

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO  
Graduação em Engenharia de Computação

Bancada A2 - 01/12/2020

Gabriel Sanefuji

NUSP: 10770141

Henrique Geribello Giabardo

NUSP: 10774142



Relatório final - Segurança para filhotes

Professor Edson Midorikawa  
PCS 3645 - Laboratório Digital 2

São Paulo - SP  
2020

<b>Objetivo</b>	<b>3</b>
<b>Projeto</b>	<b>3</b>
<b>Descrição</b>	<b>3</b>
<b>Requisitos</b>	<b>4</b>
<b>Solução Técnica</b>	<b>4</b>
<b>Sprints</b>	<b>5</b>
<b>Semana 1</b>	<b>6</b>
<b>Idealização</b>	<b>6</b>
<b>Especificações</b>	<b>7</b>
<b>Semana 2</b>	<b>10</b>
<b>Processo Experimental</b>	<b>10</b>
<b>VHDL</b>	<b>13</b>
<b>Resultados Experimentais</b>	<b>19</b>
<b>Semana 3</b>	<b>22</b>
<b>Processo Experimental</b>	<b>22</b>
<b>Script</b>	<b>22</b>
<b>Teste</b>	<b>24</b>
<b>Resultados Experimentais</b>	<b>25</b>
<b>Semana 4</b>	<b>30</b>
<b>Processo Experimental</b>	<b>30</b>
<b>Script</b>	<b>32</b>
<b>Open Lab</b>	<b>32</b>
<b>Resultados experimentais</b>	<b>38</b>
<b>Semana 5</b>	<b>40</b>
<b>OpenLab</b>	<b>40</b>
<b>Conclusão</b>	<b>41</b>
<b>Apêndice</b>	<b>42</b>
<b>VHDL</b>	<b>42</b>
<b>RTL</b>	<b>49</b>
<b>Scripts</b>	<b>51</b>

# **Objetivo**

Organizar e documentar o procedimento que será realizado durante o desenvolvimento do projeto da matéria Laboratório Digital 2.

O projeto em questão será um circuito para auxílio na preservação da segurança de filhotes de animais, no caso em que faremos, e um cachorro, de modo que acionando-se o medidor de distância, quando o pet aproximar-se a porta fecha-se. Além disso, será implementado um sistema de modo de operação, no qual cada um deles possuirá características próprias, as quais serão especificadas mais adiante.

Na experiência desta semana, finalizaremos a refatoração dos componentes já feitos, além da integração dos mesmos, e teste para verificação do funcionamento dele numa placa.

# **Projeto**

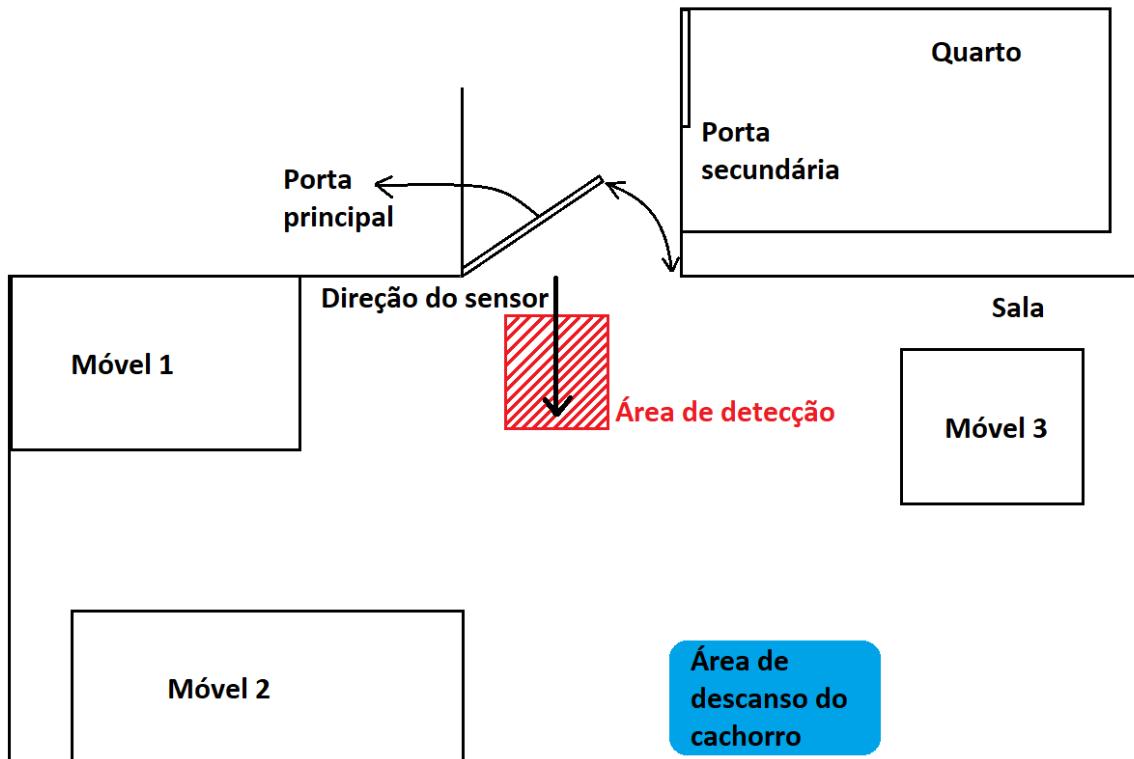
## **Descrição**

O projeto, chamado de segurança para filhotes, consistirá na ideia de que cachorros, e pets em geral, muitas vezes não podem entrar em certos cômodos, para a segurança deles e para, quando necessário, manter uma certa ordem na casa, principalmente em tempos de quarentena em que reuniões de trabalho e faculdade são realizadas neste ambiente.

Desse modo, toda vez que o cachorro em questão aproximar-se de uma porta, ela irá fechar. Além disso, será implementado um sistema de controle com LEDs para que o usuário possa identificar quais portas da casa estão devidamente abertas e fechadas. Após a implementação correta e completa do circuito, também têm-se em mente que outras portas poderão estar conectadas a esse controlador, o qual poderá abrir ou fechá-las, de acordo com o desejo do usuário.

Outro ponto que poderá ser implementado é um sistema de modo de funcionamento, ou seja, poderemos alternar entre um modo no qual admite-se que o cachorro está sozinho em casa e que, portanto, ele não pode sair do cômodo, e outro modo no qual o usuário está em casa, e o pet pode sair com o consentimento do usuário.

A seguir segue um esquemático do que foi pensado dentro de uma casa:



## Requisitos

### Funcionais:

- 1) Implementar dois modos de operação, um livre e outro restrito
- 2) Mudar o estado de um LED, ou similar que simule uma porta, a partir da posição do cachorro.

### Não funcionais:

- 1) Conexão entre componentes de servomotor e o sensor de distância;
- 2) Conexão entre dois celulares diferentes com Blynk.

## Solução Técnica

Utilizaremos os componentes já feitos de sensor de distância e servomotor, alterando-os, de maneira que o controle do servomotor tenha apenas duas saídas de estado, as quais iriam representar a porta aberta e fechada. Além disso, faremos alterações no sensor de distância para que o mesmo assinale que o cachorro está a uma distância que julgamos ser mínima para que a porta feche sem que ele se machuque, e sem que a porta feche de maneira desnecessária.

Para fazer o projeto da maneira que visualizamos, modificaremos a montagem do servomotor com o sensor de distância, de maneira que estejam acoplados separadamente, mas ainda conversem entre si, para que a porta estando aberta ou fechada, mantenha o sensor de distância funcionando e verificando a mesma área.

Para que os dois conversem entre si, iremos conceber uma nova unidade de controle, a qual receberá um sinal para verificar a distância dos objetos, fará a medição e indicará se deve-se ou não fechar a porta. Além disso, ainda irá ter como saída, LEDs que indicarão se a porta está ou não fechada e em qual cômodo o cachorro encontra-se.

Para a modificação do estado de funcionamento do circuito, há duas opções que podem ser implementadas, ou através de um botão que muda o estado do projeto, ou através de uma mensagem recebida pela UART, de maneira que poderíamos reutiliza-la.

## Sprints

### 1) Jornada de uso 1:

Verificação com o professor acerca da validade do projeto em mente e revisão dos componentes em mãos.

O que deve ser entregue ao final desse sprint:

- a) Tema;
- b) Requisitos;
- c) Solução técnica;
- d) Revisão dos componentes.

### 2) Jornada de uso 2:

Elaboração de novos componentes, em tese a unidade de controle nova e a entidade que une os componentes, e testes para verificar se os componentes novos e velhos funcionam se estiverem integrados.

O que deve ser entregue ao final desse sprint:

- a) Projeto funcional;
- b) Testes dos novos componentes;
- c) Apresentação dos componentes, assim como sua integração.

### 3) Monitoramento dos serviços:

Debate com o Professor e validação do projeto. Elaboração de modificações para deixar o projeto mais completo, com Monitoração dos Serviços e Interação com dois ou mais celulares com a FPGA em tempo real.

O que deve ser entregue ao final desse sprint:

- a) Estudo e aplicação de um segundo celular ao Blynk;
- b) Levantamento de novos componentes;
- c) Apresentar ao professor o resultado de dois celulares funcionando ao mesmo tempo.

### 4) Interação dois celulares:

No sprint final será feita a integração entre os celulares de ambos os integrantes com os componentes desenvolvidos anteriormente, de maneira a simular a conexão entre lab e home lab, assim como para em uma futura implementação, avisar ao usuário

que o seu respectivo pet encontra-se em um ou outro cômodo assim como as portas abertas ou fechadas.

O que deve ser entregue ao final desse sprint:

- a) Projeto completo funcional.

## Semana 1

Ao contrário do que foi planejado, o grupo não fez os testes dos componentes revisados. Após a apresentação e os questionamentos trazidos pelos professores, foi julgado que o projeto deveria ser melhor especificado, para que não houvessem dúvidas ou reavaliações do mesmo no meio do desenvolvimento dele. Do mesmo modo, foi definido o modo como seria apresentado o projeto ao final do curso.

## Idealização

Primeiro, revisamos como seria feito o projeto e como ele seria apresentado. Isso porque encontramos o problema de representação do cão e das portas, uma vez que, a princípio, serão utilizados apenas os recursos encontrados no laboratório digital, e portanto, para mostrar o funcionamento do circuito para a bancada seriam necessários meios alternativos.

Tendo isso em mente e com o auxílio do Victor, monitor da disciplina, pudemos alterar levemente o projeto de maneira a ficar mais visível qual seria o resultado, utilizando apenas dos recursos oferecidos pelo lab, e ainda termos maior certeza com relação a sua demonstração.

Assim como havia sido especificado, o projeto consistirá em um circuito controlador de portas, para que cãezinhos não entrem em cômodos que possuem objetos perigosos ou que simplesmente não podem ser acessados. Para isso, serão utilizados os componentes já concebidos de interface do sensor de distância, e o servomotor. Será feito uma nova unidade de controle e um circuito que une todos esses componentes.

Para representar o cachorro, utilizaremos um celular com Blynk configurado para ter a widget “slide”. Por meio da variação do valor dessa widget, iremos alterar o input do valor de posicionamento do cão. Os valores exatos serão discutidos na semana 3, quando os componentes estiverem completos.

Para as portas, utilizaremos o servomotor para representar uma delas, a principal, e a outra será através de um LED em um segundo celular com Blynk configurado. Desse modo, estaremos utilizando do recurso de dois celulares para uma placa FPGA, e essa implementação será gradual, acontecendo apenas após a primeira porta estiver funcional.

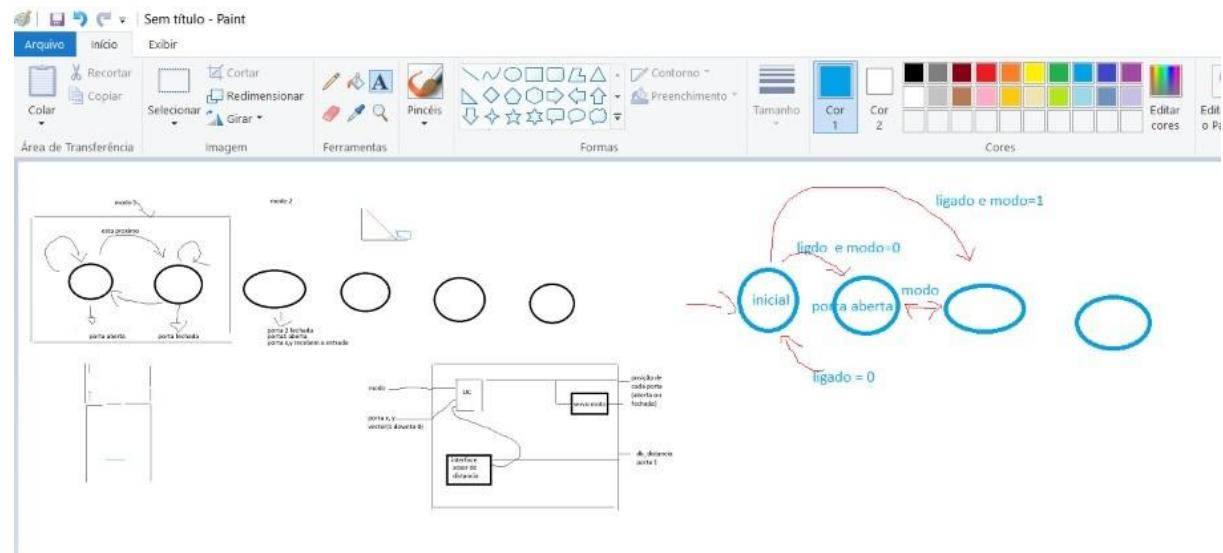
Além disso, foi decidido como operariam cada um dos modos de funcionamento do circuito. Para o “modo 0”, admite-se que ninguém está em casa, e portanto, o cãozinho deve manter-se na sala, e a qualquer aproximação dele na porta, a mesma deve ser fechada.

Para o “modo 1”, acredita-se que os donos estão em casa, e com isso, o controle de portas (neste caso apenas 2) é manual, tendo o sensor de proximidade sempre ativo para saber se ele se aproxima ou não da porta.

## Especificações

Após discutidas as ideias principais, teve início a discussão de algumas especificações, dentre elas, a distância mínima que seria utilizada para o fechamento da porta, a máquina de estados e um esquemático simplificado de como seria feita a reutilização dos componentes e suas características principais.

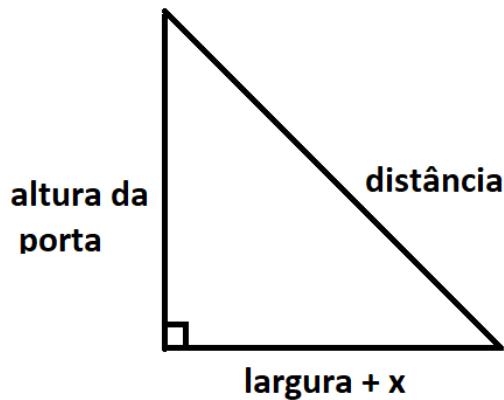
Foi feito o seguinte esquemático, que será melhor detalhado mais a frente, com a discussão entre os integrantes:



## Distância

Para definir a distância para a qual o sensor acionaria o servomotor, pensamos que o sensor estaria no topo da porta. Portanto, utilizando Pitágoras, com os catetos de valor da altura e da largura da porta mais um certo valor x, provavelmente 5 cm para manter uma distância segura para que o pet não se machuque com o movimento de fechar da porta, encontramos a distância a ser utilizada. Quando o cachorro passar por esse ponto, a distância medida pelo sensor será menor do que aquela calculada, e nesse instante a porta se fechará.

A seguir apresentamos uma figura com um esquema para o cálculo realizado:

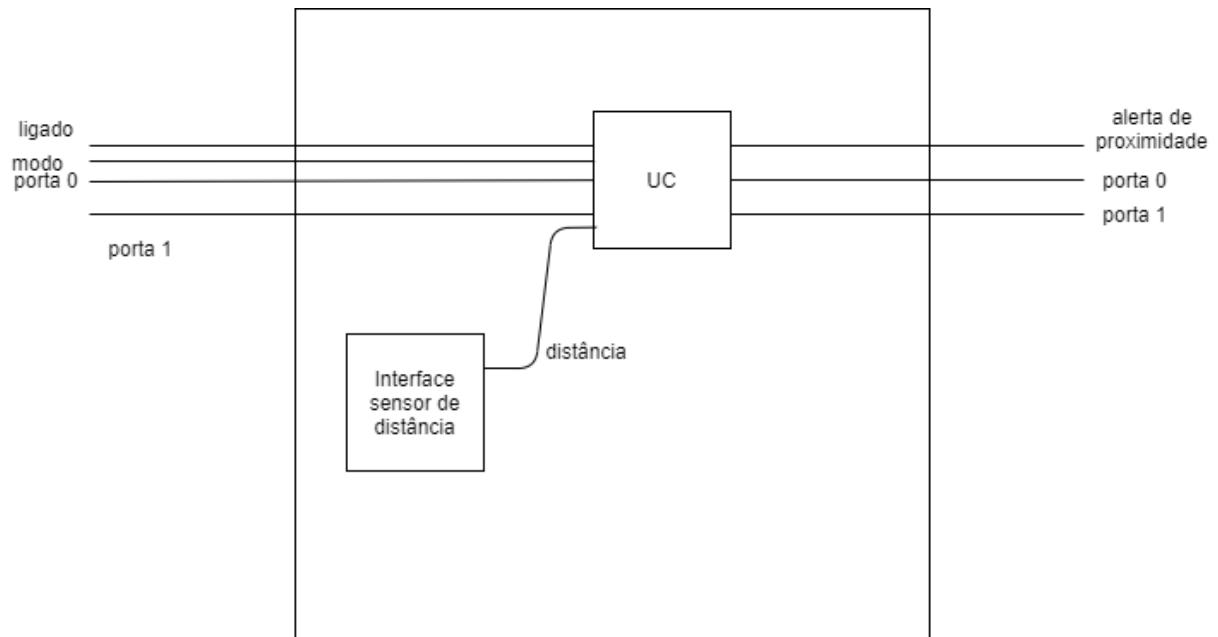


### Esquema dos componentes

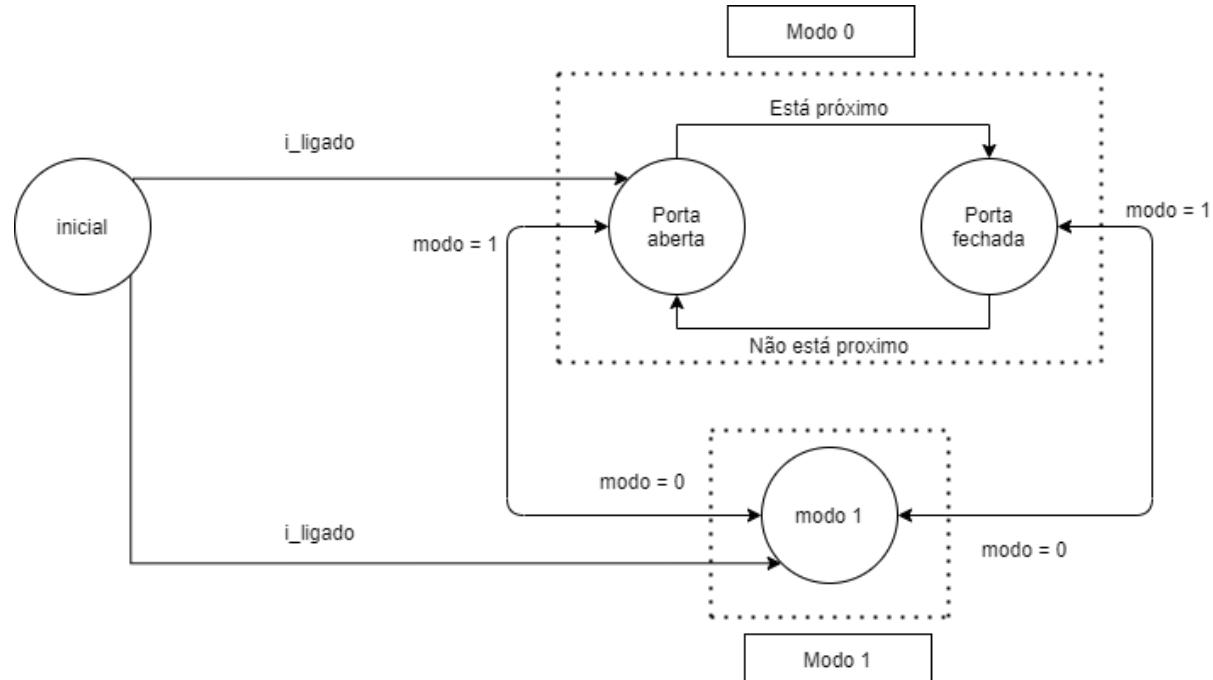
Como já havia sido pensado anteriormente, será feito uso dos componentes de servomotor, interface do sensor de distância e uma nova unidade de controle. Esse esquema serviu mais para nos guiar com relação ao número de entradas e saídas, assim como quais seriam eles.

Por meio dessa discussão, concordou-se em 4 entradas, uma para ligar efetivamente o circuito, outra para selecionar o modo de operação e 2 outras para definir o estado das portas sendo controladas, quando no modo 1.

Sabendo que debuggers serão colocados na fase de testes, seriam 3 saídas principais, das quais 2 controlariam cada uma das portas individualmente, ou seja, uma acenderia o LED no Blynk do segundo celular, e o outro controlaria o servomotor para representar o fechamento e abertura da porta, e a última saída traria a informação de proximidade do pet na porta.



## Máquina de estados



A máquina de estados apresentada é simples, contendo apenas 4 estados:

- **Inicial:** Permanece nesse estado enquanto a entrada *i\_ligado* não for acionada, e não realiza qualquer tipo de operação enquanto permanecer nele. Quando for ligado, passa para o estado *modo0\_porta\_aberta* caso a entrada modo esteja em 0, e *modo1* caso a entrada modo esteja em 1;
- **modo0\_porta\_aberta:** Neste estado, a porta principal permanece aberta, e se mantém assim até que o sensor de distância verifique que o cachorro esteja próximo, acionando o sinal *esta\_proximo*;
- **modo0\_porta\_fechada:** A porta principal permanece fechada, e se mantém assim até que o sensor de distância verifique que o cachorro não esteja próximo a uma outra distância limite;
- **modo1:** Se a qualquer momento da operação, for acionada a entrada modo, o circuito passará para este estado, no qual o controle das portas torna-se manual e dependente das entradas “porta”. O sensor de distância e o alerta de proximidade continuam funcionando, e o circuito permanece neste estado até que a entrada modo seja desativada.

## VHDL

Não foi possível realizar os testes dos componentes que já haviam sido feitos nos módulos de experiência passados. Desse modo, ficou decidido que para a semana seguinte seriam entregues os módulos revisados assim como a UC nova, visto que todo o planejamento com relação ao seu funcionamento e modo de operação já foram detalhados.

Assim, será possível demonstrar em aula, aquilo que foi feito.

# Semana 2

## Processo Experimental

Será feita a verificação dos componentes do sensor de distância e servomotor, que deveriam ter sido feitos na semana anterior, além da criação de uma UC nova. Para verificar se os componentes revisados estão de acordo com o que esperamos deles, o servomotor deve possuir apenas dois estados e o sensor de distância deve acionar uma saída com uma distância menor do que uma a ser definida, utilizaremos os seguintes planos de teste:

### a) Servomotor

Com a seguinte pinagem:

```
i_clock pin_m9  
i_reset pin_t20  
  
i_posicao[0] pin_p18  
i_posicao[1] pin_r17  
  
o_pwm pin_f15
```

E o plano de teste:

Passo	Ação
1	Configurar o osciloscópio
2	Iniciar o circuito
3	Apertar o botão de reset, DIO2
4	Acionar a posição 0, desativando os botões DIO1 e DIO 0
5	Verificar a forma de onda, apertando o botão single
6	Repetir passos 3 a 5, mas selecionando as demais posições

### b) Sensor de distância

Com a seguinte pinagem:

```

i_clock pin_m9          o_display2[0] pin_y19
i_reset pin_n16         o_display2[1] pin_ab17
i_medir pin_b16        o_display2[2] pin_aa10
i_echo pin_t17         o_display2[3] pin_y14
o_trigger pin_j17      o_display2[4] pin_v14
                           o_display2[5] pin_ab22
                           o_display2[6] pin_ab21

o_display0[0] pin_u21    o_display3[0] pin_y16
o_display0[1] pin_v21    o_display3[1] pin_w16
o_display0[2] pin_w22    o_display3[2] pin_y17
o_display0[3] pin_w21    o_display3[3] pin_v16
o_display0[4] pin_y22    o_display3[4] pin_u17
o_display0[5] pin_y21    o_display3[5] pin_v18
o_display0[6] pin_aa22   o_display3[6] pin_v19

o_display1[0] pin_aa20   o_pronto pin_aa2
o_display1[1] pin_ab20
o_display1[2] pin_aa19
o_display1[3] pin_aa18
o_display1[4] pin_ab18
o_display1[5] pin_aa17
o_display1[6] pin_u22    db_medir pin_l1
                           db_estado[0] pin_y3
                           db_estado[1] pin_n2
                           db_estado[2] pin_n1
                           db_estado[3] pin_u2
                           db_estado[4] pin_u1

```

E o plano de teste:

Passo	Ação
1	Iniciar o circuito
2	Acionar o reset
3	Apertar o botão de medir
4	Verificar se o LED para medir distância próxima está funcionando
5	Verificar a medida no display
6	Repetir o processo 2 a 5, cerca de 3 vezes, e variando a distância do objeto em questão, perto, médio e longe

### c) Circuito do projeto

Com a seguinte pinagem:

```

i_clock pin_m9
i_reset pin_t20

i_ligado pin_n16
i_modo pin_b16
i_portaX pin_m16
i_portaY pin_c16

i_echo pin_t17

o_porta0 pin_h16
o_porta1 pin_a12
o_portaX pin_h15
o_portaY pin_b12

o_pwm pin_k16
o_trigger pin_j17|

```

E o plano de testes:

Passo	Ação
1	Iniciar o circuito
2	Acionar o reset
3	modo 0
4	Pedir para Fátima colocar um objeto a uma distância menor que 30 cm
5	Observar os Leds no Blynk
6	modo 1
7	mudar os valores das entradas das portas X e Y
8	verificar os valores dos Leds

Caso seja necessário, faremos uma entrada a mais, na qual será escolhida a distância a ser medida, para verificar se o circuito está funcional e em que seja necessária a intervenção de outras pessoas para o andamento dos testes.

Ao final da experiência, o que se espera é mostrar ao professor que todos os componentes estão funcionando, provando que ao clicar no botão de escolha de modo, o circuito mude de comportamento e com o acionamento do sensor de distância, o servo motor se desloca. Além disso, a verificação do comportamento do circuito pode ser mostrada a partir do acionamento ou não dos LED que correspondem a cada porta.

# VHDL

## Alterações:

Por enquanto foram feitas poucas alterações nos componentes em que já trabalhamos nas experiências passadas, apenas modificando a saída do servomotor e entrada de posição dele, para que desse modo, haja apenas 2 posições disponíveis para o mesmo se mover:

```
8 |     i_posição : in std_logic;
--| 63 |     when '1' => s_posicao <= 50000;
64 |     when others => s_posicao <= 0;
```

Contudo, como será demonstrado nas simulações a seguir, será necessário mudar a lógica do medidor de distância ou sua Unidade de Controle. Isso se deve, pois com a medição contínua, o valor medido é perdido a cada nova medição, ainda que ele continue o mesmo, assim, por um momento a porta permanece com valor 0, ainda que a medida seja tal para que continue em 1.

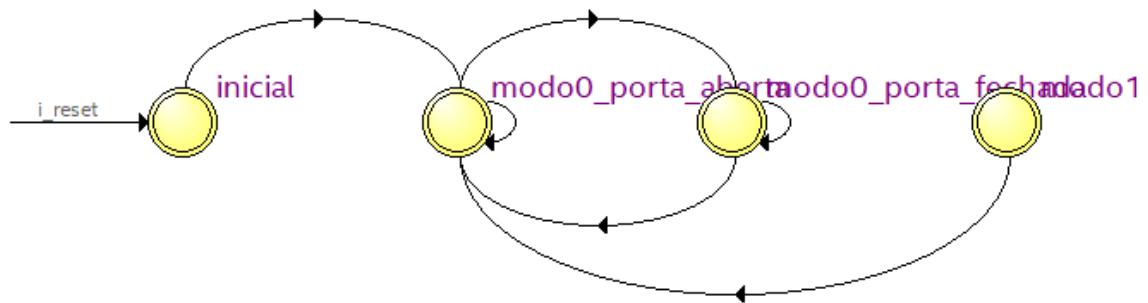
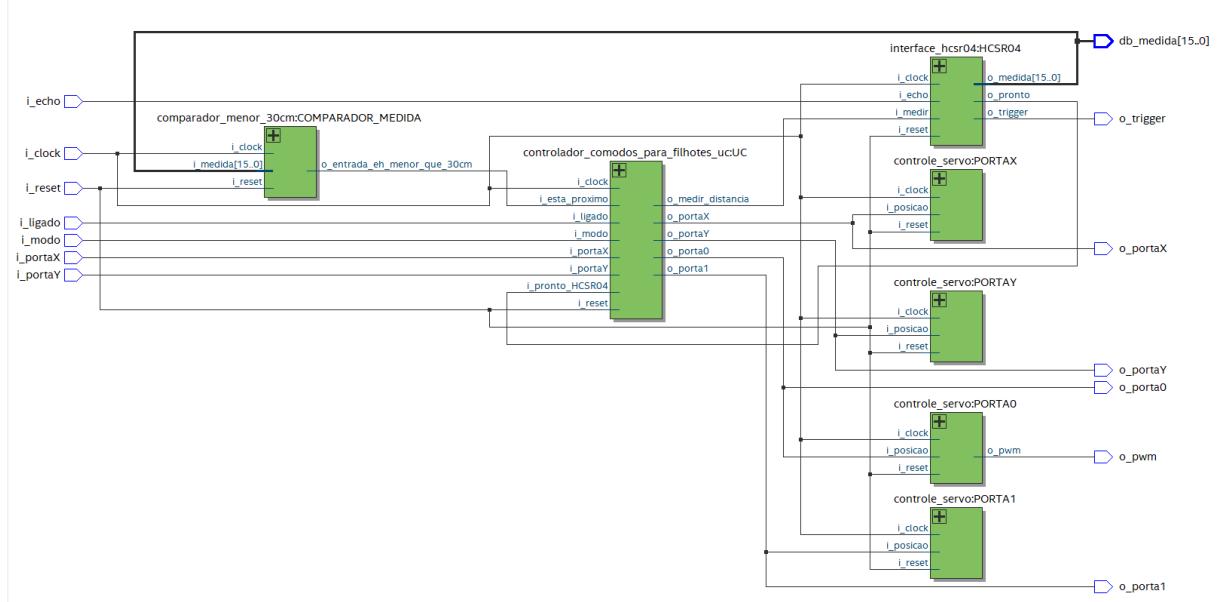
## Componentes novos:

O componente novo é uma interface que une os componentes de servomotor e medidor de distância e os manipula por meio de uma Unidade de Controle nova.

Na interface, são colocados os valores de entrada para ligar o projeto em *i\_ligado*, escolha do modo de operação em *i\_modo*, e o controle das portas manual, além da entradas para *clock* e *reset*. Para as saídas, até o momento, existe uma *pwm* para controle do servomotor, e as saídas para os LED que indicam o fechamento, quando em alto, ou abertura da porta, quando em baixo.

```
5 entity controlador_comodos_para_filhotes is
6   port
7   (
8     i_clock, i_reset: in std_logic;
9     i_ligado, i_modo, i_portax, i_portay: in std_logic;
10    o_porta0, o_porta1, o_portax, o_portay: out std_logic; -- porta em 0 é aberta em 1 é fechada
11
12    i_echo: in std_logic;
13    o_pwm, o_trigger: out std_logic;
14    db_medida: out std_logic_vector (15 downto 0)
15
16  );
17 end entity;
```

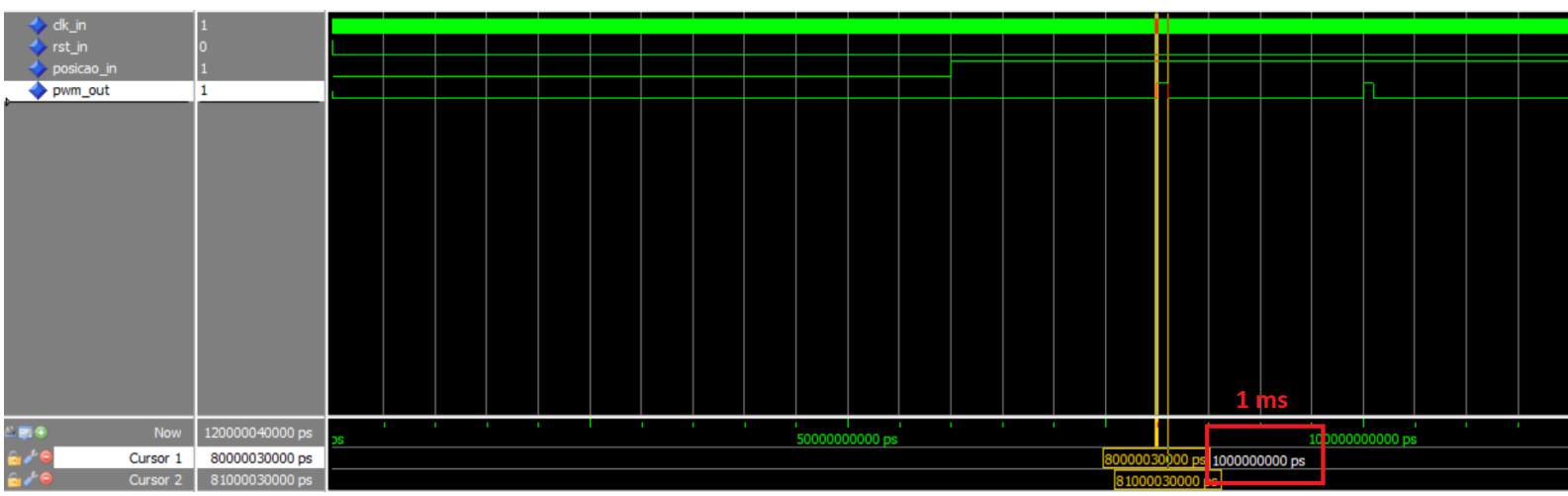
A seguir segue o diagrama RTL e de transição de estados, o qual implementa a ideia contida no diagrama de estados especificado na semana 1:



### Simulações:

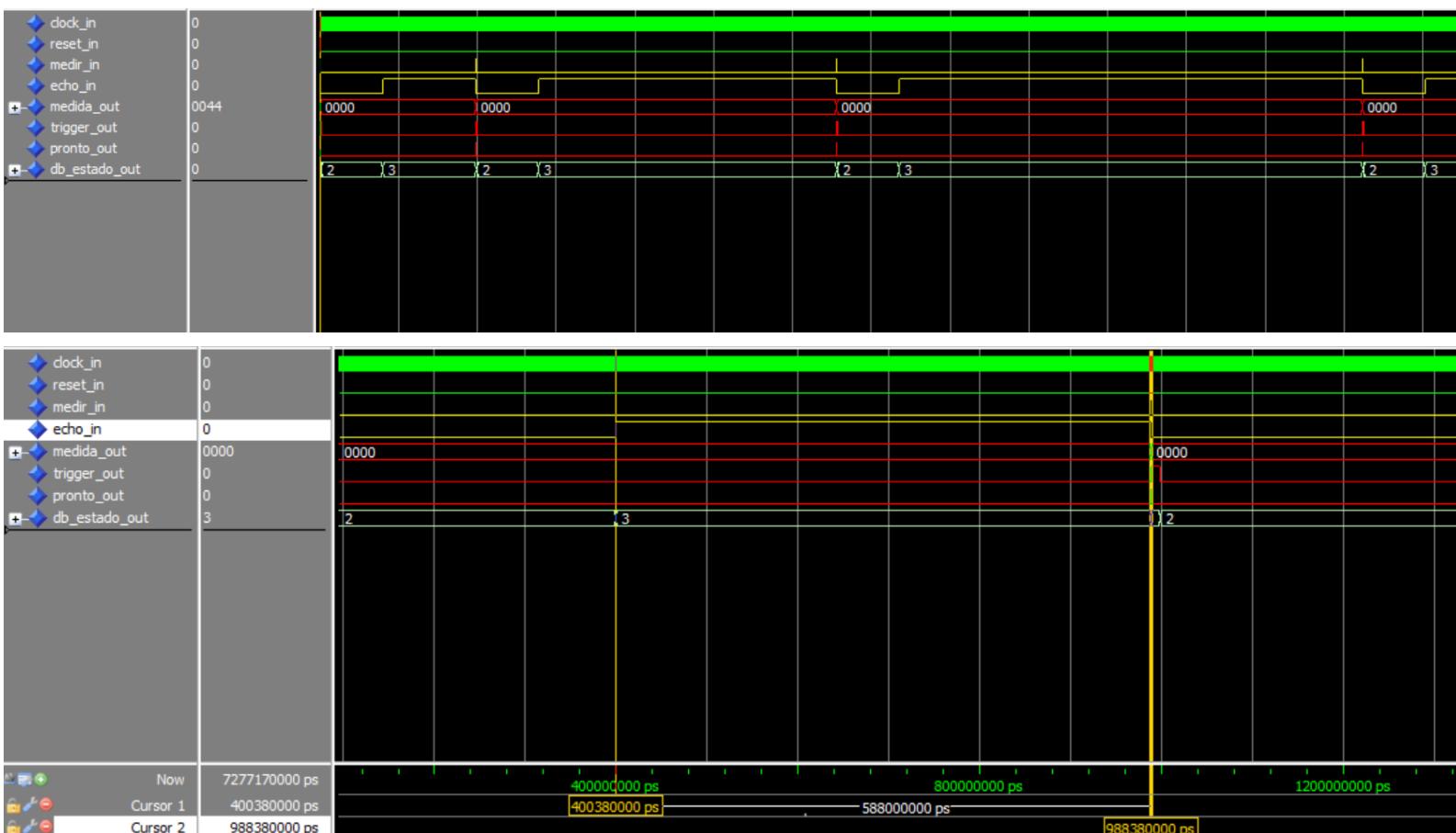
Para as simulações do servomotor e do medidor de distância, serão mostrados apenas uma de suas medidas, visto que eles sofreram poucas alterações, e portanto, o restante das medições segue a mesma lógica que a das primeiras.

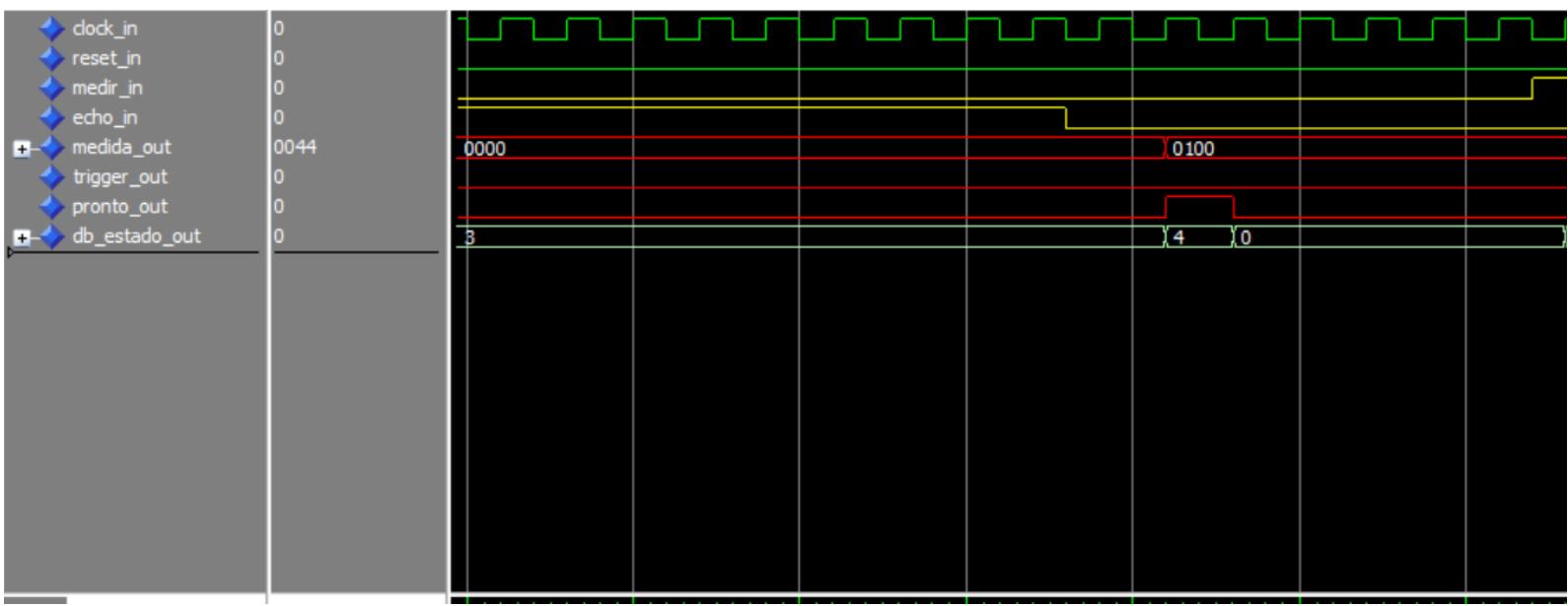
#### a) Servomotor



Apenas há duas posições, como era esperado, uma na qual o servomotor está sob uma pwm de 1 ms e outra na qual não há.

### b) Medidor de distância

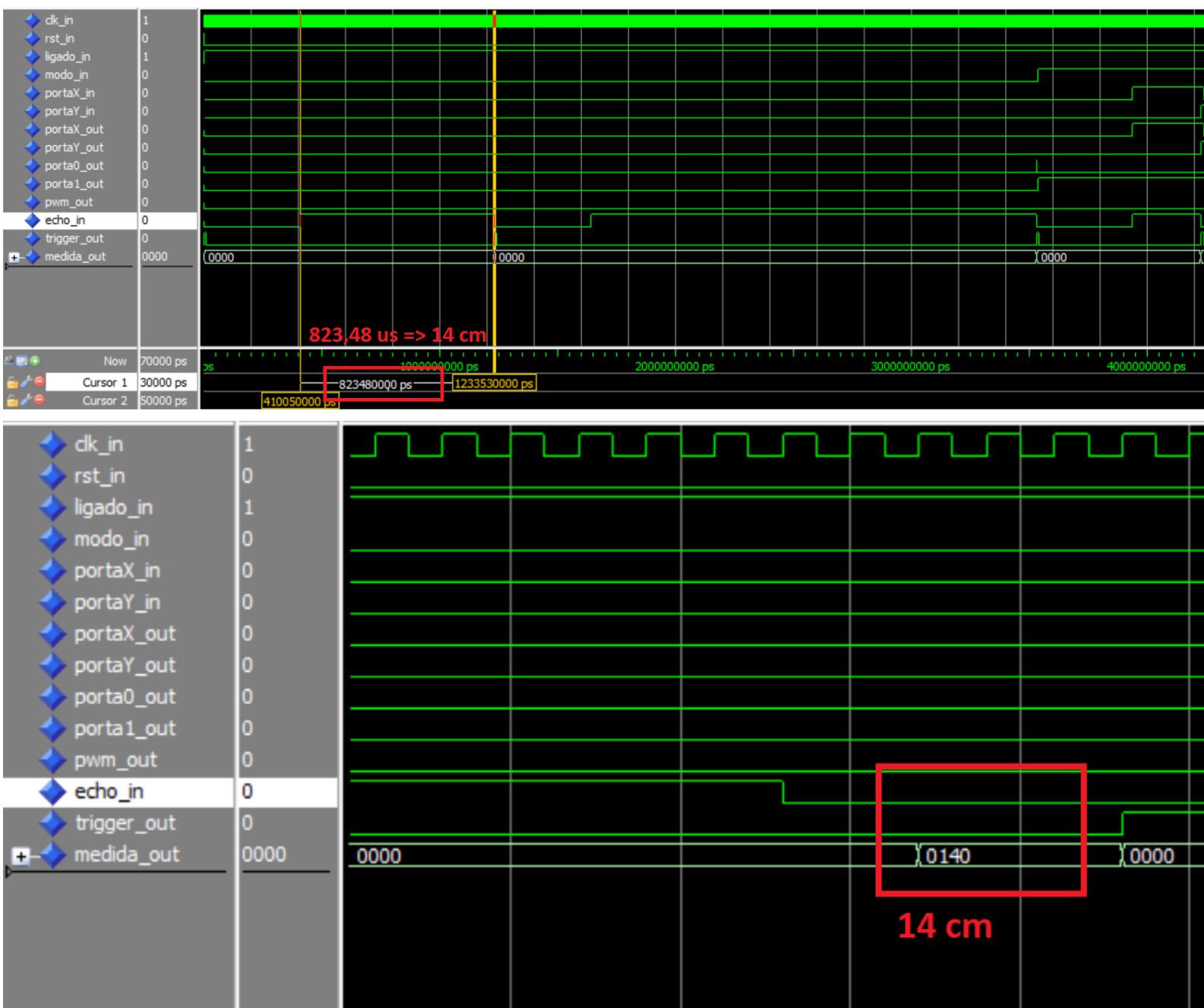




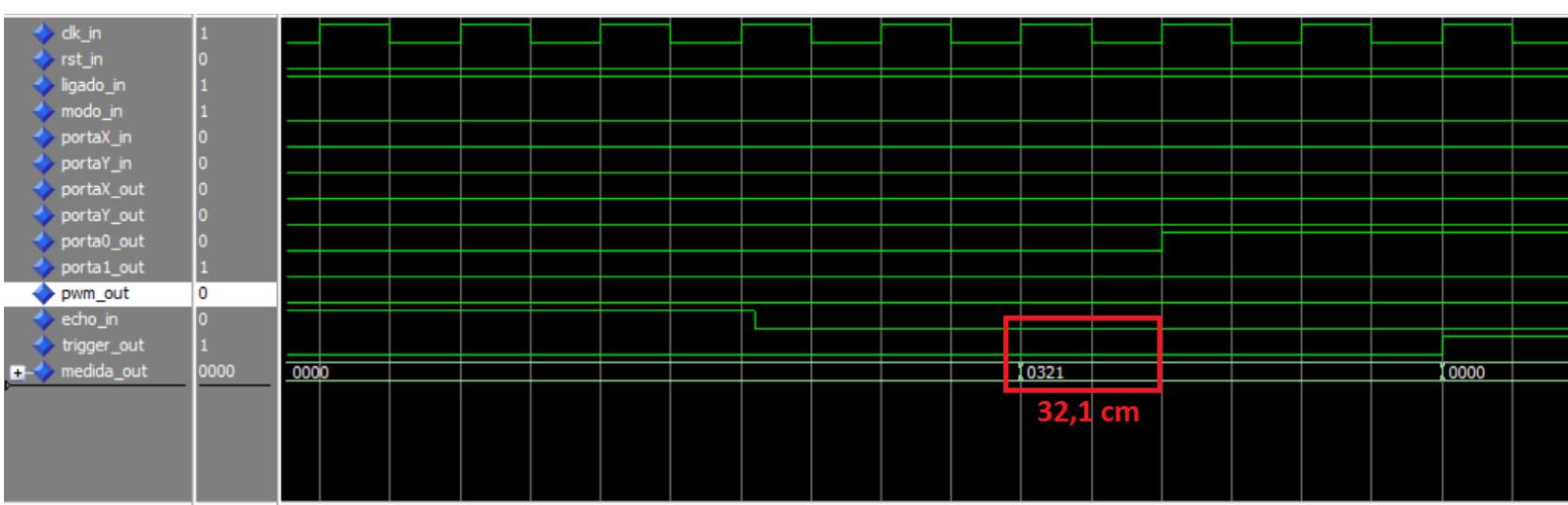
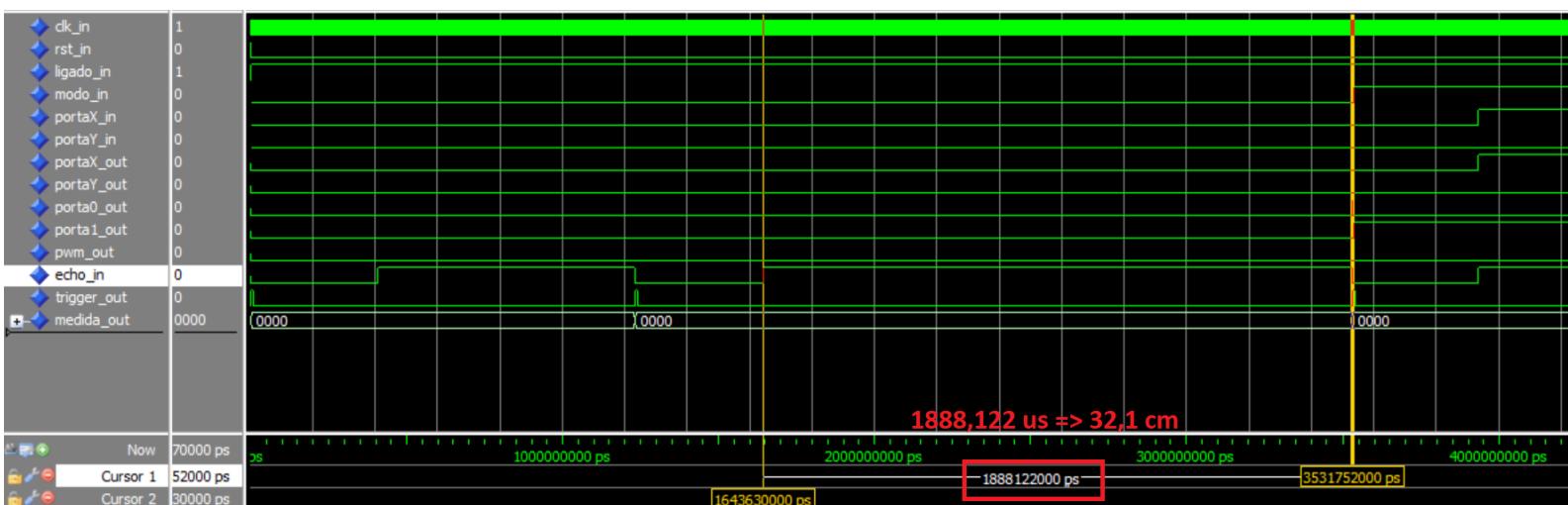
Para a medida do echo apresentado de 588 microssegundos, obtivemos o valor medido de 10 cm, visto que  $D = \frac{T}{58,82} = \frac{588}{58,82} = 10\text{cm}$ . Ou seja, o medidor está funcionando conforme o esperado.

### c) Projeto

- 1) Modo 0 e medição da primeira distância, menor que 30 cm



2) Modo 0 e medição da segunda distância, maior que 30 cm



### 3) Modo 1



Percebe-se que no modo 0 para valores menores que 30 cm, a porta 0 fica com o valor 0, e para valores maiores, ela apresenta um valor 1. Desse modo, vemos que a lógica está invertida, mas segue um padrão lógico aceitável e que pode ser até utilizado, caso mudemos a lógica do projeto.

Agora para o modo 1, qualquer valor colocado no sensor de distancia não altera o comportamento das portas 0 e 1, as quais ficam estáticas. Todavia, alterando manualmente o valor nas entradas das portas X e Y, elas tem o seu estado alterado. Portanto, para utilizarmos este circuito numa só porta, possivelmente teremos de usar um multiplexador.

# Resultados Experimentais

## a) Análise do comportamento do circuito

A fim de fazer com que o circuito funcionasse da maneira apropriada foram feitas algumas análise a respeito do comportamento do circuito. Como foi demonstrado pelas simulações, os módulos funcionaram como era esperado, sendo que o servomotor mudava a largura do pulso pwm para cada posição dada pela entrada, e o medidor de distância indicava a distância correta para cada echo enviado a ele.

Entretanto, quando foi feita a integração entre os módulos, notaram-se alguns comportamentos que não eram esperados. Primeiramente, a lógica para abertura da porta estava invertida, assim, o sinal em alto estava representando a porta aberta, enquanto o sinal em baixo representava a porta fechada. O segundo ponto que deveríamos nos atentar era que a cada início de uma nova medição, o valor medido anteriormente era apagado, e com isso, a porta se fechava, uma vez que a uma distância medida 0, espera-se que o cachorrinho esteja muito próximo da porta. Isso, garantia um comportamento indesejado pelo circuito, o qual abriria e fecharia a porta assim que uma nova edição fosse feita.

## b) Alterações no VHDL

Desse modo, os componentes de comparação da medida de distância e o próprio medidor de distância foram alterados.

### 1) Comparador

Para inverter a lógica empregada apenas mudamos os valores de saída para o inverso. Além disso, como para distâncias muito altas o sensor retorna medida 0, alteramos o código para que a porta continue aberta, com valor em baixo, para distâncias 0.

Com isso, o código ficou como se segue:

```
29 └─ process (i_clock, i_reset, s_medida0, s_medida1, s_medida2, s_0, s_3)
30   begin
31
32   if s_0 < s_medida2 then s_entrada_eh_maior_que_30cm <= '1';
33   elsif s_3 < s_medida1 then s_entrada_eh_maior_que_30cm <= '1';
34   elsif s_0 < s_medida0 and s_3 = s_medida1 then s_entrada_eh_maior_que_30cm <= '1';
35   elsif s_medida0 = s_0 and s_medida1 = s_0 and s_medida2 = s_0 then s_entrada_eh_maior_que_30cm <= '1';
36   else s_entrada_eh_maior_que_30cm <= '0';
37   end if;
38
39 end process;
```

### 2) Medidor de distância

O problema encontrado no medidor de distância era de resetar a medida calculada a cada novo início de uma medição. Verificando o código, pudemos averiguar que a entrada zera do contador estava conectada ao reset do medidor, e este ao trigger do medidor, que seria acionado a cada início de um ciclo. Ou seja, toda vez que uma nova medição era feita, o trigger era acionado, e este acionava o reset do medidor, a fim de zerar a contagem feita anteriormente.

Mudando essa lógica, separando o reset do trigger, obtivemos a seguinte entidade:

```

5  entity medidor_de_distancia is
6    port(
7      i_clock, i_reset : in std_logic;
8      i_zera: in std_logic;
9      i_echo: in std_logic;
10     o_pronto: out std_logic;
11     o_medida: out std_logic_vector (15 downto 0)
12   );
13 end entity;

```

E as seguintes linhas de código:

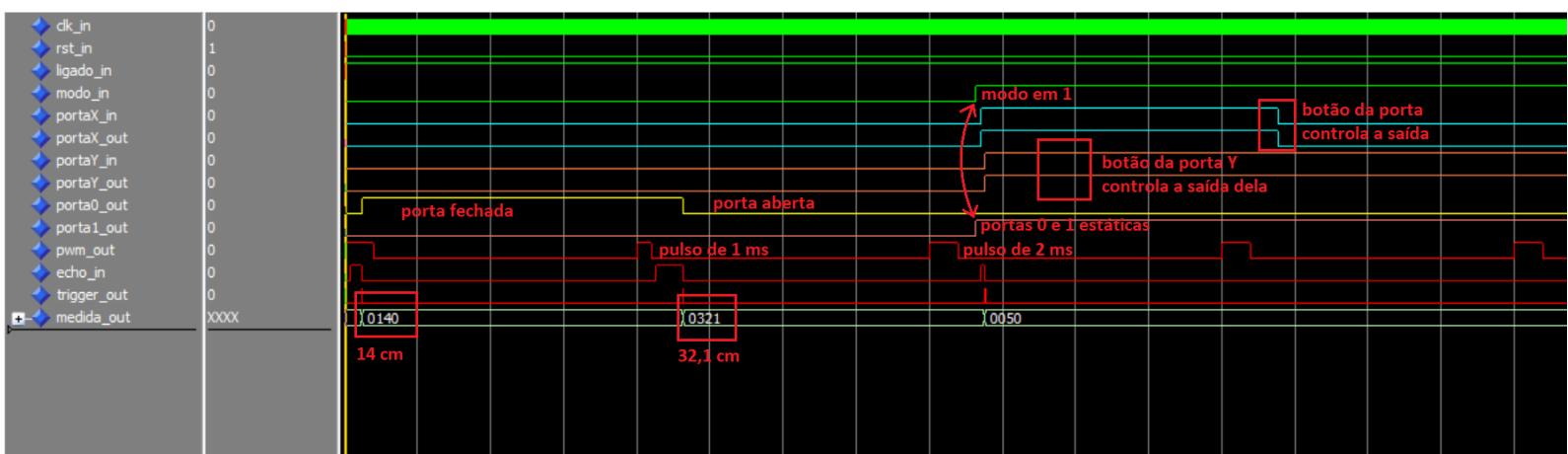
```

58 s_zera <= i_reset or i_zera;
59
60 TICK: contador_m generic map (M => 294, N => 9) port map(
61   i_clock, s_zera, i_echo,
62   open,
63   s_tick
64 );
65
66 BCD: contador_bcd_4digitos port map (
67   i_clock, s_zera, s_tick,
68   s_medida(15 downto 12), s_medida(11 downto 8), s_medida(7 downto 4), s_medida
69 );
70

```

### c) Simulações

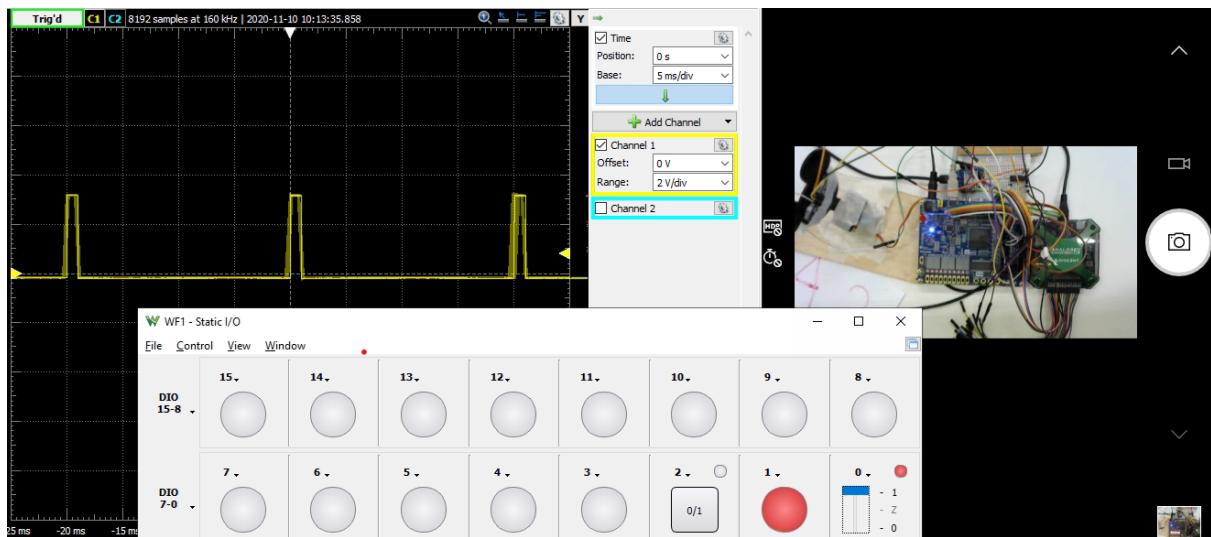
Com essas alterações, foi feita uma nova simulação, a qual apresentou o correto funcionamento do circuito:



Para a porta aberta, o pulso do servomotor difere para quando ela está fechada, e ela apenas fecha para valores menores que 30 cm, como exemplificado. Além disso, quando o modo 1 é ativado, as portas 0 e 1 ficam com a mesma posição até que esse modo seja alterado. Ademais, nestes modo, as portas X e Y são configuradas manualmente.

### d) Laboratório

Enfrentamos algumas dificuldades no laboratório. A princípio seriam feitos testes de todos os módulos incluindo o completo. Todavia, foi realizado somente o teste do servomotor, e mesmo este só foi testado no osciloscópio do Analog Discovery.



Quando fomos testar o servomotor físico, verificamos que o mesmo não se mexia, independentemente da entrada de posição que era dada. Assim, foram colocados sinais de debug para verificar se a placa estava corretamente pinada e não estaria havendo erros de conectividade com o Blynk.

A pinagem utilizada foi a seguinte:

```
i_clock pin_m9
i_reset pin_t20

i_posicao pin_b16

o_pwm pin_k16
db_reset pin_aa2
db_pwm pin_f15
db_posicao pin_aa1
```

Nos quais os debuggers estavam atrelados ou ao osciloscópio, no caso do db\_pwm, ou a LEDs como o db\_reset.

Como o comportamento atípico do servomotor continuou, mais testes foram feitos tanto para descobrir o erro do servo, quanto para fazer verificações parciais do circuito completo. Com exceção do servomotor se deslocar, o restante do circuito funcionou normalmente.

Mesmo com diversos testes e com o auxílio do professor Edson Midorikawa e da Fátima, não foi possível encontrar o erro. Como gastamos quase que o tempo inteiro para encontrar o erro nele, já que seu código e lógica de funcionamento são simples,

não foi possível concluir as demais testagens .Assim, o restante da testagem na placa física terá de ser feito na próxima aula.

## **Semana 3**

### **Processo Experimental**

Para essa semana o grupo concluirá os testes necessários para verificar se os componentes reutilizados e refatorados estão funcionando de acordo com o esperado, assim como para averiguar se a junção destes com a nova unidade de controle não apresenta qualquer comportamento indesejado. Essa parte está especificada no processo experimental da semana 2.

Após isso, faremos testes com o script teste desenvolvido durante a semana para conectar um segundo celular com Blynk ao projeto. Dentre as implementações que queremos adicionar estaria a atualização automática dos estados de um LED com a mudança do estado de um botão.

Desse modo, ao final da aula esperamos demonstrar ao professor os componentes funcionando conforme o esperado, e um teste básico com o script feito ao longo da semana e que será melhorado ao longo do processo experimental da semana.

### **Script**

O script teste realizado pelos alunos é como apresentado a seguir:

```

[ ] !pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)

[ ] import requests

[ ] auth_token1 = "u_EjRouldtNm2_eRYgvAK3JbusMiDuUq"
blynk_server = '45.55.96.146'

[ ] pin = 'V10'

[ ] rbutton = requests.get('http://'+blynk_server+'/'+auth_token1+'/get/'+pin)
print(rbutton.text)
["0"]

[ ] print(rbutton.text[2])
print(auth_token1)
pinled = 'V30'
if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=0')

0
u_EjRouldtNm2_eRYgvAK3JbusMiDuUq

```

Para criá-lo, foi utilizado o script disponibilizado pelo monitor. Desse modo, primeiramente foi instalada biblioteca requests para que ela, então, fosse importada:

```

[ ] !pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)

[ ] import requests

```

Posteriormente foram definidos o token de autenticação do projeto Blynk que se esperava conectar ao script, assim como o número IP do servidor de onde seria feita a conexão:

```

[ ] auth_token1 = "u_EjRouldtNm2_eRYgvAK3JbusMiDuUq"
blynk_server = '45.55.96.146'

```

O resto do script serviu para definir qual o botão de onde seria retirado o estado atual de operação, desligado ou ligado, e então utilizar isso para definir o estado de um LED por meio da função get da biblioteca request, e colocando nela a url correspondente para atualizar ou obter o valor de um pino virtual, V10 no caso apresentado:

```
[ ] pin = 'V10'

[ ] rbutton = requests.get('http://'+blynk_server+'/'+auth_token1+'/get/'+pin)
print(rbutton.text)

["0"]

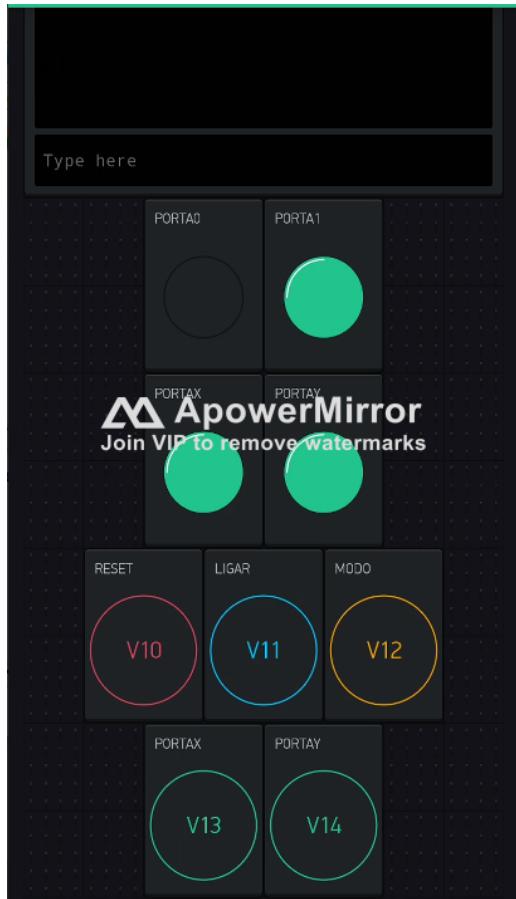
[ ] print(rbutton.text[2])
print(auth_token1)
pinled = 'V30'
if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=0')

0
u_EjRouidtNm2_eRYgvAK3JbusMiDuUq
```

Foram feitos prints para verificar quais eram as saídas obtidas, as quais estavam sendo reutilizadas como entrada para definir os estados dos LED.

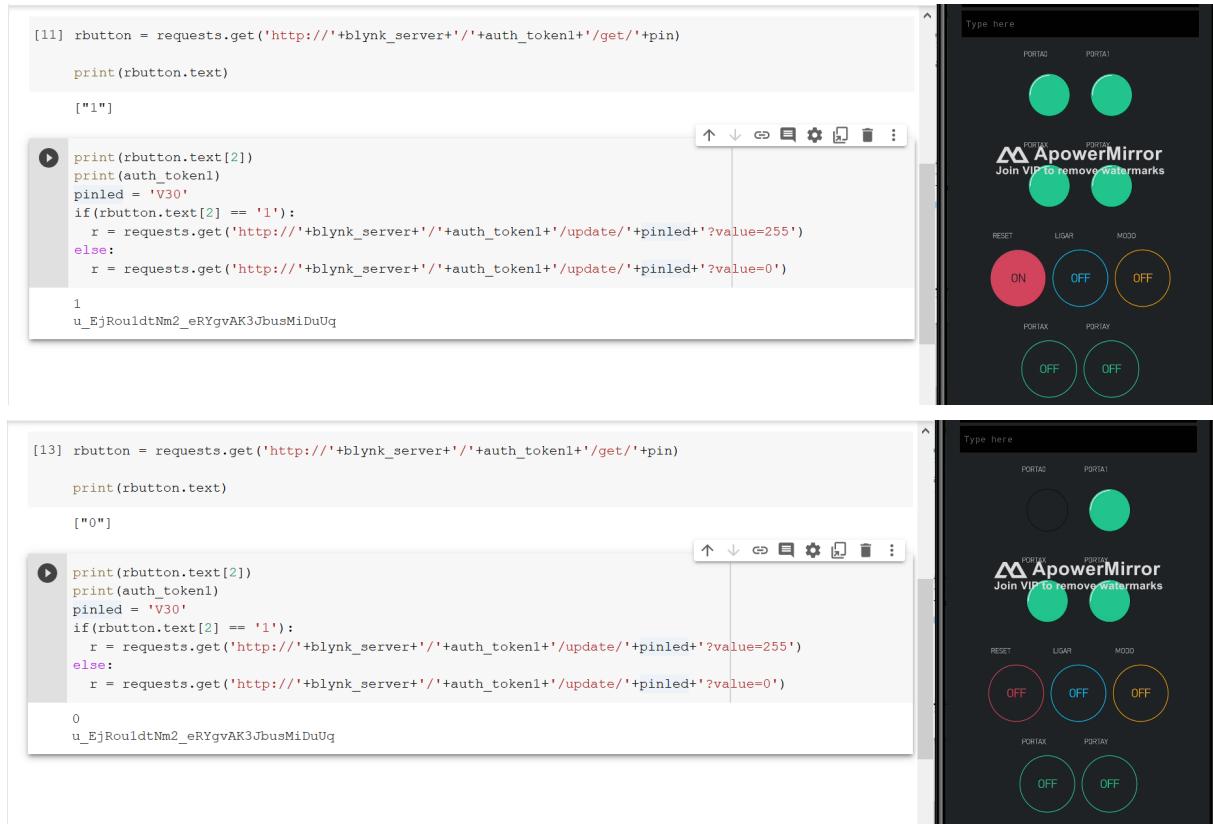
## Teste

A configuração do projeto Blynk utilizada é como se segue:



A partir das imagens a seguir, podemos assegurar que apertar o botão V10 faz com que o LED V30 se acenda, e posteriormente, se for apertado novamente, para uma posição em que o botão tenha valor de sinal baixo, o LED também é apagado.

Os prints mostram o valor, 1 ou 0, do botão, assim como o valor do token de autenticação para o qual o script está conectado.



The image shows two side-by-side screenshots of the Blynk mobile application. Both screens display a digital input interface with four circular buttons labeled 'PORTA0', 'PORTA1', 'PORTA2', and 'PORTA3'. Each button has a color-coded status: green for PORTA0 and PORTA1, and grey for PORTA2 and PORTA3. Below each button is a small text label: 'ON' for PORTA0, 'OFF' for PORTA1, 'OFF' for PORTA2, and 'OFF' for PORTA3. At the top of each screen, there is a header bar with the Blynk logo and some text. The left screenshot corresponds to a script run where the value of PORTA0 is 1, while the right screenshot corresponds to a script run where the value of PORTA0 is 0.

```
[11] rbutton = requests.get('http://'+blynk_server+'/'+auth_token1+'/get/'+pin)
    print(rbutton.text)

    ["1"]

▶ print(rbutton.text[2])
print(auth_token1)
pinled = 'V30'
if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=0')

1
u_EjRouldtNm2_eRYgvAK3JbusMiDuUq
```

```
[13] rbutton = requests.get('http://'+blynk_server+'/'+auth_token1+'/get/'+pin)
    print(rbutton.text)

    ["0"]

▶ print(rbutton.text[2])
print(auth_token1)
pinled = 'V30'
if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=0')

0
u_EjRouldtNm2_eRYgvAK3JbusMiDuUq
```

## Resultados Experimentais

Esperamos demonstrar o funcionamento de todos os componentes e alterar o script de teste para que: consiga se comunicar com dois projetos Blynk distintos; e a atualização de estados sendo consequência da mudança de um botão seja imediata, sem que seja necessário dar run novamente a cada nova atualização do circuito.

```

+ Código + Texto

[ ] !pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)

[ ] import requests
import time

[ ] auth_tokenSane = "u_EjRouIdtNm2_eRYgvAK3JbusMiDUuQ"
auth_tokenGeri = "YHL9bspUaEK2TjBIR0nBSyqZ8wNYcuu5"
blynk_server = '45.55.96.146'

[ ] pin = 'V10'
pin2 = 'V11'
pinled = 'V31'
pinlid2 = 'V32'

[ ] while 1:
    rbutton = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin)
    rbutton2 = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin2)

    print(rbutton.text)
    print(rbutton.text[2])
    print(auth_token1)
    if(rbutton.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinled+'?value=255')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinled+'?value=0')

    if(rbutton2.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinlid2+'?value=255')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinlid2+'?value=0')

```

Daquilo que foi planejado, conseguimos apenas conectar dois celulares com Blynk a um mesmo projeto em script. Desse modo, alteramos algumas linhas do código teste para que ele passasse a receber as informações de um dos celulares e se comunicasse com outro.

Para tanto, criamos 2 variáveis, cada uma contendo o auth\_token de cada aluno. Assim, poderíamos manipular da maneira que fosse mais conveniente para o projeto de maneira a pegar a informação correspondente de um e enviá-la para outro.

Com o intuito de tornar o processo contínuo, também adicionamos um while 1, o qual repetiria o processo de pegar o estado do botão de um celular e então, alteraria o valor do LED de outro, caso atingisse a condição necessária, no caso, apenas se o botão estivesse em alto.

Linhas de código para receber a informação do estado dos botões:

```

rbutton = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin)
rbutton2 = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin2)

```

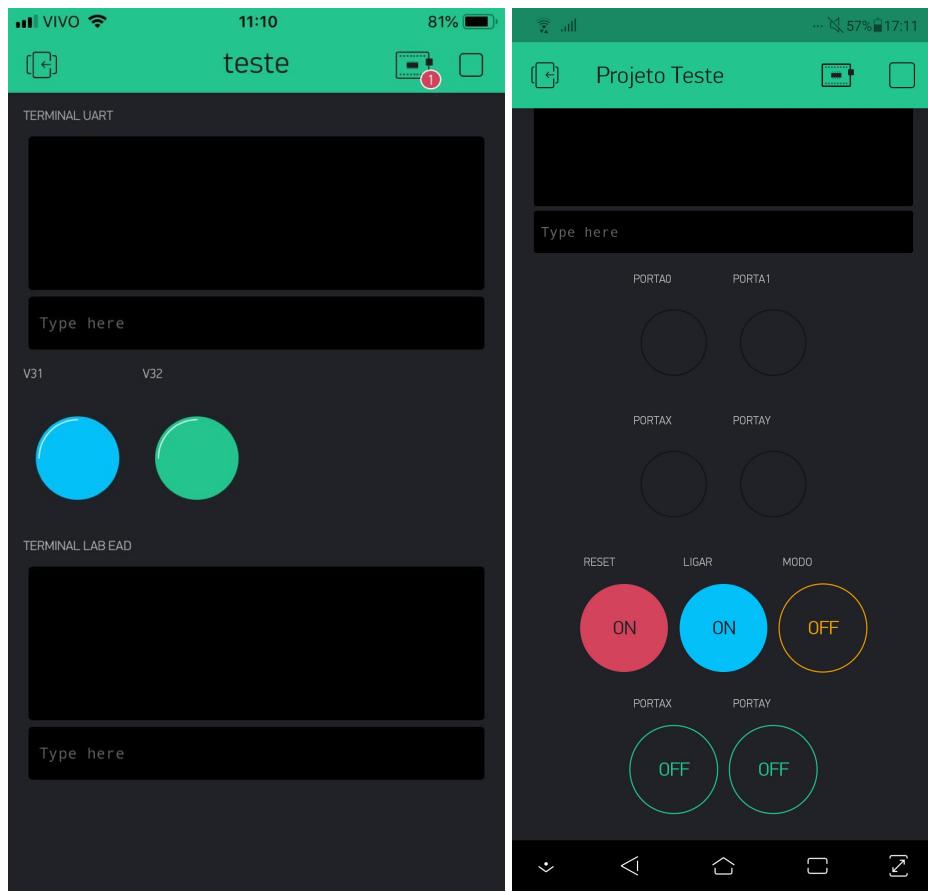
Linhas de código para atualizar o estado dos LED

```

if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/update/'+pinled+'?value=0')

```

As imagens referentes ao teste com dois celulares são apresentadas a seguir:



Sendo que os botões de reset e ligar são respectivamente ligados aos pinos V10 e V11.

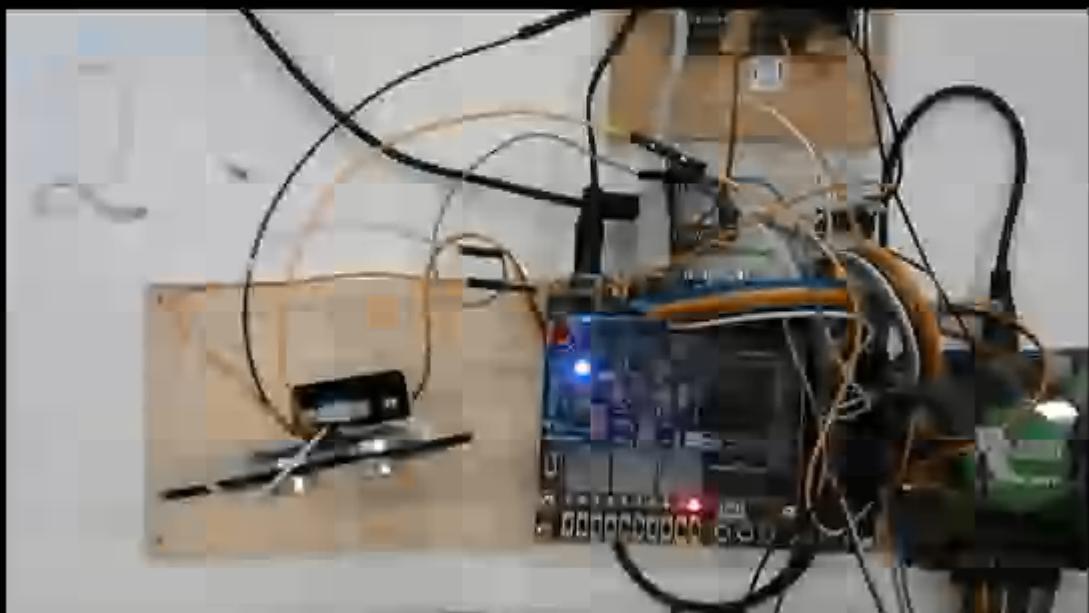
Além disso, passamos para a fase de testes que deveria ter sido feita na semana passada, mas que por problemas técnicos, não foi possível de ser concluída ou mesmo realizada. Efetuamos testes para o servomotor e para o circuito integrador. O servomotor, ao contrário da semana passada, funcionou conforme o esperado. O circuito do projeto, entretanto, não teve o mesmo resultado. Inicialmente, o circuito não detectava qualquer objeto próximo ao sensor. Uma vez trocado, ele passou a identificar, por meio de um sinal de LED que indicava quando algo estivesse a uma distância menor que 30 centímetros. O problema encontrado foi o de ele realizar a medição apenas uma vez, ainda que fosse pedido para ser realizado medições contínuas dentro da especificação do projeto e com as simulações feitas. Assim, o circuito media a primeira distância, se movimentava e então permanecia no mesmo lugar.

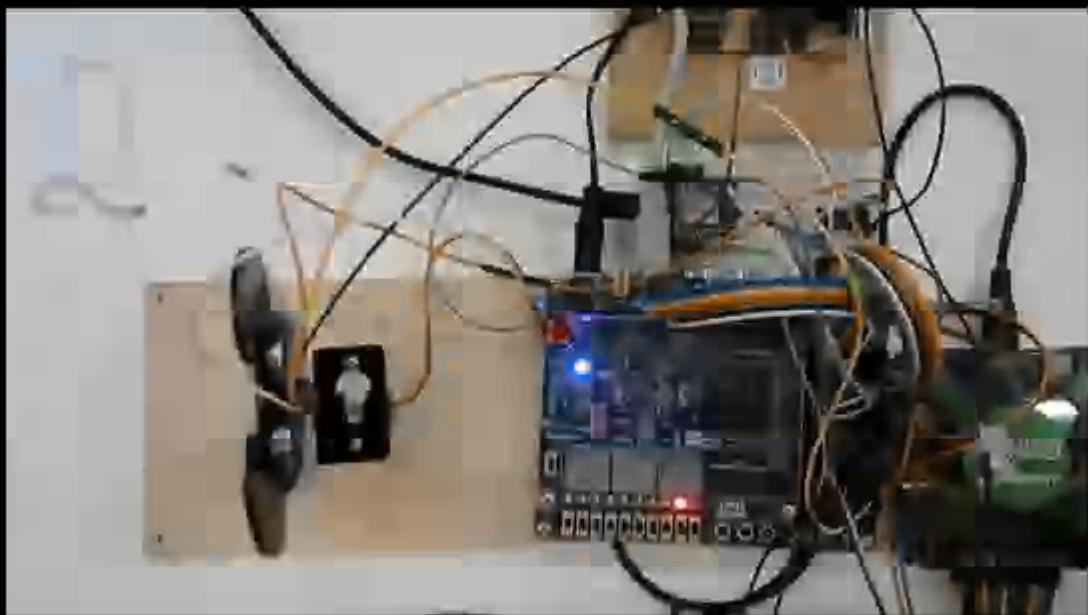
As imagens referentes a esse teste foram colocadas a seguir:



HDD

C





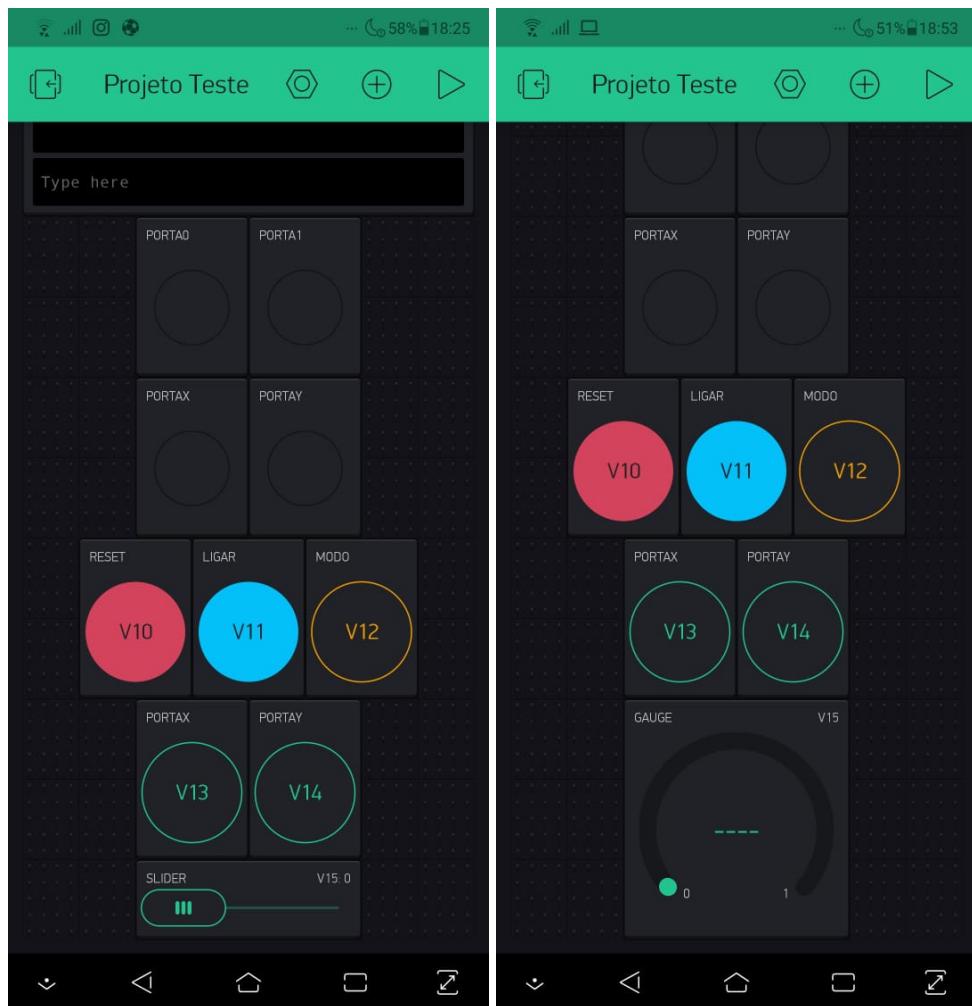
Não foi possível concluir a testagem para averiguar melhor o erro, mas julgamos ser algo que envolva ou a unidade de controle do circuito de segurança de filhote ou ao medidor de distância. Em todo caso, iremos aproveitar o uso do Openlab para verificar se o erro está em um ou outro, colocando uma entrada para definir a distância dada ao sensor e mais sinais de depuração.

# Semana 4

## Processo Experimental

Essa semana iremos testar novamente o circuito do projeto para segurança de filhotes. Dessa vez, entretanto, será feito uso de uma entrada específica, setada pelo Blynk, para determinar a distância de um filhote para o sensor de distância. Desse modo, não será necessário depender de terceiros para verificação do funcionamento.

Assim, primeiramente será feito testes com apenas um celular Blynk, o qual a partir da widget slider indicará a posição do filhote. A widget será configurada para apenas 2 posições, no momento, visto que apenas é necessário verificar se a distância é superior ou inferior a 30 cm, valor de distância apenas ilustrativo e que pode ser facilmente modificado de acordo com o tamanho de cada porta a ser utilizada. A primeira imagem a seguir ilustra um projeto Blynk com o slider configurado para 2 posições:



A ideia para utilização e representação do filhote é de que um dos celulares dos integrantes do grupo represente o cachorro, enquanto o outro representa a porta, tendo os sinais de controle. Assim, o projeto Blynk filhote terá a widget slide para escolher a posição do cachorro, enquanto o projeto Blynk porta terá um gauge que indicará a posição do cachorro e comunicará ao circuito qual a posição dele. A segunda imagem acima ilustra um possível projeto Blynk no qual o gauge é acionado por meio de um script e tem seu estado alterado e então repassado para o circuito.

Desse modo, após serem feitos os testes com apenas um celular, faremos o teste com ambos os celulares de acordo com o seguinte plano de testes:

Passo	Ação
1	Iniciar o circuito
2	Acionar o reset
3	modo 0
4	Mudar o valor do slider para 0
5	Observar os Leds no Blynk e ver se a porta abriu
6	mudar o valor do slide para 1
7	Observar os Leds no Blynk e ver se a porta abriu
8	modo 1
9	Mudar o valor do slider para 0
10	Observar os Leds no Blynk e ver se a porta abriu
11	mudar o valor do slide para 1
12	Observar os Leds no Blynk e ver se a porta abriu

Desse jeito, será possível ver se as portas abrem de acordo com o esperado no modo 0 e se permanecem estáticas no modo 1.

O que se espera apresentar ao professor ao final da aula é o projeto funcionando com um e dois celulares com Blynk conectados.

## Script

Ao contrário da semana passada, o script atual terá apenas os pins necessários para manipulação das widgets slider e gauge. Será pego o valor do slider de um dos celulares:

```
slider = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/get/'+slider)
```

E caso o valor dele seja 1, será colocado o valor 1 no gauge também:

```
if(slider.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=1')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=0')
```

Esse processo será contínuo, por meio de um while 1, e de acordo com o valor no gauge, será alterado o modo de operação do circuito

```
[ ] auth_tokenSane = "u_EjRou1dtNm2_eRYgvAK3JbusMiDuUq"
auth_tokenGeri = "VHL9bspUaEKe2TjBIR0nBSyz8wNYcuu5"
blynk_server = '45.55.96.146'

[ ] slider = 'V10'
gauge = 'V34'

[ ] while 1:
    slider = requests.get('http://'+blynk_server+'/'+auth_tokenGeri+'/get/'+slider)

    if(slider.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=1')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=0')
```

## Open Lab

O grupo foi ao Open Lab para realizar testes e verificar o funcionamento dos componentes antes da aula em si. Desse modo, o que foi feito foi testar colocando valores de distância na mão e confirmar se a lógica do circuito está falha. Além disso, foram incluídos alguns sinais de debug como o de proximidade e a distância calculada e sendo comparada nos displays de 7 segmentos.

```
160   HEX0: hexa7seg port map (
161     s_medida_distancia_em_BCD(7 downto 4),
162     o_display0
163   );
164
165   HEX1: hexa7seg port map (
166     s_medida_distancia_em_BCD(11 downto 8),
167     o_display1
168   );
169
170   HEX2: hexa7seg port map (
171     s_medida_distancia_em_BCD(15 downto 12),
172     o_display2
173   );
```

```

i_clock pin_m9
i_reset pin_n16

i_ligado pin_b16
i_modo pin_m16
i_portaX pin_c16
i_portaY pin_d17

i_filhote pin_k20

i_echo pin_t17

o_porta0 pin_h16
o_porta1 pin_a12
o_portaX pin_h15
o_portaY pin_b12

o_pwm pin_k16
o_trigger pin_j17

db_reset pin_aa2
db_proximo pin_l1          o_display1[2] pin_aa19
                            o_display1[3] pin_aa18
                            o_display1[4] pin_ab18
                            o_display1[5] pin_aa17
                            o_display1[6] pin_u22
o_display0[0] pin_u21       o_display2[0] pin_y19
o_display0[1] pin_v21       o_display2[1] pin_ab17
o_display0[2] pin_w22       o_display2[2] pin_aa10
o_display0[3] pin_w21       o_display2[3] pin_y14
o_display0[4] pin_y22       o_display2[4] pin_v14
o_display0[5] pin_y21       o_display2[5] pin_ab22
o_display0[6] pin_aa22     o_display2[6] pin_ab21

o_display1[0] pin_aa20
o_display1[1] pin_ab20

```

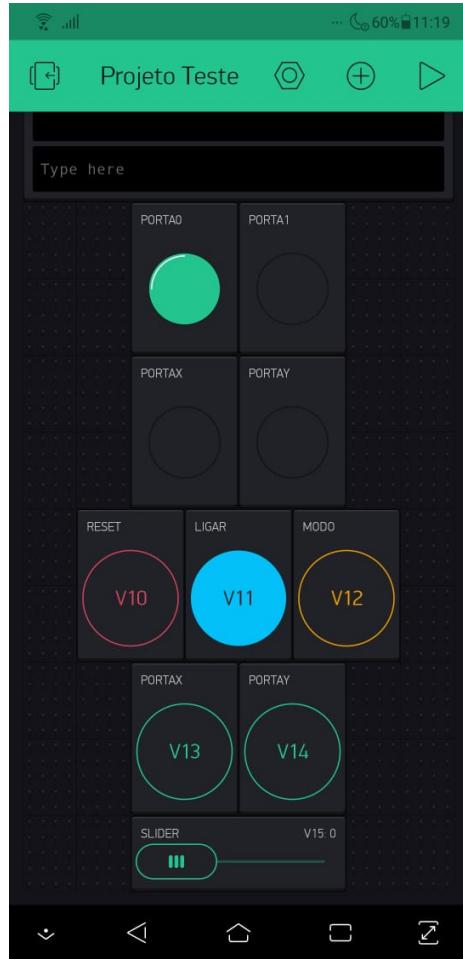
Primeiramente, o que foi feito foi colocar a distância sendo medida na mão, por meio de um seletor controlado por um slider em um projeto Blynk. Assim, quando o slider indicasse o valor 0, a distância do sensor seria 1 cm, menor que 30 cm, enquanto que para o valor 1, a distância seria 100 cm, maior que 30 cm. As linhas de código alteradas ficaram da seguinte forma:

```

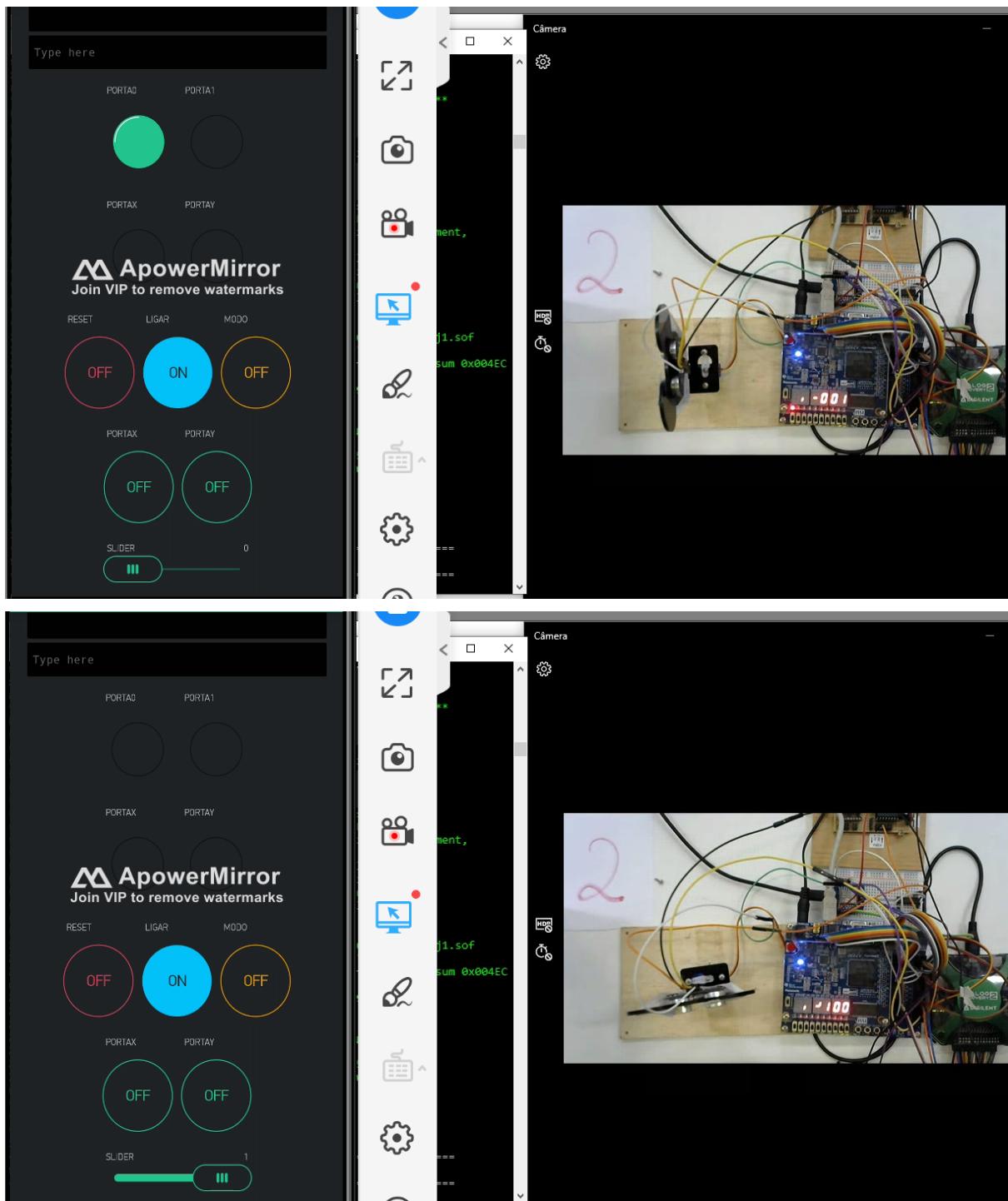
147 |     with i_filhote select
148 |         s_medida_distancia_em_BCD <= "0001000000000000" when '1', --100cm
149 |                         "000000000010000" when '0', --0,1 cm
150 |                         "0000000000000000" when others;

```

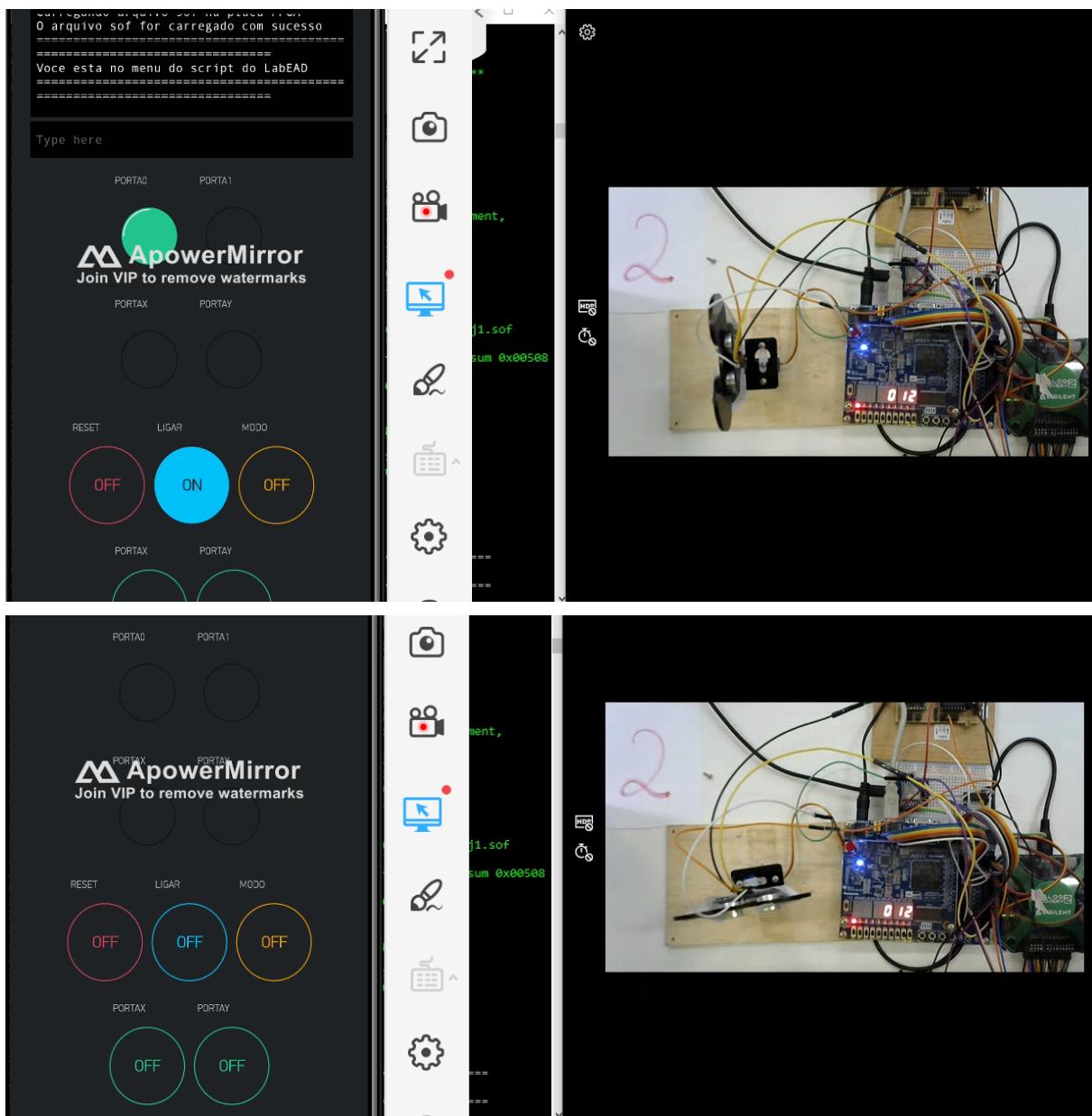
E o projeto Blynk da seguinte forma:



Com essa configuração, notou-se que o circuito funcionou conforme o esperado, e para distâncias superiores e inferiores a 30 cm, o servomotor, de fato, apresentava posições diferentes, como pode ser mostrado nas imagens a seguir:



Ou seja, o problema não se encontrava na medida, e sim na medição. Com isso concluído, Voltamos a versão inicial do projeto, que foi testada na aula passada para observar novamente o comportamento dele, com os displays funcionando dessa vez. O que se observou foi que era feito apenas uma medição independentemente se a posição do servomotor mudava ou se a distância era alterada posteriormente, sendo necessário o uso do reset para uma nova medição.



Depois de feito esse teste, propomos alterar o sinal de echo de maneira manual. Para isso, as seguintes linhas de código foram inseridas, para que o slider quando com valor 1, mudasse o echo para 1, e quando em 0, o echo para 0.

147	with i_filhote select
148	s_echo <= '1' when '1', --100cm
149	'0' when '0', --0,1 cm
150	'0' when others;

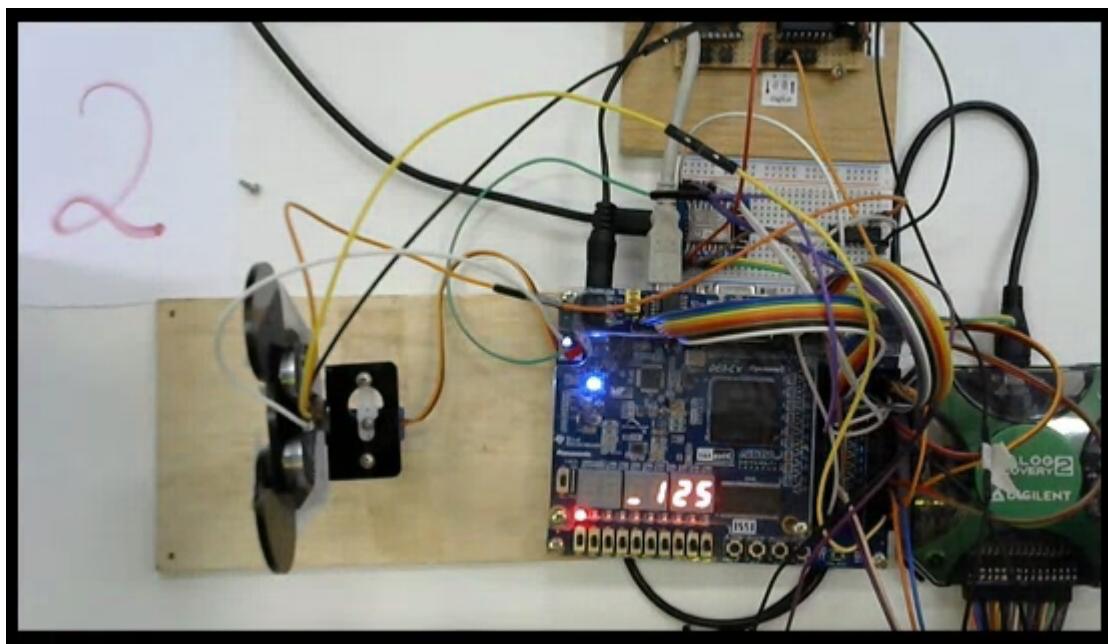
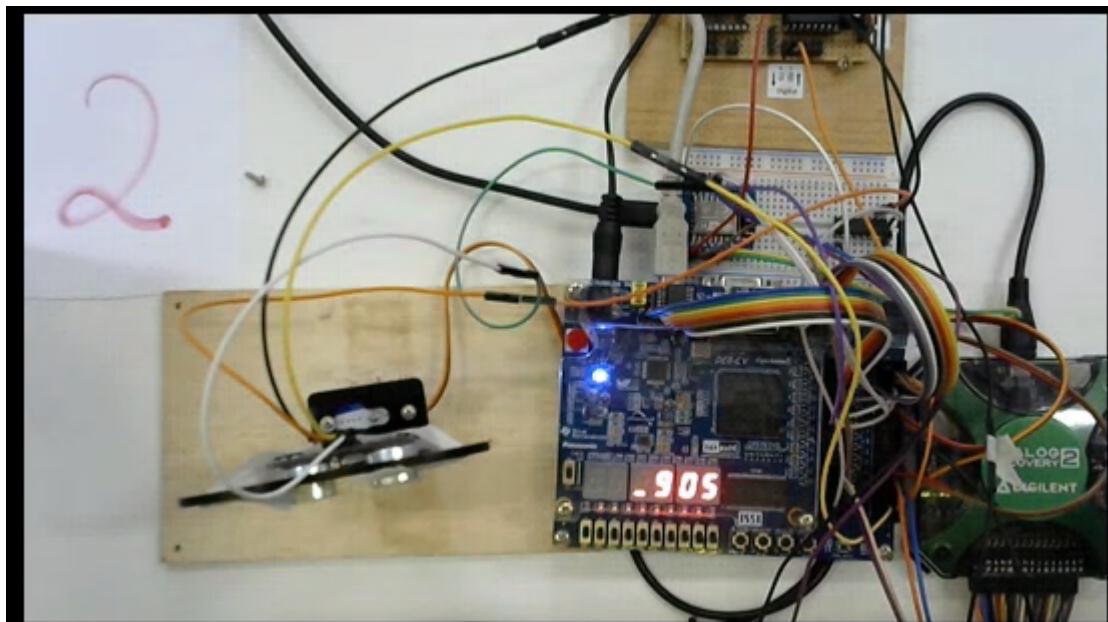
Além disso, para facilitar a comparação de números e de distância, o valor base para alterar o sinal de proximidade passou a ser 500 e não 30.

```

21    s_0 <= 0;
22    s_3 <= 3;
23    s_5 <= 5;
24
25    s_medida0 <= to_integer(unsigned(i_medida(7 downto 4)));
26    s_medida1 <= to_integer(unsigned(i_medida(11 downto 8)));
27    s_medida2 <= to_integer(unsigned(i_medida(15 downto 12)));
28
29  end process;
30  begin
31
32    if s_5 < s_medida2 then s_entrada_eh_maior_que_30cm <= '1';
33    elsif s_0 < s_medida1 then s_entrada_eh_maior_que_30cm <= '1';
34    elsif s_0 < s_medida0 and s_3 = s_medida1 then s_entrada_eh_maior_que_30cm <= '1';
35    elsif s_medida0 = s_0 and s_medida1 = s_0 and s_medida2 = s_0 then s_entrada_eh_maior_que_30cm <= '1';
36    else s_entrada_eh_maior_que_30cm <= '0';
37    end if;
38
39  end process;
40
41
42  o_entrada_eh_menor_que_30cm <= not s_entrada_eh_maior_que_30cm;
43

```

Feitas essas alterações, o resultado experimental obtido foi o seguinte:



No qual é possível observar que para distâncias menores de 500 o sinal de proximidade, representado pelo LED 9, tem seu valor convertido para 1 e o servomotor se desloca. Enquanto que para valores maiores, o LED apaga e o servomotor muda para sua posição inicial, de porta aberta. Além disso, os valores foram inseridos sem que o reset fosse acionado, e portanto, conclui-se que o circuito aceita echos contínuos, e os mede em cada instância apresentada, tendo a comparação de valores e consequente operação sendo realizados corretamente. Portanto, o circuito original não estava funcionando, pois, de alguma forma, o sensor não estaria recebendo novos echos.

## Resultados experimentais

Durante o tempo de aula dessa semana, realizamos os últimos testes referentes ao projeto. Além disso, tentamos utilizar o segundo servomotor que foi disponibilizado pelos professores.

Como feito anteriormente no OpenLab, o grupo conseguiu demonstrar o funcionamento do circuito utilizando da widget slider do projeto Blynk para determinar a posição do cachorrinho e colocar o valor de distância manualmente, visto que para a demonstração, isso se faz necessário, uma vez que não seria bom que o projeto fosse dependente de terceiros.

Em seguida passamos para uma fase de testes do segundo servomotor, e para isso, modificamos a chamada de um dos componentes do servomotor, para que sua posição determinasse a saída pwm do respectivo motor:

```
114 PORTAX: controle_servo port map (
115     i_clock, i_reset,
116     s_portax,
117     s_pwmx,
118     open,
119     open,
120     open
121 );
```

E o arquivo .txt contendo a pinagem foi atualizado:

```

i_clock pin_m9
i_reset pin_n16

i_ligado pin_b16
i_modo pin_m16
i_portaX pin_c16
i_portaY pin_d17

i_filhote pin_k20

i_echo pin_t17

o_porta0 pin_h16
o_porta1 pin_a12
o_portaX pin_h15
o_portaY pin_b12

o_pwm pin_g12
o_pwmx pin_k16|
o_trigger pin_j17      o_display1[0] pin_aa20
db_pwmx pin_f15        o_display1[1] pin_ab20
                        o_display1[2] pin_aa19
                        o_display1[3] pin_aa18
db_reset pin_aa2       o_display1[4] pin_ab18
db_proximo pin_l1      o_display1[5] pin_aa17
                        o_display1[6] pin_u22

o_display0[0] pin_u21   o_display2[0] pin_y19
o_display0[1] pin_v21   o_display2[1] pin_ab17
o_display0[2] pin_w22   o_display2[2] pin_aa10
o_display0[3] pin_w21   o_display2[3] pin_y14
o_display0[4] pin_y22   o_display2[4] pin_v14
o_display0[5] pin_y21   o_display2[5] pin_ab22
o_display0[6] pin_aa22  o_display2[6] pin_ab21

```

Foi utilizada a mesma lógica que para o servomotor já funcional, alterando-se apenas o pino, correspondente ao servomotor correto e o nome do componente sendo utilizado. Todavia, não foi observado qualquer movimento quando o mesmo foi acionado. Percebendo isso, criamos o sinal de debug db\_pwmx, para avaliar sua forma no osciloscópio presente no Analog Discovery. Utilizando desse recurso, percebeu-se que o sinal estava funcionando conforme o esperado. Posteriormente, ainda foram trocados os sinais, aquele que estava alimentando o primeiro servomotor passou a servir como sinal de entrada do segundo, para verificarmos se o problema estava na lógica ou no hardware. Mesmo assim, não houveram resultados positivos e foi concluído que ou o servomotor havia quebrado ou algum fio não estava bem conectado.

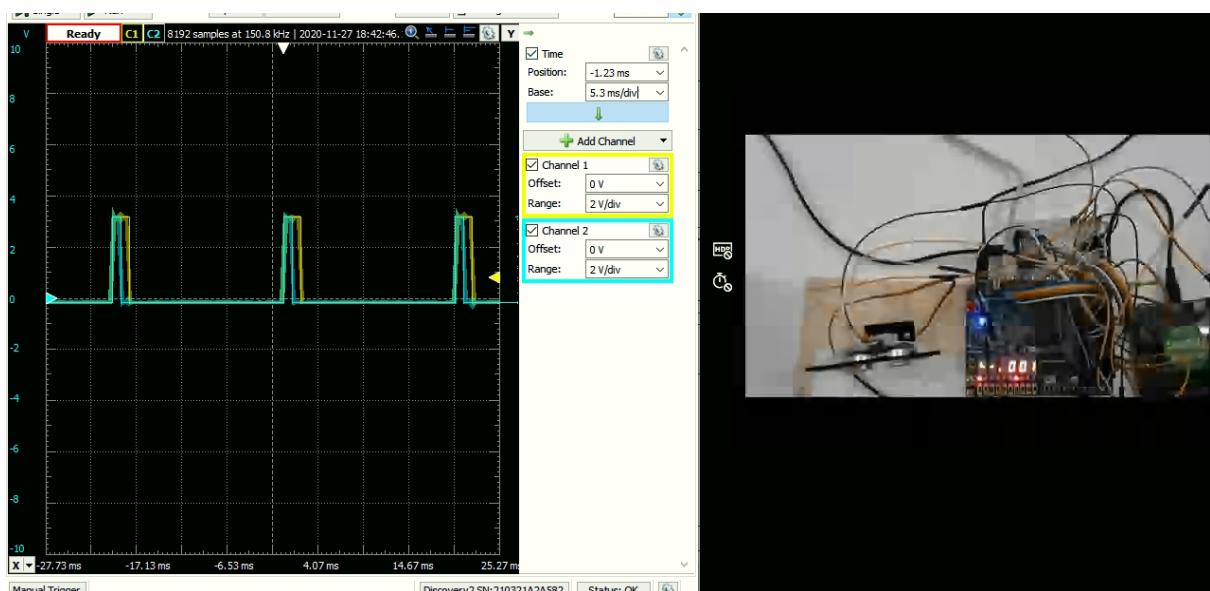
Também foi feito um teste para comunicação entre os celulares dos integrantes do grupo. Nesse teste, um dos membros teria a widget slider, a qual seleciona a posição do cachorro, e o outro a widget gauge. A lógica utilizada foi a de que o slider passaria seu valor para o gauge e este, por sua vez, atualizaria a posição do filhote no circuito.

O script apresentou inicialmente erros, mas uma vez reescrita a linha que pegava o valor do slider, o código passou a funcionar e a comunicação foi bem sucedida. Assim, um dos alunos teve controle de uma porta enquanto o outro teve controle do circuito como um todo.

# Semana 5

## OpenLab

Durante o período de OpenLab disponibilizado, o grupo realizou novos testes com o segundo servomotor, entretanto, o grupo não pode concluir os testes, visto que no meio deles o servomotor teve que ser alocado para outra bancada, para outros testes. Desse modo, foi discutido como seria feita a apresentação e foi decidido que poderia ser provado que o projeto estava funcional por meio dos LEDs e do sinal analisado pelo osciloscópio do Analog Discovery, como pode-se mostrar na imagem a seguir:



Além disso, os displays de 7 segmentos auxiliam na visualização da distância sendo medida e consequentemente na apresentação.

Ademais, foi feito o vídeo de apresentação inicial com o auxílio da ferramenta de gravação pelo Google Meets. Como era necessário espelhar dois celulares, de cada um dos alunos do grupo, foram utilizadas duas salas de Meets, cada uma compartilhando sua tela.

O vídeo produzido atingiu o tempo de 11 minutos e 51 segundos, porém, como foi todo bem explicado e conciso, julgamos não haver problema passar 51 segundos do tempo estipulado. Nesse tempo, foram apresentados o tema, problemática, soluções, simulação e script feito.

# Conclusão

Por meio da concepção e implementação do projeto de segurança de filhotes no Laboratório Digital 2, foi possível aplicar conceitos relacionados a IoT, planejamento e desenvolvimento de ideias, além da experiência com pitches de curta duração, que são muito recorrentes no meio profissional em que as ideias devem ser apresentadas de forma concisa e objetiva para se vender o produto. Ademais, também foram agregados os conceitos do sensor, sonar, transmissores e receptores seriais, assim como terminais UART por meio da primeira parte da disciplina, a qual conversou diretamente com a parte do projeto.

Com relação ao projeto em si, foi possível implementar ele inteiro, sendo que modificações foram feitas para viabilizar o processo de demonstração na feira virtual. Desse modo, quando há um filhote que se aproxima do sensor de distância, este fecha uma porta considerada principal. Quando se quer que o filhote possa sair do cômodo principal, é possível alterar o modo de operação do circuito, para que a porta principal se abra, uma porta que não pode ser acessada permaneça fechada e as demais sejam controladas por botões. Além disso, o script para conexão entre dois celulares Blynk funcionou conforme o esperado, apresentando poucos erros durante seu desenvolvimento.

Vale ressaltar que, pela indagação do professor Jorge Luis Risco Becerra e por recomendação do professor Marco Túlio Carvalho de Andrade, presentes durante a apresentação final do projeto, caso seja levado adiante, o projeto deve ser implementado de maneira a apresentar um caráter mais generalista, de maneira que mais casas pudessem receber esse produto, independente do número de portas que possam ser abertas ou que devam permanecer fora do alcance de filhotes. Outra sugestão formulada pelo grupo seria a de fazer circuitos customizados e direcionados a cada cliente.

# Apêndice

## VHDL

### a) Controlador do circuito

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity controlador_comodos_para_filhotes_uc is
5 port
6 (
7     i_clock, i_reset: in std_logic;
8     i_ligado, i_modo, i_portax, i_portaY, i_esta_proximo: in std_logic;
9     i_pronto_HCSR04: in std_logic;
10    o_porta0, o_porta1, o_portax, o_portaY: out std_logic; -- porta em 0 é aberta em 1 é fechada
11    o_medir_distancia: out std_logic;
12    db_estado: out std_logic_vector(1 downto 0)
13 );
14 end entity;
15
16 architecture arch_controlador_uc of controlador_comodos_para_filhotes_uc is
17
18 type tipo_estado is (inicial, modo0_porta_aberta, modo0_porta_fechada, modo1);
19 signal Eatual: tipo_estado; -- estado atual
20 signal Eprox: tipo_estado; -- proximo estado
21
22 begin
23
24 process (i_reset, i_clock)
25 begin
26 if i_reset = '1' then
27     Eatual <= inicial;
28 elsif i_clock'event and i_clock = '1' then
29     Eatual <= Eprox;
30
31 end if;
32 end process;
33
34 process (i_ligado, i_modo, i_esta_proximo, Eatual)
35 begin
36
37 case Eatual is
38 when inicial =>
39     if i_ligado = '0' then Eprox <= inicial;
40     elsif i_modo = '0' then Eprox <= modo0_porta_aberta;
41     else
42         Eprox <= modo1;
43     end if;
44
45 when modo0_porta_aberta =>
46     if i_ligado='0' then Eprox <= inicial;
47     elsif i_modo='1' then Eprox <= modo1;
48     elsif i_esta_proximo='1' then Eprox <= modo0_porta_fechada;
49     else
50         Eprox <= modo0_porta_aberta;
51     end if;
52
53 when modo0_porta_fechada =>
54     if i_ligado='0' then Eprox <= inicial;
55     elsif i_modo='1' then Eprox <= modo1;
56     elsif i_esta_proximo='0' then Eprox <= modo0_porta_aberta;
57     else
58         Eprox <= modo0_porta_fechada;
59
60 when others =>
61     Eprox <= inicial;
62
63 end case;
64 end process;
65
66 -- logica de saida (Moore)
67 with Eatual select
68     o_porta0 <= '1' when modo0_porta_fechada, '0' when others;
69
70 with Eatual select
71     o_porta1 <= '1' when modo1, '0' when others;
72
73 with Eatual select
74     o_portax <= i_portax when modo1, '0' when others;
75
76 with Eatual select
77     o_portaY <= i_portaY when modo1, '0' when others;
78
79 with Eatual select
80     db_estado <= "00" when inicial,
81             "01" when modo0_porta_aberta,
82             "10" when modo0_porta_fechada,
83             "11" when modo1;
84
85     o_medir_distancia <= not i_pronto_HCSR04;
86 end arch_controlador_uc;
```

### b) Interface do circuito

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity controlador_comodos_para_filhotes is
6 port (
7   i_clock, i_reset: in std_logic;
8   i_ligado, i_modo, i_portax, i_portay: in std_logic;
9   o_porta0, o_porta1, o_portax, o_portay: out std_logic; -- porta em 0 é aberta em 1 é fechada
10  i_echo: in std_logic;
11  o_pwm, o_trigger: out std_logic;
12  db_medida: out std_logic_vector (15 downto 0)
13 );
14 end entity;
15
16 architecture arch_controlador of controlador_comodos_para_filhotes is
17
18 signal s_esta_proximo, s_porta0, s_porta1, s_portax, s_portay, s_medir_distancia, s_pronto_HCSR04: std_logic;
19 signal s_medida_distancia_em_BCD: std_logic_vector (15 downto 0);
20
21 component controlador_comodos_para_filhotes_uc is
22 port (
23   i_clock, i_reset: in std_logic;
24   i_ligado, i_modo, i_portax, i_portay, i_esta_proximo: in std_logic;
25   i_pronto_HCSR04: in std_logic;
26   o_porta0, o_portax, o_portay: out std_logic; -- porta em 0 é aberta em 1 é fechada
27   o_medir_distancia: out std_logic;
28   db_estado: out std_logic_vector(1 downto 0)
29 );
30 end component;
31
32 component controle_servo is
33 port (
34   i_clock, i_reset: in std_logic;
35   i_posicao : in std_logic;
36   o_pwm: out std_logic;
37   db_reset: out std_logic;
38   db_pwm: out std_logic;
39   db_posicao: out std_logic
40 );
41 end component;
42
43 component interface_hcsr04 is
44 port (
45   i_clock, i_reset : in std_logic;
46   i_medir, i_echo: in std_logic;
47   o_trigger, o_pronto: out std_logic;
48   o_medida: out std_logic_vector (15 downto 0);
49   db_medir: out std_logic;
50   db_estado: out std_logic_vector (2 downto 0)
51 );
52 end component;
53
54 component comparador_menor_30cm is -- usei 30cm como exemplo apenas, a ideia é modificar o componente para o valor ex:
55 port (
56   i_clock, i_reset: in std_logic;
57   i_medida: in std_logic_vector (15 downto 0);
58   o_entrada_eh_menor_que_30cm: out std_logic
59 );
60 end component;
61
62 begin
63
64 UC: controlador_comodos_para_filhotes_uc port map (
65   i_clock, i_reset,
66   i_ligado, i_modo, i_portax, i_portay, s_esta_proximo,
67   s_pronto_HCSR04,
68   s_porta0, s_porta1, s_portax, s_portay,
69   s_medir_distancia,
70   open
71 );
72
73 PORTA0: controle_servo port map (
74   i_clock, i_reset,
75   s_porta0,
76   o_pwm,
77   open,
78   open,
79   open
80 );
81
82
83
84
85
86
87

```

```

88 PORTA1: controle_servo port map ( -- essa e as seguintes foram colocadas sem ter uma saida pwm pois por enquanto
89             i_clock, i_reset,
90             s_porta1,
91             open,
92             open,
93             open,
94             open
95             );
96
97 PORTAX: controle_servo port map (
98             i_clock, i_reset,
99             s_portaX,
100            open,
101            open,
102            open,
103            open
104            );
105
106 PORTAY: controle_servo port map (
107             i_clock, i_reset,
108             s_portaY,
109             open,
110             open,
111             open,
112             open
113             );
114
115 HCSR04: interface_hcsr04  port map (
116             i_clock, i_reset,
117             s_medir_distancia, i_echo,
118             o_trigger, s_pronto_HCSR04,
119             s_medida_distancia_em_BCD,
120             open,
121             open
122             );
123
124
125 COMPARADOR_MEDIDA: comparador_menor_30cm  port map (
126             i_clock, i_reset,
127             s_medida_distancia_em_BCD,
128             s_esta_proximo
129             );
130
131             o_porta0 <= s_porta0;
132             o_porta1 <= s_porta1;
133             o_portaX <= s_portaX;
134             o_portaY <= s_portaY;
135             db_medida <= s_medida_distancia_em_BCD;
136
137 end arch_controlador;
138

```

### c) Medidor de distância

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity interface_hcsr04 is
6   port(
7     i_clock, i_reset : in std_logic;
8     i_medir, i_echo: in std_logic;
9     o_trigger, o_pronto: out std_logic;
10    o_medida: out std_logic_vector (15 downto 0);
11    db_medir: out std_logic;
12    db_estado: out std_logic_vector (2 downto 0)
13  );
14 end entity;
15
16 architecture arch_interface_hcsr04 of interface_hcsr04 is
17   signal s_acionar_trigger, s_pronto_trigger, s_medir_echo, s_pronto_medidor: std_logic;
18
19 component interface_sensor_de_distancia_UC is
20   port(
21     i_clock, i_reset,
22     medir, pronto_trigger, pronto_medidor, echo: in std_logic;
23     acionar_trigger, medir_echo, fim: out std_logic;
24     db_estado: out std_logic_vector(2 downto 0)
25   );
26 end component;
27
28
29
30 component gerador_pulso is
31   generic (
32     largura: integer:= 25
33   );
34   port(
35     clock, reset: in std_logic;
36     gera, para: in std_logic;
37     pulso, pronto: out std_logic
38   );
39 end component;
40
41 component medidor_de_distancia is
42   port(
43     i_clock, i_reset : in std_logic;
44     i_echo: in std_logic;
45     o_pronto: out std_logic;
46     o_medida: out std_logic_vector (15 downto 0)
47   );
48 end component;
49
50 begin
51
52   GERADOR_DE_TRIGGER: gerador_pulso generic map (largura => 500) port map(
53     i_clock, i_reset,
54     s_acionar_trigger, '0',
55     o_trigger, s_pronto_trigger
56   );
57
58   UC: interface_sensor_de_distancia_UC port map(
59     i_clock, i_reset,
60     s_pronto_trigger, s_pronto_medidor, i_echo,
61     s_acionar_trigger, s_medir_echo, o_pronto,
62     db_estado
63   );
64   MEDIDOR: medidor_de_distancia port map(
65     i_clock, s_acionar_trigger,
66     i_echo,
67     s_pronto_medidor,
68     o_medida
69   );
70
71   db_medir <= i_medir;
72
73 end arch_interface_hcsr04;

```

#### d) Servomotor

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity controle_servo is
6  port (
7      i_clock, i_reset: in std_logic;
8      i_posicao : in std_logic;
9      o_pwm: out std_logic;
10     db_reset: out std_logic;
11     db_pwm: out std_logic;
12     db_posicao: out std_logic
13 );
14 end controle_servo;
15
16 architecture rtl of controle_servo is
17
18     constant CONTAGEM_MAXIMA : integer := 1000000; -- valor para frequencia da saida de 4KHz
19
20     signal s_contagem : integer range 0 to CONTAGEM_MAXIMA-1;
21     signal s_largura_pwm : integer range 0 to CONTAGEM_MAXIMA-1;
22     signal s_posicao : integer range 0 to CONTAGEM_MAXIMA-1;
23
24 begin
25
26     process(i_clock,i_reset,i_posicao, s_posicao)
27     begin
28         -- inicia contagem e largura
29         if(i_reset='1') then
30             s_contagem    <= 0;
31             o_pwm        <= '0';
32             db_pwm       <= '0';
33             s_largura_pwm <= s_posicao;
34         elsif(rising_edge(i_clock)) then
35             -- saida
36             if(s_contagem < s_largura_pwm) then
37                 o_pwm  <= '1';
38                 db_pwm <= '1';
39             else
40                 o_pwm  <= '0';
41                 db_pwm <= '0';
42             end if;
43             -- atualiza contagem e largura
44             if(s_contagem = CONTAGEM_MAXIMA-1) then
45                 s_contagem <= 0;
46                 s_largura_pwm <= s_posicao;
47             else
48                 s_contagem  <= s_contagem + 1;
49             end if;
50         end if;
51     end process;
52
53     process(i_posicao)
54     begin
55         case i_posicao is
56         when "001" => s_largura <= 50000; -- pulso de 1 ms
57         when "010" => s_largura <= 60000; -- pulso de 1,2 ms
58         when "011" => s_largura <= 70000; -- pulso de 1,4 ms
59         when "101" => s_largura <= 80000; -- pulso de 1,6 ms
60         when "110" => s_largura <= 90000; -- pulso de 1,8 ms
61         when "111" => s_largura <= 100000; -- pulso de 2 ms
62         when '1' => s_posicao <= 50000; -- pulso de 1 ms 30° ? nao lembro valor exato
63         when others => s_posicao <= 0; -- nulo saida 0
64     end case;
65 end process;
66
67     db_reset <= i_reset;
68     db_posicao <= i_posicao;
69
70 end rtl;
71

```

e) Interface do circuito modificada para apresentação

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity controlador_comodos_para_filhotes is
6   port (
7     i_clock, i_reset: in std_logic;
8     i_ligado, i_modo, i_portax, i_portay: in std_logic;
9     o_porta0, o_porta1, o_portax, o_portay: out std_logic; -- porta em 0 é aberta em 1 é fechada
10    i_filhote: in std_logic;
11
12    i_echo: in std_logic;
13    o_pwm, o_pwmx, o_trigger: out std_logic;
14    db_medida: out std_logic_vector(15 downto 0);
15    db_reset: out std_logic;
16    db_proximo: out std_logic;
17    db_pwmx: out std_logic;
18    db_pwm: out std_logic;
19
20    o_display0: out std_logic_vector(6 downto 0);
21    o_display1: out std_logic_vector(6 downto 0);
22    o_display2: out std_logic_vector(6 downto 0)
23  );
24 end entity;
25
26 architecture arch_controlador of controlador_comodos_para_filhotes is
27
28 signal s_esta_proximo, s_porta0, s_porta1, s_portax, s_portay, s_medir_distancia, s_pwm, s_pronto_HCSR04, s_pwmx: ;
29 signal s_medida_distancia_em_BCD: std_logic_vector (15 downto 0);
30
31
32
33
34 component controlador_comodos_para_filhotes_uc is
35   port (
36     i_clock, i_reset: in std_logic;
37     i_ligado, i_modo, i_portax, i_portay, i_esta_proximo: in std_logic;
38     i_pronto_HCSR04: in std_logic;
39     o_porta0, o_portax, o_portay: out std_logic; -- porta em 0 é aberta em 1 é fechada
40     o_medir_distancia: out std_logic;
41     db_estado: out std_logic_vector(1 downto 0)
42   );
43 end component;
44
45
46 component controle_servo is
47   port (
48     i_clock, i_reset: in std_logic;
49     i_posicao: in std_logic;
50     o_pwm: out std_logic;
51     db_reset: out std_logic;
52     db_pwm: out std_logic;
53     db_posicao: out std_logic
54   );
55 end component;
56
57 component interface_hcsr04 is
58   port
59   (
60     i_clock, i_reset : in std_logic;
61     i_medir, i_echo: in std_logic;
62     o_trigger, o_pronto: out std_logic;
63     o_medida: out std_logic_vector (15 downto 0);
64     db_medir: out std_logic;
65     db_estado: out std_logic_vector (2 downto 0)
66   );
67 end component;
68
69 component comparador_menor_30cm is -- usei 30cm como exemplo apenas, a ideia é modificar o componente para o valor ex:
70   port (
71     i_clock, i_reset: in std_logic;
72     i_medida: in std_logic_vector (15 downto 0);
73     o_entrada_en_menor_que_30cm: out std_logic
74   );
75 end component;
76
77
78 component hexa7seg is
79   port(
80     hexa : in std_logic_vector(3 downto 0);
81     sseg : out std_logic_vector(6 downto 0)
82   );
83 end component;
84
85
86 begin
87

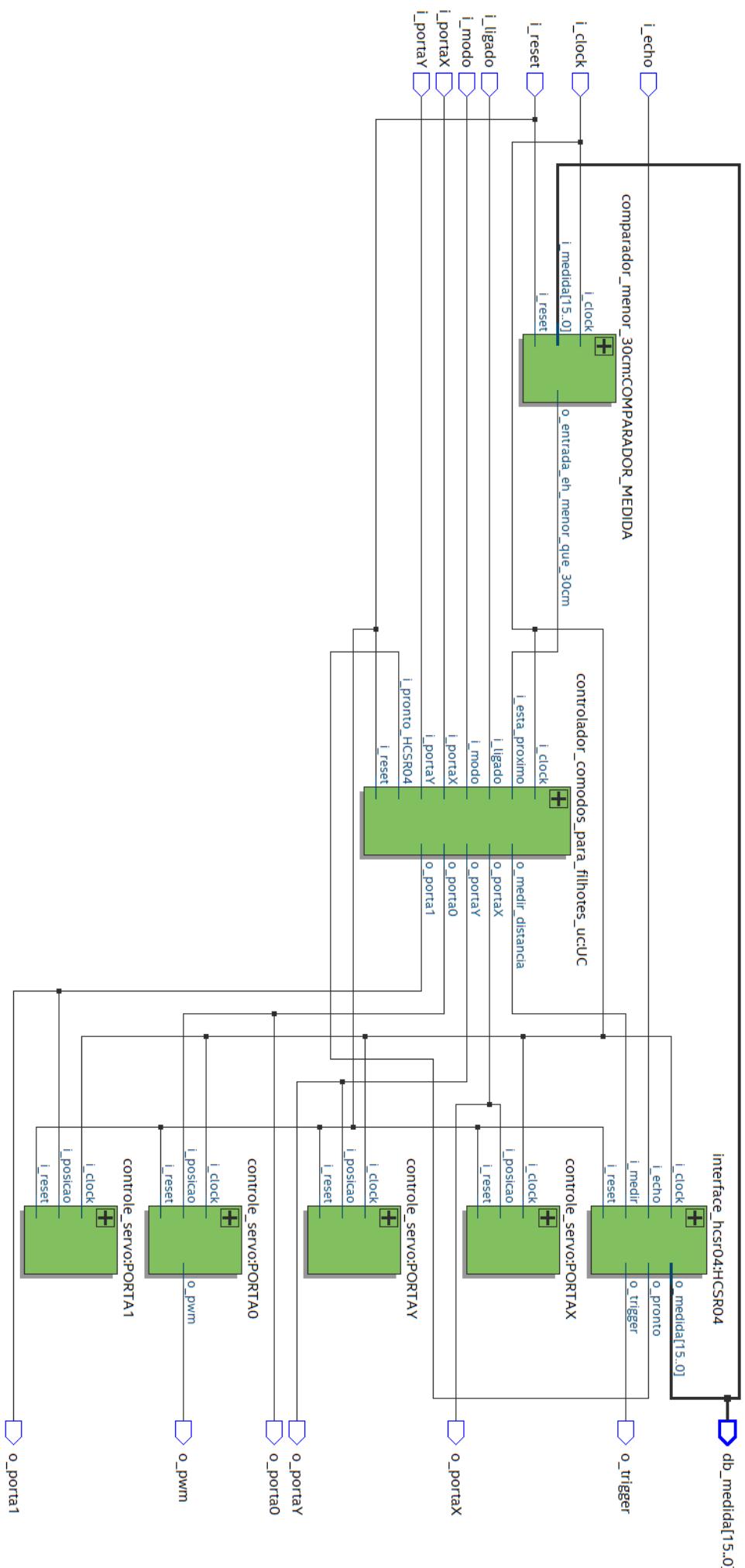
```

```

88  UC: controlador_comodos_para_filhotes_uc port map (
89    i_clock, i_reset,
90    i_ligado, i_modo, i_portax, i_portay, s_esta_proximo,
91    s_pronto_HCSR04,
92    s_porta0, s_porta1, s_portax, s_portay,
93    s_medir_distancia,
94    open
95  );
96
97  PORTA0: controle_servo port map (
98    i_clock, i_reset,
99    s_porta0,
100   s_pwm,
101   open,
102   open,
103   open
104  );
105
106  PORTA1: controle_servo port map ( -- essa e as seguintes foram colocadas sem ter uma saida pwm pois por enquanto
107    i_clock, i_reset,
108    s_porta1,
109    open,
110    open,
111    open,
112    open
113  );
114
115  PORTAX: controle_servo port map (
116    i_clock, i_reset,
117
118    s_portax,
119    s_pwmx,
120    open,
121    open,
122    open
123  );
124
125  PORTAY: controle_servo port map (
126    i_clock, i_reset,
127    s_portay,
128    open,
129    open,
130    open,
131    open
132  );
133
134  --HCSR04: interface_hcsr04 port map (
135    i_clock, j_reset,
136    s_medir_distancia, i_echo,
137    o_trigger, s_pronto_HCSR04,
138    s_medida_distancia_em_BCD,
139    open,
140    open
141  );
142
143  COMPARADOR_MEDIDA: comparador_menor_30cm port map (
144    i_clock, i_reset,
145    s_medida_distancia_em_BCD,
146
147    s_esta_proximo
148  );
149
150  with i_filhote select
151    s_medida_distancia_em_BCD <= "0001000000000000" when '1', --100cm
152    "00000000001000" when '0', --0,1 cm
153    "00000000000000" when others;
154
155  o_porta0 <= s_porta0;
156  o_porta1 <= s_porta1;
157  o_portax <= s_portax;
158  o_portay <= s_portay;
159  db_reset <= i_reset;
160  db_proximo <= s_esta_proximo;
161  db_medida <= s_medida_distancia_em_BCD;
162
163  db_pwmx <= s_pwmx;
164  o_pwmx <= s_pwmx;
165  o_pwm <= s_pwm;
166  db_pwm <= s_pwm;
167
168  HEX0: hexa7seg port map (
169    s_medida_distancia_em_BCD(7 downto 4),
170    o_display0
171  );
172
173  HEX1: hexa7seg port map (
174    s_medida_distancia_em_BCD(11 downto 8),
175    o_display1
176
177  HEX2: hexa7seg port map (
178    s_medida_distancia_em_BCD(15 downto 12),
179    o_display2
180  );
181
182
183 end arch_controlador;
184

```

**RTL**



# Scripts

## 1) Teste



```
!pip install requests
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)

[ ] import requests

[ ] auth_token1 = "u_EjRouldtNm2_eRYgvAK3JbusMiDuUq"
blynk_server = '45.55.96.146'

[ ] pin = 'V10'

[ ] rbutton = requests.get('http://'+blynk_server+'/'+auth_token1+'/get/'+pin)
print(rbutton.text)

["0"]

[ ] print(rbutton.text[2])
print(auth_token1)
pinled = 'V30'
if(rbutton.text[2] == '1'):
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=255')
else:
    r = requests.get('http://'+blynk_server+'/'+auth_token1+'/update/'+pinled+'?value=0')

0
u_EjRouldtNm2_eRYgvAK3JbusMiDuUq
```

## 2) Feito na aula do sprint 3

```
[ ] !pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)

[ ] import requests
import time

[ ] auth_tokenSane = "u_EjRou1dtNm2_eRYgvAK3JbusMiDuUq"
auth_tokenGerি = "YHL9bspuaEK2Tj8IR0nBSyqZ8wNYcuu5"
blynk_server = '45.55.96.146'

[ ] pin = 'V10'
pin2 = 'V11'
pinled = 'V31'
pinlid2 = 'V32'

[ ] while 1:
    rbutton = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin)
    rbutton2 = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/get/'+pin2)

    print(rbutton.text)
    print(rbutton.text[2])
    print(auth_token1)
    if(rbutton.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGerি+'/update/'+pinled+'?value=255')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGerি+'/update/'+pinled+'?value=0')

    if(rbutton2.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGerি+'/update/'+pinlid2+'?value=255')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenGerি+'/update/'+pinlid2+'?value=0')
```

### 3) Projeto

```
[ ] !pip install requests

Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (2.23.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests) (2020.6.20)

[ ] import requests
import time

[ ] auth_tokenSane = "u_EjRou1dtNm2_eRYgvAK3JbusMiDuUq"
auth_tokenGerি = "YHL9bspuaEK2Tj8IR0nBSyqZ8wNYcuu5"
blynk_server = '45.55.96.146'

[ ] slider = 'V10'
gauge = 'V34'

[ ] while 1:
    slider = requests.get('http://'+blynk_server+'/'+auth_tokenGerি+'/get/'+slider)

    if(slider.text[2] == '1'):
        r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=1')
    else:
        r = requests.get('http://'+blynk_server+'/'+auth_tokenSane+'/update/'+gauge+'?value=0')
```