

# CMPE242 Spring 2022

1/

Jan 26 (Wed) First Day of the Class.

Harry Li,  
E-mail: hua.li@sjtu.edu  
Text message: (650)400-1116  
Office hours: M.W. 4:30-5:30 PM.

Textbooks + References:

No Textbook, however CPU Datasheets Are employed as a Baseline Reference, and serves as a textbook.

1. ARM11 CPU Datasheet, from Samsung  
is SGH Document for the development Board.

2. NVIDIA Jetson NANO Developer

Kit. Reference Source for people Using NANO as a target platform.

System-on-module Document. (Not used that much in this class).

Design Guide As 2nd primary Ref. for Jetson NANO.

3. Broadcom Tie, BCM2835 (CPU Datasheet).

Selection of Target platform for this Course.

- a. NVIDIA Jetson NANO
- b. Broadcom Tie
- c. Samsung ARM CPU
- d. NXP i.MX 8

Note: Select your target platform from the options a-d.

(Consider NVIDIA Jetson NANO).

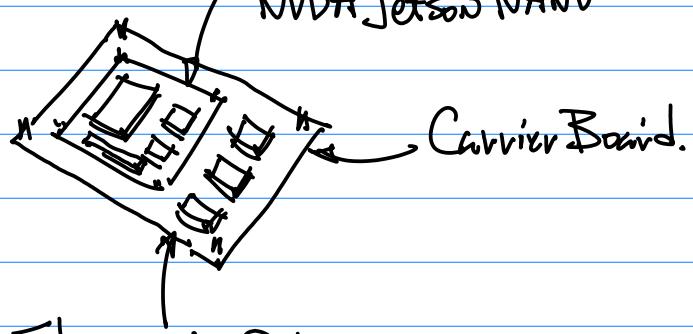
Programming Languages: { C/C++  
Python  
OS Support: Linux.

Rich I/O I/F Support.

Requirements for the Course:

1. Design / Build A State-of-the-Art Prototype System; Each Person will have to have one individual system.

NVIDIA Jetson NANO



Stepper Motors Drive...

2 Form 4 person team, Work on Homework, Project, However All Coding, Report etc have to be completed individually, no code, Report, Project etc. Can be Shared.

Grading:

- 1. Midterm Exam, Close Book / Close Notes
- 30% . Prototype System will be needed to Answer Design Questions, and to Execute programs. Need to take photos of the prototype.

2. Final Exam, Similar Format,  
40%. Prototype System is a  
Part of the Exam.

3. Homework, Projects. 30%.  
1st Project During the 1st  
Half of the Semester. 2nd Project

By Team Project, (a) End of the Semester,

Requires PPT Presentation & Live  
Demo.

Announcement in Class, in  
Written form Both in the Lecture  
Notes and ON SJSL Canvas,  
Late Projects/Homework (pt)  
Penalty.

Jan 31 (Monday)

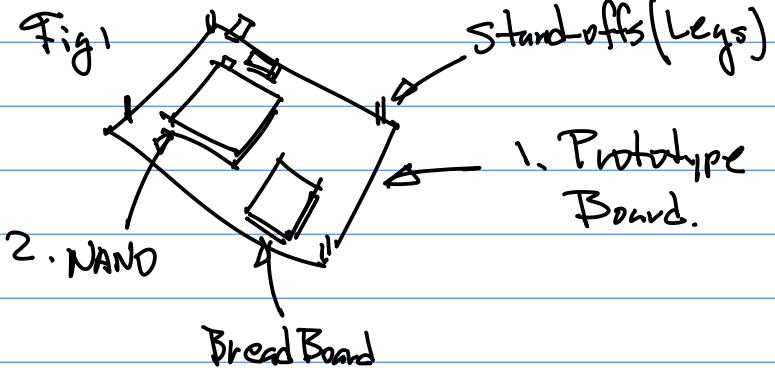
Today's Topic: Bill of Material &  
Prototype System.

Target platform to consider  
for your Prototype :

Nvidia Jetson Nano, 2GB  
Broadcom Pi e. 3B+, 4.

Note: Jetson NAND is Better  
And more powerful with Almost  
the Same Cost.

### Bill of Material for Prototype System



Note: a. Adequate size to host  
CPU module (NVIDIA Jetson Nano)

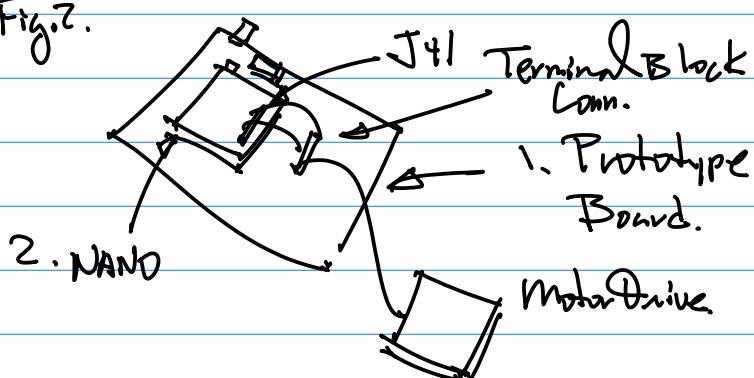
1b x 1½ Cm.

b. Power Circuit.

c. Stepper Motor Drive

Note: For Simple Testing Purpose, you can  
Choose to use Non-professional Grade Drive.  
Or Use professional Grade Stepper  
Motor Drive. (Robotics - CNC, 3D Printer  
Additive Manufacturing). Size of the  
Drive can be same size of your CPU  
Module.

Fig. 2.



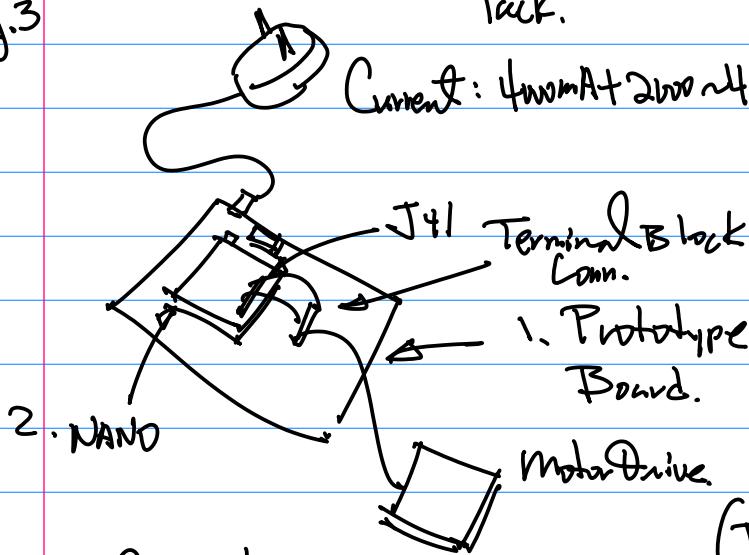
Stand-offs (lego)

Components for:

External Power Unit  
GPIO Testing Circuit

External Power Unit Option 1: Wall-Mount Adapter  
Option 2: Battery Pack.

Fig.3



Current

Note: Adaptor To Provide Power

Not just to the Target platform,

But also to the entire Board

NAND Target Requires 4A (4000mA)

Current for Peak Computing, On top of it, you will need to consider providing

Adequate Current for the Logic, for Sensor I/F (Lsm303), for Stepper Motor Drive.

Note: IC DC Regulation is Needed.

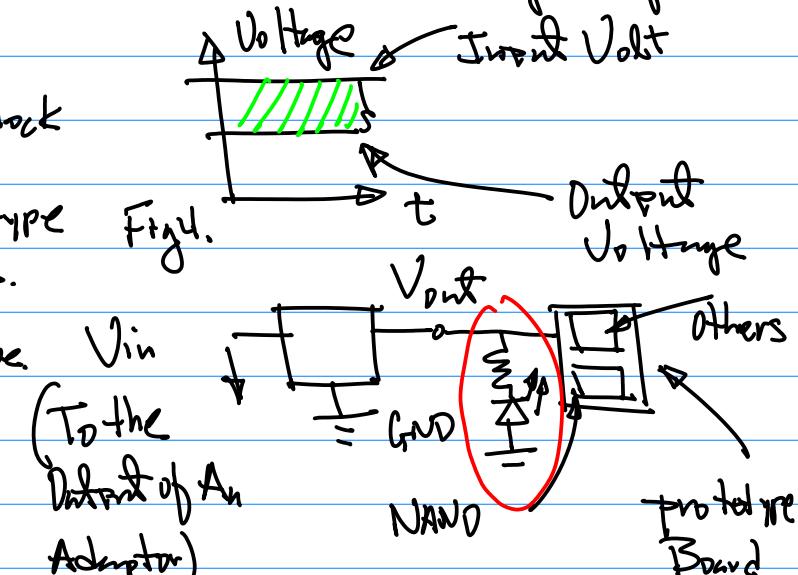
make sure DC Regulator IC Can Handle Adequate Current

Consider 7805 as an example. Only 150mA is allowed, No ground information

voltage integrated-circuit voltage regulators is designed for a wide range of include on-card regulation for elimination of noise and distribution problems assation. Each of these regulators can deliver up to 1.5 A of output current. thermal-shutdown features of these regulators essentially make them immune

(7805 Datasheet)

Note: Some power regulators will have Output voltage Drop.



②

Red LED,  $V_{LED} \approx 1.2V$ ,  $I_{LED} \approx 10mA$

③ Assorted Resistors (A few hundred of Ohms to A few Mega Ohms)

Caps. for Noise Reduction. for

IC Regulator Compensation).

④ Right Angle DC connector.

Right Angle



Voodoo Lab  
2.1mm Right  
Angle Barrel ...  
\$1.90  
West Coast ...



CUI Devices  
PJ-050AH DC  
Power ...  
\$1.14  
Mouser Elec...

Fig.5

## Anchor Electronics

Website

Directions

Save

4.6 ★★★★★ 50 Google reviews

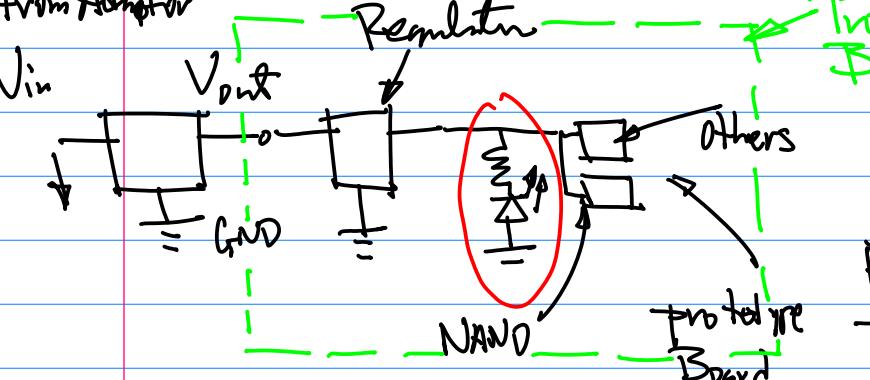
Electronic parts supplier in Santa Clara, California

Long-running supplier of a huge range of electronic components, tools, cables &amp; more.

Address: 2040 Walsh Ave, Santa Clara, CA 95050

Hours: Closes soon · 4PM · Opens 7:30AM Tue ·

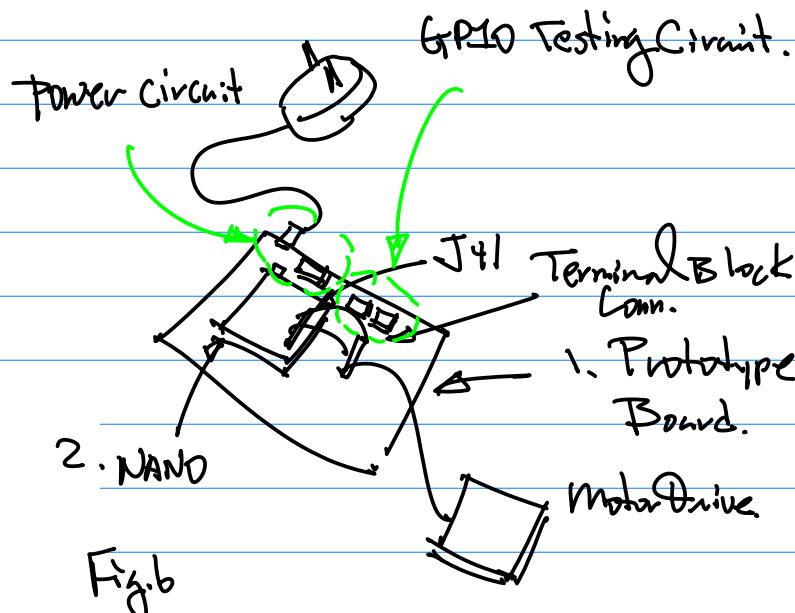
From Adapter



Components for GPIO Testing.

GPIO Testing

Input { "0"s  
"1"s  
Output { "0"s  
"1"s



Input Testing:

1. Toggles/w.

2. Assorted Resistors

(10 Ohms ~ A few hundred Ohms)



Output Testing:

1. Color LED (Red, yellow, Green)

2. Assorted Resistors.

Feb. 2nd (Wed)

Today's Topics :

1º Bill of material (Continuation)

2º Prototype System Design.

Class github

hualili Add files via upload

- 2019S
- 2022S
- 1-Lecture10\_OpAmp Circuits.pdf

Option B for the motor drive  
"Easy Drive" current ~1000mA.



EasyDriver Stepper Motor Controller  
\$1.71  
RobotShop.com

Example: Bill of material for PID Controllers (Stepper motor etc).

GR10 Testing Circuit.

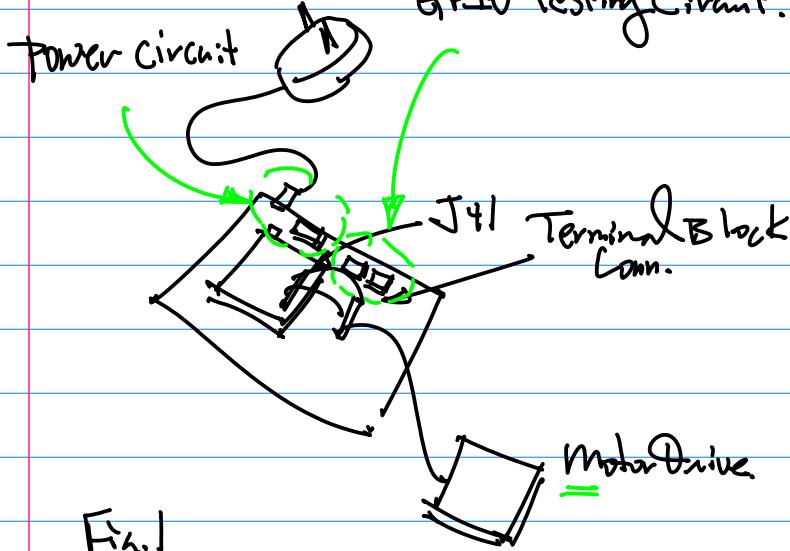


Fig. 1

PID Controller (P: Proportional, I: Integral, D: Derivative)

For CNC, 3D printer, for self-driving robot

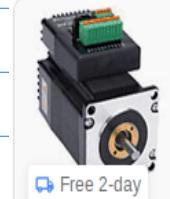
Motor Drive for Stepper motors.

Option A. Professional Drive

Option B.



InstaCNC Machining Center



Integrated Stepper  
\$265.00  
Automation...



TB6600 4.5A CNC Single-Axis For LDR-40B

Stepper Motor Controller 1 15 USD

[https://www.amazon.com/gp/product/B01DK5IRI/ref=oh\\_aui\\_detailpage\\_o02\\_s00?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B01DK5IRI/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1)

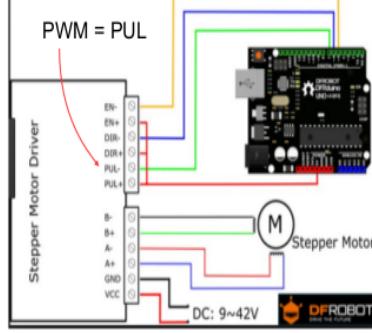


Fig. 2

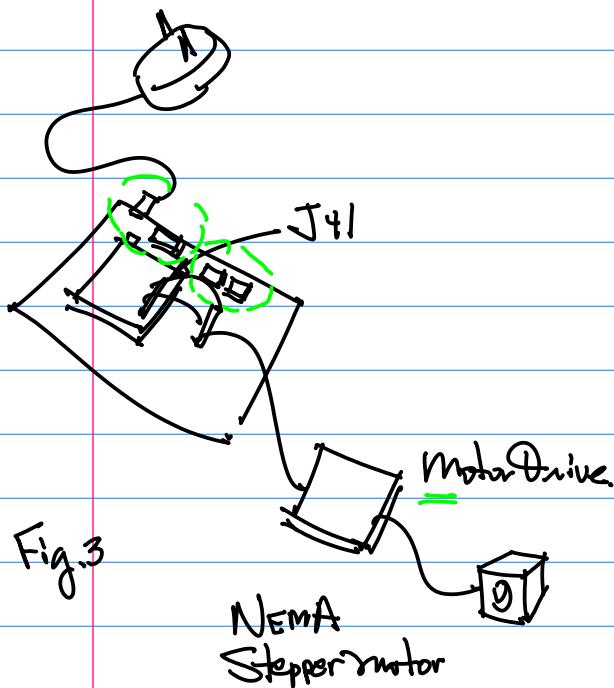
~\$15 - \$40

Note: Specs for Option A: a.

a. T85A (~250mA ~300mA for this class & Beyond) b. TB6600 (IC Chip Provides Adequate Power)

# CMP2042 Spring'22

Bill of material (motor)



[https://www.anahiemautomation.com/marketing/stepper/stepper-motors.php?gclid=EA1alQobChMlqsaW3pb19QIV-R6tBh1QBAFsEAAAYAiAAEglhx\\_D\\_BwE](https://www.anahiemautomation.com/marketing/stepper/stepper-motors.php?gclid=EA1alQobChMlqsaW3pb19QIV-R6tBh1QBAFsEAAAYAiAAEglhx_D_BwE)



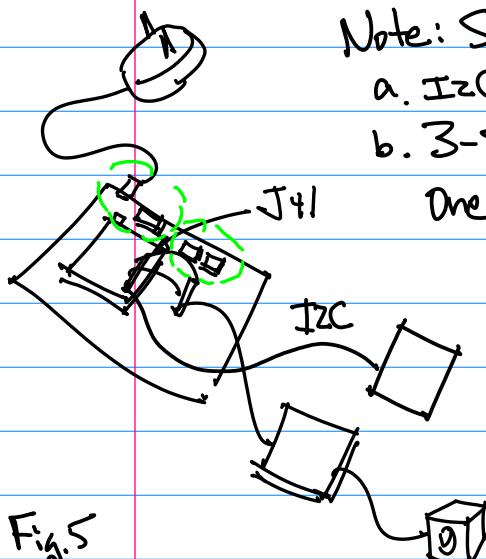
Stepper motor  
- NEMA-17

\$14.00  
Adafruit Ind...



Fig.4.

Bill of material (Sensors) for PID



Note: Sensor Lsm303

- a. I2C Sensor Interface;
- b. 3-sensing units in

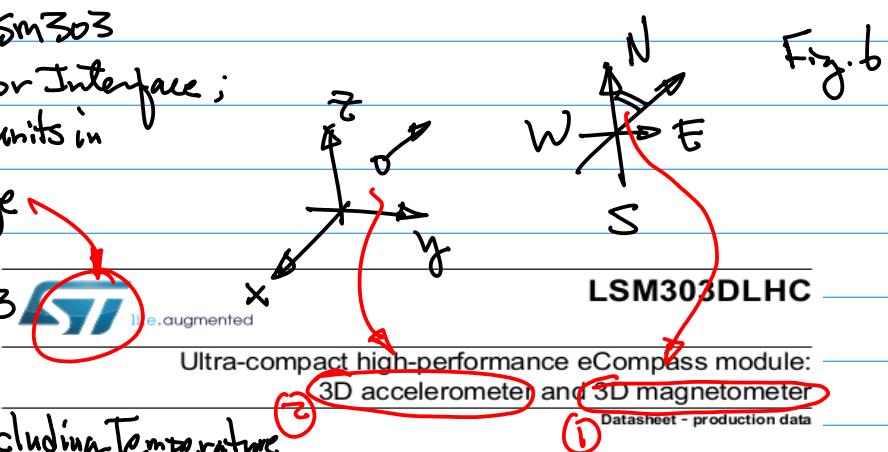
One package

Lsm303 STI  
Life.augmented

including Temperature  
Sensing (The 3rd one)

Ultra-compact high-performance eCompass module:  
3D accelerometer and 3D magnetometer

Datasheet - production data



# CmpE242 Spring'22

7

## Selection Guideline (Continued)

### 2. Sample Rate:

500KSPS or 1 MSPS (million Samples per Second)

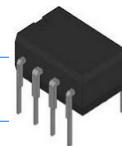
### 3. Quantization Level of ADC.

Bits per Sample. 8 or 10 bit Resolution  
minimum:

Nyquist Sampling Theorem.

$$f_{\text{Sample}} \geq 2f_{\text{max}} \dots (1)$$

Note: Optional Component — OpAmp.  
Operational Amplifier



Single OpAmp.

OpLM358  
(Quad OpAmps  
in One Package).

Texas Instruments  
LM741CN/NOPB  
\$0.91  
Digi-Key

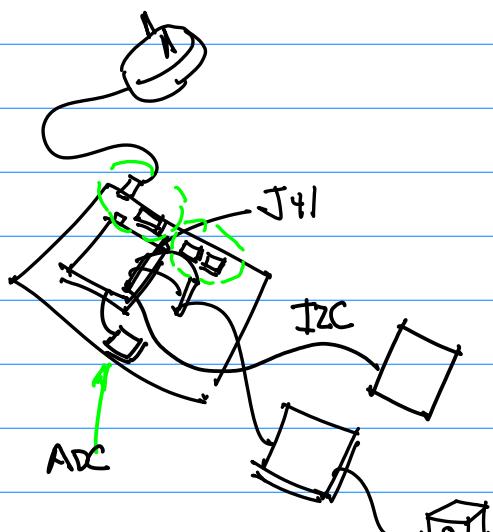


Fig.7

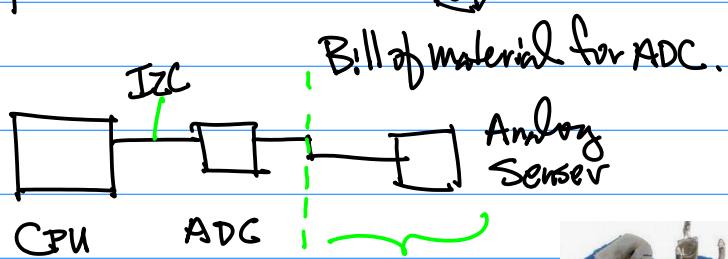
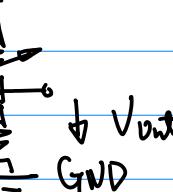
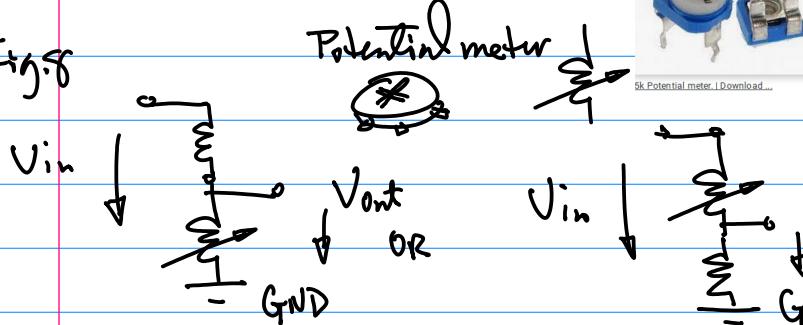


Fig.8



Note: Baseline Reference  
Samsung AR2MII.

Homework: Due A week from Today. (Feb. 9)

1. Identify your target platform, 11:59 PM

And provide A screenCapture or photo  
of its connector. With Table of Pin  
Expansion Assignment ;

2. Create A table of Bill of materials Based  
on Lecture Discussion.

3. Submission: A pdf file with Naming:  
First-Last-SID-Target-ZYZ.pdf.  
Submission to CANVAS.

Feb 7 (mon)

Today's Topics :

- 1° Bring Up the target platform.
- 2° GPIO Testing, e.g., "Hello, the world".

Target Platforms {  
a. Jetson Nano  
b. Pi3B+/4.

# CMPE242 Spring'22

Ref: CPU Datasheet of Samsung ARM-11 is on the Class github. (CMPE242)

2021F-105-#0-cpu-arm11-2018S-29...  
2021F-105-#0-nand-00-cpu-Timed

PPT for Today's Lecture

Example: Bring up The target Platform, NAND.

Note: Visit Nvidia Developer Site, And Sign up As a developer

Pre-requisites :

- 1° Developer Account ;
- 2° Linux O.S. (Kernel Source Distribution) for Device Driver Development & Kernel Source Optimization;
- 3° Download pre-Compiled, Pre-Built Kernel Image to SD Card And
- 4° Target Board, Jetson Nano 2 GB or 4 GB.

Step 1. Download Pre-Build Kernel Image

Step 1. Download SD card OS image from Nvidia to your host machine, laptop, the zipped file size is 6.1G, unzip it to get OS image, e.g., \*.img file, ref:

<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write>  
my Li, Ph.D.

Step 2. Download "Etcher" Software tool, then execute this tool to write the Downloaded Pre-Built Kernel O.S. to your SD Card. 8

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

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

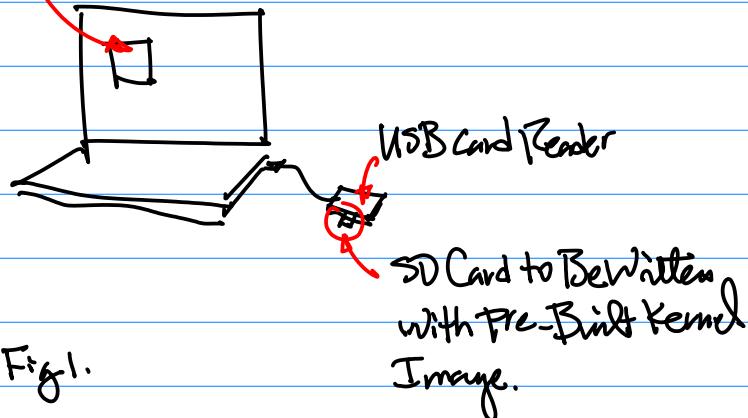


Fig 1.

Step 3. Take SD Card from the Reader, Insert the Card into the USB Slot of the target platform, then Power On the System. Then follow the steps during the Booting to initialize & Configure the System, Such as user Name, Password, Time Zone Setup etc.

Example: To prepare GRPSD Testing.  
(To Be Assigned as Homework)

Hardware Aspect :

Step 1. Identify the Expansion Connection &

# CMPE242 Spring22

9

Pin Assignment.

From NVDA Developer Site to find

Connection Information

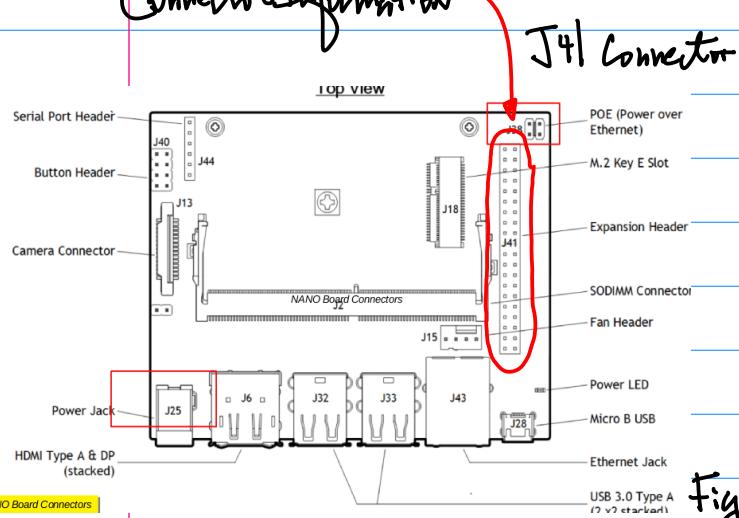


Table 1. Pin Assignment / Connectivity Table

CPU / Connector Pin	Description	Note
GPIO79/J41-H2	GPIO Input	
GPIO78/J41-G0	GPIO Output	
GND/J41-b	GND	

Step 3.  
Design Schematics

Fig.2

Step 2. Establish Pin Assignment /  
Connectivity Table

Pin Assignment → GND : Common GND  
→ GPIO : GPIO79/Pin12  
→ GPIO78/Pin10

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC Power	1	2	5.0 VDC Power	
	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power	
	I2C_2_SCL I2C Bus 1	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
	GND	9	10	UART_2_RX /dev/ttyTHS1	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3 VDC Power	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_1_CS0	gpio19
	GND	25	26	SPI_1_CS1	gpio20

gpio149	GND	25	26	SPI_1_CS1	gpio20
gpio200	I2C_1_SDA I2C Bus 0	27	28	I2C_1_SCL I2C Bus 0	
gpio38	CAM_AF_EN	29	30	GND	
gpio76	GPIO_PZ0	31	32	LCD_BL_PWM	gpio168
gpio12	GPIO_P66	33	34	GND	
	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio7
	GND	39	40	I2S_4_SDOUT	gpio78

j, Ph.D.

Fig.3

# CMPE242 Spring 22

10

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC Power	1	2	5.0 VDC Power	
	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power	
	I2C_2_SCL I2C Bus 1	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
	GND	9	10	UART_2_RX /dev/ttyTHS1	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK gpio79	
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1 gpio232	
	3.3 VDC Power	17	18	SPI_2_CS0 gpio15	
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO gpio13	
gpio18	SPI_1_SCK	23	24	SPI_1_CS0 gpio19	
	GND	25	26	SPI_1_CS1 gpio20	
	GND	25	26	SPI_1_CS1 gpio20	
	I2C_1_SDA I2C Bus 0	27	28	I2C_1_SCL I2C Bus 0	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_P20	31	32	LCD_BL_PWM gpio168	
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS gpio51	
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN gpio77	
	GND	39	40	I2S_4_SDOUT gpio78	

j, Ph.D.

Input Testing.

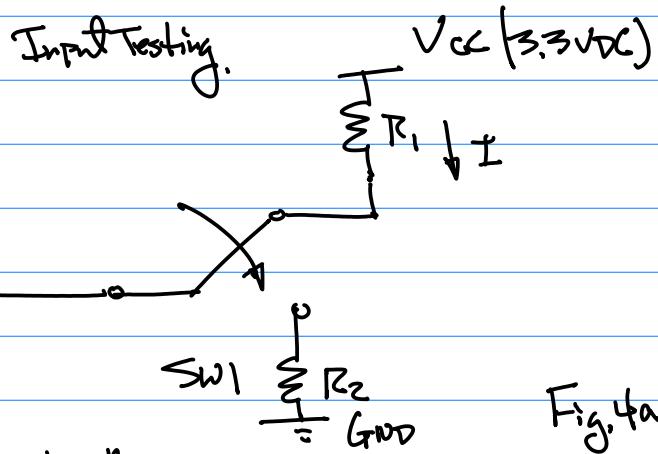
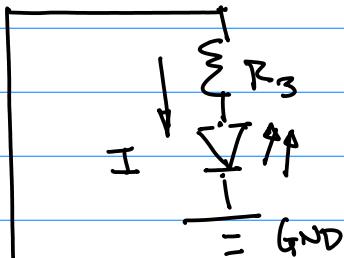


Fig.4a

Let  $I = 10mA$ .

$$R_1 = \frac{V_{cc}/I}{10 \times 10^{-3}} = 330\Omega, R_2 = R_1$$



$I \approx 10mA$

Note: Since NAND Output Current is small, so we choose not to use  $R_3$ .

Feb 9 (Wed) Topics:

- 1° Building GPIO Testing Capability On the target platform, Jetson NANO.
- 2° Kernel D.S. Sources plus toolchain

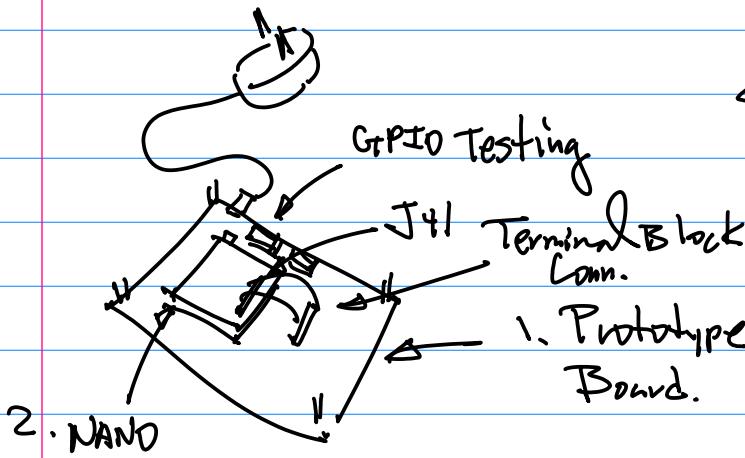
Example: Bring Up Your Target Board

Step 1. Build your prototype system with NAND Target Board mounted on it.

**Homework** (Due A week from today)

~~Feb 16~~ Moved to Feb 28th

1. Prototype System with A Carrier Board  
4" x 3" or Similar Size of your Choice;
2. GPIOD Testing Circuit for Both Inert / Duct Testing (Schematics)
3. Mount your NAND on the Carrier Board.



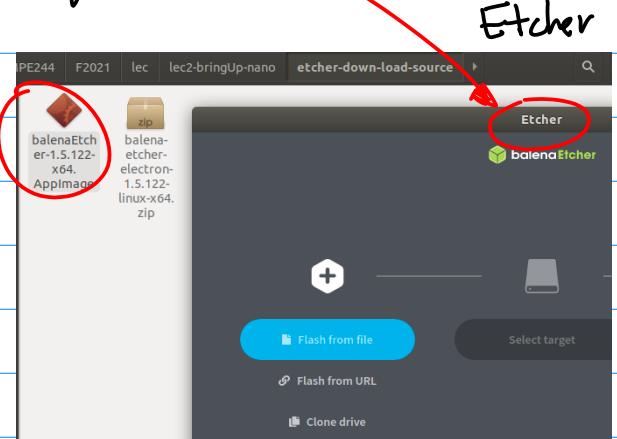
4. Take a photo of your System (with A root connection)

5. Screen Capture from your root machine, which shows NAND is running.

6. Put Photos plus Schematic into One PDF file, Zip it, Submit to SJSU CANVAS.

Step 2. Download Pre-Build Kernel

Image From NVDA Developer Site, And together with Flash Writing Software.



Step 3. Run "Etcher" program to upload the OS. Kernel Image to the target NAND Board.

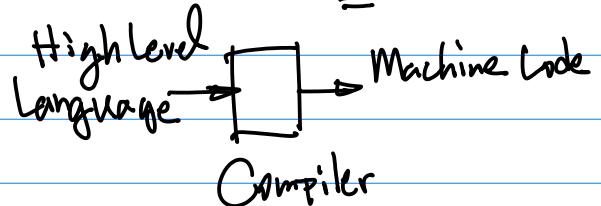
Note: Target Image for my Board is 2019.4 ? Due to my Applications in Deep Learning. For Embedded Class, we can use other / Latest Release

2022S-105-n-Jetpack-kernel-driver-simpler-2021-2

Jetpack from NVDA consists of  
a. OS Kernel (Device Driver), b.  
GPU Packages, c. OpenCV, d. DNN

OS (Operating Systems)

Tool Chain: → Cross Compiler



Compiler

Step 4. Take SD Card to the target NAND, Boot (Power Up) the NAND, finish Configuration process, then Ready to go. (To Create your Applications).

Examples:

- ✓ O.S. Kernel Source Debugging,
- ✓ Device Driver Development
- ✓ O.S. + Tool Chain for NAND

✓ O.S. + Tool Chain for Samsung ARM11

Step 1. Download And Install

Kernel Source, And Tool Chain.

Note: Purpose is to Optimize Kernel Image, Optimize/Develop/Debug Device Drivers.

→ make menuconfig

↑ Software Tool for Kernel O.S.  
Compile+Build, Arch for Device  
Driver Development & Debugging.

Feb 14 (Monday)

Topics: 1° Kernel Space Programming.  
Kernel Source Distribution, Tool Chains,  
Device Driver Development.

Example: Embedded Software Architecture

Objective: To Connect Kernel O.S. Drivers to the Target NAND plus Carrier Board

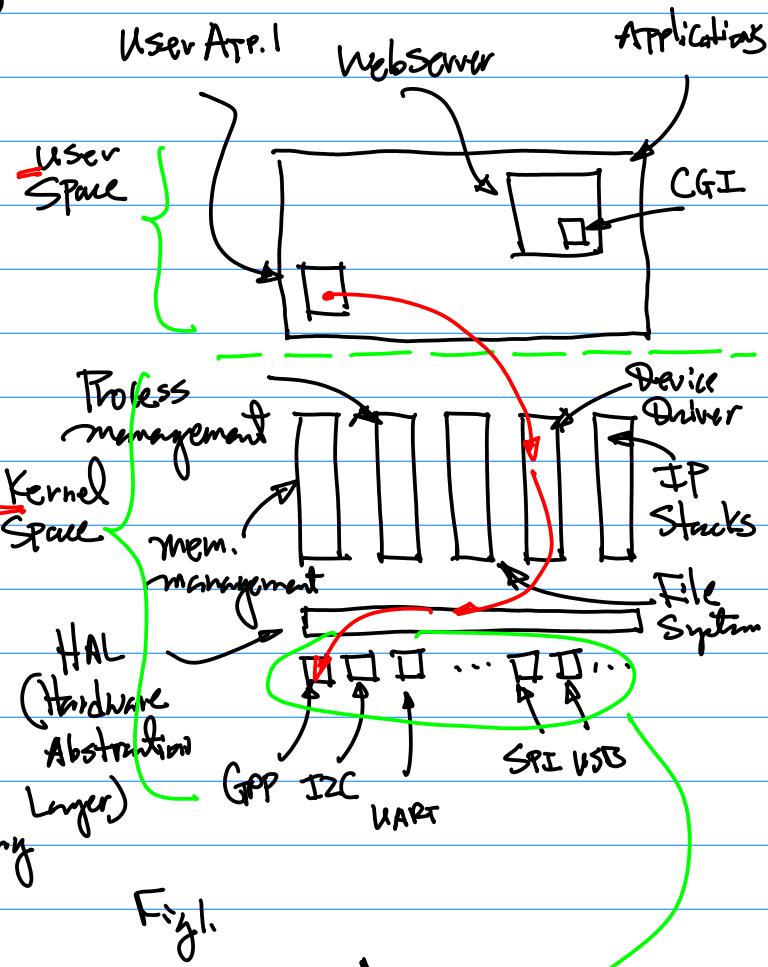
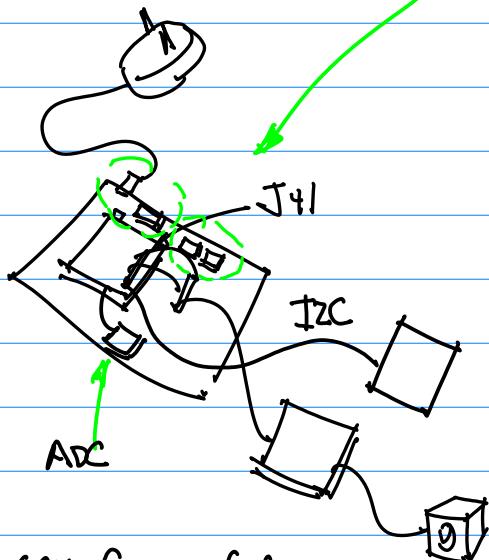


Fig. 1.



Note: CGI = Common Gateway Interface  
= Older, But Widely Adopted Technology  
for I/O Devices Interfaces.

CMPE242 Feb 14, 22

Note: Software Architecture  
(Block Diagram)  $\xrightarrow{\text{}} \text{prototype}$   
System illustration (Fig.1) are  
required.

Example Kernel Source Distribution  
 $\xrightarrow{\text{}}$  Tool Chain.

Download & Install O.S. Kernel Distribution  
(Pre-Build Image & Source Code Distribution)  
Together with the tool chain.

Target: NAND

Reference platform: Samsung ARM11.

CPU (ARM11) Datasheet

Class github

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

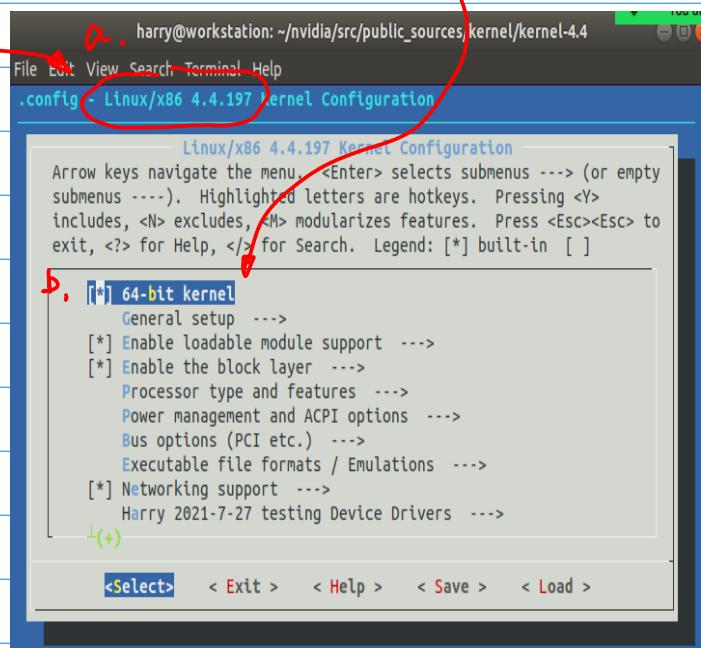
Development of Device Driver

Step 1. Run menuconfig for kernel  
Configuration.

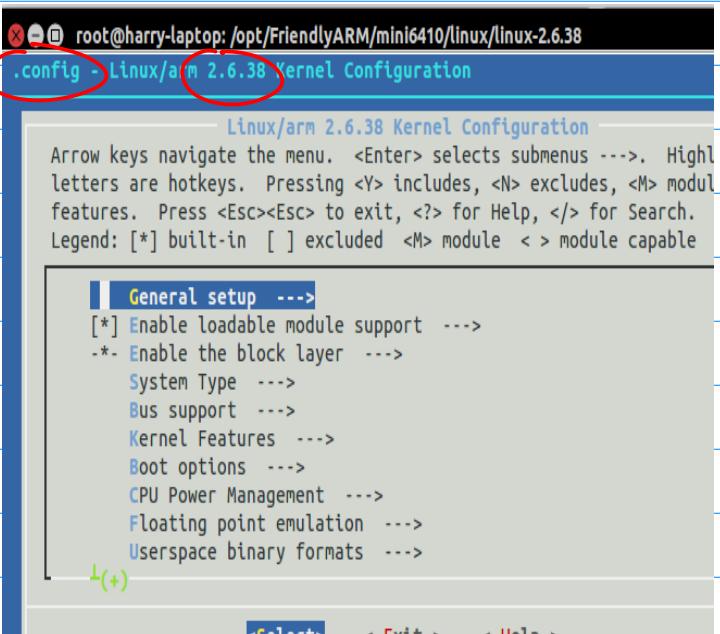
(Go to your installation directory,  
home\NVIDIA\...\Source\Kernel\...)

a. Version of the config b. Target

13



Now, for Samsung Target (ARM11)  
a. Version of Config. for 32 bit ARM11.



Step 2. Locate Device Driver for  
GPIO I/F.

Note: I2C, SPI Are 2 Device Drivers  
plus PWM are Needed for ZIF.

Take Samsung As Example first.

Select Device Driver at the Root UI Option

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

Linux/arm 2.6.38 Kernel
Arrow keys navigate the menu. <Enter> selects
letters are hotkeys. Pressing <Y> includes
features. Press <Esc><Esc> to exit, <?> for
Legend: [*] built-in [ ] excluded <M> mo
^(-)
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
Userspace binary formats --->
Power management options --->
[*] Networking support --->
  Device Drivers --->
    File systems --->
L(+)

<Select> < Exit >
```

Brows till Reach the Device Driver You Need.

GPIO Testing with Device Driver "LED"

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

Character devices
Arrow keys navigate the menu. <Enter> selects
letters are hotkeys. Pressing <Y> includes
features. Press <Esc><Esc> to exit, <?> for
Legend: [*] built-in [ ] excluded <M> mo
^(-)
  Virtual terminal
    [*] Support for binding and unbinding
    [ ] /dev/kmem virtual device support
    <M> LED Support for Mini6410 GPIO L
    <M> Harry 2021-2-3: I2C sensor module
    <M> Harry: 2016-Feb-22, CMPE 242 Mi
    <M> Harry: Mini6410 Test module
    <M> Harry: Mini6410 PWM2 module
    < > Buttons driver for FriendlyARM
    < > Buzzer driver for FriendlyARM M
L(+)

<Select> < Exit >
```

Step 3. Select Char Device (Character)

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

Device Drivers
Arrow keys navigate the menu. <Enter> selects
letters are hotkeys. Pressing <Y> includes
features. Press <Esc><Esc> to exit, <?> for
Legend: [*] built-in [ ] excluded <M> mo
^(-)
  [ ] Multiple devices driver support
  < > Generic Target Core Mod (TCM) ar
  [*] Network device support --->
  [ ] ISDN support --->
  < > Telephony support --->
  Input device support --->
  Character devices --->
    <*> I2C support --->
    [ ] SPI support --->
    PPS support --->
L(+)

<Select> < Exit >
```

Needed In the Future

Example: Now, Switch to menuconfig Version 4.4.1(a7) for NAND, Look & Jot down. Feel of the menuconfig is the same.

Now, Let's Discuss NAND GPIO I/F with Utilization of Existing (Factory Level Release) Device Driver.

### Homework

1. Write C/C++, OR Python Code to

Perform GPIO Input/Output Testing on NAND. Make Sure use your GPIO Testing Circuit Designed in the Class;

2. Input Testing has to have input "

And input "0" By Toggling the Switch;

3. Output Testing to Cover Output  
"1" & "0" to turn on/off LED.

4. Photos of Execution of the Program:

a. Input "1" & "0" Console print message.

b. Output "1" & "0", LED Light ON/OFF.

c. Entire System ;

5. Source code, Binary (for C/C++).  
as well

6. Create One PDF for All photos, Zip to include  
Source code And/Or Binary.

7. Submit Zip file to  
CANVAS. By ~~23rd (Wed)~~  
~~28th (Mon)~~

Feb 16 (Wed)

Note: 1° Homework Rescheduled  
to Feb 28th (Monday) ;

2° Bring your Prototype Board to  
the Class, Monday, Feb 21st,  
for quick assessment / evaluation

Topics Today: 1° Development Environment;  
2° Prep for the project (PID —  
Proportional, Integral, Derivative Controller  
with Stepper Motor Control & Sensor  
interface).

Example: Development Environment.

Menu config & Device Drivers.

Step 1. Have installed D.S. Distribution, and  
Enable Development Environment.

Step 2. User Space Programming to use  
Device Driver(s);

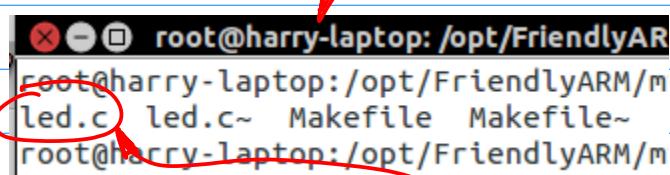
Samsung ARM11

Jetson NAND

Build GPIO Testing Capability to Turn On/Off  
LED via GPIO port.

GPIO I/F Testing (For ARM11)

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples# ls
adc-test  fft  lecSSBike  PWM  threadTest  Vision
buttons  i2c  led-player  readme.txt  usbcam  www
camtest  lecPID  leds  robotControl  vfp-test
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples#
```



```
root@harry-laptop: /opt/FriendlyAR
root@harry-laptop: /opt/FriendlyARM/m
led.c  led.c~  Makefile  Makefile~
root@harry-laptop: /opt/FriendlyARM/m
```

C Programming in the UserSpace

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
```

} for Device Driver  
Related Use

Note: This code will be posted on

a. github

```
int main(int argc, char **argv)
{
    int on;
    int led_no;
    int fd;
```

b.

```
    printf("hello\n"); //Feb. 16, 2015
    if (argc != 3 || sscanf(argv[1], "%d", &led_no) != 1 || sscanf(argv[2], "%d", &on) != 1 ||
        on < 0 || on > 1 || led_no < 0 || led_no > 3) {
        fprintf(stderr, "Usage: leds led_no 0|1\n");
        exit(1);
    }
```

c.

```
    fd = open("/dev/leds0", 0);
```

file descriptor

Path of Driver's Name

d. ioctl(arg1,arg2,arg3)

arg1 — fd — Device Driver; arg2 for sending "0"

```
if (fd < 0) {
    fd = open("/dev/leds", 0);
}
if (fd < 0) {
    perror("open device leds");
    exit(1);
}
ioctl(fd, on, led_no);
close(fd);

return 0;
```

Or "1" to Turn on/off LED;

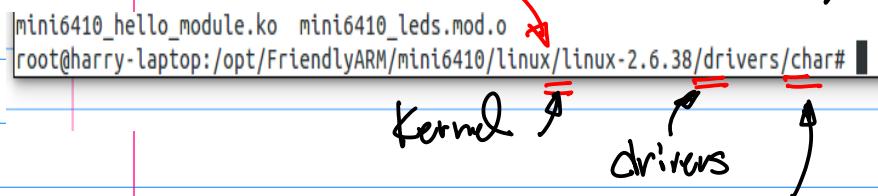
arg3 for the LED to Be Selected.

Compile & Build the User Space program, \$ make  
Cross-Compiler

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds# make
arm-linux-gcc -Wall -O2 led.c -o led
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds# ls
led led.c led.c~ Makefile Makefile~
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/examples/leds#
```

Binary Executable is generated

a. Step 3. Device Driver, find Source code in the proper folder *CharacterDevice*,



b. Find Source code of the target CPU.

c. Then, check "leds.c" related Device Driver

d. file extension ".ko": Built Device Driver.

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char# ls
mini6410*
mini6410_adc.c          mini6410_hello_module.mod.c    mini6410_leds.o
mini6410_adc.mod.c       mini6410_hello_module.mod.o   mini6410_pwm2.c
mini6410_adc.o           mini6410_hello_module.o      mini6410_pwm2.mod.c
mini6410_buttons.c       mini6410_leds.c             mini6410_pwm.c
mini6410_buttons.o       mini6410_leds.ko            mini6410_pwmHarry.c
mini6410_hello_module.c  mini6410_leds.mod.c        mini6410_pwm.mod.c
mini6410_hello_module.ko mini6410_leds.mod.o
```

Compiled into Binary

Copy this module to your target platform

Options: LAN, WIFI  
USB, RS232  
USB Dangle

Host Development platform

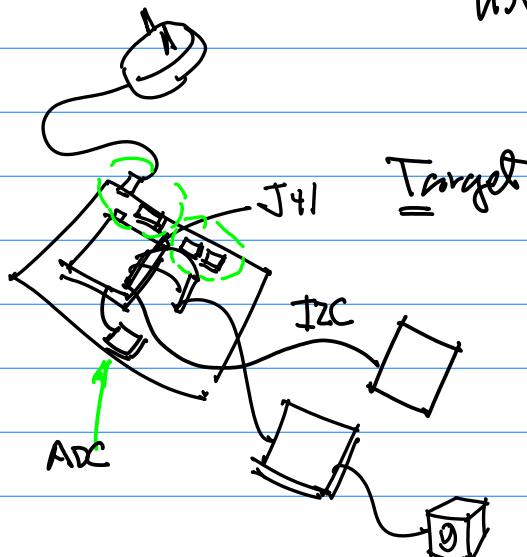
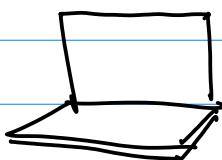


Fig. 1

Once it is copied, then you can install the .ko Driver By  
\$ insmod my\_driver.ko

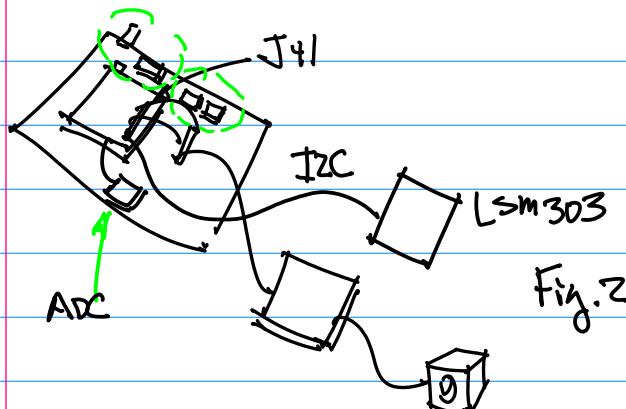
Consider Building A Prototype for

PID Controller Project.

Start from the Prototype System

Design for PID Controller Project

System Design :



Stepper motor:

1. NEMA14 or NEMA17 motor. 4 wires A+, A-, B+, B-
2. Stepper motor operates in the following fashion

200 Steps for One Revolution  
Turning of  $360^\circ$ .

Each Step is a Full Step. it gives  $1.8^\circ$  Rotation.

Table 1 Stepper motor Rotation Angles

Full Step	$\frac{360}{200} = 1.8^\circ$
Half Step	$0.90^\circ$
$\frac{1}{4}$ Step	$0.45^\circ$
$\frac{1}{8}$ Step	$0.225^\circ$

Homework (One A week from Today)

Feb 23rd. Preparation for the target, e.g. NANO to drive a stepper motor.

Pre-requisite:

1. NEMA17 or NEMA14 motor;
2. Motor Drive
3. Signal generator to provide Square waves. ( $0 \sim 1 \text{ KHz}$ )

(Note: if you do not have Signal generator, then you can use Timed GPIO Output Signal)

$$\begin{aligned} 100 \text{ ms} &\rightarrow 10 \text{ Hz} \\ 10 \text{ ms} &\rightarrow 100 \text{ Hz} \end{aligned}$$

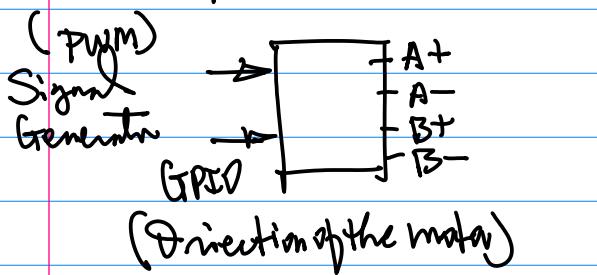
Requirements:

1. Configure your Motor Drive for full step Output;
2. Connect Input (GPIO) DR Signal (generator) to the motor Drive.

3. Connect Motor Drive to the motor to observe its operation.

4. Submit 5 seconds video clips to show your result.  
Submission on CANVAS.

Note: Motor Drive 2 pins (Joints)  
Important Among Other  
Pins



Feb 21st (Monday)

Topics: 1° GPIO Programming.  
2° Stepper motor Drive.

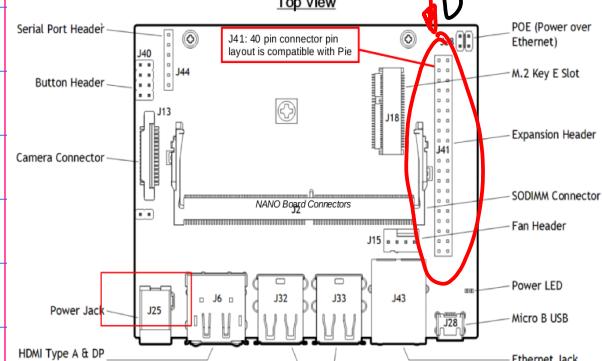
Example: GPIO Programming.

Ref:

[2022S-104-gpio-systemLevel-and-c-#202](https://www.raspberrypi.org/documentation/raspbian/manual/html/_static/gpio-systemLevel-and-c-#202)

Connector Information

J41 Connector  
for the newer Version



### NVIDIA Jetson Nano J41 Header Pinout

<https://www.jetsonhacks.com/nvidia-jetson-nano-j41-header-pinout/>

Note: I2C and UART pins are connected to hardware and should not be reassigned. By default, all other pins (except power) are assigned as GPIO. Pins labeled with other functions are recommended functions if using a different device tree.

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
3.3 VDC Power		1	2	5.0 VDC Power	
I2C_2_SDA	I2C Bus 1	3	4	5.0 VDC Power	
I2C_2_SCL	I2C Bus 1	5	6	GND	
GPIO216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
GND		9	10	UART_2_RX /dev/ttyTHS1	
GPIO50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
GPIO149	CAM_AF_EN	20	21	GND	
GPIO200	GPIO_P20	31	32	LCD_BL_PWM	gpio168
GPIO38	GPIO_P6	33	34	GND	
GPIO76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
GPIO12	SPI_2_MOSI	37	38	I2S_4_LRCK	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

Jerry Li, Ph.D.

Fig2  
Once the physical layout of the NAND Board is Ready (J41 Connector)  
Identify the pin assignment for the NAND Board.

Note: 1° All the pins are multiplexed e.g., Each pin have more than one function, such as GPIO (Ethernet), PWM, SPI etc. 2° Factory Setting is given in this color coded Table, GPIO

Fig1.

CMPE422 Feb 21, 22

### a. Choose Sysfs GPIO Setting

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC Power	1	2	5.0 VDC Power	
	I2C_2_SDA I2C Bus 1	3	4	5.0 VDC Power	
	I2C_2_SCL I2C Bus 1	5	6	GND	
gpio216	AUDIO_MCLK	7	8	UART_2_TX /dev/ttyTHS1	
	GND	9	10	UART_2_RX /dev/ttyTHS1	
gpio50	UART_2_RTS	11	12	I2S_4_SCLK	gpio79
gpio14	SPI_2_SCK	13	14	GND	
gpio194	LCD_TE	15	16	SPI_2_CS1	gpio232
	3.3 VDC Power	17	18	SPI_2_CS0	gpio15
gpio16	SPI_1_MOSI	19	20	GND	
gpio17	SPI_1_MISO	21	22	SPI_2_MISO	gpio13
gpio18	SPI_1_SCK	23	24	SPI_1_CS0	gpio19
	GND	25	26	SPI_1_CS1	gpio20

Fig.3

The configuration program

```
harry@harry-desktop: ~
Select one of the following:
Configure Jetson 40pin Header
Configure Jetson Nano CSI Connector
Configure Jetson M.2 Key E Slot
Exit
```

Ref: To config JtP pins, use Python Configuration Tool.

2022S-106-ioconfig-py-v4-hl-2021-12-19.pdf

Source code

### b. Connection of Sysfs gpio to physical pin.

User program (user space)  
Python, C/C++

Interfacing to gpio79 via  
physical connection to  
J41-12

In order to talk to kernel space  
Device Driver, use Configuration  
tool from the Target Platform Company.

User App.1

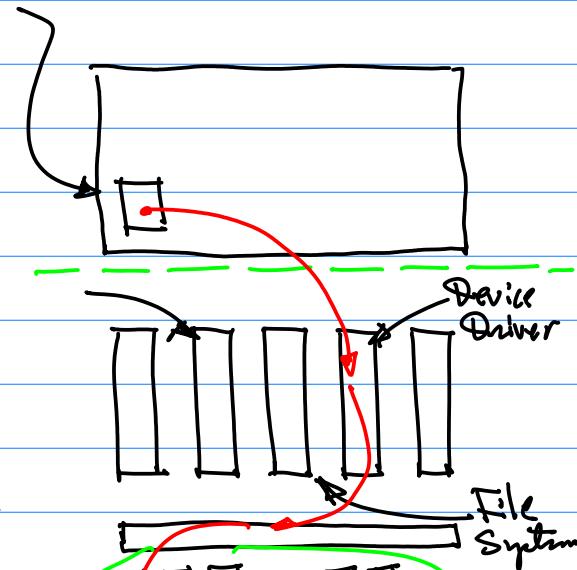


Fig.5

GPP I2C SPI UART

Fig.4

Based on the Ref.

Table GPIO Pin Assignment

GPIO Number	J41 Connector Pin
gpio79	J41-12
gpio78	J41-40
GND	J41-6

Ref: For System Level gpio, Python gpio, C/C++ gpio

[2022S-104-gpio-systemLevel-and-c-#2021F-114-gpio-n...](#)  
[2022S-104b-gpio-connector-systemCmd-python-hl-2021...](#)

First, System Level Testing, e.g., the Command line input testing, → Simple, easy to use.

To Run the Python Configuration Tool,  
 First, fix the bugs from the distribution

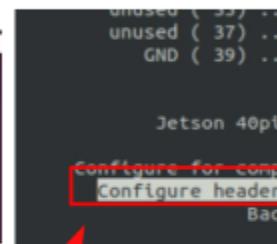
Step 1 Fix bugs  
 from the distribution

## Configuration of Pins

\$sudo find /opt/nvidia/jetson-io/ -mindepth 1 -maxdepth 1 -type d -exec touch

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

```
harry@harry-desktop:~$ sudo /opt/nvidia/jetson-io/config-by-pin.py -p 5
Traceback (most recent call last):
  File "/opt/nvidia/jetson-io/config-by-pin.py", line 84, in <module>
    main()
  File "/opt/nvidia/jetson-io/config-by-pin.py", line 39, in main
    jetson = board.Board()
  File "/opt/nvidia/jetson-io/Jetson/board.py", line 229, in __init__
    self._dtb = _board_get_dtb(self._compat, self._model, dtbdir)
  File "/opt/nvidia/jetson-io/Jetson/board.py", line 114, in _board_get_dtb
    raise RuntimeError("No DTB found for %s!" % model)
RuntimeError: No DTB found for NVIDIA Jetson Nano Developer Kit!
```



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

Step 2. Run jetson-io.py to configure pins

\$sudo /opt/nvidia/jetson-io/jetson-io.py

harry@harry-desktop:~

Fig.b

Then, Execute the Python Configuration Program.

Once, it is properly config, then There are 3 levels to test gpio interface.

ChmPE242 Feb 21, 22

22

Note : Command Line for GPIO Testing.

\$echo 79 > /sys/class/gpio/export

\$ echo out > /sys/class/gpio/gpio79/direction

\$echo 1 > /sys/class/gpio/gpio79/value

\$echo 0 > /sys/class/gpio/gpio79/value

\$echo 79 /sys/class/gpio/unexport

\$cat /sys/kernel/debug/gpio

Fig. 7.

Note: "Buffer Stage", e.g. A Transistor Based Current Amplifier Circuit. for future discussion.

```
#start
import RPi.GPIO as GPIO
import time #use for delay
```

```
GPIO.cleanup()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12,GPIO.OUT)
GPIO.output(12,1) #turn on GPIO at pin12
```

```
time.sleep(5) # sec
GPIO.output(12,0) #turn off GPIO at pin12
#end
```

Adding p2n 2222 transistor to drive GPIO load  
<https://www.jetsonhacks.com/2019/06/07/jetson-nano-gpio/>

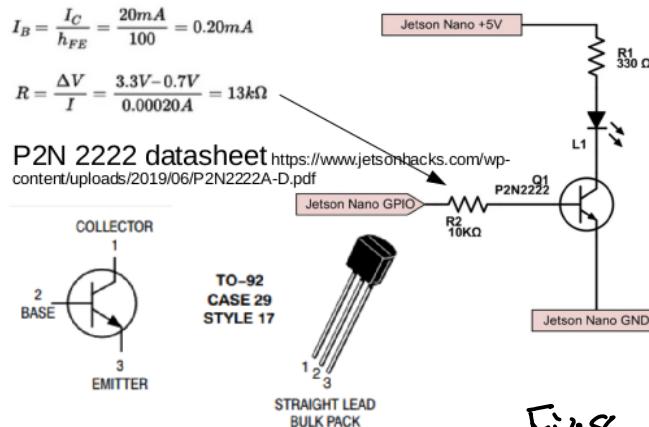


Fig. 8

Feb 23rd (Wed)

Today's Topics: GPIO

Example: Homework for Stepper motor Drive Testing

[2022s-105c-motor-drive-manual-updated-hl-2022-2-23.pdf](https://www.2022s-105c-motor-drive-manual-updated-hl-2022-2-23.pdf)

1. No MCU / No CPU Needed

## 2. Behavior of motor Drive

freq. Input Signal — Speed  
 Inputs { Directional Control Signal  
 Outputs (A+, A-, B+, B-)

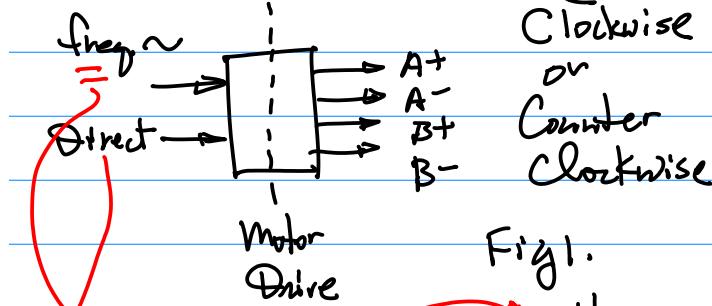


Fig. 1.  
a. Human Control Dial

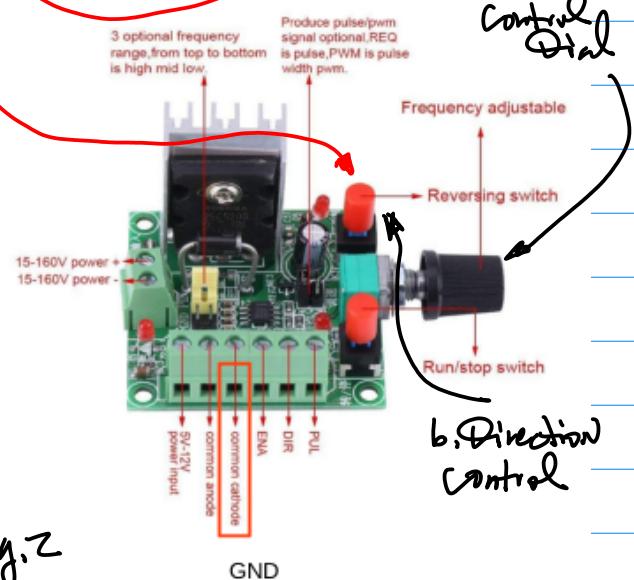


Fig. 2

b. Direction Control

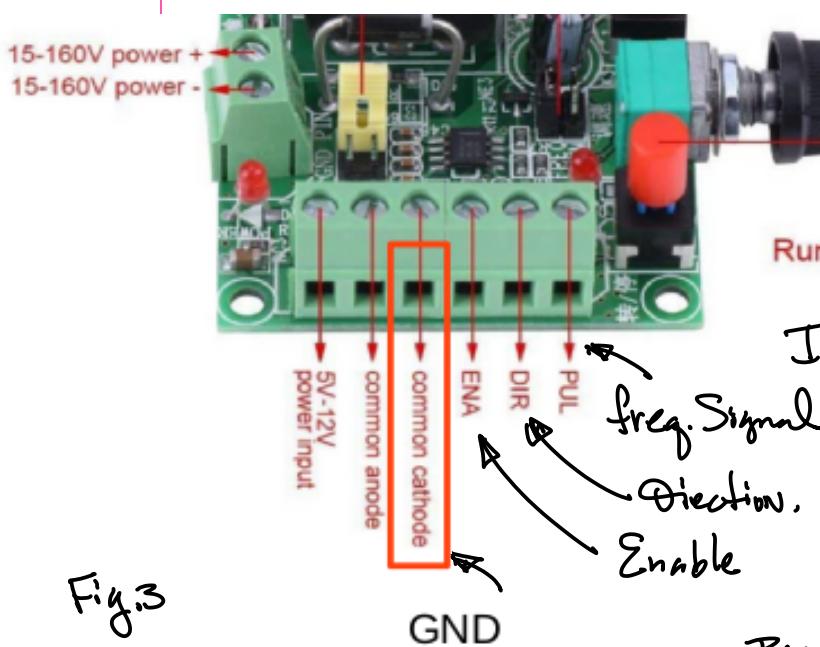


Fig. 3

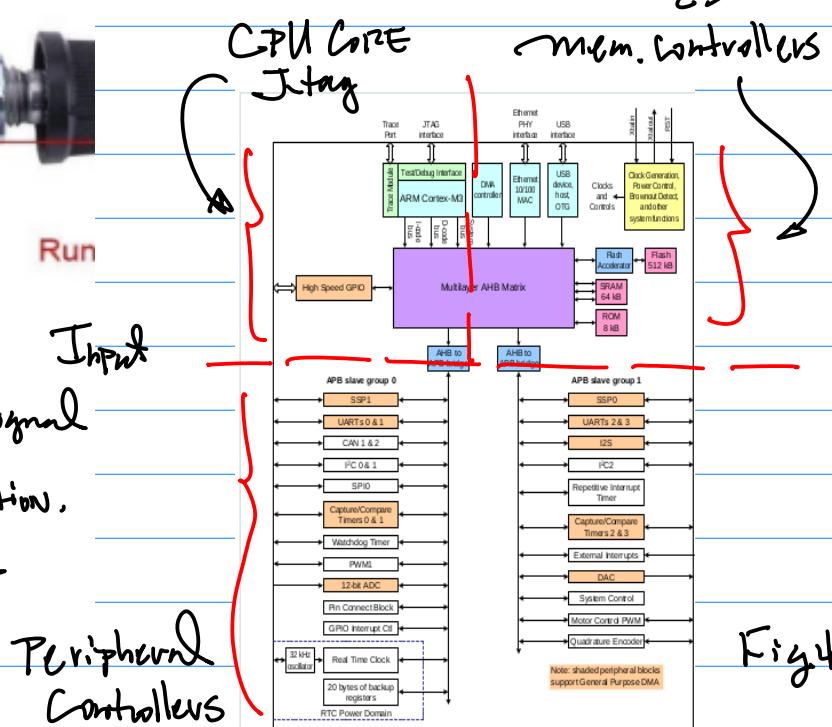


Fig. 4.

### Architecture Aspects of GPO

#### 1. CPU Block Diagram.

NXP L7G1769 ARM Cortex M-3  
~400MHz

Samsung ARM11 ~800MHz  
with Graphics Engine & Video Codec.  
~1.6GHz

Broadcom Pi3/3B+/4

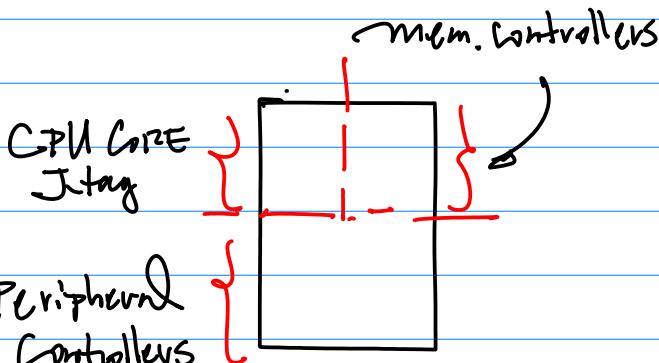
Jetson NAND (4 GBs + 128 GPU)

Baseline Ref

[CMPE244 / 2021F-105-#0-cpu-arm11-2018S-29-CPU\\_S3C6410X.pdf](#)

Use NXP L7G1769 as foundation, to Bridge the Concepts to ARM11 and Beyond.

[2021F-107-#0-cpu-UM10360.pdf](#)



#### 2. Memory Map.

NXP ARM 32 Bit  
Data & Addr. Bus — 32 Bits

$\alpha_3 \alpha_2 \dots \alpha_1 \alpha_0$   
addr[31:0]  
MSB      LSB

"Little Endian"

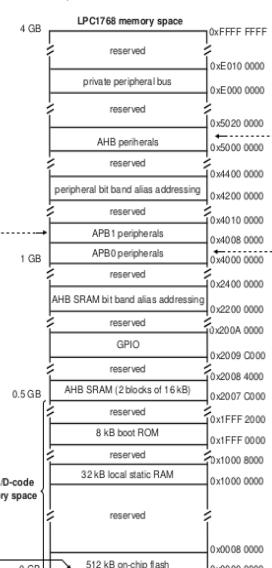
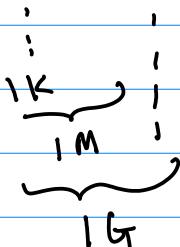


Fig. 5

$$2^{32} = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 2^2 = 4 \text{ GB}$$



**Memory Banks:**

A Block of memory.

$$2^{32}/8 = 2^{32}/2^3 = 2^{32} \cdot 2^{-3}$$

$$= 2^9 = 2^{10} \cdot 2^{10} \cdot 2^9$$

= 512 MB for each Bank.

**Address for Each memory Bank**

3 bits from the Addr. Bits to define 8 Banks

3 Most Significant, Consecutive Bits

$A_3 A_2 A_1 A_0$

0	0	0	0
0	0	1	0
:			
1	1	1	0

BANK0	1st
BANK1	2nd
BANK7 8th	

Addr. for the 1st BANK, BANK0 = 0x0000-0000

" " 2nd .. , BANK1 = 0x2000-0000

" " 3rd .. , BANK2 = 0x4000-0000

3. Peripheral controllers

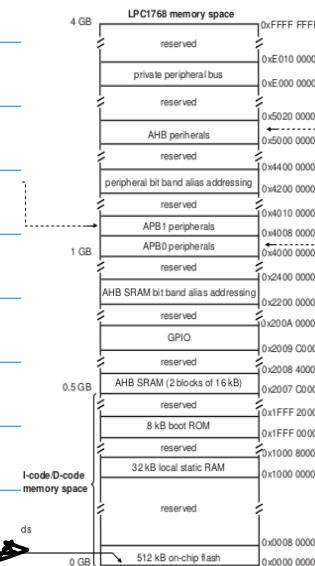


Fig. 6

Power Up  
Address 0x0000-0000

The Addr. for GPIO Controller

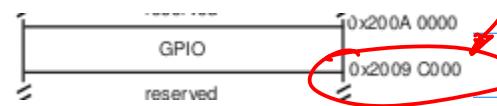


Fig. 7

0x2009-C000

use CPU Block Diagrams to populate  
memory space.

Feb 28 (Monday)

1. Check 4-Person Team Formation

2. Target Board

Billy, Barbor, Kain, Samir

Michael, Jessie, Andrew

Sourab, Xinyu

GPIO Block for ARM11, LPC1769

Naming Convention:



Size of the GPIO Block:

Fig. 1

$$\begin{aligned} & 0 \times 200A\_0000 \\ - & 0 \times 2009\_C000 \end{aligned}$$

Which gives the size of this  
GPIO Block.  $\rightarrow$  Memory for  
GPIO Controller, which  
consists of Special Purpose  
Registers. SPRs

32 bits Architecture, SPRs are  
32 bit.

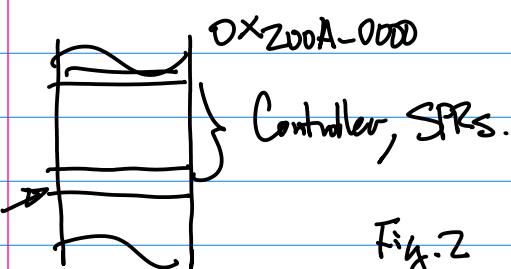


Fig. 2

Question: How many Special  
Purpose Registers are there  
to fit this memory block?

Hint: <sup>a</sup> find size. <sup>b</sup> size/4 = N.

of Special Purpose Register

Special Purpose Registers (GPIO) } Control/Configuration Register  
for Init & Config.  
Data Registers, Port for Data I/O.  
Pull-up/Down Registers.

Prefix + Root + Postscript  
3 letters 3 letters 3 letters

Example: Control/Configuration Register

Question: Design A Special Purpose  
Register for Control/Configuration By  
Selecting 3 letters?

GPP Cn → GPPCn  
Prefix Root

Question: Design By Selecting 3 Letters  
for GPIO Port? (Naming Convention)

Note: For One more than one GPIO Port,  
we will modify the prefix to  
make it fit for this type architecture

Port A, Port B, Port C, ...

Question: For Each Port what is the  
max possible pins? 32 bit Architecture,  
hence, Max Possible pins is 32

Cmpe242 Feb 28, 22

2b

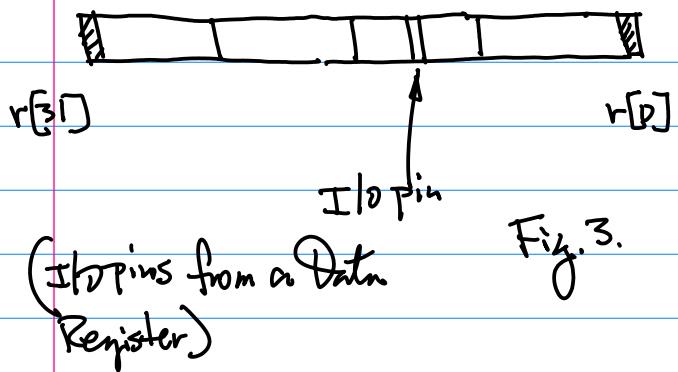


Fig. 3.

CPU Datasheet, Ref. from github ~ Z44

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

SAMSUNG  
ELECTRONICS

## USER'S MANUAL

Manufacturer Name  
ID

S3C6410X  
RISC Microprocessor

a Chapter ID      b. CPU (ARM11), c CPU  
6410X\_UM

Naming      d. No. of

GPP. 17 GPP Ports.

e. Naming of Each  
GPIO Port, A, B, ...

f. 17 pins for Each  
Port

### 10.1 OVERVIEW

S3C6410 includes 187 multi-functional input/output port pins. There are 17 ports as:

Port Name	Number of Pins.	Muxed pins
GPA port	8	UART/EINT
GPB port	7	UART/IrDA/I2C/CF/Ext.DMA/EINT
GPC port	8	SPI/SDMMC/I2S_V40/EINT
GPD port	5	PCM/I2S/AC97/EINT
GPE port	5	PCM/I2S/AC97
GPF port	16	CAMIF/PWM/EINT
GPG port	7	SDMMC/EINT
GPH port	10	SDMMC/KEYPAD/CF/I2S_V40/EINT
GPI port	16	LCD

Special-purpose Registers: a. Control/Configuration Register, b. Addr. of the Special-purpose Register

6410X UM

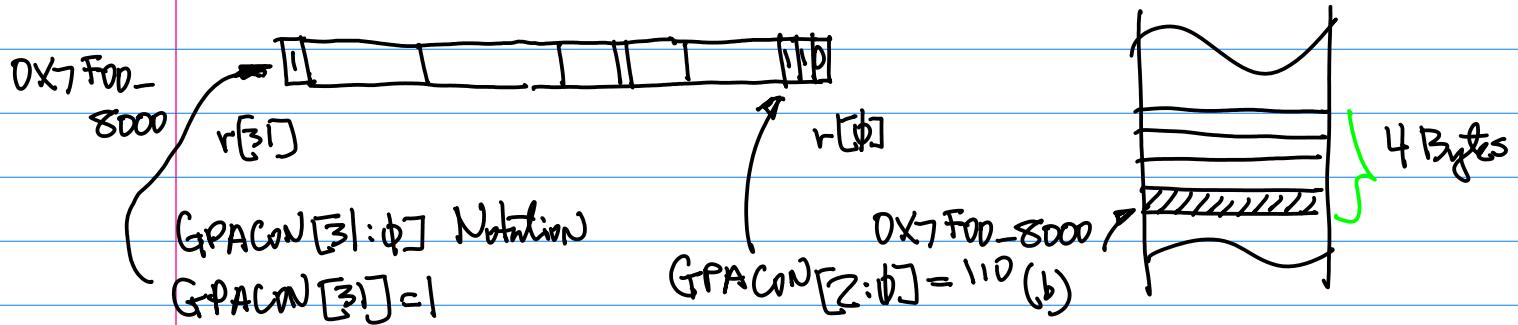
10.4 REGISTER DESCRIPTION

10.4.1 MEMORY MAP

Register	Address	R/W	Description	Reset Value
GPACON	0x7F008000	R/W	Port A Configuration Register	0x0
GPADAT	0x7F008004	R/W	Port A Data Register	Undefined
GPAPUD	0x7F008008	R/W	Port A Pull-up/down Register	0x000005555
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Configuration Register	0x0
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-up/down Register	0x0

Question: How many functions are possible for a Control/Config Register?

$2^{32}$ !



C. Data Register: GPADAT <sup>8</sup>/pins for GPIO Operation

From CPU Datasheet, PP.313

~ 8 pins

GPADAT	Bit	
GPA[7:0]	[7:0]	W
		W
		cc
		ve

6410X UM

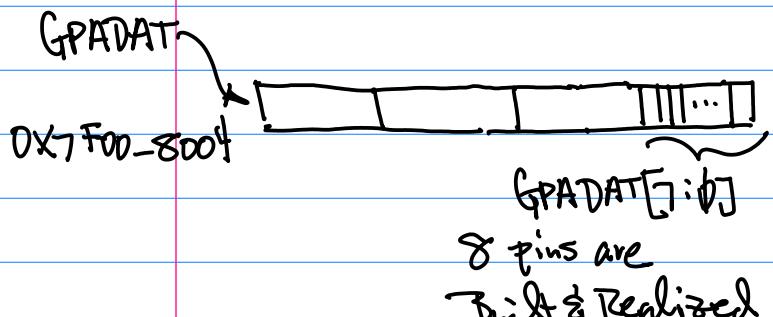
# 10

GPIO

## 10.1 OVERVIEW

S3C6410 includes 187 multi-functional input/output port pins. There are 17 ports as

PortName	Number of Pins.	Muxed pins
GPA port	8	UART/EINT
GPB port	7	UART/IrDA/I2C/CF/Ext.DMA/EINT
GPC port	8	SPI/SDMMC/I2S_V40/EINT
GPD port	5	PCM/I2S/AC97/EINT
GPE port	5	PCM/I2S/AC97
GPF port	16	CAMIF/PWM/EINT
GPG port	7	SDMMC/EINT
GPH port	10	SDMMC/KEYPAD/CF/I2S_V40/EINT
GPI port	16	LCD



Example: Suppose ARM-11 CPU is employed for I/O IF Design.

The following Design Requirements are given:

1° Use GRADAT[1] as output pin;

use GRADAT[3] as input pin;

2° Config the GPA Controller

By properly define GPA Con.

3° Suggest A way of using C code to realize this task?

March 2nd

1. Project on Stepper motor Drive

& PID Controller Design & Implementation.

Due March 21st Monday, 11:59pm.

2. Requirements for the project:

(1) Prototype Board with the target platform NANO or Tie:

(2) GPIO testing CKT & Code for testing function.

(3) PWM Output for Stepper motor Drive control.

Testing Circuit for the PWM Output in addition to Motor Drive control.

(4) Stepper motor and motor Drive, Operated under target Board Control, Both f<sub>PWM</sub>, Duty Cycle are Controllable by the target Board.

(5) Other Software Processing function.

I2C Interface Capability for LSm303 Sensor interface.

3. Take photos of the tasks from 1-2

(1) photo of the prototype (wire wrapping) Board;

(2) photo of GPIO Testing Result;

(3) photo of PWM, Stepper motor, motor Drive.

(4) 10~15 second Video Recording of Board Control of Stepper motor.

(5) Source Code & Binary of your implementation.

(6) Put all in pdf file,

Except Source Code, Binary are Standalone files, And Video Clip

(7) Zip all the above into one zip file, Submit to SJSU CANVAS.

4. Written Requirements of these will be posted on github and CANVAS.

Note: LSm303 sensor, ADC Birth Should be ready.

Homework → Due A week from Today.

2022S-101b-2homework-prototype-board-menuconfig-2...

March 9th (Wed) 11:59 pm.

Example: Continued. ARM11 CPU  
Datasheet.

#### 10.5.1 PORT A CONTROL REGISTER

There are five control registers including GPACON, GPADAT, GPAPUD, C Port A Control Registers.

Register	Address	R/W	Description
GPACON	0x7F008000	R/W	Port A Configuration Register
GPADAT	0x7F008004	R/W	Port A Data Register
GPAPUD	0x7F008008	R/W	Port A Pull-up Register
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Configuration Register
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-up Register

1.



GPAON	Bit	Description	
GPA0	[3:0]	0000 = Input 0010 = UART RXD[0] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [0]
GPA1	[7:4]	0000 = Input 0010 = UART TXD[0] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [1]
GPA2	[11:8]	0000 = Input 0010 = UART CTSn[0] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [2]
GPA3 4th pin	[15:12]	0000 = Input 0010 = UART RTSn[0] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [3]
GPA4	[19:16]	00 = Input 10 = UART RXD[1] 0100 = Reserved 0110 = Reserved	01 = Output 11 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [4]
GPA5	[23:20]	0000 = Input 0010 = UART TXD[1] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [5]
GPA6	[27:24]	0000 = Input 0010 = UART CTSn[1] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved 0111 = External Interrupt Group 1 [6]
GPA7	[31:28]	0000 = Input 0010 = UART RTSn[1] 0100 = Reserved	0001 = Output 0011 = Reserved 0101 = Reserved

2. GPAφ → Pin → GPADAT



Note with (if) connector, pin  
then physical connection  
can be established.

3. GPACON[3:0] Controls the  
Configuration of GPAφ pin.

ARM CPU pin is multiplexed

General Purpose I/O  
function → GPAφ  
multiplex

Set GPACON[3:0] = 0x1, Output pin  
GPACON[3:0] = 0x2, UART  
GPACON[3:0] = 0x3, Input pin

Note: By Default, when CPU powered up,  
GPACON[3:0] = 0x0, hence, it is set  
as GPIO Input.

Design Spec. → Init & Config.  
Binary Pattern.

Example: Design GPIO Interface,  
By making the 1st GPA pin as an  
input, the 4th GPA pin as an output.  
Find the Binary Pattern to perform  
init & Config.

$\text{GPACON}[3:0] = 0x0$  for Input

$\text{GPACON}[5:4] = 0x1$  for Output

$\text{GPACON}[3:0] = \text{GPACON}[3:28] \dots \text{GPACON}[5:4] \dots \text{GPACON}[3:0]$

$0x8$   
 $\overbrace{1 \ 0 \ 0}^{\text{(1st)}} \ 0 \ \overbrace{0 \ 0 \ 0}^{\text{(4th)}}$

To make this configuration,

$\& \text{GPACON} = 0x8;$  // Configuration of Pin0 & Pin3

(1st) (4th)

March 7th (Monday)

Topics: 1° CM Register Init &

Configuration, & Codes ; 2°

PID control for Embedded platform.

Announcement: 1° Project,

Phase I PID controller. Due March 21

2° Midterm Exam: March 23d. Online

Zoom Exam.

Example: Code Sample (Reference: ARM)

User Program      (Same code)

```
root@harri-v-laptop:/opt/FriendlyARM/minitoolbox/linux/exam
2022s-104d-userSpace-gpio.c led led.c led.c~ Makefile
root@harri-v-laptop:/opt/FriendlyARM/minitoolbox/linux/exam
```

Device Driver from User Space Program

`fd = open("/dev/leds0", 0);` Note See Next page Note 1.

`ioctl(fd, on, led_no);`

Then, the Kernel Space Device  
Driver program.

a. ARM11 CPU  
Samsung

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char# ls mini6410*
mini6410_adc.c      mini6410_hello_module.c    mini6410_leds.c      mini6410_pwm2.c
mini6410_adc.mod.c  mini6410_hello_module.ko   mini6410_leds.ko    mini6410_pwm2.mod.c
mini6410_adc.o      mini6410_hello_module.mod.c mini6410_leds.mod.c mini6410_pwm.c
mini6410_buttons.c  mini6410_hello_module.mod.o mini6410_leds.mod.o mini6410_pwmHarry.c
mini6410_buttons.o mini6410_hello_module.o     mini6410_leds.o    mini6410_pwm.mod.c
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char#
```

b. ADC (Analog-to-Digit  
Conversion)

Note: 1. Name of the Device (Driver)

```
#define DEVICE_NAME "leds0"
```

c. GPIO Device Driver

Note 2. "ioctl" module

```
static long sbc2440_leds_ioctl(struct file *filp, unsigned int cmd, unsigned long arg)
{
    switch(cmd) {
        unsigned tmp;
        case 0:
        case 1:
            if (arg > 4) {
                return -EINVAL;
            }
            tmp = readl(S3C64XX_GPKDAT);
            tmp &= ~(1 << (4 + arg));
            tmp |= ( (!cmd) << (4 + arg) );
            writel(tmp, S3C64XX_GPKDAT);
            //printk (DEVICE_NAME": %d %d\n", arg, cmd);
            return 0;
        default:
            return -EINVAL;
    }
}
```

a. readl( )

Special purpose Register  
Naming: Manufacturer ID + Special  
Purpose Register

G P K D A T

G P R D A T  
General Purpose Register  
Port K Register

Note 3: a. Init

```

static int __init dev_init(void)
{
    int ret;
    {
        unsigned tmp;
        tmp = readl(S3C64XX_GPECON);
        tmp = (tmp & ~(0xffffU<<16)) | (0x1111U<<16);
        writel(tmp, S3C64XX_GPECON);

        tmp = readl(S3C64XX_GPEDAT);
        tmp |= (0xF << 4);
        writel(tmp, S3C64XX_GPEDAT);
    }

    ret = misc_register(&misc);
    printk (DEVICE_NAME"\Harry: PGE initialized\n");
    return ret;
}

```

Defined in ".h" files to map peripheral controllers to their memory location

Find its Address:

CPU Datasheet

0x1111 U  
0001 0001; 0001 0001

0xffff  
1111 1111; 1111 1111

↓ Negation ~

↓ " & " Bitwise AND

Clear all Bits

6410X\_UM

#### 10.5.5 PORT E CONTROL REGISTERS

There are five control registers including GPECON, GPEDAT, Port E Control Registers.

Register	Address	R/W	
GPECON	0x7F008080	R/W	Port E Co
GPEDAT	0x7F008084	R/W	Port E Da
GPEPUD	0x7F008088	R/W	Port E Pu
GPECONSPL	0x7F00808C	R/W	Port E Sle
GPEPUDSLP	0x7F008090	R/W	Port E Sle

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

Which Memory Banks for this SPR?

Consider PID Controller Design  
For Embedded Platform.



Fig. 1a

NEMA  
Stepper  
motor  
For Example

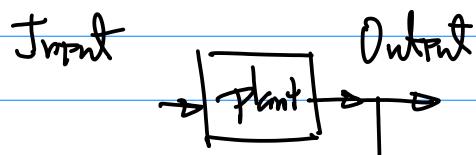
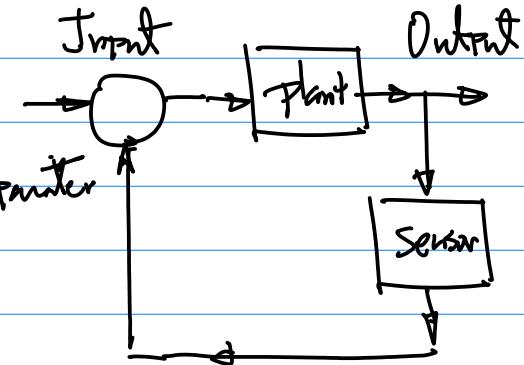
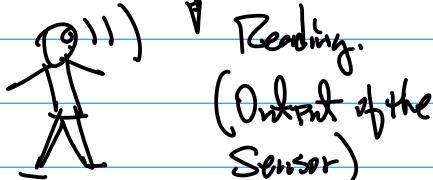


Fig. 1b



- 1° A "set-point" of an output
- 2° Computer (Input & Output)
- 3° Sensor Reads one physical (angle) quantity, produce Output of another quantity (electric)  
Transform + Produce  $\rightarrow$  "Transducer"

Computation: Subtraction (Difference)

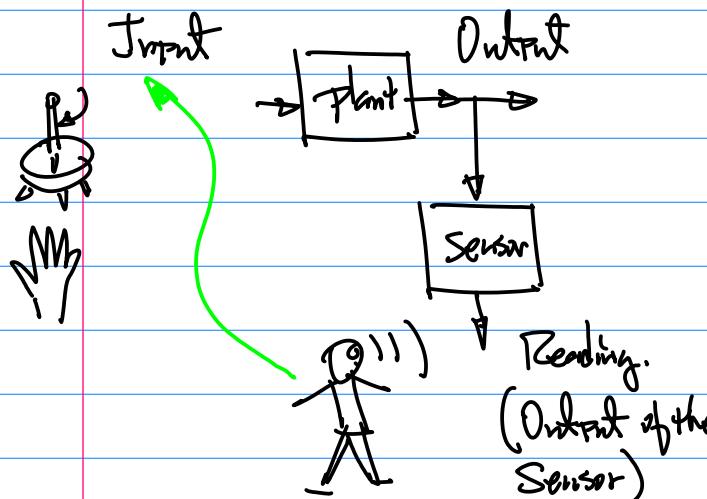
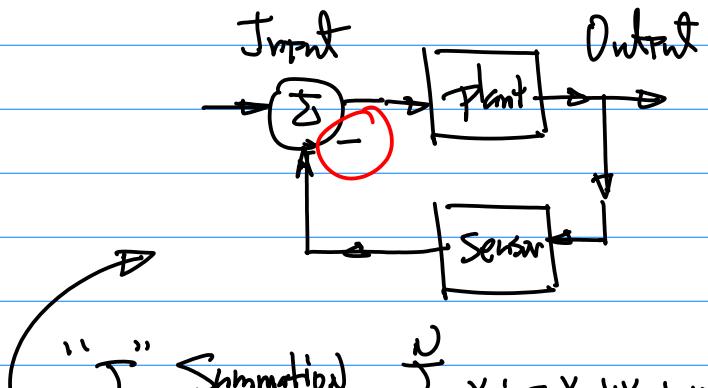


Fig. 1c

2022S-107a-PID-v5-2019-4-25.pdf  
2022S-107b-pwm-pid.pdf



Feed Back Loop  
Negative Feedback:  $\Sigma = \text{Input} + (-\text{Readout from a Sensor})$

$$\sum_{i=1}^n x_i = x_1 + x_2 + \dots + x_n$$