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) .

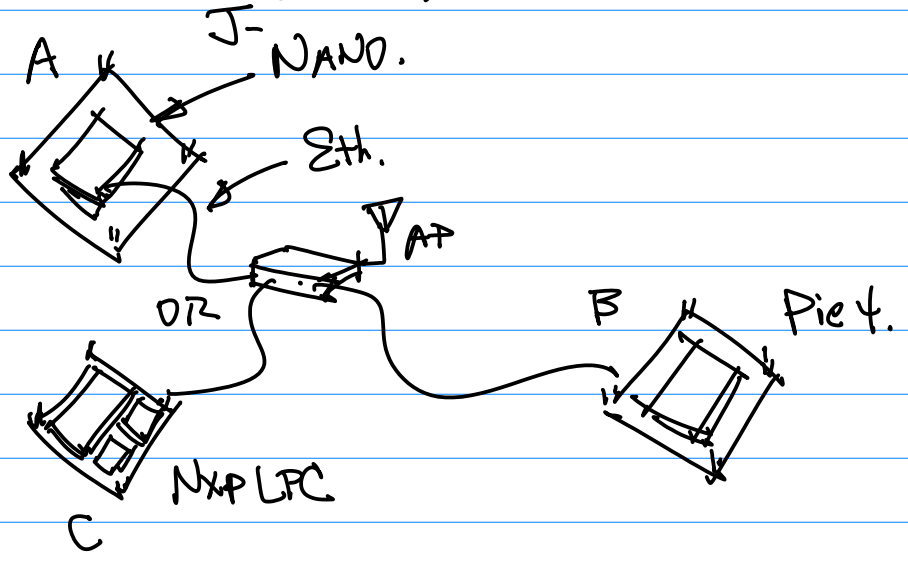
Naming Convention:

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:



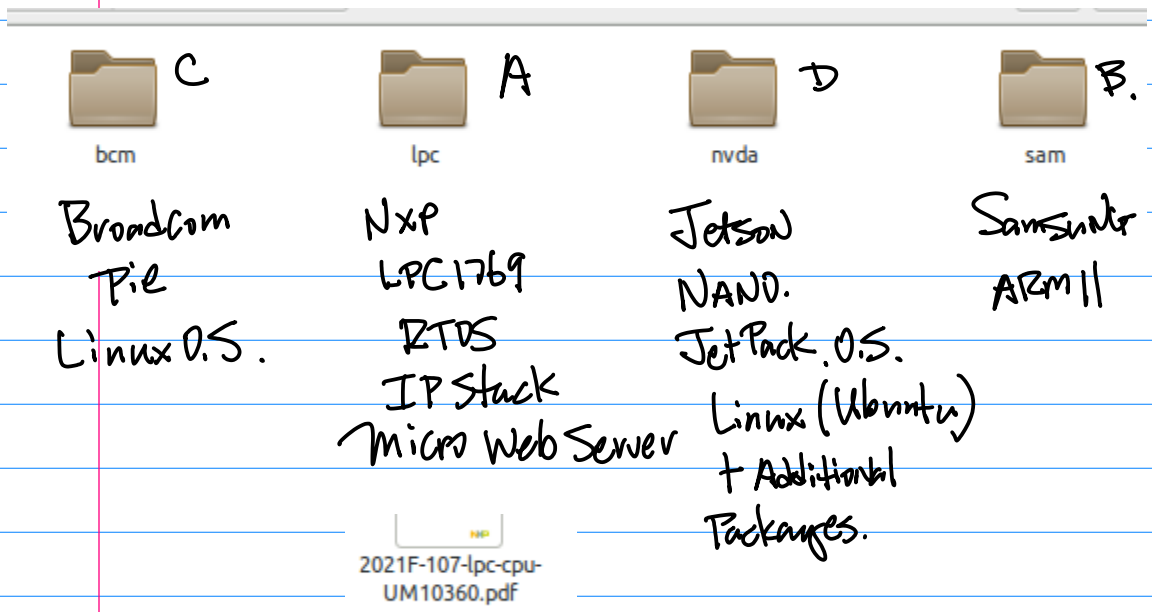
5. Grading Policy

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.



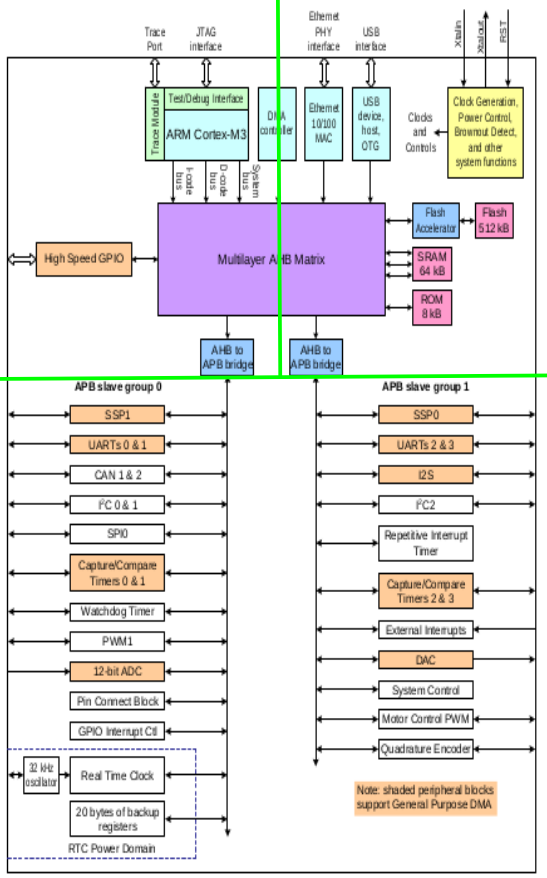


Fig 1. LPC1768 simplified block diagram

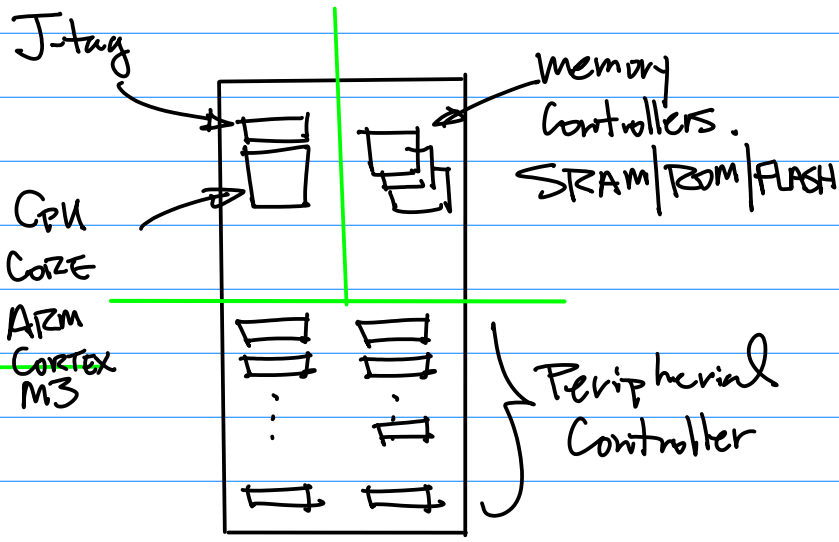


Fig. 1

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:

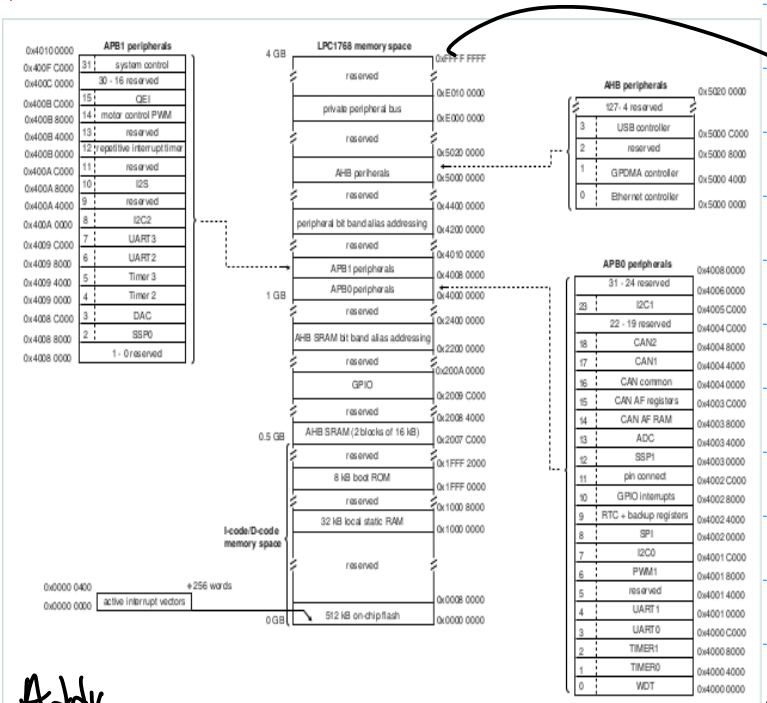
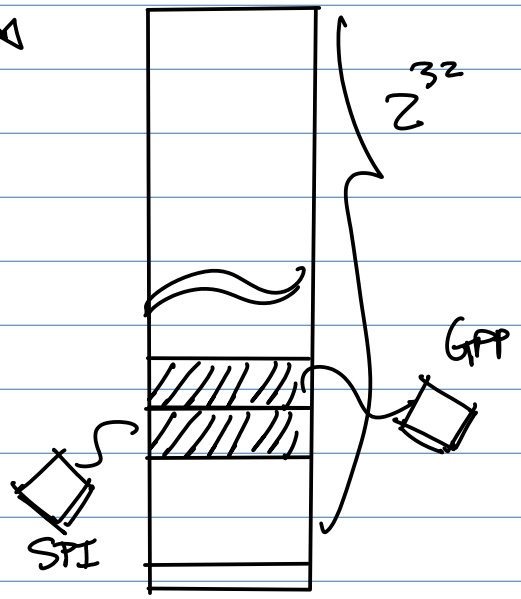


Fig.2



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

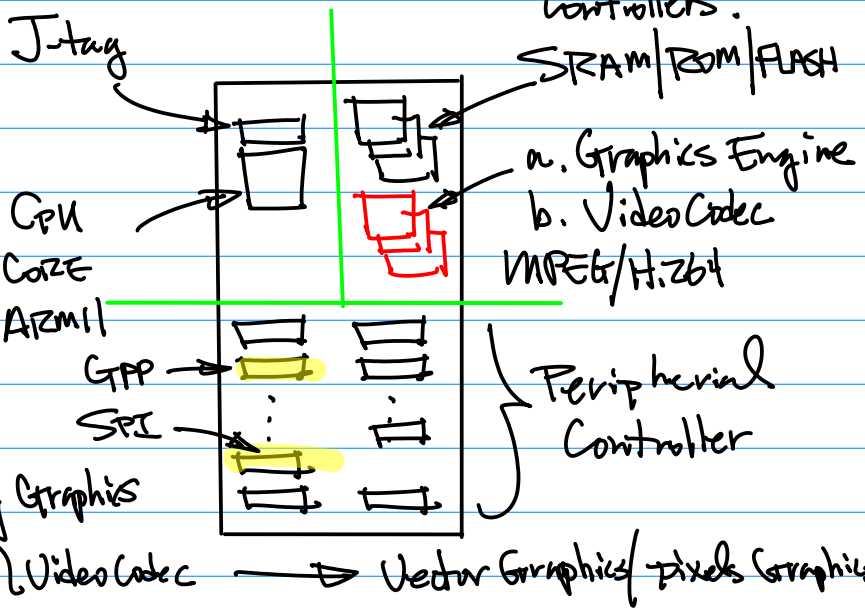


Fig.3.

Differentiation { Graphics
Video Codec → Vector Graphics / pixels Graphics

Example: Connection to (Embedded) Software Architecture

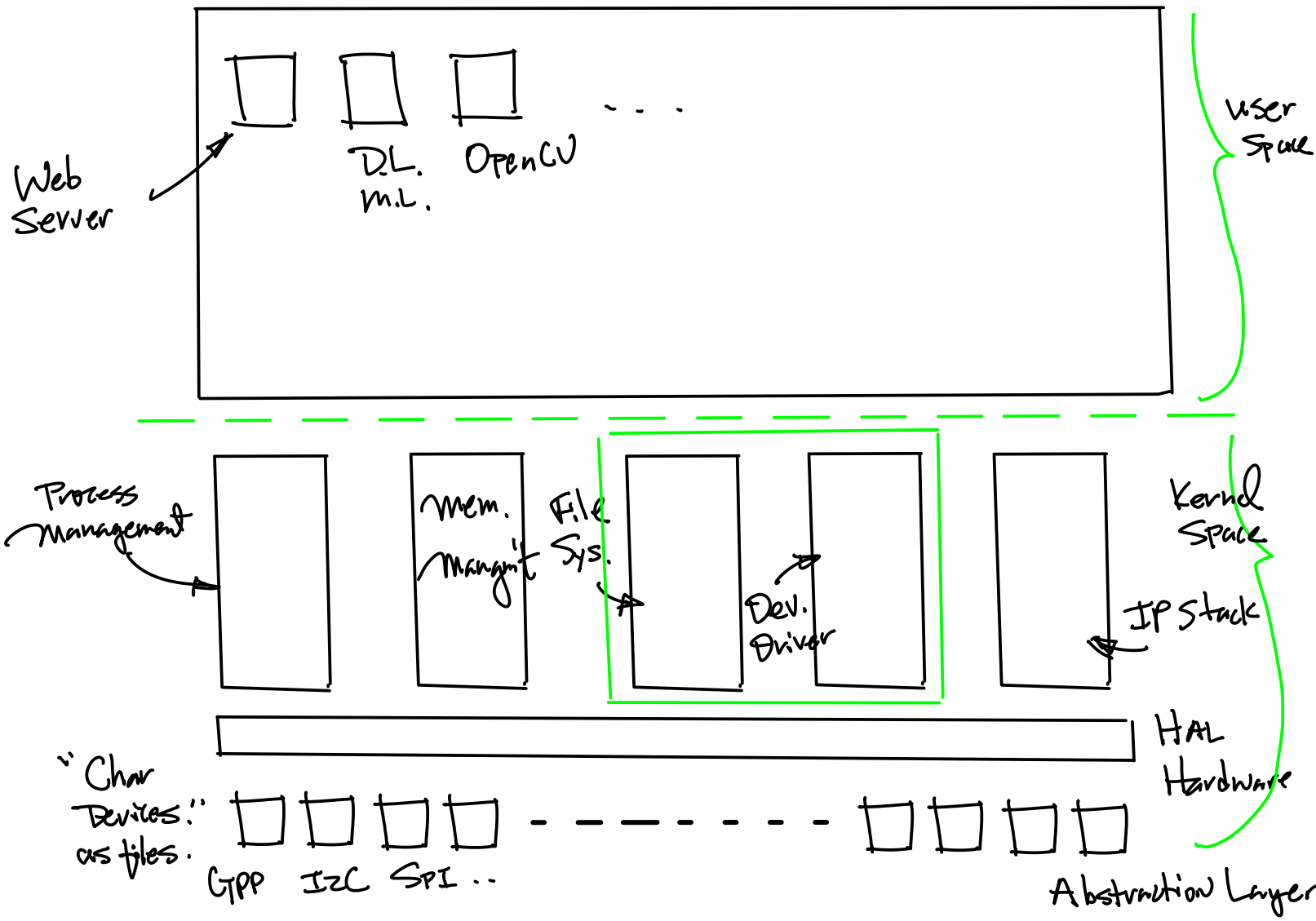


Fig. 1.

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.

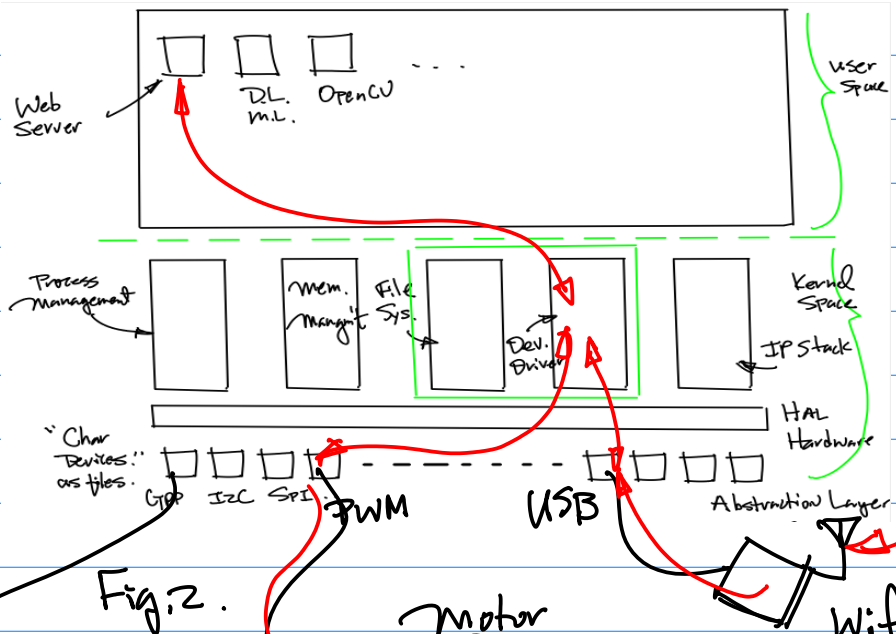
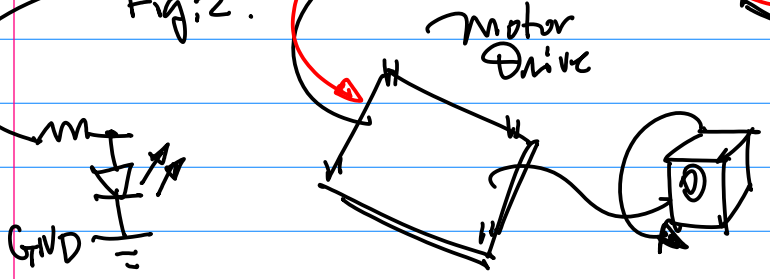


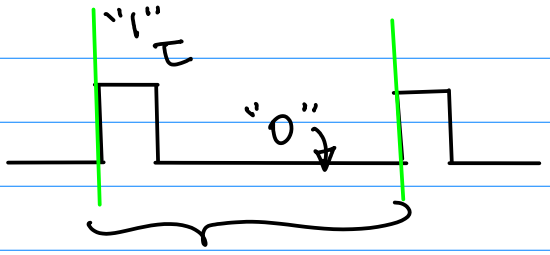
Fig.2.



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.

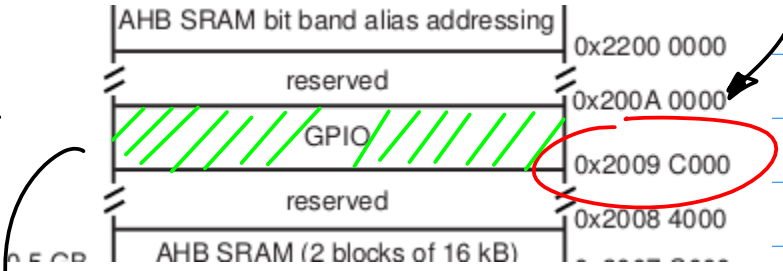
Peripheral Controller
A set of Special purpose Registers.

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.

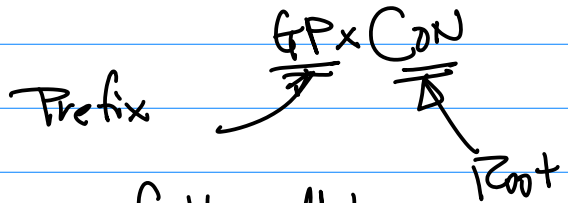
2° Please Bring the target platform to the Class. Next Wednesday.

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.

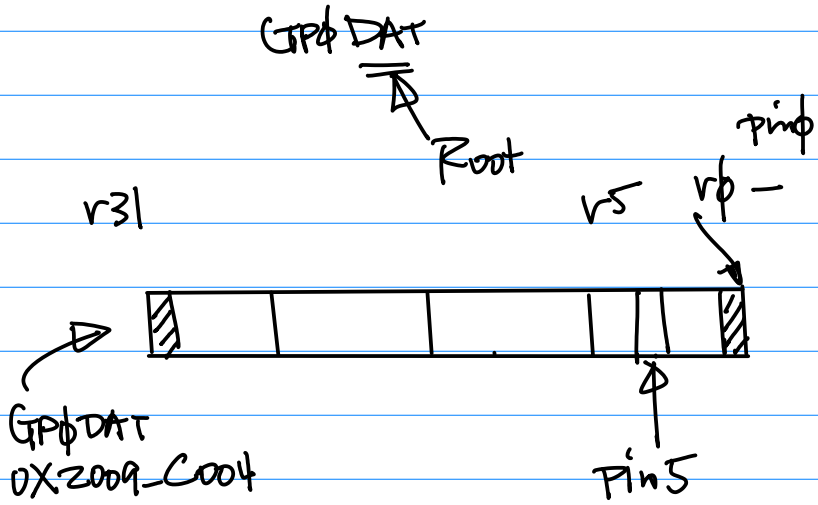
Example: Continued.
Naming Convention of the Control & Configuration Register.
By John Hennessy. Golden Rules
Uniformity,
Regularity,
Orthogonality
"3+3" Naming Convention

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)

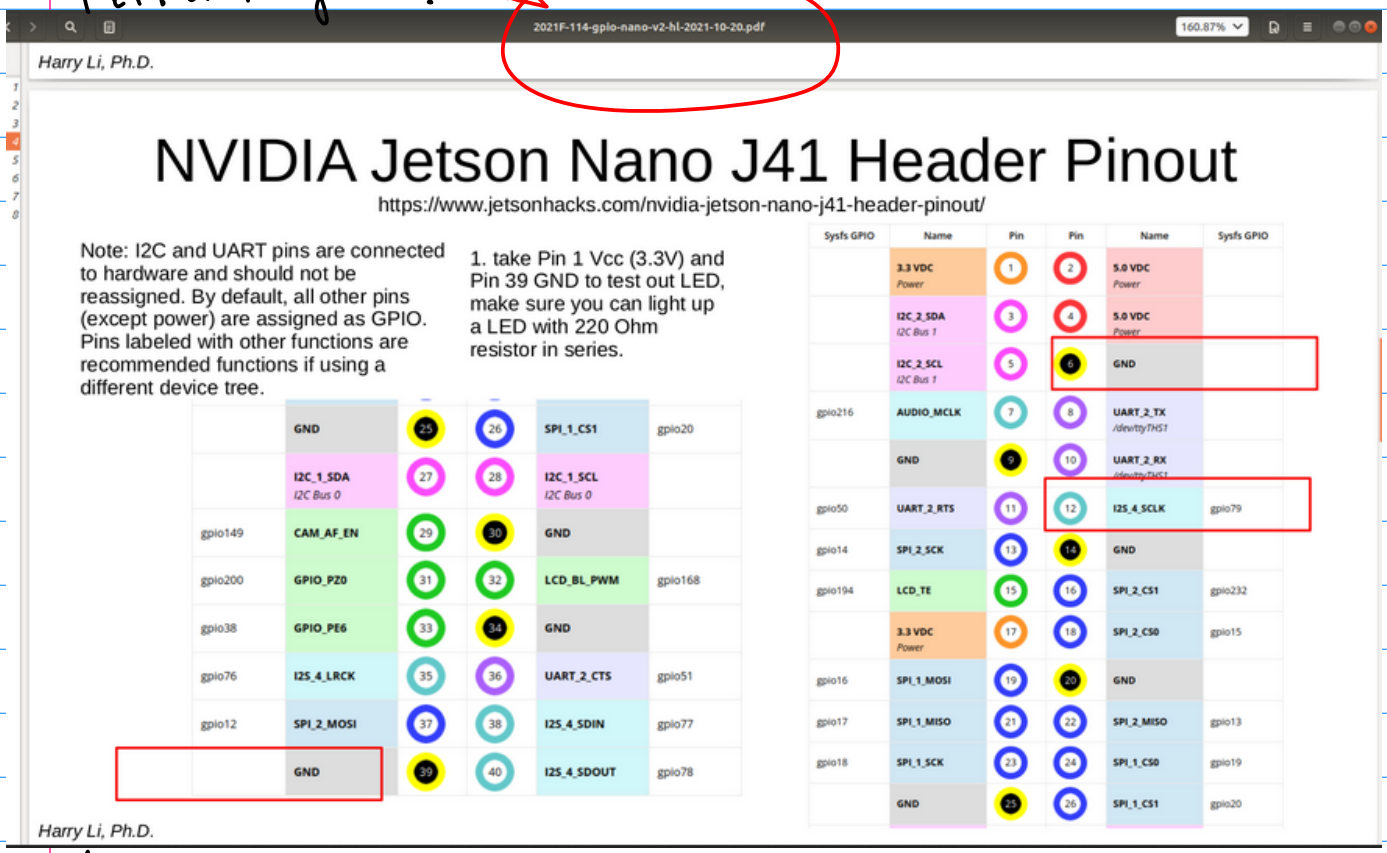
Porting Porting
gcc/g++ → ARM → CORTEX
Porting ↓
Board

Sept 6 (Wed).

Note: 1° Target Board
Inspection:

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

2021F-114~



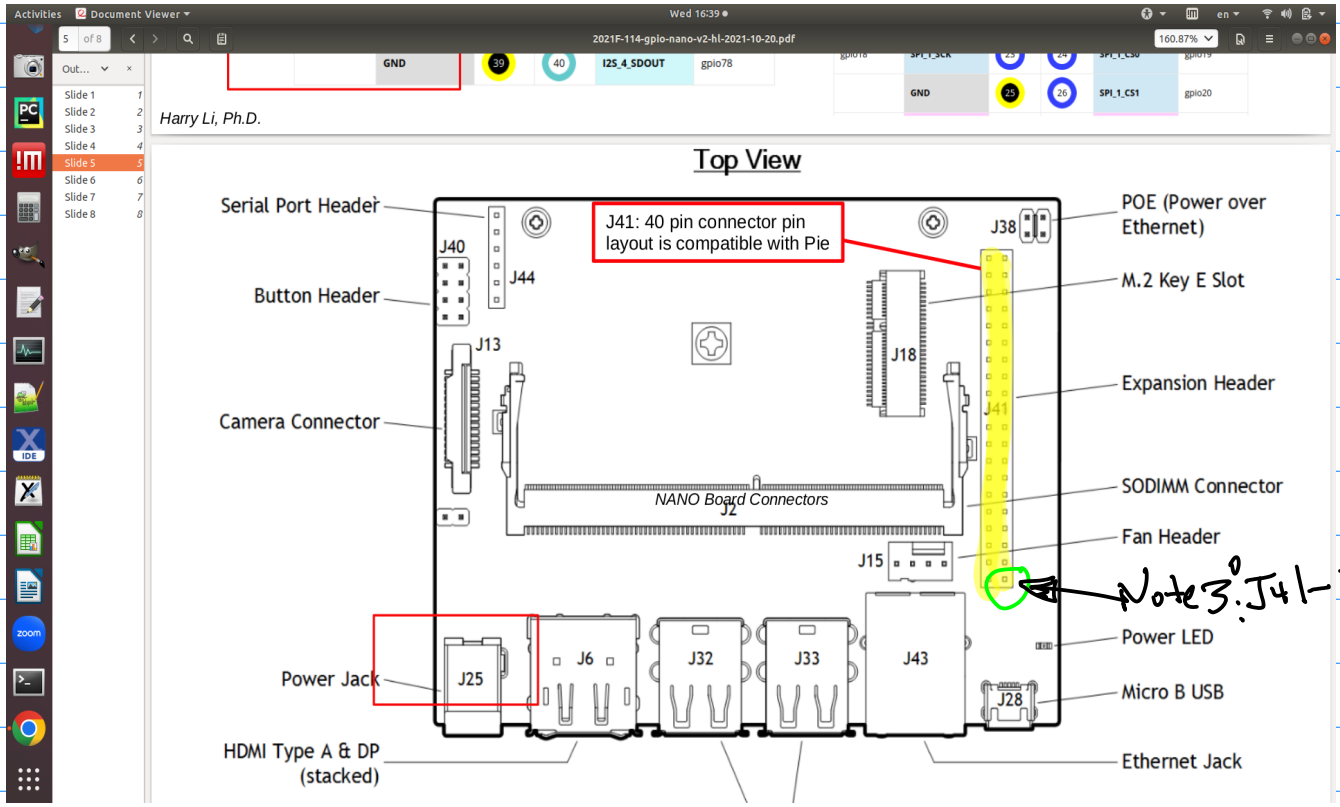
Note: 1° Power Pins
- GND: 6/9/25/39
- Vout 3P3VDC/5VDC Pin1 Pin2,4.
- Vin: J25 (5A or higher @5VDC)

Note2° GPIO

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

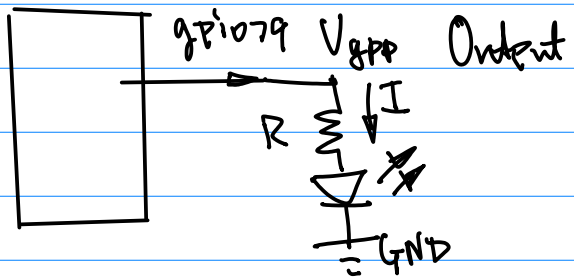
CmpE258
F2023

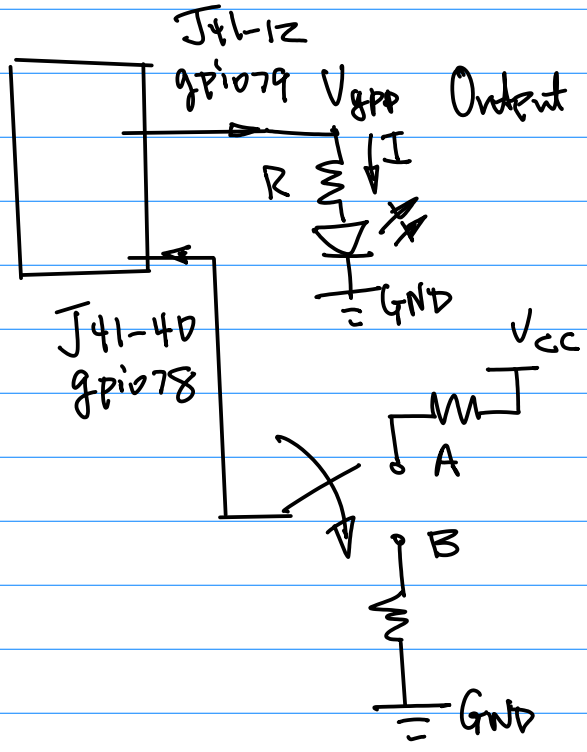
11/



Note 4°: GPIO Input/Output Testing CKT

Build the following Testing CKT.





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

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

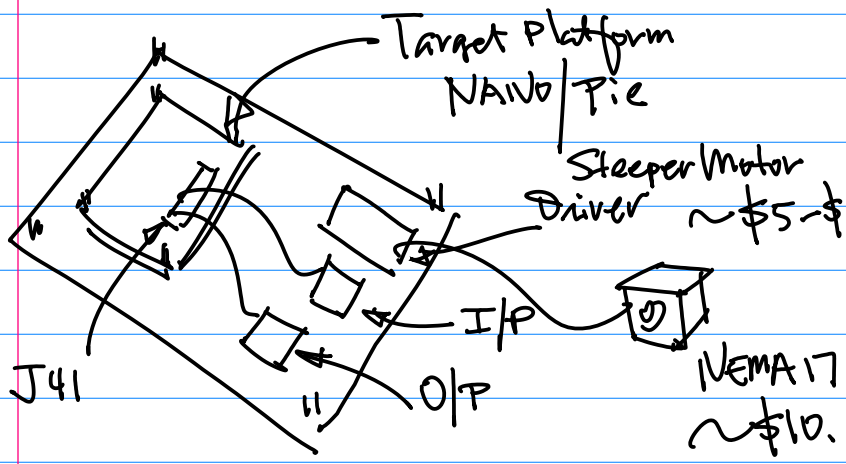
W/O Resistor with Proper selected LED.

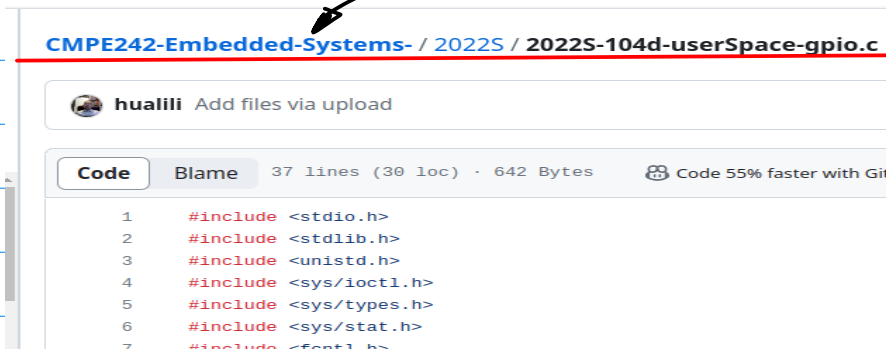
GPIO J41 Pins	CPU Functionality	Note
J41-12	gpio79	Output
J41-40	gpio78	Input

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

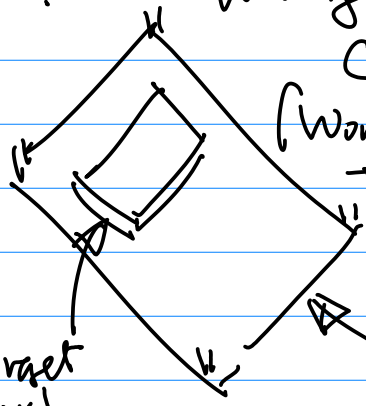
C. Smartphone { iPhone OR Android phone

↔ 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

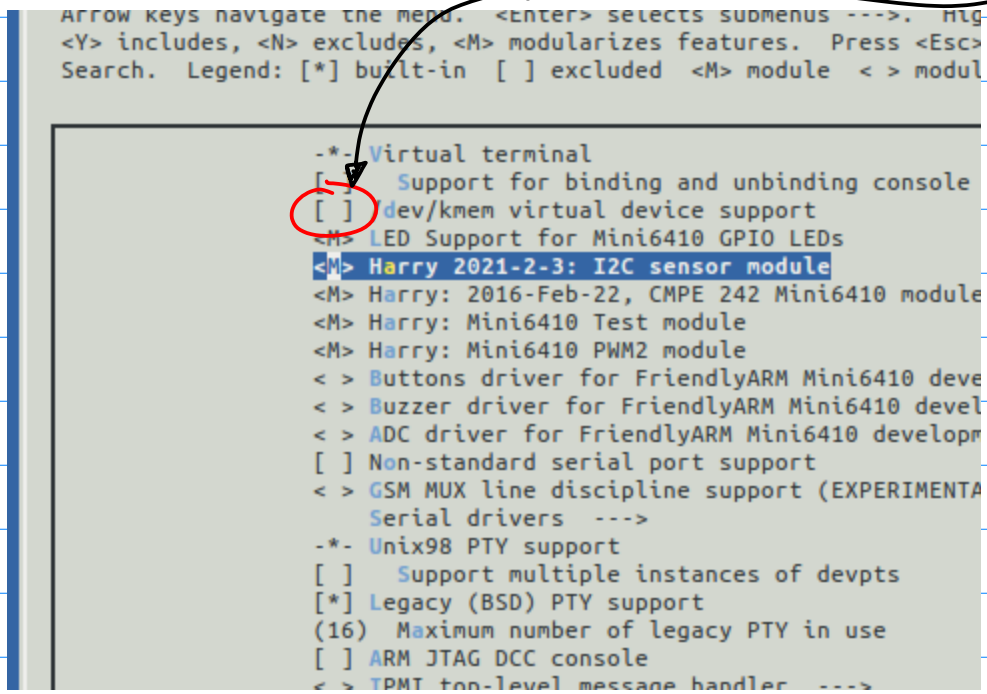
Build Kernel Image using "menuconfig" → NVDA, Broadcom, Smart phones, Android.

Embedded

```
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.

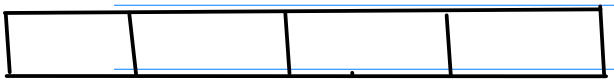
Note1: Connect the code to the CPU Datasheet

Debug

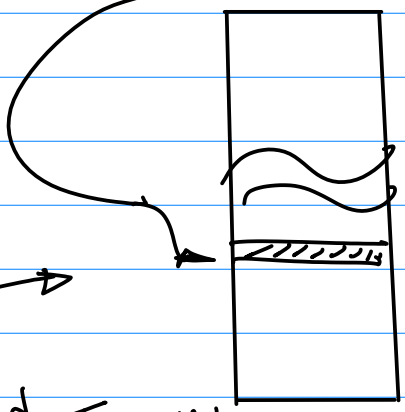
```
66 static int __init dev_init(void)
67 {
68         int ret;
69
70         {
71                 unsigned tmp;
72                 tmp = readl(S3C64XX_GPECON);
73                 tmp = (tmp & ~(0xffffU<<16))|(0x1111U<<16);
74                 writel(tmp, S3C64XX_GPECON);
75
76                 tmp = 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

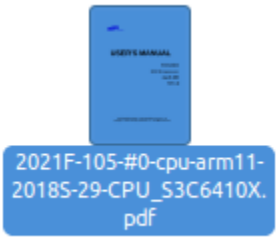
Naming: ID — Peripheral Control (Manufacturer/Part) GPP



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.)

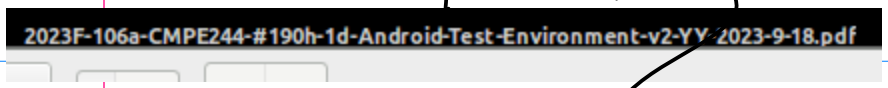
Note: Prepare the Source Distribution Download, to build Kernel Image from the distribution. (1 ~ 1½ week)

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");
```

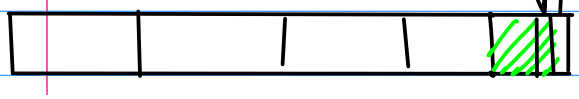
Note: 1°

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.

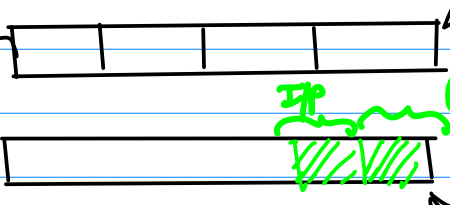
GPECON[3:2] = 0001 (O/P)

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

↓

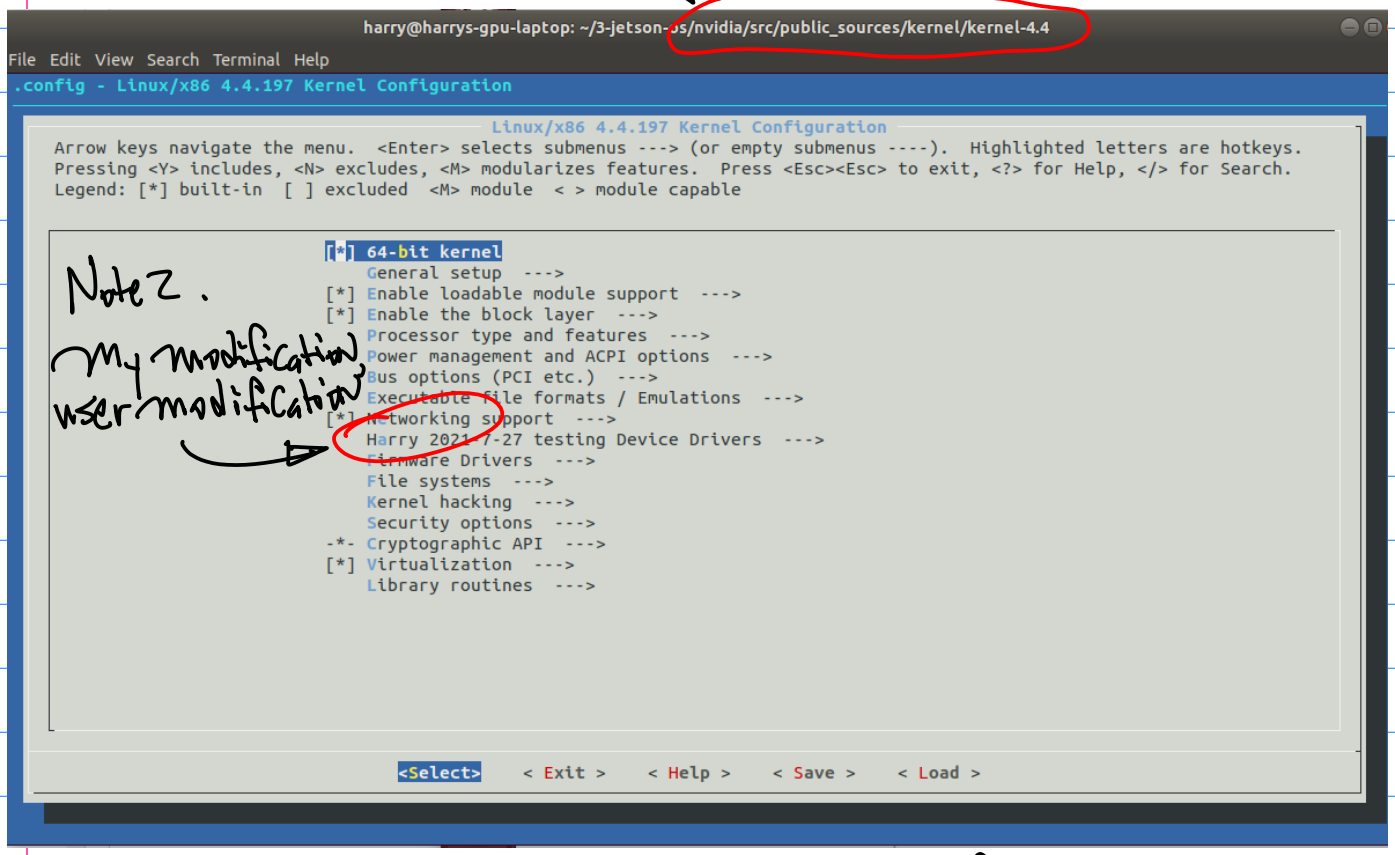
Step2. MenuConfig to Build O.S.

By the manufacturer's Default setting

↓

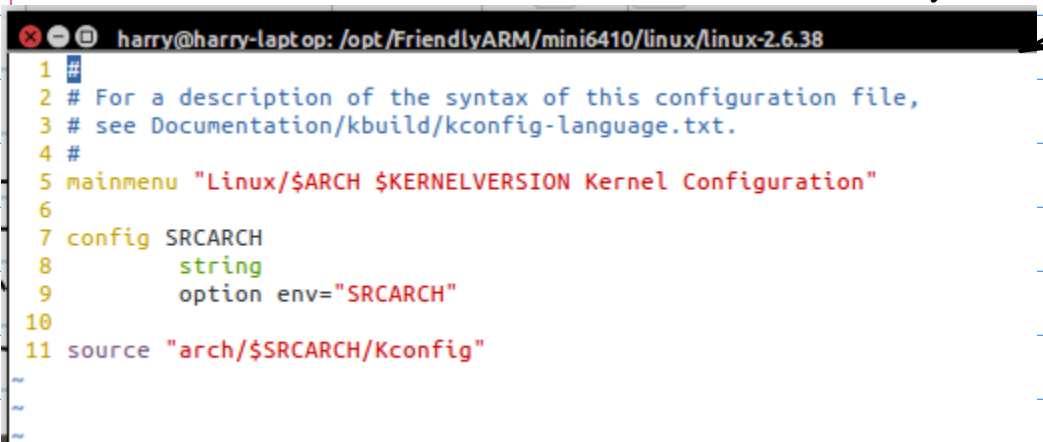
Step 3. Create User-Defined Kernel image.

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

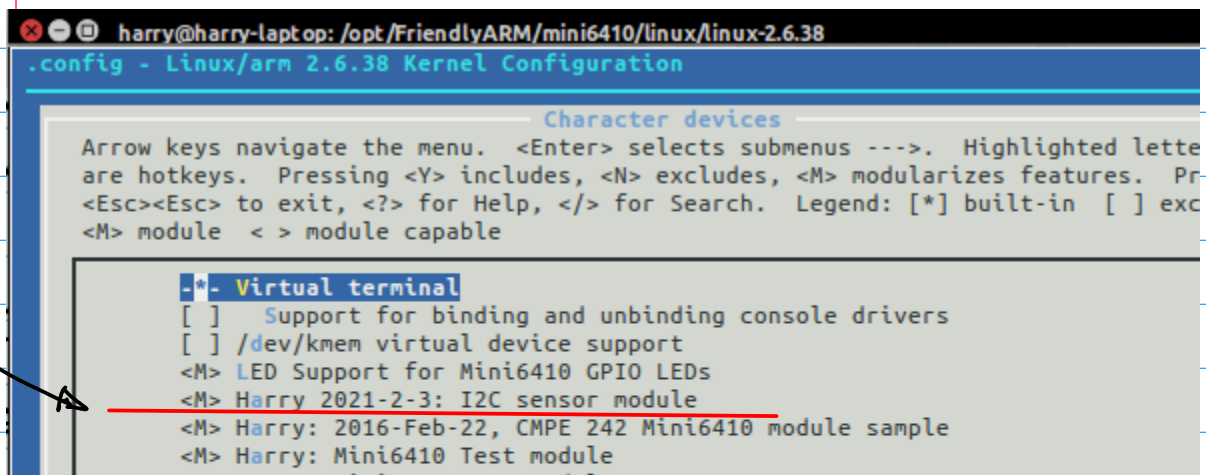
Continuation from the Homework of O.S. Kernel Source Distribution installation.

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)

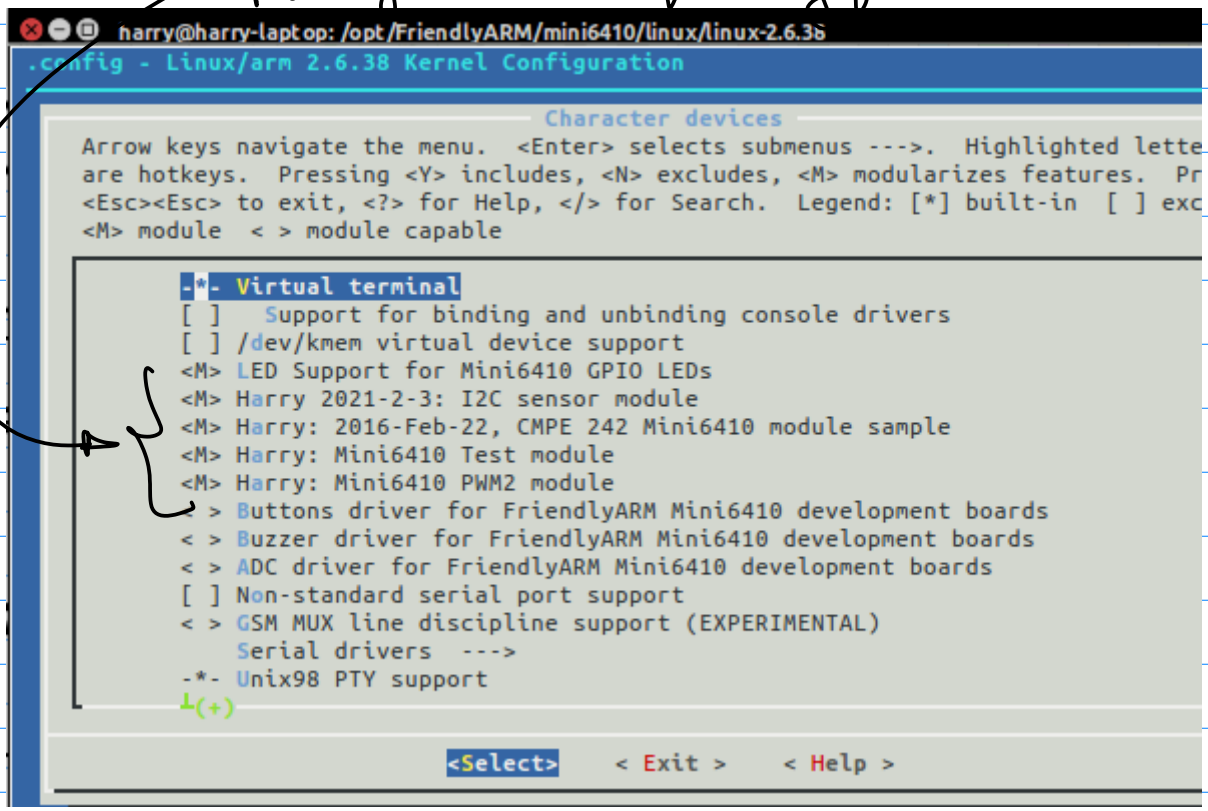
Submission to the CANVAS.

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

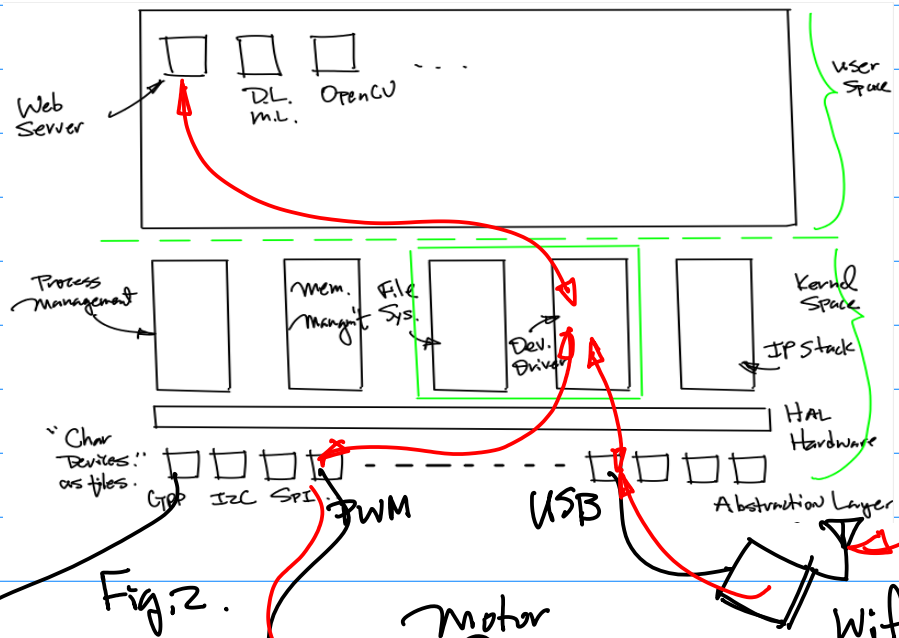
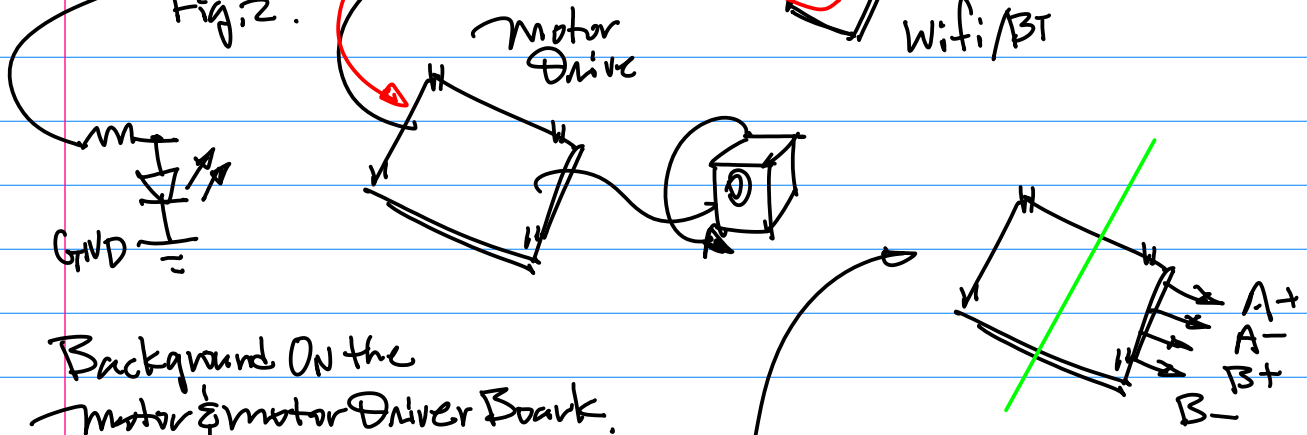


Fig. 2.



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. × 2^x No. of Config. Patterns
  Others...

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.

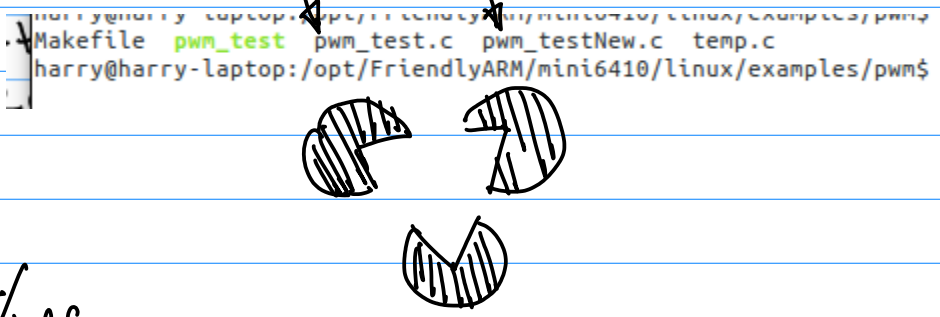
Sample code to Realize this Control function.

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 duty cycle 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
```

Duty Cycle to Be Defined and Selected by user.

Step 1. Modify | Upgrade user Space Code. to Add duty function.

⇓

Step 2. Modify | Upgrade Kernel Space Code, e.g., device Driver.

- Control/Config. f_{PWM}.
- Control/Config. Duty Cycle

⇓

Step 3. Inspection of Control/Configuration Registers, e.g., SFRs to Allow Duty to Be a part of the Driver function.

CPU Datasheet.

Step 4. Option, But if the Name of the Device Driver is Changed Due to

Oct 4th (Monday).

Example: Debugging PWM Driver (ATmel Example).

References with C code have been posted, And IDs are provided to the Class.

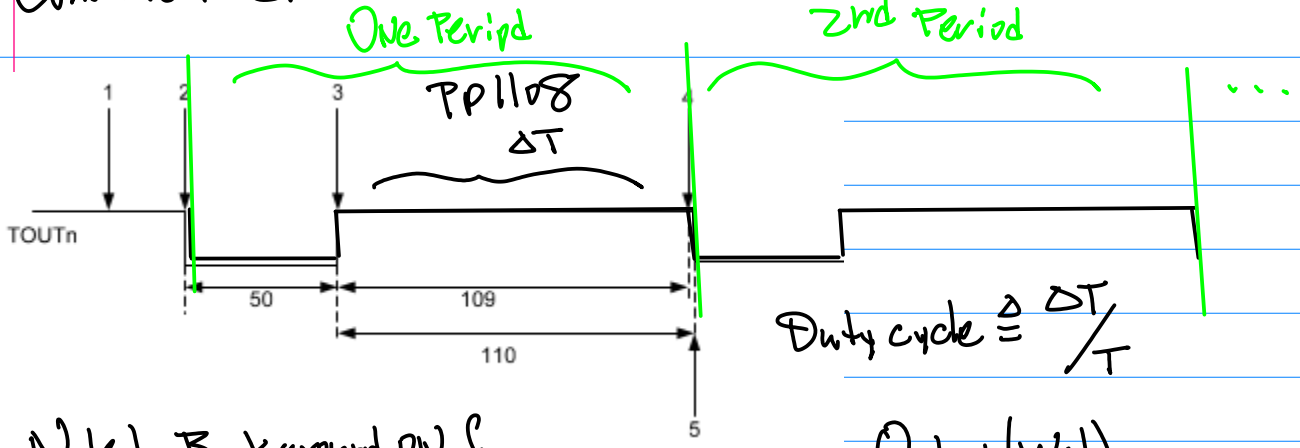
C { user Space Code. Kernel Space Code.

Objective: To modify the Device Driver Code to include Duty function.

To upgrade the user Code to Allow Both f_{PWM} and

the upgrade, then we need to change/update the script in "Kconfig"

Discussion on PWM Peripherial Controller. CPU Datasheet



Note 1. Background on f_{PWM}

System Clock → PCLK → PWM

f_{sys} peripherial clock. f_{PCLK} → f_{PWM}

SPR(s) to Control/Config. PCLK. 1/2, 1/4, 1/8, ... 16 bits.

$$f_{PWM} = \frac{f_{PCLK}}{DUSR * (\quad / Prescaler)} \quad \ldots (1)$$

Not Necessarily tied to A Commercial product.

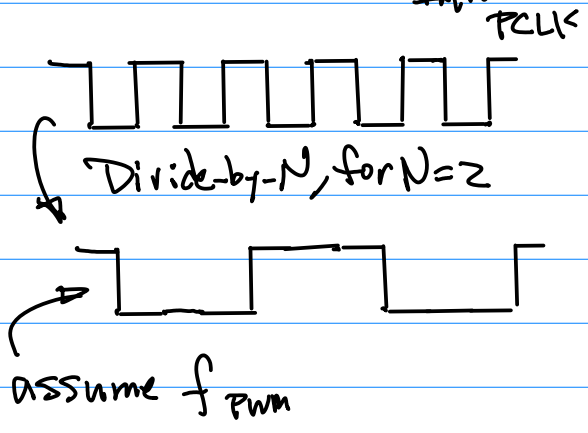
Oct. 11 (Wed)

Continuation on PWM Driver Debugging & update

f_{PWM}

Duty

Note 1. Waveform for Divide-by-N Counter



If N is 8 bits, then
$N \in [0, 255]$

Question: How many different frequency for f_{pwm} can we have with this N?

Alternative to Increase the No. of f_{pwm} by introducing a product
$N = N_1 * N_2$

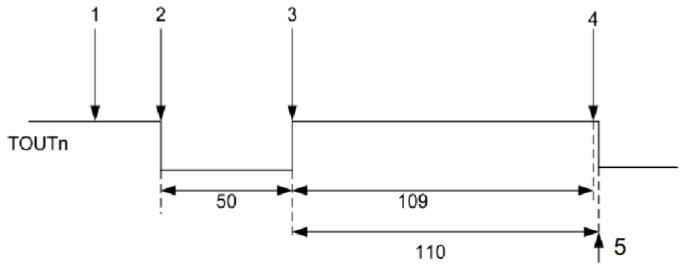
Where N_1 is 8 bit, N_2 is 8 bit
To Achieve the increase the No. of f_{pwm}.

Consider A Duty Cycle Implementation
We need { A Counter { Count up
Count Down.
Interrupt

Ref. from github for CMPE242

2022S-107a-PID-v5-2019-4-25.pdf
2022S-107b-pwm-pid.pdf
2022S-107c-pwm-config-v4-hl-2022-3-3.pdf
2022S-107d-pwm-nano-coding-hl-2021-12-19.pdf
2022S-107e-pwm-waveform-v3-2018-3-4.jpg
2022S-107f-pwm-specialPurposeRegister-v3-20...
2022S-107g-pwm-calculation-v3-2018-3-4.pdf
2022S-107h-spec-motor-drive-WS55-1800 (copy...

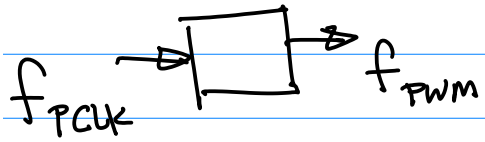
PWM Operation



Example:

1. Initialize the TCNTBn with 159(50+109) and the TCMPBn with 109.
2. Start Timer by setting the start bit and manual update bit off.
The TCNTBn value of 159 is loaded into the down-counter, the output is driven low.
3. When down-counter counts down to the value in the TCMPBn register 109, the output is changed from low to high.
4. When the down-counter reaches 0, the interrupt request is generated.
5. The down-counter is automatically reloaded with TCNTBn, which restarts the cycle.

Example: Given $f_{sys} = 800\,MHz$.
assume $f_{PCLK} = 800/8 = 100\,MHz$
assume $f_{PWM} = 2\,KHz$
find



Try:

$$TCNTB\phi = \frac{f_{PCLK}}{f_{PWM}}$$

TCFG0, TCNTBn and TCMPBn

from the given condition

$$\frac{f_{PCLK}}{f_{PWM}}$$

$f_{PCLK} = 100 \times 10^6$
$f_{PWM} = 2 \times 10^3$

32.4.1.1 TCFG0 (Timer Configuration Register) , pp 1118

Register	Offset	R/W	Description
TCFG0	0x7F006000	R/W	Timer Configuration Register 0 that configures the two 8-bit Prescaler and DeadZone Length

Timer input clock Frequency = PCLK / ({prescaler value + 1}) / {divider value}
{prescaler value} = 1~255
{divider value} = 1, 2, 4, 8, 16, TCLK

$$f_{PWM} = \frac{f_{PCLK}}{(Prescaler+1) * divider} \quad \ldots (1)$$

$$TCNTB\phi = \frac{100 \times 10^6}{2 \times 10^3} = 50 \times 10^3$$

from CPU datasheet, pp. 1120.
So, the Design holds good.

32.4.1.2 TCFG1 (Timer Configuration Register)

Register	Offset	R/W	Description
TCFG1	0x7F006004	R/W	Timer Configuration Register 1 that controls 5 MUX and DMA Mode Select Bit

32.4.1.3 TCON (Timer Control Register)

Register	Offset	R/W	Description
TCON	0x7F006008	R/W	Timer Control Register

32.4.1.5 TCMPB0 (Timer0 Compare Register)

Register	Offset	R/W	Description
TCMPB0	0x7F006010	R/W	Timer 0 Compare Buffer Register

CmpE244
F2023

32.4 SPECIAL FUNCTION REGISTERS

PP1117

32.4.1 REGISTER MAP

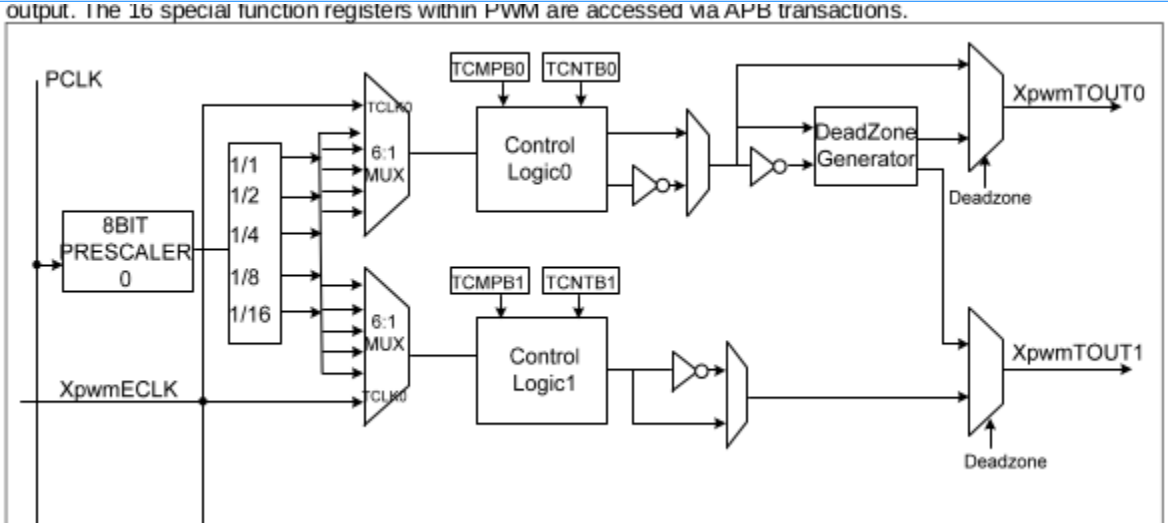
Note1. TCNTB0

CNTB : Count Buffer

Register	Offset	R/W	Description	Reset Value
TCFG0	0x7F006000	R/W	Timer Configuration Register 0 that configures the two 8-bit Prescaler and DeadZone Length	0x0000_0101
TCFG1	0x7F006004	R/W	Timer Configuration Register 1 that controls 5 MUX and DMA Mode Select Bit	0x0000_0000
TCON	0x7F006008	R/W	Timer Control Register	0x0000_0000
TCNTB0	0x7F00600C	R/W	Timer 0 Count Buffer Register	0x0000_0000
TCMPB0	0x7F006010	R/W	Timer 0 Compare Buffer Register	0x0000_0000
TCNTO0	0x7F006014	R	Timer 0 Count Observation Register	0x0000_0000
TCNTB1	0x7F006018	R/W	Timer 1 Count Buffer Register	0x0000_0000
TCMPB1	0x7F00601c	R/W	Timer 1 Compare Buffer Register	0x0000_0000
TCNTO1	0x7F006020	R	Timer 1 Count Observation Register	0x0000_0000
TCNTB2	0x7F006024	R/W	Timer 2 Count Buffer Register	0x0000_0000
TCNTO2	0x7F00602c	R	Timer 2 Count Observation Register	0x0000_0000
TCNTB3	0x7F006030	R/W	Timer 3 Count Buffer Register	0x0000_0000
TCNTO3	0x7F006038	R	Timer 3 Count Observation Register	0x0000_0000
TCNTB4	0x7F00603c	R/W	Timer 4 Count Buffer Register	0x0000_0000
TCNTO4	0x7F006040	R	Timer 4 Count Observation Register	0x0000_0000
TINT_CSTAT	0x7F006044	R/W	Timer Interrupt Control and Status Register	0x0000_0000

Reference for f_{pwm} formula.

output. The 16 special function registers within PWM are accessed via APB transactions.



Oct 16 (Mon).

Note1. Homework Assignment

On CANVAS:
PWM Device Driver + Board Implementation.

CMPE 244
Homework PWM Driver Debugging and Stepper Motor Actuation Demo

Part I:
1. Down the PWM user program and driver program from the class github, see the figure below for the document ID:

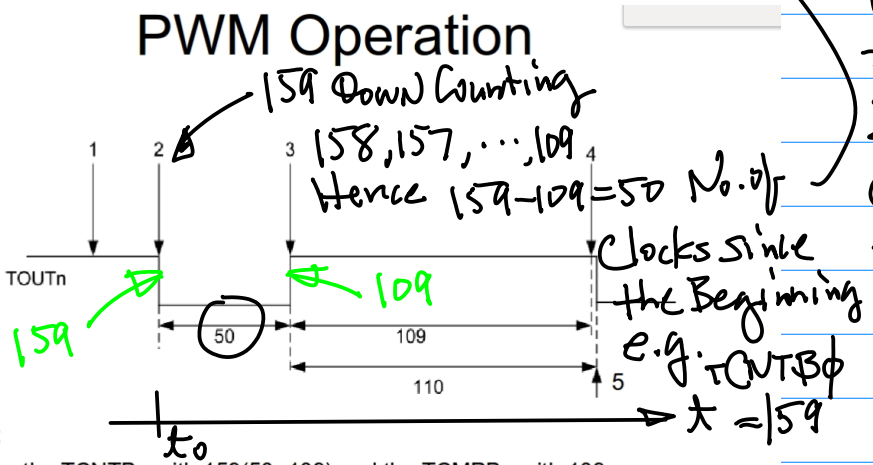
2023F-105d-user-p...	Add files via upload	3 weeks ago
2023F-105e-user-p...	Add files via upload	3 weeks ago
2023F-105f-#20-20...	Add files via upload	3 weeks ago

The github for the pwm user space program:
https://github.com/hualili/CMPE244/blob/main/2023F-105d-user-pwm_test.c

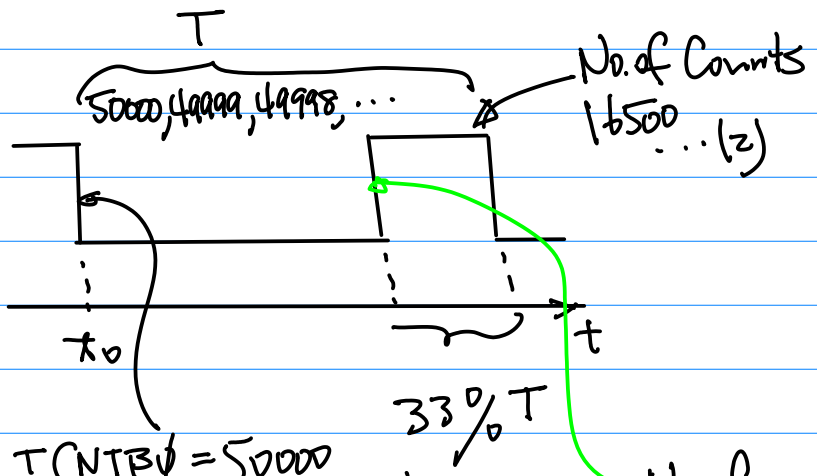
Example: Continuation of the PWM modification.
SPR.
Define the Counts value to Achieve Required duty Cycle.

TCNTBϕ is Compared Against TCMPBϕ = 109.
Then, if equal, trig the Waveform change from the default "0" to high "1"

Design Requirements for the example:
Suppose 33% Duty Cycle.
Initialize TCMPBϕ = ?
∵ TCNTBϕ = 50x10^3
we can Now,
0.33x50x10^3
= 16500
Now, to visualize it, we have



Example:
1. Initialize the TCNTBn with 159(50+109) and the TCMPBn with 109.
2. Start Timer by setting the start bit and manual update bit off.
The TCNTBn value of 159 is loaded into the down-counter, the output is driven low.
3. When down-counter counts down to the value in the TCMPBn register 109, the output is changed from low to high.
4. When the down-counter reaches 0, the interrupt request is generated.
5. The down-counter is automatically reloaded with TCNTBn, which restarts the cycle.



TCNTB0 = 50000 ... (1)

So, the Down-Counting Count

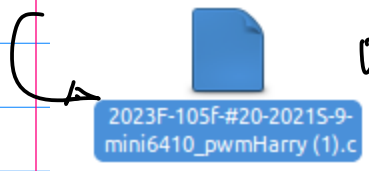
= (1) - (2) ... (3)

= 50000 - 16500 = 33500 e.g.

No. of Counts from Eqn (3).

Therefore, to get the initial value for TCMPB0 = 33500 //

Now, Read C code of Device Driver.



On the Class github.

multi-thread.py | 2022S-103c-#nn_sample_2022.py | 2023F-105f-#20..._pwmHar

```c
/*-------------------------------------------------*
 * Program: mini6410_pwmHarry.c;                   *
 * Modified by: HL;              Date: Feb 10, 2016 *
 * Purpose: To add PWM duty cycle change function   *
 *          by altering CMP compare register.       *
 *-------------------------------------------------*/

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/poll.h>
#include <asm/irq.h>
#include <asm/io.h>
#include <linux/interrupt.h>
#include <asm/uaccess.h>
#include <mach/hardware.h>
#include <plat/regs-timer.h>
#include <mach/regs-irq.h>
#include <asm/mach/time.h>
#include <linux/clk.h>
#include <linux/cdev.h>
#include <linux/device.h>
#include <linux/miscdevice.h>

#include <mach/map.h>
#include <mach/regs-clock.h>
#include <mach/regs-gpio.h>
```

Note1. Driver Name in the User Space,

```
35
36  #define DEVICE_NAME     "pwm"
37
38  #define PWM_IOCTL_SET_FREQ      1
39  #define PWM_IOCTL_STOP          0
40
41  static struct semaphore lock;
42
```

open("drivers/pwm"...)

Initially, Enabled.

Note2. Add the function to change duty cycle.
PWM_IOCTL_SET_DUTY
for Example,

```
41  static struct semaphore lock;
42
43  /* freq:  pclk/50/16/65536 ~ pclk/50/16
44   * if pclk = 50MHz, freq is 1Hz to 62500Hz
45   * human ear : 20Hz~ 20000Hz
46   */
47  //static void PWM_Set_Freq( unsigned long freq );
48  static void PWM_Set_Freq( unsigned long freq, duty )  //Ha
49  {
50      unsigned long tcon;
51      unsigned long tcnt;
52      unsigned long tduty;  //Harry
53      unsigned long tcfg1;
54      unsigned long tcfg0;
55
56      struct clk *clk_p;
57      unsigned long pclk;
58
59      unsigned tmp;
60
61      tmp = readl(S3C64XX_GPFCON);
62      tmp &= ~(0x3U << 28);
63      tmp |=  (0x2U << 28);
64      writel(tmp, S3C64XX_GPFCON);
65
66      tcon = __raw_readl(S3C_TCON);
67      tcfg1 = __raw_readl(S3C_TCFG1);
68      tcfg0 = __raw_readl(S3C_TCFG0);
69
```

Note3. Preserve the module Name for minimum modification
OR, update to
PWM_Set_Freq_Duty( ).
New Parameter for the Duty Cycle is added.

32.4.1.3 TCON (Timer Control Register) PP 1119

Register	Offset	R/W	Descr
TCON	0x7F006008	R/W	Timer Control Register

TCON	Bit	R/W	
Reserved	[31:23]	R	Reserved Bits
Timer 4 Auto Reload on/off	[22]	R/W	0: One-Shot 1:

Read Datasheet, for Example "Oneshot" V.S. "Auto-Reload" etc.

Oct.18 (Wed)

Note: Road map of the Class

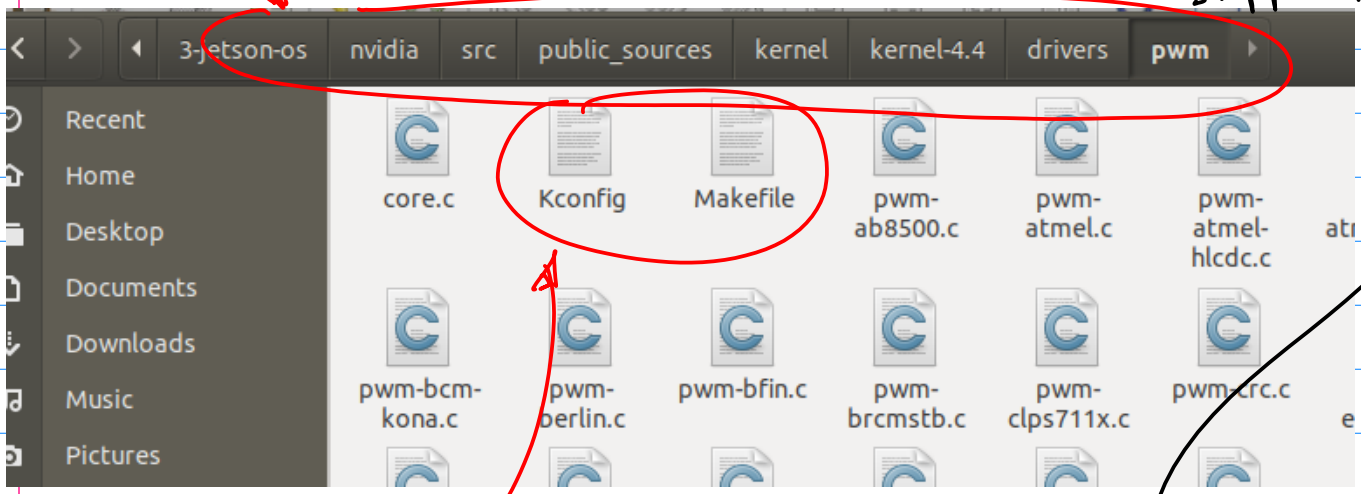
Introduction.
CPU/SPR/Driver
Device Driver Debug/Development
PWM.

In About a week Time frame, Oct.25 (Wed)
in-Class Brief Presentation (update, e.g. one-paragraph description for the Semester Long Project.)
2 person Team OR individual.

CmpE244

F2023

Q&A: Note1. ~~Device Driver~~ folder, ~~PWM Device Driver~~, Target ~~Platform Device Driver~~: Note2. Kernel-4.4 is chosen for my need with Ubuntu 18.04 on Jetson NAND. To Support AI.



Per NVDA Requirements.

Note3. Two files.

Kconfig → UI for menuconfig

↓

Script per the Developer choice To Build kernel image and Device Drivers.

Makefile.

Open ▾ Kconfig ~/3-jetson-os/nvidia/src/public_sources/

```
1 menuconfig PWM
2         bool "Pulse-Width Modulation (PWM) Support"
3         help
4           Generic Pulse-Width Modulation (PWM) support.
5
6           In Pulse-Width Modulation, a variation of the width of pulses
7           in a rectangular pulse signal is used as a means to alter the
8           average power of the signal. Applications include efficient
9           power delivery and voltage regulation. In computer systems,
10          PWMs are commonly used to control fans or the brightness of
11          display backlights.
12
13          This framework provides a generic interface to PWM devices
14          within the Linux kernel. On the driver side it provides an API
15          to register and unregister a PWM chip, an abstraction of a PWM
16          controller, that supports one or more PWM devices. Client
17          drivers can request PWM devices and use the generic framework
```

Some support Target CPU.

```
44 config PWM_ATMEL
45         tristate "Atmel PWM support"
46         depends on ARCH_AT91 || AVR32
47         help
48           Generic PWM framework driver for Atmel SoC.
49
50           To compile this driver as a module, choose M here: the module
51           will be called pwm-atmel.
```

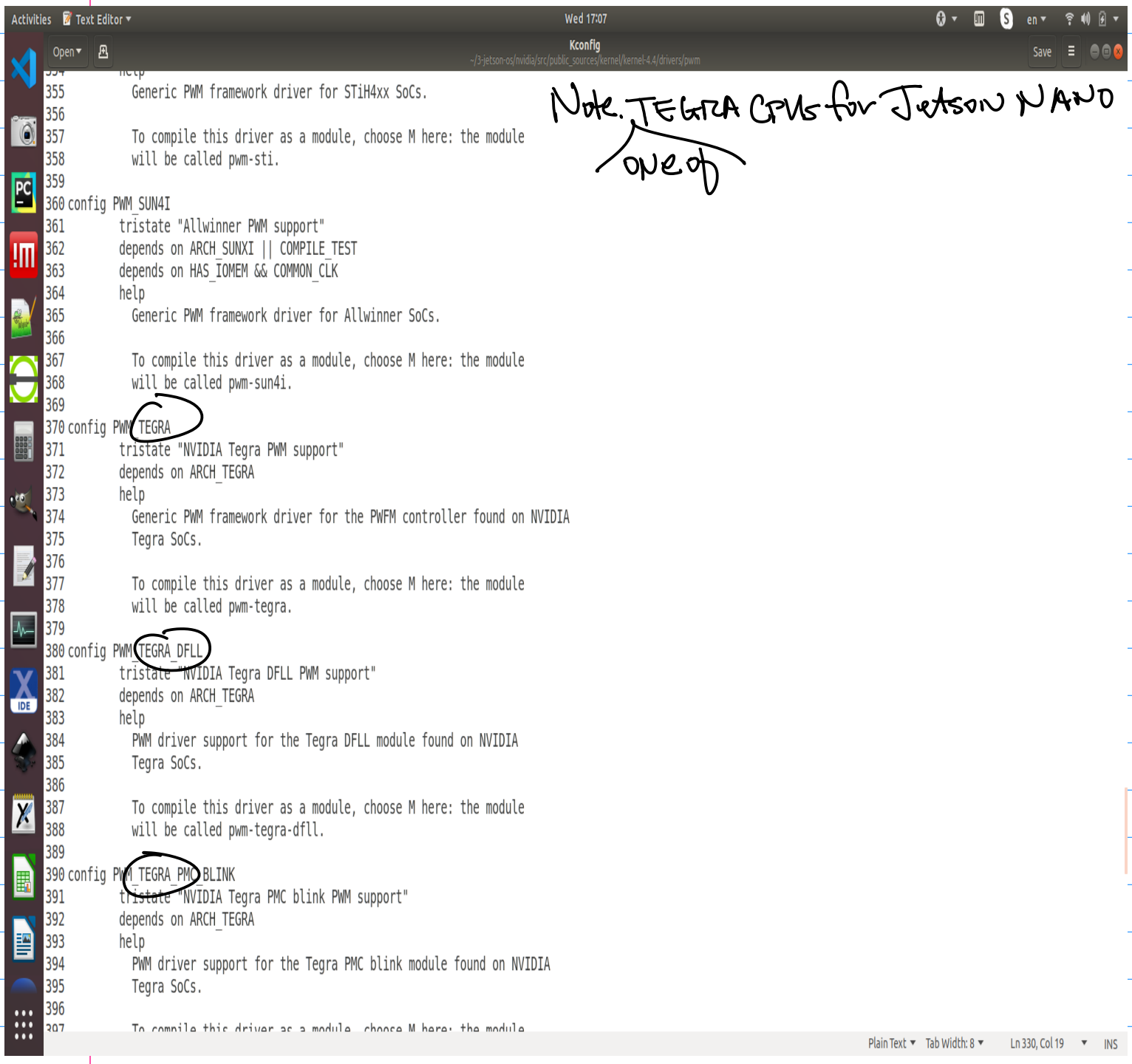
Note: BroadCom

```
77 config PWM_BCM_KONA
78         tristate "Kona PWM support"
79         depends on ARCH_BCM_MOBILE
80         help
81           Generic PWM framework driver for Broadcom Kona PWM block.
82
83           To compile this driver as a module, choose M here: the module
84           will be called pwm-bcm-kona.
85
86 config PWM_BCM2835
87         tristate "BCM2835 PWM support"
88         depends on ARCH_BCM2835
89         help
90           PWM framework driver for BCM2835 controller (Raspberry Pi)
```

Graphics Acceleration Unit.

Note: CPU from NXP.

```
202 config PWM_LPC18XX_SCT
203         tristate "LPC18xx/43xx PWM/SCT support"
204         depends on ARCH_LPC18XX
205         help
206           Generic PWM framework driver for NXP LPC18
207           supports 16 channels.
208           A maximum of 15 channels can be requested
209           must have the same period.
210
211           To compile this driver as a module, choose
212           will be called pwm-lpc18xx-sct.
213
214 config PWM_LPC32XX
215         tristate "LPC32XX PWM support"
216         depends on ARCH_LPC32XX
217         help
218           Generic PWM framework driver for LPC32XX.
```


Activities
Text Editor
Wed 17:07
en
Open
Kconfig
~/3-jetson-os/nvidia/src/public_sources/kernel/kernel-4.4/drivers/pwm
Save
Note. TEGRA GPUs for Jetson NANO
one of
355 Generic PWM framework driver for STiH4xx SoCs.
356
357 To compile this driver as a module, choose M here: the module
358 will be called pwm-sti.
359
360 config PWM_SUN4I
361 tristate "Allwinner PWM support"
362 depends on ARCH_SUNXI || COMPILE_TEST
363 depends on HAS_IOMEM && COMMON_CLK
364 help
365 Generic PWM framework driver for Allwinner SoCs.
366
367 To compile this driver as a module, choose M here: the module
368 will be called pwm-sun4i.
369
370 config PWM_TEGRA
371 tristate "NVIDIA Tegra PWM support"
372 depends on ARCH_TEGRA
373 help
374 Generic PWM framework driver for the PWFM controller found on NVIDIA
375 Tegra SoCs.
376
377 To compile this driver as a module, choose M here: the module
378 will be called pwm-tegra.
379
380 config PWM_TEGRA_DFLL
381 tristate "NVIDIA Tegra DFLL PWM support"
382 depends on ARCH_TEGRA
383 help
384 PWM driver support for the Tegra DFLL module found on NVIDIA
385 Tegra SoCs.
386
387 To compile this driver as a module, choose M here: the module
388 will be called pwm-tegra-dfll.
389
390 config PWM_TEGRA_PMC_BLINK
391 tristate "NVIDIA Tegra PMC blink PWM support"
392 depends on ARCH_TEGRA
393 help
394 PWM driver support for the Tegra PMC blink module found on NVIDIA
395 Tegra SoCs.
396
Plain Text
Tab Width: 8
Ln 330, Col 19
INS

CmpE244

F2023

Note: makefile as the result output from Kconf. Driver(s) Built per Developer Selection. 44

File Edit View Search Terminal Help

```
subdir-ccflags-y += -Werror

obj-$(CONFIG_PWM)               += core.o
obj-$(CONFIG_PWM_SYSFS)         += sysfs.o
obj-$(CONFIG_PWM_AB8500)        += pwm-ab8500.o
obj-$(CONFIG_PWM_ATMEL)         += pwm-atmel.o
obj-$(CONFIG_PWM_ATMEL_HLCDC_PWM)       += pwm-atmel-hlcdc.o
obj-$(CONFIG_PWM_ATMEL_TCB)     += pwm-atmel-tcb.o
obj-$(CONFIG_PWM_BCM_KONA)      += pwm-bcm-kona.o
obj-$(CONFIG_PWM_BCM2835)       += pwm-bcm2835.o
obj-$(CONFIG_PWM_BERLIN)        += pwm-berlin.o
obj-$(CONFIG_PWM_BFIN)          += pwm-bfin.o
obj-$(CONFIG_PWM_BRCMSTB)       += pwm-brcmstb.o
obj-$(CONFIG_PWM_CLPS711X)      += pwm-clps711x.o
obj-$(CONFIG_PWM_CRC)           += pwm-crc.o
obj-$(CONFIG_PWM_EP93XX)        += pwm-ep93xx.o
obj-$(CONFIG_PWM_FSL_FTM)       += pwm-fsl-ftm.o
obj-$(CONFIG_PWM_IMG)           += pwm-img.o
obj-$(CONFIG_PWM_IMX)           += pwm-imx.o
obj-$(CONFIG_PWM_JZ4740)        += pwm-jz4740.o
obj-$(CONFIG_PWM_LP3943)        += pwm-lp3943.o
obj-$(CONFIG_PWM_LPC18XX_SCT)   += pwm-lpc18xx-sct.o
obj-$(CONFIG_PWM_LPC32XX)       += pwm-lpc32xx.o
obj-$(CONFIG_PWM_LPSS)          += pwm-lpss.o
obj-$(CONFIG_PWM_LPSS_PCI)      += pwm-lpss-pci.o
obj-$(CONFIG_PWM_LPSS_PLATFORM) += pwm-lpss-platform.o
obj-$(CONFIG_PWM_MTK_DISP)      += pwm-mtk-disp.o
obj-$(CONFIG_PWM_MXS)           += pwm-mxs.o
obj-$(CONFIG_PWM_PCA9685)       += pwm-pca9685.o
obj-$(CONFIG_PWM_PUV3)          += pwm-puv3.o
obj-$(CONFIG_PWM_PXA)           += pwm-pxa.o
obj-$(CONFIG_PWM_RCAR)          += pwm-rcar.o
obj-$(CONFIG_PWM_RENESAS_TPU)   += pwm-renesas-tpu.o
obj-$(CONFIG_PWM_ROCKCHIP)      += pwm-rockchip.o
```

Note: Inspection of ~~PWM Driver~~ one of

Open ▾ pwm-tegra.c
~/3-jetson-os/nvidia/src/public_sources/kern

```
1  /*
2   * drivers/pwm/pwm-tegra.c
3   *
4   * Tegra pulse-width-modulation controller driver
5   *
6   * Copyright (c) 2010-2017, NVIDIA CORPORATION. All rights reserved.
7   * Based on arch/arm/plat-mxc/pwm.c by Sascha Hauer <s.hauer@pengutronix.de>
8   *
9   * This program is free software; you can redistribute it and/or modify
10  * it under the terms of the GNU General Public License as published by
11  * the Free Software Foundation; either version 2 of the License, or
12  * (at your option) any later version.
13  *
14  * This program is distributed in the hope that it will be useful, but WITHOUT
15  * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
16  * FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License for
17  * more details.
18  */
19
20 #include <linux/clk.h>
21 #include <linux/clk-provider.h>
22 #include <linux/err.h>
23 #include <linux/io.h>
24 #include <linux/module.h>
25 #include <linux/of.h>
26 #include <linux/of_device.h>
```

Example: Continued from ~~PWM Driver~~ Code.

```
66          tcon = __raw_readl(S3C_TCON);
67          tcfg1 = __raw_readl(S3C_TCFG1);
68          tcfg0 = __raw_readl(S3C_TCFG0);
69
```

32.4.1.3 TCON (Timer Control Register)

Register	Offset	R/W	Description
TCON	0x7F006008	R/W	Timer Control Register

TCON	Bit	R/W	Description
Reserved	[31:23]	R	Reserved Bits
Timer 4 Auto Reload on/off	[22]	R/W	0: One-Shot 1: Interval Mode(Auto-F
Timer 4 Manual Update	[21]	R/W	0: No Operation 1: Update TCNTB4
Timer 4 Start/Stop	[20]	R/W	0: Stop 1: Start Timer 4

Oct. 23rd (Monday).

Note1. Project Assignment on CANVAS

CMPE258
Semester Long Team Project On Embedded Software Systems
HL

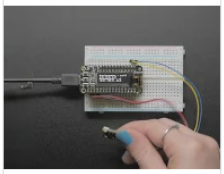
1. Design and implement a team project based on each 2-person team (or individual person). The project requirements are listed in details below. Note this project counts total 10 points and is due at the end of the semester and it requires a team presentation in the last week of the semester.

2. The technical requirements of the projects:
(2.1) Design and implement your team project which has to utilize embedded Linux OS on your chosen

Note2. This Coming Wednesday Presentation of your project Proposal, e.g., The Abstract.

3. One page project executive summary:
(3.1) Create one page executive summary of your project, with the following information
a. Title of the Project
b. List of each team members: First Name, Last Name, SID, Email Address, and Affiliation (such as Computer Engineering, Software Engineering, MS AI);
c. Team coordinator: Identify the team coordinator;
d. Abstract (up to one page):
Describe
(i) the objectives of the project ;
(ii) the technical challenges;
(iii) the proposed methodology to be employed;
(iv) the software tools and hardware platform;
(v) results and deliverable;
(vi) Experience gained and/or lessons learned.
e. table with list of the work contributed by each team member

Adafruit Industries LL…
$12.50
DigiKey
Get by 10/…



adafruit-Adafruit…
$12.50
Adafruit Indu…
30-day retur…

In-Class Presentation on Wednesday.

Homework. Nov. 1st (Wed) In-Class Demo.

1° Buy LSM303 sensor. (I2C Interface)

2° Enable PWM Driver on your target platform Then use Python, OR C/C++

To write an user program to read sensor information, and display the Sensor information on the Console.

3° In-class Demo Requirements.
Bring your prototype Board together with the sensor interface to the Class for Demo.
Ref1. from the github, First 25 slides.

CMPE244 / 2021F-116a3-i2c-v2-hl-2021-11-18.pdf

Ref 2. Command Line Commands for I2C Device Detection, and Python Sample Code.

2023S-102-i2c-command-line-2023S-104-i2c-v2-jetson-nano-2023-02-8.pdf

Note: Midterm Exam is to be scheduled Once the I2C Sensor interface work is Covered and homework is done. Most likely in the 2nd ~ 3rd week of November.

Oct. 25 (Wed).
Note1. Quiz Next Monday.
Please Bring Your Laptop / Prototype System to the Class.

Scope: OS. Kernel. Build. Device Drivers, Kconfig.

Note2: Quiz (Nov. 6).
PWM Prototype Board + Motor Drive Board + Stepper motor.
→ Code with Calculation. f_{PWM}, Duty.

Note3. midterm Exam.
Nov. 13 (Monday).
① Architecture CPU Datasheet; Device Drivers (GPP, PWM, I2C) ②
③ Coding → Register Level (SFRs)
④ Hand calculation for Code Debugging.

Oct, 30 Moday
Note 1. The road map for the 2nd half of the class.
(1) Enable/design/deploy webserver solution by using GI (Gate Way Interaces), e.g., WSGI and CGI, the first one is the backend code (engine) written in Python, the 2nd one is CGI for C/C++ based backend solution.
(2) Design/develop/deploy ChatGPT API Version 3.5 to build technical/user support system.
(3) APP development for the Android platform.

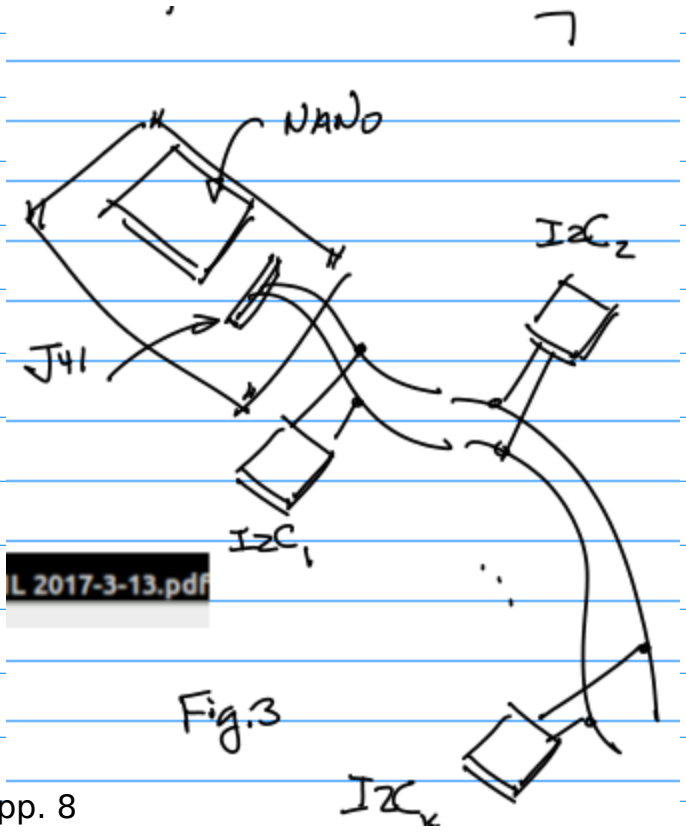
(4) Embedded software system will provide scalability and vertical integration. On hardwere side, we will add I2C sensor interface, LSM 303.

Note 2. Midterm exam is scheduled on Nov. 13 (Monday),
(1) One hour exam, one page formula is permitted. But close book and close notes.
(2) 15 minutes for prep for CANVAS submission.
Note: all the submission has to via CANVAS to be official, no late submission will be accepted. If the system is crashed or not responding to the subssion, please do screen capture to keep the record with time stamps and your ID.
(3) Scope of the midtem:
from the introducation to I2C SPR's init and configuration.
(4) format of the exam:
a. Combination of hand calculation and execution of your code.
b. you will need to bring your target platform to the exam.
c. The submitted homework, code can be reused.
d. Development platform, OS kernel compilation and build, which is a part of the subjects to be includeded.
kernel space debugging;
Kconf,
make menconfiguration;
e. Please bring your smart phone, you will need to take photos of your prototype board and the photo(s) of the actual implmentation, be sure to place SID card next to your board.
f. Scree captures may be needed, to capture the execution of your code with your ID.
(5) Prepare for the submission:
a. photos to be converted to pdf;
b. take photos of your hand calculation then convert the photos to pdf
c. combine all the pdf in a proper orcder into one pdf document.
Note: please use 1024x768 resolution as a reference when taking photos.

Example: On I2C protocole and I2C sensor interface.

Ref. 1.

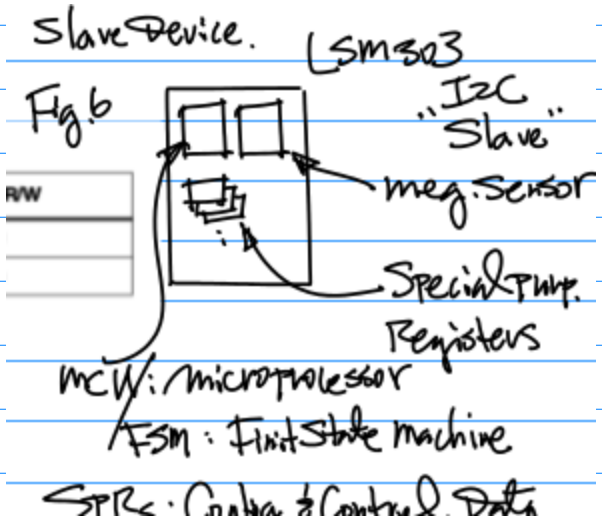
2022S-101-note-part2-cmpe242-2022-05-9.pdf



L 2017-3-13.pdf

Fig.3

pp. 8



Note: 1. Master and Slave of I2C devices;
2. Total number of slaves possible, theoretically, up to 7 bits, e.g., 2^7 = 128.
3. There is a MCU inside the slave for communication with this the master and execution of commands/instructions from the master, and perform init&config and other required task. 4. In case of LSM 303, there are 3 sensors in one package.

LSM303DLHC

Ultra compact high performance e-compass
3D accelerometer and 3D magnetometer module

Preliminary data

Features

- 3 magnetic field channels and 3 acceleration channels
- From ±1.3 to ±8.1 gauss magnetic field full-scale
- ±2g/±4g/±8g/±16g selectable full-scale
- 16 bit data output
- I^2C serial interface
- Analog supply voltage 2.16 V to 3.6 V
- Power-down mode/ low-power mode
- 2 independent programmable interrupt

LGA-14 (3x5x1mm)

Description

The LSM303DLHC is a system-in-package featuring a 3D digital linear acceleration sensor

Nov. 1st (Wed).
PWM Stepper motor Drive Demo. in-Class.
Note 1. Optional Feature:
Add Anaconda to Jetson NANO to Allow Better management of Development Environment and Packages.
Later in the Class Assignment, we may need this feature.

Example: Continuation of I2C Protocol.
1. Slave Addr. — Devive Datasheet
2. Additional Addr. for the master to Address to the SPRs (Special Purpose Registers) for the init & Config.
3. Information is then Read Back By the master. Which Can be realized By "Read" Command.

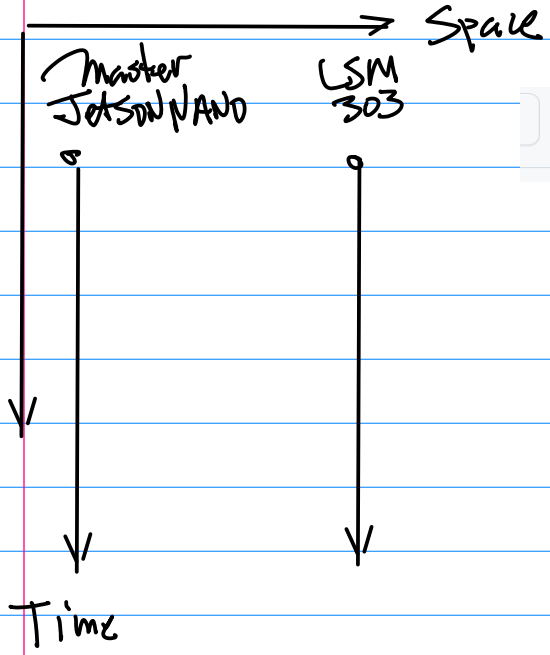
CMPE244

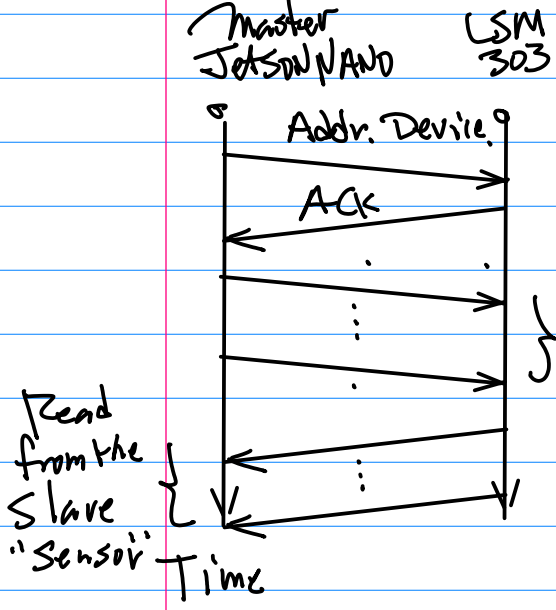
F2023

5D/

Space Time Diagram to Describe the I2C.

Spatial v.s. Temporal





Note: It's good Engineering Practice to Detect the Attached I2C Device(s).

Note: Read Datasheet to Complete the detailed Space-Time Diagram, then match up to your High Level Language Code.

Ref:

CMPE242-Embedded-Systems- / 2022S / 2022S-108b-AngularSensing-i2c-LSM303- final HL 2017-3-13.pdf

i2c

Nov.6 (Monday)

Note: Midterm Exam is scheduled on Next monday, Nov. 13th. Please Bring Your Prototype Board, And Blank Printer Paper.

Homework Nov. 19 (Sunday). a. 1° Integration of the target Board; b. motor Drive Board. (Note. The Board Has to Be able to Drive the Stepper motor per its Configuration, e.g. 1 Full Step (1.8 degree/step) OR A half step, etc. C. LSM303 Sensor to Calculate the motor angular Displacement;

2° Drive the motor 15 Degree Counter Clockwise Direction, then 15 Degree Clockwise Direction. Then Read the Angular information Back from your Sensor. (No further Calculation is Needed)

3° Submission ON CANVAS, And Bring your Prototype Board to the Class for Demo.

Note: For the team project, it is mendatory to Integrate ChatGPT into your Embedded Software System.

1. ChatGPT 3.5 API or higher;

Python Code.

Note: You may enable your Remote Access Software tool to connect to the Internet, in that case No USB wifi Needed.

2. USB wifi

Jetson NANO

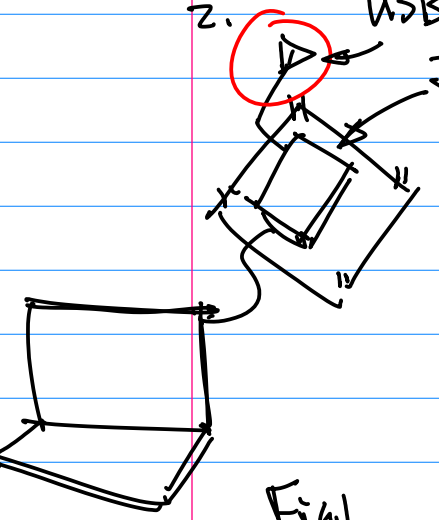


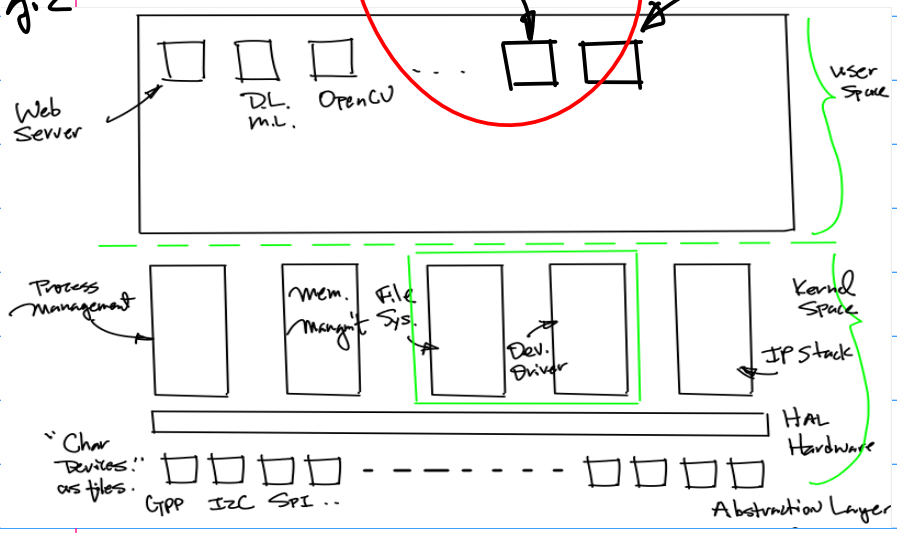
Fig1.

One of the program in the User Space.

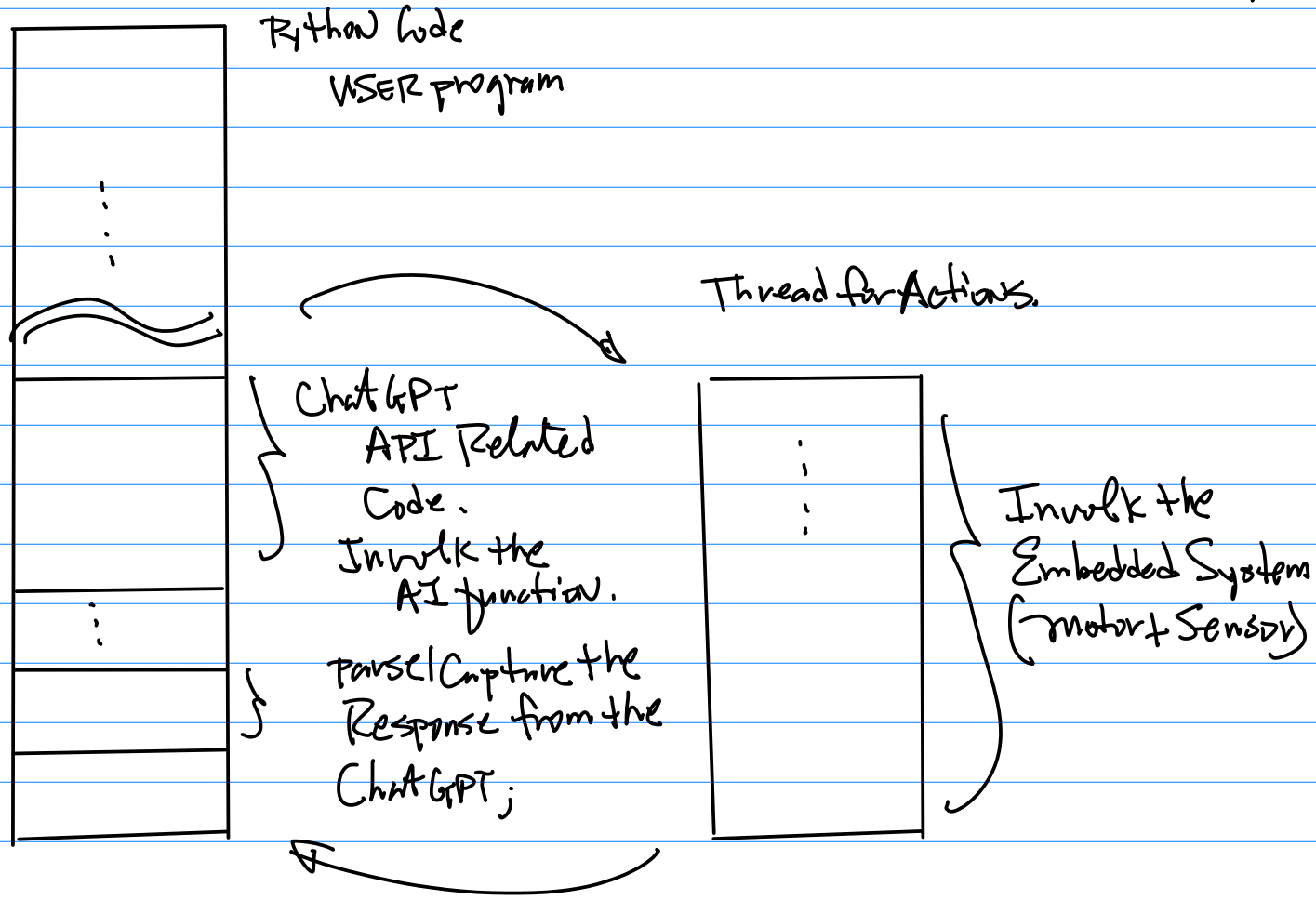
Python for ChatGPT API.

Web Server

Fig.2



Ref: github under OpenCV > DeepLearning2023S 20235-104 ~



Note: Both front End and Backend are Needed in this project.

Front End — UI (Browser)

Back End
1° Front End user input → Web Server → Back End code to Drive the motor.
2° Back End ~~Computation~~ → Server → On UI.
3° Back End Performs Some Computational Tasks. Such as Driving the motor

Get some data (From the Sensor)

Information flows One Direction

Information flows the Other Direction