# 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

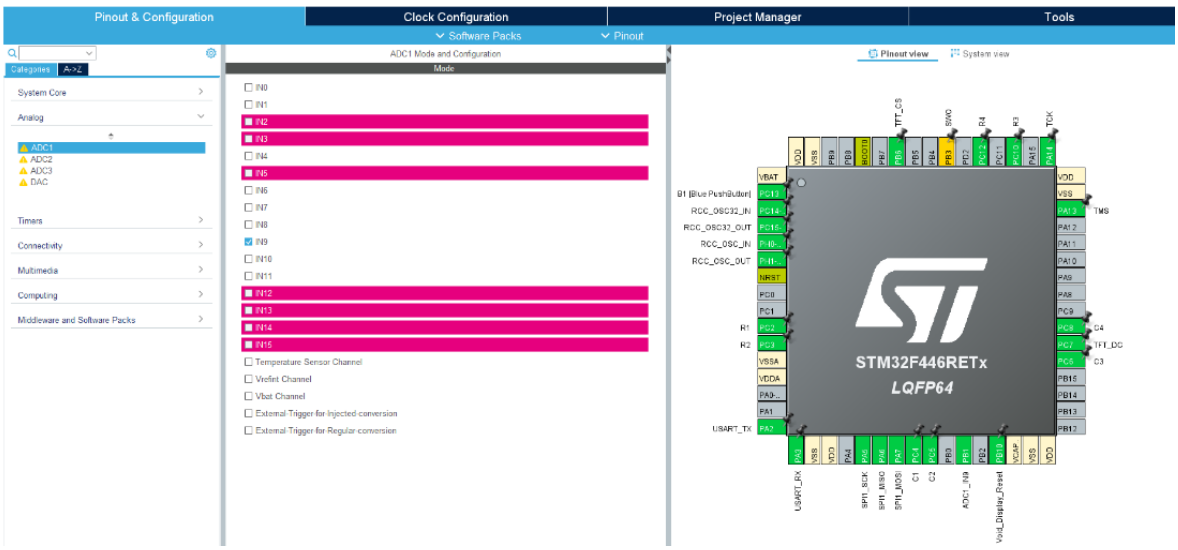
1. Découvrir et utiliser la documentation du HAL
2. Découvrir la bibliothèque CMSIS, sa documentation, et les utiliser

# 1ère 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 :

```
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */


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

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
ili9341_t *_screen;
float tab_value[256];
float FFT_value[256];
float abs_value[128];

/* USER CODE END PV */
```

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



/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
              ili9341_t *_screen;
              float tab_value[256];
              float FFT_value[256];
              float abs_value[128];

/* USER CODE END PV */
```

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

```
  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_SPI1_Init();
  MX_USART2_UART_Init();
  MX_ADC1_Init();
  /* 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 */
```

```
/* 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 :

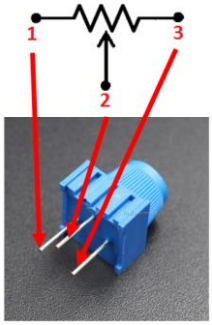
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* 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 */
}
```

```
  /* 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

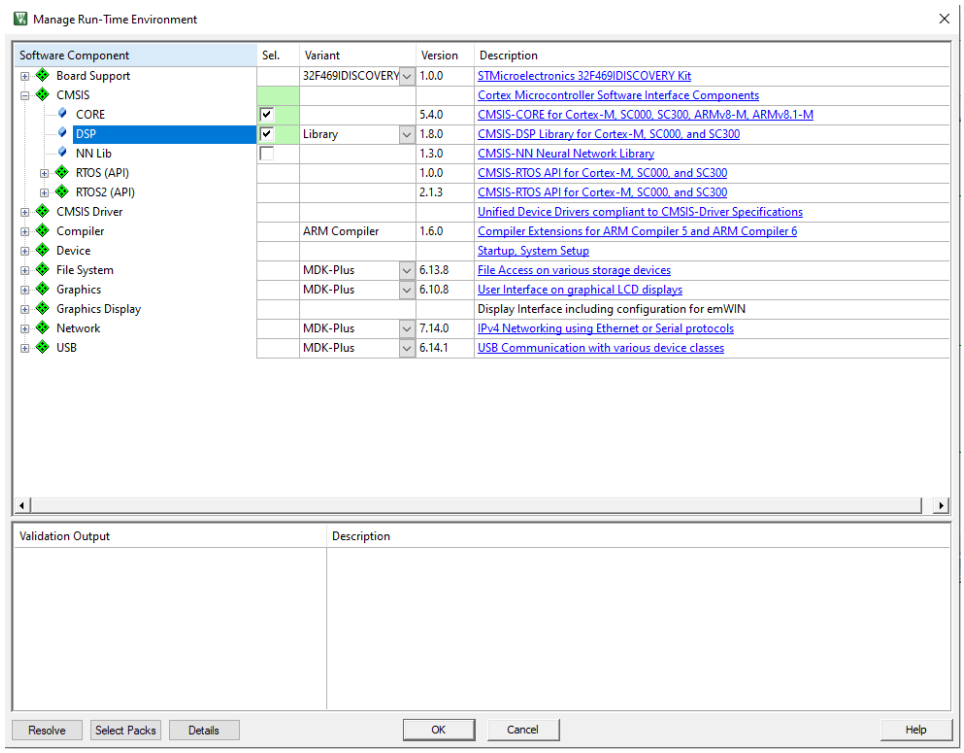
Alors que la bibliothèque HAL est régénérée chaque fois que vous utilisez STM32cube, parce qu'elle fait appel aux registres de configuration du microcontrôleur, la bibliothèque CMSIS est une bibliothèque C qui est externe, indépendante (ou presque) du matériel. Vous noterez tout de même la commande *#define 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 */
```

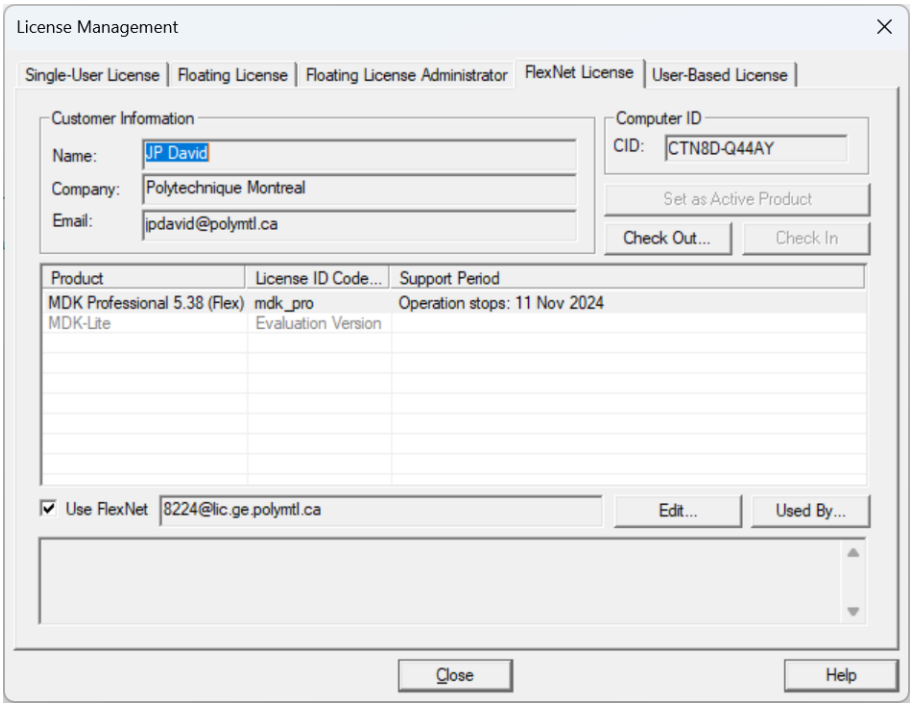
Nous allons utiliser la bibliothèque CMSIS pour calculer la transformée de fourrier du signal acquis avec l'oscilloscope.

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.

```
    /* 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 */
```

```
/* 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 librairies utilisées en lisant leur documentation. Vous serez évalués en début de laboratoire.

## 2ème 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) Fs et la fréquence de coupure normalisée (entre 0 et 1) par rapport à la fréquence de Nyquist, soit Fs/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 !!!