CMPE245
Sept.7

Sept.7.

Note: 1° Homework (RF module & RF Work-in-Progress)
Due today Inspection in Class.

Homework (1pt) Due A week from Today. Write C/C++, OR Python to Implement LISA Algorithm (Phase I) Such that:

1° Console Input from user to Select No. of Bytes for Synchronization.

For Example: 16 Bytes or 32 Bytes.

Print the payload message.

Note: Python Implementation on Jetson NANO, OR R-pie 3 B+ or 4, you can do the same.

Note: This homework is for Laptop Based Implementation.

Based on the Homework (Today. RF. Board) we will continue with "Landline" Testing Capability.

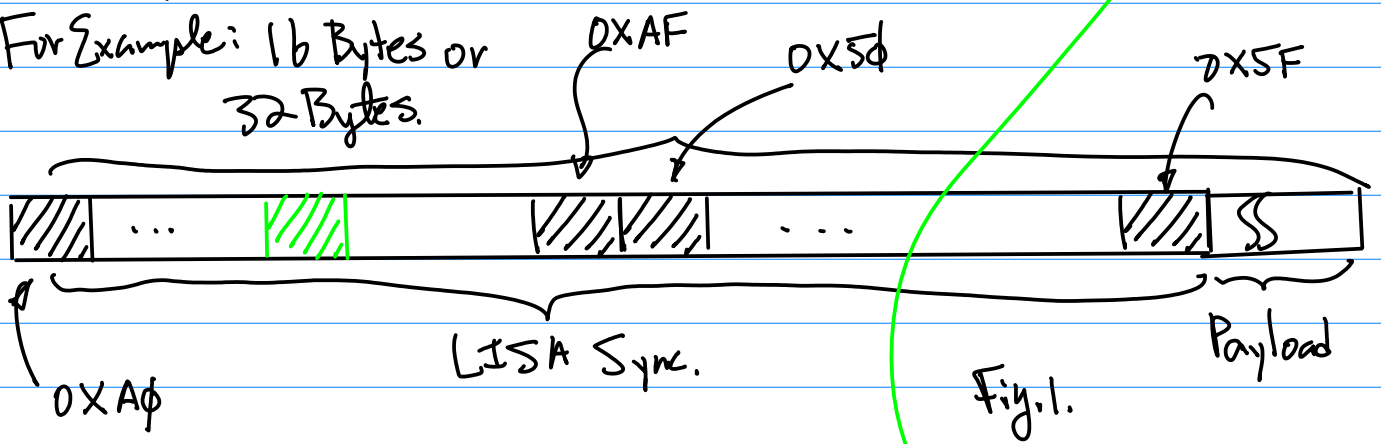
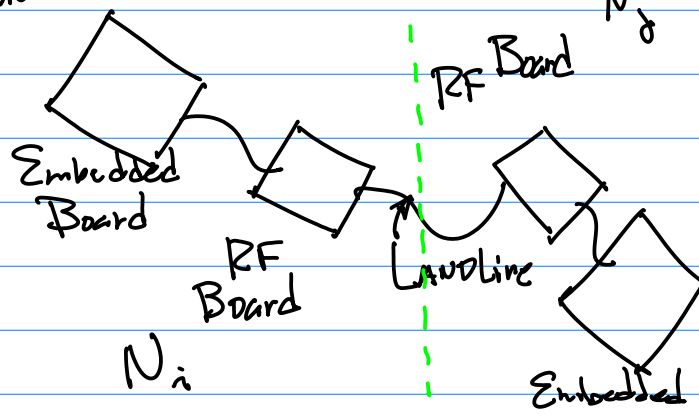


Fig.1.

2° Note in the future (phase II) we would like to Extend this Implementation to Allow a Single Byte (as "Green" in Fig.1. matching to the LISA Sync Field to Establish Synchronization.

3° Payload: First Name + Last Name + 4 Digits ID + CMPE245 + SJSU



Cmpt 245

Sept. 7

Example: Ref from the Class github, ID: 2018F-104~

Observation 1: The Minimum Number of Bytes to establish Synchronization is 1. Therefore 1/32 Bytes for the Sync. → Confidence Level/Index

$$\eta = \frac{\text{No. of Bytes to Establish Sync.}}{\text{Total Number of Bytes (32)}} \quad \cdots (1).$$

Note: In Software Defined Radio, We can Change η (Confidence Level) to trade the quality for Speed if it is allowed.

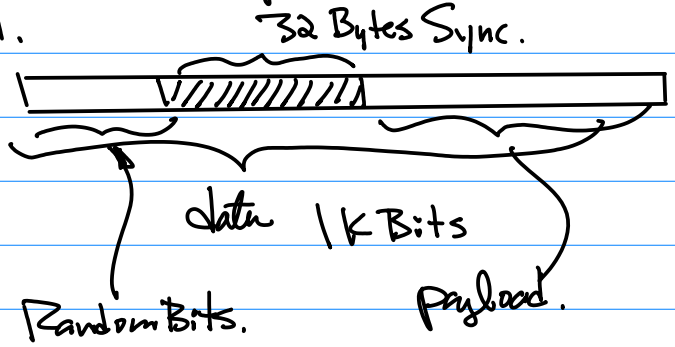
In Cognitive Radio Design, we would like to have this Ability.

Observation 2: "Linear" Characteristics is from the fact LISA Index is defined from 0 to F with Linear increment. And "Invariant" Characteristic is due to the fact the ID Index, e.g. Ranging from 0 to F will allow the Algorithm to pin point to the Beginning of the payload.

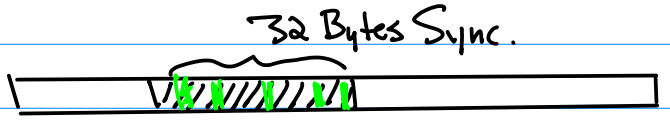
2018F-105~

Example: Homework ON LISA from the Class github.

1.



2. Sync Field is Corrupted.



256 Bits (=32 Byte in Sync Field)

↓

10% Random Bits. Corruption.

↓

256 × 10% = 26 Bits Random to alter Sync Field.

Generate Random Bits. (26 bits).

Use "XOR" Bitwise at Any Arbitrary Location within the Sync Field.

3. User Input for the No. of Bytes (as Confidence Level), then the Code will parse the input file with the Confidence Level to Establish Synchronization.

Sept. 12 (Monday)

Today's Topics: 1° LISA Homework Implementation. 2° BaseBand Signal

Example: LISA Implementation.

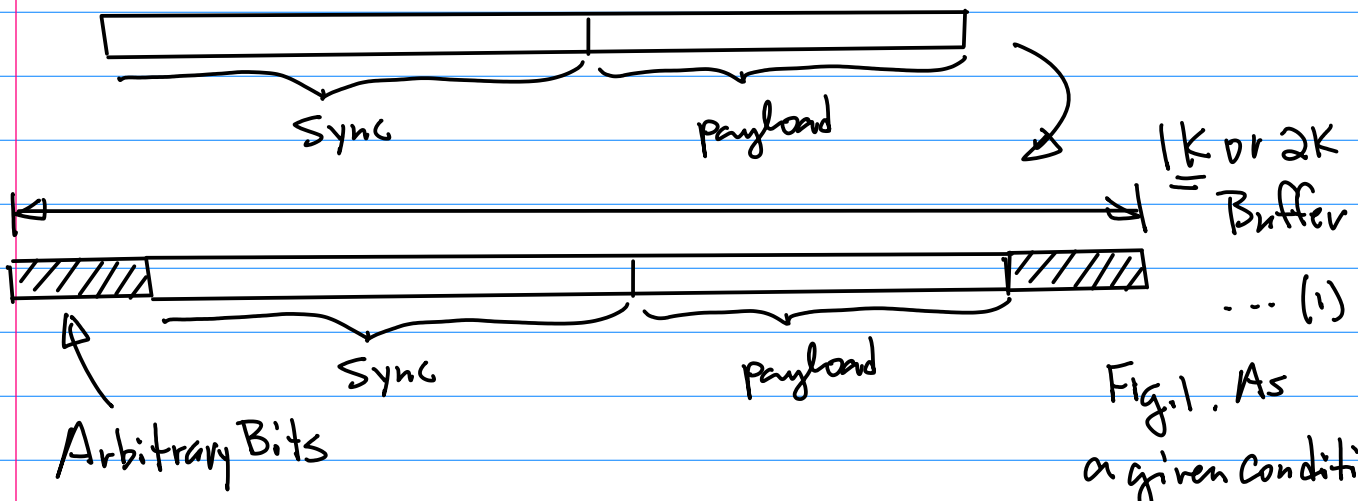
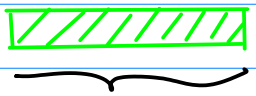


Fig.1. As a given condition.

Step2. Create a "mask" Template to Reflect the matching size that you like



Step3. Add Random Noise to Fig1. then move (2) at the beginning bit (1st Bit) of (1). Such as mask

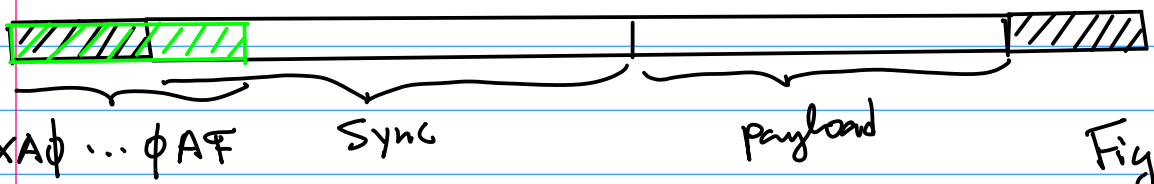
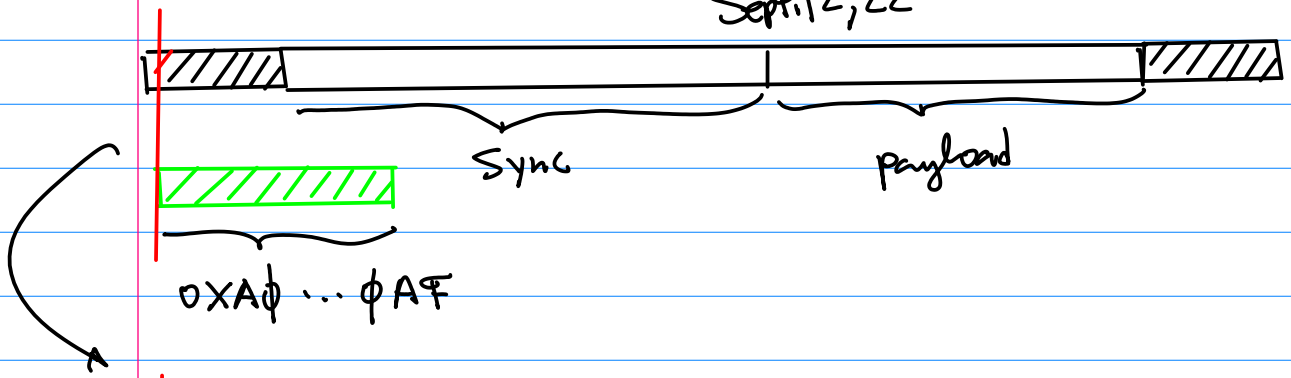


Fig.2

Check the matching result Between (1) (Black) and (2) (Green) if matched, then use the matching index to jump to the 1st Bit of the payload; O/W, continue By shifting the mask 1 bit to a newer location.



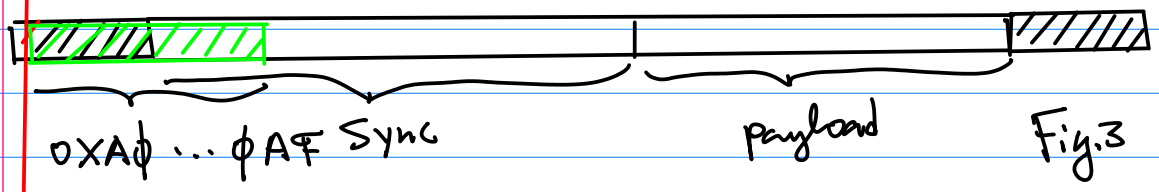


Fig.3

Perform matching operation similar as the one the previous step. if matching is confirmed, then jump to the 1st Bit of the payload Based on the matching index.

O/W. Continue this process (similar as the previous step) by shifting 1 additional Bit to the newer position, repeat the matching process.

This process Continues till the matching is found or the input Buffer (Data) is exhausted.

Note: In terms of the Implementation,
we have an inner "for-Loop," for the "green Region" in Fig.3

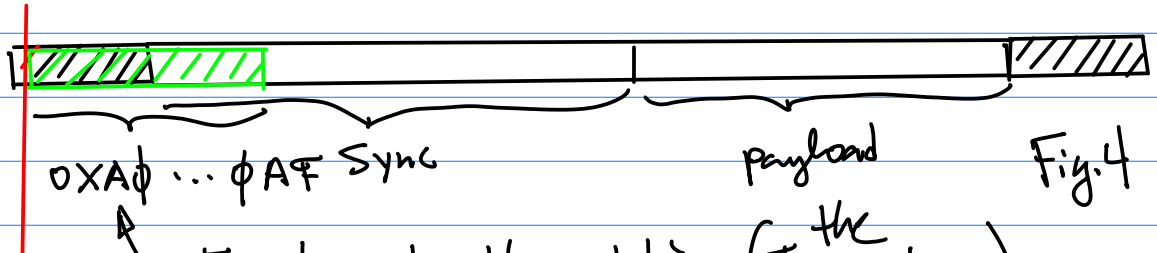


Fig.4

For-Loop for the matching (the Inner Loop).

For the "Red Line" (Newer Position), we can have an outter "for-Loop"

The outter loop will continue till the matching is found or the entire Buffer is exhausted;

Note: Reference to Convolution (1D)

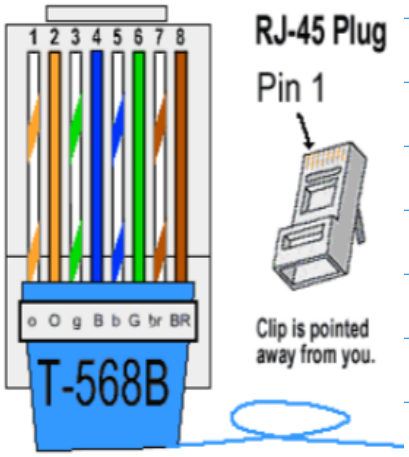
$$\sum_{k=0}^{N-1} h(k)\, g(n-k)$$

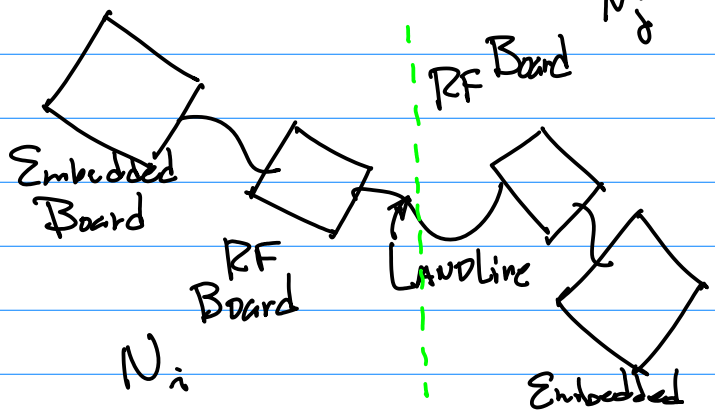
Kernel (e.g. mask)

Its implementation is similar.

Example: Landline Testing.

Note: 8 pos





Select Pos 1. for Tx (P0.2)
N_i ; N_j
Select Pos 3. for Rx (P0.3)
N_i ; N_j
Pos 5. GND.

Build One Slide PPT. With this Photo, And Connectivity table

Choose RJ-45 connector, and CAT5 or Cat6 Cable for Landline.

GPIO is the protocol for the implementation (Not Ethernet).

Hardware Design — For LPC1768 { P0.2 Output, P0.3 Input
Software Design — For NVDA NANO

Physical pin of the J2 Connector

P1.31	AD0.5	J2-20
P0.2		J2-21
P0.3		J2-22
P0.21		J2-23
P0.22		J2-24

2021F-109-II-note-2021-11-10.pdf

Pin12
Pin40

Demo Example
Pin-32

Sept.14

Homework Implementation.

Software Side { NXP LPC1769, or LPC11xx
NVDA Jetson NANO

Prerequisit:

Hardware Side: Embedded System Prototyping.

{ NXP LPC { 1769 { Prototype Board. 2017F-102-lecLayout 2017-2-7.pdf
11CZ4 { CTIONE Board-B from ebay.
NVDA NAND 2021F-114-gpio-nano-v2-hl-2021-10-20.pdf

Step1. GPIO Sample Code

{ NXP LPC1769
NVDA NANO

① Sample code from the github
CMPE240-Adv-Microprocessors / 2018S-11-GPIO-2015-1-30.zip

② NxpDeveloper. www.nxp.com
Sign up as a developer.
Down Load, Install MCUXpresso.

③ Config the IDE (MCUXpresso).

④ Down Load LPC1769 Patch.
Buildit.

C/C++ project Semihost CMPE240-Adv-Microprocessors / 1769 patch.zip

⑤ GPIO Sample code.

CMPE240-Adv-Microprocessors / 2018S-11-GPIO-2015-1-30.zip

To Run the test code, Be sure to have hardware Ready.

CMPE240-Adv-Microprocessors / 2018F / 2022F-101-notes-cmpe240-2022-09-12.pdf

LPCXpresso1769_CD_revD(1).pdf — LPCXpresso LPC1769 CMSIS-DAP rev D.sch