CmpE242 Spring 2022

1

Jan 26 (Wed) First Day of the Class.

Harry Li,
E-mail: hua.li@sjsu.edu
Text message: (650)400-1116

Office Hours: M.W. 4:30-5:30 P.M.

Textbooks + References:

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

1. ARM11 CPU Datasheet, from Samsung & SCH Document for the development Board.
2. NVDA 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 Pie, BCM2835 (CPU Datasheet).

Selection of Target platform for this Course.

a. NVDA Jetson NANO
b. BroadCom Pie
c. SamSung ARM CPU
d. NXP i8

Note: Select your target platform from the options a – d. (Consider Nvidia Jetson NANO).

Programming Languages: C/C++, python

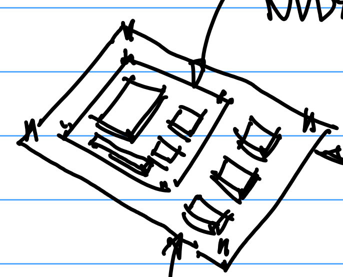
O.S. Support: Linux.

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.

NVDA Jetson NANO



Carrier Board.

Stepper motors Drive...

2. Form 4 person team, Work on Homework, Project, However All Coding, Report etc have to be complished 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, @ End of the Semester, Requires PPT Presentation & Live Demo.

Announcement in Class, in Written form Both in the Lecture Notes and ON SJSU Canvas.

Late Projects / Homework 1 pt Penalty.

Jan 31 (Monday)

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

Target platform to Consider for your Prototype:

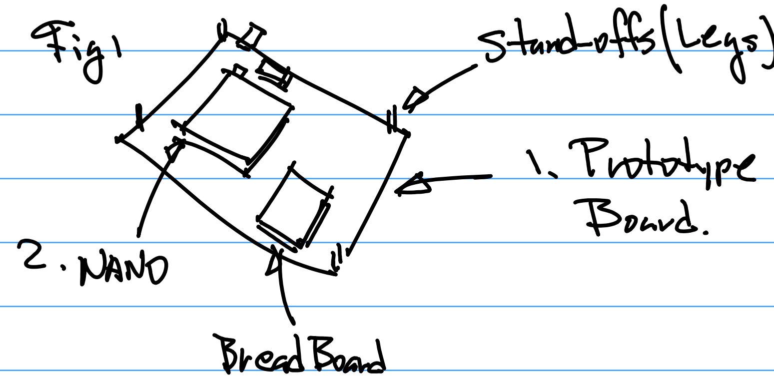
Nvidia Jetson NANO, 2GB

Broadcom Pie. 3B+, 4.

Note: Jetson NANO is Better And more powerful with Almost the Same cost.

Bill of Material for Prototype System

Fig 1



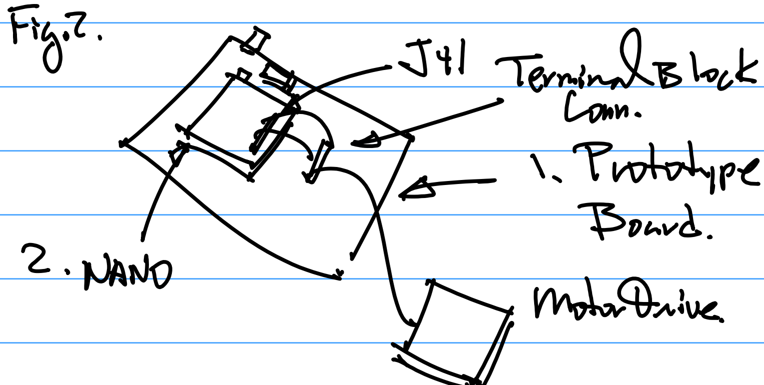
Note: a. Adequate size to host CPU-module (NVDA Jetson NANO) 16 × 11½ Cm.

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.



Prefab "Through Holes" with Coating for Soldering.

Stand-offs (Legs)

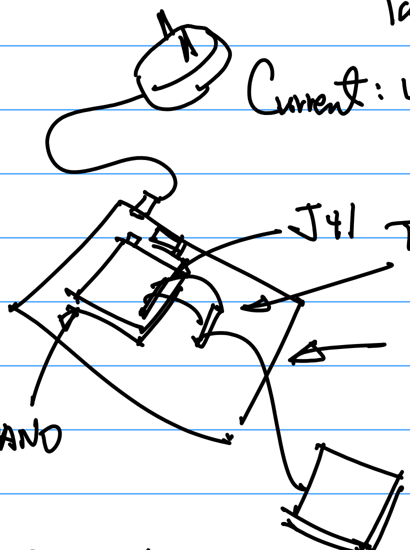
Components for:

External Power Unit

GPIO Testing Circuit

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

Fig.3



Current: 4000mA + 2000 ~ 4000mA

J41 Terminal Block Conn.

1. Prototype Board.

2. NANO

Motor Drive

Current

Note: Adaptor To Provide Power Not just to the Target platform, But also to the entire Board

NANO Target Requires 4A (4000mA) Current for Peak Computing, On top of it, you will need to consider providing Adequate Current for Glue Logic, for Sensor I/F (LSM303), for Stepper Motor Drive. ①

Note: IC DC Regulator is Needed.

Make sure DC Regulator IC Can Handle Adequate current

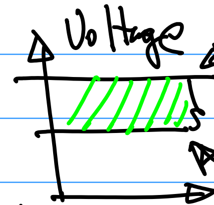
Consider 7805 as an example. Only 1500mA is allowed, No good

information

-voltage integrated-circuit voltage regulators is designed for a wide range of
include on-card regulation for elimination of noise and distribution problems ass
tion. 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)

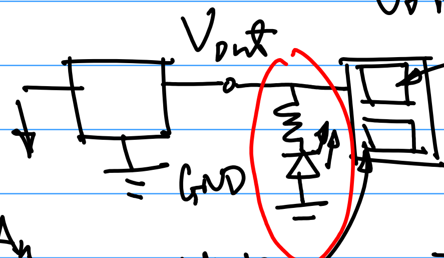
Note: Some power regulator will have Output voltage Drop.



Input Volt

Output Voltage

Fig 4.



Vin (To the Output of An Adaptor)

Others

NANO

Prototype Board

② Red LED, $V_{LED} \simeq 1.2V$, $I_{LED} \simeq 10mA$

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

GPIO Testing
- Input
  - "0"s
  - "1"s
- Output
  - "0"s
  - "1"s

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 & more.

Address: 2040 Walsh Ave, Santa Clara, CA 95050

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

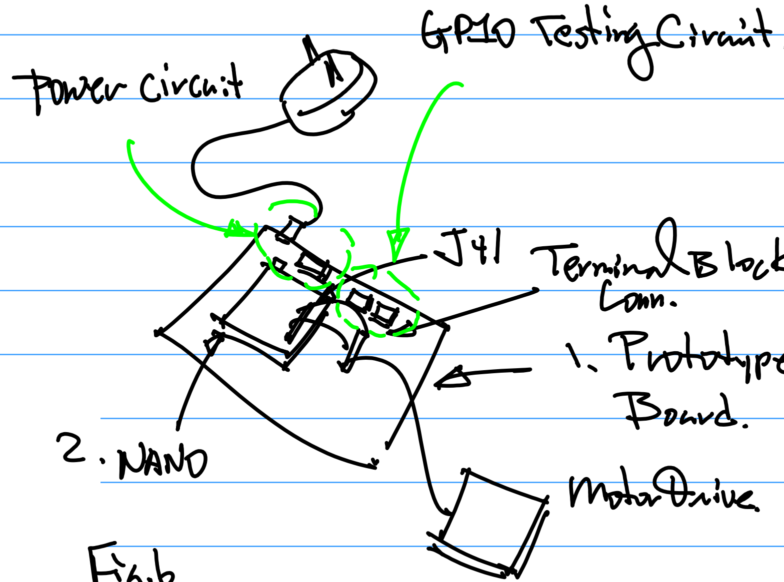
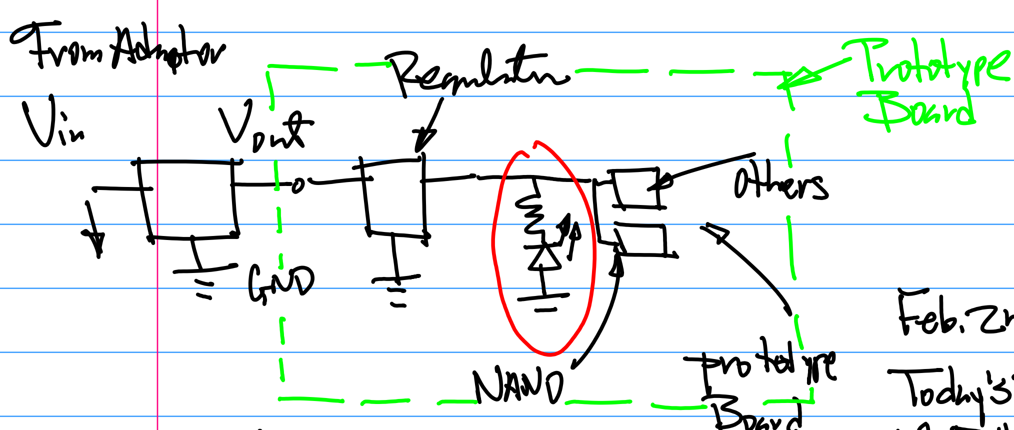


Fig.6

Input Testing:
1. Toggle s/w.
2. Assorted Resistors

(10 Ohms ~ A few Hundred Ohms)

Output Testing:
1. Color LED (Red, yellow, Green)
2. Assorted Resistors.



Components for GPIO Testing.

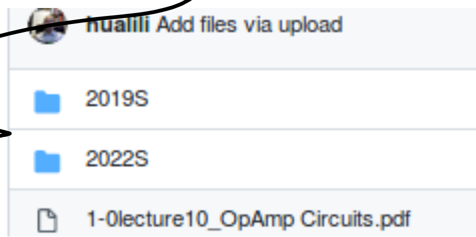
Feb. 2nd (Wed)

Today's Topics:

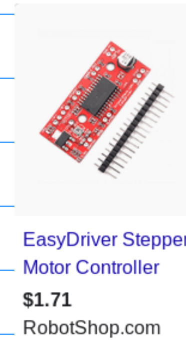
1° Bill of material (Continuation)

2° Prototype System Design.

Class github

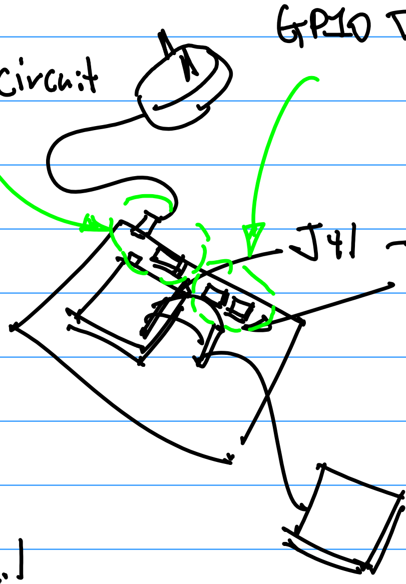


Option B for the motor drive
"Easy Drive" a. ~1000mA.



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

Power Circuit



J41 Terminal Block Conn.

Motor Drive

Fig. 1

Ref. for Stepper motor Drive
https://github.com/hualili/CMPE244

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

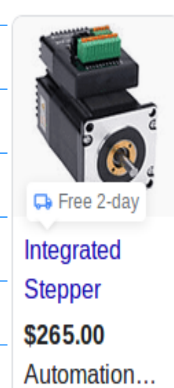
For CNC, 3D printer, for Self-Driving Robot

Motor Drive for Stepper motors.
Option A. Professional Drive
Option B.

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

Stepper Motor Controller 1 15 USD

https://www.amazon.com/gp/product/B01DIK5IRI/ref=oh_aui_detailpage_o02_s00?ie=UTF8&psc=1



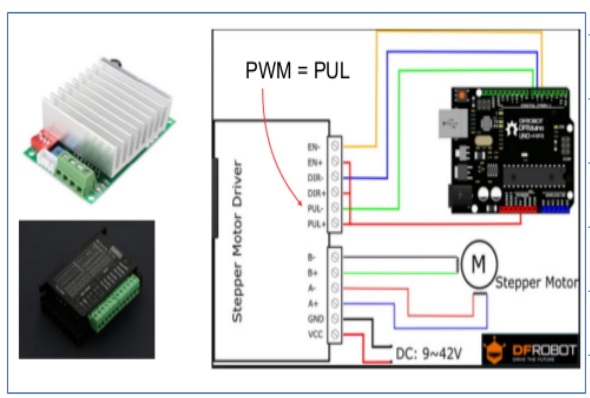


Fig. 2

~$15 – $40

Note: Specs for Option A: a. 4.5A. (2500mA ~3000mA for this class & Beyond) b. TB66xx (IC Chip Provides Adequate Power)

Bill of material (motor)

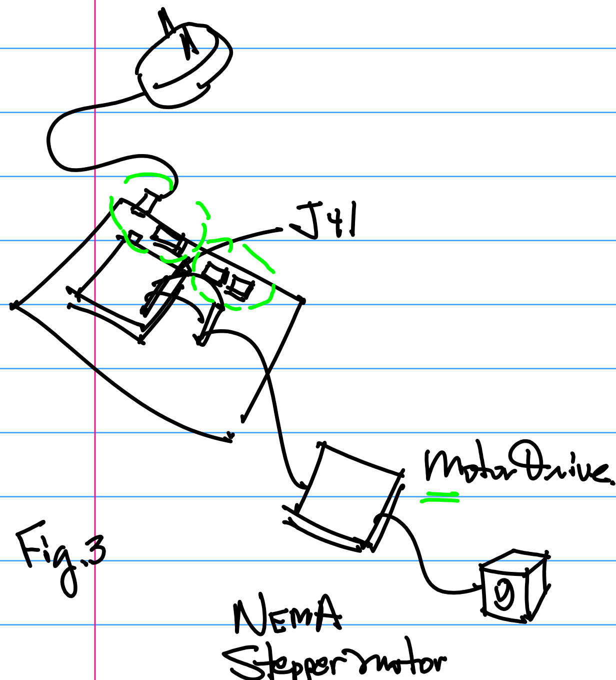


Fig.3

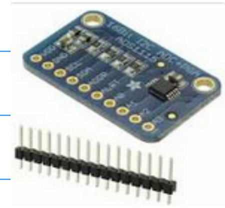
Buying LSM303 module

Bill of material (IOT Aplications
Analog Sensor I/F)

ADC module
(Analog/Digital Conversion)

{ I2C
{ SPI (Serial peripheral Interface)

Adafruit 1018
Acceleration Sensor
$17.95
Mouser Electronics



Adafruit Industries
LLC 1085 ADS1115
$14.95
Digi-Key

Adafruit Industries
LLC 4648 PCF8591
$6.50
Digi-Key

https://www.anaheimautomation.com/marketing/stepper/stepper-motors.php?gclid=EAIaIQobChMIqsaW3pbi9QIV-R6tBh1QBAFsEAAYAiAAEgIhx_D_BwE

Stepper motor
- NEMA-17
$14.00
Adafruit Ind...

Anaheim Automation

Fig4.

Selection Guideline:
1. Digital Interface Protocol (CPU and ADC Interface);

SPI: 50+ mbps, I2C: 2-4 mbps

Bill of Material (Sensors) for PID

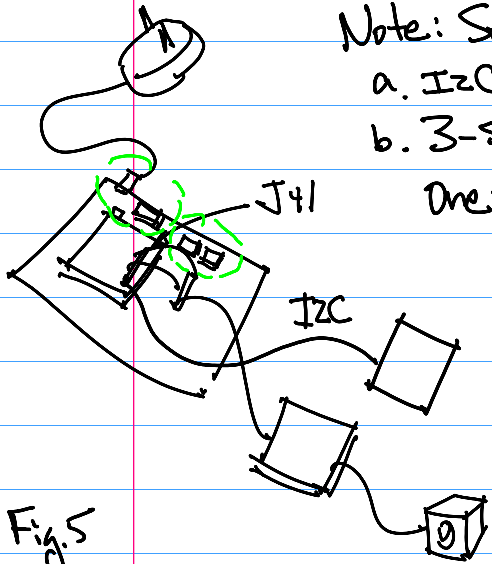


Fig.5

Note: Sensor LSM303
a. I2C Sensor Interface;
b. 3-sensing units in
One package

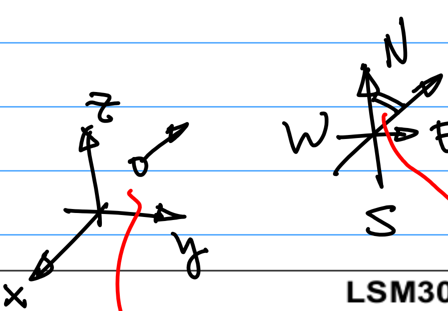


Fig.6

LSM303

LSM303DLHC

life.augmented

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

Datasheet - production data

② ①

including Temperature
Sensing (the 3rd one)

CmpE242 Spring22

7

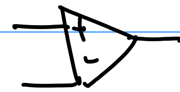
Selection Guideline (Continued)

2. Sample Rate:
   500KSPS or 1 MSPS (million Samples per Second)
3. Quantization Level of ADC.
   Bits per Sample. 8~10 bit Resolution minimum;

Nyquist Sampling Theorem.

$$f_{sample} \geq 2 f_{max} \quad \ldots (1)$$

Note: Optional Component —— OpAmp.
Operational Amplifier



Single OpAmp.

Or LM358 (Quad OpAmps in One Package).

Homework: Due A week from Today. (Feb. 9) 11:59pm

1. Identify your target platform, And provide A screen Capture or photo of its Expansion connector, with Table of pin Assignment;
2. Create A table of Bill of material Based on Lecture Discussion.
3. Submission: A pdf file with Naming: First_Last_SID_Target_242.pdf. Submission to CANVAS.

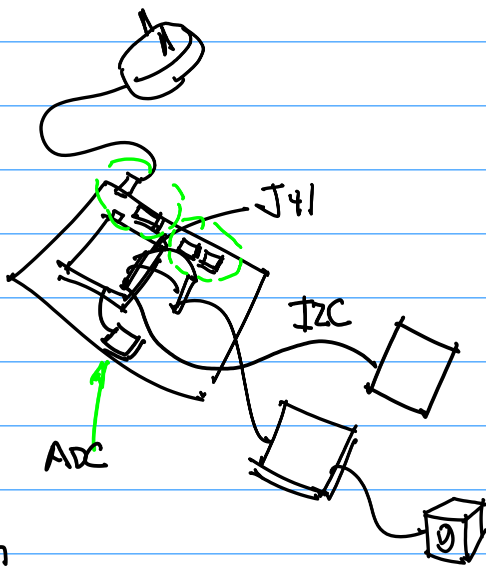
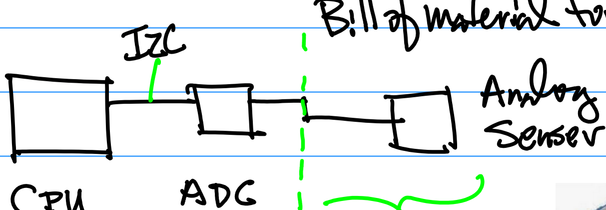


Fig.7

Bill of material for ADC.



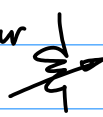
Feb 7 (Mon)

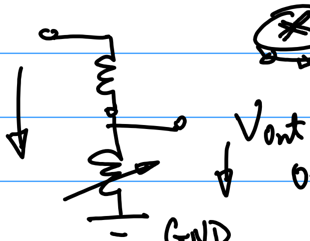
Today's Topics:

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

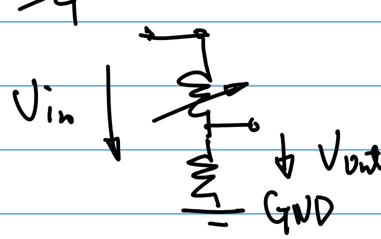
Fig.8

Potential meter





OR



Target Platforms { a. Jetson NANO; b. Pie3B+/4.

Note: Base Line Reference Samsung ARM11.

Ref. CPU Datasheet of Samsung ARM-11 is on the Class github. (Cmpe244)

2021F-105-#0-cpu-arm11-2018S-29-…

PPT for Today's Lecture

Example: Bring Up The target Platform, NANO.

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° Down Load Pre-Compiled, Pre-Built Kernel Image to SD Card And

4° Target Board, Jetson NANO 2GB OR 4GB.

Step 1. Down Load Pre-Build Kernel Image

Step 1. Down load 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:

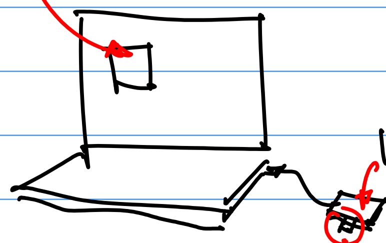
https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#write

Step 2. Down Load "Ethcher" Software tool, then execute this tool to write the Down Loaded Pre-Built Kernel O.S. to your SD Card.

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

(2.1) for Linux host, Download, install, and launch Etcher.

https://www.balena.io/etcher/



USB Card Reader

SD Card to Be Written with Pre-Built Kernel Image.

Fig 1.

Step 3. Took 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 GPIO Testing. (To Be Assigned as Homework)

Hardware Aspect:

Step 1. Identify the Expansion Connector &

Pin Assignment.

From NVDA Developer Site to find Connector Information

J41 Connector

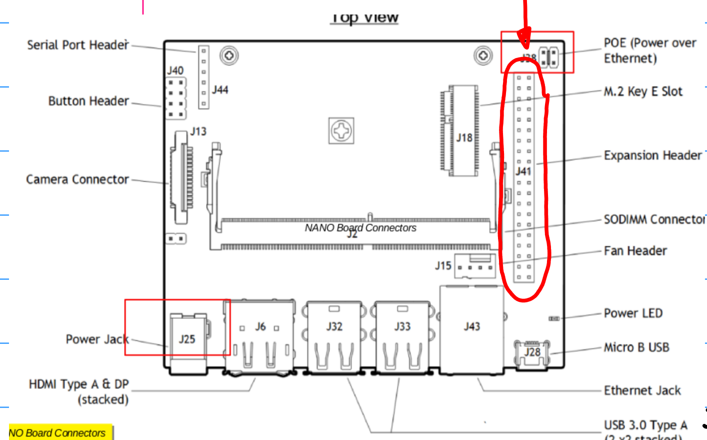


Fig.2

Table 1. Pin Assignment/Connectivity Table

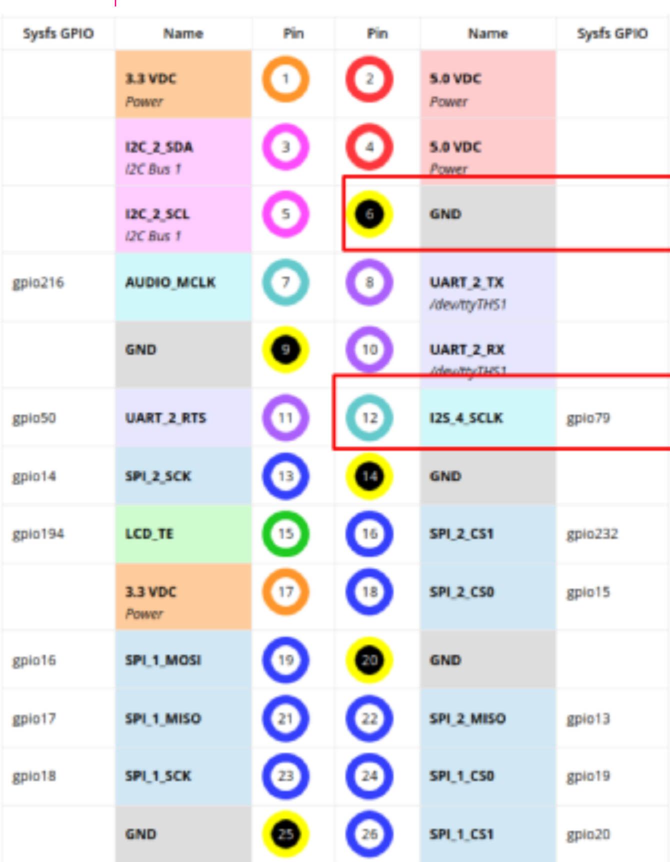
CPU Pin/Connector	Description	Note
GPIO79/J41-12	GPIO Input	
GPIO78/J41-40	GPIO Output	
GND/J41-6	GND	

Step3.

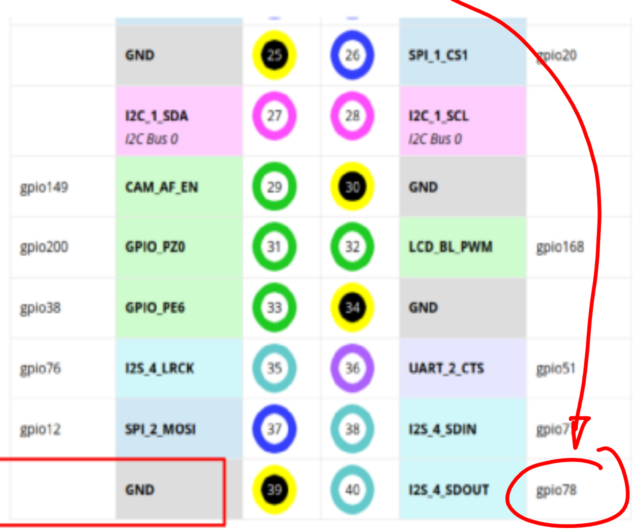
Design Schematics

Step2. Establish Pin Assignment/Connectivity Table

Pin Assignment: GND: Common GND; GPIO: GPIO79/pin12, GPIO78/pin40



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_PZ0	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	gpio7[illegible]
	GND	39	40	I2S_4_SDOUT	gpio78

i, Ph.D.

Fig.3

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_PZ0	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

…i, Ph.D.

Input Testing.

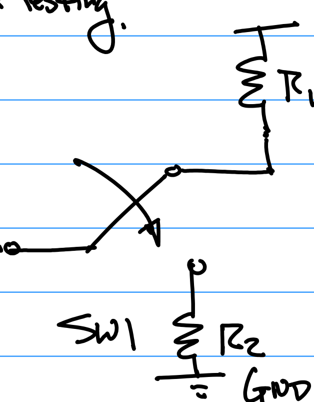
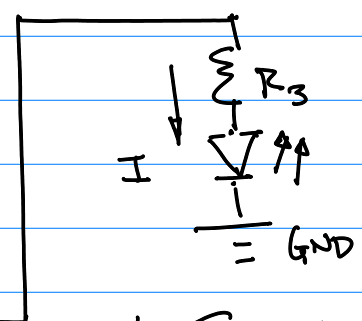


Fig. 4a

Let $I = 10mA$.

$R_1 = V_{cc}/I = 3.3/10\times10^{-3} = 330\Omega$, $R_2 = R_1$



$I \simeq 10mA$

Note: Since NANO 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.

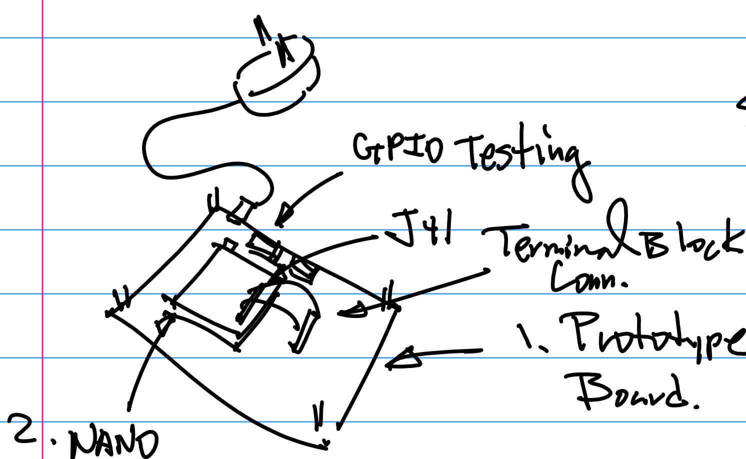
2° Kernel O.S. Sources plus toolchain

Example: Bring Up Your Target Board

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

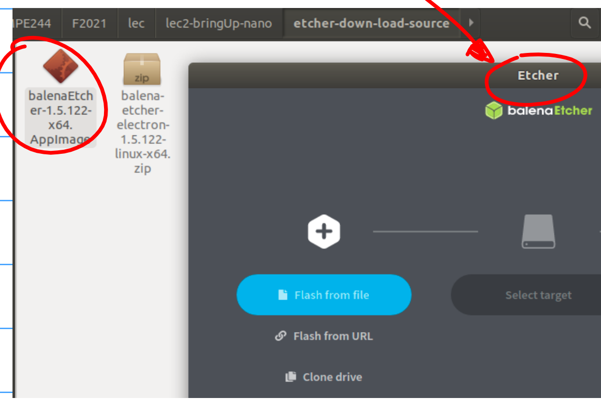
CMPE242 Feb 9, 22

Homework (Due A week from today)
~~Feb 16~~ Moved to Feb 28th
1. Prototype System with A Carrier Board 4"x3" or Similar Size of your Choice;
2. GPIO Testing Circuit for Both Input/Output Testing (schematics)
3. Mount your NANO on the Carrier Board.



4. Take a photo of your System (with A Host connection)
5. Screen Capture from your Host machine, which shows NANO is running.
6. Put Photos plus Schematic into ONE PDF file, Zip it, Submit to SJSU CANVAS.

Step 2. Down Load Pre-Build Kernel Image from NVDA Developer Site, And together with Flash Writing Software. "Etcher"



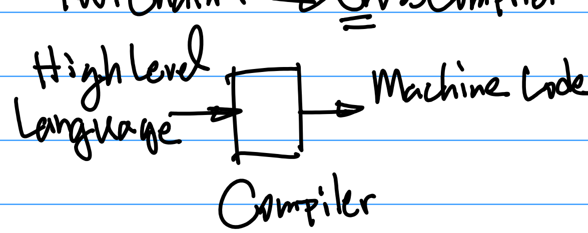
Step 3. Run "Etcher" program to Upload the O.S. Kernel Image to the target NANO Board.

Note: Target Image for my Board is 2019.4.7 Due to my Applications in Deep Learning. For Embedded Class, we can use other/Latest Release

2022S-105-#106-n-Jetpack-kernel-driver-simpler-2021-2

Jetpack from NVDA Consists of a. O.S. Kernel (Device Driver), b. GPU packages, c. OpenCV, d. DNN

O.S. (Operating Systems)
Tool Chain: → Cross Compiler



Step4. Take SD Card to the target NANO, Boot (Power Up) the NANO, finish Configuration process, then Ready to go. (To Create your Applications).

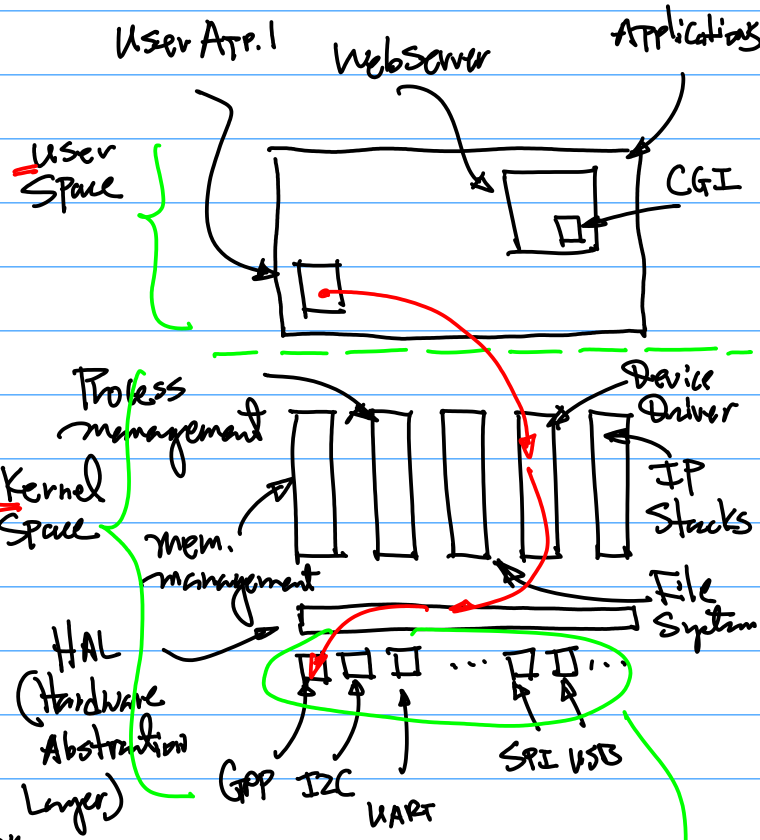


Fig. 1

Example: O.S. Kernel Source Debugging, Device Driver Development

O.S. + Tool Chain for NANO

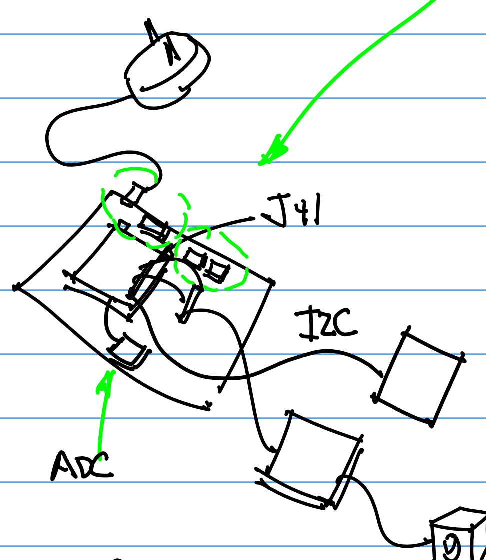
O.S. + Tool Chain for Samsung ARM11

Step1. Down Load 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, And 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, Devices Drivers to the Target NANO plus Carrier Board

Note: CGI Common Gateway Interface

Older, But Widely Adopted Technology for I/O Devices Interfaces.

CMPE242 Feb14, 22

Note: Software Architecture (Block Diagram) & Prototype System illustration (Fig.1) are required.

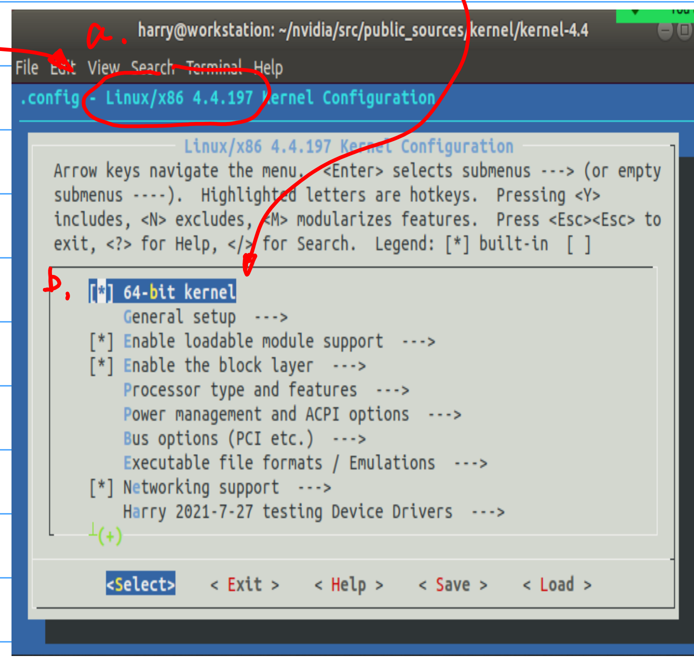
Example: Kernel Source Distribution & Tool Chain.

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

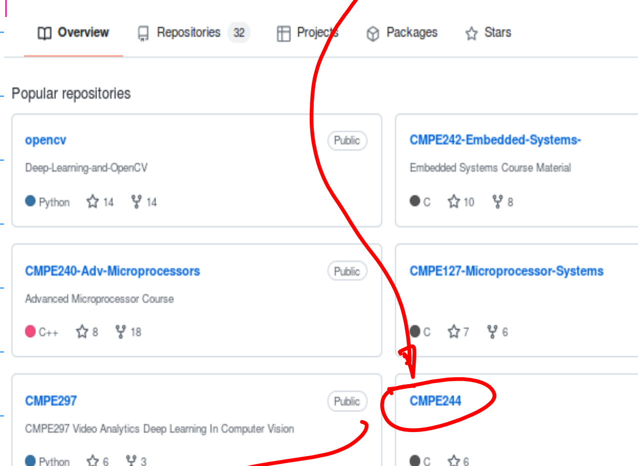
Target: NAND

Reference platform: Samsung ARM11.

a. Version of the config b. Target

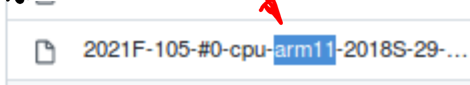


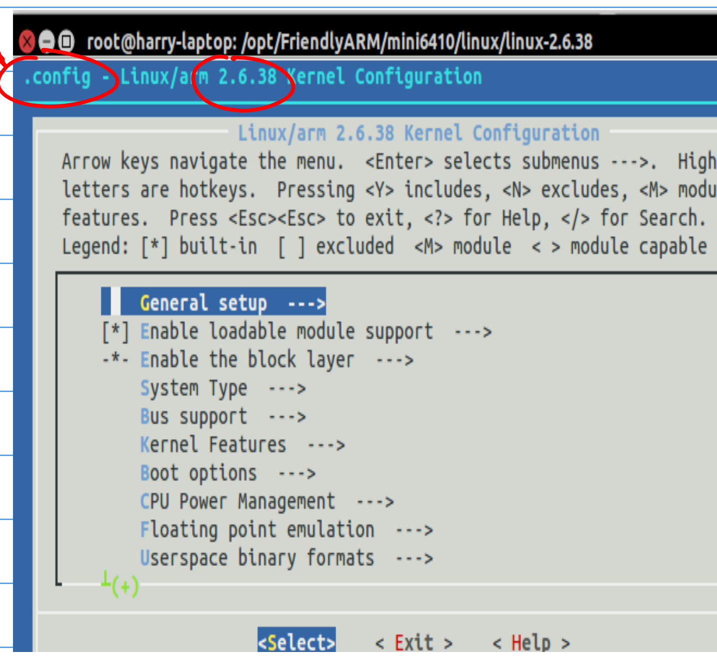
Now, for Samsung Target (ARM11)
a. Version of Config. for 32 bit ARM11.



CPU (ARM11) Datasheet

Class github





Development of Device Driver

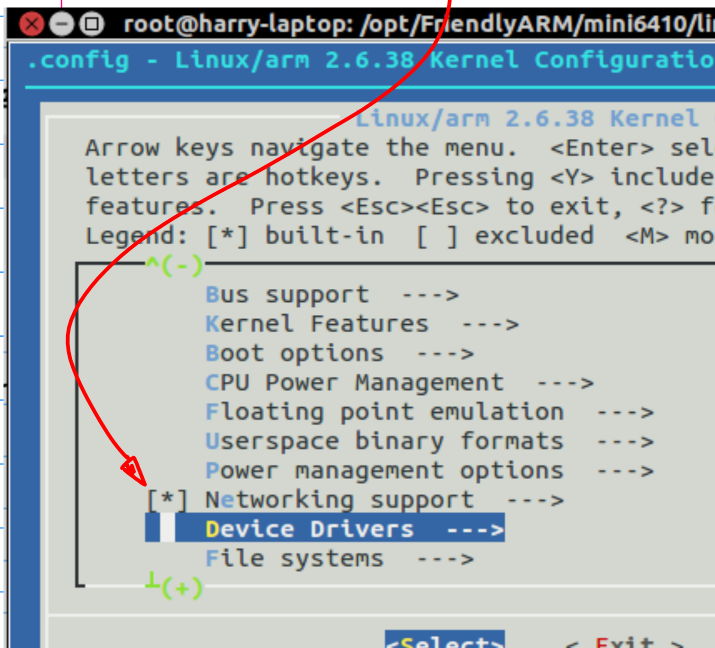
Step1. Run menuconfig for kernel Configuration.

Go to your installation directory, home\nvidia\...\source\kernel\...

Step2. Locate Device Driver for GPIO I/F.

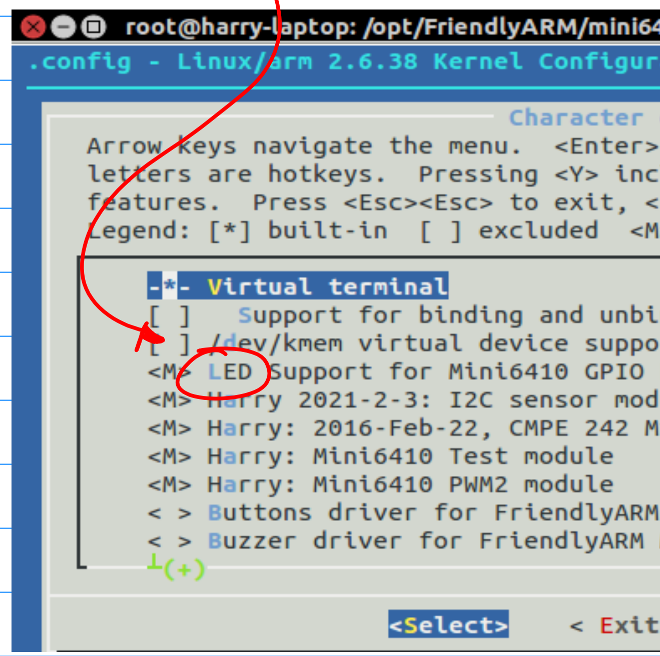
Note: I2C, SPI Are 2 Device Drivers plus PWM are Needed for 242.

Take Samsung As Example first.
Select Device Driver at the Root UI Option

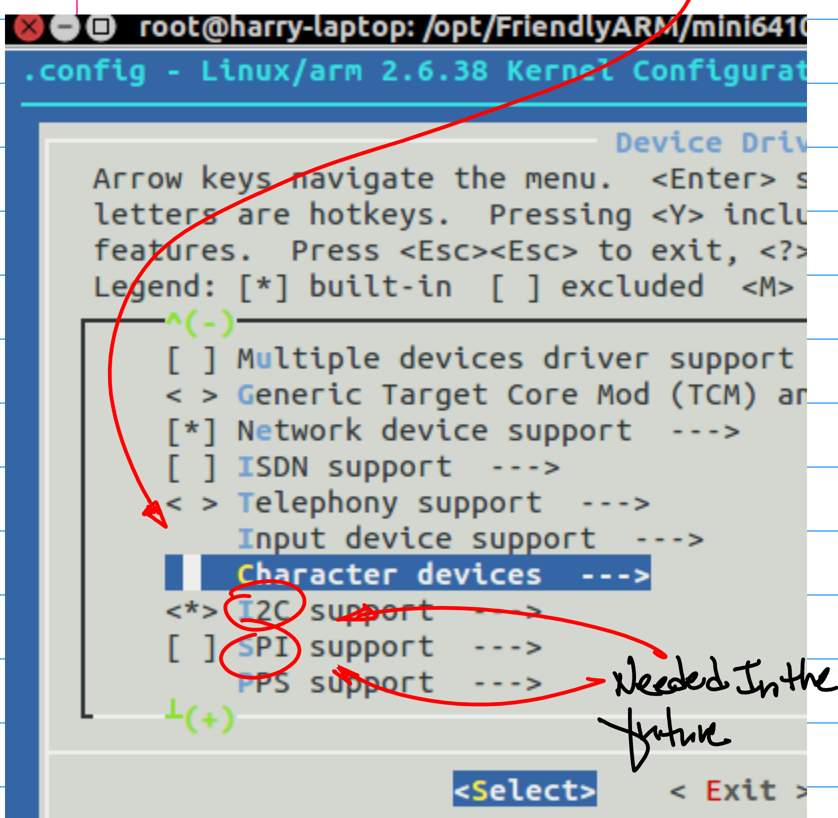


Brows till Reach the Device Driver You Need.

GPIO Testing with Device Driver "LED"



Step3. Select Char Device (Character)



Example: Now, Switch to menuconfig Version 4.4.197 for NANO, Look & Jetson. Feel of the menuconfig is the Same.

Now, Let's Discuss NANO 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 NANO. Make Sure use your GPIO Testing Circuit Designed in the Class;
2. Input Testing has to cover input "1"

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) 11:59 p.m.

Feb16 (Wed)

Note: 1° Homework Rescheduled to Feb28th (monday);

2° Bring your Prototype Board to the Class, Monday, Feb21st. 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 O.S. Distribution, and Enable Development Environment.

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

Samsung ARM11

Jetson NANO

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 Program in the User Space

```
#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 github

a. Main module; b. Input Exception/Error Handling. c. Device Driver Module to Be accessed from the User Space.

a.

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

        printf("hello\n"); //Feb. 16, 2015

b.      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 & Driver's Name

d. ioctl(arg1, arg2, arg3)

arg1 — fd — Device Driver; arg2 for Sending "0" OR "1" to Turn on/off LED; arg3 for the LED to Be Selected.

```
        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;
}
```

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

CMPE242 Feb16,22

a.
Step3. Device Driver, find Source code in the proper folder "Character Device"

```
mini6410_hello_module.ko  mini6410_leds.mod.o
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/linux-2.6.38/drivers/char#
```

Kernel

drivers

Character Device Driver

b. Find Source code of the target CPU.

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

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

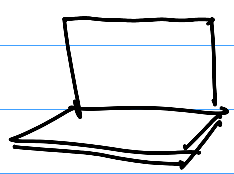
Compiled into Binary

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

Copy this module to your target platform

Options: LAN, WiFi, USB, RS232, USB Dongle

Host Development platform



Target

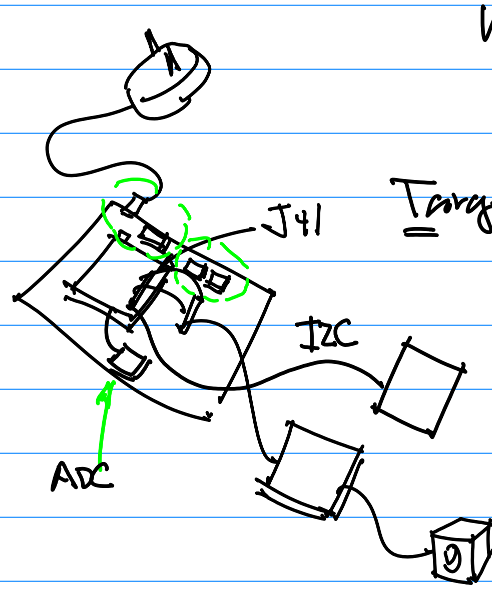


Fig.1

Once it is copied, then you can install the .ko Driver By
$ insmod my_driver.ko ↲

CMPE242 Feb16,22

Consider Building A prototype for PID Controller Project.
Start from the Prototype System Design for PID Controller Project

System Design:

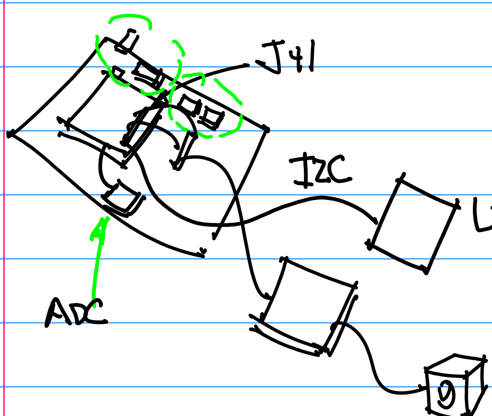


Fig. 2.

Stepper motor:

1. NEMA14~17 motor. 4 wires A+, A−, B+, B−
2. Stepper motor Operates in the following fashion

200 Steps for One Revolution Turning of 360°.

Each Step is a Full Step. it gives 1.8° Rotation.

Table 1 Stepper motor Rotation Angles

Full Step	$360/200 = 1.8°$
Half Step	0.90°
1/4 step	0.45°
1/8 step	0.225

Homework (Due A week from Today) Feb 23rd. Preparation for the target, e.g. NAND to drive a stepper motor.

Pre-requisite:

1. NEMA17 OR NEMA14 motor;
2. motor Drive
3. Signal generator to provide Square waves. (0 ~ 1 KHz)

(Note: if you do not have signal generator, then you can use timed Gpio Output Signal
100 mS → 10 Hz
10 mS → 100 Hz)

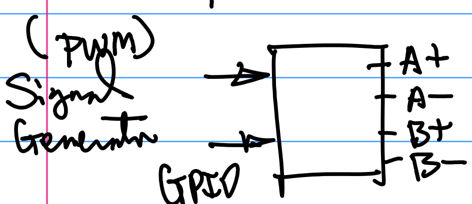
Requirements:

1. Configure your motor Drive for full step Output;
2. Connect Input (GPIO OR 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 (Inputs) Important Among Other Pins



(Direction of the motor)

Feb 21st (Monday)

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

Example: GPIO Programming.

Ref:

2022S-104-gpio-systemLevel-and-c-#202

Connector Information

J41 Connector for the Newer Version

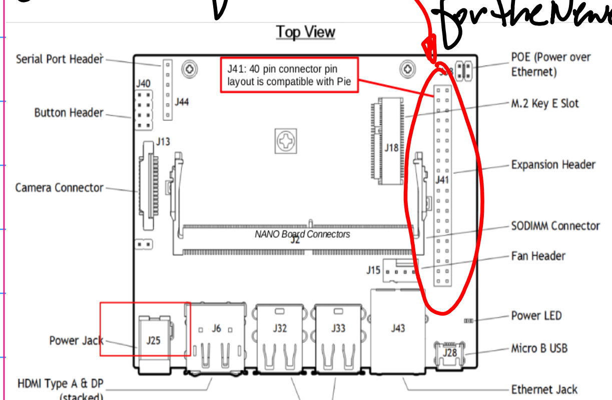


Fig1.

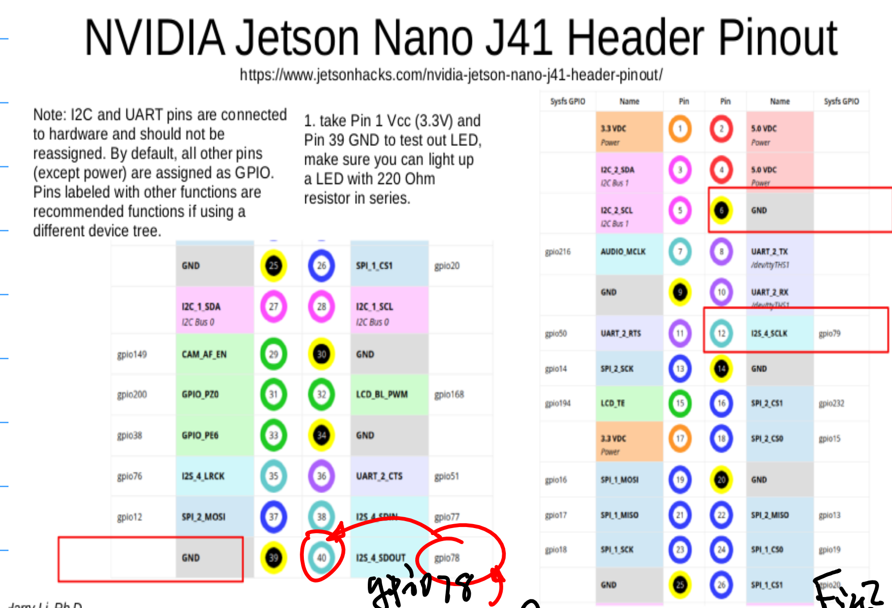


Fig2

Once the physical Layout of the NANO Board is Ready (J41 Connector) Identify the pin Assignment for the NANO Board.

Note: 1° All the pins Are multiplexed e.g., Each pin can have more than One functions, such as GPIO (GPP), PWM, SPI etc. 2° Factory Setting is given in this color coded Table, GPIO

CmpE242 Feb21,22

20

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

gpio79

Fig.3

b. Connection of Sysfs gpio to physical pin.

User program (user space)

Python, C/C++

↓

Interface 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.

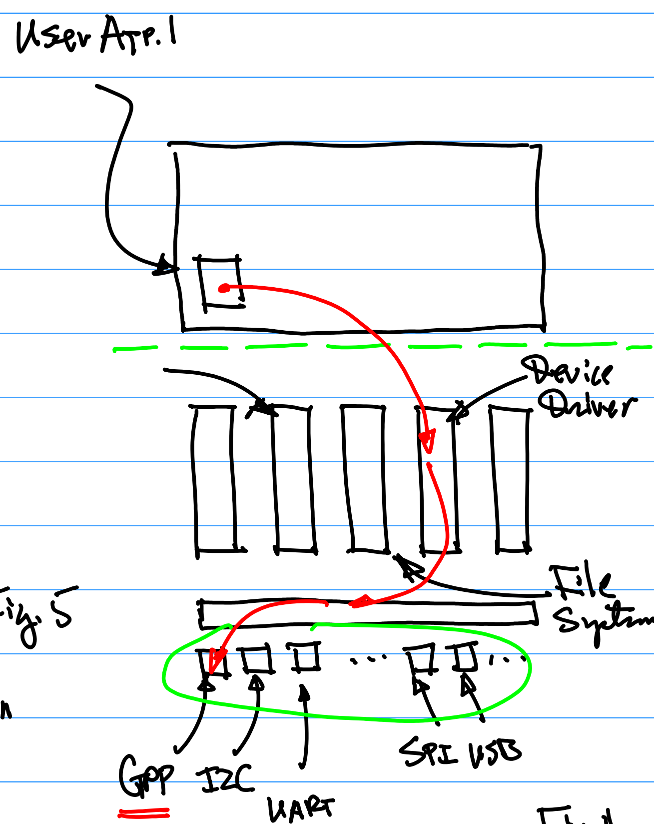


Fig.4

The configuration program

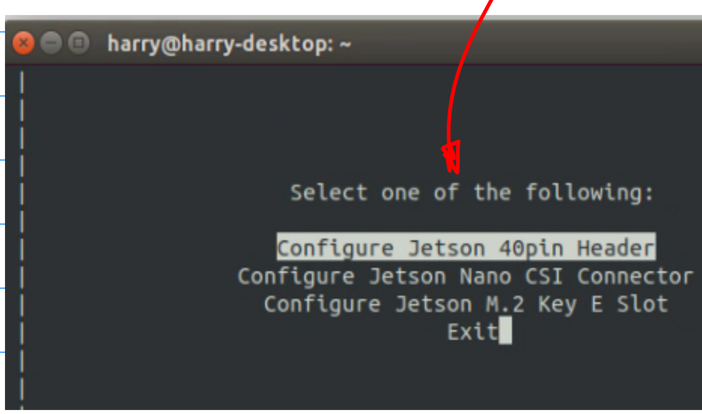


Fig. 5

Ref: To config I/O Pins, use Python Configuration Tool.

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

Source code

2022S-106b-jetson-io.py

CMPE242 Feb 21, 22

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

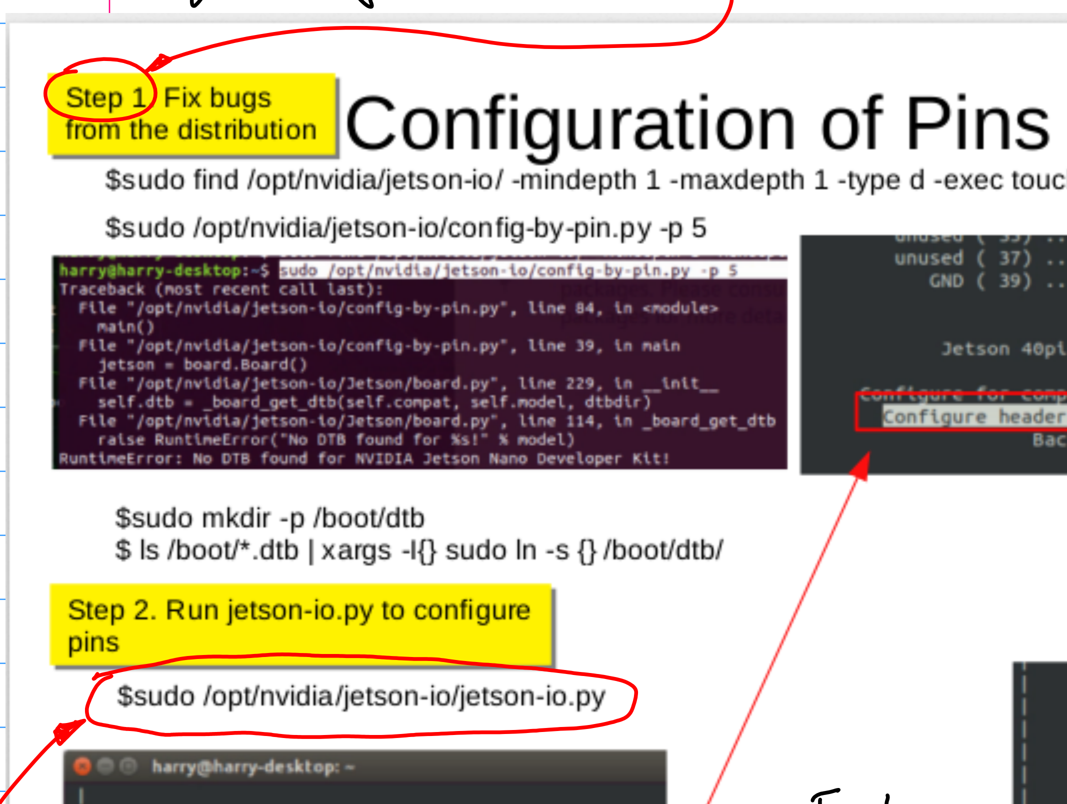


Fig.6

Then, Execute the Python Configuration Program.

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

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.

```
#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
```

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

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

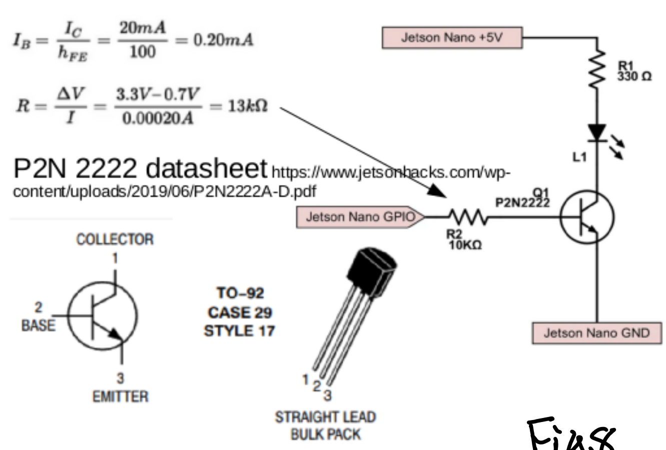
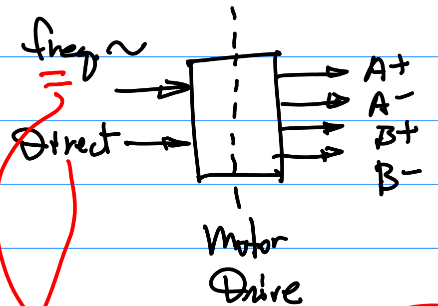


Fig.8

2. Behavior of Motor Drive

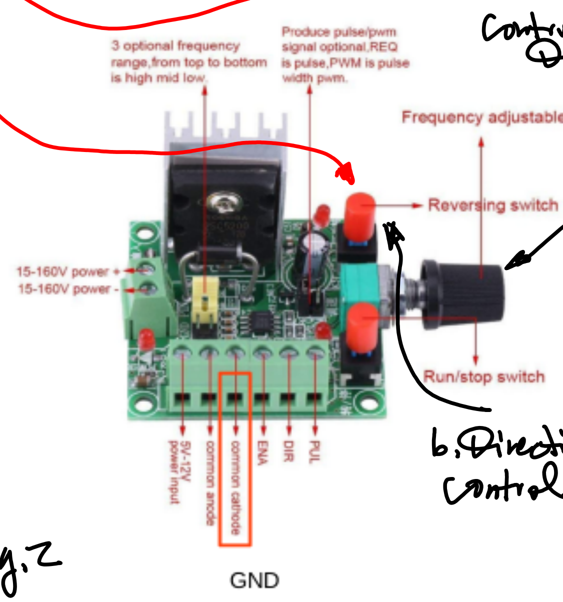
{ Inputs { freq. Input Signal — Speed
{ Directional Control Signal
{ Outputs (4 pins, A±, B±)



Clockwise or Counter Clockwise

Fig.1.

a. Human Control Dial



b. Direction Control

Fig.2

Feb 23rd (Wed)

Today's Topics: GPIO

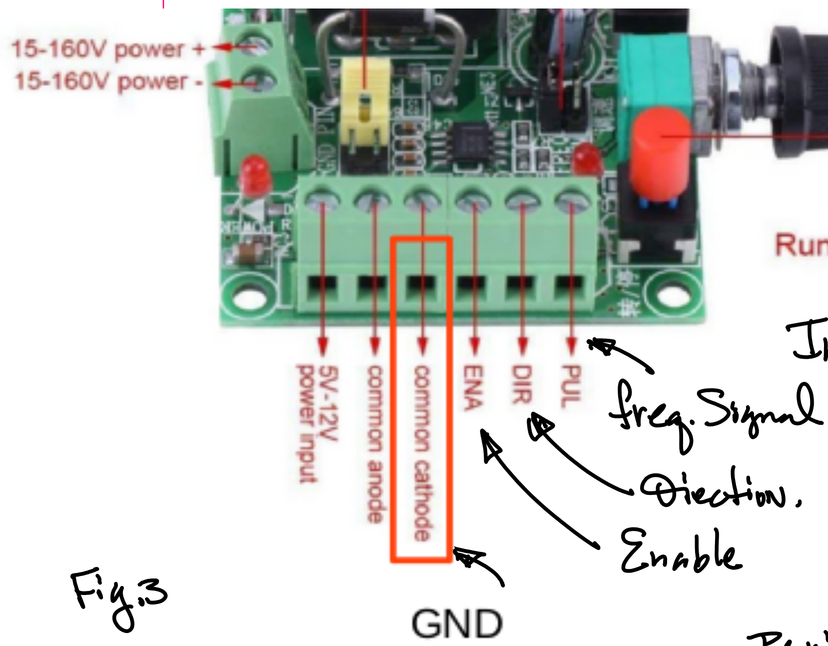
Example: Homework for Stepper motor Drive Testing

2022S-105c-motor-drive-manual-updated-hl-2022-2-23.pdf

1. No MCU / No CPU Needed

CmpE242 Feb 2nd, '22

23



Input

freq. Signal

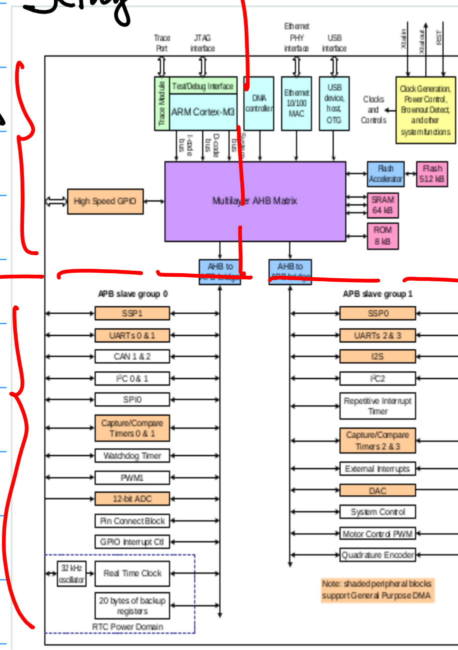
Direction.

Enable

Fig.3

CPU Core
Jtag

mem. controllers



Peripheral Controllers

Fig4.

P.8

Architecture Aspects of GPIO

1. CPU Block Diagram.

NXP LPC1769 ARM Cortex M-3
~400mHz

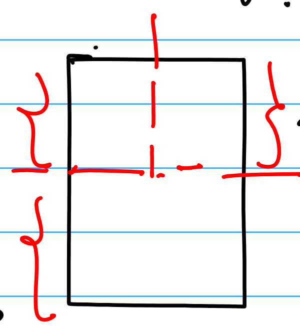
Samsung ARM11 ~800mHz
with Graphics Engine
& Video Codec.
~1GHz

Broadcom Pie3/3B+/4

Jetson NANO (4 CPUs + 128 GPU)

mem. controllers

CPU Core
Jtag



Peripheral Controllers

2. memory map.

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

$a_{31} a_{30} \cdots a_1 a_0$

addr[31:0]

MSB LSB

"Little Endian"

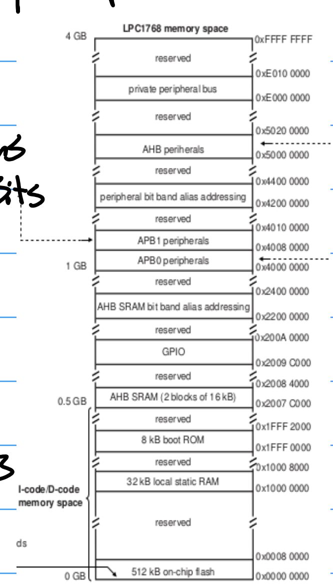


Fig5

Base Line Ref

CMPE244 / 2021F-105-#0-cpu-arm11-2018S-29-CPU_S3C6410X.pdf

Use NXP LPC1769 as foundation, to Bridge the Concepts to ARM11 and Beyond.

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

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

$\underbrace{\underbrace{\underbrace{2^{10}}_{1K} \cdot 2^{10}}_{1M} \cdot 2^{10}}_{1G}$

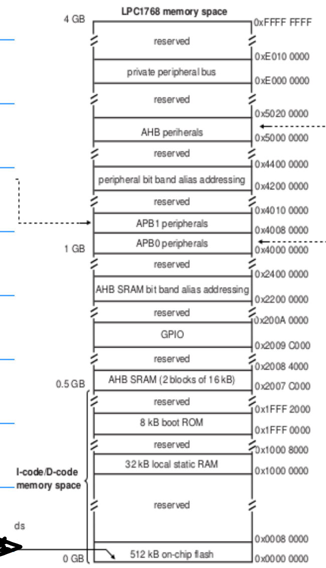


Fig. 6

Power up Address 0x0000-0000

Memory Banks:

A Block of memory.

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

$= 2^{29} = 2^{10} \cdot 2^{10} \cdot 2^{9}$

= 512 MB for Each Bank.

Address for Each memory Bank

3 bits from the Addr. Bus to define 8 Banks

3 most Significant, Consecutive Bits

The Addr. for A GPIO Controller

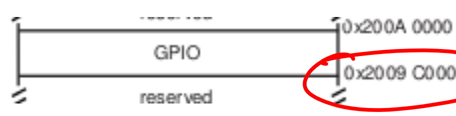


Fig. 7

0x2009-C000

a_{31}	a_{30}	a_{29}	a_{28}		
0	0	0	0	BANK0	1st
0	0	1	0	BANK1	2nd
⋮					
1	1	1	0	BANK7	8th

Use CPU Block Diagram to populate memory Space.

Addr. for the 1st BANK, BANK0 = 0x0000_0000

" " 2nd ", BANK1 = 0x2000_0000

" " 3rd ", BANK2 = 0x4000_0000

3. Peripheral controllers

Feb 28 (Monday)

1. Check 4-Person Team Formation

2. Target Board

Billy, Barbar, KaIn, Samir

Michael, Jessey, Andrew

Sourab, Xinyu

GPIO Block for ARM11, LPC1769

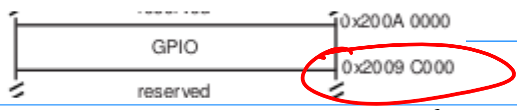


Fig. 1

Size of the GPIO Block:

$$\begin{array}{r} 0x200A\_0000 \\ -\ 0x2009\_C000 \end{array}$$

which gives the size of this GPIO Block. → Memory for GPIO Controller, which Consists of Special Purpose Registers. SPRs

32 bits Architecture, SPRs are 32bit.

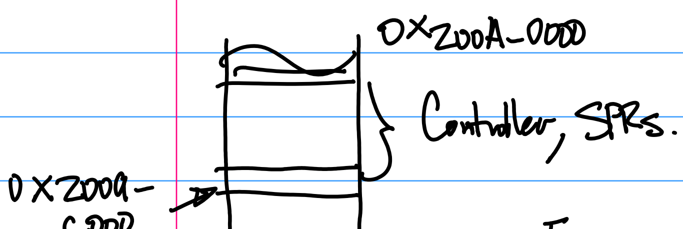


Fig. 2

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

Hint: a. find size. b. size/4 = No. 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.

Naming Convention:

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 CON → GPPCON

GPP: Prefix; CON: Root

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

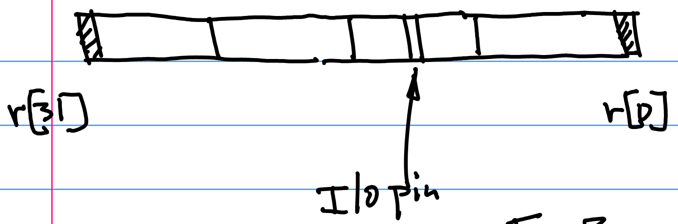
Note: For One more than one GPP port, we will modify the prefix to make it fit for this type architecture

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

Question: For each port what is the max possible pins? 32bit Architecture, hence, max possible pins is 32

CmpE242 Feb28,22

26



(I/O pins from a Data Register)

Fig. 3.

CPU Datasheet, Ref. from github ~244

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

SAMSUNG ELECTRONICS

USER'S MANUAL

Manufacture Name ID

S3C6410X

RISC Microprocessor

a. Chapter 10 b. CPU (ARM11), c. CPU Naming d. No. of GPP. 17 GPP ports.

6410X_UM

10 GPIO

f. Naming of Each port, A, B, ...

g. No. of pins for Each port

10.1 OVERVIEW

S3C6410 includes 187multi-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

CmpE242 Feb28, 22

27

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

6410X_UM GPIO

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

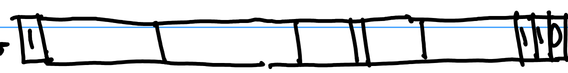
2^{32}!

10.4 REGISTER DESCRIPTION

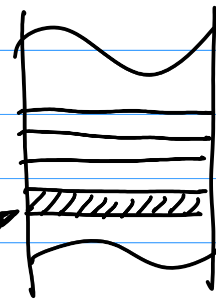
10.4.1 MEMORY MAP

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	0x00005555
GPACONSLP	0x7F00800C	R/W	Port A Sleep mode Configuration Register	0x0
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-up/down Register	0x0

0X7F00_8000



r[31] r[0]



4 Bytes

GPACON[31:0] Notation

GPACON[31]=1

GPACON[2:0]=110 (b)

0X7F00_8000

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

From CPU Datasheet, PP:313

8 pins

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

6410X_UM

10 GPIO

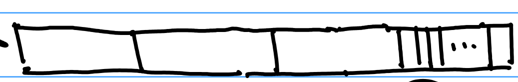
10.1 OVERVIEW

S3C6410 includes 187multi-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

GPADAT

0X7F00_8004



GPADAT[7:0]

8 pins are Built & Realized

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

The following Design Requirements are given:

1° Use GPADAT[1] as output pin;
use GPADAT[3] as input pin;

2° Config the GPA Controller By properly define GPACON.

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 Pie;

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_{PWM}, 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) Bord;

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 Stand Alone 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 Both Should be ready.

CmpE242 March 2nd, 22

Homework → Due A week from Today.

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

March 9th (Wed) 11:59pm.

Example: Continued. ARM11 CPU Datasheet.

10.5.1 PORT A CONTROL REGISTER

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

Register	Address	R/W	Descri
GPACON	0x7F008000	R/W	Port A Configuration Re
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 Conf
GPAPUDSLP	0x7F008010	R/W	Port A Sleep mode Pull-

1\.

0X7F00_8000 →

GPACON

GPACON[0]

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

2\. GPA0 → Pin → GPADAT

pin GPA0

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

3\. GPACON[3:0] Controls the Configuration of GPA0 pin.

ARM CPU pin is multiplexed

General Purpose I/O function ← GPA0 multiplexed

Set GPACON[3:0] = 0x1, output pin

GPACON[3:0] = 0x2, UART

GPACON[3:0] = 0x0, 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.

GPACON[3:0] = 0x0 for Input
GPACON[15:12] = 0x1 for Output

GPACON[31:0] = GPACON[31:28] .... GPACON[15:12] ... GPACON[3:0]

$\underbrace{1\ 0\ 0\ \ \ 0}_{0x8}$

To make this configuration,

& GPACON = 0x8; // Configuration of Pin0 (1st) & Pin3 (4th)

March 7th (Monday)

Topics: 1° CON Register Init & Configuration, & Codes; 2° PID control for Embedded platform.

Announcement: 1° Project, Phase I PID controller. Due March 21
2° Midterm Exam: March 23rd. OnLine Zoom. Exam.

Example: Code Sample (Reference: ARM11)

User Program (Same Code)

```
root@harry-laptop:/opt/FriendlyARM/mini6410/linux/exam
2022s-104d-userSpace-gpio.c  led  led.c  led.c~  Makef
root@harry-laptop:/opt/FriendlyARM/mini6410/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)

c. gpio Device Driver

Note: 1. Name of the Device (Driver)

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

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

GPK DAT

General Purpose Port K

Data Register

Note3: 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;
}
```

b. "Tmp" Buffer is defined

c. readl(S3C64xx_GPECON) memory location

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

Find its Address: CPU Datasheet

d.

e.

0x1111U

0001 0001 0001 0001

0xffff

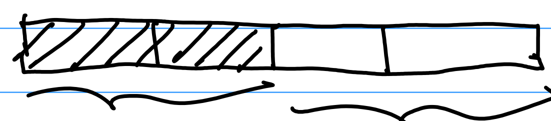
1111 1111 1111 1111

↓ Negation "~"

"&" Bitwise AND

↓

Clear all Bits



Zeroed Cleared

0001 0001 0001 0001

https://github.com/hualili/CMPE244

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

6410X_UM

10.5.5 PORT E CONTROL REGISTERS

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

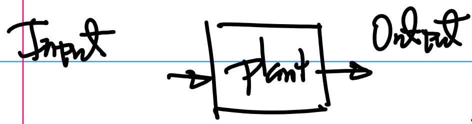
Register	Address	R/W	
GPECON	0x7F008080	R/W	Port E Co
GPEDAT	0x7F008084	R/W	Port E Da
GPEPUD	0x7F008088	R/W	Port E Pu
GPECONSLP	0x7F00808C	R/W	Port E Sle
GPEPUDSLP	0x7F008090	R/W	Port E Sle

Which memory Bank for this SFR?

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

CmpE242 March 7 (Mon), 22

Consider PID Controller Design
For Embedded platform.



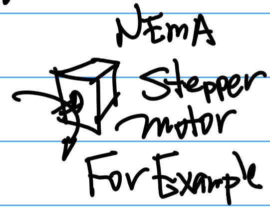
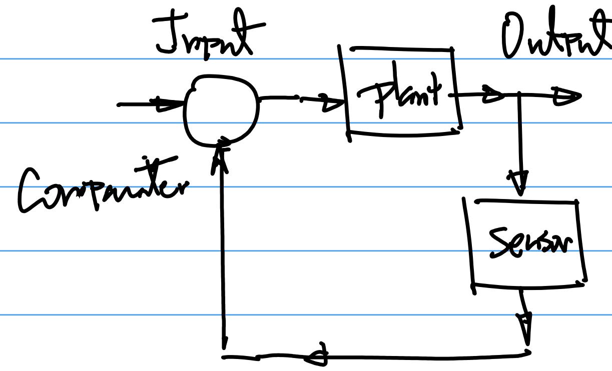


Fig. 1.a



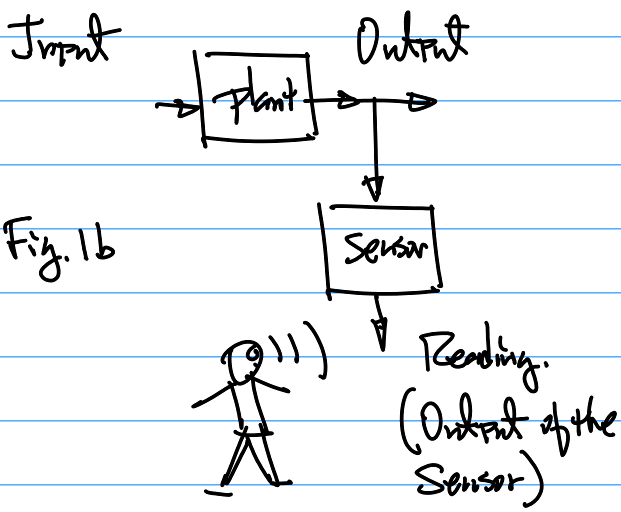
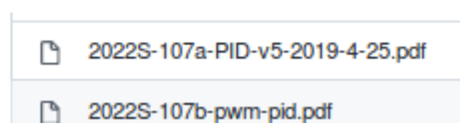


Fig. 1b

1° A "~~set point~~" of An Output
2° Comparator (Input & Output)
3° Sensor Reads one physical (Angle) quantity, produce Output of Another quantity (Electric)

Transform + Produce → "Transducer"

Comparision: Subtraction (Difference)


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

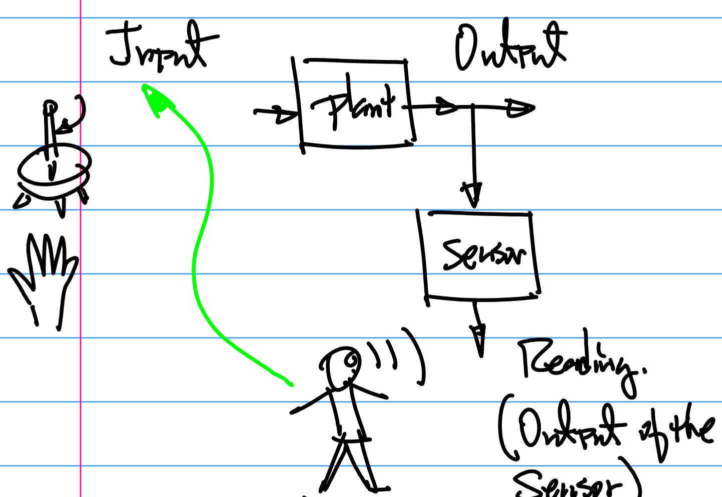
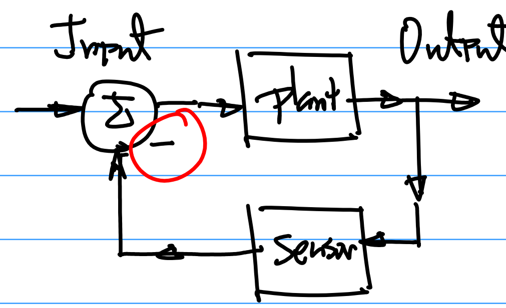


Fig. 1c



"Σ" Summation, $\sum_{i=1}^{N} x_i = x_1 + x_2 + \cdots + x_N$

Feed Back Loop

Negative Feedback: "$\sum^{N=2}$" = Input + (− Readout from a Sensor)

March 9 (Wed)

Midterm Exam: March 23d. OnLine Zoom Exam.

Note:

1° Next Monday, Midterm Exam Drill.

Bring 2 Blank Printer Papers
Write your First, Last Name
@ Top right corner.
SID (4 Digits)
CmpE242

2° Video ON During the entire session of the Exam.

3° Bring Your Prototype and have it Ready to Execute programs from your homework & Projects

4° Software tool Ready for Screen Captures.

5° No text messaging, No E-mail, Close Book, Close Notes, One page formula sheet is allowed.

6° Smart phone to take photos of your Design Implementations, Save them, Use on-Line conversion tool to convert them to pdf format.

7. Take photos of your Answer Sheets, Convert them to pdf.

Note: Keep Resolution Not too high. e.g., 1~4 MB size.
Convert them to pdf.

8. Use on-Line tool to merge all pdf files into One pdf file in Sequential order.

9. Save it with the Naming Convention:

First_Last Name_SID(4Digits)_Cmpe242_mid.pdf

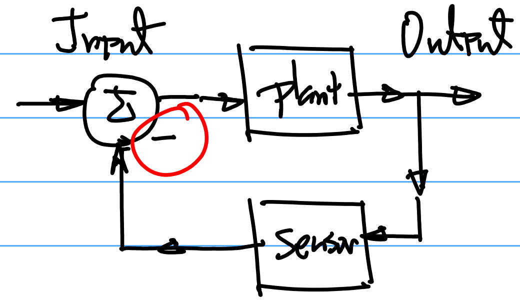
↓

Zip.

↓

Submission to SJSU Canvas.

Example: feedback loop, Negative ~



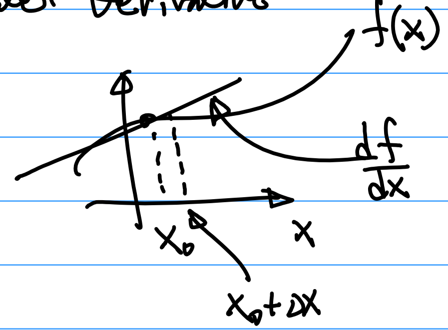
Motivations:

1° Predict the near future performance And to control the System; — Derivatives "D"

2° Evaluate the history of the System performance, And to control the System; — Integral "I"

3° Depend on the Current Perform, to Control the System. — Proportional "P"

Example: Derivative



$\frac{df}{dx} > 0$, $f \uparrow$

$\frac{df}{dx} < 0$, $f \downarrow$

$\frac{df}{dx} = 0$, f stays the Same.

Numerical Solution:

$$\frac{df(x)}{dx} \triangleq \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$\simeq \frac{f(x+1) - f(x)}{1} = f(x+1) - f(x) \quad \cdots (1)$$

$\rightarrow x$

x=0 x=1

Kernel "Mask"

$f(x+1) - f(x)$

$= 1 \cdot f(x+1) + (-1) \cdot f(x)$

from the coefficients 1, and −1 place them in the mask.

$\rightarrow x$

-1	1

find Derivative of the following Sensor Data (LSM303 for Example).

Note: use Eqn (1) or The kernel to find Derivative

t=0	t=1	t=2	t=3
1.2	1.3	1.1	0.9

-1	1

$-1 \times 1.2 + 1 \times 1.3 = -1.2 + 1.13 = 0.1$

t=0	t=1	t=2	t=3
1.2	1.3	1.1	0.9

-1	1

$-1 \times 1.3 + 1 \times 1.1 = -0.2$

then, @ t=2

$-1 \times 1.1 + 1 \times 0.9 = -0.2$

For $t=3$, No Data for $t=4$,
hence, we have done.

Forward difference. $f(x+1) - f(x)$

$\boxed{-1 \mid 1}$

Backward difference. $f(x) - f(x-1)$ ...(2a)

$\boxed{-1 \mid 1}$ ...(2b)

$x=0$

Note: Both Differences are "Skewed" Not Balanced.

2022S-104d-userSpace-gpio.c

2022S-104e-kernel-spece-gpioDriver-#mini6410_leds.c

$$\frac{1}{2}\left[\left(\text{Forward Diff}\right) + \left(\text{Backward Diff}\right)\right] \quad \ldots(3a)$$

$$\frac{1}{2}\left[\left(f(x+1) - f(x)\right) + \left(f(x) - f(x-1)\right)\right]$$

$$= \frac{1}{2}\left(f(x+1) - f(x) + f(x) - f(x-1)\right)$$

$$= \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & & 1 \\ \hline \end{array} \quad \ldots(3b)$$

Example: Use "3b" to Compute Derivative

$t=0$	$t=1$	$t=2$	$t=3$
1.2	1.3	1.1	0.9

for $t=0$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & & 1 \\ \hline \end{array}$$

No Data at $t=-1$

Skip.

for $t=1$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & & 1 \\ \hline \end{array}$$

$$\frac{1}{2}(-1 * 1.2 + 1 * 1.1) = \frac{1}{2}(-0.1)$$

$$= -0.05$$

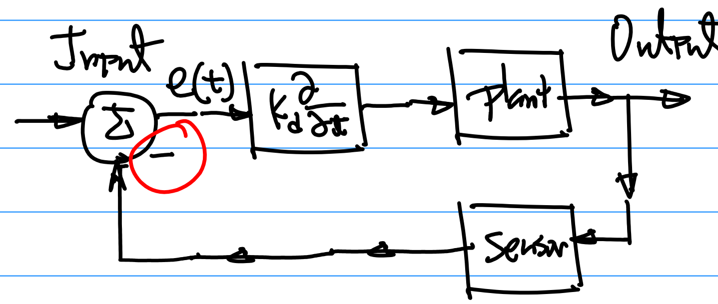
for $t=2$

$$\frac{1}{2}\begin{array}{|c|c|c|}\hline -1 & & 1 \\ \hline \end{array}$$

$$\frac{1}{2}(-1 * 1.3 + 1 * 0.9) = \frac{1}{2}(-0.4) = -0.2$$

Homework: Write code to Compute derivates of the given data By using Central Difference due next week. (March 16)

Design Derivative Controller



Embedded System Implementation.

PWM.

Ref:

- 2022S-107c-pwm-config-v4-hl-2022-3-3.pdf
- 2022S-107d-pwm-nano-coding-hl-2021-12-19.pdf

In order to write code to Access to PWM, first, do config. (Device Drivers).

Note: Read this Code (GPIO Driver)

- 2022S-104d-userSpace-gpio.c Last lecture.
- 2022S-104e-kernel-spece-gpioDriver-#mini6410_leds.c

ı. By default, all other pins
wer) are assigned as GPIO.
ed with other functions are
ded functions if using a
evice tree.

Use pin 32 for PWM

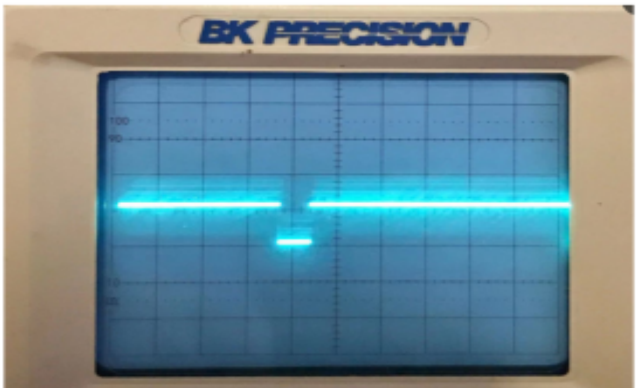
	GND	25	26	SPI_1_CS1	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	I2C_1_SCL *I2C Bus 0*	
gpio149	CAM_AF_EN	29	30	GND	
gpio200	GPIO_PZ0	31	32	LCD_BL_PWM	gpio168
gpio38	GPIO_PE6	33	34	GND	
gpio76	I2S_4_LRCK	35	36	UART_2_CTS	gpio51
gpio12	SPI_2_MOSI	37	38	I2S_4_SDIN	gpio77
	GND	39	40	I2S_4_SDOUT	gpio78

pin 12 for gpio78

```
cd /sys/class/pwm/pwmchip0
echo 0 > export
sleep 1
cd pwm0
echo 5000000 > period
echo 2500000 > duty_cycle
echo 1 > enable
```

Define as in Hz

Output high defined as in Hz




```
harry@harry-desktop: /sys/class/pwm/pwmchip0/pwm0
@harry-desktop:~$ cd /sys/class/pwm/pwmchip0
@harry-desktop:/sys/class/pwm/pwmchip0$ echo 0 > export
@harry-desktop:/sys/class/pwm/pwmchip0$ sleep 1
@harry-desktop:/sys/class/pwm/pwmchip0$ cd pwm0
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 5000000 > period
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 2500000 > duty_cycle
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 1 > enable
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 1 > enable
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 1000000 > duty_cycle
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 500000 > duty_cycle
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 250000 > duty_cycle
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 450000 > duty_cycle
@harry-desktop:/sys/class/pwm/pwmchip0/pwm0$ echo 4500000 > duty_cycle
```

march 14 (monday)
Mid-term ON 23rd (Wednesday)

Integral of Error

$$\int_I e(\tau)\,d\tau \quad \ldots (1)$$

I ← Time Interval.

Example: Integral of Error

	t=0	t=1	t=2	t=3
error	1.	1.3	1.1	0.9

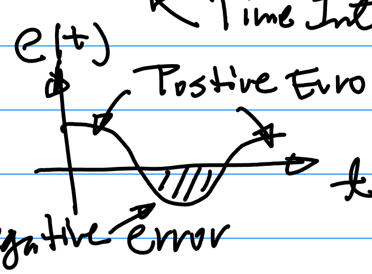


Fig. 1

To Avoid error Cancellation (positive error Added to Negative error), we have

$$\int_I e^2(\tau)\,d\tau \quad \ldots (2)$$

Introduce gain for integration controller.

$$K_I \int_I e^2(\tau)\,d\tau \quad \ldots (3)$$

Computer Computation Based on Summation.

$$K_I \int_I e^2(t)\,d\tau$$

$$\Rightarrow K_I \sum_{i=0}^{N-1} e^2(i) \quad \ldots (4)$$

Let's make $N=2$, hence

$$K_I \sum_{i=0}^{1} e^2(t-i)$$

Computation of Integral of Error

$t=0$	$t=1$	$t=2$	$t=3$
1.	1.3	1.1	0.9

1st 2nd 3rd ...

1st Integral of Error

put K_I aside for future use

$$\sum_{i=0}^{1} e^2(t-i)\Big|_{t=1} = e^2(1-0)+e^2(1-1)$$
$$= e^2(1)+e^2(0)$$

from the given condition:

$$e^2(1)+e^2(0) = 1.3^2+1^2 = 1.69+1 = 2.69$$

2nd Integral of Error.

$$\sum_{i=0}^{1} e^2(2-i) = e^2(2)+e^2(1)$$

$$= 1.69+(1.1)^2 = 1.69+1.21 = 2.90$$

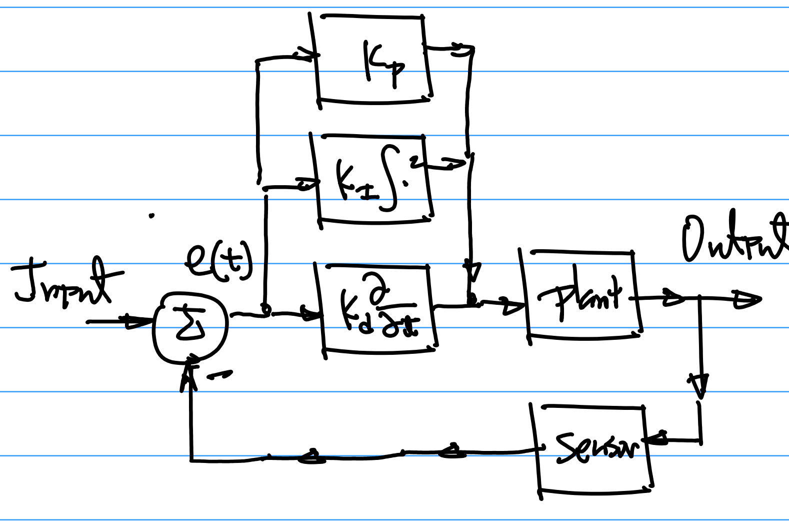
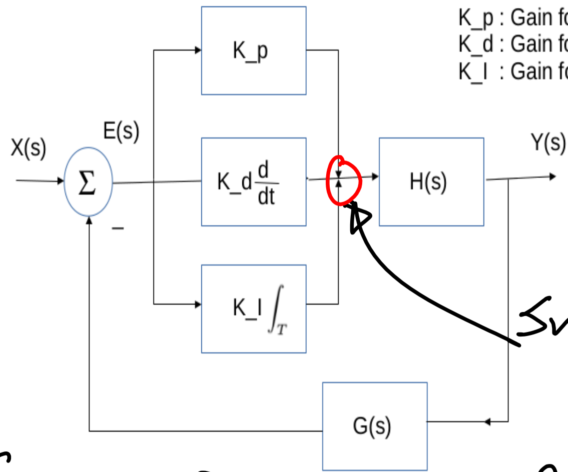


Fig. 2



K_p : Gain for proportional control
K_d : Gain for derivative control
K_I : Gain for integral control

Fig. 3

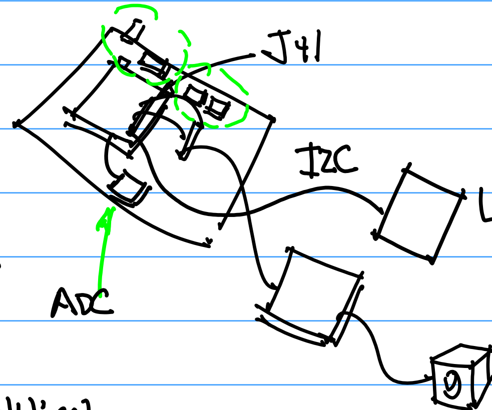
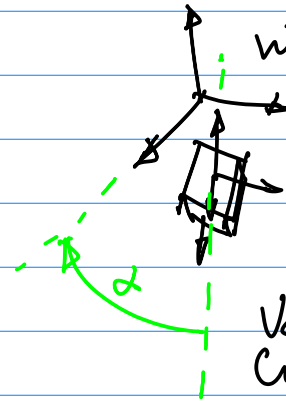


Fig. 4

Summation / Addition



Suppose $\frac{\partial}{\partial t}e(t) = 0.5$, $\int_I e^2(t)\,dt = 10$

$e(t) = 2.1$

Find the total error at the output of the PID controller (see Fig. 3)

$$e_\Sigma = K_p e(t) + K_I \int_I e^2(t)\,dt + K_d \frac{\partial}{\partial t} e(t) \quad \cdots (5)$$

$$= 2.1K_p + 10K_I + 0.5K_d$$

if $K_p = 5$, $K_I = 2$, $K_d = 1.1$

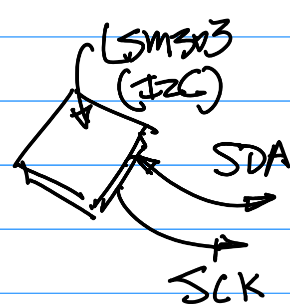
then $e_\Sigma = 2.1 \times 5 + 2 \times 10 + 1.1 \times 0.5$

Design Requirements
- Hardware
- Software
  - I2C Device I/F
  - Python or C|CPP

Hardware Design. Bi-Directional

I2C : 2 pins
- SD : Serial Data Pin (SDA)
- SCK : Clock Output, from the master

LSM303DLHC
Ultra compact high performance e-compass
3D accelerometer and 3D magnetometer module
Preliminary data



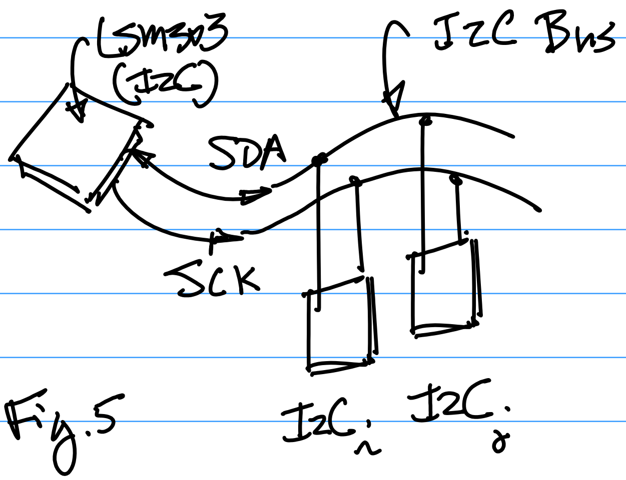


Fig. 5

Figure 4. LSM303DLHC electrical connection