

Jan 27, 21
Welcome to

CMPE242 Harry Li

Embedded Hardware Systems

1. GreenSheet github/huawili/cmpe242

Email: hua.li@sjtu.edu

(650) 400-1116 Text message

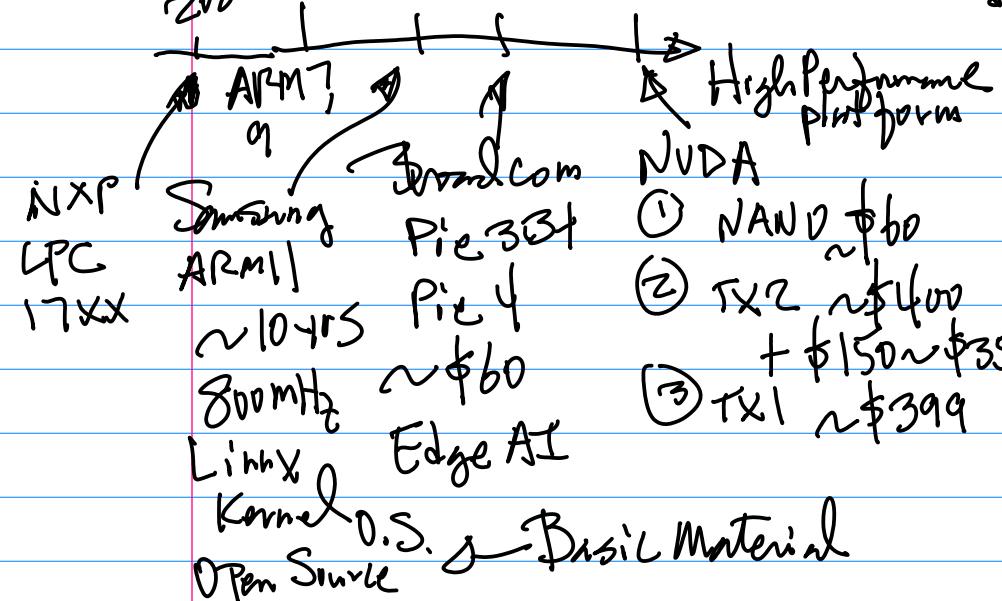
2. Pre-requist Requirements ISO A4

Course Description

Hands-ON.

Target Development Platform

ZED



Scope of the Course

Device Driver

Dep Development

C/C++, Python

O.S. Kernel

1. Assignments (No)

2. Collaboration / Submission of your Class Work

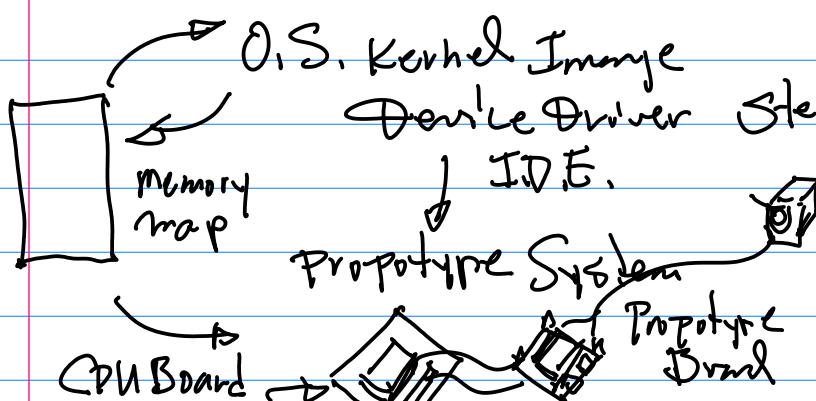
Action 1. github/huawili/cmpe242

2. Datasheet

Samsung ARM11 Datasheet

NXP i.MX6 Datasheet

Architecture PXPiPC 17xx



Action 3. Target Platform Selection

a) Unix-like OS. b) Edge AI Computing (Scalability) \Rightarrow GPU

Office hours M.W. 4:30-5:30pm
ON ZOOM.

Baseline Software

Linux Distribution
Optimized for Embedded platform
Device Drivers.

Example: Datasheet

LPC1769 ARM Cortex m3

CnPE242 Feb. 2021

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

Samsung ARM-11

NVDA

Piex

System Architecture

→ NVDA CPU/GPU platform.
→ Broadcom, Pi 3/4 G.E. (Graphics Engine)
→ Samsung ARM-11 (9, 7)

NANO
TX1, TX2

CPU Datasheet, LPC1769
2018S-3-1M10360, FP. 9

Jtag

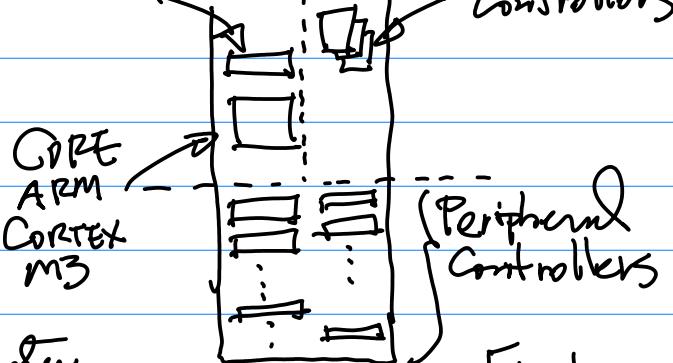


Fig 1.

* 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

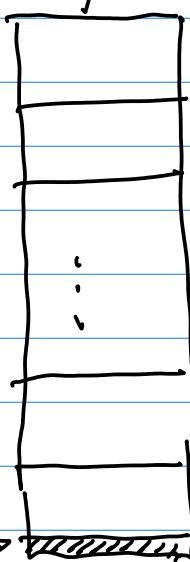
Base Line Hardware (Datasheet); ARM11

Fig 1. CPU Architecture
ARM Cortex M3

Note: To Be Able to Draw/
Design CPU Architecture

Either this OR Your Target
platform (1769 + ARM11)
Base Line

Memory map.



$$2^{32} = 4 \text{ GB (B)}$$

0x0000-
0000

Fig. 2

① 32 Bit RISC Architecture

$$2^{32} = 2^0 \cdot 2^0 \cdot 2^0 \cdot 2^2$$

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

$$= 4 \text{ GB} \quad \text{(Byte)}$$

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: 4 GB / 8

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

$$= 512 \text{ MB}$$

Starting Address of Each Bank

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

3 bits Needed

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

Little
Indian

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

Find Starting Address for Bank :

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

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

0x0000-0000

2nd Banks Starting Address

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

$$\underbrace{0 \ 0 \ 1}_{\text{ }} : 0$$

0x2000-0000

3rd Banks Starting Address

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

$$\underbrace{0 \ 1 \ 0}_{\text{ }} : 0$$

0x4000-0000

Now, Consider target Board
Conditions to qualify the Selection

Plus: Many examples
on Device Drivers (I2C,
PWM, SPI, UART, ...)

- (1) ARM Based; (2) Linux-Like O.S. (Bootloader Tool 4/5)
- (3) Establish Linux Kernel eco-System
- Developer Base (~ millions) { Some Distro.
Tool Chain }
- (4) Technology Innovators / Leaders.
- (5) External Expansion Connectors

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

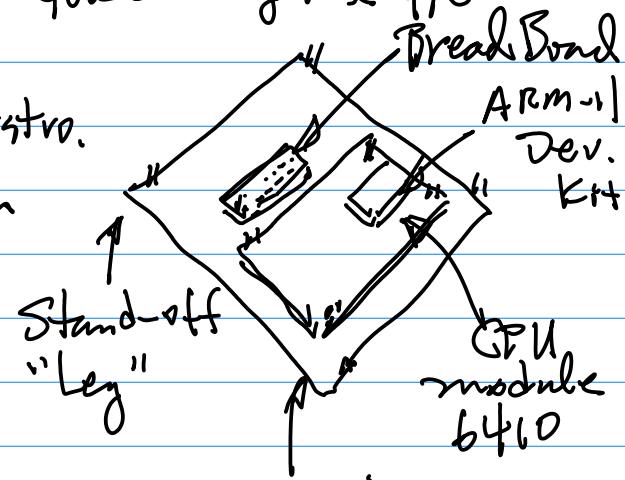
1-ZeroS-left1-hardware board...

ARM-1 | ① Coupled w/ Linux Open Source
Distro { Kernel Sources
② ARM tool Chain
Datasheet Baseline Reference
Requirements, Exams }

DrawBack: Lack of the Ability to

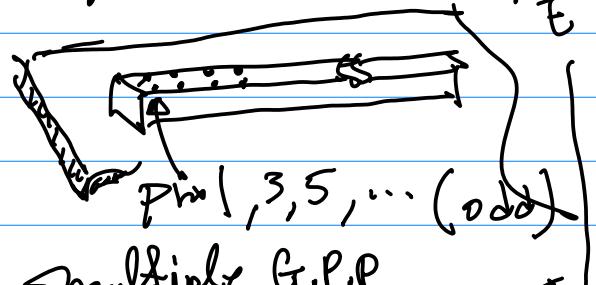
Handle Edge AI; (G.E.) Tiny b410

Kit from FriendlyARM.com, ~\$590



Note: Bread Limitation -
Run High Speed
Cannot ~20 MHz

CNN.5 GPEx → General
Connector I Purpose (out)
Pin Number physical location



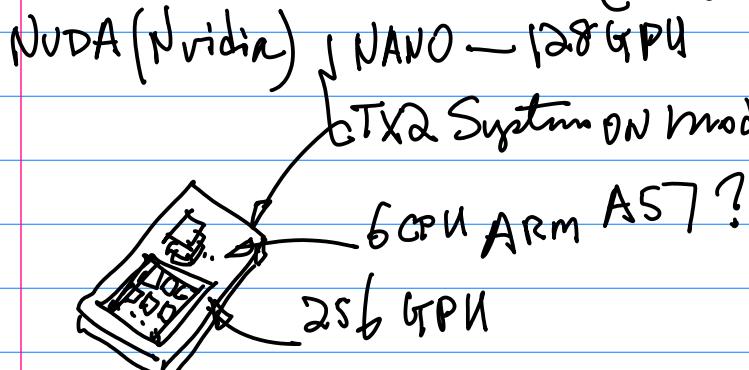
GPEx : GPE3

pin of the Port

Option (Pie Board STB+, 4)

\$75 Pie-4, 8GB mem.

ComPEasy 2



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 (NVDA TX2 System-on-Module)

1° Expensive! \$299 + Carrier Board
Edge on AI \$150 - Best

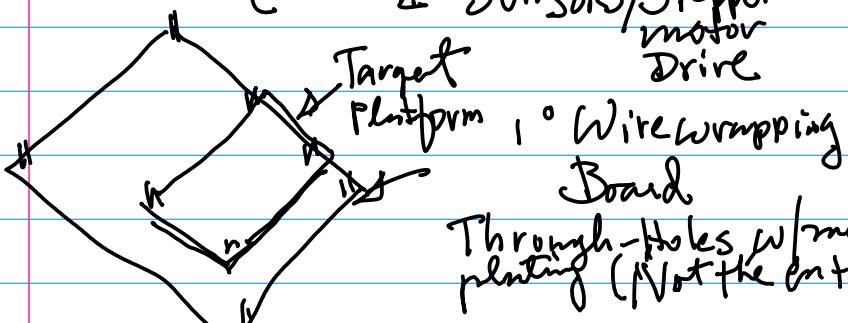
If this is selected, then Z → 4 person
to Share the Cost

Note: 1° 4-Person Team By Next Week;

2° 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

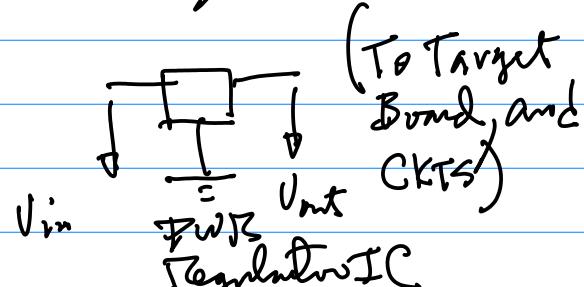


1° Wirewrapping
Through-Holes w/metal
plating (Not the entire side)

2° 4 Stand-offs ("Legs")

3° Components to Build

PWR CKT & CLK for
I/O Testing ("Hello, the
world").



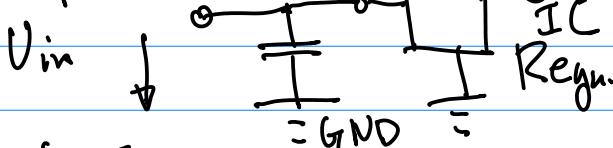
78xx 7805 7812
5VDC 12VDC

Red LED, 4 ~ 10 mA

Resistors. $V_{CC} = 5.0$

$$R = \frac{V}{I} = \frac{5}{4 \times 10^{-3}} = 1.25 \text{ k}\Omega$$

Cap for your External Power Input Clk.



LPF
(Low Pass Filter)

$$C = 4.7 \text{ nF} \quad T = RC \Rightarrow$$

3 dB

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 / OP Amps

1. Target:
 - (1) Pic3B+, Pic4
 - (2) NANO, TX2 (Edge AI)
 - (3) ARM11 — Baseline Reference platform

2. Prototype Board, 6" x 4"

POWER CKT : PWR Regulator IC

Wallmount Adapter (M7805) 2VDC dropoff Note: you may need higher voltage source to Stepper motor Drive.
3.
V_{in} C || V_{out} GND Target To CPU

Fig. 5VDC

Connections

- External PWR
- Internal PWR

4. "Glow" Logic { Asorted Resistors
Caps, 7805 200 ~ 5KΩ 10KΩ

Switch (Toggle Switch)

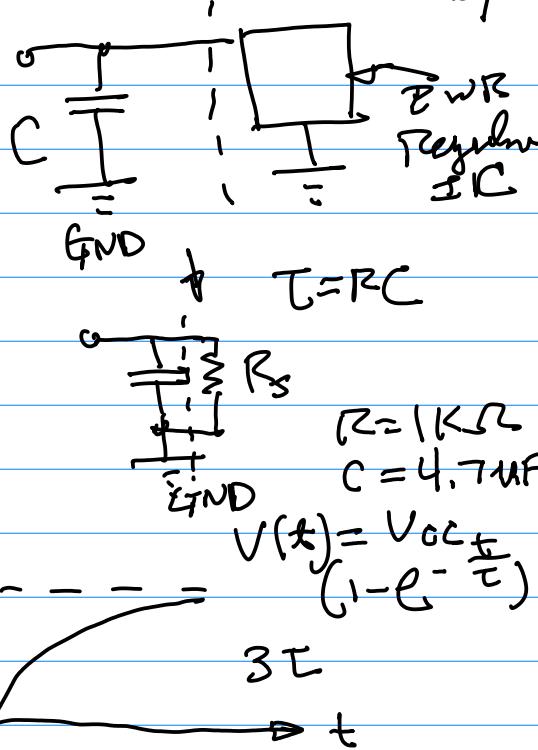


To Control PWR Source;

① PWR Distribution Node

PWR Input Node
LPF (Low Pass Filter)

To Build I/O Testing

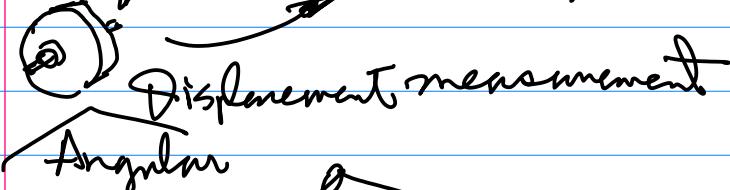


Phase II (Bill of Material):

Ref: github:

1. LSm303 (I2C) → Autonomous Robot

2. Optical Encoder (optional)



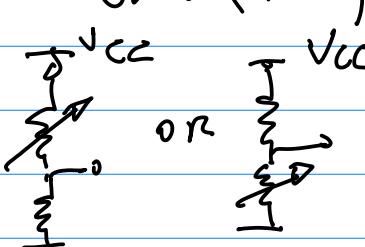
Angular
SENTR₁ OR Front wheel Drive
SENTR₂

3. Drive (Powered)
CPU module
Prototype

b Bulk Counter
= is good;



= 12VDC Commonly
used Source, Check
Current Need;



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

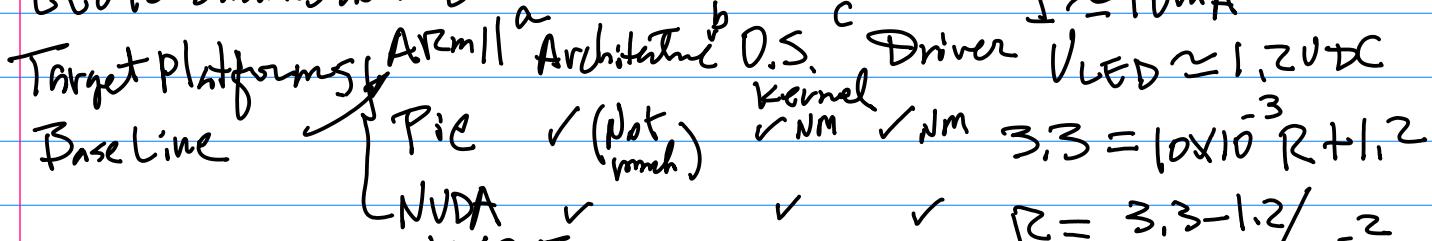
Feb 10.

Homework 1 Due Feb 24 (W) 11:59 pm.

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

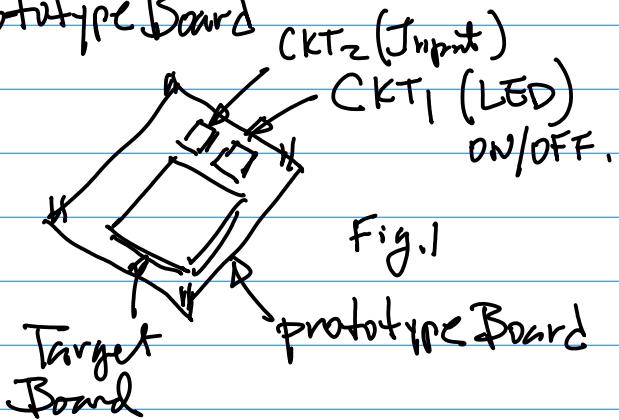
CmpE242 CANVAS

EE242 Submission to E-mail:



1° Target platform Built on

Prototype Board



Note: Find the pin(s) from Target platform.
 Pic3 GPIO14 — Pin 8

Consider Input Testing CKT

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

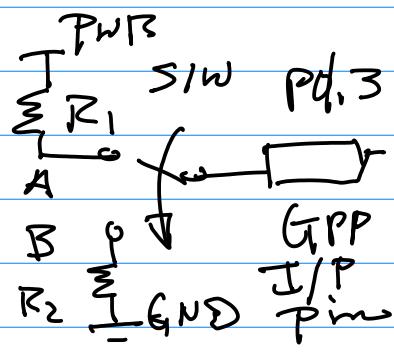
Ref: I-tec-Hardware Board —

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
 CKT,
 { "1" Turn on LED
 "0" Turn off LED }



Design:

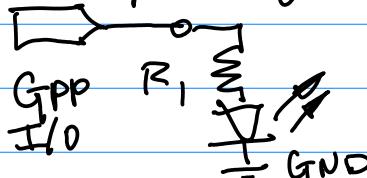
Identify the GPIO pins for Input
and Output Testing CKT.

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

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

NXP
LTC
1769

Pd.3 GPP Input pin



Pd.21 GPP Output Pin

$$R_2: 3.3 \text{ (from GPP)} / 10 \times 10^{-3} = 330 \Omega$$

9/
0.5.
I/O
function

System (Architecture) & Software Design. Boot Loader

3° Source code, Binary \rightarrow 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 Coordination 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: Z-2020S-Lec2 ...

=
Example: 1) CPU Broadcom BCM2835

2) PWR1 (SP3), 2, 4 (5VDC)

3) GPIO for Homework \rightarrow Choose
pin those not

marked w/ other
function

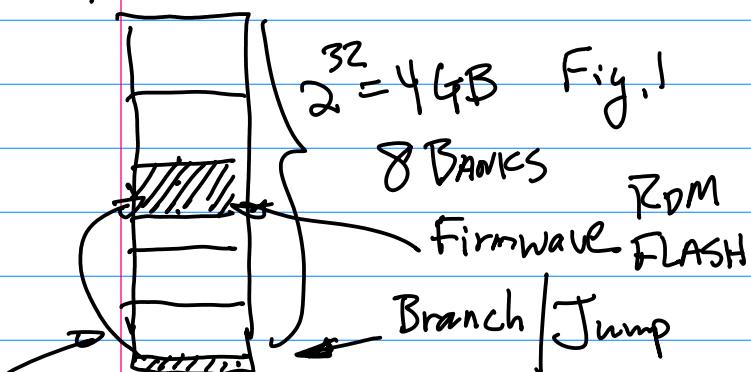
Concept of Device Driver

↳ Architecture + memory map

↳ Software + Kernel (OS)

↳ Device Drivers

Architecture

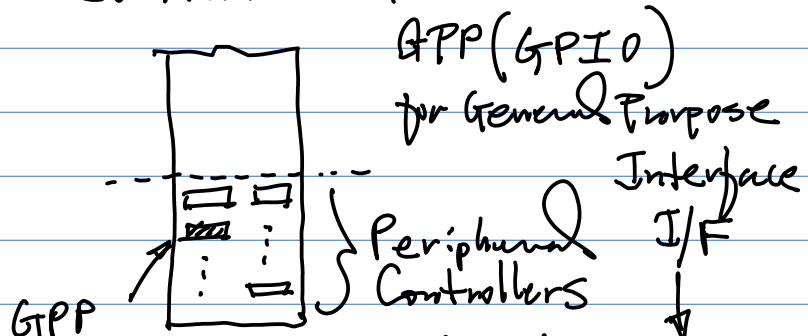


BANKs } (PP2.Fig4)

0x0000_0000 PWR-up Address

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

CPU Architecture



From Fig 1.
PP.2 A set of SPRs
(Special Purpose
Registers)

Note:

1° SPRs are 32 bits.

2° SPR's Formations into 3 Categories

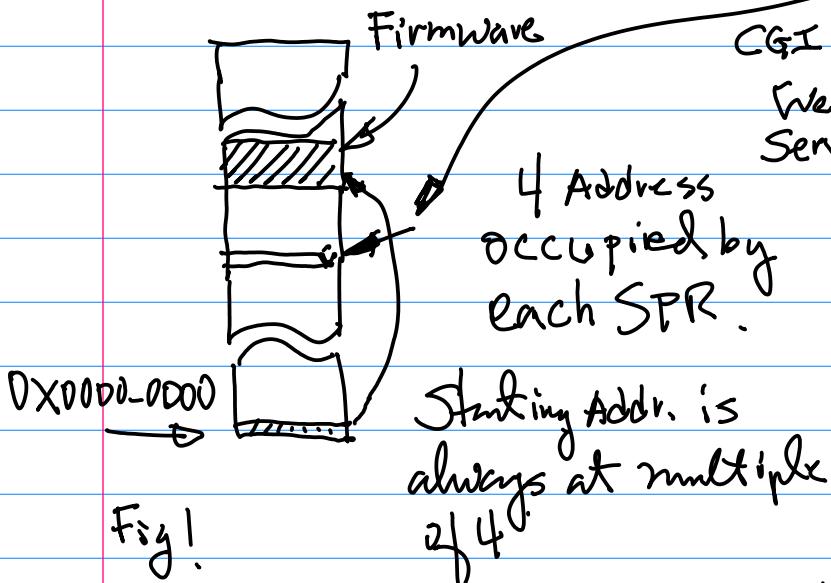
- 1° Control Function, Init & Config
- 2° Data
- 3° Pull Up/Down

3° Map 32bit SPR onto memory



32bit SPR

which is mapped to the memory location



1° PWR up Address: ~ when CPU is powered up, it will fetch the 1st Instruction from this location ~ /CMPE242/2018S-29-CPU (ARM11 CPU Datasheet)

Feb 17 (W)

Ref: CPU Datasheets

1° github ~ 2018S-29

2° Boot up Sequence

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

3° OS. Image is Being Loaded

Then user space program can be executed, And the device(s) can be accessed via Device Driver in Kernel space, open V_TF Space

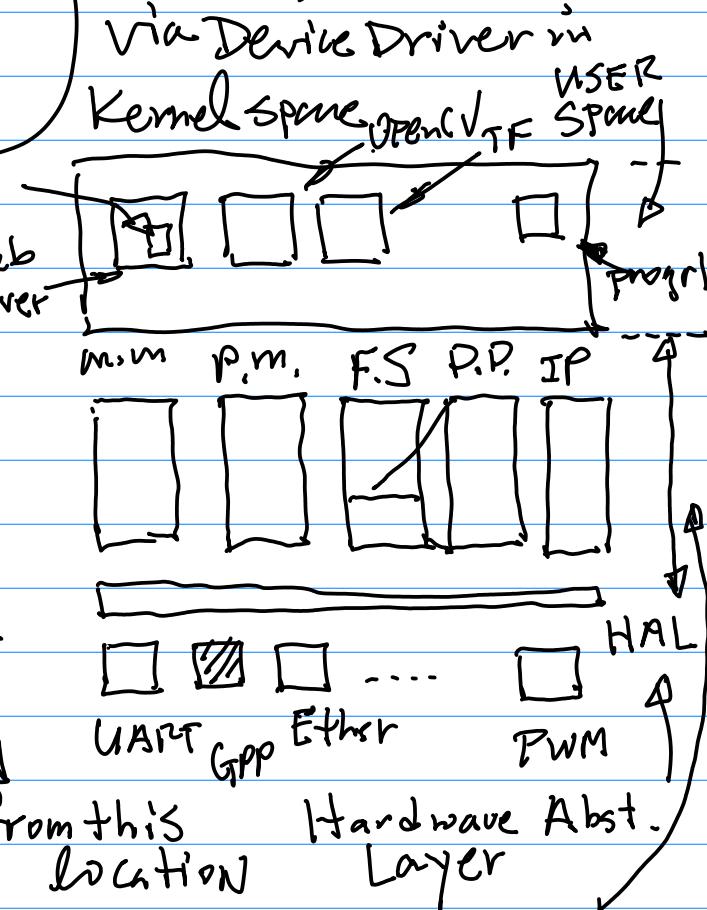


Fig 2. Kernel Space

Example: Prog1.C → USER Space

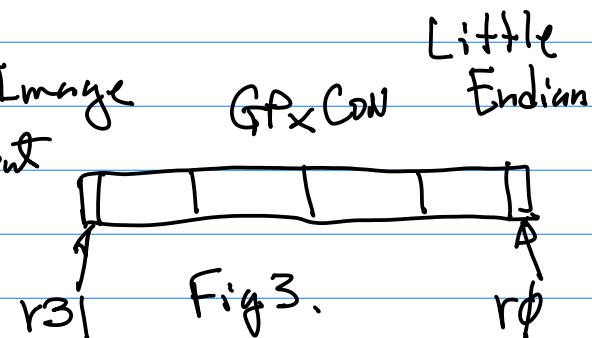
UserSpace \rightarrow open("dev");
 $fd =$

"dev"

\downarrow
 Path Device Driver
 location in
 the Kernel Image
 \downarrow
 read(fd,
 Buffer, Size for GPP Input
 Testing)

GPxCON

$$2^{32} = 4 \text{ G}$$



4. Kernel Space:

OS : manage ~ Resources &
 Policy

DeviceDriver(s) : A collection of
 program(s), manage/manipulate

Special Purpose Registers, to be
 able to utilize peripheral controller(s)

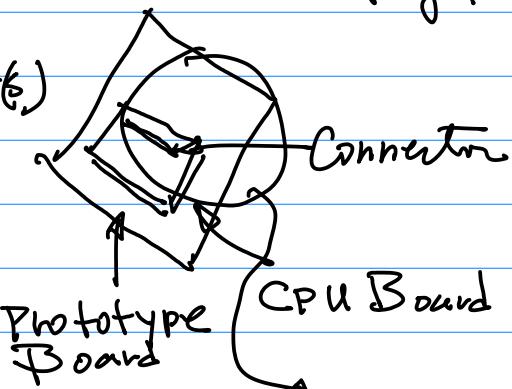
Init & Config of Special Purpose Registers

Example: Naming Convention for
 SPRs.

1° For Control Register

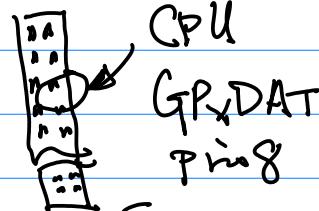
Prefix + Root + Postscript
 (3) (3) (3)

GPx CON



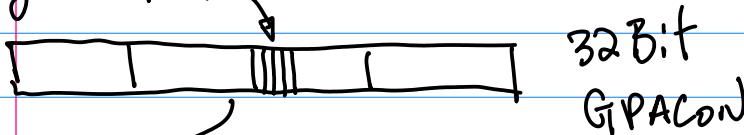
Make
 Pin 8
 as an
 output

Pin. \rightarrow Set GPxCON
 pin 8 to 1

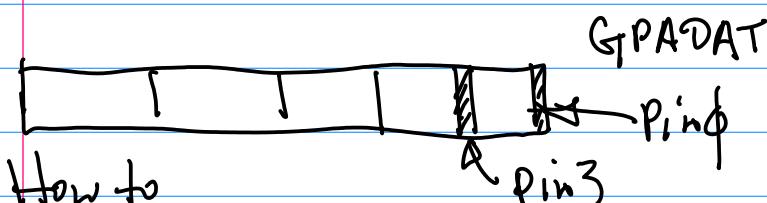


Question: How many control functions
 can one Control Register realize? $GPxCON[7] = 1$

$\text{GPACON}[3:0] = 0000$ Input
 Fig 1. $15:12 \downarrow 0001$ Output



Question: Define Pin3 GPP Input.
 $(\text{Pin}0, 1, 2, 3 \dots)$



How to Perform Init & Config?

First GPACON \Rightarrow GPA3 GPACON[15:12]

\downarrow
 $\text{GPACON}[15:12] = 0000$ for Input

$\text{GPACON}[15:12] = 0001$ for Output

Suppose that is task (Init & Config)

Question: Find Binary Patterns to

Define GPACON?

0x0000_1000 ✓

Move/Copy this Binary Pattern

to 0x7FOO_8000

Note: 1° All CONF/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 Details about
 ARM11 (Samsung)
 TX2 (Nvidia)
 Pix3/4 (Broadcom)

Note: Programming Kernel
 Sample/Drivers.

4° make menuconfig
 5° KConf Script

Note: To Be Able to Write
 Simple Script.

Config	MODULE_NAME
tristate	(1) Define 3 options to build (2) followed by "...." Verbage
depends on	CPU - manufacturer ID + Device ID
help	Text Info

Example of C Code Driver.

`printf()`

Kernel Space

Cmpe242

From Fig.3 (pp 11) $GPxCON[7] = 1$ Kernel Space Driver Development
Hands-On Experience.

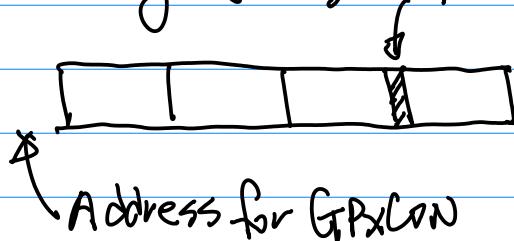


Fig.6

Suppose Addr. = 0X4008_0000

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

Note: GPxCON occupies 4 Bytes

Action 1
Kernel Source + Tool Chain
Distribution + ARM
250mB + plus
Document
CPU

Action 2
Perform Init & Configuration
Define the Behavior of
of Peripheral
Controller.

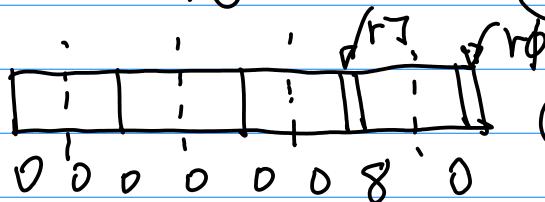
Example: Samsung CPU ARM-11
2018S-Z9 - ~ (Datasheet)

Chapter 10, GPA-GPQ Table
Look up

Section 10.4.1.

GPACON's Address

Init & Config Pattern (Binary)

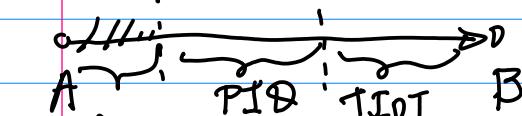


$GPxCON[7] = 1$

0x80 → Define

Homework ON CANVAS & github

Feb 22nd (Monday)



Drivers. Background → Architecture + Mem.map

Architecture → IDE → Implementation
Kernel DD

GPP I/O Testing

Section 10.5.1

within GPACON[3:0]



CMPE242

Feb 24 (Wed)

Example: ARM ToolChain Based
Environment → ARM11

Linux O.S. Kernel Target

Source → Image →
Compiled & Built Source Code
@ Kernel
Device Drivers. $\text{tmp} = \text{readl}(S3C64XX_$ $\text{GPKDAT});$
ID GPK Port K
DATA Register.

ID

*.h

posting to make
O.S. to Target CPU

#define S3C64XX_GPK

(1)

O.S. Source
Distribution → Pre-Built
ARM11
NVDA TXR
→ Compiled + BuiltDAT 0X nnnn-nnnn
→ Example $\text{tmp} \&= ~(1 \ll (4 + \text{arg}));$

User Space Example

mask
folder (Program
I/F to Device Drivers) " & " Bitwise
AND

" ~ " Negation → 0

AND

" & " Bitwise AND

Char & Drivers

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

Note: 1° Required to Be Able to "hello.c" Mask → masking SPRX
write a Simple Driver Test Code Logical Operation
2° init 2 modules @ Bit wise level
exit

3° Printk(" "); → involved

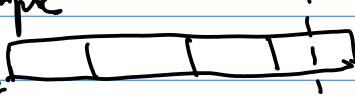
Devices is installed

Kernel Space

Check the Source Code (to Be posted
On the github)

* ioctl.h . #define DEVICE_NAME "leds0"

Example



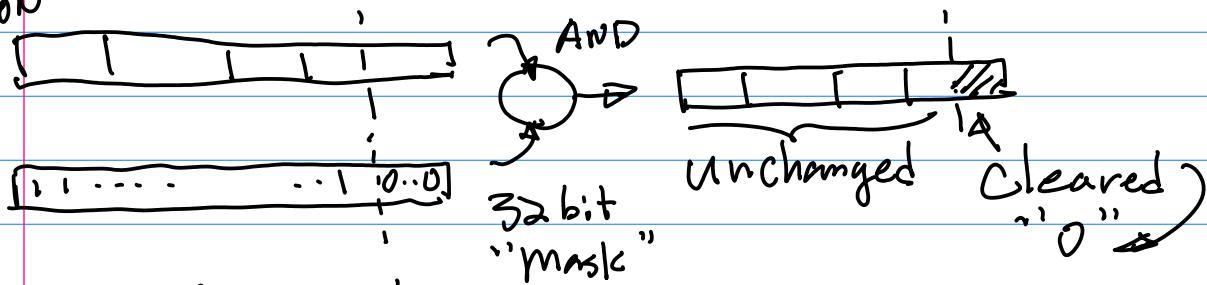
GPKCON

GPKCON[3:0]

to init GPKφ

Clear GPKCON[3:0]

GPKCON



CPU Datasheets, ARmI

Section 10.5.5, pp. 320

(1) SPRs { CON ✓
DAT ✓
PUL }

Powermanagement

```
$ insmod myledriver.ko
$ "installed Driver"
A
printf(..)
..);
$ ./testmyled
rmmod to Remove
the driver
```

(2) 0XF008080 → GPECON

(3) GPECON : Total 5 pins GRPP I/F

GPE3 GPECON[15:12]

Once Driver Done → make menuconfig

Load Kernel Image
to the target
Board

Done, Pie3B+ /
NAND
TX2

Action 1: (1) Download
NVDA TX O.S.
Distribution
(2) Download
ToolChain
(3) Download
Document

DR, Load Driver module

Copy Driver module *.ko to SD

Target platform. (O.S. Running)
Linux

insmod : Install module

NVDA
Broadcom
Samsung

Marshall (mainly)

Groups

Jonathan <β> Prototype

Akhil

PID Controller

Design

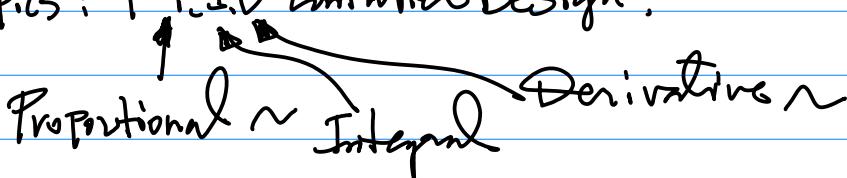
\hat{a} P.I.D. ~
 \hat{b} PWM Stepper
motor
 \hat{c} LSM303 I/F

Ref: github: 2018S-12 - ...PID

2018S-14 - ...Stepper Motor

Comparison of the Actual output with the Desired Output

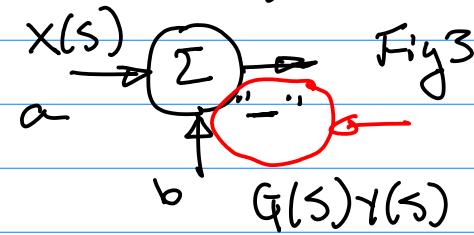
Topics: 1^o PID Controller Design.



Note:

Input
a - b

Fig 3



Consider Design CNC machine

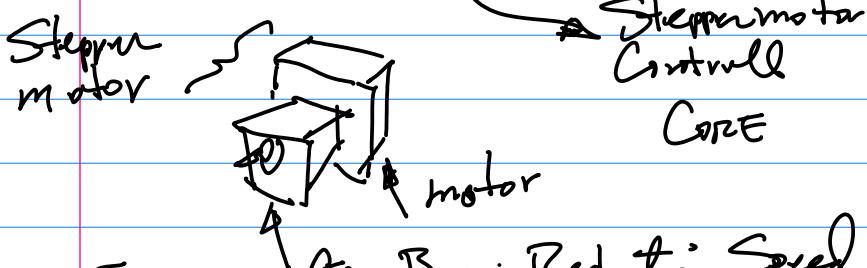
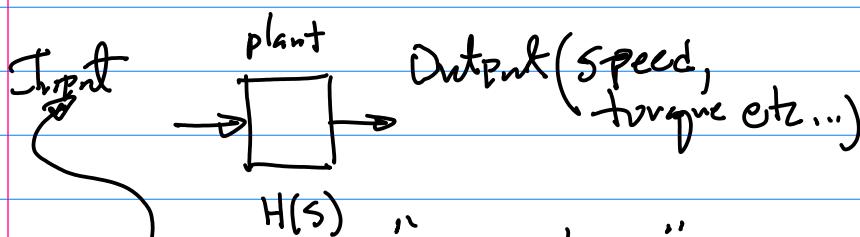


Fig 1. Gear Box: Reduction Speed
Gain Torques.

Comparison
{ a - b ✓ Difference
a/b Error

Negative Feed Back Loop



Note 2^o

Performance

Enhancement

- ✓ P: Proportional Controller
- ✓ I: Integral
- ✓ D: Derivative
- Frequency Domain

$$E(s) = \Sigma(s) - G(s)Y(s)$$

... (1)

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

Control Action Proportional to the error, in Time Domain

Note^o Stepper Motor Drives Are of the Equipped with P.I.D. Control System.
functions.

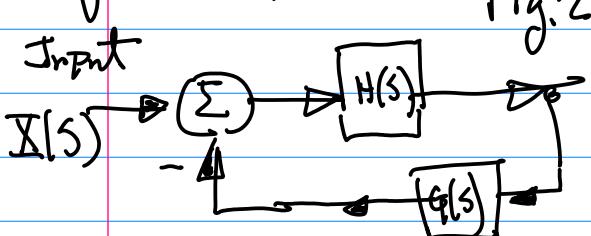
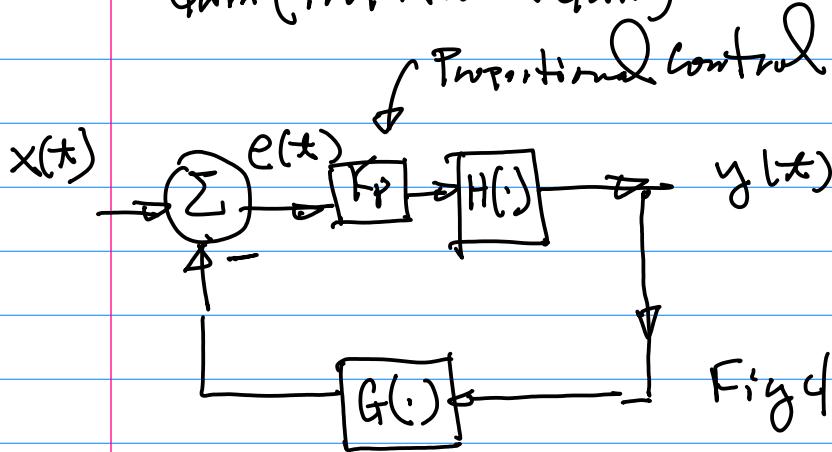


Fig 2

Output
 $Y(s)$
Sensor(s)/Transducer

$K_p e(t) \dots (z)$ (at this moment) To Solve Integration
 ↑ Gain (Proportional Gain) \rightarrow Compensation Due, \rightarrow

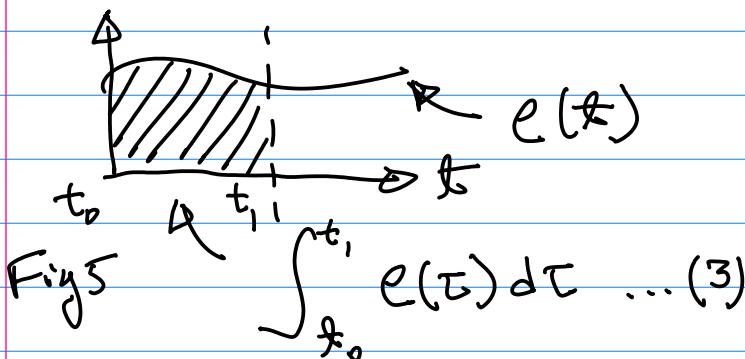


Let's ignore the error $e(t)$, e.g. $e^2(t)$

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

Fig 5. ~~|e|~~ No Derivative

Now, Integral Controller



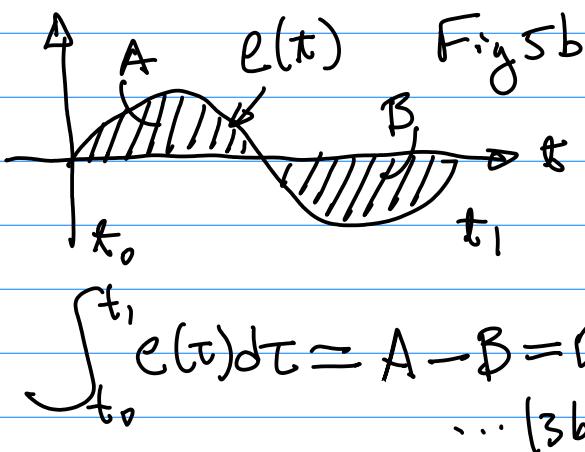
Therefore

$$\int_T e^2(\tau) d\tau \dots (3c)$$

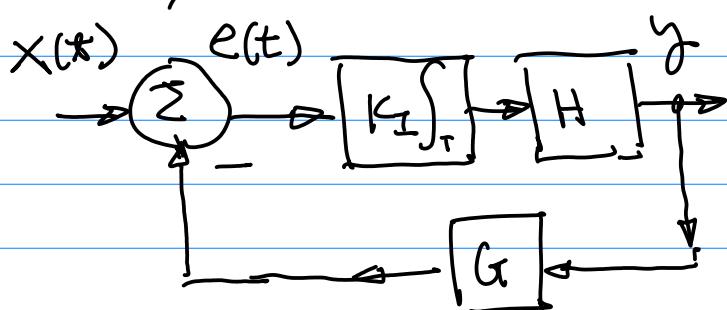
Integral Controller

$$K_I \int_T e^2(\tau) d\tau \dots (4)$$

taking into the consideration of Controller's Accumulative Performance



Hence,



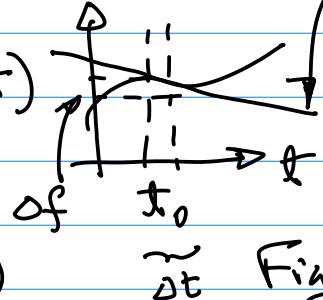
Consider Derivatives

Background: $f(t)$

$$\lim_{\Delta t \rightarrow 0} \frac{\Delta f}{\Delta t} = \frac{df}{dt}$$

Changing Rate

... (5)



Derivative

P: Proportional

I: Integral (History
Accumulative)

D: Derivative (Pre-
dictive)

- Action
- 1. Stepper motor Drive
motor Ready
 - 2 GPP+PWM Driver
(Software)

$$\frac{df}{dt} \Big|_{t_0} > 0 \rightarrow f(t+t_0) \uparrow$$

$$\frac{df}{dt} \Big|_{t_0} = 0 \rightarrow \text{unchanged}$$

$$\frac{df}{dt} \Big|_{t_0} < 0 \rightarrow f(t+t_0) \downarrow$$

March 3rd (Wed)

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

$$\frac{d}{dt} e(t) = e'(t) \quad \dots (b)$$

Build derivative controller

$$K_d \frac{d}{dt} e(t) \quad \dots (bb)$$

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

2° C/C++, Python Implementation

Convolution

3° Motor Drive

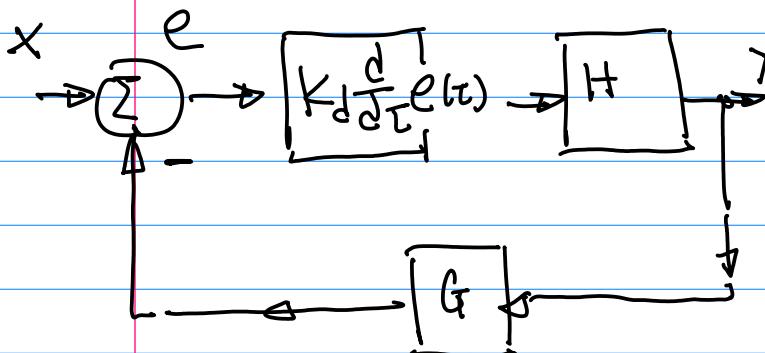


Fig 7b.

Predictive

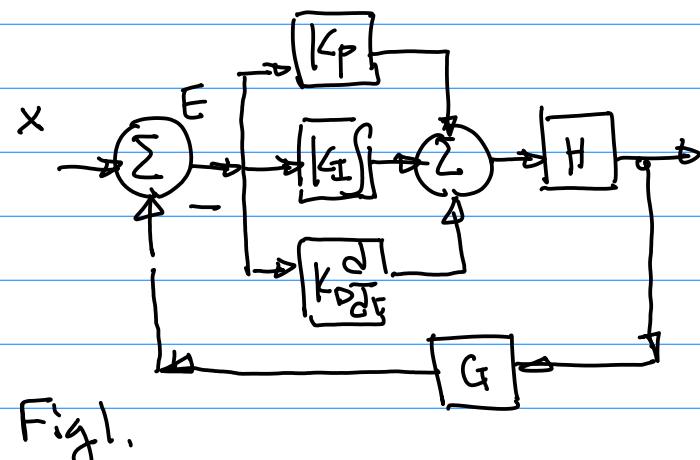


Fig 1.

2018S-12-Lecb-PID

Full Step \rightarrow ? Displacement

{ Transfer function
Characteristic Equation }

(How much can this wheel travel)

GearBox: 68:1

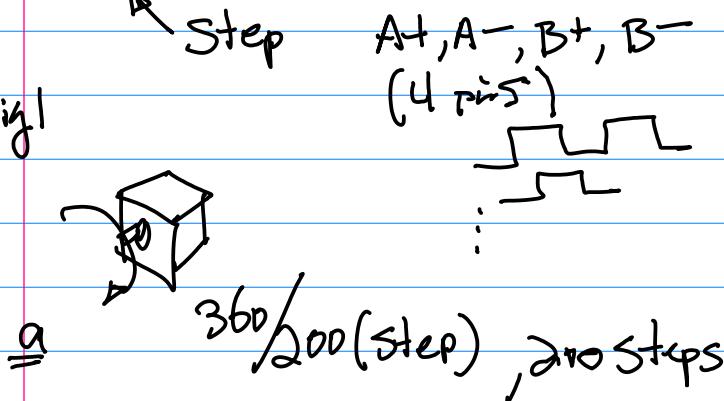
Example: Implementation Technique
P.I.D controller.

Note: Motor Drive, NEMA 14 or 17

2018S-14- Stepper Controller

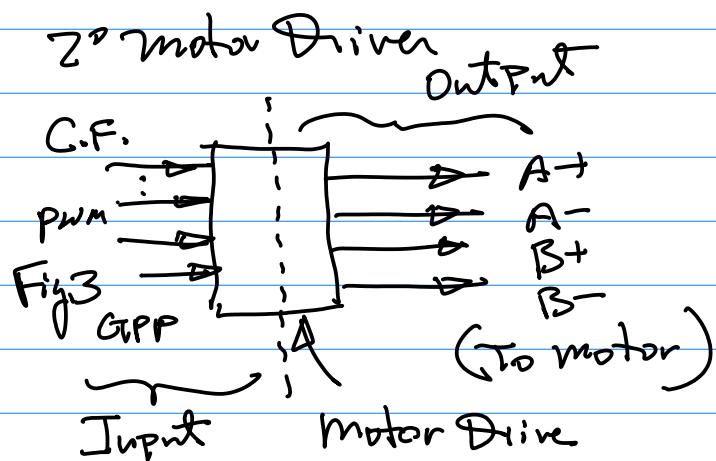
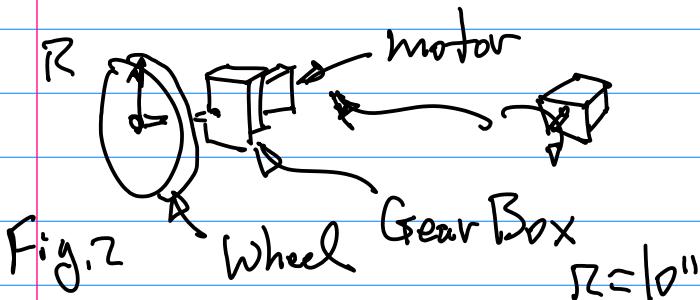
1° Stepper motor

Fig 1



$\frac{360}{200} = 1.8$ (step), 200 steps
 $\frac{1}{2}$ One step 1.8 Degree
 Half step 0.9 Degree
 $\frac{1}{4}$ 0.45 Degree
 $\frac{1}{8}$ 0.225 Degree

Example: Robotics/Self Driving.



Configuration: $K = \frac{2}{3}$
 \Rightarrow pins $K (2^k)$, $2^2 = 4$
 \Rightarrow $Z^2 = 4$
 \Rightarrow $Z^3 = 8$

PWM:

Pulse Width Modulation:

Target Platform: Pi e / NANO / TX2
 TIGSC-V
 Hardware: Pins

Software: Device Driver

Controls the speed of the motor

 $f_{PWM} \rightarrow$ Change Speed;

$f_{PWM} = 1 \text{ kHz}$, Change Only
 Cycle \rightarrow Speed

Note: CAD Design Tool

Eagle Linux & Windows

ORCAD (Cadence) → Eval Version
Free 30 Days
Link

Schematics →
Registed 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: P.I.D. Implementation.

1. Error Signal $e(t)$, From Fig. 1 P.P.17 March 8.

Find Error of Lsm303 Direction Ref: Github on PID

OptiL (self Driving
Encoder Robot)
Displacement/
Wheel travelled

Example: Continued from the
Last Example on this
page.

Suppose the error is denoted as

$e(t) \rightarrow e(k)$

K index for

Continuous Discrete time
System System

Sampling process

Time Index	Error $e(k)$	Notes	Example
K_0	0.1		
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 2nd (mandatory)

CANVAS will be posting Project 1

Ref: Github on PID

Example: Continued from the

Last Example on this

page.

Note 1. Compute Derivative

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

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

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

Δt Sample time interval
 N request Sampling Frequency

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

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

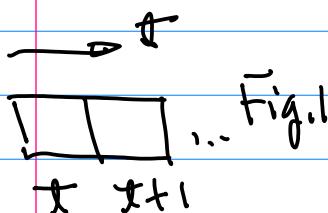
$\rightarrow 1, 2, \dots, k, \dots$

Time Index

$$\begin{aligned} \text{Eqn}(2) \quad \Delta e &= e(t+\Delta t) - e(t) \\ &= e(t+1) - e(t) \\ &\dots (4) \end{aligned}$$

Visualized way of Eqn(4)

Table ("Kernel") Convolution



From Eqn(1) & (4)

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

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

Map Eqn(5) to the Kernel

$$e(t+1) - e(t) = 1 * e(t+1) + (-1) * e(t) \quad C/C++ \text{ using double for-loop.}$$

$t \quad t+1$



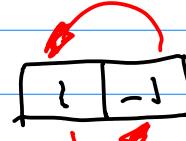
Kernel (b)

Using (b) Kernel to Compute Derivative \rightarrow Python to implement this as convolution

$$k=0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

$$0.1 \quad 1.1 \quad 1.5 \quad 0.5 \quad -0.1 \quad -2$$

$$\frac{de}{dt} \rightarrow -1.0 \quad -0.4 \quad 1.0 \quad \dots$$



$$\begin{aligned} &1 * 0.1 + (-1) * 1.1 \\ &= 0.1 - 1.1 \\ &= -1.0 \end{aligned} \quad \left. \begin{aligned} &-1 * 0.1 + 1 * 1.1 \\ &= 1.0 \end{aligned} \right\}$$



$$\begin{aligned} &1 * 1.1 + (-1) * 1.5 \\ &= -0.4 \end{aligned} \quad \left. \begin{aligned} &-1.1 + 1.5 \\ &= 0.4 \end{aligned} \right\}$$



$$\begin{aligned} &1 * 1.5 + (-1) * 0.5 \\ &= 1.0 \end{aligned} \quad \left. \begin{aligned} &-1.5 + 0.5 \\ &= -1 \end{aligned} \right\}$$

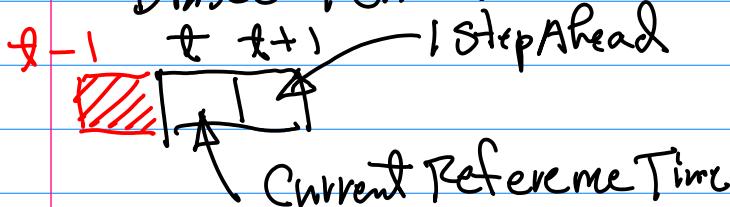
Note: The Above Graphical Way Evaluating Derivatives is based on Convolution, And Can be implemented with

C/C++ using double for-loop.

Action!: Write C/C++,

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)

$$\frac{de(t)}{dt} = \lim_{\Delta t \rightarrow 0} \frac{\Delta e}{\Delta t} \quad \left\{ \begin{array}{l} \Delta e = e(t+1) - e(t) \\ \Delta e = e(t) - e(t-1) \end{array} \right. \quad \begin{array}{l} \text{"t+1" forward} \\ \text{Backward Difference} \end{array}$$

... (6) ... (6b)

$$(F.D + B.D) \frac{1}{2} = \frac{1}{2} (e(t+1) + e(t) - e(t) - e(t-1)) = \frac{1}{2} (e(t+1) - e(t-1)) \quad \dots (7)$$

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.

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

Gaussian Function

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

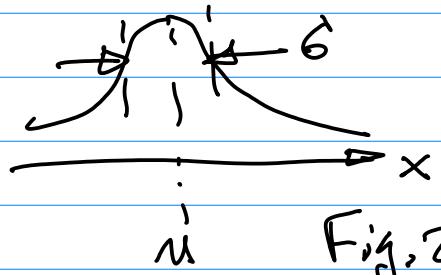
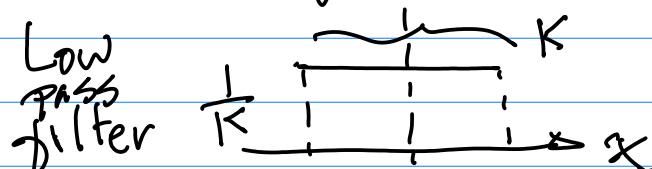
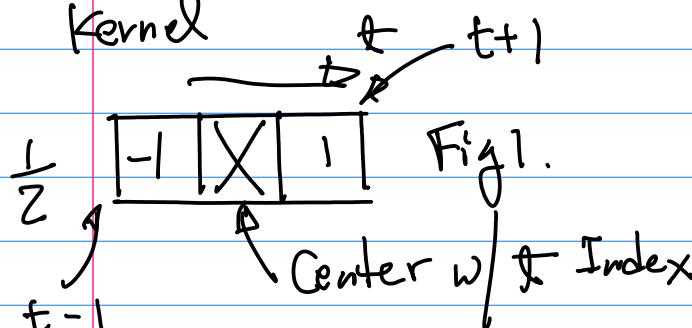


Fig. 2

Low Pass Filter
"Weighted"

Kernel



Action: Write $c(t)$, python to do this

Exercise: Compute $\frac{de(t)}{dt}$ in the table | Convolution with C.D., F.D., B.D.

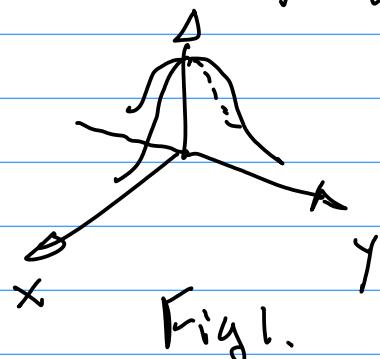
CmPE242

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \dots (1)$$

March 10 (Wed) $\frac{(x^2+y^2)}{26^2}$

$$G(x,y) = \frac{1}{2\pi 6} e^{-\frac{(x^2+y^2)}{26^2}} \dots (2)$$

Note $M_x = M_y = 0$, $G_x = G_y = 0$



Note: $\text{LoG}(x)$ is NOT ²²

Exactly the Computation
for derivatives, But we
use it, for its Low Pass
feature, and 2nd order
derivatives.

Sol.

(1) "Mapping" to a kernel
Build a kernel with
"Odd" Number of grids,
elements

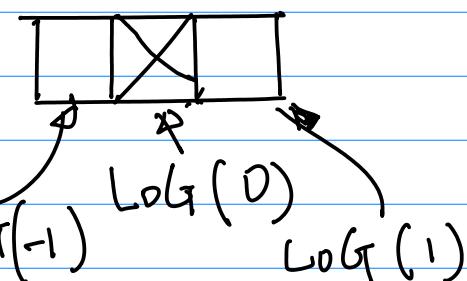
From Eqn(4), Ppt from the github
2018S-15-LecB-V3 ...

Let $y=0$, to have one indep.
Variable x .

$$\nabla^2 G(x,y) \Big|_{y=0} = \nabla^2 G(x,0) \dots (3)$$

$$-\frac{1}{2\pi 6^3} e^{-\frac{x^2}{26^2}} + \frac{x^2}{2\pi 6^5} e^{-\frac{x^2}{26^2}}$$

$K \times 1$
No. of elements One Row
for $K=3$



$\text{LoG}(x)$, or $\nabla^2 G(x,0)$

Example: ① Use $\text{LoG}(x)$ to Build
a convolutional Kernel (z) to
Compute Derivatives of the Error

\cong Identify the
Center Reference
 \Leftarrow from $\text{LoG}(x)$ (or
 $\nabla^2 G(x,0)$). map it
to the kernel

$$\text{Log}(0), \text{Log}(1), \text{Log}(-1)$$

Note: From Eqn(3)

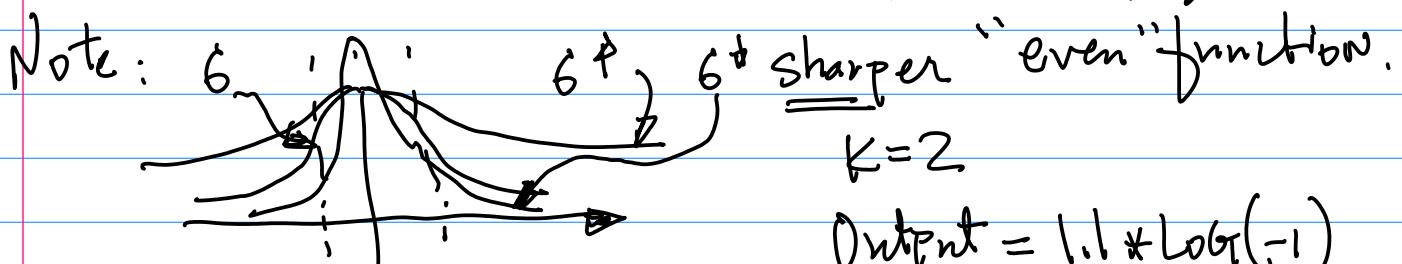
$$\begin{aligned}\text{Log}(0) &= -\frac{1}{\sqrt{2\pi}} 6^3 \cdot 1 + 0 \\ &= -\frac{1}{\sqrt{2\pi}} 6^3\end{aligned}$$

$$\text{Log}(1), \text{Log}(-1).$$

Exercise Dff-Line : Evaluate
 $\text{Log}(1)$, and $\text{Log}(-1)$.

$$\text{Log}(1) = -\frac{1}{\sqrt{2\pi}} 6^3 e^{-\frac{1}{2} 6^2} + \frac{1}{\sqrt{2\pi}} 6^5 \cdot e^{-\frac{1}{2} 6^2} + 1.5 * \text{Log}(1)$$

$$\text{Note: } \text{Log}(x) = \text{Log}(-x)$$



$$K=2$$

$$\text{Output} = 1.1 * \text{Log}(-1)$$

$$+ 1.5 * \text{Log}(0) + 0.5 * \text{Log}(1)$$

Low Pass Filter more, 6^{\ddagger}
 Low Pass Filter less, 6^{\dagger}

(6 in the range ($2.0 \sim 7$ and beyond))
 Depending on the
 physical system.

Note: Use this
 convolution output
 in your derivative
 controller design
 if required

Example (Homework "Off-line")

Use $\text{Log}(x)$ Kernel to perform
 Convolution.

$$K=3$$

$$\begin{aligned}1.5 * \text{Log}(-1) + (0.5) * \text{Log}(0) \\ + (0.1) * \text{Log}(1)\end{aligned}$$

$$\begin{array}{ccccccc}K=0 & 1 & 2 & 3 & 4 & 5 \\ 0,1 & 1.1 & 1.5 & 0.5 & -0.1 & -2\end{array}$$

$$\begin{array}{c} \text{Log}_1, \text{Log}_0, \text{Log}_1 \\ \uparrow \\ \text{Log}(0) \end{array}$$

Convolution output

$$K=1.$$

$$\begin{aligned}\text{Output} &= 0.1 * \text{Log}(-1) \\ &+ 1.1 * \text{Log}(0)\end{aligned}$$

$$+ 1.5 * \text{Log}(1)$$

$$K=4$$

$$\text{Output} = 0.5 * \text{Log}(-1) + (-0.1) * \text{Log}(0) \\ + (-2) * \text{Log}(1)$$

Suppose T (Time), 2 pts
Example: Compute Integral Controller

$$K=0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

$$0,1 \quad 1.1 \quad 1.5 \quad 0.5 \quad -0.1 \quad -2$$

$$\text{Out}(1) \quad \text{Out}(2) \quad \text{Out}(3) \quad \text{Out}(4) \quad \dots$$

$$K=0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

$$0,1 \quad 1.1 \quad 1.5 \quad 0.5 \quad -0.1 \quad -2$$

$$K=1$$

$$\text{Integral}$$

$$\text{Output}(1) = 0.1^2 + (1.1)^2$$

To Design Derivative Controller

$$K=2$$

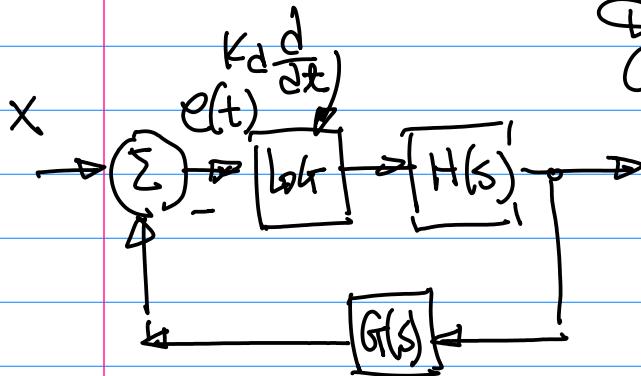
$$\text{Output}(2) = (1.1)^2 + (1.5)^2$$

$$K=3$$

$$\text{Output}(3) = (1.5)^2 + (0.5)^2$$

$$K=4$$

$$\text{Output}(4) = (0.5)^2 + (-0.1)^2 \geq 0$$



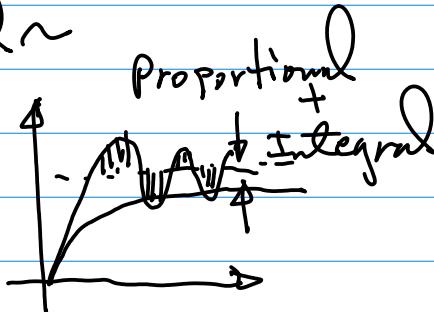
Consider integration controller

To prevent from error cancellation

$$\int_T^{\infty} K_I e(t) dt \dots (4)$$

History Accumulative Error Integral ~

$$\int_T^{\infty} K_I e^2(t) dt$$



T: Time Interval

Discrete Time

$$\int_T^{\infty} K_I e^2(t) dt = \sum_{h=K_0}^{K_T} K_I e^2(h) = K_I \sum_{h=K_0}^{K_T} e^2(h) \dots (5)$$

$$K_0, K_0+1, K_0+2, \dots, K_T$$