CmpE242

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

March 10 (Wed)

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

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

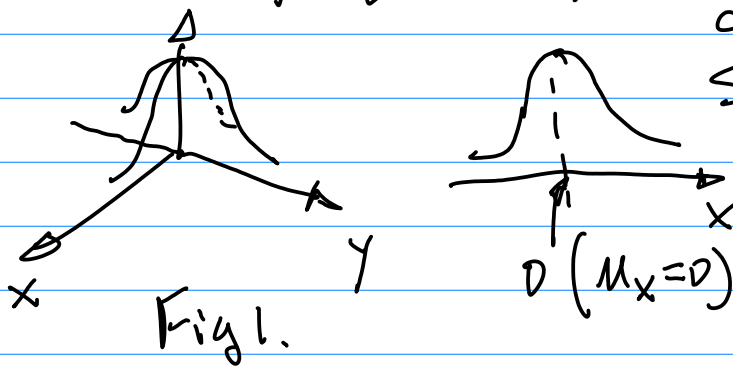


Fig 1.

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

From Eqn (4), ppt from the github 2018S-15-Lecb-V3 ...

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

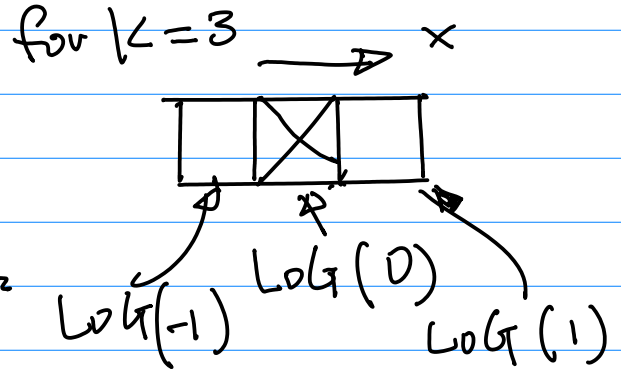
$$\nabla^2 G(x,y)\Big|_{y=0} = \nabla^2 G(x,0) \quad \dots (3)$$

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

$K \times 1$

K: No. of elements; One Row

for $K=3$



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

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

≙ Identify the Center Reference ≙ from LoG(x) (or $\nabla^2 G(x,0)$). map it to the kernel

Solve for y...

Driver Implementation. { f_{PWM}, D.C.

2018S-10-0 ~
PWM Driver

Add Duty Cycle Function to Device Driver.

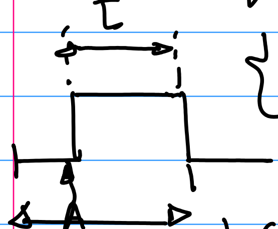
{ Theoretical Aspect
{ Implementation C Code.

SPRs { TCON ← Control Reg.
{ TCNT ← Counter Reg.
{ TCMP ← Compare Reg.
CFG0
CFG1

Root

"Prefix" "Timer"

PWM Output Square Waveform



{ TCNTB
{ TCMPB

Fig. 1

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

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

PP 1118, CPU Datasheet

Prescaler: 8 bit, [0, 255]

Divider: 1, 2, 4, 8, 16

Note CFG/CON are responsible for setting Prescalar/Divider value.

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

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

Find SPR, set SPR. to Realize this frequency.

From CPU Datasheet CFG0 CON.

Sol

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

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

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

Let Div = 16,

Solve for Pres.

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

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

Iteration,

Change PCLK to 10MHz, then, we have

CmpE/EE242

$$Pres = \frac{10\times10^6 - 16\times7\times10^3}{16\times2\times10^3}$$ 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}

TCNTB
- T: Timer
- CNT: Root
- B: Buffer

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

f ~~master~~ Clock peripheral

N Counts for CNT SPR.

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

- f: Given
- f_{PWM}: Target
- N: Unknown to be Calculated

Note: TCMPB
- CMP: 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-

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

Define the Period; CNT

Duty Cycle → % → Counts

Percentage ↓ Cmp

GPFCON PP. 522

GPFCON[29:28] = 10 → PWM 0

GPFCON[31:30] = 10 → PWM 1

#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) PWM } SPRs { CON DAT : 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.

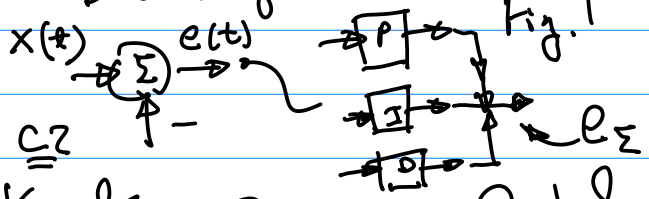


Fig. 1

C2

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

B.D. 2×1

↓

With Noise Reduction 3×1

Low Pass Filter : G(x) Gaussian.

↓ 2nd Order Derivation as in Computer Vision

∇^2 : Laplacian $\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2} \Rightarrow \frac{\partial}{\partial x}$

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

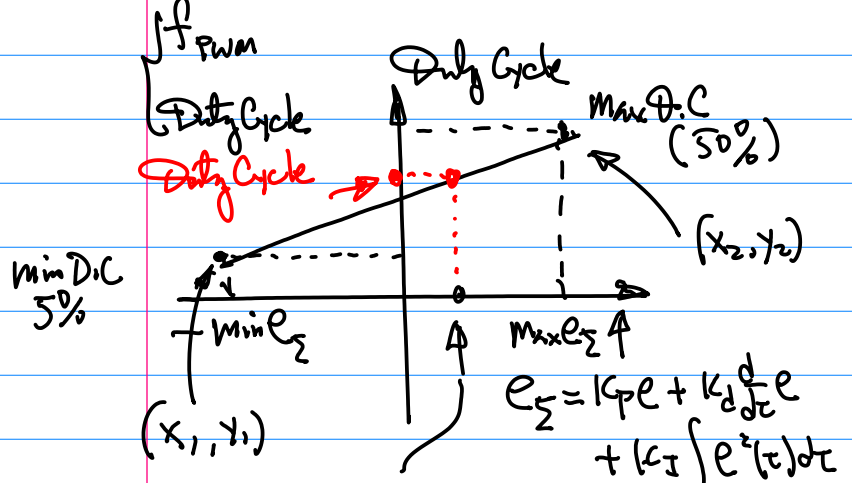
Datasheets if needed will be provided;

Convolution with Kernel(s)

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

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

Mapping to Control function PWM



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

$e_\Sigma = 0.57$ find control Action, → D.C = ?

To perform init & Config:
1° Binary Pattern for SPR.
Read/modify user Application Programs | Kernel Space Device Driver Program.
C Code for this purpose.

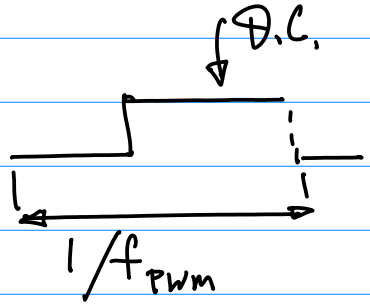
Note: PWM Waveform, e.g. Duty Cycle Calculation.

N. for CNT
N' for Cmp
T B

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

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

$\%(D.C) * N = N' \rightarrow$ Cmp



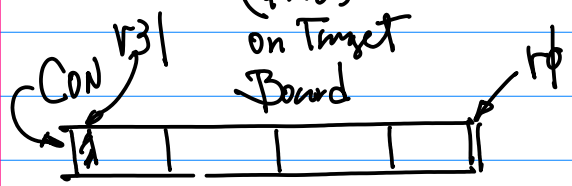
Implementation: Reference Platform ARM11

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)
PWM
GPP

GPx CON
GPx DAT
T CNT B
T Cmp B

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



Pre-requisit
1° O.S Source Distribution
2° Tool Chain Distro. "Cross Comp"
3° Datasheet.
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".

↓

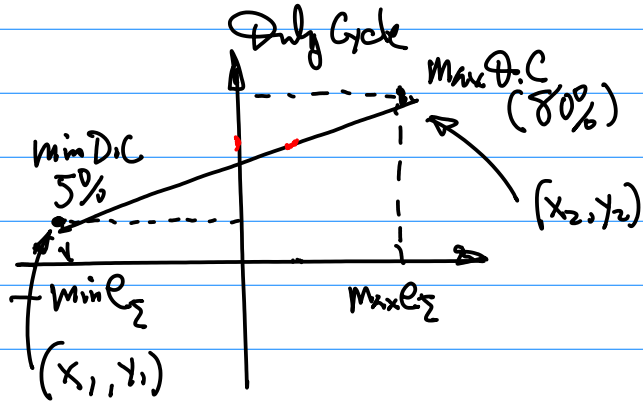
Copy "usb" "upLoad" by "cp" Copy Command to your target

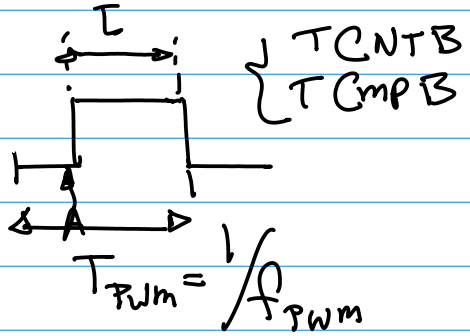
↓

"insmod" mytest.ko (To make it as a part of kernel Image)

↓

Run your user application program (By Calling the module)





CMPE/EE242

April 5th (Monday)

1. Midterm Graded, the key was Posted on Line, github, search under folder 2015, "Key"

2. 2nd half of this Course.

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

IIoT (Industrial IoT)

F.F.T to find/Characterize Analog Sensor Data (Nyquist Theorem)

Validation of Sensor Data.

OpAmps to Build Processing CKT.

"SPICE" Simulation.

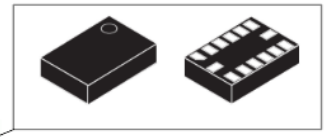
Example: LSM303

Note: Next Project use LSM303.

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

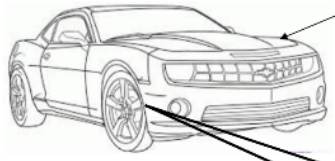
Sensors for Driving Direction and Turning Angle

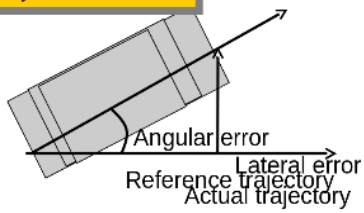
eCompass module:
3D accelerometer and 3D magnetometer

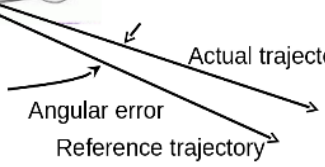


Use LSM303 or equivalent to sense the direction of the vehicle

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





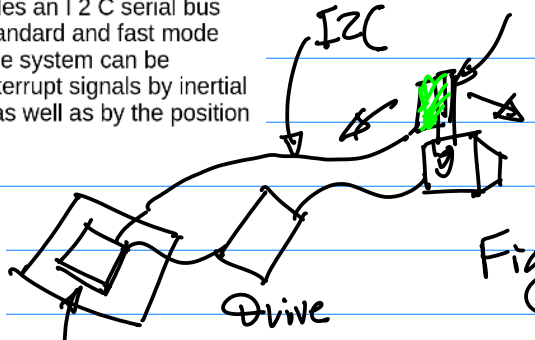


The LSM303DLHC includes an I 2 C serial bus interface that supports standard and fast mode 100 kHz and 400 kHz. The system can be configured to generate interrupt signals by inertial wake-up/free-fall events as well as by the position of the device itself.

Harry Li, Ph.D. April 2015

LSM303

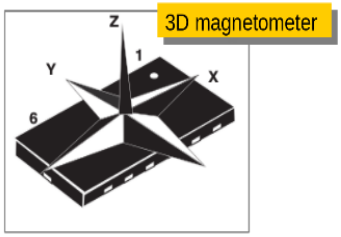
I2C



Embedded Target platform

Fig 1. Configuration for PID controller

3D Accelerometer and 3D Magnetomete LMS303



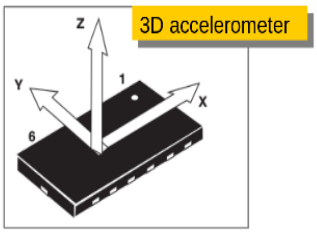


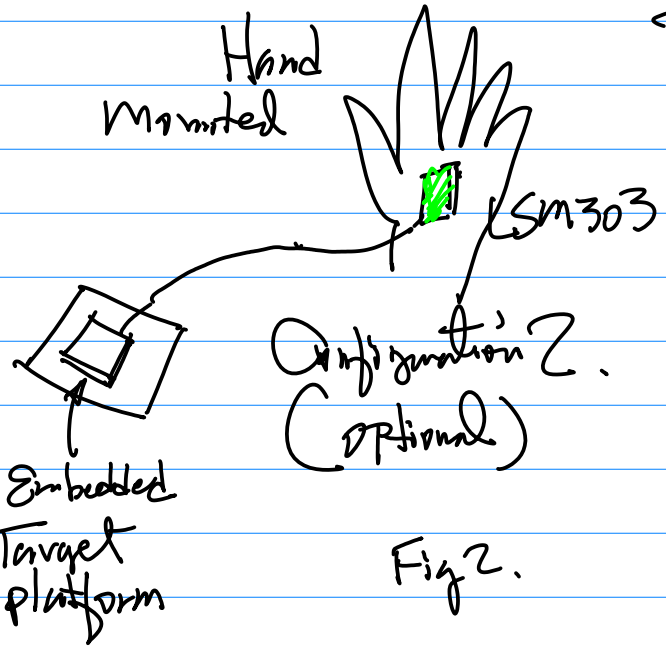
Table 9

Pin name	Pin description
SCL	I²C serial clock (SCL)
SDA	I²C 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)



Configuration 2. (Optional)

Fig 2.

3. a Space-Time Diagram

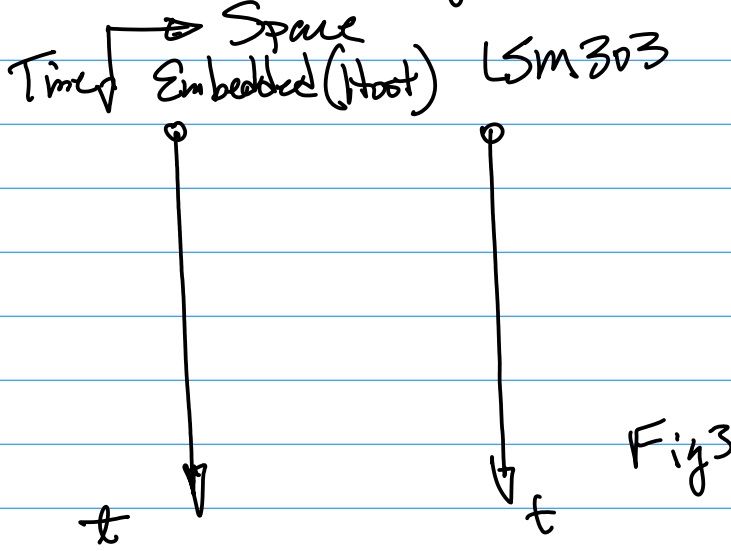


Fig 3

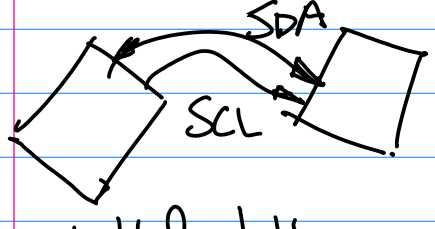
Submission On Canvas

Objective: ① To be Able to Read Sensor Input;

② To be Able Config the Sensor.

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)



I2C Device (LSM303)

Fig 4.

b To Describe "Hand-Shaking"

Three Small Steps.

Step 1. → Step 2 → Step 3

Host Slave "ACK" Data Transmission will start

Command. to the Target via Address for Init & Config

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

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

Notation: A frame

1st ST → SP

"Start" "Stop"

2° The Notations in Table11
SAD, SAD+W, SUB, DATA,
SAK. etc. pp.20

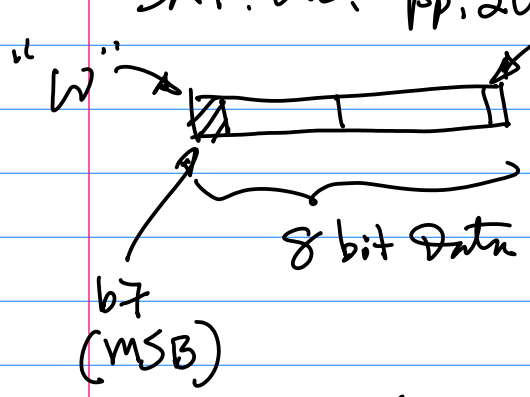
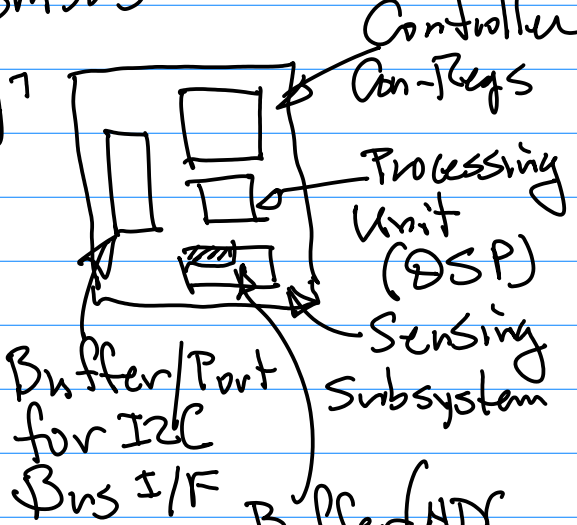


Fig5

3° from pp.20 (Datasheet)

Consider A Slave device
LSM303

Fig7



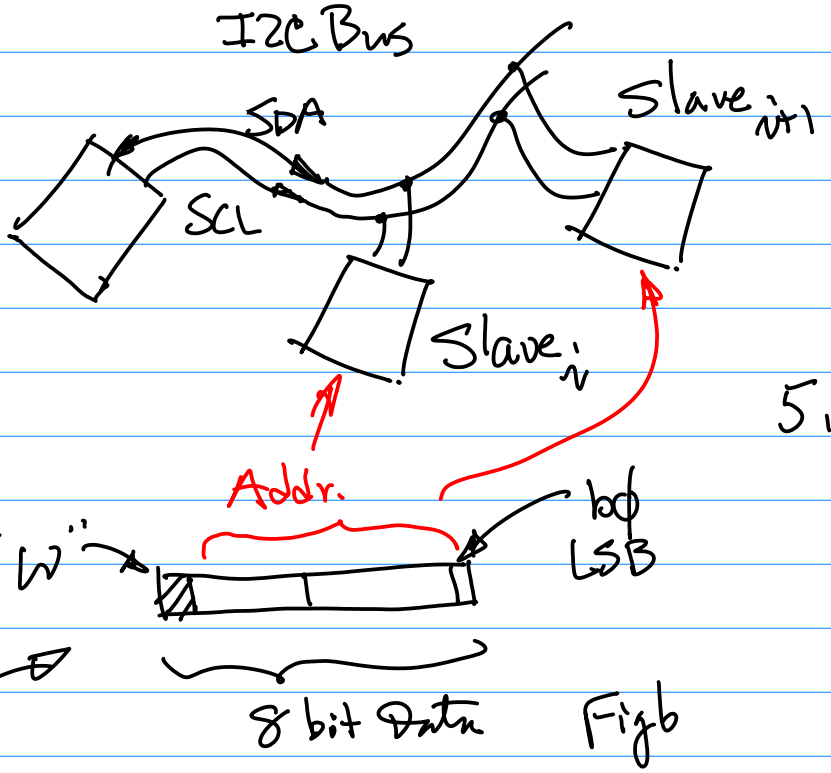


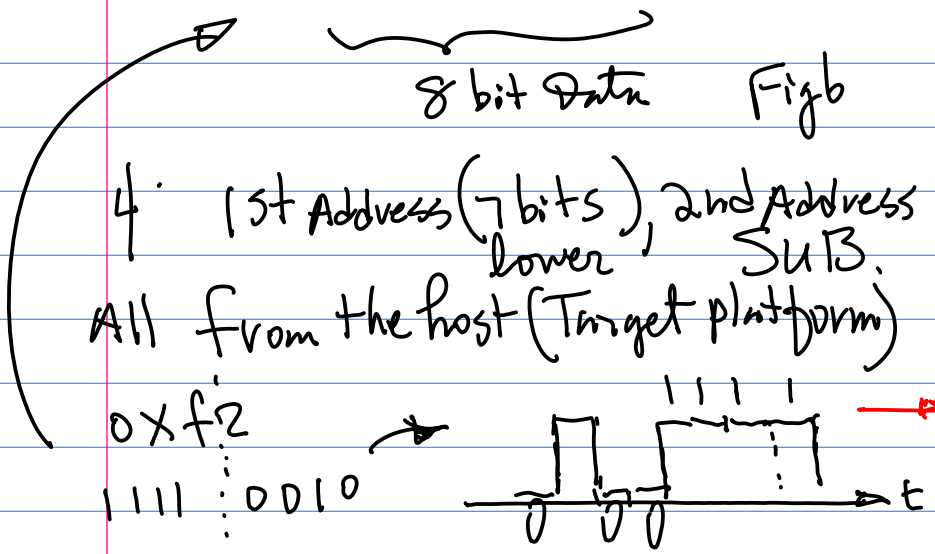
Fig6

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 Checked By "FAN-IN" "FAN-OUT"

128 Internal Addresses → Special Purpose Registers.

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



6. Most Significant Bit is transmitted first

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

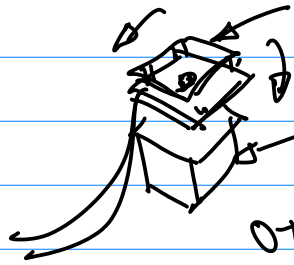
Homework (1pt) Due A week from Today, April 14, Due On CANVAS

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

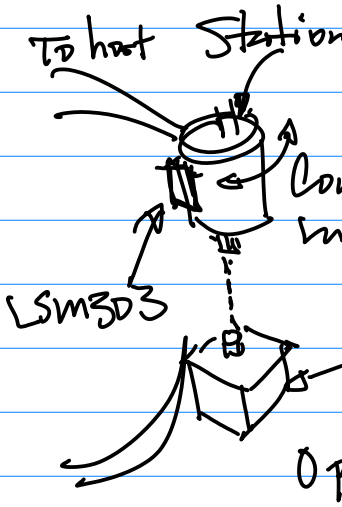
Note: Sample code is posted "as is" basis.

Repo: 20-2025-10-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 ?

fileNaming: firstName_4Digits_242.zip

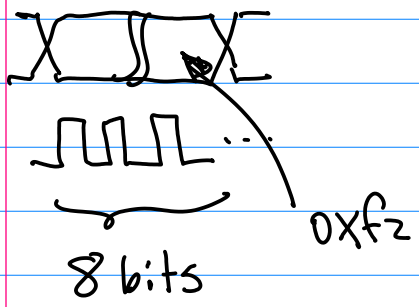
Note: Table 11 & 12 (pp19)

One Byte Writing (Host) → Multiple Bytes Writing (Host)

Table (Logical Behavior)

↓

Timing (Waveform)



Now, the Address for the Sensor(s) and Addresses for Registers

- Control Register — Init & Config
- Data Register

SAD[6:1] 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

TechSpec → Binary Pattern

TechSpec:

i. 400Hz Data Rate

ii. X-Y axis.

iii. Sensor Active (No LOWPOWR)

- 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] = 0X13 ✓

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

CTRL_REG1_A[7:0] 8 bits.

CTRL_REG1_A[7:4] = 0X7 (for 400Hz)

April 12 (Monday)

Homework Extension to April 19 (Monday)

Industrial Analog Sensor I/F Design

Example: NH_3 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

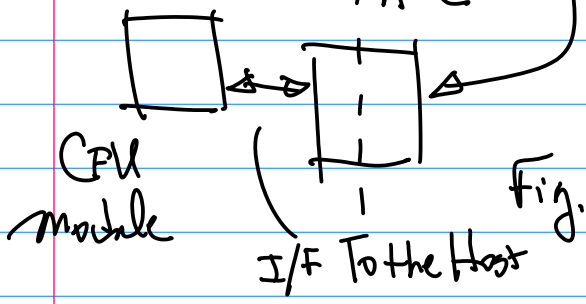


Fig. 1

2. Prototype to Build

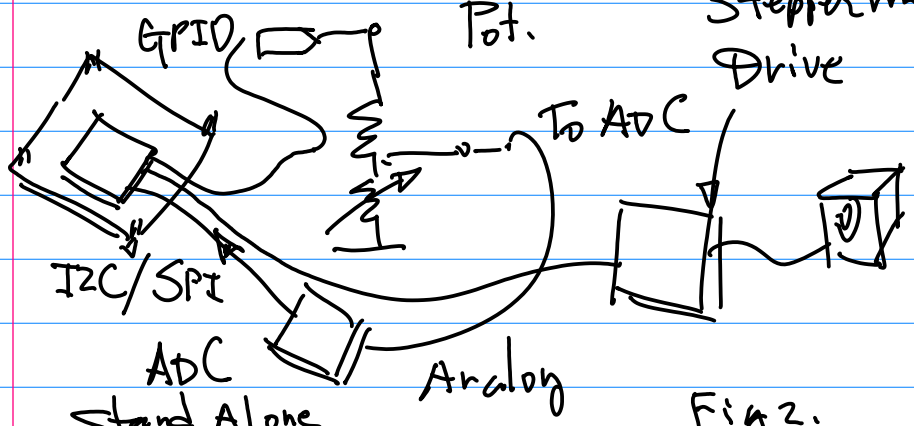


Fig 2.

3. Analog Interface Design.

Start with Characteristic Curve

↓

Linearization

↓

OpAmp Preprocessing CKT

↓

Optimized dynamic (Input) Range to ADC

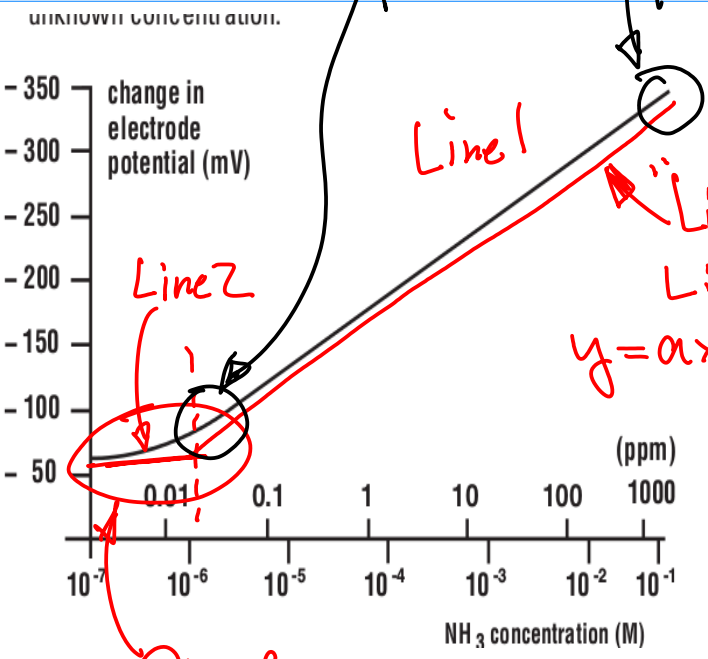
4. OpAmp's for Preprocessing

Addition
Subtraction
Division
multiplication

Math. Operations for Preprocessing

Example: from Datasheet
2018S-22-N13-
TP13, Fig. 14.

Step 1. Linearization Characteristic Curve



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

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

$y = ax^2 + bx + c$ 2nd order

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

Simplification: Just Keep Line 1.

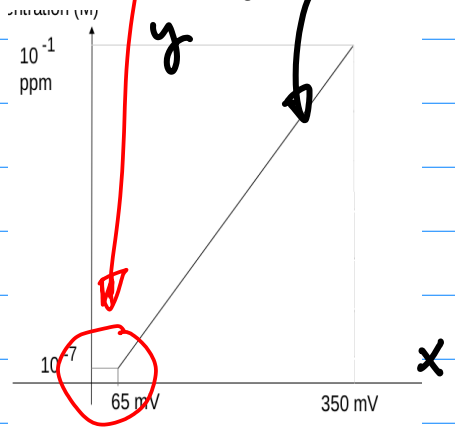
Note: Linearization By Identifying Starting and Ending points.

(V_0, C_0) (V_N, C_N)

Voltage V.S. Concentration

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

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

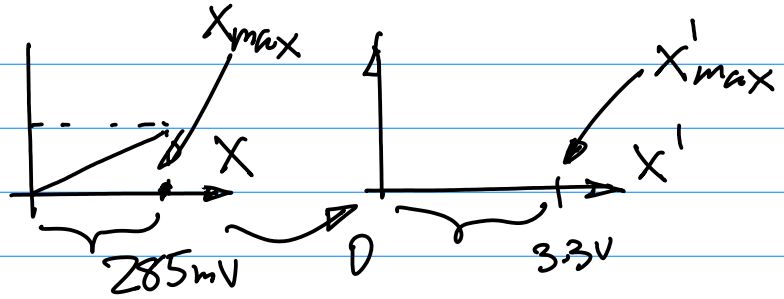


x_{New} is changed

By "Shifting"

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

2.2 Dynamic Range bits to ADC Dynamic Range



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

Gain ("Scaling Factor")

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

Review OpAmp { Inverting Configuration OpAmps; Non-Inverting Configuration

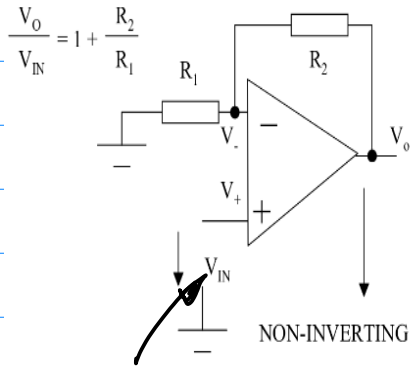
April 14 (Th).

OpAmps for Pre-processing Design

1. OpAmp { a. Very High Input Impedance; 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 }

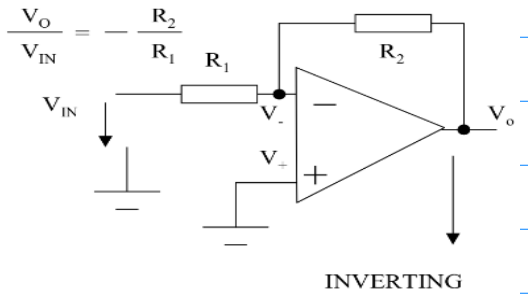


a. Input: Positive Polarity;

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

c. Draw the CKT.

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



a. Input: V_- pin

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

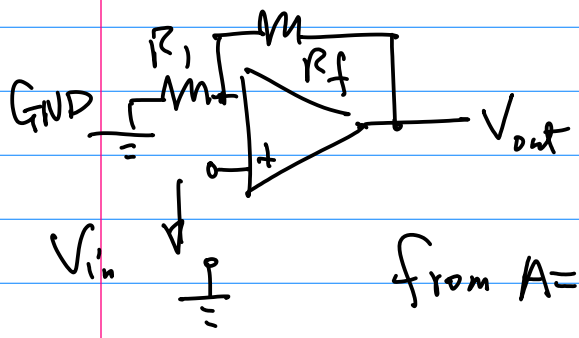
c. $$A = -\frac{R_f}{R_1} \quad \cdots (2)$$

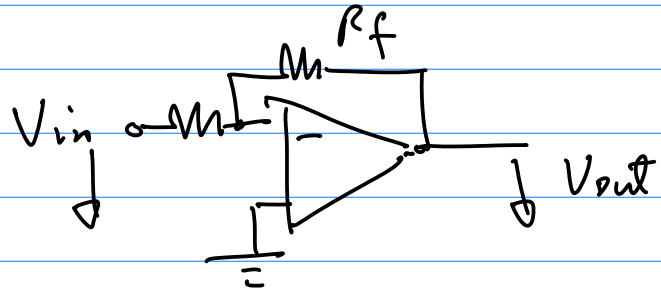
4. Use OpAmp Ckt(s) for Math. Operations.

a Addition. $X_1 + X_2$

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

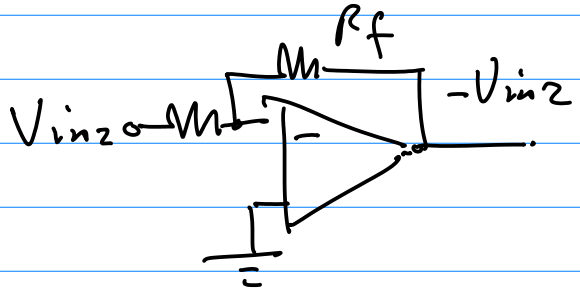
Non-Inverting Configuration





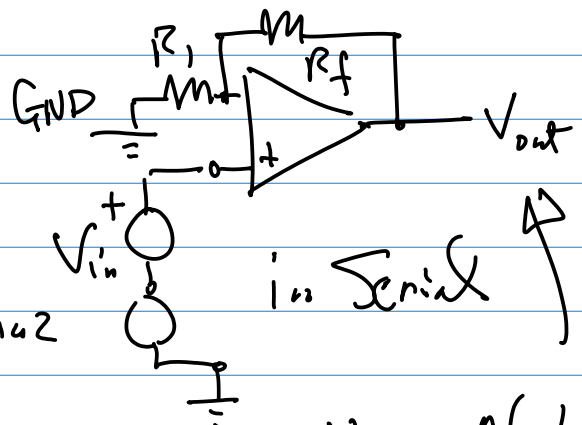
from $A = 1 + \frac{R_f}{R_1}$ $\quad \frac{V_{out}}{V_{in}} = A = -\frac{R_f}{R_1}$

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 $R_f = R_1 = 1K\Omega$

(74lopAmp.384 Quad-Pack)

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

Not too small, Noise will distore the Signal

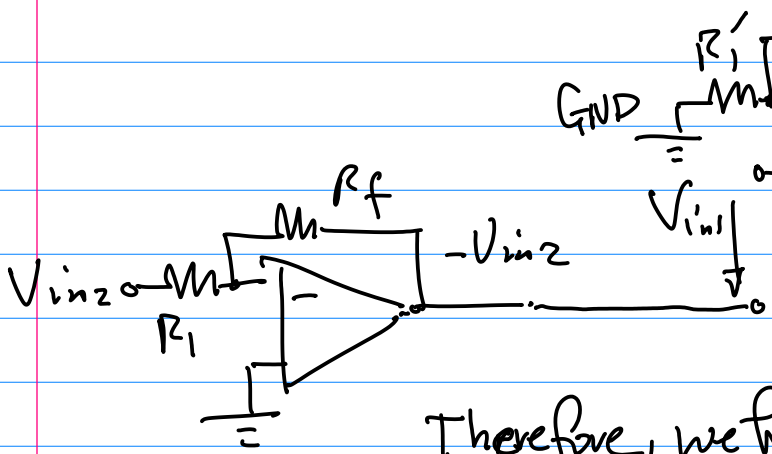
$V_{out} = A(V_{in1} + V_{in2})$

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 $V_{in_1} - V_{in_2}$



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

Therefore, we have

Subtraction

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

Multiplication. $y = ax$, $a > 1$

use Non-Inverting Configuration

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

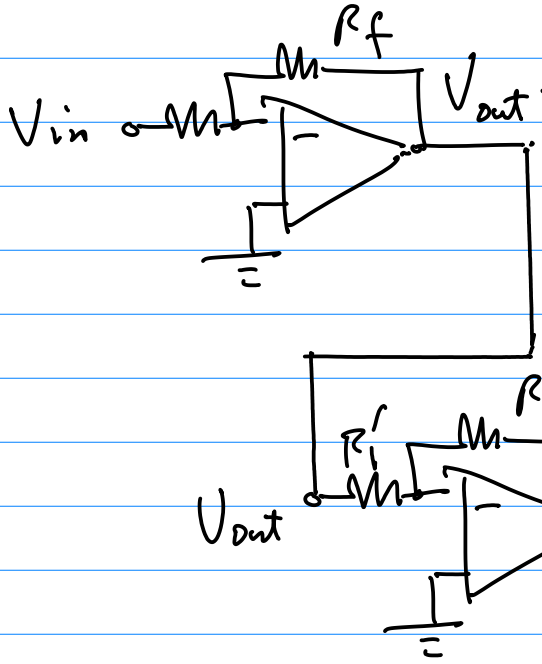
$$V_{out} = A \cdot V_{in} = \left(1 + \frac{R_f}{R_1}\right) V_{in}$$

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

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

$R'_f = R'_1 = 1K\Omega$. //

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