

TAMAGOTCHI

Implementação do jogo em Verilog para utilização em um FPGA

Pedro

Nico

André Schueda

- **O que é um Tamagotchi?**

- Dispositivos e ferramentas
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

- **O que é um Tamagotchi?**

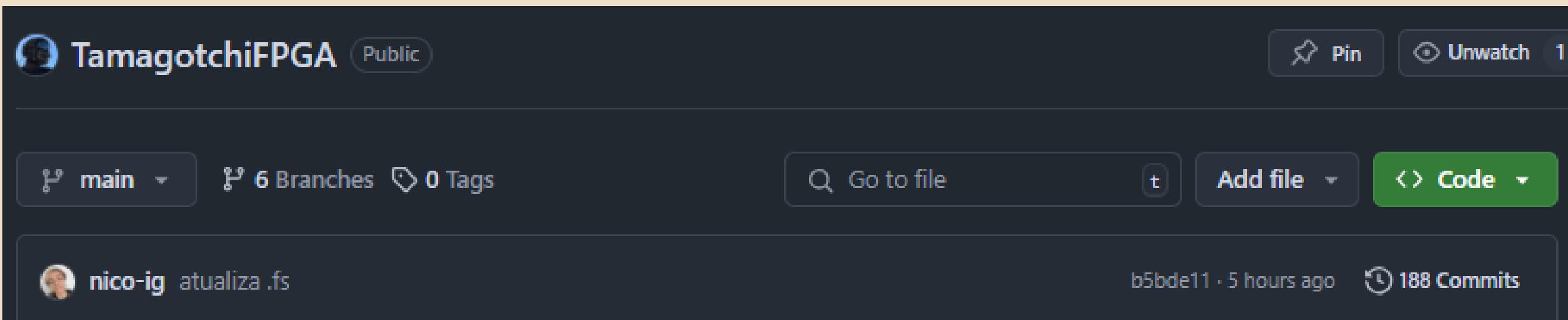
- Dispositivos e ferramentas
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final



- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

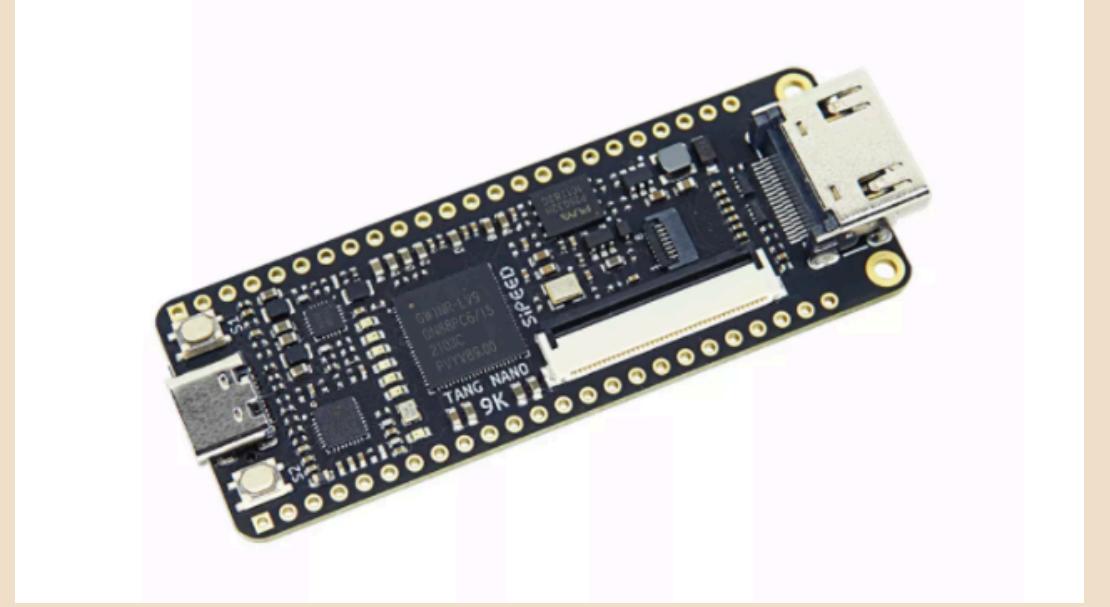
- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

- **Github**
- **Verilog**
- FPGA (*Tang Nano 9K*)
- Display (SSD1306)
- Protoboard
- Gowin
- Lushay Labs



- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

- Github
- Verilog
- **FPGA (Tang Nano 9K)**
- Display (SSD1306)
- Protoboard
- Gowin
- Lushay Labs



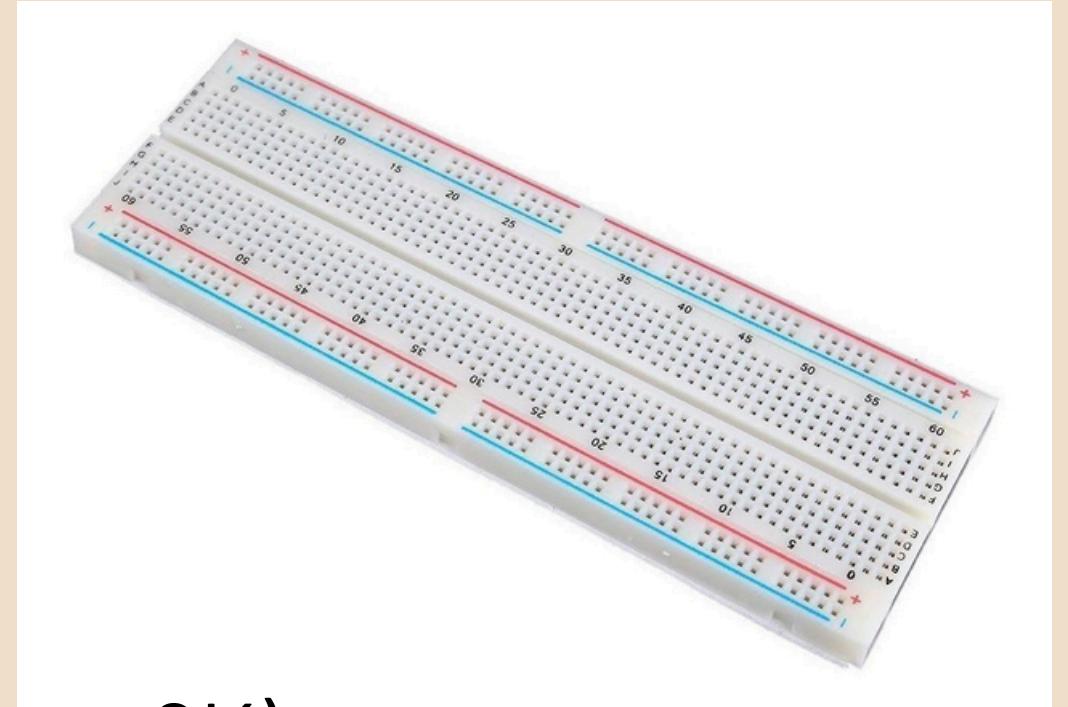
- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

- Github
- Verilog
- FPGA (Tang Nan)
- **Display (SSD1306)**
- Protoboard
- Gowin
- Lushay Labs



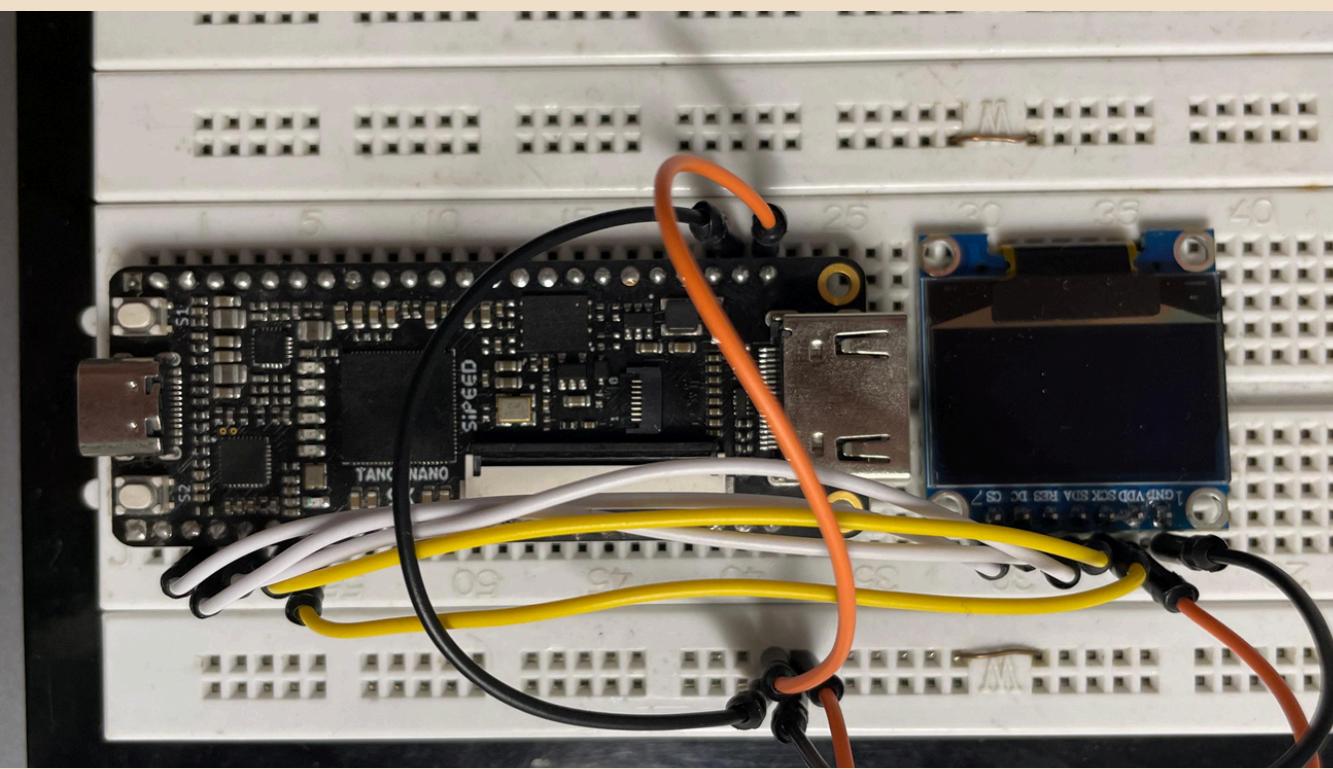
- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

- Github
- Verilog
- FPGA (Tang Nano 9K)
- Display (SSD1306)
- **Protoboard**
- Gowin
- Lushay Labs



- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final

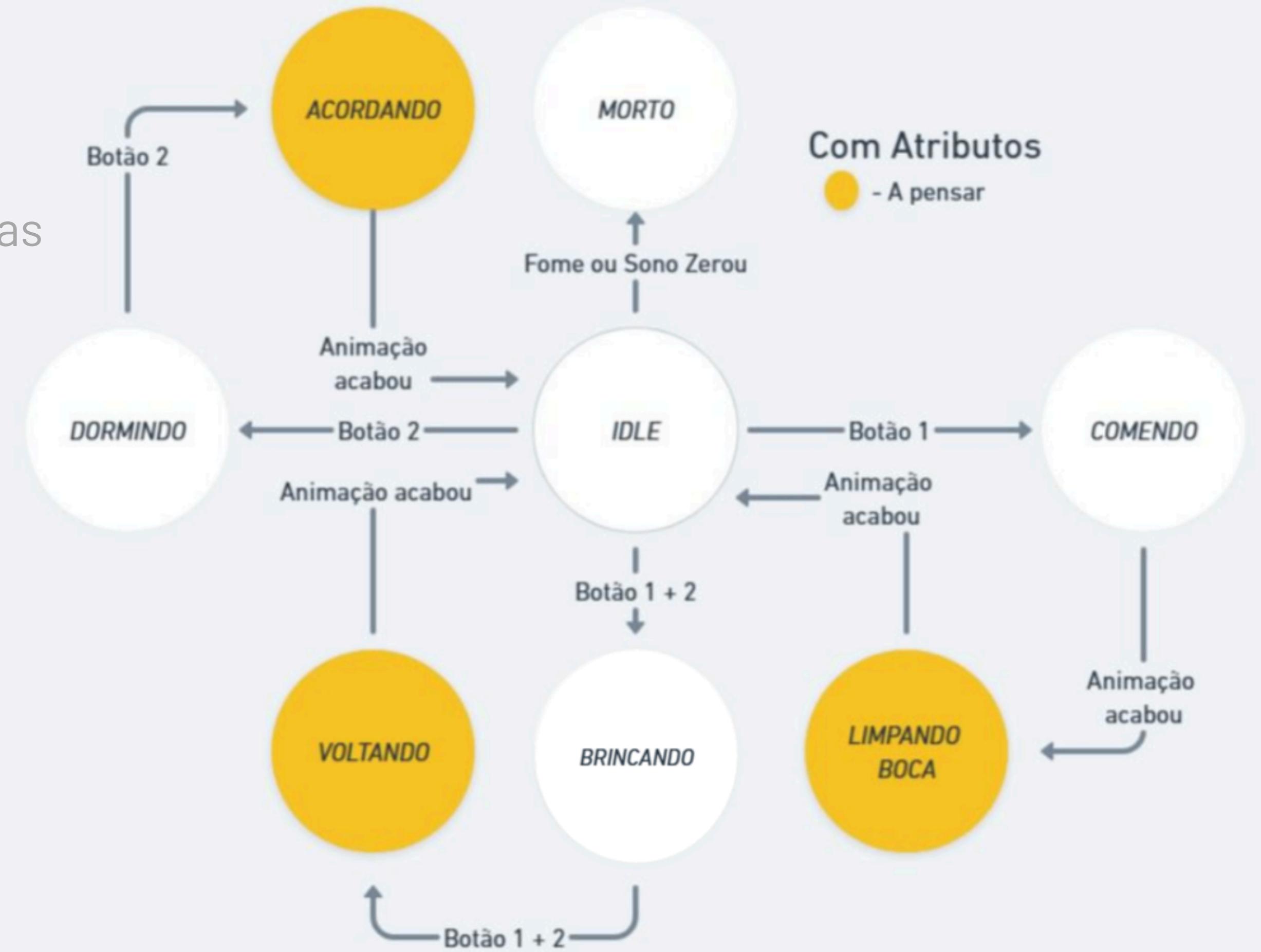
- Github
- Verilog
- **FPGA (Tang Nano 9K)**
- **Display (SSD1306)**
- **Protoboard**
- Gowin
- Lushay Labs



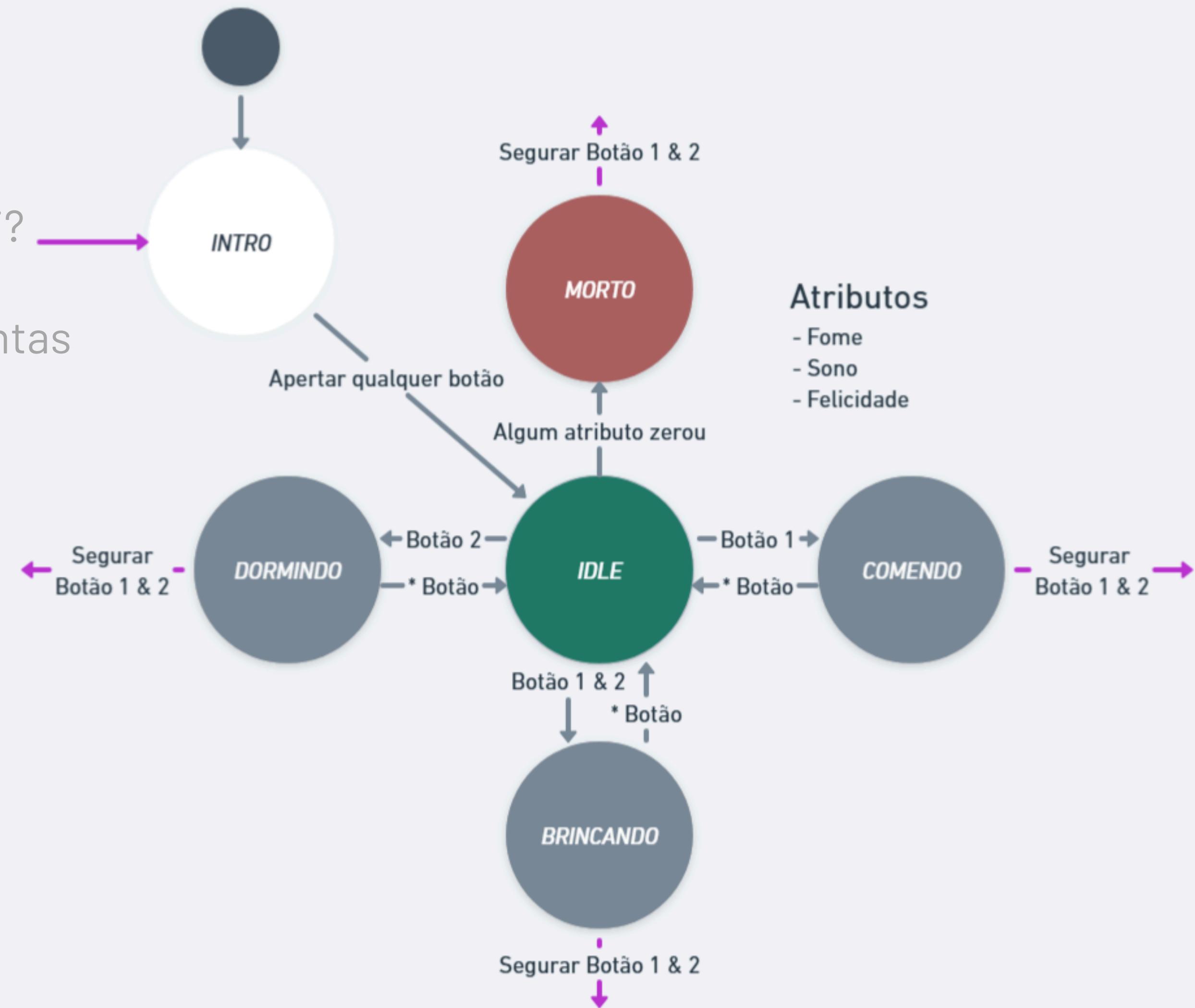
- O que é um Tamagotchi?
- **Dispositivos e ferramentas**
- Máquina de Estados
- Os módulos
- Desafios enfrentados
- Resultado final
- Github
- Verilog
- FPGA (Tang Nano 9K)
- Display (SSD1306)
- Protoboard
- **Gowin**
- **Lushay Labs**

- O que é um Tamagotchi?
- Dispositivos e ferramentas
- **Máquina de Estados**
- Os módulos
- Desafios enfrentados
- Resultado final

- O que é um Tamagotchi?
- Dispositivos e ferramentas
- **Máquina de Estados**
- Os módulos
- Desafios enfrentados
- Resultado final

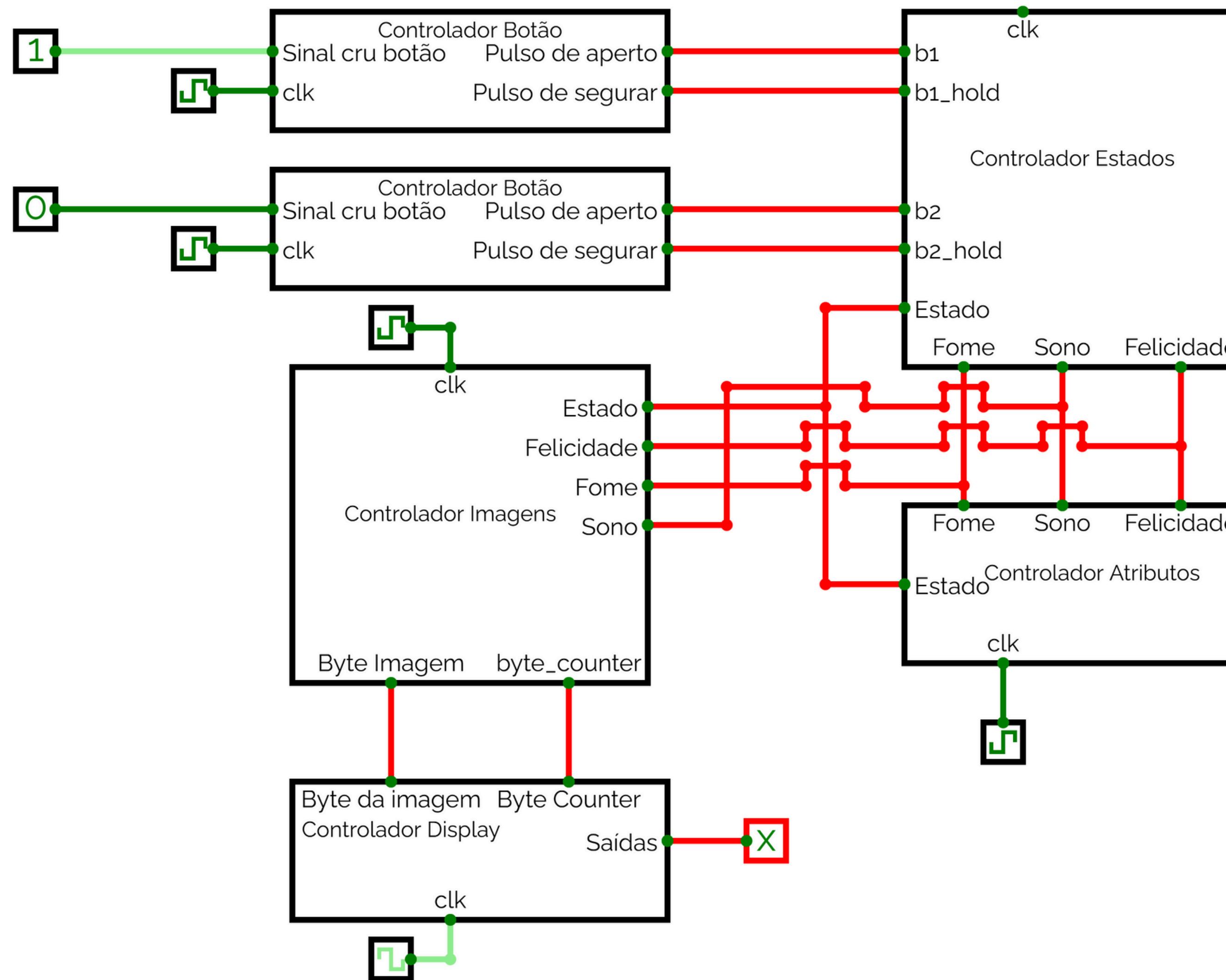


- O que é um Tamagotchi?
- Dispositivos e ferramentas
- **Máquina de Estados**
- Os módulos
- Desafios enfrentados
- Resultado final



- O que é um Tamagotchi?
- Dispositivos e ferramentas
- Máquina de Estados
- **Os módulos**
- Desafios enfrentados
- Resultado final

- O que é um Tamagotchi?
- Dispositivos e ferramentas
- Máquina de Estados
- **Os módulos**
- Desafios enfrentados
- Resultado final
- Tamagotchi
- Controlador Estados
- Controlador Botão
- Controlador Atributos
- Controlador Imagens
- Controlador Display
- Gerador Aleatório



Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

```
// Instancia o módulo de controle dos botões
controlador_botao B1
(
    .clk(clk),
    .b_in(b1),
    .b_out(b1_aux),
    .b_hold_out(b1_hold)
);

controlador_botao B2
(
    .clk(clk),
    .b_in(b2),
    .b_out(b2_aux),
    .b_hold_out(b2_hold)
);
```

Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

```
// Instancia o módulo de controle de estados
controlador_estados EST
(
    .clk(clk),
    .b1(b1_aux),
    .b2(b2_aux),
    .b1_hold(b1_hold),
    .b2_hold(b2_hold),
    .fome(fome),
    .felicidade(felicidade),
    .sono(sono),
    .estado(estado)
);
```

Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

```
// Instancia o módulo de controle de atributos
controlador_atributos ATR
(
    .clk(clk),
    .estado(estado),
    .fome(fome),
    .felicidade(felicidade),
    .sono(sono)
);
```

Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

```
// Instancia o módulo de controle das imagens para o display
controlador_imagens IMG
(
    .clk(clk),
    .estado(estado),
    .felicidade(felicidade),
    .fome(fome),
    .sono(sono),
    .byte_counter(byte_counter),
    .data_to_send(data_to_send)
);
```

Tamagotchi

Instancia todos os módulos

```
module tamagotchi
(
    input wire b1, b2, clk,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
    wire[4:0] estado;
    wire[7:0] fome, felicidade, sono;
    wire morreu, b1_aux, b2_aux, b1_hold, b2_hold;
    wire[7:0] data_to_send;
    wire[9:0] byte_counter;
```

Inputs: Botões e clock

Outputs: Saída do Display

```
// Instancia o módulo do display
controlador_display DIS
(
    .clk(clk),
    .data_to_send(data_to_send),
    .byte_counter(byte_counter),
    .io_sclk(io_sclk),
    .io_sdin(io_sdin),
    .io_cs(io_cs),
    .io_dc(io_dc),
    .io_reset(io_reset)
);
```

Controlador Estados

Controla a mudança entre estados

```
module controlador_estados
(
    input wire b1, b2, b1_hold, b2_hold, clk,
    input wire [7:0] fome, felicidade, sono,
    output reg[4:0] estado
);
    // Estados possíveis
    localparam INTRO = 5'b00000,
              IDLE = 5'b00001,
              DORMINDO = 5'b00010,
              COMENDO = 5'b00100,
              BRINCANDO = 5'b01000,
              MORTO = 5'b10000;
```

Inputs: Botões, Atributos e clock

Outputs: Estado

Controlador Estados

Controla a mudança entre estados

```
module controlador_estados
(
    input wire b1, b2, b1_hold, b2_hold, clk,
    input wire [7:0] fome, felicidade, sono,
    output reg[4:0] estado
);
    // Estados possíveis
    localparam INTRO = 5'b00000,
              IDLE = 5'b00001,
              DORMINDO = 5'b00010,
              COMENDO = 5'b00100,
              BRINCANDO = 5'b01000,
              MORTO = 5'b10000;
```

Inputs: Botões, Atributos e clock

Outputs: Estado

```
if (estado === MORTO || !fome || !sono || !felicidade)
    estado <= MORTO;

else if (estado === IDLE)
begin
    estado <= b1_reg && !b2_reg ? COMENDO :
                                !b1_reg && b2_reg ? DORMINDO :
                                b1_reg && b2_reg ? BRINCANDO : IDLE;
end
else
    estado <= b1_reg || b2_reg ? IDLE : estado;
```

Controlador Botões

Trata o clique de um botão

```
module controlador_botao
(
    input b_in, clk,
    output reg b_out,
    output wire b_hold_out
);
```

Inputs: Sinal cru botão e clock

Outputs: Pulso de segurar e pulso de apertar

Controlador Botões

Trata o clique de um botão

```
module controlador_botao
(
    input b_in, clk,
    output reg b_out,
    output wire b_hold_out
);
```

```
// O botão está pressionado e o pulso ainda não foi gerado
if (b_in === 0 && b_out === 0 && !dirty)
begin
    // Botão estavel, gera o pulso e ativa a flag,
    // não gera outro pulso até que o botão seja solto
    if (counter === 4'hF)
begin
    b_out <= 1;
    dirty <= 1;
end
    counter <= counter + 4'b1;
// O botão está solto e um pulso foi gerado
end else if (b_in === 1 && dirty)
begin
    // Botão realmente está solto, desativa
    // a flag e permite que um novo pulso seja gerado
    if (counter === 4'hF)
        dirty <= 0;
    counter <= counter + 4'b1;

end else
    b_out <= 0;
```

Controlador Botões

Trata o clique de um botão

```
module controlador_botao
(
    input b_in, clk,
    output reg b_out,
    output wire b_hold_out
);
```

Basicamente trata os pulsos (saídas) que o botão gera e cuida para que reflita a realidade

```
// O botão está pressionado e o pulso ainda não foi gerado
if (b_in === 0 && b_out === 0 && !dirty)
begin
    // Botão estavel, gera o pulso e ativa a flag,
    // não gera outro pulso até que o botão seja solto
    if (counter === 4'hF)
begin
    b_out <= 1;
    dirty <= 1;
end
    counter <= counter + 4'b1;
// O botão está solto e um pulso foi gerado
end else if (b_in === 1 && dirty)
begin
    // Botão realmente está solto, desativa
    // a flag e permite que um novo pulso seja gerado
    if (counter === 4'hF)
        dirty <= 0;
    counter <= counter + 4'b1;
end else
    b_out <= 0;
```

Controlador Atributos

Decrementa e incrementa atributos baseado no estado

```
module controlador_atributos
(
    input wire clk,
    input wire [4:0] estado,
    output reg [7:0] fome, felicidade, sono
);
```

Inputs: Estado e clock

Outputs: Atributos

Controlador Atributos

Decrementa e incrementa atributos baseado no estado

```
module controlador_atributos
(
    input wire clk,
    input wire [4:0] estado,
    output reg [7:0] fome, felicidade, sono
);
```

Inputs: Estado e clock

Outputs: Atributos

```
INTRO, MORTO;
begin
    sono <= INIT_SONO;
    fome <= INIT_FOME;
    felicidade <= INIT_FELICIDADE;
end
```

Controlador Atributos

Decrementa e incrementa atributos baseado no estado

```
module controlador_atributos
(
    input wire clk,
    input wire [4:0] estado,
    output reg [7:0] fome, felicidade, sono
);
```

```
INTRO, MORTO:
begin
    sono <= INIT_SONO;
    fome <= INIT_FOME;
    felicidade <= INIT_FELICIDADE;
end
```

Inputs: Estado e clock

Outputs: Atributos

```
DORMINDO:
begin
    sono <= sono < MAX_SONO - VEL_SUBIDA ?
        | sono + VEL_SUBIDA :
        | MAX_SONO;

    fome <= fome > VEL_DESCIDA ?
        | fome - VEL_DESCIDA :
        | 8'b0;

    felicidade <= felicidade > VEL_DESCIDA ?
        | felicidade - VEL_DESCIDA :
        | 8'b0;
end
```

Controlador Imagens

Decide qual sequência de imagens mandar para o display baseado no estado.

```
module controlador_imagens
(
    input wire clk,
    input wire [9:0] byte_counter,
    input wire [7:0] felicidade, fome, sono,
    input wire [4:0] estado,
    output reg [7:0] data_to_send
);
```

Inputs: Estado, atributos e clock

Outputs: Byte da imagem

Controlador Imagens

Decide qual sequência de imagens mandar para o display baseado no estado.

```
module controlador_imagens
(
    input wire clk,
    input wire [9:0] byte_counter,
    input wire [7:0] felicidade, fome, sono,
    input wire [4:0] estado,
    output reg [7:0] data_to_send
);
```

Atributos são escritos preenchendo posições específicas da imagem baseadas no seu valor!

Inputs: Estado, atributos e clock

Outputs: Byte da imagem



Controlador Imagens

Decide qual sequência de imagens mandar para o display baseado no estado.

Animações são inicialmente carregadas
em memórias específicas por imagem

```
// Inicializa a memória
initial
begin

    $readmemh("hexs/Intro/tamagotchi_intro1.hex", memoria_intro_0);

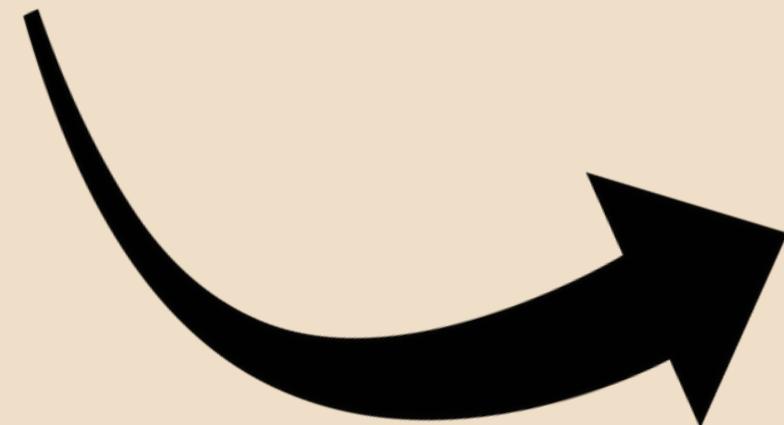
    $readmemh("hexs/Idle/tamagotchi_idle1.hex", memoria_idle_0);
    $readmemh("hexs/Idle/tamagotchi_idle2.hex", memoria_idle_1);
    $readmemh("hexs/Idle/tamagotchi_idle3.hex", memoria_idle_2);
```



Controlador Imagens

Decide qual sequência de imagens mandar para o display baseado no estado.

Animações são inicialmente carregadas
em memórias específicas por imagem
E escolhidas baseado no estado atual



```
// Inicializa a memória
initial
begin

    $readmemh("hexs/Intro/tamagotchi_intro1.hex", memoria_intro_0);

    $readmemh("hexs/Idle/tamagotchi_idle1.hex", memoria_idle_0);
    $readmemh("hexs/Idle/tamagotchi_idle2.hex", memoria_idle_1);
    $readmemh("hexs/Idle/tamagotchi_idle3.hex", memoria_idle_2);

case (estado)
INTRO:
    data_to_send <= memoria_intro_0[byte_counter_intro];
IDLE:
begin
    case (i_idle)
3'd0: data_to_send <= memoria_idle_0[byte_counter_idle];
3'd1: data_to_send <= memoria_idle_1[byte_counter_idle];
3'd2: data_to_send <= memoria_idle_2[byte_counter_idle];
```

Controlador Display

Efetivamente preenche os campos do Display com os valores recebidos.

```
module controlador_display
#(
    parameter STARTUP_WAIT = 32'd10000000
)
(
    input wire clk,
    input wire [7:0] data_to_send,
    output reg [9:0] byte_counter,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
```

Inputs: Byte para escrita e clock

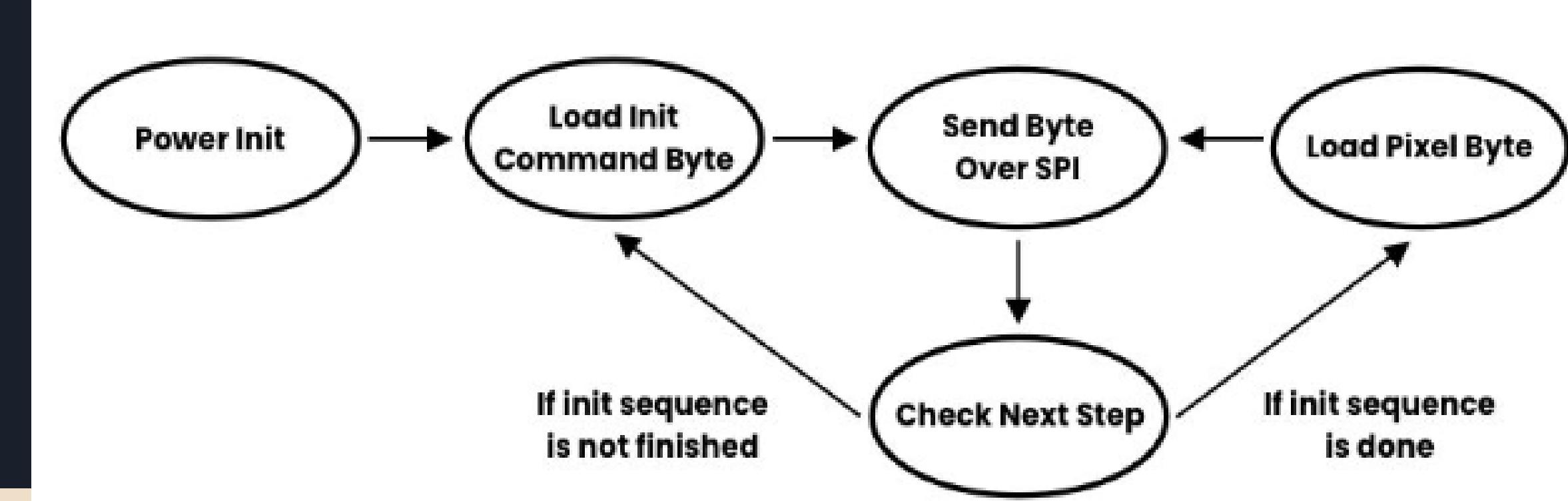
Outputs: Saídas do Display

Controlador Display

Efetivamente preenche os campos do Display com os valores recebidos.

```
module controlador_display
#(
    parameter STARTUP_WAIT = 32'd10000000
)
(
    input wire clk,
    input wire [7:0] data_to_send,
    output reg [9:0] byte_counter,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
```

Inputs: Byte para escrita e clock
Outputs: Saídas do Display



Máquina de estados que define o funcionamento do driver do display

Controlador Display

Efetivamente preenche os campos do Display com os valores recebidos.

```
module controlador_display
#(
    parameter STARTUP_WAIT = 32'd10000000
)
(
    input wire clk,
    input wire [7:0] data_to_send,
    output reg [9:0] byte_counter,
    output wire io_sclk,
    output wire io_sdin,
    output wire io_cs,
    output wire io_dc,
    output wire io_reset
);
```

Inputs: Byte para escrita e clock
Outputs: Saídas do Display

Código do estado que realiza atualização da imagem

```
STATE_LOAD_DATA: begin
    cs <= 1'd0;
    dc <= 1'd1;
    bitNumber <= 3'd7;
    dataToSend <= data_to_send;
    state <= STATE_SEND;
    byte_counter <= byte_counter + 10'd1;
end
```

Gerador Aleatório

Gera números pseudo-aleatórios para que algumas animações (os números são os índices delas em suas memórias) fiquem mais naturais.

```
module gerador_aleatorio_32 (
    input wire clk,
    output reg [31:0] data
);
```

Inputs: Dados para aleatorizar

Outputs: Dados aleatorizados

Gerador Aleatório

Gera números pseudo-aleatórios para que algumas animações (os números são os índices delas em suas memórias) fiquem mais naturais.

```
module gerador_aleatorio_32 (
    input wire clk,
    output reg [31:0] data
);
```

Inputs: Dados para aleatorizar

Outputs: Dados aleatorizados

```
wire feedback;

// Feedback calculation using the taps for a 32-bit LFSR
assign feedback = data[31] ^ data[21] ^ data[1] ^ data[0];

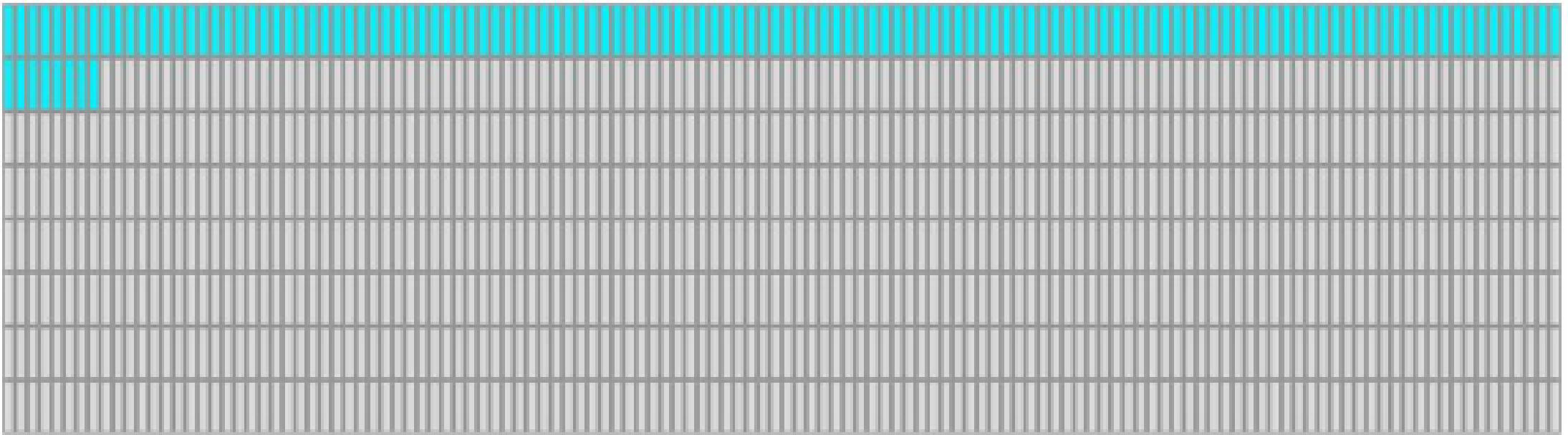
initial begin
    data = 32'h13; // Seed
end

always @(posedge clk) begin
begin
    data <= {data[30:0], feedback};
end
end
```

- O que é um Tamagotchi?
- Dispositivos e ferramentas
- Máquina de Estados
- Os módulos
- **Desafios enfrentados**
- Resultado final

- O que é um Tamagotchi?
 - Dispositivos e ferramentas
 - Máquina de Estados
 - Os módulos
- **Desafios enfrentados**
- Resultado final
- BOTÕES

- O que é um Tamagotchi?
 - Dispositivos e ferramentas
 - Máquina de Estados
 - Os módulos
 - **Desafios enfrentados**
 - Resultado final
- BOTÕES
 - Jeito que o Display funciona



128 bytes per row 8 rows, each byte controlling 8 vertical pixels

- O que é um Tamagotchi?
- Dispositivos e ferramentas
- Máquina de Estados
- Os módulos
- **Desafios enfrentados**
- Resultado final
- BOTÕES
- Jeito que o Display funciona

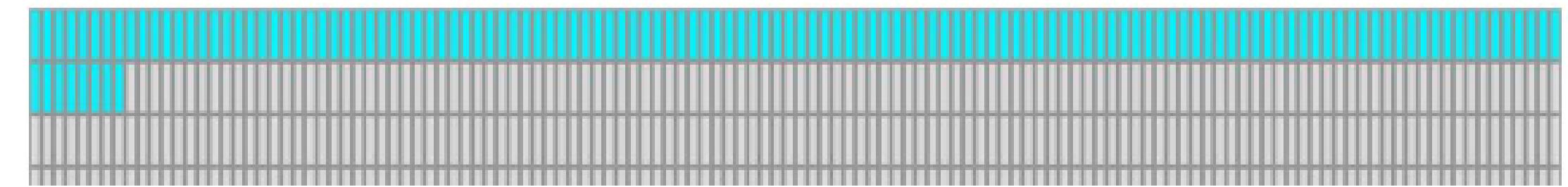


Figure 10-4 : Address Pointer Movement of Vertical addressing mode

	COL0	COL 1	COL 126	COL 127
PAGE0	↑				
PAGE1		↑			
:				
PAGE6					
PAGE7	↓	↓	↓	↓

- O que é um Tamagotchi?
 - Dispositivos e ferramentas
 - Máquina de Estados
 - Os módulos
-
- **Desafios enfrentados**
 - Resultado final
- BOTÕES
 - Jeito que o Display funciona
 - Animações e cálculos na indexação das memórias

Resource Usage Summary:

Resource	Usage	Utilization
Logic	794/8640	10%
--LUT,ALU,ROM16	776(661 LUT, 115 ALU, 0 ROM16)	-
--SSRAM(RAM16)	3	-
Register	252/6693	4%
--Logic Register as Latch	0/6480	0%
--Logic Register as FF	248/6480	4%
--I/O Register as Latch	0/213	0%
--I/O Register as FF	4/213	2%

Timing

Clock Summary:

Clock Name	Type	Period	Frequency(MHz)	Rise	Fall	Source	Master	Object
clk	Base	20.000	50.0	0.000	10.000			clk_ibuf/I

Max Frequency Summary:

No.	Clock Name	Constraint	Actual Fmax	Logic Level	Entity
1	clk	50.000(MHz)	61.716(MHz)	7	TOP

Detail Timing Paths Information

Path 1

Path Summary:

Slack	3.797
Data Arrival Time	16.529
Data Required Time	20.326
From	IMG/ i_morto_0_s6
To	IMG/ data_to_send_6_s0
Launch Clk	clk[R]
Latch Clk	clk[R]

... um rosto familiar?