DSP et calcul pour traitement de signal

Francesco Mondada, Frank Bonnet
IEM - STI - EPFL

Introduction

Les systèmes embarqués doivent effectuer des tâches de plus en plus diverses, par exemple liées aux multimédias avec les smartphones, et donc aussi des calculs de plus en plus complexes. Le traitement de signal est un élément clé de ces systèmes depuis longtemps.

Pour traiter cette question, les ingénieurs ont dû trouver des moyens pour effectuer des opérations complexes en minimisant le temps de calculs des microcontrôleurs. Le DSP (Digital Signal Processing) est un cas typique que l'on va aborder dans ce cours.

Introduction

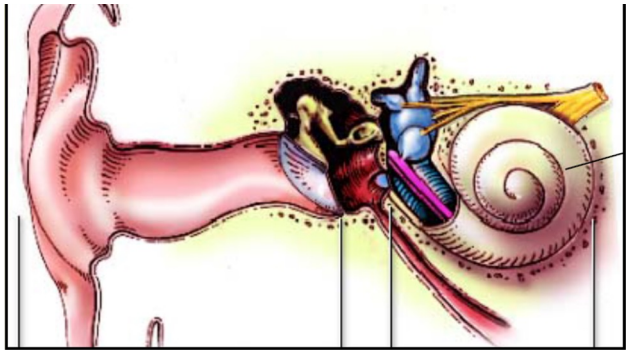
Sujets de ce cours sur le DSP:

- Exemple de traitement de signal embarqué
- Les instructions ARM liées au DSP
- FFT pour effectuer de l'analyse spectrale temps-réel sur le STM32F4

Motivation

Earing aids

Unitron.com



Signal Processing for Hearing Aids and Cochlear Implants, Alain Dufaux, EPFL course HIM, 2014

Motivation

Earing aids

- Analyse du signal requis
  - Correction de la réponse en fréquence
  - Modification d'amplitude
  - Filtrage de certaines fréquences
  - Annulation du feedback
  - Création de signaux d'interférences

⟹ **Processing compatible avec du son**

Motivation

Earing aids

Contraintes pour le design :

- Petite tailles (Parfois implanté dans le patient)
- Très faible consommation
- Très peu de délais!!
  - ➢ Délai acceptable **-> moins que 10ms,** sinon le patient est dérangé et l'appareil est inutilisable

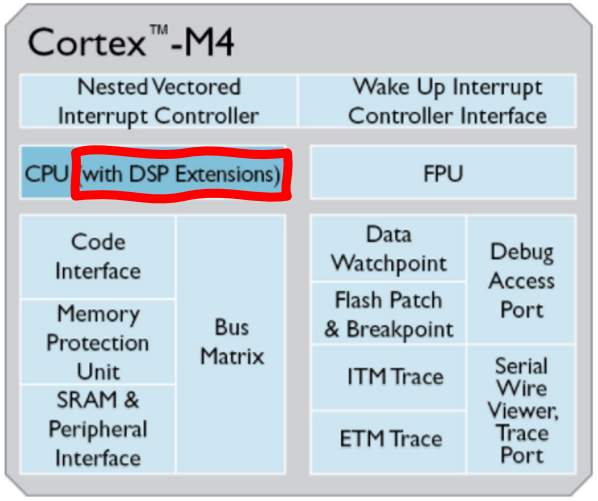
Introduction

Divers constructeurs de processeurs DSP faits spécifiquement pour ces tâches:

- Analog Devices
  - Par exemple TigerSHARC
- Texas Instruments
  - Par exemple TMS320C64xx

Introduction

Nous avons pu voir dès le premier cours que le processeur ARM Cortex-M4 dispose d'une extension DSP (Digital Signal Processing). Le DSP est une partie très commune des microcontrôleurs qui fournit des ressources de calcul très optimisées au calcul temps réel de signaux digitaux.



Coeur du processeur

https://community.arm.com/processors/b/blog/posts/armv6-m-vs-armv7-m---unpacking-the-microcontrollers

Introduction

L'opération de base du DSP est l'opération "MAC" (Multiply and ACcumulate) qui permet d'effectuer une suite de produits et leur somme, en les combinant en une seule opération.
On trouve ce type d'opérations, par ex., dans le produit scalaire:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

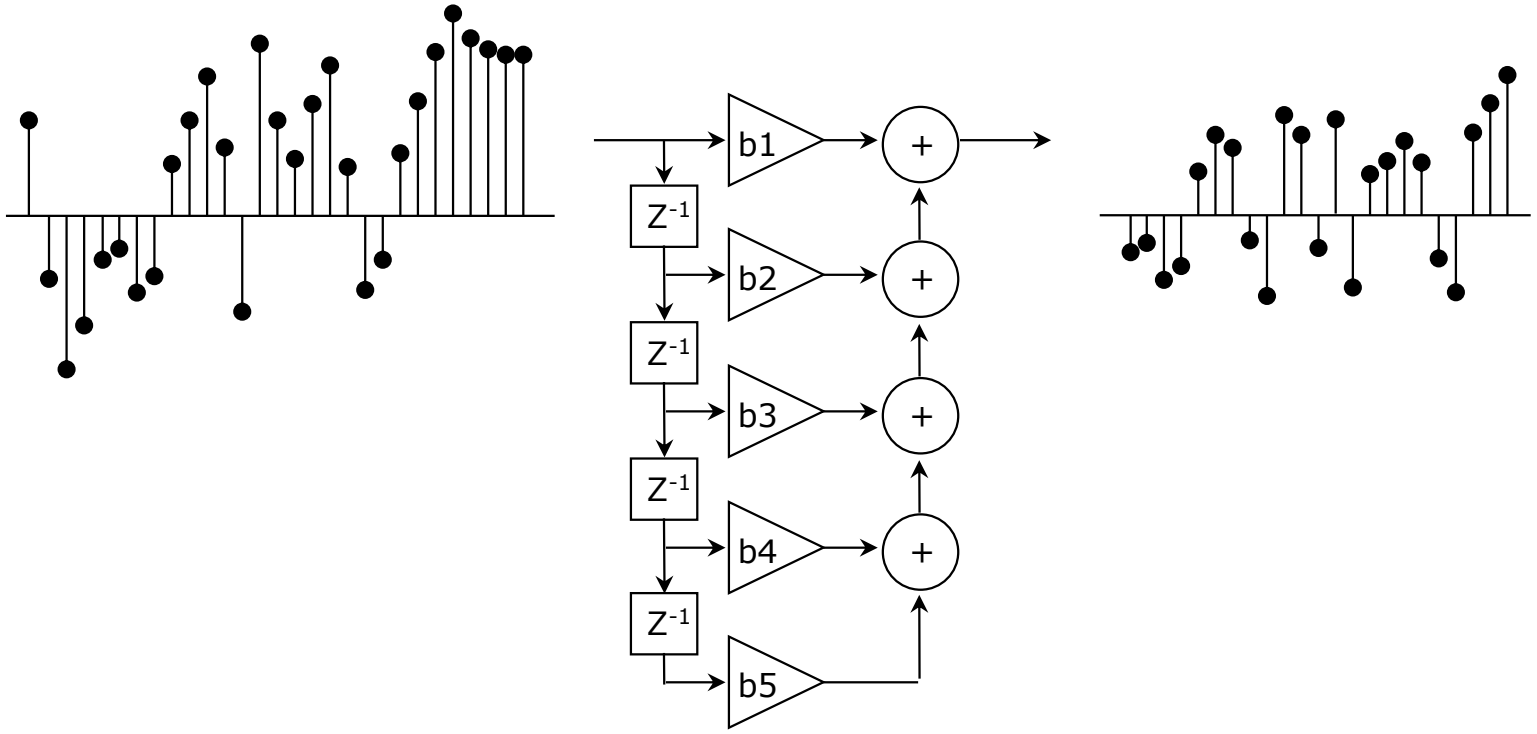
Mais attention: ces instructions, très particulières, ne sont pas forcément exploitées directement pas le compilateur!

Dans ce cadre il est important de comprendre:

- Quelles instructions permettent un gain de temps/mémoire
- Dans quel cadre de calcul elles sont utilisées
- Comment les intégrer avec du code "généraliste" en C.

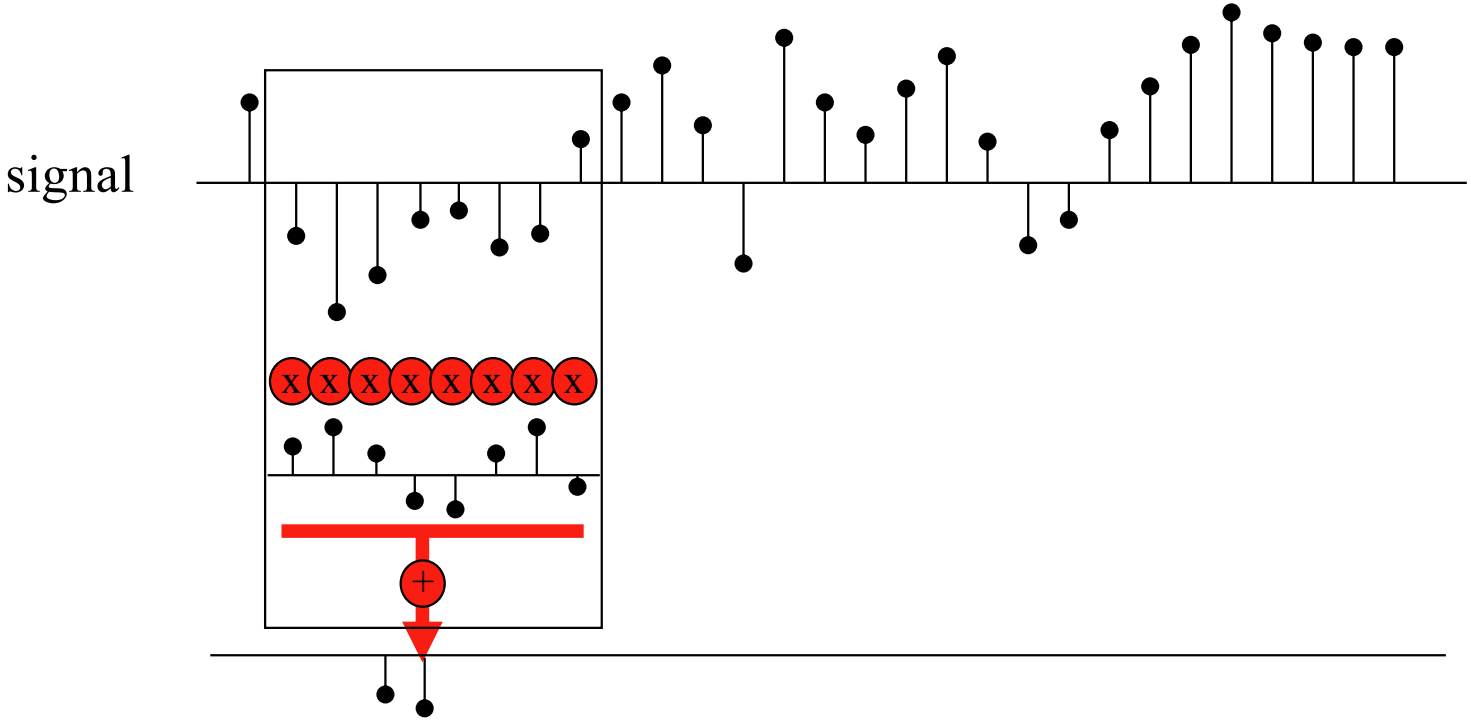
Motivation

Filtrage numérique (FIR Finite Impulse Response):



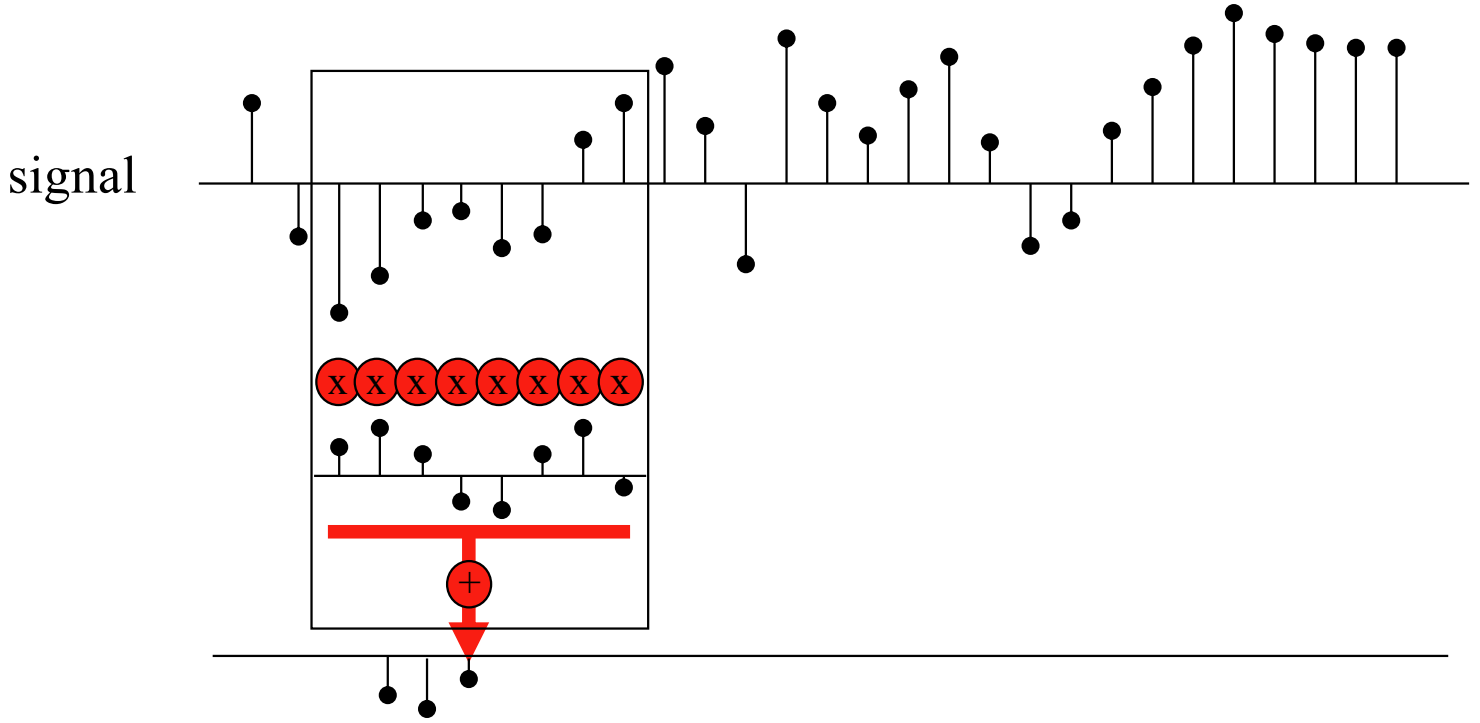
Motivation

Filtrage numérique:



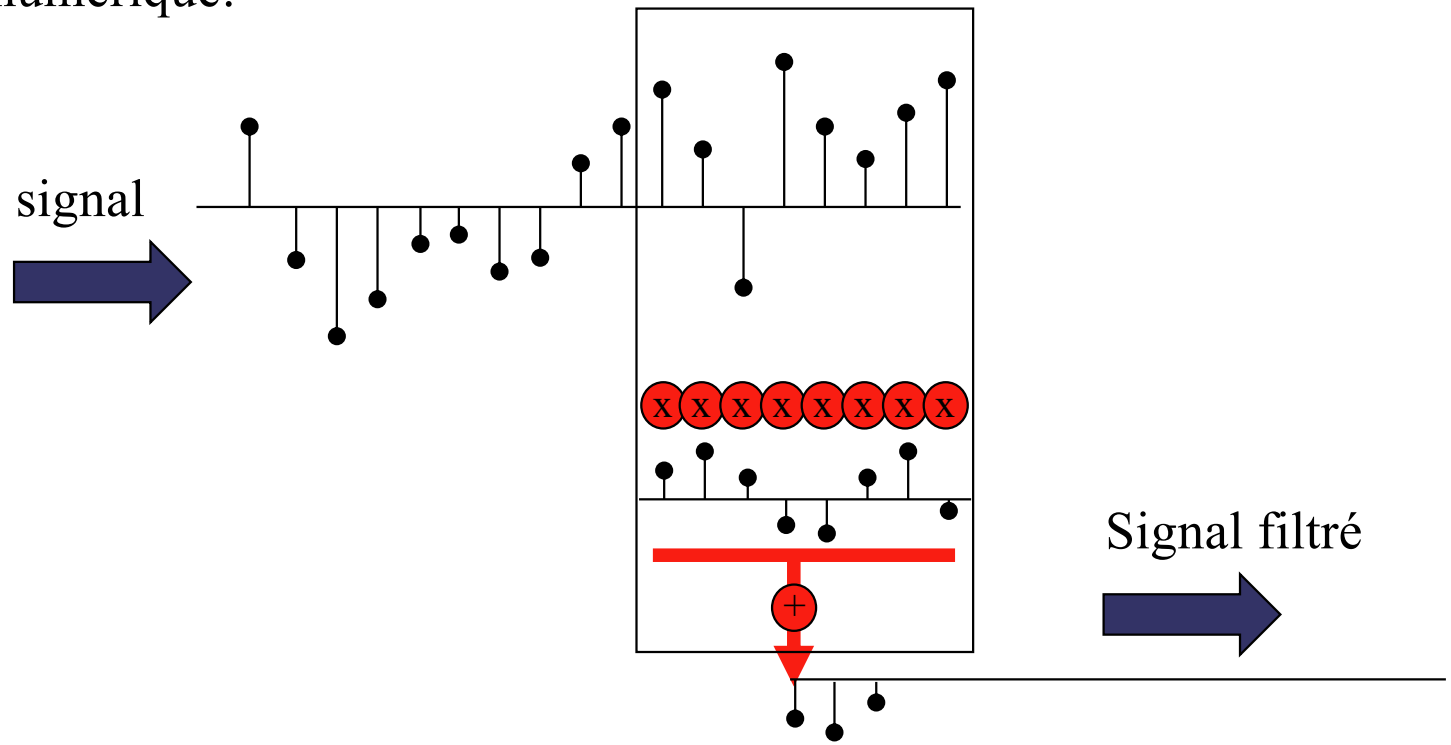
Motivation

Filtrage numérique:



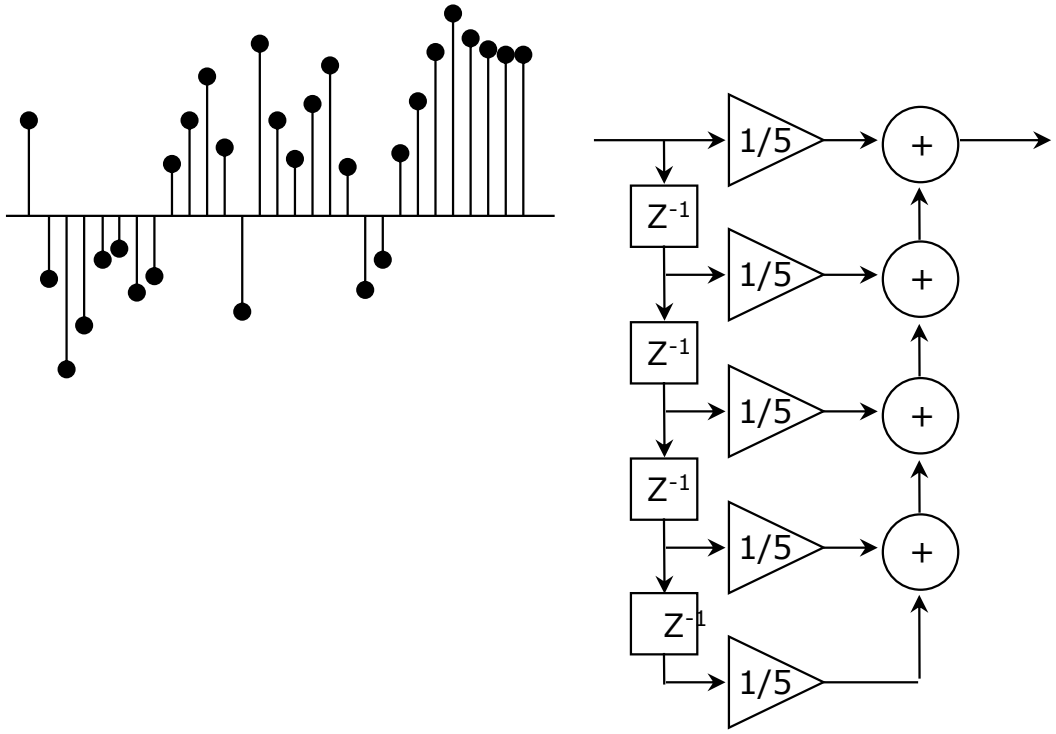
Motivation

Filtrage numérique:



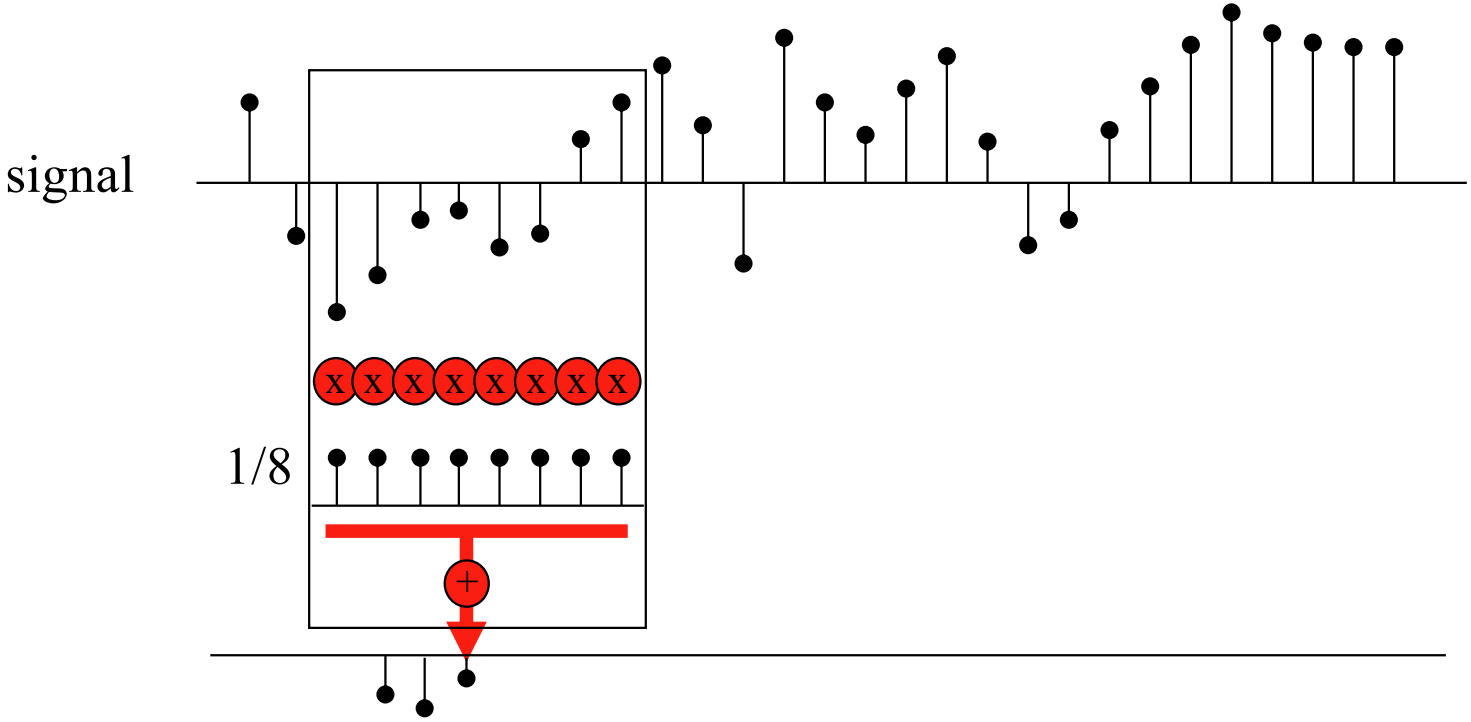
Motivation

Filtrage numérique (Moyenne mobile, filtre FIR simple):



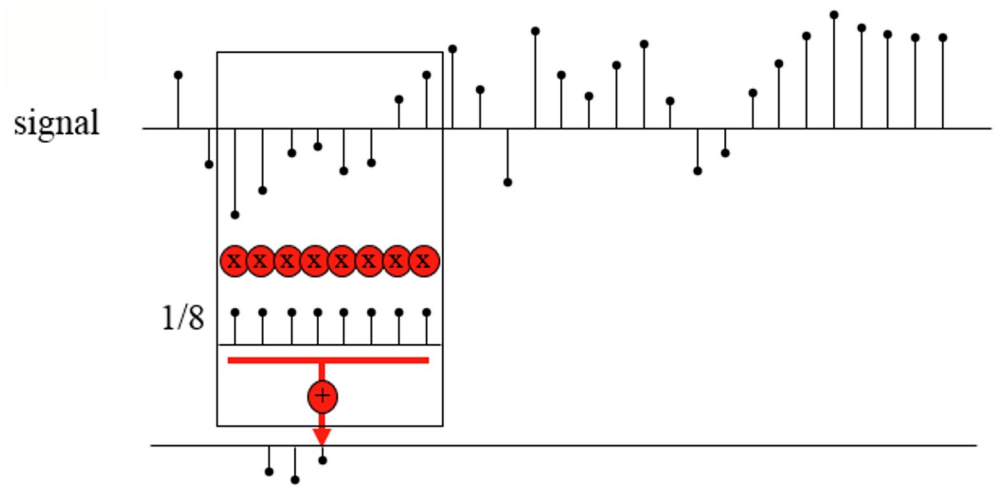
Motivation

Filtrage numérique (Moyenne mobile):



Motivation

Filtrage numérique (Moyenne mobile):

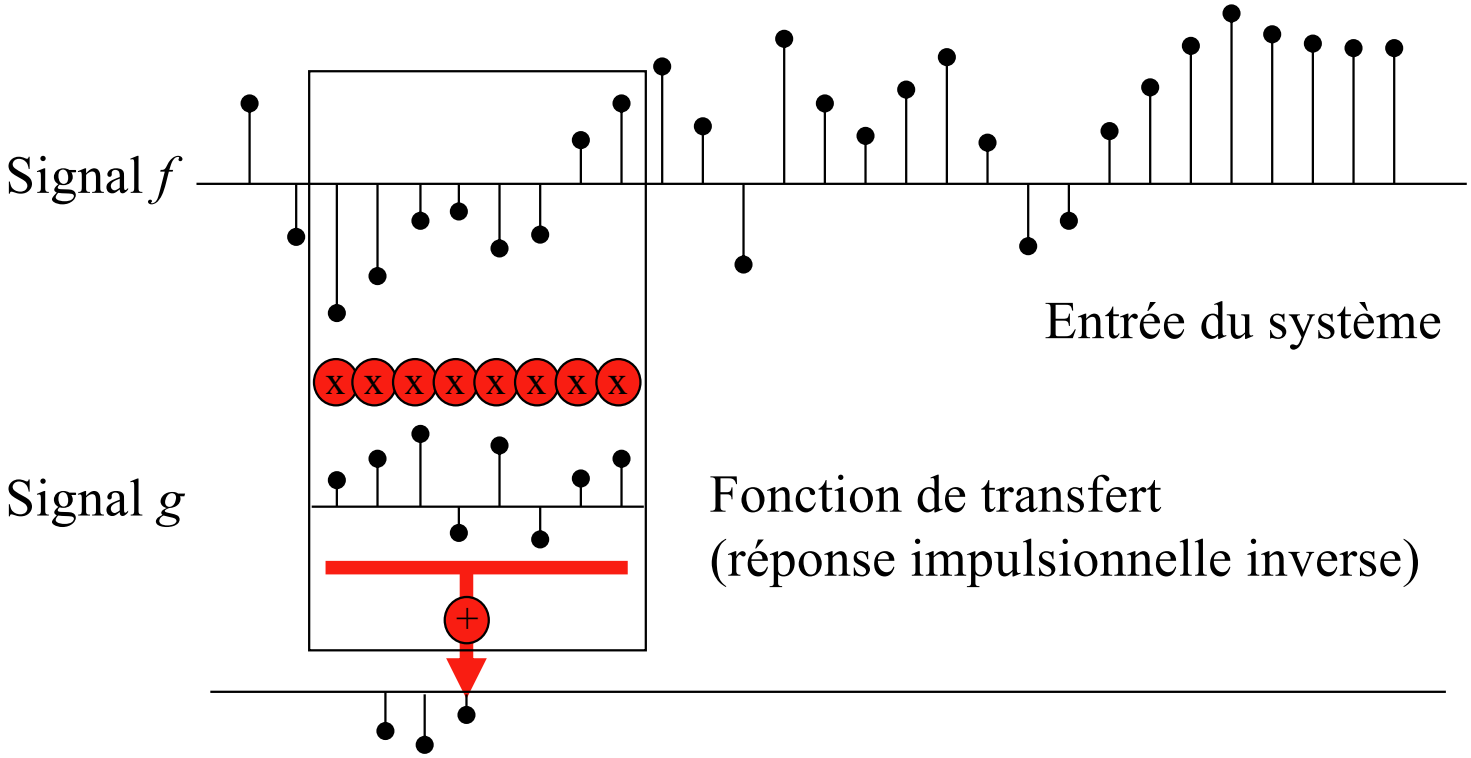


Ce type de calcul implique une série d'additions et de multiplications. Si le code n'est pas optimisé, cela engendre de gros codes assembleur à la compilation, et donc un temps d'exécution très lent.

Motivation

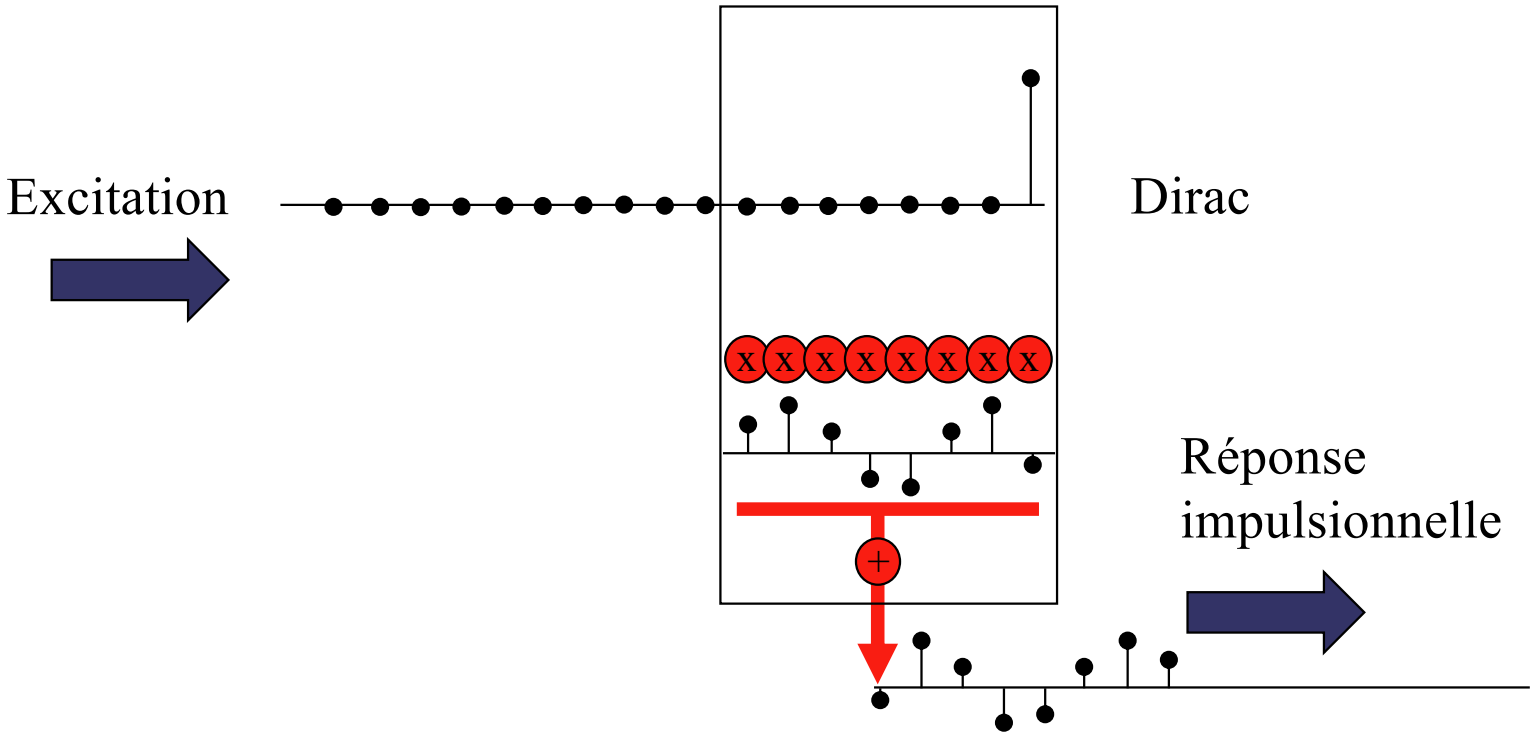
Autre exemple, la convolution:

$$(f * g)(t) = \int f(\tau) g(t - \tau)\, d\tau$$



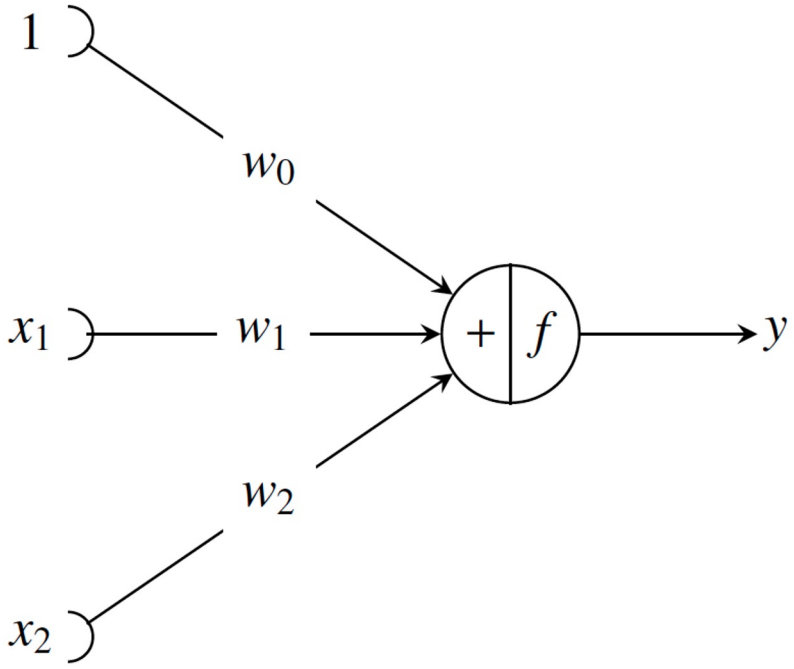
Motivation

Réponse impulsionnelle d'un système linéaire:



Motivation

Plus proche et lié à la robotique et l'intelligence artificielle: les réseaux de neurones impliquent aussi ce types d'opération à répétition



Solution, les instructions SIMD sur ARM

Pour réaliser de façon efficace ces types d'opérations, les processeurs ARM cortex-M4 (et plus, exemple M7) contiennent des sets d'instruction dites SIMD (Single Instruction Multiple Data) exploitant l'extension DSP du processeur.

Cela va permettre par exemple d'effectuer une multiplication de deux entiers suivie d'une addition de manière "hardware" = en 1 seul cycle.

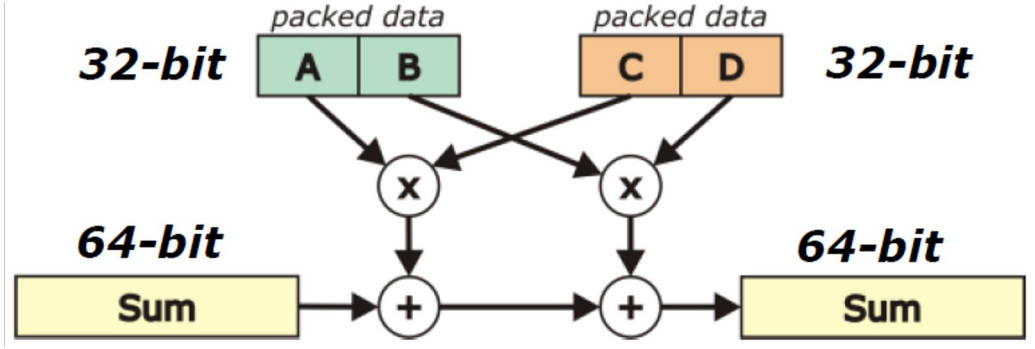
L'instruction de multiplication suivie d'une addition est communément appelée instruction MAC (Multiply and Accumulate), qui est donc l'instruction "de base" des coeurs DSP.

Les instructions SIMD

Sur le Cortex-M4, il est même possible d'effectuer 2 séries de multiplications suivies d'additions en 1 seul cycle (c'est le maximum):

A,B,C,D = entiers 16 bits **Somme = entier de 64 bits**

Somme = Somme + (A x C) + (B x D)



Les instructions SIMD

OPERATION	INSTRUCTIONS
16 x 16 = 32	SMULBB, SMULBT, SMULTB, SMULTT
16 x 16 + 32 = 32	SMLABB, SMLABT, SMLATB, SMLATT
16 x 16 + 64 = 64	SMLALBB, SMLALBT, SMLALTB, SMLALTT
16 x 32 = 32	SMULWB, SMULWT
(16 x 32) + 32 = 32	SMLAWB, SMLAWT
(16 x 16) ± (16 x 16) = 32	SMUAD, SMUADX, SMUSD, SMUSDX
(16 x 16) ± (16 x 16) + 32 = 32	SMLAD, SMLADX, SMLSD, SMLSDX
(16 x 16) ± (16 x 16) + 64 = 64	SMLALD, SMLALDX, SMLSLD, SMLSLDX
32 x 32 = 32	MUL
32 ± (32 x 32) = 32	MLA, MLS
32 x 32 = 64	SMULL, UMULL
(32 x 32) + 64 = 64	SMLAL, UMLAL
(32 x 32) + 32 + 32 = 64	UMAAL
32 ± (32 x 32) = 32 (upper)	SMMLA, SMMLAR, SMMLS, SMMLSR
(32 x 32) = 32 (upper)	SMMUL, SMMULR

Chaque instruction prend **1 cycle**

An introduction to the Cortex-M4 processor architecture, Shyam Sadasivan

Les instructions SIMD

		Cycle counts		
CLASS	INSTRUCTION	ARM9E-S	CORTEX-M3	Cortex-M4
Arithmetic	ALU operation (not PC)	1 - 2	1	1
	ALU operation to PC	3 - 4	3	3
	CLZ	1	1	1
	QADD, QDADD, QSUB, QDSUB	1 - 2	n/a	1
	QADD8, QADD16, QSUB8, QSUB16	n/a	n/a	1
	QDADD, QDSUB	n/a	n/a	1
	QASX, QSAX, SASX, SSAX	n/a	n/a	1
	SHASX, SHSAX, UHASX, UHSAX	n/a	n/a	1
	SADD8, SADD16, SSUB8, SSUB16	n/a	n/a	1
	SHADD8, SHADD16, SHSUB8, SHSUB16	n/a	n/a	1
	UQADD8, UQADD16, UQSUB8, UQSUB16	n/a	n/a	1
	UHADD8, UHADD16, UHSUB8, UHSUB16	n/a	n/a	1
	UADD8, UADD16, USUB8, USUB16	n/a	n/a	1
	UQASX, UQSAX, USAX, UASX	n/a	n/a	1
	UXTAB, UXTAB16, UXTAH	n/a	n/a	1
	USAD8, USADA8	n/a	n/a	1
Multiplication	MUL, MLA	2 - 3	1 - 2	1
	MULS, MLAS	4	1 - 2	1
	SMULL, UMULL, SMLAL, UMLAL	3 - 4	5 - 7	1
	SMULBB, SMULBT, SMULTB, SMULTT	1 - 2	n/a	1
	SMLABB, SMLBT, SMLATB, SMLATT	1 - 2	n/a	1
	SMULWB, SMULWT, SMLAWB, SMLAWT	1 - 2	n/a	1
	SMLALBB, SMLALBT, SMLALTB, SMLALTT	2 - 3	n/a	1
	SMLAD, SMLADX, SMLALD, SMLALDX	n/a	n/a	1
	SMLSD, SMLSDX	n/a	n/a	1
	SMLSLD, SMLSLD	n/a	n/a	1
	SMMLA, SMMLAR, SMMLS, SMMLSR	n/a	n/a	1
	SMMUL, SMMULR	n/a	n/a	1
	SMUAD, SMUADX, SMUSD, SMUSDX	n/a	n/a	1
	UMAAL	n/a	n/a	1
Division	SDIV, UDIV	n/a	2 - 12	2 – 12

Single cycle MAC

An introduction to the Cortex-M4 processor architecture, Shyam Sadasivan

Les instructions DSP vs FPU

Si on compare avec les instructions FPU du Cortex-M4, une “MAC” en utilisant des nombres flottants 32 bits prend 3 cycles. L’équivalent “entier” 32 bits prend 1 cycle avec le DSP.

Multiply	float	VMUL.F32	1
	then accumulate float	VMLA.F32	3
	then subtract float	VMLS.F32	3
	then accumulate then negate float	VNMLA.F32	3
	then subtract then negate float	VNMLS.F32	3
Multiply (fused)	then accumulate float	VFMA.F32	3
	then subtract float	VFMS.F32	3
	then accumulate then negate float	VFNMA.F32	3
	then subtract then negate float	VFNMS.F32	3

http://infocenter.arm.com

Gestion des instructions SIMD à la compilation

Par défaut, un code écrit en C avec des multiplications suivies d'additions ne générera pas du code assembleur exploitant les instructions SIMD.

```
int main() {

        int a = 5;
        int b = 6;
        int somme = 7;

        somme = somme + (a * b);

        return 0;
}
```

Test.c

Compilation

```
sub     sp, sp, #20
        add     r7, sp, #0
        movs    r3, #5
        str     r3, [r7, #12]
        movs    r3, #6
        str     r3, [r7, #8]
        movs    r3, #7
        str     r3, [r7, #4]
        ldr     r3, [r7, #12]
        ldr     r2, [r7, #8]
        mul     r3, r2, r3
        ldr     r2, [r7, #4]
        add     r3, r3, r2
        str     r3, [r7, #4]
        movs    r3, #0
        mov     r0, r3
        adds    r7, r7, #20
```

Test.s

Gestion des instructions SIMD à la compilation

Il faut utiliser la bonne écriture du code pour que le compilateur exploite les instructions SIMD. Exemple, l'instruction **SMMLA**

SMMLA – 32-bit multiply with 32-most-significant-bit accumulate

Fractional q31_t multiply accumulate. Multiplies two 32-bit integers, generates a 64-bit result, and adds the high bits of the result to a 32-bit accumulator.

Processor support: M4 only [1 cycle]

The instruction is available in C code via the idiom:

```
(int32_t) (((int64_t) x * y + ((int64_t) acc << 32)) >> 32);
```

C code example:

```
// Performs a true fractional MAC
int32_t x, y, acc;

acc = (int32_t) (((int64_t) x * y + ((int64_t) acc << 32)) >> 32);
acc <<= 1;
```

Gestion des instructions SIMD à la compilation

Dans le cas des TPs, ces instructions sont déjà écrites dans des **define** qu'il suffit d'utiliser lorsqu'on veut effectuer des opérations utilisant les instructions SIMD

```
/* SMMLAR */
#define multAcc_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((((q63_t) a) << 32) + ((q63_t) x * y) + 0x80000000LL ) >> 32)

/* SMMLSR */
#define multSub_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((((q63_t) a) << 32) - ((q63_t) x * y) + 0x80000000LL ) >> 32)

/* SMMULR */
#define mult_32x32_keep32_R(a, x, y) \
    a = (q31_t) (((q63_t) x * y + 0x80000000LL ) >> 32)

/* SMMLA */
#define multAcc_32x32_keep32(a, x, y) \
    a += (q31_t) (((q63_t) x * y) >> 32)

/* SMMLS */
#define multSub_32x32_keep32(a, x, y) \
    a -= (q31_t) (((q63_t) x * y) >> 32)
```

arm_math.c

Gestion des instructions SIMD à la compilation

Il est aussi possible d'appeler l'instruction assembleur à l'aide d'une fonction en C (ex: __SMMLA() dans CMSIS DSP Software Library)

```
uint32_t __SMMLA ( int32_t val1,
                   int32_t val2,
                   int32_t val3
                 )
```

This function enables you to perform a signed 32-bit multiplications, adding the most significant 32 bits of the 64-bit result to a 32-bit accumulate operand.

Parameters

val1 first operand for multiplication.
val2 second operand for multiplication.
val3 accumulate value.

Returns

the product of multiplication (most significant 32 bits) is added to the accumulate value, as a 32-bit integer.

Operation:

```
p = val1 * val2
res[31:0] = p[61:32] + val3[31:0]
```

Gestion des instructions SIMD à la compilation

L'appel d'instructions en assembleur peut aussi être fait depuis le langage C avec le mot-clé `__asm` :

```
#include <stdio.h>

int add(int i, int j)
{
  int res = 0;
  __asm ("ADD %[result], %[input_i], %[input_j]"
    : [result] "=r" (res)
    : [input_i] "r" (i), [input_j] "r" (j)
  );
  return res;
}

int main(void)
{
  int a = 1;
  int b = 2;
  int c = 0;

  c = add(a,b);

  printf("Result of %d + %d = %d\n", a, b, c);
}
```

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.100748_0606_00_en/ddx1471430827125.html

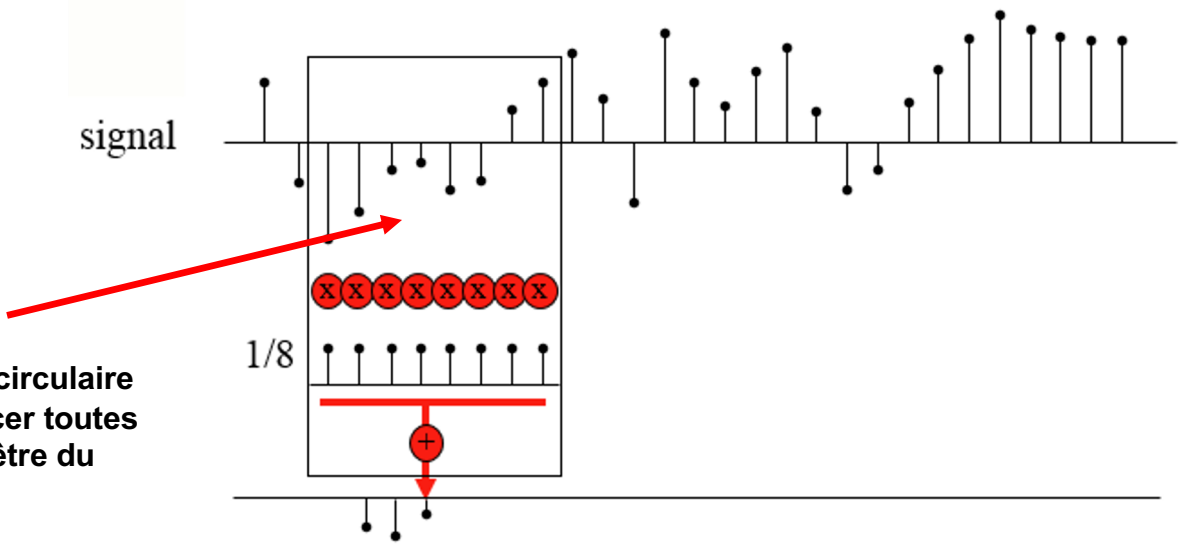
Gestion des instructions SIMD à la compilation

Il est donc possible d'écrire du code optimisé en exploitant les instructions SIMD liées à l'extension DSP du processeur pour gagner des cycles, avec des gains allant de ~4x pour des codes simples et pouvant aller beaucoup plus pour des codes compliqués avec des filtres appliqués à des signaux.

D'autres "astuces" peuvent limiter le nombres d'opération lors du traitement de signal. Par exemple lors d'opération de filtrage sur un signal, on peut utiliser un buffer circulaire pour appliquer un filtre.

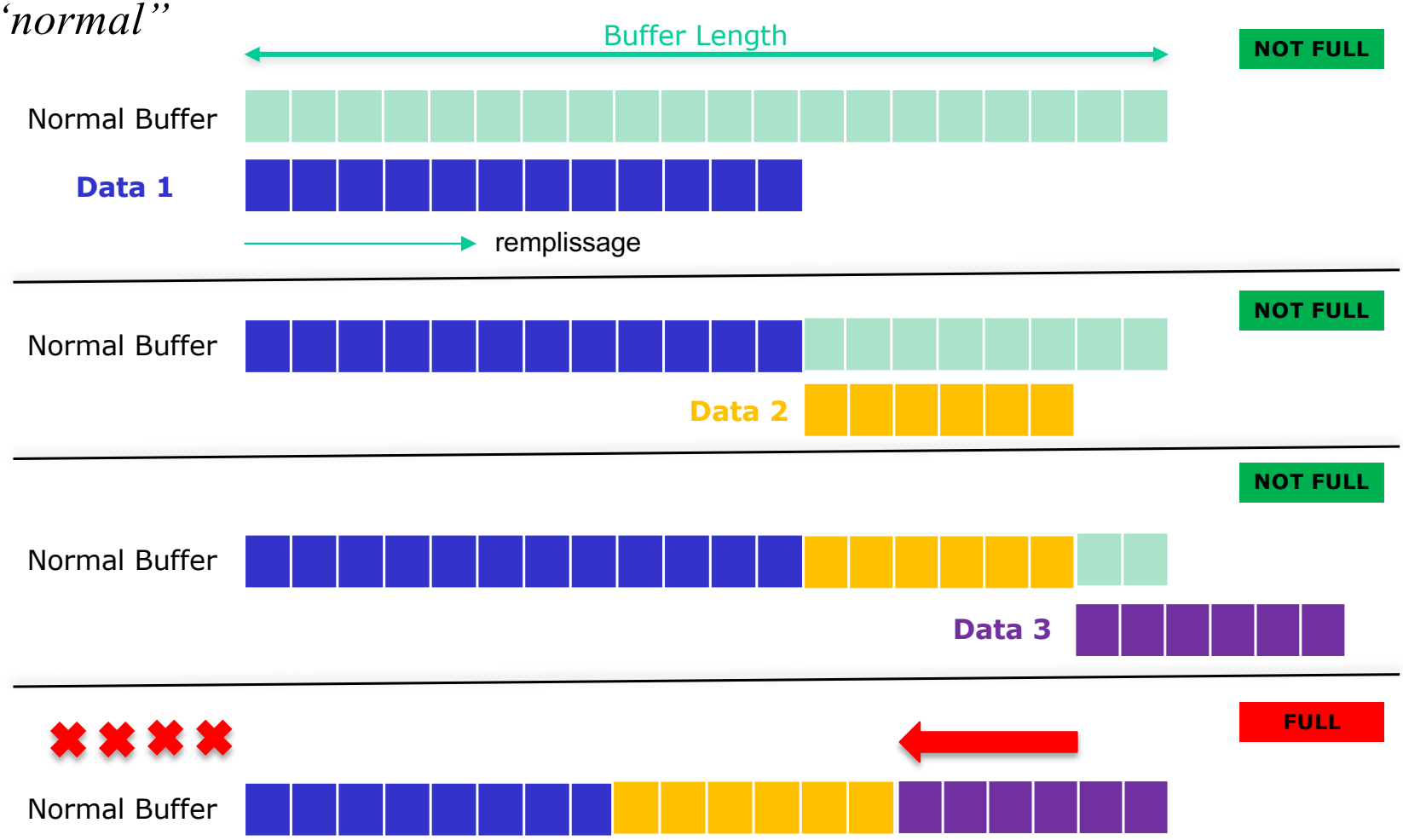
Buffer circulaire

Exemple avec le calcul de la moyenne

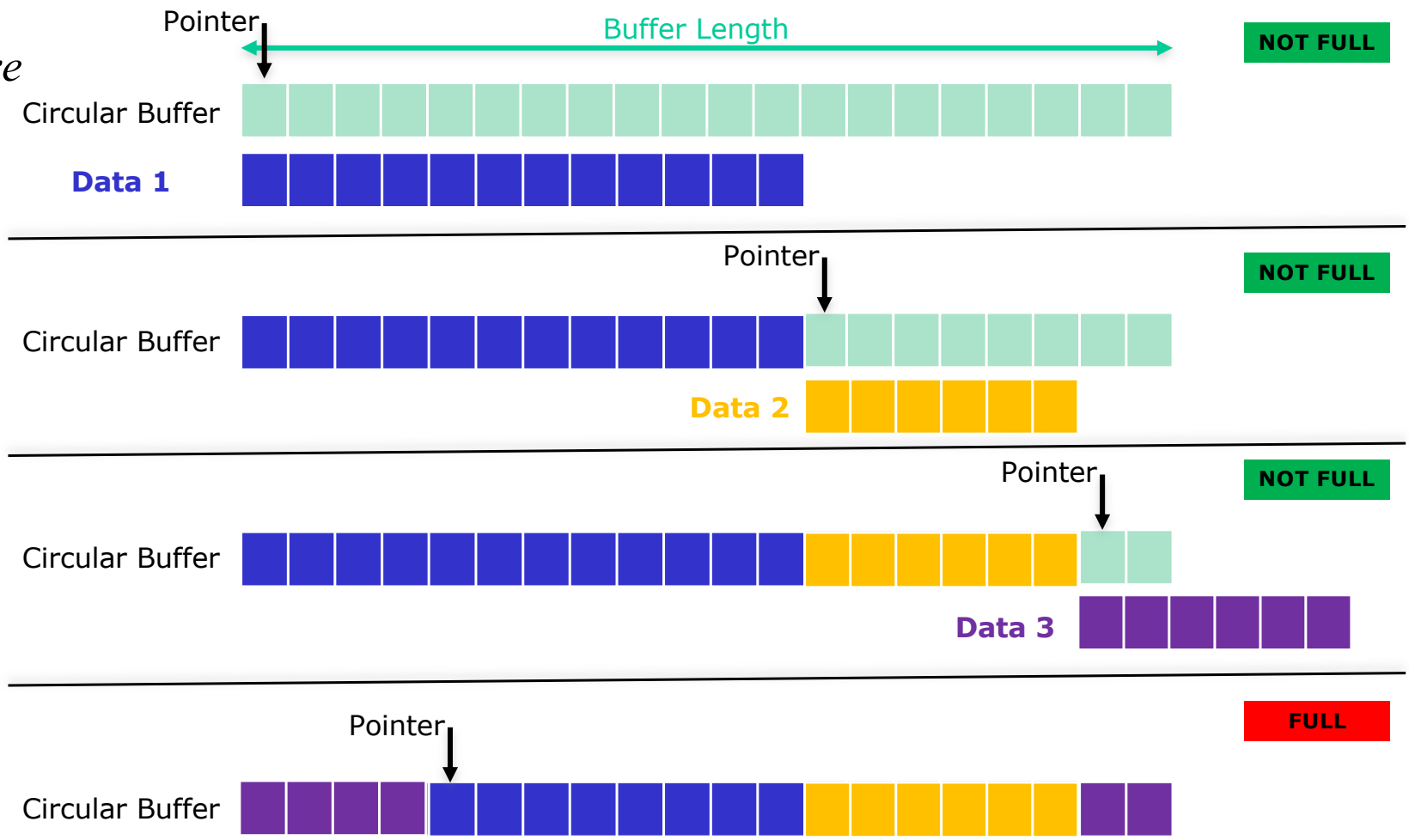


On utilise un buffer circulaire pour ne pas remplacer toutes les termes de la fenêtre du signal.

Buffer "normal"



Buffer circulaire



Buffer circulaire

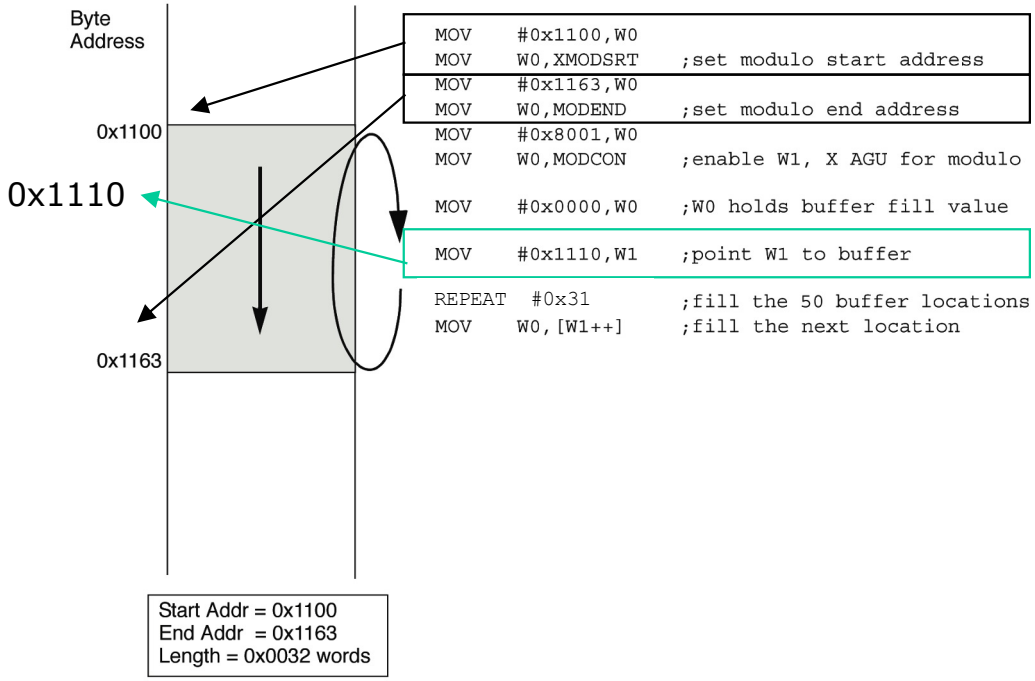
Implémentation en C

```
// Block-based FIR filter.
// N equals the length of the filter (number of taps)
// blockSize equals the number of samples to process
// state[] is the state variable buffer and contains the previous N
//  samples of the input
// stateIndex points to the oldest sample in the state buffer. It
//  will be overwritten with the most recent input sample.
// coeffs[] holds the N coefficients
// inPtr and outPtr point to the input and output buffers, respectively

for(sample=0;sample<blockSize;sample++)
{
 // Copy the new sample into the state buffer and then
    circularly wrap stateIndex
 state[stateIndex++] = inPtr[sample]
 if (stateIndex >= N)
  stateIndex = 0;

 sum = 0.0f;
 for(i=0;i<N;i++)
  {
   sum += state[stateIndex++] * coeffs[N-i];
   if (stateIndex >= N)
       stateIndex = 0;
  }
 outPtr[sample] = sum;
}
```

The definitive Guide to ARM Cortex M3 and Cortex M4 Processors, Joseph Yiu, 3rd Edition, 2013

Adressage modulo hardware
Sur microcontrôleur dsPIC

Les adresses restent toujours dans la même zone, malgré l'incrémentation (un ++ à la fin du buffer ramène au début)



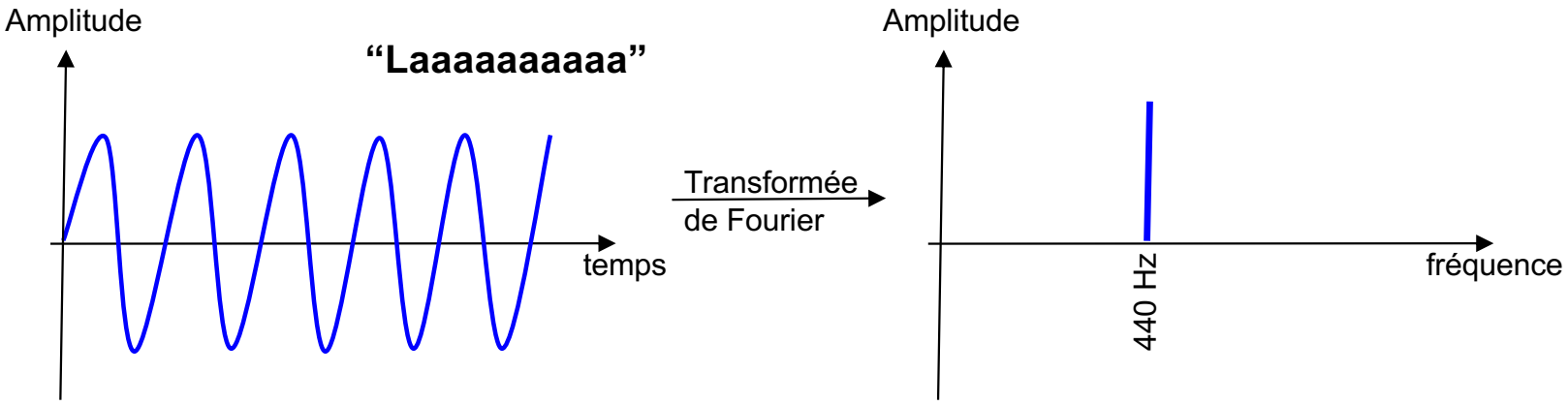

```
MOV    #0x1100,W0
MOV    W0,XMODSRT    ;set modulo start address
MOV    #0x1163,W0
MOV    W0,MODEND     ;set modulo end address
MOV    #0x8001,W0
MOV    W0,MODCON     ;enable W1, X AGU for modulo

MOV    #0x0000,W0    ;W0 holds buffer fill value

MOV    #0x1110,W1    ;point W1 to buffer

REPEAT  #0x31        ;fill the 50 buffer locations
MOV    W0,[W1++]     ;fill the next location
```

Analyse spectrale à l'aide de la transformée de Fourier

La transformée de Fourier

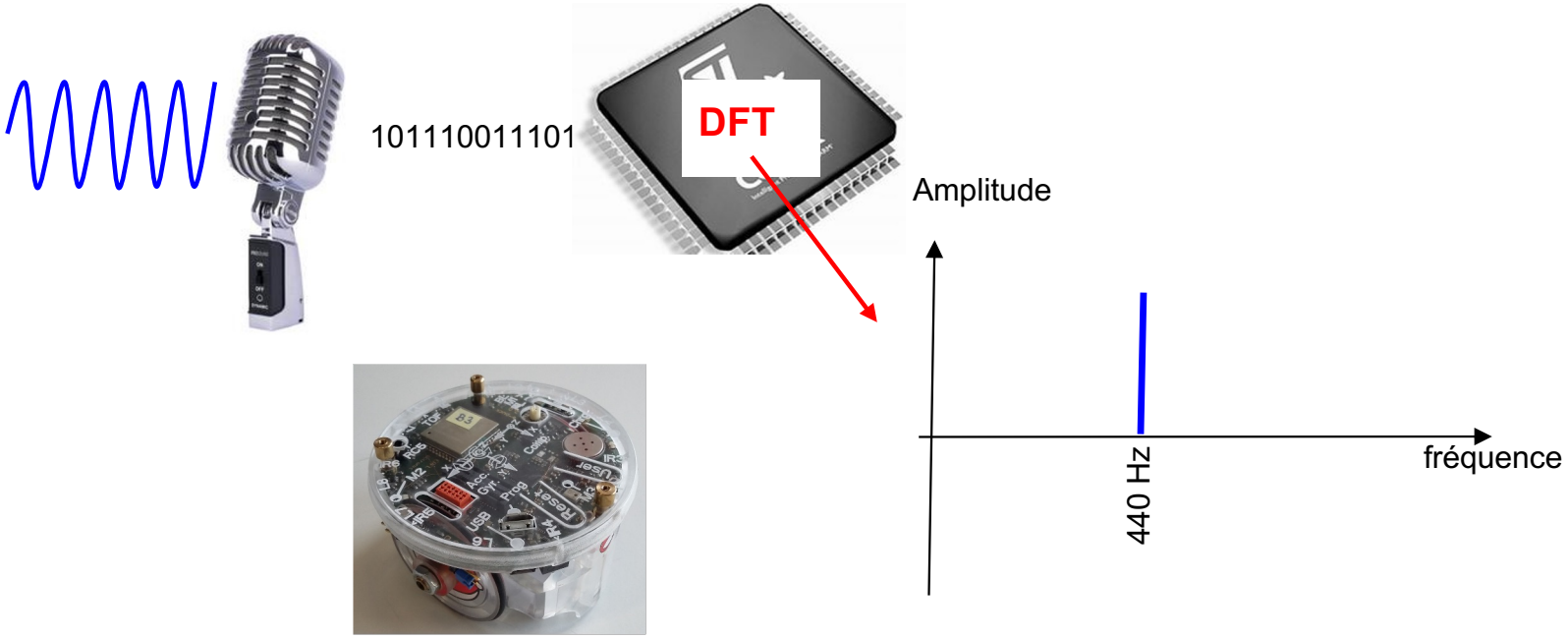
Dans le cas du traitement du son, une opération classique effectuée est la transformée de Fourier, pour passer du domaine temporel au domaine fréquentiel.

Une application typique est la détection de pics de fréquence dans le domaine fréquentiel pour déterminer la/les fréquences contenues dans un son (TP5).

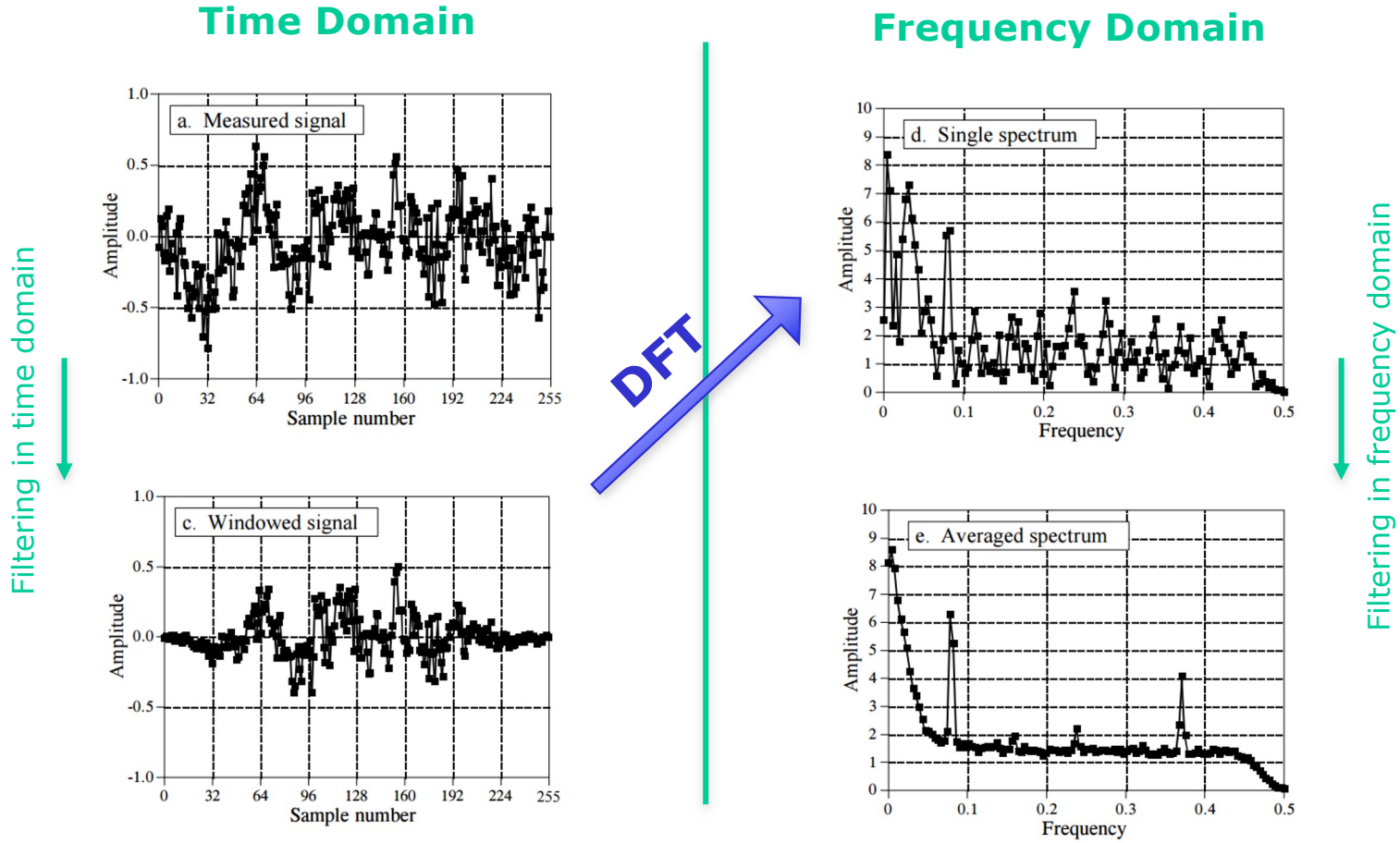


Transformée de Fourier discrète (DFT)

Dans le cadre d'un système embarqué, on travaille avec des signaux discrétisés, et donc on va appliquer une transformée de Fourier discrète.



Application de la DFT (Discrete Fourier Transform)



http://www.analog.com/media/en/technical-documentation/dsp-book/dsp_book_Ch9.pdf

DFT (Discrete Fourier Transform)

La DFT est définie par:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \ldots, N-1$$

($x_0 \ldots x_{N-1}$ nombres complexes en entrée
$X_0 \ldots X_{N-1}$ nombres complexes en sortie)

Principe: n'importe quel signal peut être re-créé par une somme de sinus et de cosinus

$$e^{ix} = cos(x) + isin(x)$$

DFT (Discrete Fourier Transform)

La DFT est définie par:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk} \qquad k = 0, \ldots, N-1$$

N^2 opérations

($x_0 \ldots x_{N-1}$ nombres complexes en entrée
$X_0 \ldots X_{N-1}$ nombres complexes en sortie)

Principe: n'importe quel signal peut être re-créé par une somme de sinus et de cosinus

$$e^{ix} = cos(x) + isin(x)$$

FFT (Fast Fourier Transform)

La Fast Fourier Transform est un algorithme de calcul rapide et efficace de la Discrete Fourier Transform

DFT: $$f_j = \sum_{k=0}^{n-1} x_k e^{-\frac{2\pi i}{n} jk} \qquad j = 0, \ldots, n-1.$$

ou en notation matricielle :

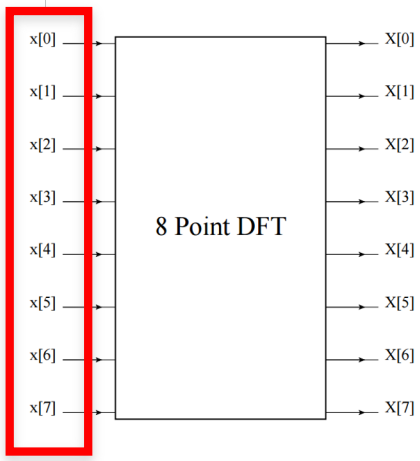
$$\begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{pmatrix}, w = e^{-\frac{2\pi i}{n}}$$

Principe de la FFT -> Modifier les opérations pour diminuer le nombre total d'opérations

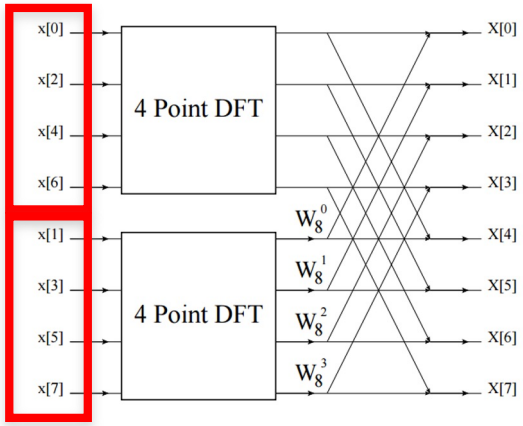
FFT (Fast Fourier Transform)

L'idée est de casser l'algorithme DFT en des sous-algorithmes DFT, et, avec une permutation de certains coefficients, cela réduira le nombre total d'opérations

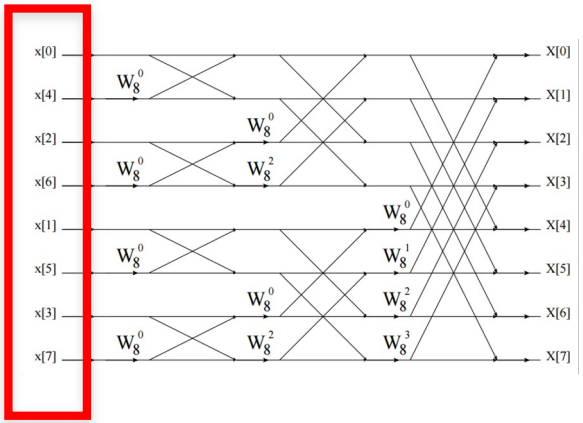
$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi nk/N}, \quad k = 0, 1, ..., N-1$$



DFT : ~N^2 opérations



DFT décomposée en deux sous DFT : <N^2 opérations



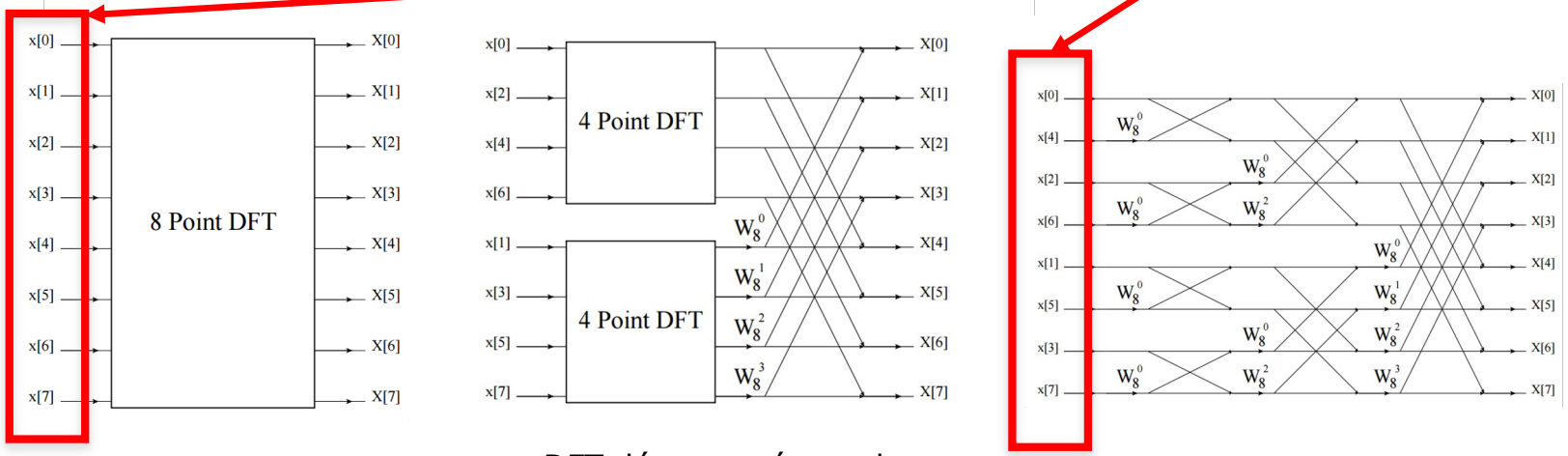
FFT: Nlog(N) opérations

FAST FOURIER TRANSFORMS ON A DISTRIBUTED DIGITAL SIGNAL PROCESSOR, Omar Sattari, University of California, 2002

FFT (Fast Fourier Transform)

L'idée est de casser l'algorithme DFT en des sous-algorithmes DFT, et, avec une permutation de certains coefficients, cela réduira le nombre total d'opérations

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi nk/N}, \quad k = 0, 1, ..., N-1$$



DFT : ~N^2 opérations

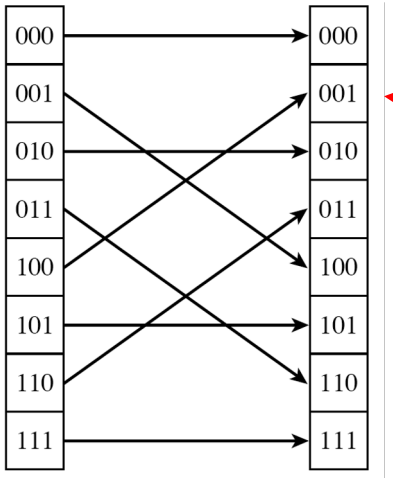
DFT décomposée en deux sous DFT : <N^2 opérations

FFT: Nlog(N) opérations

FAST FOURIER TRANSFORMS ON A DISTRIBUTED DIGITAL SIGNAL PROCESSOR, Omar Sattari, University of California, 2002

FFT: principe

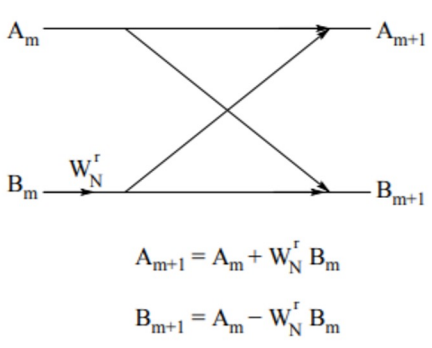
Passage de N^2 à $N \log N$ nombre d'opérations par la décomposition en FFT de taille plus petite et de façon itérative, jusqu'à atteindre la taille de 1. Cette décomposition requiert de permuter les entrées ou les sorties, selon l'usage.
L'ordre des nombres est changé suivant le schéma suivant, qui équivaut à une inversion des bits d'adressage:



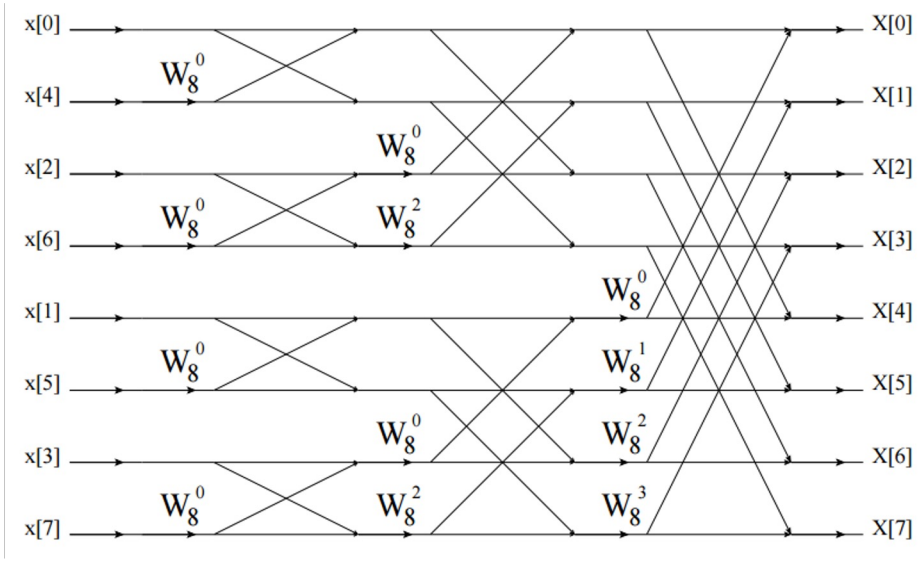
Notez l'inversion des bits d'adressage!

FFT: "butterfly" (existe en mode d'adressage)

La FFT consiste à effectuer un reverse des bits et des opérations papillons "butterfly":



1 opération "butterfly"



L'algorithme FFT en trois étapes avec chacunes 4 opérations "butterfly"

FFT: implémentation

Découpage “butterfly” aussi réalisées par des Tables.

Ici pour aller jusqu’à 0x800 on commence par 0x400, puis on coupe en moitié (0x200, 0x600), puis en Moitié (0x100, 0x500, 0x300, 0x700) etc.

```
/*
* @brief  Table for bit reversal process
*/
const uint16_t armBitRevTable[1024] = {
   0x400, 0x200, 0x600, 0x100, 0x500, 0x300, 0x700, 0x80, 0x480, 0x280,
   0x680, 0x180, 0x580, 0x380, 0x780, 0x40, 0x440, 0x240, 0x640, 0x140,
   0x540, 0x340, 0x740, 0xc0, 0x4c0, 0x2c0, 0x6c0, 0x1c0, 0x5c0, 0x3c0,
   0x7c0, 0x20, 0x420, 0x220, 0x620, 0x120, 0x520, 0x320, 0x720, 0xa0,
   0x4a0, 0x2a0, 0x6a0, 0x1a0, 0x5a0, 0x3a0, 0x7a0, 0x60, 0x460, 0x260,
   0x660, 0x160, 0x560, 0x360, 0x760, 0xe0, 0x4e0, 0x2e0, 0x6e0, 0x1e0,
   0x5e0, 0x3e0, 0x7e0, 0x10, 0x410, 0x210, 0x610, 0x110, 0x510, 0x310,
   0x710, 0x90, 0x490, 0x290, 0x690, 0x190, 0x590, 0x390, 0x790, 0x50,
   0x450, 0x250, 0x650, 0x150, 0x550, 0x350, 0x750, 0xd0, 0x4d0, 0x2d0,
   0x6d0, 0x1d0, 0x5d0, 0x3d0, 0x7d0, 0x30, 0x430, 0x230, 0x630, 0x130,
   0x530, 0x330, 0x730, 0xb0, 0x4b0, 0x2b0, 0x6b0, 0x1b0, 0x5b0, 0x3b0,
   0x7b0, 0x70, 0x470, 0x270, 0x670, 0x170, 0x570, 0x370, 0x770, 0xf0,
   0x4f0, 0x2f0, 0x6f0, 0x1f0, 0x5f0, 0x3f0, 0x7f0, 0x8, 0x408, 0x208,
   0x608, 0x108, 0x508, 0x308, 0x708, 0x88, 0x488, 0x288, 0x688, 0x188,
   0x588, 0x388, 0x788, 0x48, 0x448, 0x248, 0x648, 0x148, 0x548, 0x348,
   0x748, 0xc8, 0x4c8, 0x2c8, 0x6c8, 0x1c8, 0x5c8, 0x3c8, 0x7c8, 0x28,
   0x428, 0x228, 0x628, 0x128, 0x528, 0x328, 0x728, 0xa8, 0x4a8, 0x2a8,
   0x6a8, 0x1a8, 0x5a8, 0x3a8, 0x7a8, 0x68, 0x468, 0x268, 0x668, 0x168,
   0x568, 0x368, 0x768, 0xe8, 0x4e8, 0x2e8, 0x6e8, 0x1e8, 0x5e8, 0x3e8,
   0x7e8, 0x18, 0x418, 0x218, 0x618, 0x118, 0x518, 0x318, 0x718, 0x98,
```

https://github.com/ARM-software/CMSIS/blob/master/CMSIS/DSP_Lib/Source/CommonTables/arm_common_tables.c

FFT: implémentation

Programmation en C: ARM fournit de nombreuses librairies permettant d'effectuer des FFT sur des entiers (en utilisant la DSP) et des flottants (en utilisant la FPU)

Dans le TP5, utilisation de la FFT, mais en utilisant la FPU (choix d'implémentation):

```
79 /*
80  *   Wrapper to call a very optimized fft function provided by ARM
81  *   which uses a lot of trick to optimized the computations
82  */
83 void doFFT_optimized(uint16_t size, float* complex_buffer){
84     if(size == 1024)
85         arm_cfft_f32(&arm_cfft_sR_f32_len1024, complex_buffer, 0, 1);
86 }
87
```

Effectue la FFT

FFT: implémentation

Fonctions trigonométriques réalisées par des tables

https://github.com/ARM-software/CMSIS/blob/master/CMSIS/DSP_Lib/Source/CommonTables/arm_common_tables.c

```
/**
* \par
* Example code for Floating-point Twiddle factors Generation:
* \par
* <pre>for(i = 0; i< N/; i++)
* {
*       twiddleCoef[2*i]= cos(i * 2*PI/(float)N);
*       twiddleCoef[2*i+1]= sin(i * 2*PI/(float)N);
* } </pre>
* \par
* where N = 32  and PI = 3.14159265358979
* \par
* Cos and Sin values are in interleaved fashion
*
*/
const float32_t twiddleCoef_32[64] = {
    1.000000000f,  0.000000000f,
    0.980785280f,  0.195090322f,
    0.923879533f,  0.382683432f,
    0.831469612f,  0.555570233f,
    0.707106781f,  0.707106781f,
    0.555570233f,  0.831469612f,
    0.382683432f,  0.923879533f,
    0.195090322f,  0.980785280f,
    0.000000000f,  1.000000000f,
   -0.195090322f,  0.980785280f,
```

Récapitulation

Utilisation de ASM, C et C++/python

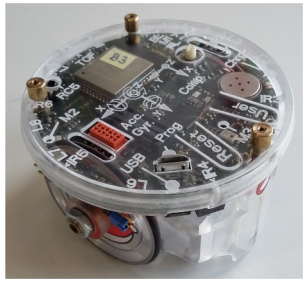
En général:

- Assembleur pour les microcontrôleurs avec des tâches peu complexes et visant faible taille / consommation. Code de startup. Aussi dans le cas d'optimisation de code sur architectures spéciales (DSP, FPU).
- C++ ou python pour des PC avec interface graphique, pour des projets complexes purement informatiques. (suite du cours)
- C pour ce qui se trouve entre deux…

Les frontières sont flexibles et dictées par le bon sens et l'expérience des personnes impliquées.

Utilisation de ASM, C et C++/python

C++ / python



C

Assembleur

Utilisation de ASM, C et C++/python

Exemple 1:

Projet: Code de startup pour le microcontrôleur STM32F4

Language: Assembleur

Motivation: Code proche de la machine car configure l'initialisation de la table des vecteurs des interruptions, Stack Pointer, Program Pointer etc. Code n'ayant pas besoin d'être recompilé et loin de l'utilisateur de la plateforme.

Utilisation de ASM, C et C++/python

Exemple 2:

Projet: Calcul de la FFT pour un traitement de son

Language: C / Assembleur

Motivation: Assembleur pour exploiter les mécanismes DSP ou FPU du processeur ARM Cortex-M4, C pour que ce soit multi-platforme.

Utilisation de ASM, C et C++/python

Exemple 3:

Projet: Régulateur PID pour un robot e-puck2

Language: C

Motivation: Doit être rapide d'exécution, mais aussi intéressant d'être multi-platforme, avec des paramètres pouvant facilement être modifiés par l'utilisateur.

Utilisation de ASM, C et C++/python

Exemple 4:

Projet: Monitor pour visualisation de données e-puck2

Language: C++ ou python

Motivation: Pour les parties graphiques il existe beaucoup de code, librairies, etc, et l'avantage de travailler avec du code orienté objet (suite du cours)

Consigne pour les miniprojets

Programme (rappel)

Faire encore un TP pour se roder et passer en revue les différents capteurs, puis test et miniprojet

Date					
18.Feb.25		Cours 1	**08.Apr.25**		8:15 test valable pour 40%
20.Feb.25		TPIntro	10.Apr.25		miniprojet
25.Feb.25		Cours 2	**15.Apr.25**		miniprojet
27.Feb.25		TP1	17.Apr.25		miniprojet
04.Mar.25		Cours 3	**22.Apr.25**		Vacances de Pâques
06.Mar.25		TP2	24.Apr.25		Vacances de Pâques
11.Mar.25		Cours 4	**29.Apr.25**		miniprojet
13.Mar.25		TP3	01.May.25		miniprojet
18.Mar.25		Cours 5	**06.May.25**		miniprojet
20.Mar.25		TP4	08.May.25		miniprojet
25.Mar.25		Cours 6	**13.May.25**		miniprojet
27.Mar.25		TP5	15.May.25		miniprojet - deadline 23h
01.Apr.25		8:15 test à blanc	**20.May.25**		présentations miniprojet
03.Apr.25		miniprojet	22.May.25		présentations miniprojet
			27.May.25		présentations miniprojet
			29.May.25		ascension

Miniprojets

Rendu du rapport + code: jeudi 15 mai, 23h00

Présentations: vendredi 16 mai -> mercredi 28 mai

Choix de l'heure de passage: Google Sheet sur moodle, publié le 1 mai. Inscrivez vous par groupe (G01, G56, etc.).

Présentations par zoom.

Méthode de travail :

La programmation à deux améliore l'apprentissage

Répartissez-vous le travail

Utilisez les outils de gestion du code (versions) partagé (git).
Ceci est un critère de l'évaluation

Donnée:

Le but du miniprojet est de partir sur la base des éléments que vous avez vu lors des TPs 1-5 pour créer plusieurs tâches plus complexes à résoudre par le robot e-puck2.

La donnée complète est sur moodle

Donnée:

Vous êtes libres de déterminer vous-même les tâches que le robot doit effectuer, et donc la forme de la démonstration de votre programme. Les contraintes sont les suivantes :

1) Le projet doit être fait sur la base de la librairie e-puck2_main-processor vue lors des TPs 4-5.

2) Vous devez obligatoirement utiliser les éléments suivants du robot e-puck2 dans votre projet :

   a. **Les deux moteurs pas-à-pas**. Par exemple Régulation PID, odométrie précise, forme géométrique, etc.
   b. **Un des capteurs de distance** (Capteurs de proximités infrarouges ou capteur Time-of-Flight). Par exemple détection d'obstacle petite ou grande distance.
   c. **Un capteur parmi ceux que vous avez investigué durant les TPs 3-5, donc un capteur parmi : la caméra, les micros, l'IMU**. Par exemple avec la caméra : détection d'objet, suivi de lignes. Par exemple avec les micros : détection de sons sur la base de l'amplitude et/ou fréquence. Par exemple avec l'accéléromètre, détection d'un plan incliné, d'un choc. Par exemple avec le gyroscope, détection du mouvement d'un plan incliné.

Donnée:

3) Chaque capteur/actuateur doit être utilisé/géré dans une Thread, à l'image de leur utilisation dans les TPs 4-5. La création de threads doit suivre les besoins, mais toujours de manière à respecter les taux de rafraichissements des capteurs et en faisant un usage intelligent des outils disponibles dans ChibiOS (messagebus, sémaphores, etc.).

4) Le code doit être rendu sous la forme d'une librairie (avec divers .c/.h) qui s'intègre avec la librairie e-puck2_main-processor que vous avez utilisé lors des Travaux pratiques.

Voici un exemple de projet : Le robot e-puck2 reproduit les mouvements d'une balle sur un plan incliné en utilisant l'accéléromètre comme détecteur de la direction du vecteur gravité. Le robot "rebondit" lorsqu'il détecte un obstacle avec ses capteurs de proximité infrarouges.

Miniprojets: travailler en groupe

On va vous demander un suivi de votre travail de groupe:

Weekly group self-assessment

Print Blank

Page 1

Planning

1 * We identified major deadlines to be met

Yes No

2 * We listed tasks to be done

Yes No

3 * We identified dependencies between tasks

Yes No

Forme du rendu:

Le rendu du miniprojet sera composé de:

- Une presentation (avec slides de preference, pas obligatoire) de **3** minutes
- Une démonstration live (pas de vidéos) du programme final, d'une durée de **2** minutes
- Un rapport de 3-4 pages (longueur indicative) qui donne un aperçu de la méthode de travail, des analyses, la conception du logiciel et des résultats obtenus. Une section doit donner un retour sur votre méthode de travail de groupe.
- Le code structuré et commenté
- Une discussion sur le projet lors du rendu/presentation

+ demo + présentation = durée totale de passage ~25 minutes

Critères d'évaluation:

Les critères d'évaluations seront :

- Clarté et propreté du code source (respect de certaines conventions du C, des commentaires utiles, des valeurs définies, des fonctions et variables avec des noms clairs, etc.)
- Clarté et propreté du rapport (Numéros de sections, légendes numérotées aux figures, numéros de page, axes aux graphiques, citations et références)
  Un point enlevé sur la note finale du projet (note entière) pour des erreurs de ce type!
- Efficacité du code en termes d'utilisation des ressources (temps, mémoire, tâches).
- Pratique d'utilization de GitHub.
- Pratique de travail de groupe.
- Originalité de la démonstration.

Miniprojets

Commencez par comprendre et structurer!!!

Code: attention aux warnings, temps d'exécution, ressources utilisées, etc.

Rapport:

- seulement ce qui n'est pas dans le code, comme structure générale, tests initiaux, graphes de capteurs, etc
- **Respectez le format (sections numérotées, figures et tables numérotées avec légendes, axes sur les graphes, références)**

GCtronic award:

Un prix sera décerné au miniprojet ayant obtenu la meilleure note sur la base de ces critères d'évaluation

Examen ***valable pour 40% de la note du cours***

Examen à blanc: mardi 1 avril

CM 1 2, CM 1 4, CM 1 5 et CO5-6 à 8:15

Examen : mardi 8 avril

CM 1 2, CM 1 4, CM 1 5 et CO5-6 à 8:15

MICRO-315 / 17 March - 23 March / Modalité de l'examen mid-term / Responses

CHOICE

Modalité de l'examen mid-term

125 étudiants inscrits au cours

Choice Settings Responses More

Responses

45 n'ont pas répondu!

Separate groups All participants

Choice options	Ordinateur personnel apporté par l'étudiant.e dans auditoire	Ordinateur de l'EPFL dans une salle informatique
Number of responses	67	13

Examen

Format:

Questions sur moodle, similaires que dans les quiz de chaque semaine

Support:

Aucune documentation autorisée

Règles générales:

Absences justifiées (certificat médical) rattrapées par une interrogation orale, pas de visite des toilettes possible. Apportez votre carte d'étudiant.

Notation de l'examen

A chaque question est associé un nombre de points qui est affiché avec la question.

- **Questions vrai-faux:** si juste, on obtient le nombre de point, si c'est faux on a 0 points.
- **Questions avec réponse numérique:** si juste, on obtient le nombre de point, si c'est faux on a 0 points.
- **Questions avec plusieurs réponses possibles:** Les réponses justes se partagent à part égale les points de la question, les réponses fausses se partagent à part égale le même nombre de points, mais négatifs. Le total est saturé vers le bas à 0.

A la fin les points sont sommés et la note est définie par le nombre de point obtenus, divisé par le nombre de points maximaux, multiplié par 6. La note est saturée vers le bas à 1.0.

TP de cette semaine

Application et compréhension de la FFT sur un signal sonore.

Exploitation de la sortie en amplitude de la FFT pour détection de fréquences.

Programmation sur l'e-puck d'un contrôle par son.

1 / 4 | Next > | Last Respondent >>
<<< List of responses | Print this Response

Respondent: **- Anonymous -**

Evaluation du cours et du TP

Ceci est un questionnaire anonyme, donnez votre avis franchement, merci.

1 * Donnez votre évaluation du cours de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	○	◉	○	○
Forme	○	○	○	○	◉	○

2 * Donnez votre évaluation du TP de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	◉	○	○	○
Forme	◉	○	○	○	○	○

3 Vos commentaires en détail (si nécessaire):

Je n'ai pas les ressources nécessaires pour effectuer les TPs. Le cours ne m'aide pas à effectuer les TPs. Une préparation au TP avant chaque TP à faire à la maison serait à mon avis nécessaire. Le guidage donné par les tasks est insuffisant.

<< First Respondent | < Previous | 2 / 4 | Next > | Last Respondent >>
<<< List of responses | Print this Response

Respondent: **- Anonymous -**

Evaluation du cours et du TP

Ceci est un questionnaire anonyme, donnez votre avis franchement, merci.

1 * Donnez votre évaluation du cours de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	○	○	◉	○
Forme	○	○	○	○	◉	○

2 * Donnez votre évaluation du TP de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	◉	○	○	○
Forme	○	○	◉	○	○	○

3 Vos commentaires en détail (si nécessaire):

malgré toute la bonne volonté possible, il est impossible d'avancer seul pour les TPs. Il est constamment nécessaire d'appeler des assistants pour des informations complémentaires et pour nous réexpliquer les concepts utiles, où trouver les infos dans les datasheets ou nous expliquer comment implémenter en C les concepts...

<< First Respondent | < Previous | 3 / 4 | Next > | Last Respondent >>
<<< List of responses | Print this Response

Respondent: **- Anonymous -**

Evaluation du cours et du TP

Ceci est un questionnaire anonyme, donnez votre avis franchement, merci.

1 * Donnez votre évaluation du cours de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	○	○	◉	○
Forme	○	○	○	○	◉	○

2 * Donnez votre évaluation du TP de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	◉	○	○	○	○
Forme	○	○	○	◉	○	○

3 Vos commentaires en détail (si nécessaire):

A mon goût, TP beaucoup trop long. Même si l'on sait que la solution est assez simple, c'est très dur de comprendre ce qu'il faut faire avec le buffer, comment prendre les éléments, comment les traiter. Donner des exemples d'applications (même simples) dans le GitHub pourrait je pense grandement aider. On se retrouve alors à perdre des heures sur des étapes relativement simples et on est frustrés de ne pas pouvoir avancer et avoir le soutient d'assistants pour les tâches suivantes + manque d'effectif ce jour là. De façon plus globale je trouve que le système d'assistants fonctionne très bien. Les TPs sont globalement beaucoup trop longs, c'est dommage.

<< First Respondent | < Previous | **4 / 4**
<<< **List of responses** | Print this Response

Respondent: **- Anonymous -**

Evaluation du cours et du TP

Ceci est un questionnaire anonyme, donnez votre avis franchement, merci.

1 * Donnez votre évaluation du cours de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	○	○	○	●
Forme	○	○	○	○	○	●

2 * Donnez votre évaluation du TP de la semaine (1 nul, 6=excellent)

	1	2	3	4	5	6
Contenu	○	○	○	○	○	●
Forme	○	○	○	○	○	●

3 Vos commentaires en détail (si nécessaire):

Les informations pour le TP étaient nécessaires mais non suffisantes :) Le chatbot est très utile pour combler les trous dans la compréhension.

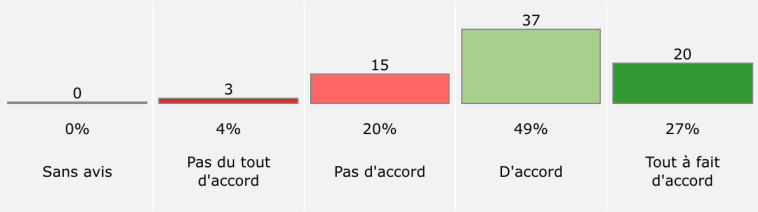
EPFL

Systèmes embarqués et robotique

Feedback (retours indicatifs)

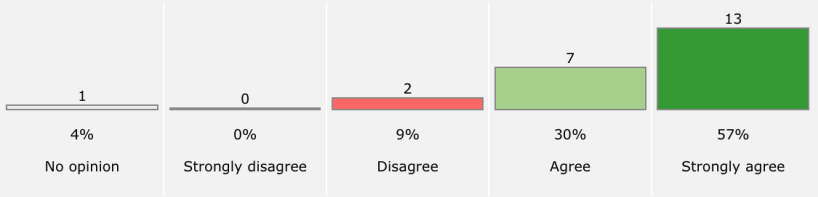
Année	2021-2022
Matière	Systèmes embarqués et robotique
Questionnaire	Retour indicatif des enseignements (dès 2020-2021)
Nb Inscrit	140
Nb Répondu	75

Dans l'ensemble, je pense que ce cours est bon.



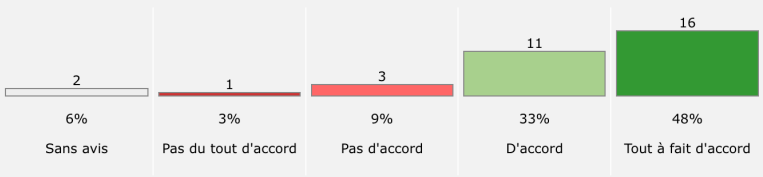
Year	2023-2024
Course	Embedded Systems and Robotics
Questionnaire	Indicative feedback of teaching (since 2022-2023)
Nb Registered	130
Nb Answered	23

The running of the course enables my learning and an appropriate class climate



Année	2022-2023
Matière	Systèmes embarqués et robotique
Questionnaire	Retour indicatif des enseignements (dès 2022-2023)
Nb Inscrit	137
Nb Répondu	33

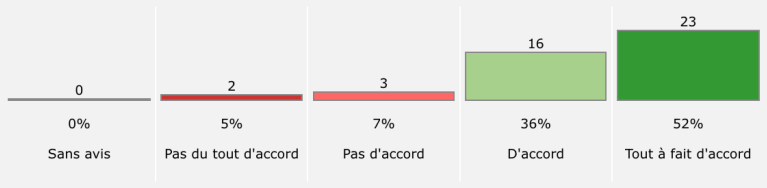
Le déroulement du cours permet ma formation et un climat de classe approprié



Année	2024-2025
Matière	Systèmes embarqués et robotique
Questionnaire	Retour indicatif des enseignements (dès 2022-2023)
Nb Inscrit	125
Nb Répondu	44

Participation : 35%

Le déroulement du cours permet ma formation et un climat de classe approprié



88% OK

✅ **Points forts du cours:**

- **Implication pédagogique** :
  - Enseignant et assistants très investis, disponibles, ouverts aux critiques et aux feedbacks réguliers.
  - Atmosphère agréable durant les cours et TP, malgré la difficulté du contenu.
- **Bonne organisation générale** :
  - Cours bien structurés, slides clairs et outils bien pensés (chatbot, formulaire pour appeler les assistants, enregistrements des cours).
  - TP bien guidés et intéressants.
- **Outils pédagogiques appréciés** :
  - Utilisation de quiz hebdomadaires utiles pour se préparer à l'examen.
  - Accès à un chatbot personnalisé, très apprécié pour la théorie.
- **Motivation suscitée** :
  - Cours qui donne envie d'approfondir la robotique.
  - Approche durable évoquée dans les cours saluée par certains.

✗ Points faibles du cours:

- **Difficulté globale élevée** :
  - Charge cognitive importante (robotique, datasheets, programmation à apprendre en parallèle).
  - Matière perçue comme difficile à digérer.
- **TP trop longs et parfois confus** :
  - Durée des TP sous-estimée, certains groupes avancent peu malgré leurs efforts.
  - Manque de guidance claire dans certains TP (besoin de plus d'instructions, de commentaires dans les solutions, etc.).
  - Informations parfois trop dispersées (multiples PDF).
- **Problèmes d'introduction aux outils** :
  - Introduction à Git jugée trop abrupte et source de décrochage.
  - Installation des outils difficile et frustrante pour certains.
- **Lien cours-TP perfectible** :
  - Certains ressentent un décalage entre les cours théoriques et leur utilité directe pour les TP.
  - Difficulté à savoir ce qui sera attendu à l'examen malgré les quiz.