

CMPE245
Sept. 7

Sept. 7.

Note: 1^o Homework (RF module & RF Work-in-Progress)

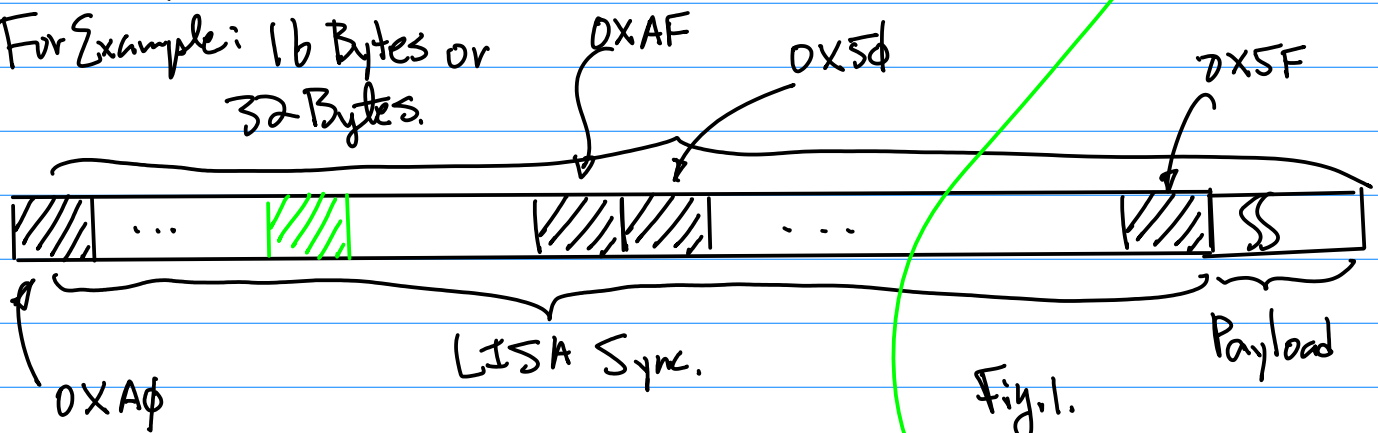
Due today Inspection in Class.

Homework (1st) Due A Week from Today. Write C/C++, OR Python to Implement LISA Algorithm (Phase I)

Such that:

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

For Example: 16 Bytes or 32 Bytes.



2^o 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.

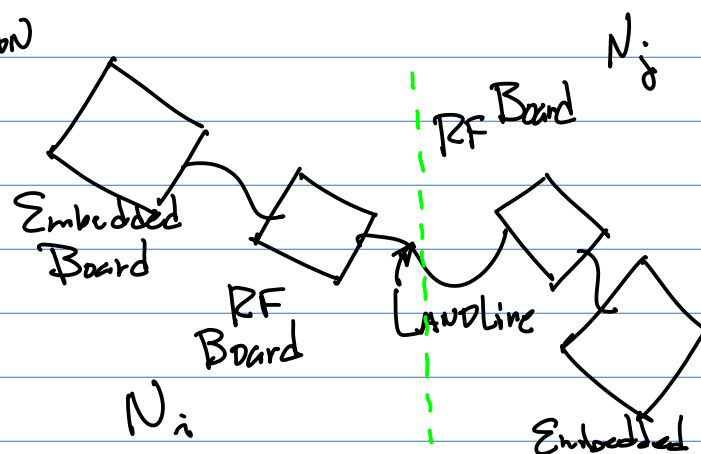
3^o Payload: First Name + Last Name + 4 Digits ID + CMPE245 + SJSU

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.



CMPE245
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. \rightarrow Confidence Level/Index η

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

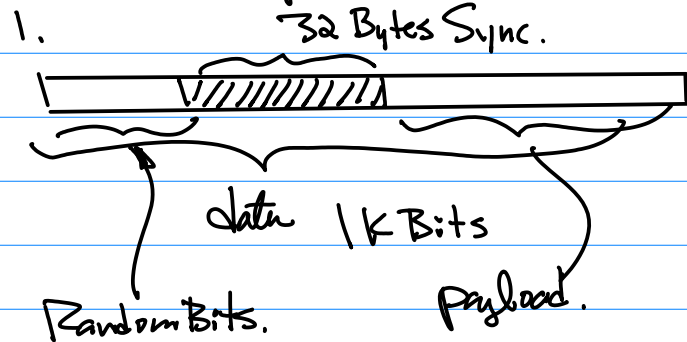
Note: In Software Defined Radio, We can Change η (Confidence Level) to trade the quality for Speed if it is allowed.

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

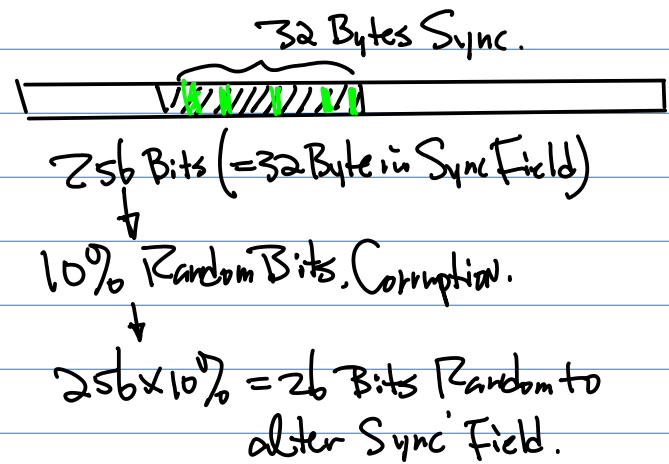
Observation 2: "LINEAR" Characteristics is from the fact LISA Index is defined from D to F with Linear increment. And "Invariant"

Characteristic is due to the fact the ID Index, e.g. Ranging from D 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.



2. Sync Field is Corrupted.



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)

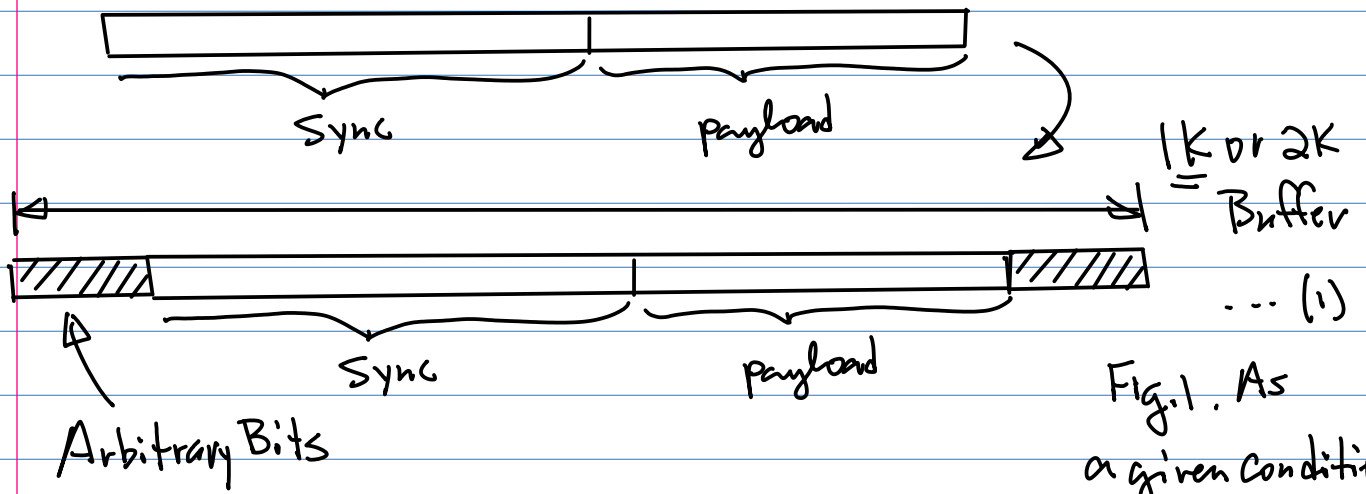
Today's Topics: 1° LISA Homework Implementation. 2° Base Band Signal

CompE245

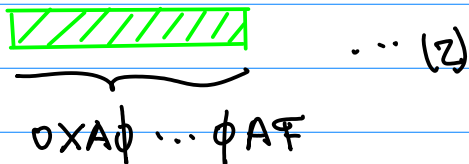
Sept. 12

1/

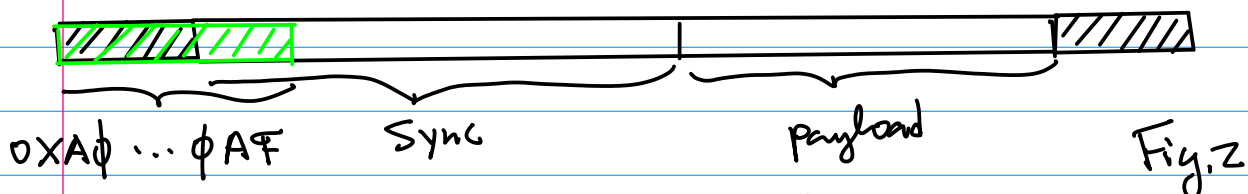
Example: LISA Implementation.



Step 2. Create a "mask" Template to Reflect the matching size that you like



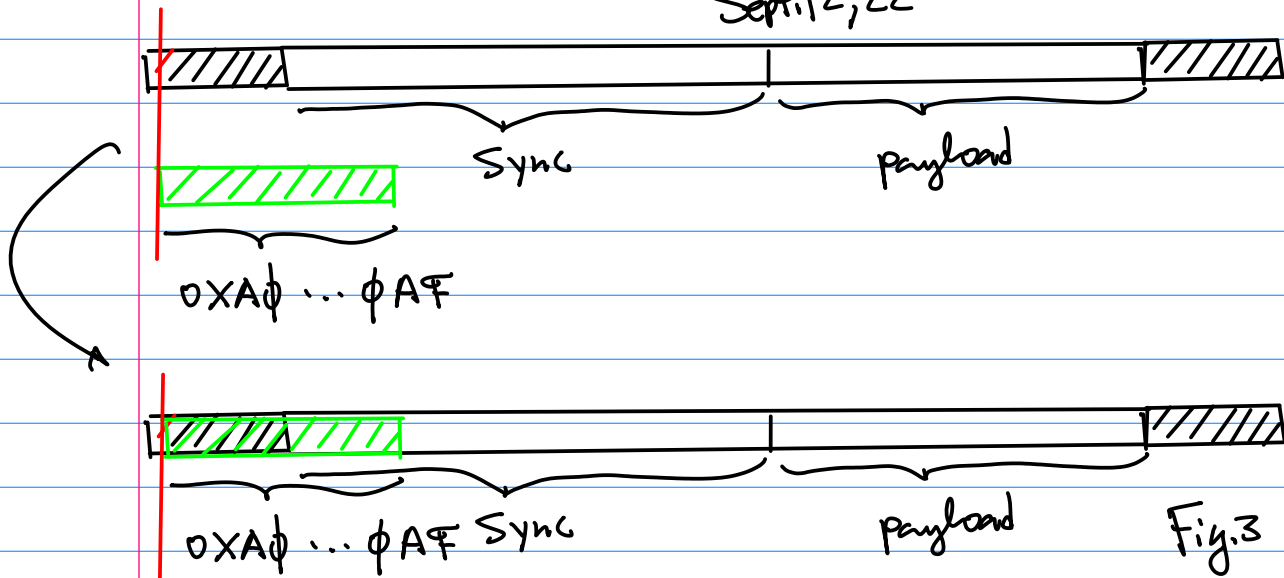
Step 3. Add Random Noise to Fig. 1. then move (2) at the beginning bit (1st Bit) of (1). Such as



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.

Ompe245
Sept. 12, 22

2/

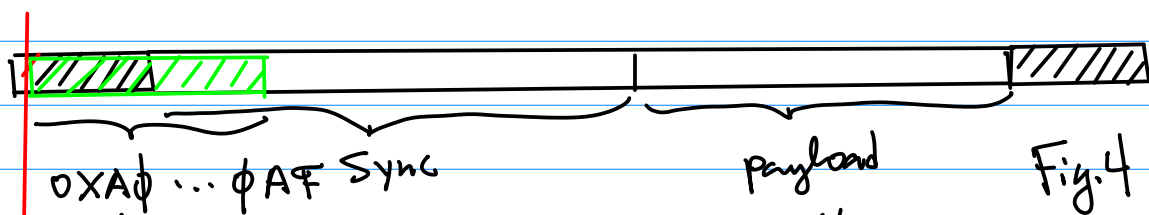


Perform matching operation similar as the one the previous step. if matching is confirmed, then jump to the 1st Bit of the payload Based on the matching index.

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



For-Loop for the matching (Inner Loop).

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

CMPE245
Sept. 12, 22.

3/

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

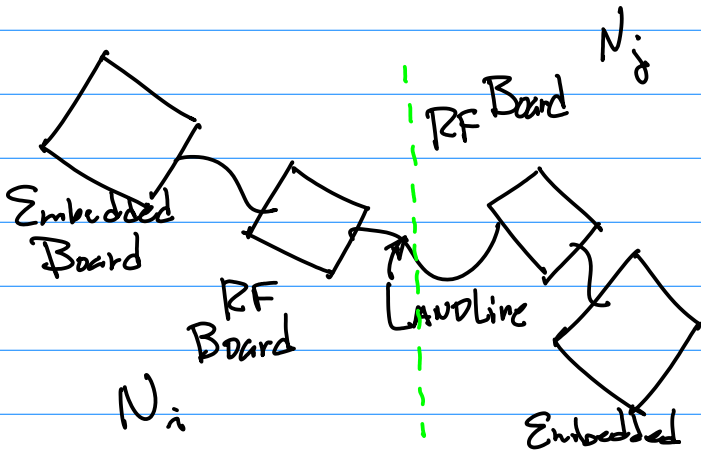
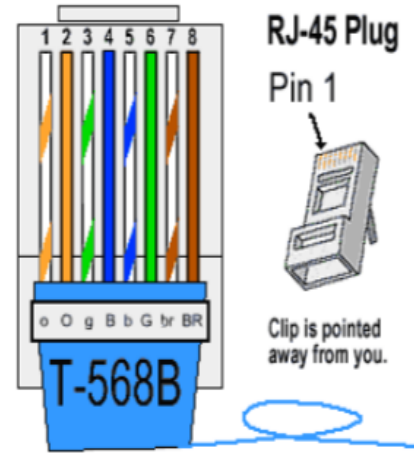
Note: Reference to Convolution (1D)

$$\sum_{k=0}^{N-1} h(k) \underset{\text{Kernel (e.g. mask)}}{g(n-k)}$$

Its implementation is similar.

Example: Landline Testing.

Note: 8 pos



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

Build PptSlide 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
Software Design

For LCR176 P0.2 Output
For NVDA NANO P0.3 Input

Physically pinning the J2 connector

P1.3V	AD0.4	P1.3V	J2-19
P1.31	AD0.5	P1.31	J2-20
P0.2		P0.2	J2-21
P0.3		P0.3	J2-22
P0.21		P0.21	J2-23
P0.22		P0.22-RED LED	J2-24

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

Pin 2
Pin 4

3.3V	5V
GPIO2 (SDA1)	GPIO14 (UART_TXD0)
GPIO3 (SCL1)	GPIO15 (UART_RXD0)
GPIO4 (GPIO_GCLK)	GPIO18 (GPIO_GEN1) PWM
GND	GND
GPIO17 (GPIO_GEN0)	GPIO23 (GPIO_GEN4)
GPIO27 (GPIO_GEN2)	GPIO24 (GPIO_GEN5)
GPIO22 (GPIO_GEN3)	GND
3.3V	GPIO25 (GPIO_GEN6)
GPIO19 (SPI0_MISO)	GPIO26 (SPI0_CE0_N)
GPIO10 (SPI0_CLK)	GPIO7 (SPI0_CE1_N)
GND	ID_0C (I2C EEPROM)
ID_0C (I2C EEPROM)	GND

Demo Example
Pin-32

Note: J

CMPE245
Sept. 14

4/

Sept. 14

Homework Implementation.

Software Side { NXP LPC1769, or
LPC1114
NVIDIA Jetson Nano

Prerequisite:

Hardware Side: Embedded System

Prototyping.

{ NXP LPC { 1769 { Prototype Board. 2017F-102-lecLayout 2017-2-7.pdf
1114 { CTONE Board-B from ebay.
NVIDIA Nano 2021F-114-gpio-nano-v2-h1-2021-10-20.pdf

Step 1. GPIO Sample Code

① Sample code from the github

{ NXP LPC1769
NVIDIA Nano

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

② NXP Developer. www.nxp.com
Sign up as a developer.
Download, Install MCUXpresso.

④ Download LPC1769
Patch.
Build it.

③ Config the IDE (MCUXpresso).

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

P0.3 — J2-22
P0.21 — J2-23

CmPE245
Sept. 21, 22

5.

Sept. 21. Due Oct. 3rd (Monday).
Note: 1^o Homework for LISA
on the target platform.

Objective:

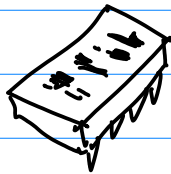
1. To Implement LISA algorithm on the target platform, e.g. LPC1114, or NVD A NAND.
2. To Establish Communication Between Node i & Node j. By Transmitting the following message:

"SJSU_CmPE245_FirstName_LastName_SID(4 Digits)" Submission is on CANVAS.

3. The Testing is Landline Testing.

Requirements:

- 1^o To Have 2 Nodes Sync'd on the Bit Rate, for Example 1 Kbps or lower;
- 2^o. Landline Communication.
- 3^o. Provide LED for Debugging Purpose, tied to GP I/O Output Port. Using "toggle" or D/I P S/W.



Submission:

- 1^o Source Code.
- 2^o Export Project Code for LPC1114 Board;

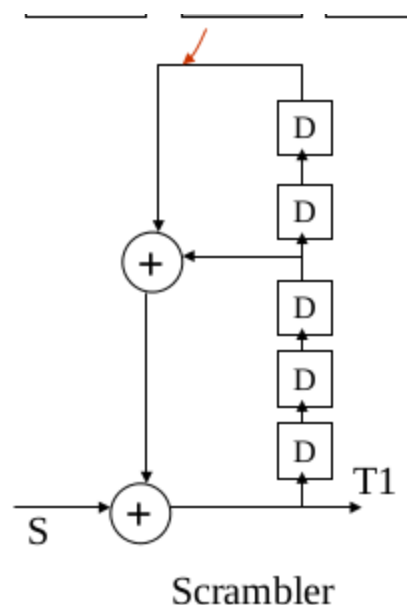
3^o No more than One Page README Document;

4^o photo of Your Testing Environment (Testing System Setup).

5^o 10~15 Seconds Video Clips that Shows the program is working

Note: please Bring your Board to the Class on the 3rd for quick Show-and-tell;

Example: Scrambler/De-Scrambler Design.



Requirements: 1. Design of the Scrambler/De-scrambler is required. for Order N, odd

$N=3,5,7,\dots,13$, etc.

Design Steps for the Scrambler:

1. Divide the Delay units (n) into 2 Banks. (By half)

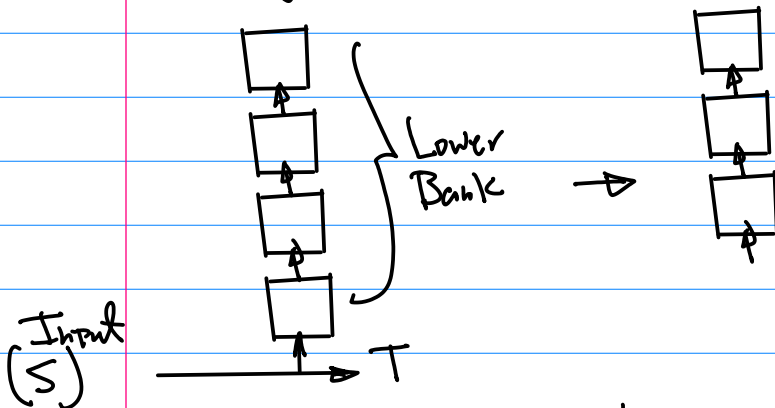
$\frac{N}{2}$ | N is odd Number Rounded up

to define the Lower Bank;
Such that it is always 1 order
higher than the Upper Bank.

Example: for $N=7$.

$7/2 = 3.5$ Rounded up, $3.5 \rightarrow 4$.

Delay unit for the Low Bank.



2. Define 2 XOR operators/Processing units to form feedback loop.

3. Testing the operation of the Scrambler By Sending:

"SJSU_CMP245_FirstName_LastName_SID(4 Digits)"

ASCII coding for Letters & Numerals.

Suppose $S \rightarrow 0X8F$

0X8F

1000 1111

Send Data Out,
MSB first

Test By generating the test
Table, Similar to Table 5.3 below.

Table 5.3. Input and
of the sc
Figure 5.2

Input S	1 0 1 0 1
D^3T_1	0 0 0 1 0
D^2T_1	0 0 0 0 0
Output T_1	1 0 1 1 1

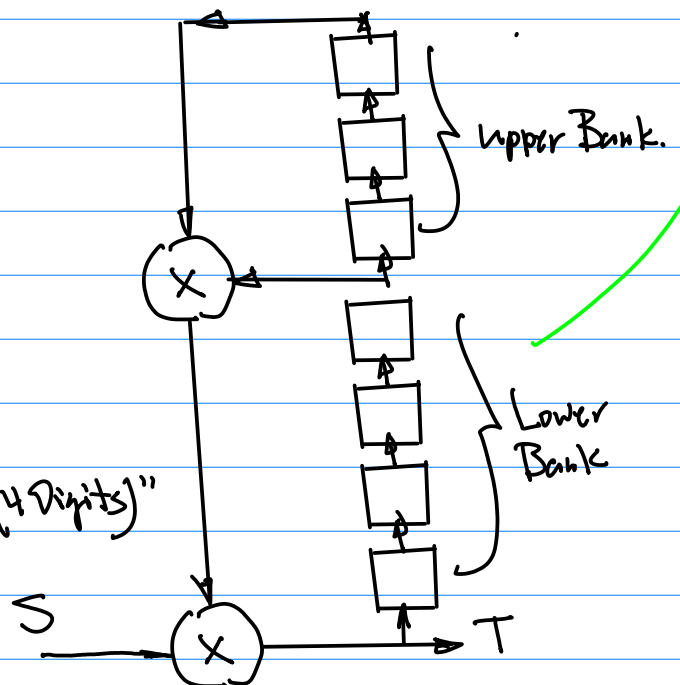
Reference: Di

Input S

Lower Bank D^4T .

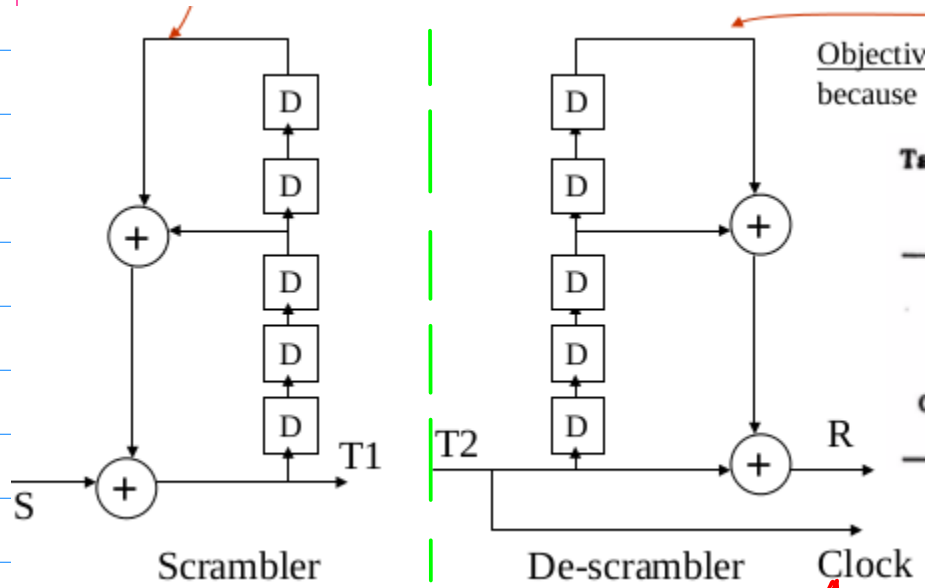
Lower + Upper Bank D^7T

Output T



CMPE245
Sept. 21, 22

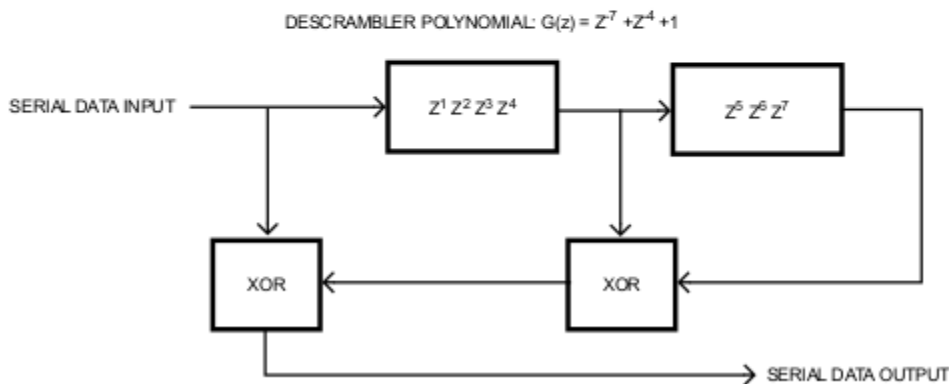
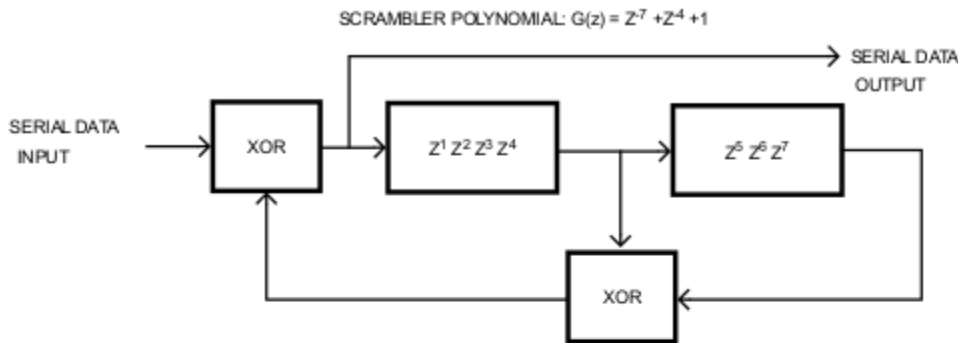
7



Objective because

Note: The Theoretical Analysis and proof of the scrambling/De-scrambling Technique can be Accomplished By using Formal math. Formulation from the Switching Theory.

Note: a. De-scrambler has "Clock" (Sync Extraction Output)
b. Mirror type of Architecture of Both System, allows Design of De-scrambler to be constructed Accordingly.



Consider Base Band Signal Analysis & Formulation.

Motivation: 1. Analyzing the WiFi Communication, Channel Availability & Allocation. a. Power Spectrum of A modulated Base Band Signal is shown in Both Fig. 141 & Fig. 142.

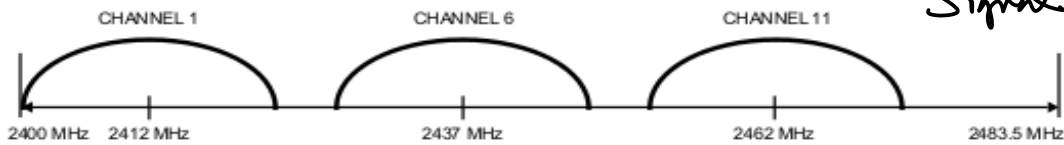


Figure 141—North American channel selection—non-overlapping



Figure 142—North American channel selection—overlapping

a. 2. Fig 145. f_c : Carrier frequency for IEEE WiFi. $f_c \sim 2.4 \text{ GHz} \rightarrow 11$ Channels.

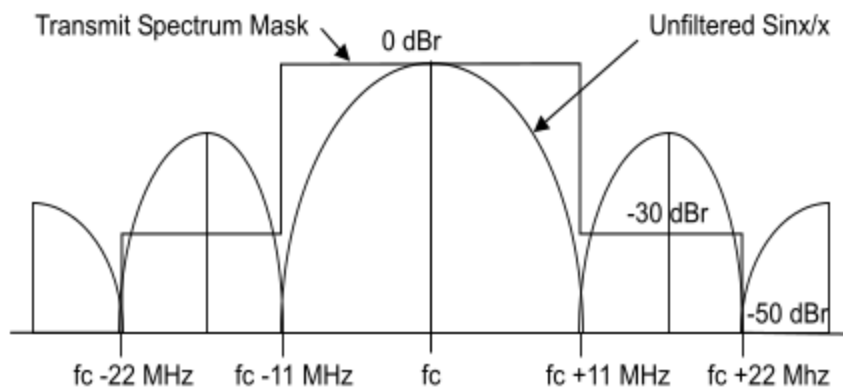
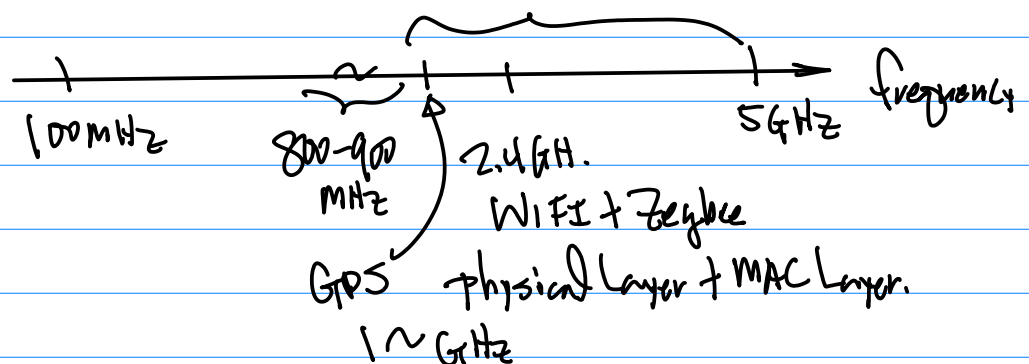


Figure 145—Transmit spectrum mask

b. Base Band Spectrum. Defines the Bandwidth: $f_c + 11 \text{ MHz} - (f_c - 11 \text{ MHz}) = 22 \text{ MHz}$.

c. 80% of the Energy of WiFi has to be kept within the Bandwidth.



CMPE 245
Sept. 21, 22

9





Sept. 26. Monday.

Note: 1. Check Homework on CANVAS. LISA on the target platform with R.F. Board.

2. Optional Target Platform, NVIDIA Jetson NANO, 5-b pieces (ppt)
Sample code have been posted on github.

 [2022F-104-#2021F-114-gpio-connector-sys...](#)

 [2022F-104b-python-gpio-jetson-nano-#202...](#)

 [2022F-105-sd-card-bring-up-nano-2021-10...](#)

 [2022F-106-nomachine-remote-nano-2021-...](#)

 [2022F-107-config#2021F-114b-pwm-nano-...](#)

CMPE245
Sept. 26

Step 1. Background, Pin Assignment. J41. Two Pins for GPIO. (Input, Output Each)

Pin 12	gpio79	input
Pin 40	gpio78	Output

NVIDIA Jetson Nano J41 Header Pino

<https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/>

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

1. take Pin 1 Vcc (3.3V) and Pin 39 GND to test out LED, make sure you can light up a LED with 220 Ohm resistor in series.

	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA	27	28	I2C_1_SCL	I2C Bus 0
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_P20	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOU	gpio78

Sysfs GPIO	Name	Pin	Pin	Name
	3.3 VDC Power	1	2	5.0 VDC Power
	I2C_2_SDA	3	4	5.0 VDC Power
	I2C_2_SCL	5	6	GND
gpio216	AUDIO_MCLK	7	8	UART_2_TX (dev/ttyTIO1)
	GND	9	10	UART_2_RX (dev/ttyTIO1)
gpio50	UART_2_RTS	11	12	I2S_4_SCLK
gpio14	SPI_1_SCK	13	14	GND
gpio194	LCD_TE	15	16	SPI_2_CS1
	3.3 VDC Power	17	18	SPI_2_CS0
gpio16	SPI_1_MOSI	19	20	GND
gpio17	SPI_1_MISO	21	22	SPI_2_MISO
gpio18	SPI_1_SCK	23	24	SPI_1_CS0
	GND	25	26	SPI_1_CS1

Step 2. Bring-up the System. Note: Power (Wall mount Adaptor, ~4000mW) 4W
SD Card. 32GB.

Download A software → Put/Copy the Pre-Built.
"Flasher" Kernel Image

<https://2022F-105-sd-card-bring-up-nano-2021-10-28.pdf>

Write Image to MicroSD Card

<https://developer.nvidia.com/embedded/community>



Prerequisite:
1. A micro-SD card (minimum 16GB) and SD card reader with USB interface;
2. A 5V 3A MicroUSB power supply;
3. An Ethernet cable;

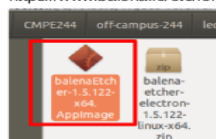
Step 1. Download SD card OS image from Nvidia to your host machine, laptop, the zipped file size is 6.1G, unzip it to get OS image, e.g., *.img file, ref:

<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write>
Harv Li, Ph.D.

```
harry@workstation: /media/harry/easydrive/backup-2020-2-15/$ ls -l
total 4
drwxr-xr-x 2 herry herry 4096 Jan 28 10:28 .
drwxr-xr-x 1 herry herry 4096 Jan 28 10:28 ..
-rw-r--r-- 1 herry herry 1048576 Jan 28 10:28 jetson-nano-j41-plus-sd-card-image.zip
```

Step 2. Write the image to your microSD card by following the instructions from Nvidia, first you will need to download the writer software "etcher" to your host machine from this site:

(2.1) for Linux host, Download, install, and launch Etcher.
<https://www.balena.io/etcher/>



Use USB card r your host mach start it to write



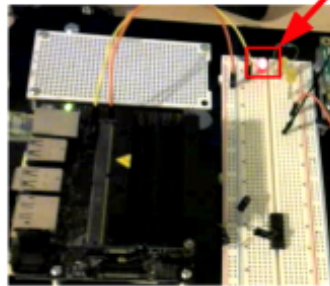
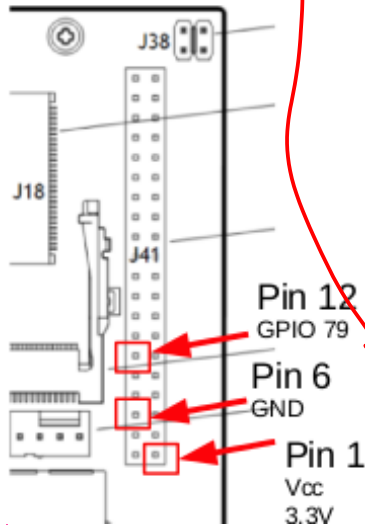
The program * 10-15 minutes then it will valid



Step 3. Test GPIO Input/Output
Ground By Command Line
Instruction.
See Ref on the
github, 2022F-104~

Command Line Information.

Testing J41 40 Pin Connector



We can control our LED

```
# Map GPIO Pin
# gpio79 is pin 12
$ echo 79 > /sys/c
# Set Direction
$ echo out > /sys/c
# Bit Bangin'!
$ echo 1 > /sys/c/
$ echo 0 > /sys/c/
# Unmap GPIO Pin
$ echo 79 > /sys/c
# Query Status
$ cat /sys/kernel/
In the above code, the l
look at the Jetson Nano
header.
```

```
$ echo 79 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio79/direction
$ echo 1 > /sys/class/gpio/gpio79/value
$ echo 0 > /sys/class/gpio/gpio79/value
```

Step 5. In order to Program GPIO Port. First Choose High Level programming Language, Such as Python or C++ . Python is recommended. Then, to Be Able to program device drivers.

Note: please choose Ubuntu 18.04 for the target platform.

Part A
O.S. Kernel Image for the target (Jetson Nano)

Jetson NAND

Part C
J41

Part B

Device Driver

Mapping of Device Drivers from the O.S. Kernel to the target hardware is done by "Configuration" process.

CmPE245
Sept. 26, 22

13

Note: Run Configuration Python Code if Pins Added Note By Factory Default.
Recommended To Use Configuration Mapping.

Step 1. Fix bugs from the distribution

Configuration of Pins with jetson-io.py

```
$sudo find /opt/nvidia/jetson-io/ -mindepth 1 -maxdepth 1 -type d -exec touch {}/_init_.py\;
```

```
$sudo /opt/nvidia/jetson-io/config-by-pin.py -p 5
```

```
harry@harry-desktop:~$ sudo /opt/nvidia/jetson-io/config-by-pin.py -p 5
Traceback (most recent call last):
  File "/opt/nvidia/jetson-io/config-by-pin.py", line 84, in <module>
    main()
  File "/opt/nvidia/jetson-io/config-by-pin.py", line 39, in main
    jetson = board.Board()
  File "/opt/nvidia/jetson-io/jetson/board.py", line 229, in __init__
    self.dtb = _board_get_dtb(self.compat, self.model, dtbdir)
  File "/opt/nvidia/jetson-io/jetson/board.py", line 114, in _board_get_dtb
    raise RuntimeError("No DTB found for %s!" % model)
RuntimeError: No DTB found for NVIDIA Jetson Nano Developer Kit!
```

```
unused ( 37) .. ( 38) unused
GND ( 39) .. ( 40) unused

Jetson 40pin Header:
Configure for compatible hardware
Configure header pins manually
Back
```

```
==== Jetson Expansion Header T
Select desired functions (for
[ ] aud_mclk (7)
[ ] i2s4 (12,35,38)
[*] pwm0 (32)
[*] pwm2 (33)
[ ] spi1 (19,21,23)
[ ] spi2 (13,16,18)
[ ] uarbt-cts/rts (11,36)
Back
```

```
$sudo mkdir -p /boot/dtb
$ls /boot/*.dtb | xargs -I{} sudo ln -s {} /boot/dtb/
```

Step 2. Run jetson-io.py to configure pins

2022F-106-nomachine-remote-nano-2021-12-7.pdf

```
Select one of the following:
Configure Jetson 40pin Header
Configure Jetson Nano CSI Connector
Configure Jetson M.2 Key E Slot
Exit
```

Be sure to choose save and reboot to reboot the system

```
NA ( 29) .. ( 30) GND
NA ( 31) .. ( 32) pwm0
pwm2 ( 33) .. ( 34) GND
unused ( 35) .. ( 36) unused
unused ( 37) .. ( 38) unused
GND ( 39) .. ( 40) unused

Jetson 40pin Header:
Configuration saved to file
/boot/tegra210-p3448-0000-p3449-0000-a02-hdr40-user-custom.d
Press any key to go back
```

Stepb. Setup Remote Access for the purpose of using your laptop Keyboard, mouse, And Display.

2022F-106-nomachine-remote-nano-2021-12-7.pdf

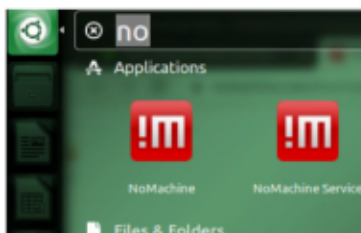
Nomachine for Jetson NANO

<https://www.nomachine.com/download/download&id=116&s=ARM>

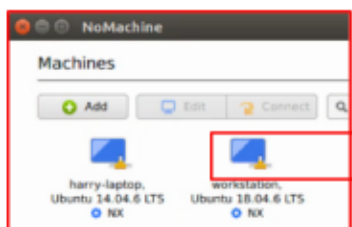
Aarch64 version 7.74_1; size: 42.29 MB, type: TAR.GZ

Follow nomachine website info for installation, I have installed it in my 'Document\NX' folder, you could installed it in 'usr\NX' folder

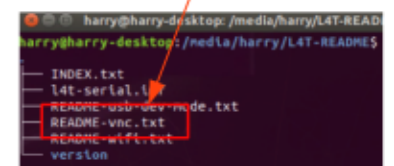
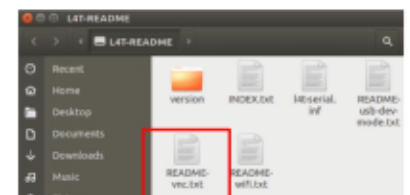
On nano after the installation, you can see this



Once you start nomachine on your laptop, and enter the right user name and password of the nano, you can see it now

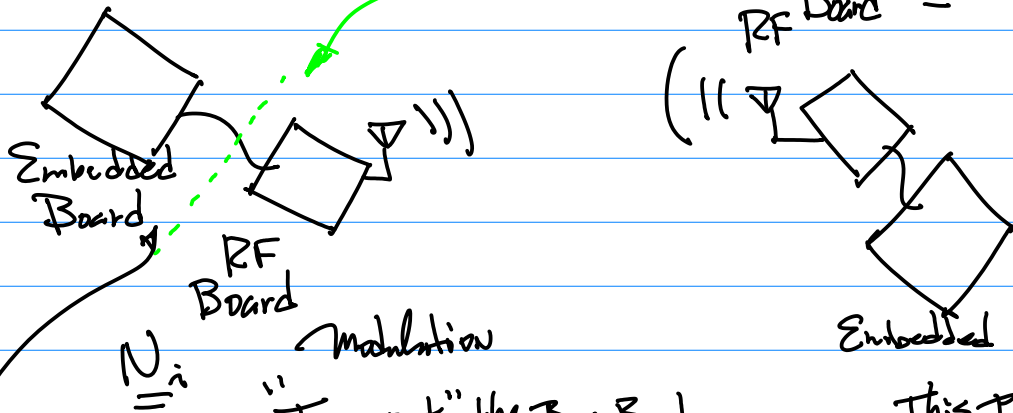


Note: From NVDA L4T installation on NANO, you can see VNC installation readme, below



Continued for the Base Band
Signal Discussion.

Example: Communication B/W N_i & N_j
"SJSU-CMPE245" To Be Sent. \rightarrow ASCII \rightarrow "0's & 1's", Base Band Signal N_j



Modulation

"Transport" the Base Band
Signal to a proper frequency
Range for Desired Characteristics.

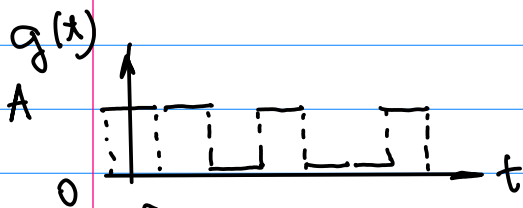
This Process (Modulation) will
 \rightarrow Satisfies 80% Energy Distribution
Requirements. By IEEE/FCC.

a. The waveform "shape" of the Base Band
Signal is well designed to meet
the 80% energy Requirement.

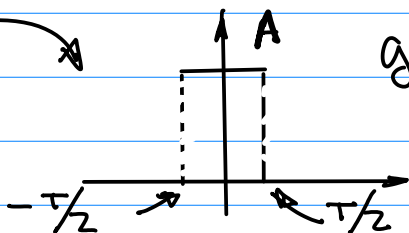
b. The "Transporting" (e.g. modulation)
is most effective and provide
"Robustness" (e.g. Resisting to Random
Noise)

Example: Base Band in Time Domain.

Given a Sequence of A.B.B. (Base
Band) Signal, in Figure 1.



$$g(t) = \begin{cases} A & t \in [-T/2, T/2] \\ 0 & \text{o/w} \\ \dots (1) \end{cases}$$



$$g(t - kT), \text{ for } k=0, 1, 2, \dots$$

For A sequence of BB:

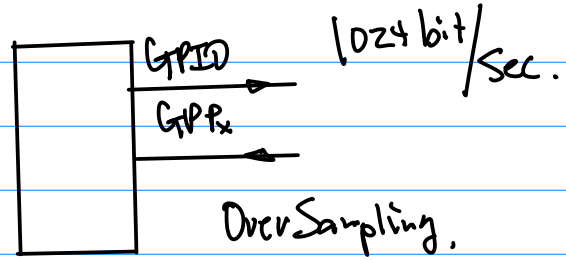
$$\sum_{k=0}^N g(t - kT)$$

Define Fourier Transform

$$F[g(t)] \triangleq \frac{1}{T} \int_T g(t) e^{j2\pi f t} dt \dots (2)$$

$$g(t) \longleftrightarrow F[g(t)]$$

$$F[g(t)] = A_T \frac{\sin \pi f T}{\pi f T} \dots (3)$$



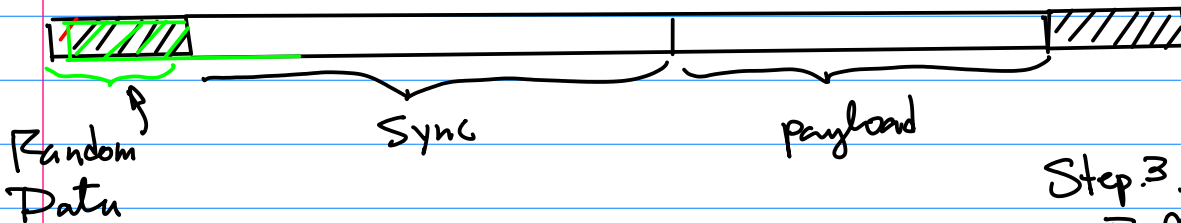
LPC1769 GPP Reading is 5×1024
OR
NAND

Sept 28 Wed.

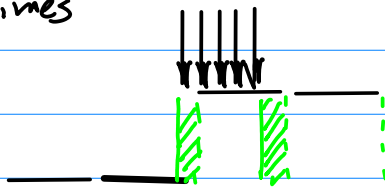
Note 1^o Regarding the Homework
on Landline testing of LISA.

Buffer of 1K or 2K bits for the
Homework can be implemented.

Step 1. \rightarrow Step 2 for Every 5
Read from GPP_x Consecutive Read,
(a) Clock Rate $5 \times$ Bit Rate Voting Mechanism
is employed
to get one
Bit



2^o Consider Using Over Sampling.
e.g. for each bit, Read 5
Times



"Voting", majority Samples
will be considered as the
actual Bit Value.

Step 4.
Start "LISA"
Parsing to
establish
Sync.

Step 3. Fill in the 1K
Buffer, Continue
till the Buffer is
filled.

Example: BaseBand Signal Analysis

\downarrow
GPPD Reading. $5 \times$ faster than
the Bit Rate. for bit Rate = 1Kbps
(= 1024 bits/Sec.) \rightarrow Need Clock
with Interrupt to define the GPPD Reading.

Base Band Signal

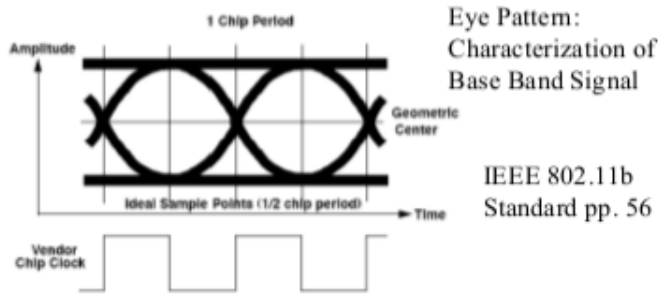
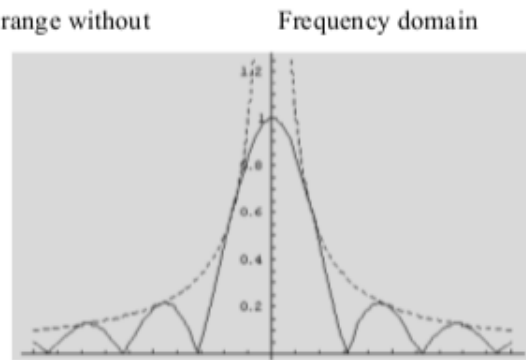
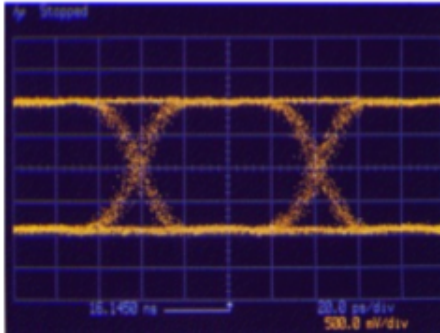
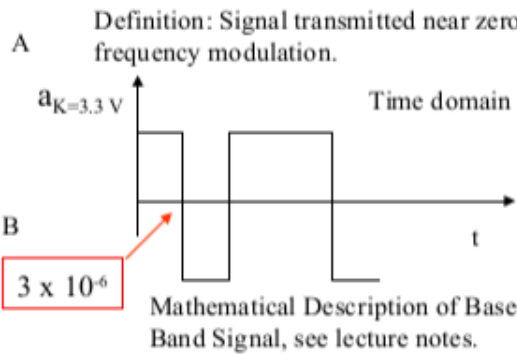
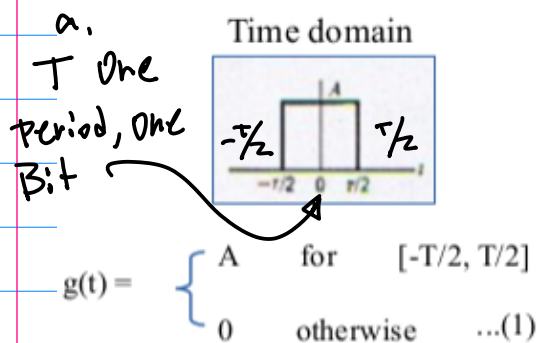


Figure 149—Chip clock alignment with baseband eye pattern

IEEE 802.11b Standard pp. 56

Base Band Signal Formulation



$$A \tau \frac{\sin \pi f \tau}{\pi f \tau} \dots(2)$$

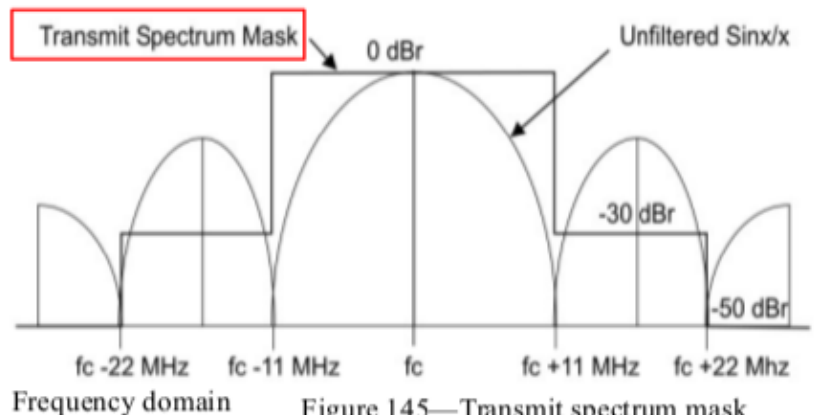
Example: Calculation of bandwidth

Let equation (2) = 0, then we have

$\pi \cdot f \cdot T = k \cdot \pi$

Let $k=1$ for the first pair of zero crossings, we have

$$f = 1/T$$



Counting both sides of the spectrum, we have

$$BW = 2/T$$

b. $f_b = \frac{1}{T_b}$
frequency

$$BW = 2 f_b$$

CompE245

Sept. 26, 22

16

Consider the Definition/Derivation of the Bandwidth (BW).

From BB Signal in Egn(1), p3. We perform F.T. in Egn(2).

$$A_T \frac{\sin \pi f T}{\pi f T}$$

We can define a Power Spectrum of the Signal By using its F.T.

$$P(f) = \sqrt{\text{Re}[F(f)]^2 + \text{Im}[F(f)]^2} \quad \dots (3)$$

Where $F(f) \triangleq \mathcal{F}[g(t)]$

Fourier Transform of $g(t)$.

Real Part of the F.T.

$$\text{Re}[F(g(t))]$$

Imaginary Part:

$$\text{Im}[F(g(t))]$$

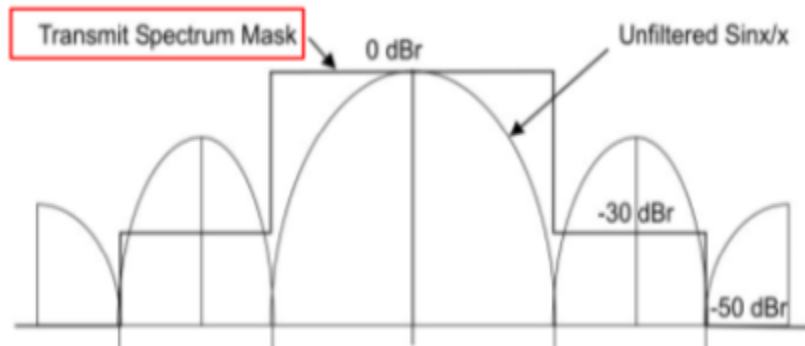
Note: $g(t) \longleftrightarrow \mathcal{F}[g(t)] = F(f)$

Note for the F.T. of the Base Band Signal $g(t)$, it is the following,

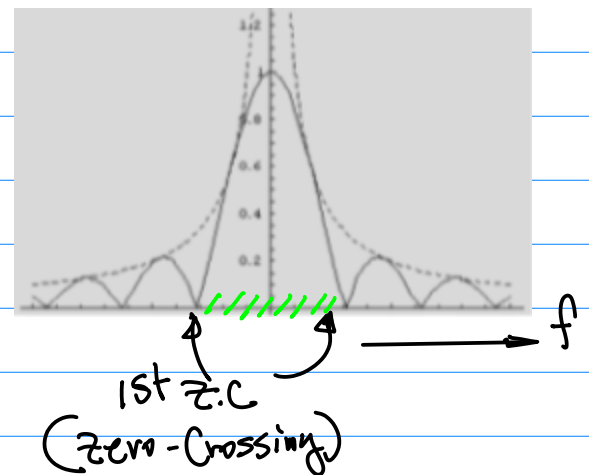
$$A_T \frac{\sin \pi f T}{\pi f T}$$

which only has the Real Part, therefore its F.T and Power Spectrum is the Same.

Plot the power Spectrum.



OR,



Define the Bandwidth of the Base Band

Note:

$$\frac{1}{T} \int_T g(t) e^{j2\pi f t} dt$$

Where

$$e^{j2\pi f t} = \cos 2\pi f t - j \sin 2\pi f t$$

(Euler Formula)

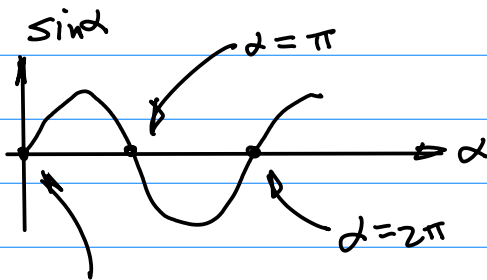
$$\text{Let } A_T \frac{\sin \pi f T}{\pi f T} = 0$$

$$\downarrow$$

$$A_T \frac{\sin \pi f T}{\pi f T} = 0$$

Multiplying $\pi f T$ By Both Sides of the Equation

$$\sin \pi f T = 0$$



$$\sin \alpha = 0 \Rightarrow \alpha = 0$$

Hence,

$$\pi f T = k\pi$$

for $k=0, 1, 2, \dots$

$$f T = k$$

$$f = k/T$$

frequency Index, where $k=1$ for the 1st z.c.

$$f = 1/T$$

Now, the other side in the Frequency Range, we have $f = -1/T$ for the 1st z.c.

To find the Bandwidth, we have

$$\text{B.W.} = 1/T - (-1/T) = 2/T \quad \dots (4)$$

Example: Find the Bandwidth of the Signal in Our Homework.

Sol:

$$\text{B.W.} = 2/T$$

$$T = 1/f, \quad f \text{ is the Bit Rate which is 1 Kbps.}$$

$$f = 1024 \text{ Hz}$$

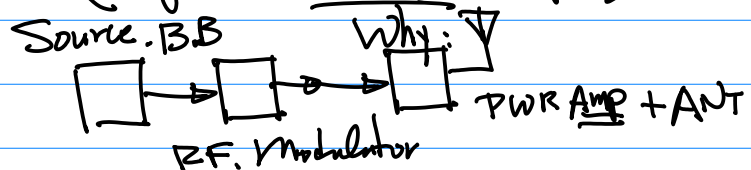
$$T = 1/1024$$

$$\therefore \text{B.W.} = 2/T = 2f = 2048 \text{ Hz}$$

Observation 1: BW and Bit Rate are

Connected by Equation (4). To Double the Bit Rate, By Eqn (4), we have to Double the Bandwidth, which is NOT desirable or practical. From Now on, we will try to Develop Technology to increase the Bit Rate While Keep the Same Bandwidth.

Consider The Tool for Analyzing the Energy Distribution of A given R.F. (e.g. of A Base Band Signal).



Define D.F.T. for this purpose:

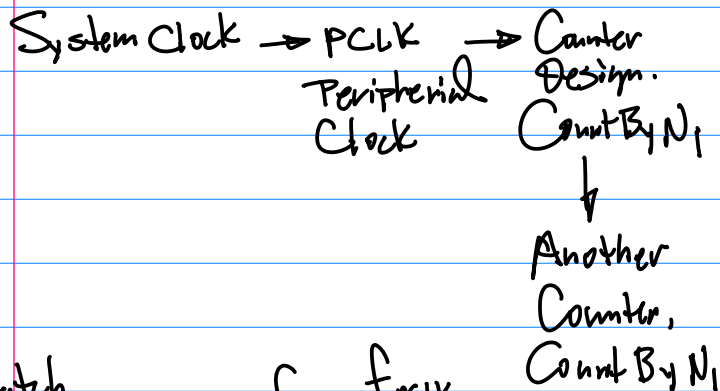
$$\bar{X}(m) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}} \dots (5)$$

Oct 5.

Q & A Session.

Timer/Timing

Background on Time Design.



Match Register.

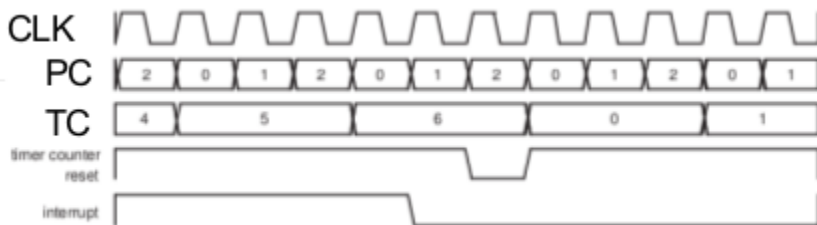
$$f = \frac{f_{\text{CLK}}}{N_1 N_2}$$

$$Reg_m = N_1 N_2$$

More Granulation/Resolution.

\downarrow
Trigger An Interrupt. (Waveform, pp. 3)

2017F-120-INT-TIMER-Part2-v3-HL-2017-12-4.pdf



1. Note: SPRs .

PR: Prescaler Register.

TC: Timer Counter.

PC: Prescale Counter.

MR: Match Register

19

(3) Set Prescaler

Note: 4 important registers, PR, PC, TC, and MR

Two steps: first, PCLKSEL register, Peripheral Clock Selection register, Timer0 clock; Section 4.7.3, pp. 57;

2nd, LPC TIM0 -> PR Prescaler register. When the

Prescale Counter (PC) is equal to PR,

Timer Counter (TC) is incremented by 1;

e.g.,

TC is incremented every PR+1 cycles of PCLK.

Bit	Symbol	Description
1:0	PCLK_WDT	Peripheral clock select
3:2	PCLK_TIMER0	Peripheral clock select
5:4	PCLK_TIMER1	Peripheral clock select

2. Algorithm.

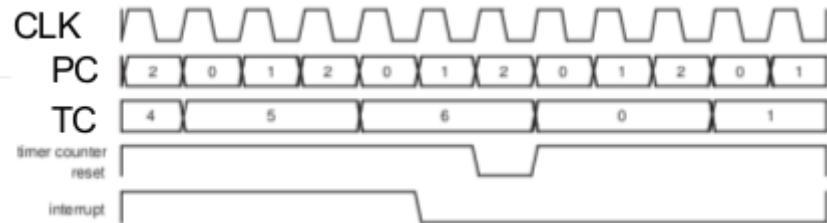
Example:

Given PR = 2, MR = 6; An interrupt generated on match.

(1) at every clock, PC -> PC+1;

(2) when PC = PR, TC -> TC+1;

(3) when TC = MR, INT generated.



PR: Prescaler Register.

TC: Timer Counter.

PC: Prescale Counter.

MR: Match Register

Suppose $f_{\text{PCLK}} = 20 \text{ MHz} = 20 \times 10^6 \text{ Hz}$.

$f_{\text{INT}} = 1 \times 10^3 \text{ Hz}$.

$$f_{\text{PCLK}} = K \cdot f_{\text{INT}} \quad \dots (1)$$

Where $K = N_1 \cdot N_2 = N_{\text{PR}} \cdot N_{\text{MR}} \quad \dots (2)$

Hence,

$$f_{\text{PCLK}} = N_{\text{PR}} \cdot N_{\text{MR}} \cdot f_{\text{INT}} \quad \dots (3)$$

$$\frac{f_{\text{PCLK}}}{f_{\text{INT}}} = N_{\text{PR}} \cdot N_{\text{MR}}$$

$$\frac{20 \times 10^6}{1 \times 10^3} = N_{\text{PR}} \cdot N_{\text{MR}}$$

$$2 \times 10^4 = N_{\text{PR}} \cdot N_{\text{MR}}$$

32 Bit Register

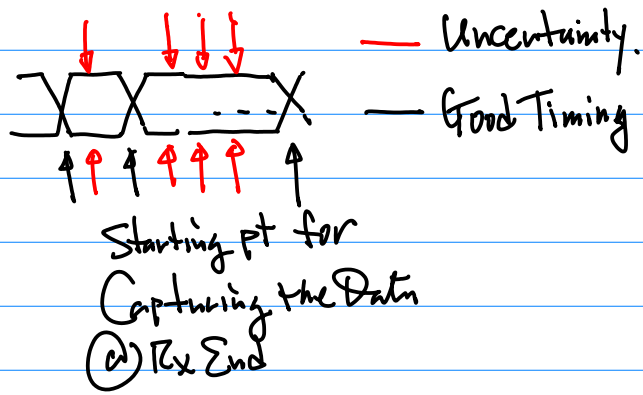
Assume 8-bit minimum usable $[0, 255]$

$$\text{Let } N_{\text{PR}} = 255$$

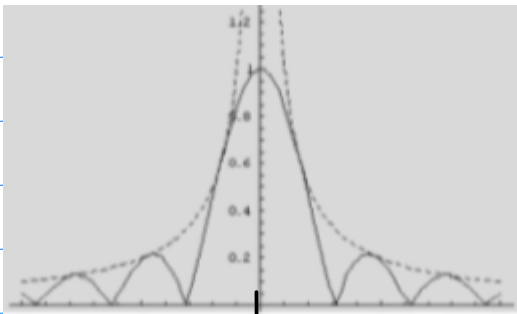
$$N_{\text{MR}} = \frac{2 \times 10^4}{255} \approx 78$$

Breadboard

B.B. Wire Wrapping Soldering
 $\sim 5 \text{ MHz} \sim 20 \text{ MHz} \sim 50 \text{ MHz}$


 $N_i(T_x)$
 f_{clk_i}
 $N_j(R_x)$
 $f_{clk_j} = 5 f_{clk_i}$

Example: Power Distribution/Power Spectrum



Continuous
F.T.

Sampling Theorem.

$$f_{\text{Sampling}} \geq 2f_{\text{Max}}$$

Discrete F.T.

