INSTITUTO FEDERAL DE SANTA CATARINA - IFSC
CURSO DE ENGENHARIA DE TELECOMUNICAÇÕES

LEONAN DA SILVA SARAIVA

Laboratório Remoto em Eletrônicos Programáveis

BILBAO, 2015

LEONAN DA SILVA SARAIVA

Laboratório Remoto em Eletrônicos Programáveis

Relatório técnico apresentado como requisito para aprovação no Programa Propicie, Edital 22/2015.

Prof. Dr. IGNACIO ANGULO
Prof. Me. ARLIONES HOELLER JR

BILBAO, 2015

RESUMO

Este relatório apresenta atividades de pesquisa científica desenvolvidas de acordo com Programa de Intercâmbio Nacional e Internacional para Estudante do Instituto federal de santa Catarina do Brasil, denominado PROPICIE. O seguinte trabalho foi desenvolvido na Universidade de Deusto na Espanha.

Esse trabalho tem como objetivo relatar o estudo e desenvolvimento de um laboratório remoto utilizando a placa de desenvolvimento ZYBO (Zynq Board) da família Xilinx Zynq-7000. O laboratório remoto deve proporcionar a conexão de um elevado usando um (microprocessador (ARM) e FPGA (XILINX)) a plataforma para laboratórios remotos, denominada Weblab-Deusto, implementar e entender os diferentes estágios para o controle do elevador, todavia aferindo os resultados da solução escolhida.

Palavras-chave: Laboratório remoto, FPGA (XILINX), microprocessador (ARM), Algoritmo de Controle de um elevador.

SUMÁRIO

1 INTRODUÇÃO

No ensino de assuntos técnicos é de suma importância disponibilizar aos alunos a possibilidade de executar tarefas práticas, para que possa relacionar os conceitos teóricos aprendidos em sala de aula a situações da vida real.

Por esse motivo muitas instituições utilizam métodos de ensino que aproximem o máximo possível as tarefas práticas de situações reais que os engenheiros terão que enfrentar. Desta forma as universidades deveriam esta providas de equipamentos reais para seus alunos.

Um dos principais problemas apontado pelos estudantes é a disponibilidade de utilizar os laboratórios a qualquer momento, muitos estudantes trabalham durante o dia, moram longe da universidade, além de ser necessário um monitor para cuidar do laboratório.

Os laboratórios remotos são uma solução extremante eficaz para os problemas citados acima. Atualmente os alunos tem acesso a internet de qualquer lugar, principalmente com o advento dos *smartphones,* dessa forma é possível que trabalhem de qualquer lugar utilizando um laboratório virtual por meio da interface Web.

2 DESENVOLVIMENTO

1.1 OBJETIVO GERAL

A universidade de Deusto dispõem de um laboratório remoto chamado Weblab-Deusto, na qual se pode realizar diversos experimentos criados e disponibilizados pela equipe do laboratório.
Os experimentos ajudam no melhor entendimento da teoria aprendia em sala de aula, dessa forma o objetivo desse trabalho é criar um laboratório remoto para controle de um elevador e disponibiliza-lo aos alunos da universidade para que possa acessá-lo sem a necessidade de ser fazer presente no laboratório.

1.1.1 Objetivos específicos

O principal objetivo da pesquisa consiste no aprendizado da criação de um sistema utilizando software Vivado Design Suite e seu SDK para desenvolver o algoritmo de controle de um elevador na linguagem de programação C. Aprofundando seu conhecimento na programação de sistemas embarcados usando a placa de desenvolvimento ZYBO (Zynq Board) da família Xilinx Zynq-7000.

1.2 METODOLOGIA

1.2.1 Estudo da placa ZYBO.

Foi realizado o estudo da placa Zybo, identificando seus recursos disponíveis como periféricos, capacidade de processamento, tensão de alimentação, entre outros com o objetivo de identifica se iria ser necessário a aquisição de componentes extras para o projeto.

1.2.2 Estudo do software Vivado e SDK.

Foi realizado a execução de cinco tutoriais disponibilizados pela Xilinx. Os tutoriais abordam a criação de um projeto, criação de um IP, codificação, simulação, depuração e implementação.

1.2.3 Criação da arquitetura de sistema embarcado utilizando o Vivado.

Nessa etapa foi criado a arquitetura do projeto do elevador, adicionando os IPs necessário para comunicar-se com a maquete do elevador.

1.2.4 Estudo da maqueta do elevador.

Nessa etapa foi estudado o funcionamento da maquete do elevador referente as entradas e saídas, bem como tensões de operação.

1.2.5 Estudo do software EAGLE para criação de uma (PCB).

Nessa etapa foi estudado o software EAGLE e as etapas necessárias para criação de um PCB, desde a criação de um esquemático ao roteamento das trilhas.

1.2.6 Criação de uma (PCB).

Com os conhecimentos adquiridos com o EAGLE foi criado uma PCB para adaptar os conectores da placa Zybo com os conectores das placas de conversões de tensões da maquete do elevador.

1.2.7 Algoritmo de controle utilizando o SDK com a linguagem C.

Nessa etapa foi desenhada um diagrama de funcionamento do elevador e posteriormente a implementação do algoritmo na linguagem de programação C.

1.2.8 Análise dos resultados obtidos.

1.3 Cronograma de Atividades.

Tabela 1: Cronograma de Atividades Relativas as seções apresentadas na metodologia.

Seções/ Mês	Setembro	Outubro	Novembro	Dezembro
1.2.1	X			
1.2.2	X			
1.2.3		X		
1.2.4		X		
1.2.5		X		
1.2.6			X	
1.2.7			X	X
1.2.8				

1.4 PROCEDIMENTOS EXPERIMENTAIS

1.4.1 ZYBO Zynq™-7000 Development Board.

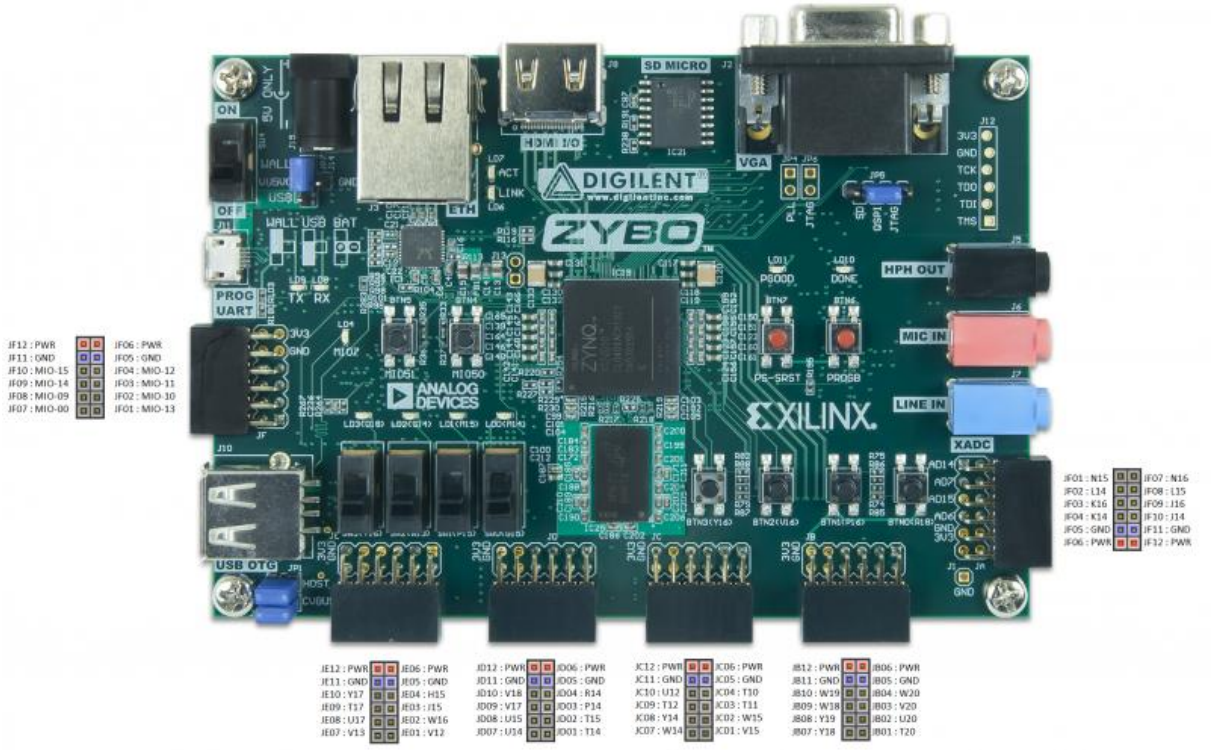


Figura 1 : Zybo Board

Para controlar a maquete do elevador foi usado a placa de desenvolvimento de circuitos digitais ZYBO (Zynq Board) da família Xilinx Zynq-7000, fabricada pela companhia Digilent Inc. Um dos principais motivos de sua escolha foi a de oferecer uma alternativa barata aos desenvolvedores sem perdem em desempenho, outro motivo foi o fato dela está sendo incorporada as disciplinas de relacionadas dispositivos lógicos programáveis e sistemas embarcados na universidade de Deusto.

1.4.1.1 Características da Zybo.

- 650 MHz Dual-core Cortex-A9 processador
- Controlador de memória DDR3 com 8 canais de DMA

- De alta largura de banda controladores periféricos: 1 G Ethernet, USB 2,0, SDIO
- Baixa largura de banda controlador periférica: SPI, UART, I 2 C
- Reprogramável lógico equivalente a Artix-7 FPGA
- 28 K lógica células
- 240KB bloco RAM
- 80 DSP fatias
- On-chip dual Channel, 12-bit, 1 Msps analógico-para-digital (xadc)

1.4.1.2 Portas e periféricos on-board:

- Zynq XC7Z010-1CLG400C.
- 512 MB x32 DDR3 W/1050 Mbps de largura de banda.
- Dual-função (fonte/sumidouro) porta HDMI.
- 16-bits por pixel porta de saída VGA.
- Trimode (1 Gbit/100 Mbit/10 Mbit) Ethernet PHY.
- Slot microSD (suporta Linux sistema de arquivos).
- OTG USB 2,0 PHY (suporta host e dispositivo).
- EEPROM externa (programado com 48-bit exclusivo global EUI-48/64 identificador™ compatível).
- Codec de áudio com fone de ouvido, microfone e LINE IN Jacks.
- 128 MB Serial Flash W/QSPI interface.
- On-board programação JTAG e UART para conversor USB.
- GPIO: 6 botões, 4 comutadores, 5 LEDs.
- Seis pmod conectores (1 processador dedicado, 1 dual analógico/digital).

1.4.2 Vivado Design Suite e SDK:

A ZYBO é compatível com o ambiente de desenvolvimento Vivado Design Suite da Xilinx, que prover uns conjuntos de ferramentas intuitivas para desenvolver IPs (intellectual property) na área da lógica programável e configurações na área de processamento (ARM).

Dentre as ferramentas então incluídas síntese, simulação, depuração temporal e funcional além do SDK para desenvolvimento em alto nível de programas a serem

executados na área de processamento e suporte à linguagem C e C++. Com o VIVADO pode-se projetar sistemas de qualquer complexidade, a partir de um sistema operacional completo é possível executar vários aplicativos de servidor em conjunto ou um simples programa que controla alguns Leds.

1.4.3 CadSoft EAGLE PCB Design Software

Se trata de um software para criar uma printed circuit board PCB com uma interface muito intuitiva, disponível para diversos sistemas operacionais como Windows, MAC e Linux. Está divido em três módulos:

1.4.3.1 Schematic editor

O editor esquemático permite que você crie uma representação simbólica fácil de ler do seu design. O objetivo do esquema é a fornecer a documentação sobre o projeto, permitindo que outros possam compreender facilmente sua intenção. EAGLE vem com milhares de peças pré-fabricados que simplificam as tarefas de captura de esquemáticos. Também possui funcionalidades de verificação ERC afim de validar suas conexões.

1.4.3.2 Layout editor

O editor de layout traz seu projeto a vida. O layout da placa representa a realidade física do seu Design. Você pode localizar e rotear seus componentes com precisão antes de fabricá-la.

1.4.3.3 Autorouter

O autorouter é uma ferramenta útil para ao roteamento automático entre os componentes oferecendo possibilidade de configuração afim de obter o menor custo, menor área, entre outros, também possibilita o roteamento manual quando necessário em aplicações mais sensíveis.

1.4.4 Elevador de três pisos.

Figura 2: Maquete do elevador de três pisos

A Figura 2 representa uma maquete educativa de um elevador de três pisos fabricado pela empresa Staudinger GMBH, equipado com todos os sensores e atuadores necessário para seu funcionamento aproximando o máximo de um elevador real.

Na figura do lado esquerdo temos um painel de controle semelhante a um painel interno de elevador comum que possui as seguintes características.

- Três portas correspondentes aos andares.
- Um botão de alarme.
- Um botão de parada de emergência.
- Seis Leds indicadores dos respectivos botões.

O lado direito da figura representa a cabine do elevador de três andares com as seguintes características:

- Três portas, uma para cada andar.
- Cinco botões, os mesmos encontrados em um elevador convencional.

- Cinco Leds na parte superior da cabine identificando o estado da mesma.
- Vinte e seis entradas digitais
- Vinte e quatro saídas digitais.
- Seis interruptores mecânicos ao longo dos andares que ajudam a identificar o estado da cabine.

1.4.4.1 Entradas e Saídas da maquete do elevador.

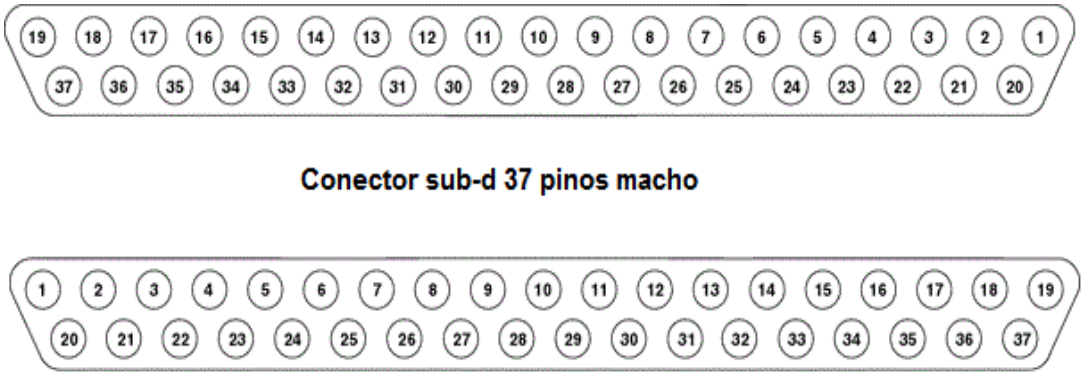


Figura 3: Pinos dos conectores sub-d 37 macho e fêmea.

Figura 4: Conectores sub-d 37 macho e fêmea.

O elevador possui um grande número de entradas e saídas que são acessadas por dois conectores do tipo D-Sub de 37 pinos como mostrado nas figuras

Figura 11 e Figura 12, um do tipo macho que corresponde as saídas e um do tipo fêmea que corresponde as entradas.

As tabelas Tabela 2 e Tabela 3 especificam a relação entre os pinos de entrada e saída e o elevador que também podem ser encontradas em (Martínez, 2015).

Tabela 2:Entradas da maquete do elevador correspondentes ao conector X1.

Pinos do conector macho X1	Nome	Descrição
Posição da cabine	Formado por todos os interruptores mecânicos que se encontram no caminho da cabine, indicando a posição da mesma gerando 24V ao ativar-se.	
1	Cabine no piso 0	Indica quando a cabine está no piso 0, devolve 24V ao ser acionada.
2	Cabine no piso 1	Indica quando a cabine está no piso 1, devolve 24V ao ser acionada.
3	Cabine no piso 2	Indica quando a cabine está no piso 2, devolve 24V ao ser acionada.
Interruptores para diminuir a velocidade	Um total de 4 interruptores situados antes de cada piso que servem para diminuir a velocidade do motor antes de que a cabine chegue a um piso.	
4	Interruptor antes do piso 0	Situado a cima do interruptor que indica que a cabine está no piso 0.
5	Interruptor antes do piso 1 Ao baixar	Situado abaixo do interruptor que indica que a cabine está no piso 1.
6	Interruptor antes do piso 1 Ao subir	Situado acima o interruptor que indica que a cabine está no piso 1
7	Interruptor antes do piso 2	Situado abaixo do interruptor que indica que a cabine está no

		piso 2.
Portas e sensores ópticos	Detectam a posição da porta (aberta ou fechada), a interrupção da luza indica se algo ou alguém se interpôs na porta. Devolvem 0 se há interrupção da luz.	
8	Porta da planta 0 aberta	Devolve 24V se aporta da planta 0 está aberta.
9	Porta da planta 0 fechada	Devolve 24V se aporta da planta 0 está fechada.
10	Porta da planta 1 aberta	Devolve 24V se aporta da planta 1 está aberta.
11	Porta da planta 1 fechada	Devolve 24V se aporta da planta 1 está fechada.
12	Porta da planta 2 aberta	Devolve 24V se aporta da planta 2 está aberta.
13	Porta da planta 2 fechada	Devolve 24V se aporta da planta 2 está fechada.
14	Sensor óptico da planta 0	Devolve 0V se o feixe de luz for interrompido, caso contrário devolve 24V.
15	Sensor óptico da planta 1	
16	Sensor óptico da planta 2	
Botões na parte externa do elevador	Situados na parte exterior do elevador ao lado das portas. Ativos por borda de descida.	
20	Botão de chamada piso 0	Botão junto da porta da planta 0
21	Botão de chamada piso 1 Para baixar	Botão da parte de baixo da porta na planta 1
22	Botão de chamada piso 1 Para subir	Botão da parte de cima da porta na planta 1
23	Botão de chamada piso 2	Botão junto da porta da planta 2
29	Botão de sobrepeso	Botão na parte inferior da maquete
Botões o painel	Referem-se aos cinco botões que se encontram no painel da maquete	
24	Botão para levar o Elevador ao piso 0	São os três primeiros botões do painel de cores verdes.
25	Botão para levar o Elevador ao piso 1	

26	Botão para levar o Elevador ao piso 2	
27	Botão de alarma	Dois botões de cores vermelha situados abaixo dos botões dos que levam aos pisos
28	Botão de parada de emergência	

Tabela 3: Saídas da maquete do elevador correspondentes ao conector X2.

Pinos do conector fêmea X2	Nome	Descrição
Movimento da cabine		
1	Cabine subindo	Se está saída está ativa e os motores alimentados a cabine inicia movimento de subida, o topo possui um interruptor limite
2	Cabine descendo	Se está saída está ativa e os motores alimentados a cabine inicia movimento de descida, a base possui um interruptor limite
3	Motor lento	Reduz a velocidade da cabine
Movimento das Portas	Abrira e fechara as portas mediante um atuador. Os motores devem estar alimentados.	
4	Abrir porta do piso 0	
5	Fecha porta do piso 0	
6	Abrir porta do piso 1	
7	Fecha porta do piso 1	
8	Abrir porta do piso 2	
9	Fecha porta do piso 2	
LEDs dos botões de clamada	Mediante estas saídas acendem-se os Leds que estão no interior dos botões da parte externa do elevador.	
10	Botões de clamada do piso 0	
11	Botões de clamada do piso 1 para bajar	
12	Botão de chamada o piso 1 para subir	
13	Botão de chamada do piso 2	
LEDs indicadores na parte de	Refere-se aos Leds que se encontram na cabine encima do piso 2	

cima	e permitem identificar a posição da cabine.
14	Indicador piso 0
15	Indicador piso 1
16	Indicador piso 2
20	Cabine subindo
21	Cabina descendo
LEDs do painel	LED que estão ao lado dos botões do painel e controle.
22	Piso 0
23	Piso 1
24	Piso 2
25	Alarma
26	Parada de emergência
27	Sobrepeso

Tabela 4: Alimentação da maquete do elevador.

Pinos dos conectores macho e fêmea	Pinos	Descrição
GND 0 Volts	18 e 19 (Macho) 18 e 19 (Fêmea)	Não é necessário a conexão de ambos os pinos.
Alimentação dos motores	36 (Macho) 36(Fêmea)	Alimentação dos motores que movem a cabine, abrir e fechar as portas. Alimentados com 24V, não é necessário a conexão de ambos os pinos.
Alimentação dos sensores	36 (Macho) 37 (Fêmea)	Alimentação necessária para os sensores das portas e dos interruptores mecânicos. Alimentados com 24V, não é necessário a conexão de ambos os pinos.

1.4.5 Fonte de alimentação HAMEG INSTRUMENTS

Figura 5: HAMEG INSTRUMENTS

Para alimentação externa foi usada a fonte de alimentação HM7044 da empresa Hameg Instruments. Que dispõem de quatro canais independentes que podem fornecer até 32 Volts e uma corrente máxima de até 3 Amperes.

1.4.6 PCB para adaptar tensões.

Como já comentado antes a maquete do elevador funciona a 24 Volts e a placa Zybo funciona a 3,3 Volts sendo necessário o uso de duas placas para converter as tensões, as placas utilizadas foram desenhadas e imprimidas pela equipe do Lab-Deusto. A primeira das PCB tem como principais componentes acopladores ópticos que faram com que os 24 Volts enviados pela maquete do elevador se convertam em 3,3 Volts no conector de entrada B1. A segunda das PCB é composta por relés que faram com que os 3,3 Volts do conector de saída A2 enviados pela Zybo se convertam em 24 Volts na entrada da maquete do elevador como mostrado na Figura 7.

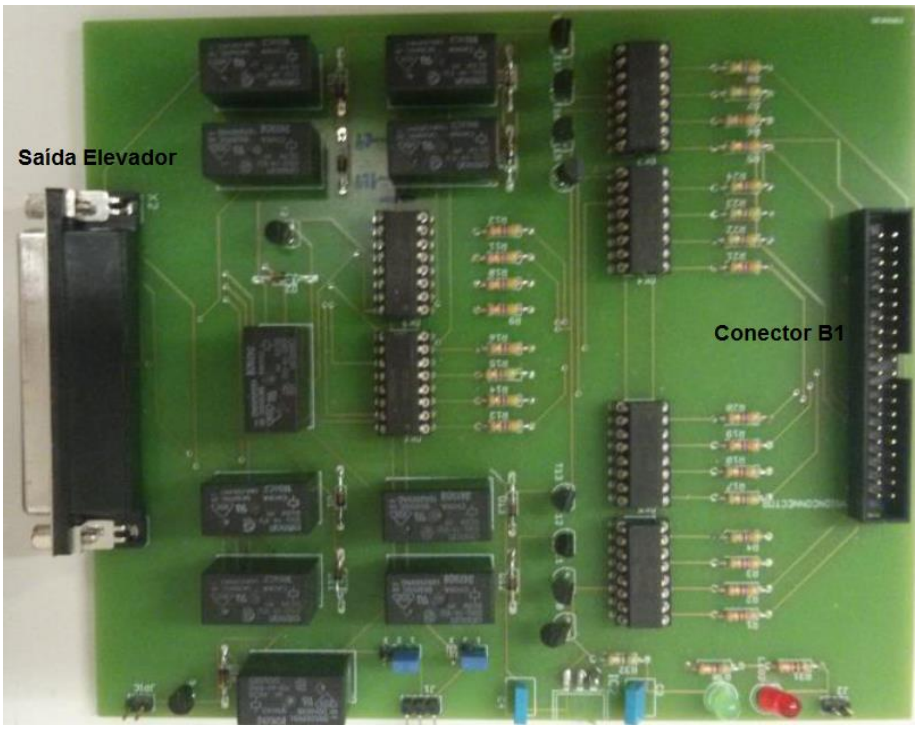


Figura 6: Abaixador de Tensão

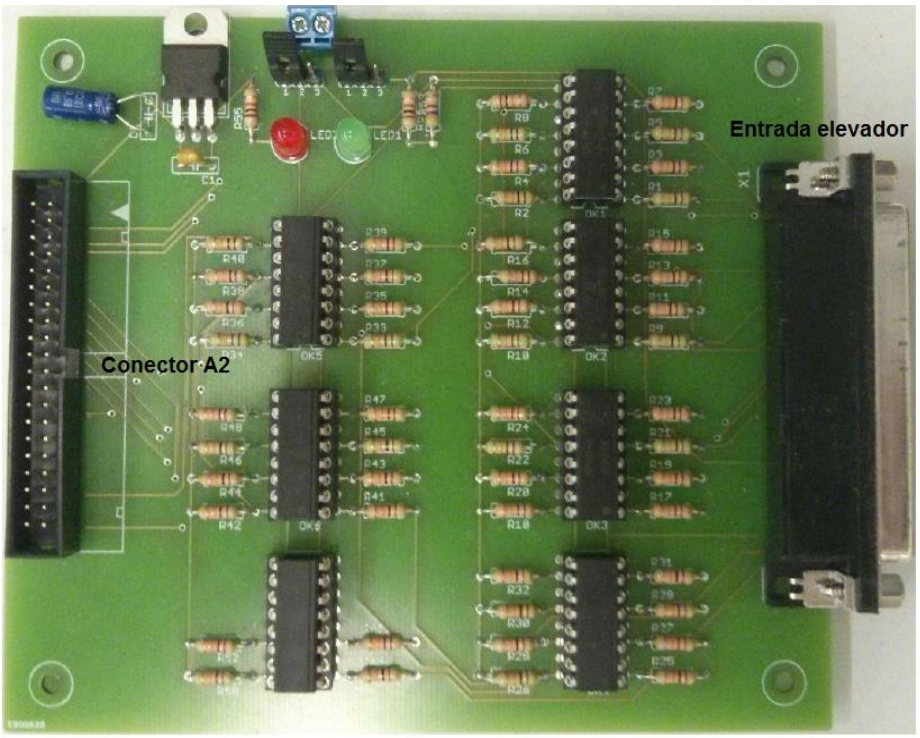


Figura 7: Elevador de Tensão

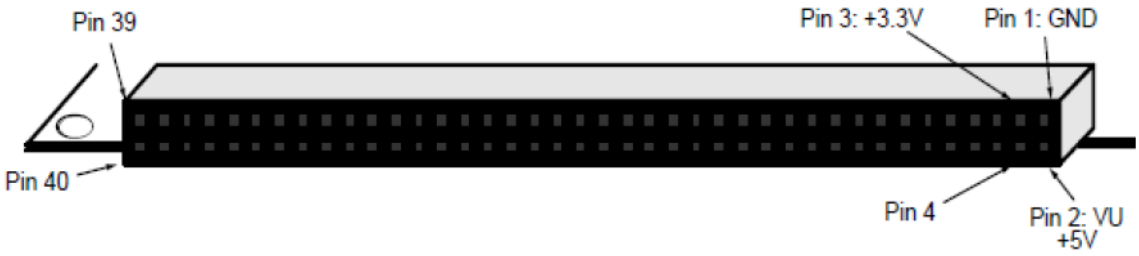


Figura 8 : Conector de 40 Pinos.

Saídas das PCB é composta por dois conectores de 40 pinos como mostrado na Figura 8, no entanto a Zybo possui 6 como mostrado na Figura 9.

Figura 9: Pmod da Zybo.

Para que seja possível a conexão da Zybo como as PCBs que possuem um conector de 40 pinos será necessária o uso de outra PCB intermediaria, o seu diagrama de conexão pode ser visto Figura 10.

1.4.7 Diagrama de montagem

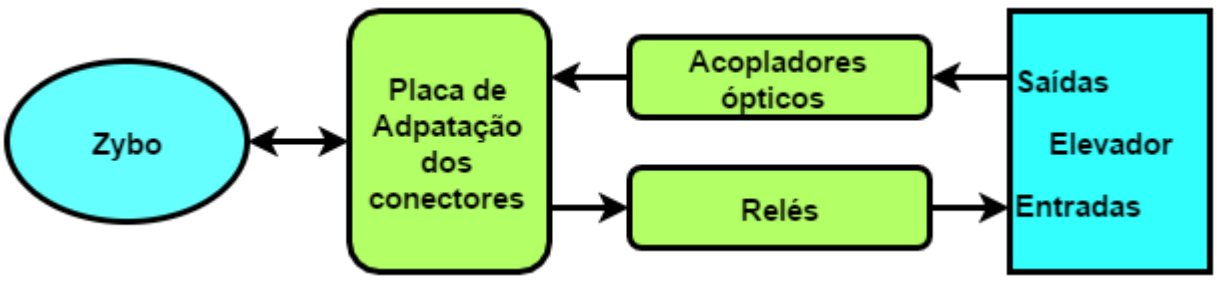


Figura 10: Diagrama de montagem do projeto

1.4.8 PCB Intermediaria para adaptar os conectores:

Para criação da PCB foi utilizado o Sfotware Eagle, primeiramente foi necessário especificar as conexões dos pinos da Zybo com as duas placas de adaptação de tensão, as tabelas Tabela 5 e Tabela 6 representam as especiações dos pinos dos Pmods da zybo correspondentes aos conectores das PCBs de conversão de tensão de saída A2 e entrada B1. As especificações dos pinos A2 e B1 pode ser encontrada em (Martínez, 2015). Em seguida foi utilizado o modulo **Schematic editor** para criar o esquemático da placa intermediaria que pode ser visto na

Figura 11, fazendo uso do modulo **Layout** editor criou-se o seu Layout que pode ser vista na Figura 12 Layout da placa *adaptação de conectores*. Por fim utilizando o modulo **Autorouter** para gerar a conexão automática das trilhas da placa. Em seguida seu projeto foi enviado a uma empresar responsável por fabricar PCBs.

Tabela 5: Saídas dos pinos da Zybo conectados a PCB intermediária.

Nome	Conector X2	Pmod - JA	Conector A2
Movimento da cabine			
Cabine subindo	1	JA01 - N15	4
Cabine descendo	2	JA02 - L14	5
Reduzir velocidade	3	JA03 - K16	6
LEDs indicadores na parte de cima			
Indicador piso 0	14	JA04 - K14	17
		JA05 - GND	1 e 2
		JA06 - VCC	
Indicador piso 1	15	JA07 - N16	18
Indicador piso 2	16	JA08 - L15	19
Cabine descendo	20	JA09 - J16	20
Cabina subindo	21	JA10 - J14	21

		JA11 - GND	1 e 2
		JA12 - VCC	Não usado

Nome	Conector X2	Pmod - JB	Conector A2
Movimento das Portas			
Abrir porta do piso 0	4	JB01 - T20	7
Fecha porta do piso 0	5	JB02 - U20	8
Abrir porta do piso 1	6	JB03 - V20	9
Fecha porta do piso 1	7	JB04 - W20	10
		JB05 - GND	1 e 2
		JB06 - VCC	Não usado
Abrir porta do piso 2	8	JB07 - Y18	11
Fecha porta do piso 2	9	JB08 - Y19	12
LEDs dos botões de chamada			
Botões de clamada do piso 0	10	JB09 - W18	13
Botões de clamada do piso 1 para bajar	11	JB10 - W19	14
		JB11 - GND	1 e 2
		JB12 - VCC	Não usado

Nome	Conector X2	Pmod - JC	Conector A2
Botão de chamada o piso 1 para subir	12	JC01 - V15	15
Botão de chamada do piso 2	13	JC02 - W15	16
LEDs do painel			
Piso 0	22	JC03 - T11	22
Piso 1	23	JC04 - T10	23
		JC05 - GND	1 e 2
		JC06 - VCC	
Piso 2	24	JC07 - W14	24
Alarma	25	JC08 - Y14	25
Parada de emergência	26	JC09 - T12	26

Sobrepeso	27	JC10 - U12	27
		JC11 - GND	1 e 2
		JC12 - VCC	Não usado

Tabela 6: Entradas da *Zybo* correspondentes aos conectores da placa intermediaria.

Nome	Conector X1	Pmod - JD	Conector B1
Posição da cabine			
Cabine no piso 0	1	JD01 - T14	4
Cabine no piso 1	2	JD02 - T15	5
Cabine no piso 2	3	JD03 - P14	6
Interruptores para diminuir a velocidade			
Interruptor antes do piso 0	4	JD04 - R14	7
		JD05 - GND	1 e 2
		JD06 - VCC	
Interruptor antes do piso 1 Ao baixar	5	JD07 - U14	8
Interruptor antes do piso 1 Ao subir	6	JD08 - U15	9
Interruptor antes do piso 2	7	JD09 - V17	10
Não usado		JD10 - V18	
		JD11 - GND	1 e 2
		JD12 - VCC	Não usado

Nome	Conector X1	Pmod - JE	Conector B1
Botões na parte externa do elevador			
Botão de chamada piso 0	20	JE01 - V12	20
Botão de chamada piso 1	21	JE02 - W16	21

Para baixar			
Botão de chamada piso 1 Para subir	22	JE03 - J15	22
Botão de chamada piso 2	23	JE04 - H15	23
		JE05 - GND	1 e 2
		JE06 - VCC	
Botão de sobrepeso	29	JE07 - V13	29
Botões o painel			
Botão para levar o Elevador ao piso 0	24	JE08 - U17	24
Botão para levar o Elevador ao piso 1	25	JE09 - T17	25
Botão para levar o Elevador ao piso 2	26	JE10 - Y17	26
		JE11 - GND	1 e 2
		JE12 - VCC	Não usado

Nome	**Conector X1**	**Pmod - JF**	**Conector B1**
Botão de alarma	27	JF01 - MIO-13 (E8)	27
Botão de parada de Emergência	28	JF02 - MIO-10 (E9)	28
Portas e sensores ópticos			
Porta da planta 0 aberta	8	JF03 - MIO-11 (C6)	11
Porta da planta 0 fechada	9	JF04 - MIO-12 (D9)	12
		JF05 - GND	1 e 2
		JF06 - VCC	
Porta da planta 1 aberta	10	JF07 - MIO-00 (E6)	13
Porta da planta 1 fechada	11	JF08 - MIO-09 (B5)	14
Porta da planta 2 aberta	12	JF09 - MIO-14 (C5)	15
Porta da planta 2	13	JF10 - MIO-15 (C8)	16

fechada			
		JF11 - GND	1 e 2
		JF12 - VCC	Não usado

1.4.8.1 Esquemático da placa intermediaria.

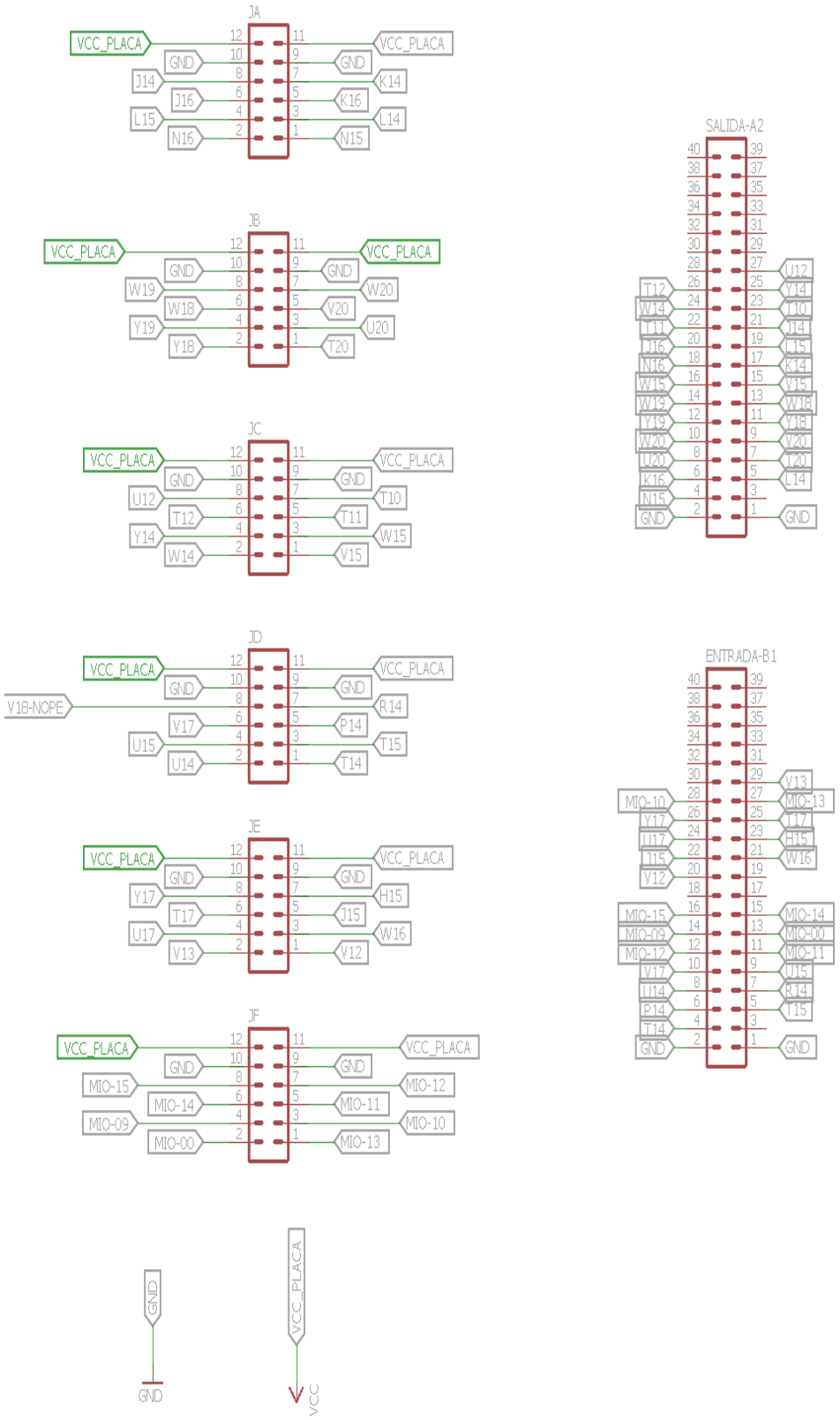


Figura 11 Esquemático da placa de adaptação de conectores.

1.4.8.2 Layout da placa intermediaria.

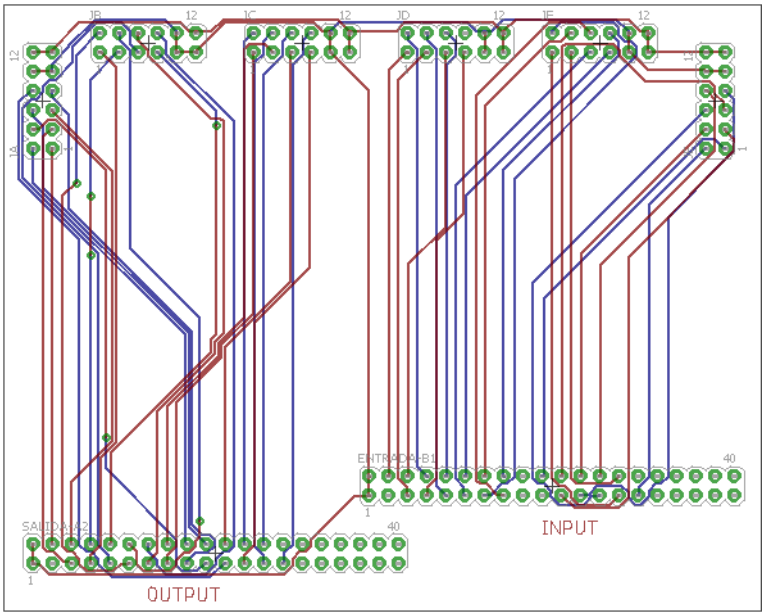


Figura 12 Layout da placa *adaptação de conectores*.

1.4.9 PCB utilizando matriz de contato.

Foi feita duas PCBs afim de validar o projeto e esquemáticos feito com o Eagle como mostra

Figura 13 PCB com matriz de contato.

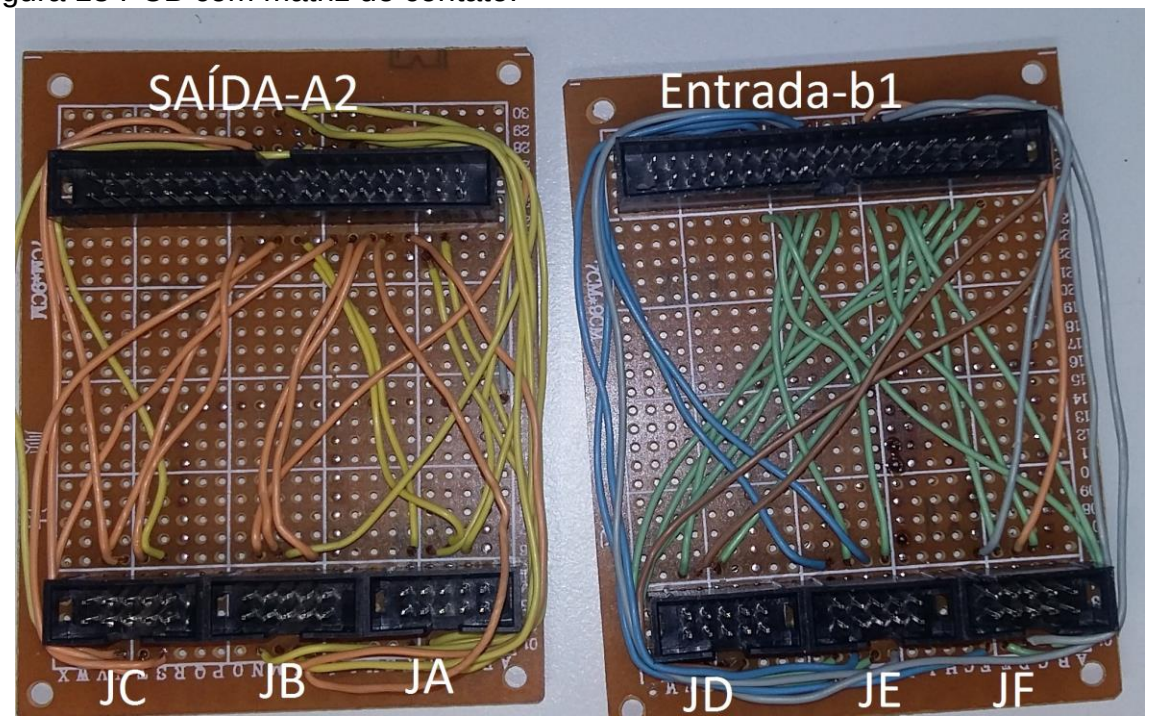


Figura 13 PCB com matriz de contato.

1.4.10 Criação do Arquivo XDC

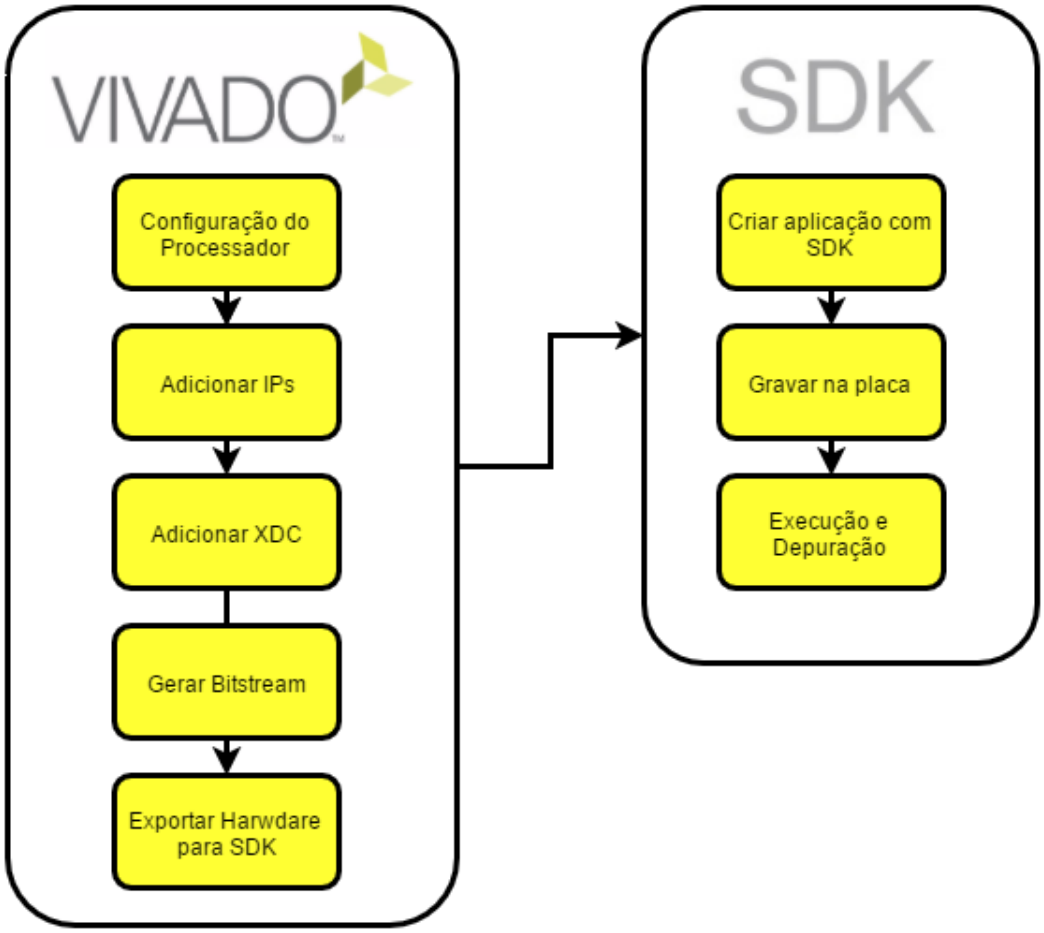
O código abaixo corresponde ao mapeamento dos pinos da Zybo de acordo com a Tabela 5 e a Tabela 6.

```
##Pmod Header JA (XADC)
set_property PACKAGE_PIN N15 [get_ports {ja_tri_o[0]}]
set_property PACKAGE_PIN L14 [get_ports {ja_tri_o[1]}]
set_property PACKAGE_PIN K16 [get_ports {ja_tri_o[2]}]
set_property PACKAGE_PIN K14 [get_ports {ja_tri_o[3]}]
set_property PACKAGE_PIN N16 [get_ports {ja_tri_o[4]}]
set_property PACKAGE_PIN L15 [get_ports {ja_tri_o[5]}]
set_property PACKAGE_PIN J16 [get_ports {ja_tri_o[6]}]
set_property PACKAGE_PIN J14 [get_ports {ja_tri_o[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ja_tri_o[0]}]
##Pmod Header JB
set_property PACKAGE_PIN T20 [get_ports {jb_tri_o[0]}]
set_property PACKAGE_PIN U20 [get_ports {jb_tri_o[1]}]
set_property PACKAGE_PIN V20 [get_ports {jb_tri_o[2]}]
set_property PACKAGE_PIN W20 [get_ports {jb_tri_o[3]}]
set_property PACKAGE_PIN Y18 [get_ports {jb_tri_o[4]}]
set_property PACKAGE_PIN Y19 [get_ports {jb_tri_o[5]}]
set_property PACKAGE_PIN W18 [get_ports {jb_tri_o[6]}]
set_property PACKAGE_PIN W19 [get_ports {jb_tri_o[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[6]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jb_tri_o[0]}]
##Pmod Header JC
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jc_tri_o[0]}]
set_property PACKAGE_PIN V15 [get_ports {jc_tri_o[0]}]
set_property PACKAGE_PIN W15 [get_ports {jc_tri_o[1]}]
set_property PACKAGE_PIN T11 [get_ports {jc_tri_o[2]}]
set_property PACKAGE_PIN T10 [get_ports {jc_tri_o[3]}]
set_property PACKAGE_PIN W14 [get_ports {jc_tri_o[4]}]
set_property PACKAGE_PIN Y14 [get_ports {jc_tri_o[5]}]
set_property PACKAGE_PIN T12 [get_ports {jc_tri_o[6]}]
set_property PACKAGE_PIN U12 [get_ports {jc_tri_o[7]}]
##Pmod Header JD
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {jd_tri_i[0]}]
set_property PACKAGE_PIN T14 [get_ports {jd_tri_i[0]}]
```

```
set_property PACKAGE_PIN T15 [get_ports {jd_tri_i[1]}]
set_property PACKAGE_PIN P14 [get_ports {jd_tri_i[2]}]
set_property PACKAGE_PIN R14 [get_ports {jd_tri_i[3]}]
set_property PACKAGE_PIN U14 [get_ports {jd_tri_i[4]}]
set_property PACKAGE_PIN U15 [get_ports {jd_tri_i[5]}]
set_property PACKAGE_PIN V17 [get_ports {jd_tri_i[6]}]
set_property PACKAGE_PIN V18 [get_ports {jd_tri_i[7]}]
##Pmod Header JE
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {je_tri_i[0]}]
set_property PACKAGE_PIN V12 [get_ports {je_tri_i[0]}]
set_property PACKAGE_PIN W16 [get_ports {je_tri_i[1]}]
set_property PACKAGE_PIN J15 [get_ports {je_tri_i[2]}]
set_property PACKAGE_PIN H15 [get_ports {je_tri_i[3]}]
set_property PACKAGE_PIN V13 [get_ports {je_tri_i[4]}]
set_property PACKAGE_PIN U17 [get_ports {je_tri_i[5]}]
set_property PACKAGE_PIN T17 [get_ports {je_tri_i[6]}]
set_property PACKAGE_PIN Y17 [get_ports {je_tri_i[7]}]
```

1.4.11 Fluxo do projeto



O projeto deve ser iniciado pelo Vivado IDE (Integrated Design Environment), e então o hardware criado é exportado para o Xilinx SDK (Software Development Kit). Primeiro passo do fluxo é a configuração do Processador no Vivado, no diagrama de bloco Zynq são configuradas as opções de memória, frequência de operação do processador, os periféricos disponíveis no hardware que vão ser instanciados, as interfaces de comunicação com o barramento AXI e as interrupções. Foram adotadas as seguintes configurações:

- M_AXI_GP0 interface
- FCLK_RESET0_N
- FCLK_CLK0 (clock frequency of 100.000000 MHz)
- GPIO MIO (Martínez, 2015)

Na etapa de inclusão de IPs, podem ser adicionados IPs disponíveis no catálogo do Vivado ou adicionado um IP desenvolvido especificamente para o projeto. Para o

controle do elevador iremos utilizar os IPS GPIO (General Purpose Input/Output) disponíveis no catalogo vivado para controlar os Pmods.

A placa possui 6 Pmods e todos serão necessário no projeto, no entanto no diagrama da Figura 14 só será adicionado cinco GPIO, isso deve se ao fato de o MIO Pmod JF ser dedicado ao processador sendo controlando diretamente na aplicação.

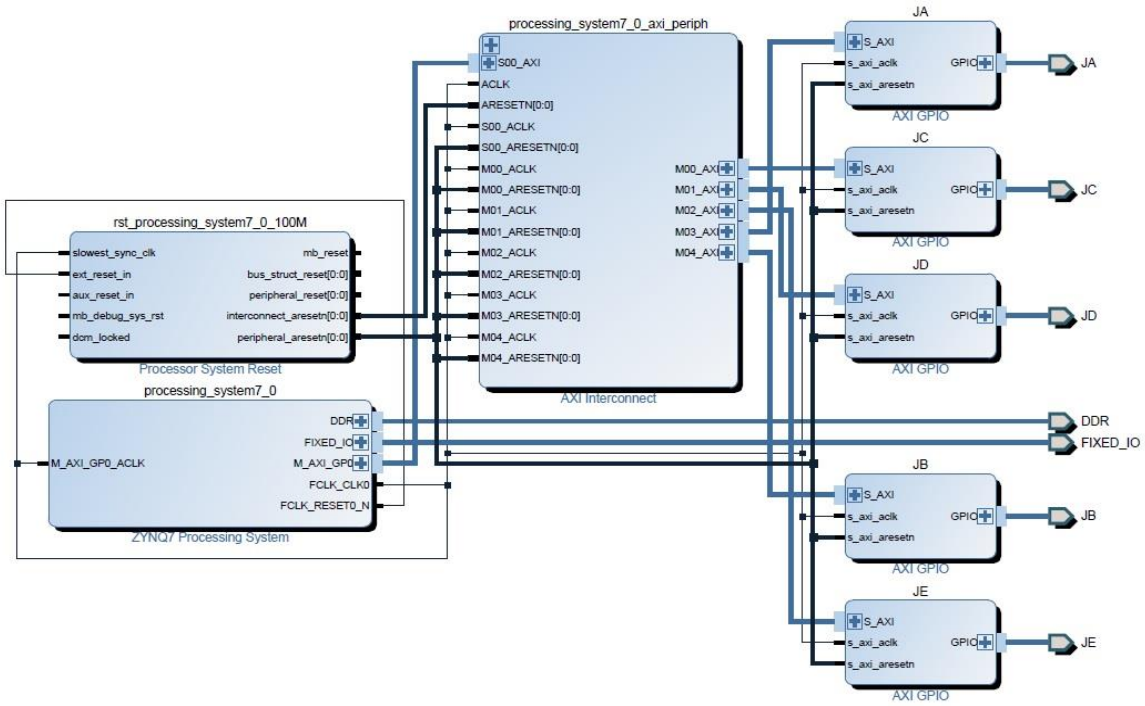


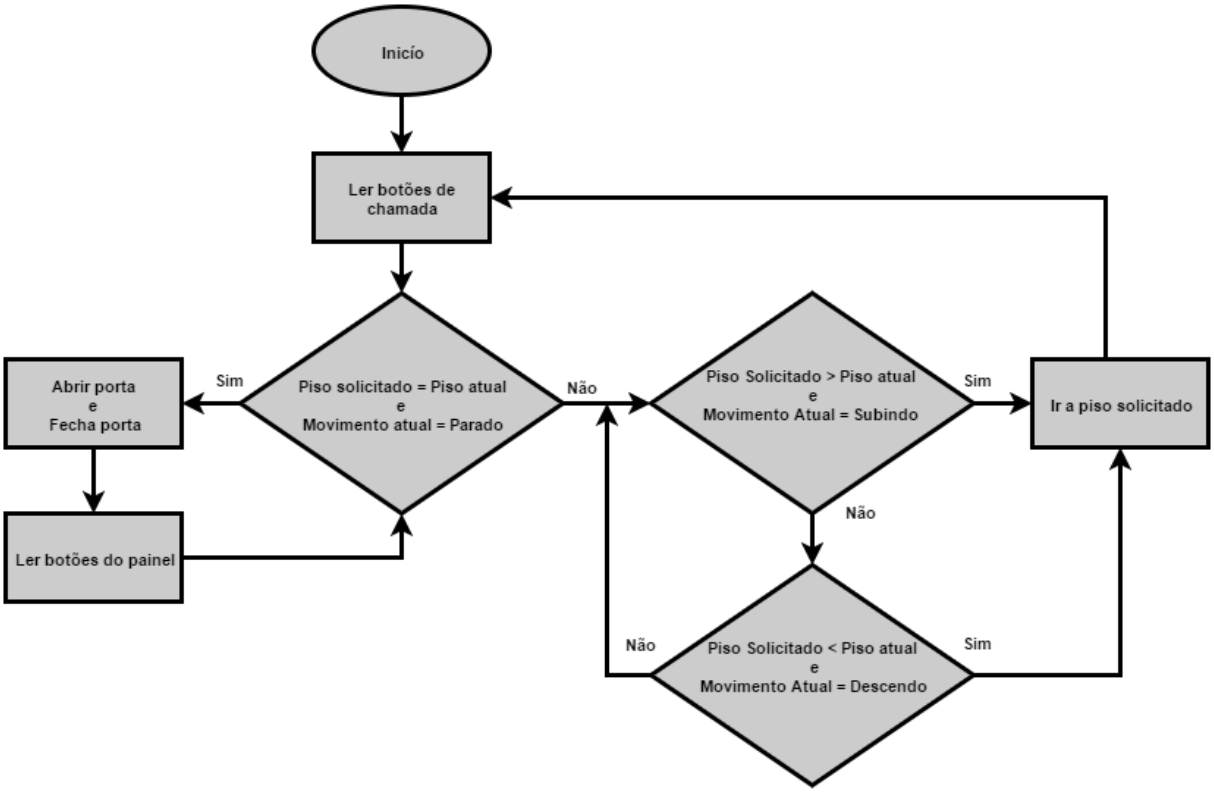
Figura 14: Diagrama da arquitetura do sistema.

Aproxima etapa consiste em importa o arquivo XDC, esse arquivo possui as especificações dos pinos de entrada e saída que foram cuidadosamente escolhidos e suas funcionalidade podem ser vistas nas tabelas Tabela 5 e Tabela 6.

Após validado o projeto é gerado o bitstream e então exportado para o SDK.

O SDK possui especificações do hardware exportado, como periféricos disponíveis e seus respectivos endereçamentos. Com base neste projeto é desenvolvido um software de controle do elevador que vai se comunicar com o processador. Após concluído e gerado um arquivo com a extensão. elf e um arquivo .bit para ser gravado na placa. Por fim é possível fazer a execução e depuração utilizando o SDK.

1.4.12 Fluxo de funcionamento do elevador



. O fluxo consiste em atender as chamadas do elevador de acordo com a memória do estado do movimento, ou seja, se ele está subido continua subindo e atendendo as chamadas para subir. Não havendo mais solicitações para subir começa a descer e atendo as chamadas para descer.

1.4.13 Algoritmo de controle

O trecho de código abaixo representa as variáveis e defines utilizados no algoritmo de controle.

```
#define turn_on 0xffffffff
#define turn_off 0x00000000
#define chanel 1
#define tempo_parado 99999999
#define GPIO_DEVICE_ID  XPAR_XGPIOPS_0_DEVICE_ID
```

```
#define INPUT_MIO_13      13  // MIO-13
#define INPUT_MIO_10      10  // MIO-10
#define INPUT_MIO_11      11  // MIO-11
#define INPUT_MIO_12      12  // MIO-12
#define INPUT_MIO_00       0  // MIO-00
#define INPUT_MIO_09       9  // MIO-09
#define INPUT_MIO_14      14  // MIO-14
#define INPUT_MIO_15      15  // MIO-14

enum BitMask {
BIT_0 = 0x00,
BIT_1 = 0x01,
BIT_2 = 0x02,
BIT_3 = 0x04,
BIT_4 = 0x08,
BIT_5 = 0x10,
BIT_6 = 0x20,
BIT_7 = 0x40,
BIT_8 = 0x80

};

enum movimentos {
Parado, Subindo, Descendo
};
enum movimentos movimento_atual;

enum Pisos {
piso_0, piso_1, piso_2
};
enum Pisos piso_atual;

struct Painel {
int B0;
int B1;
int B2;
int Alarme;
int Parada_emergencia;
int Sobrepeso;
} botoes_painel;

struct Elevador {
int B0;
int B1_Subindo;
int B1_Descendo;
int B2;
} botoes_chamada;

struct sensores_interruptores {
int I0;
int I1_up;
int I1_dw;
int I2;
} interruptores;
```

Como mencionado anteriormente o Pmod JF é dedicado ao processador, dessa forma foi nescessario configura-lo, utilizamos funções disponíveis na biblioteca xgpiops.h, para ler individualmente os pinos do Pmod JF como mostrado no trecho de código abaixo.

```
int Status;
XGpioPs Gpio_ps;
XGpioPs_Config *ConfigPtr;

ConfigPtr = XGpioPs_LookupConfig(GPIO_DEVICE_ID);
Status = XGpioPs_CfgInitialize(&Gpio_ps, ConfigPtr, ConfigPtr->BaseAddr);
if (Status != XST_SUCCESS) {
	return XST_FAILURE;
}

int	dip_mio_13, dip_mio_10, dip_mio_11, dip_mio_12,
	dip_mio_00, dip_mio_09, dip_mio_14, dip_mio_15;

XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_13, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_10, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_11, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_12, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_00, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_09, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_14, 0);
XGpioPs_SetDirectionPin(&Gpio_ps, INPUT_MIO_15, 0);
```

A seguir são mostradas as funções utilizadas:

1.4.13.1 Funções para escrever nos Pmods

A seguintes funções são responsáveis por escrever nos pinos dos Pmods de forma individual, dessa forma para escrever ‘1’ em um pino deve-se utilizado a função ligar () e para escrever ‘0’ deve-se a função desligar () no Pmod correspondente ao pino passando-se o número do pino desejado.

- void **ligar_pmod_b**(unsigned char valor)
- void **delisgar_pmod_b**(unsigned char valor)
- void **ligar_pmod_c**(unsigned char valor)
- void **delisgar_pmod_c**(unsigned char valor)
- void **ligar_pmod_a**(unsigned char valor)

- void **delisgar_pmod_a**(unsigned char valor)

1.4.13.2 Movimento do motor

O motor possui cinco movimentos possíveis, quando em movimento pode subir ou descer em velocidade normal, quando se aproxima de um piso reduz sua velocidade antes de parar, quando parado esse por sua vez mantém-se parado no piso por um certo tempo estipulado na função **delay** () que apenas é responsável por consumir tempo.

- void **motor_descendo**()
- void **motor_subindo**()
- void **motor_subindo_lento**()
- void **motor_descendo_lento**()
- void **motor_parado**()

1.4.13.3 Sensores

A função sensores tem como objetivo ler todos pinos dos Pmods de entradas em seguidas as outras funções dividem os pinos correspondentes aos pisos, as portas, interruptores antes cada piso, botões de chamada e botões do painel.

- void **sensores**()
- void **sensores_piso_atual**()
- void **sensores_interrupotores**()
- void **sensores_botones_chamada**()
- void **sesores_botones_painel**()
- void **sensor_portas**()

1.4.13.4 Leds

A seguintes funções tem como objetivo acender ou desligar todos o leds da maquete.

- void **leds_botones_chamadas**()
- void **leds_botones_painel**()

- void **leds_indicadores**()

1.4.13.5 Movimento das portas

As funções abaixo são responsáveis pelo movimentos das portas, garantido que somente se abriá a porta se o elevador estiver parado e no piso solicitado.

- void **fechar_portas**()
- void **abri_portas**()

1.4.13.6 Movimento do elevador

Sendo um elevador de três pisos, serão descritos os três possíveis movimentos do elevador de acordo com as funções abaixo.

- void **ir_piso_1**()
  Verifica se o piso atual é maior que o piso 1 se sim inicia movimento de descida, se não inicia movimento de subida.
- void **ir_piso_0**()
  Inicia movimento de descida até o piso 0, verifica se há um chamado para o piso 1, se houver e ainda não estiver passado pelo piso 1 então o elevador para no piso 1.
- void **ir_piso_2**()
  Inicia movimento de subida até o piso 2, verifica se há um chamado para o piso 1, se houver e ainda não estiver passado pelo piso 1 então o elevador para no piso 1.

3 RESULTADOS

Na realização do projeto foi constado o perfeito funcionamento da maqueta do elevador durante a execução do algoritmo de controle, A ZYBO (Zynq Board) mostrou-se uma ferramenta excelente para desenvolvimento de sistema de controle, fornecendo uma gama de funcionalidades intuitivas com o uso do ambiente de desenvolvimento VIVADO disponibilizado pela Xilinx.

Esse projeto forneceu ao aluno a capacidade de implementar e entender os diferentes estágios para criação de sistema real, a capacidade de criar um IP (intellectual property), a criação de uma **(**printed circuit board) e principalmente a capacidade de trabalhar com metas, colocando em pratica os conhecimentos até então adquiridos durante a graduação.

4 CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho foi desenvolvido um sistema capaz de controlar uma maquete de um elevador que será adicionada ao acervo da plataforma para laboratórios remotos da universidade de Deusto, denominada Weblab-Deusto. O mencionado sistema de controle foi desenvolvido utilizando a placa de desenvolvimento ZYBO (Zynq Board) da família Xilinx Zynq-7000, a mesma placa que está sendo adotada em diversas disciplinas da universidade.

Foi utilizado o microprocessador (ARM) disponível pela Zybo na construção do algoritmo de controle do elevador, que por sua vez foi criado na linguagem de programação C. Durante a construção do sistema foi necessário a construção de uma placa com objetivo de adaptar as conexões da Zybo com as placas de conversões de tensão utilizando o software Eagle.

Este projeto fornece aos alunos a possibilidade de praticar lógica de programação utilizando um sistema muito próximo do real por meio da interface Web.

O seguinte projeto também deixa com possibilidade novos estudos como um algoritmo que faça com que o elevador tenha melhor consumo energético ou a implementação de um algoritmo de controle utilizando a linguagem VHDL.

Tendo em mente que a Zybo fornece uma capacidade muito alta para desenvolvimento poderia se implementar um sistema controlado por voz, ou que através de uma *tag* RFID possa identificar e dar permissão para ir a determinado andar, sendo assim podemos ter uma gama muito alta de possibilidades.

5 REFERÊNCIAS

Martínez, J. G. (2015). *Sistema de control y programación remota de un ascensor basado en un dispositivo programable de procesamiento paralelo tipo FPGA bajo programación VHDL.* Bilbao: Universidad de Deusto.