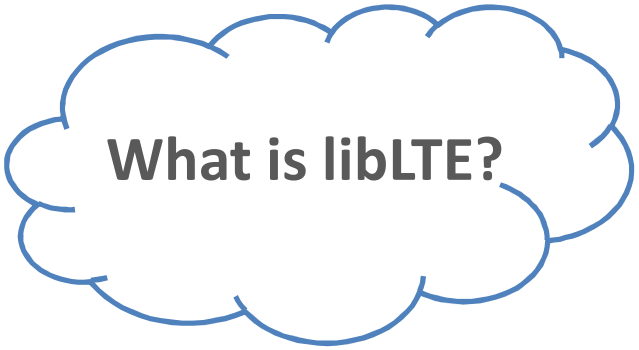
libLTE

A modular open source library for LTE

Ismael Gomez, Paul Sutton,

Vuk Marojevic, Antoni Gelonch

libLTE Project: Started Jan-14



Library

Modular

Computationally-Efficient

No external dependencies or frameworks

Automatic code-generation tools for:

GNURadio, ALOE, Iris, SCA, OSSIE, Matlab-MEX, etc.

Related Projects

Project	License	Framw.	Notes	Hardware
Amarisoft	Propietary	N/A	Full eNodeb + MME. 2-antenna, 20 Mhz in Quadcore. Requires Boost.	N210
openLTE	AGPLv3	GR	PHY + MAC + partial RRC. Not modular	osmoSDR
gr-lte	GPLv3	GR	Synch, PBCH and PCFICH. Requires Boost.	N/A
LTE Cell Scanner	AGPLv3	N/A	Synch and PBCH. Requires Boost & IT++.	rtl-sdr
OSLD	LGPLv3	ALOE++	Single antenna partial Downlink. Distributed real-time and deterministic latency.	UHD

so... why we need another LTE sw?

Library:

To enable reuse of modules

Portability:

C++ or Boost unavailable in some embedded platforms

Framework:

Sometimes too much overhead

Compatibility between frameworks

Efficiency:

It’s not just about “implementing the standard”

Objectives

Modular & Flexible

Automatic Framework Code Generation

Efficient

Design

Low-level API
Total freedom for designer

High-level API
With some rules

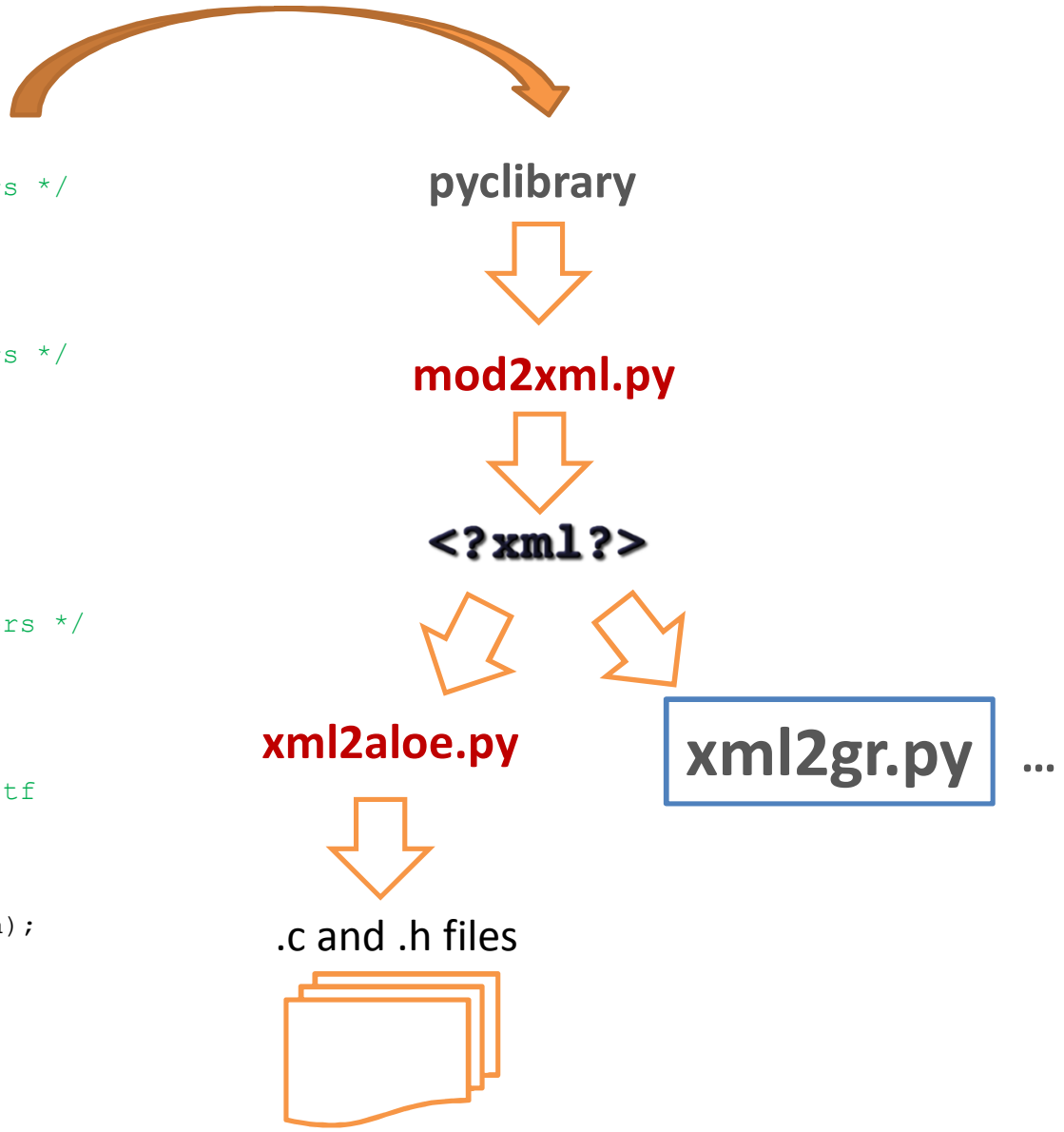
Use acceleration libs if available
e.g. libfftw or VOLK

Example – Low-level API

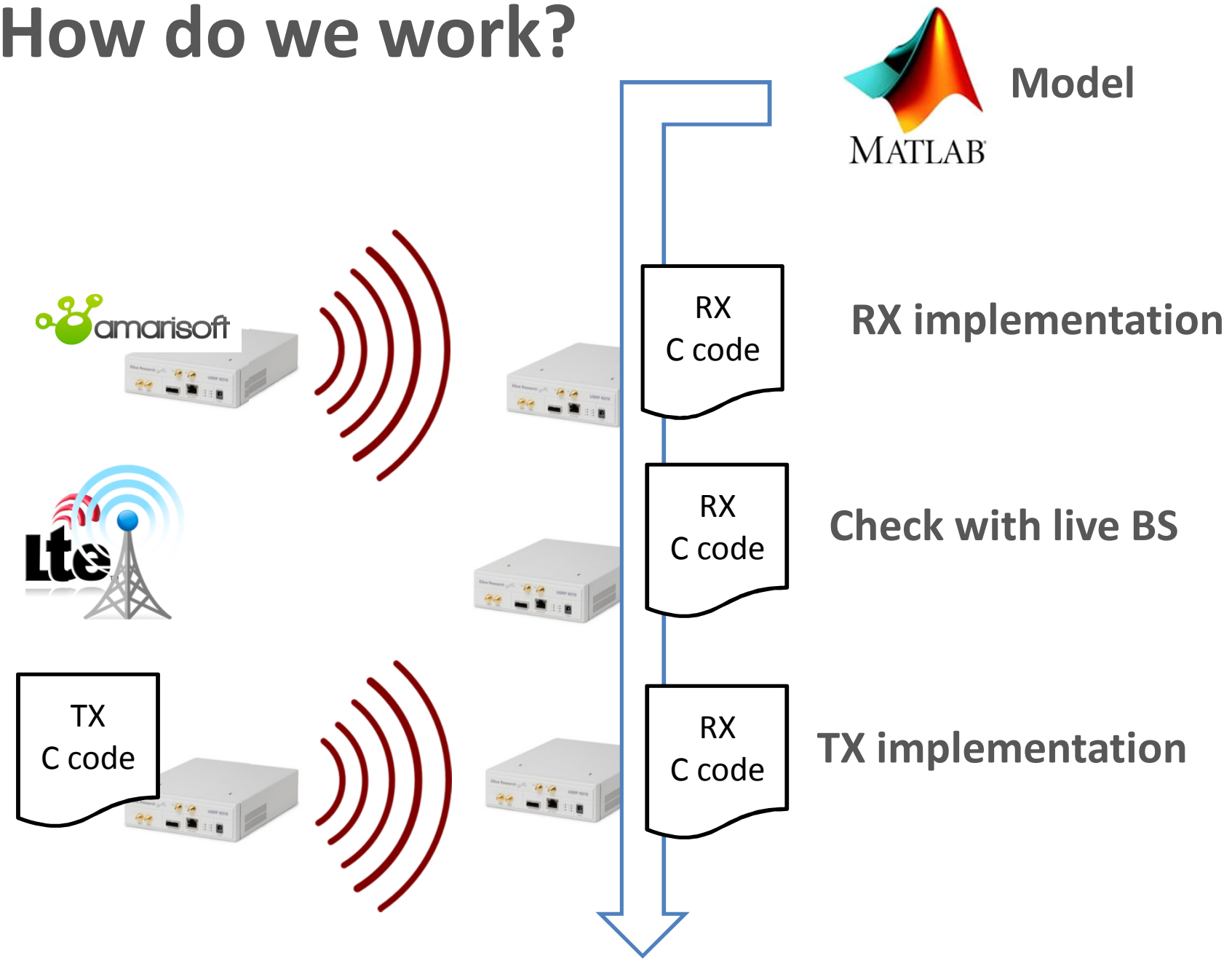
```
void mymodule_init(mymodule_t *h, int init_parameter);
void mymodule_free(mymodule_t *h);

int  mymodule_getopt(mymodule_t *h);
void mymodule_setopt(mymodule_t *h, int value);
```

Example – High-level API

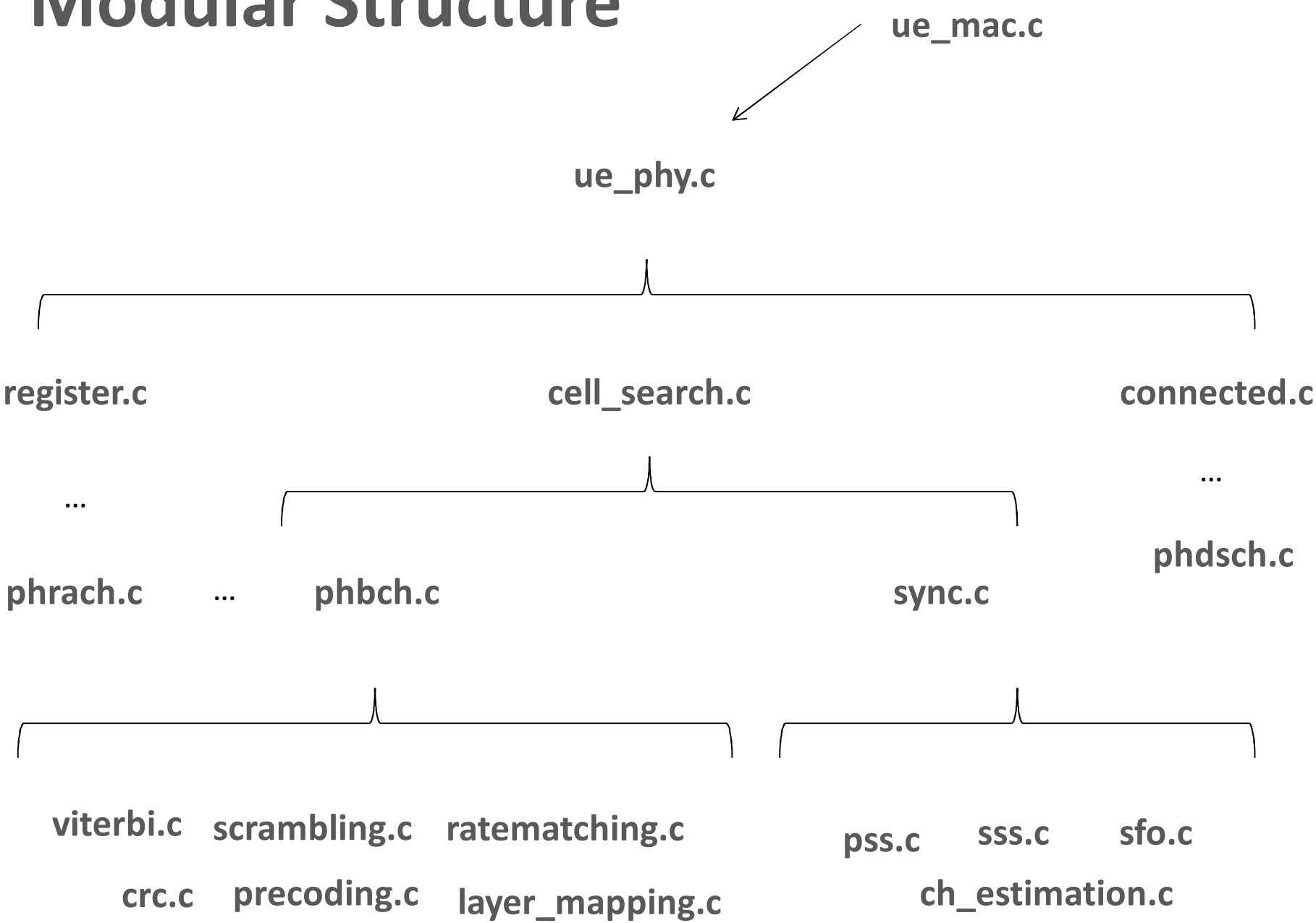
```
typedef struct {
            mymodule_hl obj;

            /* Initialization parameters */
            struct mymodule_init {
                int init_parameter;
            } init;

            /* Run-time input parameters */
            struct mymodule_input {
                int input_parameter;
            } ctrl_in;

            cf_t *input;
            int out_len;

            /* Run-time output parameters */
            struct mymodule_output {
                 int output_parameter;
            } ctrl_out;

            cf_t *output[2]; // 2 out itf
            int  *out_len[2];
 } mymodule_hl;

int mymodule_initialize(mymodule_hl *h);
int mymodule_work(mymodule_hl *h);
int mymodule_stop(mymodule_hl *h);
```



How do we work?



Modular Structure



Directory Structure

```
/
 lib/
  fec/
  filter/
  io/
  modem/
  phch/
  ratematching/
  scrambling/
  sync/
   pss.c
   sss.c
   sfo.c
   sync.c
  ...
 include/
  ...
  lte.h
examples/
matlab/
scripts/
```

Modules organized in categories

LTE PHCH processing. Uses other modules

Several modules in each category

A single .h includes all modules

Framework code-generation tools

What we've done so far:

Real-time cell search. Find LTE BS & sync:

- LTE synchronization:
  - Primary Synchronization Signal (PSS). Transmitted every 5 ms. Need to correlate with 3 possible sequences
  - Secondary Synchronization Signal (SSS). Every 5 ms. Gives Cell ID information
  - Carrier Frequency Offset (CFO). Can be estimated from the Cyclic Prefix or PSS
  - Sampling Frequency Offset (SFO). Estimated from PSS index drift

- Cell Search:
  - Scans LTE band get RSSI
  - For each RSSI > threshold,
    - Search for PSS correlation peak
    - If found, track *N* frames and get CFO, SFO and cell id

LTE Challenges

- ✓ Sampling Frequency
  - ✓ Not all HW has 3.84 Mhz clock.
  - ✓ Resampling or change DFT size?
  - ✓ libfftw good performance in arbitrary lengths
- ✓ Synchronization
  - ✓ DFT-based correlation and narrowed tracking
  - ✓ 1 MHz sampling is enough (PSS/SSS 945 Khz)
  - ✓ 64-point FFT
- ✓ Decoding is very expensive!
  - ✓ Turbo decoder. SIMD maybe required.
- ✓ Multiuser
  - ✓ Synchronization & Detection also expensive.

Roadmap

Milestone	Component	Status	Missing parts	MM
0.0.1	Cell Search	95 %	Final opt. tunning	0.25
0.0.2	eNodeB PSS + Broadcast	80 %	Fix RM and test	0.5
0.0.3	PDCCH (eNodeB + UE)	70 %	Blind decoder and test	0.75
0.0.4	PDSCH (eNodeB + UE)	70 %	Fix RM and test. Different DCI types. HARQ.	1
0.1	**Downlink Integration & Testing**	**0 %**	**-**	**1**
0.1.1	PRACH	0 %	-	0.75
0.1.2	PUSCH	50 %	Most of the components are same to DL	0.5
0.1.3	CQI + Control	0 %	-	0.75
0.1.4	Multiuser synch + detection	0 %	-	1.5
0.2	**Uplink Integration & Testing**	**0 %**	**-**	**1**
0.2.1	MIMO DL & UL	0 %	-	1.5
0.2.2	SIMD Turbo Decoder	0 %	-	1.5
0.3	MAC Layer eNodeB + UE	0 %	-	1.5
1.0	**Full eNodeB + UE Integration & Testing**	**0 %**	**-**	**1.5**

Conclusions

- libLTE is a new open-source project.
  - Simple library of LTE modules

- We want to:
  - Enable easy reuse of modules
  - Avoid having to use frameworks or libraries
  - But exploit their potential if available.

- Call for contributions:
  - Matlab models or C implementations are welcome!

Thanks!

github.com/ismagom/liblte