- **In a few months, you're going to forget most of the steps you went through when conducting the assignment.**
- **Submit documentation describing all main steps in your experiments.**
- **The text and diagrams can be sourced anywhere. Just state the source.**
- **Use your judgment whether to describe all steps or refer to another piece of documentation.**
- **Submit just 1 file. Can be PDF or DOCX.**
- **Easy full marks for:**
    - **clear and easy to follow descriptions**
    - **complete references**

This Report consists of a set of documentation which including:

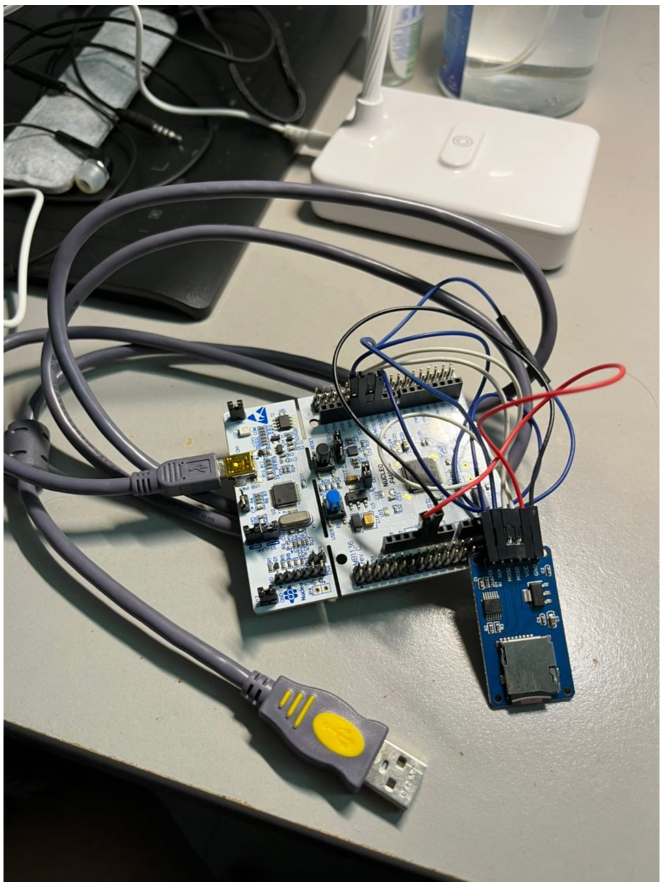
1) **Software and Tools setup**
2) **Configuration Steps**
3) **Steps for firmware development**
4) **Steps for hardware development**
5) **References**

Introduction

STM32F446RE is the main microcontroller asked by the instruction to do the DSP filtering in the project. Micro-SD card is a handy external memory device which we used to store the 32 bit single precision text file. We generated the (ECG) electrocardiogram signal from MATLAB and scrambled the clean signal with high frequency noise. The noisy ECG signal was copied to the text file in 32-bit floating point numbers format and saved in the SD card. The SD card adapter connected the SD card to the Nucleo board through the Serial Peripheral Interface (SPI) connection protocol. The connection is in full duplex mode as Nucleo-board acted as the master and SD card acted as the slave. The File Allocation Table File System (FATfs) was the middleware used to manage the data flow in SD card. The Universal Asynchronous Receiver Transmitter (UART) interface was used in our project to survellient the card mounting and read/write status on microcontroller. The Nucleo board read every single data in the text file and filtered them by using CMSIS-FIR low pass filter, followed by writing the signal to the output text file stored in the SD card. The output text file was plotted in the MATLAB to examine the effectiveness of the CMSIS-filter. The FIR filter coefficients were generated by the MATLAB filter designer and cross validation was done by convoluting the filter coefficient with the noisy ECG signal.

Hardware

1) STM32F446RE
2) SD Card Module
3) SD Card



Pin Connection: (SD Card Adapter -> Nucleo Board)

1) VCC -> 5V
2) GND -> GND
3) SCK -> PA5
4) MISO -> PA6
5) MOSI -> PA7
6) CS -> PB6

Software

1) STM32 Cube IDE
2) CMSIS DSP Library
3) MATLAB
4) Putty

Noise Generation

In MATLAB, generate the noise using code below. Save the noisy ecg signal text file into SD card.

```
openExample('dsp/RemovingHighFrequencyNoiseFromAnECGSignalExample') ;

x = ecg(500).';

y = sgolayfilt(x,0,5);

[M,N] = size(y);

x = .1*randn(M,N);

highFreqNoise = HP(x);

noisySignal = y + highFreqNoise;

ecg_noise = single(y);

ecg_noise40 = repmat( ecg_noise, 40, 1);

writematrix(ecg_noise40,'noise.txt');

type noise.txt
```

Pinout and Configuration

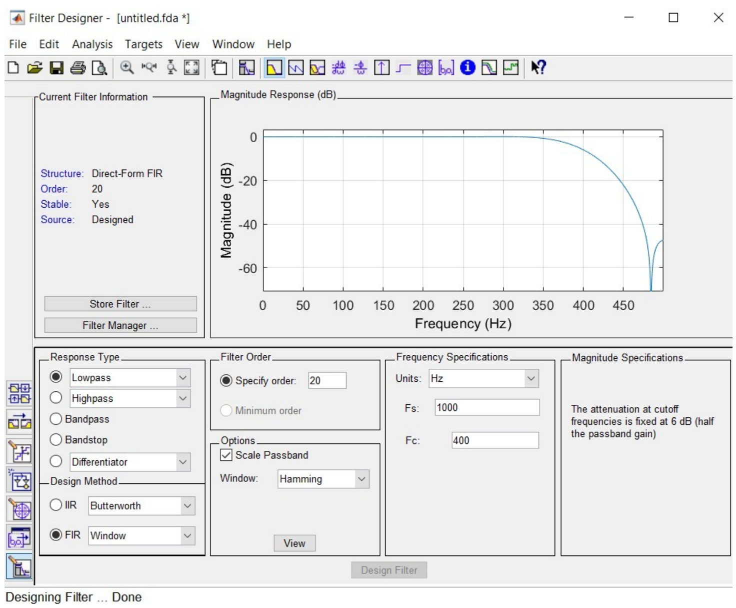
1) Open a new project at STM32 cube IDE.
2) At the pinout and configuration tab, enable the SPI 1 connection with Full Duplex Master mode. The SCK, MISO, and MOSI pins will be auto assigned to the pin connection as in the Hardware part above.
3) Reduce the baud rate by prescaling the clock to 128.

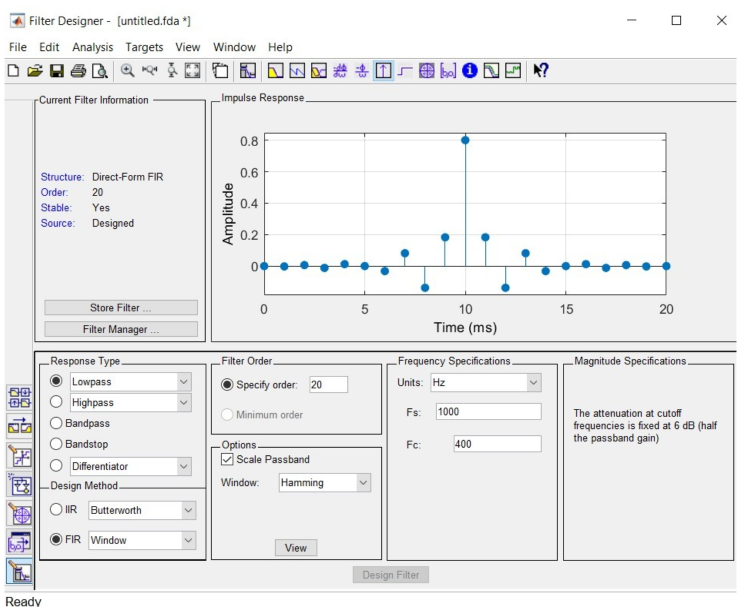
4) Connect the PB6 and CS together using GPIO connection, followed by labeling it as SD_CS.
5) Open the FAT file system in Middleware. Increase the maximum sector size from 512 to 4096 so our memory card can be supported.
6) Serial Monitor was used for us to monitor the read write status and mounting status of the SD card towards the Nucleo board.
7) Use the default UART ports, USART 2 to connect the serial monitor to the board. Set the baud rate as 9600 Bit/s.
8) Go to system mode setting, change the debug state into serial wire mode. Set the RCC into a crystal / ceramic resonator as our clock source. The maximum clock frequency was configured to 180 MHz.

Firmware Configuration

1) Import the Fatfs SD card function from their source (fatfs_sd.c) and header (fatfs_sd.h) in the Inc file. Call them in the header of user_diskio.c and main.c.
2) The data was returned to the controller through SPI1 using fatfs function.
3) Add prompt out lines in the main.c file, so that the serial monitor prompts the mounting status of the sd card using f_mount, f_open and f_close functions.
4) Step 3 is important because it helps us to locate the error caused by hardware failure, software failure or logic error in coding.
5) The main loop is in charge of reading, processing and writing data to the SD card and reporting the status via serial monitor.
6) Include the CMSIS library and arm_math library to the project. The arm_fir_32 and arm_fir_init_f32 functions were used.
7) The arm_fir_init_f32 consists of assorted initialization functions of each data type. We used this function to set the values of the internal structure fields and initialize the state buffer’s values. Since this function was used, the constant data section cannot be placed in the instance structure.
8) S parameter represents the points of instance of the single precision FIR filter structure
9) numTaps represents the filter coefficient number in the FIR filter (in our case is 21).
10) firCoeffs32 was the array consisting of the filter coefficients we obtained from the filter designer of MATLAB.
11) firStateF32 was the state buffer.
12) blockSize was the number of samples processed each time calling.
13) The complete coding is in this link: https://github.com/YeoHongChew/Microprocessor-Assignmnet.git
14) File name: Group_3.zip
15) Demo Link: https://youtu.be/Hd08rY3PnKw

Filter Coefficient Generation





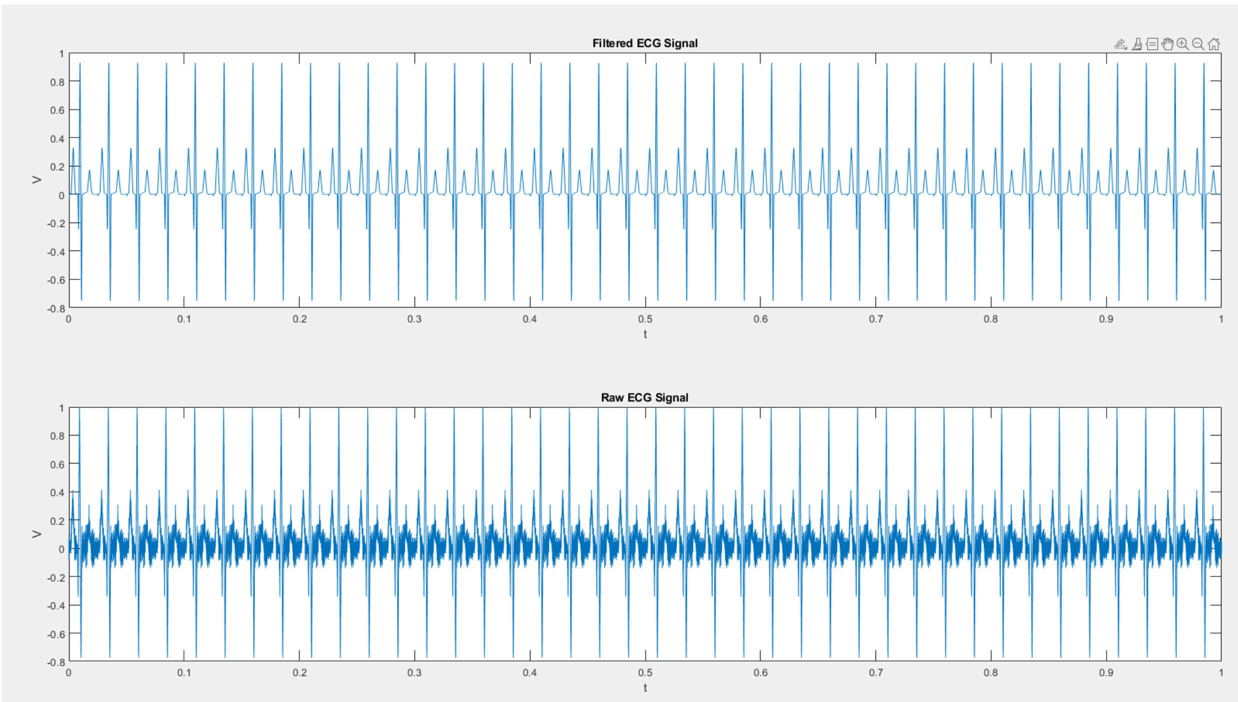
Sampling Frequency = 1000 Hz

Cut off Frequency = 400 Hz

FIR Window = Hamming Window 20 Order

Filter Type = Low Pass Filter

Result Plotting



Plot the ECG signal based on the noise.txt and noise_out.txt files in SD card based on the MATLAB script below.

```
fileID = fopen('noise_out.txt','r');

fileID2 = fopen('noise.txt','r');

 formatSpec = '%f';

 A = fscanf(fileID,formatSpec);

 B = fscanf(fileID2,formatSpec);

 x = linspace(0,1,20000);

 y = A;

 z = B;
```

```
subplot(2,1,1);

plot(x,y);

xlabel('t');

ylabel('V');

title('Filtered ECG Signal');


subplot(2,1,2);

plot(x,z);

xlabel('t');

ylabel('V');

title('Raw ECG Signal');
```

References

1) **https://www.mathworks.com/help/dsp/ug/removing-high-frequency-noise-from-an-ecg-signal.html**
2) **https://github.com/mkel1123project/Audio-Filtering-on-STM32-Using-CMSIS-DSP**
3) **https://os.mbed.com/platforms/ST-Nucleo-F446RE/**
4) **https://controllerstech.com/sd-card-using-spi-in-stm32/**
5) **https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html**