

August 21 (Monday)

Organizational Meeting.

1. The "GreenSheet" is posted on the github

Note: Bring your Laptop Computer to the Class.

<https://github.com/hualili/CMPE244>

## Course and Contact Information

Instructor:	Harry Li, Ph.D. Professor, Computer Engineering Dept., San Jose State University
Office Location:	Engineering Building 267A
Telephone:	(408) 924-4060 (650) 400-1116
Email:	hua.li@sjtu.edu
Class Days/Time:	Mondays and Wednesdays, 4:30 pm – 5:45 pm, August 21, 2023
Office Hours:	Mondays and Wednesdays, 3:00 pm – 4:00 pm
Classroom:	Engineering Building Room 295
Prerequisites:	CMPE 180A and CMPE 180D, classified standing, computer science majors or Artificial Intelligence or Computer Engineering or Software Engineering majors only.

2. Emphasis on Posix O.S. Linux OpenSource O.S. & Device Drivers Programming and Development. Scalability & Vertical Solution.

## Course Description

Experiments dealing with advanced embedded software programming concepts, interfacing techniques, hardware organization, and software development using embedded systems. Individual projects.

3. Course Format: In-Person.

Hands-on Class. Prototype System

Optional. NVIDIA Jetson Nano. GPU (128)

4 GB Version      CPU      JetPack

Option 2. Broadcom Piex3B+, Piex4.

Option 3. RISC-V FPGA Dev. Board + FreeRTOS

Selection Decision in 1 week

Option 4. NXP LPC11G24 or

LPC1768, RTOS. NXP

Dev. Forum.

has limited Processing Power.

May Not Meet the Need for Our Project

#### 4. Textbook & References

Set I: Datasheets(s), CPU Datasheet, Developer Guide; Set II: NVIDIA Developer Forum. Set III: PPTs, Sample Code, Handouts in the Class GitHub.

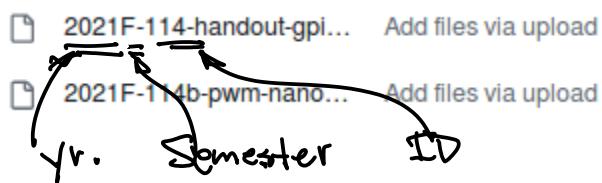
#### Course Materials

Instructor's teaching materials and online resources.

1. Professor's git: <https://github.com/hualili/CMPE244>
2. Jetson NANO Jetpack download <https://developer.nvidia.com/embedded/downloads>

#### Other Equipment / Material

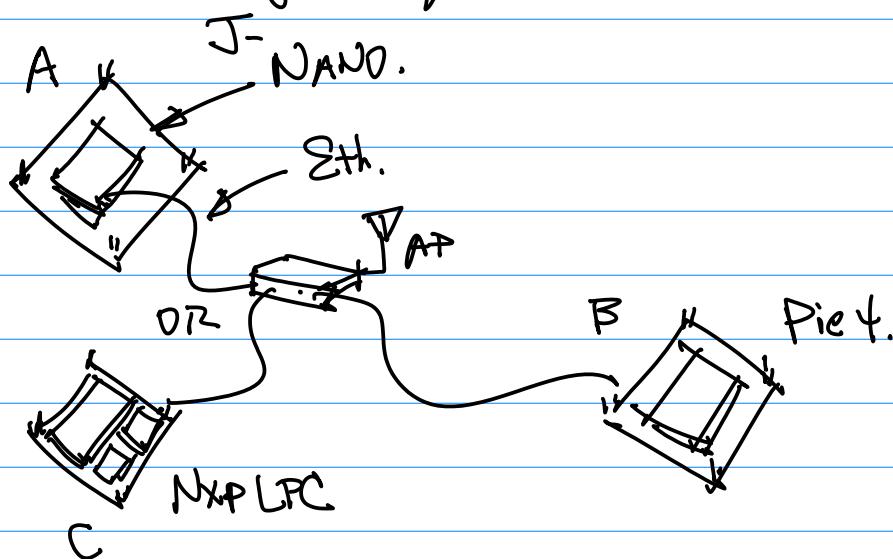
1. Hardware Equipment: You may choose any one of the following options. For detailed selection information, I will cover it in the introduction session of the class. Option 1. Nvidia Jetson NANO Board with minimum 2 GB RAM; or Option 2. Pi 3B+, or Pi 4; Option 3: Nvidia Jetson Tx2 developer kit; or Option 4: LPC1769 CPU Module: [https://www.mouser.com/NXP-Semiconductors/Embedded-Solutions/Engineering-Tools/Embedded-Processor-Development-Kits/Development-Boards-Kits-ARM/\\_/N-cxd2t2P=1z0jm4m&Keyword=LPC1769&FS=True&gclid=Cj0KCQjwqKuKBhCRAIIsACf4XuHyN8WfqtQ24WGgt0MdKd6n-k17c-YNz-r1hTcPt0ErdZN62jrM0mgaAtXZEALw\\_wcB](https://www.mouser.com/NXP-Semiconductors/Embedded-Solutions/Engineering-Tools/Embedded-Processor-Development-Kits/Development-Boards-Kits-ARM/_/N-cxd2t2P=1z0jm4m&Keyword=LPC1769&FS=True&gclid=Cj0KCQjwqKuKBhCRAIIsACf4XuHyN8WfqtQ24WGgt0MdKd6n-k17c-YNz-r1hTcPt0ErdZN62jrM0mgaAtXZEALw_wcB) or Option 5: Samsung ARM11 developer platform.
2. Linux Host Machine (Ubuntu 18.04).



Naming Convention:

A & B  
A & C

Note: Regarding the Selection of  
A Target Platform:



## 5. Grading Policy

Project Assignment (Two Projects)	
Phased	
Assignments and projects:	15% (pts) for the assigned projects.
Midterm Exam:	30%
Final Exam:	30%
Total:	40%
	100%

August 23rd (Wed)

Introduction

Note: Rm268

Ref:

Datasheets.



bcm



lpc



nvda



sam

Broadcom

Pi e

Linux O.S.

NXP

LPC1769

RTOS

IP Stack

Micro Web Server

Jetson

NAND.

JetPack O.S.

Linux (Ubuntu)

+ Additional  
Packages.

Samsung

ARM11

2021F-107-lpc-cpu-  
UM10360.pdf

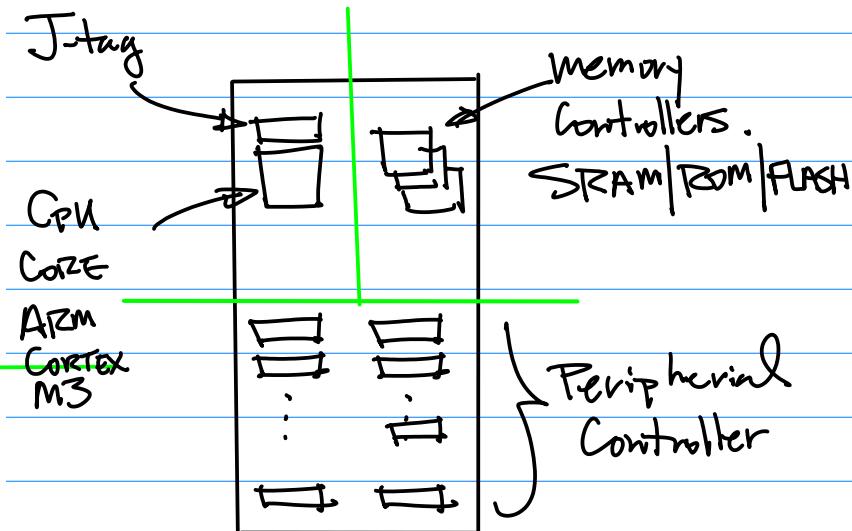
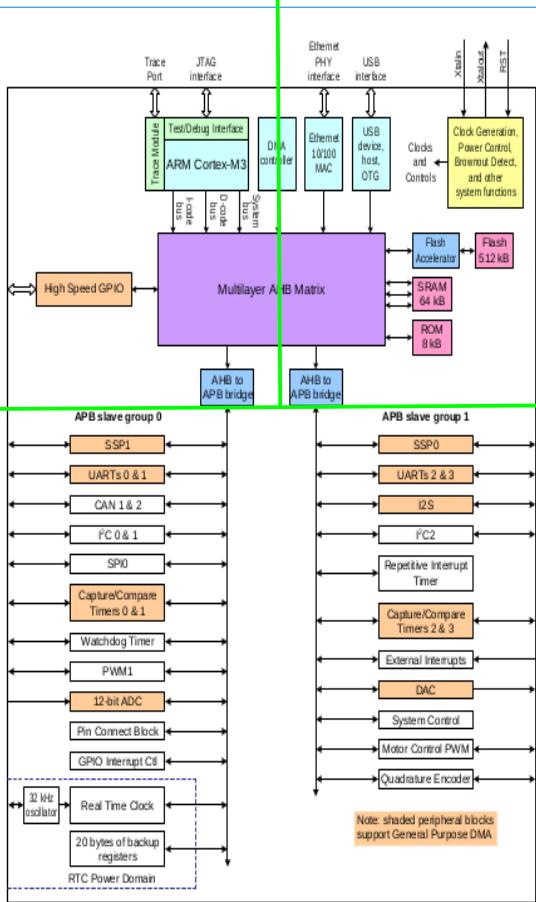
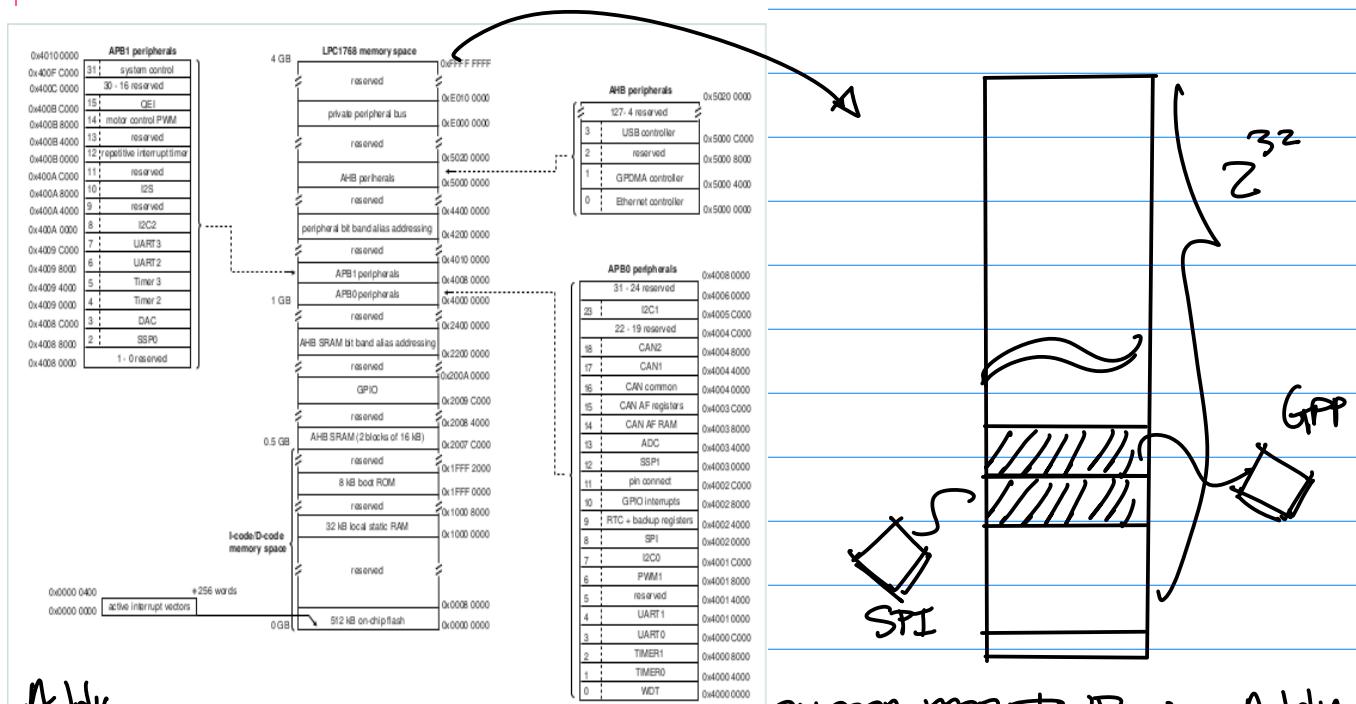


Fig. 1

Note: The CPU Block Diagram for LPC1769 is a Sample for the Rest of the target platforms.

e.g. Pic34 ; Sam's ARM 11 ;  
NVIDIA Jetson NANO

Note 2 :



Addrs.

$$Z^{32} = Z^0 \cdot Z^1 \cdot Z^2 \cdot Z^3$$

1024    :    1    4  
1K        :    1  
1Meg.     :    1  
1G.        :    1

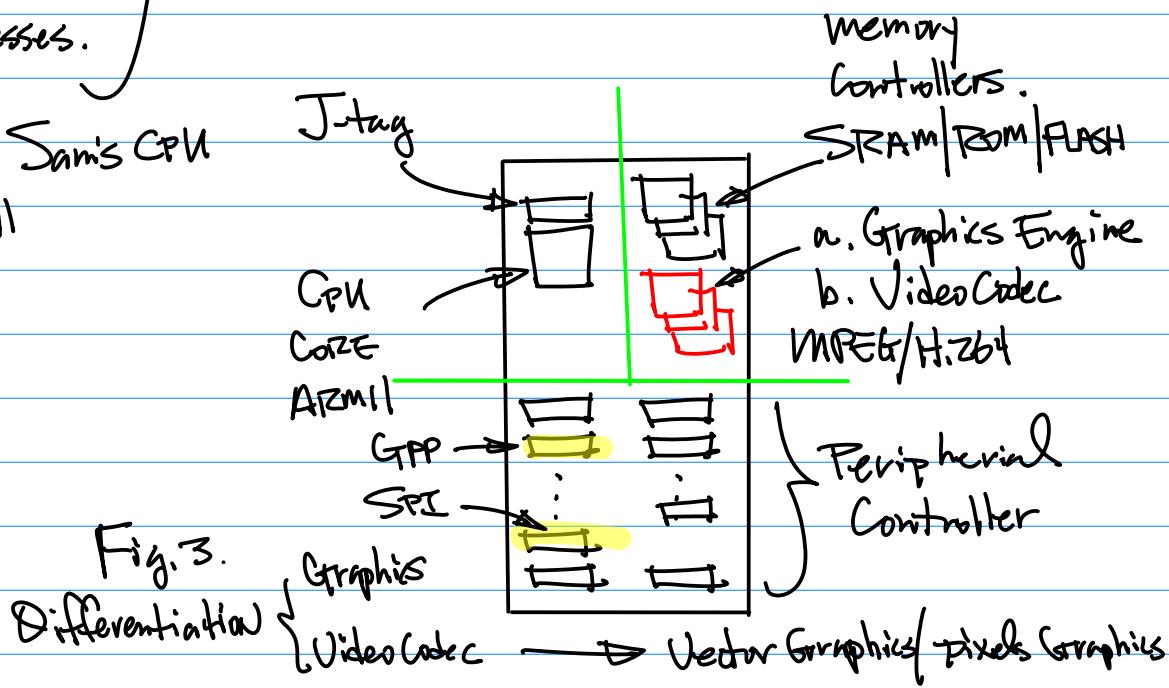
4 G Addressess.

Example: "B", Sam's CPU  
ARM11

Fig. 2

0X0000-0000 PWR-up Addr.

Datasheets.

2021F-105-#0-cpu-arm11-  
2018S-29-CPU\_S3C6410X.  
pdfLocate the page with the top level  
Description of the CPU Architecture

Example: Connection to (Embedded) Software Architecture

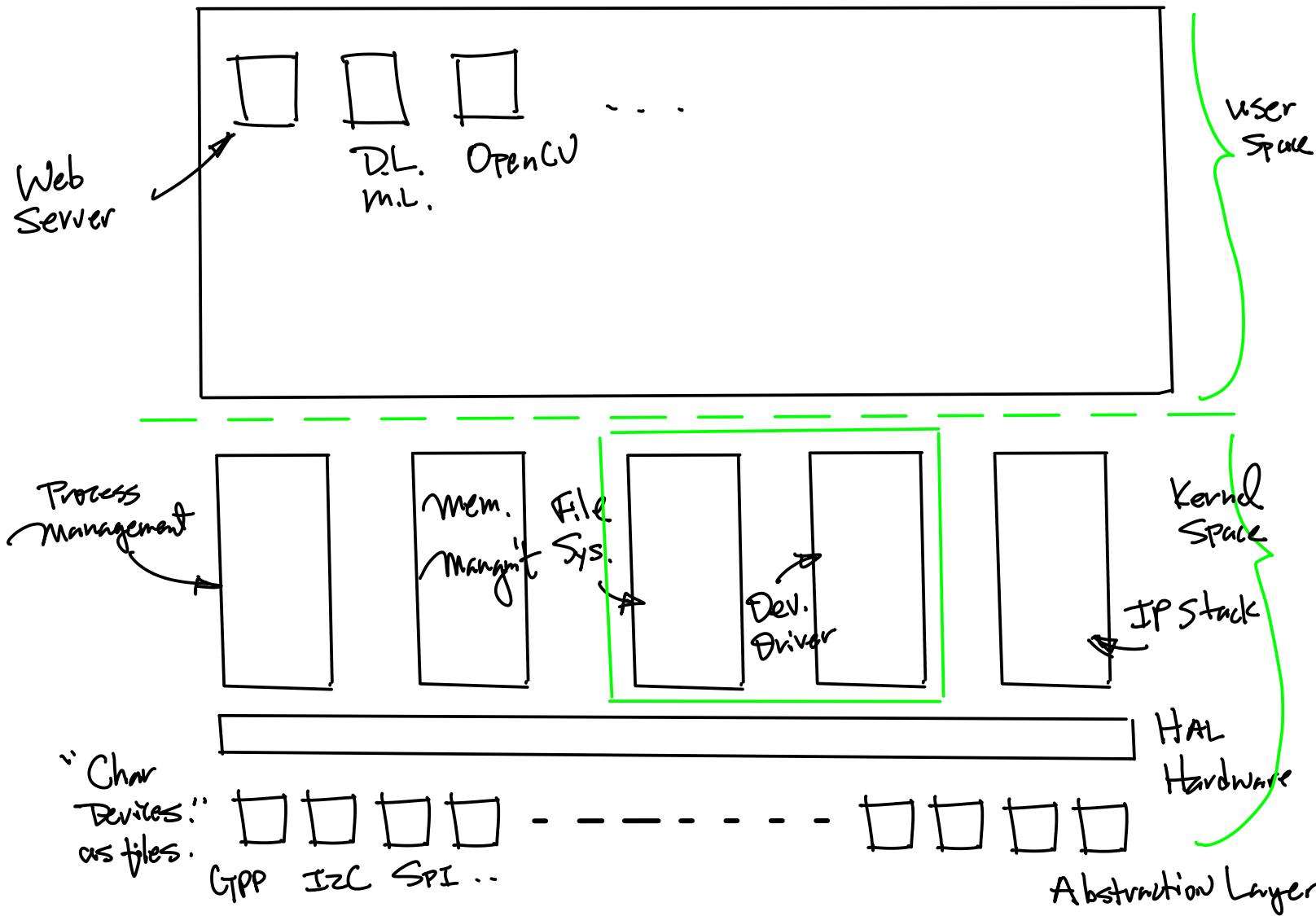


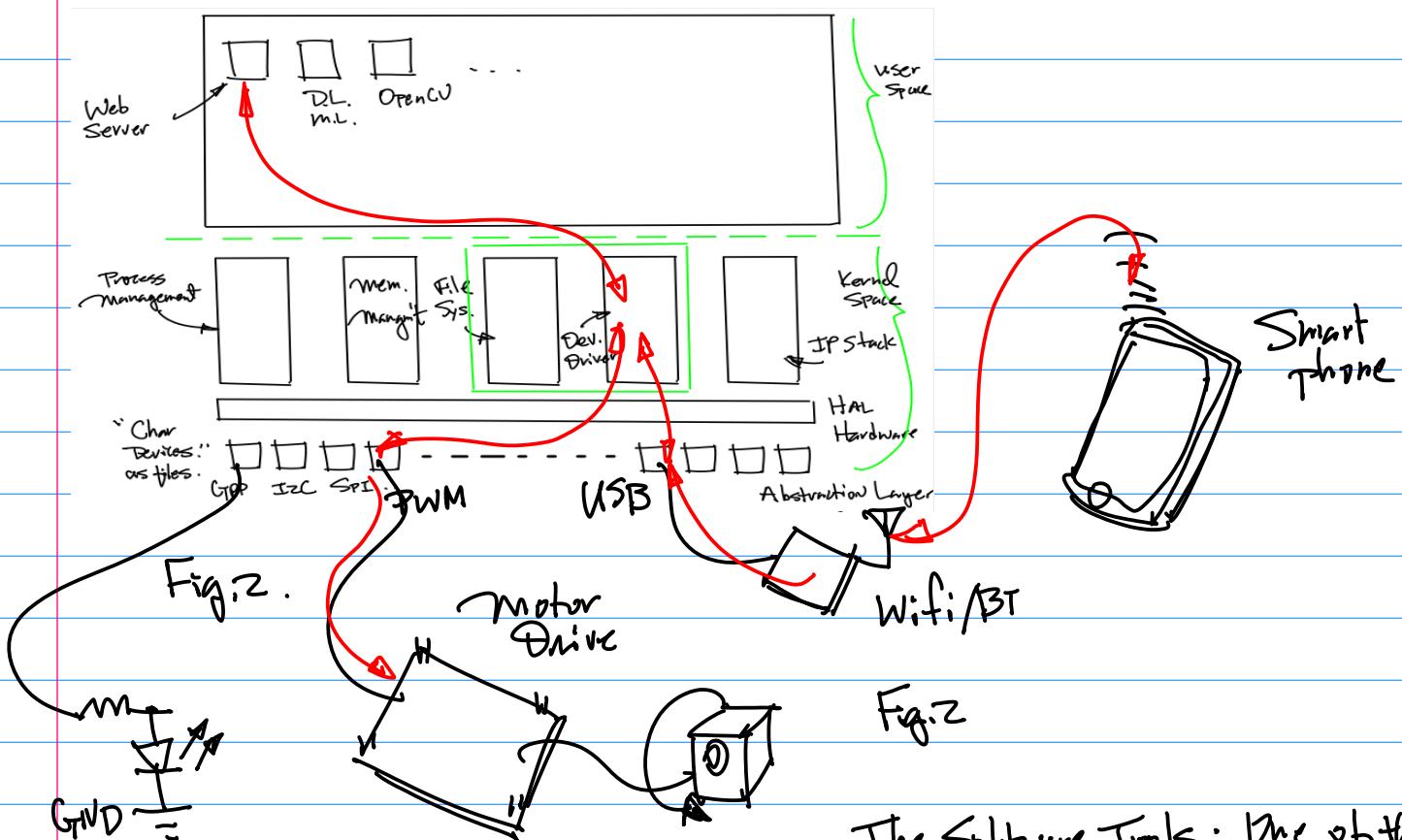
Fig.1.

Note: Data Size for 1080P  
Image OR 720P

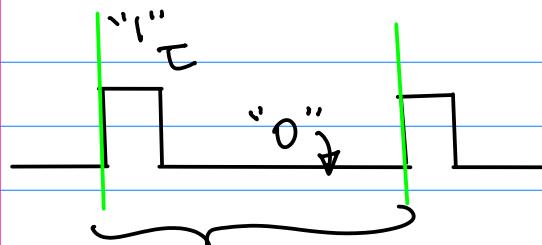
August 28 (Monday)  
Note: 1<sup>o</sup> Brief Description D.V.  
the Scope of Semester-Long  
Project.

- Embedded Software; Kernel D.S.
- Device Driver → APPS for iPhone/Android phone
- 2<sup>o</sup> CANVAS is up.
- Honesty pledge
- 3<sup>o</sup> Target platform → Minor upgrade to Enable RTC by Adding On-Board Battery

Example: Continuation of the Introduction/Embedded Software Architecture.



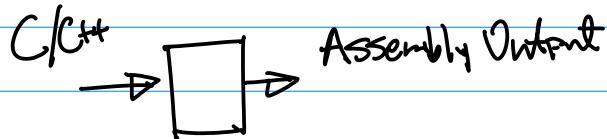
Note: PWM — Pulsewidth Modulation.



$T$  One Period

$$\left\{ \begin{array}{l} \text{Duty Cycle} = \frac{T}{f_{\text{PWM}}} \dots (1) \\ f_{\text{PWM}} \dots (2) \end{array} \right.$$

The Software Tools: One of them is open source gcc, or g++ Compiler.



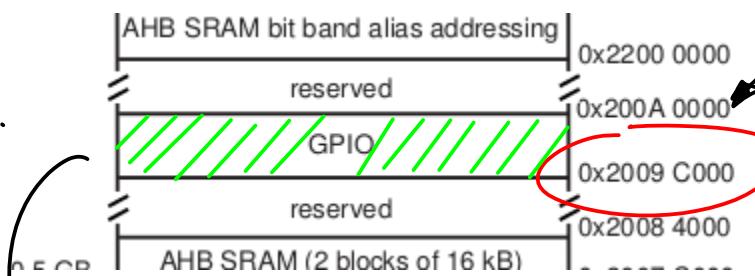
Porting. → Match to the Core  
(ISA: Instruction Set Architecture)  
Device Drivers Customization.

✓ Peripheral Controller  
A Set of Special Purpose Registers.

Most Likely this SPR has  
the addv in the Block.

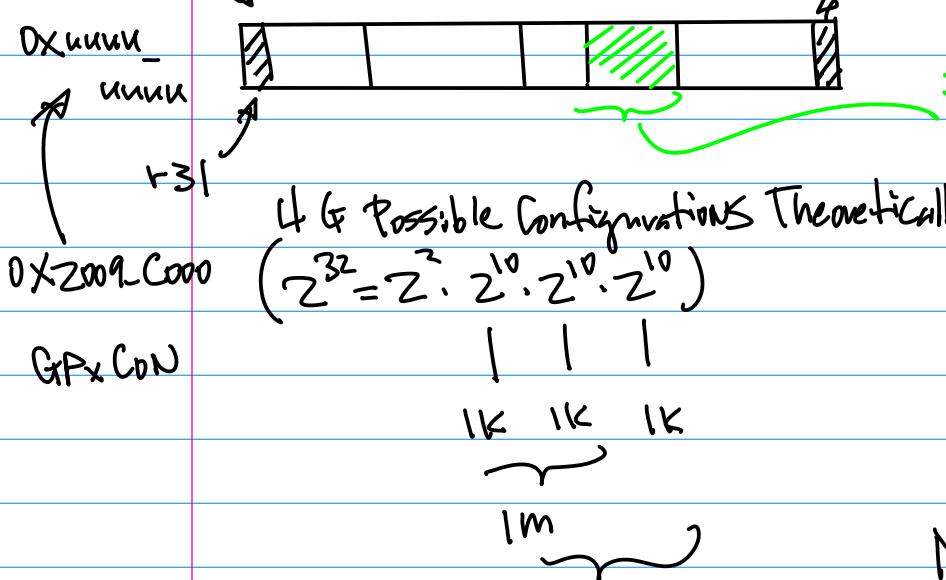
Identify A peripheral controller, GPP

Fig. 3.



→ Memory is dedicated for  
SPR's (Special Purpose Registers)

## Control & Configuration Register



It has its unique Address. (at the multiple of 4).

## Ext. Naming Convention.

Guideline :

Timeline:  
RISC → UC Berkeley David Patterson  
Stanford, John Hennessy

August 30 (Wed)

Note: 1<sup>o</sup> CANVAS is up.

Honesty Pledge to Be Signed  
And Submit on CANVAS  
By this Friday 11:59 pm.

2<sup>o</sup> Please Bring the target platform to the Class. Next Wednesday.

3<sup>o</sup> (Written Requirements) <sup>in 2 weeks</sup>

Bring up your target platform  
By Downloading Kernel OS.

Image to A micro-SD Card,  
then boot the System.

then Screen Capture with your personal identifier, Submit it on CANVAS.

4<sup>o</sup> Create A ChatGPT Account, Python interface to ChatGPT (3.5) API.

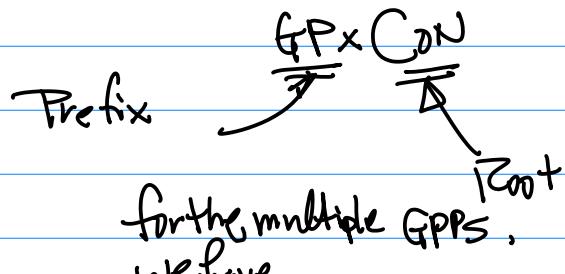
Example: Continued.

Naming Convention of the Control & Configuration Register.

By John Hennessy . Golden Rules

Uniformity, "3+3"  
Regularity,  
Orthogonality Naming Convention

Prefix + Root  
3 letters 3 letters.

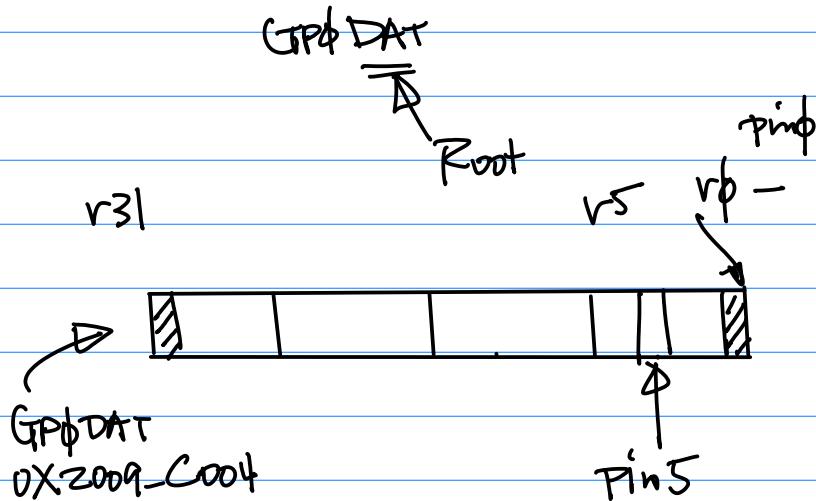


for the multiple GPPS,  
we have

GPPCON, GPICON, etc.

Suppose we want to use GP0 pin 5 as an output to turn on/off LED.

Design 2nd SPR.



place "1" @ R5 to Output logical high.  
"0" ..  
Low.

#define GP0CON  
GP0 DAT

Turing Turing  
gcc/g++  $\rightarrow$  ARM  $\rightarrow$  CORRECT

# Porting .f Board

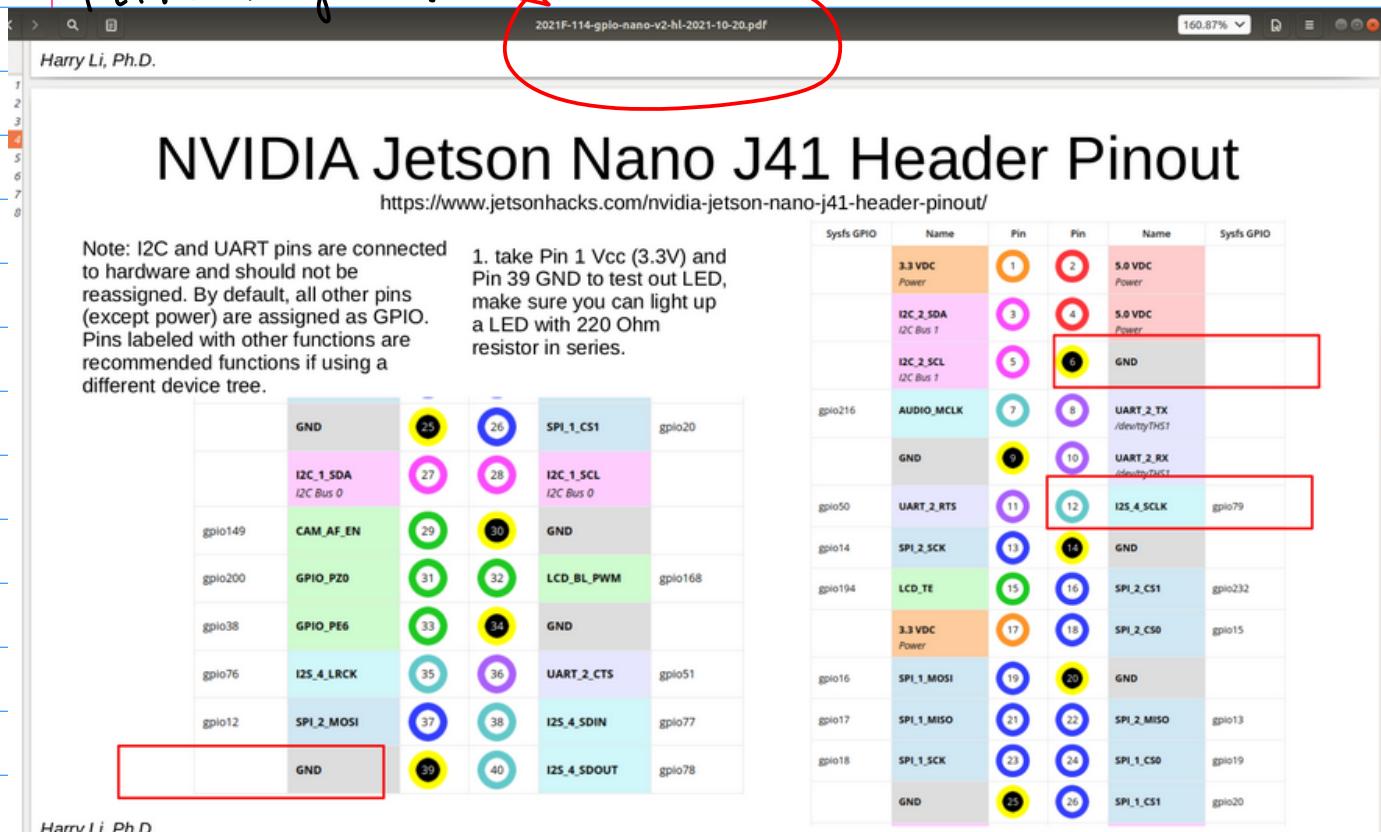
Sept 6 (Wed).

Note: 1° Target Board

## Inspection :

Purpose: J41 Connector

RTC Battery  
Ref: on the github. 2021-F-114 ~



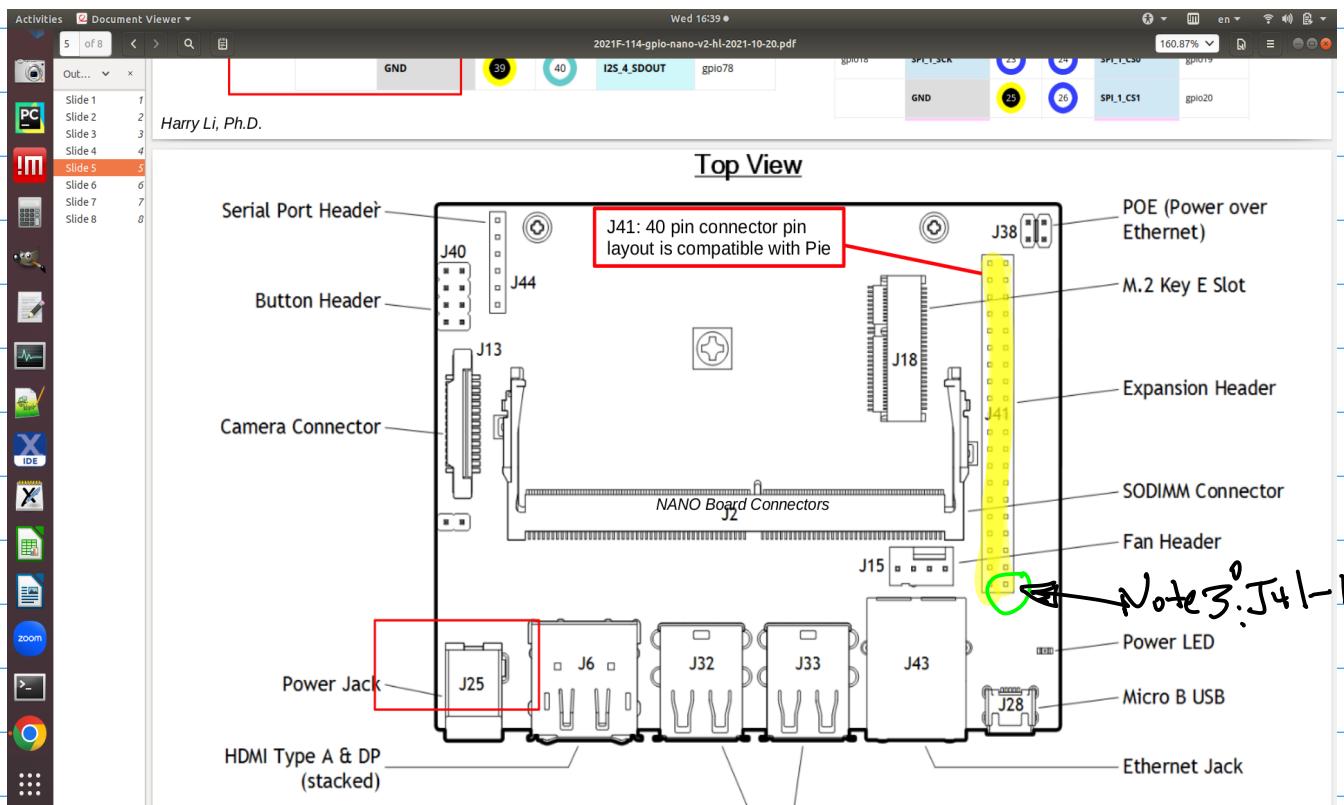
Note: 1° Power Pins

GND: b19/z5/zsa  
 Vout 3.3VDC/5VDC  
 Pin 1 Pin 2,4.  
 Vin: J25 (5A or higher  
 @ 5VDC)

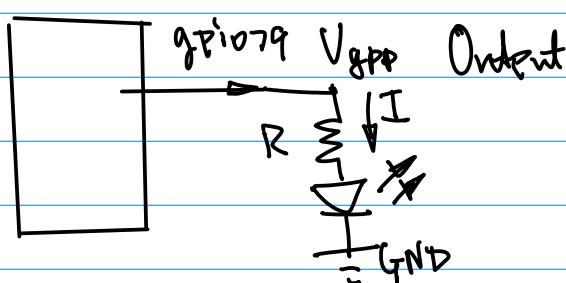
## Notez vos GPIO

J41 Pins  
J41-12  
J41-40

CPU Functionality



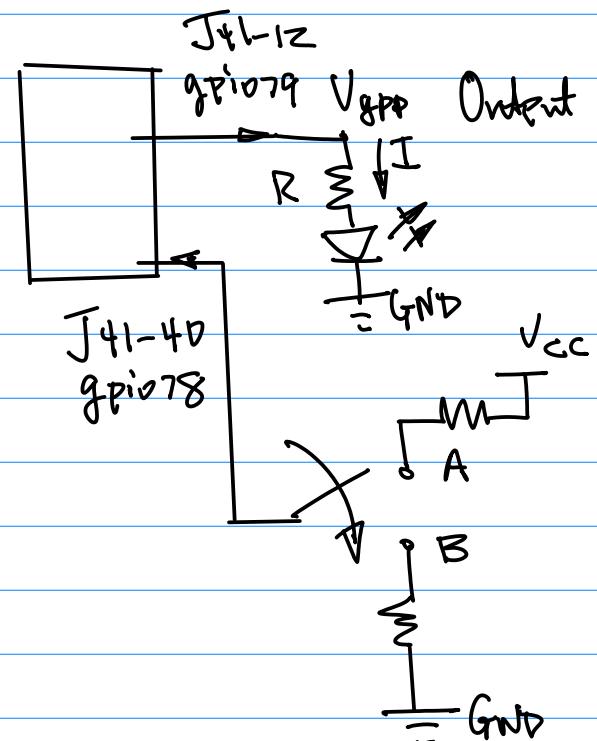
Note 4<sup>o</sup>: GPIO Input/Output Testing CKT  
Build the following Testing CKT.



Let  $I = 4 \text{ mA}$ ,  $V_{LED} \approx 1.8 \text{ V}$

$$V_{gpp,H} = IR + V_{LED} \dots (1)$$

w/o Resistor With Proper Selected LED.

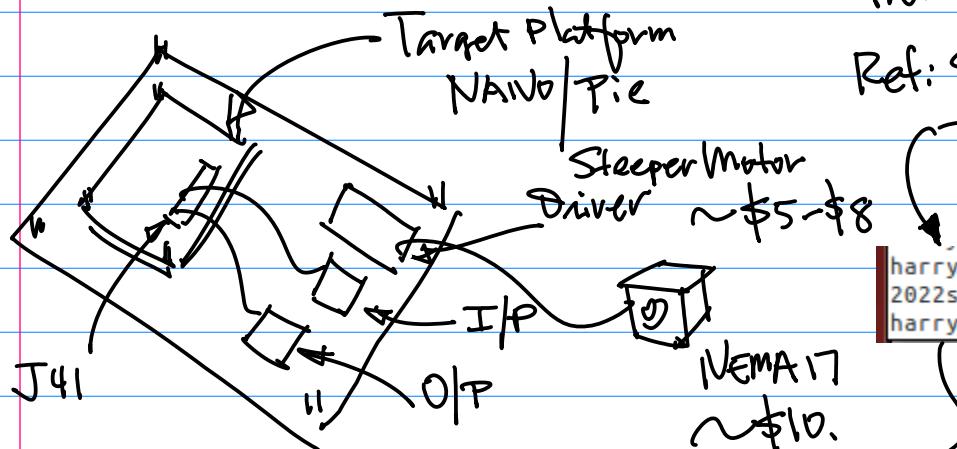


GPIO J41 Pins	CPU Functionality	Note
J41-12	gpio79	Output
J41-40	gpio78	Input

User Space Code  
Kernel Space Code

Take a Reference Design  
from Arm11, Samsung CPU.

Ref: Sample code has been  
Posted on the github.



```
harry@harry-laptop:/opt/FriendlyARM/min
2022s-104d-userSpace-gpio.c led led.c
harry@harry-laptop:/opt/FriendlyARM/min
```

### CMPE242-Embedded-Systems- / 2022S / 2022S-104d-userSpace-gpio.c

hualili Add files via upload

Code	Blame	37 lines (30 loc) · 642 Bytes	Code 55% faster with Git
------	-------	-------------------------------	--------------------------

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/ioctl.h>
5 #include <sys/types.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
```

Note: Form 2 person Team for  
A Semester Long Project.

Sharp: Hardware Layer  
(Sensors/Actuators)  
↓  
"Security"  
Device Driver/Kernel Space  
↓  
Process Management

↓  
Web Server

↓  
Smartphone APPs.

ChatGPT 3.5 API + Python Interface

Example: Sample Code for GPIO Device  
Driver

Sept. 11 (Monday)

Notel: Homework Due in 1½ weeks.

To Be Posted on Line today;

a. Bring up the target, Screen  
Capture with Personal Identifier.

b. GPIO Testing (Python + Hardware  
Circuit). (1) Input Testing (KT).

Output Testing (KT). (2) Coding:

Python.

Note 2: B.o.M. (Bill of Material)

for the class/Project

a. Motor (Stepper Motor, NEMA.

17



Nema17  
Stepper Motor  
\$8.99  
Amazon.com

(2) BLDC  
Brushless DC  
motor

c. Smartphone (iPhone

OR Android phone

Swift Mac OS. as Development  
platform.

Ubuntu Linux.  
18.04.

NVIDIA JetPack (OS.)

Note:

With GPIO Testing

Ckt.

(Work-In-  
progress)

Target  
Board

NVIDIA Jetson Nano Bread Board/

OR Wine Wrapping  
Board.



350W Brushed  
Electric Motor



10 Inch Hub  
Motor 1000w, ...



48V 500W  
Wheel Motor ...

b. Motor Drive Unit.



EASON  
Stepper Motor  
\$9.69  
Amazon.com



SparkFun  
Electronics ...  
\$12.09  
DigiKey



WWZMDiB  
A4988 Stepper  
\$7.99  
Amazon.com



Pololu  
Corporation  
\$8.49  
DigiKey



[STEPPERONLINE](#)  
[CNC Stepper Motor](#)  
[Driver 1.0-4.2A](#)  
20-50VDC 1/128

## Example: Continuation on Linux D.D.

on ARM-11 (Samsung). Note! userSpace Code Samples, Kernel Space code

```
linux
harry@harry-laptop:/opt/FriendlyARM/mini6410$ cd linux/
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux$ ls
arm-qte-4.7.0      examples          u-boot-mini6410
arm-qt-extended-4.4.3  linux-2.6.38    x86-qte-4.6.1
arm-qtopia          rootfs_qtopia_qt4  x86-qt-extended-4.4.3
busybox-1.17.2     rootfs_qtopia_qt4-s x86-qtopia
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux$
```

Ref. Sample code on github.

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds$ ls
2022s-104d-userSpace-gpio.c  led  led.c  led.c~  Makefile  Makefile~
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds$
```

```
22
23      fd = open("/dev/leds0", 0);
24  if (fd < 0) {
25      fd = open("/dev/leds", 0);
26  }
27  if (fd < 0) {
28      perror("open device leds");
29      exit(1);
30  }
31
32      ioctl(fd, on, led_no);
33  close(fd);
34 }
```

Note1: "Char" Device. Open the Device just like a file.

Kernel (OS. Image)  
path to the Device.  
The Device Driver can be either integrated as the whole kernel image or module (installed/removed)

Note2 for passing control parameter(s) to the Device for Control Action.

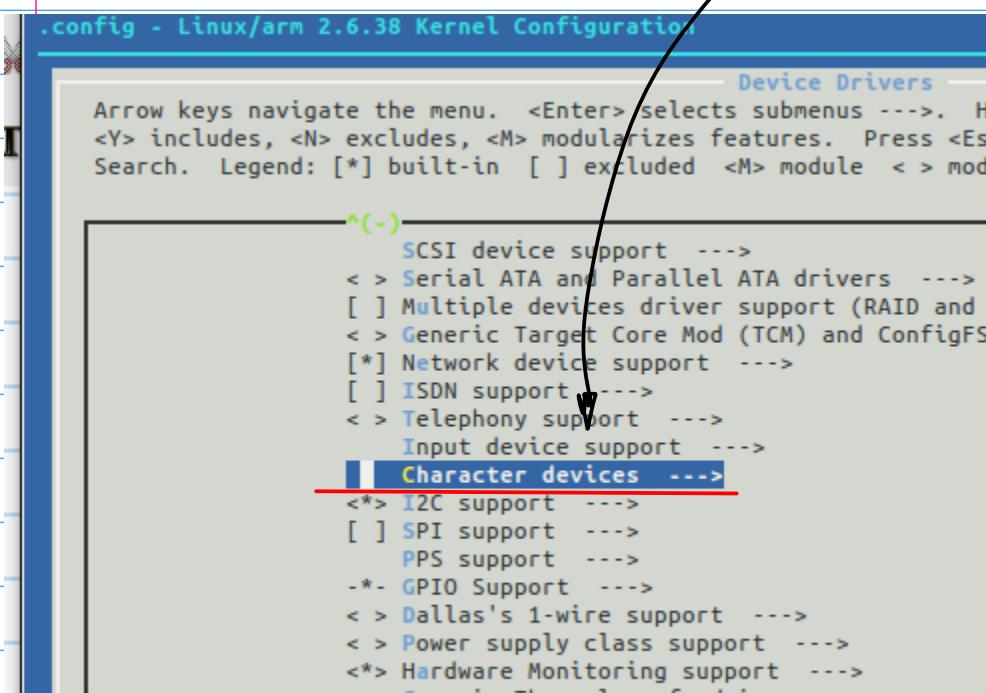
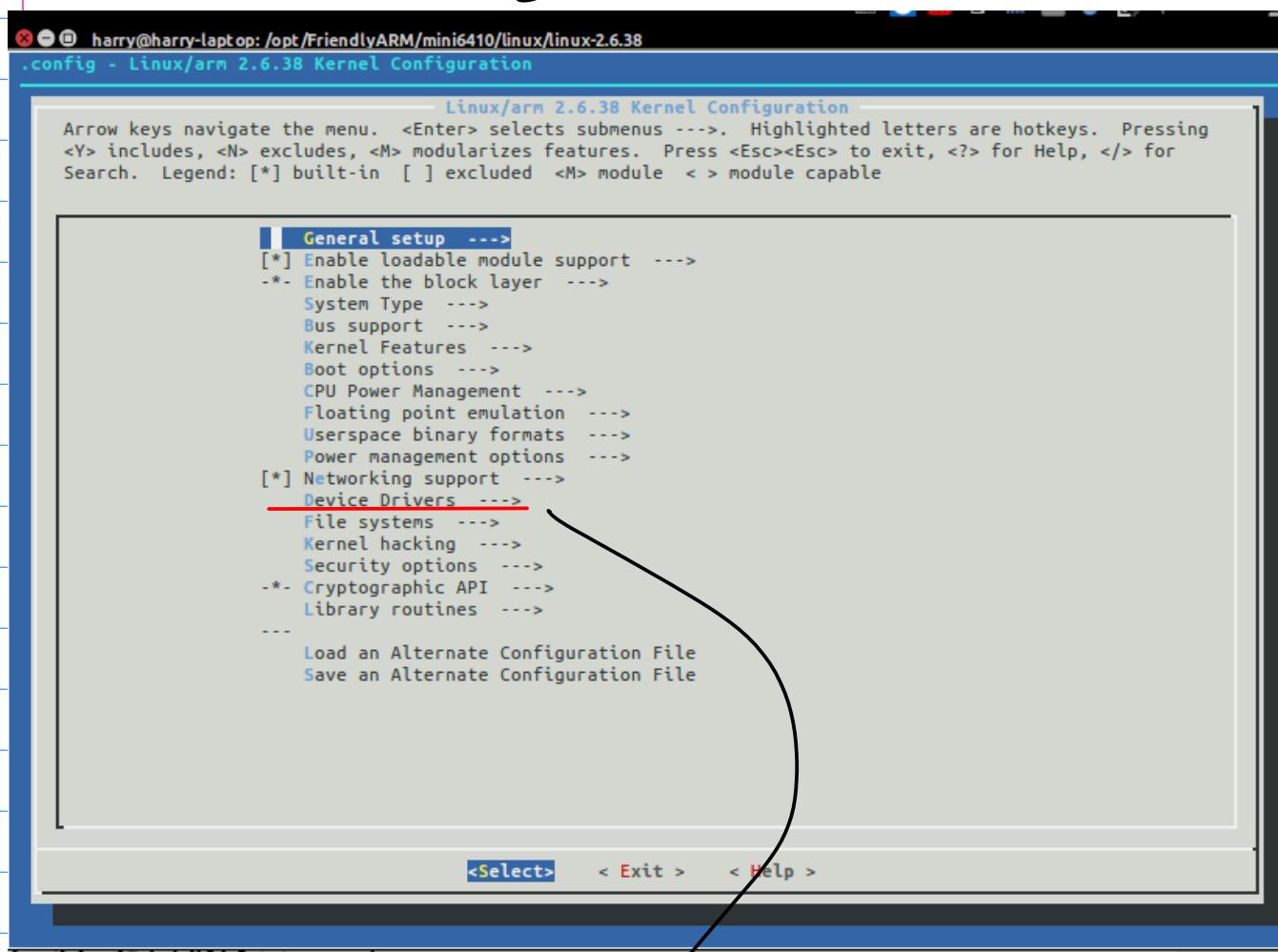
Note3. Close it, when done!

Build

Kernel image using

"menuconfig" → NVDA, Broadcom, Smart phones  
Embedded

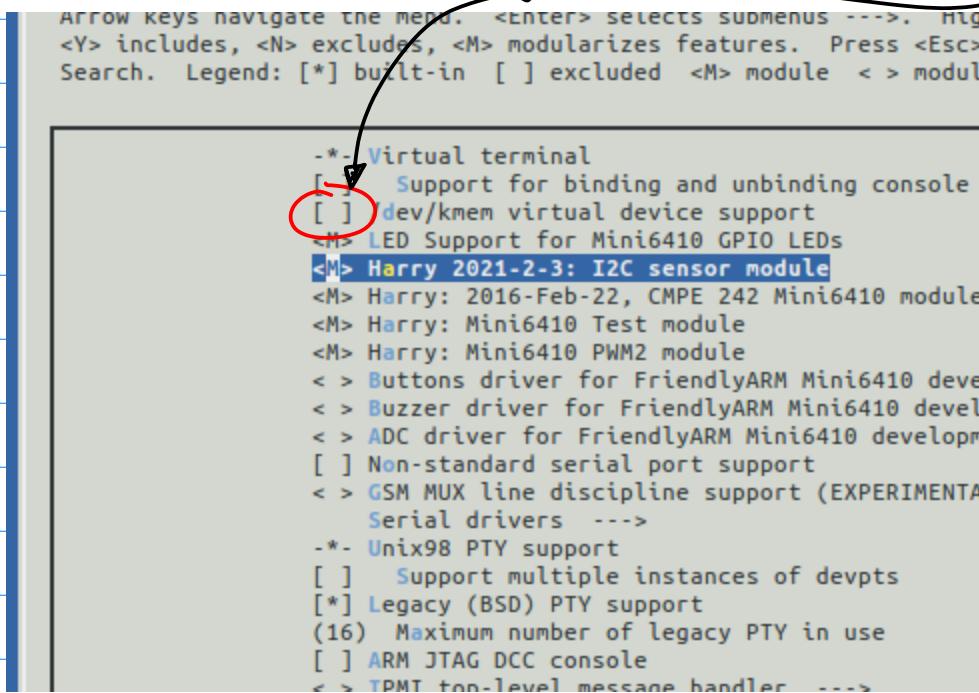
Android.



Note 4. menutools controls how the Kernel Image is built.  
 Here, the "Char" Device Driver can be Selected/Deselected.  
 b. Use "Space Bar" to toggle between 3 options.

```
Arrow keys navigate the menu. <Enter> selects submenus ---. Hig
<Y> includes, <N> excludes, <M> modularizes features. Press <Esc>
Search. Legend: [*] built-in [ ] excluded <M> module <> modul

```



(Non | M | \*)  
↓  
Module  
Integrated Kernel Image

Note 5. Folder for the "Char" D.D., The "gpio" (leds) Device Driver

```
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers$ cd char
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
20-2021S-9-mini6410_pwmHarry.c  ip2          misc.c      rtc.c
2q                           ipmi         misc.o      scc.i
agg                          isicom.c    mmtimer.c  scx2i
amiserial.c                  istallion.c modules.builtin ser_i
apm-emulation.c              Kconfig     modules.order moxa.c
applicom.c                   lp.c        moxa.h      moxa.h
applicom.h                   Makefile   mspec.c      mspec.c
bfin_jtag_comm.c             Makefile-backup  mwave       mxser.c
bfin-otp.c                   mbcs.c     mxser.h      mxser.h
briq_panel.c                mbcs.h      nozomi.c    nsc_gpio.c
bsr.c                        mem.c       nvram.c      nwbutton.c
built-in.o                   mem.o       nwbutton.h   nwbutton.h
cd1865.h                     mini6410_adc.c  nvram.c      nwflash.c
cyclades.c                  mini6410_adc.mod.c mini6410_adc.o pc8736x_gpio.c
digi1.h                      mini6410_buttons.c mini6410_buttons.o pxmcia
digiFep1.h                  mini6410_buttons.o mini6410_hello_module.c ppdev.c
digiPCI.h                   mini6410_hello_module.c mini6410_hello_module.mod.c ps3flash.c
ds1302.c                     mini6410_hello_module.ko mini6410_hello_module.mod.o ramosops.c
ds1620.c                     mini6410_hello_module.mod.o mini6410_hello_module.o random.c
dsp56k.c                     mini6410_hello_module.o mini6410_leds.c random.o
dtlk.c                       mini6410_leds.ko
efirtc.c
epca.c
epcaconfig.h
```

2022S-104e  
(CMPE242)



## Example: Kernel Space

Device Driver Code Requirements: Code Spec.

Note1:  
 Connect the Code to the CPU  
 Datasheet  
 ↓  
 Debug

```

66 static int __init dev_init(void)
67 {
68     int ret;
69
70     {
71         unsigned tmp;
72         tmp = readl(S3C64XX_GPECON);
73         tmp = (tmp & ~(0xffffU<<16))|(0x1111U<<16);
74         writel(tmp, S3C64XX_GPECON);
75
76         tmp = readl(S3C64XX_GPEDAT);
77         tmp |= (0xF << 4);
78         writel(tmp, S3C64XX_GPEDAT);
79     }
80
81     ret = misc_register(&misc);
82
83     printk (DEVICE_NAME"\nHarry: PGE initialized\n");
84 }
```

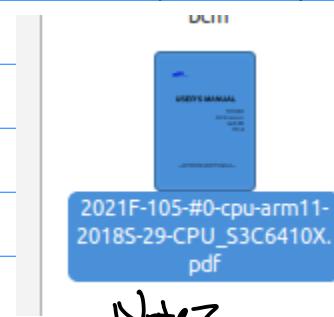
Note2. Read long 32bit

Naming: ID — Peripheral Control  
 (Manufacturer Part) GPP



Memory map.

Ref: CPU Datasheet.



Note2.

TP320.

a) GPE Con

6410X\_UM

## 10.5.5 PORT E CONTROL REGISTERS

There are five control registers including GPECON, GPEDAT, GPEPUD, GPECONS, and GPEPUDS in the Port E Control Registers.

Register	Address	R/W	Description	Reset Value
GPECON	0x7F008080	R/W	Port E Configuration Register	0x00
GPEDAT	0x7F008084	R/W	Port E Data Register	Undefined
GPEPUD	0x7F008088	R/W	Port E Pull-up/down Register	0x00000155
GPECONS	0x7F00808C	R/W	Port E Sleep mode Configuration Register	0x0
GPEPUDS	0x7F008090	R/W	Port E Sleep mode Pull-up/down Register	0x0

C. GPEDAT.

d. Two Addition types:

GPEPUD

GPECONS  
GPEPUDSPower  
Control

GPDCON	Bit	Description		Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0001 = Output 0011 = I2S CLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0001 = Output 0011 = I2S CDCLK[1] 0101 = Reserved 0111 = Reserved	0000

Note 3. From the D.D. Code, GPE1 is utilized for the I/O function

$$GPECON[7:4] = 0001$$

5  
GPIO  
pins

GPDCON	Bit	Description		Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0001 = Output 0011 = I2S CLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0001 = Output 0011 = I2S CDCLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE2	[11:8]	0000 = Input 0010 = PCM FSYNC[1] 0100 = AC97 SYNC 0110 = Reserved	0001 = Output 0011 = I2S LRCLK[1] 0101 = Reserved 0111 = Reserved	0000
GPE3	[15:12]	0000 = Input 0010 = PCM SIN[1] 0100 = AC97 SDI 0110 = Reserved	0001 = Output 0011 = I2S DI[1] 0101 = Reserved 0111 = Reserved	0000
GPE4	[19:16]	0000 = Input 0010 = PCM SOUT[1] 0100 = AC97 SDO 0110 = Reserved	0001 = Output 0011 = I2S DO[1] 0101 = Reserved 0111 = Reserved	0000

How To

Question: Make GPE1 as an output pin,  
But keep the rest unchanged?

Sept. 18 (Monday)

Note: 1<sup>o</sup> Homework Posted on the  
CANVAS.

2<sup>o</sup> Homework, Due Sept. 27  
(Wed), PWM Testing.

① Enable PWM Device  
Driver via Driver mapping  
utility By NVIDIA

② Test PWM Output By  
Changing f<sub>PWM</sub> from 2 kHz  
to 50 Hz; By changing  
Duty Cycle from 5% to 90%.

Then, Observe its Output

Note: Output with 2222 npn Transistor  
to drive a LED. (see the  
details in the Class PPT on  
github). (U.S.)

Note: Prepare the Source Distribution  
Download, to build Kernel  
Image from the distribution.  
(1 ~ 1½ week).

Note: Ref. on Smartphone Apps.  
Android APPs Development.

## Note: Device Driver Sample Code

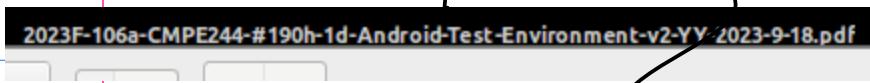
- 2023F-104-gpio-command-line-2023...
- 2023F-105a-2022s-104d-userSpace...
- 2023F-105b-mini6410\_leds.mod.c
- 2023F-106a-CMPE244-#190h-1d-A...

README.md

Update README.md

APPS .

Note: Install Android Studio  
ON your Laptop .



## Install Android Studio on Ubuntu 18.04 (9/15-8/3/2023)

nnn-n-Android-Development-HelloWorld-Calculator-v2-XW-2023-7-7.odt

### 1. Check Java version

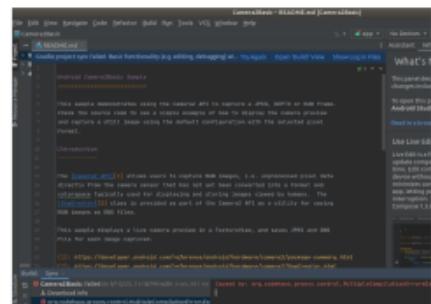
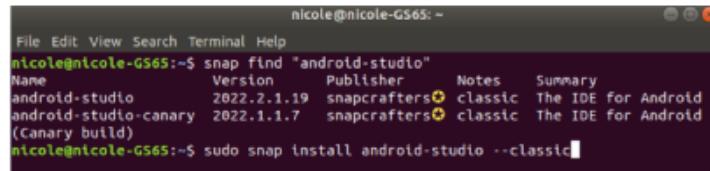
```
nicole@nicole-GS65:~$ java --version
openjdk 11.0.19 2023-04-18
OpenJDK Runtime Environment (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1)
OpenJDK 64-Bit Server VM (build 11.0.19+7-post-Ubuntu-0ubuntu118.04.1, mixed mode, sharing)
nicole@nicole-GS65:~$
```

3. Once installed, double click the icon to start the studio



### 2. Install Android Studio using snap

```
$ snap find "android-studio"
$ sudo snap install android-studio --classic
```



```
(base) harry@harrys-gpu-laptop:~$ sudo snap install android-studio --classic
[sudo] password for harry:
android-studio 2022.3.1.18 from Snapcrafters● installed
```

Note: Team Project (1) Formation of  
The Team ; (2) Selection of Smart phone  
for the APP. (3) Stepper motor  
Drive Board and Stepper motor.

Example: GPIO & PWM .

2023F-104

GPIO Command Line  
<https://jetsonhacks.com/2019/06/07/jetson-nano-gpio/>

```
# Map GPIO Pin
# gpio79 is pin 12 on the Jetson Nano
$ echo 79 > /sys/class/gpio/export
# Set Direction
$ echo out > /sys/class/gpio/gpio79/direction
# Bit Bangin'!
$ echo 1 > /sys/class/gpio/gpio79/value
$ echo 0 > /sys/class/gpio/gpio79/value
# Unmap GPIO Pin
$ echo 79 > /sys/class/gpio/unexport
# Query Status
$ cat /sys/kernel/debug/gpio
```

PWM

2021F-114b - ~

Note: Establish Remote Access to your target, e.g. Laptop to Access your target Board.

→ Next Monday, Show+Tell in class

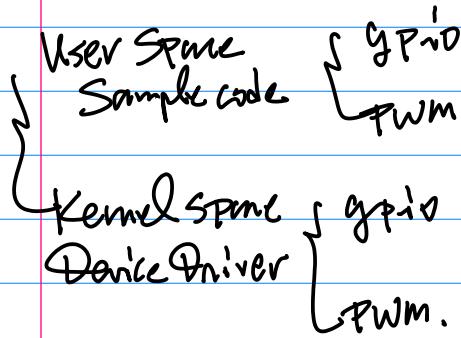
Sept 25 (Monday).

Note: 1<sup>o</sup> Updated github Sample code.

(1) 2023F-107.

Booting Jetson Nano without HDMI Connection.

(2) 2023F-105a to 2023F-105f.



Target Board Connection

Requirements.

a. The Demo/Show+Tell in Class is required, Also

Midterm Exam, and Final Exam are Based on Analyzing & Executing your Implementation Code on the target platform;

b. Bring gpio Homework with the target Board to the Classroom

Wednesday for Quick Inspection,

Show+Tell; (Sept. 27, Wednesday).

Note: HDMI Monitor is optional, To Be Able to Boot your System is the requirement.

Consider the github Sample code.

Ref: -105a, User Space program.

for GPIO.   
 ↳ open()   
 ↳ ioctl()   
 ↳ close()

-lvsb & c

```

1 #include <mach/gpio-bank-k.h>
2
3 #define DEVICE_NAME "leds0"
4
5 static long sbc2440_leds_ioctl(struct file *filp, unsigned
6                                long arg)
7 {
8     switch(cmd) {
9         case 0:
10            unsigned tmp;
11
12            if(arg)
13                tmp = 1;
14            else
15                tmp = 0;
16
17            /* ... */
18
19            /* ... */
20
21            /* ... */
22
23            /* ... */
24
25            /* ... */
26
27            /* ... */
28
29            /* ... */
30
31            /* ... */
32
33            /* ... */
34
35            /* ... */
36
37            /* ... */
38
39
  
```

Device\_NAME  
 ↳ a. User Space Program.  
 ↳ b. make

make menuconfig  
 for the Kernel  
 Driver Configuration.

## Kconf

Kernel configuration for make menuconfig.

```

80
81 source "drivers/hid/Kconfig"
82
83 source "drivers/usb/Kconfig"
84
85 source "drivers/uwb/Kconfig"
86
87 source "drivers/mmc/Kconfig"
88
89 source "drivers/memstick/Kconfig"
90
91 source "drivers/leds/Kconfig" Red circle around this line
92
93 source "drivers/nfc/Kconfig"
94
95 source "drivers/accessibility/Kconfig"
96

```

Kernel folder/driver folder

```

if (argc != 3 || sscanf(argv[1], "%d", &led_no)
    on < 0 || on > 1 || led_no < 0
    fprintf(stderr, "Usage: leds led_no 0|1"
    exit(1);
}

fd = open("/dev/leds0", 0); Red circle around /dev/leds0
if (fd < 0) {
    fd = open("/dev/leds", 0);
}
if (fd < 0) {
    perror("open device leds");
}

```

Note: 1°

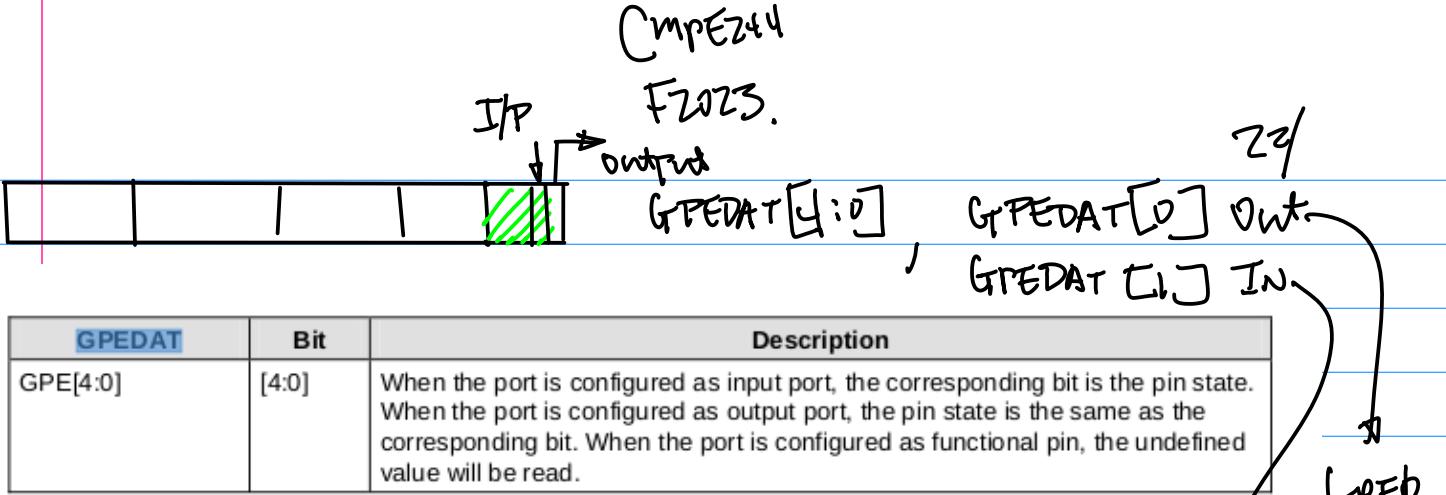
Note 2° SPR. Naming convention/addr. e.g.  
pointed from CPU Data sheet.

```

35 static long sbc2440_leds_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
36 {
37     switch(cmd) {
38         unsigned tmp;
39     case 0:
40     case 1:
41         if (arg > 4) {
42             return -EINVAL;
43         }
44         tmp = readl(S3C64XX_GPKDAT);
45         tmp &= ~(1 <(4 + arg));
46         tmp |= ((!cmd) <(4 + arg));
47        	writel(tmp, S3C64XX_GPKDAT);
48         //printk (DEVICE_NAME": %d %d\n", arg, cmd);
49         return 0;
50     default:
51         return -EINVAL;
52     }
53 }

```

↳ Bitwise Operations.



Note: Each bit (pin) can be set as an input or output by GPECON.

PP 320.

GPDCON	Bit	Description	Initial State
GPE0	[3:0]	0000 = Input 0010 = PCM SCLK[1] 0100 = AC97 BITCLK 0110 = Reserved	0000
GPE1	[7:4]	0000 = Input 0010 = PCM EXTCLK[1] 0100 = AC97 RESETn 0110 = Reserved	0000
GPE2	[11:8]	0000 = Input 0010 = PCM FSYNC[1] 0100 = AC97 SYNC 0110 = Reserved	0000
GPE3	[15:12]	0000 = Input 0010 = PCM SIN[1] 0100 = AC97 SDI 0110 = Reserved	0000
GPE4	[19:16]	0000 = Input 0010 = PCM SOUT[1] 0100 = AC97 SDO 0110 = Reserved	0000

Find the Binary Pattern to set these 2 pins.

$$GPECON[3:4] = 0001 \quad (O/P)$$

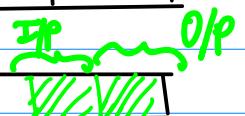
$$GPECON[7:4] = 0000 \quad (I/P)$$

for init & config.

Define Bitwise Operation mask

Design the Binary Pattern.

GPECON



mask to be designed

static void \_\_exit dev\_exit(void)

CmPE244

F2023

23/

Bitwise "OR"

Note: 1° Init module

2°

```
66 static int __init dev_init(void)
67 {
68     int ret;
69
70     {
71         unsigned tmp;
72         tmp = readl(S3C64XX_GPECON);
73         tmp = (tmp & ~(0xffffU<<16)) | (0x1111U<<16);
74        	writel(tmp, S3C64XX_GPECON);
75
76         tmp = readl(S3C64XX_GPEDAT);
77         tmp |= (0xF << 4);
78        	writel(tmp, S3C64XX_GPEDAT);
79     }
80
81     ret = misc_register(&misc);
82
83     printk (DEVICE_NAME"\Harry: PGE initialized\n");
84
85     return ret;
86 }
```

From 2023F-105c-mini6410\_leds.c

#define DEVICE\_NAME "leds0"

static long sbc2440\_leds\_ioctl(struct file \*filp, unsigned int cmd, unsigned long arg)

static int \_\_init dev\_init(void)

static void \_\_exit dev\_exit(void)

GPIO SPI for Configuration

Hardware  
Software  
User Code  
Driver Code  
CPU  
Datasheet

Sept. 27 (Wed).

Note 1: Homework 1 #2.

Note 2: Inspection of the Prototype System

GPIO Testing.

Example: Architecture

Device Driver

OS Kernel Image

Note: PWM + I2C  
for the future discussion

Let's consider Configure + Build OS. Kernel Image.

Step 1. Download OS Distribution

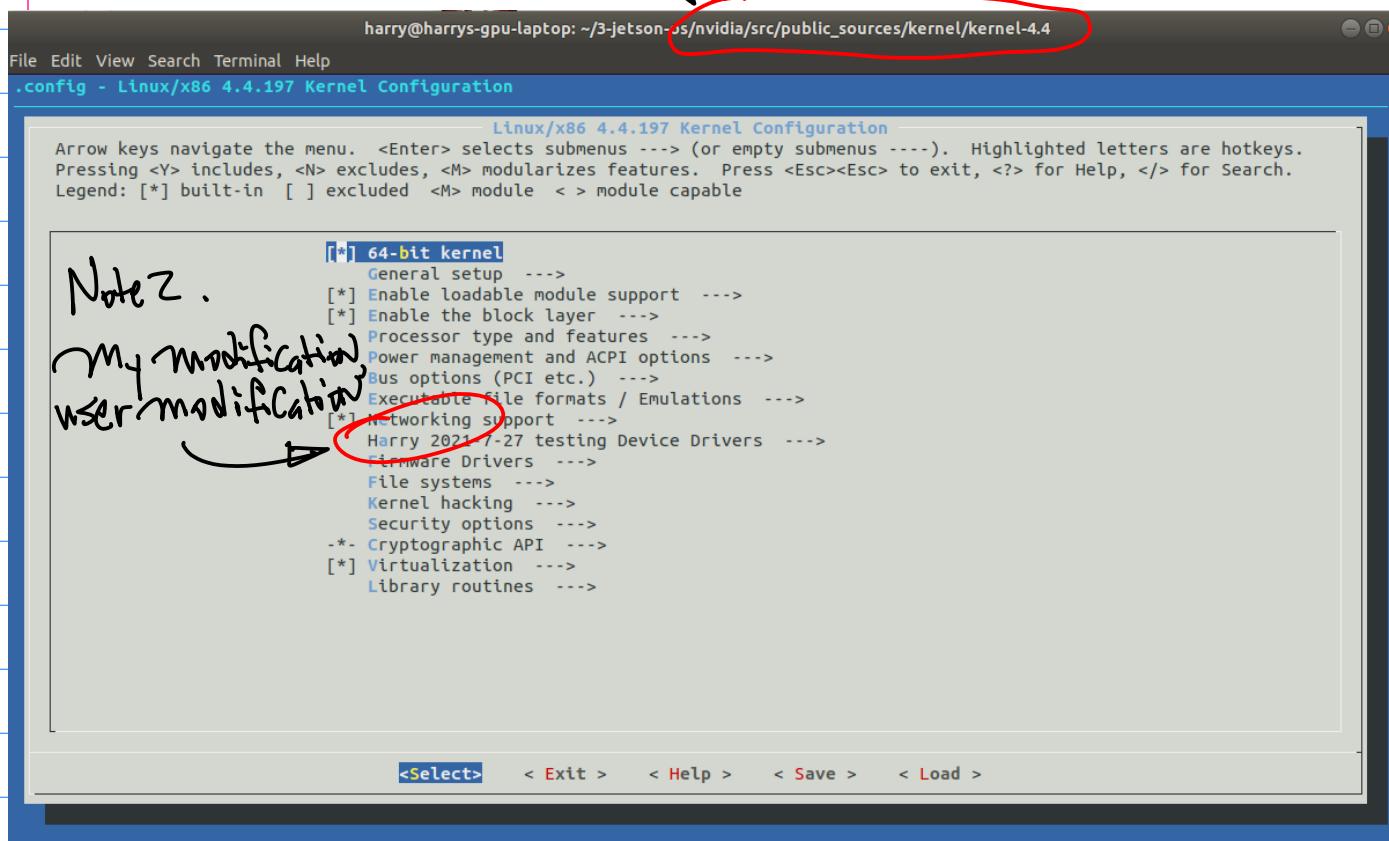
↓  
Step 2. Menutools to Build OS.

By the manufacturer's Default  
setting



Step 3. Create User-Defined  
Kernel image.

Note1: menuconfig UI



Now, Similar Setup for Sam's Arm11, Check Kconf for UI Customization  
at the Root folder.)

```

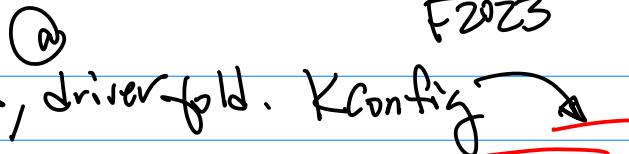
harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
1 #
2 # For a description of the syntax of this configuration file,
3 # see Documentation/kbuild/kconfig-language.txt.
4 #
5 mainmenu "Linux/$ARCH $KERNELVERSION Kernel Configuration"
6
7 config SRCARCH
8     string
9     option env="SRCARCH"
10
11 source "arch/$SRCARCH/Kconfig"

```

CmREZt4

F2023

25/

Next Level, driver-fold. Kconfig 

```
x - harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers
1 menu "Device Drivers"
2
3 source "drivers/base/Kconfig"
4
5 source "drivers/connector/Kconfig"
6
7 source "drivers/mtd/Kconfig"
8
9 source "drivers/of/Kconfig"
10
11 source "drivers/parport/Kconfig"
12
13 source "drivers/pnp/Kconfig"
14
15 source "drivers/block/Kconfig"
16
17 # misc before ide - BLK_DEV_SGIIOC4 depends on SGI_IOC4
18
19 source "drivers/misc/Kconfig"
20
21 source "drivers/ide/Kconfig"
22
```

Next Level to "Char" folder, KConfig.

```
x - harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char
1 #
2 # Character device configuration
3 #
4
5 menu "Character devices"
6
7 config VT
8     bool "Virtual terminal" if EXPERT
9         depends on !S390
10        select INPUT
11        default y
12        ---help---
13            If you say Y here, you will get support for terminal devices with
14            display and keyboard devices. These are called "virtual" because you
15            can run several virtual terminals (also called virtual consoles) on
16            one physical terminal. This is rather useful, for example one
17            virtual terminal can collect system messages and warnings, another
18            one can be used for a text-mode user session, and a third could run
19            an X session, all in parallel. Switching between virtual terminals
20            is done with certain key combinations, usually Alt-<function key>.
21
22            The setterm command ("man setterm") can be used to change the
23            properties (such as colors or beeping) of a virtual terminal. The
```

```

100 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112     tristate "Harry 2021-2-3: I2C sensor module"
113     depends on CPU_S3C6410
114     help
115         I2C sensor module Feb 17, 2016.
116 #
117 #
118 #
119 # Harry: Here is my modification
120 #
---[END_OF_KERNEL_MODULE]

```

Ref: 2023 F-109, Readme.

Oct. 2nd (Monday) . Oct. 8 (Sun)

Note1. Homework, Due 1 week

Continuation from the Homework  
of O.S. Kernel Source Distribution  
Installation.

Modify Kconfig, to

1° Add user defined device  
Driver Option as a char  
device.

The option appears as.

FirstName\_LastName\_CMPE244\_driverX, see Ref Below.

```

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
.config - Linux/arm 2.6.38 Kernel Configuration

Character devices

Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded [M] module < > module capable

-# Virtual terminal
[ ] Support for binding and unbinding console drivers
[ ] /dev/kmem virtual device support
<M> LED Support for Mini6410 GPIO LEDs
<M> Harry 2021-2-3: I2C sensor module
<M> Harry: 2016-Feb-22, CMPE 242 Mini6410 module sample
<M> Harry: Mini6410 Test module

```

2° Provide Screen Capture  
of your UI.

3° provide Kconfig file  
which realizes this  
function.

4° Hardware Side.  
Motor Drive Board.

Note: place comments to highlight the modification

```

105      This option enables support for LEDs connected to GPIO lines
106      on Mini6410 boards.
107
108 #-----#
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112     tristate "Harry 2021-2-3: I2C sensor module"
113     depends on CPU_S3C6410
114     help
115         T2C sensor module Feb 17 2016

```

Option to consider.

Be Careful, Power budget  
for A4988 is  
16mA.



A4988 Stepper  
Motor Driver...

\$5.89

Amazon.com  
Free shipping

Note 2: Quiz is scheduled A  
week from today Oct. 9th  
(Monday).

1° Bring the target Board &  
Prototype Board to the Class

2° Be sure to have a way  
to capture the UI.

3° Scope of the Quiz:

CPU Datasheet, CPU Architecture  
Memory-map, SPRs (Special  
Purpose Registers) for GPP.

→ ARM11, Background Info  
from LPC;

4° Execute GPIO Code

5° take a photo of your  
entire System Setup.

Cmpe244

F2023

28/

6° Screen Capture from your target Board to show the execution of the code with Personal ID is in the Screen Capture.

Time to Complete: 20 min  
for the Quiz, 15 min. to  
Prepare the Submission

Example: Continuation on KConfig.

7° Python Code.

8° Submission.

→ photos in png/jpg.

Code

A piece of paper with  
handwritten Answer.



Take a photo to Capture  
your answer sheet, then  
Convert it to pdf.

Then, place all the files  
(2 photos, 1 pdf, 1 code)

into one package, Zip it.

Naming the file:

FirstName - LastName - SID - 244.zip.  
(4 digits)

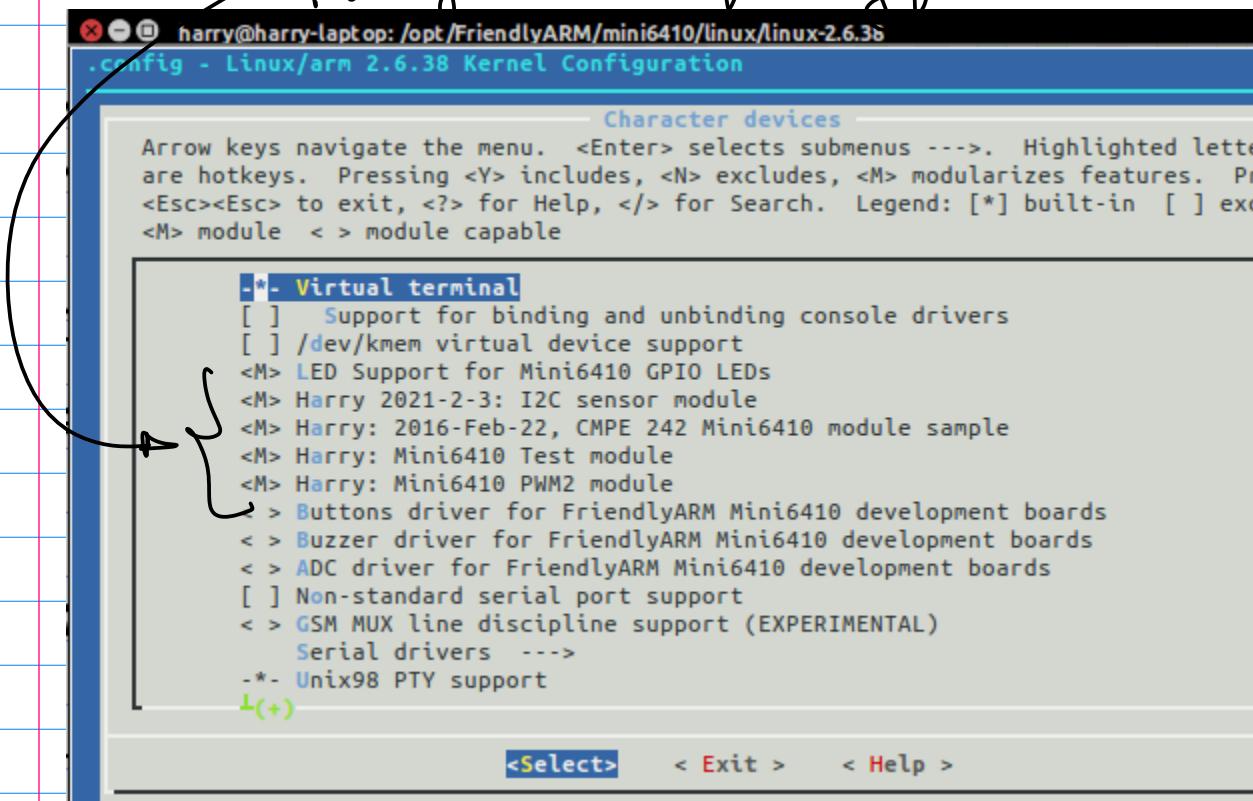
Submission to the CANVAS.

CMPE242

F2023.

29/

Objectives: To Be Able to Customize OS, Kernel, e.g. by modifying KConfig to have the following feature.



Note 1. The Architecture of the KConfig.

```
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd drivers/char/
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ ls
Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ vi Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$ cd ..
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd ..
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd drivers/
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd char
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ cd char
harry@harry-laptop:~/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls
```

Annotations:

- a) at the Root
- b. drivers
- c. Char. ==

Note: To Create an entry, use "Config"

```

107
108 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112 tristate "Harry 2021-2-3: I2C sensor module"
113 depends on CPU_S3C6410
114 help
115     I2C sensor module Feb 17, 2016.
116 #
117
118 #

```

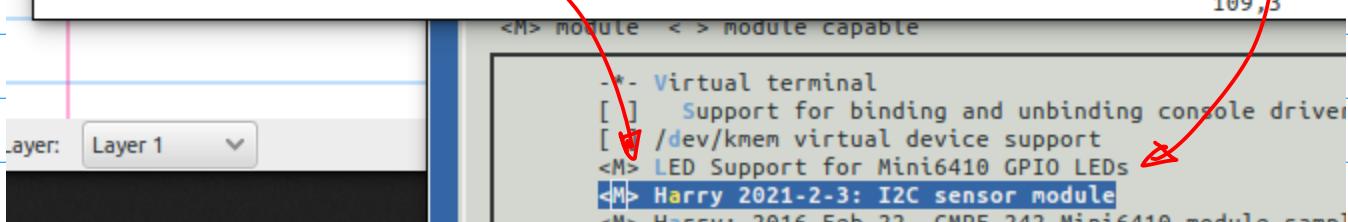
Note: To define the target CPU, use "depends on" followed by Manufacturer ID & Device ID.

```

107
108 #
109 # Harry: Feb 17, 2016
110 #
111 config MINI6410_I2CSEN_MODULE
112 tristate "Harry 2021-2-3: I2C sensor module"
113 depends on CPU_S3C6410
114 help
115     I2C sensor module Feb 17, 2016.
116 #
117
118 #
119 # Harry: Here is my modification
120 #

```

Note:



Note 2: To pre-select, use default "y"

```

70
91 config DEVKMEM
92     bool "/dev/kmem virtual device support"
93     default y
94     help
95         Say Y here if you want to support the /dev/kmem device. The
96         /dev/kmem device is rarely used, but can be used for certain
97         kind of kernel debugging operations.
98         When in doubt, say "N".

```

Note: To provide Description for the option, use "help"

Once the KConfig is updated with matching Name of the Device Driver, then \$make all

\$insmod *driverName*  
\$printk( )  
\$rmmod *driverName*