CmpE244
Fall 2023

1/

August 21 (Monday)

Organizational Meeting.

1. The "Greenshot" 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 D State University
Office Location:	Engineering Building 267A
Telephone:	(408) 924-4060 (650) 400-1116
Email:	hua.li@sjsu.edu
Class Days/Time:	Mondays and Wednesdays, 4:30 pm – 5:45 pm, Aug
Office Hours:	Mondays and Wednesdays, 3:00 pm – 4:00 pm
Classroom:	Engineering Building Room 295
Prerequisites:	CMPE 180A and CMPE 180D, classified standing, c Artificial Intelligence or Computer Engineering or S majors only.

2. Emphasis on POSIX O.S. Linux Open Source 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

Option1. NVDA Jetson NANO. GPU (128) CPU
4 GB Version JetPack

Option2. BroadCom Pie3B+, Pie4.

option3. RISC-V FPGA Dev. Board. + FreeRTOS

Has limitted Processing Power. May Not meet the Need for Our Project

Selection Decision in 1 week

Option4. NXP LPC11C24 or LPC17xx, RTOS. NXP Dev. Forum.

CmpE244
Fall2023

4. TextBook & References

Set I: Datasheet(s), CPU Datasheet, Developer Guide; Set II: NVDA 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. Pie 3B+, or Pie 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-cxd2t?P=1z0jm4m&Keyword=LPC1769&FS=True&gclid=Cj0KCQjwqKuKBhCKARIsACf4XuHyN8WfqtQ24WGgtoMdKd6n-kI7c-YNz-r1hTcPt0ErdZN62jrMQmgaAtXZEALw_wcB or Option 5: Samsung ARM11 developer platform.
2. Linux Host Machine (Ubuntu 18.04) .

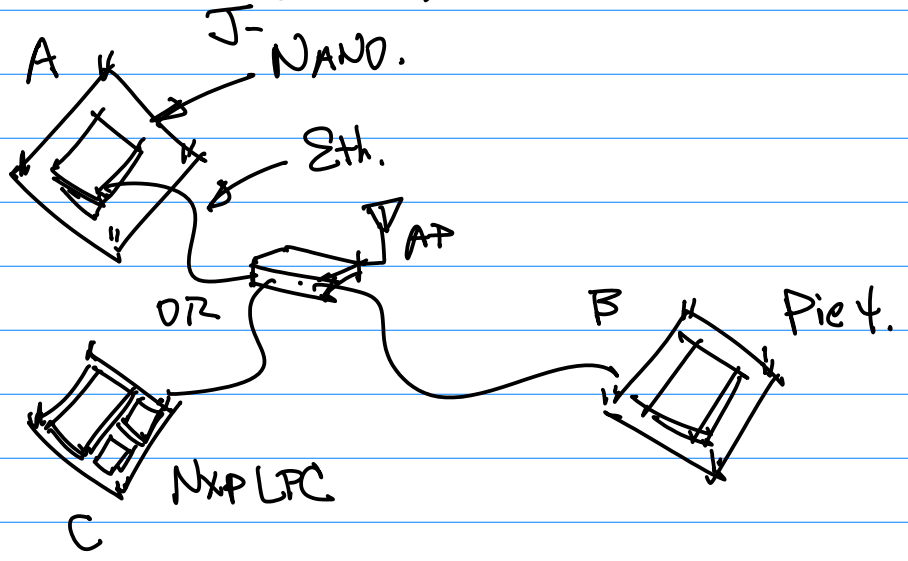
Naming Convention:

2021F-114-handout-gpi... Add files via upload
2021F-114b-pwm-nano... Add files via upload

yr. Semester ID

A & B
A & C

Note: Regarding The Selection of A Target Platform:



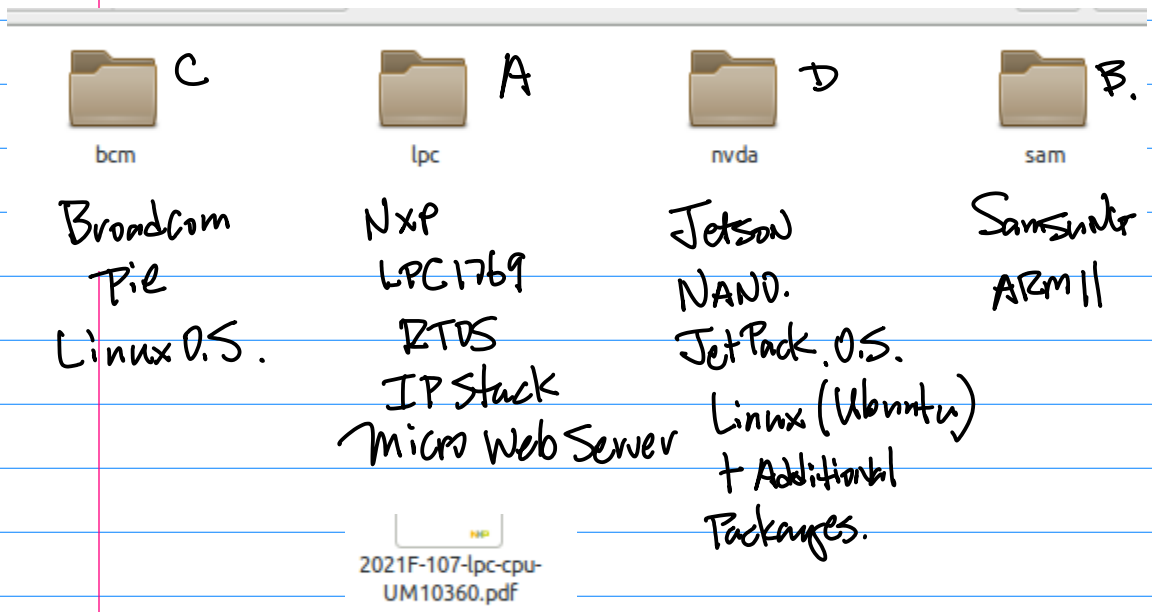
5. Grading Policy

Project Assignment: (Two Projects) Phased
15% (pts) for the assigned projects.
15% for the Semester Long Project

Assignments and projects:	30%
Midterm Exam:	30%
Final Exam:	40%
Total:	100%

August 23rd (Wed)
Introduction
Note: IZm268

Ref:
Datasheets.



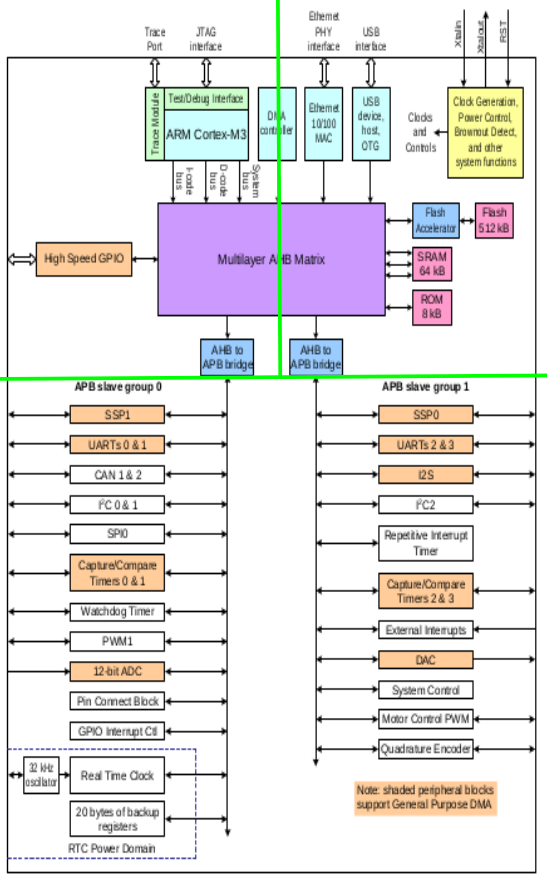


Fig 1. LPC1768 simplified block diagram

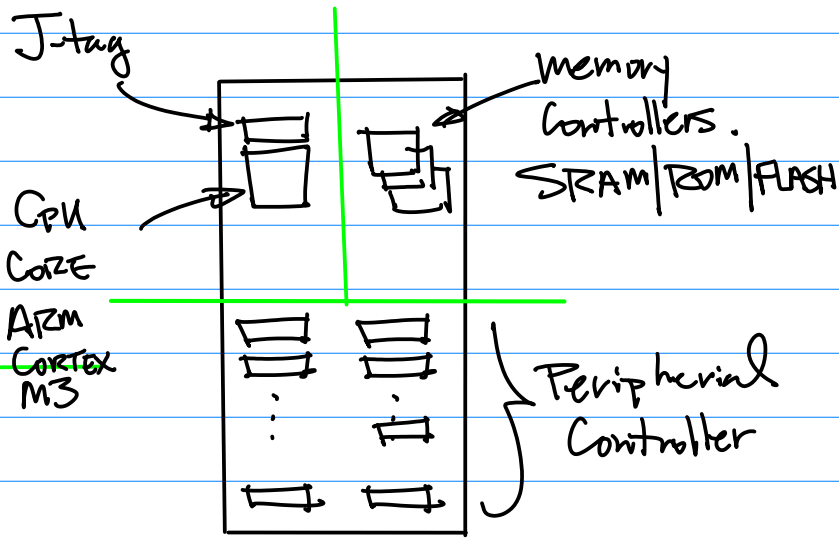


Fig. 1

Note1: The CPU Block Diagram for LPC1769 is a Sample for the Rest of the target platforms, e.g. Pre3/4; Sam's ARM 11; NVDA Jetson NANO

Note2:

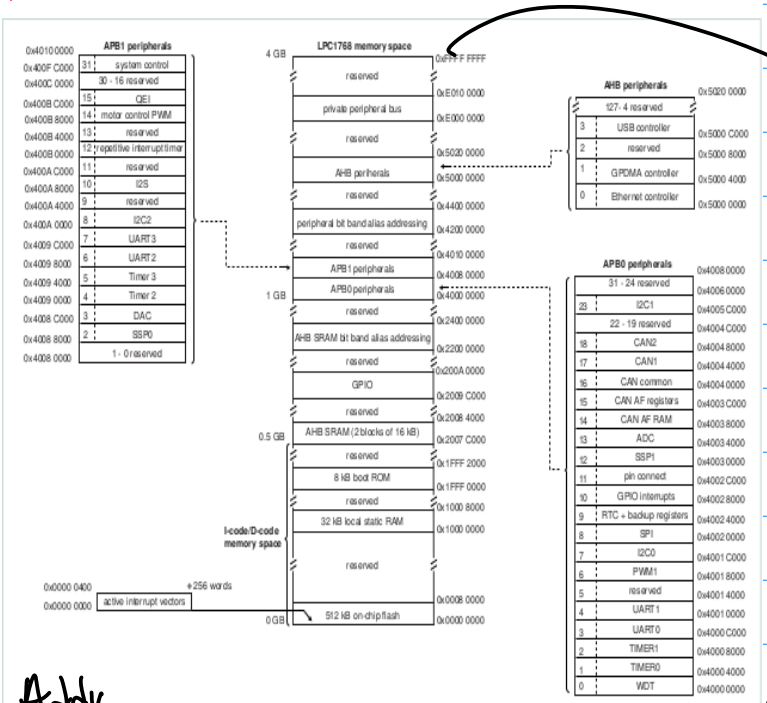
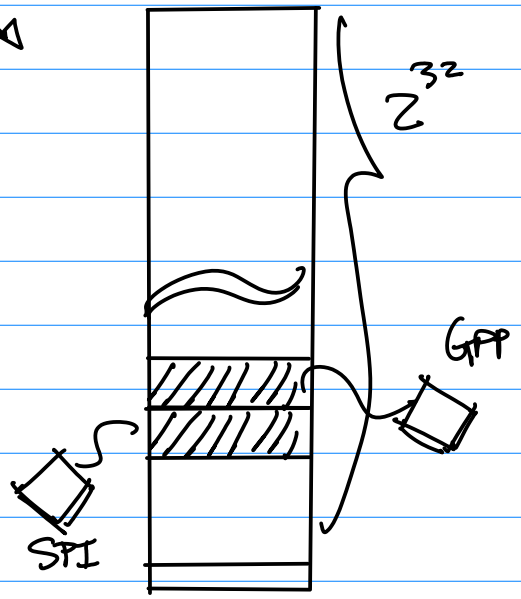


Fig.2



0x0000_0000 PWR-up Addr.

Addr.

$2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^{2}$

1024 : : 4

1K : :

1meg. :

1G.

4 G Addresses.

Datasheets.

2021F-105-#0-cpu-arm11-2018S-29-CPU_S3C6410X.pdf

Locate the page with the top level Description of the CPU Architecture

Example: "B", Sam's CPU ARM11

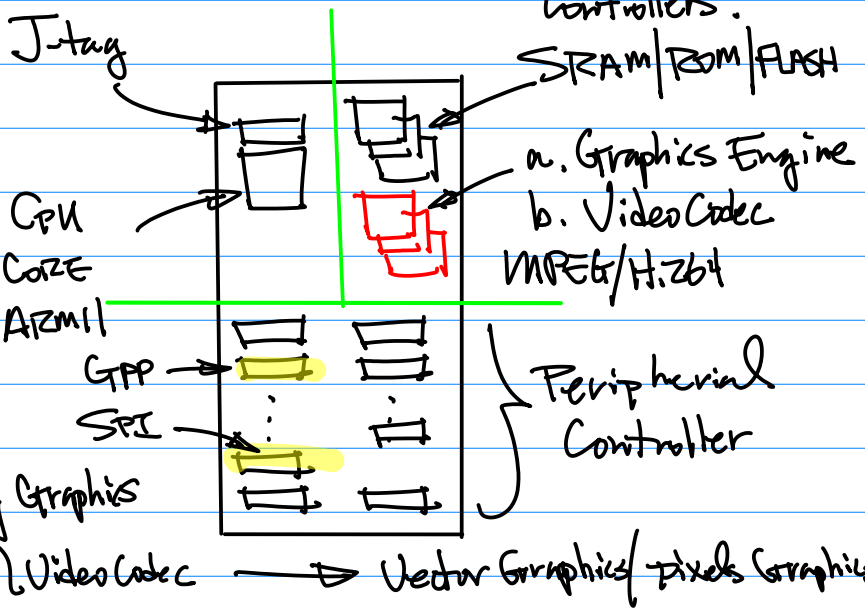


Fig.3.

Differentiation { Graphics
Video Codec → Vector Graphics / pixels Graphics

Example: Connection to (Embedded) Software Architecture

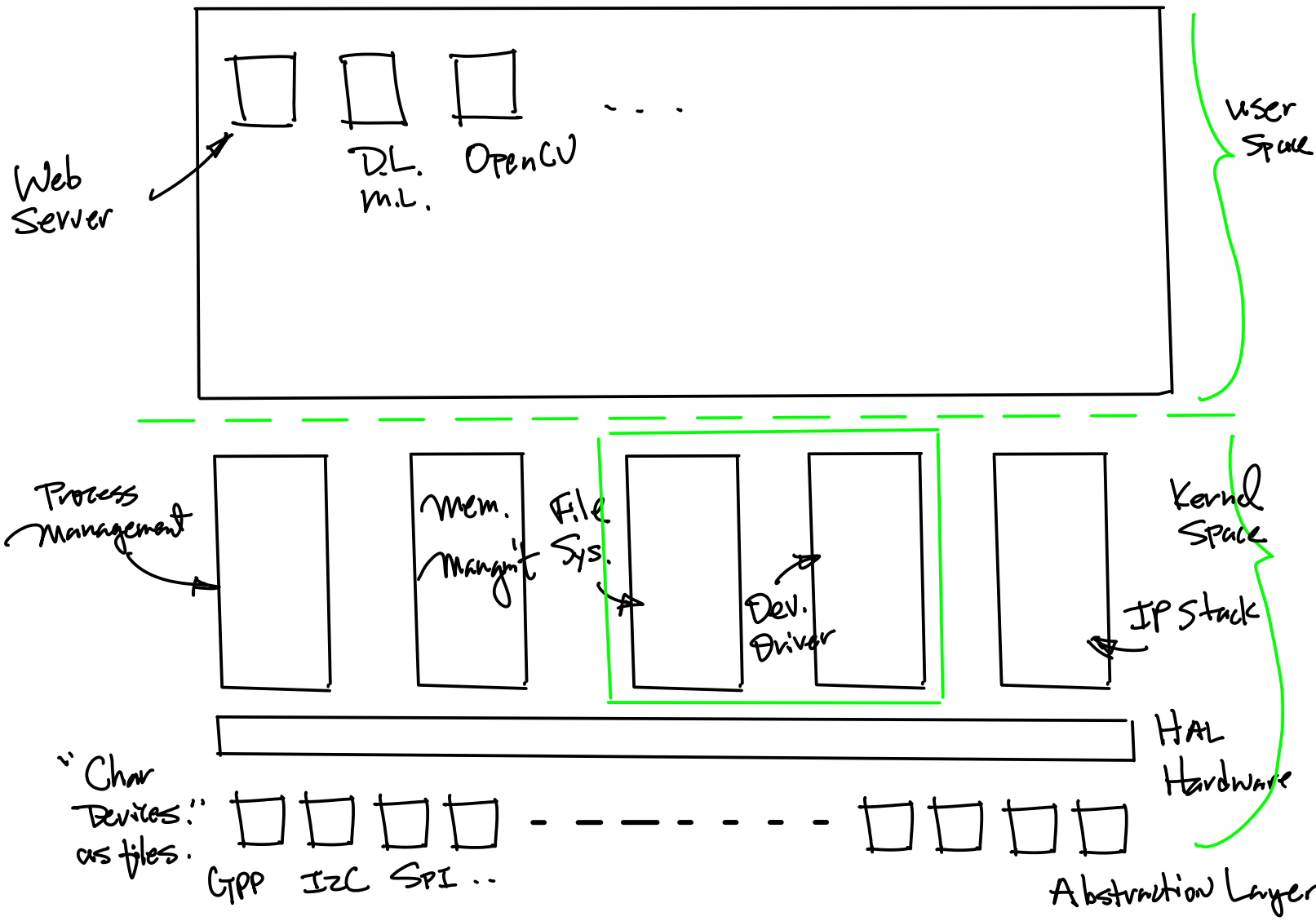


Fig. 1.

Note: Data Size for 1080P Image or 720P

August 28 (Monday)

Note: 1° Brief Description on the Scope of Semester-Long Project.

Embedded Software; Kernel O.S. Device Driver → Apps for iphone/Android phone

2° CANVAS is up. Honesty pledge

3° Target platform → Minor upgrade to Enable RTC By Adding On-Board Battery

Example: Continuation of the Introduction/Embedded Software Architecture.

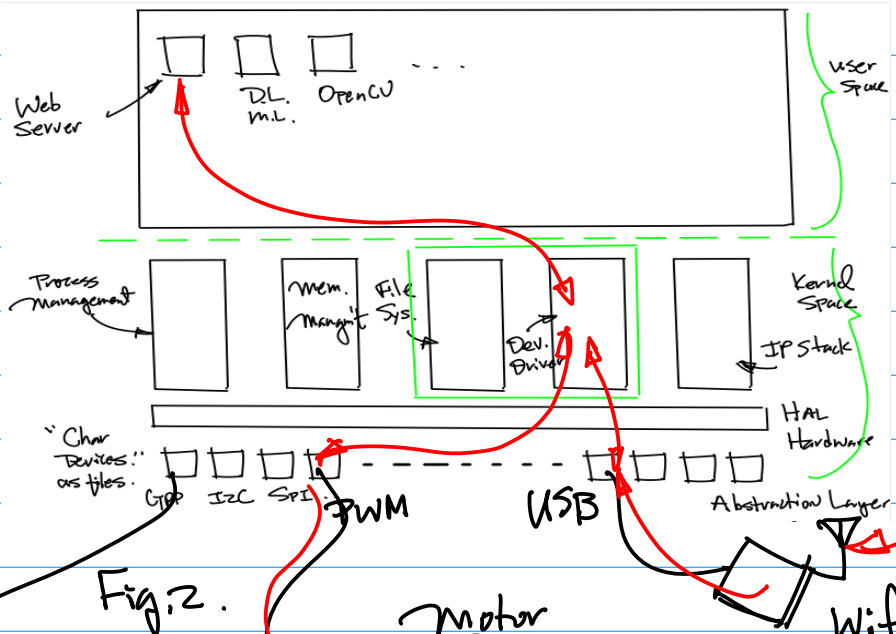
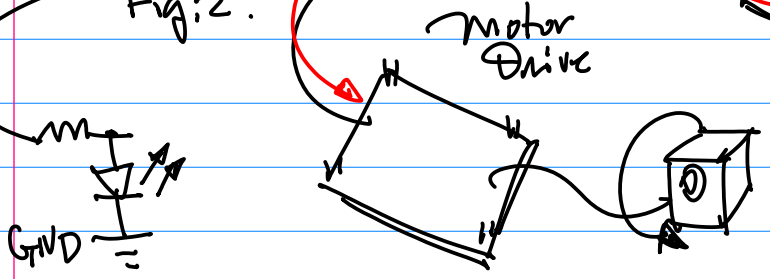


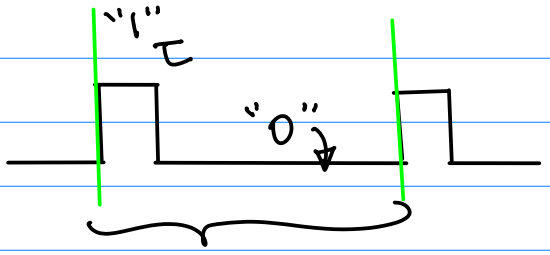
Fig.2.



Wifi/BT

Fig.2

Note: PWM — Pulse Width Modulation.



T One Period

Duty Cycle $\triangleq \tau / T$ ... (1)

f_{PWM} ... (2)

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

C/C++ → Assembly Output

Porting.

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

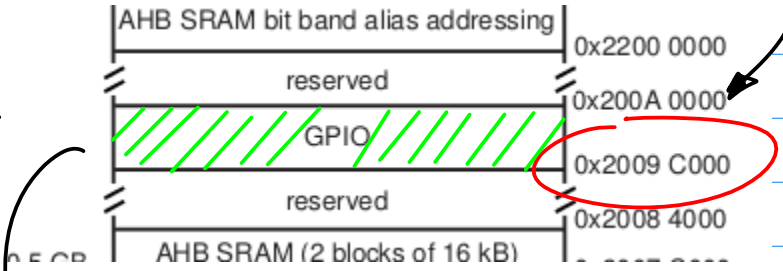
Peripheral Controller
A set of Special purpose Registers.

Identify A peripheral controller, GPP

Most Likely this SPR has the Addr in the Block.
ISA
0x2009_C000

Then, make a GPP as
(1) Output Port
(2) Turn on "LED"
(3) Turn off the "LED"
See Fig.3.

Fig.3.



Memory is designated for SPR's (Special Purpose Registers)

Go to 32Bit Control/Configuration Register. To Define Init & Config. for GPP output.

Control & Configuration Register

0xuuuu_uuuu

0x2009_C000

GPx_CON

For Selection of Port & Pin(s).

4 G Possible Configurations Theoretically
$(2^{32} = 2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10})$

1K 1K 1K

1M

1 G.

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

Coding (Software Aspect)
Write 32 bits unsigned Data as Init & Config pattern to select the GPP & the pin as output.

Next. Naming Convention.
Guideline:
RISC → UC Berkeley, David Patterson
Stanford, John Hennessy.

August 30 (Wed)

Note: 1° CANVAS is up.
Honesty Pledge to Be Signed
And Submit ON CANVAS
By this Friday 11:59 pm.

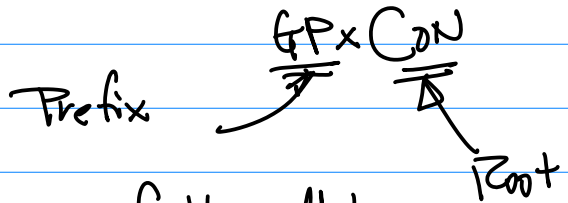
2° Please Bring the target platform to the Class. Next Wednesday.

3° (Written Requirements) in 2 Week
Bring up your target platform
By Down Loading Kernel OS.
Image to A micro-SD Card,
then boot the System.
then Screen Capture with your personal identifier, Submit it ON CANVAS.

4° Create A ChatGPT Account, Python interface to Chat GPT (3.5) API.

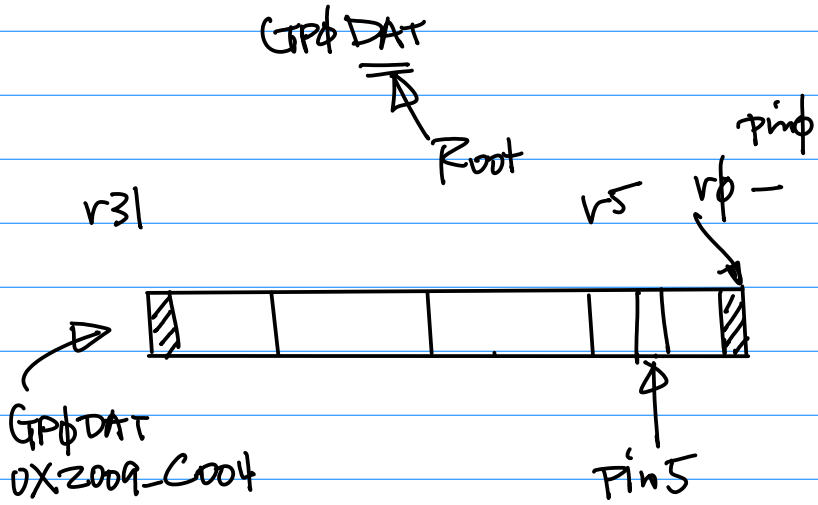
Example: Continued.
Naming Convention of the Control & Configuration Register.
By John Hennessy. Golden Rules
Uniformity,
Regularity,
Orthogonality
"3+3" Naming Convention

Prefix + Root
3 Letters 3 Letters.



for the multiple GPPs,
we have
GPBCON, GPICON, etc.
Suppose we want to use GPB pin 5 as an Output to turn ON/off LED.

Design 2nd SPR.



place "1" @ r5 to Output Logical high.
"0" .. ...
Low.

#define GPBCON
.. GPBDAT

CmpE244
F2023

10)

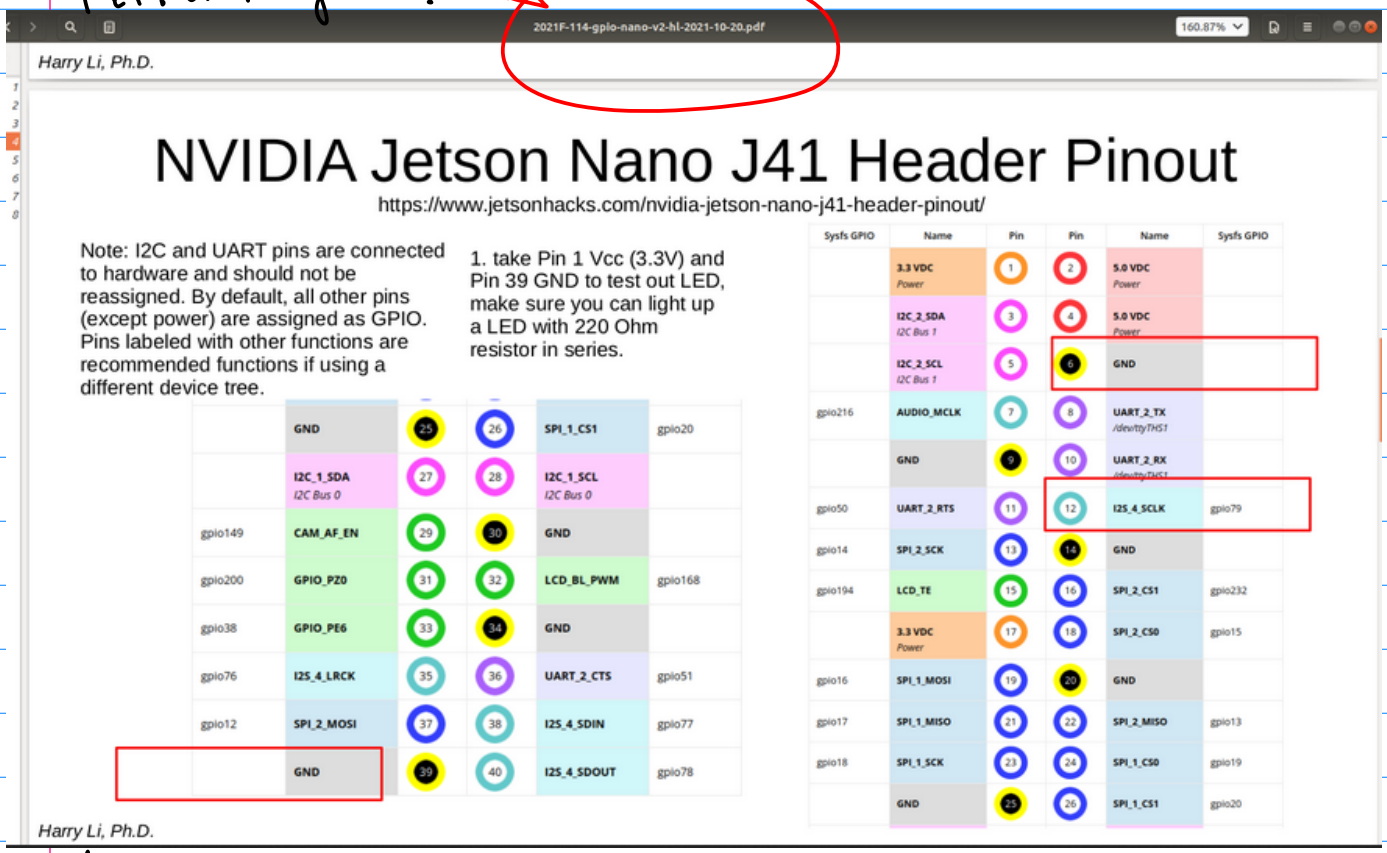
Porting Porting
gcc/g++ → ARM → CORTEX
Porting ↓
Board

Sept 6 (Wed).

Note: 1° Target Board
Inspection:

Purpose: J41 Connector
RTC Battery
Ref: on the github.

2021F-114~



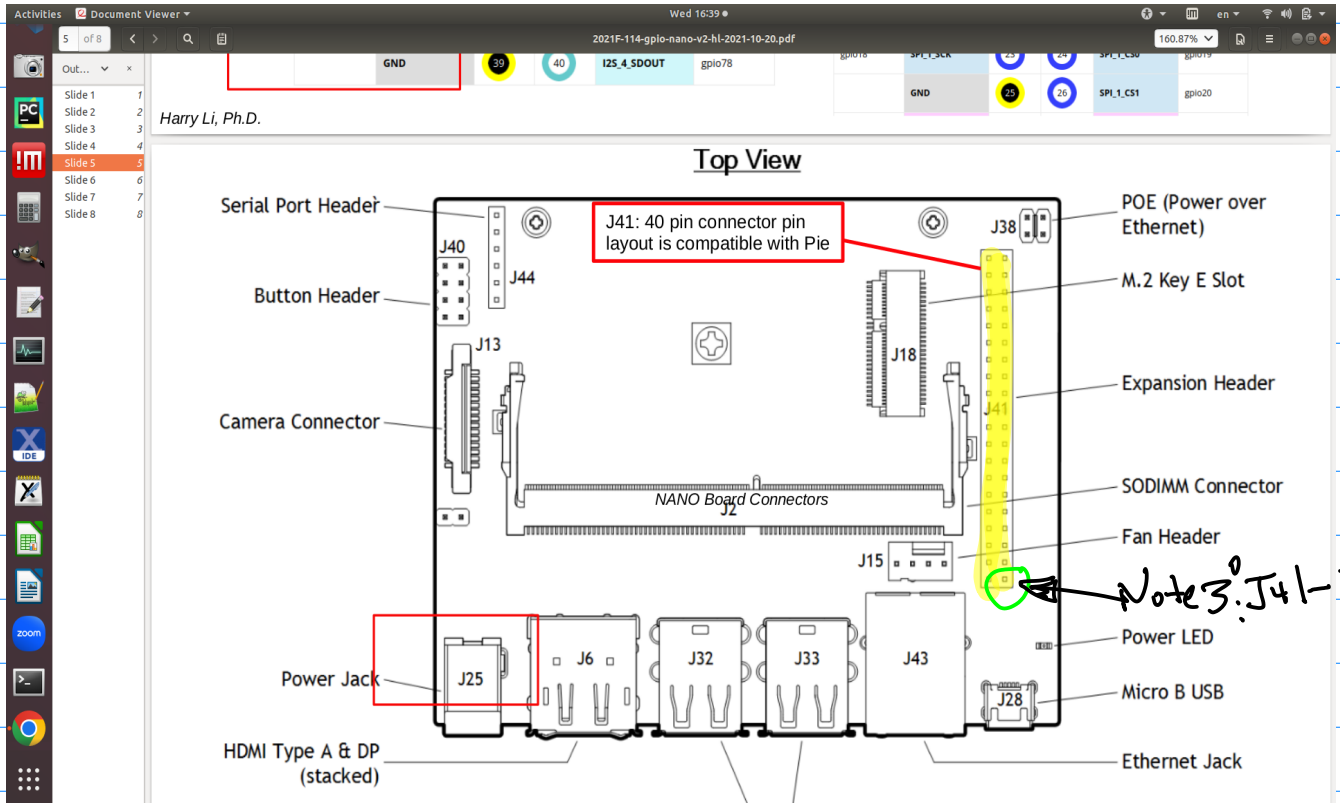
Note: 1° Power Pins
- GND: 6/9/25/39
- Vout 3P3VDC/5VDC Pin1 Pin2,4.
- Vin: J25 (5A or higher @5VDC)

Note2° GPIO

J41 Pins	CPU Functionality
J41-12	gpio79
J41-40	gpio78

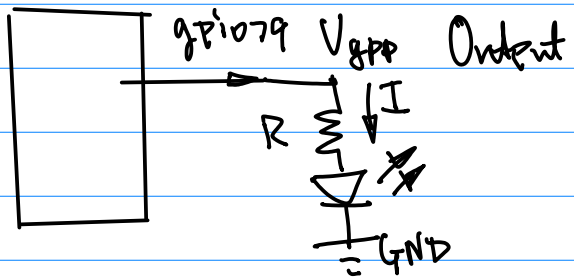
CmpE258
F2023

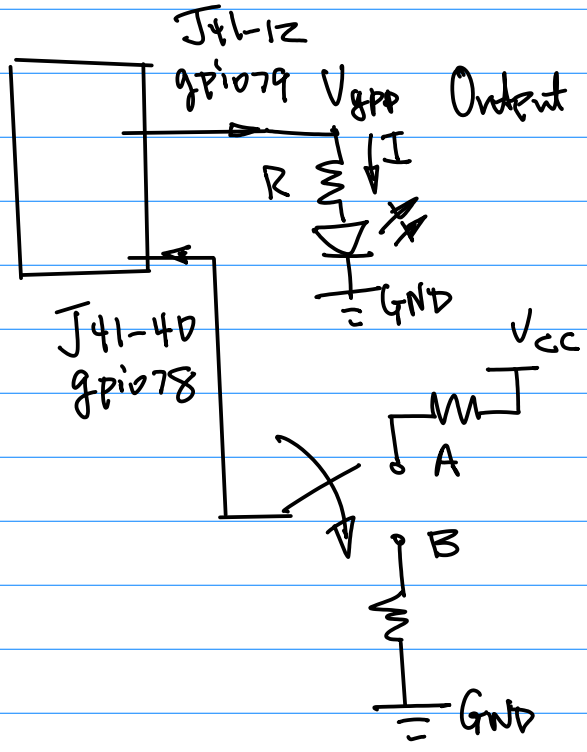
11/



Note 4°: GPIO Input/Output Testing CKT

Build the following Testing CKT.





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

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

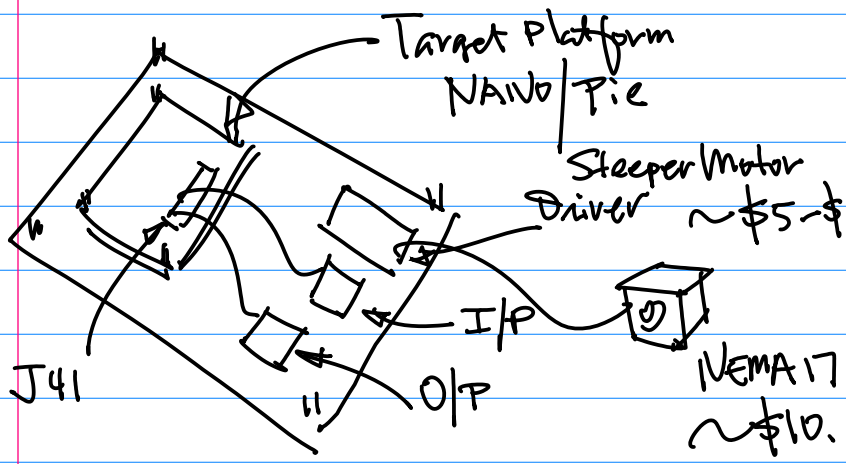
W/O Resistor with Proper selected LED.

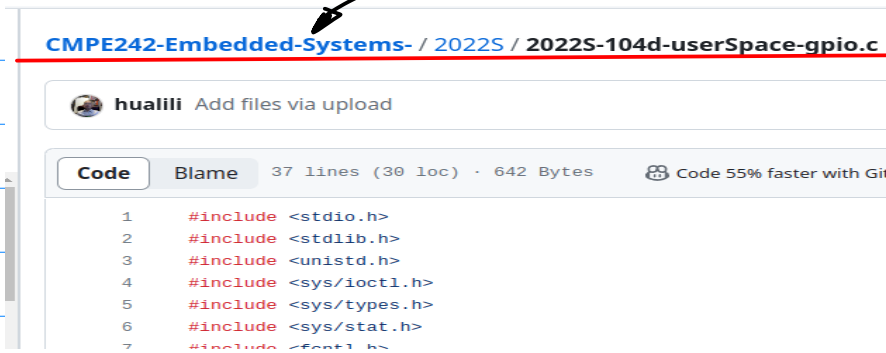
GPIO J41 Pins	CPU Functionality	Note
J41-12	gpio79	Output
J41-40	gpio78	Input

Code { User Space Code
Kernel Space Code

Take a Reference Design from ARM11, Samsung CPU.

Ref: Sample code has been posted on the github.





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

Scoop: Hardware Layer (Sensors/Actuators) "Secuator"

↓

Device Driver/Kernel Space Process Management

↓

Web Server

↓

Smart phone APPs.

Chat GPT 3.5 API + Python Interface

Example: Sample Code for GPP Device Driver

Sept. 11 (Monday)

Note 1. 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 CKT. Output Testing CKT. ② Coding: Python.

Note 2: B.o.M. (Bill of Material) for the class/Project

a. Motor (1) Stepper Motor. NEMA 17

Nema17 Stepper Motor
$8.99
Amazon.com

(2) BLDC Brushless DC motor

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

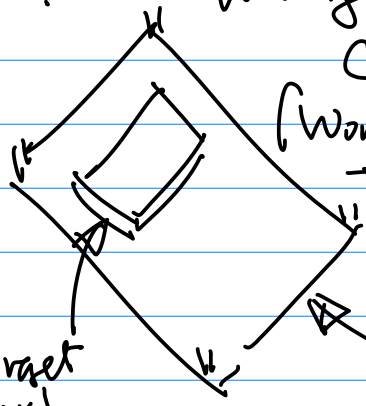
C. Smartphone { iPhone OR Android phone

↔ Swift macOS. as Development platform.

Ubuntu Linux. 18.04.

NVDA JetPack (O.S.)

Note: With Gpio Testing CKT. (Work-In-progress).



Example: Continuation on Linux D.D.

on ARM-11. (Samsung). Note! : userspace Code Samples, Kernel space code

```
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 (O.S. 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!

CmpE244

F2023

Build Kernel Image using "menuconfig" → NVDA, Broadcom, Smart phones, Android.

Embeded

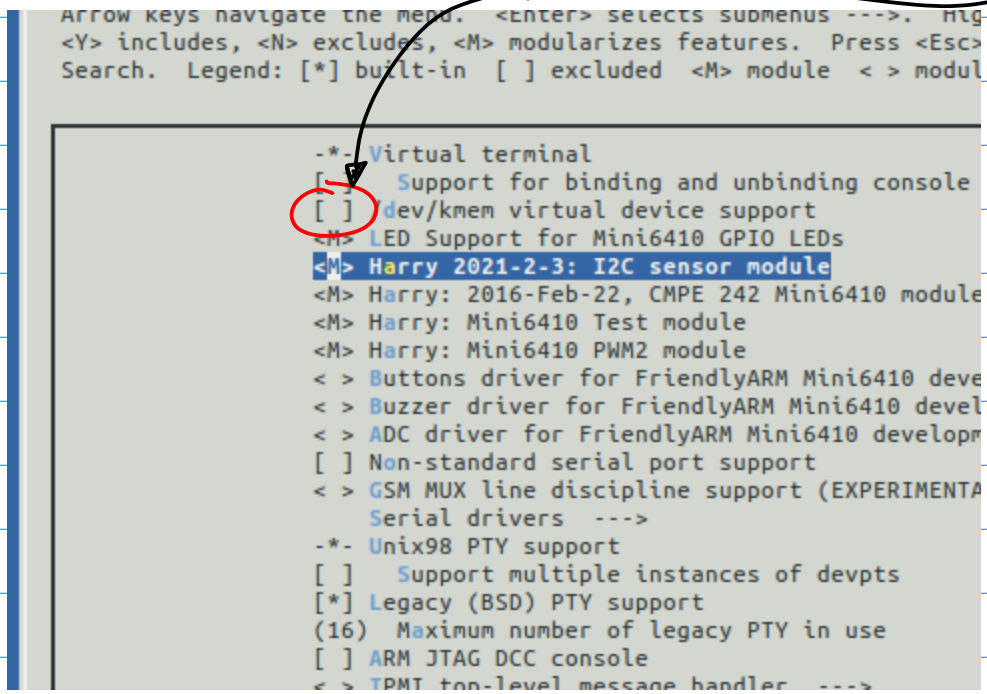
```
harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38
.config - Linux/arm 2.6.38 Kernel Configuration
                    Linux/arm 2.6.38 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  Highlighted letters are hotkeys.  Pressing
 <Y> includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for
 Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

        General setup  --->
    [*] Enable loadable module support  --->
    -*- Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Userspace binary formats  --->
        Power management options  --->
    [*] Networking support  --->
        Device Drivers  --->
        File systems  --->
        Kernel hacking  --->
        Security options  --->
    -*- Cryptographic API  --->
        Library routines  --->
    ---
        Load an Alternate Configuration File
        Save an Alternate Configuration File

              <Select>    < Exit >    < Help >
```

```
.config - Linux/arm 2.6.38 Kernel Configuration
                                    Device Drivers
 Arrow keys navigate the menu.  <Enter> selects submenus --->.  H
 <Y> includes, <N> excludes, <M> modularizes features.  Press <Es
 Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > mod
                 ^(-)
        SCSI device support  --->
    < > Serial ATA and Parallel ATA drivers  --->
    [ ] Multiple devices driver support (RAID and
    < > Generic Target Core Mod (TCM) and ConfigFS
    [*] Network device support  --->
    [ ] ISDN support  --->
    < > Telephony support  --->
        Input device support  --->
        Character devices  --->
    <*> I2C support  --->
    [ ] SPI support  --->
        PPS support  --->
    -*- GPIO Support  --->
    < > Dallas's 1-wire support  --->
    < > Power supply class support  --->
    <*> Hardware Monitoring support  --->
```

Note4. menuconfig controls how the Kernel Image is built.
Here, a. the "Char" Device Drive can be selected/Deselected.
b. Use "Space Bar" to toggle between 3 options.



(N|M|*)
Module
Integrated Kernel Image

Note5. Folder for the "char" D.D., The "gpio" (leds) Device Driver

2022-10-4e (CMPE242)

```
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.h
agp                             isicom.c                mmtimer.c               scx200_gpio.c
amiserial.c                     istallion.c             modules.builtin         ser_a2232.c
apm-emulation.c                 Kconfig                 modules.order           ser_a2232.h
applicom.c                      lp.c                    moxa.c                  ser_a2232fw.ax
applicom.h                      Makefile                moxa.h                  ser_a2232fw.dl
bfin_jtag_comm.c                Makefile-backup         mspec.c                 serial167.c
bfin-otp.c                      mbcs.c                  mwave                   snsc.c
briq_panel.c                    mbcs.h                  mxser.c                 snsc.h
bsr.c                           mem.c                   mxser.h                 snsc_event.h
built-in.o                      mem.o                   nozomi.c                sonypi.c
cd1865.h                        mini6410_adc.c          nsc_gpio.c              specialix.c
cyclades.c                      mini6410_adc.mod.c      nvram.c                 specialix_io8.h
digi1.h                         mini6410_adc.o          nwbutton.c              stallion.c
digiFep1.h                      mini6410_buttons.c      nwbutton.h              sxboards.h
digiPCI.h                       mini6410_buttons.o      nwflash.c               sx.c
ds1302.c                        mini6410_hello_module.c pc8736x_gpio.c          sx.h
ds1620.c                        mini6410_hello_module.ko pcmcia                 sxwindrv.h
dsp56k.c                        mini6410_hello_module.mod.c ppdev.c             synclink.c
dtlk.c                          mini6410_hello_module.mod.o ps3flash.c          synclink_gt.c
efirtc.c                        mini6410_hello_module.o ramoops.c               synclinkmp.c
epca.c                          mini6410_leds.c         random.c                tb0219.c
epcaconfig.h                    mini6410_leds.ko        random.o                tlclk.c
```

Example: Kernel Space Device Driver Code. Requirements: Code ↔ Spec.

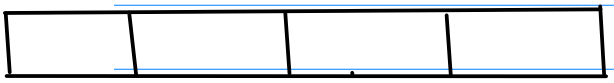
Note1: Connect the code to the CPU Datasheet

Debug

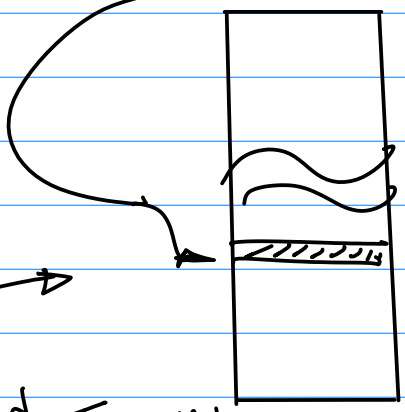
```
66 static int __init dev_init(void)
67 {
68         int ret;
69
70         {
71                 unsigned tmp;
72                 tmp = readl(S3C64XX_GPECON);
73                 tmp = (tmp & ~(0xffffU<<16))|(0x1111U<<16);
74                 writel(tmp, S3C64XX_GPECON);
75
76                 tmp =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

Note2. Readlong. 32bit

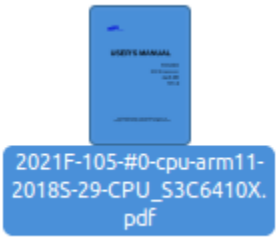
Naming: ID — Peripheral Control (Manufacturer/Part) GPP



Memory map.



Ref: CPU Datasheet.



Note2. pp320.

a) GPE CON b) Addr. / On the memory. c. GPEDAT . d. Two Addition types: Pull up down.

GPEPUD

GPECONSLP

GPEPUDSLP

Power Control

6410X_UM GPIO

10.5.5 PORT E CONTROL REGISTERS

There are five control registers including GPECON, GPEDAT, GPEPUD GPECONSLP and GPEPUDSLP 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
GPECONSLP	0x7F00808C	R/W	Port E Sleep mode Configuration Register	0x0
GPEPUDSLP	0x7F008090	R/W	Port E Sleep mode Pull-up/down Register	0x0

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

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

5 GPIO pins

How to

Question: Make GPE1 as an output pin, But Keep the Rest unchanged?

Sept. 18 (Monday)

Note: 1° Homework Posted on the CANVAS.

2° Homework, Due Sept. 27 (Wed), PWM Testing.

① Enable PWM Device Driver via Driver mapping utility By NVDA

② Test PWM Output By Changing f_{PWM} from 2KHz 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). (0.5.)

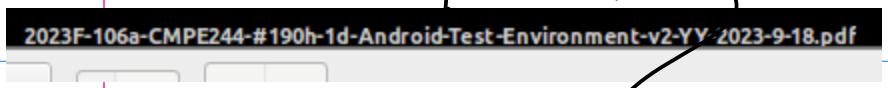
Note: Prepare the Source Distribution Download, to build Kernel Image from the distribution. (1 ~ 1½ week)

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

Note: Device Driver Sample Code

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 mod
e, sharing)
nicole@nicole-GS65:~$
```

2. Install Android Studio using snap
   $ snap find "android-studio"
   $ sudo snap install android-studio --classic

```
nicole@nicole-GS65: ~
File  Edit  View  Search  Terminal  Help
nicole@nicole-GS65:~$ snap find "android-studio"
Name                Version      Publisher      Notes    Summary
android-studio      2022.2.1.19  snapcrafters✪  classic  The IDE for Android
android-studio-canary  2022.1.1.7  snapcrafters✪  classic  The IDE for Android
(Canary build)
nicole@nicole-GS65:~$ 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
```

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

Note: Team Project ① Formation of the Team; ② Selection of Smart-home 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-1146 ~

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° Updated github Sample code.

① 2023F-107. Booting Jetson NANO without HDMI Connection.

② 2023F-105a to 2023F-105f.

User Space Sample code { gpio, pwm

Kernel Space Device Driver { gpio, pwm.

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 Class on 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( )

-105b&c

Device_NAME

a. User Space Program.

b. make menuconfig for the Kernel Driver Configuration.

```
33 #define DEVICE_NAME "leds0"
35 static long sbc2440_leds_ioctl(struct file *filp, unsigned
   long arg)
36 {
37         switch(cmd) {
38                 unsigned tmp;
39         case 0:
```

CmpE244
F2023

21

Kconf

R Kernel configuration for make menuconfig.

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers

```
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"
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);
    if (fd < 0) {
            fd = open("/dev/leds", 0);
    }
    if (fd < 0) {
            perror("open device leds");
```

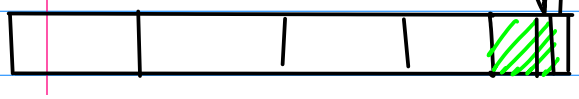
Note: 1°

Note 2° SPR. Naming convention/Addr. e.g. pointer from CPU Datasheet.

```
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.

CmpE244
F2023.



GPEDAT[4:0], 22/

GPEDAT[0] Out

GPEDAT[1] IN

GPED (I/P)

GPEI (O/P)

GPEDAT	Bit	Description
GPE[4:0]	[4:0]	When the port is configured as input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

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	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

Find the Binary Pattern to set these 2 pins.

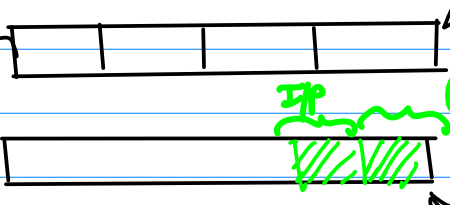
GPECON[3:2] = 0001 (O/P)

GPECON[1:0] = 0000 (I/P)

for Init & Config.

Define Bitwise Operation mask

GPECON



mask to Be Designed

DR

Design the Binary Pattern.

static void __exit dev_exit(void)

CmpE244
F2023

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)

Sept. 27 (Wed).

Note1: Homework #2.

Note2: Inspection of the prototype System GPIO testing.

Example: Architecture

↓

Device Driver

↓

O.S. Kernel Image

GPIO SFR for Configuration

↓ Hardware

↓ Software

User Code

Driver Code

CPU Datasheet

Note: PWM + I2C for the future Discussion

Let's consider Configure + Build O.S. Kernel Image.

Step1. Down load O.S. Distribution

↓

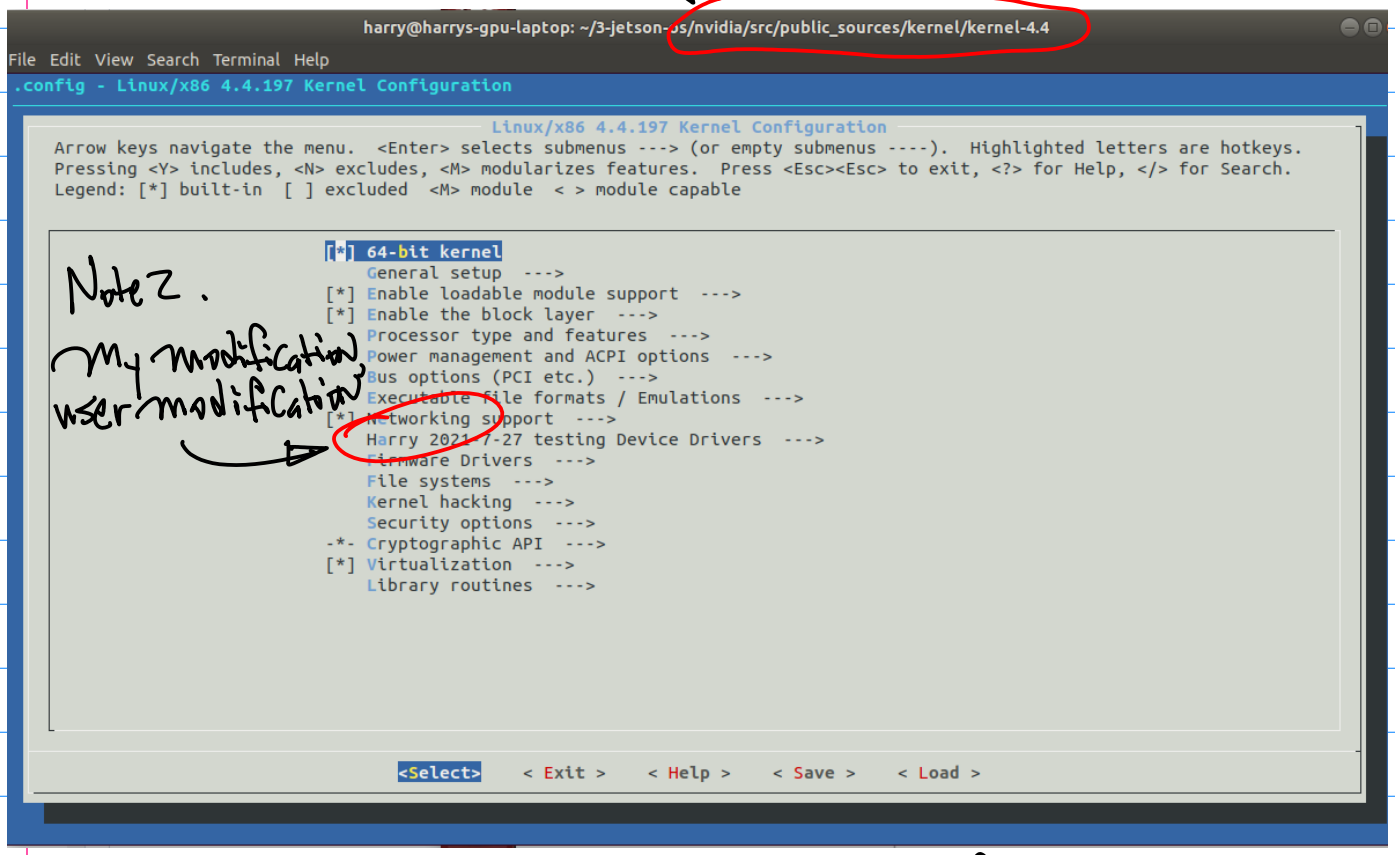
Step2. MenuConfig to Build O.S.

By the manufacturer's Default setting

↓

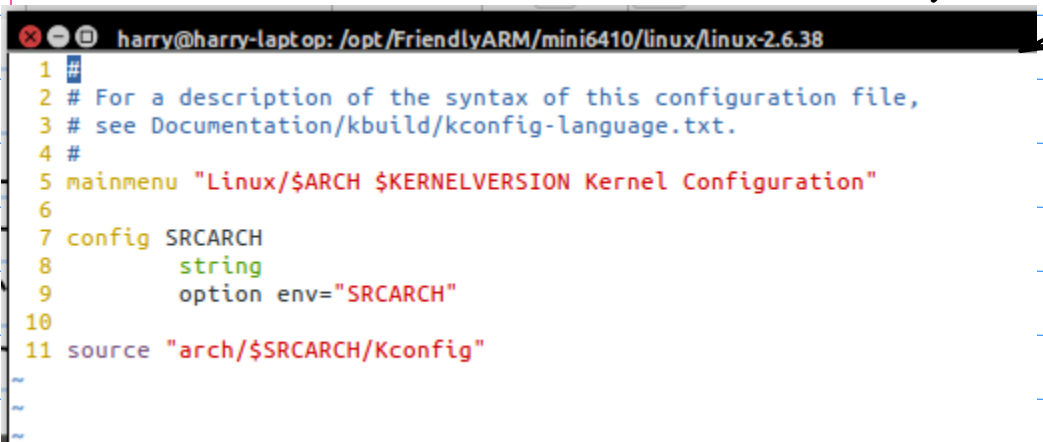
Step 3. Create User-Defined Kernel image.

Note1. menuconfig UI



Note2.
My modification, user modification

Now, similar setup for Sam's Arm11, Check Kconf for UI Customization at the Root folder.



Next Level @ driver fold. Kconfig

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers

```
menu "Device Drivers"

source "drivers/base/Kconfig"

source "drivers/connector/Kconfig"

source "drivers/mtd/Kconfig"

source "drivers/of/Kconfig"

source "drivers/parport/Kconfig"

source "drivers/pnp/Kconfig"

source "drivers/block/Kconfig"

# misc before ide - BLK_DEV_SGIIOC4 depends on SGI_IOC4

source "drivers/misc/Kconfig"

source "drivers/ide/Kconfig"
```

Next Level to "Char" folder. Kconfig.

harry@harry-laptop: /opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char

```
#
# Character device configuration
#

menu "Character devices"

config VT
        bool "Virtual terminal" if EXPERT
        depends on !S390
        select INPUT
        default y
        ---help---
          If you say Y here, you will get support for terminal devices with
          display and keyboard devices. These are called "virtual" because you
          can run several virtual terminals (also called virtual consoles) on
          one physical terminal. This is rather useful, for example one
          virtual terminal can collect system messages and warnings, another
          one can be used for a text-mode user session, and a third could run
          an X session, all in parallel. Switching between virtual terminals
          is done with certain key combinations, usually Alt-<function key>.

          The setterm command ("man setterm") can be used to change the
          properties (such as colors or beeping) of a virtual terminal. The
```

```
#---------------------------------------------
# Harry: Feb 17, 2016
#---------------------------------------------
config MINI6410_I2CSEN_MODULE
        tristate "Harry 2021-2-3: I2C sensor module"
        depends on CPU_S3C6410
        help
          I2C sensor module Feb 17, 2016.
#---------------------------------------------

#---------------------------------------------
# Harry: Here is my modification
#---------------------------------------------
```

2° Provide Screen Capture of your UI.

3° provide Kconfig file which Realizes this function.

4° Hardware Side.
MotorDriveBoard.

Ref: 2023F-109, Readme.

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

Note1. Homework, Due 1 week

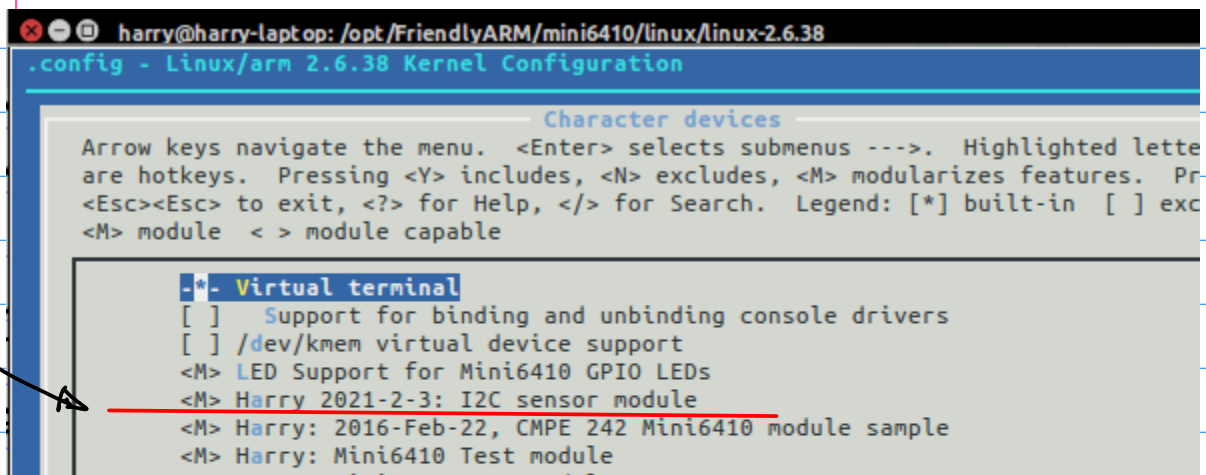
Continuation from the Homework of O.S. Kernel Source Distribution installation.

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.



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           I2C sensor module Feb 17, 2016
```

Option to Consider.

Be Careful, Power budget for A4988 is 1000mA.

A4988 Stepper Motor Driver...

$5.89

Amazon.com

Free shipping

Note2: 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, SFRs (Special Purpose Registers) for GPIO.

→ ARM11, Background Info from LPC;

4° Execute GPIO Code

5° take a photo of your entire System Setup.

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

7° Python Code.

8°. Submission.

2 photos in png/jpeg.

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)

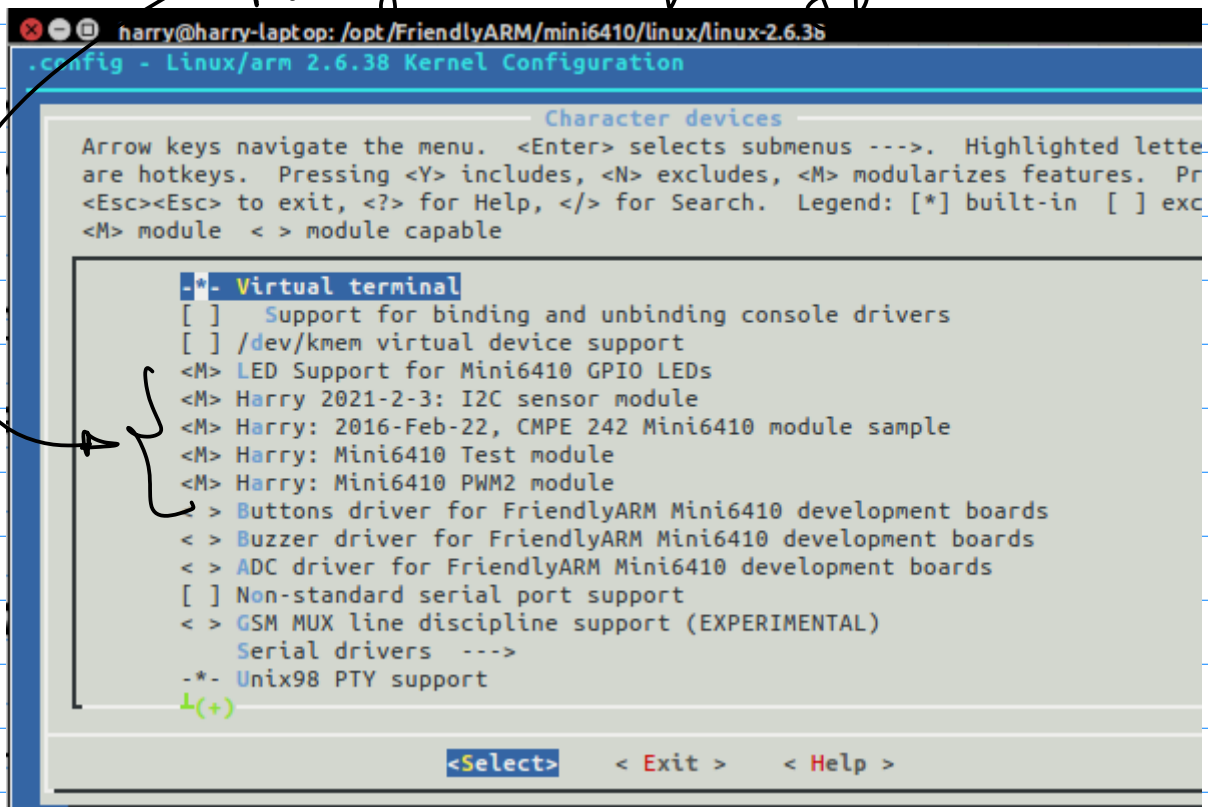
Submission to the CANVAS.

Time to complete: 20 min for the Quize, 15 min. to Prepare the Submission

Example: Continuation on Kconfig.

CmpE244
F2023.

Objectives: To Be Able to customize O.S, 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/drivers/char$ ls
 Kconfig
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/drivers$ cd ..
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38$ ls Kconfig
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/drivers$ ls Kcon
fig
Kconfig
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
 Kconfig
Kconfig
harry@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char$
```

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 manufacture ID + DeviceID.

```
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 #---------------------------------------------
                                                  109,3
```

Note:

```
<M> module  < > module capable
    -*- Virtual terminal
    [ ]   Support for binding and unbinding console driver
    [ ] /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 sampl
Layer: Layer 1
```

Note2: To pre-select, use default "y"

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

```
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".
```

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

$insmod driverName

printk( )

$rmmod driverName

Oct. 4 (Wed).

Consider Debugging the Device Driver(s) from Manufacturer's Distribution.

Example: Embedded Software in this Example: User/Smartphone → Jetson NANO WebServer → User App. Program → Device Driver

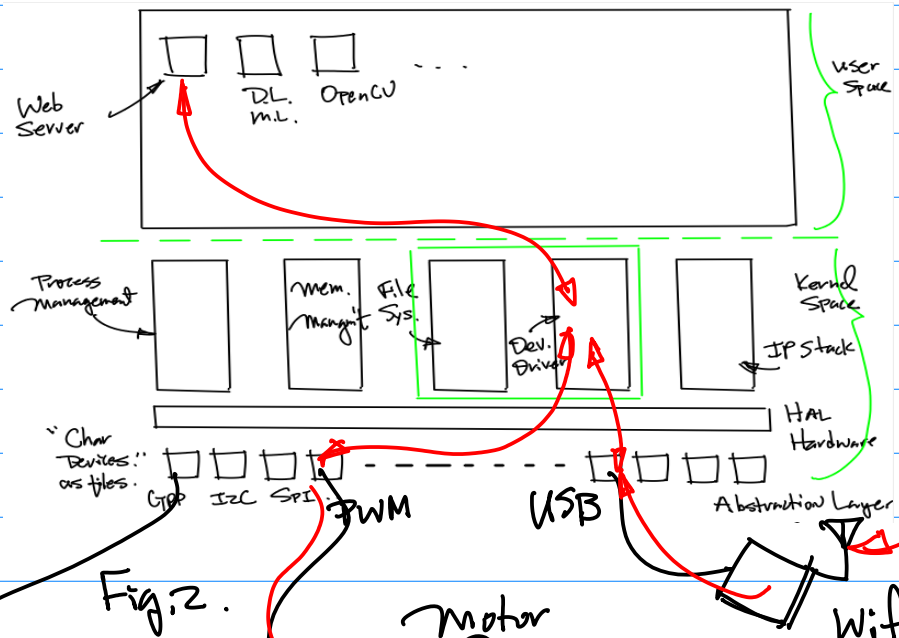
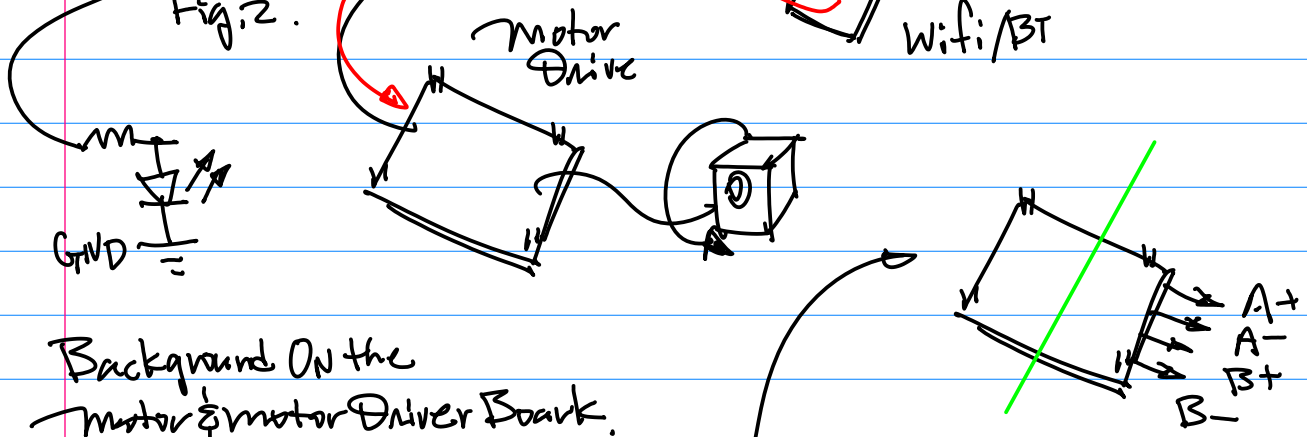


Fig. 2.



Background ON the Motor & Motor Driver Board.

1° Stepper motor pins. (4)

A+, A−, B+, B−.

2° Driver Board Pins

{ Input Group
{ Output Group. (4 pins) A+, A−, B+, B−.

{ ENABLE/Select.
Speed Input pin. PWM { f_{PWM} Duty Cycle
Direction Control. GPP
Others. { Configuration Pins. X
Others...

2^X No. of Config. Patterns

Table1. Connectivity Table Driver Board to the Stepper motor.

Driver Board	NEMA17
A+	A+
A-	A-
B+	B+
B-	B-

Now, for the Configuration of the driver board.

Background ON the Stepper motor Drive Board.

Note: 1 Full step of the Stepper motor.

$2\pi/200 \Rightarrow 360/200$

1.8 degree/step

1 half step. 0.9 Degree/Half Step

1/4 step 0.45 Degree

1/8 step 0.225 Degree

⋮

A typical Driver Board consists of 4 pins for Configuration CON1, CON2, CON3, CON4

Example: f_{pwm} is set to its target frequency, $f_{pwm} = 2 \times 10^3$

And Duty Cycle 5% to 95% Allows the control of the Speed.

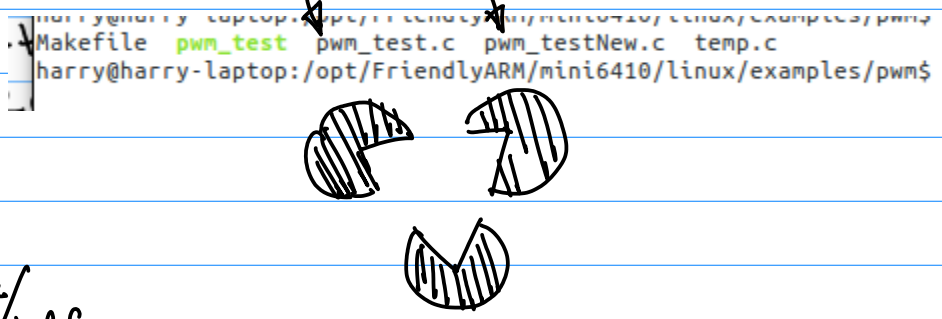
Sample code to Realize this Control function.

a. Set f_{pwm}

b. set & Change Duty Cycle.

Ref: Ref code to Be posted on line. To Be Referenced for your design & debugging

Note1.



```
 99        open_buzzer();
100
101        printf( "\nBUZZER TEST ( PWM Control )\n" );
102        printf( "Press +/- to increase/reduce the frequency of the BUZZER\n" ) ;
103        printf( "Press 'ESC' key to Exit this program\n\n" );
104
105
106        while( 1 )
```

f_{PWM}

Note: set f_{PWM}, But Need to set dutyCycle as well.

```
109
110                set_buzzer_freq(freq);
111                printf( "\tFreq = %d\n", freq );
112
113                key = getch();
114
115                switch(key) {
```

f_{PWM}.

modification of this module?

Add, duty-cycle if possible.

Note: Access to the Device Driver By "fd" (file Descriptor).

```
47
48 static int fd = -1;
49 static void close_buzzer(void);
50 static void open_buzzer(void)
51 {
52         fd = open("/dev/pwm", 0);
53         if (fd < 0) {
54                 perror("open pwm_buzzer
55                 exit(1);
56         }
57
```