CMPE245
Sept. 7

Sept 7.
Note: 1° 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° 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.

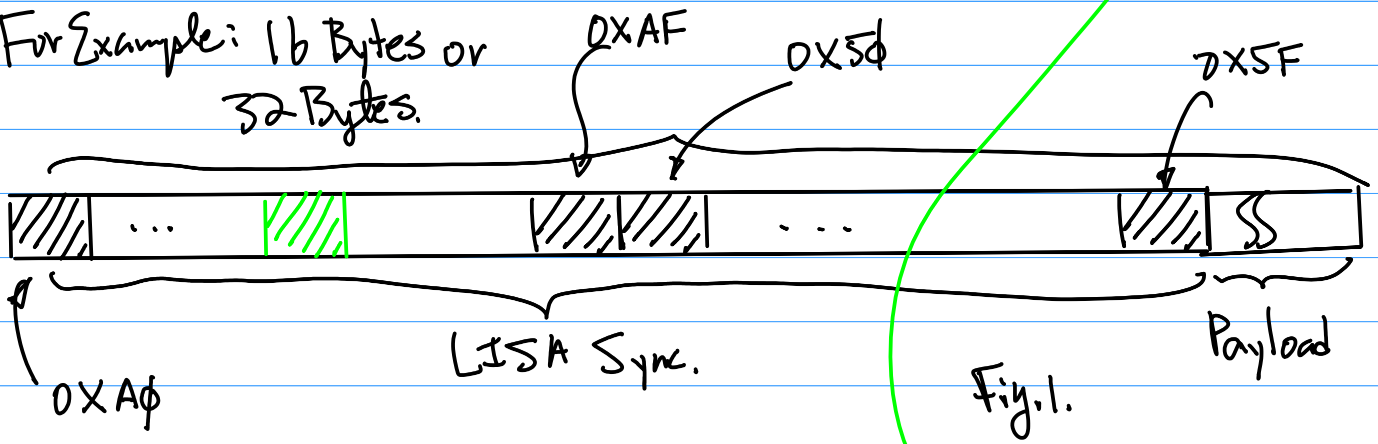
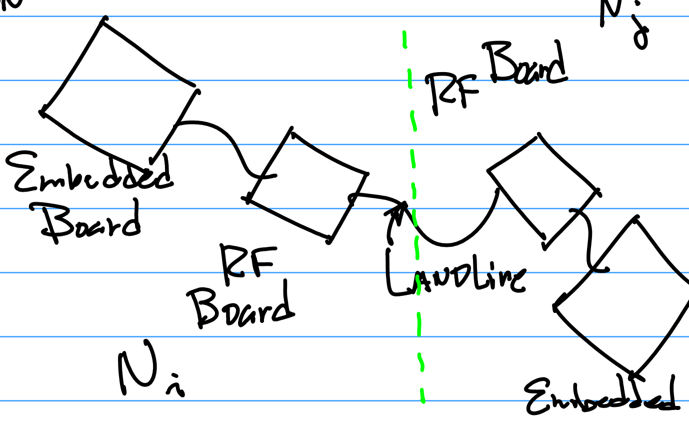


Fig. 1.

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



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. → Confidence Level / Index η

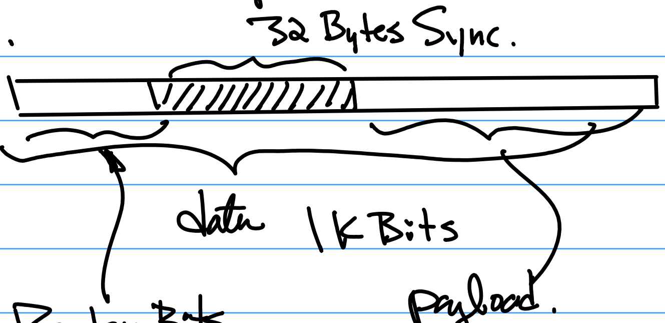
$$\eta = \frac{\text{No. of Bytes to Establish Sync.}}{\text{Total Number of Bytes (32)}} \quad \cdots (1).$$

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.

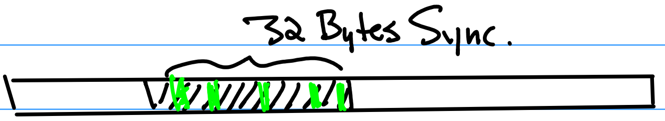
Observation 2: "LINEAR" characteristics is from the fact LISA Index is defined from 0 to F with Linear increment. And "Invariant" Characteristic is due to the fact the ID Index, e.g. Ranging from 0 to F will allow the Algorithm to pin point to the Begining of the payload.

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

CmpE245
Sept.12

Example: LISA Implementation.

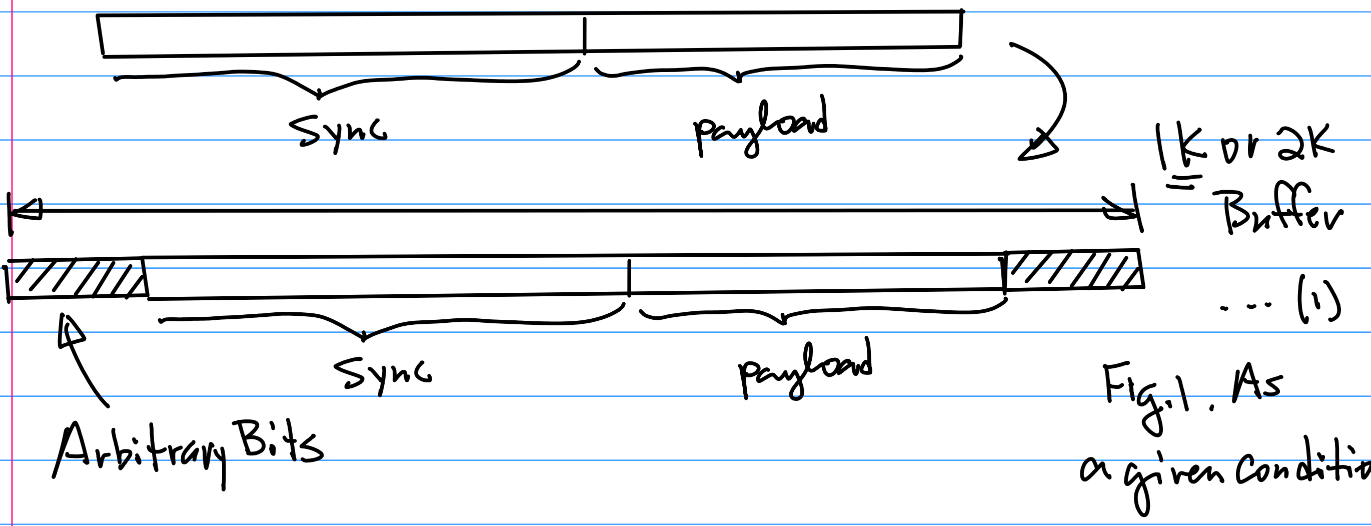
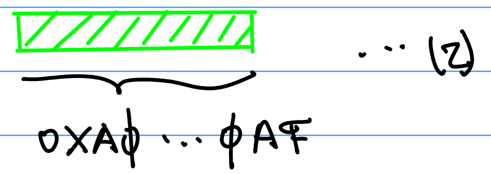


Fig.1. As a given condition.

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) mask. Such as

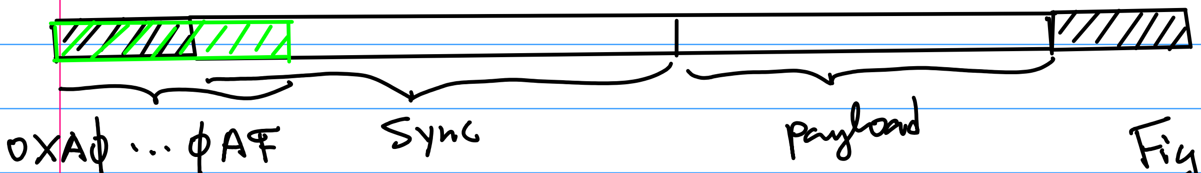


Fig.2

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.

CmpE245
Sept.12,22

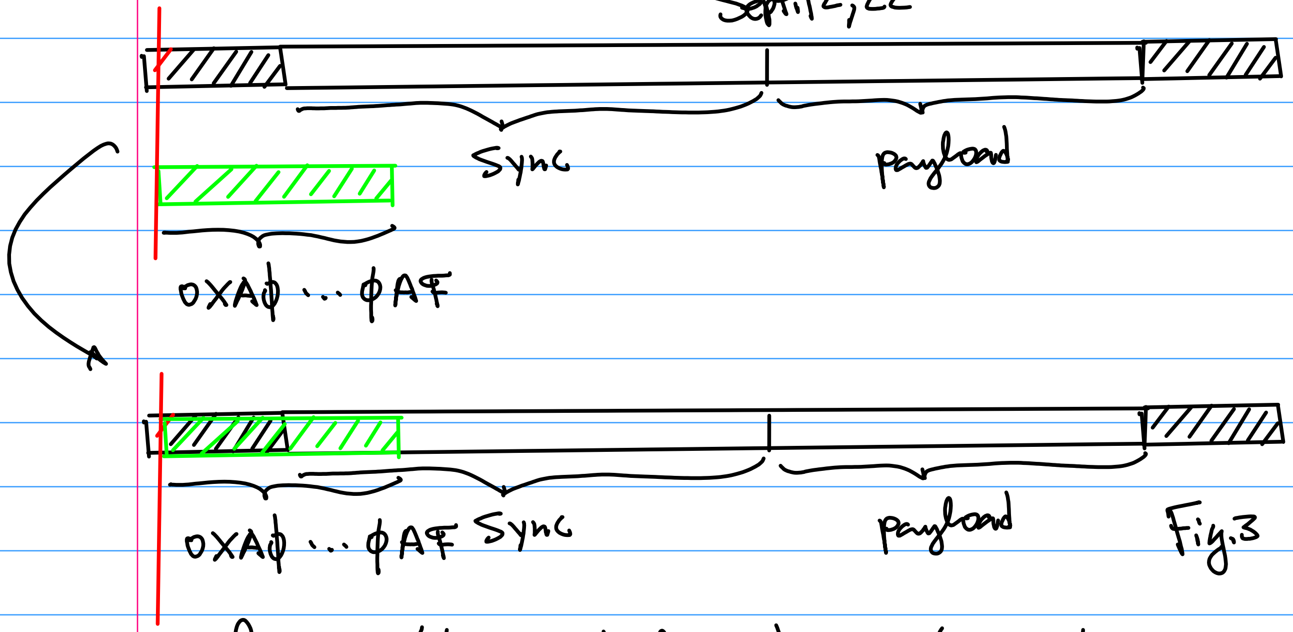


Fig.3

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 Fig3

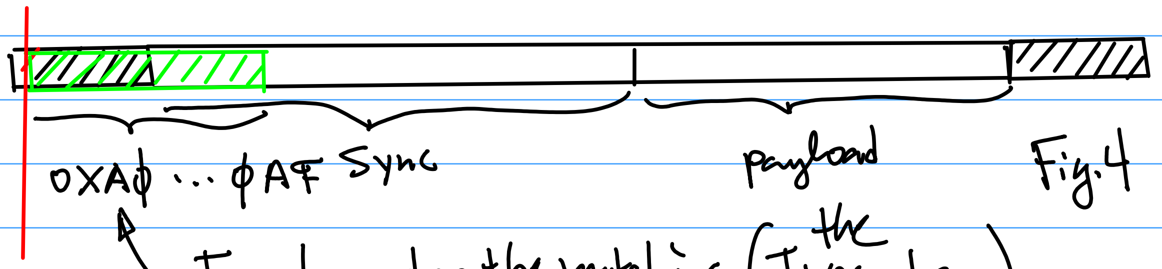


Fig.4

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

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

CmpE245

Sept. 12, 22.

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

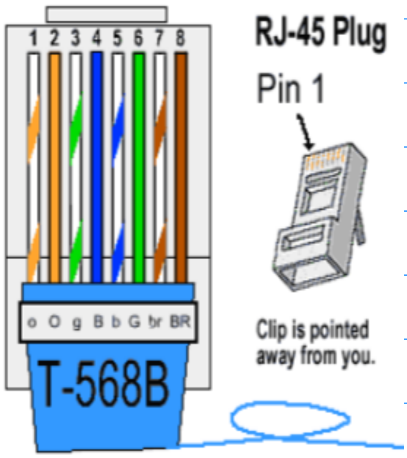
Note: Reference to Convolution (10)

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

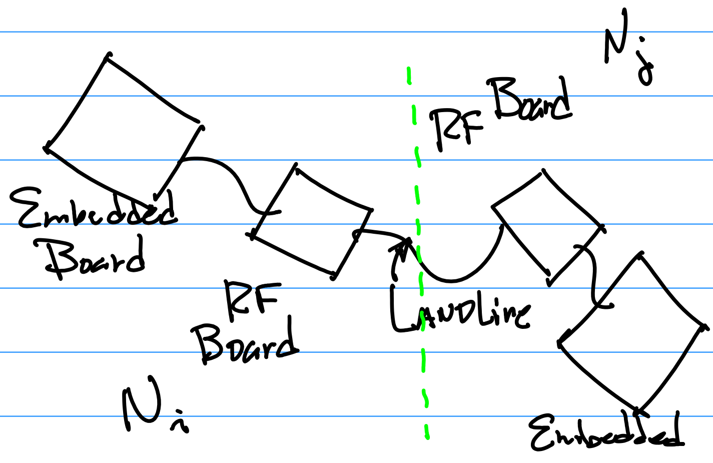
Kernel (e.g. mask)

Its implementation is similar.

Note: 8 pos



Example: Landline Testing.



Select Pos 1. for Tx (P0.2) N_i; j

Select Pos 3. for Rx (P0.3) N_i; j

Pos 5. GND.

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

Choose RJ45 Connector, and CAT5 or CAT6 Cable for Landline.

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

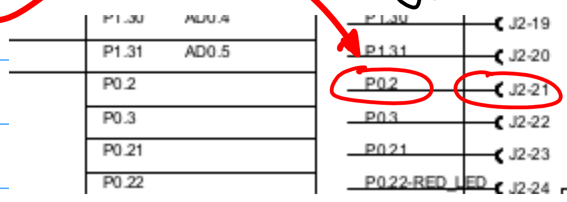
Hardware Design — For LPC1769 { P0.2 Output, P0.3 Input }

Software Design For NVDA NANO

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

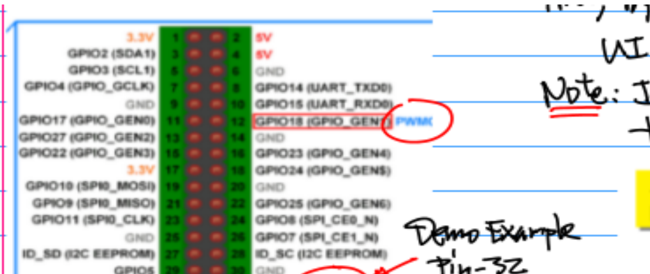
Pin 12
Pin 40

Physical pin @ the J2 Connector



UI

Note: J


3.3V 1 2 5V
GPIO2 (SDA1) 3 4 5V
GPIO3 (SCL1) 5 6 GND
GPIO4 (GPIO_GCLK) 7 8 GPIO14 (UART_TXD0)
GND 9 10 GPIO15 (UART_RXD0)
GPIO17 (GPIO_GEN0) 11 12 GPIO18 (GPIO_GEN1) PWM
GPIO27 (GPIO_GEN2) 13 14 GND
GPIO22 (GPIO_GEN3) 15 16 GPIO23 (GPIO_GEN4)
3.3V 17 18 GPIO24 (GPIO_GEN5)
GPIO10 (SPI0_MOSI) 19 20 GND
GPIO9 (SPI0_MISO) 21 22 GPIO25 (GPIO_GEN6)
GPIO11 (SPI0_CLK) 23 24 GPIO8 (SPI_CE0_N)
GND 25 26 GPIO7 (SPI_CE1_N)
ID_SD (I2C EEPROM) 27 28 ID_SC (I2C EEPROM)
GPIO5 29 30 GND

Demo Example Pin-32

Sept.14

Homework Implementation.

Software Side { NXP LPC1769, or LPC11xx
NVDA Jetson NANO

Prerequsit:

Hardware Side: Embedded System Prototyping.

{ NXP LPC { 1769 / 11C24 — { Prototype Board. 2017F-102-lecLayout 2017-2-7.pdf / CTIONE Board-B from ebay.
NVDA NAND 2021F-114-gpio-nano-v2-hl-2021-10-20.pdf

Step 1. GPIO Sample Code

{ NXP LPC1769 / NVDA NANO

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

↓

② Nxp Developer. 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

↓

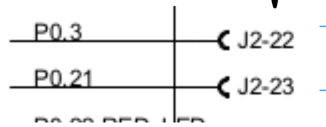
⑤ GPIO Sample code.

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

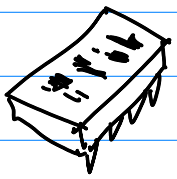
LPCXpresso1769_CD_revD(1).pdf — LPCXpresso LPC1769 CMSIS-DAP rev D.sch



CmpE245
Sept. 21, 22

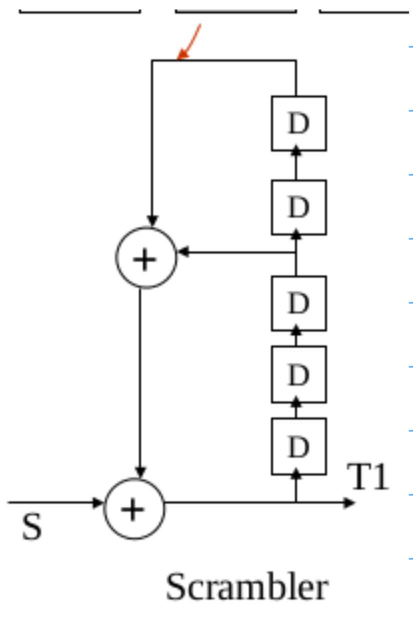
Sept. 21. Due Oct. 3rd (Monday).
Note: 1° Homework for LISA on the target platform.
Objective:
1. To Implement LISA algorithm on The target platform, e.g. LPC1769, or NVDA NANO.
2. To Establish Communication Between Node i & Node j. By Transmitting the following message:
"SJSU_CmpE245_FirstName_LastName_SID(4 Digits)"
3. The Testing is Land line Testing.

Requirements:
1° To have 2 Nodes Sync'd on the Bit Rate, for Example 1Kbps or lower;
2°. Land line Communication.
3°. Provide LED for Debugging Purpose, tied to GPIO Output Port. using "toogle" or DIP S/W.



Submission:
1° Source Code.
2° Export Project Code for LPC1769 Board;
3° No more than One page Readme Document;
4° photo of Your Testing Environment (Testing System Setup).
5° 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;
Submission is on CANVAS.

Example: Scrambler/De-scrambler Design.



Scrambler

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

CmpE245
Sept. 21, 22

N=3, 5, 7, ..., 13, etc.

Design Steps for the Scrambler:

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

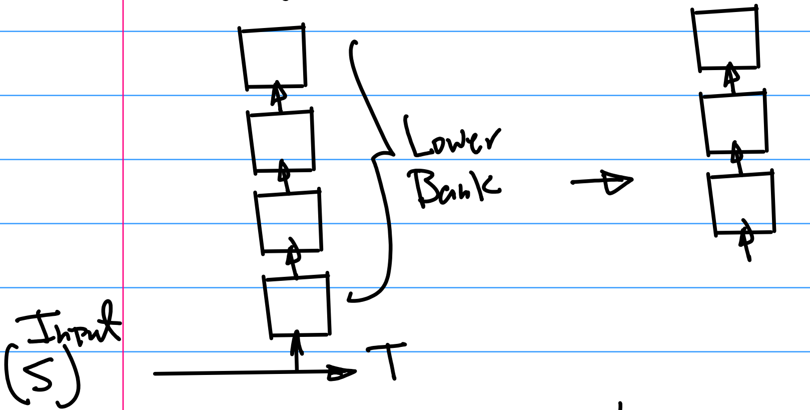
$\frac{N}{2}\Big|_{N \text{ is odd Number}}$ Rounded up

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, $3.5 \rightarrow \underline{\underline{4}}$.

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_CmpE245_FirstName_LastName_SID(4 Digits)"

ASCII Coding for Letters & Numerals.

Suppose $S \rightarrow 0X8F$

0X8F

1000 1111 Send Data Out, $\underline{\text{MSB}}$ first

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

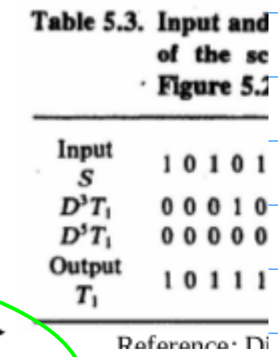
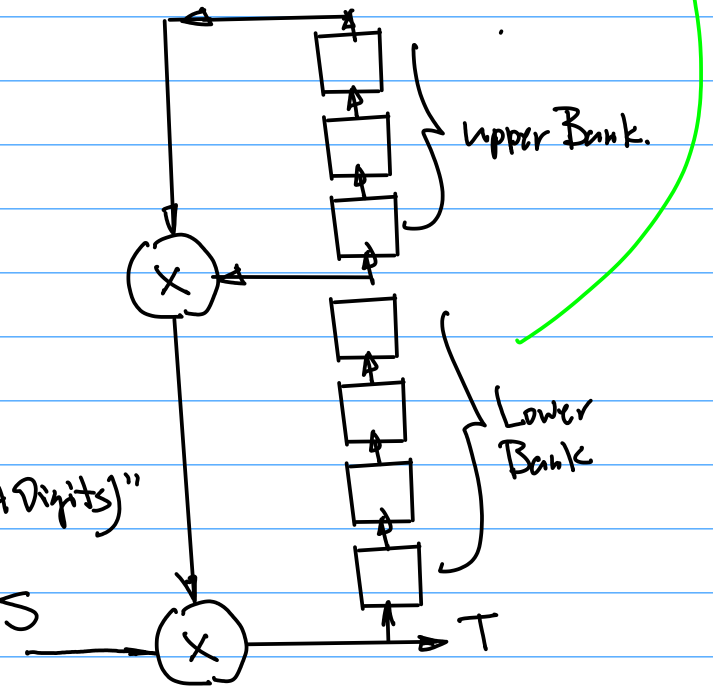


Table 5.3. Input and
of the sc
Figure 5.

Input S	1 0 1 0 1
D^3T_1	0 0 0 1 0
D^5T_1	0 0 0 0 0
Output T_1	1 0 1 1 1

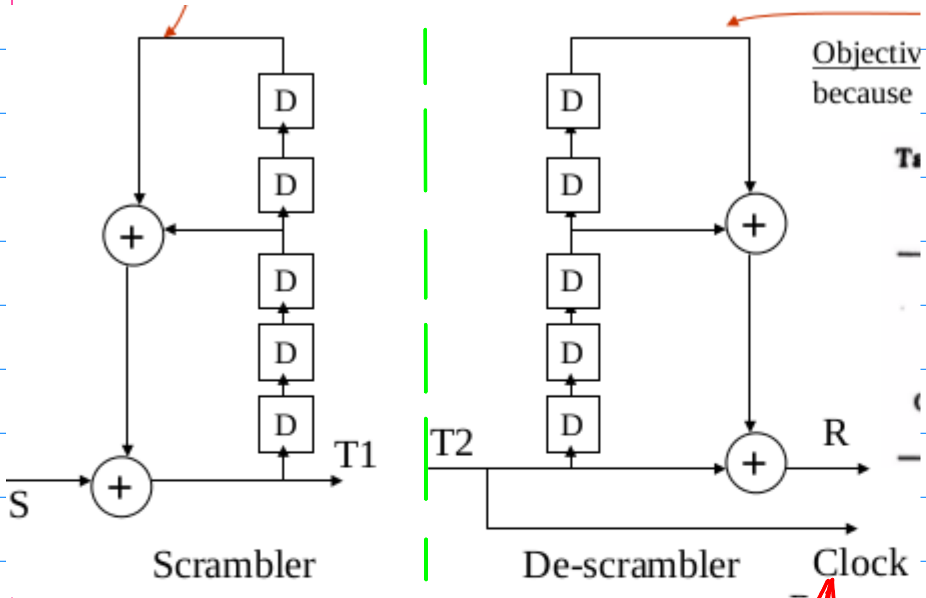
Reference: Di

Input S
Lower Bank D^4T.
Lower + Upper Bank D^7T
Output T



CmpE245
Sept. 21, 22

7



Note: The Theoretical Analysis and proof of the scrambling/Descrambling Technique Can be Accomplished By using Formal Math. Formulation from The Switching Theory.

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

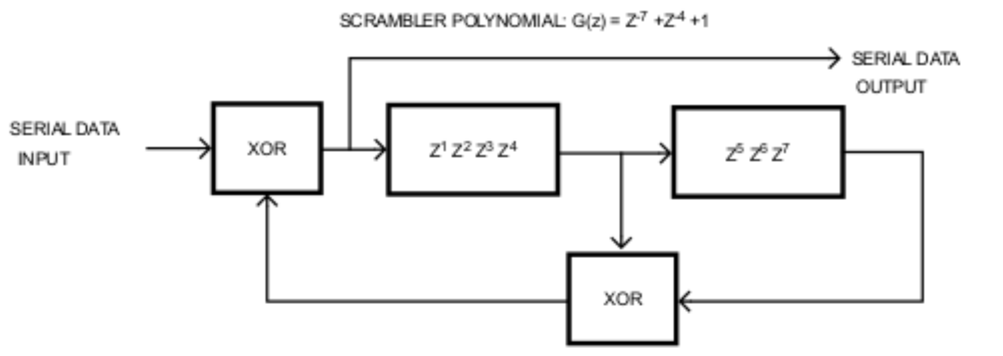


Figure 131—Data scrambler

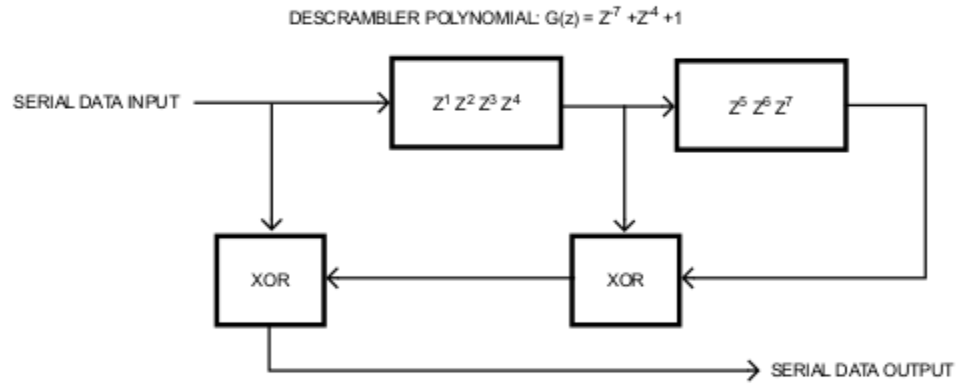


Figure 132—Data descrambler

CMPE258
Sept.21,22

Consider BaseBand 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.

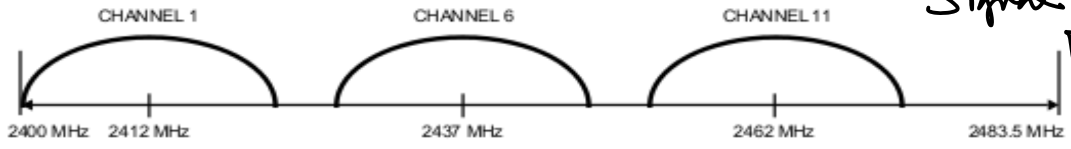


Figure 141—North American channel selection—non-overlapping

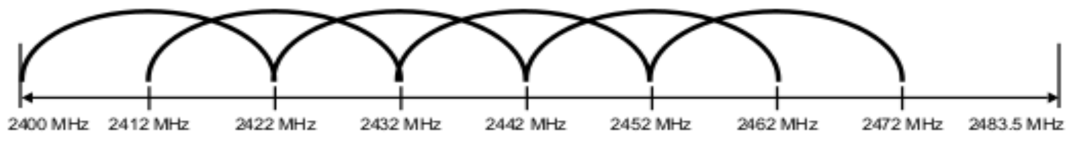


Figure 142—North American channel selection—overlapping

2. Fig145. a. fc: Carrier frequency for IEEE Wifi. fc ~ 2.4 GHz → 11 Channels. then 11 fc

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.

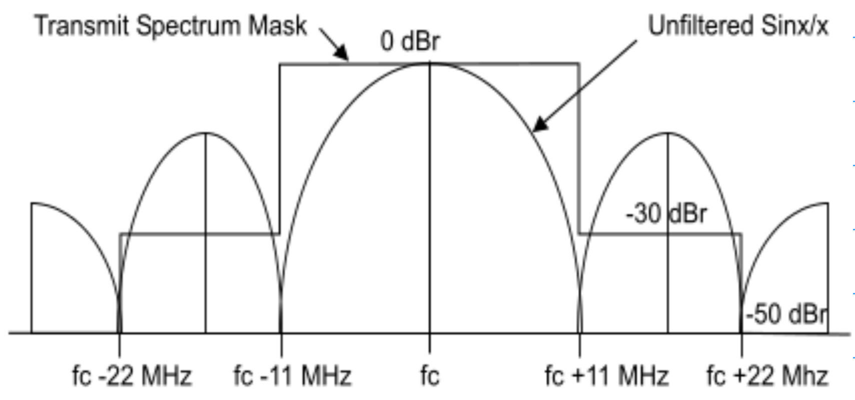
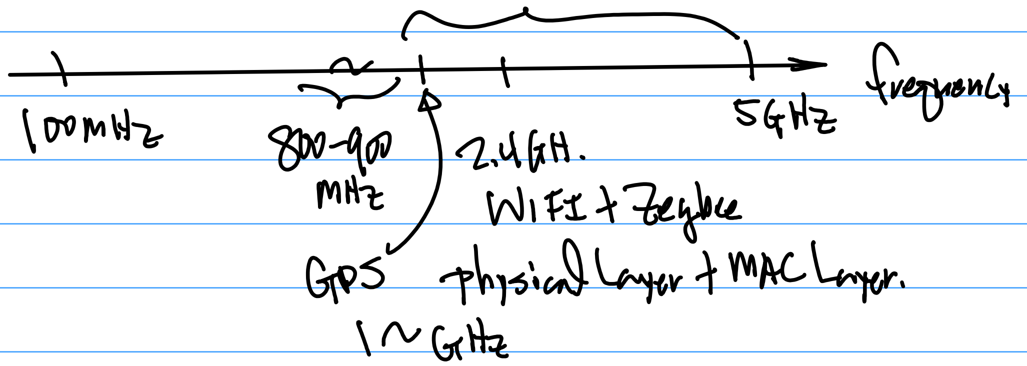
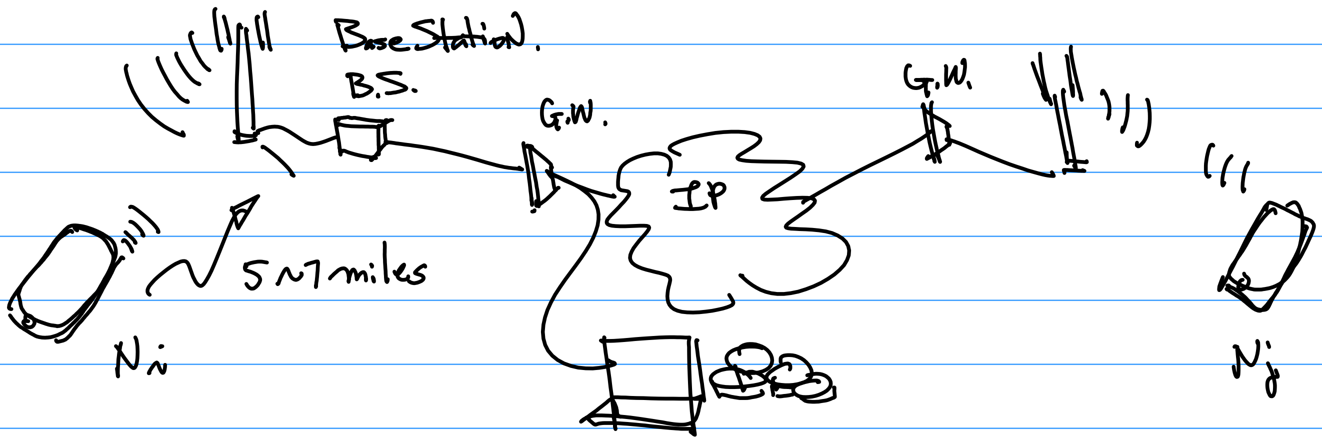


Figure 145—Transmit spectrum mask pp.60

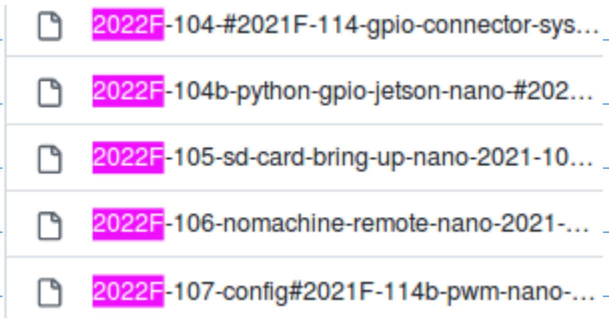


Physical Layer + MAC Layer.

CMPE245
Sept.21,22



Sept.26. Monday.
Note: 1. Check Homework on CANVAS. LISA on the target platform with R.F. Board.
2. Optional Target Platform, NVDA Jetson NANO, 5-6 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

Step1. Background, Pin Assignment. J41. Two Pins for GPIO. (Input, Output Each) 10

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.

Sysfs GPIO	Name	Pin	Pin	Name
	3.3 VDC *Power*	1	2	5.0 VDC *Power*
	I2C_2_SDA *I2C Bus 1*	3	4	5.0 VDC *Power*
	I2C_2_SCL *I2C Bus 1*	5	6	GND
gpio216	AUDIO_MCLK	7	8	UART_2_TX */dev/ttyTHS1*
	GND	9	10	UART_2_RX */dev/ttyTHS1*
gpio50	UART_2_RTS	11	12	I2S_4_SCLK
gpio14	SPI_2_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

a. Pin 12

	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	I2C_1_SCL *I2C Bus 0*	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZ0	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_SDOUT	gpio78

b. Pin 40.

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

SD Card. 32GB.

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

/ 2022F-105-sd-card-bring-up-nano-2021-10-28.pdf

Write Image to MicroSD Card

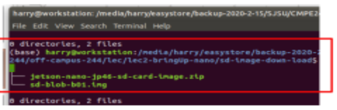
NVIDIA

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. Down load 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:

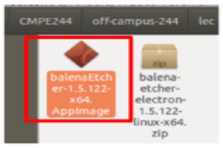
https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write



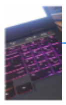
```
harry@workstation: /media/harry/easystore/backup-2020-2-15/SJSU/CMPE2
File  Edit  View  Search  Terminal  Help
0 directories, 2 files
(base) harry@workstation:/media/harry/easystore/backup-2020-2
244/off-campus-244/lec/lec2-bringUp-nano/sd-image-down-load$
├── jetson-nano-jp46-sd-card-image.zip
└── sd-blob-b01.img

0 directories, 2 files
```

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

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


CMPE244
off-campus-244
lec
balenaEtcher-1.5.122-x64.AppImage
balena-etcher-electron-1.5.122-linux-x64.zip

Use USB card r
your host mach
start it to write



The program '
10-15 minutes
then it will vali

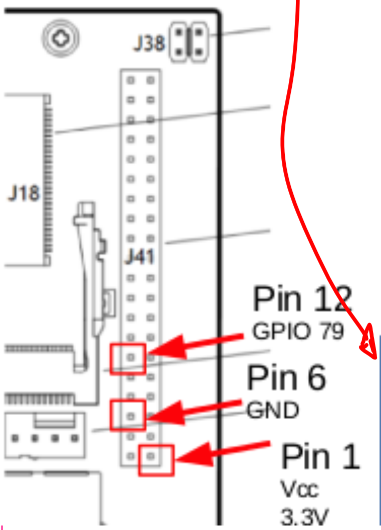
Harry Li, Ph.D.

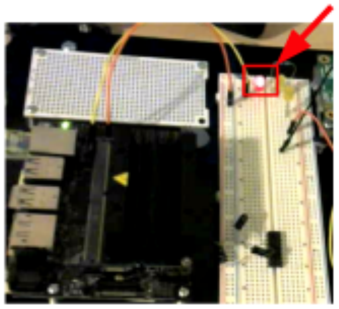
Step3. Test GPIO Input/Output
Ground By Command Line
Instruction.
See Ref on the
github, 2022F_104~

CmpE245
Sept. 26, 22

Command Line Information.

Testing J41 40 Pin Connect





We can control our LED

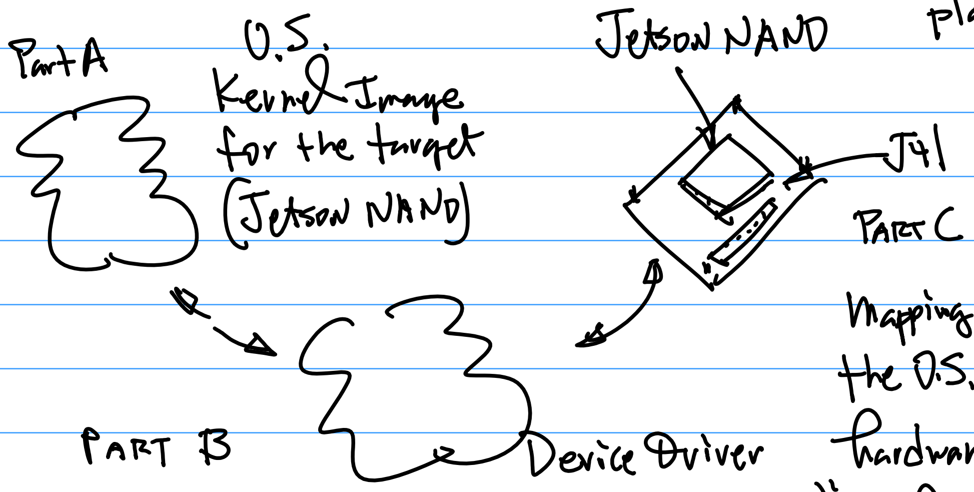
```
# Map GPIO Pin
# gpio79 is pin 12
$ echo 79 > /sys/c
# Set Direction
$ echo out > /sys/
# Bit Bangin'!
$ echo 1 > /sys/cl
$ echo 0 > /sys/cl
# Unmap GPIO Pin
$ echo 79 > /sys/c
# Query Status
$ cat /sys/kernel/
```

In the above code, the
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.



Mapping of Device Driver(s) from the O.S. Kernel to the target hardware is done by "Configuration" process.

CMPE245

Sept. 26, 22

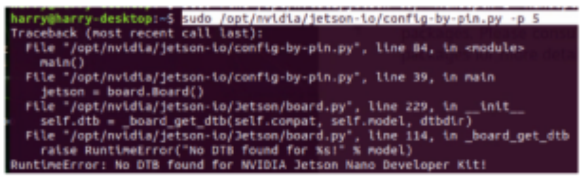
Note: Run Configuration Python Code if Pins Added Note By Factory Default.
Recommended To use Configuration Mapping.

Step 1. Fix bugs from the distribution

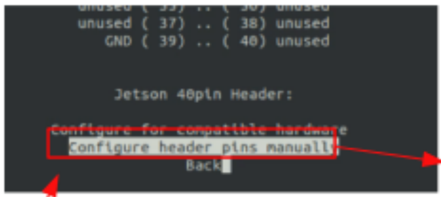
Configuration of Pins with jetson-io.py

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

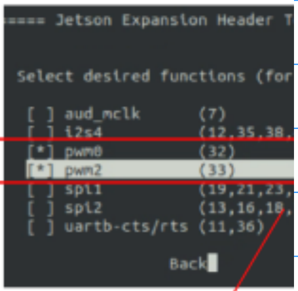
$sudo /opt/nvidia/jetson-io/config-by-pin.py -p 5



```
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!
```



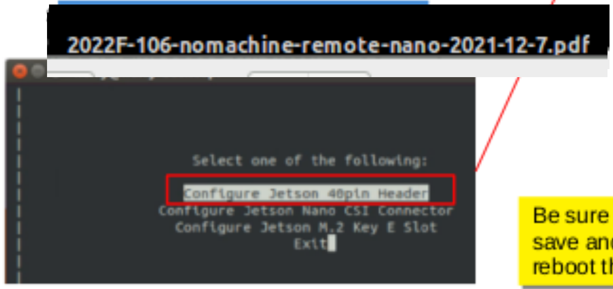
```
   unused ( 37) .. ( 38) unused
      GND ( 39) .. ( 40) unused

          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,
  [ ] uartb-cts/rts  (11,36)

                Back
```

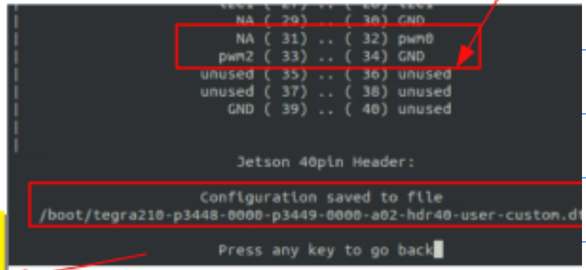
$sudo mkdir -p /boot/dtb
$ ls /boot/*.dtb | xargs -I{} sudo ln -s {} /boot/dtb/

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
```

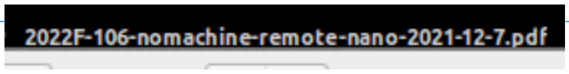


```
       NA ( 29) .. ( 30) GND
       NA ( 31) .. ( 32) pwm0
     pwm2 ( 33) .. ( 34) GND
   unused ( 35) .. ( 36) unused
   unused ( 37) .. ( 38) unused
      GND ( 39) .. ( 40) unused

          Jetson 40pin Header:

        Configuration saved to file
 /boot/tegra210-p3448-0000-p3449-0000-a02-hdr40-user-custom.dt

          Press any key to go back
```

Be sure to choose save and reboot to reboot the system

Harry Li, Ph.D

Step6. Setup Remote Access for the purpose of using your laptop KeyBoard, mouse, And Display.



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

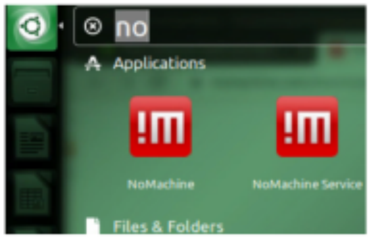
Nomachine for Jetson NANO

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

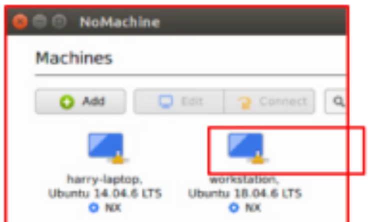
Aarch64 version 7.74_1; size: 42.29 MB, type: TAR.GZ

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

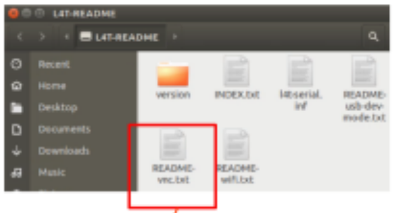


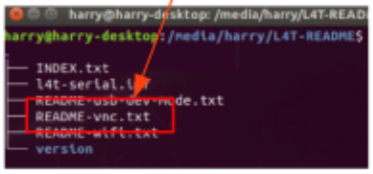
Once you start nomachine on your laptop, and enter the right user name and password of the nano, you can see it now



NoMachine
Machines
Add
Edit
Connect
harry-laptop, Ubuntu 14.04.6 LTS
NX
workstation, Ubuntu 18.04.6 LTS
NX

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



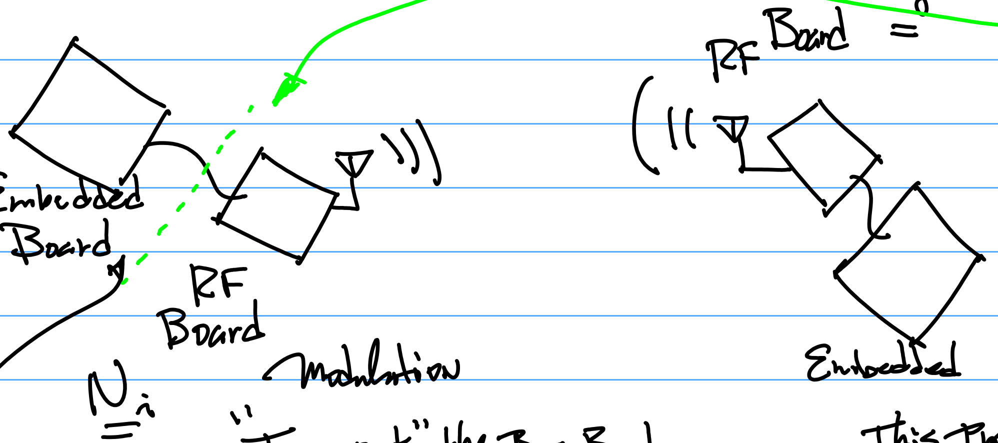


```
harry@harry-desktop: /media/harry/L4T-READ
harry@harry-desktop:/media/harry/L4T-README$
.
├── INDEX.txt
├── l4t-serial.inf
├── README-usb-dev-mode.txt
├── README-vnc.txt
├── README-wifi.txt
└── version
```

CMPE245
Sept. 26, 22

Continued for the Base Band Signal Discussion.

Example: Communication B/W N_i & N_j
"SJSU_CMPE245" To Be Sent. → ASCII → "0's & "1", Base Band Signal



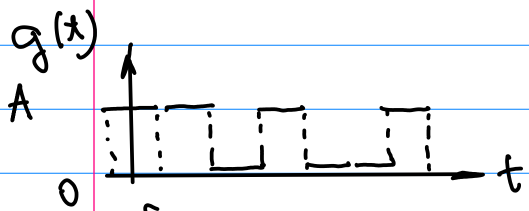
"Transport" the Base Band Signal to a proper frequency Range for Desired Characteristics.

This Process (modulation) will → Satisfies 80% Energy distributi Requirements. By IEEE/FCC.

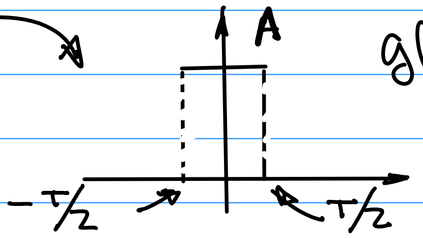
a. The waveform "shape" of the Base Band Signal is well designed to meet the 80% energy Requirement.

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.



$$g(t) = \begin{cases} A & t \in [t_0 + T/2, t_0 - T/2] \\ 0 & o/w \end{cases} \quad \ldots (1)$$



$g(t - kT)$, for $k = 0, 1, 2, \ldots$

For A sequence of BB:

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

Define Fourier Transform

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

CmpE 245
Sept. 26, 22

$$\begin{tikzcd} g(t) \arrow[r, leftrightarrow] & \mathcal{F}[g(t)] \end{tikzcd}$$

$$\mathcal{F}[g(t)] = A\tau \frac{\sin \pi f \tau}{\pi f \tau} \quad \cdots (3)$$