

ELE3312
Microcontrôleurs et applications
Laboratoire 6

Auteur : Jean Pierre David

Introduction

Grâce à la puissance toujours grandissante des microcontrôleurs, les applications qui tournent sur ces circuits sont de plus en plus sophistiquées. La plupart des applications partagent des fonctionnalités communes, qui ont été développées une fois pour toute et mises à disposition dans des bibliothèques. Ces bibliothèques permettent aux développeurs de gagner beaucoup de temps. En outre, le code est généralement plus performant, plus lisible, et plus facilement réutilisable. En principe, il contient aussi moins d'erreur qu'un code écrit à partir de rien. Toutefois, paradoxalement, cela peut aussi le rendre plus vulnérable aux attaques car les bogues inconnus sont communs à toutes les applications qui utilisent ces bibliothèques. Dans ce laboratoire vous allez approfondir la bibliothèque HAL et découvrir la bibliothèque CMSIS.

Objectifs

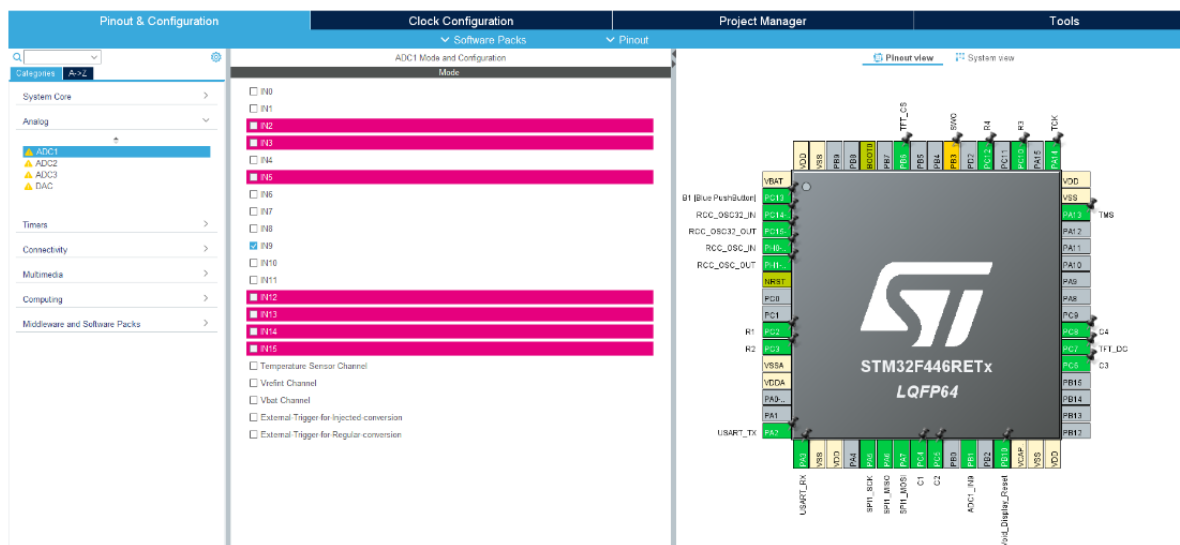
1. Découvrir et utiliser la documentation du HAL
2. Découvrir la bibliothèque CMSIS, sa documentation, et les utiliser

1^{ère} partie : préparation à la maison

Ce laboratoire se fonde sur le laboratoire 4. Commencez par faire une copie de son répertoire et ouvrez le projet STM32cube. Vous devriez donc déjà avoir la configuration pour les broches de l'écran et le clavier.

Expérience 1

1. Ouvrez le projet STM32cube et activez l'ADC1 sur le canal 9 (qui correspond à la broche PB1, une des rares broches qui ne soient pas encore utilisées par le clavier ou par le LCD). Voyez l'image ci-dessous :



2. Régénérez le code (Ctrl-Shift-G) et ouvrez le projet Keil correspondant.
3. Nettoyez le code de tout ce qui concernait le laboratoire 4.

4. Ajoutez les inclusions nécessaires et définissez les variables statiques comme illustré ci-dessous :

```
27  /* Private includes -----*/
28  /* USER CODE BEGIN Includes */
29
30
31  #include "ili9341.h"
32  #include "ili9341_gfx.h"
33  #include "stdio.h"
34  /* USER CODE END Includes */
35
36  /* Private typedef -----*/
37  /* USER CODE BEGIN PTD */
38
39  /* USER CODE END PTD */
40
41  /* Private define -----*/
42  /* USER CODE BEGIN PD */
43
44  /* USER CODE END PD */
45
46  /* Private macro -----*/
47  /* USER CODE BEGIN PM */
48
49  /* USER CODE END PM */
50
51  /* Private variables -----*/
52
53  /* USER CODE BEGIN PV */
54  ili9341_t *_screen;
55  float tab_value[256];
56  float FFT_value[256];
57  float abs_value[128];
58
59  /* USER CODE END PV */
60
61  /* USER CODE BEGIN Includes */
62  #include "ili9341.h"
63  #include "ili9341_gfx.h"
64  #include "stdio.h"
65  /* USER CODE END Includes */
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

5. Faites les initialisations nécessaires pour le `_screen` et l'ADC1 :

```

100  /* USER CODE END SysInit */
101
102  /* Initialize all configured peripherals */
103  MX_GPIO_Init();
104  MX_DMA_Init();
105  MX_SPI1_Init();
106  MX_USART2_UART_Init();
107  MX_ADC1_Init();
108  /* USER CODE BEGIN 2 */
109  _screen = ili9341_new(
110      &hspi1,
111      Void_Display_Reset_GPIO_Port, Void_Display_Reset_Pin,
112      TFT_CS_GPIO_Port, TFT_CS_Pin,
113      TFT_DC_GPIO_Port, TFT_DC_Pin,
114      isoLandscape,
115      NULL, NULL,
116      NULL, NULL,
117      itsNotSupported,
118      itnNormalized);
119
120  ili9341_fill_screen(_screen, ILI9341_BLACK);
121  ili9341_text_attr_t text_attr = {&ili9341_font_11x18, ILI9341_WHITE, ILI9341_BLACK, 0, 0};
122
123
124  /* USER CODE END 2 */

/* USER CODE BEGIN 2 */
    _screen = ili9341_new(&hspi1, Void_Display_Reset_GPIO_Port,
Void_Display_Reset_Pin, TFT_CS_GPIO_Port, TFT_CS_Pin, TFT_DC_GPIO_Port, TFT_DC_Pin,
isoLandscape, NULL, NULL, NULL, NULL, itsNotSupported, itnNormalized);

ili9341_fill_screen(_screen, ILI9341_BLACK);
ili9341_text_attr_t text_attr = {&ili9341_font_11x18, ILI9341_WHITE, ILI9341_BLACK, 0, 0};
/* USER CODE END 2 */

```

6. Écrivez le code suivant dans la boucle main pour réaliser un oscilloscope miniature qui va lire la tension du ADC1 (sur l'entrée du port PB1) et l'afficher sur le LCD :

```

126  /* Infinite loop */
127  /* USER CODE BEGIN WHILE */
128  while (1)
129  {
130      /* USER CODE END WHILE */
131
132      /* USER CODE BEGIN 3 */
133      float scale = 120.0/4095.0;
134      for (int x=0;x<256;x++) {
135          HAL_ADC_Start(&hadc1);
136          HAL_ADC_PollForConversion(&hadc1,100);
137          float value = tab_value[x] = HAL_ADC_GetValue(&hadc1)*scale;
138          char buffer[15] = {0};
139          sprintf(buffer, "Value : %-6.2f",value);
140          ili9341_draw_string(_screen, text_attr,buffer);
141          ili9341_draw_pixel(_screen, ILI9341_BLUE, x,(int) (120-value));
142          HAL_Delay(100);
143      }
144
145      HAL_Delay(5000);
146  }
147  /* USER CODE END 3 */
148  }

/* USER CODE BEGIN 3 */
float scale = 120.0/4095.0;
for (int x=0;x<256;x++) {
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,100);
    float value = tab_value[x] = HAL_ADC_GetValue(&hadc1)*scale;
    char buffer[15] = {0};
    sprintf(buffer, "Value : %-6.2f",value);
    ili9341_draw_string(_screen, text_attr,buffer);
    ili9341_draw_pixel(_screen, ILI9341_BLUE, x,(int) (120-value));
    HAL_Delay(100);
}
HAL_Delay(5000);

```

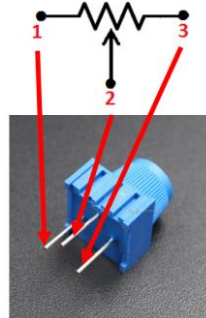
```

}
/* USER CODE END 3 */

```

7. Pour faire varier la tension sur le port d'entrée PB1, utilisez un potentiomètre comme vous l'aviez fait au laboratoire 1 (les broches externes sont connectées au GND et 3.3V alors que la broche centrale est connectée au port PB1).

ATTENTION, NE PAS UTILISER LE 5V. IL POURRAIT ENDOMMAGER VOTRE MICROCONTROLEUR.



8. Si tout fonctionne bien, vous devriez voir une trace d'oscilloscope qui suit parfaitement vos mouvements sur le potentiomètre.
9. Vous venez d'utiliser la puissance de la bibliothèque HAL. Assurez-vous de bien comprendre chacune des fonctions utilisées. La documentation du HAL est sur Moodle. **Le chargé de laboratoire pourrait vous demander de lui montrer ou se trouve telle ou telle fonction.**
10. Vous allez maintenant utiliser une autre bibliothèque : la bibliothèque CMSIS

Expérience 2

Alors que la bibliothèque HAL est régénérée chaque fois que vous utilisez STM32cube, parce qu'elle fait appel aux registres de configuration du microcontrôleur, la bibliothèque CMSIS est une bibliothèque C qui est externe, indépendante (ou presque) du matériel. Vous noterez tout de même la commande `#define ARM_MATH_CM4` en début de code qui permet à la bibliothèque de faire certaines optimisations en fonction du cœur du processeur qui est disponible. Rajoutez les lignes suivantes au début du main.c :

```

27  /* Private includes -----*/
28  /* USER CODE BEGIN Includes */
29  #define ARM_MATH_CM4
30  #include "arm_math.h"
31
32  #include "ili9341.h"
33  #include "ili9341_gfx.h"
34  #include "stdio.h"
35  /* USER CODE END Includes */

/* USER CODE BEGIN Includes */

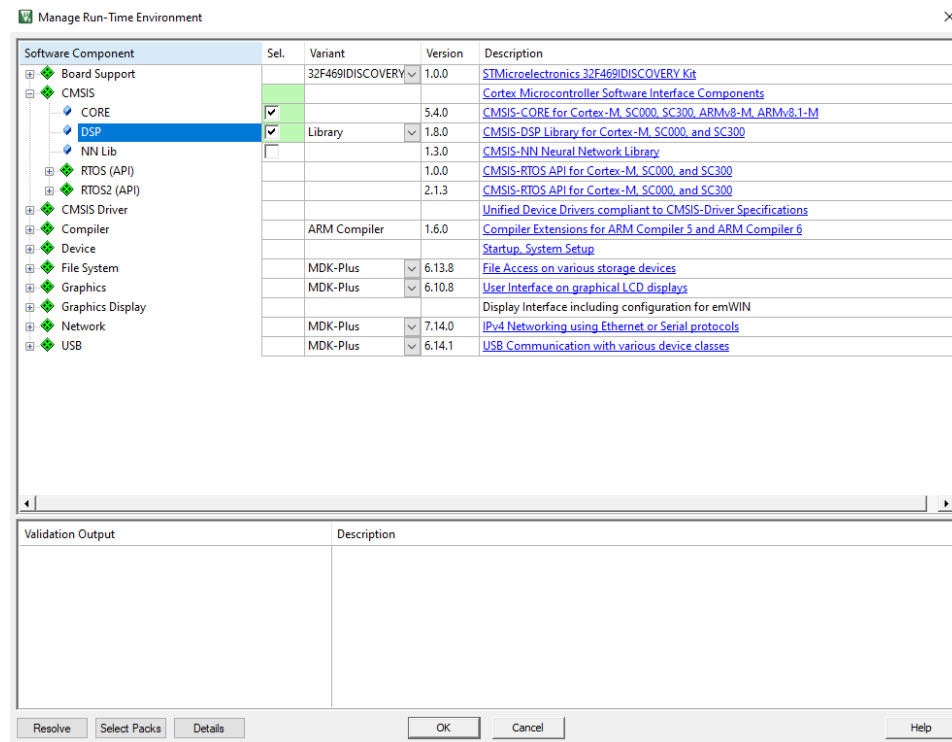
#define ARM_MATH_CM4
#include "arm_math.h"

#include "ili9341.h"
#include "ili9341_gfx.h"
#include "stdio.h"
/* USER CODE END Includes */

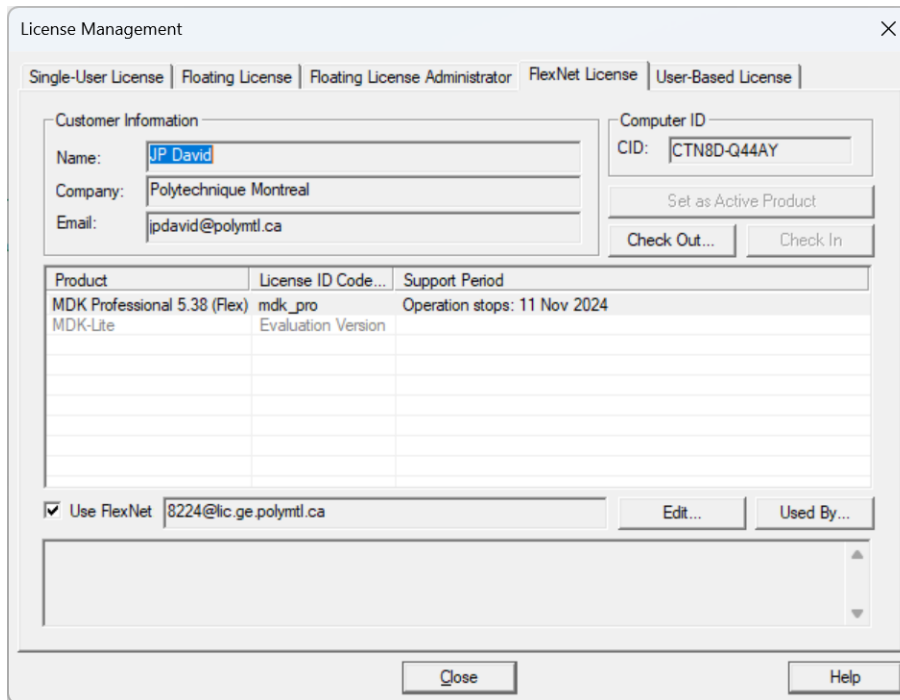
```

Nous allons utiliser la bibliothèque CMSIS pour calculer la transformée de fourrier du signal acquis avec l'oscilloscope.

1. Pour commencer, il faut déclarer que nous utilisons la bibliothèque dans le menu Project/Manage/Run-Time environment (cocher la case DSP et choisissez la variante «Library») :



2. Vous pouvez ouvrir la documentation de la librairie en cliquant sur l'hyperlien à droite de la case cochée. Assurez-vous d'y trouver l'information sur les fonctions qui vont être utilisées par la suite.
3. La bibliothèque nécessite un espace mémoire plus grand pour stocker votre programme. Il se peut que la licence gratuite de Keil ne fonctionne plus. Si c'est le cas, utilisez le VPN de Polytechnique et configurez la licence de Keil comme suit dans le menu File->Licence Management :



4. Écrivez le code suivant qui permet de faire la FFT du signal.

```

133      /* USER CODE BEGIN 3 */
134      float scale = 120.0/4095.0;
135      for (int x=0;x<256;x++) {
136          HAL_ADC_Start(&hadc1);
137          HAL_ADC_PollForConversion(&hadc1,100);
138          float value = tab_value[x] = HAL_ADC_GetValue(&hadc1)*scale;
139          char buffer[15] = {0};
140          sprintf(buffer, "Value : %-6.2f",value);
141          ili9341_draw_string(_screen, text_attr,buffer);
142          ili9341_draw_pixel(_screen, ILI9341_BLUE, x,(int) (120-value));
143          HAL_Delay(100);
144      }
145      arm_rfft_fast_instance_f32 fftInstance;
146      arm_rfft_fast_init_f32(&fftInstance, 256);
147      arm_rfft_fast_f32(&fftInstance, tab_value, FFT_value, 0);
148      arm_cmplx_mag_f32(FFT_value, abs_value, 128);
149
150      float max_value;
151      unsigned int max_index;
152      arm_max_f32(abs_value, 128, &max_value, &max_index);
153      scale = 120.0/max_value;
154      ili9341_fill_screen(_screen, ILI9341_BLACK);
155      for (int x=0;x<128;x++) {
156          float value = abs_value[x]*scale;
157          ili9341_draw_line(_screen, ILI9341_RED, 2*x,(int) (240-value), 2*x, 239);
158          ili9341_draw_line(_screen, ILI9341_RED, 2*x+1,(int) (240-value), 2*x+1, 239);
159          char bufferFFT[20] = {0};
160          sprintf(bufferFFT, "FFT : %-6.2f ",value);
161          ili9341_draw_string(_screen, text_attr,bufferFFT);
162      }
163      HAL_Delay(5000);
164  }
165      /* USER CODE END 3 */
/* USER CODE BEGIN 3 */
float scale = 120.0/4095.0;
for (int x=0;x<256;x++) {
    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1,100);
    float value = tab_value[x] = HAL_ADC_GetValue(&hadc1)*scale;
    char buffer[15] = {0};
    sprintf(buffer, "Value : %-6.2f",value);

```

```

        ili9341_draw_string(_screen, text_attr,buffer);
        ili9341_draw_pixel(_screen, ILI9341_BLUE, x,(int) (120-value));
        HAL_Delay(100);
    }
    arm_rfft_fast_instance_f32 fftInstance;
    arm_rfft_fast_init_f32(&fftInstance, 256);
    arm_rfft_fast_f32(&fftInstance, tab_value, FFT_value, 0);
    arm_cmplx_mag_f32(FFT_value, abs_value, 128);

    float max_value;
    unsigned int max_index;
    arm_max_f32(abs_value, 128, &max_value, &max_index);
    scale = 120.0/max_value;
    ili9341_fill_screen(_screen, ILI9341_BLACK);
    for (int x=0;x<128;x++) {
        float value = abs_value[x]*scale;
        ili9341_draw_line(_screen, ILI9341_RED, 2*x,(int) (240-value), 2*x, 239);
        ili9341_draw_line(_screen, ILI9341_RED, 2*x+1,(int) (240-value), 2*x+1,
239);

        char bufferFFT[20] = {0};
        sprintf(bufferFFT, "FFT : %-6.2f ",value);
        ili9341_draw_string(_screen, text_attr,bufferFFT);
    }
    HAL_Delay(5000);
}
/* USER CODE END 3 */

```

5. Testez votre oscilloscope en faisant appel à vos connaissances en traitement de signal. Essayez d'entrer un signal impulsionnel ou un signal périodique pour voir si sa transformée est cohérente.

Assurez-vous de bien comprendre le code en général et toutes les fonctions des bibliothèques utilisées en lisant leur documentation. Vous serez évalués en début de laboratoire.

2^{ème} partie : Laboratoire à réaliser en salle à Polytechnique

En utilisant la bibliothèque CMSIS, ajoutez une deuxième trace à l'oscilloscope qui affiche la tension après être passée dans un filtre passe-bas. La trace doit s'afficher en temps réel en même temps que la trace qui affiche le signal original.

Le chargé de laboratoire vous assignera une fréquence d'échantillonnage (approximative) F_s et la fréquence de coupure normalisée (entre 0 et 1) par rapport à la fréquence de Nyquist, soit $F_s/2$. Vous utiliserez un filtre FIR d'ordre 15 (16 coefficients appelés « TAPs »).

En résumé, on vous demande de :

- a) Calculer les 16 TAPs du FIR avec Matlab, Octave ou tout autre outil de traitement de signal
- b) Instancier et initialiser le filtre FIR de CMSIS dans Keil
- c) Dès qu'un nouvel échantillon arrive (soit un bloc de UN SEUL échantillon), l'envoyer dans le FIR et afficher la sortie du FIR

Les liens suivants pourraient vous être utiles :

<https://www.mathworks.com/help/signal/ref/fir1.html>

https://www.keil.com/pack/doc/CMSIS_Dev/DSP/html/group__FIRLPF.html

Et en particulier :

https://www.keil.com/pack/doc/CMSIS_Dev/DSP/html/arm_fir_example_f32_8c-example.html

Si vous n'avez pas accès à Matlab, vous pourriez utiliser le logiciel (gratuit et open source) Octave :

<https://octave.org/>

Ou encore sa version en ligne :

<https://octave-online.net/>

!!! Il vous est grandement conseillé de préparer cette partie AVANT le laboratoire !!!