UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Documentação de Projeto – Parte 2
Design, Estudo da Plataforma

Projeto: Sistema de controle de temperatura veicular

Autores: João Victor Laskoski
Luis Camilo Jussiani Moreira

Versão: 31-Out-2022

Parte 2a – Design

1 Introdução

Este documento em questão visa detalhar a projeção do sistema de controle de temperatura veícular e o estudo para confecção do sistema, tal projeto que foi introduzido no documento "CONOPS, Domínio do Problema, Especificação" apresentado anteriormente.

Desse modo, o presente documento aprofundará sobre as arquiteturas do projeto, definindo:

- Estruturas de hardware e software
- Fluxo do sistema
- Design da interface de usuário
- Funcionamento da interface de usuário

Os itens serão construídos de acordo com os requisitos funcionais/não funcionais e restrições que foram definidos previamente no documento "CONOPS, Domínio do Problema, Especificação".

Além do mais, será demonstrado o estudo de hardware e software feito pelos desenvolvedores para confecção do projeto, visando atingir com êxito o propósito, missão e visão pontuados no primeiro documento.

2 Arquitetura Funcional

Dessa forma, a Figura 1 apresenta o fluxo das funcionalidades do sistema, a qual toma base do diagrama de casos de uso confeccionado no primeiro documento para o projeto. Sendo assim, grande parte das funcionalidades parte da ação do usuário com a interface de usuário do sistema, que é dividida entre botões de controle de temperatura, botões de controle da velocidade do ventilador e seletor de saídas de ar. Vale ressaltar que o seletor pode desativar o sistema se nenhuma saída for selecionada.

Além disso, ao modificar a temperatura desejada, ocorrerá a mudança da abertura das válvulas para que se obtenha tal temperatura. Por fim, com as modificações das configurações realizadas pelo usuário, as mesmas serão enviadas para o visor (tela VGA) qual um kit FPGA fará o envio dos dados para a tela.

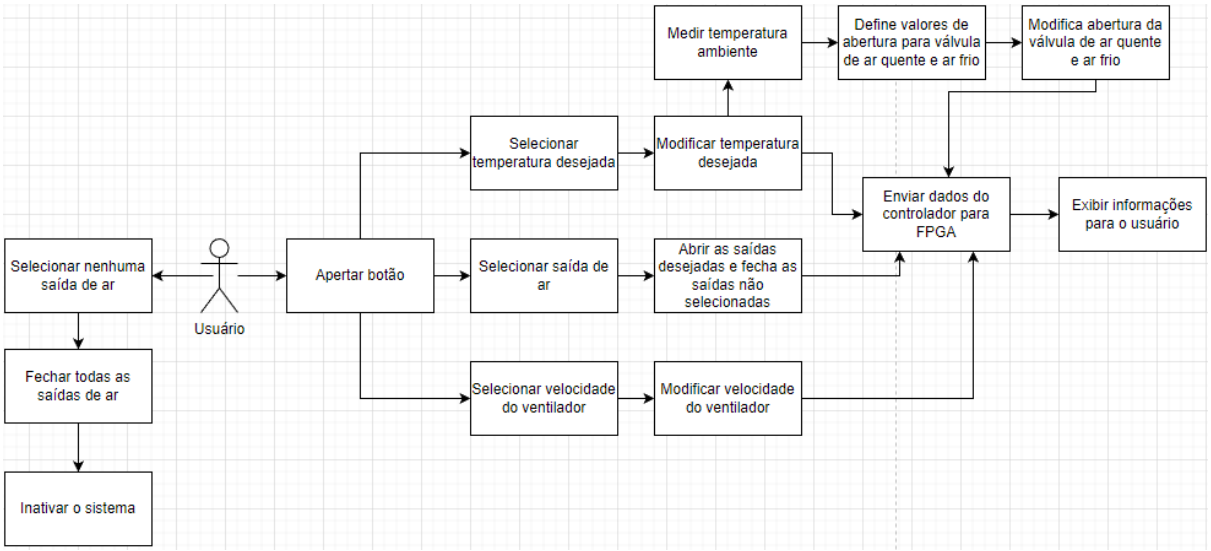

Figura 1 - Diagrama da arquitetura funcional

3 Arquitetura Física

Na figura 2 está apresentada a arquitetura física do sistema, contendo as threads de execução e o relacionamento dos componentes de hardware com os componentes de software.

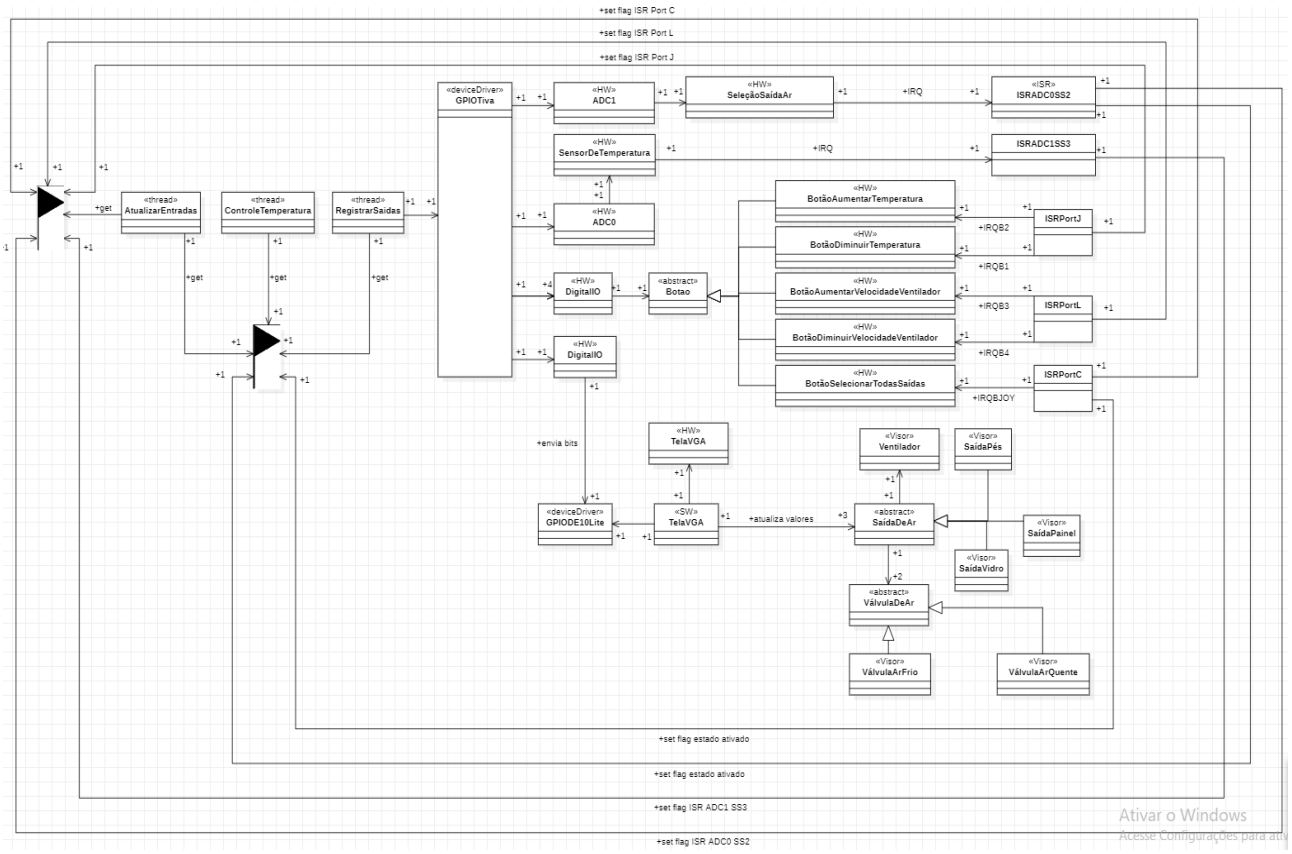

Figura 2 - Diagrama da arquitetura física (imagem)

4 Interface com o Usuário

Sendo assim, a interface de usuário contém duas partes, as quais serão detalhadas a seguir:

Entradas:

As entradas do sistema são compostas pelos botões e joystick, em que o usuário poderá interagir com o sistema para definir algumas configurações para o controle de temperatura interna do veículo. Sendo assim, conforme representado nas imagens abaixo, os dois botões de usuário presentes na placa Texas Tiva TM4C1294XL serão utilizados para o controle da temperatura desejada pelo usuário, sendo o botão B1 utilizado para diminuição da temperatura e o botão B2 para o aumento da temperatura.

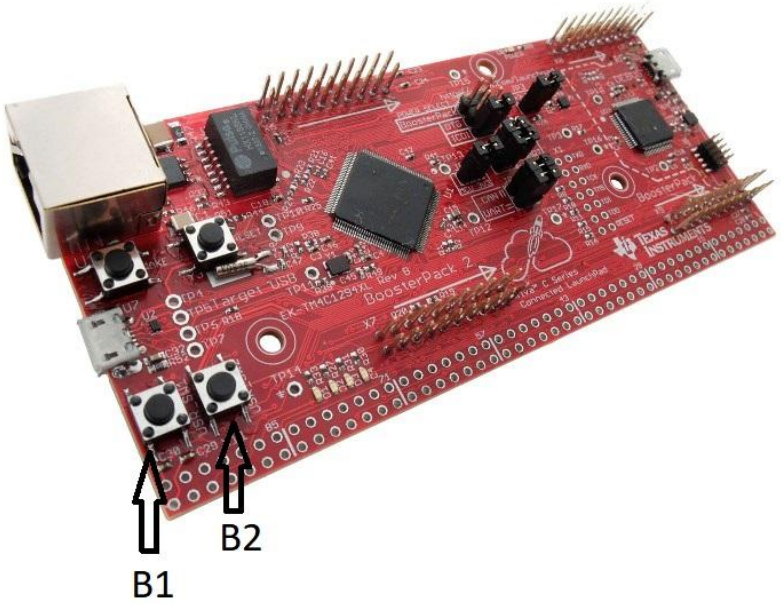

Figura 3 - Representação dos botões utilizados na placa Tiva TM4C1294XL

A seguir, os dois botões de usuário presentes na placa BOOSTXL-EDUMKII Educational BoosterPack serão utilizados para definição da velocidade do ventilador, no qual B4 será utilizado para redução da velocidade e B3 para o aumento da velocidade. Ademais, o joystick presente na mesma placa é utilizado para seleção das saídas, sendo que as saídas estão representadas nas possíveis posições do joystick presente na Figura 4. Vale ressaltar que se o joystick estiver na posição padrão, não é reconhecido como uma escolha, mantendo assim a última configuração realizada pelo movimento do joystick. Contudo, ao pressionar o joystick são selecionadas todas as saídas.

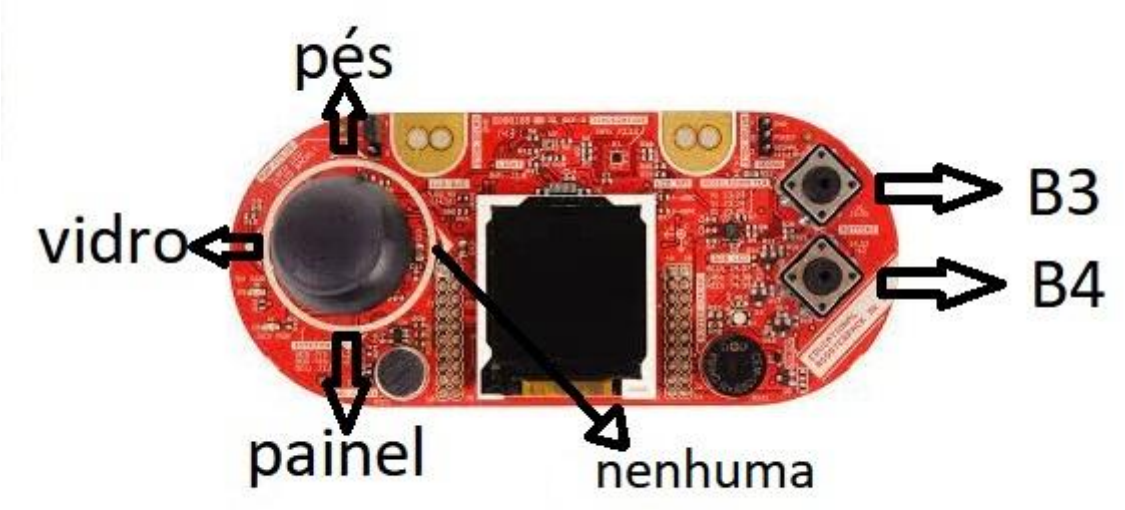

Figura 4 - Representação dos componentes utilizados na placa BOOSTXL-EDUMKII

Saídas:

As saídas do sistema são as informações do estado do sistema, as quais são demonstradas em uma tela VGA, sendo que a imagem abaixo possui uma representação simulada das saídas para o usuário.

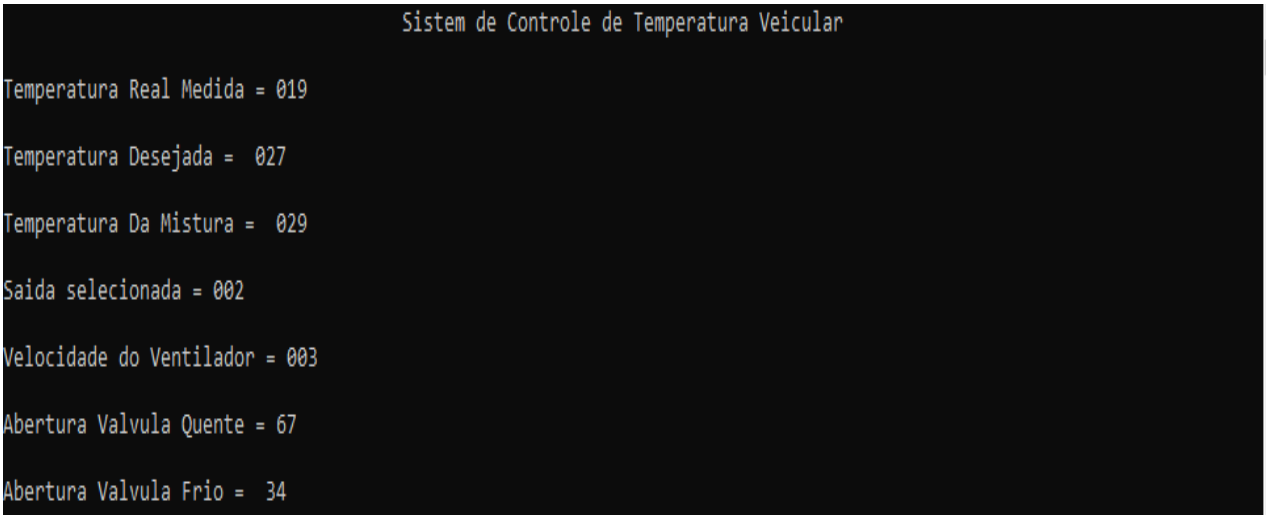

Figura 5 - Representação das saídas presentes na tela VGA

5 Mapeamento da Arquitetura Funcional à Arquitetura Física

Na figura 5, está contido o relacionamento entre os componentes da arquitetura física com as funcionalidades presentes na arquitetura funcional.

Mapeamento da Arquitetura Funcional à Arquitetura Física	AtualizarEntradas	ControleTemperatura	RegistrarSaidas	GPIOTiva	TelaVGA	GPIODE10Lite	ISRPortC	ISRPortJ	ISRPortL	SeleçãoSaídaAr	ISRADC0SS2	ISRADC1SS3
Selecionar nenhuma saída de ar										X		
Selecionar saida de ar							X			X		
Abrir as saídas desejadas e fecha as saídas não selecionadas											X	
Fechar todas as saídas de ar											X	
Inativar o sistema											X	
Apertar o botão				X			X	X	X			
Selecionar a velocidade do ventilador									X			
Modificar velocidade do ventilador	X											
Selecionar temperatura desejada								X				
Modificar temperatura desejada	X											
Medir temperatura ambiente												X
Definir valores de abertura para válvula de ar quente e ar frio		X										
Modificar abertura da válvula de ar quente e ar frio		X										
Enviar dados do controlador para FPGA			X	X		X						
Exibir informações para o usuário					X							

Figura 6 - Representação do mapeamento entre arquiteturas funcional e física.

6 Arquitetura do Hardware

Portanto, para composição do sistema de temperatura veicular, foram utilizados os seguintes itens:

- Microcontrolador Texas Tiva TM4C1294XL, que executará o software de controle de temperatura veicular, além de possuir alguns dos componentes da interface de usuário.
- Módulo BOOSTXL-EDUMKII Educational BoosterPack que conterá alguns dos componentes da interface de usuário.
- Potênciometro, para simular o sensor de temperatura.
- Tela VGA para exibir as informações para o usuário.
- Kit *MAX 10 10M50DAF484C7G* para comunicação com a tela VGA.

Ainda mais, a Figura 7 contém o diagrama da arquitetura de hardware, especificando os pinos e componentes utilizados pelos itens listados acima.

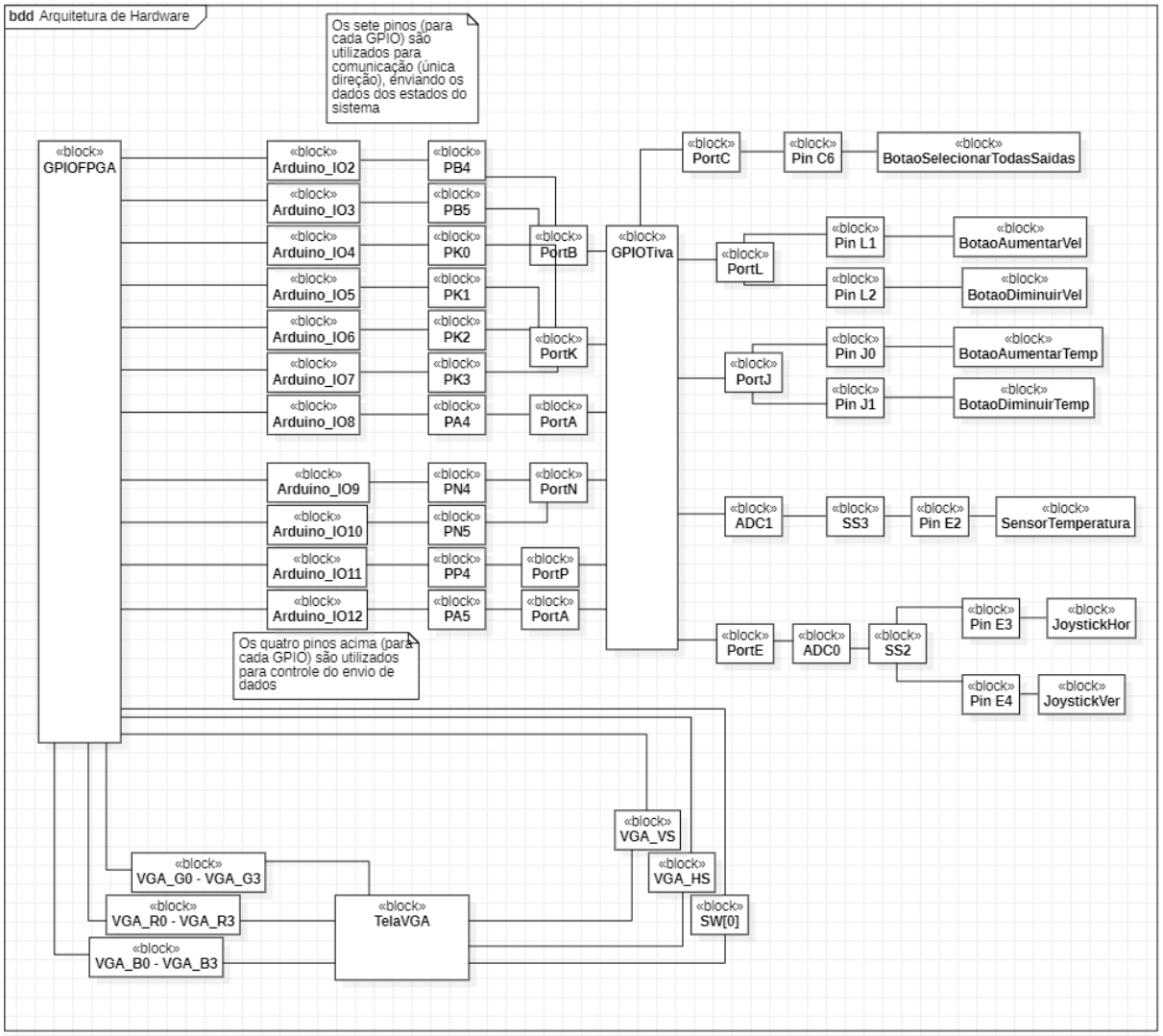

Figura 7 - Representação da arquitetura de hardware.

7 Design Detalhado

Desse modo, a Figura 7 contém as máquinas de estados que representam o funcionamento do sistema, na qual a primeira demonstra o funcionamento generalista do sistema e as Figuras 8, 9 e 10 apresentam com detalhes os super estados presentes no primeiro diagrama.

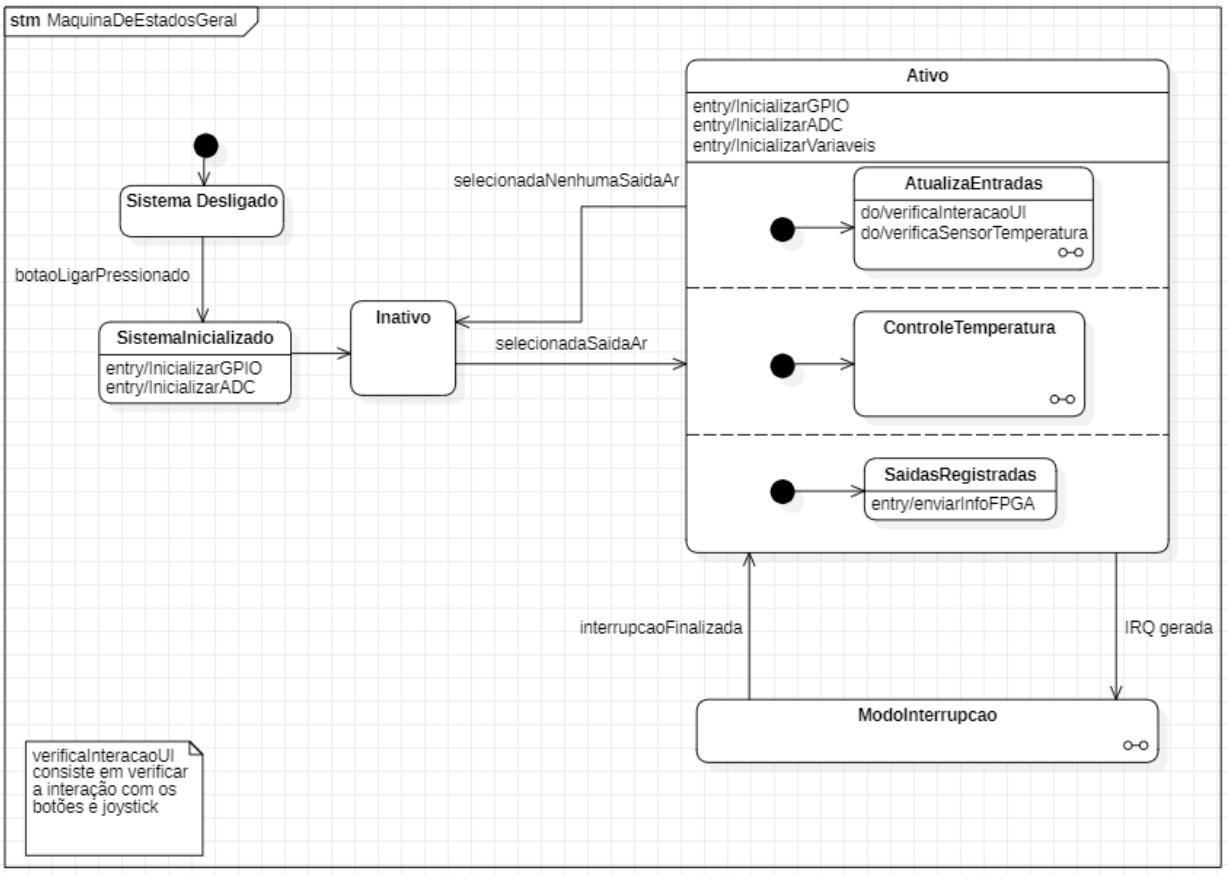

Figura 8 - Diagrama de estados geral

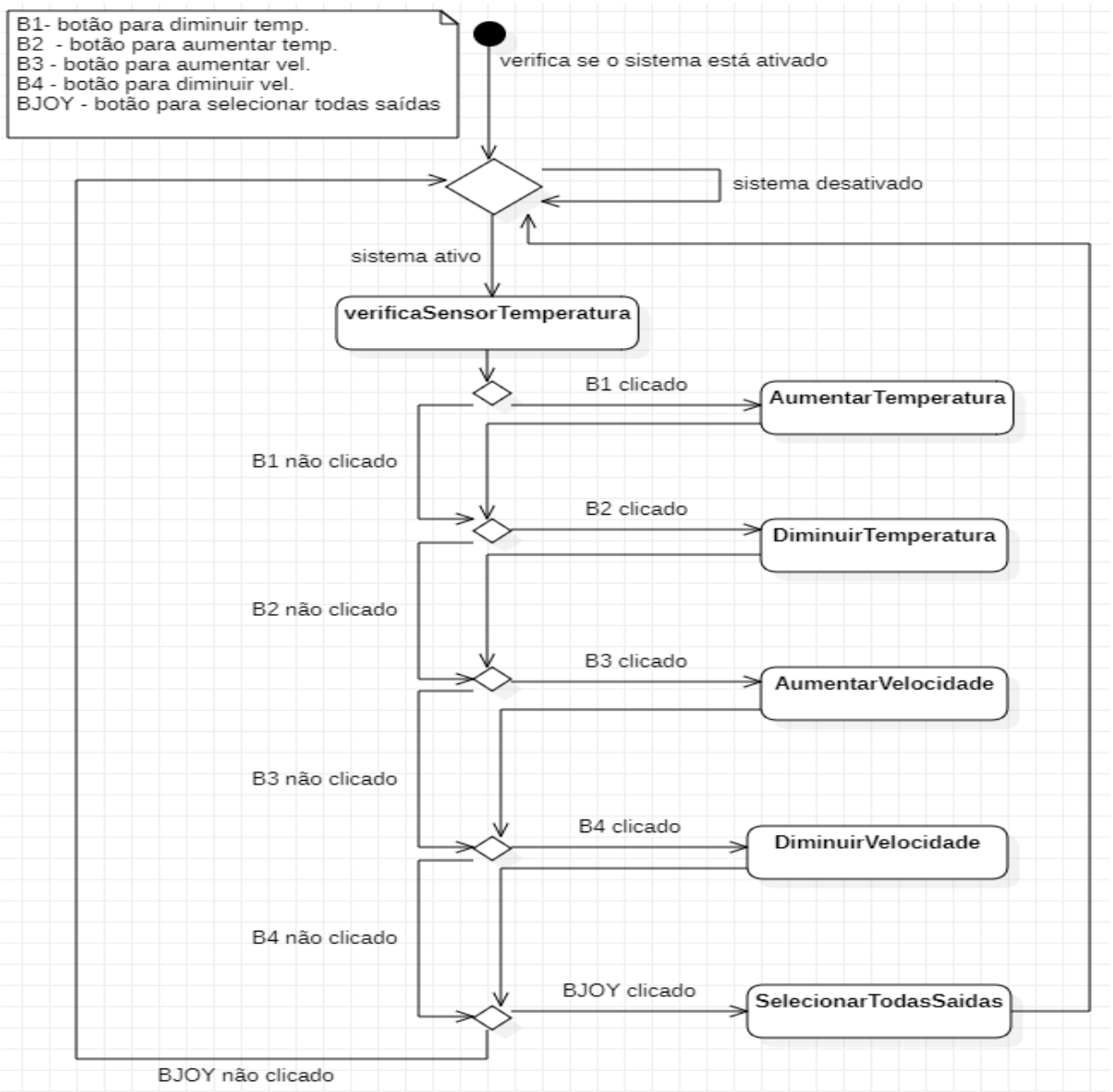

Figura 9 - Diagrama de estados (AtualizarEntradas)

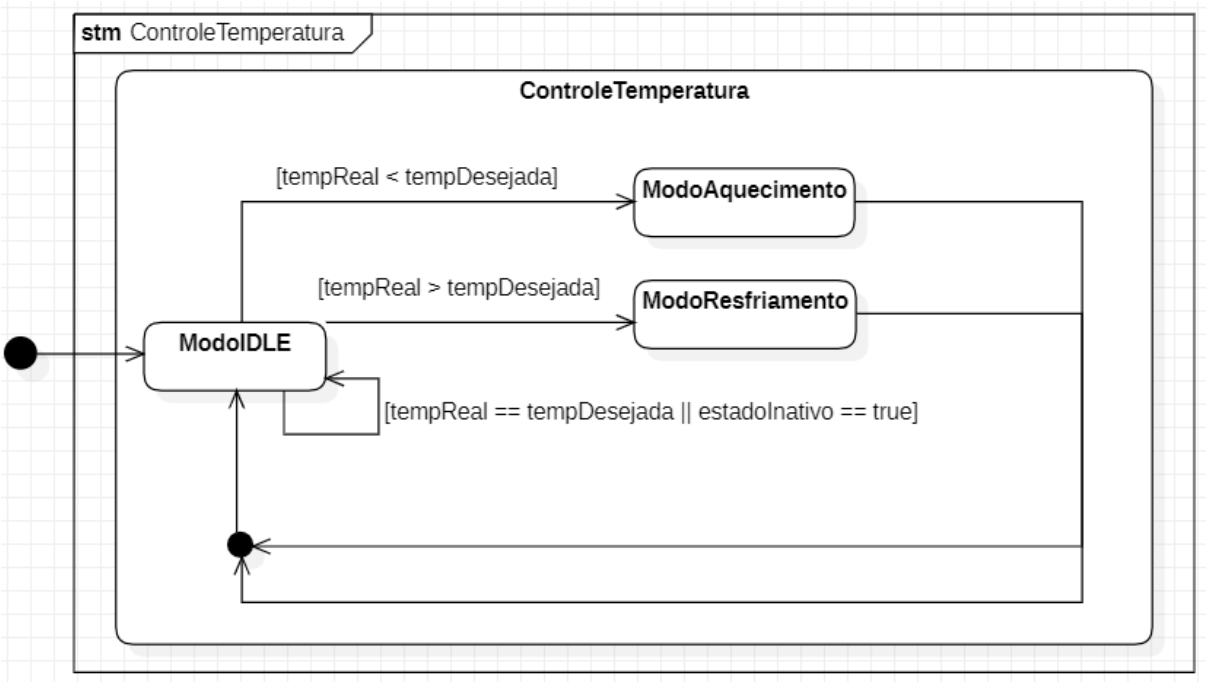

Figura 10 - Diagrama de estados (ControleTemperatura)

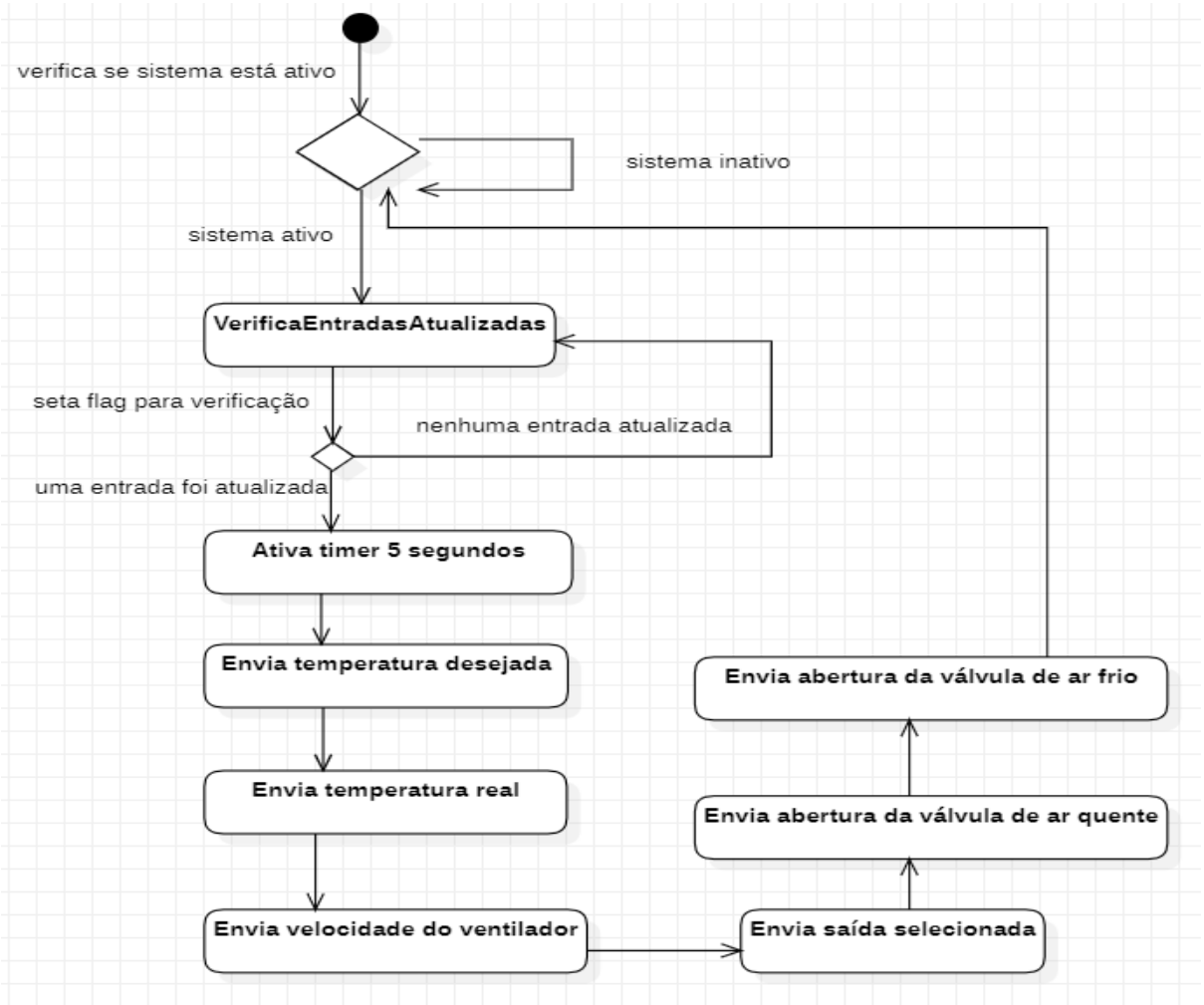

Figura 11 - Diagrama de atividades (RegistrarSaidas)

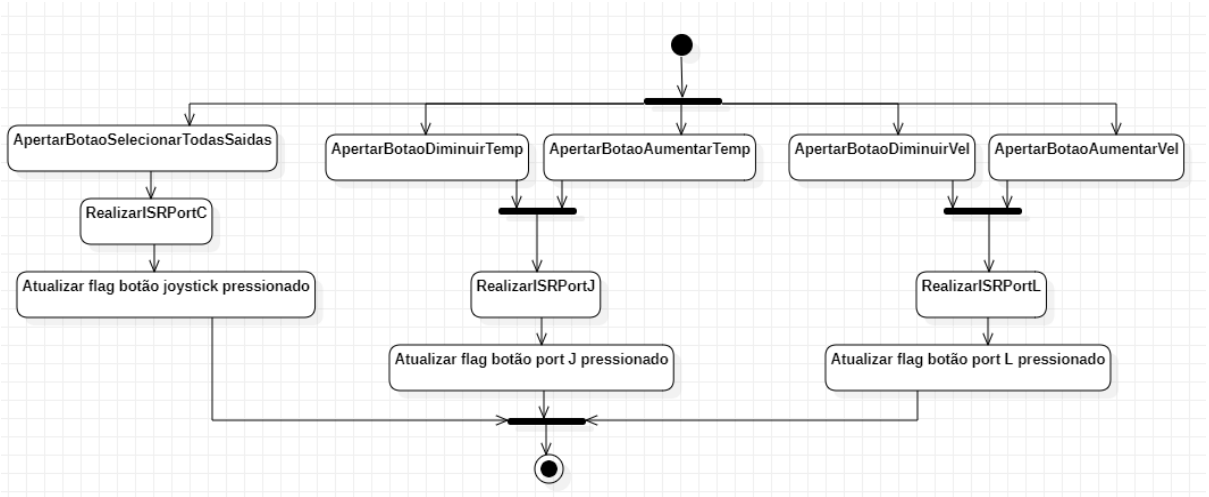

Figura 12 - Diagrama de atividades (ModoInterrupcao)

Parte 2b – Estudo da Plataforma

1 Estudo da plataforma de hardware

1.1 Placa Tiva

1.1.1 GPIO

Para tal sistema, foi utilizado o GPIO para comunicar com os botões presentes, como também para comunicação com o kit FPGA. Desse modo, a seguir encontra-se o mapeamento dos pinos de GPIO utilizados:

- Botão B1 (aumentar temperatura) → PJ0
- Botão B2 (diminuir temperatura) → PJ1
- Botão B3 (aumentar velocidade do ventilador) → PL1
- Botão B4 (diminuir velocidade do ventilador) → PL2
- Botão Joystick (selecionar todas as entradas) → PC6
- Barramento de dados de comunicação → (PB4, PB5, PK0, PK1,PK2, PK3, PA4)
- Barramento de controle de comunicação → (PN4, PN5, PP4, PA5)

1.1.2 Conversor A/D

Para o controle de temperatura veicular, o uso do conversor A/D tem utilidade para converter a tensão do potenciômetro (simulando o sensor de temperatura) e do joystick (simulando o seletor de saídas de ar) para valores digitais. Desse modo, os componentes vinculados com o conversor, estão mapeados com os seguintes pinos:

- Potenciômetro → PE2
- Joystick → PE3 (horizontal) e PE4 (vertical)

1.1.3 Clock do microcontrolador

Para o clock do sistema, será utilizado o overclock sobre o clock padrão do microcontrolador, para que o microprocessador execute em 125MHz.

1.2 Placa *DE10-Lite*

1.2.1 GPIO

O GPIO tem a finalidade de receber os dados do estado do sistema advindo do microcontrolador TIVA, sendo assim, servindo para comunicação. Desse modo, os pinos utilizados (sendo todos de entrada) são:-

- Arduino_IO2 ao Arduino_IO8, para receber o barramento de dados de comunicação enviado pelo microcontrolador TIVA.
- Arduino_IO9 ao Arduino_IO11, para receber o barramento de controle de comunicação enviado pelo microcontrolador TIVA.

1.2.2 VGA

Desse modo, para a parte executada no kit FPGA, a VGA tem o propósito de informar os dados do estado do sistema para o usuário, sendo um componente da interface de usuário. Assim sendo, a seguir está o mapeamento das entradas e saídas dos componentes da VGA, como também a relação com os pinos da placa:

Entradas:

- Clock, o qual atualiza o pixel corrente verificado.
- Switch, o qual desliga ou liga a tela VGA (PIN_C10 ou SW[0]), servindo como o reset.

Saídas:

- Saída para sincronização horizontal (PIN_N3 ou VGA_HS).
- Saída para sincronização vertical (PIN_N1 ou VGA_VS).
- Valor X e Y do pixel corrente (utilizado apenas para controle e lógica de qual pixel preencher).
- Doze saídas, sendo três conjuntos com quatro saídas cada. No qual, cada conjunto é referente a uma única cor entre vermelho, verde ou azul da escala RGB, definindo assim a escala para cada cor (sendo assim, cada cor possui 16 níveis de coloração). Sendo os pinos VGA_R[0..3] para coloração vermelha, VGA_B[0..3] para coloração azul e VGA_G[0..3] para coloração verde.

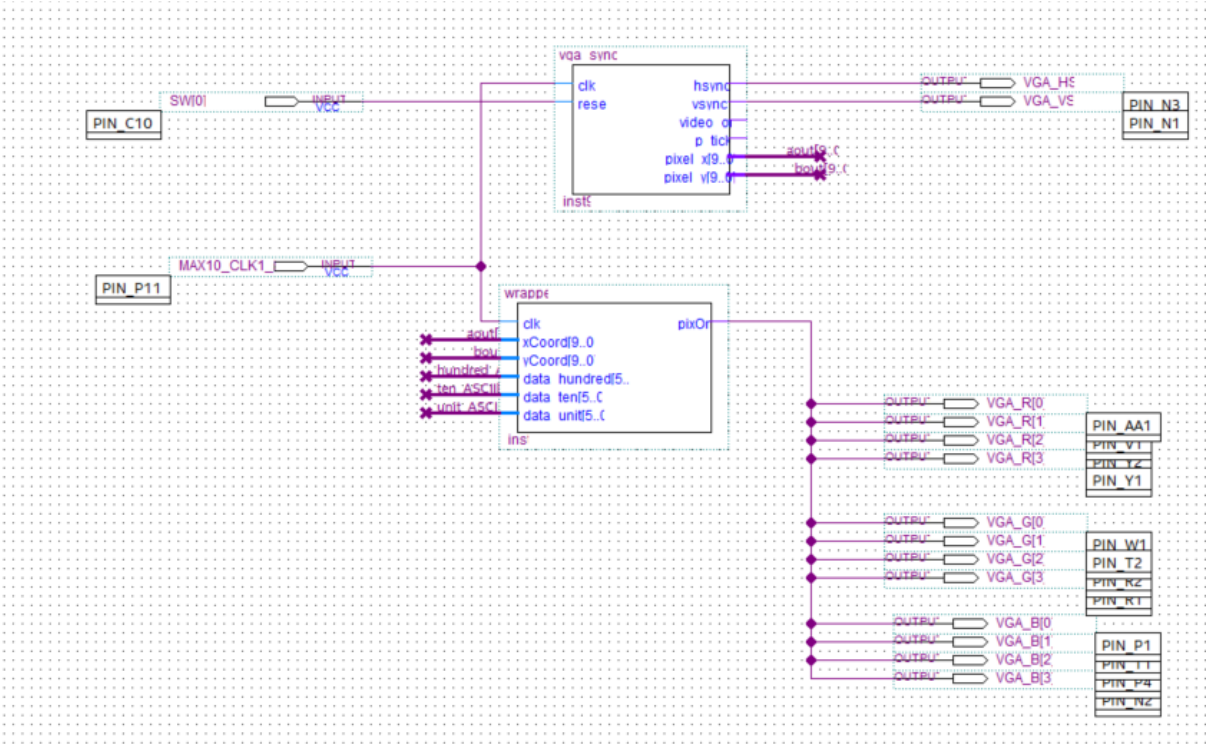

Figura 13 - Bloco do device driver da VGA ilustrado no *Quartus Prime Lite Edition 18.1*.

2 Estudo da plataforma de software

2.1 **Placa Tiva**

A implementação das funcionalidades utilizadas na placa Tiva será via biblioteca *TivaWare*, a qual facilita o uso e configuração dos componentes da placa. Sendo assim, abaixo estão descritas as funções que foram utilizadas por cada componente empregado.

2.1.1 **Funções do sistema**

-uint32_t **SysCtlClockFreqSet**(uint32_t ui32Config, uint32_t ui32SysClock)
Description: Configures the system clock.

-uint32_t **SysCtlClockGet**(void)
Description: Gets the processor clock rate.

-void **SysCtlPeripheralEnable**(uint32_t ui32Peripheral)
Description: Enables a peripheral.

-bool **SysCtlPeripheralReady**(uint32_t ui32Peripheral)
Description: Determines if a peripheral is ready.

2.1.2 **GPIO**

Para a configuração e uso do GPIO, foram avaliadas o uso das seguintes funções:

-void **GPIOADCTriggerDisable** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Disable a GPIO pin as a trigger to start an ADC capture.

-void **GPIOADCTriggerEnable** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Enables a GPIO pin as a trigger to start an ADC capture.

-void **GPIODirModeSet** (uint32_t ui32Port, uint8_t ui8Pins, uint32_t ui32PinIO)
Description: Sets the direction and mode of the specified pin(s).

-void **GPIOIntClear**(uint32_t ui32Port, uint32_t ui32IntFlags)
Description: Clears the specified interrupt sources.

-void **GPIOIntDisable**(uint32_t ui32Port, uint32_t ui32IntFlags)
Description: Disables the specified GPIO interrupts.

-void **GPIOIntEnable** (uint32_t ui32Port, uint32_t ui32IntFlags)
Description: Enables the specified GPIO interrupts.

-uint32_t **GPIOIntStatus**(uint32_t ui32Port, bool bMasked)
Description: Gets interrupt status for the specified GPIO port.

-void **GPIOPinConfigure** (uint32_t ui32PinConfig)
Description: Configures the alternate function of a GPIO pin.

-int32_t **GPIOPinRead** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Reads the values present of the specified pin(s).

-void **GPIOPinTypeADC** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Configures pin(s) for use as analog-to-digital converter inputs.

-void **GPIOPinTypeGPIOInput** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Configures pin(s) for use as GPIO inputs.

-void **GPIOPinTypeGPIOOutput** (uint32_t ui32Port, uint8_t ui8Pins)
Description: Configures pin(s) for use as GPIO outputs.

-void **GPIOPinWrite** (uint32_t ui32Port, uint8_t ui8Pins, uint8_t ui8Val)
Description: Writes a value to the specified pin(s).

2.1.3 **Timer**

Desse modo, para a configuração e utilização do componente Timer da biblioteca TivaWare, foram utilizadas as funções abaixo. Vale ressaltar que o timer não está relacionado com nenhum componente de hardware diretamente, mas apenas para gerar o trigger de interrupção para os conversores ADC utilizados no sistema.

-void **TimerConfigure** (uint32_t ui32Base, uint32_t ui32Config)
Description: Configures the timer(s).

-void **TimerControlTrigger** (uint32_t ui32Base, uint32_t ui32Timer, bool bEnable)
Description: Enables or disables the ADC trigger output.

-void **TimerEnable** (uint32_t ui32Base, uint32_t ui32Timer)
Description: Enables the timer(s).

-void **TimerIntClear** (uint32_t ui32Base, uint32_t ui32IntFlags)
Description: Clears timer interrupt sources.

-void **TimerIntEnable** (uint32_t ui32Base, uint32_t ui32IntFlags)
Description: Enables individual timer interrupt sources.

-uint32_t **TimerLoadGet** (uint32_t ui32Base, uint32_t ui32Timer)
Description: Gets the timer load value.

2.1.4 **Conversor A/D**

Sendo assim, para configuração e uso do conversor, as funções utilizadas estão descritas abaixo:

-uint32_t **ADCClockConfigGet** (uint32_t ui32Base, uint32_t *pui32ClockDiv)
Description: Returns the clock configuration for the ADC.

-void **ADCClockConfigSet** (uint32_t ui32Base, uint32_t ui32Config, uint32_t ui32ClockDiv)
Description: Sets the clock configuration for the ADC.

-void **ADCHardwareOversampleConfigure** (uint32_t ui32Base, uint32_t ui32Factor)
Description: Configures the hardware oversampling factor of the ADC.

-void **ADCIntDisable**(uint32_t ui32Base, uint32_t ui32SequenceNum)
Description: Disables a sample sequence interrupt.

-void **ADCIntClear** (uint32_t ui32Base, uint32_t ui32SequenceNum)
Description: Clears sample sequence interrupt source

-void **ADCIntEnable** (uint32_t ui32Base, uint32_t ui32SequenceNum)
Description: Enables a sample sequence interrupt.

-void **ADCSequenceConfigure** (uint32_t ui32Base, uint32_t ui32SequenceNum, uint32_t ui32Trigger, uint32_t ui32Priority)
Description: Configures the trigger source and priority of a sample sequence.

-int32_t **ADCSequenceDataGet** (uint32_t ui32Base, uint32_t ui32SequenceNum, uint32_t *pui32Buffer)
Description: Gets the captured data for a sample sequence.

void **ADCSequenceDisable**(uint32_t ui32Base, uint32_t ui32SequenceNum)
Description: Disables a sample sequence.

-void **ADCSequenceEnable** (uint32_t ui32Base, uint32_t ui32SequenceNum)
Description: Enables a sample sequence.

-void **ADCSequenceStepConfigure** (uint32_t ui32Base, uint32_t ui32SequenceNum, uint32_t ui32Step, uint32_t ui32Config)
Description: Configure a step of the sample sequencer.

2.2 Placa *DE10-LITE*

A implementação do software executado no kit *DE10-Lite* foi feita sobre o quartus, no qual foi utilizada linguagem VHDL e blocos de diagrama, nos quais muitos dos componentes já existentes possuem o código VHDL pronto.

2.2.1 GPIO

Para utilização do GPIO, tendo configurado previamente o mapeamento dos pinos do kit (*pin planner*) via arquivo fornecido pela própria fabricante, foi apenas nomeados os pinos com o os nomes pré definidos no *pin planner*, além de setar se o pino é de entrada/saída, como também a tensão default. Como por exemplo, se desejada a utilização do pino GPIO1, é necessário nomear a entrada como GPIO[1].

2.2.2 VGA

Na questão da VGA, por meio de um componente (bloco da Figura 11), no qual as suas entradas recebem 12 pinos (4 pinos para cada cor), além dos pinos de sincronização vertical e horizontal, é possível verificar qual é o pixel corrente analisado e decidir se o mesmo será preenchido. Desse modo, a coloração do pixel será de acordo com os valores passados para cada barramento R, G e B de 4 pinos.

3 Estudo do *ThreadX*

No quesito do sistema *Azure RTOS ThreadX*, para o sistema de controle de temperatura veicular, foram utilizados os seguintes componentes providos pelos serviços do RTOS:

3.1 Thread

Sendo um segmento de programa semi-independente, utilizada para execução "parelela" de código para um propósito dedicado. Sendo assim, as funções utilizadas para configuração, criação e manipulação de uma thread estão as seguir:

-UINT **tx_thread_create**(TX_THREAD *thread_ptr, CHAR *name_ptr, VOID (*entry_function)(ULONG), ULONG entry_input, VOID *stack_start, ULONG stack_size,
UINT priority, UINT preempt_threshold, ULONG time_slice, UINT auto_start);
Descrição: cria uma thread e também configura a forma como a mesma irá atuar(prioridade, função a executar, uso de preempção e tempo de time slice, entre outras configurações).

-UINT **tx_thread_sleep**(ULONG timer_ticks);
Descrição: suspende a thread pelo tempo especificado.

3.2 Sinalizadores de eventos

Os sinalizadores de eventos são uma ferramenta para sincronização entre threads, na qual é um recurso público. Sendo assim, cada sinalizador de evento é representado por um único bit contido dentro do grupo de sinalizadores de eventos de 32 bits. No qual, o grupo será utilizado para comunicação entre as threads presentes, verificando se o sistema está ativado ou desativado. Ainda mais, foram utilizadas as seguintes funções providas pela API da ThreadX:

-UINT **tx_event_flags_create**(TX_EVENT_FLAGS_GROUP *group_ptr, CHAR *name_ptr);
Descrição: cria um grupo de sinalizadores de 32 bits, todos iniciados em 0.

-UINT **tx_event_flags_get**(TX_EVENT_FLAGS_GROUP *group_ptr, ULONG requested_flags, UINT get_option, ULONG *actual_flags_ptr, ULONG wait_option);
Descrição: Obtém o grupo de sinalizadores.

-UINT **tx_event_flags_set**(TX_EVENT_FLAGS_GROUP *group_ptr, ULONG flags_to_set, UINT set_option);
Descrição: Define valores para os sinalizadores de um grupo de sinalizadores.

3.3 Byte pool

O byte pool possui propósito semelhante ao block pool, no qual é para alocação de memória, em contrapartida, é possível especificar a quantidade de memória desejada para alocação. Ainda mais, tal componente é utilizado para conter as pilhas das threads utilizadas.

4 Estudo da operação do controle da temperatura

Desse modo, para operação do controle de temperatura do sistema, foram considerados os conceitos desenvolvidos no documento "CONOPS, Domínio do Problema, Especificação", no capítulo sobre o domínio do problema.

Sendo assim, a equação fundamental da calorimetria indica a quantidade de calor que um corpo de massa *m* e calor específico c absorve ou libera para variar a sua temperatura. Além disso, em um sistema fechado que não troca calor com o ambiente externo, os corpos do interior do sistema trocam calor entre si de modo que a soma de todas as energias térmicas é nula.

$$\Sigma Q = 0$$

Assim, considerando o interior do veículo como um sistema fechado onde somente o ar emitido pelas válvulas são capazes de alterar a temperatura no interior do veículo, então a temperatura desejada (Td) pode ser escrita da seguinte maneira:

$$Qf + Qq = 0$$
$$mf(Td - Tf) + mq(Td - Tq) = 0$$
$$mf(Td - 15) + mq(Td - 40) = 0$$
$$Td = (15mf + 40mq) \div (mf + mq) \quad (1)$$

Assim, a equação (1) contém duas incógnitas: *mf* e *mq*, que estão relacionadas diretamente com a abertura das válvulas, uma vez que a massa de ar frio no interior do veículo depende da quantidade de ar frio está entrando, que por sua vez depende da abertura da válvula de ar frio. O mesmo raciocínio é válido para o ar quente. Por fim, é importante salientar que a equação (1) terá variação de acordo com a temperatura real verificada pelo sensor de temperatura (o qual é simulado por um potenciômetro). Portanto, haverá os seguintes casos:

- Temperatura real maior que a temperatura desejada
  Esse caso é quando ocorre o resfriamento do ambiente, sendo assim, a equação(1) ficará do seguinte modo:

$$Td = (15mf + 40mq) \div (mf + mq) - 2 \quad (2)$$

- Temperatura real menor que a temperatura desejada
  Esse caso é quando ocorre o aquecimento do ambiente, sendo assim, a equação(1) ficará do seguinte modo:

$$Td = (15mf + 40mq) \div (mf + mq) + 2 \quad (3)$$

Portanto, para o cálculo da temperatura, primeiro é verificado em que situação se encontra (aquecimento, resfriamento ou IDLE), visto que a temperatura desejada e temperatura real já estão obtidas. Sendo assim, é considerado um valor inicial da abertura da válvula de ar quente, o qual é 20% em caso de resfriamento e 80% em caso de aquecimento. Desse modo, utilizando as equações acima, é obtido a abertura da válvula de ar fria e verificada se é válida (está entre 0% e 100%). Na situação que não é válida, é tomado um novo valor da abertura da válvula de ar quente (acrescido em um para o estado de resfriamento e decrescido em um na situação de aquecimento), até que seja encontrada uma saída válida.