Internet of Things and Software Defined Radio

Danny Webster




Introduction - What is Internet of Things?

- **What is Internet of Things?**
- **Innovative Products**
  - Using Radio Connectivity
  - Includes Artificial Intelligence?
- **Applications**
  - Home
  - Cars
  - Industrial Sites

- **Possible Applications include…**
  - A fridge that automatically orders luxury chocolate from Amazon for you because you are depressed.
  - An alarm clock that tells you where next door's cat has gone during the night.
  - Custom made devices to help people with memory loss or unique learning difficulties or other disabilities.

Lime microsystems

Introduction - What is Software Defined Radio?

- **What is Software Defined Radio?**
  - Is **low cost 'field programmable'** radio that **connects to a computer.**
  - End user selects different software for different radio standards.
- **What is the minimum computer hardware do we need for SDR?**
  - Raspberry Pi Zero,
  - I7 Tower up in the Cloud
  - Depends on the link.
- **What computer languages do we need to use?**
  - Do we need more than one computer language?
- **Which Radio Standard to use?**
  - Depends on link.
  - Video – WiFi or LTE
- **Is our application 'Safety Critical'?**
  - What if our psychoanalysing fridge malfunctions and starts ordering **gift wrapped rat poison** from Amazon instead of luxury chocolate?
  - How do we make it fail safely?

Lime microsystems

The Low Cost SDR

RF Parts

Antennas
SAW Filters
RF Switches
Power Amps

TRX RFIC

RF and DSP
Field Programmable

FPGA

Data link and
Extra DSP

PCIe Link

USB3 Link

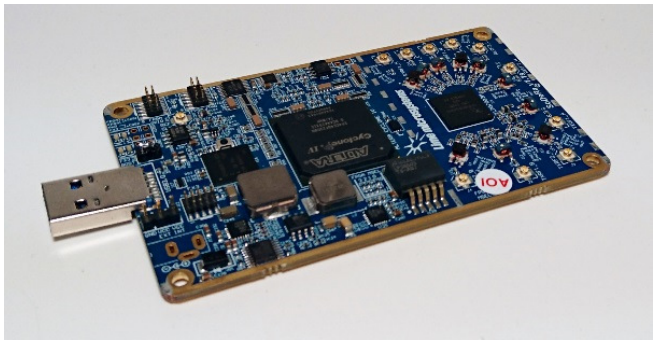
Open Source Software/Apps

COMPUTER

Multicore
GHz Processor
And Memory

WiFi/Ethernet/ ADSL Network

Optimal partitioning of the radio system leads to an easy to use low cost solution.



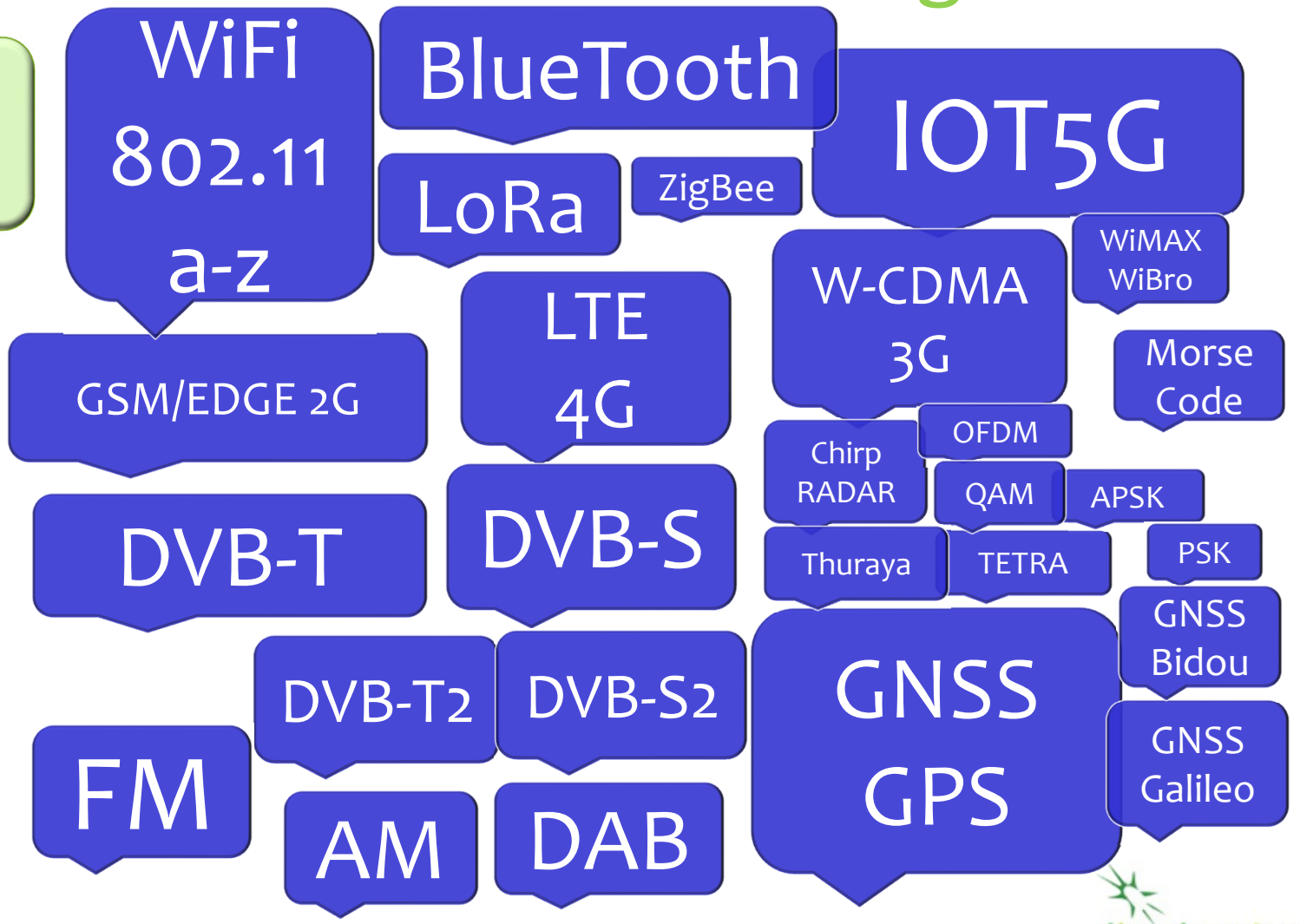
Lime microsystems

So why do we need SDR for Things?

<1940s Morse AM FSK
1950s FM appeared
1978 Military GPS (CDMA)

Since 1991, an explosion of Radio Standards, many are GMSK, CDMA and OFDM based

We need SDR!!!



Which Programming Language for SDR and Things?

- **Tradeoffs**
  - Development Time, Execution Speed & Interconnectivity with other languages
- **C/C++, Fortran95**
  - Software for **high speed real time** links.
- **GNU Radio, Pothos, Scratch**
  - **Integrated environments** for 'real time' software defined radio systems.
- **Octave/Matlab etc.**
  - **Easy to use highly interactive** languages for education and for R&D
  - **Efficient development** of algorithms
- **Python/Ruby**
  - Interactive languages
  - Compact, **good for GUI and Networks**
- **Perl/PHP/Java/JavaScript etc.**
  - Scripting and remote **database** links.
- **LISP/Prolog/Mathematica etc.**
  - **Artificial intelligence & expert systems**
  - Rules and 'real time' knowledge
- **ADA**
  - **Safety Critical Systems.**
  - Extended Error checking

Lime microsystems

Languages from SDR/Things Perspective…

GNU	TCP + UDP Sockets	Linux Sys Calls	ARGV	Text Files	Binary Files	Pipe Output	Complex Numbers	Links to C	TK	Irregular Lists	Compiled?
ADA	Yes?	via C?	YES	YES	YES	YES	?	?	?	?	YES
C/C++	YES	YES	YES	YES	YES	YES	YES	YES	YES	USER DEF	YES
Fortran95	Yes?	Send only	YES	YES	YES	YES	Vector/Matrix	YES	Ex C?	NO	YES
Com LISP	Yes?	YES	NonStd	YES	YES	YES	YES	??	Ex Lib	YES	Optional
Java	YES	YES	YES	YES	YES	YES	USER DEF	?	?	USER DEF?	YES
Julia	YES	YES	YES	YES	YES	YES	Vector/Matrix/FFT	Shared Lib	?	?	Interpreted?
Octave	YES	YES	YES	YES	YES	YES	Vector/Matrix/FFT	YES	?	NO	Optional?
Perl	YES	YES	YES	YES	YES	YES	YES	Shared Lib	YES	NO	Interpreted?
Prolog	Yes?	?	?	?	?	interactive	USER DEF?	?	?	?	Interactive
Python	YES	YES	YES	YES	YES	YES	Vector/Matrix/FFT	YES	YES	YES	Optional
Ruby	YES	YES	YES	YES	YES	YES	YES	Shared Lib	YES	NO	Inteprted?

Most languages do sockets, system calls, command lines, files and pipes.
How *PAINFUL* is it to link between languages? Language should define basic constants.
Artificial intelligence often needs irregular and flexible data structures for real life.
Language independent GUI can help speed development.
User defined structures often have additional burdens that slow down programming.

Languages from SDR/Things perspective…

		(-Ofast)	(-Ofast)	(Numpy)	(Repository)	(Repository)
Time in ms	**Target Time**	**C 99**	**FORTRAN 95**	**PYTHON 2.7**	**OCTAVE 4.0.3**	**JULIA 0.4.5**
BPSK Hadamard 256 build (W-CDMA)	initial	0.517	1	6.1	15	763
Binary PRS 4096 (WiFi 802.11a)	0.1	0.038	2	153	4465	80.4
Complex Number Spread 300*256 (W-CDMA CPICH)	0.1	2.615	3	2.8	13	4204
Complex Number Despread 300*256 (W-CDMA CPICH)	0.1	0.579	1	2.1	18	0.605
RRC FIR Complex Number 76800pts x4 OSR +/-3symbols	1	0.059	125	1005		9231
Complex Number FFT 4096 points	0.2	0.806	1	2.176	24	1496
Software tested on a 1.44GHz Z83 Atom with Ubuntu 16.04 Low Latency		Fortran was originally developed by IBM in the 1950s				
Vectorisation used where possible for speed		C was originally developed by Bell Labs in the 1970s.				
Functions inlined for speed		Octave is a GNU clone of the Matlab language developed in 1970s				
GNU compilers gfortran5, g++5		Python was originally developed in the Netherlands in the 1990s				
Timing not usually 7repeatable, best of 3 runs		Julia began development in 2009				
Fortran CPU_TIME only measures to 1ms accuracy						
Libfftw3.3 installed on ubuntu						

PRS/CRC are a requirement of almost all digital radio standards.
FIR filtering is vital for most radio standards, but some SDRs provide on chip FIRs

Lime microsystems

Languages – Simple FSK Example

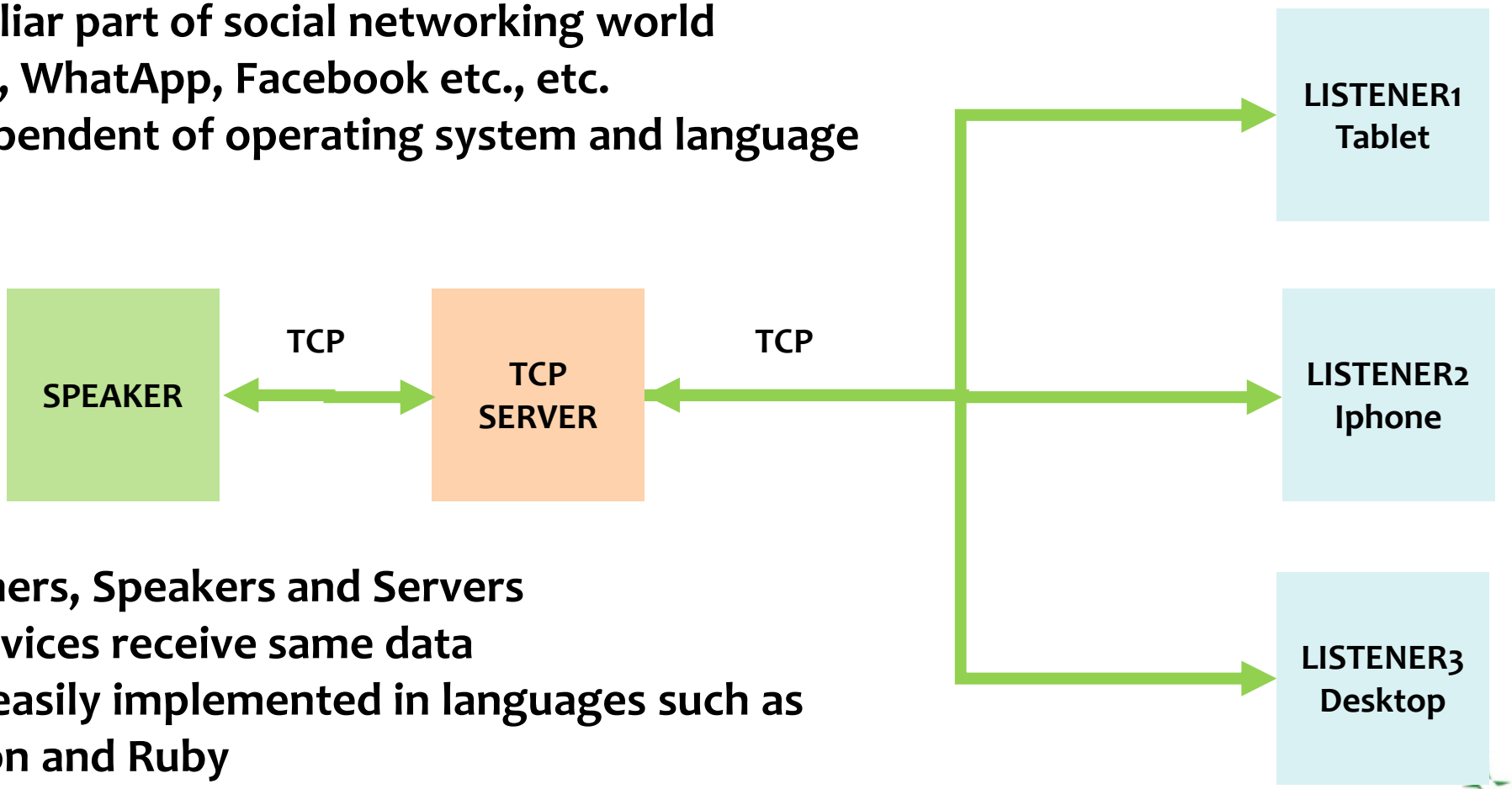
- **OCTAVE**
  - 26 Lines, including graph plot
  - About 20mins to code and debug
- **FORTRAN90**
  - 86 Lines
  - about 3 hours to code and debug
- **C99 (Complex numbers)**
  - 110 Lines,
  - about 3 hours to code and debug
- **(Shorter line counts possible at expense of readability)**

- **High level numerical languages**
  - Vectorised maths is compact to code.
  - 'roots' in Fortran,
    - So map more easily to FORTRAN than C
  - Interactive, speeds up prototyping
- **FORTRAN**
  - Fast, Compiler optimised for large data objects.
  - Compiler error messages often obtuse.
- **C compilers**
  - Very fast, heavily optimised for speed.
  - Work very well with small data objects.
  - Compiler much more 'static analysis' aware.
  - Unsafe programming structures easily used

Lime microsystems

Classic Networking – Chat Network

Familiar part of social networking world
MSN, WhatApp, Facebook etc., etc.
Independent of operating system and language



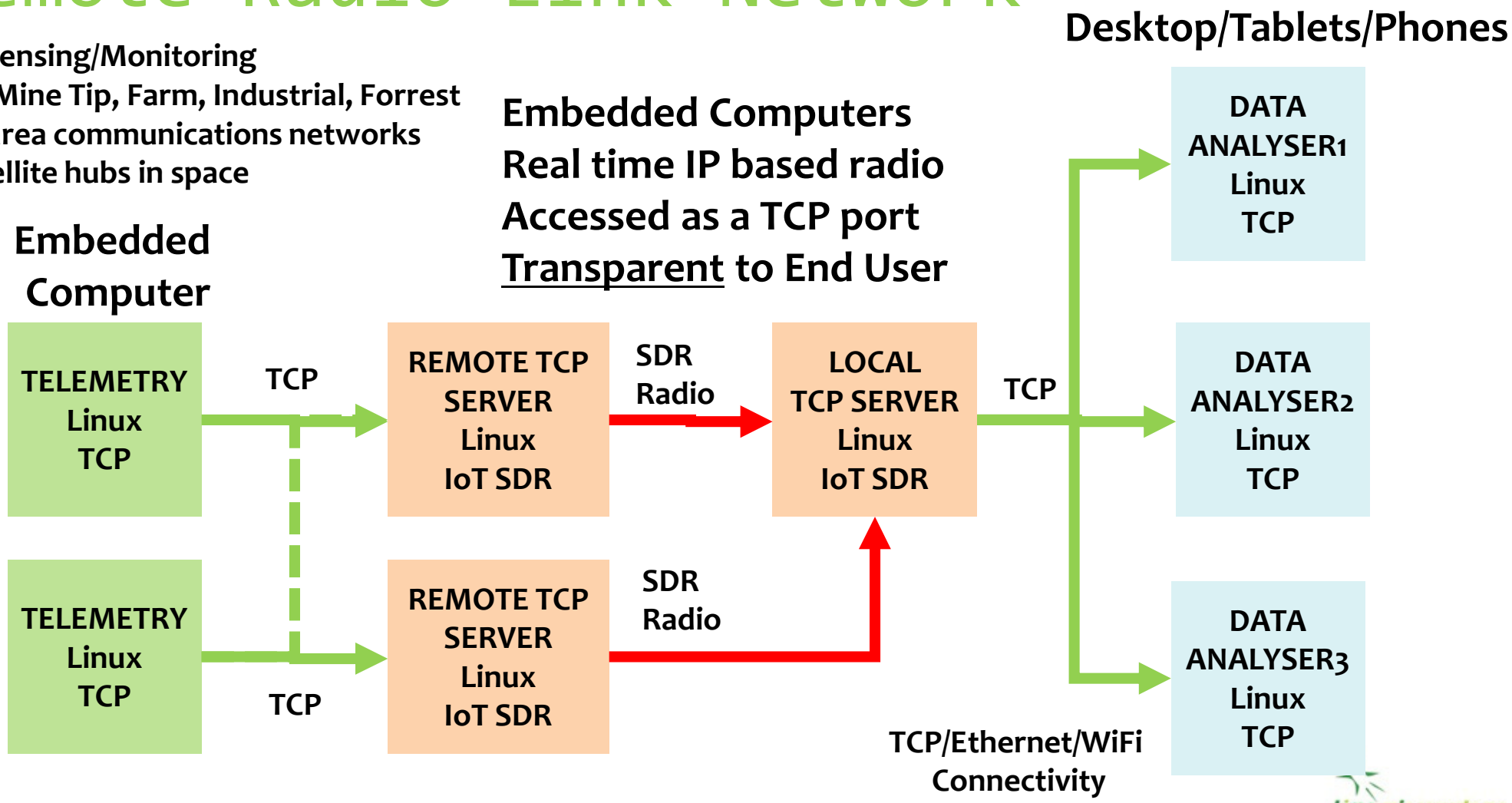
Listeners, Speakers and Servers
All devices receive same data
Very easily implemented in languages such as Python and Ruby

Lime microsystems

Remote Radio Link Network

Remote Sensing/Monitoring
Volcano, Mine Tip, Farm, Industrial, Forrest
Disaster area communications networks
Microsatellite hubs in space

Embedded Computers
Real time IP based radio
Accessed as a TCP port
Transparent to End User



Philosophy and SDR

- **Unix/Linux Philosophy**
  - "Do one thing, and do it well", McIlroy (Bell Sys Tech Jour 1978)
- **SDR Philosophy**
  - **Rapid low cost innovation** through:
  - **Low cost RF hardware**
    - '**Field Programmable**', Widely available.
  - **Low cost computers**
    - Even **'entry level'** processors are now 1GHz **Quad Core** (Raspberry Pi, Intel Atom Z83)
    - Ideal for real time systems.
  - **Open Source Software**
    - **Repo**sitory and **App store** based distribution

Lime microsystems

Conclusions

- **Internet of Things Fuelled by…**
  - Very low cost ‘Field Programmable’ Software Defined Radios,
  - Very low cost multicore computers
  - Open source software, easily obtainable.
- **Most ‘Things’ systems will be IP based**
  - Next generation SDR drivers needs to be become much more ‘**network friendly**’
  - SDR needs to become ‘**invisible**’ to the end user.
  - Modern high level languages make network GUI software very easy to write.
- **Many programming languages for Things and SDR**
  - Vector Maths important for low level radio development.
  - Processor and compilers need to be improved for calculating PRS/CRC and FIR

Lime microsystems

Conclusions

- **Many programming languages for Things and SDR (ctd)**
  - Modern interactive programming languages can significantly speed up software development and help debug lower level historic languages.
  - Historic programming languages perform remarkably well in modern real time systems if they are well supported.
    - FORTRAN90 family is particularly interesting as it more closely maps with R&D programming languages such as Octave/Matlab.
    - Some further work is required optimising and improving Compilers
  - Most existing languages are not well matched to Artificial Intelligence
  - A number of programming languages are held back by poor connectivity to other computer languages and standard interfaces such as TCP Sockets.
    - Open source artificial intelligence languages are particularly poor in this area.
  - Improvements in Compiler 'Static Analysis' helps 'Safety Critical' development but do not eliminate all 'unsafe' programming.

Lime microsystems