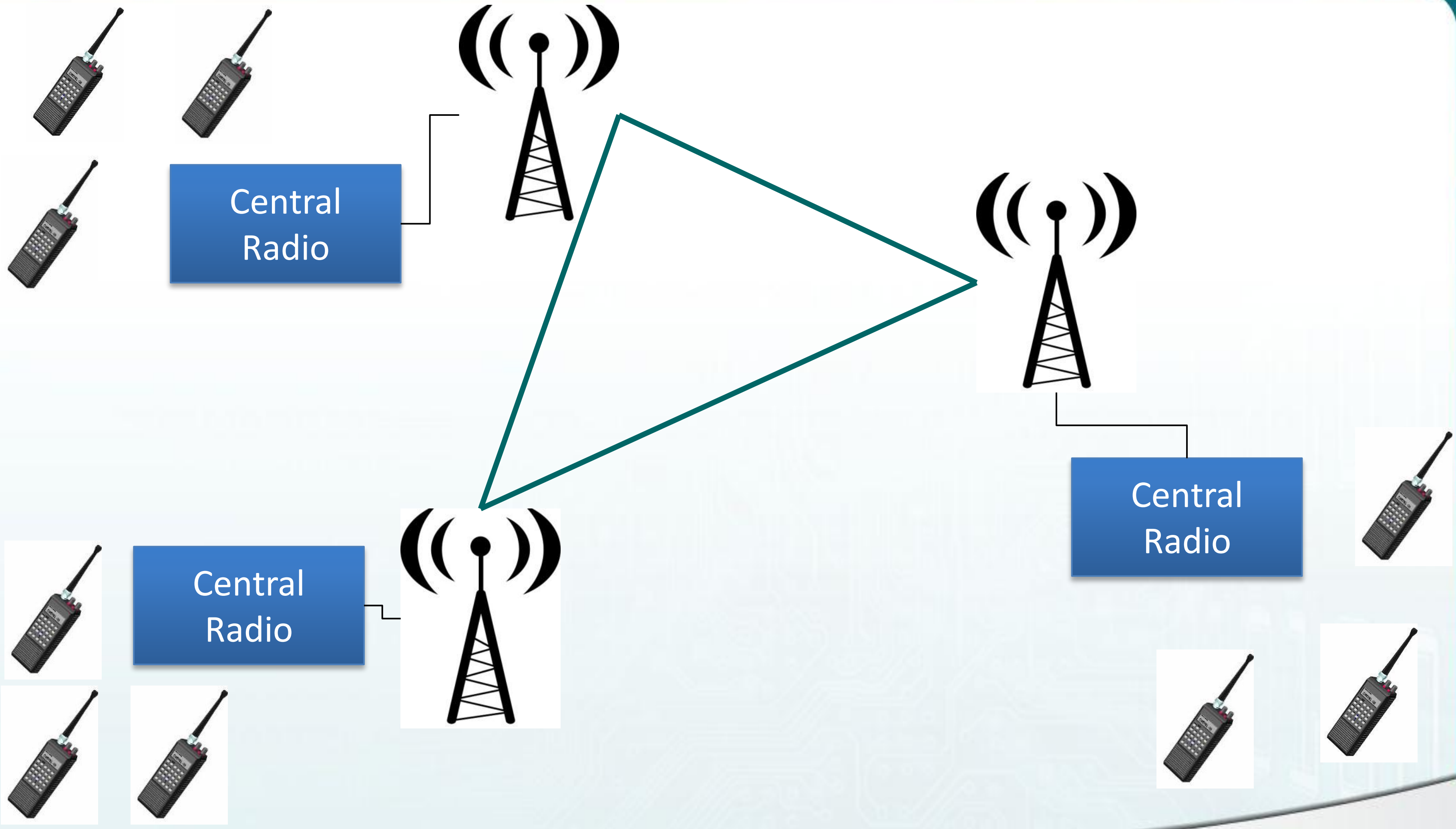
SYSTEM OVERVIEW

About RoIP

Radio over Internet Protocol is a way to transmit and receive radio communications over Internet Protocol.

It uses the same concept of VoIP with a command layer to control the direction of the radio through PTT (push-to-talk).

Why RoIP?

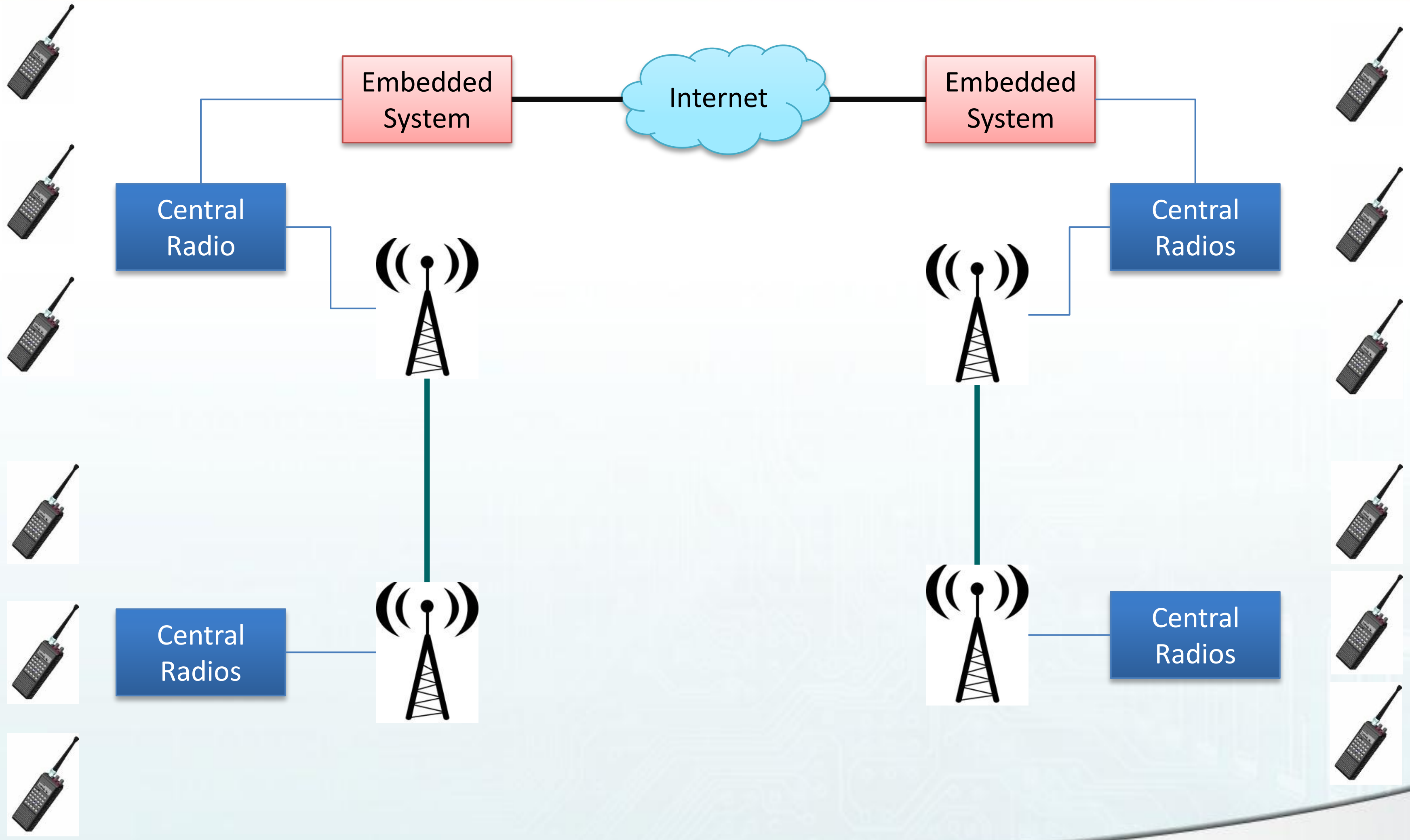


Central Radio

Central Radio

Central Radio

Why RoIP?



Embedded System

Internet

Embedded System

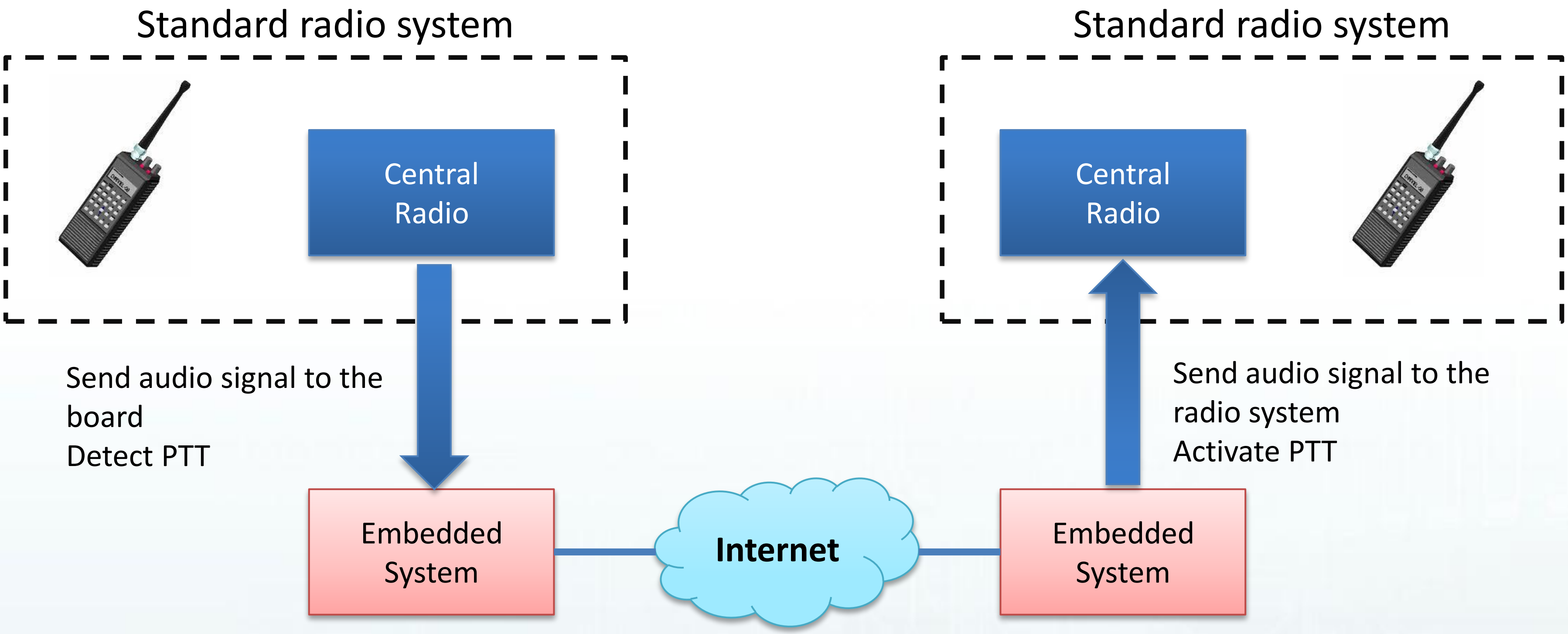
Central Radio

Central Radios

Central Radios

Central Radios

System Architecture



Standard radio system

Central Radio

Send audio signal to the board
Detect PTT

Embedded System

Internet

Embedded System

Send audio signal to the radio system
Activate PTT

Central Radio

Standard radio system

Apply audio codification
Exchange data with similar equipment

Same embedded system for both functionalities
Running specific Android Operating System

Apply audio decodification
Exchange data with similar equipment

HARDWARE DESCRIPTION

Embedded system

Freescale iMX53 Quick Start Board

A complete embedded computer:

- iMX53 ARM Cortex A8 processor
- Audio, Video, storage capabilities
- Serial, USB, Ethernet communication

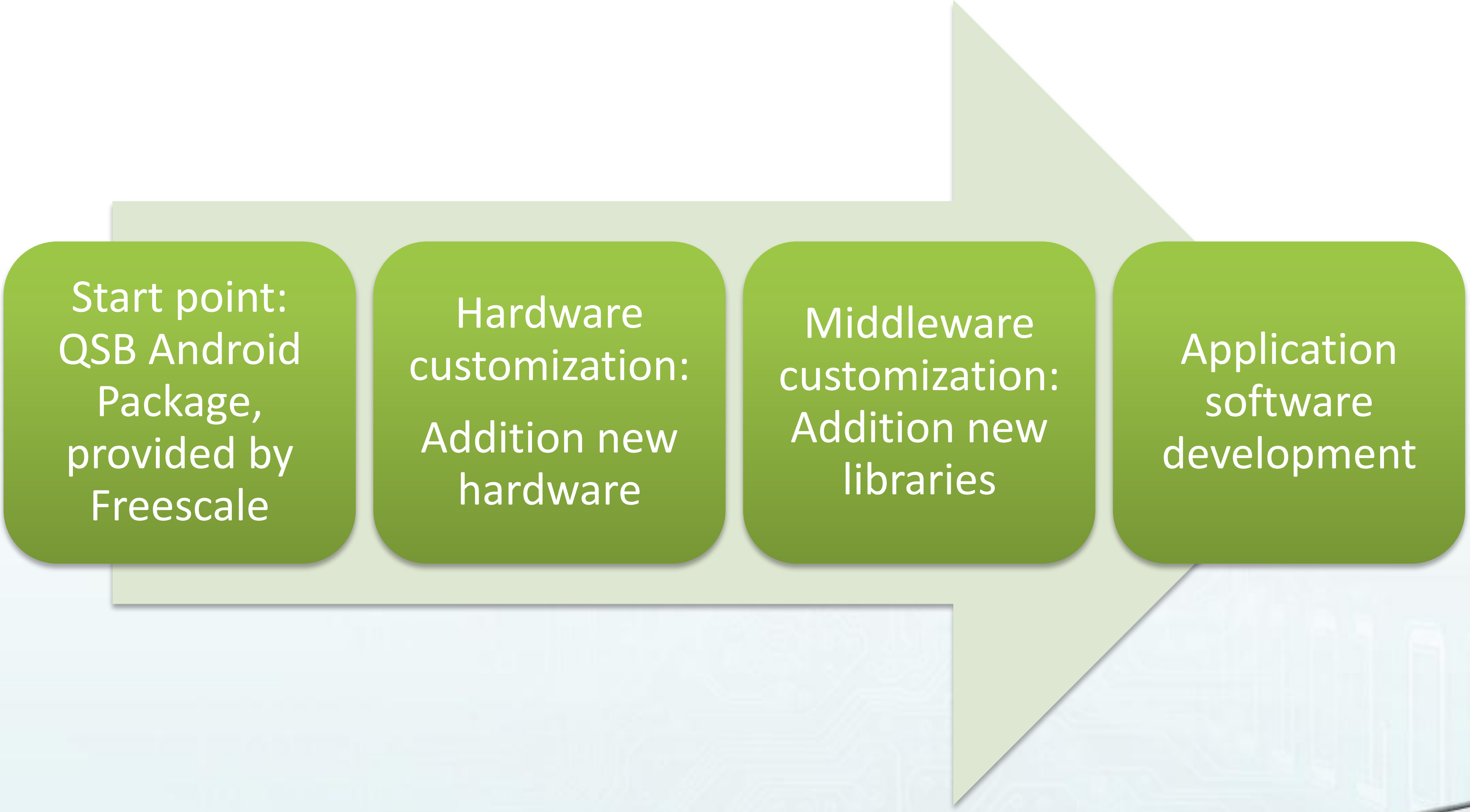
New features to this board was added, as equipment add-on:

- GPRS Module
  - Communicating with the serial interface
- WiFi Module
  - Communicating with the USB interface
- Bluetooth Module
  - Communicating with the USB interface
- GPIO activation
  - For PTT

For this project, we used COTS assembly hardware

ANDROID SYSTEM DEVELOPMENT

Android deployment (1/5)



Start point: QSB Android Package, provided by Freescale

Hardware customization:

Addition new hardware

Middleware customization: Addition new libraries

Application software development

Android deployment (2/5)

- Download BSP provided by Freescale
  - From Adeneo Embedded Website

- Android compilation from sources
  - Patches applied from AOSP
  - Android version 3.2 (Froyo)
  - Using build scripts provided by the BSP

Android deployment (3/5)

- Kernel customization
  - Custom changes for the provided kernel from Freescale
  - Rebuild made using build scripts provided by the BSP
- WiFi dongle
  - Added kernel patches provided by the manufacturer
- Bluetooth dongle
  - Added kernel patches provided by the manufacturer
- GPRS modules
  - No need to patch the kernel. It uses serial interface for CPU-Module communication
- GPIO configuration – Pin muxing
  - Needed to add support for PTT activation and detection
  - Interfacing to GPIO subsystem from kernel

Android deployment (4/5)

- Middleware configuration
- Installation of additional libraries and packages into Android system
  - New open source packages
  - Package sources added directly to Android build system
  - In some cases, changes was needed to be made. Specially on build scripts (Makefiles)
- PPPd – current installation didn’t work as expected. Added a newer version of the project
- Speex – codec for audio communication. This open source codec was added to Android build structure

Android deployment (5/5)

- Application development

- It was needed to add SDK support for these new features added to the Android BSP
- New functions was added
  - Using JNI (Java Native Interface) technology

- GPRS – module control functions
  - AT commands conversion
- WiFi / Bluetooth – no changes needed
  - Android OS automatically detected the hardware
- GPIO – new functions required
- Audio codec – new functions required
  - Mapping from C functions in low level

APPLICATION SOFTWARE

Application software architecture (1/3)

Configuration engine

- Responsible for network setup
- Responsible for point-to-point connection establishment
- Responsible for operation logging retrieval
- Responsible for operation status

Communication engine

- Wait for data from PTT
- Codification and decodification of audio
- Send audio information to radio

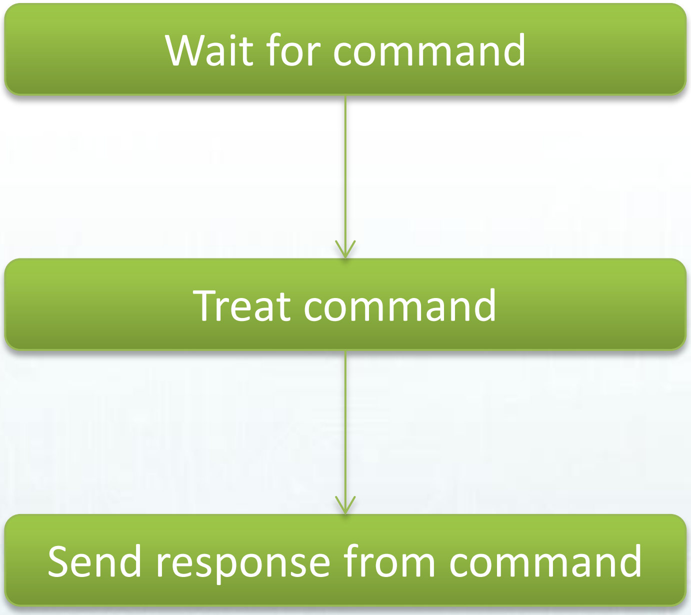
Application software architecture (2/3)

Implemented a command-line interface (CLI) to setup and get status from the equipment.

Embedded systems – no need for graphical user interface (GUI)

Implemented commands:

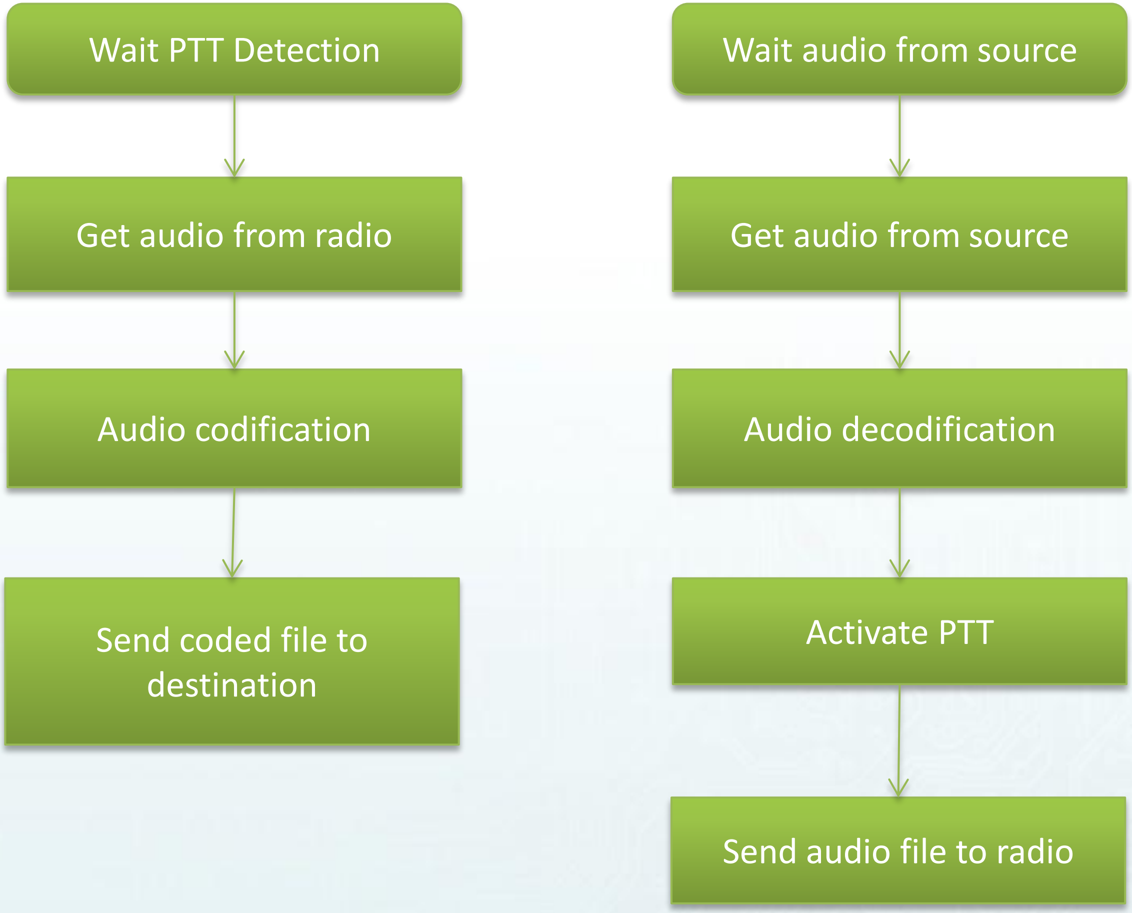
- Setup local IP
- Setup remote IP
- Get log file
- Read communication engine status


Wait for command

Treat command

Send response from command

Application software architecture (3/3)



The communication engine is composed by two main threads:

- Audio send data
- Audio receive data

CONSIDERATIONS

Considerations

- The main objective of the project is to use Android OS in an embedded system in an application generally used by a Linux system
- The software behavior was excellent, even considering the fact that an Android system is bigger than Linux system
- Android customization was challenging, but as much as difficult as deploying an embedded Linux system
- Java language was not an issue, regarding performance
- Much possibilities to improve the solution
  - Addition of Graphical User Interface
  - Addition of custom protocols

Thank you (contact@phiinnovations.com)