

Universidade de São Paulo

Escola Politécnica, Engenharia de Computação
PCS3645 - Laboratório Digital II
Turma 3 - Professor Paulo Cugnasca
Bancada B6



DoseMinder

Semana 3.0 do projeto - Relatório

| Nome Completo | N USP |
|---------------------------|--------------|
| Henrique Freire da Silva | 12555551 |
| Mariana Dutra Diniz Costa | 12550841 |

São Paulo, 21 de Novembro de 2023

0 ESPECIFICAÇÕES DO SISTEMA

0.1 PROPOSTA

Em prescrições de uso contínuo de fármacos, é natural que possa se esquecer de ingerir o medicamento correto na hora adequada. Essa condição, porém, tende a ser problemática para um público com condições especiais, tal como idosos e indivíduos que sofrem do mal de Alzheimer.

Propõe-se, então, o desenvolvimento de um dispensador de medicamentos para controle de receita prescrita através de interface por aplicativo mobile e comunicação por protocolo MQTT. Dessa forma, poderia se facilitar a descarga de pílulas/comprimidos sem depender unicamente da recordação do cronograma por parte do paciente, bem como de sua capacidade de distinguir os medicamentos que deve ingerir. Os principais componentes que descrevem o sistema são como segue:

1. Interface gráfica desenvolvida em Flutter;
2. Sensor ultrassônico HC-SR04;
3. Micro servomotor;
4. ESP32;
5. Botão;
6. LEDs e/ou buzzer ativo;
7. LED IR (emissor infravermelho);
8. Fototransistor IR.

De acordo com os recursos de tempo e execução disponíveis, alguns requisitos da ideia de projeto inicial foram restringidos. Ao acionar o sistema manualmente via interface, o usuário define quantos comprimidos precisa tomar. Uma notificação é enviada ao dispositivo do usuário e uma mensagem é publicada no respectivo tópico MQTT. Com isso, a informação estruturada e agora contida na ESP32 é serializada e enviada à FPGA para registro das dosagens de cada fármaco disponível. O escopo do protótipo foi definido para dispor de um tipo de fármaco, no entanto o circuito será desenvolvido de forma generalizada, para que seja possível expandi-lo em outro momento.

Logo após o registro da dosagem e a verificação da presença de medicamento no reservatório, há o acionamento de sinais sonoros e visuais via LEDs e buzzers da maquete, para indicar ao usuário que o circuito está esperando para liberar o medicamento.

O sistema espera, durante certo tempo de timeout, pela proximidade de um recipiente ao sensor ultrassônico e, quando essa aproximação é efetivada, o servomotor responsável pelo reservatório é, então, reposicionado de forma a empurrar o próximo comprimido da fila. Esse processo é repetido até que a dose completa seja despejada. Da mesma forma, caso seja removido o recipiente de forma precipitada, deve-se interromper o processo de dispensa e avisar ao usuário que ainda há remédios a se coletar. O fluxo descrito pode ser verificado nas interconexões apresentadas na Figura 1.

O conteúdo dos reservatórios será organizado em pilha, sendo adequado para um determinado intervalo de tamanho de comprimido e permitindo que o servomotor "empurre" apenas um por vez. Este conteúdo será sempre monitorado via sensor de presença infravermelho para que, quando se chegue ao fim da disponibilidade de um fármaco, seja acionada uma notificação para que o usuário recarregue o dispositivo de forma correta. O sistema ficará travado e voltará a funcionar apenas após a realização da recarga.

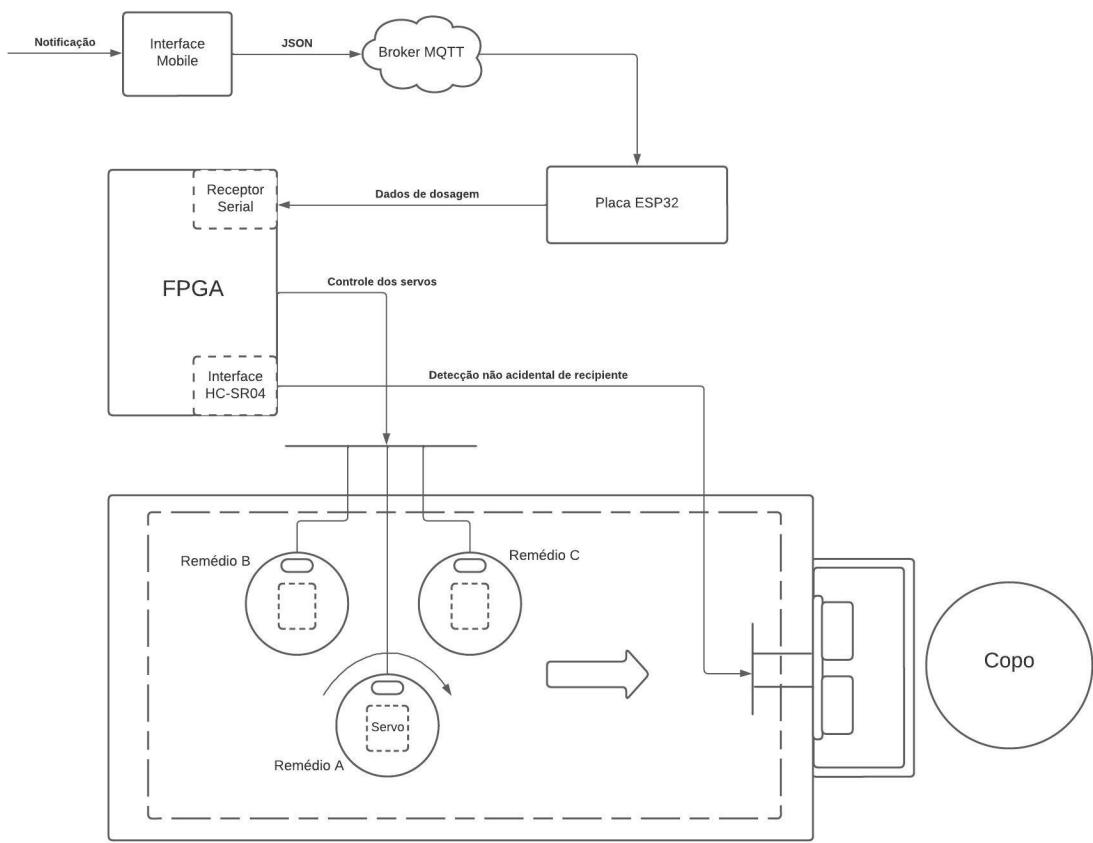


Figura 1: Diagrama de blocos de alto nível.

0.2 REQUISITOS

0.2.1 Requisitos Funcionais

1. Despejo do medicamento quando o sistema for acionado.
2. Capacidade do dispositivo para 1 tipo de medicamento.
3. Sinalização ao usuário no momento de despejo.
4. Interface de monitoramento via *software*.
5. Protótipo em maquete do produto final.
6. Acionamento manual do sistema pelo aplicativo com definição da dosagem.

0.2.2 Requisitos Não Funcionais

7. Usabilidade da Interface.
8. Tratamento da indisponibilidade do medicamento no dispenser.
9. Comunicação interna do sistema por protocolo MQTT.
10. Desenvolvimento descentralizado e integração estável entre os componentes.

1 SEMANA 1

1.1 DESCRIÇÃO DO PROJETO

O projeto discorrido nas seguintes seções descreve as conexões expostas pelo circuito que combina a transmissão serial e o servomotor, bem como a lógica adotada em etapas de processamento entre os componentes mencionados e demais fluxos para a construção do sistema parcial de dispensar de medicamentos.

1.1.1 Projeto do circuito

A entidade principal do sistema desenvolvido, de nome *pill_dispenser.vhd*, contém em sua arquitetura a interconexão entre sua unidade de controle, seu o fluxo de dados e demais portas de interface declaradas, além de sinais de depuração (Apêndice B.1).

A entrada "serial" é recebida via porta serial para definir a dosagem e a saída "pwm" provém do circuito controlador do atuador de forma a reger sua posição. Os sinais "trigger", "echo" e "alert" são deixados para a interface com o sensor ultrassônico, que ainda não faz parte do escopo de desenvolvimento para esta semana.

O projeto dispõe de um pacote *pill_package* que declara os componentes reutilizados nos módulos do design e a tipagem que modela o tema genericamente, tal que possa se centralizar mudanças que afetarão diversos arquivos. O pacote é tal como o Apêndice A.

O conjunto "db_switch" faz a seleção do sinais de depuração de estado dos módulos internos através de multiplexação, conforme C.3, em que "s_state" designa o estado da unidade de controle de cada componente. Os sinais de depuração apresentados a partir da linha 12 de B.1 são conectados diretamente a portas de módulos internos ao fluxo de dados.

1.1.1.1 Unidade de controle

A unidade de controle desenvolvida reage, inicialmente, ao recebimento serial pelo port "serial", aguardando um tempo definido (C.3, linha 2) para garantir que o recipiente permanece no lugar, porém com um *delay* incondicional até posterior implementação da interface de sensor; e persistindo em um ciclo de acionamento do servo, aguardo de um tempo de segurança (C.3, linha 4, em ciclos de *clock*), retorno para a posição de repouso do servo e decremento de registrador(es) aplicável(is) do fluxo de dados (discorridos na seção 1.1.1.2). A entidade desse módulo segue o Apêndice B.2.

A transição de estados de execução segue o algoritmo seguinte, descrito em linguagem natural:

```
1 Aguarda deteccao de dosagem por DETECTED;
2 Enquanto DETECTED alto :
3     Espera pelo CHECK_TIMEOUT;
4     Avanca o motor para despejo do remedio pelo sinal MOVE
5     Espera pelo SAFETY_TIMEOUT;
6     Retorna o motor a posicao inicial e levanta DISCOUNT;
```

Esse mesmo comportamento pode ser verificado na arquitetura apresentada no Apêndice C.2 entre as linhas 19 e 39, em que é descrita a lógica de transições e as saídas de cada estado (máquina de Moore).

O sinal de contagem (*count*, C.2, linha 44) está ativo em 3 estados do ciclo de execução: *check*, *dispense* e *await*. O compartilhamento desse sinal rege a contagem dos dois temporizadores utilizados no fluxo de dados e, para os estados vizinhos (*dispense* e *await*) a contagem do contador

SAFETY_TIMER (Apêndice C.3, linhas 17-26) é repetida uma vez ao se concatenar ciclos de contagem com o fim circular sem desativar o sinal de contagem.

1.1.1.2 Fluxo de dados

O Fluxo de Dados (Apêndice B.3) realiza a integração entre diferentes módulos, que consistem atualmente no controle de pwm, a recepção serial, dois contadores de tempo e um contador regressivo de garantia da dosagem correta.

A transmissão serial foi padronizada no protocolo 7O1, onde os 3 bits menos significativos representam a quantidade de comprimidos que deve ser liberada naquela dose, os próximos 4 identificam o container de remédios e, por fim, um último bit de paridade ímpar. É importante salientar que a identificação de container foi uma decisão de projeto para generalizar o sistema e possibilitar expansão, porém, o escopo só abordará um único reservatório de medicamento.

Ao receber a palavra serial pelo sinal *s dosage*, verifica-se o container definido e, caso seja 0 (o único disponível nesse escopo), o contador inverso *downwards counter* é habilitado. Seu objetivo é realizar uma contagem regressiva, partindo dos 3 bits menos significativos recebidos até o zero. A realização de uma contagem, ou seja, o decremento de um valor, é indicado pelo sinal de controle *discount* vindo da Unidade de Controle no momento oportuno do ciclo. Assim que a contagem chega ao fim, um sinal de alerta é ativado e enviado para a Unidade de Controle pela interface principal do circuito.

O circuito de pwm, definido para gerar pulsos de 4 diferentes larguras, foi instanciado e seu parâmetro *width* foi modulado de forma a assumir sempre um de dois valores: 0 ou 75000 (representativo de um pulso de 1500 ns). Sendo assim, ao receber o sinal de controle *move*, o servomotor é movimentado, empurrando consigo a pílula que deve ser despejada. Da mesma forma, quando o despejo é finalizado, ele retorna para a posição inicial até que o próximo comprimido precise ser empurrado.

Por fim, os dois contadores *check timer* e *safety timer* ficam ativos durante os estados de "check", "dispense" e "await" e são garantidores da segurança do sistema. O primeiro deles, que será posto à prova nas semanas subsequentes, está relacionado ao sensor ultrassônico e existe para garantir que a detecção de um recipiente à espera de medicamento foi efetivamente realizada durante 2 segundos, tempo considerado suficiente para realizar essa verificação com estabilidade. Já o temporizador de segurança conta 0,5 segundo e assegura que o atuador não correrá risco de ser forçado por instantes curtos demais, sendo sempre ativado por um tempo mínimo de funcionamento.

1.1.2 Estratégia de testes

O *testbench* do circuito de interface foi configurado para adotar os casos da Tabela 1. Para cada iteração rodada, a largura de pulso é definida PWM de acordo com a terceira coluna da tabela.

Com a simulação do envio do sinal serial, o circuito deve realizar a movimentação do atuador, ou seja, enviar o pulso pwm de acordo com a quantidade de vezes requisitada. Isso só deve ocorrer, porém, quando o container selecionado for o "0", ou seja, o único atualmente disponível. A nível de teste laboratorial, a estratégia será verificar efetivamente o movimento do servomotor para cada reprodução desses envios via terminal TeraTerm.

| Teste | Recepção Serial (big endian) | Largura de pulso (ns) | Ciclos de <i>dispense</i> |
|-------|------------------------------|-----------------------|---------------------------|
| 1 | 10000000 | 1500 | 1 |
| 2 | 11000001 | 1500 | 3 |
| 3 | 11101001 | 0 | 0 |
| 4 | 01100001 | 1500 | 6 |

Tabela 1: Casos de teste definidos no *testbench*.

1.1.3 Simulação e Síntese do Design

Depois de definidos os casos de testes em *testbench*, conforme apresentados na Tabela 1, o projeto foi integrado ao *software ModelSim* para fins da simulação final e geral. Desse modo, foi possível simular os cenários propostos e acompanhar o comportamento de cada sinal de entrada, saída e depuração ao longo do funcionamento do circuito.

É importante ressaltar que, apenas para fins de simulação, os tempos de *check_timeout*, *safety_timeout*, foram alterados para otimizar o processo de testagem e serão reconfigurados para os testes laboratoriais. No entanto, não foi possível adequar o período e a largura do pulso PWM para parâmetros tangíveis de simulação, que conta então com o sinal "move" para verificação. A movimentação do atuador será, portanto, testada em laboratório.

Foram realizados, portanto, 4 casos de teste, estando o primeiro representado na Figura 2. Nele, inicia-se recebendo uma entrada serialmente e, assim que a leitura é finalizada, o circuito entra no estado "check", enquanto o sinal de dosagem e o contador de pílula são configurados inicialmente. Depois que ocorre o timeout desse estado, inicia-se o "dispense", em que o sinal de "move" é ativado e, assim que termina o "safety timeout", a quantidade de pílulas é descontada em um por meio da ativação do sinal de controle "discount". Assim, com a contagem regressiva atingindo zero, nenhuma pílula mais deve ser liberada e o circuito entra em "idle" até que receba uma nova transmissão serial, já no caso de teste seguinte. É importante perceber, também, o sinal de saída "alert" que, além de indicar para o usuário que o circuito está pronto para liberar um comprimido, ajudará no controle do funcionamento do sensor ultrassônico.

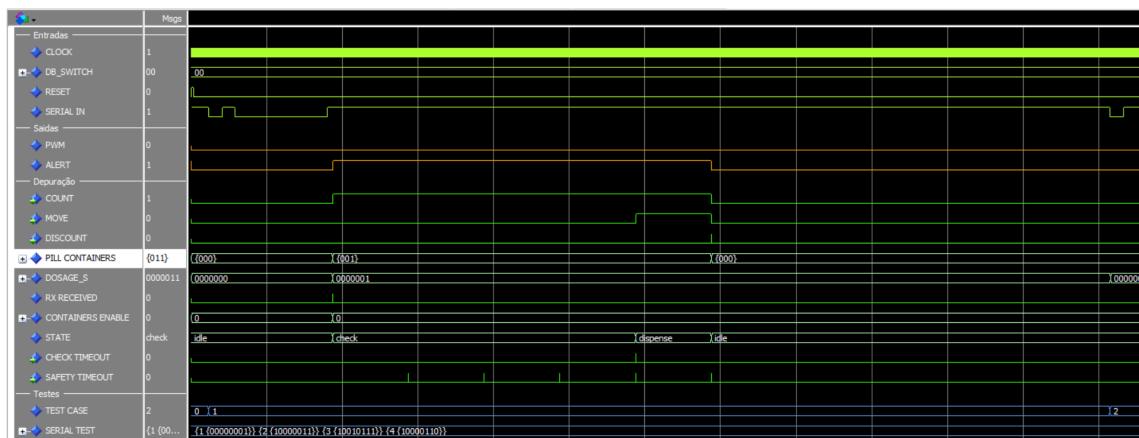


Figura 2: Formas de onda da simulação do circuito para o caso de liberar 1 pílula.

Os 3 casos de teste seguintes, dispostos entre as Figuras 3 e 5, seguem as conformidades do primeiro, registrando sempre o sinal recebido serialmente e contando regressivamente a liberação de cada comprimido, enquanto se completa os ciclos dos estados de "check", "dispense" e "await". É importante destacar que, na Figura 4, a entrada serial seleciona um reservatório inexistente, visto

que o projeto abarca apenas o dispenser 0. É por isso que, nesse caso, o circuito se comporta de modo a não movimentar o atuador, não acionar sinais de atenção e não dispensar nenhum comprimido.

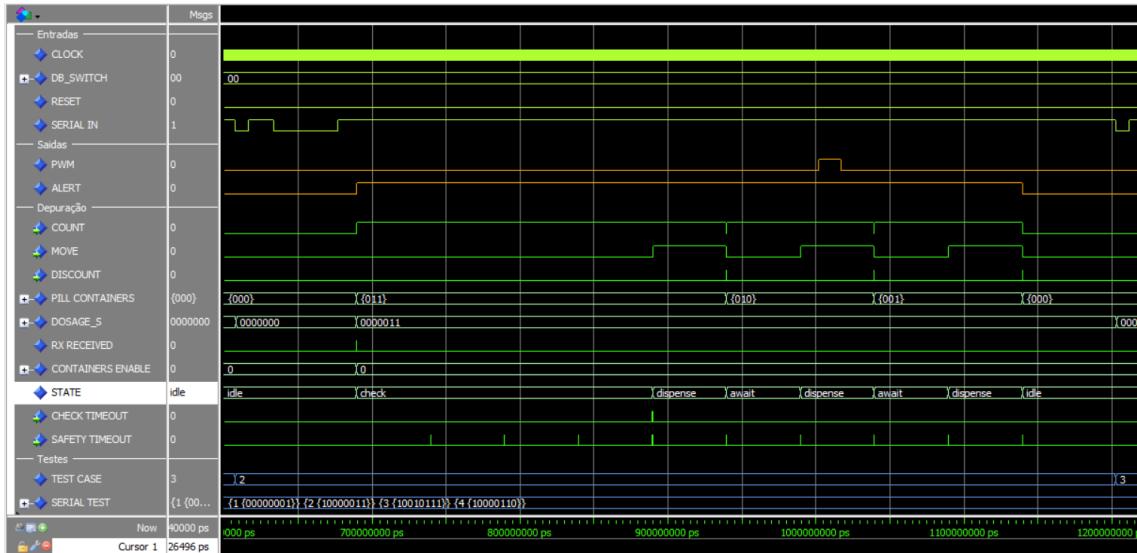


Figura 3: Formas de onda da simulação do circuito para o caso de liberar 3 pílulas.

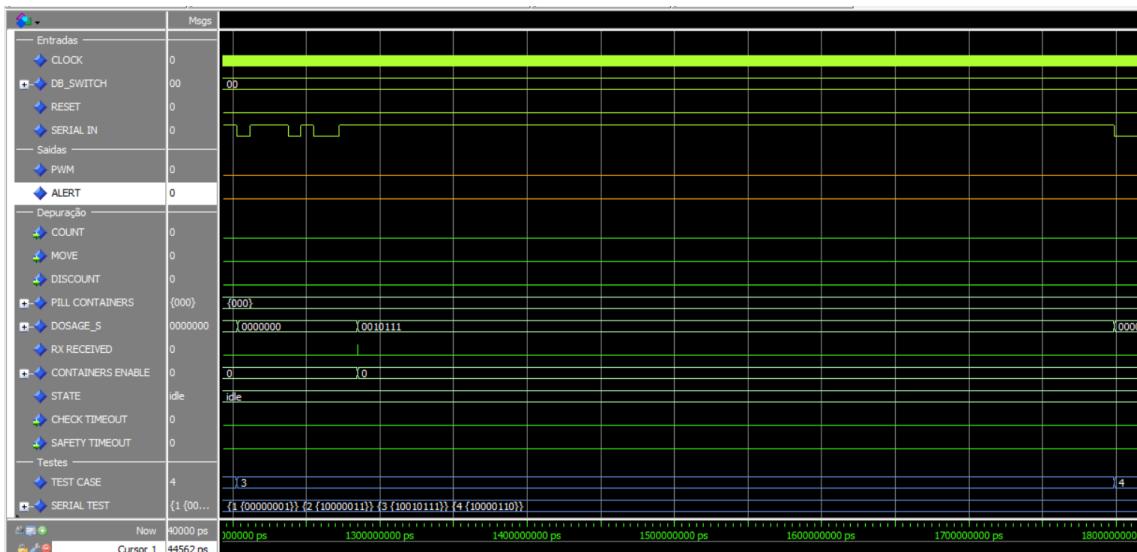


Figura 4: Formas de onda da simulação do circuito para o caso de selecionar um dispenser inexistente.

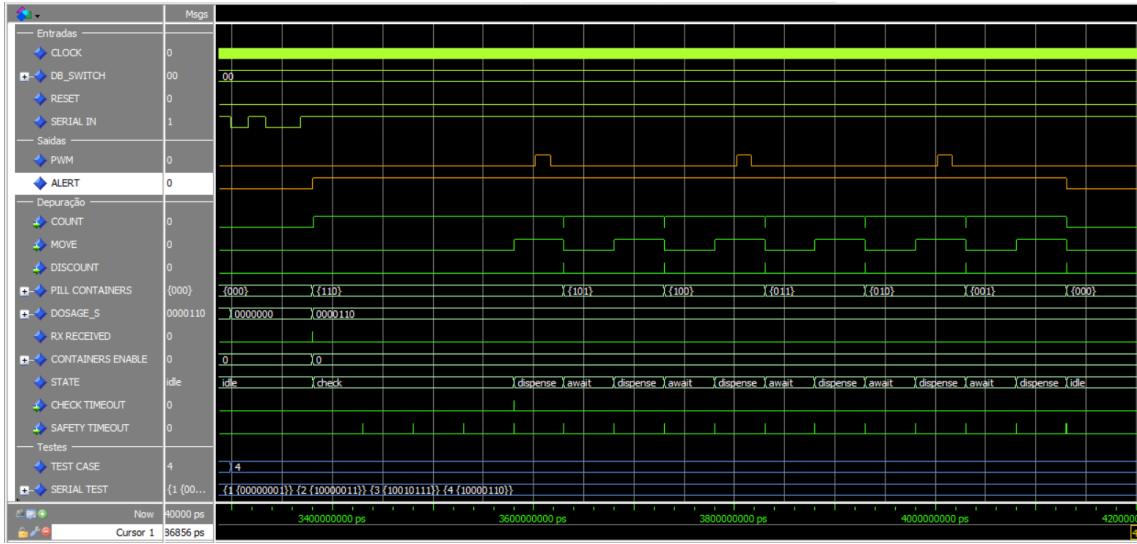


Figura 5: Formas de onda da simulação do circuito para o caso de liberar 6 pílulas.

Após realizar as simulações e confirmar, de fato, o bom funcionamento do circuito projetado para a integração do servomotor com a porta serial, os módulos do projeto foram importados para um novo arquivo na ferramenta *Intel Quartus Prime*, onde foi possível analisar cada componente e sintetizar o circuito final.

Partindo dessa síntese, foram criadas visualizações por meio da ferramenta *RTL viewer*, através da qual foi possível verificar os diagramas de bloco do circuito e confirmar sua compatibilidade com as especificações definidas no capítulo 3. Nesse sentido, foram ilustrados o detalhamento de implementação do novo componente contador regressivo (Figura 6), os componentes formadores do Fluxo de Dados (Figura 7) e o circuito completo do dispenser (Figura 8).

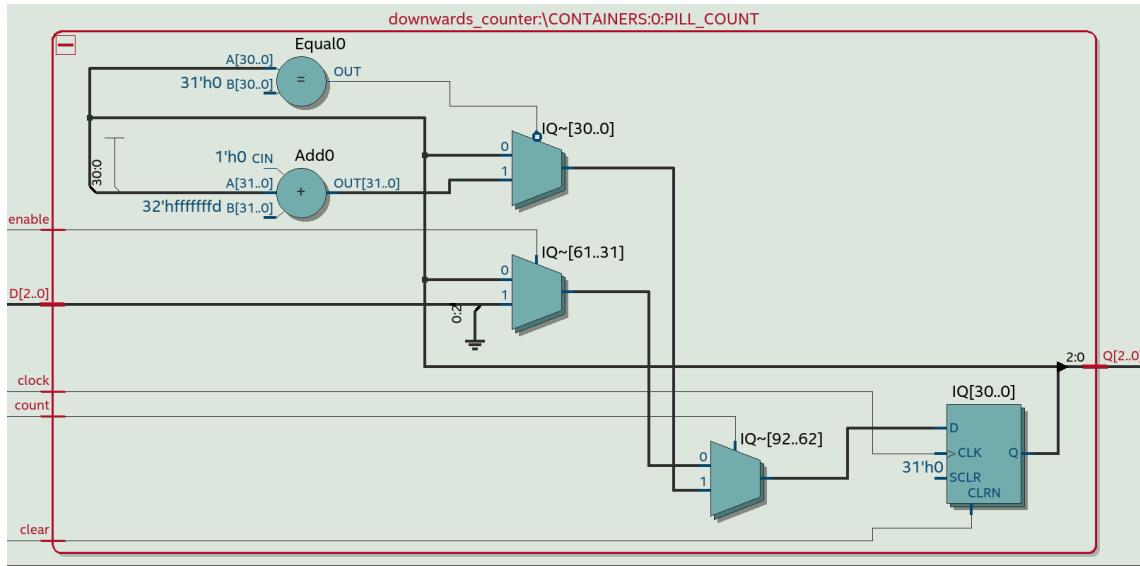


Figura 6: Diagrama de blocos RTL do novo componente "downwards counter".

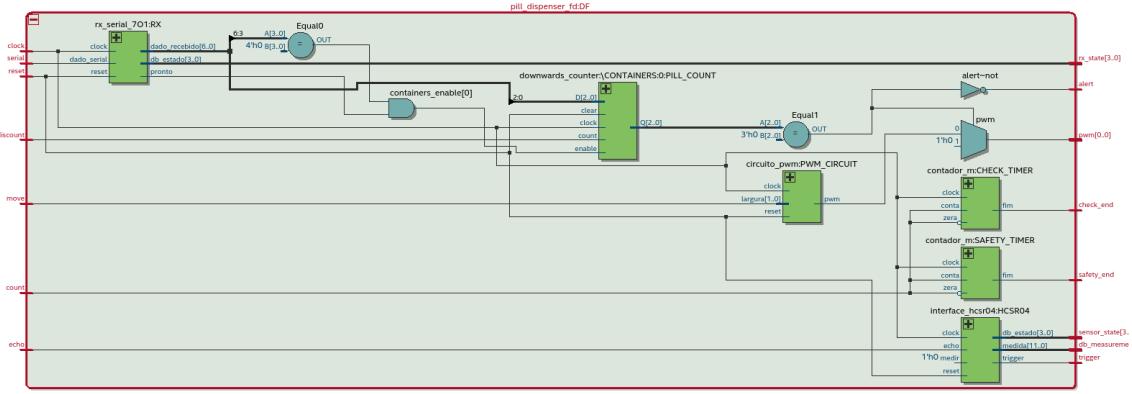


Figura 7: Diagrama de blocos RTL para o fluxo de dados do circuito.

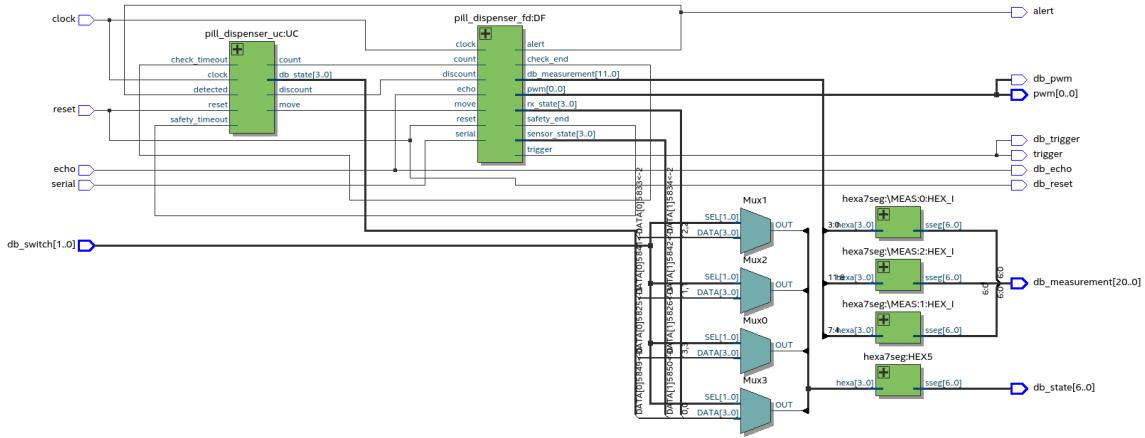


Figura 8: Diagrama de blocos RTL para a entidade principal do circuito.

Com isso, a ferramenta interna *State Machine Viewer* foi utilizada para conferir a estrutura da máquina de estados que rege a Unidade de Controle do projeto, assim como as transições entre estados e os sinais de condição que influenciam no seu funcionamento. Esses aspectos estão representados na Figura 9.

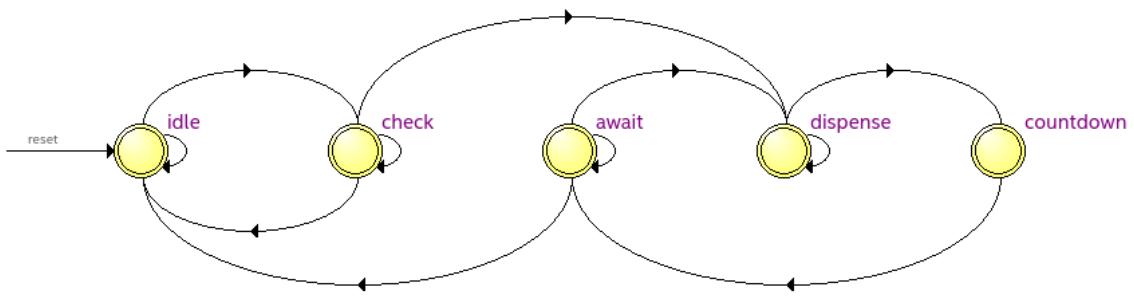


Figura 9: Diagrama da máquina de estados que rege a Unidade de Controle do circuito.

Finalmente, foi configurada a pinagem conforme a Tabela ?? descrita na Seção 4 e o projeto foi exportado para um arquivo de extensão *qar* que poderá ser diretamente utilizado em laboratório para elaboração na placa FPGA.

1.2 ATIVIDADES EXPERIMENTAIS

Em laboratório, o circuito final do projeto foi sintetizado e programado na placa FPGA DE0-CV. A montagem foi realizada em etapas intermediárias, seguindo o que havia sido anteriormente definido no Plano de Execução Experimental. Primeiramente, foi realizada a montagem intermediária para testar a comunicação serial. Nesse sentido, o pino GND foi conectado à referência de Terra da FPGA, os terminais RD e TD da placa MAX3232 foram colocados em curto e a porta serial foi conectada à entrada USB computador. Com isso, o terminal TeraTerm foi utilizado e com ele verificada a transmissão de caracteres via serial no protocolo 7O1. Com isso, foi possível conectar o terminal TD ao pino GPIO do sinal *serial* de entrada do circuito para possibilitar, enfim, a transmissão da dosagem requisitada.

O próximo passo foi integrar o módulo do servomotor, por meio da conexão de seus terminais GND e VCC (5V), além do sinal de controle *pwm* vindos da GPIO. Com isso, foi realizada a integração dos dois módulos e finalizada a montagem do sistema atual, conforme apresentado na Figura 10. Nela, o display hexadecimal mais à esquerda reflete o estado do componente principal do circuito (o que pode ser multiplexado pelas chaves SW9 e SW8) e o LED0, mais à direita, foi utilizado como saída do sinal de alerta.

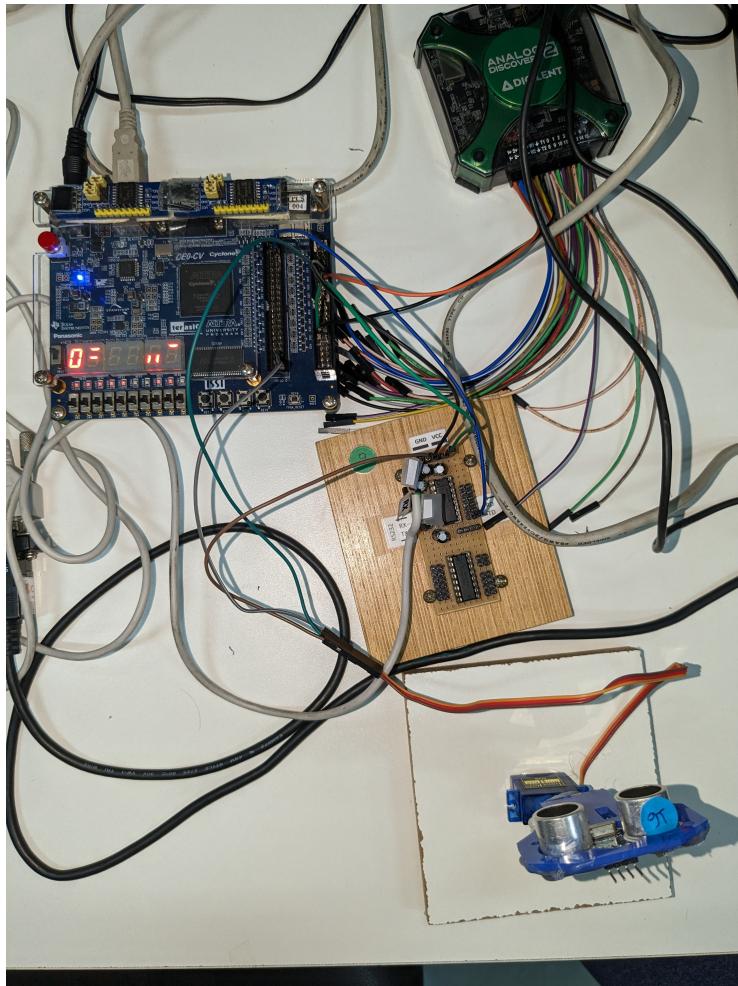


Figura 10: Montagem experimental do sistema atual.

Nesse momento seriam, então, iniciados os testes propostos na Tabela 1. No entanto, ao codificar os valores seriais dos testes para ASCII, percebeu-se que se tratavam de caracteres especiais, o que dificultou o envio por meio do teclado. Por isso, foi feita uma adequação temporária no circuito, configurando o "dispenser" padrão (único existente) com o valor 14 ("1110") em vez de 0 ("0000"). Dessa forma, os casos de teste foram alterados (Tabela 2) de modo a abarcarem o novo valor de identificação do dispenser, assim possibilitando o trabalho com caracteres ASCII entre "q" e "w".

| Teste | Recepção Serial (big endian) | Caractere | Largura de pulso (ns) | Ciclos de <i>dispense</i> |
|-------|------------------------------|-----------|-----------------------|---------------------------|
| 1 | 10001110 | q | 1500 | 1 |
| 2 | 11001111 | s | 1500 | 3 |
| 3 | 11101010 | W | 0 | 0 |
| 4 | 01101111 | v | 1500 | 6 |

Tabela 2: Casos de testes experimentais para envio de dosagens.

Com esses novos valores de teste, adequados à restrição dos caracteres, foi possível verificar o bom funcionamento do circuito projetado por meio da movimentação de ida e volta do servomotor na quantidade exata de vezes determinada pela entrada serial. Este movimento representa, por fim, o ato de empurrar uma pílula para a saída do dispenser e retornar para a posição inicial, seja para empurrar uma nova ou finalizar o processo daquela dose. Além disso, a indicação da saída de alerta no LED0 ocorreu conforme esperada, permanecendo ativa até que o despejo daquela requisição seja finalizado.

Como depuração, foram utilizados o display HEX5 para o estado e o canal 1 do osciloscópio do Analog Discovery para conferir o correto envio do sinal de PWM, conforme representa a Figura 11 a seguir.

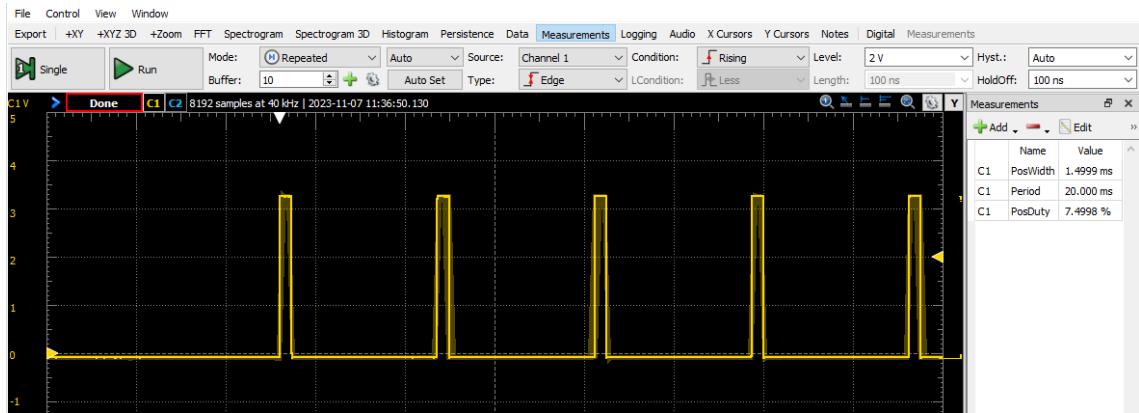


Figura 11: Verificação do envio e da largura do sinal PWM no osciloscópio.

2 SEMANA 2

Para a segunda semana do projeto, foi dada continuidade aos requisitos #1, #2 e #3 com a finalização da implementação do circuito lógico por meio da integração com a interface do sensor ultrassônico medidor de distância. Além disso, ao longo dessa semana foi dado início à montagem da maquete física do produto final, representada pelo requisito #5.

2.1 DESCRIÇÃO DO PROJETO LÓGICO

O projeto lógico do sistema descreve as conexões expostas pelo circuito para integração da transmissão serial, do sensor ultrassônico e do servomotor. A lógica adotada entre os componentes e demais fluxos foi minimamente alterada em relação à semana anterior para adequar a inserção da medida de distância e funcionamento do dispensador apenas quando um recipiente for detectado à espera. Essas alterações estão descritas nas seções.

2.1.1 Projeto do circuito lógico

A entidade principal do sistema desenvolvido, de nome *pill_dispenser.vhd*, foi mantida conforme sua interconexão entre sua unidade de controle e seu fluxo de dados, além dos sinais de depuração (Apêndice B.1). A principal alteração em relação à semana anterior foi a adição do sensor ultrassônico HC-SR04 no fluxo de dados, agora dando seguimento aos sinais de "trigger" e "echo" de interface do circuito completo.

2.1.1.1 Fluxo de dados

No Fluxo de Dados, que pode ser visto no Apêndice C.3, foi adicionado o componente *interface_hcsr04*, conectando respectivamente os sinais de "echo" e "trigger" de interface do circuito. O sinal de "medir" do sensor foi ativado conforme "s_alert", o que garante que sempre que o sistema estiver em alerta, isto é, no momento de dispensa da dosagem solicitada, o sensor estará constantemente medindo sua distância ao objeto mais próximo.

A medida realizada pelo sensor é comparada com o valor de 6 cm e, caso seja menor que este limite, ativa o sinal *s_close* indicador de que há um recipiente perto. Este sinal, por sua vez, aliado ao alerta do circuito, é utilizado para ativar o sinal de controle *detected* que será enviado para a UC. Além disso, a medida também é direcionada para o sinal de depuração que aparecerá nos displays de 7 segmentos da placa FPGA.

À entidade principal do Fluxo de Dados (Apêndice B.3) foi adicionado apenas este sinal de saída *detected*, que indica para a Unidade de Controle que o sistema tem remédios para liberar e que há um recipiente pronto para recebê-los.

Além disso, foi adicionado um contador de *timeout* ao componente *interface_hcsr04* para abarcar possíveis erros operacionais do sensor. Com isso, caso não seja detectada uma resposta de *echo* dentro de 0.5 segundo, o sensor envia um novo pulso de *trigger*, assim impedindo que o funcionamento do dispositivo seja interrompido.

2.1.1.2 Unidade de controle

A entidade da Unidade de Controle não foi modificada, mas agora o sinal de entrada *detected* indica não apenas o alerta do sistema, mas também a proximidade de um recipiente e a permissão para liberar os medicamentos.

A lógica da máquina de estados (Apêndice C.2) foi alterada de modo que o estado *dispense* só seja atingido caso *detected* e o *timeout* de checagem estejam ativos, indicando que o recipiente foi

detectado por tempo suficiente e seguro para iniciar a liberação dos remédios.

2.1.2 Estratégia de testes

O *testbench* do circuito de interface foi configurado para adotar os casos da Tabela 1. Foi definida uma única situação do envio serial da palavra "01110110", que solicita a liberação de 6 doses do medicamento.

Para cada caso de teste, a largura de pulso do echo é definida de acordo com a segunda coluna da tabela. Esse sinal de echo é enviado periodicamente durante 1,5 ms, quando o caso de teste é trocado para a próxima largura. Dessa forma, pode-se verificar o funcionamento do circuito para o caso de esperar a presença de um recipiente para começar a liberação de remédio, interromper o funcionamento caso o recipiente se distancie e retornar de onde parou quando ele voltar para o limite estabelecido. A nível de teste laboratorial, a estratégia será verificar efetivamente o movimento do servomotor e dos leds de depuração para cada reprodução desses pulsos via variação da distância ao sensor.

| Teste | Largura do Echo (ns) | Distância (cm) | Funcionamento do <i>dispense</i> |
|-------|----------------------|----------------|----------------------------------|
| 1 | 1200 | 20 | interrompido |
| 2 | 294 | 5 | ativo |
| 3 | 580 | 10 | interrompido |
| 4 | 235 | 4 | ativo |

Tabela 3: Casos de teste definidos no *testbench*.

2.1.3 Simulação e síntese do design

Depois de definidos os casos de testes em *testbench*, conforme apresentados na Tabela 3, o projeto foi integrado ao software *ModelSim* para fins da simulação final e geral. Desse modo, foi possível simular os cenários propostos e acompanhar o comportamento de cada sinal de entrada, saída e depuração ao longo do funcionamento do circuito.

É importante ressaltar que, apenas para fins de simulação, os tempos de *check_timeout*, *safety_timeout*, foram alterados para otimizar o processo de testagem e serão reconfigurados para os testes laboratoriais. No entanto, não foi possível adequar o período e a largura do pulso PWM para parâmetros tangíveis de simulação, que conta então com o sinal "move" para verificação. A movimentação do atuador será, portanto, testada em laboratório.

Foi realizado, então um caso de teste único que engloba diferentes cenários, demonstrado na Figura 12. Ele é iniciado ao receber uma entrada serialmente para a dispensa de 6 comprimidos. Assim que a leitura é finalizada, o circuito envia um pulso de Trigger por meio do sensor. Alguns instantes depois, o testbench inicia o envio do pulso de Echo de acordo com os cenários de teste definidos. A partir disso, é possível verificar no sinal *measurement* as medidas em centímetros calculadas logo após cada pulso e, no sinal *move* a correta indicação para movimentação do servomotor ocorrendo apenas quando a distância medida é menor ou igual a 5 cm.

Quando, no meio do processo, o pulso de echo é alargado para 10 cm, representando a retirada do recipiente da frente do sensor, o sistema interrompe seu funcionamento, voltando ao estado de *idle* até que seja detectada uma distância dentro dos limites. É importante notar que a contagem regressiva de *pill containers* continua válida e o sistema é capaz de retornar a dispensa de medicamentos exatamente de onde havia parado, podendo assim finalizar a liberação dos 6 comprimidos.

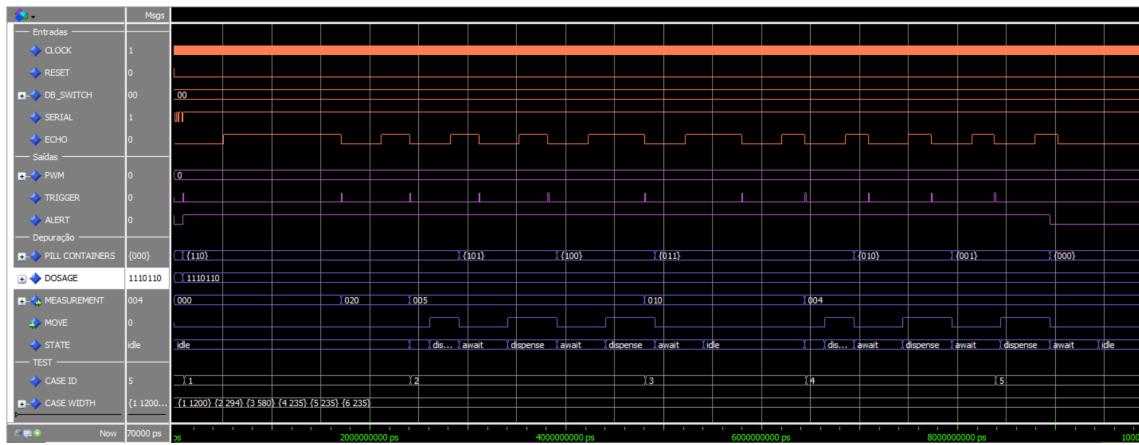


Figura 12: Formas de onda da simulação do circuito para diferentes larguras de echo.

Após realizar as simulações e confirmar, de fato, o bom funcionamento do circuito projetado para a integração do sensor ultrassônico ao restante do dispositivo, os módulos do projeto foram importados para um novo arquivo na ferramenta *Intel Quartus Prime*, onde foi possível analisar cada componente e sintetizar o circuito final.

Partindo dessa síntese, foram criadas visualizações por meio da ferramenta *RTL viewer*, através da qual foi possível verificar os diagramas de bloco do circuito e confirmar sua compatibilidade com as especificações definidas no capítulo 2.1. Nesse sentido, foram ilustrados os componentes formadores do Fluxo de Dados (Figura 13) e o circuito completo do dispenser (Figura 14).

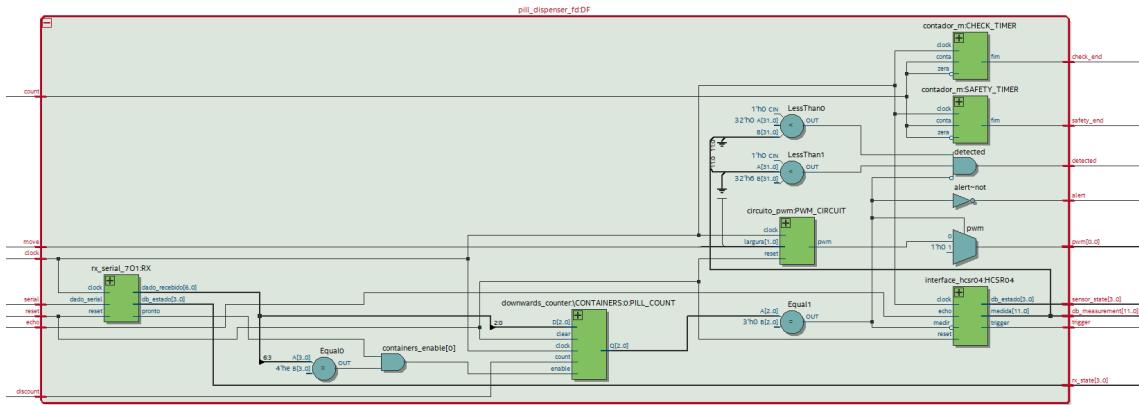


Figura 13: Diagrama de blocos RTL do Fluxo de Dados do circuito final do *dispenser*.

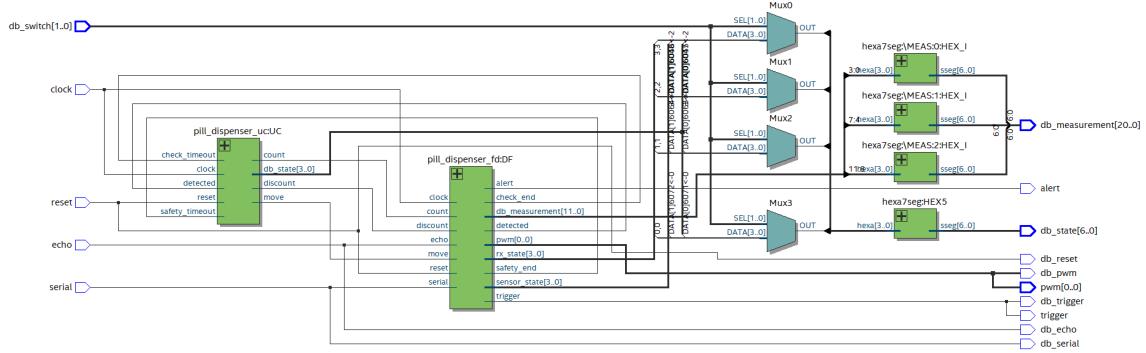


Figura 14: Diagrama de blocos RTL do circuito completo final do *dispenser*.

Com isso, a ferramenta interna *State Machine Viewer* foi utilizada para conferir a estrutura da máquina de estados que rege a Unidade de Controle do projeto, assim como as transições entre estados e os sinais de condição que influenciam no seu funcionamento. Esses aspectos estão representados na Figura 15.

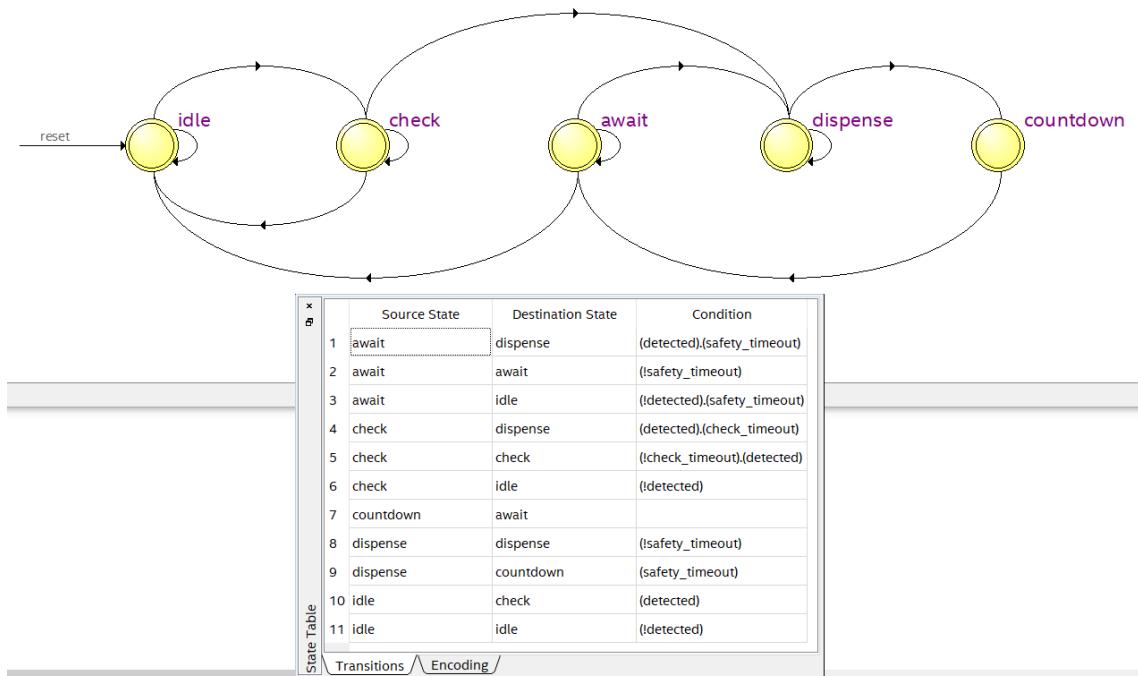


Figura 15: Diagrama da máquina de estados que rege a Unidade de Controle do circuito completo.

Finalmente, foi configurada a pinagem conforme a Tabela 4 descrita na Seção 2.3 e o projeto foi exportado para um arquivo de extensão *qar* que poderá ser diretamente utilizado em laboratório para elaboração e programação na placa FPGA.

2.2 MODELAGEM DA MAQUETE

A estrutura de suporte do projeto foi desenvolvida em CAD através do software Fusion 360, utilizado em nuvem através da aplicação web licenciada. O resultado final é como apresentado no Apêndice E.1.

O design consiste de 5 partes principais: base; suporte do motor; rampa de despejo; pilha de armazenamento; êmbolo. A identificação individual desses componentes se encontra na vista explodida em E.2.

Os componentes foram desenvolvidos em módulos, de forma a facilitar sua montagem física, reduzir a complexidade e tempo de impressão, bem como possibilitar a reimpressão de partes específicas, a depender da necessidade de projeto. Além disso, utilizou-se do recurso de casca (comumente conhecido por *shell*) para remover preenchimentos de formas e produzir paredes de espessura constante de 2.5 mm.

Na base, foram definidas aberturas para o perfil do sensor (transmissor, receptor e furos de fixação) e uma ranhura em 'T' para a passagem de fios dos LEDs da vista frontal do modelo e do motor no suporte sobre essa peça. Como escolha de projeto, o sensor fica posicionado na parte traseira do modelo, de tal forma que aumente a distância percebida em suas medições e se reduza mínima de detecção de objetos pelo sistema, tendo por referência a face frontal da montagem, assim viabilizando uma rampa de até 3 cm que alcance um recipiente de até 6.5 cm de distância do sensor (4.5 cm da maquete). O componente base em vista posterior se encontra em E.3.

Para o suporte, elaborou-se um berço com dois pontos de fixação (por parafusos ou abraçadeiras) para o micro servomotor (vide E.4). Além disso, a seção com o encaixe para a peça superior - a pilha (conforme E.2, anotação 5) - contém um degrau que permite a união da rampa (anotação 4) ao conjunto sem a necessidade de demais fixadores.

O êmbolo responsável pelo devido despejo de comprimidos com a movimentação do servomotor possui furos para alinhamento e fixação àqueles das hélices fornecidas com o produto, seja por arame ou abraçadeira. As dimensões da face que atravessa a pilha são 75% do próprio rasgo, tal que não se tenha interferência durante o movimento circular imposto pelo motor.

Por fim, a pilha possui 75 mm de altura, com uma abertura de 19x12 mm para a passagem de comprimidos dimensionados em 16x8 mm, tal que haja uma tolerância de manufatura sem folga no sistema.

2.3 PLANEJAMENTO DA AULA PRÁTICA

Em laboratório, será realizada a síntese do projeto arquivado (extensão *qar*) na placa FPGA DE0-CV segundo a pinagem da Tabela 4.

A montagem experimental será, então, realizada em etapas intermediárias para testar o correto funcionamento de todos os componentes do projeto e suas relações. Esse processo seguirá de acordo com o aqui descrito Plano de Execução Experimental.

2.3.1 Plano de Execução Experimental

Inicialmente, será testada a comunicação pela porta serial - USB, conectando o GND, TX e RX nos pinos correspondentes da placa MAX3232 e deixando em curto os terminais TD e RD. Do outro lado, a porta USB será conectada à entrada devida no computador. Com isso, a ferramenta *TeraTerm* será configurada de acordo com a codificação 7O1 utilizada e serão realizados testes de eco ao digitar um caractere ASCII no terminal, verificando o correto retorno dele pela porta.

Em seguida, será realizada a integração com o módulo do servomotor acoplado ao sensor ultrassônico. Para isso, ele será apropriadamente alimentado pelos terminais VCC e GND da placa FPGA, além de conectado ao sinal de saída *pwm* do circuito de controle do sistema por meio dos acessos GPIO; e aos sinais de *trigger* e *echo*, conforme 4. É importante recordar, desde o início, de referenciar todos os componentes experimentais a um mesmo potencial terra (GND).

A partir desse momento, a placa será ligada e programada com o projeto em questão para a realização dos testes propostos, sempre de forma a se atentar aos sinais de depuração definidos. Além da depuração definida na placa, ainda há os pinos GPIO que apresentam a possibilidade de serem conectados ao Analog Discovery para verificação das ondas, que pode ser bem utilizada na depuração da recepção serial, medição do sensor e controle do motor.

| Sinal | Ligaçāo na placa FPGA | Pino na FPGA | Analog Discovery |
|-------------------|-----------------------|---|------------------|
| clock | CLOCK_50 | PIN_M9 | - |
| reset | chave SW0 | PIN_U13 | - |
| serial | GPIO_0_D1 | PIN_B16 | CH1+ (Scope)* |
| pwm | GPIO_0_D35 | PIN_T15 | - |
| trigger | GPIO_1_D1 | PIN_A12 | - |
| echo | GPIO_1_D3 | PIN_B12 | - |
| alert | led LEDR[0] | PIN_AA2 | - |
| db_measurement[0] | display HEX0 | PIN_U21 PIN_V21 PIN_W22 PIN_W21 PIN_Y22 PIN_Y21 PIN_AA22 | - |
| db_measurement[1] | display HEX1 | PIN_AA20 PIN_AB20 PIN_AA19 PIN_AA18 PIN_AB18 PIN_AA17 PIN_U22 | - |
| db_measurement[2] | display HEX2 | PIN_Y19 PIN_AB17 PIN_AA10 PIN_Y14 PIN_V14 PIN_AB22 PIN_AB21 | - |
| db_switch | chaves SW8 e SW9 | PIN_AB13 PIN_AB12 | - |
| db_state | display HEX5 | PIN_N9 PIN_M8 PIN_T14 PIN_P14 PIN_C1 PIN_C2 PIN_W19 | - |
| db_reset | led LEDR[3] | PIN_Y3 | - |
| db_pwm | GPIO_1_D11 | PIN_J18 | CH2+ (Scope)* |
| db_trigger | GPIO_1_D15 | PIN_J11 | CH1+ (Scope)* |
| db_echo | GPIO_1_D17 | PIN_A15 | CH2+ (Scope)* |

Tabela 4: Pinagem para a montagem experimental.

* Canais de osciloscópio duplicados para depuração em partes do sistema: interface com sensor (*echo* e *trigger*); transmissão serial e controle do motor.

2.4 ATIVIDADES EXPERIMENTAIS

Em laboratório, o circuito final do projeto foi sintetizado e programado na placa FPGA DE0-CV. A montagem foi realizada em etapas intermediárias, seguindo o que havia sido anteriormente definido no Plano de Execução Experimental. Primeiramente, foi realizada a montagem intermediária para testar a comunicação serial com os pinos RD e TD em curto. Com isso, o terminal TeraTerm foi utilizado e com ele verificada a transmissão de caracteres via serial no protocolo 7O1. Então, foi possível conectar o terminal TD ao pino GPIO do sinal *serial* de entrada do circuito para possibilitar, enfim, a transmissão da dosagem requisitada.

O próximo passo foi integrar o módulo do servomotor, por meio da conexão de seus terminais GND e VCC (5V), além do sinal de controle *pwm* vindos da GPIO. Por fim, foi feito o acoplamento do sensor ultrassônico com os testes de conversão de tensão 5V para 3V por meio do osciloscópio do WaveForms.

Com isso, foi realizada a integração de todos os módulos do dispositivo e finalizada a montagem do sistema atual, conforme apresentado na Figura ???. Nela, o display hexadecimal mais à esquerda reflete o estado do componente de interface do sensor (o que pode ser multiplexado pelas chaves SW9 e SW8) e os três à direita mostram a medida de distância em centímetros realizada pelo sensor. Já o LED0, mais à direita, foi utilizado como saída do sinal de alerta.

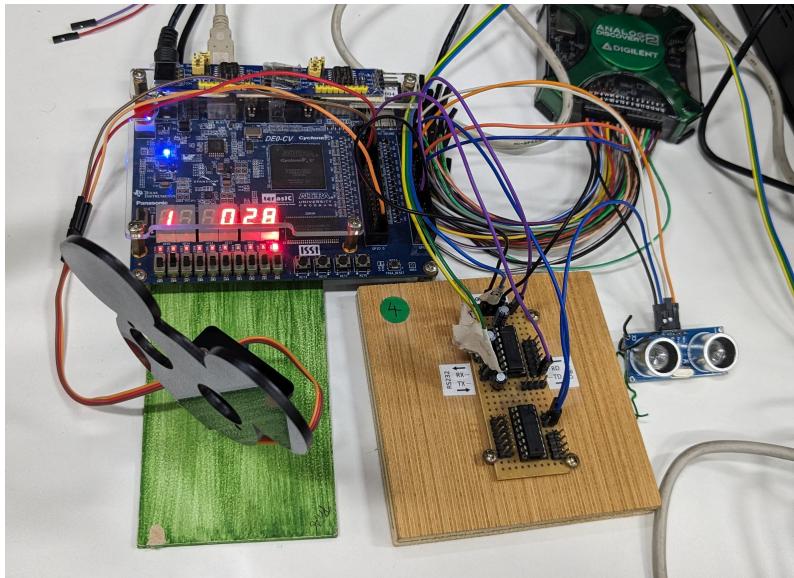


Figura 16: Montagem experimental do circuito completo.

Dando início aos testes, percebeu-se uma dificuldade de funcionamento do sensor ao realizar medidas constantemente, isto é, enviar o trigger da próxima medida logo após o fim do echo recebido da anterior. Por isso, foi instanciado um novo contador de *delay* no Fluxo de Dados da interface do sensor, que aguarda 100ms entre o fim da recepção do echo e o início do novo trigger. Desse modo, o circuito realiza aproximadamente 10 medidas por segundo, que é uma frequência suficiente para não sobrecarregá-lo e ainda garantir o funcionamento dentro do esperado para o usuário. A nova relação periódica de Trigger e Echo pode ser vista na Figura 17 construída pelo osciloscópio da ferramenta WaveForms.

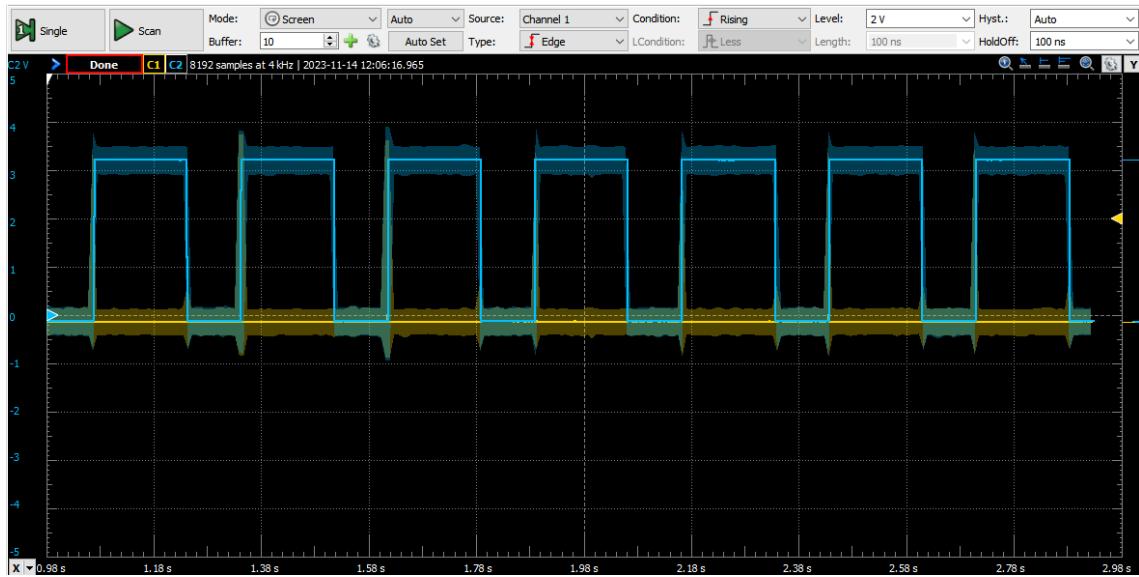


Figura 17: Sinais de Trigger e Echo no osciloscópio

Após essa readequação, os testes correram como previsto: ao enviar serialmente o caractere correspondente ao despejo de 6 pílulas, o servomotor só se movimentou quando o sensor identificou um objeto a, no máximo, 5 cm de distância. Durante o processo, quando o objeto foi reposicionado mais longe, o servo teve seu movimento interrompido, enquanto o LED de alerta continuou aceso em aviso ao usuário de que o despejo ainda não havida sido finalizado. Em seguida, ao reaproximar o objeto do sensor, o servomotor voltou à movimentação partindo da etapa em que havia parado, até completar o despejo dos 6 medicamentos.

3 SEMANA 3

Para a terceira semana do projeto, foram iniciados os requisitos #4, #6, #7 e #9 com o desenvolvimento do aplicativo mobile e da implementação de comunicação via MQTT com o microcontrolador ESP32.

3.1 DESENVOLVIMENTO MOBILE

Para o desenvolvimento mobile focado em sistema Android, decidiu-se utilizar o *framework* Flutter em linguagem Dart. A base de dados, capaz de armazenar os remédios, dosagens e recipientes de cada usuário autenticado, foi integrada por meio do sistema Firebase fornecido pela Google.

Foram criadas, até o momento, duas telas simples para teste de funcionalidades que poderão ter recursos visuais melhor aproveitados posteriormente. A tela inicial contém a logo e o título do produto *DoseMinder*, além de indicar para o recurso de login com a conta Google do usuário, a fim de permitir a autenticação e a integração com a base de dados pessoal de cada um.

Em seguida, direcionado para uma nova tela, o usuário pode verificar as possíveis requisições que pode fazer para o dispositivo e, ao clicar em uma dessas opções, ele dispara a publicação de uma mensagem MQTT. Nesse sentido, o circuito lógico programado na FPGA recebe, por meio da intermediação da ESP32 via WiFi, a palavra serial que indica o recipiente e a dosagem do remédio a ser despejado. Em versões futuras, será possível alterar a base de dados diretamente pelo aplicativo, permitindo que o usuário crie, atualize ou apague requisições cadastradas.

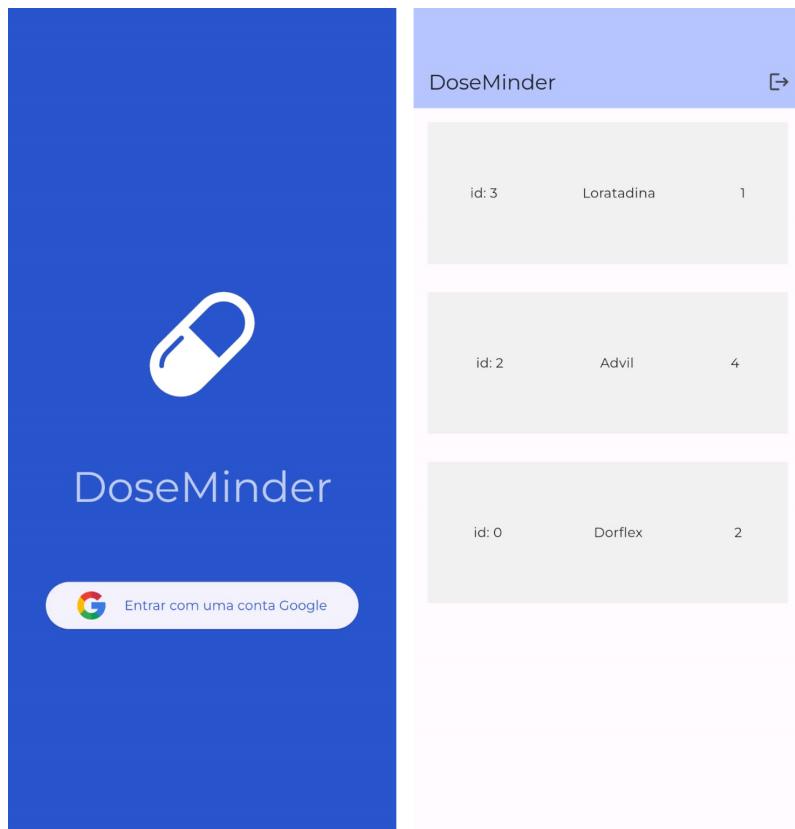


Figura 18: Telas de autenticação e acionamento do dispositivo via interface mobile.

3.1.1 Base de dados e autenticação Firebase

De forma a melhor complementar a solução proposta pelo projeto, configurou-se um projeto Firebase em nuvem com autenticação por conta Google e banco de dados NoSQL. A autenticação garante uma partição da base de dados em coleções identificados pelo UID gerado no registro de conta e permitindo um fácil controle de acesso pela descrição:

```
rules_version = '2';

service cloud.firestore {
    match /databases/{database}/documents {
        match /{userId}/{document=**} {
            allow read, write: if request.auth != null && request.auth.uid == userId;
        }
    }
}
```

em que toda subsequente documento ou coleção é de livre acesso ao usuário dono da coleção raiz, definido pelo *id* recebido na requisição.

3.2 COMUNICAÇÃO MQTT

A publicação de mensagens via protocolo MQTT por parte do aplicativo foi implementada por meio da biblioteca em Dart *mqtt.client*. Depois de autenticar a conta do usuário, o aplicativo estabelece a conexão MQTT ao broker privado EMQX Cloud, passando a ser capaz de publicar e se inscrever em tópicos definidos para a comunicação com o circuito lógico.

Com base na classe modelo definido, contendo as informações pertinentes à receita, tal como o nome do medicamento, dose por ciclo e o recipiente de estoque (para o produto a ser apresentado, sempre o 0); é gerado um mapa JSON que represente esse objeto selecionado e enviado em forma de *string* para o tópico definido.

Do outro lado da operação, a ESP32 programada com o *firmware* desenvolvido, conectada ao mesmo broker MQTT, se inscreve no mesmo tópico e, ao receber um *payload* desse, executa uma função de *callback* que deserializa o json recebido e produz um byte que contém os dados de dose e recipiente no formato apresentado na seção 1.1.1.2 e implementado conforme Apêndice D.

3.3 PLANEJAMENTO DA AULA PRÁTICA

Em laboratório, será realizada a síntese do projeto arquivado (extensão *qar*) da semana 2 para a placa FPGA DE0-CV segundo a pinagem da Tabela 4.

A montagem experimental será, então, realizada em etapas intermediárias para testar o correto funcionamento de todos os componentes do projeto e suas relações. Esse processo seguirá de acordo com o aqui descrito Plano de Execução Experimental.

3.3.1 Plano de Execução Experimental

Inicialmente, será testada a integração com o sensor ultrassônico. Para isso, os componentes externos ao circuito projetado (Analog Discovery, módulo do sensor HC-SR04 e CI conversor de nível de tensão 74HC4050) serão referenciados em um mesmo potencial da placa programada (GND). Em seguida, os pinos de alimentação da FPGA serão ligados a esses componentes de acordo com o nível de tensão esperado: 3,3 V de referencial nas placas conversoras 74HC4050; e 5 V para o funcionamento do sensor ultrassônico. Nesse momento, o osciloscópio do Analog Discovery poderá ser usado para verificar a adequação da tensão dos sinais e garantir que não haverá sobrecarga.

Após averiguação desse componente e correção de quaisquer falhas, os sinais de interface do circuito poderão, então, ser conectados. A saída *trigger* na ponte da FPGA pode ser ligada diretamente ao pino de mesmo nome no sensor ultrassônico, o *echo* do sensor deve ser ligado à entrada de um par entrada-saída do CI 7HC4050 e coletado em seu dual para a entrada do sinal *echo* efetivo do circuito pela porta B12.

Em seguida, será realizada a integração com o módulo do servomotor acoplado ao sensor ultrassônico. Para isso, ele será apropriadamente alimentado pelos terminais VCC e GND da placa FPGA, além de conectado ao sinal de saída *pwm* do circuito de controle do dispenser por meio dos acessos GPIO. É importante recordar, desde o início, de referenciar todos os componentes experimentais a um mesmo potencial terra (GND).

Em seguida, a placa será ligada e programada com o projeto em questão para a realização dos testes propostos, sempre de forma a se atentar aos sinais de depuração definidos. Além da depuração definida na placa, ainda há os pinos GPIO que apresentam a possibilidade de serem conectados ao Analog Discovery para verificação das ondas, que pode ser bem utilizada na depuração da recepção serial, medição do sensor e controle do motor.

O objetivo principal da prática em laboratório será verificar a correta comunicação serial via MQTT, garantindo a correta transição e nova independência do projeto em relação à porta serial física. Sendo assim, serão realizados os mesmos testes descritos antes pela Tabela 2, porém agora com o controle via aplicativo Mobile e transmissão via WiFi.

3.4 ATIVIDADES EXPERIMENTAIS

Em laboratório, o circuito do projeto foi sintetizado e programado na placa FPGA DE0-CV. A montagem foi realizada em etapas intermediárias, seguindo o que havia sido anteriormente definido no Plano de Execução Experimental. Primeiramente, foi realizada a montagem do módulo do servomotor, por meio da conexão de seus terminais GND e VCC (5V), além do sinal de controle *pwm* vindos da GPIO. Por fim, foi feito o acoplamento do sensor ultrassônico com os testes de conversão de tensão 5V para 3V por meio do osciloscópio do WaveForms.

A placa ESP32 foi acoplada a uma *protoboard* e alimentada por uma porta USB do computador para ser programada. Sua porta de saída de transmissão TX2 foi conectada a entrada serial da placa FPGA por meio do pino GPIO correspondente.

Com isso, foi realizada a integração de todos os módulos do dispositivo, que agora tornou-se independente do barramento serial, e foi finalizada a montagem do sistema atual, conforme apresentado na Figura ???. Nela, o display hexadecimal mais à esquerda reflete o estado do componente de interface do sensor (o que pode ser multiplexado pelas chaves SW9 e SW8) e os três à direita mostram a medida de distância em centímetros realizada pelo sensor. Já o LED0, mais à direita, foi utilizado como saída do sinal de alerta.

O aplicativo mobile foi aberto e, por meio dele, foram feitas as publicações MQTT que puderam ser depuradas pelo terminal do computador, tanto em relação ao envio pelo app quanto à recepção pela ESP e consequente envio para o circuito digital.

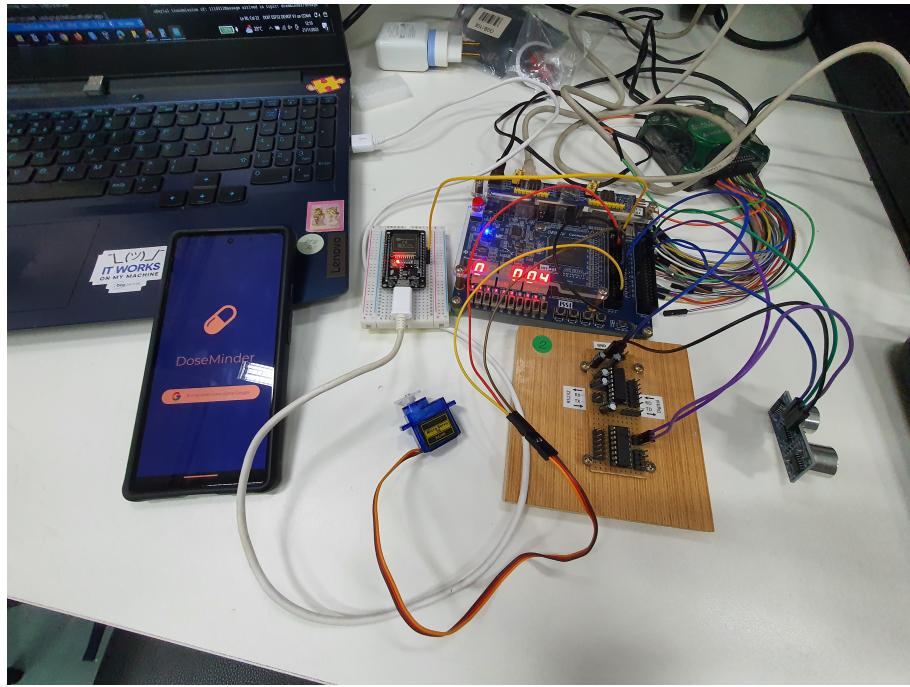


Figura 19: Montagem experimental do circuito completo.

Depois de um procedimento de testes e depurações, foi necessário aumentar o tempo de delay do servomotor para 0.5ms com o parâmetro de 25 milhões de ciclos de *clock*. Também foi preciso, ao longo do processo, utilizar o barramento serial via USB para verificar a estabilidade do funcionamento do circuito. Foi observada, por fim, a movimentação do servomotor conforme o esperado a partir das publicações enviadas pelo aplicativo, o que confirmou o bom funcionamento do dispositivo e da nova integração implementada pela comunicação.

4 CONSIDERAÇÕES FINAIS

A integração de diferentes áreas de desenvolvimento do projeto - hardware, software (interface e serviços) e firmware - se mostrou de suma importância para o devido funcionamento do sistema. A criação de módulos contidos em si com comunicação bem definida com os demais (chamadas de API, protocolo MQTT e transmissão serial) permitiu testes unitários do design, com depuração em console e nos *displays* da FPGA, e separação de responsabilidades no projeto.

APÊNDICES

A Pacote do projeto

```
1 package pill_package is
2     constant CONTAINERS : natural range 1 to 16 := 1;
3
4     subtype containers_range is natural range 0 to CONTAINERS-1;
5     type pill_count is array (containers_range) of
6         std_logic_vector(2 downto 0);
7
8     component downwards_counter is
9         generic (constant N: integer);
10        port (
11             clock   : in  std_logic;
12             clear   : in  std_logic;
13             count   : in  std_logic;
14             enable  : in  std_logic;
15             D       : in  std_logic_vector(N-1 downto 0);
16             Q       : out std_logic_vector(N-1 downto 0)
17         );
18     end component downwards_counter;
19
20     component circuito_pwm is
21         generic (
22             conf_período          : integer;
23             largura_00 , largura_01 : integer;
24             largura_10 , largura_11 : integer
25         );
26         port (
27             clock    : in  std_logic;
28             reset    : in  std_logic;
29             largura  : in  std_logic_vector(1 downto 0);
30             pwm      : out std_logic
31         );
32     end component circuito_pwm;
33
34     component contador_m is
35         generic (constant M, N : integer);
36         port (
37             clock : in  std_logic;
38             zera  : in  std_logic;
39             conta : in  std_logic;
40             Q     : out std_logic_vector(N-1 downto 0);
41             fim   : out std_logic;
42             meio  : out std_logic
43         );
44     end component contador_m;
45
46     component interface_hcsr04 is
```

```

47  port (
48    clock      : in  std_logic;
49    reset      : in  std_logic;
50    medir      : in  std_logic;
51    echo       : in  std_logic;
52    trigger    : out std_logic;
53    medida    : out std_logic_vector(11 downto 0);
54    pronto    : out std_logic;
55    db_reset   : out std_logic;
56    db_medir   : out std_logic;
57    db_estado  : out std_logic_vector(3 downto 0)
58  );
59 end component interface_hcsr04;
60
61 component rx_serial_7O1 is
62   generic (constant samples : natural := 16);
63   port (
64     clock      : in  std_logic;
65     reset      : in  std_logic;
66     dado_serial : in  std_logic;
67     dado_recebido : out std_logic_vector(6 downto 0);
68     paridade_recebida : out std_logic;
69     pronto    : out std_logic;
70     db_dado_serial : out std_logic;
71     db_estado  : out std_logic_vector(3 downto 0)
72  );
73 end component rx_serial_7O1;
74
75 component hexa7seg is
76   port (
77     hexa : in  std_logic_vector(3 downto 0);
78     sseg : out std_logic_vector(6 downto 0)
79  );
80 end component hexa7seg;
81
82 end package;

```

B Declaração de entidades

B.1 Entidade principal

```
1 entity pill_dispenser is
2     port (
3         clock      : in  std_logic;
4         reset      : in  std_logic;
5         db_switch  : in  std_logic_vector(1 downto 0);
6         echo       : in  std_logic;
7         serial     : in  std_logic;
8         pwm        : out std_logic_vector(CONTAINERS-1 downto 0);
9         trigger    : out std_logic;
10        alert      : out std_logic;
11        — debug
12        db_reset   : out std_logic;
13        db_pwm     : out std_logic;
14        db_echo    : out std_logic;
15        db_trigger : out std_logic;
16        db_serial  : out std_logic;
17        db_state   : out std_logic_vector(6 downto 0);
18        db_measurement : out std_logic_vector(20 downto 0)
19    );
20 end entity;
```

B.2 Unidade de Controle

```
1 entity pill_dispenser_uc is
2     port (
3         clock      : in  std_logic;
4         reset      : in  std_logic;
5         check_timeout : in  std_logic;
6         safety_timeout : in  std_logic;
7         detected   : in  std_logic;
8         — control
9         count      : out std_logic;
10        move       : out std_logic;
11        discount   : out std_logic;
12        — debug
13        db_state   : out std_logic_vector(3 downto 0)
14    );
15 end entity;
```

B.3 Fluxo de Dados

```
1 entity pill_dispenser_fd is
2     port (
3         clock      : in  std_logic;
4         reset      : in  std_logic;
5         serial     : in  std_logic;
```

```

6      echo      : in  std_logic;
7      count     : in  std_logic;
8      move      : in  std_logic;
9      discount   : in  std_logic;
10     — outputs
11     trigger    : out std_logic;
12     check_end  : out std_logic;
13     safety_end : out std_logic;
14     pwm        : out std_logic_vector(CONTAINERS-1 downto 0);
15     alert       : out std_logic; — system is active and alert
16     detected    : out std_logic; — system can dispense pill
17     — debug
18     db_measurement : out std_logic_vector(11 downto 0);
19     — states
20     rx_state    : out std_logic_vector(3 downto 0);
21     sensor_state: out std_logic_vector(3 downto 0)
22 );
23 end entity;

```

C Implementação de arquiteturas

C.1 Contador decrescente

```
1 entity downwards_counter is
2     generic (constant N: integer := 3);
3     port (
4         clock : in std_logic;
5         clear : in std_logic;
6         count : in std_logic;
7         enable : in std_logic;
8         D : in std_logic_vector (N-1 downto 0);
9         Q : out std_logic_vector (N-1 downto 0)
10    );
11 end entity downwards_counter;
12
13 architecture behavioral of downwards_counter is
14     signal IQ : natural;
15 begin
16
17     process(clock, clear, enable, IQ)
18     begin
19         if (clear = '1') then IQ <= 0;
20         elsif (rising_edge(clock)) then
21             if (count='1') then
22                 if (IQ /= 0) then IQ <= IQ - 1;
23                 end if;
24             elsif (enable='1') then IQ <= to_integer(unsigned(D));
25             else IQ <= IQ;
26             end if;
27         end if;
28         Q <= std_logic_vector(to_unsigned(IQ, Q'length));
29     end process;
30
31 end architecture behavioral;
```

C.2 Unidade de Controle

```
1 architecture fsm_arch of pill_dispenser_uc is
2     type state is (idle, check, dispense, countdown, await);
3     signal current_state, next_state : state;
4 begin
5
6     — state
7     process (reset, clock)
8     begin
9         if reset='1' then
10             current_state <= idle;
11         elsif clock'event and clock='1' then
12             current_state <= next_state;
13         end if;
```

```

14  end process;
15
16  — next state
17  process (current_state , detected , check_timeout , safety_timeout )
18  begin
19    case current_state is
20      when idle      => if detected='1' then next_state <= check;
21                      else                         next_state <= idle;
22                      end if;
23      when check     => if detected='0'           then next_state <=
24                           idle;
25                           elsif check_timeout='1' then next_state <=
26                               dispense;
27                           else                         next_state <=
28                               check;
29                           end if;
30      when dispense  => if safety_timeout='1' then next_state <=
31                           countdown;
32                           else                         next_state <=
33                               dispense;
34                           end if;
35      when countdown => next_state <= await;
36      when await     => if safety_timeout='0' then next_state <=
37                           await;
38                           elsif detected='1' then next_state <=
39                               dispense;
40                           else                         next_state <=
41                               idle; — detected 0 and safety timeout 1
42                           end if;
43      when others    => next_state <= idle;
44    end case;
45  end process;
46
47  — control signals
48  with current_state select
49    count    <= '1' when check | dispense | await , '0' when others;
50  with current_state select
51    move     <= '1' when dispense , '0' when others;
52  with current_state select
53    discount <= '1' when countdown , '0' when others;
54
55 with current_state select
56   db_state <= "0000" when idle ,
57                  "0001" when check ,
58                  "0010" when dispense ,
59                  "0011" when countdown ,
60                  "0100" when await ,
61                  "1110" when others; — Error
62
63 end architecture;

```

C.3 Fluxo de Dados

```

1 ...
2 constant check_timeout : natural := 100_000_000;
3 ...
4 constant safety_timeout : natural := 25_000_000;
5 ...
6     CHECK_TIMER: contador_m
7         generic map (M => check_timeout, N => check_timeout_bits)
8             port map (
9                 clock => clock,
10                zera => s_count_reset,
11                conta => count,
12                Q => open,
13                fim => check_end,
14                meio => open
15            );
16
17     SAFETY_TIMER: contador_m
18         generic map (M => safety_timeout, N => safety_timeout_bits)
19             port map (
20                 clock => clock,
21                 zera => s_count_reset,
22                 conta => count,
23                 Q => open,
24                 fim => safety_end,
25                 meio => open
26            );
27
28     HCSR04: interface_hcsr04
29         port map (
30             clock => clock,
31             reset => reset,
32             medir => s_alert,
33             echo => echo,
34             trigger => trigger,
35             medida => s_measurement,
36             pronto => s_meas_ready,
37             db_reset => open,
38             db_medir => open,
39             db_estado => sensor_state
40         );
41
42     — 4 bits to identify the container (6 downto 3) and
43     — the 3 least significant bits to indicate the proper dosage of
44     — the container (2 downto 0)
45     CONTAINERS: for i in pill_containers'range generate
46         containers_enable(i) <= '1' when
47             i = to_integer(unsigned(s_dosage(6 downto 3))) and
48             s_rx_received='1'
49             else '0';
50         pwm(i) <= '0' when pill_containers(i) = "000" else s_pwm;

```

```

50
51     PILL_COUNT: downwards_counter
52         generic map (N => 3)
53         port map (
54             clock => clock,
55             clear => reset,
56             count => discount,
57             enable => containers_enable(i),
58             D      => s dosage(2 downto 0),
59             Q      => pill_containers(i)
60         );
61     end generate;
62
63     alert <= '0' when
64         pill_containers = (pill_containers'range => (others => '0'))
65     else
66         '1';
67
68     meas_cm <= to_integer(unsigned(s_measurement));
69     s_close <= '1' when (0 < meas_cm and 6 > meas_cm) else '0';
70     detected      <= s_alert and s_close;

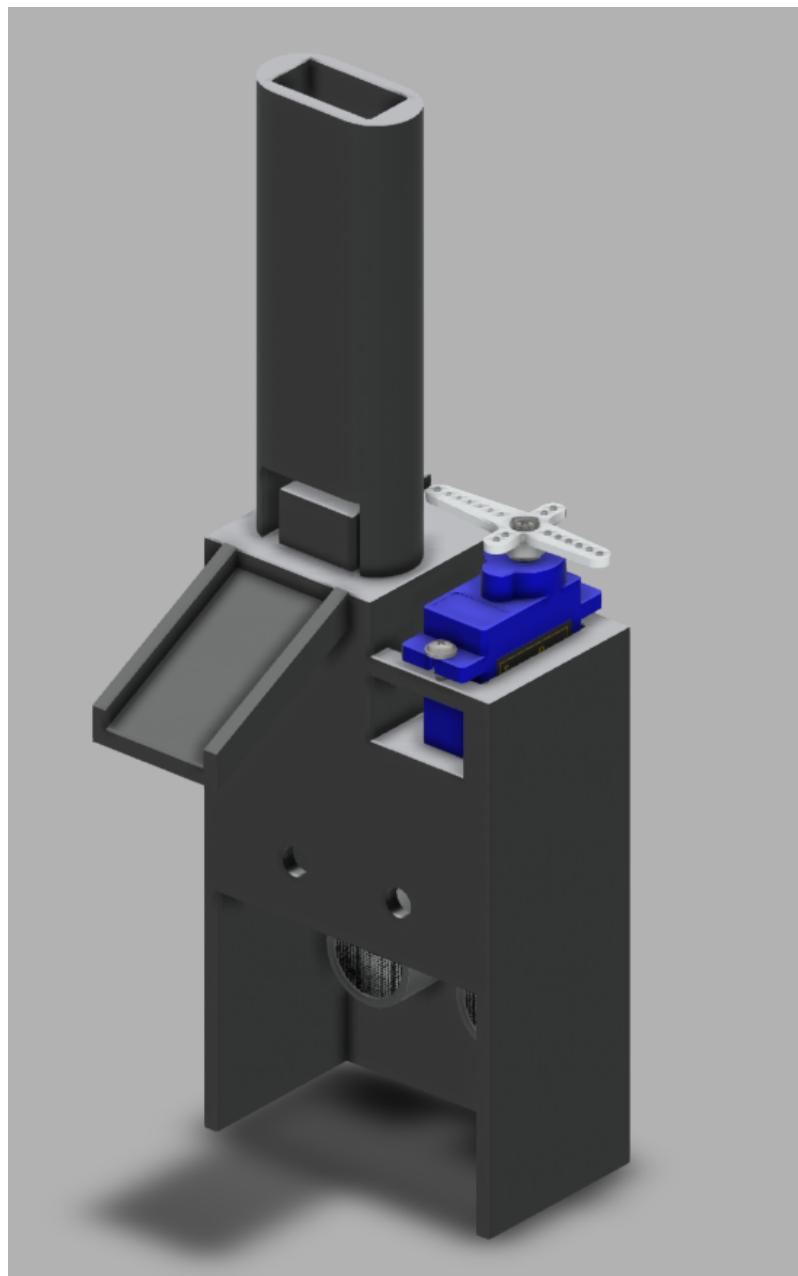
```

D Deserialização em firmware MQTT

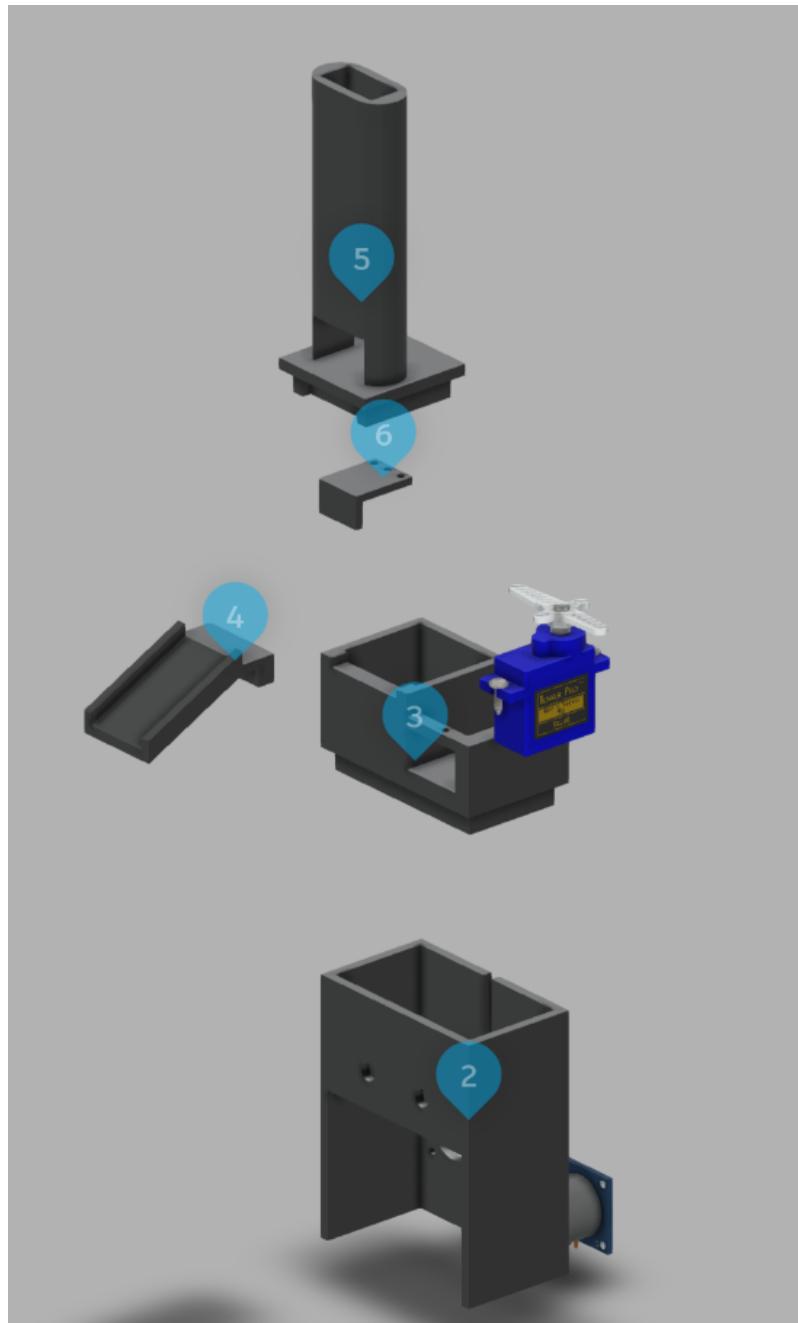
```
1 void callback(char *topic, byte *payload, unsigned int length) {
2     StaticJsonDocument<60> doc;
3     DeserializationError error;
4
5     byte data, dose, recipient;
6
7     Serial.print("Message - arrived - in - topic : -");
8     Serial.println(topic);
9     Serial.print("Message : -");
10    for (int i = 0; i < length; i++) {
11        Serial.print((char) payload[i]);
12    }
13    Serial.println();
14    Serial.println("-----");
15
16    // Unmarshalling errors
17    error = deserializeJson(doc, (char*) payload);
18    if (error) {
19        Serial.print("deserializeJson() - failed : -");
20        Serial.println(error.c_str());
21        return;
22    }
23
24    dose = (byte) doc["dose"] << 3;
25    recipient = doc["recipient"];
26    data = dose + recipient;
27    Serial2.write(data);
28    Serial.printf("Serial - transmission - of -%x: -", data);
29 }
```

E Modelagem 3D

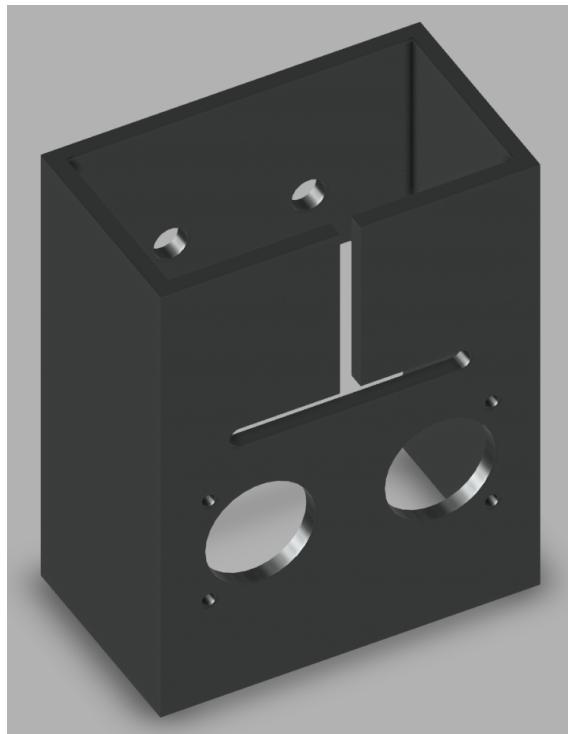
E.1 Montagem



E.2 Vista explodida



E.3 Base do sensor



E.4 Suporte do motor

