Sistema de Teste Automático de Componentes Eletrónicos

Daniela Alexandra Dias Baião

Orientadora

Professora Doutora Paula Cristina Alves Pereira

Trabalho de Projeto apresentado à Escola Superior de Tecnologia para cumprimento dos requisitos necessários à obtenção do grau de Licenciada em Engenharia Eletrotécnica e das Telecomunicações, realizada sob a orientação científica da Professora Doutora Paula Pereira, do Instituto Politécnico de Castelo Branco.

outubro de 2024

Composição do júri

Presidente do júri

Professor Doutor, Rogério Pais Dionisio

Professor Coordenador, EST-IPCB

Vogais

Professor Doutor, José António Barros Vieira

Professor Adjunto, EST-IPCB

Professora Doutora, Paula Cristina Alves Pereira (orientadora)

Professora Adjunta, EST-IPCB

Dedicatória

Aos meus pais e amigos que sempre me apoiaram nesta nova loucura de realizar uma segunda licenciatura e que me deram força nos momentos de maior cansaço.

À Niki, que apesar de me tentar roer os fios do projeto sempre que possível, esteve sempre lá a fazer me companhia.

Agradecimentos

Gostaria de expressar a minha sincera gratidão a todas as pessoas que contribuíram, de diferentes maneiras, para a realização deste projeto.

Em primeiro lugar, agradeço à minha orientadora, Professora Paula Pereira, pela orientação, paciência e ajuda ao longo de todas as fases deste trabalho. Os seus conselhos e o seu conhecimento técnico foram essenciais para o sucesso deste projeto.

Agradeço também à Escola Superior de Tecnologia de Castelo Branco, pela infraestrutura e recursos disponibilizados, que me permitiram desenvolver este projeto. E aos professores da Licenciatura que sempre tiveram uma palavra de incentivo para com os alunos e pelos conhecimentos transmitidos durante estes três anos.

Aos meus colegas e amigos, que ao longo deste percurso académico ofereceram suporte e incentivo, deixando sempre uma palavra de motivação nos momentos mais desafiantes.

Por fim, um agradecimento muito especial à minha família, pelo apoio incondicional, paciência e compreensão desde sempre.

A todos, o meu sincero agradecimento.

Resumo

Este projeto teve como objetivo o desenvolvimento de um Sistema de Teste Automático de Componentes Eletrónicos, focado na verificação de portas lógicas utilizando o microcontrolador ESP32-S3 e uma configuração de relés. A motivação principal foi criar uma solução prática e eficiente para testar a funcionalidade de circuitos lógicos, algo essencial em ambientes escolares e industriais, onde o correto funcionamento de componentes eletrónicos é essencial.

O sistema foi realizado para testar portas lógicas da família 74LS, especificamente as portas AND, OR, XOR, NAND, NOR, XNOR e NOT. Para isso foram utilizados relés como elementos de comutação, controlados pelo ESP32, para configurar as entradas e saídas de cada porta lógica. Esta solução permitiu a criação de um sistema flexível, capaz de realizar testes automáticos sem necessidade de modificar a configuração física entre diferentes testes de portas.

O projeto incluiu a utilização de diversos componentes eletrónicos, tais como díodos, transístores BC546C, relés SRD-05VDC-SL-C, e circuitos integrados da série 74HCT00. A escolha do ESP32-S3, devido à sua capacidade de lidar com múltiplos pinos de entrada/saída e a sua compatibilidade com o ambiente de desenvolvimento Arduino IDE, foi essencial para a implementação prática.

O sistema foi testado com sucesso, e os resultados mostraram que a solução desenvolvida é eficaz para identificar o estado de funcionamento das portas lógicas. O teste de cada porta lógica foi realizado com precisão, confirmando se as portas estavam a funcionar corretamente conforme a sua tabela verdade.

Palavras chave

Sistema de Testes Automático; Portas Lógicas; ESP32-S3; 74LS

Abstract

This project aimed to develop an Automatic Testing System for Electronic Components, focused on verifying logic gates using the ESP32-S3 microcontroller and a relay configuration. The main motivation was to create a practical and efficient solution to test the functionality of logic circuits, something essential in both academic and industrial environments, where the correct operation of electronic components is critical.

The system was designed to test logic gates from the 74LS family, specifically the AND, OR, XOR, NAND, NOR, XNOR, and NOT gates. To achieve this, relays were used as switching elements, controlled by the ESP32, to configure the inputs and outputs of each logic gate. This solution allowed the creation of a flexible system, capable of performing automatic tests without the need to modify the physical configuration between different gate tests.

The project involved the use of various electronic components, such as diodes, BC546C transistors, SRD-05VDC-SL-C relays, and integrated circuits from the 74HCT00 series. The choice of the ESP32-S3, due to its ability to handle multiple input/output pins and its compatibility with the Arduino IDE development environment, was essential for practical implementation.

The system was successfully tested, and the results showed that the developed solution is effective in identifying the operational state of logic gates. Each gate was tested with precision, confirming whether the gates were functioning correctly according to their truth table.

Keywords

Automatic Testing System; Logic Gates; ESP32-S3; 74LS

Índice Geral

Índice de figuras

Lista de tabelas

Lista de abreviaturas, siglas e acrónimos

A - Amperes

AC - Corrente Alternada

ADC - *Analog to Digital Converter*

B- Base

CI- Circuito Integrado

C- Coletor

DC- Corrente Continua

DUT- *Device Under Test*

ATE- Equipamentos de Teste Automático

E- Emissor

GND - terra

HCT- *High-Speed CMOS with TTL compatibility*

I2C- *Inter-Integrated Circuit*

I2S- *Integrated Interchip Sound*

IIH- *Input High Current*

IIL- *Input Low Current*

IOH- *Output High Current*

IOL- *Output Low Current*

IoT- *Internet of Things*

LED- *Light Emitting Diodes*

LSI- *Large-scale integration*

MSI- *Medium-scale integration*

NO- Normalmente Aberto

NC- Normalmente Fechado

PCBs- Placas de Circuito Impresso

PWM- *Pulse Width Modulation*

SRAM- *Static Random Access Memory*

SPDT- *Single Pole Double Throw*

SPI- *Serial Peripheral Interface*

SSI- *Small-scale integration*

TTL- *Transistor-Transistor Logic*

UART- *Universal Asynchronous Receiver-Transmitter*

V- Volts

VIH- *High-level input voltage*

VIL- *Low-level input voltage*

VLSI- *Very-Large-Scale Integration*

VOH- *Voltage Output High*

VOL- *Voltage Output Low*

ZIF- *Zero Insertion Force*

1. Introdução

O presente projeto, intitulado "Sistema de Teste Automático de Componentes Eletrónicos", teve como objetivo a criação de uma solução prática para o teste de portas lógicas básicas utilizando relés controlados por um microcontrolador ESP32 S3. A necessidade de garantir o correto funcionamento de componentes eletrónicos é crucial em indústrias que dependem de sistemas digitais, como automóveis, semicondutores e dispositivos de consumo. Tendo em vista a importância desses testes, este projeto foi desenvolvido com o intuito de oferecer um sistema eficiente e de baixo custo que automatiza o processo de verificação de portas lógicas como AND, NAND, OR, XOR, XNOR, NOR e NOT.

A escolha do ESP32 S3 como microcontrolador foi essencial devido à sua capacidade de controlo de múltiplos pinos GPIO, conectividade com outros dispositivos e facilidade de programação utilizando o *software* Arduino IDE. Além disso, foram utilizados componentes acessíveis, como relés, transístores, díodos e circuitos integrados da série HCT, nomeadamente a porta 74HCT00, de forma a garantir a compatibilidade e eficiência do sistema.

Neste relatório, são detalhadas as fases de conceção e implementação do projeto. Começamos por apresentar inicialmente uma introdução a sistemas de teste automático de componentes eletrónicos, de forma a contextualizar o projeto. No capítulo seguinte são apresentados os componentes de *hardware* utilizados, tal como as suas principais características que influenciaram a realização do projeto e o seu comportamento neste. No quarto capítulo são dados a conhecer os programas de *software* utilizados no projeto e o que foi realizado em cada um deles enquanto que no capitulo seguida, falamos sobre a implementação prática do projeto, aqui podemos verificar o funcionamento do circuito, onde são apresentados tanto o esquema funcional como o elétrico do sistema implementado, alguns dos resultados dos testes das portas lógicas, os fluxogramas que orientaram o código do programa, o desenho do PCB e finalmente os custos associados ao projeto.

O objetivo final é proporcionar uma solução automatizada que permita a rápida validação de componentes em diversos contextos, minimizando o risco de erros e aumentando a produtividade.

2. Sistema de Teste Automático de Componentes Eletrónicos

Quando falamos em Sistemas Automáticos de Componentes Eletrónicos pensamos em conjuntos de instrumentos e *software* destinados para testar e avaliar a funcionalidade, o desempenho e a qualidade de componentes eletrónicos. Estes tipos de sistemas são utilizados na indústria de fabricação de componentes eletrónicos, especialmente quando falamos em produção em massa, para garantir que os componentes fabricados e os seus dispositivos finais funcionem corretamente e atendam às especificações necessárias.

Os equipamentos de teste automático (ATE-*Automatic Test Equipment)* referem-se a um conjunto de ferramentas e sistemas controlados por um computador, criados com o objetivo de executar testes e medições em dispositivos e em componentes eletrónicos.

O teste automático de equipamentos é utilizado em várias indústrias de forma a testar a qualidade dos produtos produzidos, como na Indústria dos semicondutores, onde o ATE é muito utilizado na produção de circuitos integrados de forma a realizar os testes funcionais dos circuitos. Na indústria automóvel, a ATE desempenha um papel muito importante na fabricação de peças automóveis como nos testes do controlo elétrico, nos sensores entre outros, sendo útil também para a avaliação do desempenho e segurança automóvel para corresponder aos padrões exigidos pela indústria automóvel. Na indústria aerospacial e na defesa o ATE também desempenha um papel fundamental, pois permite testar sistemas de radar, equipamentos de navegação e orientação, e outros elementos eletrónicos críticos na indústria, uma vez que estes testes permitem garantir um correto funcionamento dos mesmos, confiabilidade e desempenho dos sistemas que são essenciais para a segurança e proteção nacional. Também na indústria dos dispositivos de consumo comum o ATE desempenha um papel fundamental pois durante a produção de *smartphones*, tablets, eletrodomésticos, entre outros equipamentos que utilizamos no nosso dia-a-dia, é necessário verificar o correto funcionamento dos mesmos de forma a proporcionar a experiência desejada aos utilizadores (Sciotex, n.d.). Estes são apenas alguns exemplos de indústrias onde o ATE desempenha um papel crucial, sendo que existem muitas outras onde o teste dos equipamentos produzidos é essencial para o bom desempenho das indústrias que o utilizam.

No passado, a maioria das medições era realizada manualmente, sendo necessário ler os resultados diretamente do aparelho de medição. Devido á evolução da automação, surgiu a necessidade de automatizar essas medições manuais, uma vez que era despendido muito tempo em medições de rotina. No início dos anos setenta, surgiram vários instrumentos no mercado capazes realizar estas medições. O instrumento era conectado através de um cabo a um computador, frequentemente chamado de controlador. Este controlador era conectado mecânica e eletricamente aos instrumentos via uma interface, normalmente chamada de barramento de interface. A este conjunto de instrumentos com o controlador e a interface era chamada de sistema de instrumentação. Como o teste e a medição eram os principais propósitos do sistema, ele também era chamado de Sistema de Teste e Medição ou um sistema ATE. Com o passar dos anos e o desenvolvimento da indústria foi necessário criar sistemas de medição mais modernos e rápidos, apesar de manter a lógica de funcionamento que no início dos anos setenta (Pieper, 2014).

O ATE é composto por vários componentes de *hardware* e *software*, cada um contribuindo para o processo geral dos testes realizados, sendo alguns dos principais componentes os seguintes:

- Instrumentos de teste, refere-se ao *hardware* utilizado para medir os sinais elétricos e características dos componentes eletrónicos, como por exemplo osciloscópios, geradores de sinais, fontes de alimentação, multímetros, etc.
- *Test Fixtures*, são estruturas mecânicas que fazem de interface de dispositivos eletrónicos com o sistema ATE. Estes fornecem conexões elétricas, suporte mecânico e controlo térmico durante os processos dos testes. Normalmente são desenhados com o objetivo de testar placas de circuito impresso (PCBs) ou circuitos integrados;
- *Software* de teste, os sistemas ATE dependem de programas de *software* para controlar os componentes de *hardware*, executar os testes, adquirir dados e gerar relatórios. O *software* de testes fornece a interface para definir a sequência dos testes, definir os parâmetros a avaliar e analisar os resultados;
- Sistemas de aquisição de dados, são os responsáveis por converter os sinais analógicos em sinais digitais, de forma que possam ser analisados posteriormente;
- Pontas de teste e Interfaces, permitem estabelecer as ligações elétricas entre o sistema ATE e o DUT (*Device Under Test*), é através delas que se estabelece o contato físico com os terminais do dispositivo, permitindo a transmissão e medição de sinais.

Estes são alguns dos componentes que desempenham papéis muito importantes na automação e eficiência dos testes realizados (Sciotex, n.d.).

A indústria dos semicondutores é uma das mais importantes e estratégicas do mundo moderno, é fundamental para a fabricação de equipamentos eletrónicos, como *smartphones*, computadores, carros, eletrodomésticos entre outros equipamentos com que trabalhamos no nosso dia-a-dia. Os semicondutores são materiais com propriedades elétricas intermediarias entre condutores e isolantes, o que os torna ideais para controlar o fluxo de corrente elétrica em circuitos eletrónicos (Diegues & Roselino, 2012).

Semiconducter ATE, referem-se a sistemas automáticos utilizados para testar a qualidade e a funcionalidade de semicondutores durante a sua produção, estes podem testar uma grande variedade de dispositivos eletrónicos desde componentes simples, como resistências, condensadores, bobines, até circuitos integrados, placas de circuito impresso (PCBs) e sistemas eletrónicos mais complexos já montados. Este tipo de sistema, são desenhados para reduzir a quantidade de tempo necessária para verificar se um componente específico funciona, de forma a encontrar rapidamente as suas possíveis falhas antes que seja utilizado em um produto final. De forma a reduzir os custos na fabricação e melhorar o rendimento da produção, o ideal é avaliar estes componentes logo após a sua fabricação (TERADYNE, 2023).

O ATE é um elemento essencial para os testes de semicondutores que garante a sua qualidade e confiabilidade. Um ATE pode ir desde um simples multímetro digital controlado por um computador até um sistema complicado contendo dezenas de instrumentos complexos capazes de testar e diagnosticar falhas em componentes eletrónicos ou em testes

de *wafer* (verificam a funcionalidade dos circuitos eletrónicos individuais enquanto eles ainda estão na forma de *wafer*, antes de serem cortados em componentes individuais). O ATE é amplamente utilizado na indústria eletrónica, no processo de fabricação de componentes e sistemas eletrónicos após serem fabricados (Filho, 2022).

O funcionamento de um sistema ATE abrange várias etapas que desempenham um papel crucial em garantir a qualidade e a confiabilidade dos dispositivos semicondutores. A primeira etapa no funcionamento de um sistema ATE é o desenvolvimento de um programa de teste que define as condições de teste, parâmetros e critérios de aceitação para o dispositivo em teste. O programa de teste geralmente é criado pelo fabricante do dispositivo sendo escrito em uma linguagem de programação especializada que permite controlar o sistema ATE, e envolve a criação de um programa que define os parâmetros de teste, as condições e os critérios de aceitação para o dispositivo em teste, garantindo a precisão e a eficiência do processo de teste. O programa de teste interage com o sistema ATE, permitindo que ele controle os instrumentos de teste, sinais e aquisição de dados durante o processo de teste (Deshpande, Epili, Ghule, & Ratnaparkhi, 2023).

Após o desenvolvimento do programa de teste, o sistema ATE é configurado para testar o dispositivo em questão. Esta etapa envolve a configuração dos componentes de *hardware* do sistema ATE, os acessórios de teste, como *sockets* e pontas de teste, são ligados ao dispositivo para facilitar a conexão elétrica entre o dispositivo e o sistema de teste. Os sinais introduzidos ao dispositivo são controlados através do programa desenvolvido e aplicados em diversas condições de teste, como em altas e baixas temperaturas, diferentes frequências, de forma a verificar como se comporta o dispositivo em diferentes condições operacionais. O sistema ATE então mede as características elétricas e funcionais do dispositivo em teste usando instrumentos de teste especializados, como multímetros, osciloscópio, entre outros. As medições são comparadas com os critérios de teste definidos no programa de teste para determinar se o dispositivo atende aos critérios de aceitação. Os dados obtidos durante esse processo são analisados para otimizar o desempenho do dispositivo, identificar problemas de design e defeitos, e garantir a conformidade com padrões e especificações da indústria. Com os dados obtidos é possível realizar a caracterização dos componentes estudados, o principal objetivo disso é estabelecer os limites operacionais do dispositivo. Esses limites definem o intervalo de condições operacionais sob as quais podem funcionar de forma confiável, sem falhas e preservação do desempenho (Deshpande, Epili, Ghule, & Ratnaparkhi, 2023).

Por fim, o sistema ATE classifica o dispositivo em teste como aprovado ou reprovado com base nos critérios de aceitação especificados no programa de teste. Se o dispositivo atender aos critérios de aceitação, ele é classificado como aprovado e avança para a próxima etapa do processo de produção. Se o dispositivo não passar no teste, ele é classificado como reprovado, e o processo de produção é interrompido até que o problema seja resolvido (Deshpande, Epili, Ghule, & Ratnaparkhi, 2023).

O funcionamento de um sistema ATE é fundamental para garantir a qualidade e a confiabilidade dos dispositivos semicondutores. As várias etapas envolvidas no funcionamento de um sistema ATE, como o desenvolvimento do programa de teste, a configuração do teste, a aplicação de sinais, a medição, a análise de dados e a classificação do dispositivo, permitem testar e verificar com precisão a funcionalidade dos dispositivos semicondutores. Isso garante que apenas dispositivos de alta qualidade sejam produzidos,

melhorando a eficiência do processo de produção e aumentando a satisfação dos clientes (Deshpande, Epili, Ghule, & Ratnaparkhi, 2023).

3. *Hardware*

O *Hardware* é um termo em inglês que se utiliza quando nos queremos referir às partes físicas e tangíveis de um computador ou dispositivo eletrónico, ou seja, tudo aquilo em que podemos ver e tocar. No desenvolvimento deste projeto foram utilizados uma série de componentes essenciais para garantir um bom desempenho do circuito, componentes estes que iremos apresentar de seguida.

3.1 Microcontrolador Esp-32 S3 WROOM-1

O ESP32-S3-WROOM-1 é um módulo de desenvolvimento muito utilizado em projetos de eletrónica e IoT (*Internet of Things*) devido à sua versatilidade, potência de processamento e recursos de conectividade. Este módulo, desenvolvido pela *Espressif Systems*, faz parte da família ESP32-S3, que se destaca por oferecer um bom desempenho, ideal para aplicações que exigem um processamento rápido, conectividade avançada, e eficiência energética.

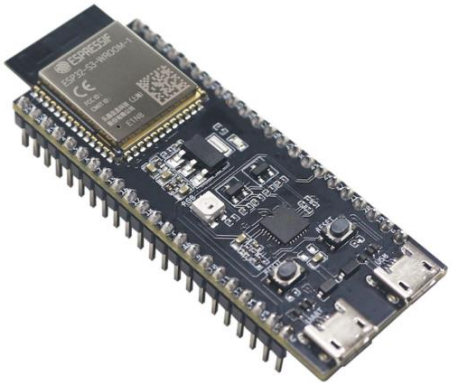

Figura 1: ESP32-S3-WROOM-1

3.1.1 Estrutura do *Hardware* do ESP32-S3-WROOM-1

A estrutura de *hardware* do ESP32-S3-WROOM-1 foi idealizada de forma a obter um equilíbrio ideal entre o desempenho, conectividade e eficiência energética, tornando-o uma solução versátil para uma variedade de aplicações em IoT e sistemas embebidos. Neste módulo encontramos componentes com alta qualidade, como um processador dual-core avançado, uma boa memória, interfaces de entrada e de saída diversificadas, e capacidades integradas de segurança e inteligência artificial, de modo a fornecer um dispositivo eficiente para o desenvolvimento de projetos tecnológicos. Abaixo vamos desenvolver de forma mais detalhada cada um dos constituintes do ESP32-S3-WROOM-1.

- Processador (CPU)

O ESP32-S3-WROOM-1 possui um processador dual-core Xtensa LX7, que funciona a uma frequência máxima de 240MHz. Este processador apresenta um bom desempenho na execução de tarefas complexas, como processamento de dados em tempo real, controlo de dispositivos, e execução de algoritmos de inteligência artificial (AI). A arquitetura dual-core

permite a execução de múltiplas tarefas em simultâneo, o que é essencial para projetos que exigem a execução de multitarefas.

- Memória

O ESP32-S3-WROOM-1 possui uma memória de 512Kb de SRAM (*Static Random Access Memory*) interna o que permite a execução de operações de processamento intensivo em tempo real uma vez que a SRAM é muito mais rápida que outros tipos de memória, o que significa que o processador pode aceder aos dados armazenados na SRAM quase instantaneamente, pois consegue aceder a qualquer parte desta memória rapidamente. Para além disso, integra uma memória Flash de 384 KB de *firmware* (O *firmware* é um tipo especial de *software* que é gravado permanentemente no *hardware*, contendo instruções básicas necessárias para o seu funcionamento). Esta combinação de memórias garante que o ESP consiga trabalhar com aplicações complexas, armazenando e processando grandes volumes de dados.

- Conectividade

O ESP32-S3-WROOM-1 apresenta capacidades de conectividade que o tornam ideal para projetos de IoT como Wi-Fi 802.11 b/g/n que funciona na faixa dos 2,4GHz e Bluethooth 5.0, o que permite conectividade com uma vasta gama de dispositivos que possuam Bluethooth, o que o torna ideal para aplicações que requerem baixo consumo de energia e comunicações a curtas distâncias.

- Interfaces de Entrada/Saída (I/O)

O ESP32-S3-WROOM-1 possui um total de 45 pinos de entrada/saída (GPIO), o que permite suporte a várias interfaces para controlar os periféricos, as interfaces suportadas incluem:

- 4 pinos de SPI (*Serial Peripheral Interface*);
- 2 pinos para I2C (*Inter-Integrated Circuit*);
- 3 pinos UART (*Universal Asynchronous Receiver-Transmitter*);
- 2 pinos para I2S (*Integrated Interchip Sound*);
- PWM (*Pulse Width Modulation*) que são multiplexados com outros pinos GPIO;
- ADC (*Analog to Digital Converter*), possui 20 canais ADC, permitindo a leitura de sinais analógicos;

Para além das interfaces mencionadas, os restantes pinos GPIO podem ser utilizados para outras funções, como controlo de relés, botões, LEDs, comunicar com outros dispositivos como sensores, displays, etc., o que o torna ideal para diferentes tipos de projetos. Na figura 2, podemos verificar os pinos referente ao ESP32-S3-WROOM-1.

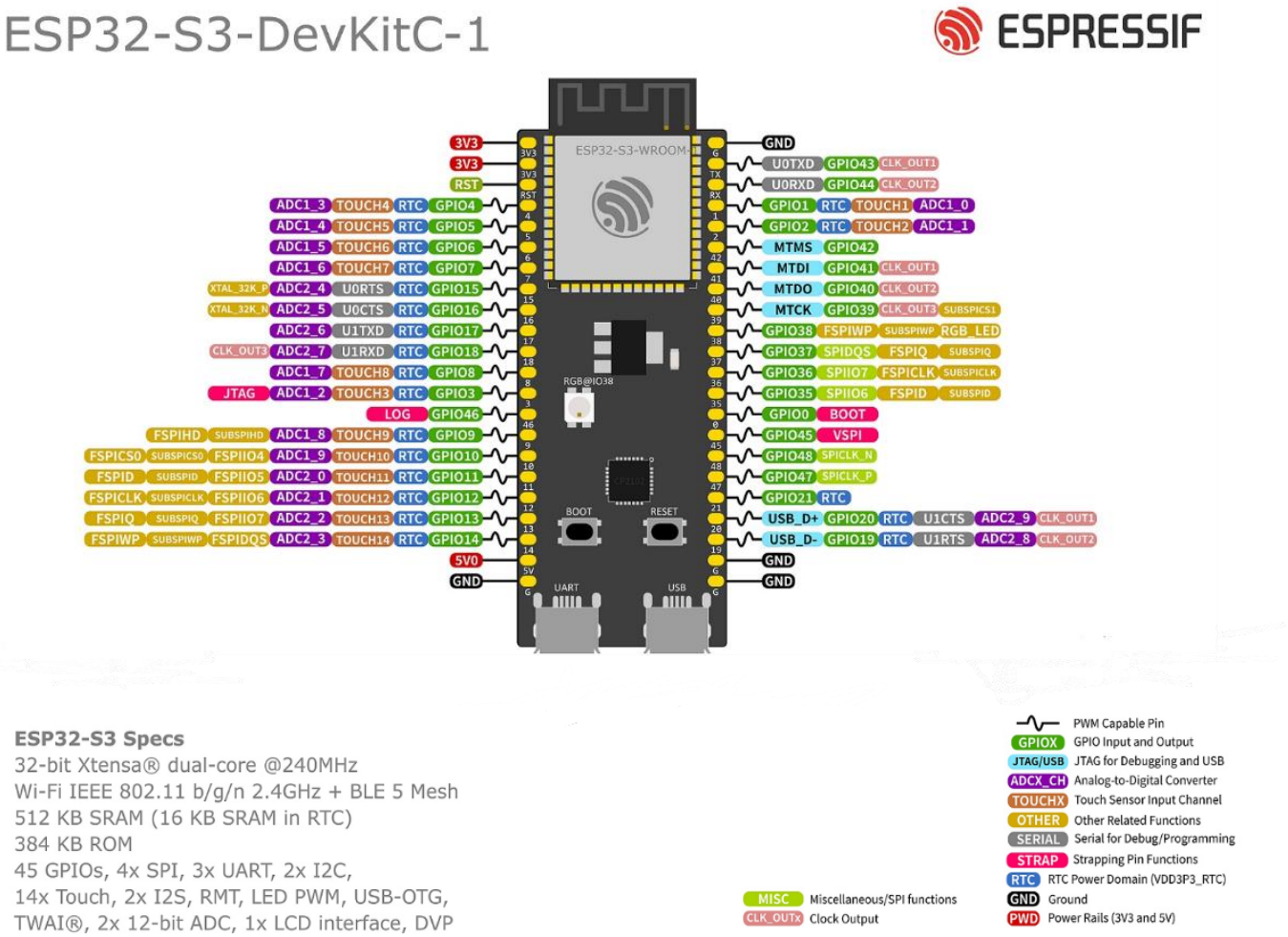

Figura 2: *Pinout* do ESP32 S3 WROOM-1

– Segurança

O ESP32-S3-WROOM-1 incorpora uma serie de recursos de segurança como Criptografia de *Hardware*, Arranque seguro (*Secure Boot*) e armazenamento seguro de chaves. Todas elas cruciais para proteger os dados e as comunicações quando é necessário proteger informações sensíveis ou operações críticas.

– Consumo de energia

O ESP32-S3-WROOM-1 foi desenvolvido de forma a funcionar com baixo consumo de energia. Por esse motivo, possui uma variedade de modos de funcionamento que permitem equilibrar o desempenho com o consumo de energia, tornando-o ideal para aplicações IoT e sistemas que necessitem de funcionar por um período prolongado sem acesso a fontes de energia constantes.

Devido às suas características o ESP32-S3-WROOM-1 torna-se ideal para uma grande variedade de aplicações, o que nos levou a escolhê-lo para o presente projeto. O que nos levou a escolher este modelo em particular, dentro da família ESP foi a quantidade de pinos disponíveis para entrada e saída, pois só para os reles seriam necessários 24 pinos GPIO disponíveis.

O datasheet do ESP32-S3-WROOM-1 pode ser consultado através do seguinte link: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

3.2 Relé SRD – 05VDC-SL-C

O relé SRD-05VDC-SL-C trata-se de um componente eletromecânico utilizado em circuitos de controlo e de automação. A sua principal função é permitir que um circuito de baixa potência controle um circuito de maior potência, funcionando como uma interface entre um sinal de controlo e outros elementos (Petruzella, 2014).

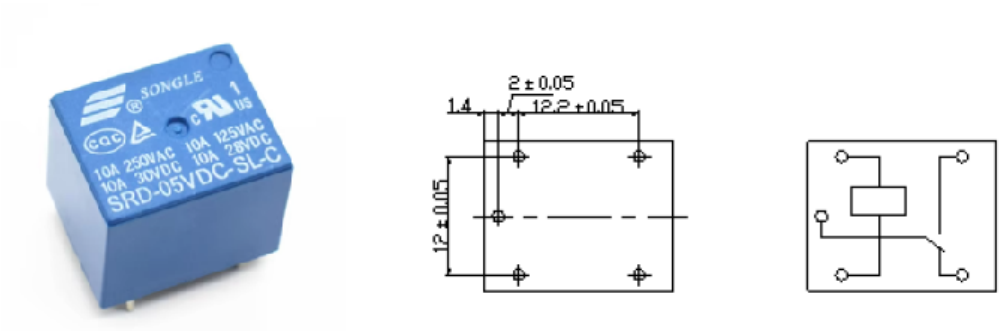

Figura 3: Relé SRD-05VDC-SL-C e pinout

O relé SRD-05VDC-SL-C é um relé de utilização geral, caracterizado pelas suas especificações que se encontram na tabela abaixo.

Tabela 1: Principais especificações técnicas do relé SRD-05VDC-SL-C

Tensão de funcionamento da bobine	5V DC
Corrente da Bobine	Aproximadamente 70mA
Tensão máxima de Comutação	250V AC / 30V DC
Corrente Máxima de Comutação	10A

Configuração	SPDT (*Single Pole Double Throw),* o que significa que possui um contato comum (COM), um normalmente aberto (NO) e um normalmente fechado (NC)
Tempo de ativação	≤ 10 ms
Tempo de desativação	≤ 5 ms

O relé SRD-05VDC-SL-C funciona com uma bobine, quando alimentado a +5V, cria um campo magnético que atrai uma alavanca interna, mudando a posição dos contactos. Por norma antes de a bobine do relé ser alimentada a +5V o contato do rele encontra-se no NO (*Normally Open* ou normalmente aberto) após a receção de energia a bobine gera um campo magnético que faz com que o contato se mova para a posição de NC (*Normally Closed* ou normalmente fechado) permitindo então que a corrente siga através do NC, quando deixa de ser alimentada, o campo magnético desaparece e o contacto volta á posição NO como inicialmente (Malvino & Bates, 2016).

Este relé em particular foi escolhido por ser compatível com o sistema de controlo escolhido, nomeadamente o ESP32-S3, como funciona com uma tensão de 5V é adequado para ser controlado diretamente pelo ESP utilizado. Algo importante também, esta associado à questão de o relé conseguir lidar com uma corrente de até 10 Amperes, até 250 Volts em tensão alternada (AC) ou até 30 Volts em tensão continua (DC), permitindo uma margem de segurança elevada para o nosso projeto. Aliados a estas características, temos ainda o fato de ser um componente de baixo custo e fácil de encontrar no mercado, e a sua fiabilidade e simplicidade de utilização.

3.3 Díodo 1N4002

No projeto realizado foram utilizados 12 díodos 1N4002, estes desempenham um papel importante na retificação da corrente e na proteção do circuito contra possíveis danos causados por polaridade reversa.

Figura 4: Díodo 1N4002

O díodo 1N4002 é um díodo retificador de silício muito utilizado em circuitos eletrónicos para aplicações com baixa potência. Este díodo faz parte da série 1N400x, uma linha de díodos retificadores conhecidos pela sua robustez e confiabilidade em uma vasta gama de aplicações. Na tabela abaixo podemos encontrar algumas especificações técnicas deste díodo.

Tabela 2: Principais especificações técnicas do díodo 1N4002

Peak Reverse Voltage	100 V
Continuous Forward Current	1 A
Maximum Surge Current	30 A
Forward Voltage	Entre 0,7V e 1V
Encapsulamento	DO-41
Temperatura de operações	-65°C a 150°C

Estas são algumas das especificações que tornam este díodo adequado para várias aplicações em circuitos eletrónicos, onde a fiabilidade e o desempenho são essenciais.

Neste projeto em particular, o díodo foi utilizado entre os terminais do relé, funcionando como um "díodo de *flyback*", sendo a sua principal função proteger os outros componentes do circuito contra picos de tensão que podem acontecer quando o relé é ativado e/ou desativado.

3.4 Transístor BC546C

O BC546C é um transístor de junção bipolar (BJT) de baixa potência muito utilizado em circuitos eletrónicos. Este transístor pertence á família dos transístores NPN, sendo conhecido pela sua versatilidade em diferentes aplicações. O BC546C possui uma série de especificações que o tornam adequado para uma variedade de fins em eletrónica, onde podemos destacar as que se encontram na tabela 3.

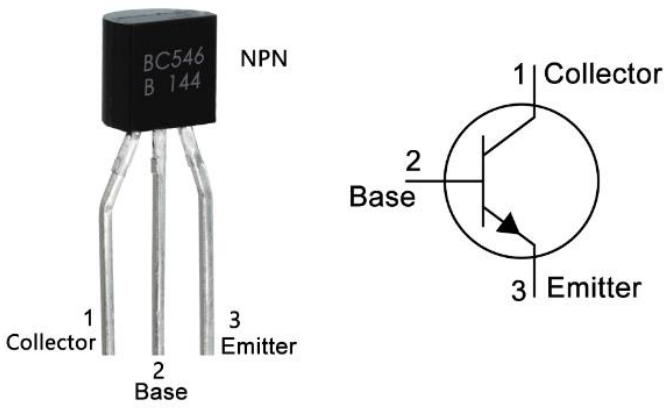

Figura 5: Transístor BC546C pinout

Tabela 3: Principais especificações técnicas do transístor BC546C

Tipo	NPN
Corrente máxima do coletor	100mA
Tensão máxima coletor-emissor	65V
Tensão máxima coletor-base	80V
Tensão máxima emissor-base	6V
Frequência de transição	300MHz
Potência máxima dissipada	500mW

O BC546C sendo um transístor NPN funciona com três terminais principais, o coletor (C), base (B) e o emissor (E). O funcionamento básico de um transístor é muito simples, um pequeno sinal de corrente aplicado na base do transístor controla a corrente entre o coletor e o emissor. O que significa que o transístor pode amplificar sinais, já que a corrente do coletor é proporcional á corrente base multiplicada pelo ganho da corrente. Por outro lado, quando a corrente base é suficiente para saturar o transístor, ele conduz totalmente (saturação), quando não há corrente base, o transístor não conduz (corte).

O BC546C pode ser utilizado em diversas aplicações devido às suas características de baixa potência e alta frequência, sendo as aplicações mais comuns a amplificação de sinais, comutação de baixa potência quando é necessário controlar dispositivos de baixa potência como LEDs, relés, entre outros, proteção de circuitos de forma a desligar ou a limitar o funcionamento de circuitos principais em caso de picos de tensão ou corrente, entre outras aplicações (Platt, 2012).

Neste projeto utilizamos o transístor BC546C como um interruptor de forma a controlar a ativação dos relés, isto foi possível uma vez que conduz a corrente quando a base está em um potencial mais baixo do que o emissor. Desta forma, quando a corrente flui pela base do transístor, permite a circulação de corrente entre o emissor e o coletor, fechando o circuito e ativando o relé. Como tal, podemos indicar que o transístor atua como um interruptor elétrico, controlado pela tensão no seu terminal de base, o que se tornou importante para automatizar o acionamento dos relés utilizados.

3.5 Circuito Integrado 74HCT00

O 74HCT00 é um circuito integrado (CI) digital que pertente á família HCT (*High-Speed CMOS with TTL compatibility*). Este CI contém 4 portas NAND de duas entradas, conhecidas pela sua rápida comutação e baixo consumo de energia, características que o tornam ideal para uma variedade de aplicações e circuitos lógicos digitais. Cada uma destas portas NAND é capaz de realizar operações lógicas NAND, no entanto, no contexto deste projeto, foi adaptado para uma função mais especifica, foi utilizada como uma porta NOT, uma vez que a porta NAND é considerada uma porta universal, ou seja, com este tipo de porta podemos implementar todos os outros tipos de porta lógica

Tabela 4: Principais especificações técnicas do CI 74HCT00

Tensão de alimentação (VCC)	4.5V a 5.5V
Tempo de propagação	Aproximadamente 14ns (nanossegundos)
Corrente de saída máxima	25mA por porta
Consumo de corrente	Muito baixo (típico dos circuitos CMOS)
Impedância de entrada	Alta
$V_{IH\ min}$	2V
$V_{IL\ max}$	0.8V

Na tabela acima podemos verificar algumas das especificações técnicas que levaram a escolha deste CI para o projeto. É de destacar o valor de $V_{IH\ min}$ (*High-level input voltage*) que se refere ao nível de tensão mínima para que o circuito reconheça um sinal como nível alto (1) e $V_{IL\ max}$ (*Low-level input voltage*) da tabela, que corresponde ao nível de tensão máximo para que o circuito reconheça o sinal como nível baixo (0). Isto foi essencial uma vez que este circuito integrado funciona como uma interface crítica entre o sistema digital de 3.3V (ESP) e o sistema de 5V, onde os circuitos integrados são testados. A capacidade do 74HCT00 de receber sinais de entrada de 3.3V e funcionar a 5V permitiu que o ESP32 controle diretamente os circuitos, evitando problemas de compatibilidade de tensão. Para além disso, a baixa corrente de entrada do 74HCT00 permite proteger os pinos do ESP32, que possuem uma limitação na capacidade de corrente, minimizando a carga nos GPIOs sem a necessidade de utilizar amplificadores adicionais. Essa interface permite uma comunicação estável e eficiente entre os dois sistemas de tensão diferentes, garantindo que o projeto funcione sem falhas devido a incompatibilidades de nível lógico.

A resposta rápida do 74HCT00 também permite que os sinais enviados pelo ESP32 sejam processados com baixa latência, um ponto importante para o controlo preciso dos relés. Desta forma, o 74HCT00 não só facilita a comunicação entre diferentes sistemas de tensão, mas também garante a confiabilidade e a velocidade do controlo.

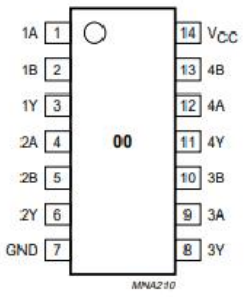

Figura 6: 74HCT00 e Pinout

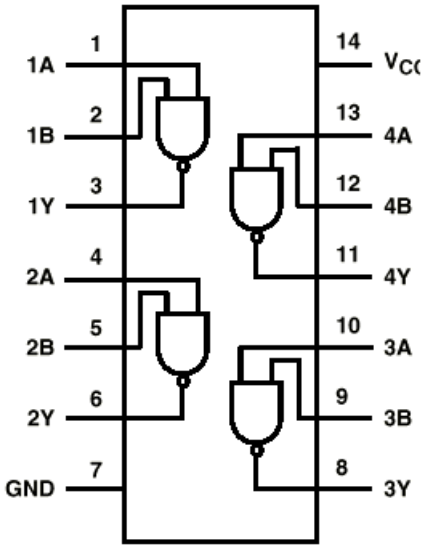

Figura 7: Esquema funcional do 74HCT00

A utilização do 74HCT00 no projeto foi estratégica para garantir a integridade dos sinais lógicos enviados pelo microcontrolador ESP às portas lógicas. A capacidade deste circuito integrado de proteger contra ruídos, isolar sinais e compatibilizar níveis lógicos, assegurou que as operações lógicas fossem realizadas de forma confiável e precisa.

3.6 Circuitos Integrados

Os circuitos digitais são projetados para produzir tensões de saída e responder a tensões de entrada que estejam dentro do intervalo determinado para os binários 0 e 1. Praticamente todos os circuitos digitais existentes são circuitos integrados (CI), o que permitiu a construção de sistemas digitais menores e mais confiáveis do que aqueles construídos com circuitos lógicos discretos.

Um circuito integrado (CI) é um componente eletrónico que combina vários componentes como transístores, resistências e díodos em um único chip de silício. O fato destes componentes serem microscópicos permite a construção de circuitos eletrónicos mais complexos e eficientes, ocupando menos espaço físico e consumindo menos energia. CIs são amplamente utilizados em sistemas digitais, desde computadores até dispositivos móveis e equipamentos de automação, desempenhando um papel crucial no processamento de dados e controlo lógico (Malvino & Brown, 1993).

Os CI podem ser classificados em diferentes categorias de acordo com o numero de portas lógicas ou transístores que possuam, nomeadamente, em pequena escala temos os SSI (*Small-scale integration*) com menos de 12 portas no mesmo componente, em média escala temos os MSI (*Medium-scale integration*) que contem entre 12 a 100 portas no mesmo componente, em grande escala temos os LSI (*Large-scale integration*) que se refere aos componentes com mais de 100 portas e por fim quando temos os VLSI (*Very-Large-Scale Integration*) os circuitos com

milhões portas lógicas em um só componente. Estas classificações definem o nível de complexidade e capacidade de processamento de cada CI (Tocci, Widmer, & Moss, 2007).

3.6.1 Portas Lógicas

Este projeto teve como intuito realizar um sistema de testes para componentes eletrónicos, dentro destes foram escolhidos os CI para fazer os testes, nomeadamente, portas lógicas básicas, pelo que iremos falar sobre estas de seguida.

As portas lógicas são os blocos fundamentais dos sistemas digitais, são componentes que realizam operações lógicas sobre um ou mais sinais de entrada binários para produzir um sinal de saída, também binário. Essas operações seguem as regras da álgebra booleana, que se baseia no uso de dois estados, geralmente representados por 0 (falso) e 1 (verdadeiro). As portas lógicas são cruciais para a construção de circuitos digitais, sendo responsáveis pela execução de operações aritméticas, controlo de fluxo, e tomada de decisões lógicas em sistemas eletrónicos. Dentro dos principais tipos de portas lógicas temos as portas AND, OR, NOT, NAND, NOR, XOR e XNOR. Estas portas lógicas podem ser combinadas em diferentes configurações como somadores, multiplexadores e flip-flops, que são essenciais nos sistemas digitais (Wakerly, 2006).

Dentro dos CI digitais, existem várias famílias de portas lógicas, classificadas de acordo com a tecnologia utilizada para implementar as operações lógicas. Nas famílias mais importantes está a TTL (*Transistor-Transistor Logic*), que utiliza transístores para processar sinais digitais. Neste projeto foram utilizadas portas lógicas da família 74LS (*Low Power Schottky*) esta é uma variante específica da tecnologia TTL que utiliza díodos Schottky (são uma espécie de "válvulas unidirecionais" para a corrente elétrica, permitem que a corrente flua facilmente em uma direção, mas bloqueia essa mesma corrente na direção oposta), para reduzir os tempos de comutação dos transístores e, ao mesmo tempo, diminuir o consumo de energia. As portas 74LS foram desenvolvidas para resolver as limitações de velocidade e eficiência das versões originais do TTL, proporcionando uma solução intermediária entre o desempenho e o consumo (Tocci, Widmer, & Moss, 2007). Entre algumas das características das portas lógicas 74LS podemos destacar o baixo consumo de energia, velocidade moderada e a tensão de funcionamento (normalmente funcionam com uma tensão de 5V). Este tipo de portas lógicas é muito utilizado em circuitos que exijam um bom equilíbrio entre desempenho e eficiência energética.

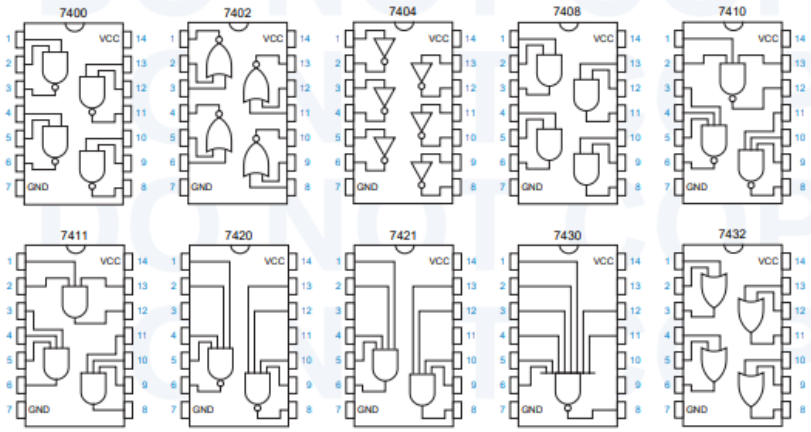

Figura 8: Diagrama de pinos de algumas portas lógicas da família 74LS

Neste projeto foram utilizadas portas lógicas da série 74LS para os testes, que são CI que realizam funções lógicas básicas. Cada uma das portas lógicas utilizadas realizam operações especificas na álgebra booleana, sendo essenciais para o funcionamento de sistemas digitais complexos. Posto isto, iremos de seguida falar sobre as portas utilizadas especificamente para os testes deste projeto.

Estas portas lógicas da família 74LS funcionam em um intervalo de tensão de 4.75V a 5.25V. O consumo de corrente é relativamente baixo, sendo adequado em circuitos digitais onde a eficiência energética é importante. O seu tempo de propagação é de aproximadamente 10ns e é caracterizado pela impedância de saída, pois apresenta alta impedância em estado aberto, que permite que o CI não interfira com os outros componentes a que esta ligado.

Tabela 5: Valores típicos de entrada e saída para a família TTL (casos extremos de funcionamento)

TTL – 74LS	
V_{IH} (min)	2.0 V
V_{IL} (máx)	0.8 V
V_{OH} (min)	2.4 V
V_{OL} (máx)	0.5 V
I_{IH} (máx)	20 µA
I_{IL} (máx)	400 µA
I_{OH} (máx)	0.4 mA
I_{OL} (máx)	8 mA

Na tabela 5 é possível verificar alguns dos valores típicos de entrada e de saída para as os circuitos integrados da família TTL 74LS. Verificamos que o valor mínimo do sinal de entrada para que seja considerado alto (1) será no mínimo de 2.0V, enquanto para que seja considerado como baixo (0) o valor máximo devera ser de 0.8V. O valor de V_{OH} (*Voltage*

Output High), que diz respeito ao nível de tensão mínimo que a saída do CI irá gerar quando tiver em nível lógico alto (1) é de 2.4V enquanto o V_{OL} (*Voltage Output Low*), ou seja, o valor máximo de tensão que o CI vai gerar quando o nível lógico for baixo (0) será no máximo de 0.5V. Por sua vez, quanto às correntes também temos alguns valores padrão para esta família de circuitos integrados, verificamos que o valor de I_{IH} (*Input High Current*) que diz respeito ao valor máximo da corrente de entrada quando o valor lógico da entrada é 1 é no máximo de 20μA, em oposição o valor máximo da corrente de entrada quando o nível logico é 0, I_{IL} (*Input Low Current*) é de 400 μA. Por fim, temos o valor de I_{OH} (*Output High Current*) que é de 0.4mA, que nos indica que a corrente máxima de saída quando está em um nível lógico alto (1), e o valor de I_{OL} (*Output Low Current*) que diz respeito ao valor máximo de corrente que a saída pode fornecer quando estiver em nível lógico baixo (0) que será de 8mA.

3.6.1.1 AND

Como AND utilizamos o CI 74LS08 que é composto por quatro portas lógicas AND de duas entradas.

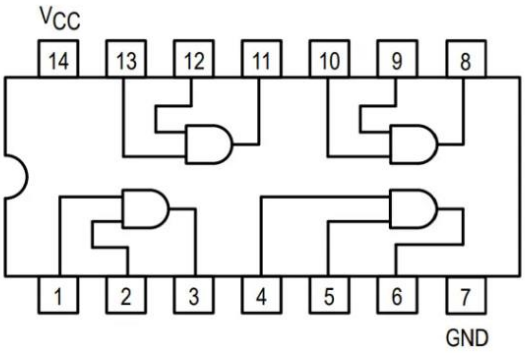

Figura 9: Pinout da porta lógica 74LS08

Tabela 6: Tabela de verdade da porta AND

Entrada A	Entrada B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

A porta AND realiza a operação lógica que só resulta em uma saída verdadeira (1) quando todas as entradas forem verdadeiras (1). Caso contrário, a saída será falsa (0).

Em termos matemáticos, a operação da porta AND pode ser representada como Saída = A ·B, onde A e B são entradas da porta.

3.6.1.2 NAND

O Circuito Integrado 74LS00 foi utilizado como porta NAND e contém quatro portas lógicas NAND de duas entradas

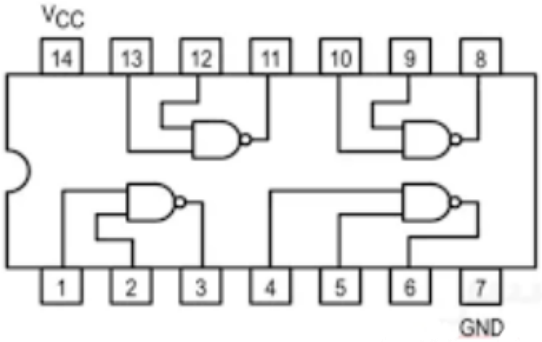

Figura 10: Pinout da porta lógica 74LS00

As portas NAND realizam a operação lógica NAND, que é a negação da operação AND. A tabela verdade para uma porta NAND é a seguinte:

Tabela 7: Tabela de verdade da porta NAND

Entrada A	Entrada B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

A porta NAND realiza a negação da operação AND. A saída será falsa (0) somente quando todas as entradas forem verdadeiras (1). Se qualquer uma das entradas for falsa (0), a saída será verdadeira (1).

Em termos matemáticos, a operação da porta NAND pode ser representada como Saída = $\overline{A \cdot B}$, onde A e B são entradas da porta.

3.6.1.3 XOR

O CI 74LS86 foi utilizado no projeto como XOR, e fornece funções lógicas fundamentais em circuitos digitais. Especificamente, o 74LS86 contém quatro portas lógicas XOR (Exclusive OR) de duas entradas. As portas XOR são utilizadas para realizar operações de comparação e adição.

As portas XOR realizam a operação lógica XOR, que é uma forma de "ou exclusivo". A tabela verdade para uma porta XOR é a seguinte:

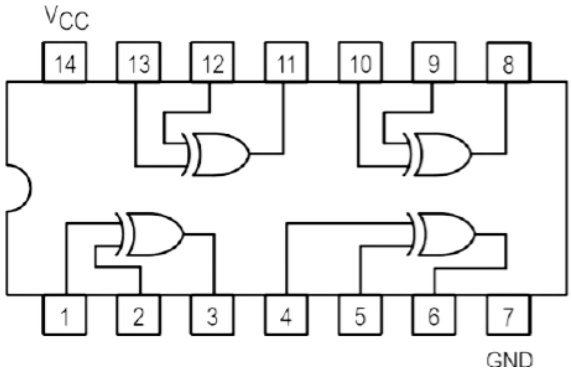

Figura 11: Pinout da porta lógica 74LS86

Tabela 8: Tabela de verdade da porta XOR

Entrada A	Entrada B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

A porta XOR realiza a operação "OU Exclusivo", ou seja, a saída será alta (1) somente quando as entradas forem diferentes (ou seja, uma entrada for alta e a outra for baixa). Se ambas as entradas forem iguais, a saída será baixa (0).

Esta operação pode ser representada matematicamente por Saída $=$ A $\oplus$ B, onde A e B são entradas da porta.

3.6.1.4 XNOR

O CI 74LS266 foi utilizado para implementar a função XNOR, que se trata de uma porta lógica complementar a XOR. Este é composto por quatro portas lógicas XNOR (Exclusive NOR) de duas entradas. Este tipo de portas é utilizado para realizar operações de igualdade e comparação.

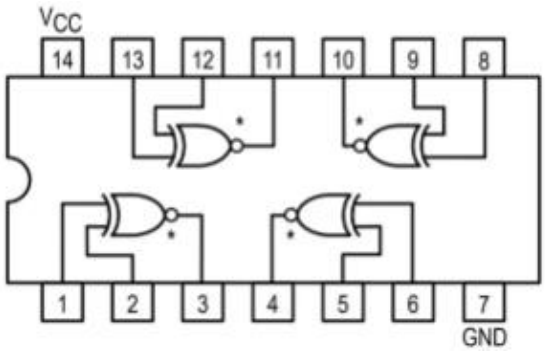

Figura 12: Pinout da porta lógica 74LS266

Tabela 9: Tabela de verdade da porta XNOR

Entrada A	Entrada B	Saída
0	0	1
0	1	0
1	0	0
1	1	1

A operação lógica XNOR pode ser descrita como um "ou exclusivo-negado". A saída de uma porta XNOR será verdadeira quando ambas as entradas forem iguais, ou seja, ambas 0 ou 1. Pelo que a saída será falsa quando as entradas forem diferentes. Esta característica faz com que as portas XNOR sejam ideias para realizar comparações lógicas simples, por exemplo, quando é necessário detetar quando duas entradas são iguais.

A expressão matemática da porta XNOR pode ser representada por Saída= A ⊙B, onde A e B são entradas.

3.6.1.5 OR

Como porta lógica OR foi utilizado o CI 74LS32, é conhecido por fornecer quatro portas lógicas OR de duas entradas, que são essenciais para diversas operações lógicas em circuitos digitais.

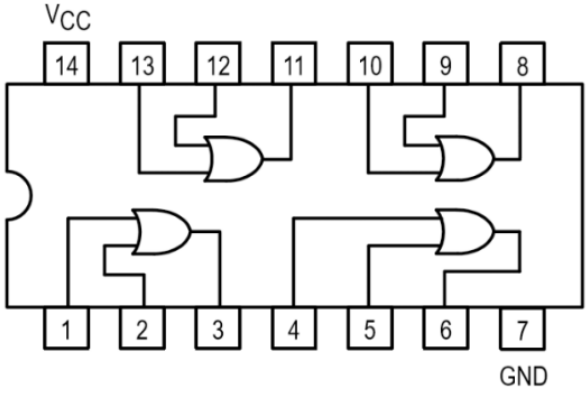

Figura 13: Pinout da porta lógica 74LS32

Tabela 10: Tabela de verdade da porta OR

Entrada A	Entrada B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

A porta OR realiza a função lógica "OU", onde a saída será verdadeira (1) se qualquer uma das entradas for verdadeira (1). Se ambas as entradas forem falsas (0), a saída será falsa (0).

A operação matemática da porta OR pode ser representada matematicamente por Saída = $A + B$, onde A e B são entradas da porta.

3.6.1.6 NOR

Como porta lógica NOR foi utilizado o CI 74LS02, este CI contém quatro portas lógicas NOR de duas entradas.

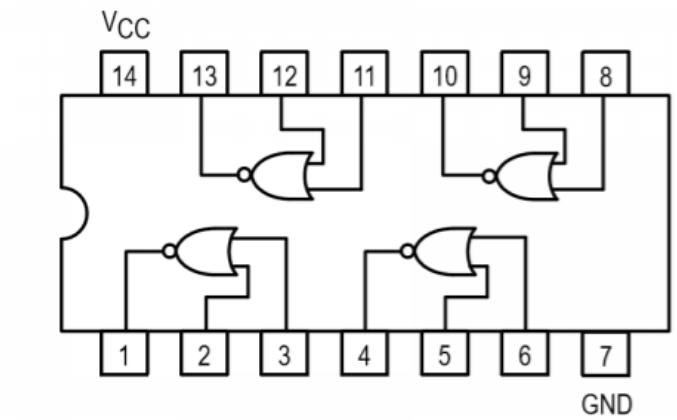

Figura 14: Pinout da porta lógica 74LS02

Tabela 11: Tabela de verdade da porta NOR

Entrada A	Entrada B	Saída
0	0	1
0	1	0
1	0	0
1	1	0

A porta NOR realiza uma combinação das operações lógicas OR e NOT. Ela produz uma saída falsa (0) se qualquer uma das entradas for verdadeira (1). Somente quando todas as entradas são falsas (0), a saída será verdadeira (1).

Em termos matemáticos, a operação da porta NOR pode ser representada como Saída = $\overline{A + B}$, onde A e B são entradas da porta.

3.6.1.7 NOT

Como porta NOT foi utilizado o CI 74LS04, este contém seis portas lógicas NOT independentes. A porta NOT, também conhecida como inversor, é uma das portas lógicas mais básicas e realiza a inversão de sinal.

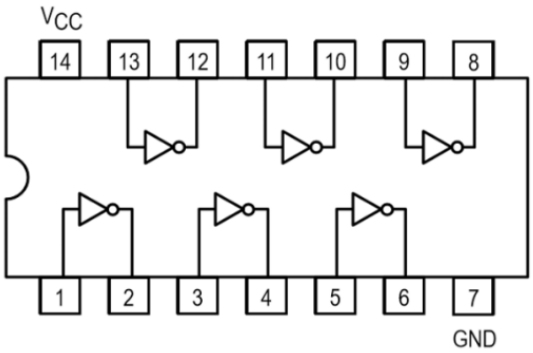

Figura 15: Pinout da porta lógica 74LS04

Tabela 12: Tabela de verdade da porta NOT

Entrada A	Saída
0	1
1	0

A porta NOT realiza a operação lógica de negação ou inversão da entrada. Ela inverte o estado lógico da entrada, transformando um 1 em 0 e um 0 em 1, ou seja, quando a entrada é verdadeira (1), a saída será falsa (0) e, quando a entrada é falsa (0), a saída será verdadeira (1).

Em termos matemáticos, a operação da porta NOT pode ser representada como Saída = $\overline{A}$, onde A é entrada da porta

3.7 Interfaces mecânicas

No desenvolvimento de circuitos eletrónicos e dos respetivos testes, existe muitas vezes a necessidade de colocar e retirar componentes constantemente, de forma a facilitar este processo foi utilizado um ZIF (*Zero Insertion Force*) *Socket* de 20 pinos, de forma a simplificar o processo.

Figura 16: ZIF Socket de 20 pinos

O ZIF *socket* permite colocar e retirar os componentes eletrónicos sem a necessidade de aplicar força significativa, ao contrário dos *sockets* tradicionais, onde os pinos do componente precisam ser pressionados para se conectarem aos terminais, o ZIF *socket* utiliza um mecanismo de alavanca que abre os contactos internos, permitindo que o componente seja inserido ou removido com facilidade.

O ZIF *socket* de 20 pinos foi pensado de forma a permitir componentes até 20 pinos, sendo por isso uma boa opção para o teste de Circuitos Integrados como as portas lógicas. É conhecido também pela sua durabilidade, suportando múltiplas inserções e remoções sem desgaste significativo, sendo também de fácil utilização, a alavanca que o *socket* possui permite inserir e remover os componentes rapidamente e sem esforço, minimizando o risco de danos aos pinos do CI.

3.8 LEDs

Os LEDs (*Light Emitting Diodes*) foram utilizados neste projeto como indicadores visuais do estado das portas lógicas, ou seja, quando a porta lógica está funcional é ativado o LED verde, caso não esteja funcional é acionado o LED vermelho. Os LEDs são componentes eletrónicos semicondutores que emitem luz quando é atravessado por uma corrente elétrica, apresentando vantagens como baixo consumo de energia, tamanho reduzido, alta durabilidade e respostas rápidas.

Figura 17: Leds utilizados

Tal como num díodo comum o LED tem um ânodo e um cátodo que precisam de ser adequadamente polarizados para que funcionem corretamente. Do lado de fora de um LED com encapsulamento de plástico típico, existe um corte plano que indica o lado do cátodo (figura 18) e o material semicondutor utilizado na sua fabricação é que ira determinar as suas características essenciais.

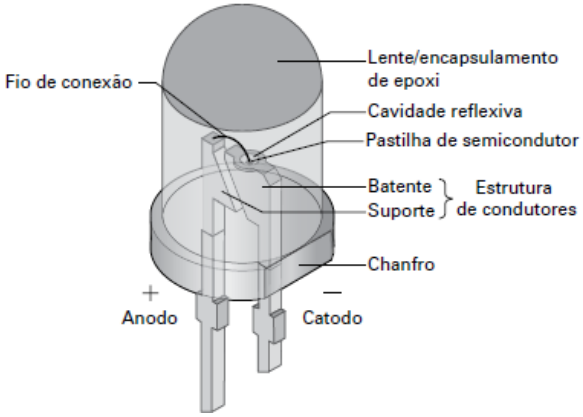

Figura 18: Partes constituintes de um LED

Na figura 19 podemos verificar uma fonte conectada a uma resistência em série com um LED. As setas para fora simbolizam a luz irradiada. Quando um LED é polarizado diretamente, os eletrões livres atravessam a junção *pn* e recombinam-se nas lacunas. Á medida que os eletrões passam de um nível de maior energia para um menor, eles irradiam energia na forma de fotões. Enquanto nos díodos comuns esta energia é irradiada sobre a forma de calor em um LED, a energia é libertada sobre a forma de luz. Este efeito é conhecido como eletroluminescência (Malvino & Bates, 2016).

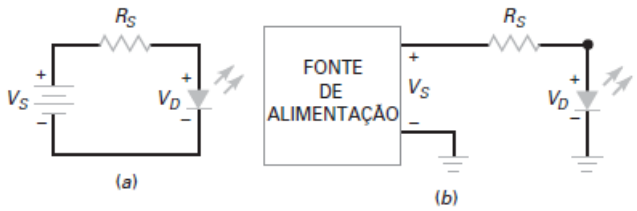

Figura 19: Circuito básico com um LED

A cor da luz, que corresponde ao comprimento de onda de energia dos fotões, é determinada pelo *gap* nas bandas de energia dos materiais semicondutores que são utilizados. Quando são utilizados elementos como o gálio, arsénio e fósforo, um fabricante pode produzir LEDs que irradiam luz vermelha, verde, amarela, azul, laranja, branca ou infravermelha (invisível) (Capuano, 2008).

4. *Software*

O *Software* refere-se a um conjunto de instruções, dados ou programas que são utilizados por dispositivos eletrónicos, como computadores e microcontroladores, para realizar tarefas específicas. Diferente do *hardware*, que diz respeito aos componentes físicos, o *software* é a parte intangível que define como o *hardware* deve comportar-se. Ele pode ser dividido em várias categorias, como sistemas operacionais, aplicações e ferramentas de desenvolvimento, cada um com funções e propósitos distintos.

No contexto de sistemas embebidos e microcontroladores, o *software* desempenha um papel fundamental na programação e controlo dos dispositivos. Ferramentas de desenvolvimento, como o Arduino IDE e o EasyEDA, que foram utilizados neste projeto, são utilizados para escrever, compilar e transferir o código para o microcontrolador, permitindo que ele execute as funções pretendidas e o segundo para o desenho do esquema elétrico e da Placa de circuito impresso (PCB).

Essas aplicações desempenharam papéis complementares, garantindo tanto o correto funcionamento lógico do projeto quanto o seu *design* físico, permitindo a integração de *hardware* e *software* de forma eficiente no projeto.

4.1 Arduino IDE

O Arduino IDE (*Integrated Development Environment*) foi o *software* escolhido para o desenvolvimento do código no projeto, especialmente pela sua simplicidade, versatilidade e compatibilidade com uma ampla gama de microcontroladores, incluindo o ESP32-S3 que foi o utilizado no desenvolvimento deste projeto. O IDE oferece uma interface simples que facilita a programação, compilação e upload de código para o microcontrolador, tornando-o ideal para atingir o objetivo deste projeto.

Figura 20: Logotipo do Arduino IDE

O Arduino IDE fornece uma série de funcionalidades úteis, para o desenvolvimento do código do projeto como o suporte a várias bibliotecas, uma vez que é adicionar facilmente bibliotecas específicas caso exista essa necessidade. É uma plataforma de código aberto, por ser *open-source*, facilita a solução de problemas e a obtenção de exemplos de código para diversas aplicações. E é compatível com vários microcontroladores, pois apesar de ter sido originalmente desenvolvido para Arduíno, o IDE suporta uma vasta gama de microcontroladores, como o ESP32-S3.

No projeto, o Arduino IDE foi utilizado para desenvolver e testar todo o código que executa no ESP32-S3, incluindo controlo das entradas e saídas digitais e a sua leitura. O IDE permitiu a programação através da linguagem C++, com a utilização de bibliotecas específicas para o ESP32, o que facilitou a implementação das funcionalidades desejadas. Associado a isto, através do monitor serial do IDE foi possível realizar o *debug* e verificar a comunicação entre o microcontrolador e outros componentes utilizados. Isso possibilitou ajustes no código, sempre que foi necessário, a interação do utilizador com o programa e a leitura dos resultados dos testes realizados.

4.2 EasyEDA

O EasyEDA foi o *software* escolhido para o desenvolvimento e design do esquema elétrico e do PCB do projeto. O EasyEDA é uma aplicação muito utilizada por engenheiros e para desenhar circuitos, realizar simulações e gerar arquivos prontos para fabricação de PCBs. Apresenta uma interface simples, combinada com um vasto conjunto de bibliotecas de componentes, o que o torna uma escolha acessível e eficiente para projetos com vários níveis de complexidade.

Figura 21: Logotipo do EasyEDA

O EasyEDA oferece diversas funcionalidades que ajudaram o desenvolvimento do projeto, como os desenhos esquemáticos, uma vez que permite desenhar esquemas detalhados, conectando componentes eletrónicos de forma visual e intuitiva. Permite ainda o design do PCB como falamos anteriormente, pois a partir do esquema elétrico o *software* permite criar o *layout* do PCB, posicionando componentes e realizando as conexões com base no projeto. Isto é possível devido á existência de uma biblioteca que disponibiliza inúmeros componentes pré-definidos como resistências, condensadores, relés, microcontroladores, o que foi um fator decisivo na escolha deste *software*. Ainda permite simular os circuitos desenhados, o que permite testar o comportamento do mesmo antes do seu fabrico, evitando erros na fase de implementação física. E por fim, este cria os arquivos Gerber, que são padrões industriais utilizados para enviar o design do PCB a fabricantes para produção.

Neste projeto o EasyEDA foi utilizado para criar o esquema elétrico do projeto e o PCB, através das suas funcionalidades de desenho e simulação, foi possível projetar de forma eficiente as ligações entre o ESP32-S3 e os outros componentes. Embora o PCB não tenha sido reproduzido fisicamente, a utilização do EasyEDA garantiu que todas as etapas de design

fossem realizadas com precisão, permitindo a futura produção da placa de circuito impresso sem a necessidade de alterações significativas. Este programa mostrou-se ser muito acessível e útil para o desenvolvimento do projeto.

5. Implementação Prática

Neste capítulo abordamos a fase crucial do projeto, nomeadamente a sua implementação, onde as ideias e os conceitos teóricos adquiridos ganharam forma física em um circuito funcional. Esta fase envolveu a montagem do circuito elétrico, o programa de teste e a validação do seu desempenho conforme o planeado.

Sendo que o objetivo principal foi realizar um circuito de teste para portas lógicas, começamos por planear o circuito elétrico consoante os materiais à nossa disposição, desenhamos o esquema elétrico, procedemos à sua montagem, escrevemos o código de teste e começamos a realizar os testes de forma a realizar as alterações necessárias no programa e no circuito sempre que necessário. Estas diferentes fases serão desenvolvidas nos subcapítulos seguintes.

5.1 Esquema Funcional

Neste capítulo podemos encontrar o esquema funcional associado ao nosso projeto, no entanto, em primeiro lugar podemos verificar o funcionamento básico de cada pino, quando está associado a uma entrada ou a uma saída do CI em teste, uma vez que foi necessário criar um circuito de interface entre o pino do ESP32 e o ZIF *Socket*, de modo que o encaminhamento dos sinais seja de acordo com o objetivo do pino. Desta forma, o diagrama de ligação do pino do ESP a 1 pino do ZIF *Socket* é o seguinte:

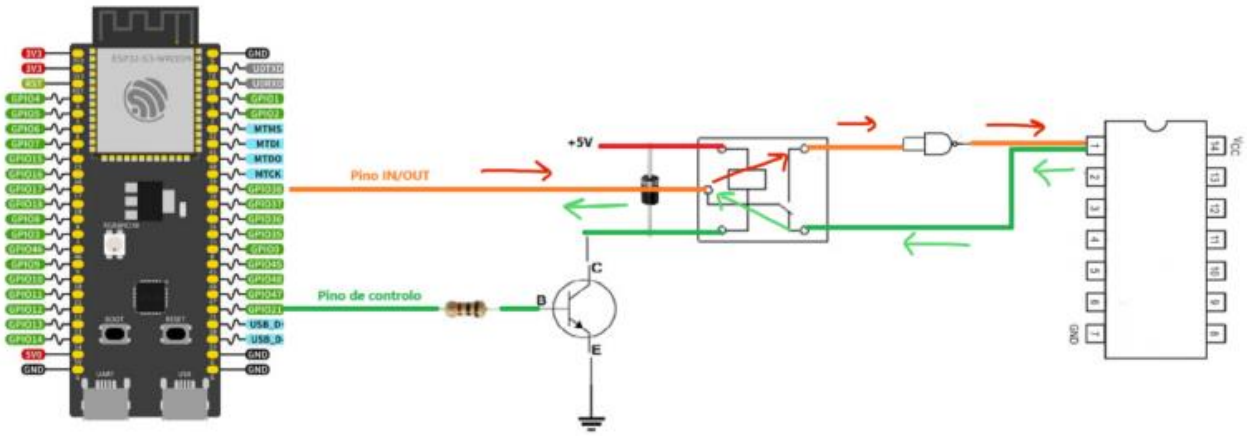

Figura 22: Esquema Funcional para um pino IN/OUT

Analisando a figura 22 podemos verificar que o relé permite o encaminhamento do sinal de acordo com o pino a estudar, caso seja um pino de entrada da porta lógica, o pino de controlo é ativado provocando a comutação do relé, que encaminha o sinal para a porta NAND adaptada para funcionar como uma NOT e encaminhando o sinal para o pino 1. Por sua

vez, se o pino 1 da porta lógica em estudo foi um pino de saída, esta é encaminhada diretamente pelo relé pelo canal NO e é lido pelo ESP32.

Na figura seguinte é possível verificar o esquema funcional correspondente a funcionamento de uma porta lógica de duas entradas e uma saída de forma a apresentar o funcionamento do nosso projeto.

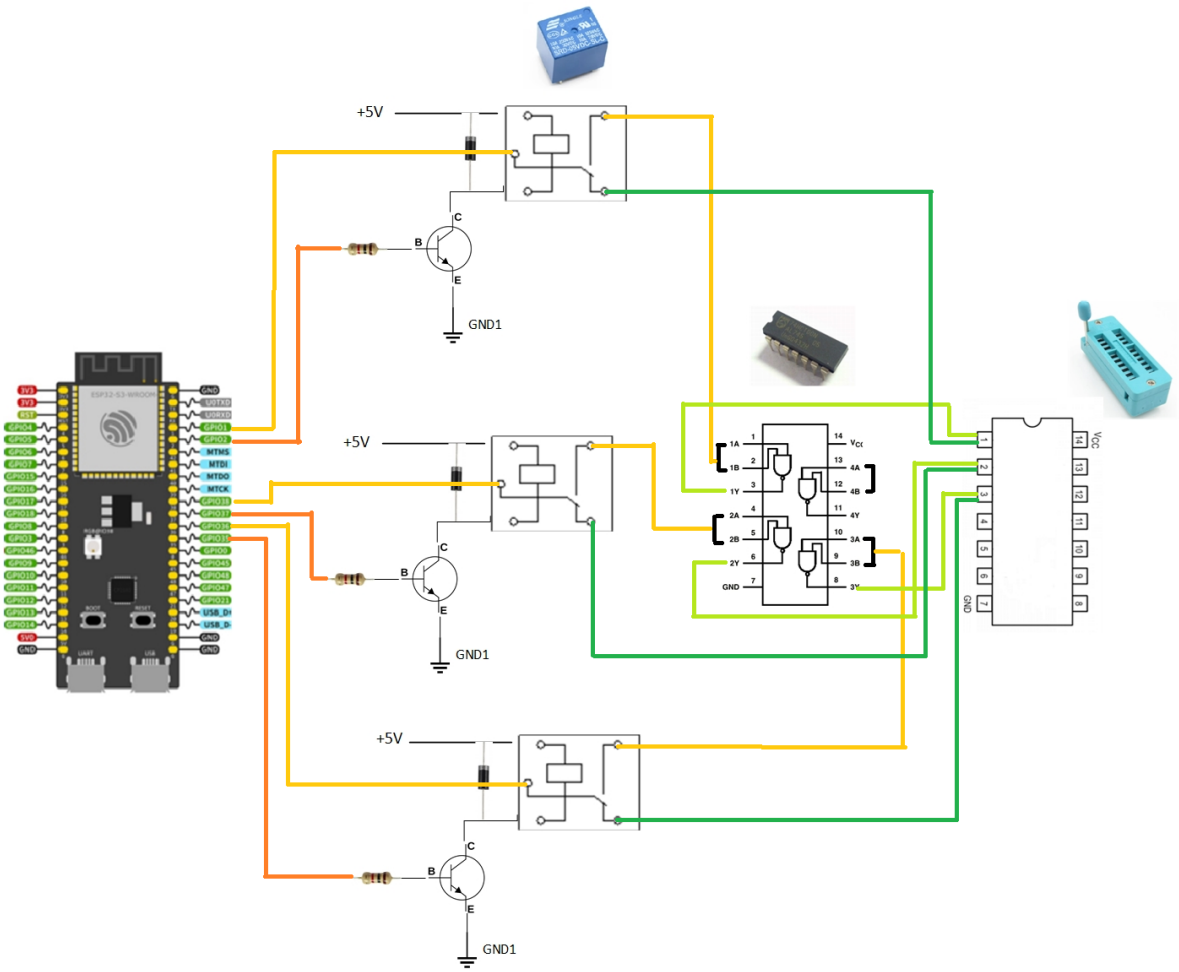

Figura 23: Esquema funcional de uma porta de 2 entradas e 1 saída

De forma a explicar mais detalhadamente o funcionamento do projeto realizamos o esquema acima. Enquanto na figura 22 podemos verificar a configuração para um pino IN ou OUT, de forma a verificar o encaminhamento do sinal nas duas situações, pois o circuito foi projetado de forma que cada um dos relés consiga efetuar as duas funções *IN* ou *OUT*, sendo o seu encaminhamento previamente configurado através do ESP32. Na figura 23, foi utilizado como exemplo uma das configurações mais frequentes das portas lógicas, que consiste em

duas entradas e uma saída. Para tal são utilizados três relés, um para cada pino das portas lógicas. Todos eles são controlados pelo ESP32, o microcontrolador utilizado neste projeto, cada um dos relés está conectado a 2 pinos do ESP32, um ligado diretamente à resistência e por conseguinte ao transístor que funcionará como pino de controlo e outro ligado ao COM responsável pela leitura ou envio de sinais, dependendo do tipo de porta lógica a analisar.

Cada relé está conectado a um transístor que é ativado ou desativado pelo ESP32, permitindo controlar a comutação do relé

Vamos supor por exemplo, que estaríamos a estudar uma porta AND, nesse caso sabemos que o relé 1 e 2 serão entradas e o relé 3 será uma saída (do ponto de vista da porta lógica). Neste exemplo, no relé 1 e 2, será enviado um sinal para o pino de controlo que vai ativar a comutação desse relé, e será enviado o valor 1 (5V), de seguida o sinal é encaminhado para o CI 74HCT00, que foi adaptado para funcionar como uma porta NOT, alterando assim o sinal enviado para a porta lógica, pois a sua saída será 0 (0V), este sinal então será enviado para a porta lógica que, de acordo com a tabela de verdade das portas AND, esperamos que a saída seja 1 (5V). Este sinal de saída é então enviado através do relé 3, que não sofreu comutação, e segue para o ESP32 onde será lido, caso seja conforme o esperado, declaramos que a porta esta OK, caso contrário, declaramos que a porta esta NOT OK.

No nosso projeto, em cada teste, é realizado esta operação para os 12 pinos das portas lógicas, o que faz com que o os 12 reles sejam utilizados em simultâneo em cada teste. Estes são previamente configurados pelo ESP32, após solicitar ao utilizador qual a porta lógica que pretende avaliar.

O procedimento de teste permite garantir a integridade e o correto funcionamento das portas lógicas, enquanto o ESP32 executa o controle e análise dos sinais, verificando a consistência dos resultados obtidos nas saídas com as operações lógicas previstas.

5.2 Esquema Elétrico

Após serem consideradas várias opções de materiais disponíveis para a realização do projeto e de se terem escolhido os componentes mais adequados, procedeu-se ao desenho do esquema elétrico através do programa EasyEDA. Tendo em conta a dimensão do esquema elétrico ele encontra-se dividido nas próximas três imagens, onde podemos verificar todas as ligações e componentes utilizados na elaboração do circuito de testes dos componentes eletrónicos, nomeadamente, de circuitos integrados que foi o que foi testado neste projeto.

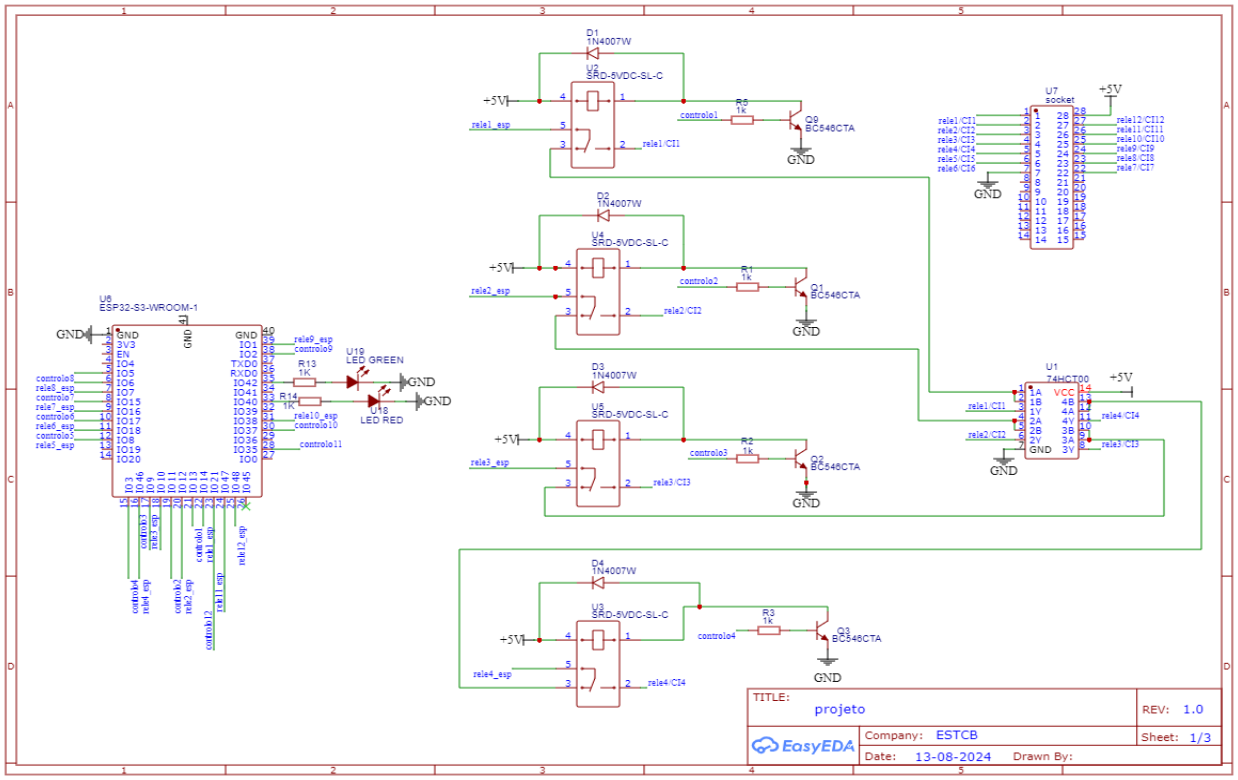

Figura 24: Esquema elétrico parte 1/3

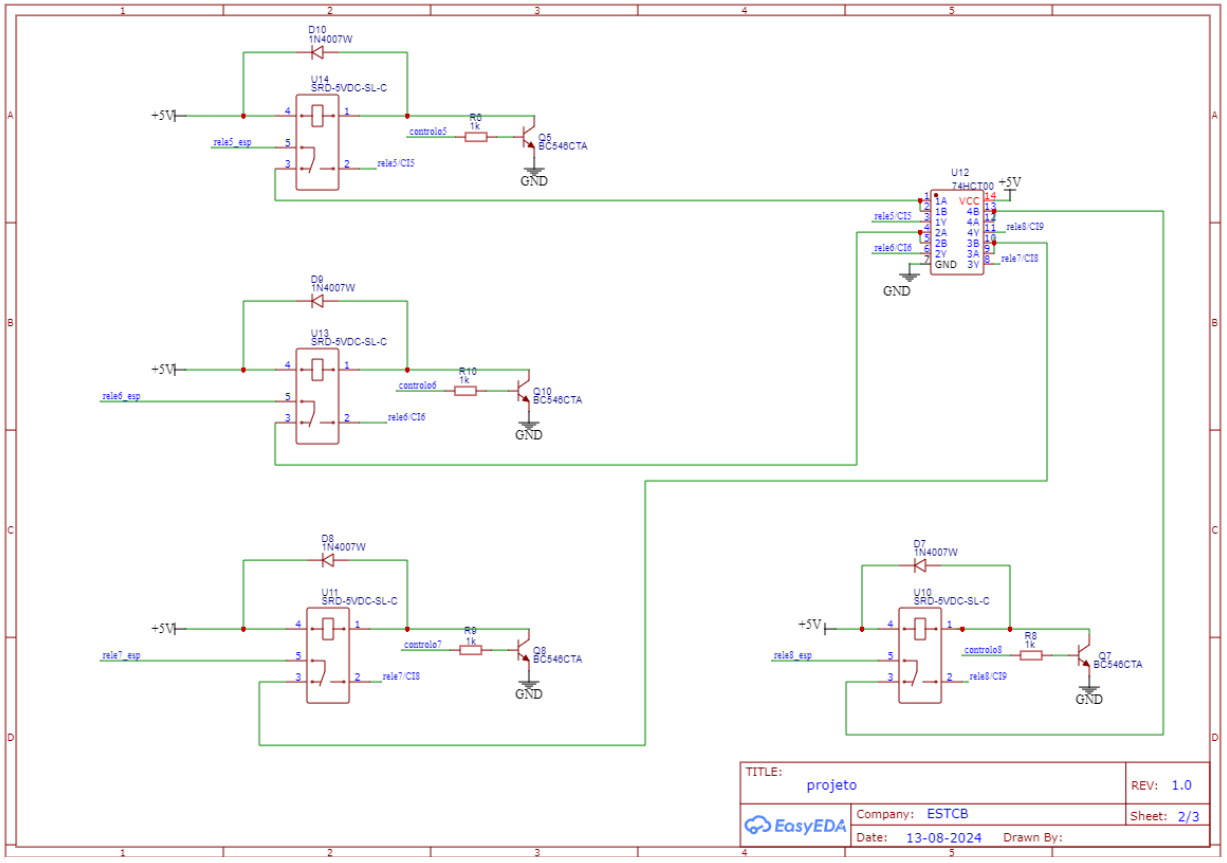

Figura 25: Esquema elétrico parte 2/3

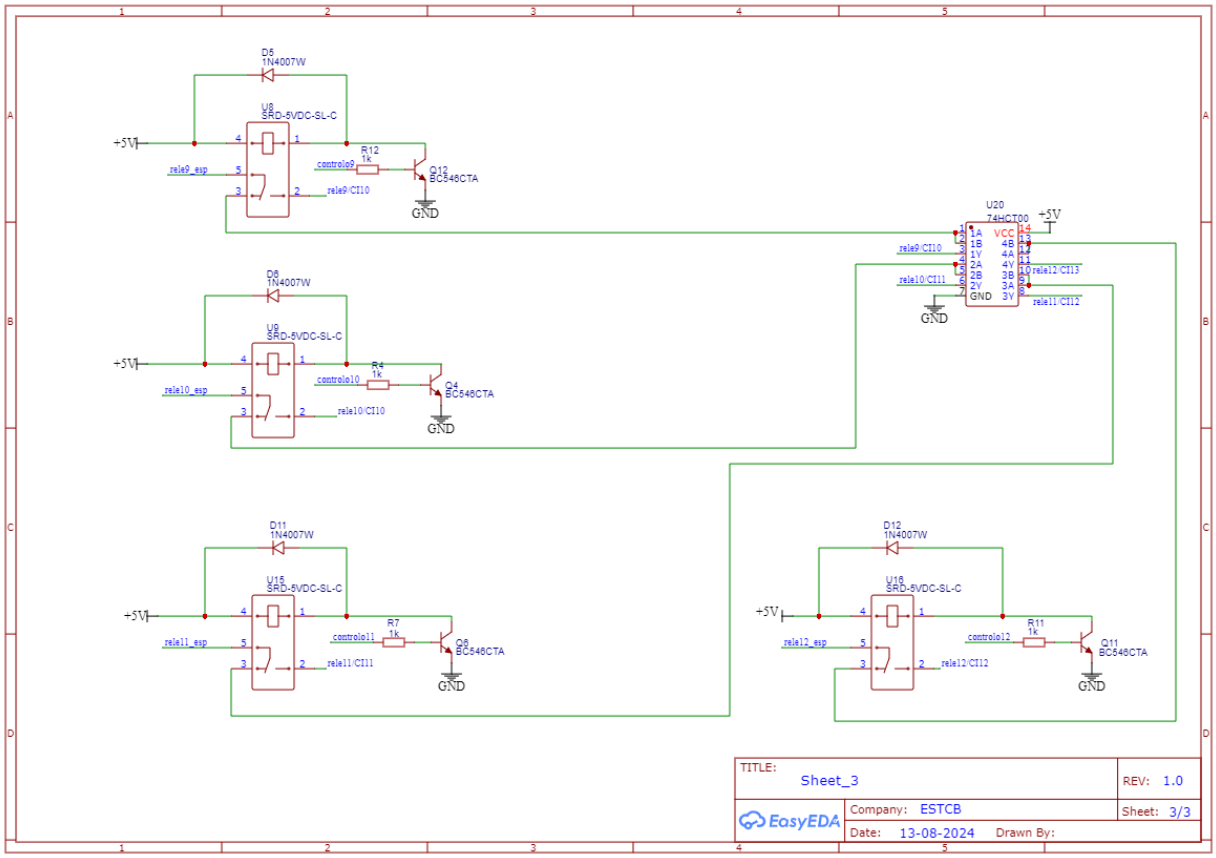

Figura 26: Esquema elétrico parte 3/3

Nas figuras 24, 25 e 26 podemos verificar o esquema elétrico desenhado para o projeto, para a montagem do mesmo foram necessários os seguintes materiais:

- 1 Fonte de tensão;
- 1 breadboard;
- 1 placa de circuito perfurada;
- 1 Esp-32 S3 WROOM-1;
- 12 Relés SRD – 05VDC-SL-C;
- 12 Díodos 1N4002;
- 12 Transístores BC546C;
- 3 Circuito Integrado 74HCT00;
- Portas Lógicas para os testes;
- 1 ZIF *Socket* de 20 pinos;
- 1 LED vermelho;
- 1 LED verde;
- 14 resistências de 1000Ω.

O desenho do projeto foi pensado de forma a conseguir realizar os testes de várias portas lógicas utilizando apenas uma montagem do circuito, tendo em conta que consoante o tipo de

porta a analisar as entradas e saídas seriam diferentes entre si. Por esse motivo foram também escolhidos os relés pois trata-se de um componente eletrónico que nos permite utilizá-lo estando ativo ou desativo para fazer envio de dados ou leitura dos dados.

O circuito é alimentado a 5V, e todas os componentes do circuito são alimentados por essa fonte de alimentação, também as conexões do GND são distribuídas por todo o circuito de forma a alimentar adequadamente todos os componentes.

Observando o esquema elétrico é possível identificar o Esp-32 S3 WROOM-1 como componente principal do circuito, pois é através deste microcontrolador que são enviados os sinais para controlar os relés, ativando-os quando é necessário enviar dados e mantendo desativados no caso de leitura de dados, para tal foram utilizados vários pinos GPIO do ESP32 12 para controlo dos relés e outros 12 para enviar e receber dados para os dispositivos em teste quando necessário, e 2 pinos para os LEDs perfazendo um total de 25 pinos utilizados.

Os transístores são utilizados para controlar os relés, tal como verificamos na figura 5, o transístor possui 3 terminais, o Coletor, Base e Emissor. No nosso circuito o coletor está conectado a um dos terminais da bobine no relé, a base do transístor está conectada a um pino de controlo do ESP32 através de uma resistência de 1KΩ e por fim, o emissor está conectado diretamente a GND (*ground*). Os transístores funcionam como uma espécie de interruptor no circuito uma vez que, quando um sinal positivo e maior que 0.7V chega na base do transístor este começa a conduzir, o que permite que a corrente flua do coletor para o emissor, consequentemente ativando o relé. Quando o relé é ativado começa a passar corrente na bobine do relé, o que gera um campo magnético que aciona o relé, alterando o seu estado de NO para NC. No caso, de o sinal que está a ser enviado para a base for desativo, o transístor deixa de conduzir, interrompendo a corrente que passa pela bobine do relé, e desativa o relé, alterando novamente os seus estados de NC para NO. A resistência colocada entre a base do transístor e o ESP assume duas funções importantes, permite limitar a corrente que chega a base do transístor, pois sem a resistência esta corrente poderia ser excessiva para o funcionamento do mesmo, podendo comprometer o seu funcionamento e protege o transístor de oscilações indesejadas de corrente ou ruídos do circuito, pois ao limitar a corrente na base, a resistência contribui para a estabilidade do transístor, evitando que ele ligue e desligue quando não é desejado.

Tendo em conta que decidimos avaliar apenas as portas lógicas básicas, e que cada uma delas tem 14 pinos, sendo um deles para o VCC e outro para a GND, restam-nos 12 pinos a testar, motivo pelo qual foram utilizados 12 relés no projeto para a leitura e controlo das portas lógicas a estudar. Os relés foram soldados a uma placa de circuito perfurada ao contrário de todos os outros componentes, foi necessário fazer esta adaptação tendo em conta o design do relé, pois como podemos verificar na figura 3 o pino do COM encontra-se no centro mas não em linha reta com os outros pinos, o que o impede de entrar em uma *breadbord* comum.

Através dos relés foi possível configurar o circuito, uma vez que foi através destes que foi possível controlar a passagem de dados para as entradas ou saídas das portas lógicas em estudo. Quando estávamos perante um pino de entrada da porta lógica, o pino de controlo do relé era acionado encaminhando os dados para a pino de entrada da porta lógica, quando se tratava de um pino de saída, não era acionado o pino de controlo, e a saída da porta lógica era encaminhada pelo pino do rele NO, sendo posteriormente lido através do ESP32. A flexibilidade permitida pelos relés foi o que nos levou a escolha dos mesmos para o projeto.

Os relés foram dispostos em uma matriz de modo a permitir a sua reconfiguração de acordo com o tipo de portas a estudar, tendo em conta a rápida comutação dos relés com a sua resposta quase imediata, foi possível garantir a fiabilidade dos resultados face ao comportamento das portas lógicas.

Cada relé possui um díodo de *flyback* conectado em paralelo à bobine para proteger o transístor contra picos de tensão. Quando o relé é desativado, ou seja, quando a corrente que passa pela bobine é interrompida, a energia armazenada no campo magnético precisa de ser dissipada sem a utilização de um díodo, esta energia é dissipada através de um pico de tensão que pode danificar componentes do circuito como transístores, circuitos integrados ou mesmo a própria fonte de alimentação (Malvino & Bates, 2016). Por esse motivo, o díodo foi colocado em paralelo com a bobine do relé, com a sua polaridade invertida em relação à tensão de alimentação. Quando o relé está ativo, o díodo não conduz e não interfere com o funcionamento normal esperado, no entanto, quando o relé é desativado ou desligado, o campo magnético "colapsa" e gera um pico de tensão inversa. O díodo de *flyback* age nesta situação, conduzindo esta corrente de retorno, permitindo que a energia seja dissipada de forma segura na resistência da própria bobine, evitando que a tensão seja aplicada em outros componentes do circuito. Ao utilizar o díodo de *flyback*, reduzimos de forma significativa o stress elétrico sobre a bobine do relé, prolongando a vida útil do sistema. Além disso, evita a ocorrência de possíveis falhas ou danos que poderiam acontecer devido aos picos de tensão não controlados que podem gerar interferências eletromagnéticas nos outros componentes, pelo que a utilização deste díodo ajuda a reduzir o ruído elétrico e as interferências, contribuindo para o funcionamento estável e confiável do circuito.

Foram utilizados três CI 74HCT00, entre o relé e o pino do ZIF *Socket*. Para receber estes sinais foram realizados curto-circuitos nas entradas de cada portas lógicas, fazendo com que esta funcionasse como uma porta NOT para cada sinal recebido através desta ligação, a saída de cada um foi ligada a um pino do ZIF *socket*. A utilização do 74HCT00 permitiu compatibilizar os níveis lógicos, uma vez que o ESP e as portas lógicas podem funcionar com diferentes níveis de tensão, o 74HCT00 permite regular estas diferenças, fazendo com que os sinais enviados para as portas lógicas sejam adequados às mesmas. Durante os testes, a inclusão do 74HCT00 mostrou-se eficaz em proteger as portas lógicas de oscilações indesejadas e de ruídos que poderiam comprometer os resultados. O desempenho do sistema foi consistentemente preciso, com as portas lógicas respondendo de acordo com as entradas esperadas, o que validou a eficácia do 74HCT00 como um componente de proteção essencial no projeto.

As portas lógicas a testar são colocadas todas no ZIF *socket*, isto permite que sejam testados diferentes tipos de CI podendo ser inseridos e removidos sem a necessidade de soldar os componentes.

5.3 Testes das portas lógicas

Como foi mencionado anteriormente foi utilizado um ZIF *socket* no circuito, esta foi uma solução técnica muito vantajosa, uma vez que permitiu a substituição rápida e segura dos circuitos integrados, minimizando o risco de danificar os componentes utilizados devido a um excesso de força ao inserir ou remover os CI. Essa flexibilidade foi importante, visto que o

projeto tinha como objetivo testar diferentes CI de forma eficiente, garantindo que o mesmo circuito pudesse ser reaproveitado sem necessidade de ajustes físicos frequentes.

Foram ainda utilizados 12 relés, controlados através do microcontrolador ESP32 S3. Os relés tiveram a função de agir como intermediários entre o microcontrolador e as portas lógicas, permitindo controlar a entrada e saída de sinais de forma isolada. A configuração dinâmica dos pinos do ESP32, alternando entre estados de entrada (*input*) e saída (*output*) de acordo com o tipo de porta lógica testada, permitiu o teste de diferentes tipos de portas lógicas. Na prática, os relés designados como *output* eram responsáveis por enviar os sinais aos CI. O ESP32 S3 ativa os relés através dos seus pinos de controlo, que por sua vez acionavam o transístor, permitindo que o sinal fosse encaminhado para a entrada da porta lógica em teste. Já os relés configurados como *input* realizavam a função contrária, uma vez que recebem o sinal de saída das portas lógicas, para que seja lido e interpretado pelo ESP32.

Na tabela seguinte é possível observar a lógica aplicada para cada tipo de porta em teste, sabendo que foram testadas portas AND, NAND, XOR, OR, NOR e NOT.

Tabela 13: Configuração dos Relés

Relé	AND	NAND	XOR	OR	NOR	NOT
Relé 1	*Output*	*Output*	*Output*	*Output*	*Input*	*Output*
Relé 2	*Output*	*Output*	*Output*	*Output*	*Output*	*Input*
Relé 3	*Input*	*Input*	*Input*	*Input*	*Output*	*Output*
Relé 4	*Output*	*Output*	*Output*	*Output*	*Input*	*Input*
Relé 5	*Output*	*Output*	*Output*	*Output*	*Output*	*Output*
Relé 6	*Input*	*Input*	*Input*	*Input*	*Output*	*Input*
Relé 7	*Input*	*Input*	*Input*	*Input*	*Output*	*Input*
Relé 8	*Output*	*Output*	*Output*	*Output*	*Output*	*Output*
Relé 9	*Output*	*Output*	*Output*	*Output*	*Input*	*Input*
Relé 10	*Input*	*Input*	*Input*	*Input*	*Output*	*Output*
Relé 11	*Output*	*Output*	*Output*	*Output*	*Output*	*Input*
Relé 12	*Output*	*Output*	*Output*	*Output*	*Input*	*Output*

Como podemos verificar na tabela acima de acordo com cada porta lógica em teste é configurado cada relé. Relembro que a cada relé está associado um pino de controlo, que no caso de o relé ser de *Output* (sempre que o relé é *Output*, envia um sinal através do relé que corresponde á entrada da porta lógica) o pino de controlo é também de *Output* e envia um sinal através do transístor ao relé, que vai ativar a comutação do relé. Pelo contrário, quando o pino do relé é declarado como *input* irá receber o sinal de saída vindo da porta lógica, fazendo a sua leitura e apresentando o resultado.

5.4 Fluxograma

Os fluxogramas são representações gráficas que ilustram a sequência de fases e processos de um sistema, código, algoritmo, etc. Para realizar um fluxograma são utilizados símbolos padrão, como retângulos para processos, losangos para decisões e setas para direcionar o fluxo, os fluxogramas permitem ter uma visão clara e estruturada do funcionamento de um programa, permitindo identificar facilmente a lógica utilizada, pontos de melhoria e a dependência entre diferentes fases do processo.

Os fluxogramas que apresentamos nas figuras 27, 28, 29, 30, 31, 32, 33, 34 e 35 foram realizados de forma a estruturar as operações no código desenvolvido. Desta forma ao analisar os fluxogramas, é possível ter uma visão objetiva sobre a sequência de acontecimentos que levam ao teste das portas lógicas e como este teste é conseguido.

Inicialmente podemos observar o fluxograma correspondente ao programa principal, a imagem seguinte permite visualizar o fluxo principal do programa desde o seu início, configuração dos relés e dos pinos de controlo, a seleção do tipo de porta lógica a ser testada, tal como o *reset* dos pinos após cada teste. Esta representação gráfica facilita a compreensão do ciclo de operações do código e a interação com o utilizador através da comunicação serial. De seguida, é apresentado o fluxograma para cada uma das funções inseridas no programa, que são essenciais para o correto funcionamento do programa.

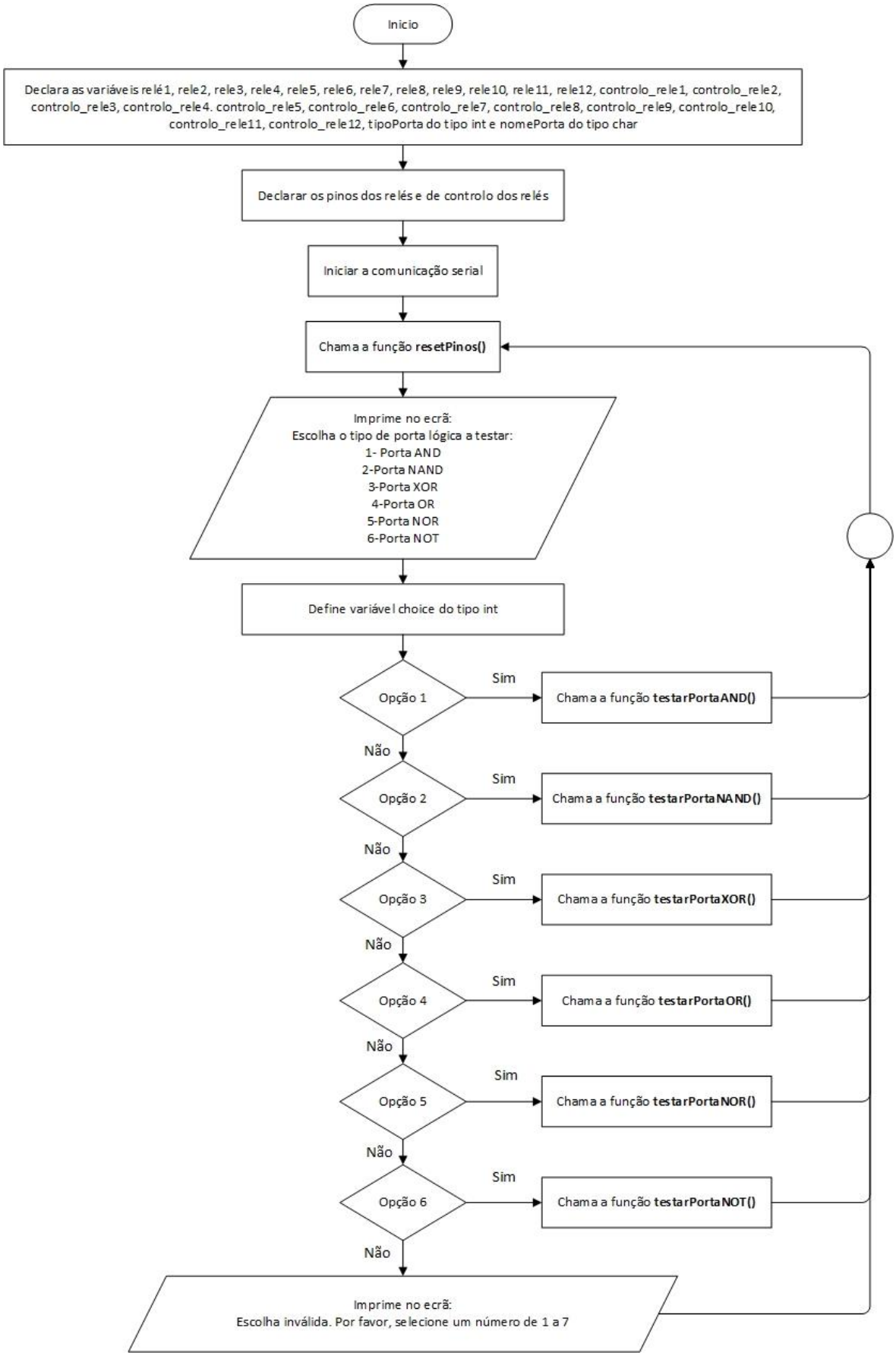

Figura 27: Fluxograma do Programa Principal

De seguida podemos verificar todas as funções, associadas ao teste das portas lógicas, pois depois de o utilizador escolher a opção de porta que pretende analisar, associada a cada opção está uma função responsável por configurar todos os pinos do relé e de controlo do relé consoante a configuração das portas.

Figura 28: Fluxograma da função testarPortasAND()

Figura 29: Fluxograma da função testarPortasNAND()

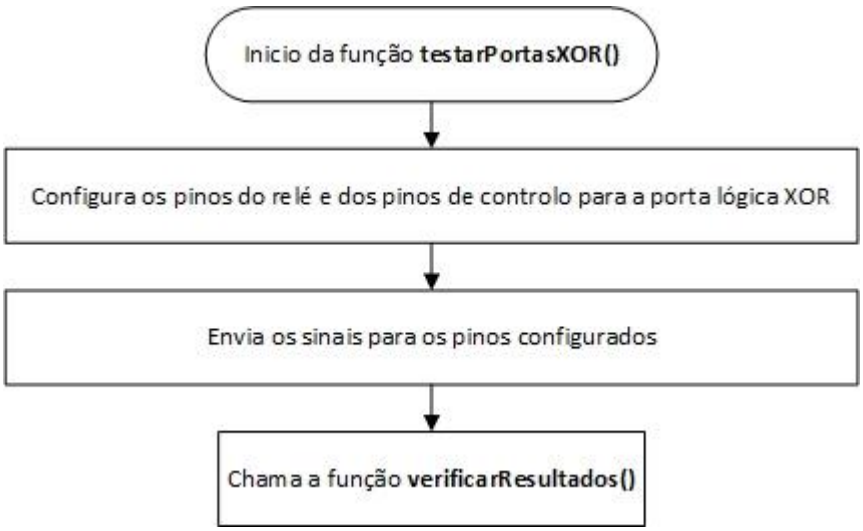

Figura 30: Fluxograma da função testarPortasXOR()

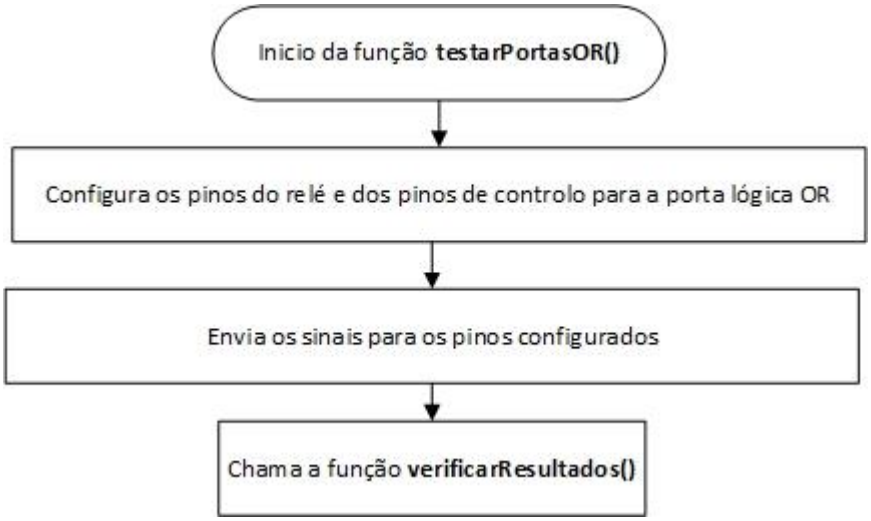

Figura 31: Fluxograma da função testarPortasOR()

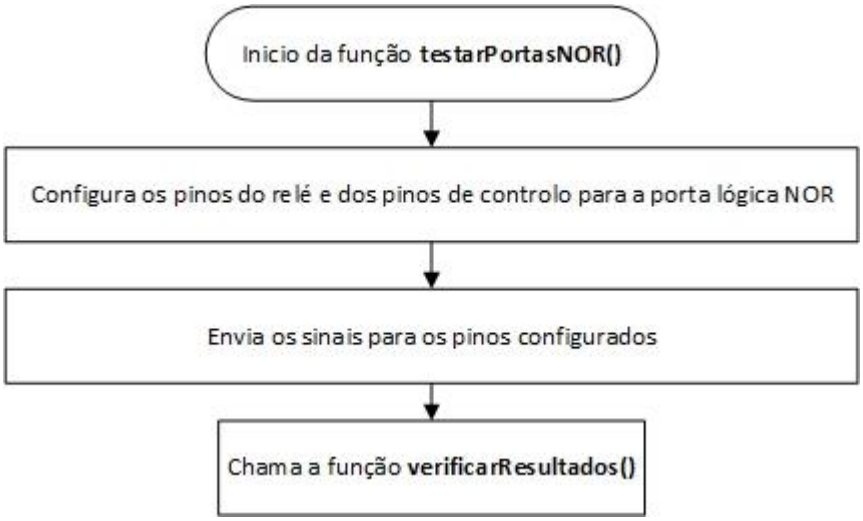

Figura 32: Fluxograma da função testarPortasNOR()

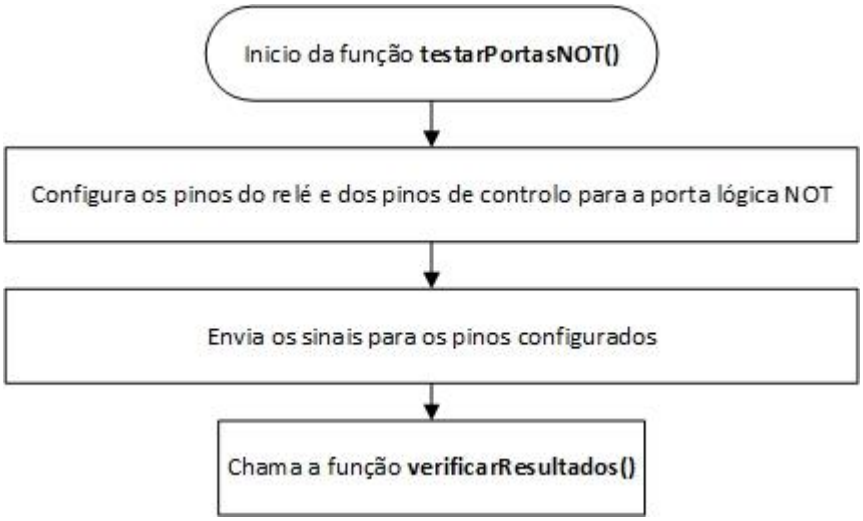

Figura 33: Fluxograma da função testarPortasNOT()

Após a configuração das portas apresentamos o fluxograma responsável pela verificação dos resultados de cada teste, esta última parte é a que nos permite determinar se a porta esta OK, ou seja, está a funcionar corretamente e em condições de ser utilizada, ou se pelo contrário está NOT OK, ou seja, não está a funcionar corretamente não sendo aconselhada a sua utilização.

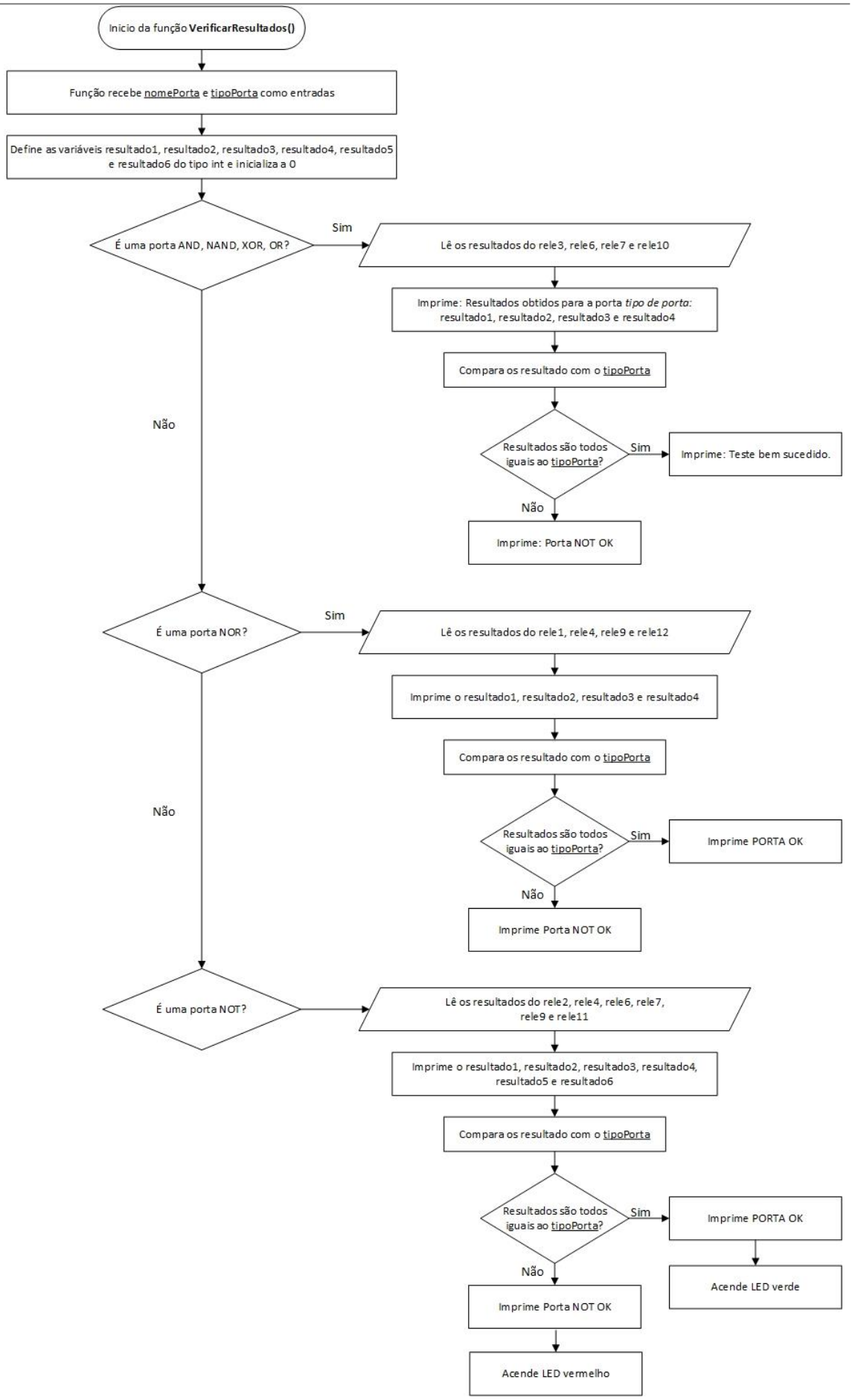

Figura 34: Fluxograma da função verificarResultados()

A função verificarResultados() é a função mais importante do código, uma vez que esta é a responsável pela leitura e análise dos resultados obtidos e compara com os resultados esperados, motivo este pelo qual apresentamos o código implementado de seguida:

```
void verificarResultados(const char* nomePorta, int tipoPorta) {
 int resultado1 = 0, resultado2 = 0, resultado3 = 0, resultado4 = 0, resultado5 = 0, resultado6 = 0; //declara as variáveis dos resultados

 // Lê os resultados dos pinos conforme o tipo de porta
 //caso 0, 1, 2, 3 e 4 de acordo com os tipos de leitura para cada porta
 switch (tipoPorta) {
  case 0:  // AND, XOR, OR
   resultado1 = digitalRead(rele3);
   resultado2 = digitalRead(rele6);
   resultado3 = digitalRead(rele7);
   resultado4 = digitalRead(rele10);
   Serial.print("Resultados obtidos para a porta ");
   Serial.print(nomePorta);
   Serial.print(": ");
   Serial.print(resultado1);
   Serial.print(" ");
   Serial.print(resultado2);
   Serial.print(" ");
   Serial.print(resultado3);
   Serial.print(" ");
   Serial.println(resultado4);
   break;

  case 1:  // NAND
   resultado1 = digitalRead(rele3);
   resultado2 = digitalRead(rele6);
   resultado3 = digitalRead(rele7);
   resultado4 = digitalRead(rele10);
   Serial.print("Resultados obtidos para a porta ");
```

```
  Serial.print(nomePorta);
  Serial.print(": ");
  Serial.print(resultado1);
  Serial.print(" ");
  Serial.print(resultado2);
  Serial.print(" ");
  Serial.print(resultado3);
  Serial.print(" ");
  Serial.println(resultado4);
  break;

 case 2:  // NOR
  resultado1 = digitalRead(rele1);
  resultado2 = digitalRead(rele4);
  resultado3 = digitalRead(rele9);
  resultado4 = digitalRead(rele12);
  Serial.print("Resultados obtidos para a porta ");
  Serial.print(nomePorta);
  Serial.print(": ");
  Serial.print(resultado1);
  Serial.print(" ");
  Serial.print(resultado2);
  Serial.print(" ");
  Serial.print(resultado3);
  Serial.print(" ");
  Serial.println(resultado4);
  break;

 case 3:  // NOT
  resultado1 = digitalRead(rele2);
  resultado2 = digitalRead(rele4);
  resultado3 = digitalRead(rele6);
  resultado4 = digitalRead(rele7);
```

```
resultado5 = digitalRead(rele9);
resultado6 = digitalRead(rele11);
Serial.print("Resultados obtidos para a porta ");
Serial.print(nomePorta);
Serial.print(": ");
Serial.print(resultado1);
Serial.print(" ");
Serial.print(resultado2);
Serial.print(" ");
Serial.print(resultado3);
Serial.print(" ");
Serial.print(resultado4);
Serial.print(" ");
Serial.print(resultado5);
Serial.print(" ");
Serial.println(resultado6);
break;

 case 4:  // XNOR, resultado esperado é 1
resultado1 = digitalRead(rele3);
resultado2 = digitalRead(rele4);
resultado3 = digitalRead(rele9);
resultado4 = digitalRead(rele10);
Serial.print("Resultados obtidos para a porta ");
Serial.print(nomePorta);
Serial.print(": ");
Serial.print(resultado1);
Serial.print(" ");
Serial.print(resultado2);
Serial.print(" ");
Serial.print(resultado3);
Serial.print(" ");
Serial.println(resultado4);
```

```
        break;

      default:
        Serial.println("Tipo de porta desconhecido.");
        return;
    }

    // Verificar resultados de acordo com o tipo de porta
     if (strcmp(nomePorta, "AND") == 0 || strcmp(nomePorta, "XOR") == 0 || strcmp(nomePorta, "OR") == 0) {
      // As saídas têm que ser 0 para essas portas estarem OK
      if (resultado1 == LOW && resultado2 == LOW && resultado3 == LOW && resultado4 == LOW) {
        Serial.println("Teste bem sucedido.");
        digitalWrite(ledVerde, HIGH);  // Acende LED verde
        digitalWrite(ledVermelho, LOW); // Apaga LED vermelho
      } else {
        Serial.println("Porta NOT OK.");
        digitalWrite(ledVermelho, HIGH);  // Acende LED vermelho
        digitalWrite(ledVerde, LOW); // Apaga LED verde
      }
    } else if (strcmp(nomePorta, "NAND") == 0 || strcmp(nomePorta, "NOR") == 0) {
      // As saídas têm que ser 1 para essas portas estarem OK
      if (resultado1 == HIGH && resultado2 == HIGH && resultado3 == HIGH && resultado4 == HIGH) {
        Serial.println("Teste bem sucedido.");
        digitalWrite(ledVerde, HIGH);  // Acende LED verde
        digitalWrite(ledVermelho, LOW); // Apaga LED vermelho
      } else {
        Serial.println("Porta NOT OK.");
        digitalWrite(ledVermelho, HIGH);  // Acende LED vermelho
        digitalWrite(ledVerde, LOW); // Apaga LED verde
      }
    } else if (strcmp(nomePorta, "XNOR") == 0) {
```

```
        // As saídas têm que ser 1 para a porta XNOR estar OK
        if (resultado1 == HIGH && resultado2 == HIGH && resultado3 == HIGH && resultado4
== HIGH) {
          Serial.println("Teste bem sucedido.");
          digitalWrite(ledVerde, HIGH);  // Acende LED verde
          digitalWrite(ledVermelho, LOW); // Apaga LED vermelho
        } else {
          Serial.println("Porta NOT OK.");
          digitalWrite(ledVermelho, HIGH);  // Acende LED vermelho
          digitalWrite(ledVerde, LOW); // Apaga LED verde
        }
      } else if (strcmp(nomePorta, "NOT") == 0) {
        // As saídas têm que ser 1 para a porta NOT estar OK
        if (resultado1 == HIGH && resultado2 == HIGH && resultado3 == HIGH && resultado4
== HIGH && resultado5 == HIGH && resultado6 == HIGH) {
          Serial.println("Teste bem sucedido.");
          digitalWrite(ledVerde, HIGH);  // Acende LED verde
          digitalWrite(ledVermelho, LOW); // Apaga LED vermelho
        } else {
          Serial.println("Porta NOT OK.");
          digitalWrite(ledVermelho, HIGH);  // Acende LED vermelho
          digitalWrite(ledVerde, LOW); // Apaga LED verde
        }
      }

      delay(2000);

      // Apagar ambos os LEDs após o teste
      digitalWrite(ledVerde, LOW);
      digitalWrite(ledVermelho, LOW);

      //volta ao início
    }
```

Por último, mas não menos importante, podemos verificar o fluxograma correspondente á função *reset* pinos.

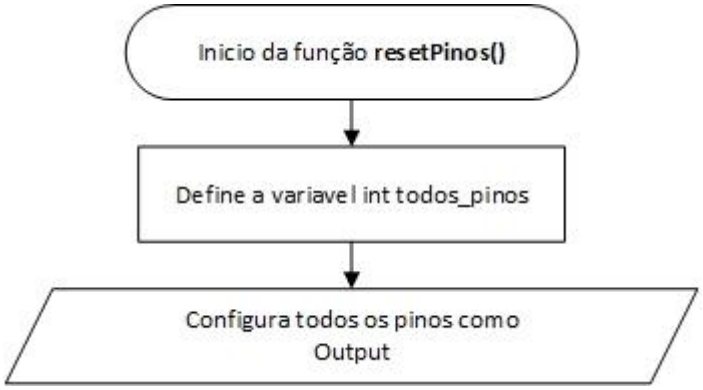

Figura 35: Fluxograma da função resetPinos()

5.5 Análise de Resultados

Após planear e definir os componentes do projeto, passamos á sua execução. Em uma primeira fase realizamos a montagem do circuito elétrico, de forma a termos o sistema de testes para os circuitos integrados pronto a utilizar. E de seguida, seguindo os fluxogramas apresentados no capítulo anterior prosseguimos á escrita do código de programação que nos permitiu avaliar se os CI em teste estariam funcionais ou não.

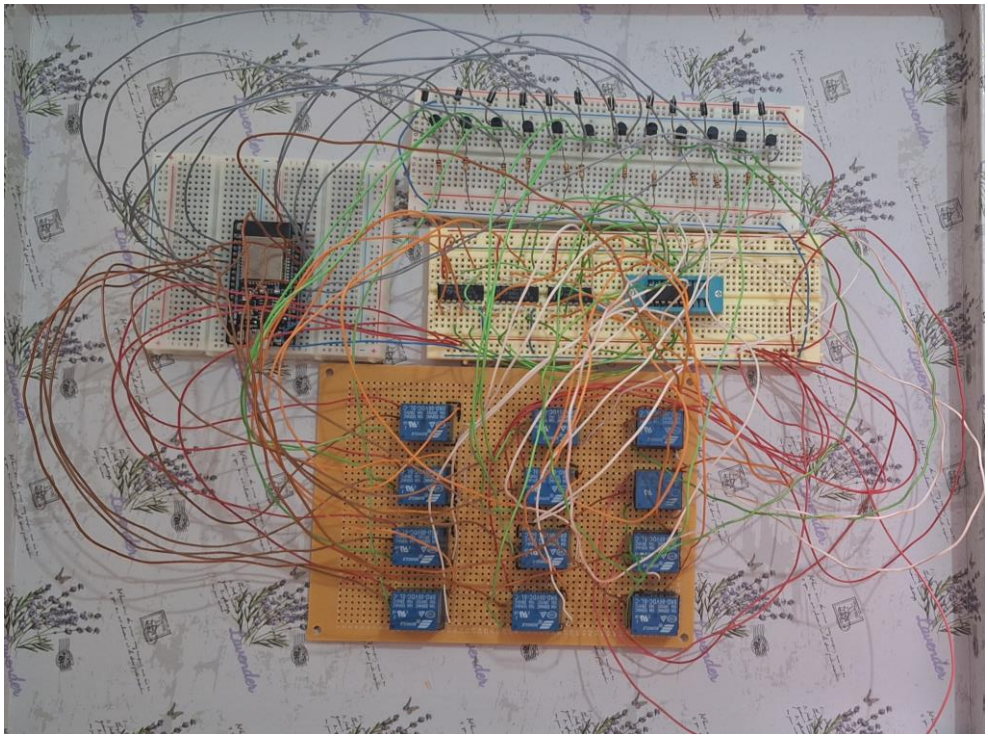

Figura 36: Circuito elétrico

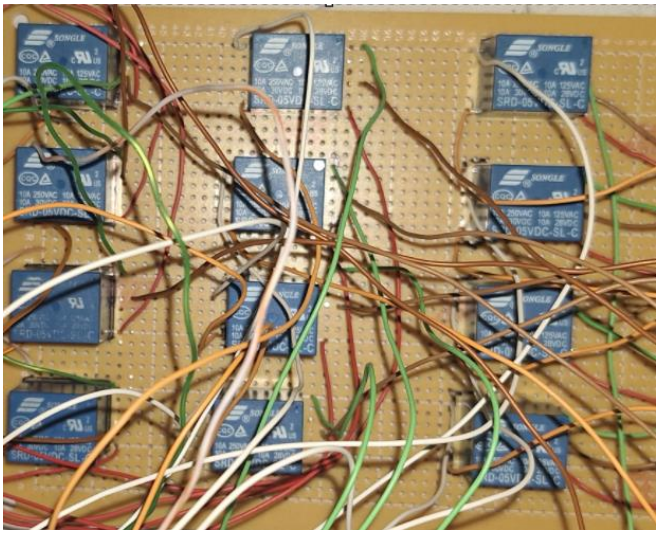

Figura 37: Relés no circuito

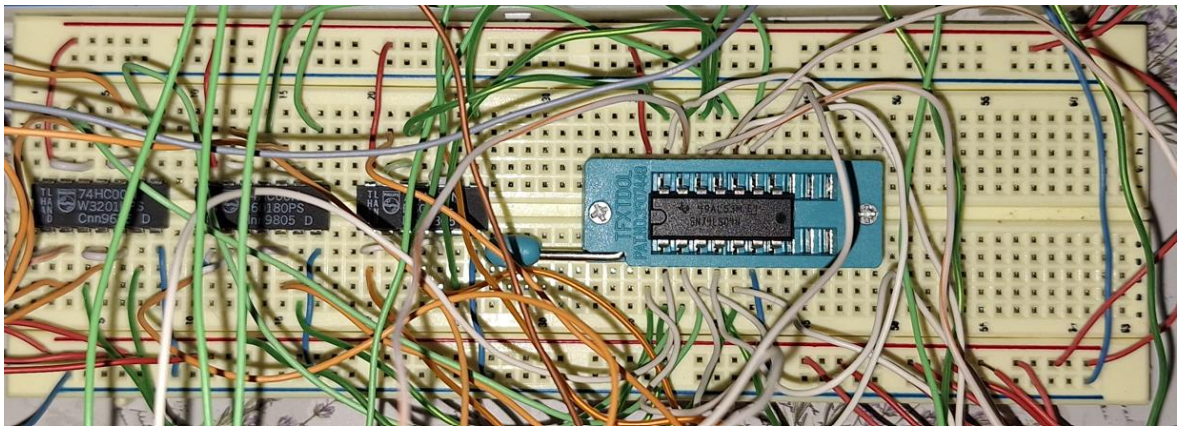

Figura 38: *Breadboard* com HCT00 e a *Socket*

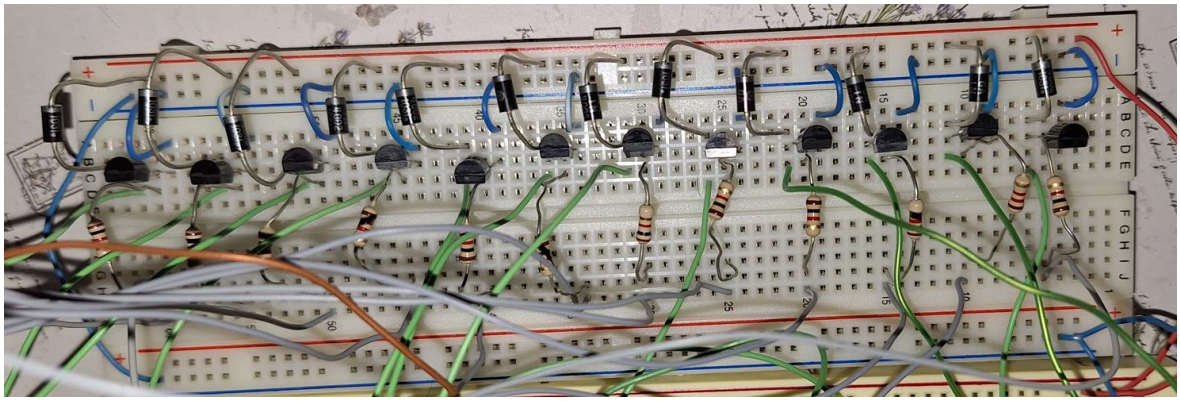

Figura 39: ***Breadbord*** **com os transístores, díodos e resistências**

Na figura 36 podemos verificar o nosso circuito após a sua montagem, no entanto nas figuras seguintes podemos verificar mais detalhadamente o circuito. Na figura 37 verificamos os relés, estes e os fios ligados a eles foram soldados em uma placa de circuito perfurada, uma vez que as disposições dos pinos dos relés não permitiam utilizar uma *breadboard* comum. Na figura 38 podemos verificar uma outra *breadboard* onde foram dispostos os CI HCT00 e o *socket.* Por sua vez, na figura 39 observamos uma outra *breadboard* com os transístores, díodos e resistências, responsáveis por acionar os pinos de controlo dos relés. A alimentação do circuito foi realizada utilizando uma fonte de alimentação e uma *Power Adapted Board* (adaptador de tensão), através dela foi utilizado o pino dos +5V e do GND, que foi o necessário para o funcionamento do circuito.

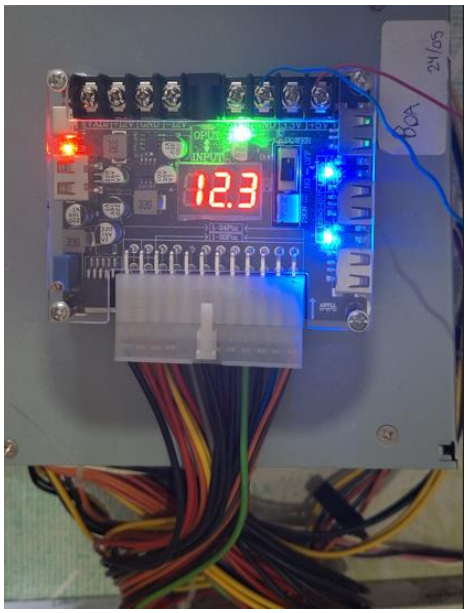

Figura 40: Alimentação do circuito

A fase seguinte consistia em enviar o programa para o ESP32 de forma a fazer o teste dos CI. Para isso utilizados o programa ARDUINO IDE para escrever o nosso programa em C++ (código no Anexo I). Uma vez enviado o programa para o microcontrolador, passamos para o teste dos nossos CI. Antes do teste surgia o menu, que podemos verificar na figura 41, de forma a selecionar o tipo de porta que pretendíamos analisar, pois consoante o tipo de porta escolhida as configurações dos relés e dos pinos do ESP32 são distintas.

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
```

Figura 41: Menu inicial do programa

Este é o menu que nos surge inicialmente de forma a escolher o tipo de porta lógica que queremos avaliar, após escolhida a opção pretendida o ESP32 configura os pinos dos relés e os pinos de controlo conforme a porta escolhida, realiza o envio dos dados para a porra lógica e a leitura dos respetivos pinos. Depois compara o resultado pretendido com as leituras obtidas do ESP32 e caso coincidam é declarado que o teste foi bem-sucedido, ou então, que a porta esta NOT OK, ou seja, não está a funcionar corretamente pelo que não se aconselha a sua utilização nos circuitos. Abaixo podemos verificar alguns dos testes realizados e seus respetivos resultados.

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
Resultados obtidos para a porta NAND: 1 1 1 1
Teste bem-sucedido.
```

Figura 44: Teste realizado à porta NAND

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
Resultados obtidos para a porta NOR: 1 1 1 1
Teste bem-sucedido.
```

Figura 43: Teste realizado à porta NOR

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
Resultados obtidos para a porta OR: 0 0 0 0
Teste bem-sucedido.
```

Figura 42: Teste realizado à porta OR

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
Resultados obtidos para a porta NOT: 1 1 1 1 1 0
Porta NOT OK.
```

Figura 45: Teste realizado à porta NOT

As figuras anteriores referem-se a alguns dos testes que realizamos a algumas das portas em análise, podemos verificar que quando as portas lógicas estão funcionais é indicado que o teste foi bem-sucedido(figura 42, 43 e 44), caso contrário é indicada a mensagem que a porta está NOT OK (figura 45). Foi também adicionado uma funcionalidade ao programa, de forma a prevenir que o utilizador selecione outra opção que não seja corresponda às pretendidas, caso isso aconteça, quando o utilizador seleciona alguma tecla que não seja 1, 2, 3, 4, 5, 6 ou 7 surge a seguinte mensagem “Escolha inválida. Por favor, selecione um número de 1 a 7”, tal como podemos verificar na figura seguinte.

```
Escolha o tipo de porta lógica a testar:
1 - Porta AND
2 - Porta NAND
3 - Porta XOR
4 - Porta XNOR
5 - Porta OR
6 - Porta NOR
7 - Porta NOT
Escolha inválida. Por favor, selecione um número de 1 a 7.
```

Figura 46: Mensagem de erro do programa

De forma a calibrar os nossos resultados com os que eram obtidos, foi utilizado um multímetro. Desta forma, inicialmente em cada teste realizado eram medidas também as tensões em cada ponto do circuito, de forma a verificar o funcionamento do circuito conforme o pretendido e com os resultados.

Após os testes realizados e confirmado o correto funcionamento do circuito foram adicionados LEDs ao mesmo que permitem identificar visualmente os testes realizados.

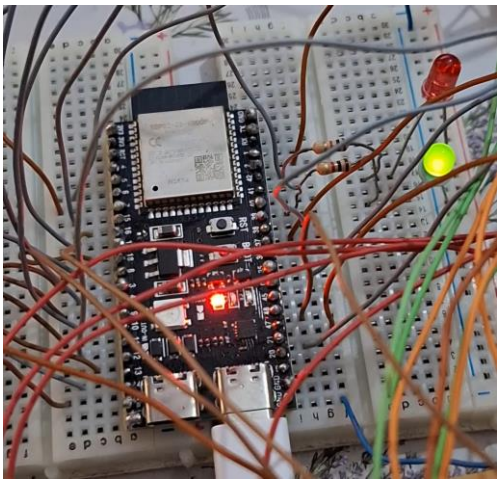

Figura 47: LED de confirmação do estado do teste

De um modo geral o objetivo definido no inicio da realização deste projeto foi atingido, o circuito funciona corretamente, mesmo quando alteramos os dados enviados para o circuito, por padrão estamos a enviar a partida o valor 1 para as portas lógicas de forma a facilitar a programação do mesmo, mas quando alterado, ou seja, em fase de testes enviamos o valor 0 á partida para o circuito através do ESP32, o funcionamento do circuito comprovou-se eficaz, devolvendo os resultados esperados e comprovados pelo multímetro. Pelo que podemos dizer que o objetivo final do projeto foi atingido com estes resultados. Ainda assim o ideal seria termos utilizado o PCB correspondente ao projeto, de forma a melhorar visualmente o circuito, tendo em conta as várias ligações necessárias existe uma grande quantidade de fios que seriam suprimidos com a utilização do PCB mas devido ao tempo limitado não foi possível obtê-lo fisicamente, mas ainda assim foi desenhado, como vamos verificar de seguida.

5.6 PCB

O desenho do circuito impresso (PCB - *Printed Circuit Board*) foi realizado através do *software* EasyEDA, uma ferramenta de *design* online que permite a criação de circuitos eletrónicos e o *layout* de PCBs. Embora o projeto do PCB tenha sido completamente desenvolvido, o processo de fabricação não foi concluído devido à limitação de tempo disponível. No entanto, são apresentadas neste relatório as imagens geradas no *software* EasyEDA, que ilustram o *layout* final do PCB.

O PCB foi desenhado de forma a conter todos os componentes utilizados no circuito, incluindo o microcontrolador ESP32 S3. Este foi desenhado com o intuito de otimizar a organização dos componentes e minimizar problemas como *crosstalk* (interferência de sinais) e ruído de sinal. Cada componente foi colocado de forma a reduzir o comprimento das trilhas e melhorar a eficiência do circuito.

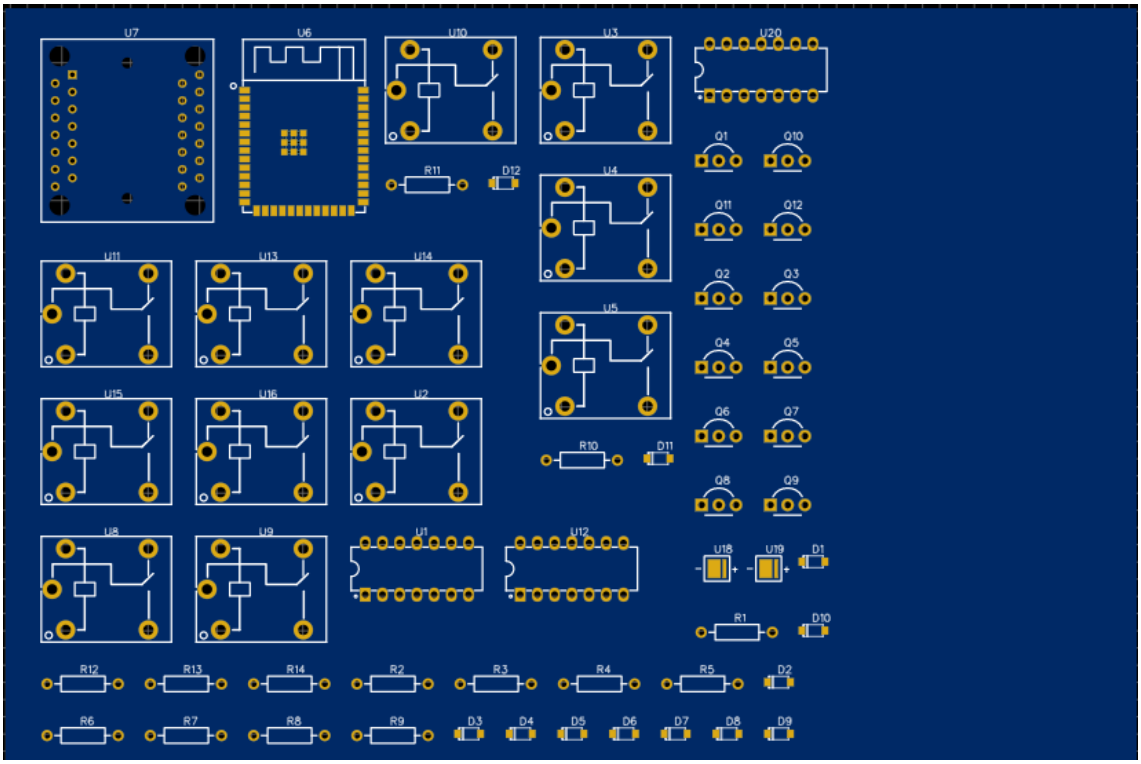

Figura 48: *Layout* do PCB visão 2D

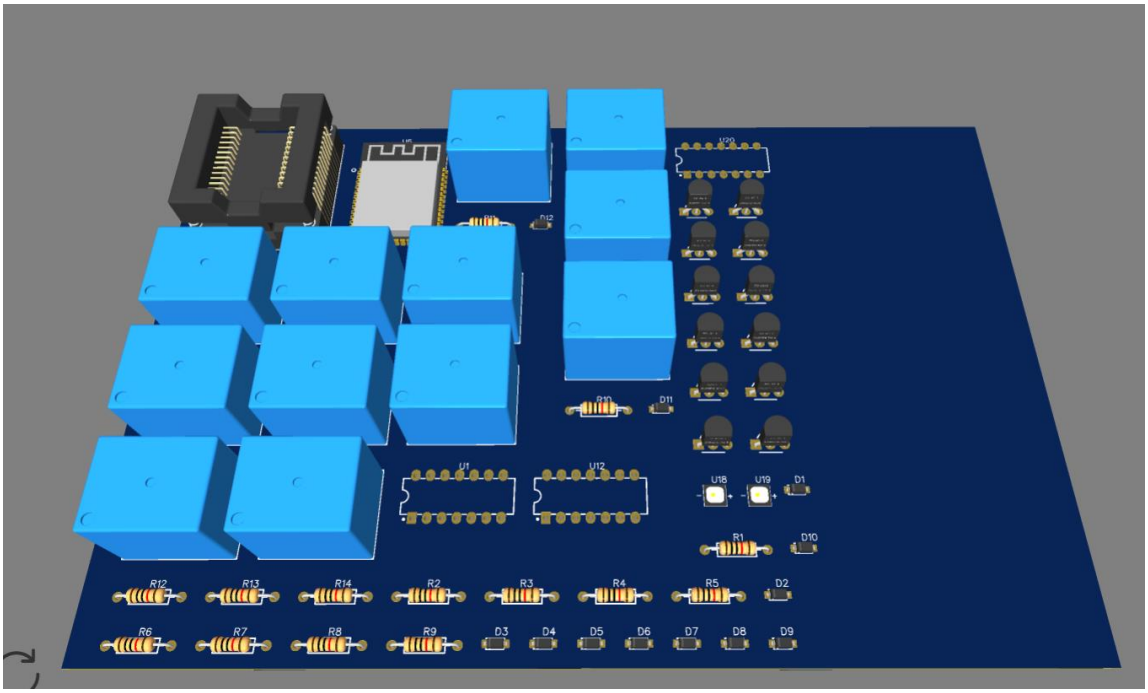

Figura 49: *Layout* do PCB visão 3D

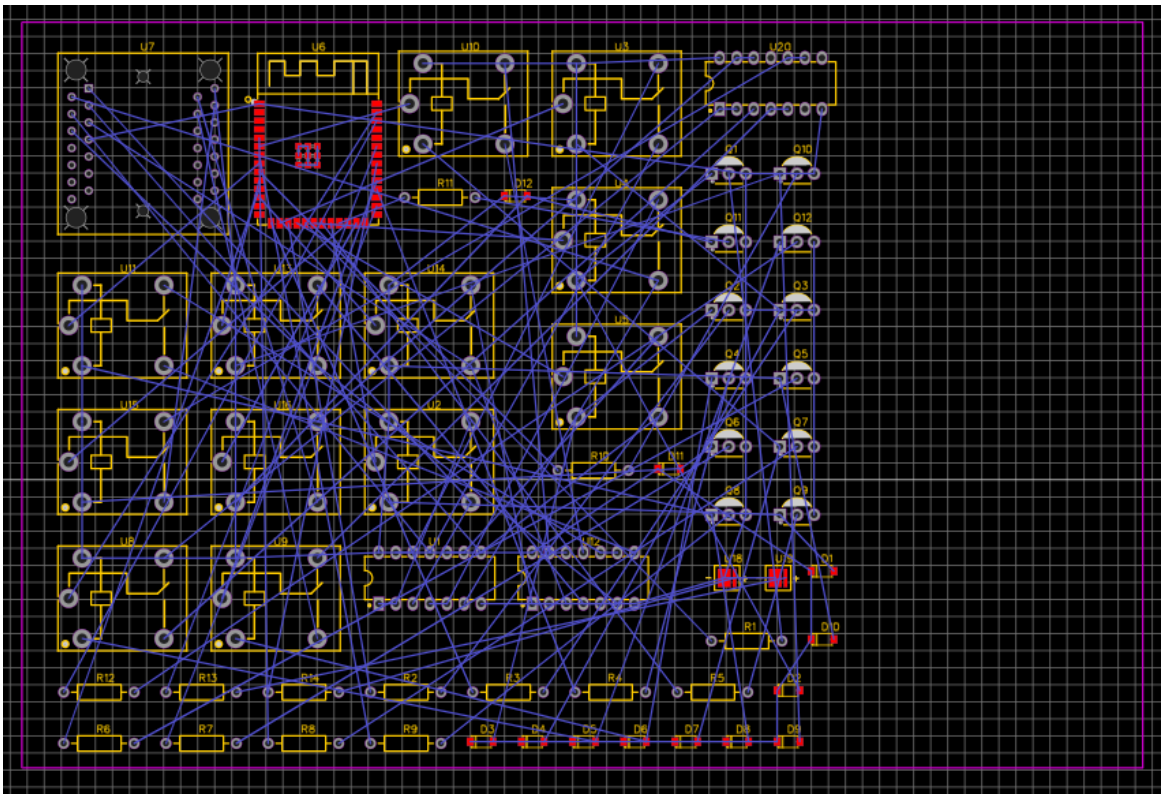

Figura 50: Trilhas do PCB

O projeto foi desenvolvido como um PCB de camada única, de forma a simplificar o processo de produção e reduzir custos. Entretanto, o layout foi pensado para minimizar cruzamentos e interferências. Estas imagens são uma representação visual do que seria o PCB final caso tivesse sido fabricado. Embora o PCB não tenha sido fabricado, o projeto está preparado para futuras implementações, e o design pode ser fabricado sem ajustes significativos.

5.7 Custos

Após a realização do projeto foi realizado um balanço dos custos do mesmo, de forma a verificar a sua viabilidade económica. No contexto deste projeto, que tinha como objetivo implementar um sistema para testar portas lógicas básicas utilizando relés controlados por um microcontrolador ESP32 S3, foi realizado um levantamento detalhado de todos os matérias e componentes utilizados. A tabela abaixo apresenta todos os materiais e quantidades dos mesmos, como o valor unitário e total de cada componente.

Tabela 14: Tabela dos custos associados ao projeto

Material	Quantidade	Valor unitário	Total
Fonte de tensão	1	0€ (cedida pela escola)	0€
Adaptador der tensão	1	6,60€	6,60€

Breadbord	2	3,99€	7,98€
Placa de circuito perfurada	1	1,20€	1,20€
Esp-32 S3 WROOM-1	1	7,07€	7,07€
Relés SRD – 05VDC-SL-C	12	0,92€	11,04€
Díodos 1N4002	12	0,10€	1,20€
Transístores BC546C	12	0,07€	0,84€
Circuito Integrado 74HCT00	3	0,61€	1,83€
ZIF *Socket* de 20 pinos	1	1,66€	1,66€
LED	2	0,17€	0,34€
Resistência 1000Ω	14	0,42€	5,88€
Total			**45,64€**

O custo total do projeto foi de 45,64€, isto diz respeito aos custos diretos do projeto, no entanto, houve matérias fornecidos pela Escola Superior de Tecnologia de Castelo Branco como a fonte de tensão que tornou o projeto mais económico. O fato de não ter sido fabricado o PCB aumentou a economia do projeto, pois seria algo que iria aumentar os custos do mesmo. Ainda assim, tendo em conta a diversidade de matérias utilizados verificamos que o projeto é viável do ponto de vista económico tendo em conta o seu custo-benefício.

Conclusão e Sugestões

O desenvolvimento do "Sistema de Teste Automático de Componentes Eletrónicos" demonstrou ser uma solução eficiente e acessível para a validação de portas lógicas básicas. A escolha de componentes de fácil acesso, juntamente com a utilização de tecnologias open-source como o Arduino IDE e o EasyEDA, permitiu a criação de um sistema funcional capaz de testar diversas configurações de portas lógicas de forma automatizada.

Este tipo de opções já existem em produções de larga escala, sendo muito utilizadas pelas industrias, no entanto, este projeto foi pensado de forma a ser implementado na vida escolar pelos alunos, aquando da montagem de pequenos circuitos elétricos, tornou-se muitas vezes necessário identificar se os componentes estavam funcionais ou não, de forma a serem utilizados, pois em alguns casos, somente quando algo não funciona como esperado, verificamos que a origem do problema pode estar associado ao mau funcionamento de algum componente. Este tipo de situações levou a ideia de construir um sistema automático para avaliar componentes eletrónicos, de forma a facilitar a identificação dos componentes em bom estado e consequentemente tornar mais rápida e eficaz a montagem de circuitos elétricos.

Os testes realizados comprovaram a eficácia do sistema, que foi capaz de identificar com precisão o estado de funcionamento das portas lógicas em diferentes cenários e configurações. A utilização de relés, controlados pelo ESP32 S3, proporcionou uma flexibilidade significativa na configuração das entradas e saídas, permitindo que o mesmo circuito fosse utilizado para testar diferentes tipos de portas lógicas sem a necessidade de alterações físicas.

Embora o PCB não tenha sido fabricado devido a limitações de tempo, o projeto demonstrou grande potencial para futuras expansões e melhorias. A fabricação do PCB, aliada à otimização do layout, permitiria uma implementação mais compacta e profissional do sistema, aumentando ainda mais sua eficiência e facilidade de utilização.

Em resumo, o projeto atendeu aos objetivos propostos, oferecendo uma ferramenta útil para o teste automatizado de componentes eletrónicos, com potencial de aplicação em laboratórios de eletrónica.

No que se refere a trabalhos futuros penso que algo importante seria a fabricação do PCB, como podemos verificar sem o PCB o circuito possui muitas ligações, o que visualmente pode não ser apelativo e funcionalmente também, pois intencionalmente pode ser retirado algum fio e prejudicar todo o funcionamento do circuito, o que seria ultrapassado com a utilização de uma placa de circuito impresso.

Seria interessante também associar um teclado e um ecrã ao sistema, pois permitiria uma utilização mais portátil do sistema, pois deixaria de ser obrigatório ter sempre disponível um computador de forma a interagir com o sistema.

Em situações futuras este projeto poderia ainda incluir várias opções, mantendo aquilo que já foi realizado, poderia ser ampliado de forma a verificar a funcionalidade de outros componentes eletrónicos, como outras famílias de Circuitos integrados, bobines, condensadores, medições de resistências, medições de corrente e tensão, sendo estes apenas de alguns exemplos de algumas funcionalidades que poderiam ser adicionadas ao projeto.

Referências

Capuano, F. G. (2008). *Elementos da Eletrônica Digital.* São Paulo: Editora Érica.

Deshpande, P., Epili, V., Ghule, G., & Ratnaparkhi, A. (2023). Digital Semiconductor Testing Methodologies. *Fourth International Conference on Electronics and Sustainable Communication Systems*, (pp. 316-321). Coimbatore, India.

Dhaker, P. (30 de December de 2019). *Introduction to the SPI Interface*. Obtido de Embbedded Computing Design: https://embeddedcomputing.com/technology/software-and-os/ides-application-programming/introduction-to-spi-interface?gad_source=1&gclid=CjwKCAjw74e1BhBnEiwAbqOAjNRNXGEcUNM5lQcsEo_-x2RL-wVaFVjd9TNWhusZV7k4YyLkDcv3UhoCwxQQAvD_BwE

Diegues, A. C., & Roselino, J. E. (2012). Competitive dynamics and innovation in high tech activities: an analysis of Computer Equipment and Semiconductor. *Gestão & Produção*, pp. 481-492.

Filho, A. A. (2022). *Engenharias - pesquisas sobre desenvolvimentos e inovações.* Brasil: Editora Dialética.

Malvino, A. P., & Brown, J. A. (1993). *Digital Computer Electronics.* United States of America: GLENCOE.

Malvino, A., & Bates, D. (2016). *Eletrônica.* Porto Alegre: AMGH Editora LTda.

Petruzella, F. D. (2014). *Eletrotécnica I.* Porto Alegre: BOOKMAN.

Pieper, J. M. (2014). *Automatic Measurement Control.* Munchen, Germany: ROHDE&SCHWARZ.

Platt, C. (2012). *Encyclopedia of Eletronic Components vol.1.* Sebastopol, CA: O'REILLY.

Sciotex. (n.d.). *Automatic Test Equipment – A Guide to ATE Test Systems*. Obtido de Sciotex: https://sciotex.com/what-is-ate-automatic-test-equipment-defined/#

Sedra, A. S. (2017). *Instructor's Solutions Manual for Microeletronic Circuits.* New York: OXFORD UNIVERSITY PRESS.

Silva, M. d. (1996). *Introdução aos Circuitos Elétricos E Eletrónicos.* Lisboa: Fundação Calouste Gulbenkian.

Silva, M. d. (2003). *Circuitos com Transistores Bipolares e Mos.* Lisboa: Fundação Calouste Gulbenkian.

Técnico, I. S. (1998). *Instrumentação e Medidas.* Lisboa: Secção Folhas.

TERADYNE. (21 de Fevereiro de 2023). *Advanced Digital Process Nodes Drive Semiconductor Test Innovations*. Obtido de TERADYNE: https://www.teradyne.com/2023/02/21/advanced-digital-process-nodes-drive-semiconductor-test-innovations/

Tocci, R. J., Widmer, N. S., & Moss, G. L. (2007). *Digital Systems Principles and Applications.* New Jersey : Pearson Education International.

Wakerly, J. F. (2006). *Digital Design: Principles and Practices.* New Jersey: Pearson Education.