



Técnico de Gestão e Programação de Sistemas Informáticos

Prova de Aptidão Profissional

## Casa Inteligente



Cantanhede

Tiago Dinis, 12º TGPSI, Nº 17

2023 / 2024

Cofinanciado por:



Escola Técnico Profissional de Cantanhede

Técnico de Gestão e Programação de Sistemas Informáticos

Prova de Aptidão Profissional

Casa Inteligente

Tiago Dinis, Nº 17, 12º TGPSI

Equipa de Acompanhamento

Michael Teixeira | Sónia Moço | Elisabete Cavaleiro | Ana Marques

Cantanhede,

Ano Letivo 2023/2024

O sucesso é a soma de pequenos esforços repetidos dia após dia.

*Robert Collier*

## Agradecimentos

Em primeiro lugar, quero agradecer à minha família pelo apoio incondicional dado durante todo o meu percurso escolar. Estiveram sempre presentes para me apoiar e ajudar nos momentos difíceis e festejar comigo nos momentos de sucesso. A família deve ser a base onde nos abrigamos em tempos de dificuldade, ajudando na nossa recuperação e servindo com um escudo contra os problemas que nos poderiam afetar ainda mais.

Depois, quero agradecer aos amigos que fiz ao longo deste percurso e que me proporcionaram momentos incríveis. Para mim, os amigos são como uma família “não de sangue”, que estão lá para nos ajudar e dar o apoio necessário quando a família não pode.

Quero também agradecer a todas as escolas que me acolheram durante o meu percurso académico e a todos os professores que me acompanharam nesta jornada, dando o seu melhor para transmitir o seu conhecimento para as gerações mais novas, proporcionando-lhes o conhecimento necessário para se tornarem bons cidadãos na sociedade e ingressar no mercado de trabalho.

Por último, mas não menos importante, quero agradecer à empresa que se disponibilizou a acolher-me durante o meu estágio nestes últimos meses do meu percurso académico, disponibilizando-me o conhecimento, o tempo e o material de que precisava para desenvolver a minha Prova de Aptidão Profissional, para além de novas experiências e aprendizagens no contexto real do trabalho.

## Índice Geral

Introdução (Português).....	1
Introduction (Inglês).....	3
Planeamento .....	5
Fundamentação.....	5
Recursos .....	6
Hardware .....	6
Software.....	8
Metodologia .....	10
Atividades .....	11
Fase de Definição .....	11
Fase de Desenvolvimento .....	11
Fase de Manutenção.....	12
Cronograma.....	12
Instalação de Software .....	13
Arduino IDE.....	13
Visual Studio Community 2022 .....	19
FlatCAM .....	22
Projeto.....	24
Criação do Modelo 3D .....	24
Criação do Circuito Elétrico .....	28
Tinkercad .....	28
Teste de Componentes .....	29
Sensor de Temperatura/Humidade .....	30
Sensor de Luminosidade.....	33
Círculo Final .....	35
Aplicação .....	43

Visual Studio – Interface Gráfica .....	45
Visual Studio – Objetos e Variáveis.....	47
Visual Studio - Formulário Principal.....	47
Visual Studio - Temporizador de Atualização da ComboBox.....	48
Visual Studio – Botão “Conectar” .....	49
Visual Studio – Botão “Escrever” .....	52
Visual Studio – Timer de Inatividade do Botão “Escrever” .....	55
Visual Studio – Botão “Limpar”.....	56
Visual Studio - Re却ão dos Dados .....	56
Visual Studio – Comunicação com o Arduino .....	60
Visual Studio - Botões de Controlo.....	63
Visual Studio - Habilitar/Desabilitar Botões .....	63
Visual Studio – Atualizar Botões.....	64
Visual Studio – Botão “Sair” .....	65
Visual Studio – Formulário Principal Fechado .....	66
ArduinolDE – Bibliotecas e Variáveis .....	67
ArduinolDE – Iniciação do Programa .....	68
ArduinolDE – Loop.....	69
ArduinolDE – Verificação do Estado dos LED’s .....	73
ArduinolDE – Controlo do Sistema de Luzes UV.....	74
ArduinolDE – Apresentação dos Valores dos Sensores.....	75
ArduinolDE – Função dos Sistemas da Casa.....	76
Placa de Circuitos Impressos (PCI) .....	77
EasyEDA.....	77
FlatCAM .....	85
Construção da Maquete.....	97
Conclusão .....	105
Webgrafia.....	107

## Índice de Figuras

Figura 1: Metodologia Em Cascata .....	10
Figura 2: Cronograma .....	12
Figura 3: Site Arduino - Página Principal .....	13
Figura 4: Site Arduino - Página de Download .....	14
Figura 5: Site Arduino - Página de Doação .....	14
Figura 6: Site Arduino - Página de Vinculação do E-mail.....	15
Figura 7: Arduino IDE - Janela de Aceitação do Termos da Licença .....	16
Figura 8: Arduino IDE - Janela de Seleção de Utilizadores .....	17
Figura 9: Arduino IDE - Janela de Seleção da Localização .....	17
Figura 10: Arduino IDE - Janela do Progresso da Instalação .....	18
Figura 11: Arduino IDE - Janela de Finalização da Instalação.....	18
Figura 12: Site Visual Studio - Página Principal.....	19
Figura 13: Visual Studio Installer - Janela Inicial .....	19
Figura 14: Visual Studio Installer - Janela de Download do Instalador .....	20
Figura 15: Visual Studio Installer - Janela de Edição do Visual Studio Community ....	20
Figura 16: Visual Studio Installer - Janela de Instalação do Visual Studio Community	21
Figura 17: Visual Studio Installer - Janela de Programas Instalados .....	21
Figura 18: FlatCAM - Página de Download .....	22
Figura 19: FlatCAM - Página de Documentação da Instalação.....	22
Figura 20: FlatCAM – Lista de Downloads .....	23
Figura 21: 1 <sup>a</sup> Fase da Construção do Modelo .....	24
Figura 22: 2 <sup>a</sup> Etapa da Construção do Modelo .....	25
Figura 23: Tinkercad – Versão 1 do Modelo 3D da Casa (Frente).....	25
Figura 24: Tinkercad - Versão 1 do Modelo 3D da Casa (Trás).....	26
Figura 25: Tinkercad - Versão 2 do Modelo 3D da Casa (Frente).....	26
Figura 26: Tinkercad - Versão 2 do Modelo 3D da Casa (Trás).....	27
Figura 27: Tinkercad - Versão 1 do Circuito Elétrico (Sem Sensores) .....	28
Figura 28: Aba De Pesquisa De Bibliotecas Do Arduino IDE.....	29
Figura 29: Biblioteca - DHT sensor library .....	30
Figura 30: Biblioteca - Adafruit Unified Sensor .....	30
Figura 31: Circuito Para O Sensor De Humidade/Temperatura .....	31
Figura 32: Circuito Para O Sensor De Luminosidade .....	33
Figura 33: Circuito Elétrico Final - Versão 1 .....	35
Figura 34: Exemplo De Um Dos Estados Do Dia .....	35

Figura 35: Janela Introdutória do Visual Studio .....	43
Figura 36: Janela de Escolha do Modelo do Projeto.....	44
Figura 37: Janela de Introdução do Nome e Localização do Projeto .....	44
Figura 38: Interface da Aplicação – Versão 1 .....	45
Figura 39: Interface da Aplicação - Versão 2.....	46
Figura 40: Interface da Aplicação - Versão 3.....	46
Figura 41: EasyEDA – Página Principal .....	77
Figura 42: EasyEDA – Janela de escolha da Versão .....	78
Figura 43: EasyEDA – Editor.....	79
Figura 44: EasyEDA - Página de Inicio de Sessão .....	79
Figura 45: EasyEDA - Escolha do Tipo de Uso e Nome.....	80
Figura 46: EasyEDA - Escolha dos Objetivos e do Nível de Experiência.....	80
Figura 47: EasyEDA - Escolha do Modo de Execução .....	81
Figura 48: EasyEDA - Página de Visualização dos Projetos .....	81
Figura 49: EasyEDA - Página de Criação do Projeto.....	82
Figura 50: EasyEDA - Esquema Elétrico da PCI .....	82
Figura 51: EasyEDA - "PCI Editável" .....	83
Figura 52: EasyEDA - PCI (Camada de Cima) .....	84
Figura 53: EasyEDA - PCI (Visualização 3D) .....	84
Figura 54: EasyEDA - Guardar como Ficheiro Gerber.....	85
Figura 55: Explorador de Ficheiros - Ficheiros Gerados pelo EasyEDA .....	85
Figura 56: FlatCAM - Camada de Baixo da PCI .....	86
Figura 57: FlatCAM - Aba Lateral .....	86
Figura 58: FlatCAM - Propriedades do Objeto Gerber.....	87
Figura 59: FlatCAM - Ferramenta de Isolamento.....	87
Figura 60: FlatCAM – Geometria da Placa .....	88
Figura 61: FlatCAM - Propriedades do Objeto de Geometria .....	89
Figura 62: FlatCAM - Rota da CNC .....	90
Figura 63: FlatCAM - Excerto de Código G-Code.....	90
Figura 64: FlatCAM - Propriedados do Objeto de Trabalho da CNC .....	91
Figura 65: MECH3 – Interface do MECH3.....	91
Figura 66: MECH3 - Controlo das Coordenadas da CNC.....	92
Figura 67: MECH3 - Controlo do ficheiro G-Code .....	92
Figura 68: MECH3 - Controlo do código G-Code .....	93
Figura 69: Teste - Marcador no Cartão.....	93
Figura 70: 1º Teste - Dremel na Madeira.....	94

---

Figura 71: 2º Teste - Dremel na Madeira.....	94
Figura 72: Foto 1 – Desenho das Trilhas.....	95
Figura 73: Foto 2 – Desenho das Trilhas.....	95
Figura 74: Foto 3 - Perfuração .....	96
Figura 75: Foto 4 - Resultado Final .....	96
Figura 76: Maquete - Furos das Bases e Parede .....	97
Figura 77: Maquete - Colagem da folha de espuma EVA.....	98
Figura 78: Maquete - Início da Construção da Casa.....	98
Figura 79: Maquete - Colagem das Peças .....	99
Figura 80: Maquete – Resultado Final (Sem Telhado) .....	99
Figura 81: Maquete - Retirada de Imperfeições da PCI.....	100
Figura 82: Maquete - Verificação da Continuidade da Corrente .....	100
Figura 83: Maquete - Aplicação de Massa de Soldar na PCI.....	101
Figura 84: Maquete - Soldadura dos Componentes .....	101
Figura 85: Maquete - PCI Finalizada (Trás).....	102
Figura 86: Maquete - PCI Finalizada (Frente).....	102
Figura 87: Maquete - Instalação da PCI na Casa .....	103
Figura 88: Maquete - Circuito Instalado na Casa.....	103
Figura 89: Resultado Final .....	104

## Índice de Tabelas

Tabela 1: Tabela dos Objetivos.....	5
-------------------------------------	---

## Introdução (Português)

Como projeto de Prova de Aptidão Profissional (PAP) decidi construir uma casa inteligente por várias razões. Um dos motivos e talvez o mais influenciador foi o facto de que, durante a minha vida, sempre tive bastante contacto com aparelhos eletrónicos que o meu pai utilizava em casa visto que é um profissional especializado nesta área. Sendo assim, sempre tive interesse em perceber como estes aparelhos funcionavam. Outro motivo para ter escolhido este projeto foi tentar automatizar algumas funcionalidades dentro e fora de casa com o objetivo de perceber que tarefas do dia-a-dia podem ser automatizadas de forma simples. Por último, mas não menos importante, outra importante razão para ter realizado este projeto foi para simplesmente divertir-me com o Arduino e explorar as suas funcionalidades, visto que este é o meu primeiro projeto com este tipo de material.

Este projeto consiste numa maquete de uma casa (no meu caso utilizei a casa dos Simpsons com modelo) com um pequeno jardim. As funcionalidades que implementei na casa são o abrir e fechar as portas e as janelas, ligar e desligar luzes, um sistema de rega automático e um sistema de luz ultravioleta. Todas estas funcionalidades estão representadas por LED's na maquete e cada sistema tem a sua cor que escolhi, dentro das cores que tinha disponíveis, consoante achei mais apropriado para o sistema.

Para controlar estas funcionalidades utilizei um Arduino Mega, alguns sensores (DHT11, LDR) e uma aplicação Windows Forms que desenvolvi. Para simular o circuito (embora apenas dos LED's) e criar um modelo 3D do projeto utilizei o Tinkercad e para desenvolver código utilizei o Arduino IDE para o Arduino e o Visual Studio Community 2022 para a aplicação.

Este trabalho está dividido em três etapas: fase de definição, fase de desenvolvimento e fase de manutenção.

Na primeira fase comecei por escolher que tipo de tecnologia queria usar neste trabalho. Depois, dentro das tecnologias escolhidas, tive que pensar de que forma podia aplicá-las num projeto. Com um tema em mente, precisei de procurar sobre o que era necessário para o realizar como aplicações, sites, componentes e materiais.

Cofinanciado por:



SELO DE CONFORMIDADE EQAVET  
GARANTIA DA QUALIDADE  
NA EDUCAÇÃO E FORMAÇÃO PROFISSIONAL

Na segunda fase, antes de tudo, decidi criar um modelo 3D da casa para poder ter uma melhor noção das dimensões, escala, estilo, materiais e a quantidade de componentes necessários para a sua realização. Para isso, utilizei a mesma ferramenta usada para a construção do modelo, o Tinkercad, e criei um esboço do circuito com os componentes disponíveis. Em relação aos componentes que não estavam disponíveis, optei por criar um circuito e código simples de forma individualmente para testar e perceber o seu funcionamento. No fim destes testes, juntei todos os componentes num circuito só e desenvolvi um código base com as funcionalidades essenciais.

Por fim, comecei a desenvolver a aplicação e, em simultâneo, a modificar o código base que já tinha criado para o Arduino, visto que ambos precisam de colaborar para o circuito funcione conforme a aplicação ordena.

Na última fase, a fase de manutenção, fiz testes e modificações constantemente a ambos os códigos, livrando-os de possíveis erros e tornando-os mais compactos e eficientes.

Este relatório pode ser dividido em quatro partes principais: criação do circuito, onde mostro o processo realizado para criar o circuito utilizado no projeto; criação da aplicação, onde mostro as etapas pelas quais passei para a criar assim como o código utilizado na mesma e no Arduino; criação da PCI (Placa de Circuitos Impressos), onde mostro o processo de criação da PCI utilizando uma máquina CNC; e construção da maquete, onde mostro como montei a maquete utilizada no projeto e a implantação do circuito na casa.

## Introduction (Inglês)

As my Professional Aptitude Teste (PAP) project, I decided to build a smart house for several reasons.

One of the main reasons, and perhaps the most influential, was the fact that throughout my life, I have always had significant contact with electronic devices that my father used at home, as he is a professional specialized in this area. Therefore, I have always been interested in understanding how these devices work. Another reason for choosing this project was to try to automate some functionalities inside and outside the house with the aim of understanding which daily tasks can be automated simply. Last but not least, another important reason for undertaking this project was simply to have fun with Arduino and explore its functionalities, as this is my first project with this type of material.

This project consists of a model of a house (in my case, I used the Simpsons' house as a model) with a small garden. The functionalities I implemented in the house include opening and closing doors and windows, turning lights on and off, an automatic irrigation system, and an ultraviolet light system. All these functionalities are represented by LEDs in the model, with each system having its color that I choose, from the available colors, as I found most appropriate for the system.

To control these functionalities, I used an Arduino Mega, some sensors (DHT11, LDR), and a Windows Forms application that I developed. To simulate the circuit (although only the LEDs) ad create a 3D model of the project, I used Tinkercad, and to develop the code, I used Arduino IDE for the Arduino and Visual Studio Community 2022 for the application.

This project is divided into three stages: definition phase, development phase, and maintenance phase.

In the first phase. I started by choosing what type of technology I wanted to use in this project. Then, within the chosen technologies, I had to think about how I could apply them to a project. With a theme in mind, I needed to research what was necessary to accomplish it, such as applications, websites, components, and materials.

In the second phase, first of all, I decided to create a 3D model of the house to have a better sense of the dimensions, scale, style, materials, and the number of components needed for its realization. For this, I used the same tool used for building the model, Tinkercad, and created a circuit sketch with the available components. Regarding the components that were not available, I chose to create a simple circuit and code individually to test and understand their functioning. At the end of these tests, I combined all the components into a single circuit and developed a base code with the essential functionalities.

Finally, I started developing the application and, simultaneously, modifying the base code I had already created for the Arduino, as both need to work together for the circuit to function as the applications commands.

In the last phase, the maintenance phase, I constantly teste and modified both codes, eliminating possible errors and making them more compact and efficient.

This report can be divided into four main parts: circuit creation where I show the process undertaken to create the circuit used in the project; application creation, where I show the stages I went through to create it, as well as the code used in it and in the Arduino; PCB (Printed Circuit Board) creation, where I show the process of creating the PCB using a CNC machine; and model construction, where I show how I assembled the model used in the project and implemented the circuit in the house.

## Planeamento

### Fundamentação

Escolhi fazer este projeto com o objetivo de aprofundar os meus conhecimentos na área da robótica e explorar novos componentes e suas funcionalidades. Para além disso, com este trabalho pretendo procurar e descobrir as automatizações que podem ser feitas numa casa, tornando-a mais eficiente e sustentável.

Apesar de já ter algum interesse por esta área há alguns anos, o meu primeiro contacto com este tipo de material foi no 11º ano, na disciplina de Arquitetura de Computadores, onde desenvolvemos um sistema de contagem de entradas e saídas para a cantina da escola com o objetivo de controlar o número de pessoas lá presentes.

Objetivos	
Gerais	Específicos
Terminar o Curso	<ul style="list-style-type: none"><li>• Terminar o curso com uma boa média.</li></ul>
Desenvolvimento da PAP	<ul style="list-style-type: none"><li>• Aumentar os meus conhecimentos em programação e eletrónica;</li><li>• Melhorar as minhas competências de concretização de projetos;</li><li>• Adquirir conhecimentos nas linguagens C++, C#</li><li>• Aplicar os conhecimentos adquiridos ao longo dos três anos do curso.</li><li>• Explorar novas tecnologias utilizadas para o desenvolvimento da PAP.</li><li>• Desenvolver competências na área de desenvolvimento de projetos de software e hardware.</li></ul>

Tabela 1: Tabela dos Objetivos

## Recursos

### **Hardware**

#### Computador Laptop (Pessoal)

- Intel® Core™ i7-8750H CPU @ 2.20GHz 2.21GHz
- NVIDIA GeForce GTX 1060 3GB
- RAM 16GB
- Disco M.2 NVMe 256GB
- Disco SSD 256GB
- Windows 11 Home 64 bits

#### Computador Desktop (CNC)

- Intel® Atom™ 230 1.60GHz
- RAM 0.99GB
- Disco HDD 160GB
- Windows XP Home 32 bits

#### T51 Starter Kit

- Arduino Mega 2560
- Gas Sensor MQ-2
- HC-SR04
- Stepper Driver
- ADXL345
- 8\*8 Matrix
- 1602 Display
- 4\*7 Segment
- 7 Segment

- 10k Potentiometer

- 74HC595

- SW-520D (x2)

- Active Buzzer

- Passive Buzzer

- SG90 Servo

- 830 Breadboard

- F-M Dupont wire

- Soil Humidity

- 2.54mm Pin

- Remote Control

- PIR Motion

- DHT11

- Joystick

- Battery Pack

- Key Switch (x4)

- Clock DS3231

- USB Cable

- Jumper Wire (x30)

- 5v Step Motor

- 5v Relay

- Flame

- RGB LED

- LM35

- IR Receiver

- CDS (x3)

Cofinanciado por:



- RFID Module
- Red LED (x5)
- Yellow LED (x5)
- Blue LED (x5)
- Resistência 220Ω (x8)
- Resistência 10kΩ (x5)
- Resistencia 1kΩ (x5)
- Cartão de Resistências
- CD
- Firm Packing

## **Software**

- Microsoft Office Professional Plus 2019
  - Requisitos Mínimos
    - Sistema Operativo – Windows 10
    - Processador – 1.6 GHz ou Mais Rápido
    - Memória RAM – 4Gb (64 bits)
    - Disco – 4Gb Espaço Disponível
    - Resolução Mínima – 1280px X 768px
- Microsoft Visual Studio Community 2022
  - Requisitos Mínimos
    - Sistema Operativo – Windows 10, Windows Server 2016
    - Processador – ARM64 ou x64
    - Memória RAM – 4Gb
    - Disco – 850Mb
    - Resolução Mínima – 1366px X 768px

- Arduino IDE

Requisitos Mínimos

- Sistema Operativo – Windows 10 ou Superior
- Arquitetura – x86
- Disco – 2Gb Espaço Disponível

- FlatCam

Requisitos

- Python 2.7 32-bit
- PyQt4
- Matplotlib 1.3.1
- Numpy 1.8
- Shapely 1.3 - GEOS
- RTree - SpatialIndex

## Metodologia

A metodologia que escolhi para a criação deste projeto foi a metodologia em cascata. Esta metodologia consiste em seis fases: requisitos, design, implementação, testes, implantação e manutenção.

**Requisitos:** São recolhidos e identificados os requisitos do sistema;

**Design:** É elaborado um design para o sistema de acordo com os seus requisitos;

**Implementação:** O código-fonte é escrito e o sistema é construído com base nas instruções estabelecidas na fase de design;

**Testes:** Após a implementação, são realizados testes ao sistema para verificar se cumpre com requisitos definidos;

**Implantação:** O software é implantado no hardware-alvo e ambiente para onde foi desenvolvido;

**Manutenção:** Depois de feita a implantação é feita um último apuramento, verificando se é necessário algum tipo de manutenção como correção de bugs, adaptação para novos requisitos ou alterações no ambiente ao redor.

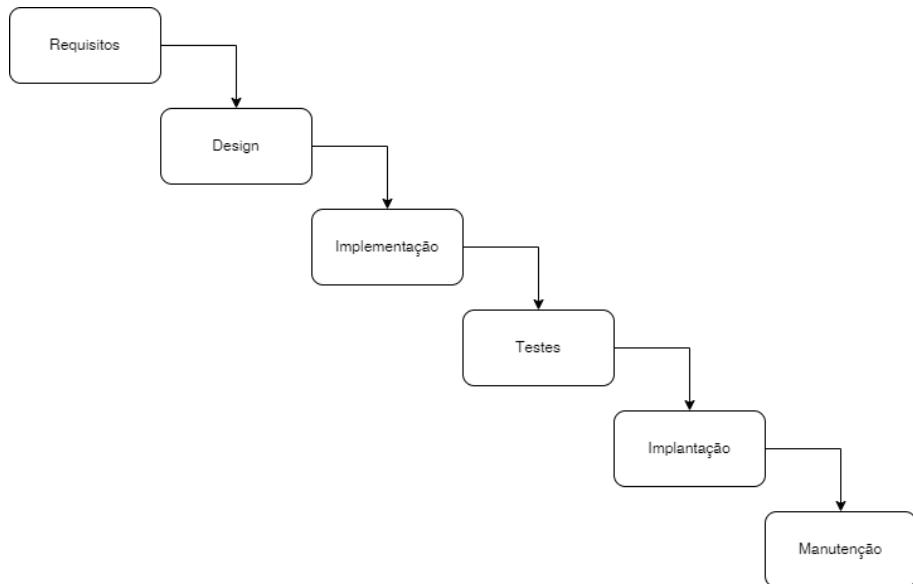


Figura 1: Metodologia Em Cascata

Eu escolhi esta metodologia porque acho que é a que mais se enquadra com o tipo de projeto que desenvolvi, já que é adequada para projeto com um objetivo bem definido, onde o desenvolvimento de cada etapa depende da etapa anterior.

## Atividades

A Prova de Aptidão Profissional irá ser desenvolvida em três fases. Cada fase é constituída por várias atividades. A primeira fase, a Fase de Definição tem como principais atividades: pesquisa bibliográfica, análise do sistema, análise dos requisitos e planeamento do projeto. A segunda fase, a Fase de Desenvolvimento é constituída pelas seguintes atividades, desenho, codificação e testes. A terceira e última fase, a Fase de Manutenção é onde encontramos as atividades de correção, adaptação e evolução.

### **Fase de Definição**

- ▶ Pesquisa Bibliográfica – etapa inicial do trabalho com o objetivo de reunir todas as informações necessárias para o desenvolvimento do projeto.
- ▶ Análise do Sistema – atividade onde se realizou o levantamento de todas as funcionalidades do projeto e a forma como elas vão funcionar. Nesta atividade foi também realizada a escolha das tecnologias para a realização do mesmo.
- ▶ Análise dos Requisitos – nesta atividade foi necessário fazer a verificação dos requisitos mínimos do Software necessário para implementação do projeto.
- ▶ Planeamento do Projeto – com esta atividade foi desenvolvido o cronograma em que se estabelece as balizas temporais para cada uma das fases do projeto.

### **Fase de Desenvolvimento**

- ▶ Desenho – nesta atividade faz-se o levantamento dos dados necessários para criação da Base de Dados utilizada no projeto, implementando o Diagrama Entidade Relacionamento e o Modelo de Dados. Faz-se ainda a estruturação da navegação do website e do backoffice.
- ▶ Codificação – atividade em que toda a parte de codificação é realizada.
- ▶ Testes – ao longo desta atividade realizam-se os testes às funcionalidades implementadas por forma a garantir que nesta fase as mesmas estão a funcionar de acordo com o que foi planeado.

## Fase de Manutenção

- ▶ Correção – é nesta atividade em que irei realizar correções ao projeto. Estas correções tanto podem ser à interface gráfica ou ao código realizado até ao momento. Se for necessário também se poderá fazer alterações estruturais à base de dados.
- ▶ Adaptação – depois de executado todos os testes e correções, é nesta atividade que vão ser realizadas adaptações ao projeto. Estas adaptações podem ser ao nível da interface gráfica ou ao nível da otimização de código. Estas adaptações surgem da necessidade do projeto ser testado por outras pessoas.
- ▶ Evolução – depois de ter os objetivos iniciais concluídos, é nesta fase, havendo tempo que irei tentar implementar novas funcionalidades que foram surgindo ao longo do desenvolvimento do projeto e que inicialmente não estavam previstas.

## Cronograma



Figura 2: Cronograma

## Instalação de Software

### Arduino IDE

Para conseguir programar e fazer upload do código para o Arduino escolhi a IDE fornecida pelo próprio site do Arduino, o Arduino IDE. Neste site, para além de fornecer o software necessário para programar o Arduino, é disponibilizado outro tipo de opções, como a venda dos vários tipos existentes de Arduino, Kits, documentos de apoio, tutoriais ou redes sociais.

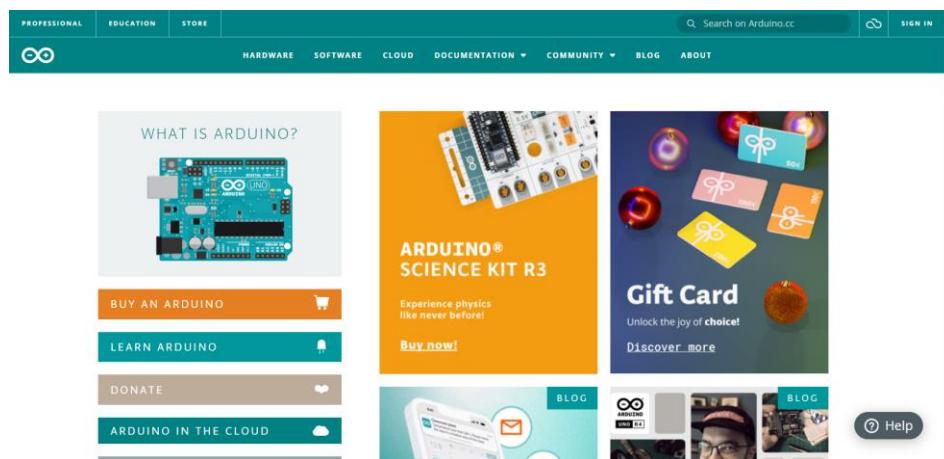


Figura 3: Site Arduino - Página Principal

Para poder instalar esta IDE, já com o site aberto, acedi à aba de software e escolhi a versão 2.2.1 do Arduino IDE por ser a mais recente, mais rápida, mais desenvolvida e com mais ferramentas de apoio. Como posso o sistema operativo Windows 11 no meu computador, instalei a IDE através da opção que corresponde ao Windows 10 ou mais recente.

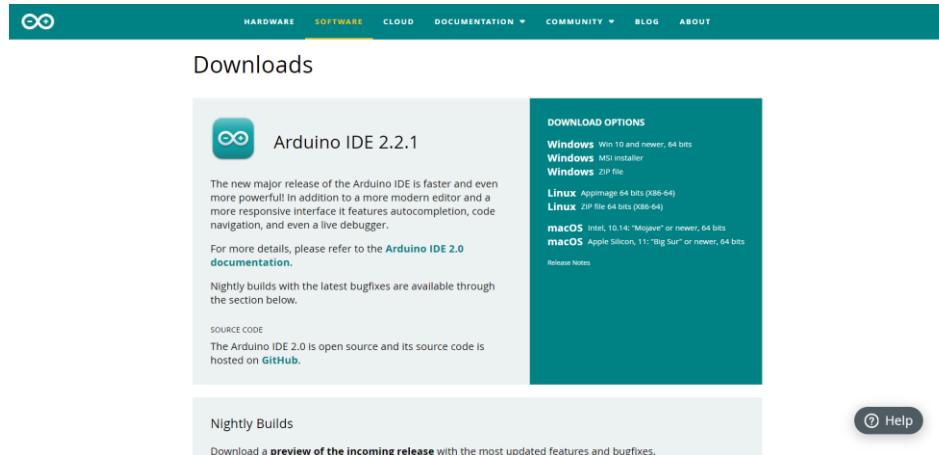


Figura 4: Site Arduino - Página de Download

Após selecionada a opção correspondente ao sistema operativo, é apresentada uma página onde é possível fazer uma doação à plataforma do Arduino, porém esta etapa é opcional. Para continuar basta pressionar a opção “Just Download”.



Figura 5: Site Arduino - Página de Doação

Continuando com a instalação, é exibida uma página onde é possível vincular o e-mail com o objetivo de receber notícias e projetos relacionados às novidades lançadas pelo Arduino. Este passo também é opcional e, mais uma vez, basta clicar no botão “Just Download”.

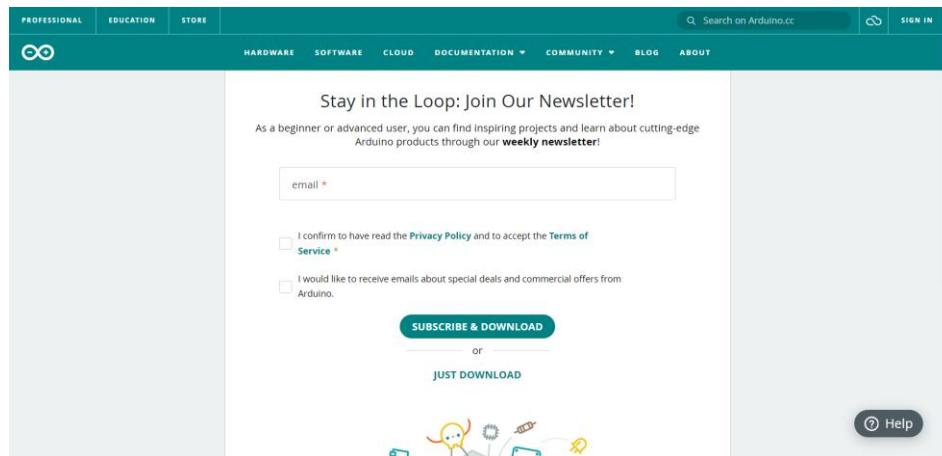


Figura 6: Site Arduino - Página de Vinculação do E-mail

Após realizadas todas estas etapas é, finalmente, iniciado o download do executável da IDE.

Agora, para instalar o Arduino IDE no computador basta ir ao local onde o ficheiro foi baixado e executá-lo. Antes de ser iniciada a instalação, são apresentadas algumas janelas: aceitação dos termos de licença, para que utilizadores será feita a instalação e a localização.

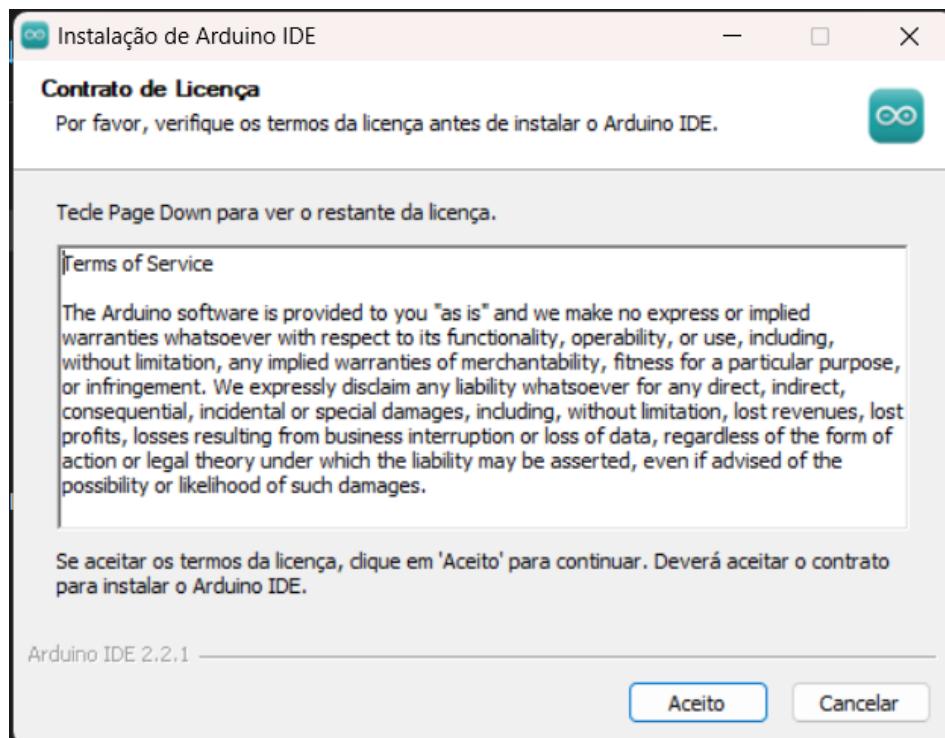


Figura 7: Arduino IDE - Janela de Aceitação do Termos da Licença

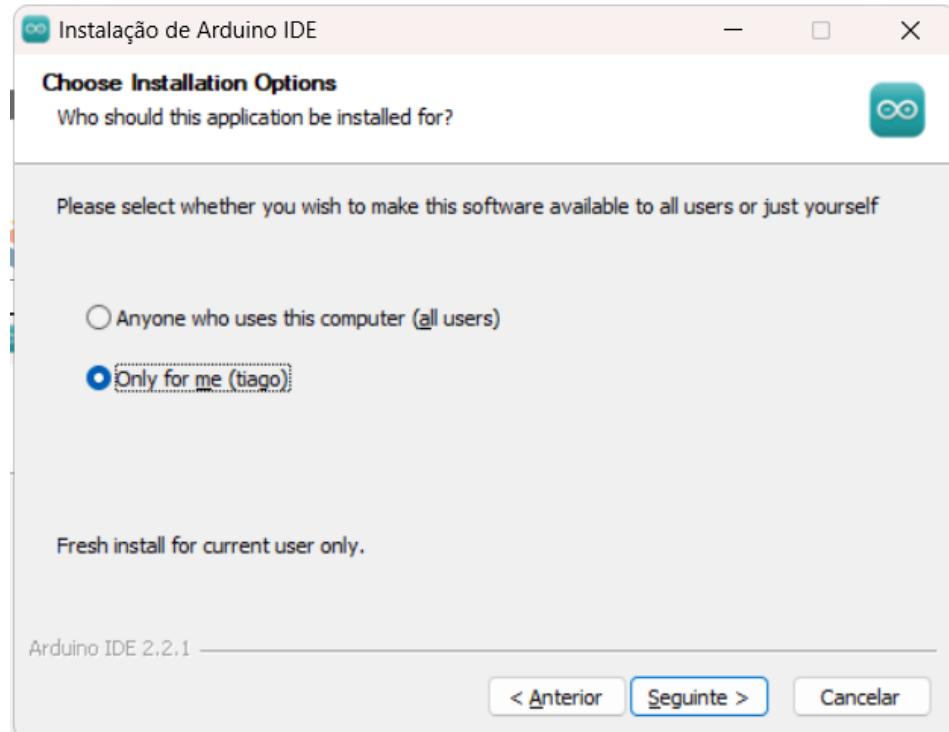


Figura 8: Arduino IDE - Janela de Seleção de Utilizadores

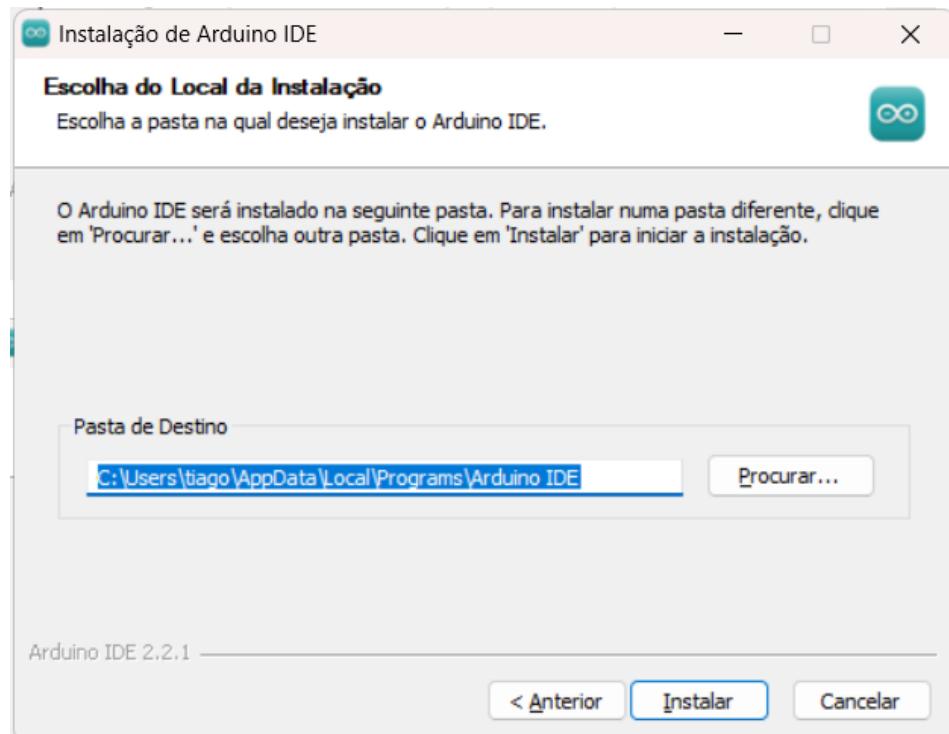


Figura 9: Arduino IDE - Janela de Seleção da Localização

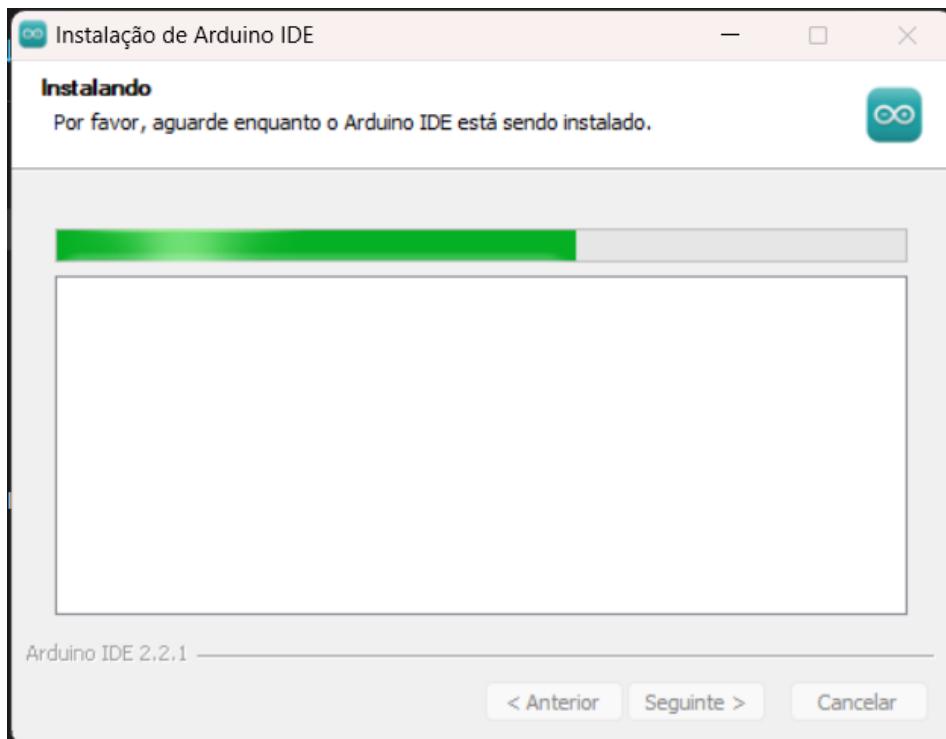


Figura 10: Arduino IDE - Janela do Progresso da Instalação

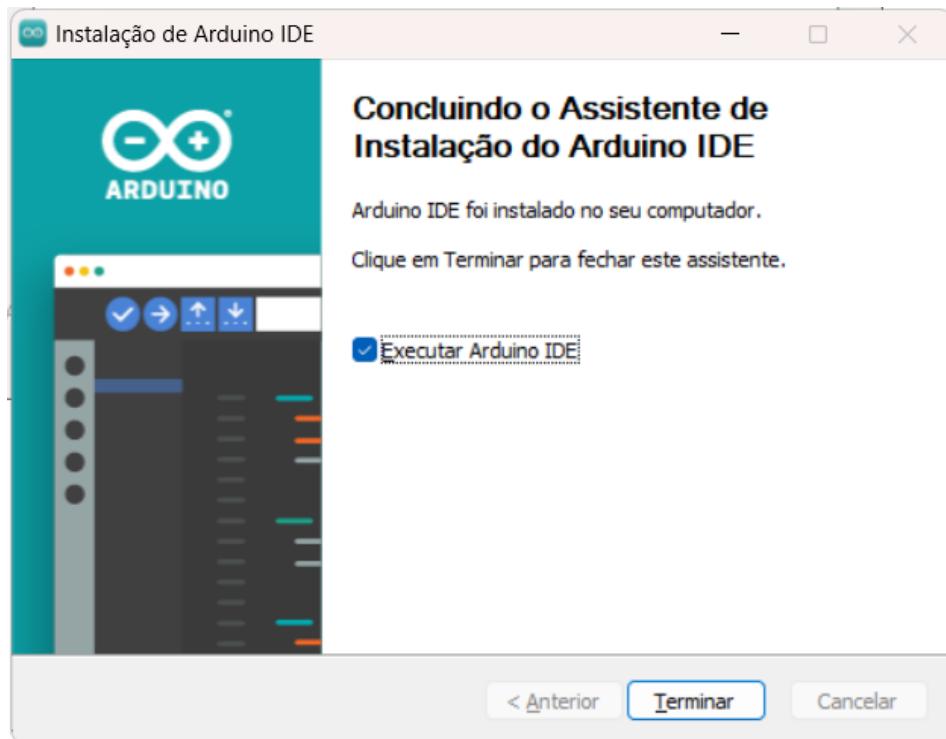


Figura 11: Arduino IDE - Janela de Finalização da Instalação

## Visual Studio Community 2022

Para criar a aplicação que controla o Arduino utilizei o Visual Studio Community 2022. Para instalar esta ferramenta existem duas opções: instalar pela web ou instalar pela Microsoft Store. Eu optei por instalar pela web.

Para começar, acedi ao site oficial da Microsoft do Visual Studio e cliquei no botão para baixar a versão Visual Studio Community.

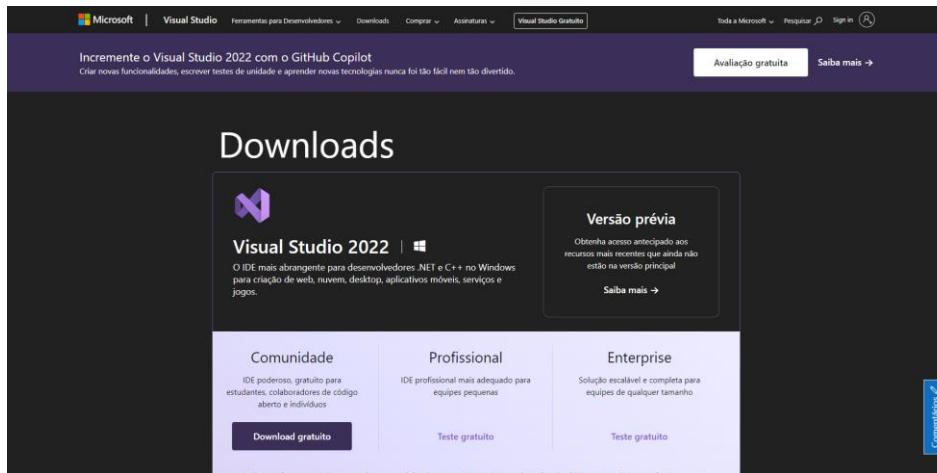


Figura 12: Site Visual Studio - Página Principal

Após isso, é baixado um executável com o nome “VisualStudioSetup.exe”. Para começar a instalar o Visual Studio basta executar esse ficheiro.

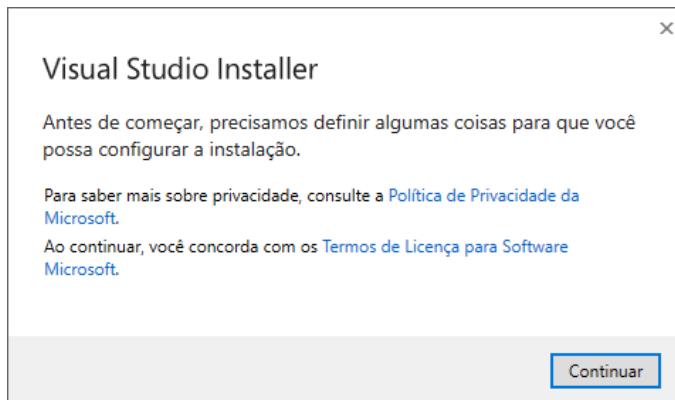


Figura 13: Visual Studio Installer - Janela Inicial

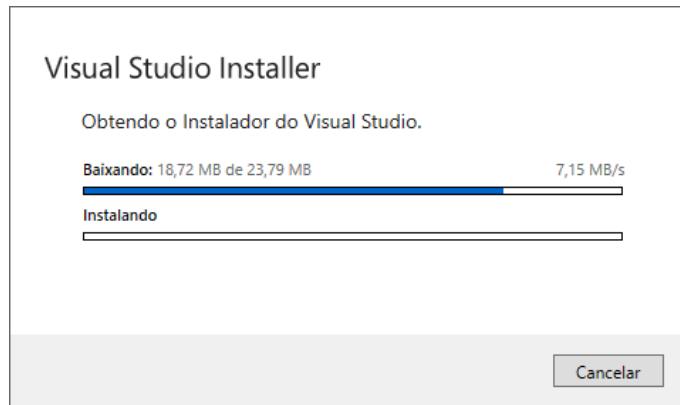


Figura 14: Visual Studio Installer - Janela de Download do Instalador

No fim da instalação, será aberta uma janela no Visual Studio Installer para editar o núcleo do Visual Studio Community. No meu caso, como pretendo criar uma aplicação para desktop, selecionei a opção “Desenvolvimento para desktop com .NET”.

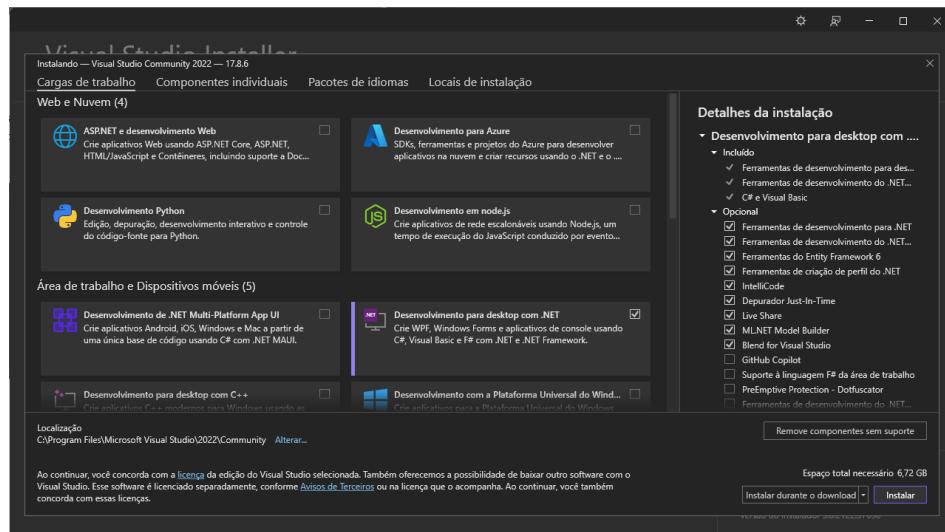


Figura 15: Visual Studio Installer - Janela de Edição do Visual Studio Community

Depois de feita a edição do núcleo, é começado o download do Visual Studio Community.

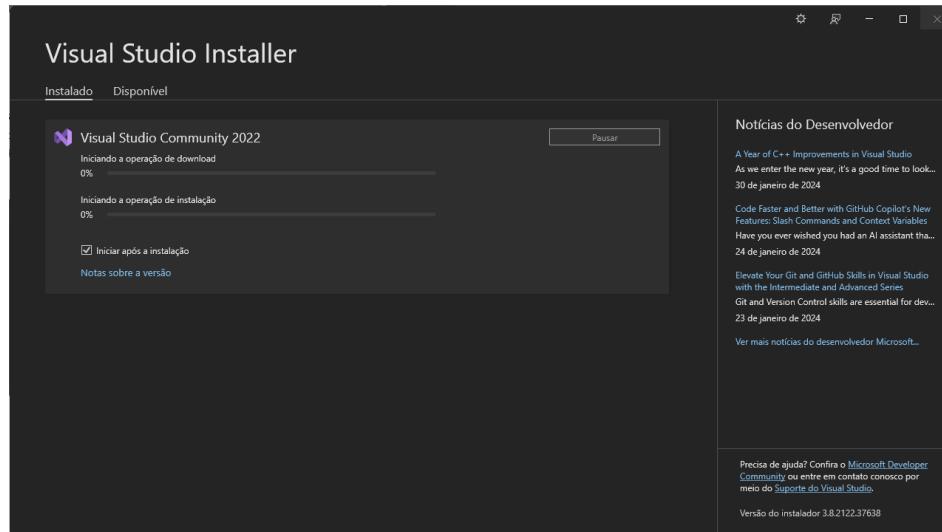


Figura 16: Visual Studio Installer - Janela de Instalação do Visual Studio Community

No fim da instalação, o Visual Studio Community será executado e aberto.

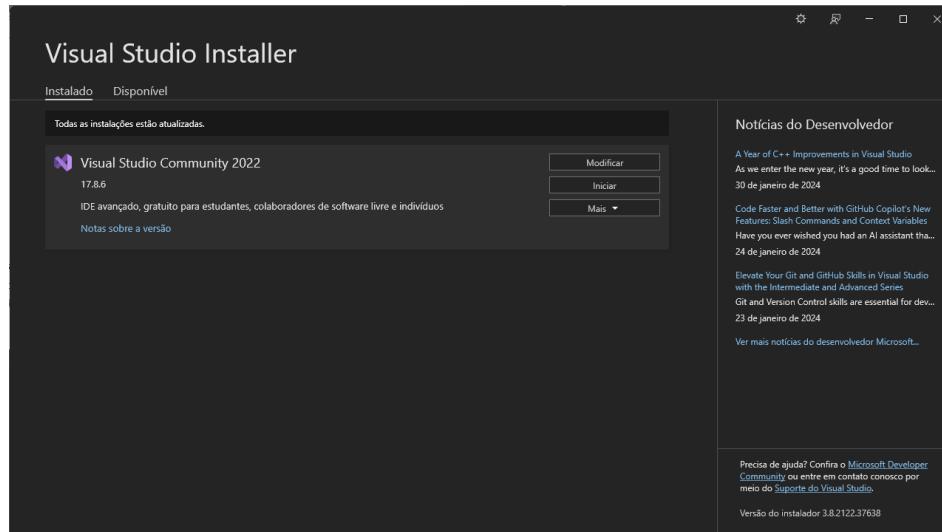


Figura 17: Visual Studio Installer - Janela de Programas Instalados

## **FlatCAM**

Para poder utilizar realizar a PCI (placa de circuitos impressos) precisei de utilizar um software que convertesse ficheiros Gerber para instruções em G-Code para poderem serem interpretadas pela CNC (Computer Numeric Control), por isso utilizei o FlatCAM por recomendação.

Para instalar este software, em primeiro lugar, acedi à sua página web, fui à aba “Download” e cliquei na hiperligação “Installation”, no fundo da página.

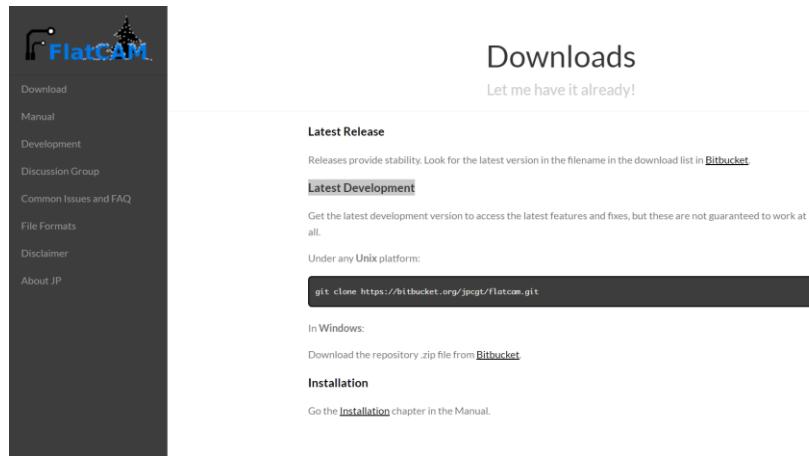


Figura 18: FlatCAM - Página de Download

Depois, na aba “2.1 Microsoft Windows”, cliquei na hiperligação “repository”.

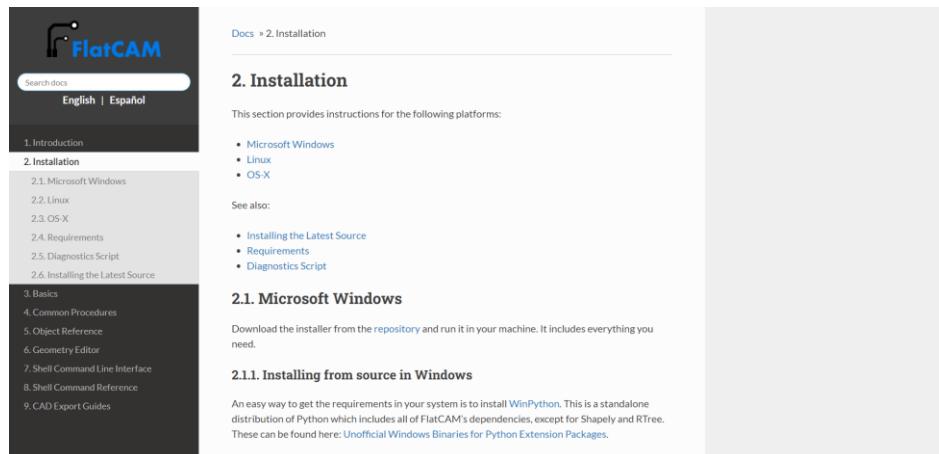


Figura 19: FlatCAM - Página de Documentação da Instalação

Por fim, no repositório, baixei a versão “FlatCAM\_beta\_8.994\_x64\_installer.exe”.

Name	Size	Uploaded by	Downloads	Date
Download repository	113.2 MB			
FlatCAM_beta_8.994_x86_installer.exe	64.7 MB	Marius Stanciu	12171	2023-05-04
FlatCAM_beta_8.994_x64_installer.exe	112.7 MB	Marius Stanciu	35225	2023-05-04
request_FlatCAM_beta_8.993_x64_installer.exe	110.4 MB	Marius Stanciu	4436	2023-05-04
request_FlatCAM_beta_8.993_x86_installer.exe	64.3 MB	Marius Stanciu	930	2023-05-02
request_FlatCAM_beta_8.991_x64_installer.exe	90.8 MB	Marius Stanciu	6558	2023-04-04

Figura 20: FlatCAM – Lista de Downloads

Feito isso, é baixado um executável que, ao executar, será feita a configuração da instalação, semelhante à que foi mostrada anteriormente para o ArduinolDE.

## Projeto

### Criação do Modelo 3D

Antes de iniciar a montagem do circuito e a programação, optei por criar um modelo 3D da casa que pretendia fazer, proporcionando-me uma referência inicial das suas dimensões e a quantidade de componentes necessários para o circuito.

Para esta tarefa, aproveitei uma das ferramentas disponibilizadas no site onde iria posteriormente montar o circuito elétrico, o Tinkercad, visto que para além de possibilitar a preparação de um circuito, também permite criar projetos em 3D.

Primeiramente, comecei por criar uma casa rés-do-chão com telhado com dois polígonos simples.

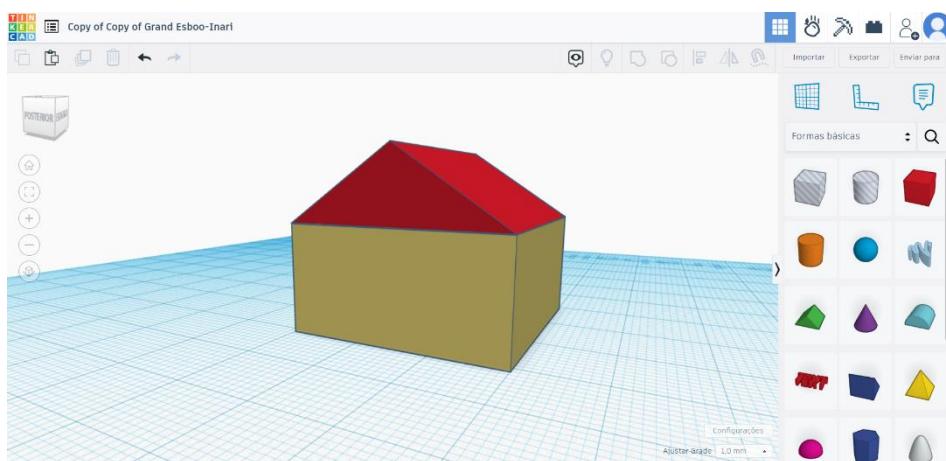


Figura 21: 1<sup>a</sup> Fase da Construção do Modelo

Depois, para representar a área da horta, utilizei um cubo e redimensionei-o para formar apenas uma fina camada verde no chão.

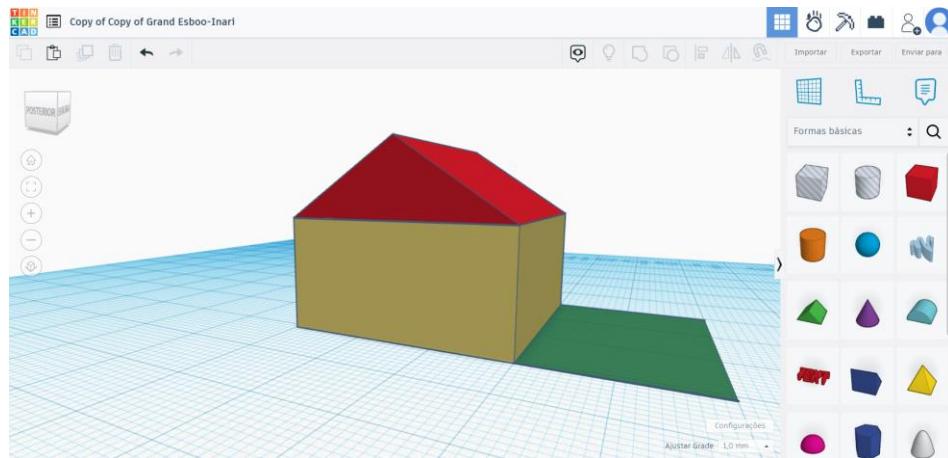


Figura 22: 2<sup>a</sup> Etapa da Construção do Modelo

Por fim, para identificar os locais das portas e janelas e determinar a sua quantidade utilizei novamente cubos e alterei as suas dimensões para formarem estes objetos, diminuindo a sua espessura para ficarem alinhados com as paredes.

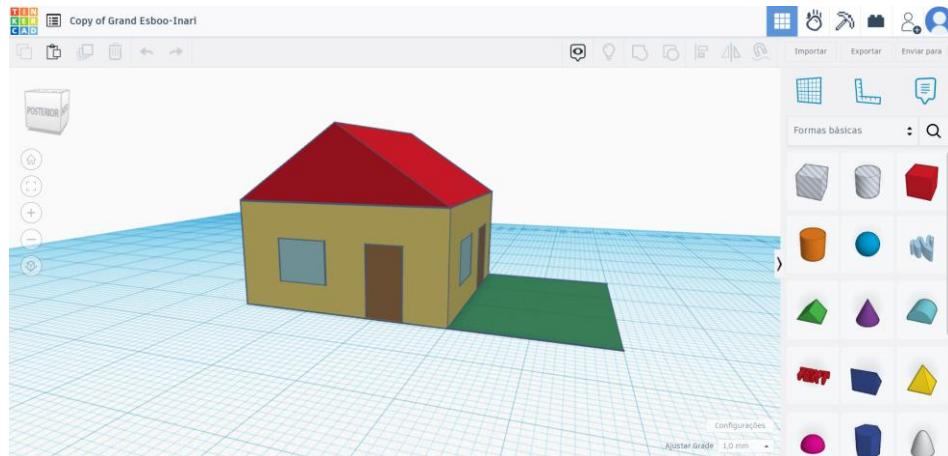


Figura 23: Tinkercad – Versão 1 do Modelo 3D da Casa (Frente)

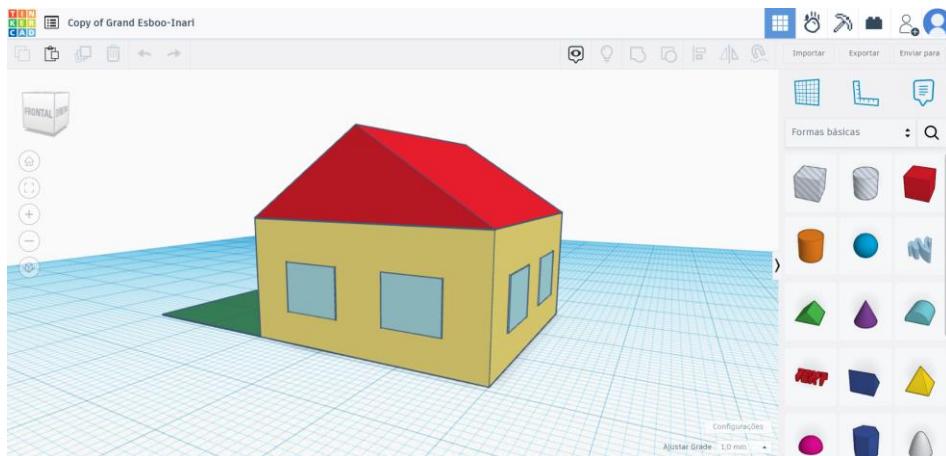


Figura 24: Tinkercad - Versão 1 do Modelo 3D da Casa (Trás)

Ao explorar melhor esta ferramenta descobri alguns objetos pré-fabricados que poderia implementar no meu modelo, como portas e janelas detalhadas e uma cerca para a horta.

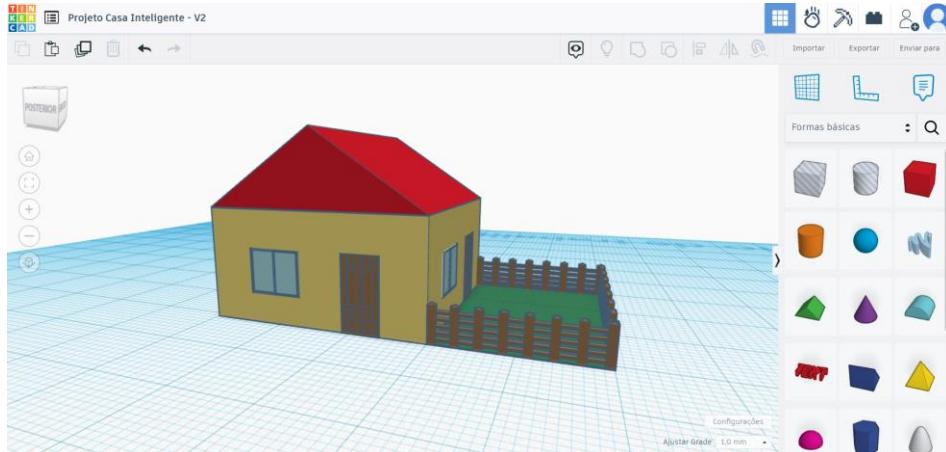


Figura 25: Tinkercad - Versão 2 do Modelo 3D da Casa (Frente)

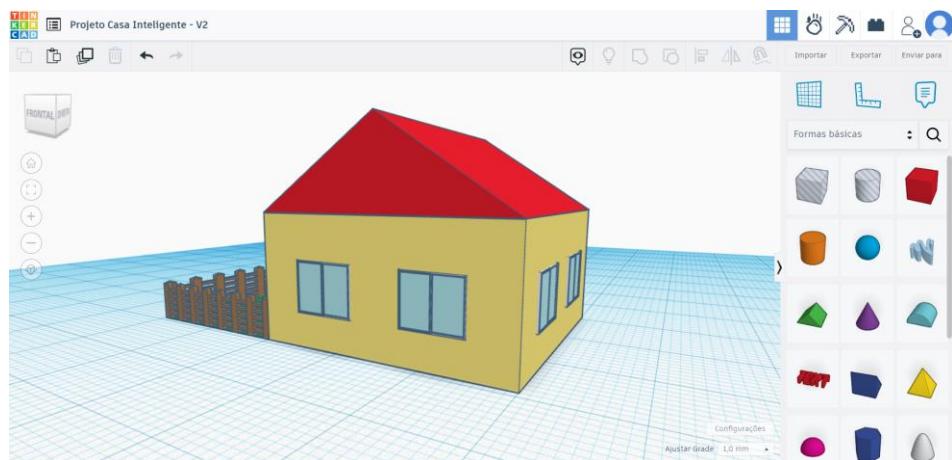


Figura 26: Tinkercad - Versão 2 do Modelo 3D da Casa (Trás)

## Criação do Circuito Elétrico

### Tinkercad

Como referi anteriormente, utilizei também o Tinkercad para planejar o circuito elétrico para evitar a probabilidade de estragar ou queimar componentes. Ao criar um modelo 3D da casa, tive uma melhor noção da quantidade de materiais que iria precisar.

Para representar todas a funções da casa decidi utilizar luzes LED para indicar se o sistema correspondente está ativo ou não. Como é possível ver no modelo anterior, a casa que idealizei possuí 6 janelas (1 na parte da frente, 2 do lado esquerdo, 2 atrás, 1 do lado do jardim) e 2 portas (1 na parte da frente e 1 do lado do jardim). Para além disso possui iluminação interior e, para o jardim, um sistema de rega e um sistema de luz ultravioleta.

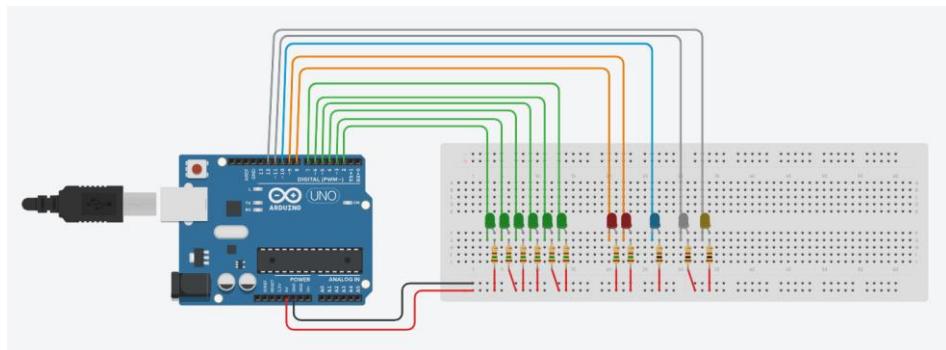


Figura 27: Tinkercad - Versão 1 do Circuito Elétrico (Sem Sensores)

Para tornar a distinção entre sistemas mais rápida e fácil, escolhi atribuir uma cor diferente a cada um sendo o verde para as janelas, o vermelho para as portas, o azul para a rega, o branco para a luz ultravioleta e o amarelo para a iluminação interior.

Como o Tinkercad não possui todos os componentes necessários para a realização deste projeto, tive que adaptar e simplificar o circuito com as partes disponíveis.

## Teste de Componentes

Como referido anteriormente, o Tinkercad não possui todos os componentes necessários para a realização do meu projeto. Por esse motivo, para realizar o trabalho sem danificar os componentes e entender melhor o seu funcionamento decidi criar um circuito simples individual para cada componente.

Para realizar os testes que pretendia fazer era necessário instalar as bibliotecas específicas de cada componente. Estas bibliotecas podem ser encontradas na internet em diversos sites ou, como no meu caso, estão disponíveis para pesquisa na própria IDE do Arduino.

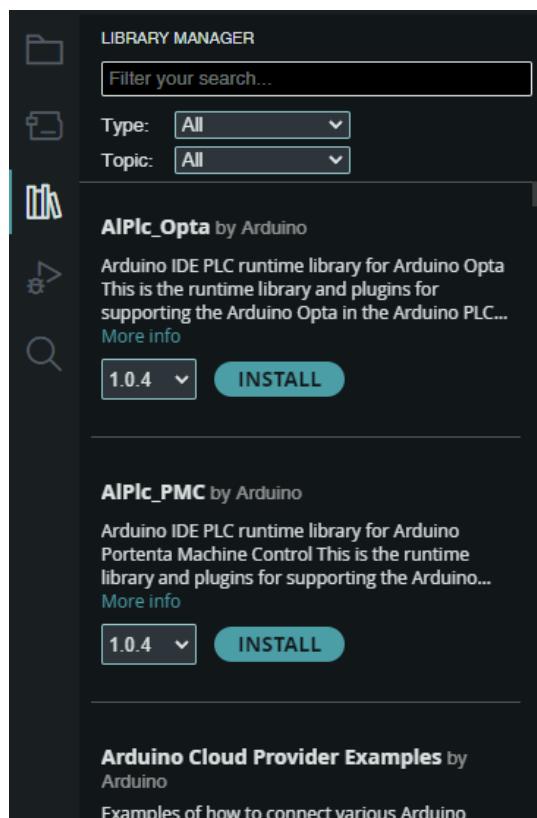


Figura 28: Aba De Pesquisa De Bibliotecas Do Arduino IDE

### Sensor de Temperatura/Humidade

Para começar escolhi testar o sensor de temperatura/humidade (DHT11). O sensor DHT11 (Digital Humidity and Temperature) é um sensor que é capaz de medir tanto a temperatura, entre 0º e 50º, como a humidade, entre 20% a 90%, o que tornou a configuração dos sensores mais rápida.

Para poder utilizar este sensor no circuito, precisei de instalar duas bibliotecas: *DHT sensor library*, *Adafruit Unified Sensor*.

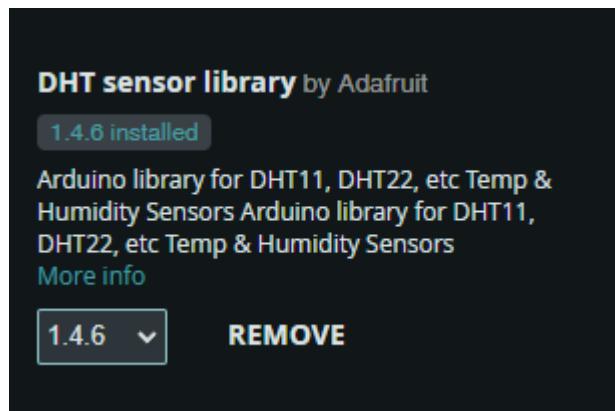


Figura 29: Biblioteca - DHT sensor library

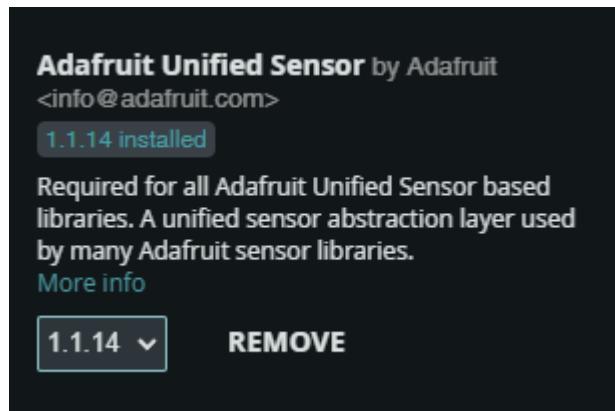


Figura 30: Biblioteca - Adafruit Unified Sensor

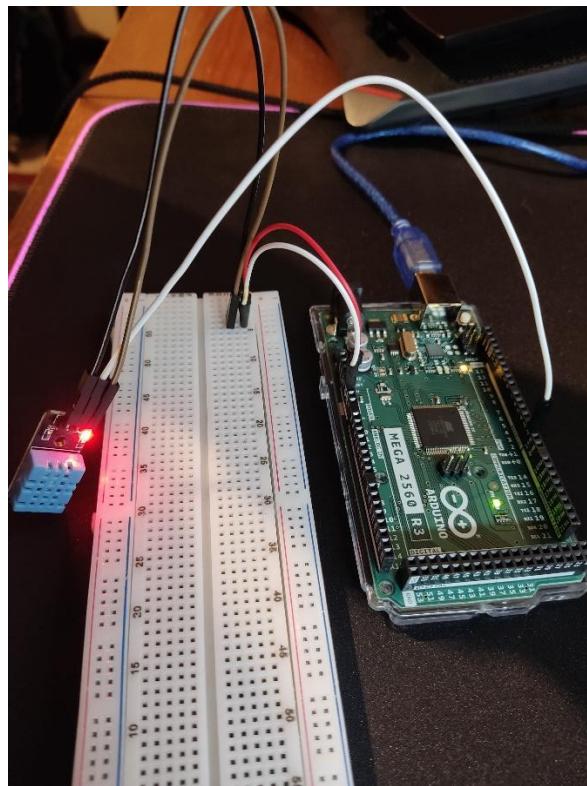


Figura 31: Circuito Para O Sensor De Humidade/Temperatura

```
//Inclui a biblioteca do sensor
#include "DHT.h"

//Define o pino 2 do sensor
#define DHTPIN 2

//Indicação da versão do sensor
#define DHTYPE DHT11

//Cria um objeto dht com o pino e tipo de sensor
DHT dht(DHTPIN, DHTYPE);

void setup() {
    //Inicia o monitor serial e o sensor
    Serial.begin(9600);
    dht.begin();
}
```

Cofinanciado por:

```
void loop() {  
  
    //Criar uma variável para cada medição e atribui-lhe o respetivo valor  
    float h = dht.readHumidity();  
    float t = dht.readTemperature();  
  
    //Verifica se a leitura foi bem-sucedida  
    if (isnan(h) || isnan(t)){  
        Serial.println("Falha de leitura do sensor DHT!");  
    }  
  
    //Apresenta o resultado para o monitor serial  
    Serial.println("Temperatura:");  
    Serial.println(t);  
    Serial.println("Humidade:");  
    Serial.println(h);  
    delay(1000);  
}
```

### **Sensor de Luminosidade**

Para medir a luminosidade utilizei o sensor LDR (Light Dependent Resistor). Este componente é classificado como um componente passivo, ou seja, não gera sinais. Este sensor funciona, basicamente, da mesma forma que uma resistência comum com a única diferença de variar a sua capacidade de resistência consoante a luz que lhe está a incidir. Por ser um componente passivo, não é necessário instalar nenhuma biblioteca para o utilizar.

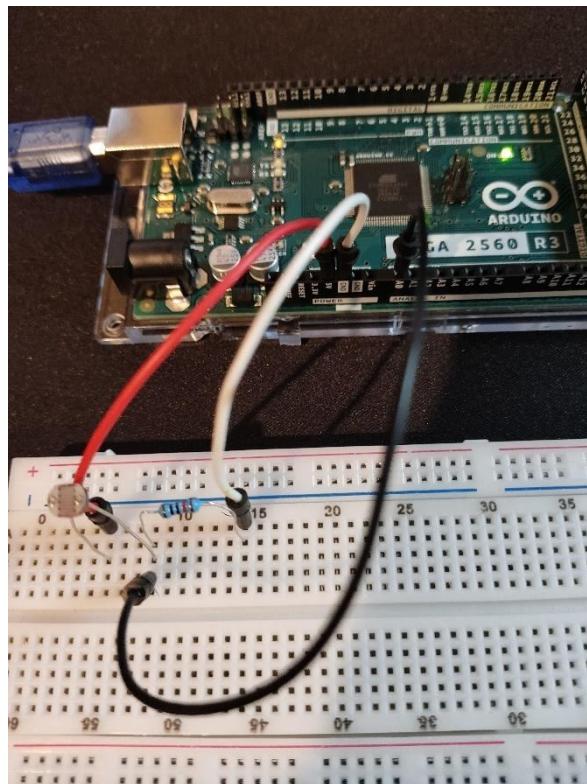


Figura 32: Circuito Para O Sensor De Luminosidade

#### **//Atribui o pino A0 à variável Idr**

```
int Idr = A0;
```

#### **//Define a variável do valor do sensor**

```
int valorldr;
```

```
void setup() {
```

#### **//Inicia o monitor serial**

Cofinanciado por:



```
Serial.begin(9600);

//Define que a variável Idr é um input
pinMode(Idr, INPUT);

}

void loop() {

//Atribui o valor do sensor à variável
valorldr = analogRead(Idr);

//Apresenta o valor para o monitor serial
Serial.println("Valor Recebido:");
Serial.println(valorldr);
delay(1000);

}
```

### Círculo Final

Depois de ter testado todos os componentes e entender a forma como funcionam, montei o circuito final do projeto. Neste circuito estão conectados os dois sensores que testei anteriormente assim como um LED para cada sistema da casa.

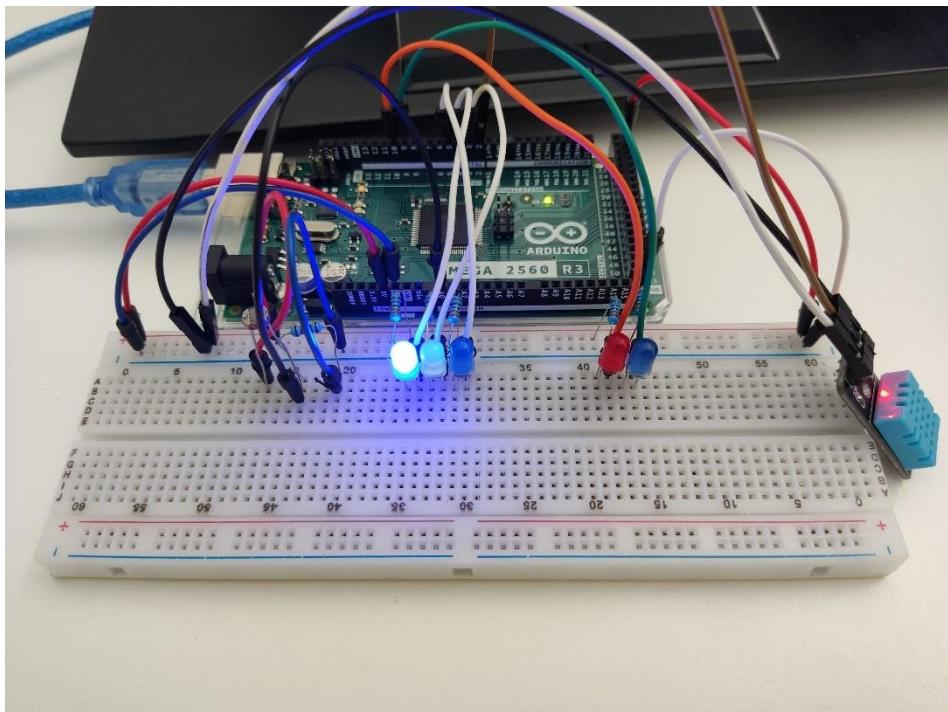


Figura 33: Circuito Elétrico Final - Versão 1

Tendo já um circuito funcional, criei um código básico para fazer o seu controlo e mostrar os dados recolhidos pelos sensores para o monitor serial do ArduinoIDE.

Para isso, comecei por elaborar um bloco de notas com os estados do dia para determinar quais sistemas deveriam estar ligados ou desligados com base nos valores dos sensores.

<b>Dia, Seco, Quente</b>
- Rega: Ligado
- Luz: Desligado
- Janelas: Ligado
- Portas: Ligado

Figura 34: Exemplo De Um Dos Estados Do Dia

Na primeira linha está o estado de cada sensor. Neste caso é um dia seco e quente, fazendo com que o sistema de rega ligue e as portas e janelas abram. Os parâmetros que usei para determinar o estado do ambiente envolvente do circuito foram os seguintes:

- **Sensor Luminosidade**
  - 0 – 499: Noite
  - 500 – 1000: Dia
- **Sensor DHT / Humidade**
  - 0% - 59%: Seco
  - 60% - 100%: Húmido
- **Sensor DHT / Temperatura**
  - °C < 25: Frio
  - °C >= 25: Quente

E para determinar que sistemas deviam ou não estar ligados utilizei os seguintes critérios:

- **Dia / Seco / Quente**
  - Luzes: Desligado
  - Rega: Ligado
  - Janelas: Ligado
  - Portas: Ligado
- **Dia / Seco / Frio**
  - Luzes: Desligado
  - Rega: Ligado
  - Janelas: Desligado
  - Portas: Desligado
- **Dia / Húmido / Quente**
  - Luzes: Desligado
  - Rega: Desligado
  - Janelas: Ligado
  - Portas: Ligado

- **Dia / Húmido / Frio**

- Luzes: Desligado
- Rega: Desligado
- Janelas: Desligado
- Portas: Desligado

- **Noite / Seco / Quente**

- Luzes: Ligado
- Rega: Desligado
- Janelas: Ligado
- Portas: Desligado

- **Noite / Seco / Frio**

- Luzes: Ligado
- Rega: Desligado
- Janelas: Ligado
- Portas: Desligado

- **Noite / Húmido / Quente**

- Luzes: Ligado
- Rega: Desligado
- Janelas: Ligado
- Portas: Desligado

- **Noite / Húmido / Frio**

- Luzes: Ligado
- Rega: Desligado
- Janelas: Desligado
- Portas: Desligado

Em relação ao sistema de luz ultravioleta, o seu funcionamento é diferente dos outros quatro (luz, rega, janelas e portas) porque esta função não depende da humidade, temperatura ou luminosidade. Este sistema é controlado por uma variável que se auto incrementa (simulando o tempo a passar) e é ativado por 15 segundos de 2 em 2 minutos.

```
#include "DHT.h" //Inclui a biblioteca do sensor DHT
```

```
#define DHTYPE DHT11 //Indicação da versão do sensor
```

```
//Atribui os pinos aos respetivos sistemas
#define DHTPIN 2 //Sensor DHT
#define ldr A0 //Sensor de luminosidade

#define luz 3 //Luzes
#define agua 4 //Rega
#define uv 5 //Luz uv
#define j 9 //Janelas
#define p 10 //Portas

DHT dht(DHTPIN, DHTTYPE); //Cria um objeto dht com o pino e versão do sensor

//Declaração das variáveis
int valorldr; //Luminosidade
float h; //Humidade
float t; //Temperatura
int temp = 0; //Tempo

void setup() {
    Serial.begin(9600); //Inicia o monitor serial
    dht.begin(); //Inicia o sensor dht
    //Declara as variáveis como INPUT's ou OUTPUT's
    pinMode(ldr, INPUT);
    pinMode(luz, OUTPUT);
    pinMode(agua, OUTPUT);
    pinMode(uv, OUTPUT);
    pinMode(j, OUTPUT);
    pinMode(p, OUTPUT);
}

void loop() {
```

Cofinanciado por:



```
valorldr = analogRead(ldr); //Lê o valor do sensor de luz
```

#### //Lê os valores do sensor DHT

```
float h = dht.readHumidity(); //Humidade  
float t = dht.readTemperature(); //Temperatura
```

#### //Verifica se a leitura do sensor DHT foi bem sucedida

```
if (isnan(h) || isnan(t)){  
    Serial.println("Falha de leitura do sensor DHT!");  
}
```

#### //Ativação/Desativação Dos Componentes

```
if (valorldr >= 500 && h < 60 && t >= 25){
```

#### //Dia, Seco, Quente

```
Serial.println("-----");  
Serial.println("");  
Serial.println("Dia | Seco | Quente");  
digitalWrite(agua, LOW);  
digitalWrite(luz, HIGH);  
digitalWrite(j, LOW);  
digitalWrite(p, LOW);
```

```
}else if (valorldr >= 500 && h < 60 && t < 25){
```

#### //Dia, Seco, Frio

```
Serial.println("-----");  
Serial.println("");  
Serial.println("Dia | Seco | Frio");  
digitalWrite(agua, LOW);  
digitalWrite(luz, HIGH);  
digitalWrite(j, HIGH);  
digitalWrite(p, HIGH);  
}else if (valorldr >= 500 && h >= 60 && t >= 25){
```

#### //Dia, Húmido, Quente

Cofinanciado por:



```
Serial.println("-----");
Serial.println("");
Serial.println("Dia | Húmido | Quente");
digitalWrite(agua, HIGH);
digitalWrite(luz, HIGH);
digitalWrite(j, LOW);
digitalWrite(p, LOW);
}else if (valordr >= 500 && h >= 60 && t < 25){
//Dia, Húmido, Frio
Serial.println("-----");
Serial.println("");
Serial.println("Dia | Húmido | Frio");
digitalWrite(agua, HIGH);
digitalWrite(luz, HIGH);
digitalWrite(j, HIGH);
digitalWrite(p, HIGH);
}else if (valordr < 500 && h < 60 && t >= 25){
//Noite, Seco, Quente
Serial.println("-----");
Serial.println("");
Serial.println("Noite | Seco | Quente");
digitalWrite(agua, HIGH);
digitalWrite(luz, LOW);
digitalWrite(j, LOW);
digitalWrite(p, HIGH);
}else if (valordr < 500 && h < 60 && t < 25){
//Noite, Seco, Frio
Serial.println("-----");
Serial.println("");
Serial.println("Noite | Seco | Frio");
digitalWrite(agua, HIGH);
digitalWrite(luz, LOW);
```

```
digitalWrite(j, HIGH);
digitalWrite(p, HIGH);
}else if (valorldr < 500 && h >= 60 && t >= 25){
```

#### //Noite, Húmido, Quente

```
Serial.println("-----");
Serial.println("");
Serial.println("Noite | Húmido | Quente");
digitalWrite(agua, HIGH);
digitalWrite(luz, LOW);
digitalWrite(j, LOW);
digitalWrite(p, HIGH);
}else if (valorldr < 500 && h >= 60 && t < 25){
```

#### //Noite, Húmido, Frio

```
Serial.println("-----");
Serial.println("");
Serial.println("Noite | Húmido | Frio");
digitalWrite(agua, HIGH);
digitalWrite(luz, LOW);
digitalWrite(j, HIGH);
digitalWrite(p, HIGH);
}
```

#### //Controlo do sistema de luzes ultravioleta

```
temp = temp + 1;
if (temp <= 15){
    digitalWrite(uv, LOW);
}else if (temp >= 15 && temp <= 135){
    digitalWrite(uv, HIGH);
}else if(temp > 135){
    temp = 0;
}
```

### //Apresenta os resultados para o monitor serial

```
Serial.print("Luminosidade: ");
Serial.print(valorldr);
Serial.println(" lm");
Serial.print("Humidade: ");
Serial.print(h);
Serial.println(" %");
Serial.print("Temperatura: ");
Serial.print(t);
Serial.println(" °C");
Serial.print("Tempo: ");
Serial.print(temp);
Serial.println(" minutos");

delay(1000);
}
```

## Aplicação

Para facilitar a gestão das funções do Arduino, desenvolvi uma aplicação utilizando o Visual Studio Community 2022 e a linguagem de programação C#. Esta aplicação estabelece uma ligação serial entre o Arduino e o computador para fazer a comunicação. O seu propósito é possibilitar o controlo dos sistemas da casa de forma manual através de botões ou fazer com que o Arduino faça esse controlo de forma automática, para além de monitorar os dados vindos dos sensores.

Para desenvolver a aplicação, no Visual Studio, comecei por criar um novo projeto, escolher o modelo que iria usar e o seu nome e localização. Como modelo, optei pelo “Aplicativo do Windows Forms (.NET Framework)”, ideal para uma aplicação simples compatível com o Windows.

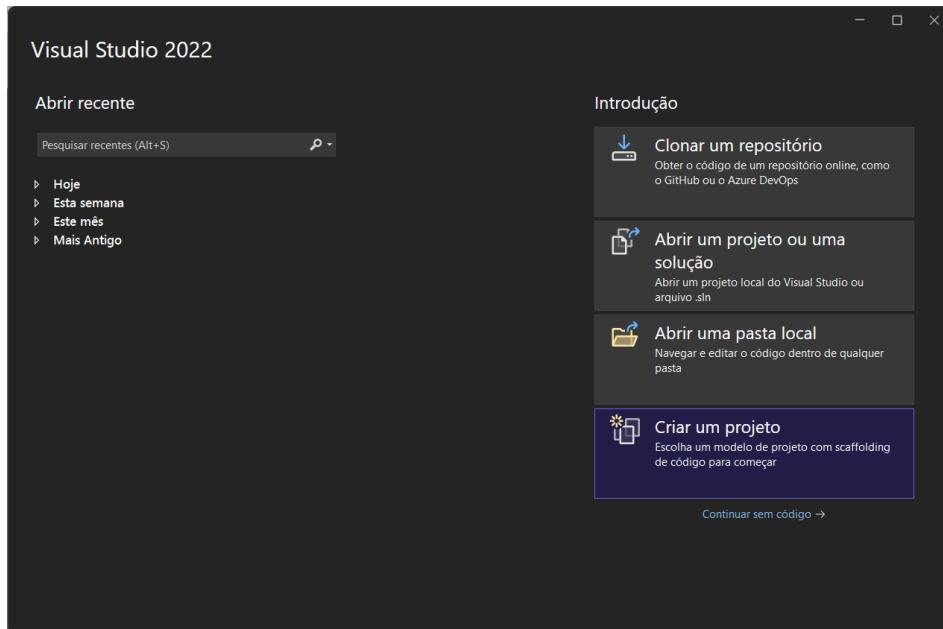
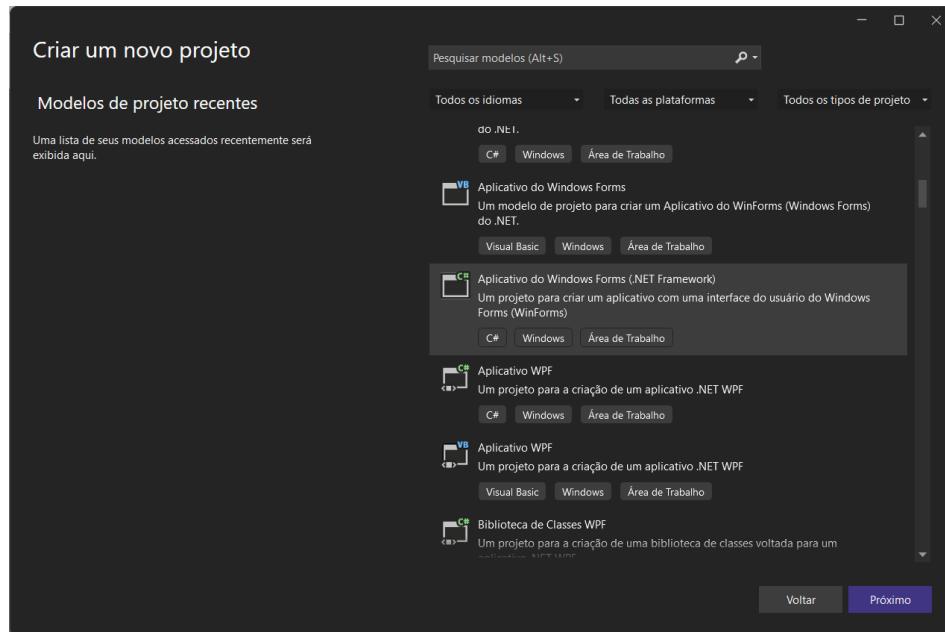
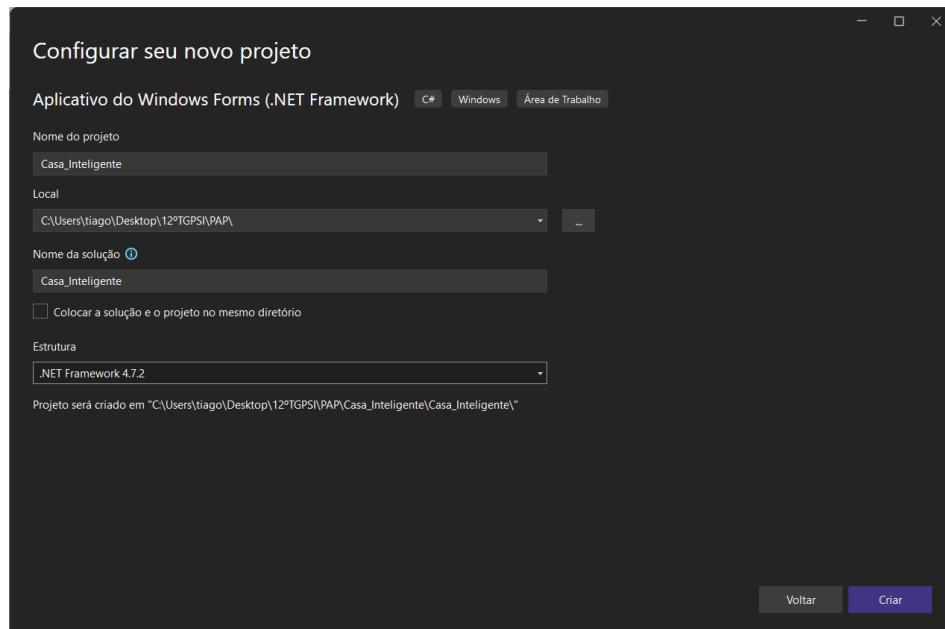


Figura 35: Janela Introdutória do Visual Studio



*Figura 36: Janela de Escolha do Modelo do Projeto*



*Figura 37: Janela de Introdução do Nome e Localização do Projeto*

## Visual Studio – Interface Gráfica

Com o projeto criado, comecei por colocar os elementos essenciais para o funcionamento da aplicação: um botão e uma *combobox* para selecionar a porta e estabelecer a conexão; uma *listbox* para apresentar os dados vindos dos sensores; uma *checkbox* e botões para controlar manualmente os sistemas do Arduino.

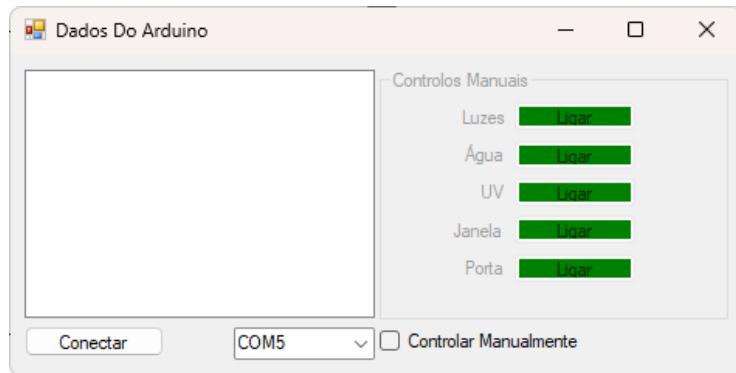


Figura 38: Interface da Aplicação – Versão 1

À medida que fui desenvolvendo a aplicação apercebi-me que utilizar apenas uma *checkbox* para controlar a ativação/desativação dos botões e da leitura dos dados estava a tornar-se complicado. Para além disso, por ser a primeira interface gráfica que desenvolvi para a aplicação, não planeei de forma detalhada a maneira como estariam organizados os elementos.

Com isso, decidi alterar o seu visual, colocando o botão e a *listbox* de conexão no topo visto que são os elementos mais importantes da aplicação. Depois, abaixo coloquei a *listbox* onde são apresentados os dados e dois botões: “Escrever” e “Limpar”. Além disso, decidi mover a *combobox* com os botões para a parte de baixo da aplicação, dando-lhe um aspeto mais simétrico. Por último, adicionei um botão para sair da aplicação já que removi a barra de controlo da janela.

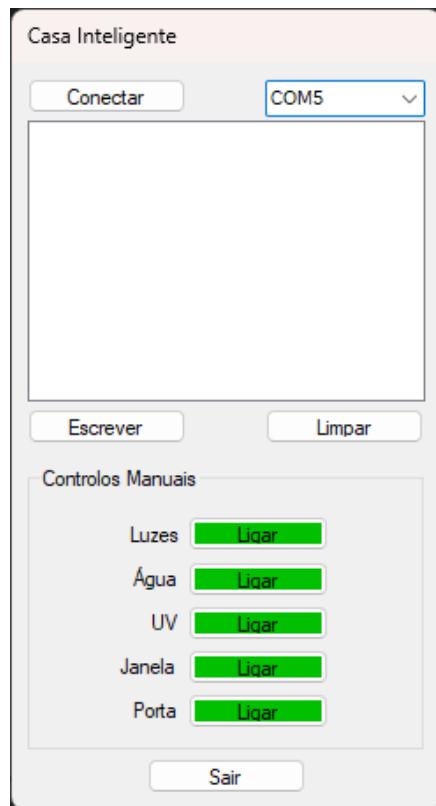


Figura 39: Interface da Aplicação - Versão 2

Posteriormente, ao explorar mais profundamente o Visual Studio, encontrei um elemento na lista de ferramentas que me poderia ser conveniente adicionar à aplicação: um gráfico.

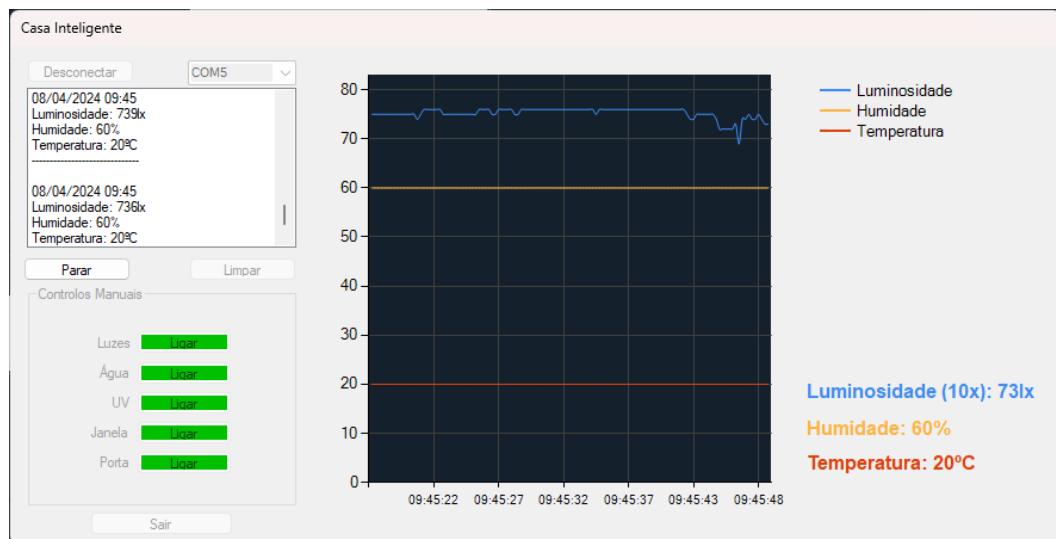


Figura 40: Interface da Aplicação - Versão 3

## **Visual Studio – Objetos e Variáveis**

Visto que estes objetos e variáveis não foram declarados dentro de nenhuma função em específico, estes podem ser usados em qualquer parte do código, sendo esta uma característica importante para que a aplicação funcione.

```
int temp_escrever;  
bool dados;  
  
private Thread receber;  
private object my_lock = new object();
```

## **Visual Studio - Formulário Principal**

```
//FORMULÁRIO PRINCIPAL  
public Form1()  
{  
    //Inicia os componentes do formulário e o temporizador  
    InitializeComponent();  
    timer_lista_portas.Enabled = true;  
}
```

## **Visual Studio - Temporizador de Atualização da ComboBox**

**//TIMER - ATUALIZAR LISTA**

```
private void timer_lista_portas_Tick(object sender, EventArgs e)
{
    //Verifica se o número de items na ComboBox é diferente do número de portas
    //disponíveis
    if (lista_portas.Items.Count != SerialPort.GetPortNames().Length)

    {
        //Limpa a ComboBox
        lista_portas.Items.Clear();

        //Adiciona cada porta disponível à ComboBox
        foreach (string porta in SerialPort.GetPortNames())
        {
            lista_portas.Items.Add(porta);
        }

        //Pre-Seleciona o item de index 0
        lista_portas.SelectedIndex = 0;
    }
}
```

## Visual Studio – Botão “Conectar”

**//BOTÃO - CONECTAR PORTA SERIAL**

```
private void botao_conectar_Click(object sender, EventArgs e)
{
    //Para o timer do botão "Conectar" para não criar nenhum conflito com o
    //estado da conexão
    timer_botao_escrever.Stop();

    //Verifica se a porta serial está fechada
    if (!serialPort1.IsOpen)
    {
        //Tenta executar o seguinte código
        try
        {
            //Define o nome da porta serial como o item selecionado na ComboBox
            serialPort1.PortName =
                lista_portas.Items[lista_portas.SelectedIndex].ToString();

            //Abre a porta serial
            serialPort1.Open();
        }
        catch (Exception ex) //Caso não consiga
        {
            //Exibe uma mensagem de erro
            MessageBox.Show("Não foi possível conectar ao Arduino: " + ex.Message,
                "Erro de Conexão", MessageBoxButtons.OK, MessageBoxIcon.Error);

            //Encerra a execução desta função nesta linha
            return;
        }

        //Se a porta serial foi aberta
        if (serialPort1.IsOpen)
        {
    }
```

Cofinanciado por:



```
//Altera o texto do botão para "Desconectar"
botao_conectar.Text = "Desconectar";

//Desativa a ComboBox para impedir a mudança de porta durante a
ligaçāo
lista_portas.Enabled = false;

//Desativa o botão "Sair" para impedir que a aplicação seja fechada
durante a ligação
botao_sair.Enabled = false;

//Ativa os botões "Escrever" e "Limpar", responsáveis por escrever ou
limpar a ListBox dos dados
botao_escrever.Enabled = true;
botao_limpar.Enabled = true;

//Garante acesso exclusivo ao objeto "my_lock" para que não seja
interferido por outra Thread
lock (my_lock)

//Envia um "M" para o Arduino, ativando como predefinição o modo
manual
serialPort1.Write("M");

//Atribui o valor "true" às funções "atualizar_botoes" e "controlos",
ativando os botões de controlo do Arduino e atribui-lhes a cor verde ou
vermelho consoante o estado do LED correspondente
controlos(true);
atualizar_botoes(true);

}

}

else //Se a porta serial estiver fechada
{
    //Tenta executar o seguinte código
    try
    {
```

```
//Garante acesso exclusivo ao objeto "my_lock" para que não seja
interferido por outra Thread

lock (my_lock)

//Envia um "A" para o Arduino para que, após o fechamento da
aplicação, este possa funcionar de forma independente

serialPort1.Write("A");

//Fecha a porta serial

serialPort1.Close();

}

catch (Exception ex) //Caso não consiga

{

    //Exibe uma mensagem de erro

    MessageBox.Show("Não foi possível desconectar do Arduino" + ex.Message,
    "Erro de Conexão", MessageBoxButtons.OK, MessageBoxIcon.Error );

    //Encerra a execução desta função nesta linha

    return;

}

//Se a porta serial estiver fechada

if (!serialPort1.IsOpen)

{

    //Altera o texto dos botões para "Conectar" e "Escrever"

    botao_conectar.Text = "Conectar";

    botao_escrever.Text = "Escrever";



    //Ativa a ComboBox de seleção da porta serial

    lista_portas.Enabled = true;

    //Ativa o botão "Sair"

    botao_sair.Enabled = true;




    //Desativa o botão "Escrever" para que não ser possível iniciar uma
    leitura com a porta serial fechada

    botao_escrever.Enabled = false;
```

**//Desativa o botão "Limpar" apenas por aspetto gráfico, ajudando a passar uma imagem de aplicação desativada**

```
botao_limpar.Enabled = false;
```

**//Atribui o valor "false" às funções "atualizar\_botoes" e "controlos", desativando os botões de controlo do Arduino e retornando-os para a sua cor pre-definida para, assim como o botão "Limpar", simular uma aplicação desativada**

```
atualizar_botoes(false);
controlos(false);
}
}
}
```

### **Visual Studio – Botão “Escrever”**

#### **//BOTÃO - ESCREVER DADOS**

```
private void botao_escrever_Click(object sender, EventArgs e)
```

```
{
```

**//Desativa o próprio botão para que não possa ser pressionado repetidamente, desativa o botão "Conectar" para que a ligação não seja interrompida durante a leitura dos dados e inicia um timer que, posteriormente, reativa o botão**

```
botao_conectar.Enabled = false;
botao_escrever.Enabled = false;
temp_escrever = 0;
timer_botao_escrever.Start();
```

**//Verifica se o texto do botão é "Escrever", ou seja, se não está a ser feita a leitura dos dados**

```
if (botao_escrever.Text == "Escrever")
{
```

**//Altera o texto do botão para "Parar" e torna a variável de controlo da função "receber\_dados" verdadeira**

Cofinanciado por:



SELO DE CONFORMIDADE EQAVET  
GARANTIA DA QUALIDADE  
NA EDUCAÇÃO E FORMAÇÃO PROFISSIONAL

```

botao_escrever.Text = "Parar";
dados = true;

//Desativa o botão "Conectar" e o botão "Limpar"
botao_conectar.Enabled = false;
botao_limpar.Enabled = false;

//Garante acesso exclusivo ao objeto "my_lock" para que não seja
interferido por outra Thread

lock (my_lock)

//Envia um "A" para o Arduino, ativando o modo automático
serialPort1.Write("A");

//Limpa todos os dados anteriormente recebidos do Arduino para não
criarem nenhuma confusão com os dados novos

serialPort1.DiscardInBuffer();

//Atribui o valor "false" às funções "atualizar_botoes" e "controlos",
desativando os botões de controlo do Arduino e retornando-os para a sua cor
pre-definida

atualizar_botoes(false);
controlos(false);

//Verifica se a Thread "receber" é nula ou se está inativa

if (receber == null || !receber.IsAlive)
{
    //Associa a Thread "receber" à função "receber_dados" e inicia-a
    receber = new Thread(receber_dados);
    receber.Start();
}

else if (botao_escrever.Text == "Parar") //Se o texto do botão "Parar", ou seja, se
está a ser feita a leitura dos dados

{
}

```

**//Altera o texto do botão para "Escrever" e torna a variável de controlo da função "receber\_dados" falsa**

```
botao_escrever.Text = "Escrever";  
dados = false;
```

**//Ativa o botão "Conectar" e o botão "Limpar"**

```
botao_conectar.Enabled = true;  
botao_limpar.Enabled = true;
```

**//Verifica se a Thread "receber" não é nula e se está ativa**

```
if (receber != null && receber.IsAlive)  
{  
    //Termina a Thread "receber"  
    receber.Join();  
}
```

**//Limpa todos os dados anteriormente recebidos do Arduino para não criarem nenhuma confusão com os dados novos**

```
serialPort1.DiscardInBuffer();
```

**//Garante acesso exclusivo ao objeto "my\_lock" para que não seja interferido por outra Thread**

```
lock (my_lock)
```

**//Envia um "M" para o Arduino, ativando o modo manual**

```
serialPort1.Write("M");
```

**//Atribui o valor "true" às funções "atualizar\_botoes" e "controlos", ativando os botões de controlo do Arduino e atribui-lhes a cor verde ou vermelho consoante o estado do LED correspondente**

```
controlos(true);  
atualizar_botoes(true);  
}  
}
```

## Visual Studio – Timer de Inatividade do Botão “Escrever”

Este timer possui um intervalo de 0.5 segundos entre cada iteração, ou seja, assim que o botão “Escrever” seja desativado e seja iniciado o timer, este só voltará a ficar ativo 0.5 segundos após o clique.

### //TIMER - ESCREVER DADOS

```
private void timer_botao_escrever_Tick(object sender, EventArgs e)
{
    //Incrementa a variável "temp_escrever"
    temp_escrever = temp_escrever + 1;

    //Quando a variável chegar ao valor 1
    if (temp_escrever == 1)
    {
        //Ativa novamente o botão "Escreve"
        botao_escrever.Enabled = true;
        //Para o próprio timer
        timer_botao_escrever.Stop();
    }
}
```

### **Visual Studio – Botão “Limpar”**

#### **//BOTÃO - LIMPAR**

```
private void botao_limpar_Click(object sender, EventArgs e)
{
    //Limpa todos os itens da ListBox
    lista_dados.Items.Clear();

    //Limpa os valores das labels
    label6.Text = "Luminosidade (10x):";
    label7.Text = "Humidade:";
    label8.Text = "Temperatura:";
}
```

### **Visual Studio - Recepção dos Dados**

Como esta função precisa de estar em execução continuamente é necessário colocar o modificador “async” na sua declaração, indicando que esta será realizada em segundo plano. Isso permite que esta função opere de forma assíncrona, possibilitando que outras partes do programa continuem a ser executadas.

#### **//RECEBER DADOS**

```
private async void receber_dados()
{
    //Tenta executar o seguinte código
    try
    {
        //Declaração de uma variável do tipo string que, diferente das string's regulares, pode ser manipulada e modificada de forma simples e fácil
        StringBuilder buffer = new StringBuilder();
```

Cofinanciado por:



```

//Executa enquanto a variável de controlo for verdadeira
while (dados)
{
    string texto;

    //Garante acesso exclusivo ao objeto "my_lock" para que não seja
interferido por outra Thread

    lock (my_lock)
    {
        //Envia um "6" para o Arduino, retornando os dados dos sensores
        serialPort1.WriteLine("6");

        //Lê os dados vindos do Arduino e guarda numa variável
        texto = serialPort1.ReadExisting().Trim();

    }

    //Divide em partes a variável texto e guarda num vetor do tipo string
    string[] partes = texto.Split(new char[] { '\n' },
StringSplitOptions.RemoveEmptyEntries);

    //Verifica se o tamanho do vetor é o correto
if (partes.Length >= 3)
{
    Invoke((MethodInvoker)delegate
    {

        //Converte os valores guardados no vetor para inteiros e guarda em
variáveis

        int luminosidade = Convert.ToInt32(partes[0]);
        int humidade = Convert.ToInt32(partes[1]);
        int temperatura = Convert.ToInt32(partes[2]);

        //Adiciona os valores dos sensores às labels do gráfico
label6.Text = "Luminosidade (10x): " + luminosidade / 10 + "lx";
label7.Text = "Humidade: " + humidade + "%";
    });
}
}

```

```

label8.Text = "Temperatura: " + temperatura + "ºC";

//Adiciona os valores dos sensores e a hora atual à ListBox

lista_dados.Items.Add("-----");
lista_dados.Items.Add("\n");
lista_dados.Items.Add(DateTime.Now);
lista_dados.Items.Add("Luminosidade: " + luminosidade + "lx");
lista_dados.Items.Add("Humidade: " + humidade + "%");
lista_dados.Items.Add("Temperatura: " + temperatura + "ºC");
lista_dados.TopIndex = lista_dados.Items.Count - 1;

//Adiciona um ponto novo ao gráfico para cada variável

chart1.Invoke((MethodInvoker)() =>
chart1.Series["Luminosidade"].Points.AddXY(DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"), 
luminosidade / 10));

chart1.Invoke((MethodInvoker)() =>
chart1.Series["Humidade"].Points.AddXY(DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
humidade));

chart1.Invoke((MethodInvoker)() =>
chart1.Series["Temperatura"].Points.AddXY(DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"),
temperatura));

//Define o limite máximo do gráfico como o valor mais alto das três variáveis + 10

chart1.ChartAreas[0].AxisY.Maximum = Math.Max(luminosidade / 10,
Math.Max(humidade, temperatura)) + 10;

//Verifica se o número de pontos no gráfico é maior que 250

if (chart1.Series["Luminosidade"].Points.Count > 250 ||
chart1.Series["Humidade"].Points.Count > 250 ||
chart1.Series["Temperatura"].Points.Count > 250)
{
    //Remove o ponto na posição 0 do gráfico para que não sobrecarregar a aplicação

    chart1.Series["Luminosidade"].Points.RemoveAt(0);
    chart1.Series["Humidade"].Points.RemoveAt(0);
}

```

```
chart1.Series["Temperatura"].Points.RemoveAt(0);
}

});

}

//Intervalo de 1/4 de segundo entre iterações
await Task.Delay(250);

}

}

catch (ThreadAbortException) //Torna a variável "dados" falsa quando a Thread é abolida
{
    dados = false;
}

catch (Exception ex) //No caso de a exceção não corresponder à verificação anterior
{
    //Exibe uma mensagem de erro
    MessageBox.Show("Não foi possível ler os dados do Arduíno" + ex.Message,
    "Erro de Leitura", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

## Visual Studio – Comunicação com o Arduino

Esta função foi criada com o objetivo de juntar todo o código que envolve enviar e receber dados do Arduino (exceto o código responsável por pedir e receber os dados dos sensores) num local só. Desta forma, elimino a possibilidade de existir mais que uma função a pedir ou receber dados do Arduino em simultâneo.

(Nota: Como o código é idêntico para todos os sistemas, coloquei aqui apenas o código de um deles como exemplo.)

### //COMUNICAÇÃO COM O ARDUINO

```
private void geral(string mensagem)
{
    //Variável que guardará os dados do Arduino
    string resposta;
    //Limpa todos os dados anteriormente recebidos do Arduino para não criarem
    //nenhuma confusão com os dados novos
    serialPort1.DiscardInBuffer();

    //Garante acesso exclusivo ao objeto "my_lock" para que não seja interferido
    //por outra Thread
    lock (my_lock)
        //Envia o valor da variável mensagem para o Arduino
        serialPort1.Write(mensagem);

    switch (mensagem)
    {
        //Se o valor da variável "mensagem" for "1"
        case "1":
            //Guarda os dados vindos do Arduino
            resposta = serialPort1.ReadLine().Trim();

        //Verifica se o valor guardado é "Luz Ligada"
```

```

if (resposta == "Luz Ligada")
{
    //Altera o texto do botão para "Desligar"
    botao_luzes.Text = "Desligar";
    //Altera a cor do botão para vermelho
    botao_luzes.BackColor = Color.Red;
}

else if (resposta == "Luz Desligada") //Verifica se o valor guardado é "Luz Desligada"
{
    //Altera o texto do botão para "Ligar"
    botao_luzes.Text = "Ligar";
    //Altera a cor do botão para verde
    botao_luzes.BackColor = Color.FromArgb(0, 192, 0);
}

//Limpa todos os dados anteriormente recebidos do Arduino para não criarem nenhuma confusão com os dados novos
serialPort1.DiscardInBuffer();
break;

//BOTÃO - REGA
//BOTÃO - LUZ UV
//BOTÃO - JANELAS
//BOTÃO – PORTAS

case "7":
    //Cria um vetor do tipo string
    string[] partes = new string[5];

    for (int i = 0; i < 5; i++)
{

```

```

//Guarda no vetor todas as linhas vindas do Arduino, ou seja, 5
partes[i] = serialPort1.ReadLine().Trim();
}

//Verifica se o valor do vetor na posição 0 é "Luz Ligada"
if (partes[0] == "Luz Ligada")
{
    //Altera o texto do botão para "Desligar"
botao_luzes.Text = "Desligar";
    //Altera a cor do botão para vermelho
botao_luzes.BackColor = Color.Red;
}
else if (partes[0] == "Luz Desligada") //Verifica se o valor do vetor na
posição 0 é "Luz Desligada"
{
    //Altera o texto do botão para "Ligar"
botao_luzes.Text = "Ligar";
    //Altera a cor do botão para verde
botao_luzes.BackColor = Color.FromArgb(0, 192, 0);
}

//BOTÃO - REGA
//BOTÃO - LUZ UV
//BOTÃO - JANELAS
//BOTÃO – PORTAS

//Limpa todos os dados anteriormente recebidos do Arduino para não
criarem nenhuma confusão com os dados
serialPort1.DiscardInBuffer();
break;
}
}

```

## **Visual Studio - Botões de Controlo**

O código dos botões que controlam manualmente as funções da casa é praticamente o mesmo, variando apenas no valor que enviam ao Arduino e a mensagem que recebem.

```
//BOTÃO - LUZ
```

```
//Envia o valor "1" para a função "geral"
```

```
private void botao_luzes_Click(object sender, EventArgs e) => geral("1");
```

## **Visual Studio - Habilitar/Desabilitar Botões**

```
//CONTROLOS
```

```
private void controlos(bool habilitar)
```

```
{
```

```
//Ativa ou desativa os objetos consoante o valor da variável "habilitar"
```

```
controlos_manuais.Enabled = habilitar;
```

```
label1.Enabled = habilitar;
```

```
label2.Enabled = habilitar;
```

```
label3.Enabled = habilitar;
```

```
label4.Enabled = habilitar;
```

```
label5.Enabled = habilitar;
```

```
botao_luzes.Enabled = habilitar;
```

```
botao_agua.Enabled = habilitar;
```

```
botao_uv.Enabled = habilitar;
```

```
botao_janela.Enabled = habilitar;
```

```
botao_porta.Enabled = habilitar;
```

```
}
```

## Visual Studio – Atualizar Botões

Sempre que este código é ativado, é dado o valor verdadeiro ou falso à variável “n”, responsável por conferir o texto e as cores certas aos botões de acordo com o estado do LED's.

(Nota: Como o código é idêntico para todos os sistemas, coloquei aqui apenas o código de um deles como exemplo.)

### //ATUALIZAR BOTÕES

```
private void atualizar_botoes(bool n)
{
    //Se a variável "n" for verdadeira
    if (n)
    {
        //Envia o valor "7" para a função "geral"
        geral("7");
    }
    else if (!n) //Se a variável "n" for falsa
    {
        //Altera o texto do botão para "Ligar"
        botao_luzes.Text = "Ligar";
        //Altera a cor do botão para verde
        botao_luzes.BackColor = Color.FromArgb(0, 192, 0);

        //BOTÃO - REGA
        //BOTÃO - LUZ UV
        //BOTÃO - JANELAS
        //BOTÃO – PORTAS
    }
}
```

## Visual Studio – Botão “Sair”

### //BOTÃO - SAIR

```
private void botao_sair_Click(object sender, EventArgs e)
{
    //Mostra uma caixa de mensagem de confirmação e guarda a opção
    //selecionada numa variável

    var result = MessageBox.Show("Sair da Aplicação?", "Casa Inteligente",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    //Se a opção selecionada foi "Sim"
    if (result == DialogResult.Yes)
    {
        //Fecha o formulário
        this.Close();
    }
}
```

## **Visual Studio – Formulário Principal Fechado**

Como esta aplicação estabelece uma conexão serial e possui funções assíncronas a executar em segundo plano é importante que, quando fechada, estes processos sejam terminados corretamente.

### **//AO FECHAR**

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)  
{  
    //Torna a variável de controlo da função "receber_dados" falsa  
    dados = false;  
  
    //Termina a Thread "receber"  
    receber.Join();  
  
    //Fecha a porta serial  
    serialPort1.Close();  
}
```

## **ArduinoIDE – Bibliotecas e Variáveis**

Para que a aplicação funcione, o Arduino precisa de receber e enviar dados para a mesma. Sendo assim, para que isto seja possível, fiz algumas alterações ao código apresentado na fase de testes dos componentes.

```
#include "DHT.h" //Inclui a libraria do sensor DHT

#define DHTYPE DHT11 //Indicação da versão do sensor
//Atribui os pinos aos respetivos sistemas
#define DHTPIN 2 //Sensor DHT
#define pinldr A0 //Sensor de luminosidade

#define sluz 3 //Luzes
#define sagua 4 //Rega
#define suv 5 //Luz UV
#define sjanela 9 //Janelas
#define sporta 10 //Portas

DHT dht(DHTPIN, DHTYPE); //Cria um objeto dht com o pino e tipo de sensor

//Declaração das variáveis
int ldr; //Luminosidade
int h; //Humidade
int t; //Temperatura
int tempo; //Tempo
int ant = 0; //Variável que guardará o tempo passado em determinado momento
int dif = 0; //Variável que será auto incrementada no timer
bool manual = true; //Variável de controlo do modo manual
```

## **ArduinoIDE – Iniciação do Programa**

Nesta parte do programa é colocado o código que precisa de iniciar assim que é feita a sua execução, como a ativação do monitor serial e sensores ou definição dos pinos como entrada ou saída de dados.

### **//SETUP**

```
void setup() {
```

#### **//Declara os LED's como OUTPUT's**

```
pinMode(3, OUTPUT);
```

```
pinMode(4, OUTPUT);
```

```
pinMode(5, OUTPUT);
```

```
pinMode(9, OUTPUT);
```

```
pinMode(10, OUTPUT);
```

#### **//Inicia o programa com os LED's desligados**

```
digitalWrite(3, LOW);
```

```
digitalWrite(4, LOW);
```

```
digitalWrite(5, LOW);
```

```
digitalWrite(9, LOW);
```

```
digitalWrite(10, LOW);
```

```
Serial.begin(9600); //Inicia o monitor serial
```

```
dht.begin(); //Inicia o sensor DHT
```

```
pinMode(pinldr, INPUT); //Declara o sensor de luminosidade como INPUT
```

```
}
```

## ***ArduinoIDE – Loop***

Como esta parte do programa está constantemente em execução, pode ser considerado como o local onde é colocado o código principal. É nesta parte do código que é feita a leitura e apresentação dos valores dos sensores e o controlo das funcções dos LED's.

*(Nota: Como o código é idêntico para todos os sistemas, coloquei aqui apenas o código de um deles como exemplo.)*

### **//LOOP**

```
void loop() {  
  
    //Verifica se a porta serial está disponível  
    if (Serial.available()){  
  
        //Guarda os valores recebidos da porta serial numa variável  
        char op = Serial.read();  
  
        switch(op){  
            //  
            //Se o valor recebido for "1"  
            case '1':  
                //Executa a função "luz"  
                luz();  
                break;  
  
                //FUNÇÃO - REGA  
                //FUNÇÃO - LUZ UV  
                //FUNÇÃO - JANELAS  
                //FUNÇÃO – PORTAS
```

**//Se o valor recebido for "7"**

```
case '7':  
    //Executa a função "estado_luzes"  
    estado_luzes();  
    break;
```

**//Se o valor recebido for "A"**

```
case 'A':  
    //Atribui à variável "manual" o valor falso, ativando o modo automático  
    manual = false;  
    break;  
}
```

**//Executa enquanto a variável manual for falsa**

```
while (manual == false){  
    //Lê o valor do sensor de luz  
    ldr = analogRead(pinldr);
```

**//Lê os valores do sensor DHT**

```
h = dht.readHumidity();  
t = dht.readTemperature();
```

**//Verifica se as leituras do sensor DHT foram bem sucedidas**

```
if(isnan(h) || isnan(t)){  
    Serial.println("Falha de leitura do sensor DHT!");  
}
```

**//ATIVAÇÃO/DESATIVAÇÃO DOS COMPONENTES**

**//Cada If verifica os valores dos sensores para determinar quais sistemas ativar/desativar**

```
if (ldr >= 500 && h < 60 && t >= 25){
```

**//DIA, SECO, QUENTE**

```
digitalWrite(sagua, HIGH);
digitalWrite(sluz, LOW);
digitalWrite(sjanelas, HIGH);
digitalWrite(sporta, HIGH);

}else if (ldr >= 500 && h < 60 && t < 25){

    //DIA, SECO, FRIO
    digitalWrite(sagua, HIGH);
    digitalWrite(sluz, LOW);
    digitalWrite(sjanelas, LOW);
    digitalWrite(sporta, LOW);

}else if (ldr >= 500 && h >= 60 && t >= 25){

    //DIA, HÚMIDO, QUENTE
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, LOW);
    digitalWrite(sjanelas, HIGH);
    digitalWrite(sporta, HIGH);

}else if (ldr >= 500 && h >= 60 && t < 25){

    //DIA, HÚMIDO, FRIO
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, LOW);
    digitalWrite(sjanelas, LOW);
    digitalWrite(sporta, LOW);

}else if (ldr < 500 && h < 60 && t >= 25){

    //NOITE, SECO, QUENTE
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, HIGH);
    digitalWrite(sjanelas, HIGH);
    digitalWrite(sporta, LOW);

}else if (ldr < 500 && h < 60 && t < 25){

    //NOITE, SECO, FRIO
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, HIGH);
```

```

digitalWrite(sjanela, LOW);
digitalWrite(sporta, LOW);
}else if (ldr < 500 && h >= 60 && t >= 25){
    //NOTE, HÚMIDO, QUENTE
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, HIGH);
    digitalWrite(sjanela, HIGH);
    digitalWrite(sporta, LOW);
}else if (ldr < 500 && h >= 60 && t < 25){
    //NOITE, HÚMIDO, FRIO
    digitalWrite(sagua, LOW);
    digitalWrite(sluz, HIGH);
    digitalWrite(sjanela, LOW);
    digitalWrite(sporta, LOW);
}

```

**//Guarda, em segundos, o tempo passado desde o início da execução do programa**

```

tempo = millis() / 1000;
//Executa a função que controla o sistema de luzes UV
acende_uv();

```

**//Guarda os valores recebidos da porta serial numa variável**

```
char op = Serial.read();
```

```
switch (op)
```

```
{
```

**//Se o valor recebido for "6"**

```
case '6':
```

**//Executa a função "sensores"**

```
sensores();
```

```
break;
```

```
//Se o valor recebido for "M"
case 'M':
    //Atribui à variável "manual" o valor verdadeiro, ativando o modo manual
    manual = true;
    break;
}
}
}
}
```

### **ArduinoIDE – Verificação do Estado dos LED's**

(Nota: Como o código é idêntico para todos os sistemas, coloquei aqui apenas o código de um deles como exemplo.)

```
//Função que verifica o estado das luzes
void estado_luzes(){

    //Verifica se o LED está ligado
    if (digitalRead(3) == HIGH){

        //Escreve para a porta serial "Luz Ligada"
        Serial.println("Luz Ligada");

    }else{ //Se o LED está desligado

        //Escreve para a porta serial "Luz Desligada"
        Serial.println("Luz Desligada");
    }

    //FUNÇÃO - REGA
    //FUNÇÃO - LUZ UV
    //FUNÇÃO - JANELAS
    //FUNÇÃO – PORTAS
}
```

Cofinanciado por:



## **ArduinoIDE – Controlo do Sistema de Luzes UV**

**//Função que controla o sistema de luzes UV**

```
void acende_uv(){  
    //Verifica se o valor da variável "ant" não corresponde ao valor da variável  
    "tempo"  
    if (tempo != ant){  
        //Atribui o valor da variável "tempo" à variável "ant"  
        ant = tempo;  
        //Incrementa o valor 1 à variável dif  
        dif = dif + 1;  
        if (dif < 15){ //Liga o LED nos primeiros 15 segundos  
            digitalWrite(suv, HIGH);  
        }else if (dif >= 15 && dif < 135){ //Desliga o LED nos seguintes 120 segundos  
            digitalWrite(suv, LOW);  
        }else if (dif >= 135){ //Reinicia o valor da variável no fim do ciclo  
            dif = 0;  
        }  
    }  
}
```

## ***ArduinoIDE – Apresentação dos Valores dos Sensores***

```
void sensores(){

    //Escreve os valores dos sensores para o monitor serial
    //Valor do sensor de luz
    Serial.print(ldr);
    //Quebra de linha
    Serial.println();
    //Valor do sensor de humidade
    Serial.print(h);
    //Quebra de linha
    Serial.println();
    //Valor do sensor de temperatura
    Serial.print(t);
    //Quebra de linha
    Serial.println();
}
```

## **ArduinoIDE – Função dos Sistemas da Casa**

Assim como no código da aplicação no Visual Studio, o código das funções da casa, com exceção do sistema de luzes UV, é praticamente igual. As únicas diferenças são o nome que têm e o pino que controlam.

*(Nota: Como o código é idêntico para todos os sistemas, coloquei aqui apenas o código de um deles como exemplo.)*

### **//Função do sistema de luz**

```
void luz(){  
    //Verifica se o LED está desligado  
    if (digitalRead(3) == LOW){  
        //Liga o LED  
        digitalWrite(3, HIGH);  
        //Escreve para porta serial "Luz Ligada"  
        Serial.println("Luz Ligada");  
    }else{ //Se o LED está ligado  
        //Desliga o LED  
        digitalWrite(3, LOW);  
        //Escreve para a porta serial "Luz Desligada"  
        Serial.println("Luz Desligada");  
    }  
}
```

## Placa de Circuitos Impressos (PCI)

Como forma de prevenir que os componentes se soltem do circuito e como recomendação, decidi fazer uma placa de circuitos impressos. Esta PCI, para além de evitar que os componentes se soltem, também permite que eu possa reutilizar a placa de ensaio para outros projetos, visto que esta já não é preciso para manter o circuito conectado.

Para criar esta placa utilizei uma CNC (Computer Numeric Control) que me foi disponibilizada no estágio. Como esta máquina utiliza instruções em G-Code, o processo para criar a PCI foi mais complexo do que pensava. Em primeiro lugar, assim como para a criação de qualquer PCI, tive que criar o seu esquema elétrico. Depois, tive que exportar esse esquema como um ficheiro Gerber. Para converter esse ficheiro para G-Code, precisei de o passar por um terceiro software chamado FlatCAM. Por fim, carreguei os ficheiros gerados para a CNC.

### EasyEDA

Para começar a criar a minha PCI, primeiro precisei de criar o seu esquema elétrico. Para isso, utilizei uma ferramenta chamada EasyEDA. Esta plataforma pode ser utilizada de forma online, através dum browser ou localmente, baixando um executável. Eu optei por usar a versão online.



Figura 41: EasyEDA – Página Principal

Ao selecionar a opção “Design Online”, é dada a opção de escolher que versão da plataforma se pretende usar. Como o meu objetivo é fazer uma placa simples, escolhi a versão “Std Edition”.



Figura 42: EasyEDA – Janela de escolha da Versão

Após isso, será aberto o editor do EasyEDA. Para começar a criar um projeto, primeiro, é necessário criar uma conta ou iniciar sessão. Já que esta ferramenta o permite, eu decidi iniciar sessão com a conta Google.

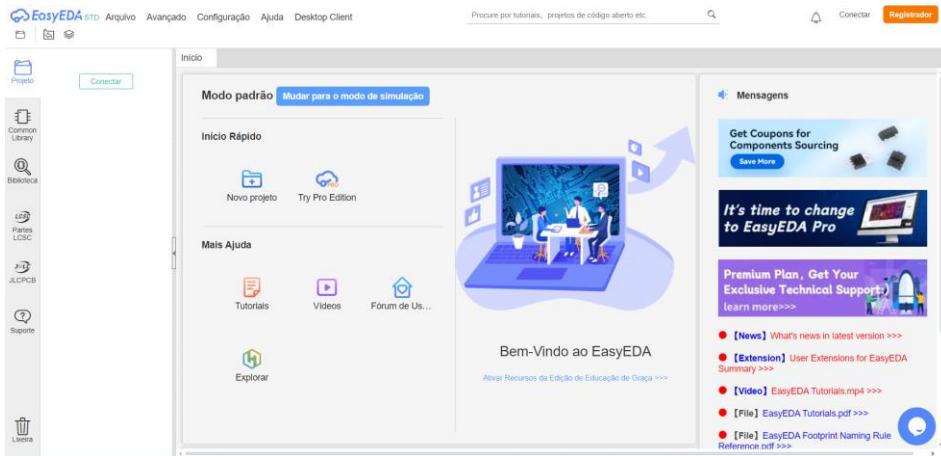


Figura 43: EasyEDA – Editor

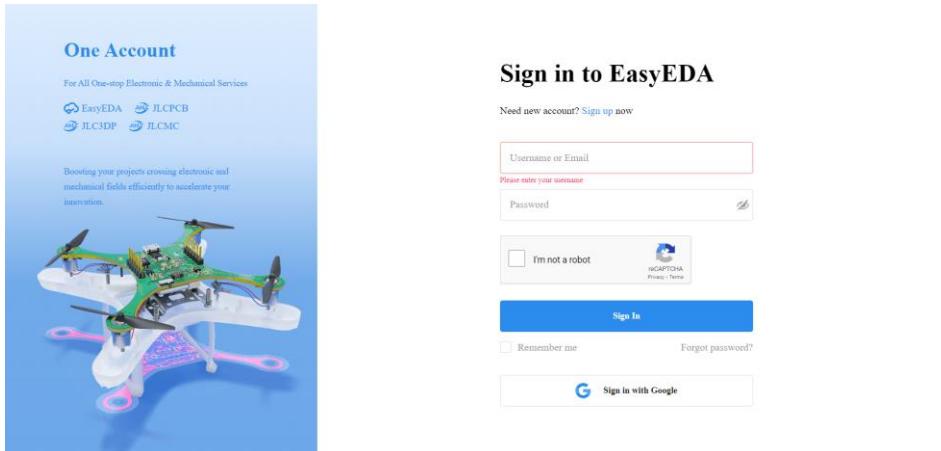


Figura 44: EasyEDA - Página de Início de Sessão

Como esta é a minha primeira vez a utilizar o EasyEDA, após ter iniciado sessão, foram apresentadas algumas janelas para definir alguns aspetos.

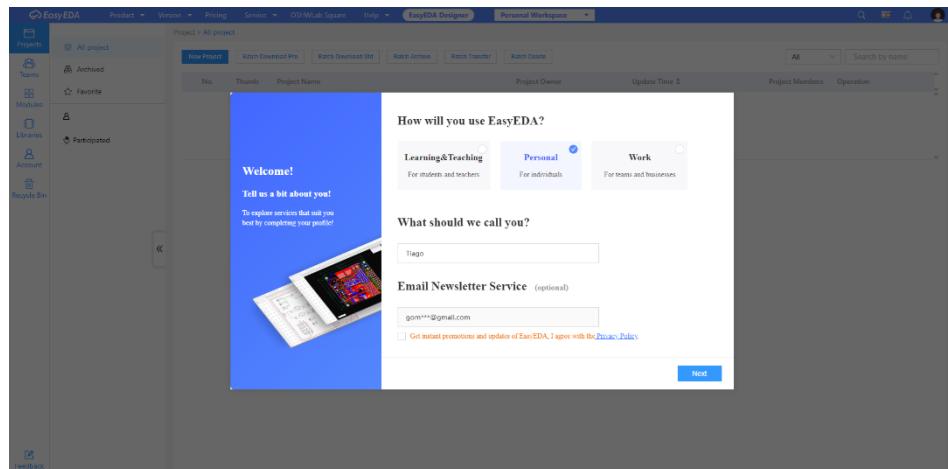


Figura 45: EasyEDA - Escolha do Tipo de Uso e Nome

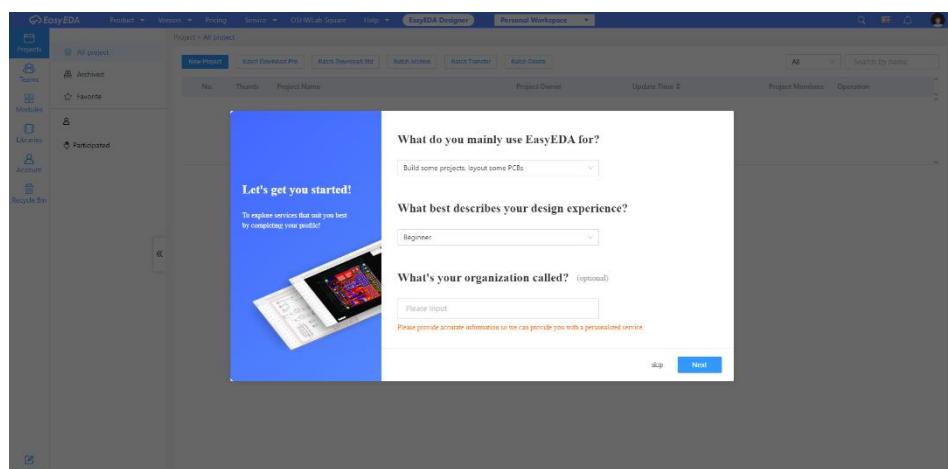
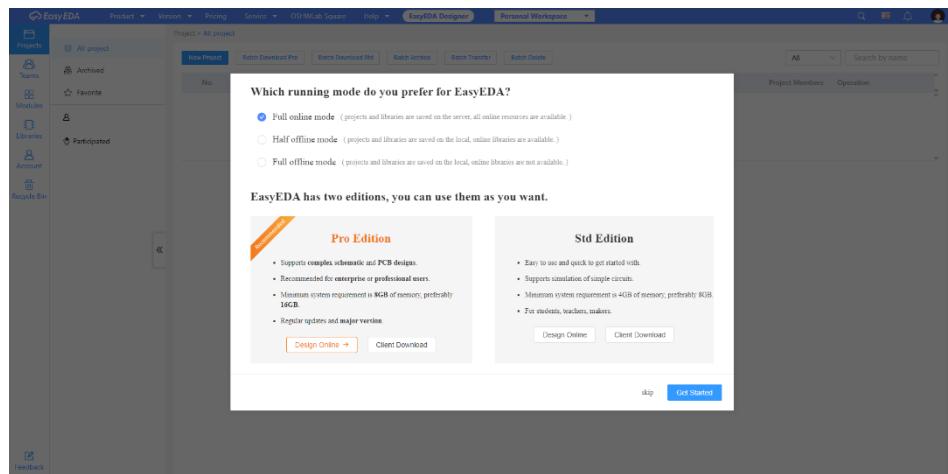
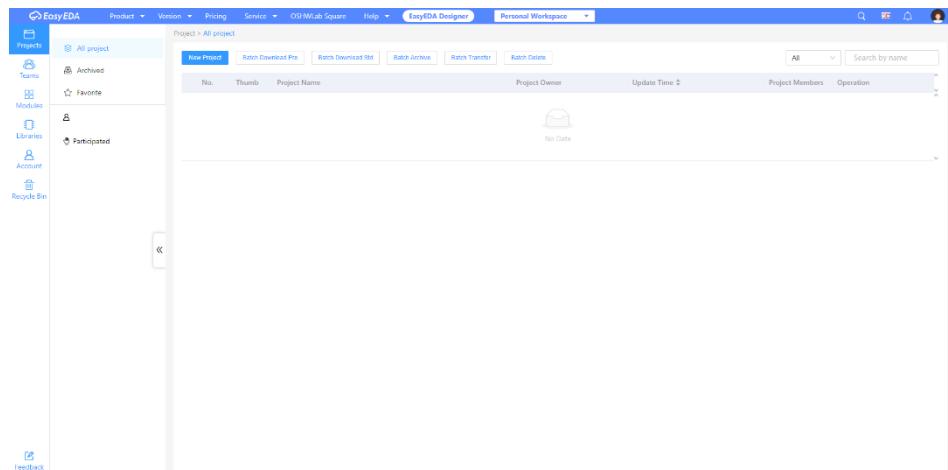


Figura 46: EasyEDA - Escolha dos Objetivos e do Nível de Experiência



*Figura 47: EasyEDA - Escolha do Modo de Execução*

Com estes passos todos realizados basta apenas clicar no botão “New Project” e preencher os campos com as informações do projeto.



*Figura 48: EasyEDA - Página de Visualização dos Projetos*

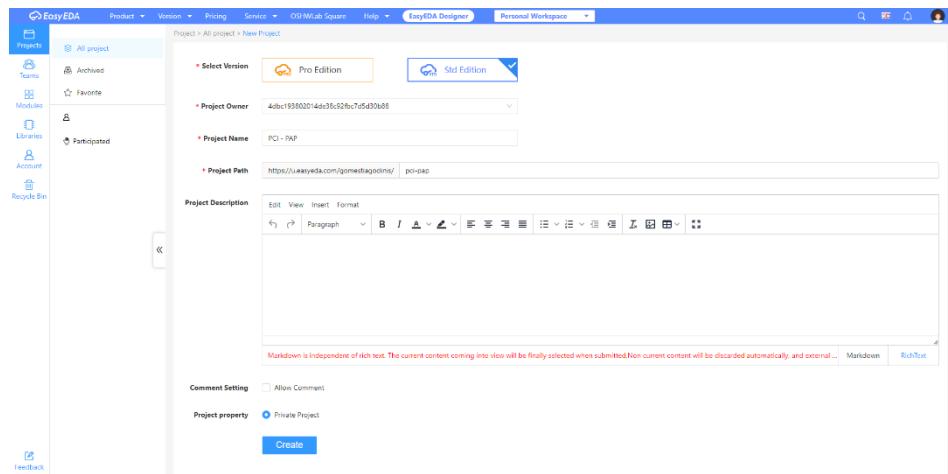


Figura 49: EasyEDA - Página de Criação do Projeto

Tendo o projeto criado, comecei a criar o esquema elétrico da placa. Em primeiro lugar, coloquei todos os componentes necessários na folha e organizei-os. Depois, liguei-os de forma a que nenhuma linha se cruze.

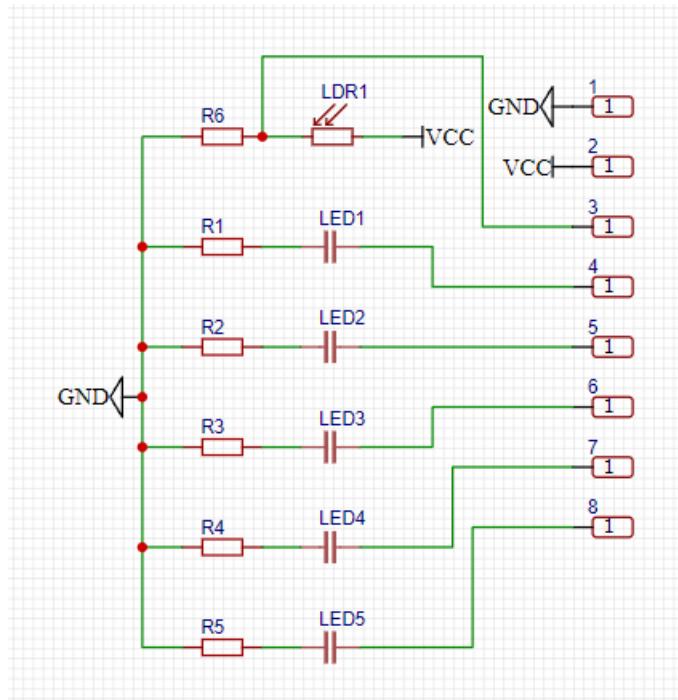


Figura 50: EasyEDA - Esquema Elétrico da PCI

Depois, na aba “Design”, converti o esquema para PCB, gerando uma “PCI editável”.

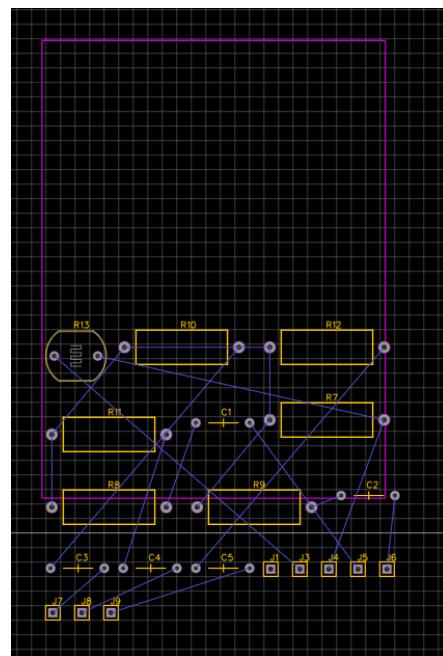


Figura 51: EasyEDA - "PCI Editável"

Como os componentes são todos colocados de forma desorganizada, tive que organizá-los manualmente. Para além disso, precisei de configurar alguns detalhes em relação ao tamanho das trilhas e dos buracos.

Com a placa organizada e configurada, é possível ver a sua previsualização 2D ou 3D.

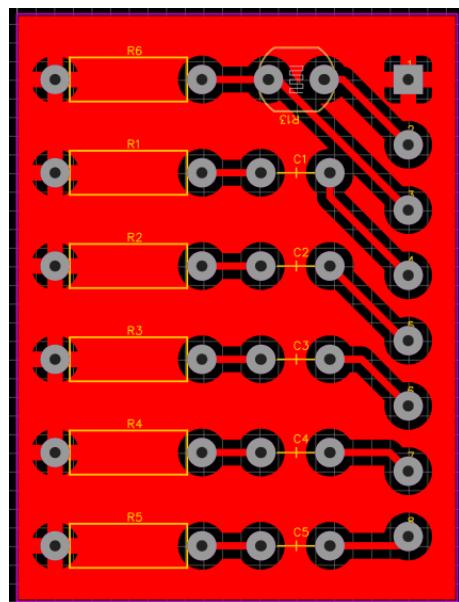


Figura 52: EasyEDA - PCI (Camada de Cima)

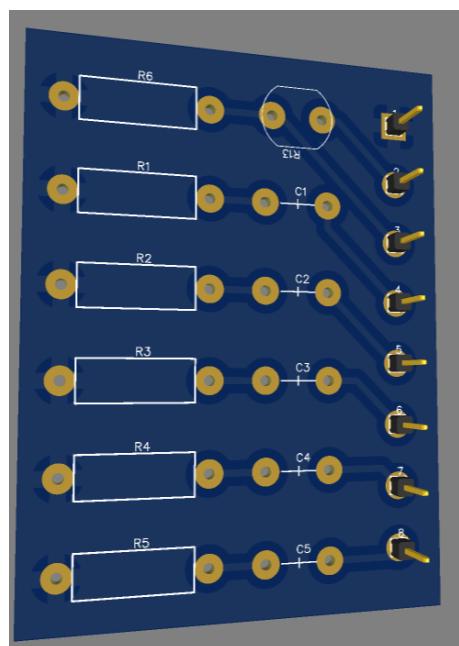


Figura 53: EasyEDA - PCI (Visualização 3D)

Por fim, guardei o esquema da PCI para um ficheiro Gerber.

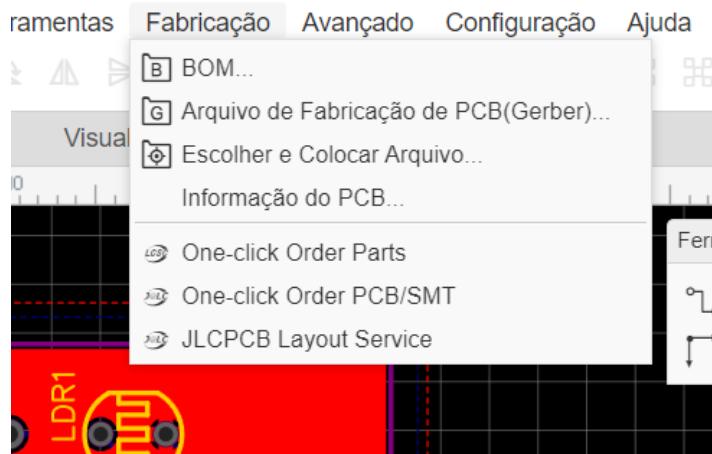


Figura 54: EasyEDA - Guardar como Ficheiro Gerber

## FlatCAM

Depois, tendo já extraído os ficheiros gerados pelo EasyEDA para uma pasta, carreguei para o FlatCAM apenas o ficheiro que me interessava. No meu caso, carreguei o ficheiro “Gerber\_BottomLayer”, ou seja, o ficheiro correspondente à camada de baixo.

Nome	Data de modificação	Tipo	Tamanho
Drill_PTH_Through	09/05/2024 17:17	Program FlatCAM	1 KB
Drill_PTH_Through_Via	09/05/2024 17:17	Program FlatCAM	1 KB
Gerber_BoardOutlineLayer	09/05/2024 17:17	Program FlatCAM	1 KB
<b>Gerber_BottomLayer</b>	09/05/2024 17:17	Program FlatCAM	33 KB
Gerber_BottomSolderMaskLayer	09/05/2024 17:17	Program FlatCAM	2 KB
Gerber_TopLayer	09/05/2024 17:17	Program FlatCAM	36 KB
Gerber_TopSilkscreenLayer	09/05/2024 17:17	Program FlatCAM	14 KB
Gerber_TopSolderMaskLayer	09/05/2024 17:17	Program FlatCAM	2 KB
How-to-order-PCB	09/05/2024 17:17	Documento de Te...	1 KB

Figura 55: Explorador de Ficheiros - Ficheiros Gerados pelo EasyEDA

Para abrir o ficheiro, no FlatCAM selecionei “File” > “Open” > “Open Gerber”. Depois, no explorador de ficheiros que foi aberto, selecionei o ficheiro apresentado anteriormente.

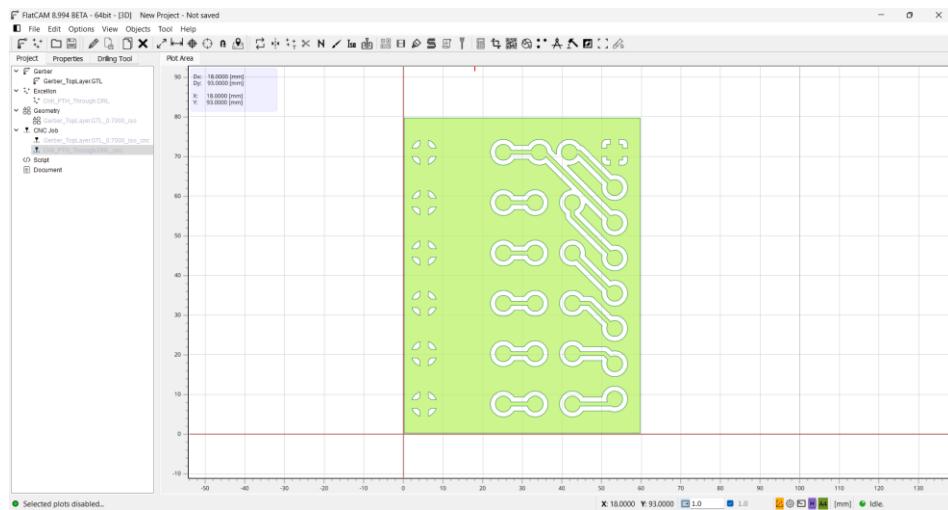


Figura 56: FlatCAM - Camada de Baixo da PCI

Depois, na aba do lado esquerdo, dei duplo-clique no ficheiro aberto na aba “Gerber”, abrindo um menu com várias opções de modificação para este objeto. Dentro de todas estas opções, a única que me interessa é a “Isolation Routing”, ou seja, o caminho isolante.

Como o tamanho **da fresa de utilizei na CNC era 0,7 milímetros**, defini no software o “Tool Diameter” como 0,7.

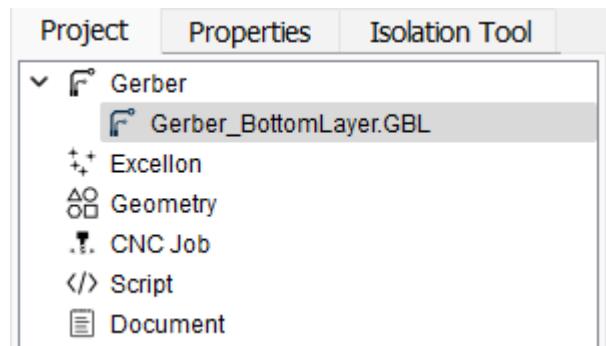


Figura 57: FlatCAM - Aba Lateral

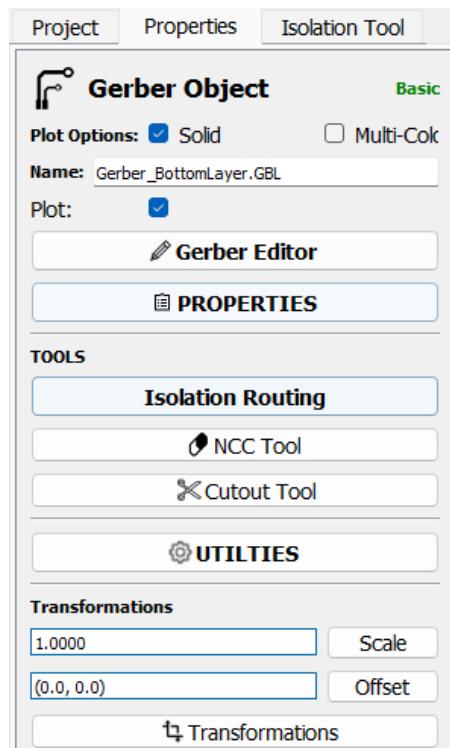


Figura 58: FlatCAM - Propriedades do Objeto Gerber

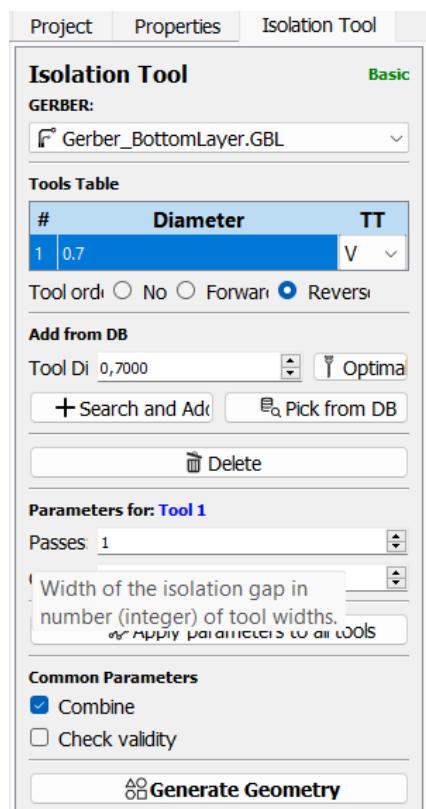


Figura 59: FlatCAM - Ferramenta de Isolamento

Depois, ao clicar no botão “Generate Geometry” será gerado um “Geometry Object”. Este objeto serve para que o FlatCAM seja capaz de manipular a geometria da placa para, posteriormente, criar o código G-Code com os locais onde a máquina deve passar.

Na aba deste objeto, para além de definir novamente o tamanho da fresa, tive que configurar outros parâmetros como o “feedrate” dos eixos X, Y e Z. Como a máquina que me foi disponibilizada já não era utilizada há algum tempo e, consequentemente, não recebia manutenção, tive que baixar as velocidades destes eixos para que esta passasse pelos sítios definidos sem perder passos, ou seja, contar passos a mais dos que realmente foram feitos.

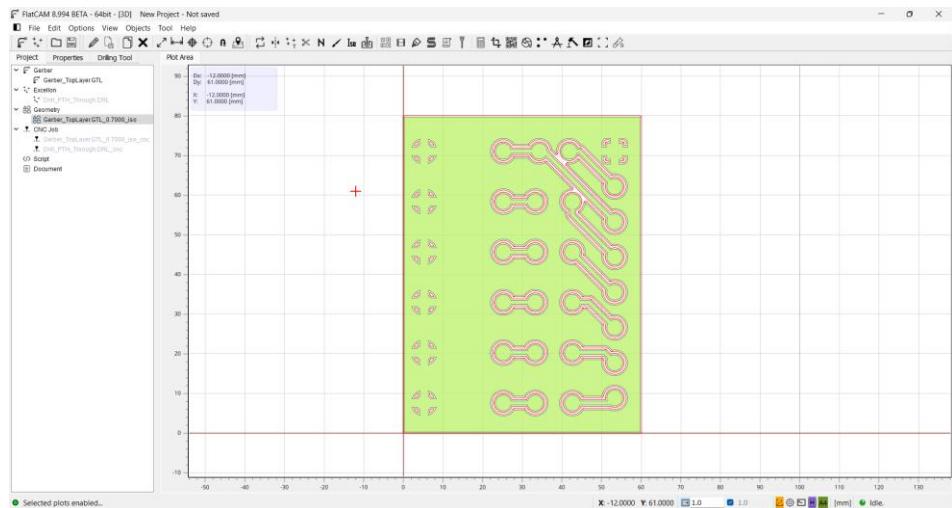


Figura 60: FlatCAM – Geometria da Placa

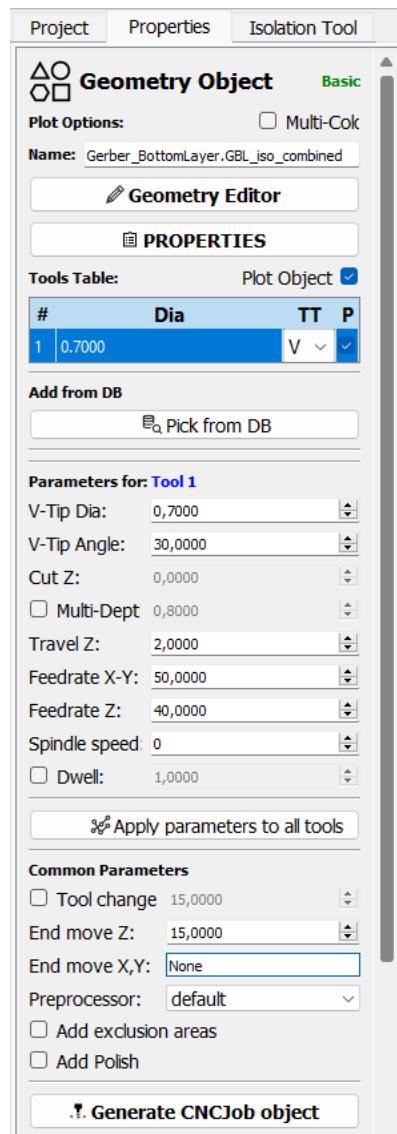


Figura 61: FlatCAM - Propriedades do Objeto de Geometria

De seguida, é só clicar no botão “Generate CNCJob object”. Este objeto irá calcular a rota que a CNC deve tomar e gerar um código G-Code. Por fim, salvei este último ficheiro gerado e transferi-o para o computador conectado à CNC.

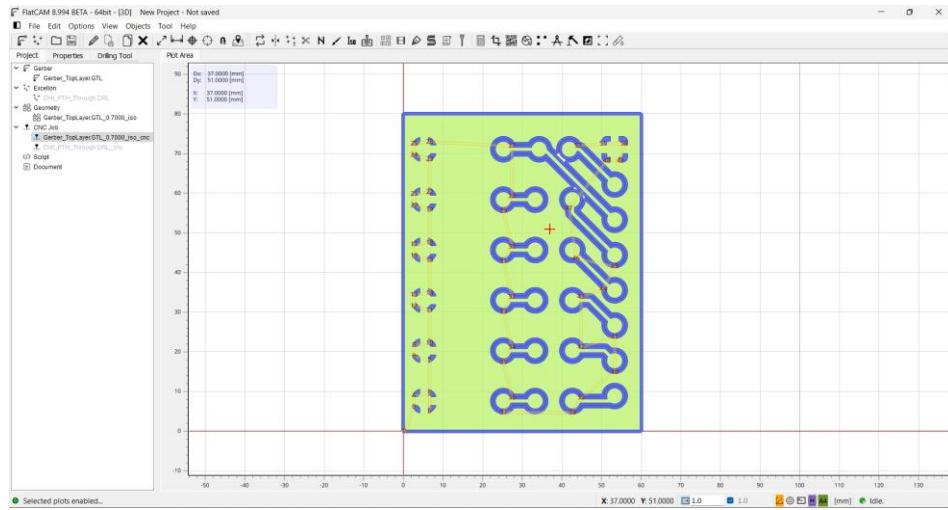


Figura 62: FlatCAM - Rota da CNC

```

41 M0
42 G00 Z15.0000
43
44 M03
45 G01 F50.00
46 G00 X0.3607 Y0.2091
47 G01 F40.00
48 G01 Z0.0000
49 G01 F50.00
50 G01 X30.1193 Y0.2091
51 G01 X30.1277 Y0.2098
52 G01 X30.1680 Y0.2176
53 G01 X30.1738 Y0.2191
54 G01 X30.1783 Y0.2208
55 G01 X30.1858 Y0.2248
56 G01 X30.2203 Y0.2486
57 G01 X30.2259 Y0.2533
58 G01 X30.2307 Y0.2587
59 G01 X30.2552 Y0.2942
60 G01 X30.2596 Y0.3028
61 G01 X30.2624 Y0.3120
62 G01 X30.2702 Y0.3523
63 G01 X30.2709 Y0.3607
64 G01 X30.2709 Y39.6392
65 G01 X30.2702 Y39.6476
66 G01 X30.2624 Y39.6879
67 G01 X30.2609 Y39.6937
68 G01 X30.2592 Y39.6983
69 G01 X30.2552 Y39.7057
70 G01 X30.2314 Y39.7402
71 G01 X30.2267 Y39.7458
72 G01 X30.2212 Y39.7506
73 G01 X30.1858 Y39.7751
74 G01 X30.1772 Y39.7796
75 G01 X30.1680 Y39.7823
76 G01 X30.1277 Y39.7901
77 G01 X30.1193 Y39.7908

```

Figura 63: FlatCAM - Excerto de Código G-Code

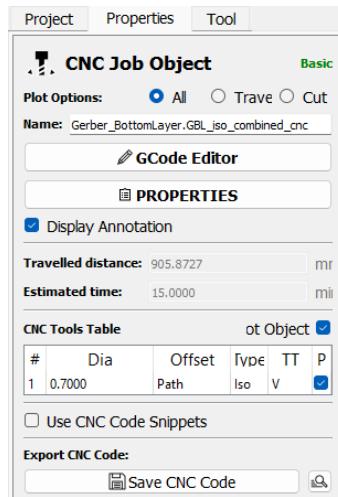


Figura 64: FlatCAM - Propriedados do Objeto de Trabalho da CNC

Para que a CNC comence a trabalhar, em primeiro lugar, tive que definir os 0's da placa, isto é, definir qual é o ponto de referência. Depois, precisei de carregar o ficheiro que criei no meu computador para o software que controla a CNC, neste caso o MECH3. Por fim, basta clicar no botão “Cycle Start”.

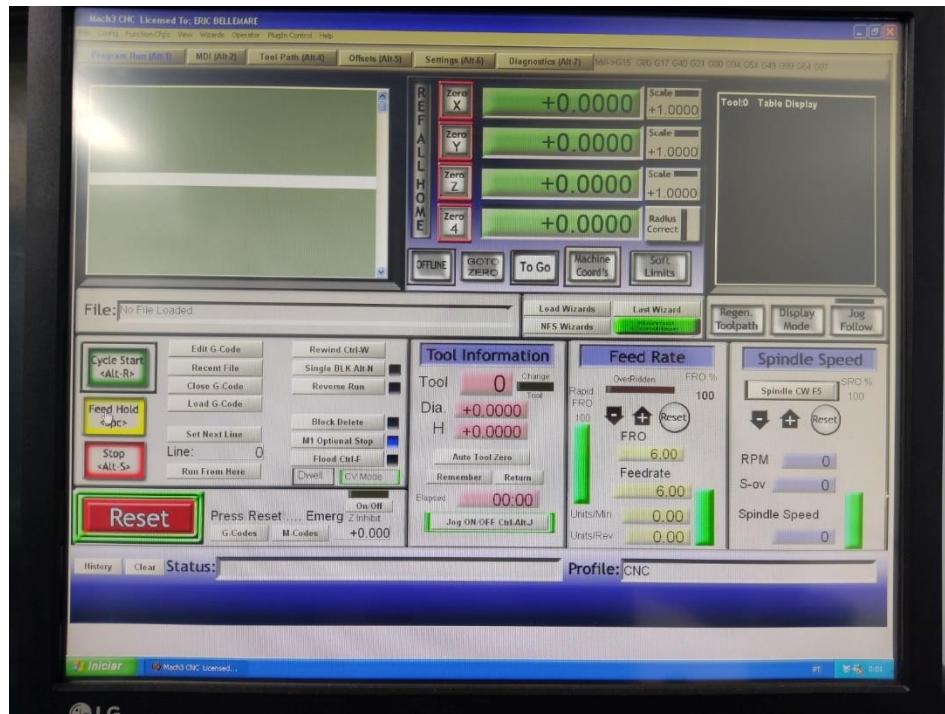


Figura 65: MECH3 – Interface do MECH3

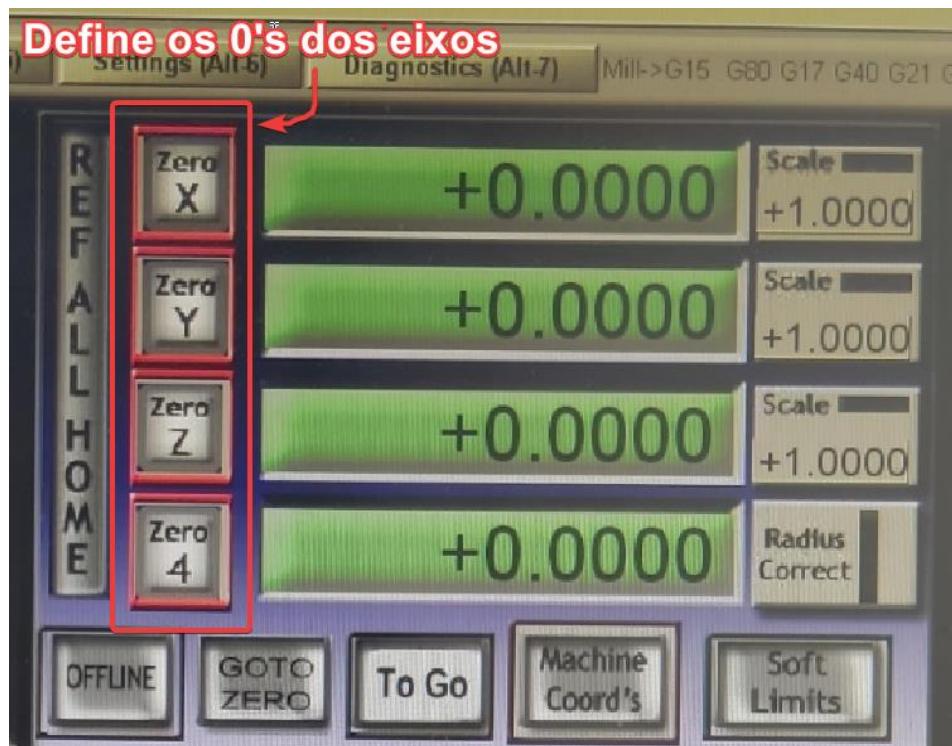


Figura 66: MECH3 - Controlo das Coordenadas da CNC

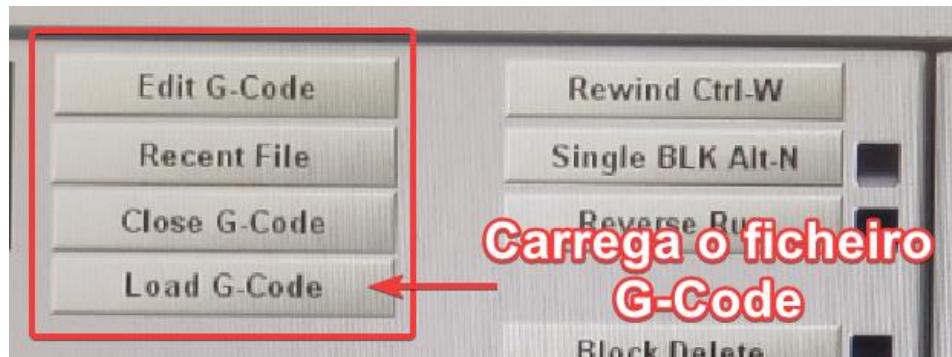


Figura 67: MECH3 - Controlo do ficheiro G-Code



Figura 68: MECH3 - Controlo do código G-Code

Com o software da CNC preparado para “imprimir” a minha placa, antes de passar para a placa de cobre, comecei por realizar alguns testes noutros materiais.

Em primeiro lugar, como ainda não tinha a máquina que iria fazer o desenho na placa, optei por utilizar um marcador e um pedaço de cartão.

Como é possível ver na imagem abaixo, como ainda não tinha ajustado a velocidade da CNC, esta saltou alguns passos e o desenho das trilhas ficou ligeiramente para a esquerda.

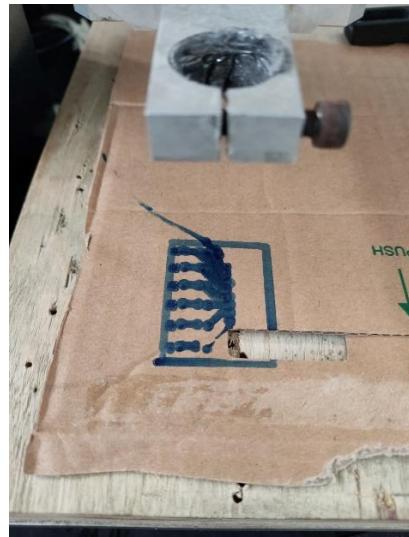


Figura 69: Teste - Marcador no Cartão

Depois, tendo já a máquina, realizei alguns testes na madeira antes de passar para o cobre para verificar certos parâmetros da máquina como a velocidade a que CNC tinha de estar para efetuar o desenho sem perder passos ou a altura que tinha de descer para que as trilhas ficassem o mais fino possível.



Figura 70: 1º Teste - Dremel na Madeira



Figura 71: 2º Teste - Dremel na Madeira

Com tudo testado e tendo a certeza de que o desenho da placa seria bem realizado, coloquei a CNC para “imprimir” no cobre. Para além disso, após a CNC ter terminado de desenhar as trilhas, utilizei um outro ficheiro que criei no FlatCAM para esta desenhar um pequeno buraco nos locais onde seriam feitas as perfurações.



Figura 72: Foto 1 – Desenho das Trilhas



Figura 73: Foto 2 – Desenho das Trilhas

No fim da execução dos dois ficheiros G-Code, utilizei os controlos manuais da CNC para terminar as perfurações na placa marcadas anteriormente.



Figura 74: Foto 3 - Perfuração

Para terminar, utilizei o dremel para separar a placa de cobre desenhada da placa de cobre original e passei álcool etílico para limpar as dedadas e a sujidade causada durante o desenho.



Figura 75: Foto 4 - Resultado Final

## Construção da Maquete

Ao iniciar a etapa da construção da maquete, acabei por abandonar o modelo que tinha inicialmente planeado e escolhi utilizar uma casa pré-fabricada em madeira.

Tomei esta decisão porque durante o meu estágio, que foi realizado relativamente longe de minha casa, optei por ir para um apartamento mais próximo por questões de praticidade. Sendo assim, não tinha os materiais, os meios ou tempo necessário para construir uma maquete do zero, visto que apenas estava em casa durante os fins-de-semana.

Já que tive a oportunidade de escolher um modelo pré-fabricado, escolhi utilizar a casa da icónica série animada “Simpsons” porque apresentava as características que procurava para o meu projeto.

Em primeiro lugar, comecei por fazer um furo na base da casa e numa das paredes para poder passar o cabo que liga o Arduino ao computador e os fios que vão da PCI (placa de circuitos impressos) para os seus respetivos componentes.



Figura 76: Maquete - Furos das Bases e Parede

Depois, para simular o jardim que idealizei, utilizei uma folha de espuma EVA verde que colei à base com fita de dupla face.



Figura 77: Maqueta - Colagem da folha de espuma EVA

Tendo a base pronta, comecei a montar a casa.



Figura 78: Maqueta - Início da Construção da Casa

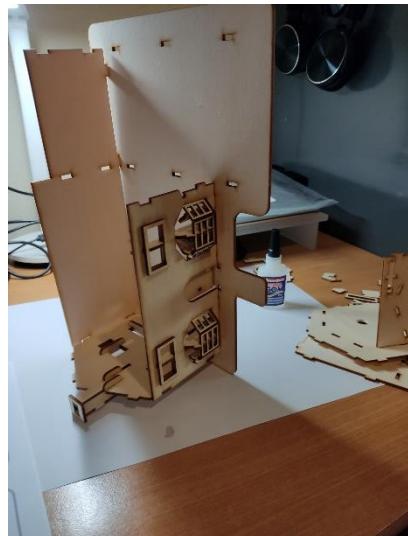


Figura 79: Maquete - Colagem das Peças



Figura 80: Maquete – Resultado Final (Sem Telhado)

Com a casa também já pronta, só restava implementar o circuito. Para isso comecei por dar uns toques finais à PCI, retirando algumas imperfeições, lixando e limpando com álcool etílico.

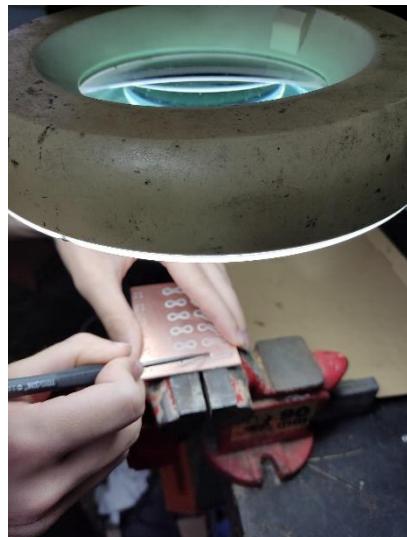


Figura 81: Maquete - Retirada de Imperfeições da PCI

Após este processo, confirmei se havia continuidade de corrente nos devidos locais utilizando um multímetro.



Figura 82: Maquete - Verificação da Continuidade da Corrente

Depois de todas estas etapas, comecei finalmente a soldar os componentes à placa. Para facilitar a fixação da solda na placa, coloquei um pouco de massa de soldar em todos os locais onde se iriam soldar os componentes.

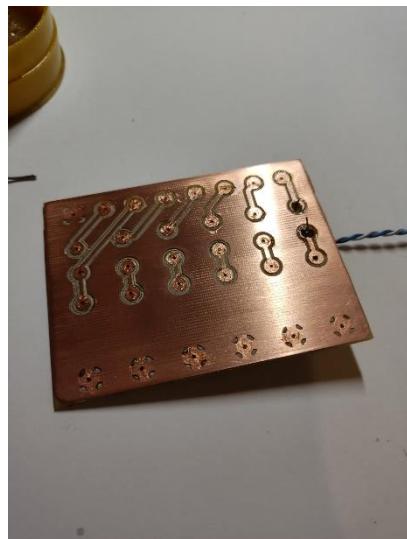


Figura 83: Maquete - Aplicação de Massa de Soldar na PCI

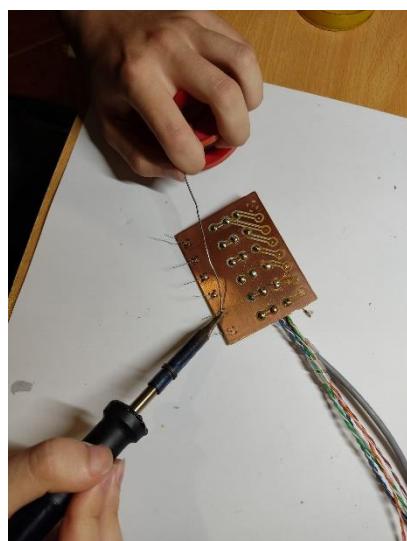


Figura 84: Maquete - Soldadura dos Componentes

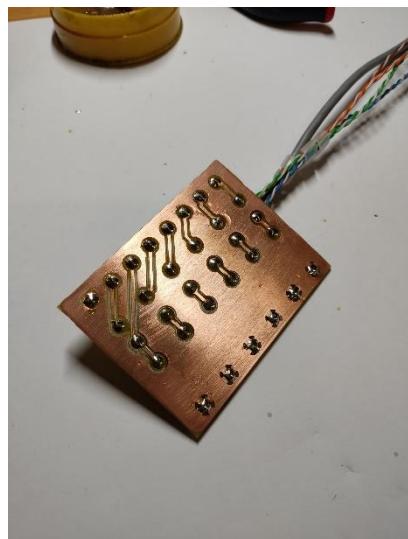


Figura 85: Maquete - PCI Finalizada (Trás)

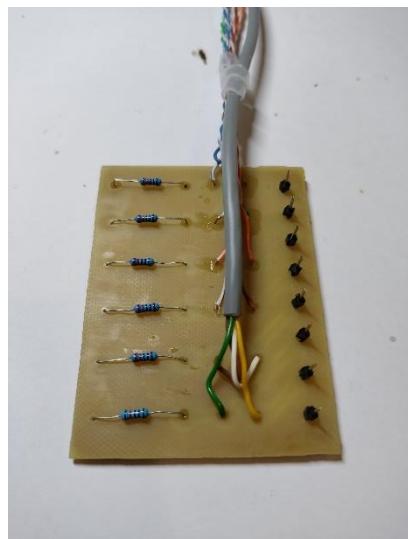


Figura 86: Maquete - PCI Finalizada (Frente)

Com a PCI pronta e todos os componentes soldados, instalei-a na maquete. Para finalizar, instalei todos as luzes LED na maquete e soldei-as ao circuito.



Figura 87: Maquete - Instalação da PCI na Casa



Figura 88: Maquete - Circuito Instalado na Casa

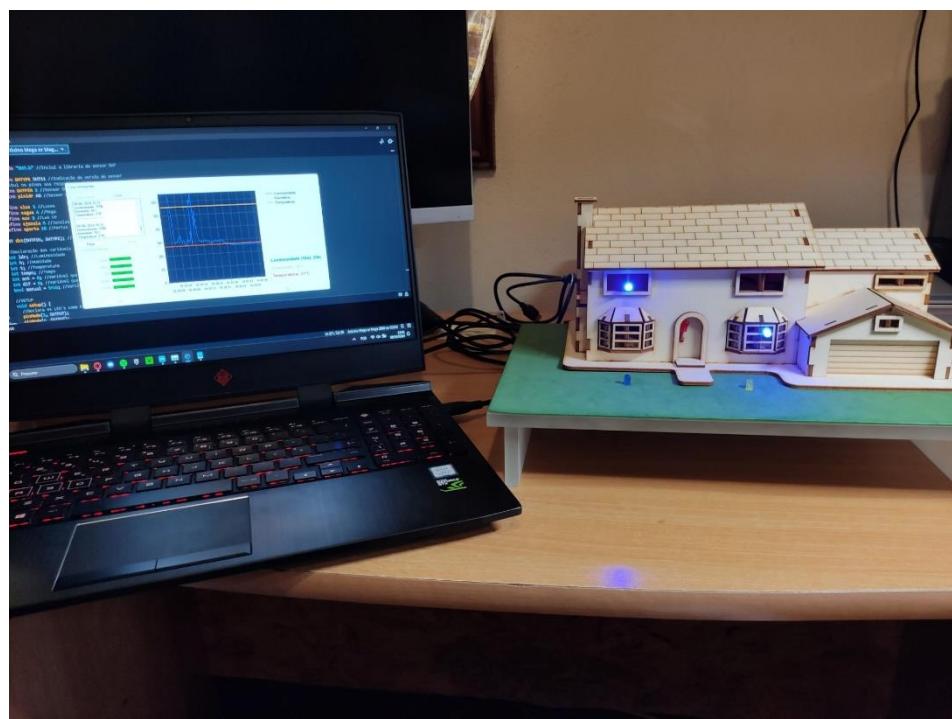


Figura 89: Resultado Final

## Conclusão

Acho que a escolha deste tipo de projeto foi uma mais valia para mim devido à quantidade de informação e conhecimento que adquiri durante o seu desenvolvimento. Pude aprender e/ou fortalecer as minhas bases em diversas áreas como a modelagem 3D, eletrónica, programação e trabalhos manuais.

Para começar, a criação de um modelo 3D, tanto da casa como do circuito, facilitou bastante o planeamento e desenvolvimento do projeto, dando-me desde o princípio uma ideia visual de como ficaria o resultado final e o que seria necessário para lá chegar.

Depois, a testagem dos componentes de forma individual garantiu-me que pudesse perceber o seu funcionamento de forma mais simples e rápida e sem a possibilidade de danificar outros componentes accidentalmente. Com estes componentes testados, criei o circuito final e um código-base para o controlo das funções principais do projeto, fundamentando as minhas bases na eletrónica e programação de circuitos.

De seguida, com a criação da aplicação, tive a oportunidade de trabalhar com uma linguagem de programão nova, C#, que apesar de ser similar à linguagem utilizada pelo Arduino (C++ levemente modificado) e ter facilitado no seu processo de aprendizagem, também teve os seus obstáculos.

Posteriormente, contruí uma placa de circuitos impressos para garantir que o circuito não se desfizesse. Durante o seu processo de criação, tive a possibilidade de trabalhar com outra tecnologia nova. Para criar a PCI foi-me disponibilizada uma máquina CNC no local de estágio, onde tive que aprender a desenvolver um modelo da placa e transformar em código G-Code, o código utilizado na CNC.

Por fim, ao fazer o trabalho de solda e a montagem da maquete, pude desenvolver a minha experiência e domínio em trabalhos manuais, para além de ter aprendido a soldar componentes.

Durante o desenvolvimento deste projeto, as partes que mais gostei foram o desenvolvimento da aplicação para o Arduino e a criação da PCI. Através destas duas etapas, pude aprender uma linguagem nova, como comunicar e controlar o Arduino

através de uma comunicação serial e o processo de criação de uma PCI utilizando uma máquina CNC.

Por outro lado, a parte que menos gostei e que até pode ser visto como uma desvantagem deste projeto foi conciliar o seu desenvolvimento com o estágio. Durante este período de meses, estive num apartamento perto da minha empresa de estágio por questões de conveniência, onde não dispunha de qualquer ferramenta manual para o seu desenvolvimento. Sendo assim, a única altura em que podia desenvolver o a parte física do projeto, como a maquete e a soldadura, era durante os fins-de-semana, quando regressava a casa.

Durante o seu avanço, fui fazendo algumas alterações ao projeto devido a vários fatores, como o referido anteriormente, sendo as principais a remoção do painel solar do circuito e a alteração do modelo da casa utilizado.

Resumindo, gostei bastante de ter desenvolvido este projeto e poder explorar e “brincar” com as funcionalidades do Arduino. A sua escolha trouxe bastantes vantagens para a minha aprendizagem, tendo adquirido conhecimentos tanto a nível de software como a nível de hardware. Para além disso, com a construção da maquete e da PCI pude aperfeiçoar as minhas habilidades em trabalhos manuais e soldadura.

## Webgrafia

Marciovcampos – *Casa Inteligente Com Arduino* [Em linha]. [Consult. 07 Nov. 2023]. Disponível na Internet: <https://www.instructables.com/Casa-Inteligente-Com-Arduino/>

Vida de Silício – *Como controlar Entradas e Saídas Digitais do Arduino* [Em linha]. 17 Abr. 2017. [Consult. 07 Nov. 2023]. Disponível na Internet: <https://www.arduinoportugal.pt/entradas-saidas-digitais/>

Mattede, Henrique – *Aprenda como Calcular Resistor para LED?* [Em linha]. [Consult. 16 Nov. 2023]. Disponível na Internet: <https://www.mundodaeletrica.com.br/aprenda-como-calcular-resistor-para-led/>

Icarooo – *Esquema para chavear entre 2 fontes de energia com relê* [Em linha]. 9 Jul. 2016. [Consult. 16 Nov. 2023]. Disponível na Internet: <https://eletronicabr.com/forums/topic/78396-esquema-para-chavear-entre-2-fontes-de-energia-com-rel%C3%A9/>

Robot!Education – *Não queime seu Arduino* [Em linha]. [Consult. 21 Nov. 2023]. Disponível na Internet: <https://roboteducation.com.br/planos-de-aula/arduino/aula-sobre-seguranca-como-nao-queimar-seu-arduino/>

Mouro, Catarina – *Como criar Sensor Temperatura e Umidade Arduino | DHT11* [Em linha]. 2021. [Consult. 04 Jan. 2024]. Disponível na Internet: <https://www.youtube.com/watch?v=obmUmRir6Fk>

Correia Viana, Carol – *Controle de Luz com sensor LDR e Arduino* [Em linha]. 2 Out. 2020. [Consult. 05 Jan. 2024]. Disponível na Internet: <https://www.youtube.com/watch?v=obmUmRir6Fk>

Alves, Pedro – *LDR – Como é e como funciona!* [Em linha]. [Consult. 08 Jan. 2024]. Disponível na Internet: <https://www.manualdaeletronica.com.br/ldr-o-que-e-como-funciona/>

TME Eletronic Componentes – *Elementos Passivos E Ativos: Exemplos, Funções e Diferenças* [Em linha]. 28 Nov. 2022. [Consult. 08 Jan. 2024]. Disponível na Internet: <https://www.tme.eu/pt/news/events/page/48387/elementos-passivos-e-ativos-exemplos-funcoes-e-diferencias/>

MASTERPLANTS – *Luz Ultravioleta no Cultivo: como obter rendimento máximo nas colheitas com o correto dimensionamento da iluminação no cultivo indoor* [Em linha]. [Consult. 8 Jan. 2024]. Disponível na Internet: <https://masterplants.com.br/luz-ultravioleta-no-cultivo-como-obter-rendimento-maximo-nas-colheitas-com-o-correto-dimensionamento-da-iluminacao-no-cultivo-indoor/>

Souza, Fábio – *Comunicação Serial em C# e Arduino – Parte 1* [Em linha]. 5 Ago. 2014. [Consult. 30 Jan. 2024]. Disponível na Internet: <https://embarcados.com.br/comunicacao-serial-c-arduino-parte-1/>

Guimarães, Fábio – *Arduino-Executar duas coisas ao mesmo tempo – Aula 9 – AI* [Em linha]. 8 Nov. 2017. [Consult. 31 Jan. 2024]. Disponível na Internet: <https://mundoprojetado.com.br/executar-duas-coisas-ao-mesmo-tempo/>

Macoratti, José Carlos – *C# - Trabalhando com Threads* [Em linha]. [Consult. 31 Jan. 2024]. Disponível na Internet: [https://www.macoratti.net/10/09/c\\_thd1.htm](https://www.macoratti.net/10/09/c_thd1.htm)