UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Documentação de Projeto – Parte 2
Design, Estudo da Plataforma

Projeto: Sistema de Filtro Digital Parametrizado

Autores: Eric Yutaka Fukuyama e Fabio Seiti Fukuda

Versão: 26-Maio-2024

Parte 2a – Design

1 Introdução

Esse documento de Design e Estudo da Plataforma tem como objetivo o design e planejamento da solução do projeto. Dessa forma, esse projeto seria uma continuação ao documento anterior que incluía as partes de CONOPS e Especificação do sistema, pois a partir do estudo do que é necessário para os stakeholders, é possível detectar o que deve estar presente no sistema e o planejamento das próximas seções das arquiteturas.

2 Arquitetura Funcional

O diagrama da arquitetura funcional apresenta as funcionalidades do projeto. Dessa forma, o sistema deve ser iniciado com uma leitura do input do usuário. Assim, há duas possibilidades, ou seja, o usuário pode ligar/desligar o filtro ou configurar os parâmetros do filtro. Portanto, caso seja feita a configuração, então ocorre o cálculo dos coeficientes do filtro FIR, os quais são passados como parâmetro para o filtro. Caso o usuário decida desligar o filtro, essa informação é passada diretamente para o filtro.

Ainda, o filtro recebe a leitura do sinal de entrada e retorna, finalmente, a geração do sinal de saída. Dessa forma, o diagrama com arquitetura funcional pode ser visto a seguir na Figura 1.

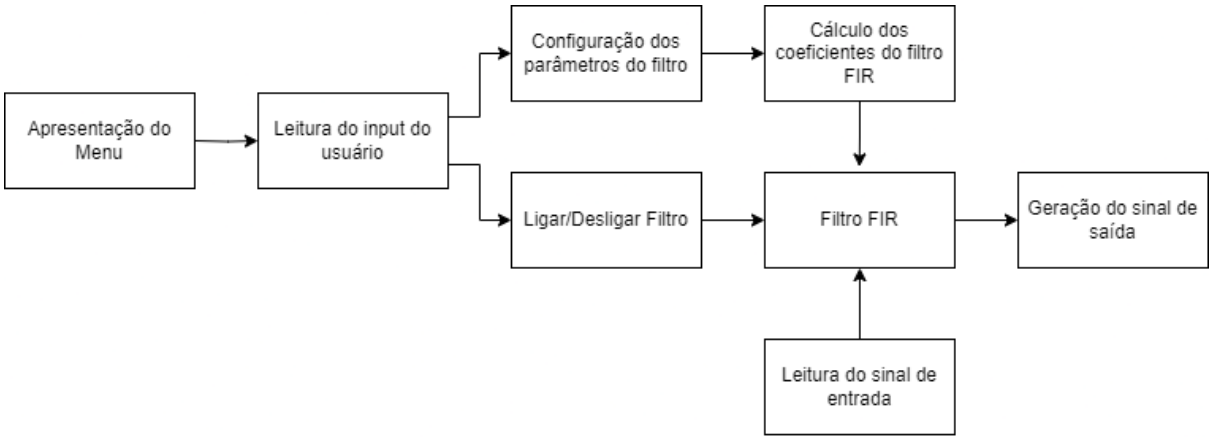

Figura 1 - Diagrama com a arquitetura funcional.

3 Arquitetura Física (Arquitetura da Solução)

A Arquitetura da Solução é apresentada na Figura 2. É possível, assim, dividir a arquitetura do sistema em três partes.

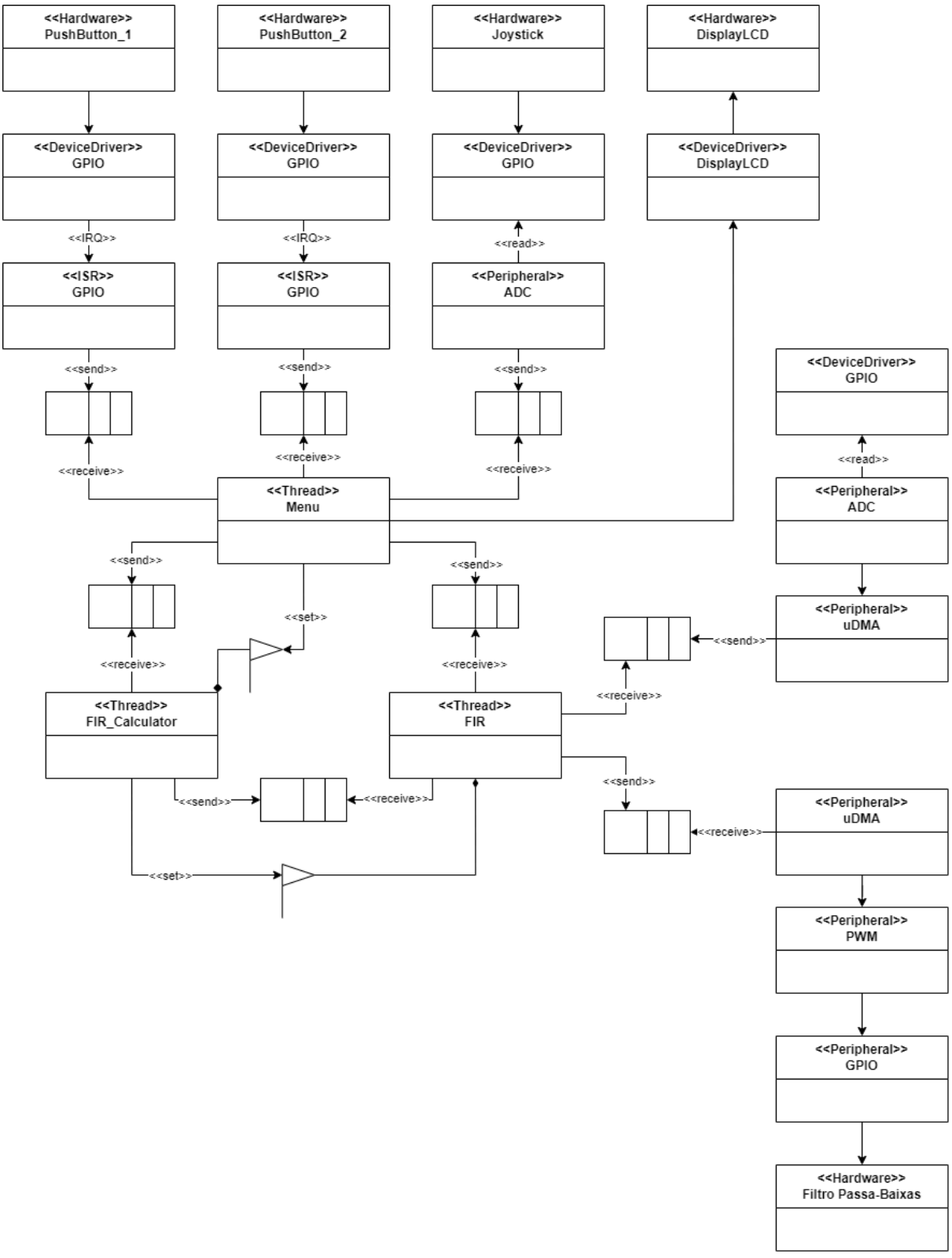

Figura 2 - Arquitetura da Solução

A primeira diz respeito à interface com o usuário. Seguindo o levantamento de requisitos, podemos observar que há dois botões e um joystick para o input do usuário. Assim, os botões estão conectados a pinos GPIO, os quais correspondem a linhas de interrupções. Já para o joystick, é necessário fazer uma leitura analógica do pino no qual ele está conectado.

Isso justifica a presença de um conversor ADC. Com isso, todas as informações dos inputs do usuário são colocadas em filas de mensagens.

A segunda divisão da arquitetura da solução diz respeito à implementação do software utilizando RTOS. Observa-se, assim, que há uma thread Menu responsável por gerenciar o fluxo de informação vinda do usuário para o sistema, e vice-versa. Dessa maneira, essa thread também é responsável por gerenciar a interface gráfica.

Não só isso, como também é o Menu quem avisa a thread que calcula os coeficientes do filtro FIR (FIR_Calculator) a respeito de mudanças de parâmetro dadas pelo usuário. Por isso se faz necessário tanto uma flag de evento quanto uma fila de mensagens.

Assim, se faz preciso também uma flag de evento entre a thread FIR_Calculator e a thread FIR, a qual faz as operações para a filtragem do sinal de entrada. Logo, quando houver uma mudança de parâmetros dos coeficientes do filtro, a thread FIR_Calculator avisa a FIR através de uma flag de evento.

Por fim, a última divisão do sistema se refere à leitura do sinal de entrada e a geração do sinal de saída. Assim, a amostragem do sinal se dá por um conversor ADC, enquanto a saída do sinal é dado por um sinal PWM em cascata com um filtro passa-baixas.

Para tanto, o uso de um DMA na amostragem se faz necessário, pois levando em consideração, pelo levantamento de requisitos, um sinal de 20kHz, seria necessário uma taxa de amostragem mínima de 40kHz. Ou seja, se a amostragem fosse implementada por interrupções no sistema, seria necessário uma frequência de interrupções de pelo menos 40kHz, o que não é eficiente principalmente levando em consideração um ambiente multi-thread. Se faz necessário também um DMA na saída do sinal por razões análogas.

4 Interface com o Usuário

A interface com o usuário será composta por dois push-buttons, um display LCD e um joystick. Com o display será possível visualizar as configurações e com os botões e o joystick serão feitas as alterações nos valores.

Desse modo, o menu inicial pode ser visualizado na Figura 3. Logo, com o joystick será possível percorrer de uma opção e outra e com um push-button haverá a seleção

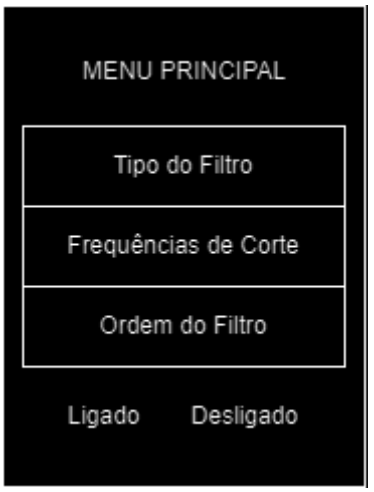

Figura 3 - Menu Principal.

Por fim, se o usuário selecionar uma das opções de Tipo de Filtro, Frequências de Corte e Ordem do Filtro abrirá um submenu como visualizado na Figura 4 para a seleção.

Logo, um push-button será usado para a confirmação da configuração e o outro para voltar para o menu principal.

Figura 4 - Submenu “Tipo de Filtro”.

5 Mapeamento da Arquitetura Funcional à Arquitetura Física

O mapeamento proposto entre as arquiteturas pode ser visto na Figura 5. Dessa forma, os elementos da arquitetura física estão presentes nas linhas enquanto os elementos da arquitetura funcional estão nas colunas. Ainda, é denotado com um símbolo “X” na tabela o elemento da arquitetura física que implementa a funcionalidade demonstrada na arquitetura funcional.

	Apresentação do Menu	Leitura do Input do usuário	Configuração dos parâmetros do filtro	Ligar/desligar filtro	Cálculo dos coeficientes do filtro FIR	Filtro FIR	Leitura do sinal de Entrada	Geração do sinal de Saída
<<Hardware>> PushButton_1		X	X					
<<Hardware>> PushButton_2		X	X	X				
<<Hardware>> Joystick		X	X	X				
<<Hardware>> DisplayLCD	X							
<<Hardware>> Filtro Passa-Baixa								X
<<DeviceDriver>> GPIO		X		X			X	X
<<DeviceDriver>> DisplayLCD	X							
<<ISR>> GPIO		X	X					
<<Peripheral>> ADC		X	X				X	
<<Peripheral>> uDMA							X	X
<<Peripheral>> PWM								X
<<Thread>> Menu	X							
<<Thread>> FIR_Calculator					X			
<<Thread>> FIR						X		

Figura 5 - Tabela Arquitetura Funcional x Arquitetura Física

6 Arquitetura do Hardware

Esse projeto final utiliza a placa da Tiva EK-TM4C1294XL em conjunto com a Boosterpack BOOSTXL-EDUMKII como plataforma de hardware. Desse modo, os push-buttons, joystick e o display LCD descritos nas seções anteriores serão os que estão integrados na Boosterpack. A placa Tiva pode ser visualizada pela Figura 6. A Boosterpack utilizada e os periféricos mencionados podem ser vistos na Figura 7.

Figura 6 - Tiva EK-TM4C1294XL.

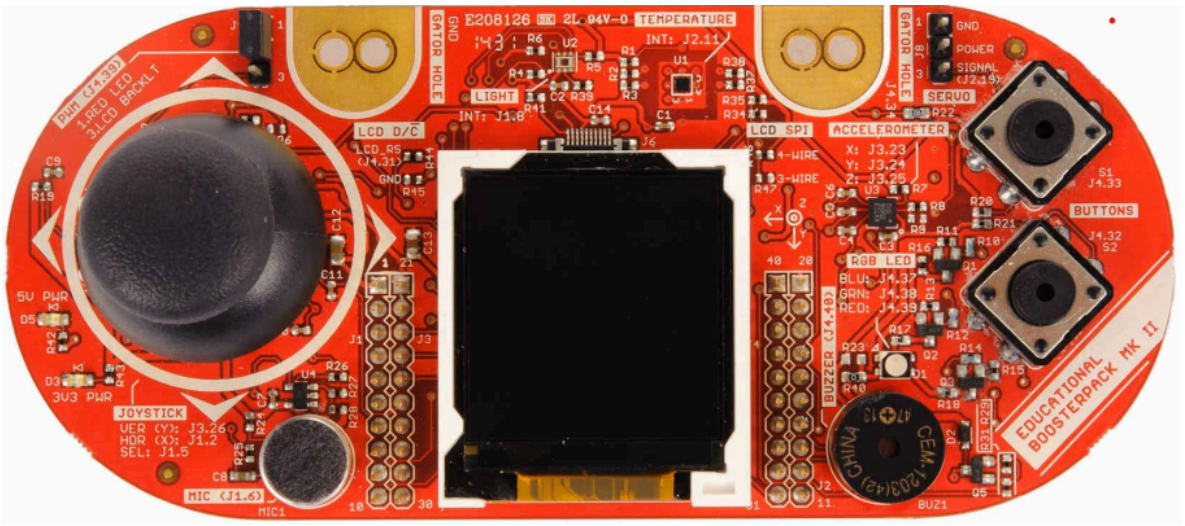

Figura 7 - Boosterpack e os periféricos Joystick, Buttons e o Display.

Assim, para a leitura dos push-buttons deverá ser feito um tratamento de interrupção por software, enquanto que para a leitura do joystick deverá haver uma conversão de um sinal analógico para digital.

Por fim, para a configuração do display LCD será utilizado o código disponibilizado pelo professor da disciplina no GitHub como base.

7 Design Detalhado

Sobre o funcionamento do sistema como um todo, a Figura 8 apresenta um diagrama de estados. Observa-se assim, o funcionamento do Menu e do Filtro em paralelo.

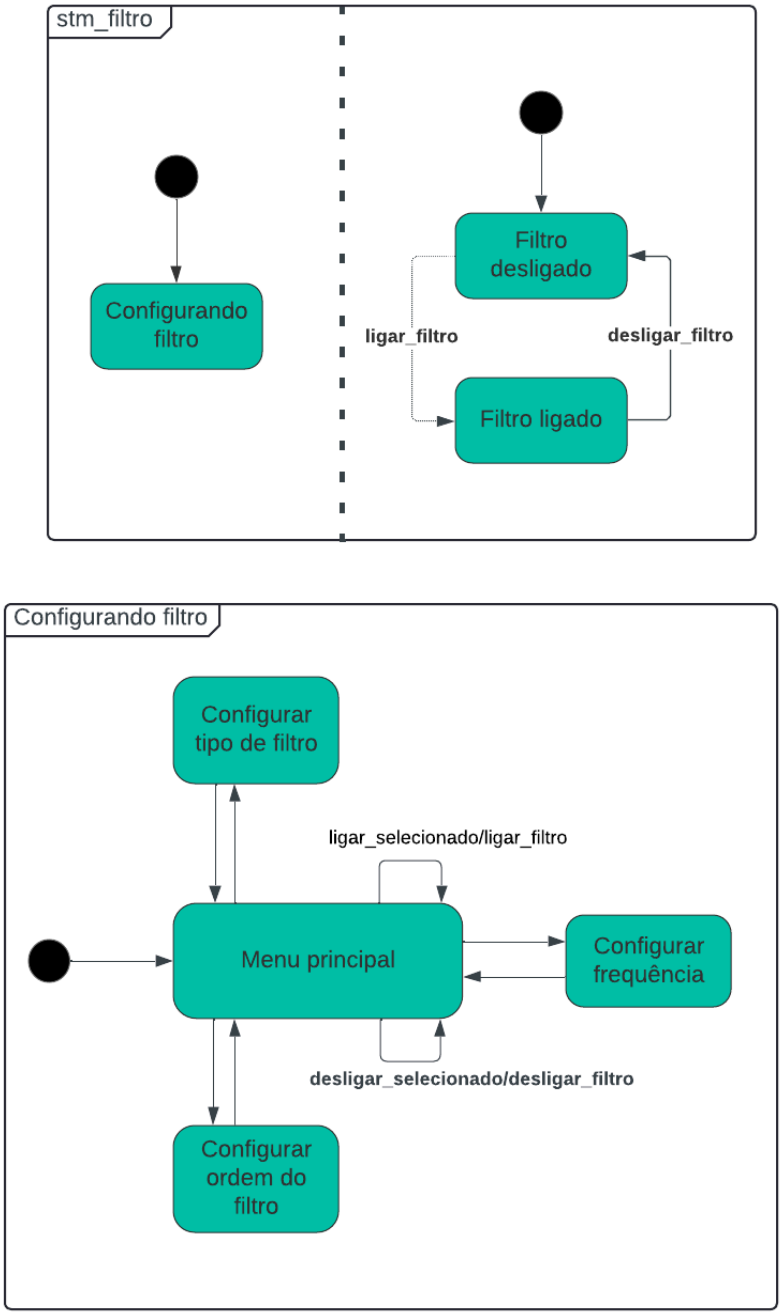

Figura 8 - Diagrama de Estados do Sistema de Filtro.

Assim, pela arquitetura física é possível observar que as rotinas de tratamento de interrupção informam os menus sobre os inputs dados pelos botões através de uma fila de mensagens. Portanto, na thread Menu será necessário constantemente verificar se há algum input novo. Em outras palavras, será necessário executar um polling, como mostra o diagrama de atividades da Figura 9.

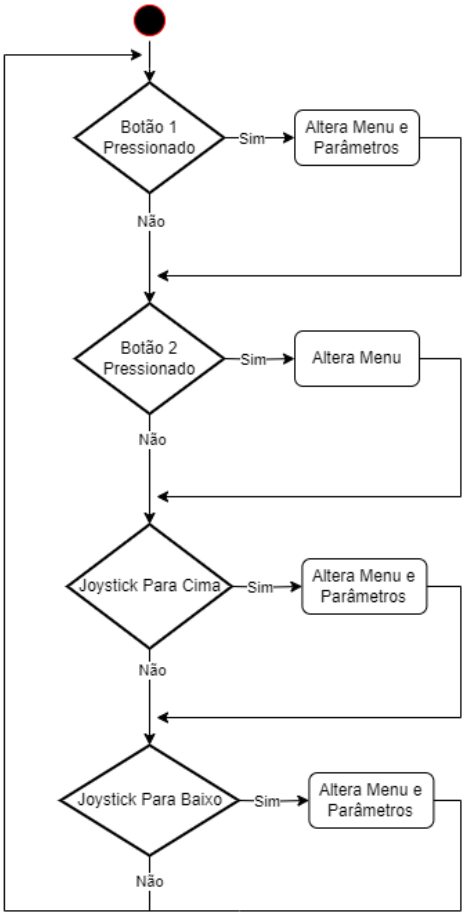

Figura 9 - Diagrama de Atividades do Menu.

Assim, as configurações captadas pelo Menu são repassadas para a thread FIR_Calculator. Este, porém, não precisa fazer polling para verificar se há uma mudança de parâmetros. Isso porque ele possui uma flag de evento. Assim, essa thread fica suspensa até que haja alguma mudança. Em caso de qualquer alteração no filtro FIR, os novos coeficientes deste são enviados para a thread FIR pela fila de mensagens. Assim, o fluxo de mudança de parâmetros do filtro é mostrado na Figura 10.

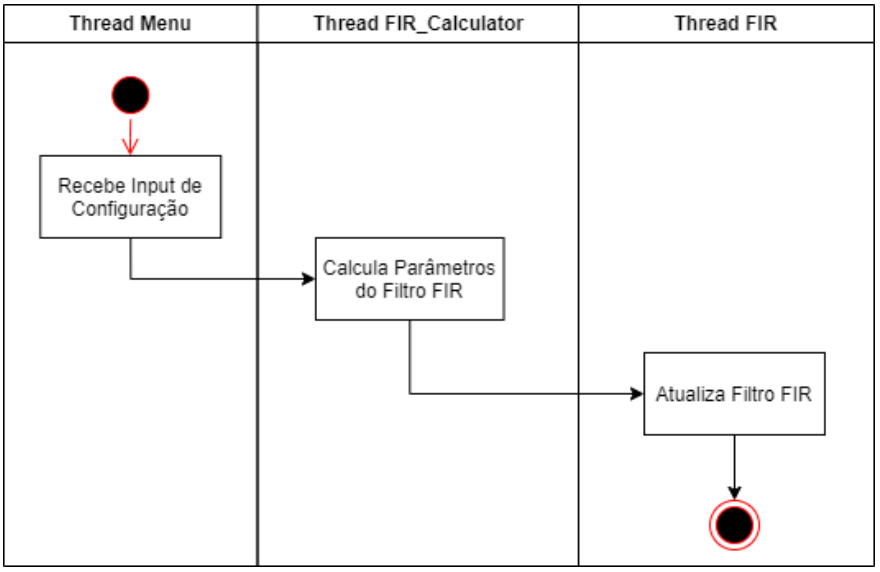

Figura 10 - Diagrama de Atividades
da mudança de parâmetros.

Sobre o funcionamento do processo de filtragem em si, a Figura 11 apresenta um diagrama de atividades que mostra este processo.

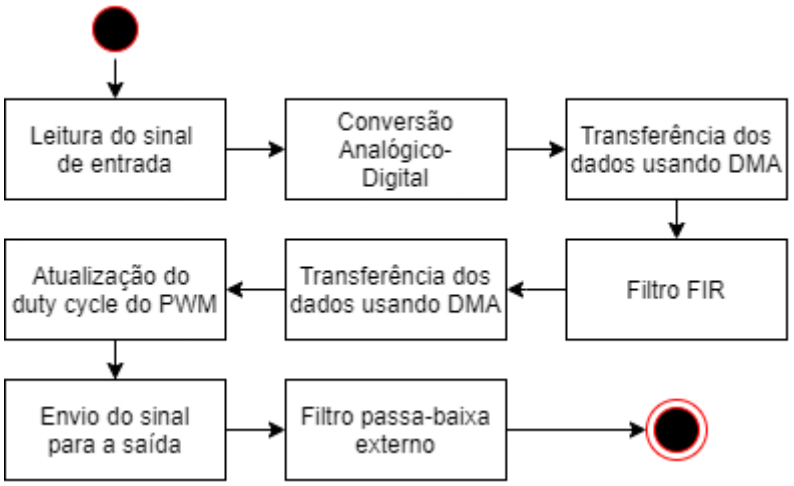

Figura 11 - Fluxograma do Processo de
Filtragem do Sinal

É importante ressaltar que uma defasagem entre o sinal de entrada e de saída é inerente à solução proposta. O primeiro argumento para isso é que o simples uso de um processador para fazer este procedimento já gera atrasos no sinal de saída.

Além disso, a filtragem e geração do sinal de saída será dada em cima de amostras consecutivas com tamanho maior que 1. Assim, é necessário acrescentar uma defasagem entre o sinal de entrada e saída pois, enquanto se aplica a filtragem em um sinal amostrado, de forma concomitante o sinal da amostra anterior está sendo gerado na saída. Ou seja, o sinal de saída de um dado instante é referente à amostragem anterior daquela que está sendo processada no momento.

Há de se pensar, além disso tudo, na taxa de geração do sinal de saída. Isso porque é necessário, pelo levantamento de requisitos, gerar amostras na saída a uma taxa mínima de 40kHz, isto é, uma amostra a cada 25µs. Ainda, levando em consideração que a função que aplica o filtro no sinal de saída gera algo em torno de 5 µs para gerar uma amostra, observa-se, então, que a thread FIR consumirá boa parte do processador.

Levando tudo isso em consideração, podemos montar um planejamento para o escalonamento das tarefas, a qual pode ser vista na Figura 12.

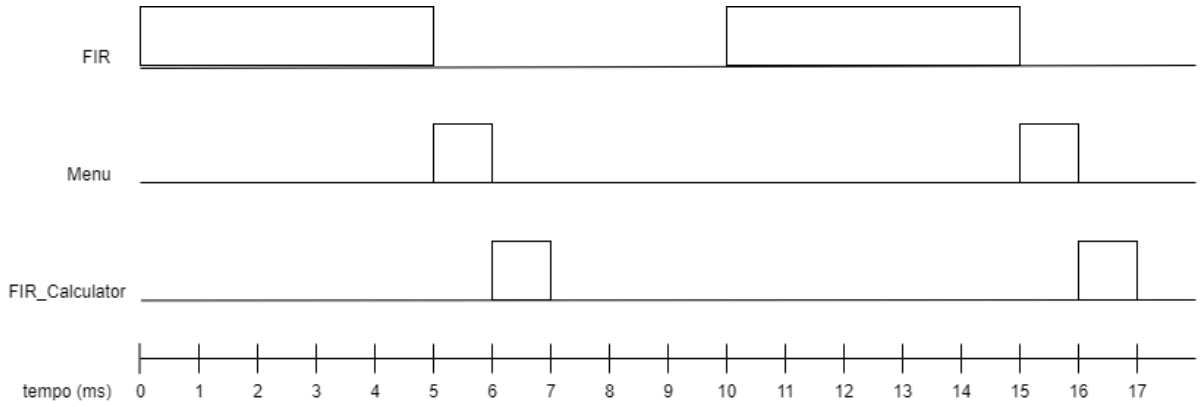

Figura 12 - Planejamento do escalonamento

Optou-se, assim, por dar menos tempo de processador para as threads Menu e FIR_Calculator, uma vez que as saídas que elas geram não são críticas em relação ao tempo. Ou seja, elas podem demorar um pouco mais sem comprometer o funcionamento do sistema.

Parte 2b – Estudo da Plataforma

1 Periféricos da Boosterpack

Primeiramente, é necessário saber quais são os pinos usados para a comunicação entre a placa e os periféricos utilizados. Desse modo, a Figura 13 ilustra um diagrama ilustrando essa pinagem.

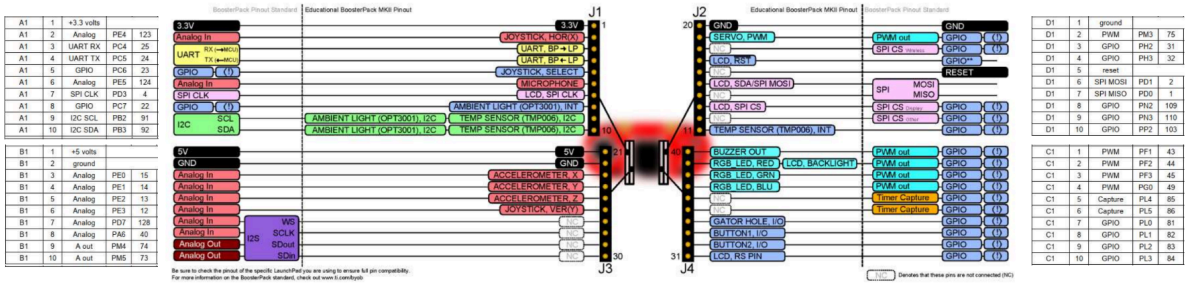

Figura 13 - Diagrama de Pinos da Boosterpack

Portanto, para o joystick serão usados os pinos da BoosterPack J1 2(JOYSTICK, HOR(X)) e J3 26(JOYSTICK, VER(Y)) que correspondem aos pinos da placa Tiva PE4 e PE3, respectivamente. Ainda, os botões correspondem aos pinos da BoosterPack J4 33(BUTTON1, I/O) e J4 32(BUTTON2, I/O) e estes correspondem aos pinos da placa PL1 e PL2, respectivamente. Por fim, para o Display LCD é possível notar pelo Diagrama de Pinos que os pinos J1 7(LCD, SPI CLK), J2 13(LCD, SPI CS), J2 15(LCD, SDA/SPI MOSI), J2 17(LCD, RST) e o J4 31(LCD, RS PIN) e, respectivamente, os pinos PD3, PN2, PD1, PH3 e PL3 da Tiva.

2 Entrada e saída de sinal

Para a entrada do sinal deverá ser usado um pino de ADC. Já para a saída de sinal será utilizado um pino de PWM. Desse modo, a Tiva possui dois módulos ADC, os quais compartilham 20 canais. Além disso, para cada módulo há 4 sequenciadores: SS0 (de 8 amostras), SS1 (de 4 amostras), SS2 (de 4 amostras) e SS3 (1 amostra).

Além disso, a Tiva TM4C1294x possui 4 geradores PWM e, com isso, é capaz de gerar 8 sinais PWM. O diagrama da Figura 14 ilustra os geradores e as saídas PWM.

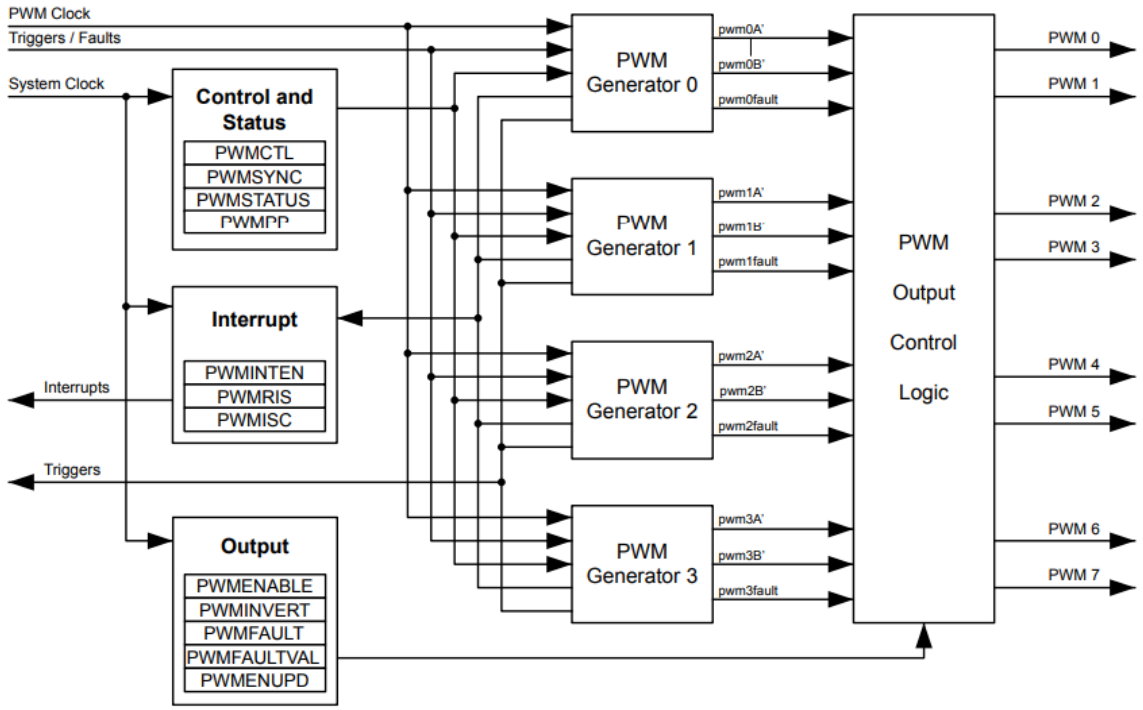

Figura 14 - Diagrama de Geradores e saídas do PWM.

Ademais, para setar o período do PWM será utilizada a função PWMGenPeriodSet. Porém, o registrador que guarda o período do PWM possui 16 bits, o que representa um tick máximo de 65535 clocks. Dessa forma, o valor do período do PWM será, portanto, calculado pela fórmula clock_do_sistema/(div x frequência_pwm) ticks em que div é um divisor para poder usar uma quantidade menor de ticks.

Ainda, algumas funções da TivaWare devem ser necessárias para a implementação dessa parte como:

- ADCSequenceConfigure(para a inicialização do ADC): configura a fonte do acionamento e a prioridade da sequência de amostra;
- ADCSequenceEnable(para a inicialização do ADC): habilita uma sequência de amostra;
- ADCSequenceStepConfigure(para a inicialização do ADC): configura o passo da sequência de amostra(deve ser chamada antes do ADCSequenceEnable);
- GPIOPinTypePWM(para a inicialização do PWM): configura o pino para o uso do PWM;
- PWMClockSet(para a inicialização do PWM): define a configuração de clock do PWM;
- PWMGenConfigure(para a inicialização do PWM): configura o gerador do PWM;
- PWMGenEnable(para a inicialização do PWM): desabilita o timer/counter para um bloco do gerador do PWM;

- PWMGenPeriodSet(para a inicialização do PWM): configura o período do gerador do PWM;
- PWMPulseWidthSet(para a inicialização do PWM): configura a largura do pulso para a saída do PWM específico;
- PWMOutputState(para a inicialização do PWM): desabilita ou habilita as saídas do PWM;
- ADCIntStatus: obtém o status atual da interrupção;
- ADCSequenceDataGet: obtém o dado obtido pela sequência de amostra;
- ADCProcessorTrigger: causa um acionamento do processador para a sequência de amostra;

Para o projeto, os pinos selecionados para a entrada e saída do sinal foram, respectivamente, os E3 e F2.

3 Tratamento de interrupção

As interrupções produzidas por sinais no GPIO serão tratadas por meio da biblioteca da Tivaware em que é possível utilizar as funções de IntEnable para habilitar as interrupções e definir as funções de Handler para o tratamento dessas exceções. Por fim, para não precisar alterar a posição do vetor de exceções para a RAM, não será usado as funções do tipo "IntRegister". Dessa forma, é importante saber a pinagem correta descrita na subseção "Periféricos da BoosterPack".

4 ThreadX

O ThreadX é o sistema operacional em tempo real(RTOS) que será utilizado para esse projeto. Dessa forma, esse sistema será importante para o uso das threads. Assim, algumas funções serão essenciais para o desenvolvimento. Primeiramente, as funções tx_kernel_enter e tx_aplcation_define são essenciais para a inicialização do ThreadX, assim, a primeira é responsável pela inicialização das estruturas de dados do próprio RTOS e a segunda serve para o instanciamento de algumas estruturas de dados. Ainda, o tx_thread_create(para criação das threads), tx_byte_allocate (aloca bytes de memória), tx_event_flag_create(cria as event flags), tx_event_flags_set(configura a event flag). tx_event_flags_get(lê a event flag) e o tx_byte_pool_create(cria a memory pool de bytes).

5 Interface gráfica

A interface gráfica será visualizada por meio do display LCD embutido na Boosterpack. Ainda, para a utilização desse display será necessário o uso das funções presentes na biblioteca grlib.lib. Tal biblioteca possui as seguintes funções que devem ser utilizadas no projeto: GrContextInit(para inicializar um contexto gráfico), GrFlush(atualiza a tela para mostrar as alterações gráficas), GrContextFontSet(define a fonte), GrContextForegroundSet(define a cor do plano da frente), GrContextBackgroundSet(define a cor do plano de fundo), GrStringDraw(escreve uma string na posição especificada), GrContextDpyWidthGet(obtém a largura da área de exibição) e GrContextDpyHeightGet(obtém a altura da área de exibição). Ainda, as funções do arquivo Crystalfontz128x128_ST7735.c serão importantes também para a configuração do display. Dessa forma, duas dessas funções devem ser utilizadas como a Crystalfontz128x128_Init(inicializa o módulo do display) e a Crystalfontz128x128_SetOrientation(define a orientação do display). Por fim, é válido ressaltar que o código usado para essa parte será inspirado no código disponível pelo professor da disciplina.

6 CMSIS-DSP

O CMSIS-DSP é uma biblioteca de Software que possui funções de processamento de sinais para o Cortex-M e o Cortex-A. Dessa forma, para esse projeto, as funções de filtro FIR são importantes para construção do filtro. Logo, as funções de inicialização do filtro FIR tais como as próprias funções de processamento do filtro FIR são as que devem ser usadas com maior frequência nesse sistema.