Jan 27, 21

Welcom to

CMPE242 Harry Li

Embedded Hardware Systems

1. Green Sheet github/hualili/cmpe242

E-mail: hua.li@sjsu.edu

(650) 400-1116 Text message

2. Pre-requist Requirements 180A & D

Course Description

Hands-ON.

Target Development platform

200

ARM7, 9 — NXP LPC 17xx

Samsung ARM11 ~10yrs 800mHz Linux Kernel O.S. Open Source

Broadcom Pie 3B+ Pie 4 ~$60 Edge AI

High Performance platform

NVDA

① NANO ~$60

② TX2 ~$400 + $150 ~ $350

③ TX1 ~$399

Basic Material

Scope of the Course { Device Driver Development C/C++, Python; O.S. Kernel

Before: 3 Labs/Projects

Device Driver — Sensor LSM 303 — I/F to Target Board. — Human Readable file. → Feed Back Loop P.I.D Controller Convolution

Optional subjects — Integration (HDL)

1° RISC-V Project — FPGA

Device Driver Development — O.S. Kernel Image

2° ROS (Robotics Operating System) platform Visualization Tool.

* Grading Policy

3-3-4 mid Projects ← Final

* CANVAS — EE242

① Assignments (NO)

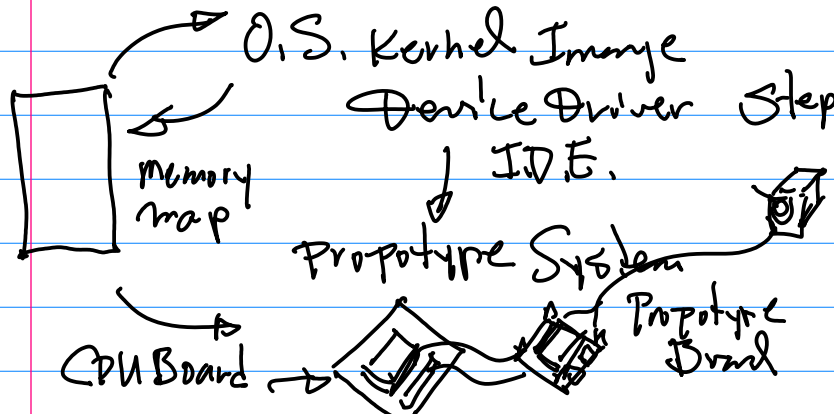
② Collection/Submission of your class work

Action 1. github/hualili/cmpe242

2. Datasheet

{ Samsung ARM11 Datasheet
NXP LPC1769 Datasheet

Architecture NXP LPC 17xx



Action3. Target Platform Selection of Unix-Like OS. 6) Edge AI Computing (Scalability) ⇒ GPU

Office Hours M.W. 4:30-5:30pm ON Zoom.

CMPE242 Feb1. 2021

Today's Topics: 1° System Architecture Review, CPU Datasheet; 2° Target platform

System Architecture

- NVDA CPU/GPU platform. { NANO TX1, TX2
- BroadCom, Pi 2/3/4 G.E. (Graphics Engine)
- Samsung ARM11 (9,7)

Optional Architecture RISC-V

Common Characteristics RISC

Reduced Instruction Set Computer

Optimization Not Only on the Hardware But On the Compiler Design, and System Software Design.

MIPS, ARM (Most Widely Adopted) RISC

ARM Architecture ——— Common Core

Base Line Hardware (Datasheet): ARM11

Base Line Software { O.S. Kernel Aspects, Device Drivers.

Linux Distribution Optimized for Embedded platform

Example: Datasheet LPC1768 ARM Cortex M3

Samsung ARM-11

NVDA ... PiEX

CPU Datasheet, LPC1769 20185-3-UM10360, PP. 9

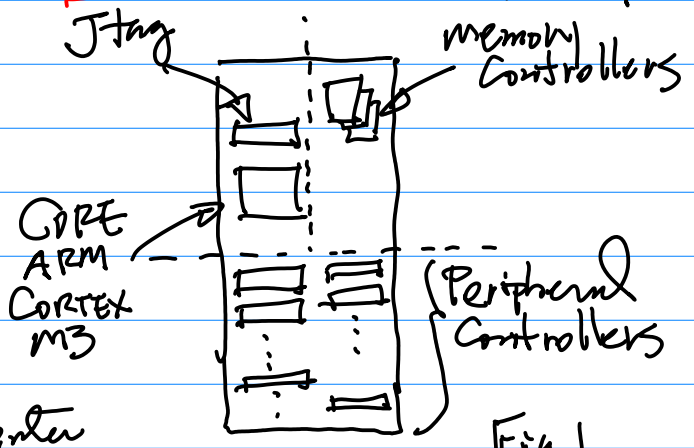


Fig 1.

Fig 1. CPU Architecture ARM Cortex M3

Note: To Be Able to Draw/Design CPU Architecture Ether this OR Your Target platform. (176a + ARM11) Base Line

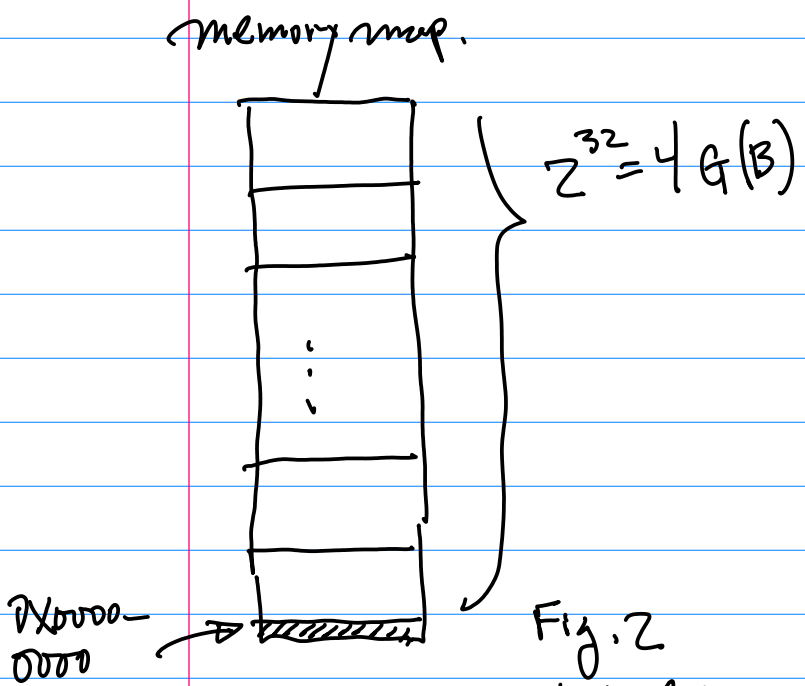


Fig. 2

① 32 Bit RISC Architecture

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

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

$= 4\ GB \quad (1G\ Byte)$

② Byte Addressable Machine

~ whose minimum memory cell with an unique Address is a Single Byte

③ Memory Banks, 8 BANKS

Size of Each Bank: 4GB/8

$= 2^{32}/2^{3} = 2^{32-3} = 2^{29} = 2^{9} \cdot 2^{20}$

$= 512$ MB

Starting Address of Each Bank

Question: How many Bits needed to define the Starting Address of Each Bank? $\underline{3}$ bits, $2^3 = 8$

3 bits Needed

$a_{31}\ a_{30}\ a_{29} : a_{28} \cdots a_1\ a_0$

Little Endian

ARM CPU can be configured, at Boot Stage as either "Little Endian" or "Big Endian".

Find Starting Address for BANK the 1st:

$a_{29} = a_{30} = a_{31} = 0$

a_{28} has to be added, to form a Hex

0X0000_0000

2nd Bank's Starting Address

$a_{31}\ a_{30}\ a_{29} : a_{28}$

0 0 1 : 0

0X2000_0000

3rd Banks Starting Address

$a_{31}\ a_{30}\ a_{29} : a_{28}$

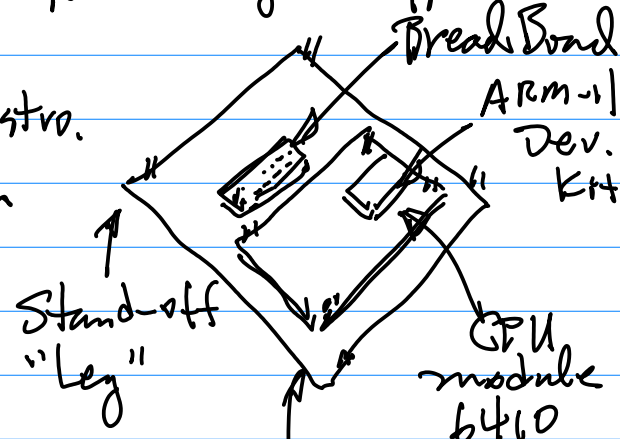
0 1 0 0

0X4000_0000

Now, Consider target Board
Conditions to qualify the selection

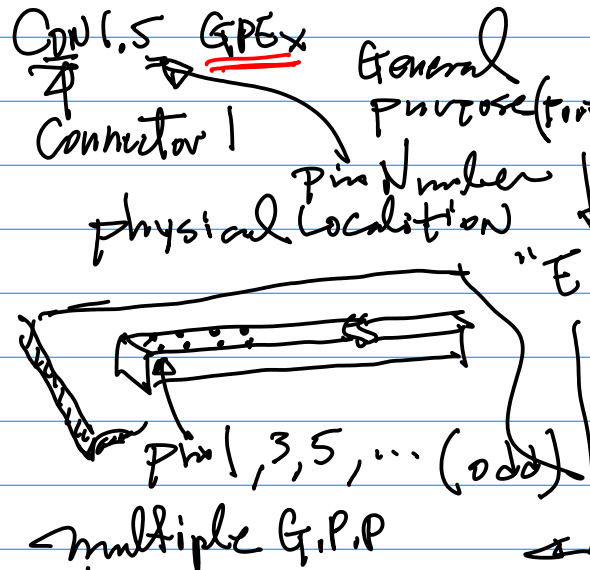
① ARM Based ; ② UNIX-Like O.S. Linux { Kernel Image, Source Distro, Tool Chain
③ Establish eco-system Developer Base (~millions)
④ Technology Innovators / Leaders.
⑤ External Expansion Connectors

Plus: many examples on Device Drivers (I2C, PWM, SPI, UART, ...)
Good learning Tool 4/5



Feb 3rd (Wed) CMPE242
Note: 1° Submission of Honest Pledge Form (Signed), CANVAS, By Sat 11:59 pm ; EE242 submission to e-mail ;
Today's Topics: 1° CPU Architecture
2° Target Board Selection — Bill of material
Ref: github
1-2020S-lect1_hardware board...
ARM-11 ① Compliant w/ Linux Open Source Distro { Kernel Sources, ARM tool chain
② Datasheet Baseline Reference Requirements, Exams
DrawBack: Lack of the Ability to handle Edge AI ; (G.E.) Tiny 6410 Kit from FriendlyARM.com, ~$90

Note: Bread Limitation — Raw High Speed ~20 MHz Cannot

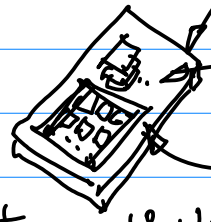


Option (Pie Board 3B+, 4) $75 Pie-4, 8GB mem.

CMPE242

NUDA (Nvidia) ↓ NANO — 128 GPU

↓ TX2 System on module



6 GPU ARM A57?

256 GPU

Note: 1° NANO Expansion Connector, Yes (Limited I/O Function) Compare to ARM11 — SPI ✓, I2C ?, PWM

2° Kernel Source Distribution, Yes

To Become a developer, to Sign up ~$59

Option (NUDA TX2 System on module)

1° Expensive! $299 + Carrier Board $150 – Bust

Edge on AI

If this is seleted, then { 2~4 person to Share the Cost

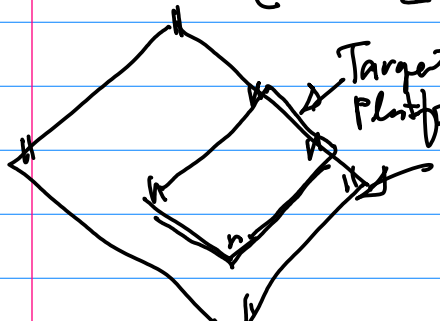
Note: 1° 4-Person Team By Next Week;

{ or Homework/Project has to Individual, Each person has your own Board;

Bill of materials:

Phase I & II { Phase I — HW1 – "Hello,..."

" II Sensors/Stepper motor Drive



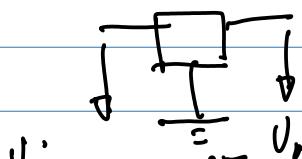
Target platform

1° Wirewrapping Board

Through-Holes w/metal plating (Not the entire side)

2° 4 Stand-offs ("Legs")

3° Components to Build PWR CKT & CKT for I/O Testing ("Hello, the world").



(To Target Board, and CKTs)

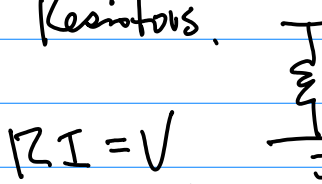
Vin — PWR Regulator IC — Vout

78xx 7805 7812

5VDC 12VDC

Red LED, 4~10mA

Resistors.



Vcc = 5.0

I = 4~10mA

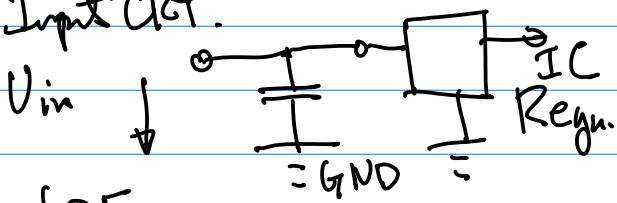
GND

$RI = V$

$R = V/I = 5/4\times10^{-3}$

$= 1.25\ K\Omega$

Cap for your External Power Input Clkt.



Vin

IC Regu.

GND

LPF (Low Pass Filter)

$C = 4.7\ \mu F$

$T = RC \rightarrow$ 3dB

4° Toggle s/w

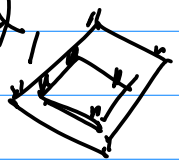
Feb 8, (Monday) CmpE242
Harry Li

Today's Topics: 1° Bill of material
2° Prepare for the 1st Assignment
"Hello, the world". 3° CPU Architecture

Bill of Material { Phase I: Target platform "Hello, the world"
Phase II: Sensors / Drive / Stepper motor / OpAmps

1. Target: ① Pic3Bt, Pic4
② NANO, TX2 (Edge AI)
③ ARM11 — Base Line Reference platform

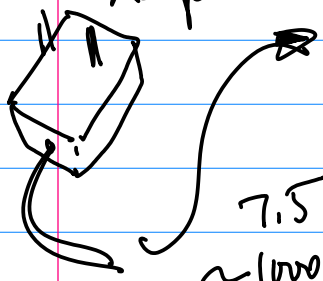
2. Prototype Board, 6" x 4"



I/O CKT
Sensors { Analog / Digital
Drive

POWER CKT: PWR Regulator IC
LM7805 20VDC drop off

3. Wall mount Adapter



7.5 VDC ~1000-1500 mA.

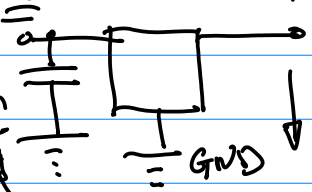
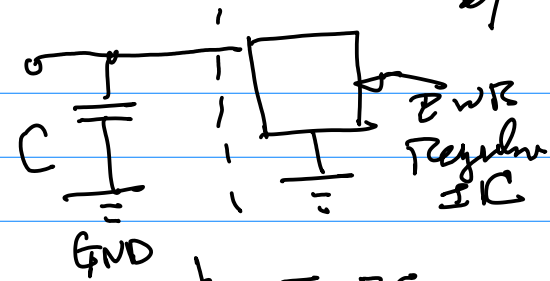


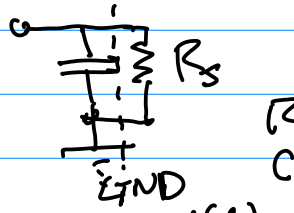
Fig1. 5 VDC

4. "Glue" Logic { Assorted Resistors 200 ~ 5 KΩ 10 KΩ
Caps { 7805
PWR Input Node
LPF (Low Pass Filter)

⑤ PWR Distribution Node

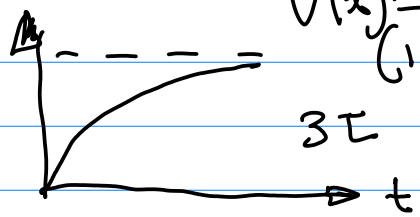


$\tau = RC$



R = 1KΩ
C = 4.7uF

$V(t) = V_{DC}(1 - e^{-\frac{t}{\tau}})$



Caps for 7805 → Datasheet
Polarity "⊥"

74xx NAND, NOR,
LED, Red, yellow, green
4~10mA

Note: you may need higher voltage source to Stepper motor Drive.
LM7812?

Connectors { External PWR
Switch (Toggle Switch)

To control PWR Source;
To Build I/O Testing

Phase II (Bill of material):

Ref: github:

1. LSM303 (I2C) → Autonomous Robot

2. optical encoder (optional) = LSM303

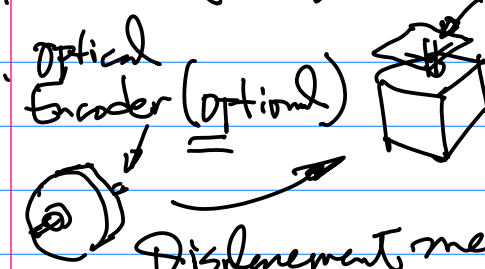
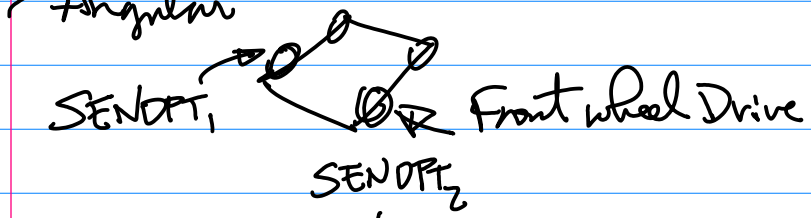
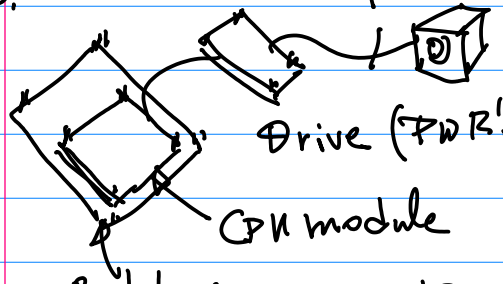


Fig2

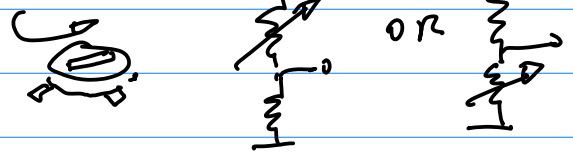


3. 4 NEMA14, 17



a. 12VDC Commonly used Source, Check Current Need!

b. Buck Converter = is good;



Consider Building "Hello, the world" Test CKT

Objectives 1. Bring up the target Board and print "your Name, Last 4 Digits Student ID"

2. Test GPIO Interface

2.1 Send Logic "1" to Turn ON On-Board LED, then flash @ 1 Hz Frequency

2.2 Send Logic "0" via gpio port to turn off on-Board LED;

2.3 Read input via GPIO port, if the input is "1", then send command to Turn ON On-Board LED

2.4. Read Inputs via GPIO Port, if the input is "0" then, Send command to Turn off the on-Board LED.

Discussion on CPU Architecture then on this Assignment.

Action 1: Form 4 person Team, Send me email First, Last Name, SID

Feb10.

Homework 1 Due Feb24 (W) 11:59pm.

CmpE/EE242 Canvas

EE242 Submission to E-mail:

Target platforms: ARM11[a] Architecture[b] O.S.[c] Driver

Base Line
- Pic ✓ (Not mbed) — Kernel ✓ NM — ✓ NM
- NVDA ✓ — ✓ — ✓
  - NANO, TX2

1° Target platform Built on Prototype Board

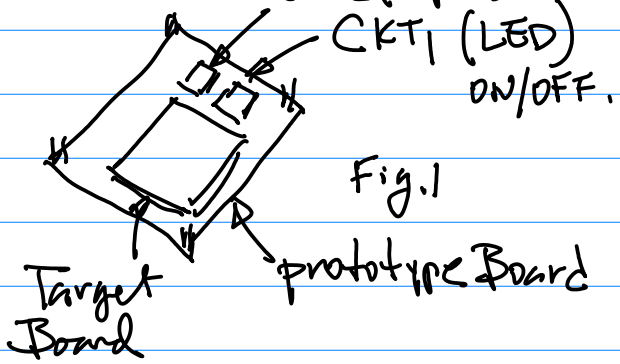


Fig.1

Ref: 1-lec1-Hardware Board-

print "Hello, the world", Drive LED ON/OFF

C/C++ or Python — Has to be the LED ON your prototype

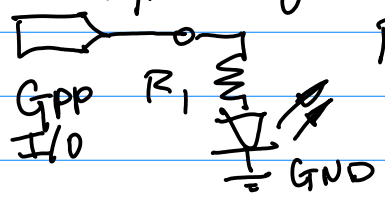
2° GPIO Testing
- GPP Output CKT1
  - "1" Turn on LED
  - "0" Turn off LED

Design:

Identify the GPIO pins for Input and Output Testing CKT.

NXP LTC 1769
- Pf.3 GPP Input pin
- Pf.21 GPP Output Pin

Pf.21 Fig2



$V_{GPP} = IR + V_{LED}$ ... (1)

CMOS $V_{GPP} = 3.3$ VDC

$I \simeq 10$ mA

$V_{LED} \simeq 1.2$ VDC

$3.3 = 10\times10^{-3} R + 1.2$

$R = (3.3 - 1.2) / 10^{-2}$

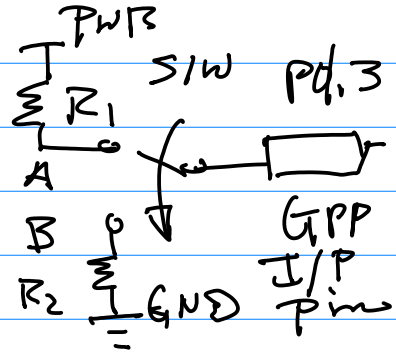
$= 210\ \Omega$

Note: Find the pin(s) from Target platform.

Pie3 GPIO14 — Pin 8

Consider Input Testing CKT

Input pin GPP { Read "1", PWR; Read "0", GND } with toggle switch



R1: $3.3 / 10\text{mA} = R_1$

$3.3 / 10\times10^{-3} = 330\ \Omega$

R2: 3.3 (from GPP) $/ 10\times10^{-3} = 330\ \Omega$

System (Architecture) & Software Design. Boot Loader { O.S. / I/O Function

3° Source code, Binary → Zip.

4° One page Report IEEE paper format (Template is given on-Line) github

5° Form 4-person Team. Submit First, Last Name, 4 Digits SID E-mail Contact Information Indicate Coordinator of the Group By Thursday.

Note: All work has to be individual

6° Short video Clip (5–10 seconds) Shows the Prototype Board and Screen Capture.

Feb 15, Monday

Topics: GPP I/O, Device Driver.

github: 2-2020S-Lec2 ...

Example: 1) CPU Broadcom BCM2835

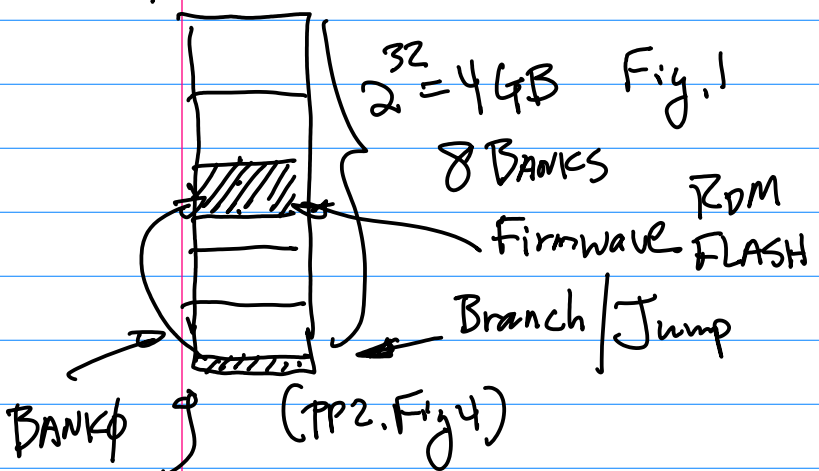
2) PWR 1, (3P3), 2, 4 (5VDC)

3) GPIO for Homework → Chose pin those not marked w/ other function

Concept of Device Driver

{ Architecture + memory map
{ Software { Kernel (OS)
{ Device Drivers

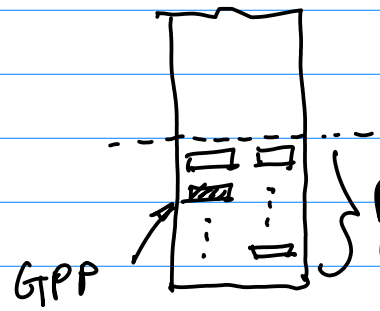
Architecture



0X0000_0000 PWR-up Address

Addr. when CPU is powered up, it will fetch the 1st Instruction from this addr.

CPU Architecture.



GPP (GPIO) for General Purpose Interface I/F

A set of SPRs (Special Purpose Registers)

Note:

1° SPRs are 32 bits.

Feb 17 (W)

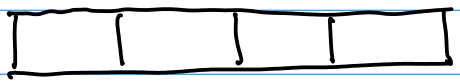
Ref: CPU Datasheets

1° github ~ 20185-29

2° SPR's Functions into 3 Categories

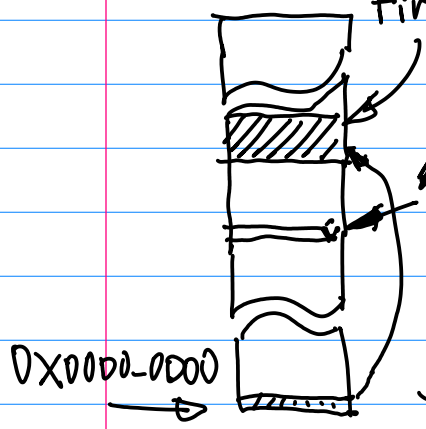
a. Control Function, Init & Config
b. Data
c. Pull up/Down

3° Map 32bit SPR onto The memory



32 bit SPR

which is mapped to the memory location



4 Address occupied by each SPR.

Starting Addr. is always at multiple of 4

Fig 1

1° PWR_up Address: ~ when CPU is powered up, it will fetch the 1st Instruction from this location

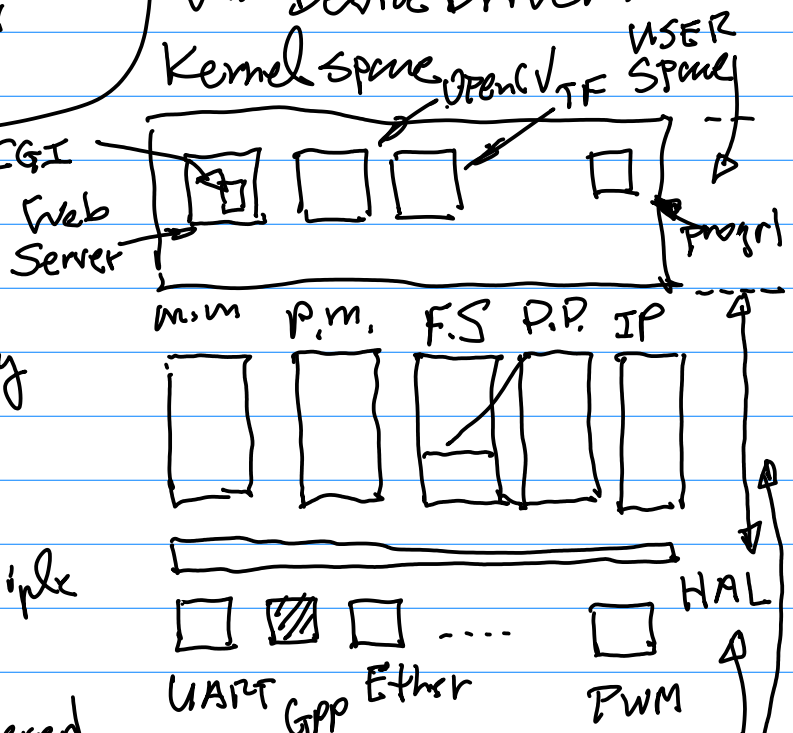
~/cmpe242/20185-29-CPU

(ARM11 CPU Data Sheet)

2° Boot up Sequence

Firmware: ROM/FLASH { Boot Loader, I/F

3° O.S. Image is Being Loaded Then USER Space program can be executed, And the Device(s) can be accessed via Device Driver in Kernel space



Hardware Abst. Layer

Kernel Space

Fig 2.

Example: Prog1.C → USER Space

UserSpace → open("/dev/..."); fd = ; path Device Driver location in the Kernel Image for GPP Input Testing

read(fd, Buffer, size)

write(fd, Buffer, size)

GPxCON $2^{32} = 4G$

Little Endian

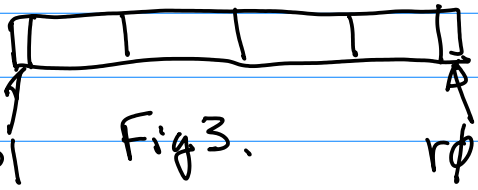


Fig 3.

Test GPP Output GPx pin 8 as an output pin

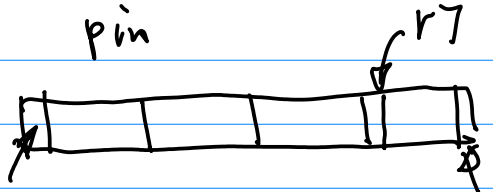


Fig 4

4. Kernel Space:

OS: manage ~ Resources & Policy

DeviceDriver(s): A collection of program(s), manage/manipulate Special Purpose Registers, to be able to utilize peripherial controller(s)

Init & Config of Special Purpose Registers

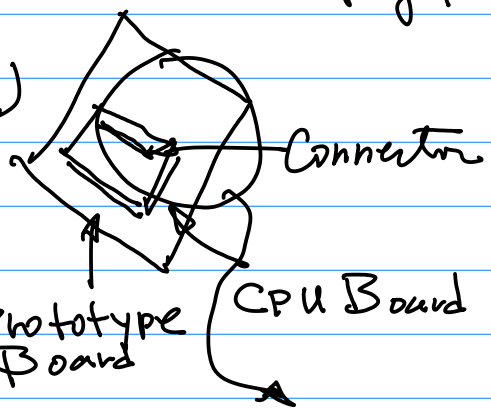


Fig 5

Example: Naming Convention for SPRs:

1° For Control Register

Prefix (3) + Root (3) + Postscript (3)

GPx CON

Make pin 8 as an output pin. → Set GPxCON pin 8 to 1

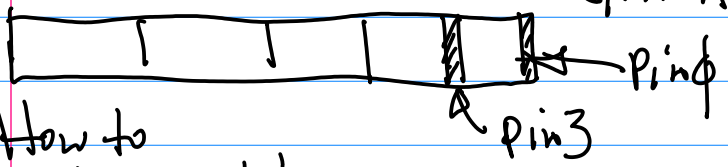
GPxCON[7] = 1

Question: How many control functions can one Control Register realize?

GPACON[3:0] = 0000 Input
Fig1. 15:12 0001 Output

32 Bit GPACON

Question: Define Pin3 GPP Input.
(Pin0, 1, 2, 3 ...)



How to Perform Init & Config?

First GPACON → GPA3 GPACON[15:12]

GPACON[15:12] = 0000 for Input
GPACON[15:12] = 0001 for Output

Suppose that is task (Init & Config)

Question: Find Binary Pattern to Define GPACON ?

0X0000_1000 ✓

Move/Copy this Binary Pattern to 0X7F00_8000

Note: 1° All CON/DAT Registers for GPA ~ GPR

2° To Be Able to Define Input/Output Based on Table Look up.

3° To Be Able to Generate Binary Pattern for Init & Config.

Action 2: Read CPU Datasheet
ARM11 (Samsung)
TX2 (Nvidia)
Pie 3/4 (BroadCom)

Note: Programming Kernel Source/Drivers.

4° make menuconfig
5° Kconfig Script

Note: To Be Able to Write Simple Script.

Config	MODULE_NAME
tristate	① Define 3 options to Build ② followed by "_____" Verbose
depends on	CPU_ manufactor ID + Device ID
help	Text Info

Example of C Code Driver.

printk( )

Kernel Space

Cmpe242

From Fig.3 (pp 11) GPxCON[7] = 1

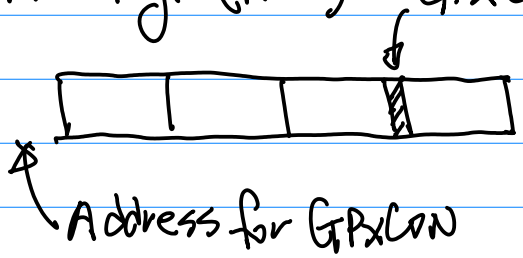
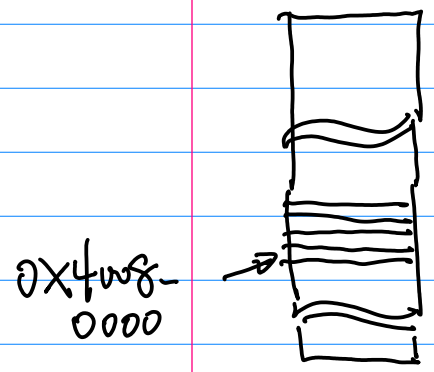


Fig.6

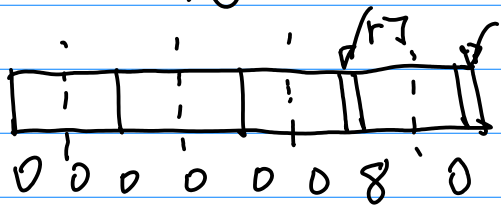
Suppose Addr. = 0x4008_0000

Find the BANK which holds this SFR, addr, GPx Controller

Note: GPxCON occupies 4 Bytes



Init & Config Pattern (Binary)

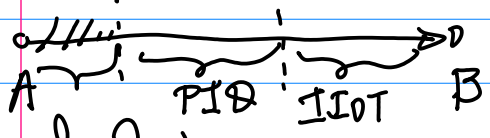


GPxCON[7] = 1

0x80 → Define

Homework ON CANVAS & github

Feb 22nd (Monday)



Kernel + Device Drivers. Background → Architecture + mem. map

1° Kernel Space Driver Development 12

Hands-ON Experience.

Action!

Kernel Source Distribution 250mB + Tool Chain ARM plus Document CPU

2° Perform Init & Configuration of SFRs

Define the Behavior of Peripheral Controller.

Example: Samsung CPU ARM-11

2018S-29-~ (Datasheet)

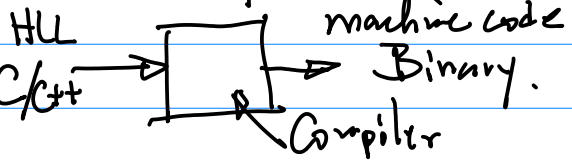
Chapter 10, GPA-GPQ Table Lookup

Section 10.4.1.

GPACON's Address.

0x7F00_8000

ptr move/copy Data 4 Bytes into this memory location



Architecture → IDE → Implementation

Kernel DD

GPP I/O Testing

Section 10.5.1

Write GPACON[3:0]

CMPE242

Feb24 (Wed)

Example: ARM Tool Chain Based Enviroment → ARM11

Linux O.S. Kernel Target

Source → Image → Source Code
Compiled & Built @ Kernel Device Drivers.

(1) O.S. Source Distribution
- Pre-Built
- Compiled + Built

{ ARM11
NVDA TX2

→ Example User Space Example folder (Programs I/F to Device Drivers)

(2) Tool Chain (Compiler + UI)

Char ← Drivers

locate Drivers for the Target CPU "mini6410 ..."

Note: 1° Required to Be Able to write a Simple Driven Test code "hello.c"

2° init, exit — 2 modules

3° PrintK(" "); → Device is installed invoked

Kernel Space

Check the Source code (to Be posted on the github)

* ioctl.h . #define DEVICE_NAME "leds0"

tmp = readl(S3C64XX_GPKDAT);

IO → GPK Port K DATA Register.

*.h porting to make O.S. to Target CPU

#define S3C64XX_GPK

DAT 0X uuuu_uuuu

tmp &= ~(1<<(4+arg));

mask

"~" Negation → "0"

"&" Bitwise AND

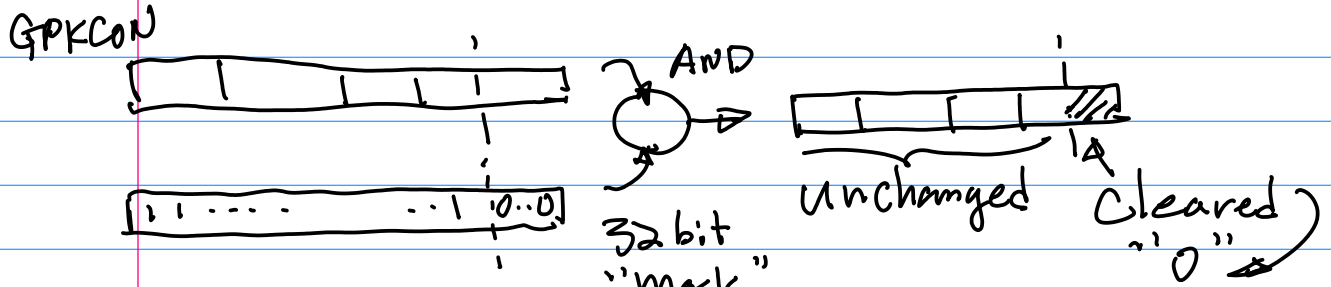
SPRx

Mask → masking Logic Operation @ Bit wise Level

4 bit

Example

GPKCON
GPKCON[3:0]
to init GPK0
Clear GPKCON[3:0]



CPU Datasheet, ARM11,
Section 10.5.5, pp.320
(1) SFRs { CON ✓ DAT ✓ PUL PWR management

(2) 0x7F008080 → GPECON
(3) GPE0~4 : Total 5 pins GPP I/F

GPE3 GPECON[15:12]

Once Driver Done → make menuconfig → KConfig
module Built { a "M" b "*" c Void

Load Kernel Image to the target Board
Done, Pie3B+/4
NAND
TX2

DF, Load Driver module
Copy Driver module *.ko to SD Card
Target platform. (O.S. Running) Linux
insmod : Install module

$ insmod myleddriver.ko
$ "installed Driver"
printk(" ");
$./testmyled

rmmod to Remove the driver

Action 1: (1) Download NVIDIA TX O.S. Distribution 250MB
(2) Download Toolchain
(3) Download Document

Linux O.S.
ARM Tool Chain
→ NVDA
→ Broadcom
→ Samsung
→ Google

Marchist (Mastry)

Jonathon < B T Prototype
Akhil
PID Controller Design { a P.I.D. ~ b PWM Stepper motor c LSM303 I/F

Ref: github : 20185-12-...PID
20185-14-...Stepper motor

Topics : 1° P.I.D Controller Design.

Proportional ~ Integral Derivative ~

Consider Design CNC machine → Stepper motor Control Core

Stepper motor

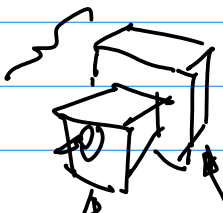
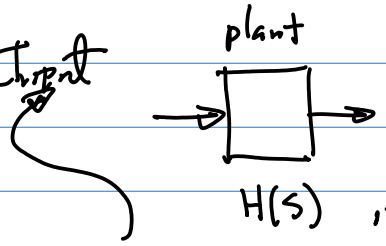
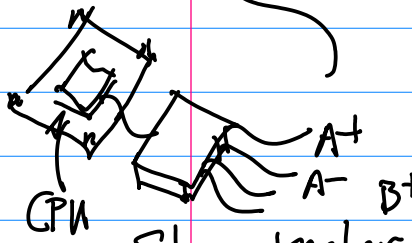


Fig 1.

Gear Box : Reduction Speed ↓ Gain Torques.



" Open-Loop " System → Challenges Uncertainty of the System.



Stepper Motor Drive

Fig 1.6.

Note 1° Stepper Motor Drives Are Equiped with P.I.D. Control functions.

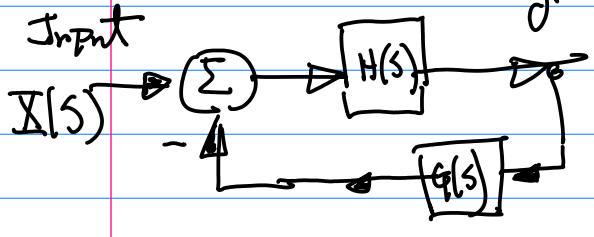


Fig. 2

Comparison of the Actual output with the Desired Output

Input

Note :

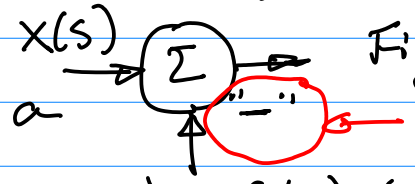


Fig 3

Comparison { a - b → Difference → Error ; a/b }

Negative FeedBack Loop

Note 2°

Performance Enhancement

✓ P: Proportional Controller
✓ I: Integral ~
✓ D: Derivative ~

Frequency Domain

$$E(S) = X(S) - G(S)Y(S) \quad \dots (1)$$

$e(t)$, $e(\tau)$ Time Domain

Control Action Proportional to the error, i.e. Time Domain

$K_e e(t)$ ... (2) (at this moment t)

↳ Gain (Proportional Gain)

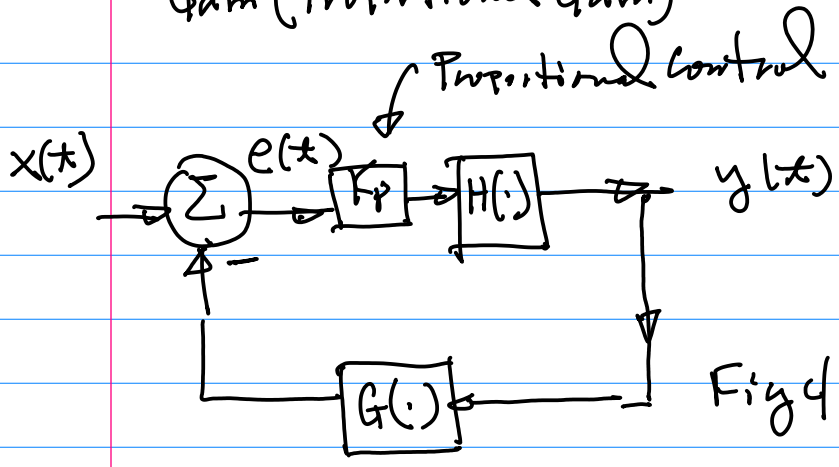


Fig4.

Now, Integral Controller

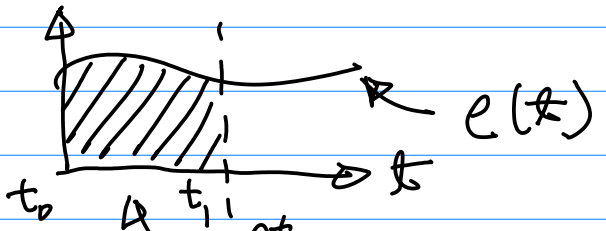


Fig5 $\int_{t_0}^{t_1} e(\tau)\,d\tau$ ... (3)

taking into the consideration of Controller's Accumulative Performance

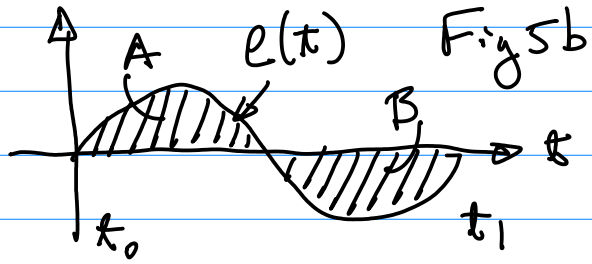


Fig5b

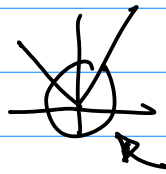
$$\int_{t_0}^{t_1} e(\tau)\,d\tau = A - B = 0 \quad \text{... (3b)}$$

To solve Integration Cancellation Due,

↳

Let's square the error $e(t)$, e.g. $e^2(t)$

Note: Don't use Absolute Value of the error.

Fig6. 

Therefore

$$\int_T e^2(\tau)\,d\tau \quad \text{...(3c)} \checkmark$$

↓

Integral controller

$$K_I \int_T e^2(\tau)\,d\tau \quad \text{... (4)}$$

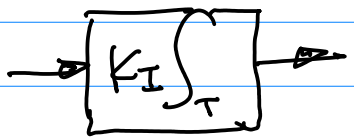
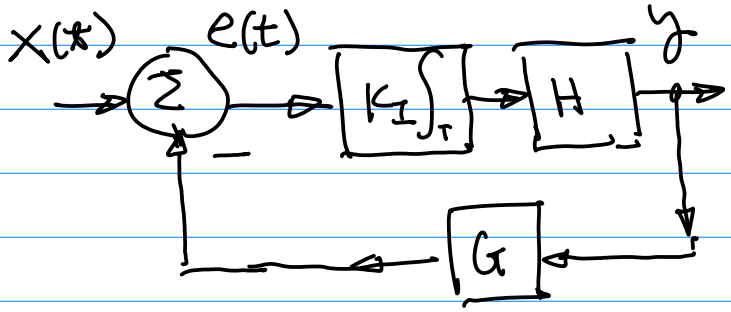


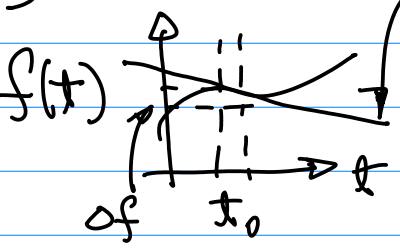
Fig6.

Hence,



Consider Derivatives

Background: $f(t)$



$$\lim_{\Delta t \to 0} \frac{\Delta f}{\Delta t} = \frac{df}{dt} \quad \cdots (5)$$

Changing Rate

Fig 7a

P: Proportional

I: Integral (History Accumulative)

D: Derivative (Predictive)

$\left.\frac{df}{dt}\right|_{t_0} > 0 \rightarrow f(t+t_0)$ ↑

$\left.\frac{df}{dt}\right|_{t_0} = 0 \rightarrow$ f unchanged

$\left.\frac{df}{dt}\right|_{t_0} < 0 \rightarrow f(t+t_0)$ ↓

Action
1. Stepper Motor Drive motorReady
2. GPP + PWM Driver (Software)

March 3rd (Wed)

Topics: 1° PID Controller with Each Subsystem (3) Combined together.

2° C/C++, Python Implementation

Convolution

3° motorDrive

Let $e(t) = f(t)$

$$\frac{d}{d\tau} e(\tau) = e'(\tau) \quad \cdots (6)$$

Build derivative Controller

$$K_d \frac{d}{d\tau} e(\tau) \quad \cdots (66)$$

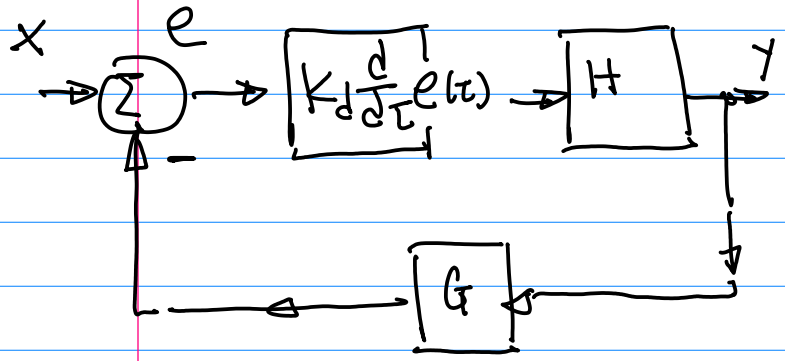


Fig 7b.

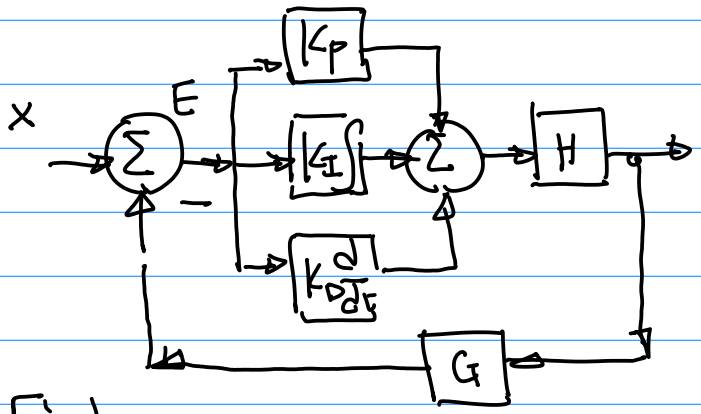


Fig 1.

Predictive

2018S-12-Lecb-PID

{ Transfer function
Characteristic Equation

Example: Implementation Technique
P.I.D controller.

Note: Motor Drive, NEMA 14 or 17

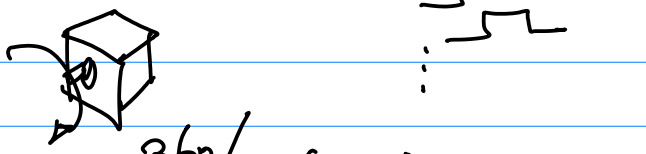
2018S-14- Stepper Controller

1° Stepper motor

Step

Fig1

A+, A−, B+, B− (4 pins)



a) 360/200 (step), 200 steps

b) One step 1.8 Degree
Half Step 0.9 Degree
1/4 0.45 Degree
1/8 0.225 Degree
⋮

Example: Robotics/Self Driving.

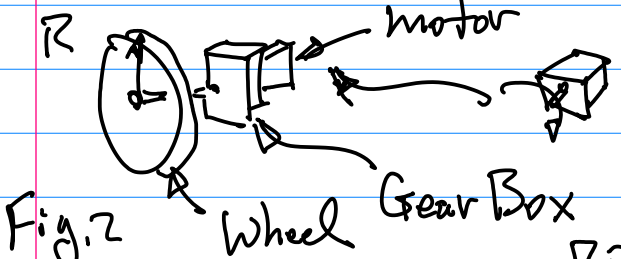


Fig.2

R=10"

Full Step → ? Displacement
(How much can this wheel travels)

GearBox: 58:1

2° Motor Driver

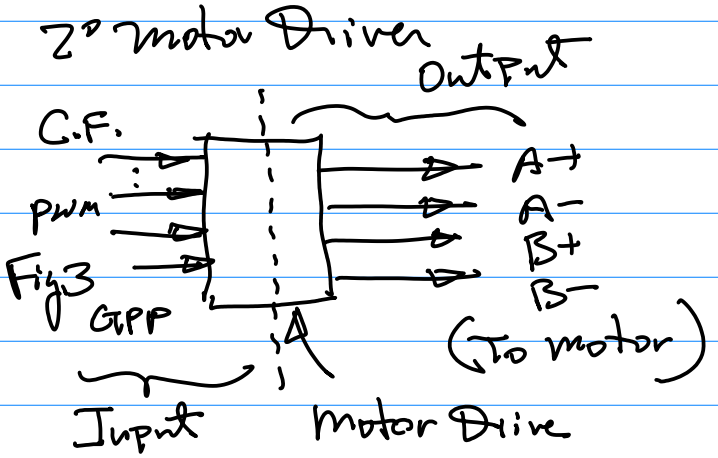


Fig3

Input
a) Configuration: pins K (2^K) K=2 or 3
b) PWM (Speed)
GPP (Direction)

$2^2=4$
$2^3=8$

CF1	CF0	
0	0	f_0
1		f_1

PWM:
Pulse width modulation

Target platform. Pie/NANO/TX2
TCSC-V { Hardware: Pin(s)
Software: Device Driver

Controls the speed of the motor

f_{PWM} to change Speed;

f_{PWM} = 1 KHz, Change Duty Cycle → Speed

Note: CAD Design Tool

- Eagle Linux & Windows
- ORCAD (Cadence) → Eval Version Free 30 Days Win

Schematics → Required in Project Report

Action 1: Build motor Drive and Connect to NEMA 14 or Equivalent, Try to Drive it, please bring your Prototype Board to the class for show & Tell.

Example: PID Implementation.

1. Error Signal $e(t)$, From Fig. 1 P.P. 17

Find Error { LSM303 Direction
Optic Encoder (self Driving Robot)
Displacement / Wheel traveled

Suppose the error is denoted as

$e(t) \longrightarrow e(k)$ k index for time

Continous System — Discrete System

Sampling process

Suppose the set of error data is tabulated in the following Table:

Time Index k	Error $e(k)$	Note
k_0	0.1	Example
k_1	1.1	
k_2	1.5	
k_3	0.5	
k_4	-0.1	
k_5	-2	
⋮		

Table 1

midterm 24 (Wed) 1 hr.
Close Book / Close Notes
Project 1 22nd (monday)
CANVAS will be posting Project 1 March 8.
Ref: Github on PID

Example: Continued from the Last Example on this page.

Note 1. Compute Derivative From

$$\frac{de(t)}{dt} = \lim_{\Delta t \to 0} \frac{\Delta e}{\Delta t} \quad \cdots (1)$$

$$\Delta e = e(t+\Delta t) - e(t) \quad \cdots (2)$$

Δt Sample time interval
Nyquest Sampling Frequency

$f_{Sampling} \geq 2 f_{max}$ ... (3)

$\Delta t, 2\Delta t, 3\Delta t, \ldots, k\Delta t, \ldots$

$1, 2, \ldots, k, \ldots$

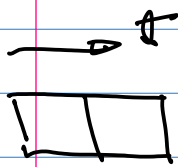
Time Index

Eqn(2) $\Delta e = e(t+\Delta t) - e(t)$

$= e(t+1) - e(t)$ ... (4)

Visualized way of Eqn(4)

Table ("Kernel") — Convolution



... Fig.1

From Eqn(1) & (4)

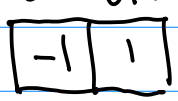
$$\frac{de(t)}{dt} \triangleq \lim_{\Delta t \to 0} \frac{\Delta e}{\Delta t} \simeq \frac{e(t+\Delta t) - e(t)}{\Delta t}$$

$$= \frac{e(t+1) - e(t)}{1} = e(t+1) - e(t) \quad \ldots (5)$$

Map Eqn(5) to the Kernel

$e(t+1) - e(t) = 1 * e(t+1) + (-1) * e(t)$

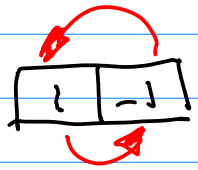
t t+1



Kernel (6)

Using (6) Kernel to Compute Derivative of the error (table)

k=0	1	2	3	4	5
0.1	1.1	1.5	0.5	-0.1	-2
$\frac{de}{dt}$ =	-1.0	-0.4	1.0	..	..



$1 \times 0.1 + (-1) * 1.1$

$= 0.1 - 1.1$

$= -1.0$

$-1 * 0.1 + 1 * 1.1 = 1.0$

$1 * 1.1 + (-1) * 1.5$

$= -0.4$

$-1.1 + 1.5 = 0.4$

$1 * 1.5 + (-1) * 0.5$

$= 1.0$

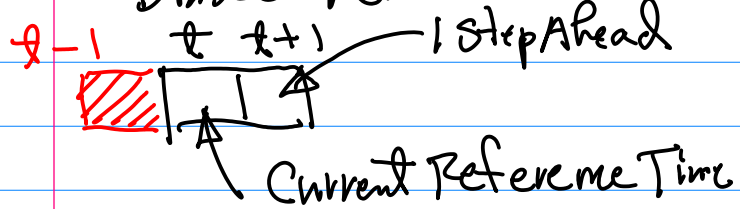
$-1.5 + 0.5 = -1$

Note: The Above Graphical Way Evaluating Derivatives is based on Convolution And Can be implement with C/C++ using double for-loop.

Action1: Write C/C++, Python to implement this as Convolution

Improvement I

"Biased" Kernel



Note 1° Add time index such that the contributing error terms are taken from evenly Balanced Time interval.

2° Forward Difference for Derivative Computation (Numerical)

From Eqn(5)

$$\frac{d}{dt}e(t) = \lim_{\Delta t \to 0} \frac{\Delta e}{\Delta t}$$

"t+1" forward

$$\begin{cases} \Delta e = e(t+1) - e(t) & \cdots (6) \\ \Delta e = e(t) - e(t-1) & \cdots (6b) \end{cases}$$

Backward Difference

$$(F.D + B.D)\frac{1}{2} = \frac{1}{2}\big(e(t+1) + e(t) - e(t) - e(t-1)\big)$$

$$= \frac{1}{2}\big(e(t+1) - e(t-1)\big) \quad \cdots (7)$$

Kernel

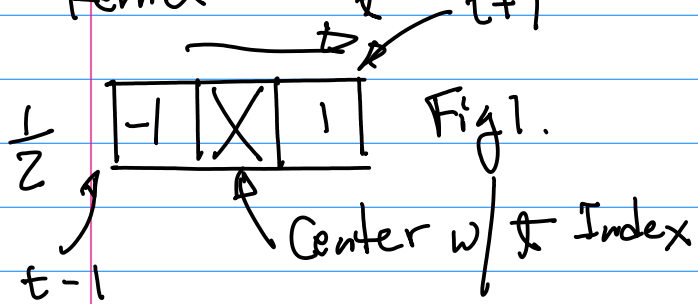


Fig 1.

Exercise: Compute $\frac{d}{dt}e(t)$ in the table! TP19

Note: C.D. takes care of Smoothing the Random Noise.

Low pass Filter.

Improvement II.

"Smooth" Filter out Random Noise while taking derivative

LoG. Laplace of Gaussian.

$$\nabla^2 G(x) \quad \cdots (8*)$$

Gaussian Function

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}} \quad \cdots (8)$$

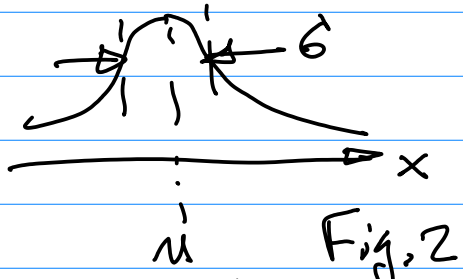


Fig. 2

Low Pass Filter

"Weighted"

Low pass filter

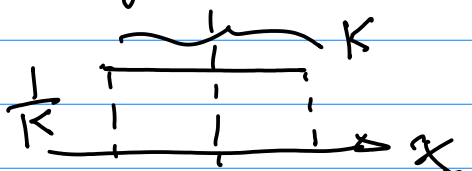


Fig 3.

→ Action: Write C/C++, python to do this convolution with C.D, F.D., B.D.

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$