CmpE245

August 25 (Wed)

Introduction.

Today's Topics
- Bill of materials
- Target platforms For Prototyping

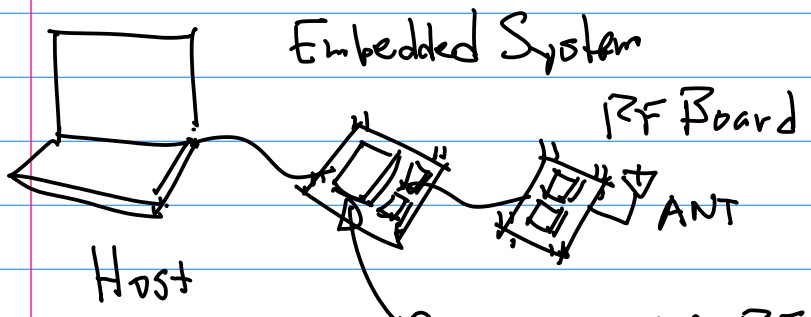


Fig. 1.

Model:

Bill of Material

1. Target CPU NXP LPC1769 OR NANO (NVDA)

2. RF modules, physical Layer only.

(2) ASK R.F. module, (1) frequency ~433 mHz

Amplitude Switching

(3) F.C.C. certified Open Spectrum.

Power ≤ 1000 mW. (1W)

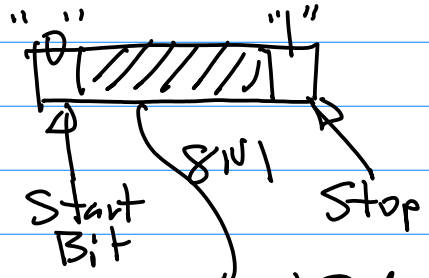
Tx: Transmitter

(4) No MAC (media Access Control) Needed

Note: Your RF module(s) is to be interfaced to GPIO of your target Board.

Why? General Purpose → No Data Framing.

for UART, Data Frame



Note: CPU Target

Option to use NVDA Jetson NANO

Guide Line for selection of target platform.

1. Register Level Control of GPIO, SPI Controllers

For LPC1769 ✓

For NANO → Devices Drivers.

(1) Datasheet 4 root pages

(2) Software Dev. Environment, tools.

Jetpack (OS + Libs)

(3) O.S. Distr.

August 25 (Wed)

2° Access to CPU pins, e.g.
GPIO pins, SPI pins.

Homework: Purchase ASK RF module By Sept. 8th (Wed) OR ideally Sept 3rd (Fri)

Note: To provide hardware Debugging Ckt. on the RF. Board

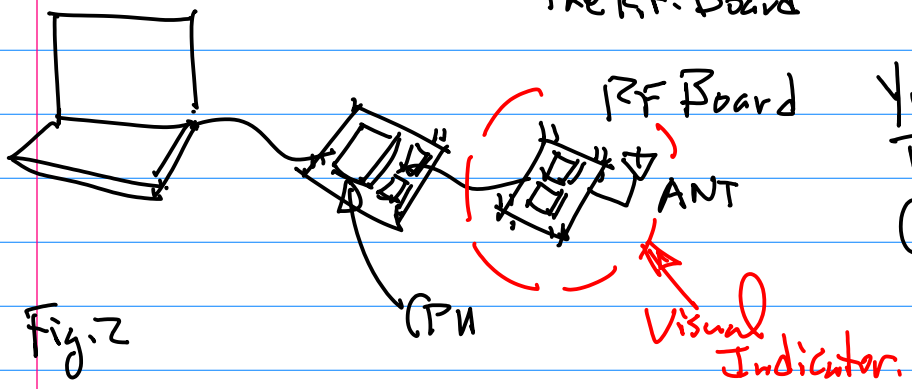


Fig. 2

Debugging Capability on the RF. Board:

① Objective: To visualize/observe GPIO Output.

Means: LED.

material:

a. LED (Red, Green), 4 ~ 10 mA

b. Connectors (to cable to RF Board)
RJ45 Right Angle Connectors (2)

A piece of CAT 5 Cable (Ethernet)

c. Components { Resistors, Caps. (1117?)

d. External PWR Regulator (7805, 7812 OR

Note: On your RF Board.

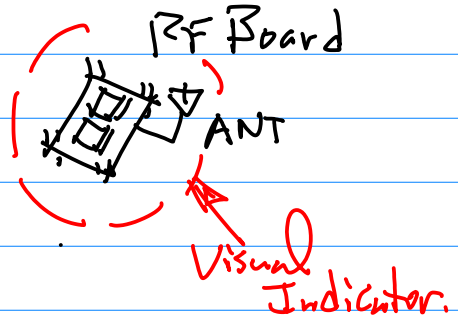


Fig 3.

2 Blocks { RX, TX

Both Need to Be Powered.

You may want to have the DC PWR Delivered via CAT 5 Cable from your Embedded Board.

RJ45 POS (Pin): 8

August 30th.

RF. module, to Build RF Board.

1° ASK RF.
Amplitude Switch Keying

2° RX: Receiving
TX: Transmission

3° GPIO I/F
Data Pin

Homework: Identify/Bring Your Wire Wrapping Board for RFB Design. 4x3 Inch;

To Build RFB

1° Board

2° 4 standoffs

3° Build I/O I/F Testing CKT.

- To Light up LED when CPU output "1"
- To Turn off LED when CPU output "0"

Output Testing

- S/W Toggles to Vcc, when Vcc, CPU Reads as "1"
- S/W " " to GND, CPU Reads as "0"

Input Testing

4° CAT-5 Ethernet Cable

RJ45 Right Angle Connectors (2)

One for Embedded;

One for RF Board,

5. PWR Distribution to RF Board:

a = 5VDC is adequate But R.F. module can be operated with Bigger Power, 9VDC OR 7.5VDC may be needed during Debugging;

Sept 1. (Wed)

Ref: IEEE802.6

1. github/hualili/cmpe245/2018F "2021F-"

2. Topics: Design R.F. Board for the 1st homework. 2pts. (Hardware)

Example:

From pp.2. System View

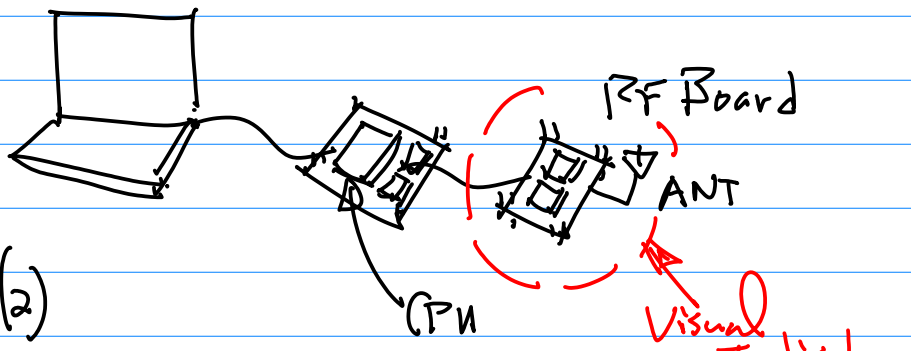
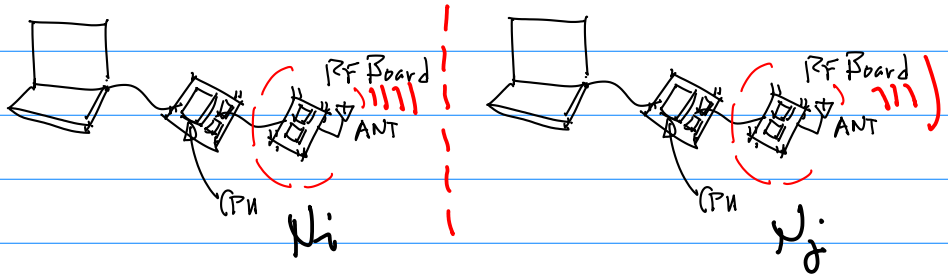


Fig.1 Node i, Wi



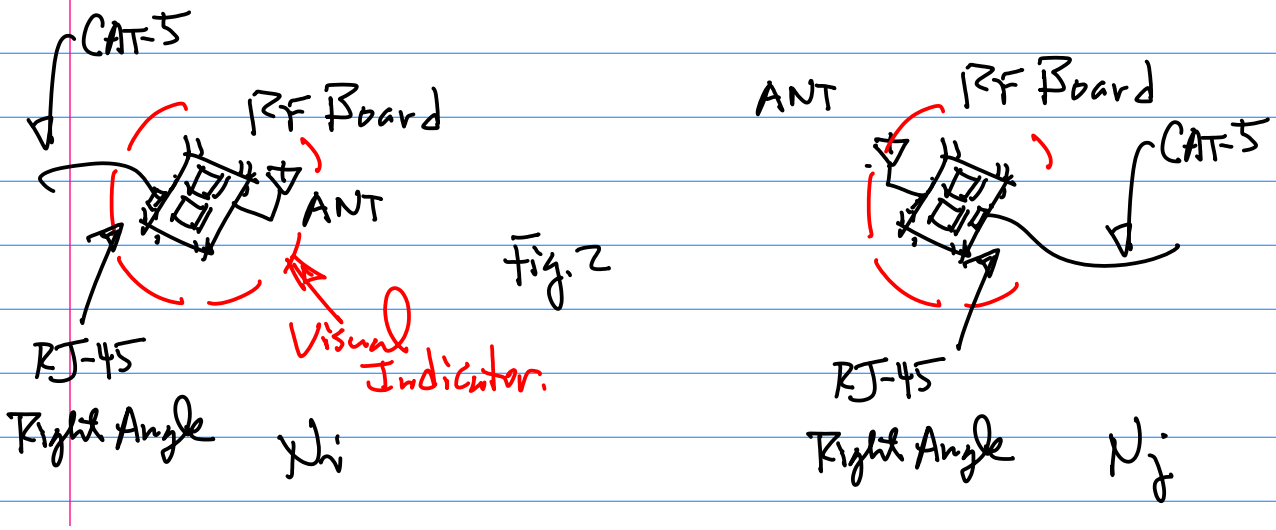
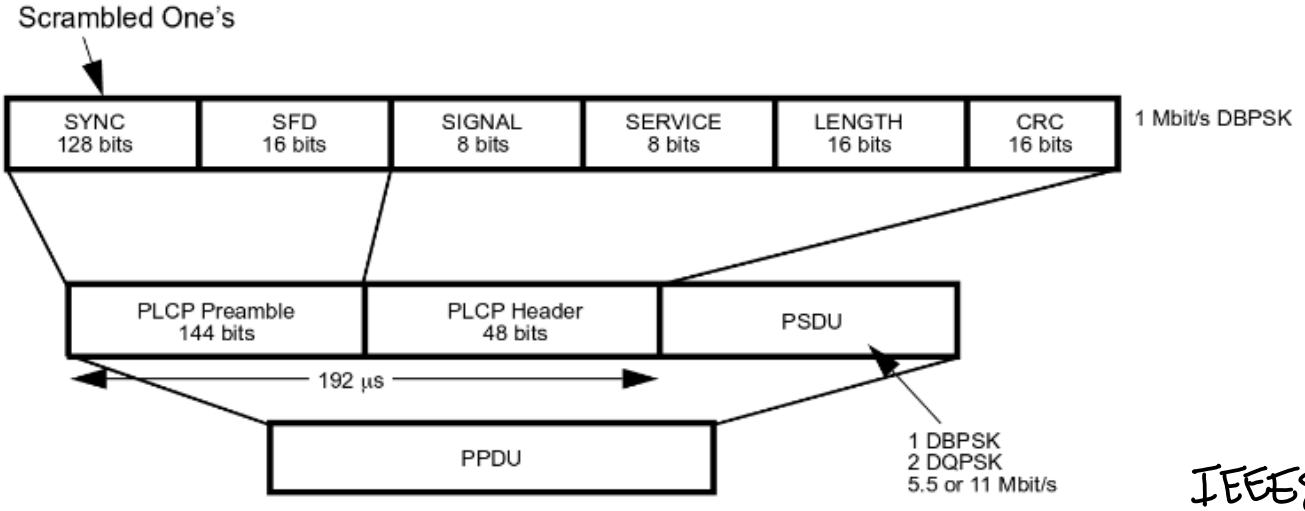


Fig. 2



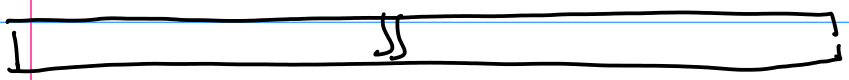
IEEE802.6

Figure 127—Long PLCP PPDU format

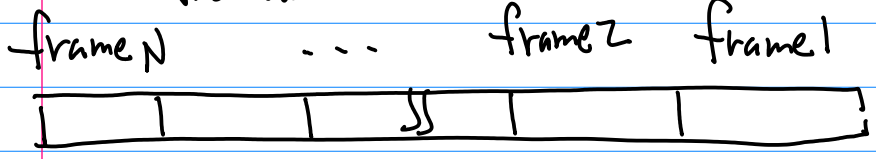
Fig. 3. pp. 13

Node Ni, Node Nj, Each Node is illustrated in Fig 1.

Sending Stream of Data from Ni to Nj



String of Bits sent in A "Framed" fashion.



Note: Consider the construction of Each frame, frame i, i = 1, 2, ..., N;

What are the contents to be placed into A Frame?

like start bit(s) → to mark the Beginning of the Communication.

→ Timing/Sync.

C.R. (Cognitive Radio), SDR (Software Defined Radio)

multiple Bits in Sync. Design → more Reliable → "Robust" performance
Maybe

Question: What is the general guideline in terms of designing "Sync" Field?

multiple Bits of what?

3 bits Example: b2 b1 b0

$2^3 = 8$

0 0 0
0 0 1
0 1 0
⋮
1 1 1

Transition → Change of State

"Zero Crossing"

for Example

Fig 5.

To provide best possible "Transitions", e.g. "ZC"

Pick 0x5 Binary: 101

The Objective for R.F. Board Homework/Design:
To Allow Ni, Nj to Sync.

2-Step Approach:
Step 1. Based on Land Line;
Step 2. then R.F (Wireless)

Hence, R.F. Board will have to Support 2 Options.

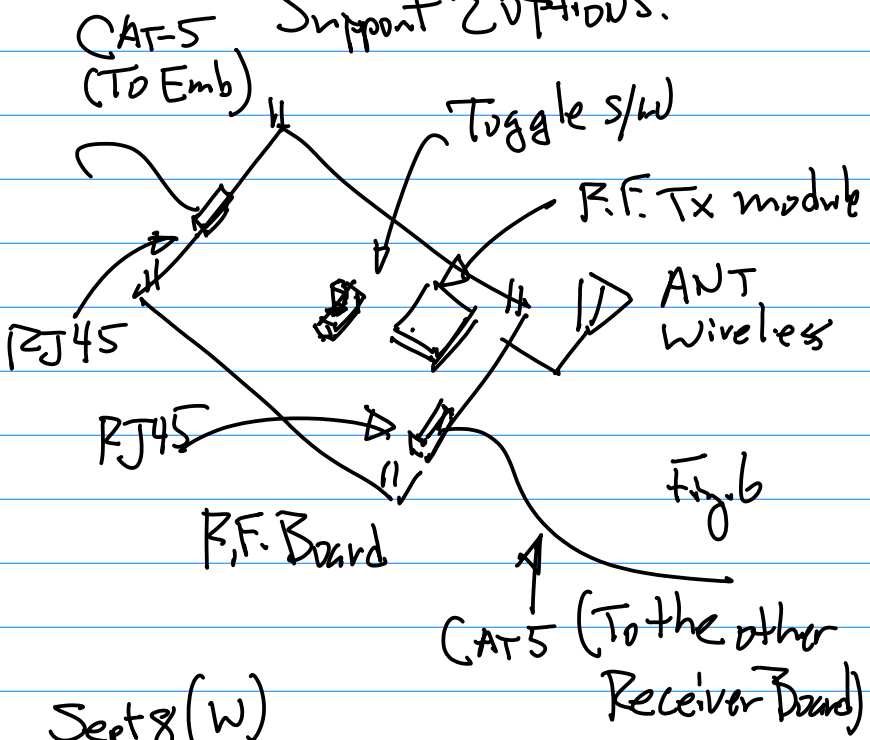


Fig. 6

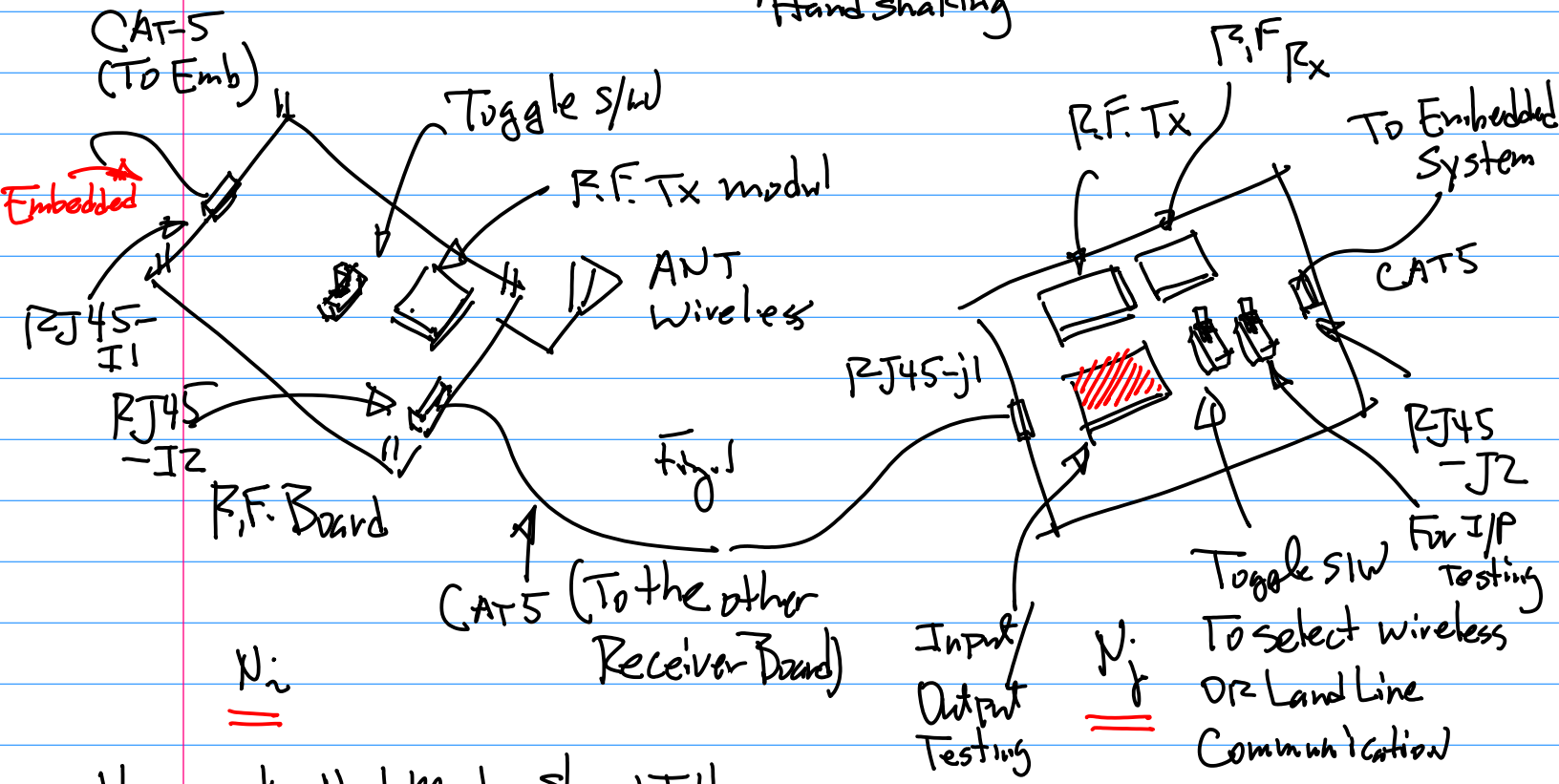
Sept 8 (W)

Topics: 1° Prototype for Handshaking (for LPC1769, NANO, OR your choice, such as Pic)

Note: NANO Boards are not delivered yet. So, please find your solution.

Note 2. Semester Long Project, please form 4 person team;

Example: "LAND-Line" Based Design for Handshaking



Homework: Next Monday, Show + Tell for Each Person's R.F. Prototype Board.

Things to show for the next class: 1. 4x3" Prototype Board; the Bare Through-Hole; metal coating of the through-Holes only; 2. Stand-offs 3. RJ45 Connectors (2x), Right Angle; Mounted on the Board;

4. Testing Circuit
   - Output Testing
     - Output "1" from the Host/Embedd System
     - Output "0" .. ..
   - Input Testing
     - Input "0" originated on your R.F. Board
       - ↓ RJ45-J1
       - ↓ RJ45-I2
     - Input "1" .. .. ..

Continued,

Input Testing { Input "0" originated on your RF Board
Input "1", ... (Similar to Input "0")

RJ45-J1
↓
RJ45-I2
↓
RJ45-I1 → Embedded System/Host PC Laptop

Testing CKT:

Output Testing CKT → Red LED.

Embedded → RFB (N_i) → LED1 → RJ45-I2
via RJ45-I1
RFB (N_i)

RJ45-I2 → CAT5 ⋯⋯ RJ45-J1
RFB (N_j)
RJ45-J1 → LED2 → RJ45-J2 → Embedded

On RFB (N_i & N_j)

GPIO Output from LPC/NAND

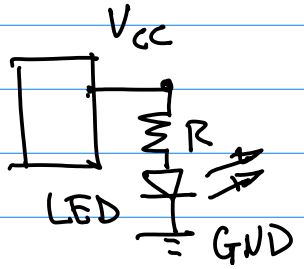


Fig. 2

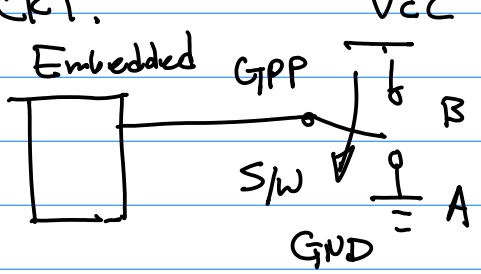
$V_{CC} = IR + V_{LED}$ ... (1)

V_{CC} (CMOS) = 3.3VDC

V_{LED} = 1.8V DC ; I = 10mA ;

Solve for R ~ 250 ± Ω or higher ;

Input Testing CKT.



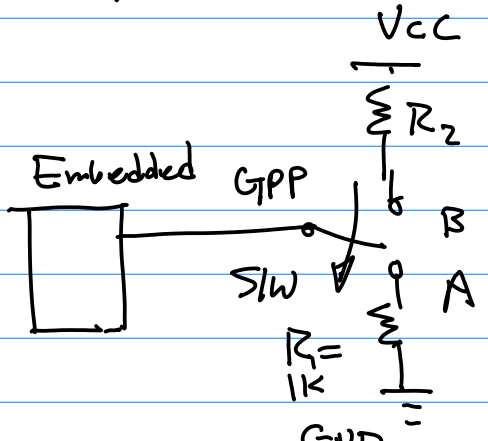


Fig. 3a

Sept13 (Monday)

Today's Topics: 1° Hardware platform.

2° Sync Algorithm, LISA

(Linear Invariant Sync. Algorithm)

Ref: 1° ... github, ID: ~106...

2021F-106

for Target RISC-V

(System On Chip)

2° github/hualili/Cmpe245/...

2021F-105~ (NAND Connector)

FPGA Solution (RISC-V), Iglu2.

a Board from Future Electronics

b IP-Core for CPU in Verilog

Can be down Loaded;

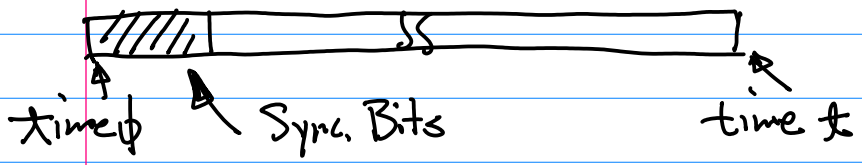
c TAP Plastics

NAND Board. Connector J-41

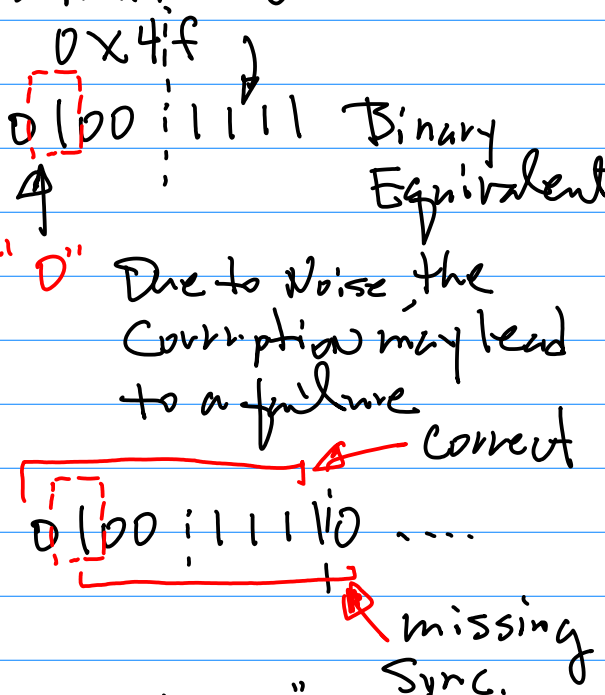
Identify GPIO

Example: Let's consider the Design of Sync. Field in Wireless Communication

Consider 1° Place to define sync. Bits.



Suppose we have a letter in a Hex Number



"1001 : 1110"

2° Sync is established Based on "2C"

The change of the State of a Signal, For Example

"0" → "1" OR "1" → "0"

Preferred Change is to include "1".

Requirements:

1° Establish Sync with Change of State

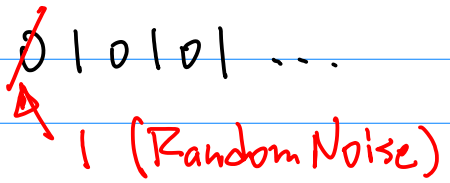
01010101 ...

OR

101010...

{ 01010101 ... OR ... (1a)
{ 101010... ... (1b)

Consider Random Disturbances, alters one bit in the Sync field. (1a)



Suppose we know the sync. field Consists of 10 Bits. → Discard the 1st 2 bits until we have 0101...

Pattern again, to satisfy the total Number of bits in the Sync, Definition.

Question: How to utilize the Sync Pattern even when this pattern is corrupted, And not to start over again?

(1) The Need to Re-use the Sync Field Even if it is corrupted;

(2) The Need to Known where is the end of the Sync Field, e.g. the Beginning of the payload.

Sept 15 (W)

Homework: Due Next Wed. Sept. 22nd.

Official (2 pts)

1° R.F Board Prototype (finish this Prototype for Landline I/F).

2° Run a simple program C/C++ or Python to perform Testing of Input/Output function.

Input Testing: GPIO Reads Input "0" AND/OR "1", Note Toggle the Switch on the RF Board to produce "0" & "1".

Output Testing: LED ON when Output = "1"
LED Off " " = "0"

What to Submit:

1° One Description of your design implementation;

2° photo of the System Set up
Host Laptop + Embedded Board + RF LPC/NANO Board

3° Source (soft Copy) LPC1769
Export your Work as a project
NANO, Python or C/C++ Code

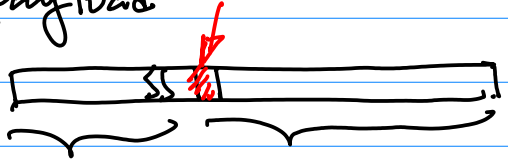
4° Video Clip 5~10 Sec.

~~ON CANVAS or~~ Submit the Zip file via E-mail;

Consider Sync. Field Design:

Design Requirements:

1° Re-use the k-bit Sync to Be able to identify the start bit of the payload



K bits Sync. Field payload

2° To Establish Timing
Even if Random disturbances Corrupted some bits in Sync. Field, in addition to pin point to where the Corruption occurs.

Embedding ID Index into Sync Field.

From 10101010 ....
Change it to the following.
e.g. Take 8 bits
10101010 at a time

Taking the 1st 4 bits of this Segment (1 byte), preserve its pattern
prefix: 1010
Then, Taking the 2nd 4 bits modify it to make it as an index to Reflect this Byte position in the Sync Field. e.g.

0 000 ... "0"
0 001 ... "1"
0 010 ... "2"
0 011 ... "3"

0 0 0 0 ... "0"
0 0 0 1 ... "1"
0 0 1 0 ... "2"
0 0 1 1 ... "3"
...

1 1 1 1 ... "f"

Assemble prefix and ID together

1 0 1 0 ; 0 0 0 0
1 0 1 0 ; 0 0 0 1
⋮
1 0 1 0 ; 1 1 1 1

16 Bytes
a0 ; a1 ; a2 ; ... ; af ... (1)

Now, Change 1010 to 0101 = 0x5
therefore,

50 ; 51 ; 52 ... ; 5f ... (2)
16 Bytes

Integrate (1) & (2) together,

a0, a1, a2, ..., af, 50, 51, ..., 5f

e.g.
0xa0, 0xa1, ..., ..., 0x5f
... (3)
32 Bytes (256 bits)

Design An Algorithm for Sync. Extraction @ Rx End

Question: What is the minimum Number of Bits in (3) do we need to establish Sync.?

Remark 1: Minimum 8 consecutive Bits with first 4 bits from alternating Bit Patter, e.g. 0xA, or 0x5 is Needed to Establish Sync. Given

Question: Sync. Based 8 bits V.S. Sync. Based on 16 bits, which one gives higher Confidence? Ans: 16 bits.

Question: What is the number bits that gives the highest confidence level?
256 bit.

Let's define Confidence level η as follows,

$$\eta = \left(\begin{array}{c}\text{Sync. Establish} \\ \text{with K bits}\end{array}\right) \Big/ \left(\begin{array}{c}\text{Total No. of} \\ \text{bit in Sync.} \\ \text{Field}\end{array}\right)$$

... (4)

Sept. 20 (Mon)

Topics: 1° LISA Conclusion

2° Base Band Signals with Definition modulation Techniques, ASK, FSK, PSK.

3° Project 1. Assignment

Software Defined Radio Implementation of LISA.

Due Oct. 8th (Fri), 11:59am.

Written Requirements to Be Posted on git & SJSU CANVAS.

Official (10pts)

a. Land Line "LISA" (Based on Homework of Input/Output Testing)

b. RF. "LISA".

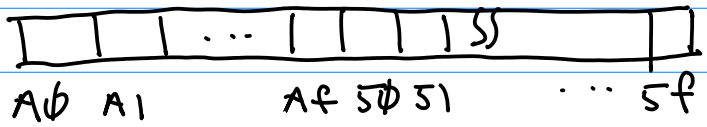
ASK RF Rx & Tx for the implementation.

Example: Compute "n" (confidency Level, CL) - for the sync established Based 8 bits data from the Sync. Field.

Sol. From Eqn (4),

$$n = \frac{8}{\text{Total bits in the sync Field}} = \frac{8}{256} = \frac{2^3}{2^8} = 2^{-5} = \frac{1}{32}$$

Fig. 1



Linear: Index Arranged in a Linear fashion;

Invariant: Capture 8 bit sync pattern regardless of its/their position(s);

Conclusion: LISA algorithm Provides Robust, Versatile Sync. Scheme to Allow N_i, N_j to Establish sync. (Handshaking).

Homework: Write C/C++ (OR python) for your hardware platform, to Realize Base Line "LISA". (Due Sept. 27 monday)

a. Prompt the user for his/her input for the Number of Bytes to establish Sync.

1 Byte, 2 Bytes, ..., 32 Bytes

b. Implementation for "the Base Line"

BaseLine: Consecutive Bytes.

C. Creates A Test Pattern as follows, "testsync.txt"

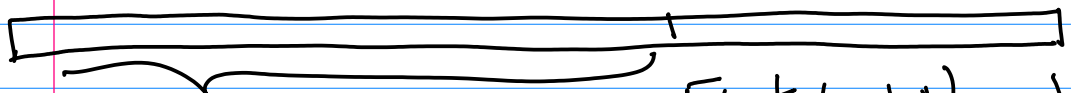


Fig. 2

plain text file.

d. Run your program to process the testsyn.txt, to extract Sync. By Printing the payload.

Note, then future homework will allow the Bytes from Non-Consecutive order.

Note: Optional Requirements

Pseudo Code.

What to Submit:

a. Source Code (Indicate if on Linux, or Windows, or LPC, or NANO)

b. Binary Executable;

C. Optional, Pseudo Code.

Base Band Signal:

without

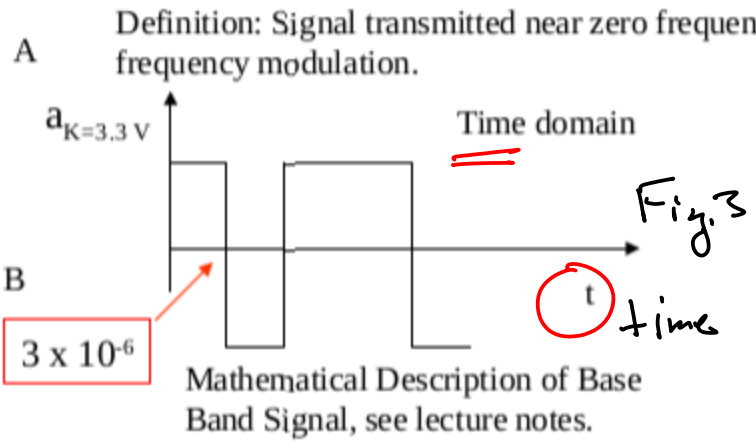


Fig. 3

Signals from GPIO, PWM, etc.

for Example.

Suppose the Signal is periodic Signal. Concentrate on a Single Bit.

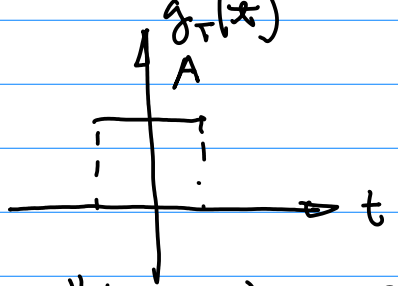


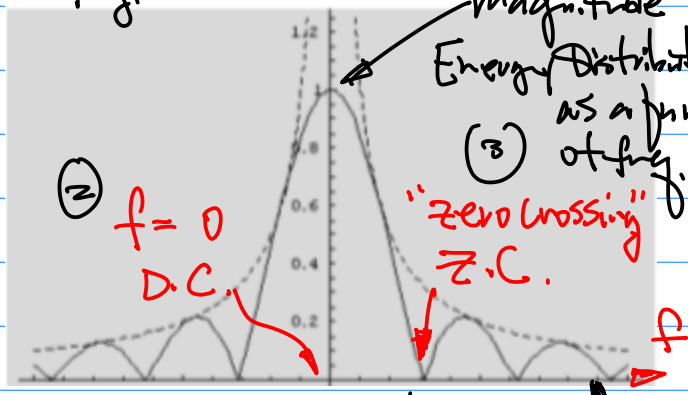
Fig. 4

Subscript "T" in $g_T(t)$ for One bit

$$g_T(t) = \begin{cases} 1 & \text{for } t \in [-T/2, T/2] \\ 0 & \text{otherwise} \end{cases} \quad \cdots (1)$$

Frequency Characteristics of the B.B. Signal:

Fig.5



(5) magnitude Energy Distribution as a function of freq. index

(2) f = 0 D.C.

(3) "zero crossing" Z.C.

f

(4) Based Band Defined between 2 Z.C.s "1st zero Crossing": Bandwith.

(1) frequency Index f

Eye Pattern: Characterization of Base Band Signal

IEEE 802.11b Standard pp. 56