CmpE242

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

March 10 (Wed)

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad \ldots (2)$$

Note $\mu_x = \mu_y = 0$, $\sigma_x = \sigma_y = \sigma$

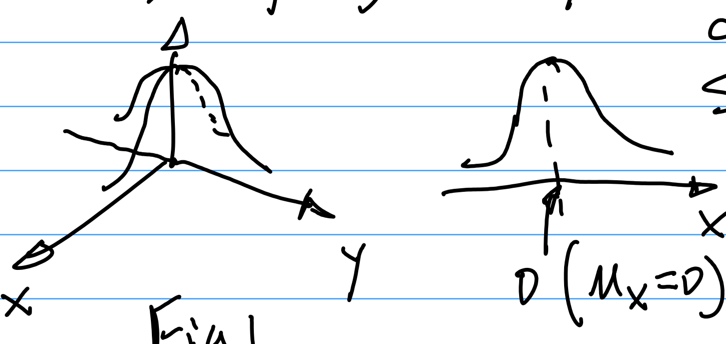


Fig 1.

From Eqn (4), Plot from the github
2018S-15-Lec6-V3 ...

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

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

$$= -\frac{1}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}} + \frac{x^2}{\sqrt{2\pi}\sigma^5} e^{-\frac{x^2}{2\sigma^2}}$$

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

Example: (1) Use LoG(x) to Build a convolutional kernel (2) to Compute Derivatives of the Error

Note: LoG(x) is NOT Exactly the Computation for derivatives, But we use it, for its Low pass feature, and 2nd order derivative.

Sol.

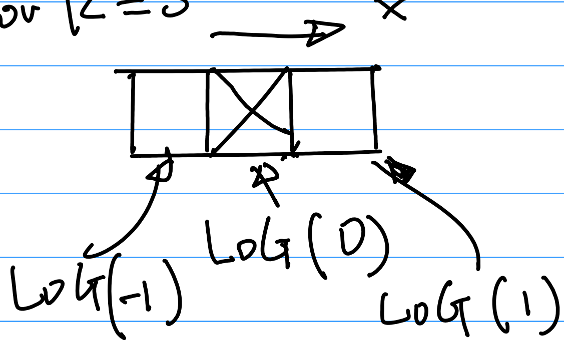
(1) "Mapping" to a Kernel

Build a kernel with "odd" Number of grids, elements

K × 1

No. of elements — One Row

for K = 3



a. Identify the Center Reference

b. from LoG(x) (or $\nabla^2 G(x,0)$), map it to the Kernel

Solve for y ...

Driver Implementation $\begin{cases} f_{PWM} \\ D.C. \end{cases}$

2018S-10-0 ~
PWM Driver

Add Duty Cycle Function to Device Driver.

$\begin{cases} \text{Theoretical Aspect} \\ \text{Implementation C Code.} \end{cases}$

SPRs $\begin{cases} \text{TCON} \leftarrow \text{Control Reg.} \\ \text{TCNT} \leftarrow \text{Counter Reg.} \\ \text{TCMP} \leftarrow \text{Compare Reg.} \end{cases}$

Root

Prefix "Timer"

CFG0
CFG1

PWM Output Square Waveform

T

$\begin{cases} \text{TCNTB} \\ \text{TCmpB} \end{cases}$

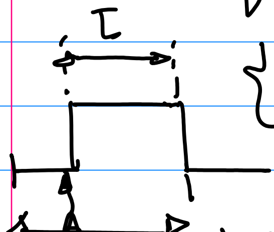


Fig. 1

$T_{PWM} = 1/f_{PWM}$

$$f_{PWM} = \frac{CLK_P}{(\text{Prescaler}+1)(\text{divider})} \quad \cdots (3)$$

PP 1118, CPU Datasheet

Prescaler: 8 bit, [0, 255]

Divider: 1, 2, 4, 8, 16

Note CFG/CON are responsible for Setting Prescaler/Divider value.

$$f_{PWM} = \frac{50\times10^6}{(\text{Pres}+1)\cdot\text{Div.}} \quad \cdots (*)$$

If we need $f_{PWM} = 2\times10^3$

Find SPR, Set SPR. to Realize this frequency.

From CPU Datasheet CFG0 CON.

Sol

$$2\times10^3 = \frac{50\times10^6}{(\text{Pres}+1)\text{Div}}$$

$$(\text{Pres}+1)\text{Div} = \frac{50\times10^6}{2\times10^3}$$

$$(\text{Pres}+1)\text{Div.} = 25\times10^3$$

Let Div = 16,
Solve for Pres.

$$\text{Pres}+1 = \frac{25\times10^3}{16}$$

$$\text{Pres} = \frac{25\times10^3 - 16}{16} \simeq > 255$$

Iteration,

Change PCLK to 10MHz,
then, we have

CMPE/EE242

$$Pres = \frac{10\times10^6 - 16\times2\times10^3}{16\times2\times10^3}$$ if it is still too big

therefore, then Low the CLK_p

try $CLK_p \simeq 2\times10^6$. please verify it!

Note: SPR Responsible for f_{PWM}

T CNT B

Timer, Root, Buffer

$f_{PWM} = 1\times10^3$ given.

f ~~master~~ Clock peripherial

N Counts for CNT SPR.

$$f = f_{PWM} \cdot N \quad \ldots (4)$$

Given, Target, Unknown to be Calculated

Note: T Cmp B

Comparison Register for Duty Cycle

2nd Counts Value for "Cmp" Derived from Duty Cycle.

March 17 (Wed)

{ f_{PWM} By Setting SPR's value
{ Duty Cycle

Arm11 , ~~Datasheet~~

{ TCNTB ← "N" Counts ↕ f_{PWM}
{ TCmpB

20185-10-

$$f_{PCK}/f_{PWM} = N \quad \ldots (1)$$

~~Define~~ ~~one~~ the period; CNT

Duty Cycle → % → Counts
Percentage ↓ Cmp

GPFCON PP. 522

GPFCON[29:28] = 10 → PWM

GPFCON[31:30] = 10 → PWM

#define S3C64xx_GPFCON

0x7...

*.h

&= ~(0x3u << 28)

"AND" "Neg." "00" "11"

|= (0x2u << 28)

"OR" "10" unsigned

Set 2 Bits

GPFCON[29:28] = 10

March 22nd (Mon)
Review.

1° 3± Questions.

a Basic Concepts CPU Architecture, Block Diagram,
Memory Map, Peripheral Controllers
GPP (GPIO) } SPRs CON, DAT
PWM
PWM TCNTB, TCMPB, CFG0, 1

Architecture → Mem → SPR
Code

{ User Space Programming
{ Kernel Space Device Driver

Kconf Script ←

Define Compilation + Build Process.

Programming Requirements, No Program Code Writing
However! Debug/Change the existing Code is Needed;

b Design-Related Question(s)

SCH. Design, CKT for PWM
Pin(s), f_{PWM} — GPIO

Motor Drive
Pin(s), Label(s) Stepper motor I/F

GPP I/O Testing ("Hello, the World")

{ Input Testing CKT
{ Output Testing CKT

Resistance Value Calculation

c Theoretical Aspects

C1. PID Controller Design
Basic Concepts
Block Diagram.

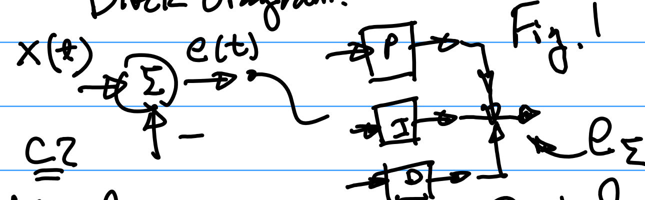


Fig. 1

C2 Kernels
F.D. 2x1 → Central D. $\frac{1}{2}$(F.D+B.D)
B.D. 2x1

With Noise Reduction 3x1

Low Pass Filter: G(x) Gaussian.

+ 2nd Order Derivation as in Computer Vision

∇^2 : Laplacian $\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2} \rightarrow$ 1D $\frac{\partial^2}{\partial x}$

LoG(x)

Note: One page formula sheet is Allowed, However No verbal Description And/Or Examples Allowed.

Note: Calculator Is Allowed.

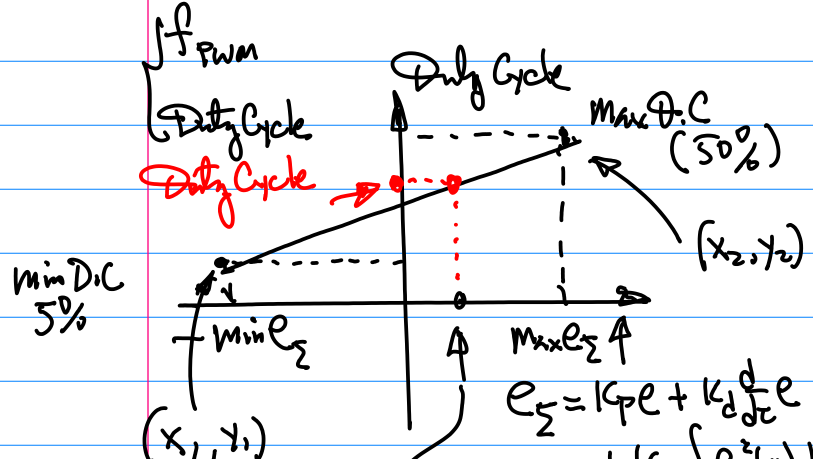
Close Book, Close Notes
Datasheets if needed will be provided;

Convolution with Kernel(s)
Table of E(t), find $\frac{d}{dt}E(t)$ Convolution

$$\int K_I e^2(\tau)d\tau \approx \sum_{i=0}^{I} K_I e^2(\tau)$$

Mapping to Control function PWM

$\begin{cases} f_{PWM} \\ \text{Duty Cycle} \end{cases}$



$$e_\Sigma = K_p e + K_d \frac{d}{dt} e + K_I \int_I e^2(\tau)d\tau$$

$e_\Sigma = 0.57$

find Control Action, → D.C = ?

Implementation: Reference Platform ARM11

Architecture CPU Block Diagram → memory map.

2^{32} (4 GB)

8 Equal Bank ↓ 3 Addr. Bits

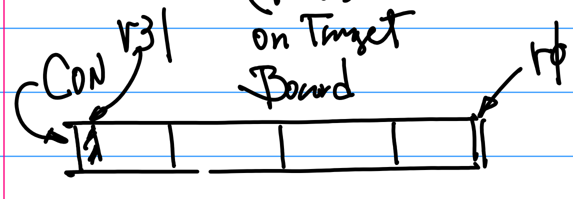
$a_{31} a_{30} a_{29}$

* SPRs. (Peripheral Controllers) ←

$\begin{cases} \text{PWM} \\ \text{GPP} \end{cases}$

GPx CON
GPx DAT
T CNT B
T Cmp B

Design Spec. → SPRs → CPU Datasheet (Pins) on Target Board



To perform init & Config:
1° Binary Pattern for SPR.

Read/modify user Application Programs | Kernel Space Device Driver Program.

C Code for this purpose.

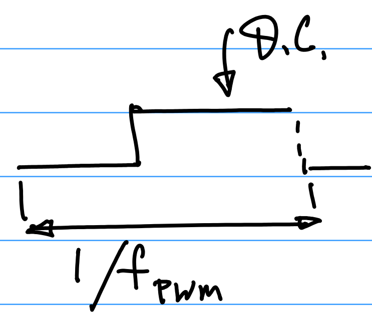
Note: PWM Waveform, e.g. Duty Cycle Calculation.

N. for CNT
N' for Cmp
T B

$f_{PCLK} / f_{PWM} = N$

$$f_{PWM} = \frac{PCLK}{(\text{Prest}+1) * DVR}$$

%(D.C) * N = N' → Cmp



Pre-requisit $\begin{cases} 1° \text{ O.S Source Distribution} \\ 2° \text{ Tool Chain Distro. "Cross Comp"} \\ 3° \text{ Datasheet.} \end{cases}$

↓

Tool Chain Installed Running

make menuconfig

Continued → Kconf (at \drivers ~\char)

↓

Script. Add your Device Driver

↓

make menuconfig

↓

invoke your change, Compile & Build (Module Only for Simplicity Purpose)

↓

Object "KO".

↓

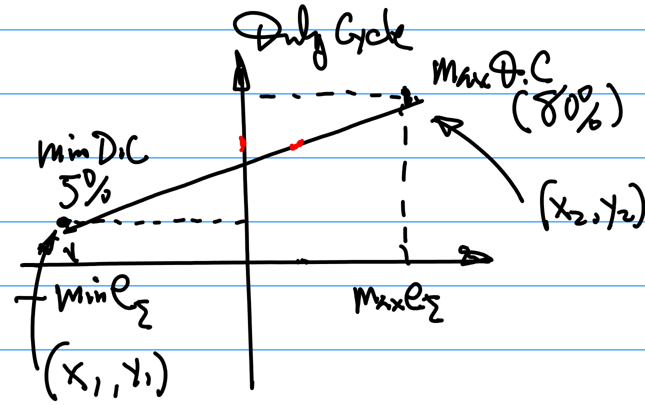
Copy "USB", upload by "CP" Copy Command to your target

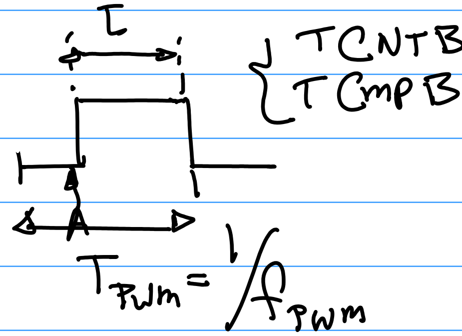
↓

"insmod" mytest.KO (To make it as a part of Kernel Image)

↓

run your user application program (By calling the module)





https://github.com/hualili/CMPE242-Embedded-Systems-/blob/master/2018S-16-AngularSensing-i2c-L8

CMPE/EE242

April 5th (Monday)

1. midterm Graded, the key was posted online, github, search under folder 2019S, "key"

2. 2nd half of this Course. IIoT (Industrial IoT)

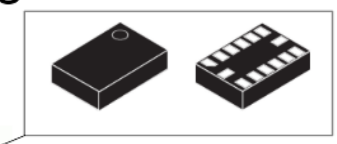
Sensors I/F { Digital Sensors — I2C I/F. / Analog Sensors — ADC }

FFT to find/Characterize Analog Sensor Data (Nyquist Theorem) 32
Validation of Sensor Data.
OpAmps to Build Processing CKT. "SPICE" Simulation.

CMPE242-Embedded-Systems- / 2018S-16-AngularSensing-i2c-LSM303- final HL 2017-3-13.pdf

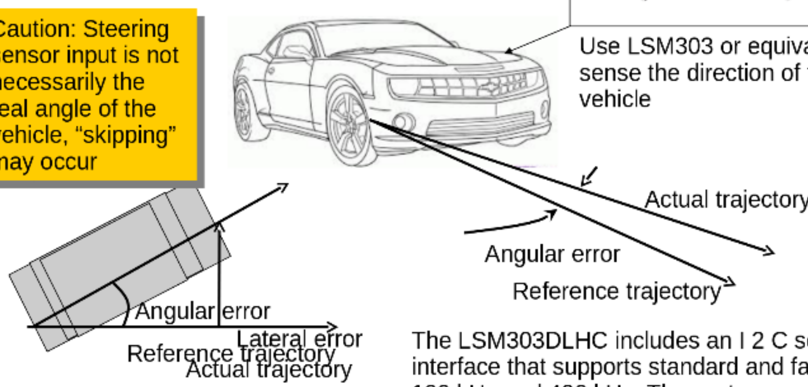
Sensors for Driving Direction and Turning Angle

eCompass module:
3D accelerometer and 3D magnetometer



Use LSM303 or equivalent to sense the direction of the vehicle

Caution: Steering sensor input is not necessarily the real angle of the vehicle, "skipping" may occur



The LSM303DLHC includes an I 2 C serial bus interface that supports standard and fast mode 100 kHz and 400 kHz. The system can be configured to generate interrupt signals by inertial wake-up/free-fall events as well as by the position of the device itself.

Harry Li, Ph.D. April 2015

Example: LSM303

Note: Next Project use LSM303.

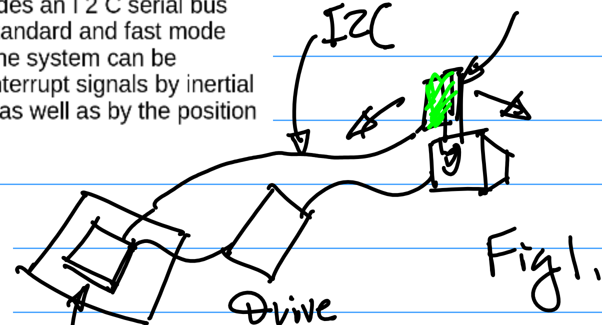
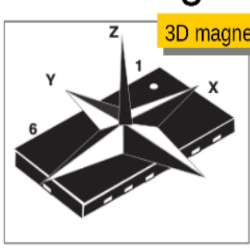


Fig 1. Configuration I for PID controller

3D Accelerometer and 3D Magnetomete LMS303



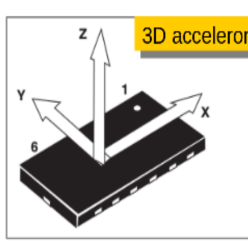


Table 9

Pin name	Pin description
SCL	I^2C serial clock (SCL)
SDA	I^2C serial data (SDA)

I2C Interface

(1) The transaction started through a START (ST) signal, defined as a high-to-low on the data line while the SCL line is held high.
(2) After ST, the next byte contains the slave address (the first 7 bit), bit 8 for if the master is receiving or transmitting data.

Homework: Implementation I2C LSM303 Sensor I/F. Due April 16 (Fri)

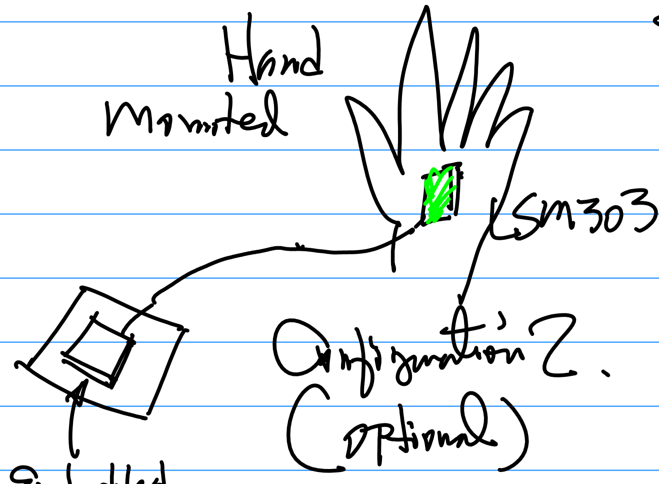


Fig 2.

3. a Space-Time Diagram

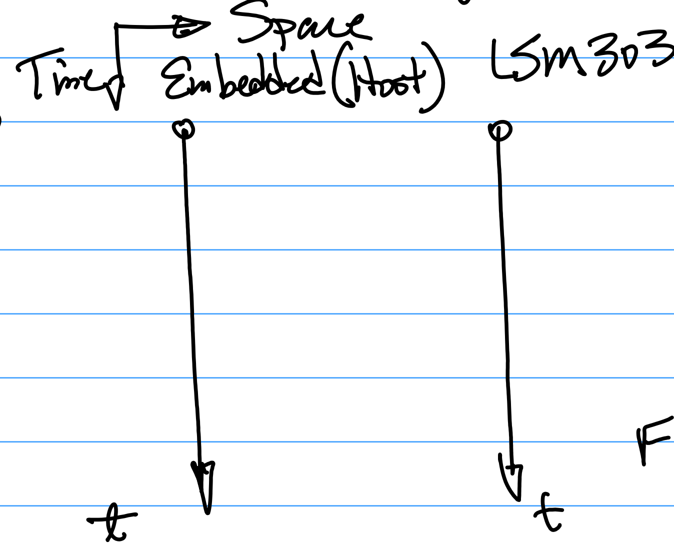


Fig 3

Submission On CANVAS

Objective ① To be Able to Read Sensor Input,
② To be Able Config the Sensor.

b To Describe "Hand-Shaking"

Three Small Steps.

Step 1. → Step 2 → Step 3

Step 1: Host Command. to the Target via Address for Init & Config

Step 2: Slave "Ack"

Step 3: Data Transmission will start

1. Note: LSM303 for ST-micro

Sensor Supports { Acceleration X-Y-Z axis, Magnetometer, Temperature

2. I2C

{ SDA (Serial Data) Bidirectional Data;
SCL (Serial clock)

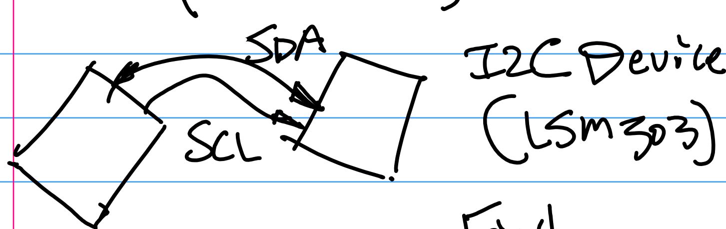


Fig 4.

* Be sure read Datasheet to map the Steps of the I/F to Space-Time Diagram.

3. Datasheet Table 9 & 11. Pp. 20.

Notation:

A Frame

1° ST → SP
"Start" "Stop"

2° The Notations in Table 11
SAD, SAD+W, SUB, DATA,
SAK, etc. pp. 20

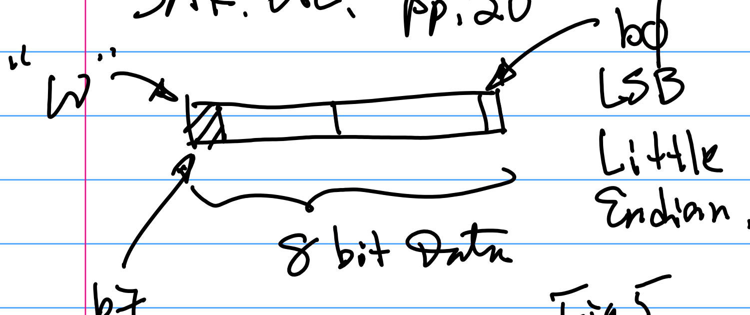
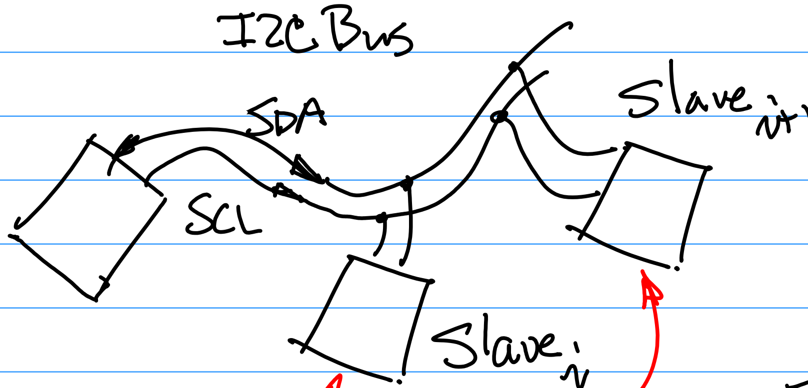


Fig 5

3° from pp. 20 (Datasheet)



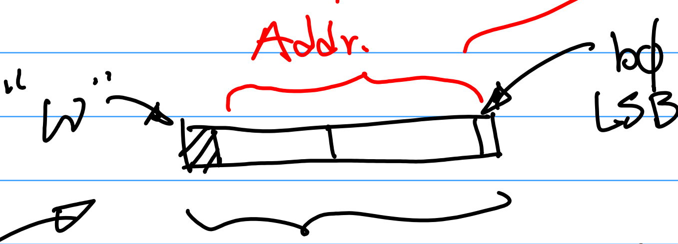
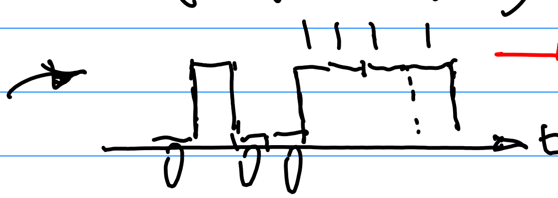


Fig 6

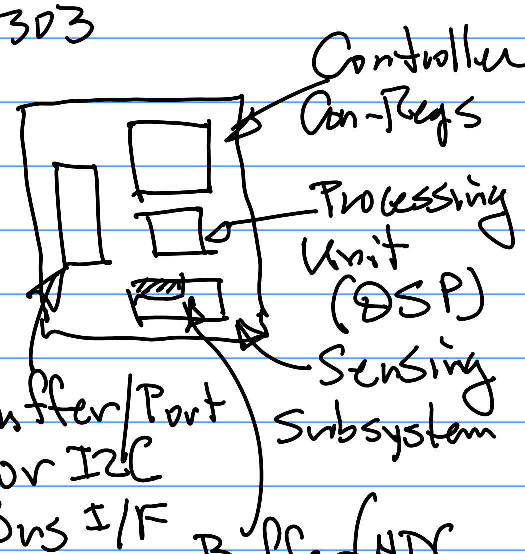
4. 1st Address (7 bits), 2nd Address
lower SUB.
All from the host (Target platform)

0xf2
1111 : 0010



Consider A slave device
LSM303

Fig 7



2nd Address "SUB"
is for Identifying the
target inside the
Slave Device.

5. 127 Devices Possible
(Theoretically) on
I2C Bus, In Reality
this has to be Checked
By "FAN-IN" "FAN-OUT"

128 Internal Addresses
→ Special Purpose
Registers.

6. Most Significant Bit is
transmitted first

Example: From Datasheet (LSM303)
PP.19 Table 11, 12, 13

Homework (1pt) Due A week from Today, April 14, Due On CANVAS

1° Build I2C Bus Interface with your target platform as a host, LSM303 Slave.
To be able:

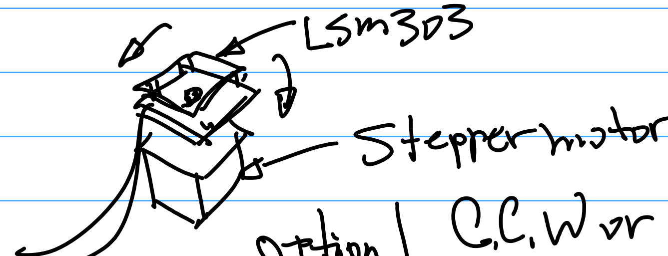
a Hardware Implementation. (e.g. mount LSM303 on the Stepper motor, OR mount it to your hand)

b Read Acceleration Data X, Y, Z., Display it on your terminal.

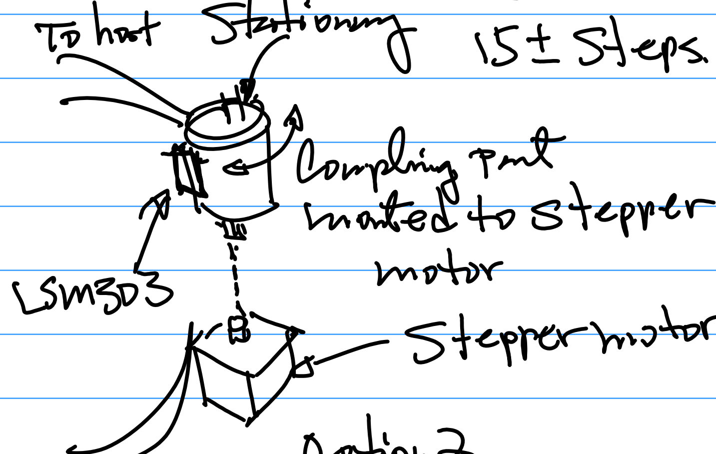
c Read Magnetometer Data and display it on the terminal

Note: Sample code is posted "as is" basis.
Repo: 20-2021S-10-lsm

2° Submission
a Source Code, b Readme.txt
c photo(s) of your Implementation
3x photos, 1 for the entire System (with Laptop); 2nd for the Host Side, Expansion Connector is the focus; 3rd for LSM303



Option 1. C.C.W or CW. 15± steps.



Option 2.

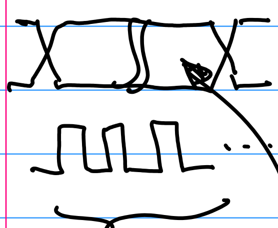
d 5 seconds Video Clip(s) 720P or 1080P (1920x1280) Compressed, mpeg4, .avi ?

file Naming: firstName_4Digits_242.zip

Note: Table 11 & 12 (PP19)

OneByte Writing → Multiple Bytes Writing
Host → Host

Table (Logical Behavior)
↓
Timing (Waveform)



0xf2

8 bits

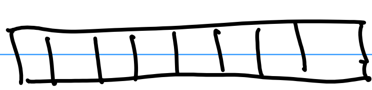
Now, the Address for the Sensor(s) and Addresses for Registers
- Control Register — Init & Config
- Data Register

SAD[7:1] Address + SAD[0] for W/R
- =1 for W
- =0 for R

Note: Use info from Table 14 to fill in SAD+W, SAD+R in tables 11~13.

Note: Section 5.1.3 Magnetometer

Example: Table 18. Control Register A for Magnet CTRL_REG1_A[7:0] 8 bits.



CTRL_REG1_A[7:4]=0x7 (for 400Hz)

TechSpec → Binary Pattern

TechSpec:
i. 400Hz Data Rate
ii. X-Y axis.
iii. Sensor Active (No LowPower)

CTRL_REG1_A[3]=0
CTRL_REG1_A[2]=0
CTRL_REG1_A[1]=1
CTRL_REG1_A[0]=1
→ 0x3

Hence,
CTRL_REG1_A[7:0] = 0x73 ✓

Section 7.1.8 Status Registers

Section 7.1.9~7.1.11 Data Registers.

2's Complement form
↓
1's Complement By Negation "0"→"1" "1"→"0" +1

April 12 (Monday)

Homework Extension to April 19 (Monday)

Industrial Analog Sensor I/F Design

Example: NH3 Analog Sensor (Ion Selective Electrode) Interface.

1. ADC (Analog to Digital Conversion Unit)

LPC1769 6x ADC

ARM11 ADC

Pie Stand Alone ADC

Jetson NANO/Tx2

1° Sampling Rate
500 KSPS

2° Quantization
8 bits, 10 bits.
12 bits.

3° SPI/I2C
Faster ≤ 10MHz
≤ 100MHz

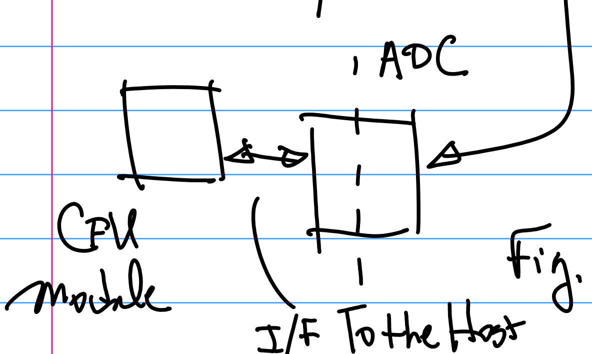


Fig. 1

3. Analog Interface Design.

Start with Characteristic Curve

↓

Linearization

↓

OpAmp Preprocessing CKT

↓

Optimized dynamic (Input) Range to ADC

4. OpAmp's for Preprocessing

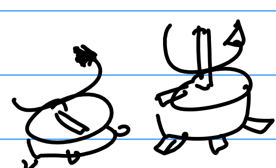
Addition
Subtraction
Division
multiplication

Math. Operations for Preprocessing

Example: from Datasheet 2018S-22-NH3-TP13, Fig. 14.

Step 1. Linearization Characteristic Curve

2. Prototype to Build



Stepper motor Drive

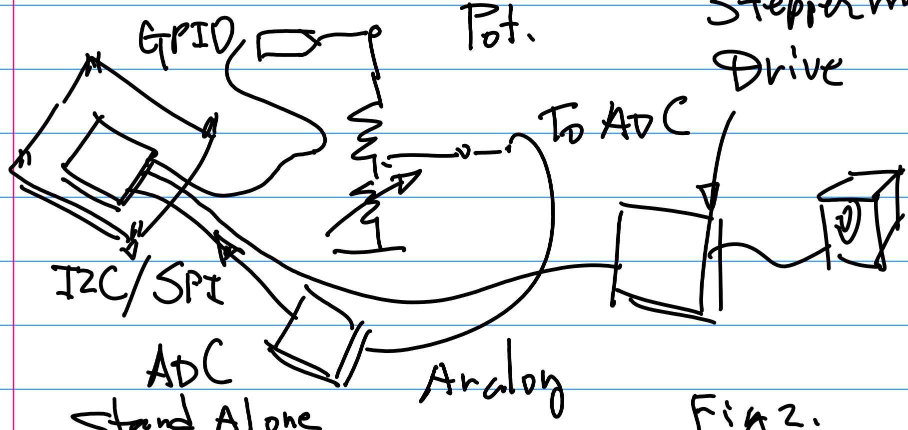
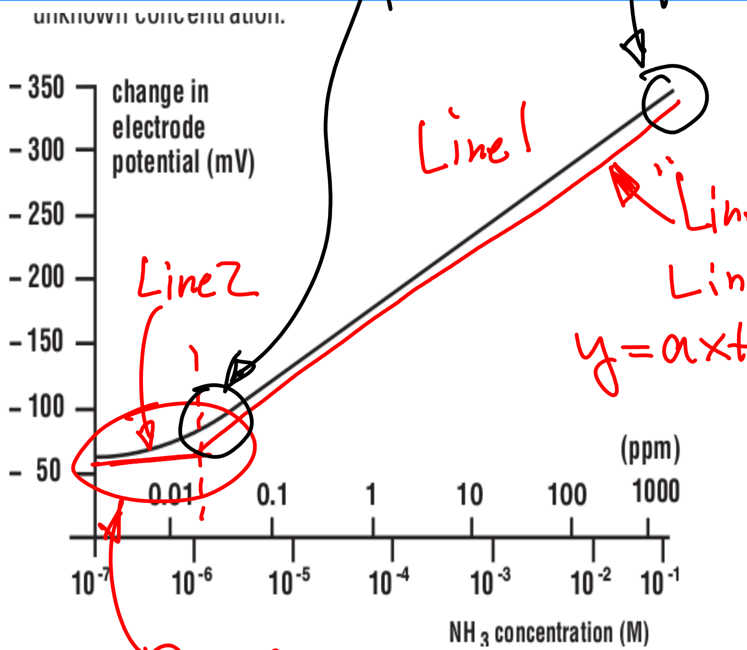


Fig 2.

Starting pt.

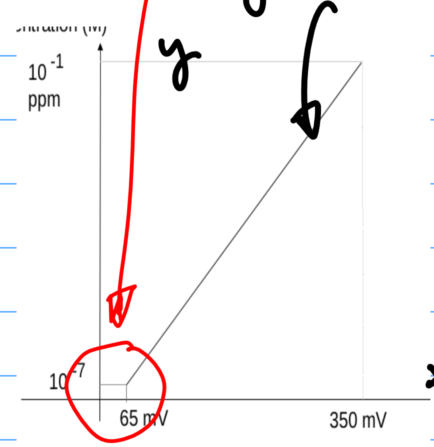
Ending pt.



Non-Linear. $y = ax^2 + bx + c$ 2nd order

Step 2. Map the Dynamic Range of Sensor ~~to~~ Output to the Dynamic Range of ADC Input.

2.1 "Shifting" OR Offset to move the Characteristic Curve to the origin.



Linearization: Line Equation(s) to Replace Non-Linear Curve.

Simplification: Just Keep Line 1.

Note: Linearization By Identifying Starting and Ending Points.

(V_0, C_0) (V_N, C_N)

Voltage v.s. Concentration

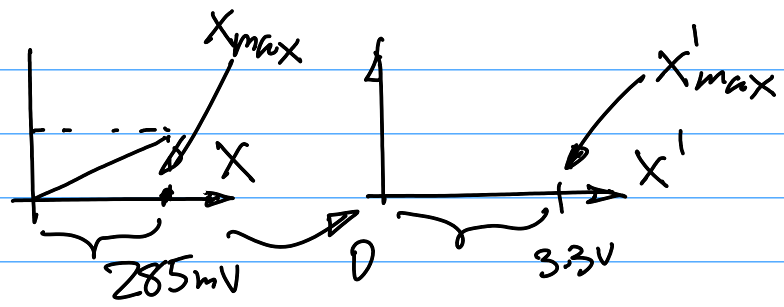
$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \quad \ldots (1)$$

$(x_1, y_1) \to (V_0, C_0)$, $(x_2, y_2) \to (V_N, C_N)$

x_{New} is changed By "Shifting"

$ax \to a(x - \Delta x)$

2.2 Dynamic Range fits to ADC Dynamic Range



$$\frac{X'_{max}}{X_{max}} = \frac{3.3V}{0.285V} \overset{Let}{=} A \quad \cdots (2)$$

Gain
("Scaling Factor")

$$X \cdot A = X' \quad \cdots (3)$$

Review OpAmp { Inverting Configuration OpAmps
Non-Inverting Configuration

April 14 (Th).

OpAmps for Pre-processing Design

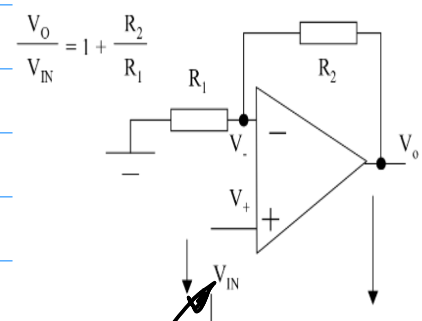
1. OpAmp { a Very High Input Impedence
b Very Big Open-Loop Gain
c Very Small output Resistance

2. Using OpAmps As Basic Building Blocks (B^3) for

Arithmetic Computation { Add / Sub
Multiplication / Division

Integral / Derivatives

3. Configurations { Non-Inverting ~
Inverting ~



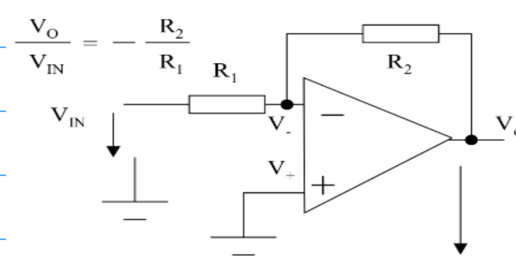
NON-INVERTING

a Input: Positive Polarity;

b feedback CKT $V_{out} \rightarrow$ Input (V_- pin) via R_f

c Draw the CKT.

$$A = V_{out}/V_{in} = 1 + R_f/R_1 \quad \cdots (1)$$



INVERTING

a Input: V_- pin

b feedback CKT $V_o \rightarrow V_-$ (in) Via R_f

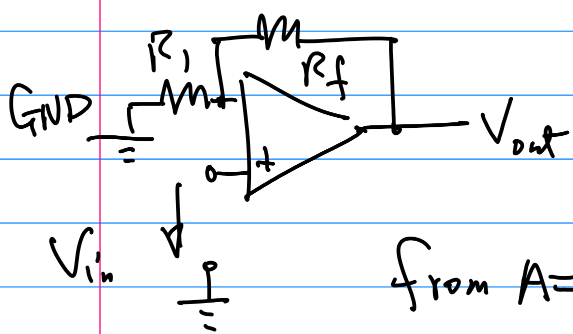
c $A = -\frac{R_f}{R_1} \quad \cdots (2)$

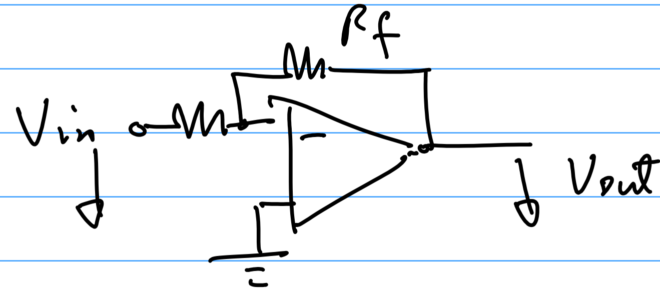
4. Use OpAmp CKT(s) for Math. Operations.

More than One Approach is possible, But Let's use Inverting Configuration

a Addition. $X_1 + X_2$

Non-Inverting Configuration

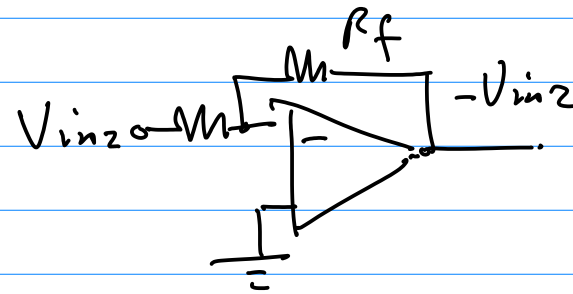




from $A = 1 + \frac{R_f}{R_1}$

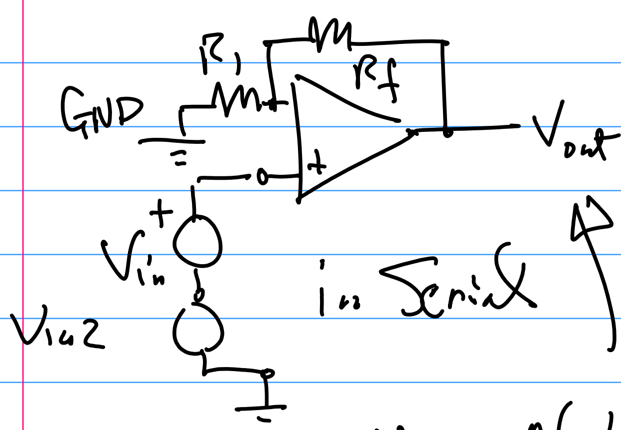
$\frac{V_{out}}{V_{in}} = A = -\frac{R_f}{R_1}$

OR, $A = V_{out}/V_{in}$, $V_{out} = A \cdot V_{in}$ ...(3)



$V_{in1} + V_{in2}$ Addition

Let input Circuit as follows



make $V_{out} = -V_{in2}$, Let $R_f = R_1 = 1K\Omega$

(741 opAmp, 384 Quad-Pack)

Note: Not too Big Current, PWR Consumption is going to be a problem.

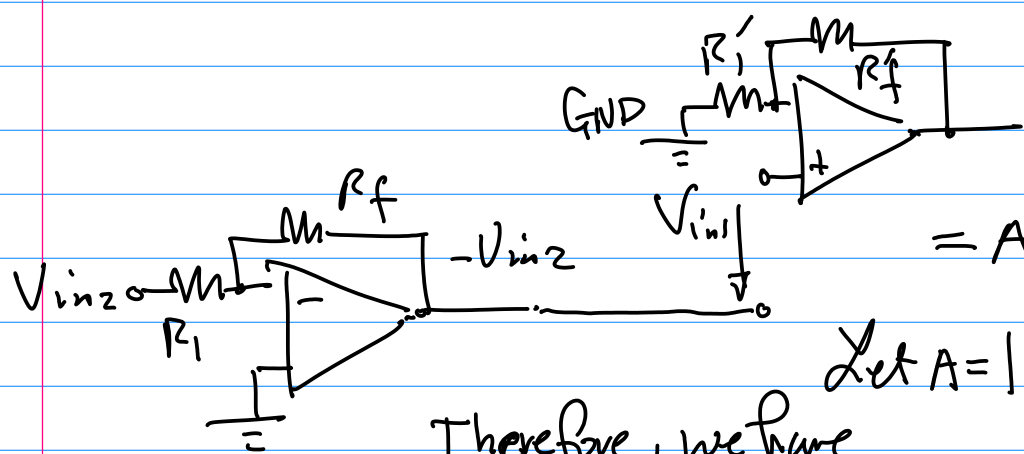
Not too small, Noise will distore the Signal

$V_{out} = A(V_{in1} + V_{in2})$

Let $A = 1$, $\Rightarrow R_f = 0\Omega$

b Subtraction $X_1 - X_2$ $(V_{in1} - V_{in2})$

Then, Combine it with Add CKT
So, we have $V_{in_1} - V_{in_2}$



$V_{out} = A\,V_{in}$

$= A\left(V_{in_1} - V_{in_2}\right)$

Let $A = 1$, Hence $R_f' = 0\Omega$

Therefore, we have

Subtraction

$V_{out} = V_{in_1} - V_{in_2}$

c. multiplication. $y = ax$
$a > 1$

use Non-Inverting Configuration

$\left.\frac{V_{out}}{V_{in}} = A\right|_{A = 1 + \frac{R_f}{R_1}}$

$V_{out} = A \cdot V_{in} = \left(1 + \frac{R_f}{R_1}\right) V_{in}$

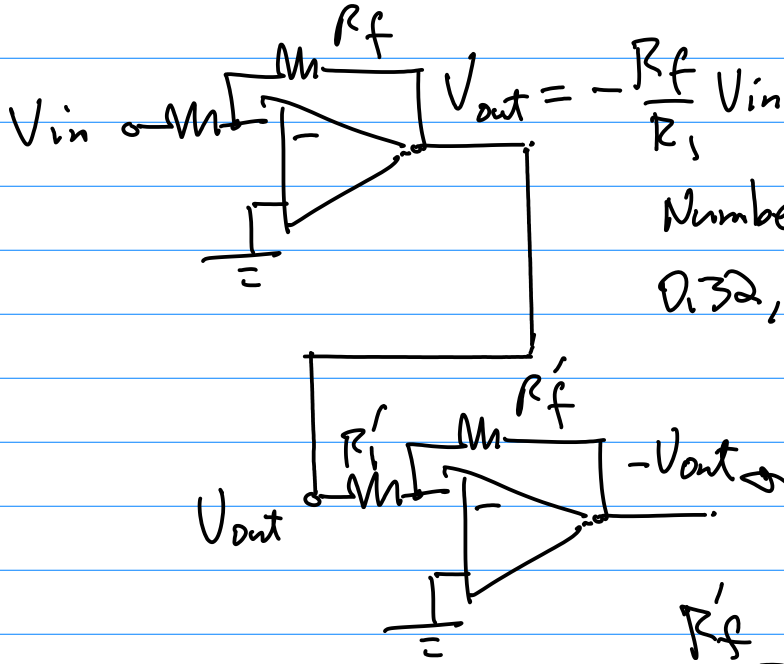
Note: For Linear System, the multiplication is done by multiplying a gain, But Not Another X (or V_{in}).

d. Division (is a multiplication!)

for the multiplier A less than 1.

Two Stage Inverting Configuration, 1st Stage does the division But with negative Sign, 2nd stage with gain = 1, But Change to positive by 2nd negative.

Example: 2 stage Inverting



$V_{out} = -\frac{R_f}{R_1} V_{in}$, where $\frac{R_f}{R_1}$ is a fractional Number for division, for example 0.32, $R_f/R_1 = 0.32$, if $R_1 = 1K$

$R_f = 320\Omega$

to make gain $= -1$

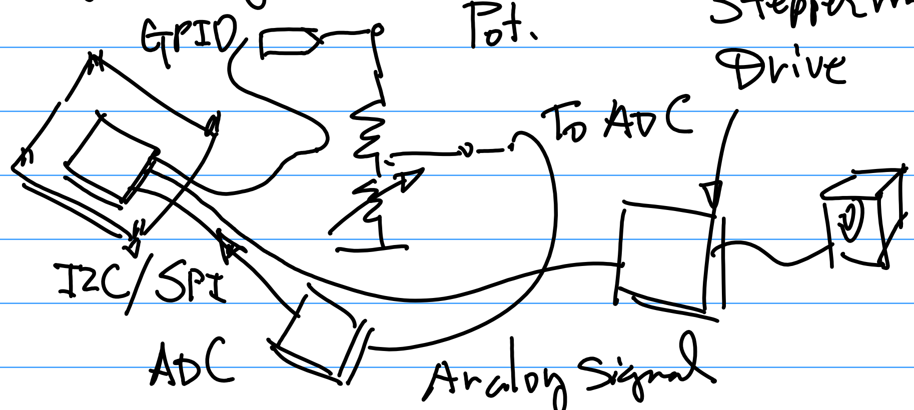
$\frac{R_f'}{R_1'} = 1$, Let $R_1' = 1K\Omega$ then Solve for

$R_f' = R_1' = 1\,K\Omega$.

Note: Analog Sensors have to meet the Output Current Requirement, e.g. 4~20mA

April 19 (Mon).

1° Preparation for the Coming Project. Fig 2, PP. 37.

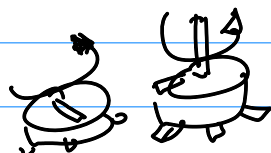


Project Due May 9 (Sunday) 11:59 P.m.

April 19 (Mon)

Design ON Preprocessing CKT.

Example: See Fig. on github



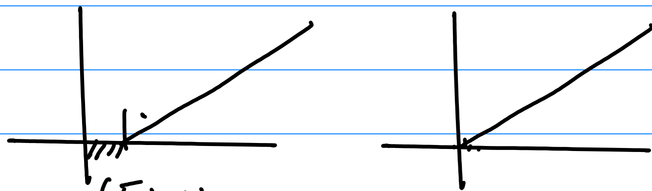
Stepper motor Drive

Characteristic Curve. Shifting → Enlargement of the Dynamic Range.

a Shifting. Add/Sub.

b Dynamic Range : Gain mult.

Shifting (Subtraction).



65mV

Inverting Configuration

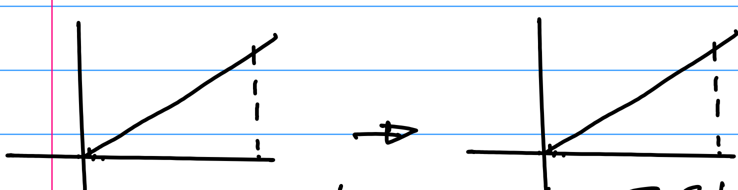
$$\frac{V_o}{V_{in}} = -\frac{R_f}{R_1}$$

Desired Shifting

Output from the Reference (External Voltage Source)

$$\frac{65\times10^{-3}}{5.0} = \frac{R_f}{R_1},\quad \text{Let } R_f = 1K\Omega \quad \cdots (1)$$

Now, Dynamic Range Design



285mV → 3.3V.

$$\text{Gain} = \frac{3.3V}{285mV} = \frac{3.3}{0.285} \simeq 11.58 \quad \cdots (2)$$

After

Before

Since Gain is positive, we have Non-inverting configuration

$A = \left(1 + \frac{R_f}{R_1}\right)$, substitute (2) into this Eqn.

$11.58 = 1 + R_f/R_1$

Let $R_1 = 1K\Omega$, find/solve for R_f. Hence $R_f = 10.58 K\Omega$

Then, Integrate OPAmps Together to form Preprocessing CKT.

Note:

1° Analyze Sensor Characteristic Curve, Define Arithmetic/Math Operations, for Shifting and magnification (of the Gain, to Cover 3.3V Dynamic Range)

2° To be Able to use Inverting And/OR Non-inverting Configuration to Realize the design Requirements, e.g. Shifting, and magnification

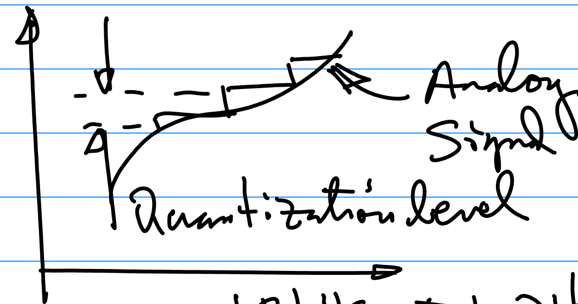
Now, Let's Consider the ADC Design. Scope:

- System Level for ADC
- Data Validation

D.F.T. (Discrete Fourier Transform) → Power Spectrum

↓

Nyquist Sampling Theorem.

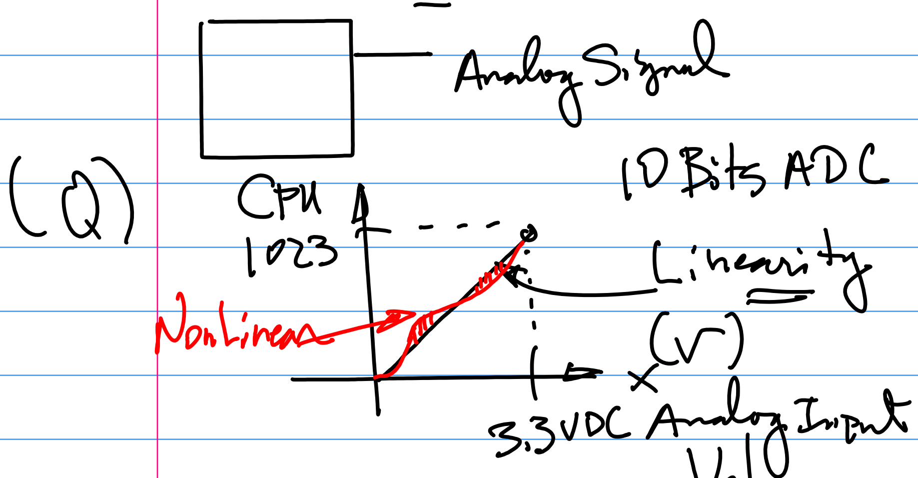


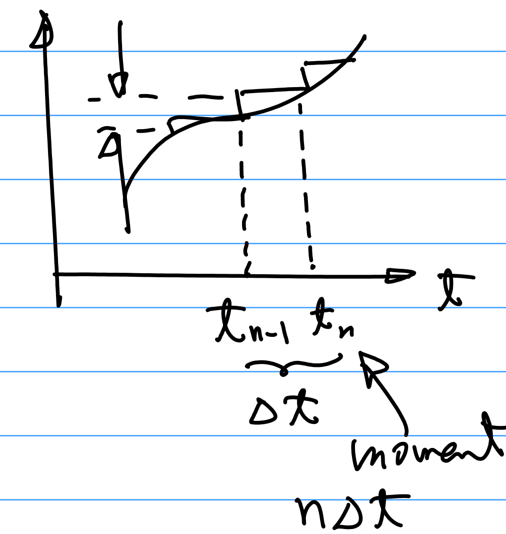
10 bits → 1024 $= 2^{10}$

April 21 (Wed)

Data Validation: To make sure ADC Output (to CPU)

Δt : Sampling interval





$\Delta Q / \Delta V = K$

Nyquist Theorem

$$f_{sampling} \geq 2 f_{max} \quad \cdots (1)$$

Sampling frequency has to be greater than OR Equal to Twice the highest frequency of the Signal itself.

2. Introduce Discrete Fourier Transform
   - a Background : Basic Building Blocks to Characterize or to Build a given Signal.
   - b Formulation.

↓

Fourier Transform

Note:

1. Denote Digital signal as $x(n)$ or $x(k)$

Digitized, limited Quantization level

Index integer $n\Delta t$

$$C_n = \int_T x(\tau) e^{-j2\pi f \tau}\, d\tau \quad \ldots(1)$$

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x-x_0) + \frac{f''(x)}{2!}(x-x_0)^2 + \cdots + R_n(x) \quad \ldots(2)$$

Basic Building Blocks

$$f(x) \cong \sum_{n=-\infty}^{+\infty} C_n \left(\cos 2\pi n f t + \phi\right) \quad \ldots(3)$$

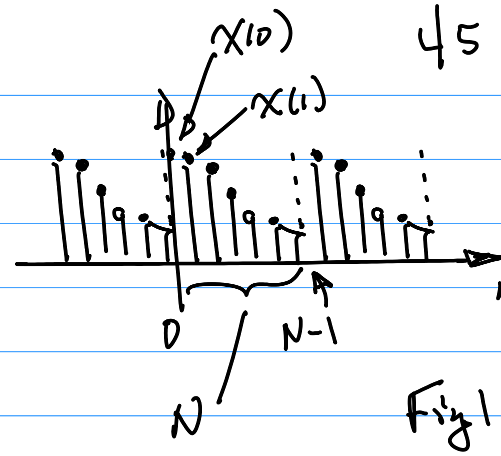


Fig 1.

$n=0,\ n=1,\ n=2,\ \ldots$

$x(0),\ x(1),\ x(2),\ \ldots$

$e^{-j2\pi \frac{mn}{N}}$ for Imaginary Axis

m: Frequency Index

$X(m)\big|_{m=0} = X(0)$ D.C. Component

$X(1), X(2), \ldots X(N-1)$ Higher Freq. Comp.

3. Notation

$X(m)$ Discrete Fourier Transform of a Signal $x(n)$

m: Index in Frequency Domain

Hence, D.F.T. (Discrete Fourier Transform)

$$X(m) = \frac{1}{N}\sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}} \quad \ldots(4)$$

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi \frac{mn}{N}}$$

$$= x(0)e^{-j2\pi \frac{m\cdot 0}{N}} + x(1)e^{-j2\pi \frac{m\cdot 1}{N}} + \cdots + x(N-1)e^{-j2\pi \frac{m(N-1)}{N}}$$

from Right Hand Side of Eqn.(4). (For Simplicity, $\frac{1}{N}$ Removed)

Physical meaning:

$x(n) = x(n+kN)$ periodic function.

Period: N, where $k = 0, 1, 2, \ldots$ (Natural Number)

$\frac{1}{N}$ Scaling factor, N No. of Total pts for One Period.

for $m=0$,

$$X(0) = x(0)e^{-j2\pi\frac{0\cdot0}{N}} + x(1)e^{-j2\pi\frac{0\cdot1}{N}} + \cdots + x(N-1)e^{-j2\pi\frac{0(N-1)}{N}}$$

D.C.

$$= x(0)\cdot1 + x(1)\cdot1 + \cdots + x(N-1)\cdot1$$

$$= \underbrace{x(0)+x(1)+\cdots+x(N-1)}$$

Divided by $1/N$, Average → D.C.

for $m=1$.

$$X(1) = x(0)e^{-j2\pi\frac{1\cdot0}{N}} + x(1)e^{-j2\pi\frac{1\cdot1}{N}} + \cdots + x(N-1)e^{-j2\pi\frac{1(N-1)}{N}}$$

$\vdots$

for $m=2,3,\ldots$

$$X(N-1) = x(0)e^{-j2\pi\frac{(N-1)\cdot0}{N}} + x(1)e^{-j2\pi\frac{(N-1)1}{N}} + \cdots + x(N-1)e^{-j2\pi\frac{(N-1)(N-1)}{N}}$$

D.F.T.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{bmatrix} = \frac{1}{N}\begin{bmatrix} \end{bmatrix}_{N\times N}\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ \vdots \\ x(N-1) \end{bmatrix} \quad \cdots (5)$$