



**RELATÓRIO DO PROJETO DE
PROJECT FACTORY**

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Professor Orientador: André Sabino

André Custódio, 20220112

André Mendes, 20220355

Steve Vilas, 20200856

28/05/2025

O relatório encontra-se em condições para ser apresentado

Ciclo de Formação 2022/2025
Ano Letivo 2024/2025

Agradecimentos

Agradecemos ao nosso Coordenador de Curso, Professor André Sabino, por nos ter providenciado e proporcionado um ensino de qualidade, puxando sempre por nós para dar o nosso melhor e tentar-nos sempre ajudar. Agradecer também ao professor Nathan Campos, por ser um excelente professor, ensinar-nos a ser boas pessoas, bons programadores e estar sempre disponível para dar uma mãozinha.

Também queremos agradecer ao professor Vasco, por dar conselhos para o veículo, ajudar a passar alguns obstáculos que apareceram na nossa construção e abrir os olhos para os desafios do veículo. Também gostávamos de agradecer ao Bernardo por disponibilizar a sala 14/13 para testarmos o veículo e possibilitar usarmos os objetos que irão ser utilizados no nosso desafio de *PBL*.

Por fim, agradeço a todos os professores a disponibilidade e ajuda, que nos têm dado no nosso percurso escolar, sem os quais não o conseguiria concluir estes três anos com o sucesso que tivemos. E por último e não menos importante, agradecer aos meus colegas do Grupo 4, sem eles, não era possível concluir este projeto! Um grande obrigado!

Índice

Agradecimentos	2
Índice	3
Índice de Imagens.....	5
Introdução	6
Capítulo I – Cronograma Inicial	7
Descrição do Cronograma.....	8
Capítulo II – Conceção do Projeto	9
Objetivos	9
Tecnologias	10
<i>Framework</i>	10
O que é um <i>framework</i> ?	10
Vantagens da <i>Framework</i>	10
Protocolo de Comunicação	11
<i>MQTT</i>	11
Código inicial de <i>MQTT</i>	11
Peças para o Arduino.....	12
➤ Bateria <i>Lipo 2s 5200 mAh 50C 7,4 V</i>	12
Linguagem de Programação.....	13
C#.....	13
C	13
Recursos Necessários para o Programa.....	14
<i>Visual Studio Code</i>	14
<i>Visual Studio 2022</i>	14
<i>Arduino IDE</i>	15
<i>MQTT Explorer</i>	15
Capítulo III – O Projeto	16
Projetos que nos inspiraram	16
Ideia 1.....	16
Ideia 2.....	17
Ideia 3.....	17
Ideia do nosso projeto	18

Arquitetura do Programa.....	19
Imagens antigas do veículo.....	19
Imagens atuais do veículo.....	20
Diagrama de Circuitos Necessários	22
Dashboard	23
Planeamento do Projeto	25
Distribuição de Tarefas	25
Proposta de plataformas padrão, funcionalidades a implementar.....	25
Análise Ética do PBL.....	26
Recursos Utilizados.....	27
Ferramentas de Desenvolvimento:	30
Ferramentas para desenvolvimento de apresentação e relatório:.....	30
Aplicação/Site de Comunicação:	30
Capítulo IV – Cronograma Final e Justificação de desvios	31
Justificação.....	32
Capítulo V – Análise do percurso pessoal	33
Conclusão.....	34
Bibliografia e Web Grafia.....	35

Índice de Imagens

Imagem 1 - Cronograma inicial	7
Imagem 2 - Imagem do Logo	9
Imagem 3 - Logo das linguagens de Programação	10
Imagem 4 - MQTT.....	11
Imagem 5 - Codificação para ligar ao MQTT	11
Imagem 6 - C#	13
Imagem 7 - C	13
Imagem 8 - IDE Visual Studio Code	14
Imagem 9 - IDE Visual Studio 2022.....	14
Imagem 10 - IDE Arduino IDE	15
Imagem 11 - MQTT Explorer	15
Imagem 12 - Carro ideia 1.....	16
Imagem 13 - Ideia carro 2	17
Imagem 14 - Carro ideia 3.....	17
Imagem 15 - esboço do veículo 3D em blender	18
Imagem 16 - esboços em papel	18
Imagem 17 - Arquitetura do projeto S.T.A.R	19
Imagem 18 – Imagens antigas do Veículo do PBL.....	19
Imagem 19 - Imagem atual do veículo visto de lado, sem caixa.....	20
Imagem 20 - Imagem atual do veículo vista de cima, sem caixa	20
Imagem 21 - Imagem atual do veículo vista de lado com caixa.....	20
Imagem 22 - Imagem atual do veículo visto de cima, com caixa e um cubo.....	21
Imagem 23 - Sistema de suporte do chassi e parte de cima do chassi	21
Imagem 24 - Diagrama de Circuito do Veículo.....	22
Imagem 25 - Dashboard do PBL	23
Imagem 26 - Dashboard com alguns dados vindos do ESP32.....	24
Imagem 29 - Imagem de Logo.....	26
Imagem 30 - Visual Studio Code.....	27
Imagem 31 - <i>Visual Studio 2022</i>	27
Imagem 32 - Arduíno IDE.....	27
Imagem 33 - Microsoft Word	27
Imagem 34 - Microsoft Excel	28
Imagem 35 - Microsoft Power Point	28
Imagem 36 - Brave	28
Imagem 37 - Discord.....	28
Imagem 38 - GitHub	28
Imagem 39 - <i>Docker</i>	29
Imagem 40 - MQTT Explorer	29
Imagem 41 - Cronograma Final.....	31

Introdução

A faculdade IADE – Faculdade de Design, Tecnologia e Comunicação tem como principal objetivo ajudar a desenvolver nos alunos competências para o exercício de uma profissão. A faculdade destaca-se pela sua articulação com as empresas, garantindo uma forte ligação ao mundo do trabalho e permitindo o prosseguimento de estudos. O meu curso, técnico de Licenciatura em Engenharia Informática, tem como principal objetivo dar competências na área de criação de software, instalação e manutenção do hardware do computador, gestão de redes informáticas e desenvolver aplicações desktop e web, aos alunos.

No seu plano curricular existe vários PBL (*Project Base Learning*) onde os alunos têm de desenvolver softwares ou soluções durante um semestre com certas restrições ou regras. Este semestre temos 250 horas para desenvolver um veículo que transporta material. No fim deste relatório irei fazer uma análise acerca do meu percurso pessoal deste semestre, finalizando com uma conclusão e uma *Webgrafia*.

Capítulo I – Cronograma Inicial

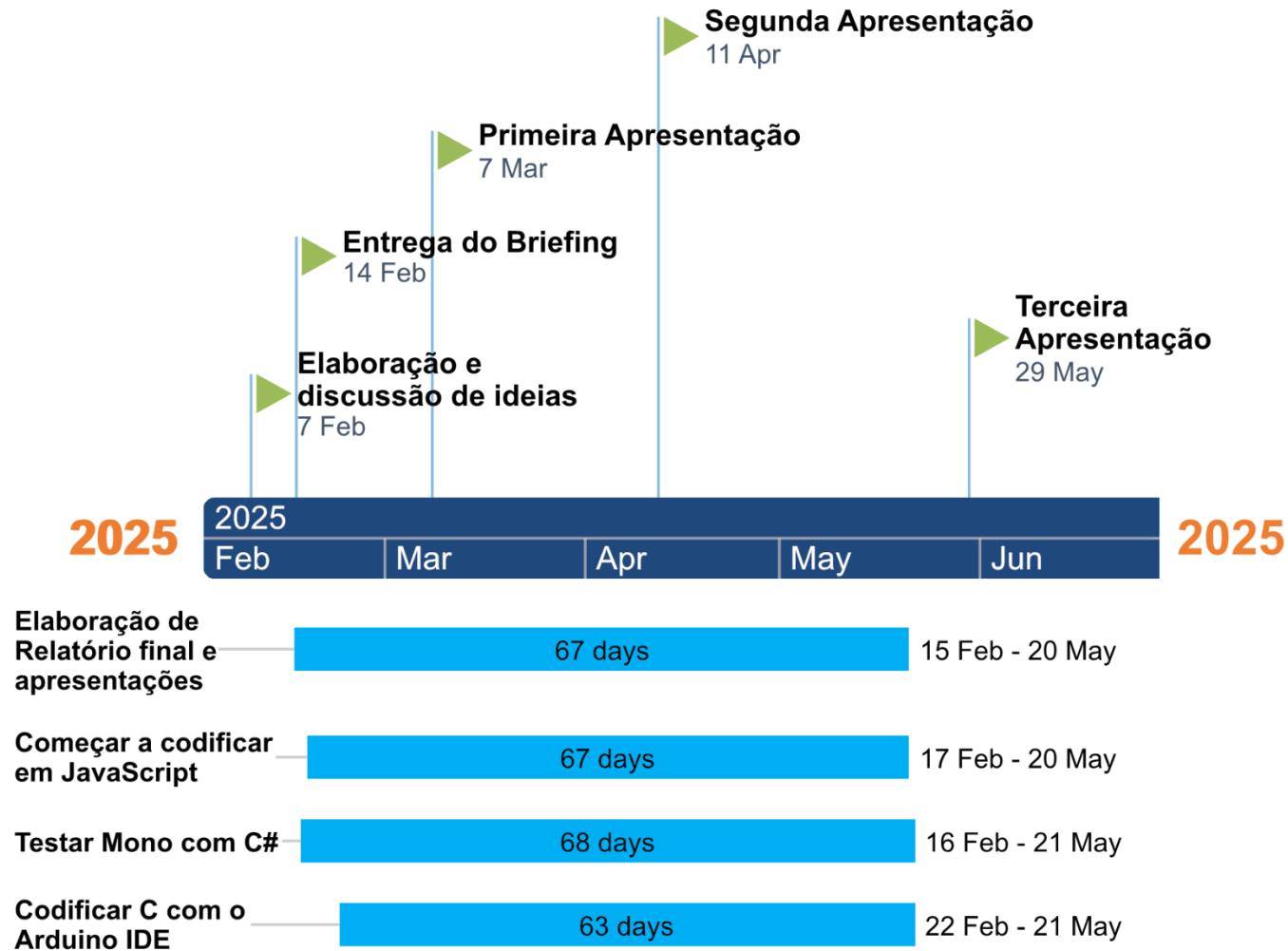


Imagem 1 - Cronograma inicial

Descrição do Cronograma

O desenvolvimento deste trabalho irá decorrer entre os dias 7 de fevereiro a 29 de maio de 2025.

Elaboração de Relatório final e apresentações começou no dia 15 de fevereiro a 20 de maio. A testagem de *Mono* com *C#* começou dia 16 de fevereiro a 21 de maio de 2025.

Começo da codificação em *Javascript* foi 17 de fevereiro a 20 de maio. A codificação *C* com o *Arduíno IDE* foi de 22 de fevereiro a 21 de maio de 2025.

Durante o desenvolvimento deste projeto irá haver três momentos de avaliação, o primeiro está agendado para 7 de março de 2025, a segunda avaliação está marcada para 11 de abril de 2025 e para a terminar a terceira e última avaliação está marcada para 29 de maio de 2025.

Capítulo II – Conceção do Projeto

Projecto: *S.T.A.R*

Descrição do Projeto: *Surveillance & Tactical Autonomous Rover* (Sistema Terrestre de Análise e Reconhecimento) ou o seu acrónimo *S.T.A.R* é um veículo com a capacidade de ajudar o homem em algumas tarefas difíceis ou impossíveis. Este veículo irá possibilitar ajudar meteorologistas em algumas atividades como, transporte de material de locais perigosos, medir temperatura do espaço, gravar o som do espaço, entre outras tarefas mais pequenas.

Estes valores irão estar todos recebidos pelo *ESP32* e visualizados numa *dashboard*.



Imagem 2 - Imagem do Logo

Objetivos

- Ambiente de utilização
 - Ouvir o que se passa no espaço;
 - Transporte de material para locais perigosos;
 - Medir a temperatura do espaço;
 - Sensor de Fumo;
 - Reproduzir som;
 - Medir Humidade no espaço;
- Rapidez e fluidez
 - Ser rápido e responsivo
 - Ter uma interface apelativa e intuitiva para os utilizadores, facilitando o uso de quem está a usar.

Tecnologias

O nosso projeto utiliza uma *Framework* (.Net Framework) e duas linguagens de programação (C# e C).



Imagem 3 - Logo das linguagens de Programação

Framework

O que é um *framework*?

Um *framework* consiste numa abstração que une códigos entre vários projetos de *software*, fornecendo uma funcionalidade sem pormenores. Trata-se de um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação. Em vez de ser as bibliotecas a controlar é o *framework* quem dita o controlo da aplicação.

Vantagens da *Framework*

A *framework* é rápida no seu *debug*, na solução do projeto. A linguagem de programação ou o *Software Development Kit* evolui com ajuda da comunidade ou das empresas, tem uma grande auxílio em fóruns caso os programadores tenham dúvidas.

A comunidade quando deteta um problema de segurança é rapidamente atualizada e modificado esse problema.

Protocolo de Comunicação

MQTT

O *MQTT* foi criado pela *IBM* em 1999 como um protocolo de comunicação leve, eficiente e ideal para comunicações em redes instáveis ou com pouca largura de banda. Baseado no modelo *publish/subscribe*, permite que dispositivos troquem mensagens através de um *broker*, sem necessidade de conexão direta entre eles.



Imagem 4 - MQTT

É muito utilizado em sistemas de *IoT* (*Internet das Coisas*), automação residencial, sensores remotos e aplicações que exigem comunicação em tempo real. O protocolo é simples, rápido e consome poucos recursos, o que o torna ideal para dispositivos com capacidade limitada.

Código inicial de MQTT

```
public void StartMQTT(string IPV4, string Path, Form1 frm)
{
    // create client instance
    warning disable CS0618 // Type or member is obsolete
    client = new MqttClient(brokerIpAddress: (IPAddress.Parse(IPV4)));
    warning restore CS0618 // Type or member is obsolete

    client.MqttMsgPublishReceived += client_MqttMsgPublishReceived;

    string clientId = Guid.NewGuid().ToString();
    client.Subscribe(new string[] { Path }, new byte[] { MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE });
    client.Connect(clientId);
    this.frm = frm;
}
```

→ Ip da Maquina

→ Topic do MQTT

→ fazer uma comunicação!

Imagem 5 - Codificação para ligar ao MQTT

Peças para o Arduíno

Para o nosso veículo estar bem estruturado e preparado para os desafios precisamos que ele não colida contra paredes, consiga subir obstáculos, virar, entre outros desafios. Para tal, iremos precisar de:

- 4 peças de *Geared Motoro* DC3V-12V + 4 rodas;
- *BreadBoard*
- *Ultrasonic Sensor*;
- *Fire Sensor*;
- *Temperature Sensor + Humidity Sensor*;
- *Sound Sensor*;
- 2 *h-Bridges*;
- Bateria *Lipo* 2s 5200 mAh 50C 7,4 V
- *Buck-Converter*

Linguagem de Programação

C#

Microsoft em 2000 criou o *C#*, uma linguagem simples, moderna, orientada por objetos, flexível e versátil. É semelhante ao *C++* e *Java*, só em 2002 foi lançada para a comunidade.

As suas implementações mais utilizadas são *.Net Framework* que utiliza o *form (Framework)* e *.Net Core* que utiliza a linha de comandos. É uma linguagem utilizada em jogos, aplicações de clientes, aplicações webs, inteligência artificial e muitos mais.

A comunidade tem dado uma grande ajuda, na evolução da linguagem e na criação de bibliotecas, que auxiliam na codificação da aplicação.

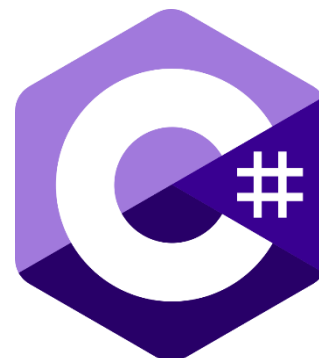


Imagem 6 - C#

C

O Sr. Dennis MacAlistair Ritchie foi o criador da linguagem *C*, uma linguagem que tem suporte a *structured Programing*, *lexical variables* e recursão.

Originalmente a linguagem foi pensada para o desenvolvimento de sistemas operativos, incluindo o *Unix*. Atualmente a linguagem continua a ser usada, devido a ser uma linguagem de baixo nível, fazendo uma codificação próxima do hardware.



Imagem 7 - C

Recursos Necessários para o Programa

Visual Studio Code

Visual Studio Code é um editor de código fonte, possível usar em todos os sistemas operativos. O *Visual Studio*, contem extensões para ajudar o utilizador a programar ou a publicar o seu código, o exemplo é o *GitLens* que envia os projetos para o *GitHub*.

Foi desenvolvida pela *Microsoft* e foi programado com o *TypeScript*, *JavaScript* e *Css*.

Algumas linguagens dependem de alguns recursos, um deles é a depuração, processo que tenta encontrar erros, tanto no hardware ou software.

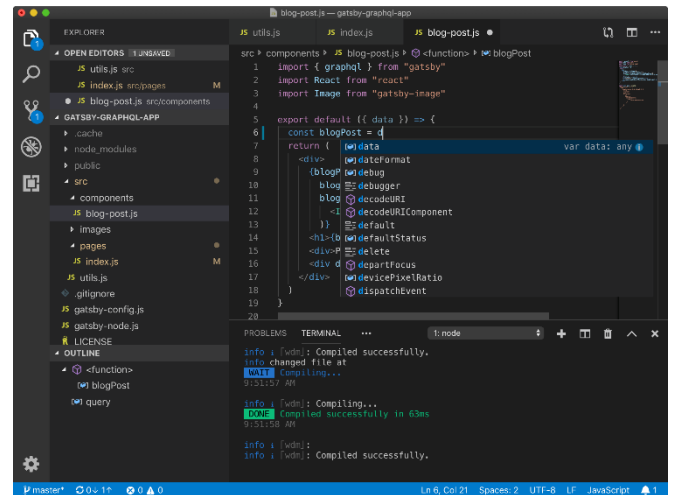


Imagem 8 - IDE Visual Studio Code

Visual Studio 2022

O *Visual Studio 2022* é uma aplicação desenvolvida pela *Microsoft*, a sua primeira versão foi em 1997, *Visual Studio 97*. É uma *IDE*, ambiente de desenvolvimento integrado que auxilia na criação dos objetos e na sua localização.

A mesma contem *Visual Basic*, *C*, *C++*, *C#*, *F#*, *JavaScript*, *Python*, *Type Script* e outras mais. Este contem *templates .Net Framework*, *.Net Core*, *AspNet* e muitos mais.

O *Visual Studio* tem muitas boas vantagens, desde auxílio na codificação, rápido *debug* e possível colaboração de trabalho entre colegas.

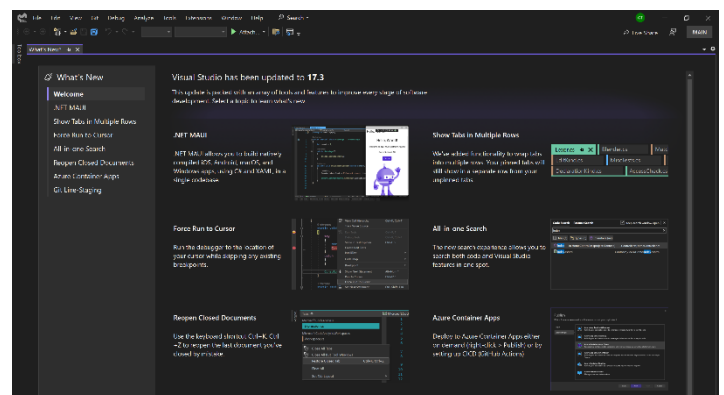


Imagem 9 - IDE Visual Studio 2022

Arduino IDE

O *Arduino IDE (Integrated development environment)* é um editor de código fonte, dedicada ao desenvolvimento. Este editor ajuda a enviar o código em *flash* para os arduíno que tiverem ligados ao computador do programador. Este editor foi feito pela Arduino Software, disponibilizado aos clientes no ano 2021.

A mesma originalmente foi escrita em *Java, C* e *C++*, mas a versão mais atualizada (20 de fevereiro de 2024) está escrita em *TypeScript, JavaScript* e *Go*. A versão mais atual contém nova gestão de *boards*, nova gestão de bibliotecas, novo explorador de projetos, *dark mode* e suporte a 64 bits.

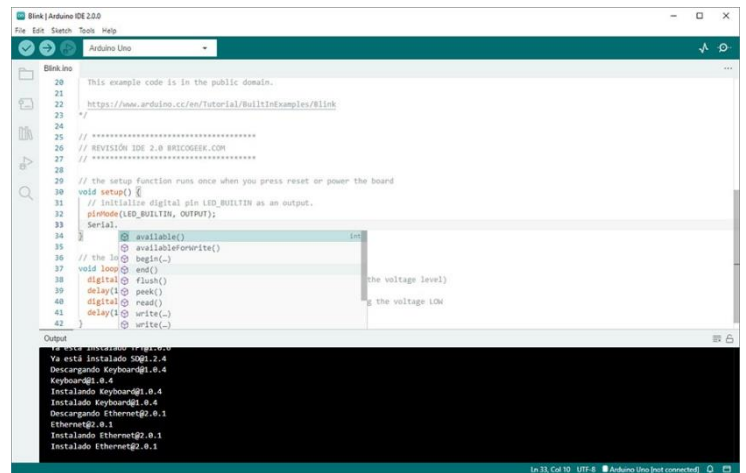


Imagem 10 - IDE Arduino IDE

Atualmente este *IDE* encontra-se disponibilizado para os vários sistemas operativos, como *Windows, Mac* e *Linux*.

MQTT Explorer

MQTT Explorer é uma aplicação para visualizar e ter gestão de tópicos *MQTT* em tempo real. Esta aplicação permite visualizar e interagir com os dados publicados em um broker *MQTT*, tornando mais simples a análise e o *debug* de sistemas *IoT* que utilizam esse protocolo. O software foi desenvolvido por *Thomas Nordquist* e encontra-se disponível gratuitamente desde 2018.

O *MQTT Explorer* foi desenvolvido em *Electron, JavaScript* e *React*, oferecendo uma interface moderna e intuitiva de modo a visualizar hierarquia dos tópicos, histórico de mensagens, publicação direta de mensagens e suporte a múltiplas conexões. A versão mais recente, lançada em 2023, inclui melhorias de desempenho, suporte para *payloads* em formato *JSON*, texto ou binário, e modos de visualização avançados.

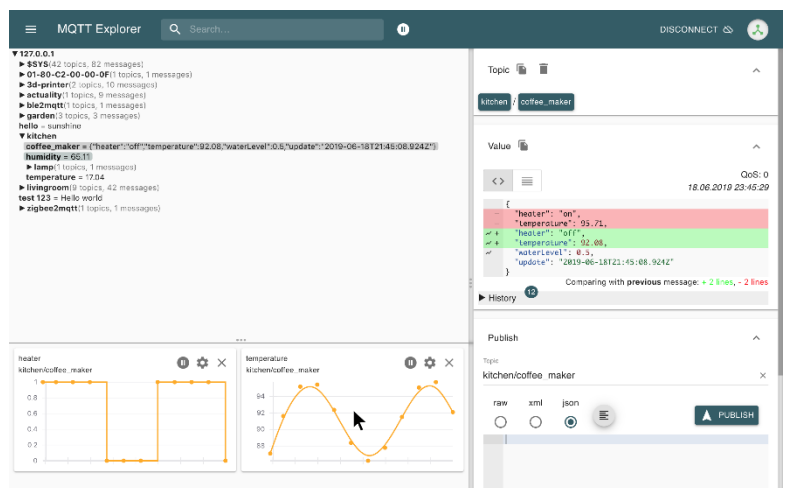


Imagem 11 - MQTT Explorer

Atualmente, o *MQTT Explorer* está disponível para os sistemas operacionais *Windows, Mac* e *Linux*.

Capítulo III – O Projeto

Projetos que nos inspiraram

Para sentir-nos inspirados e pensarmos na estrutura do nosso projeto, pesquisamos alguns trabalhos já existentes na internet e encontramos 3 projetos similares a nossa ideia. Componentes de *arduino* que detetam valores e enviam para o *Software* do computador, com algumas ideias diferentes e alguns ajustes que o nosso projeto irá ter.

Ideia 1

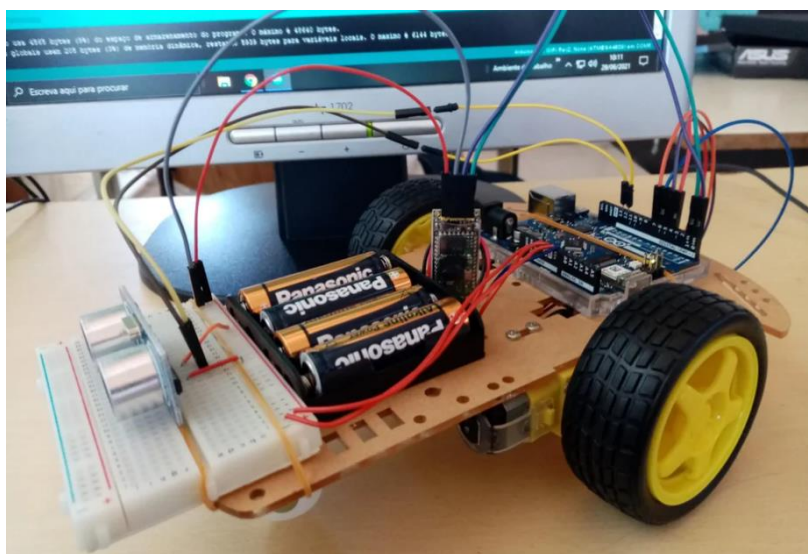


Imagem 12 - Carro ideia 1

Pequeno projeto realizado por um estudante do 12º ano do curso técnico de *GPSI* (Gestão e Programação de Sistemas Informáticos), que o estudante fez um veículo telecomandado com a adição de um maquinismo que faz parar, quando o veículo se encontra em risco de impacto frontal. Este projeto tem *Arduino*, *Bluetooth*, *breadboard*, *chassis* robótico, *Driver Motores*, Suporte de pilhas 9V, cabos *jumper* e muito mais.

Este projeto irá ter uma arquitetura parecida a nossa, usar peças parecidas a nossa e também tem usa a mesma linguagem que a nossa (C).

Ideia 2

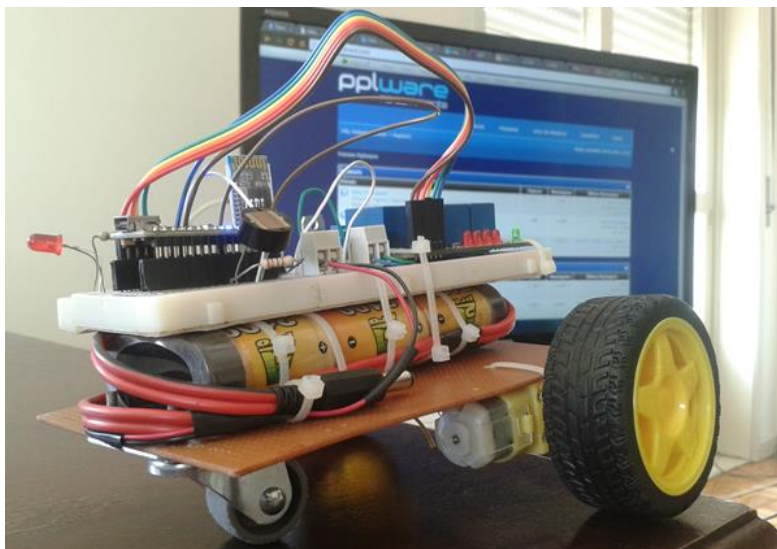


Imagem 13 - Ideia carro 2

O outro projeto que nos inspirou foi este do site *pplware*, que faz análise da temperatura e essa leitura é enviada para uma aplicação do telemóvel. Essa aplicação também tem uma funcionalidade adicional de controlar o carro.

As características deste veículo são *Arduino*, Base de 4 relés, *Breadboard*, Bateria 7.2V-2100mA, Roda livre, *LDR*, entre outros.

Ideia 3

O outro e último projeto que nos inspirou foi o robô da *NASA*. Apesar de o nosso projeto ser um “pouco mais amador” comparado ao da *NASA*, achamos interessante apresentar aqui algumas ideias desde linguagens, peças, programas, desafios, etc. Ambos os veículos, deslocam-se autonomamente ou manualmente e ambos recolhem dados meteorológico para ser analisado.

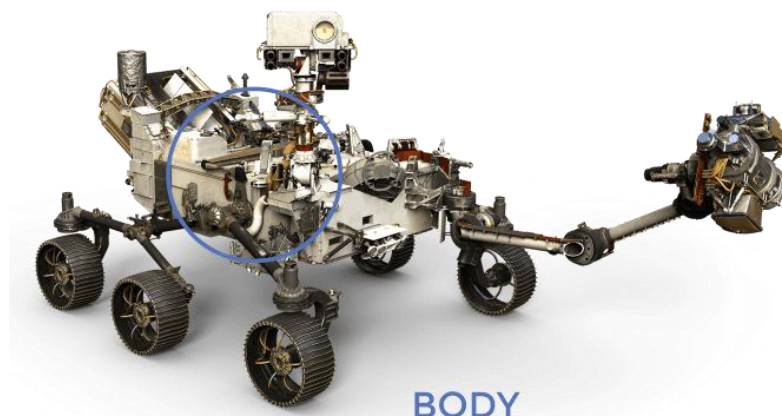


Imagem 14 - Carro ideia 3

Apesar de a *NASA*, ser uma empresa grande e ter muito dinheiro, achamos sensato analisar detalhadamente os seus documentos e desafios, pois os “problemas” passados deles, podem ou poderão ser os nossos problemas de hoje, para ter sucesso na entrega deste projeto.

A *NASA* usa no seu robô *Python* para cálculo e simulação e juntamente tem *Assembly* incorporado para fácil controlo no robô evitando *glitches* e/ou erros.

Ideia do nosso projeto

A ideia do grupo é fazer um carro com uma boa estrutura para andar em todo o terreno e conseguir passar vários obstáculos e vários terrenos. Desenhamos o carro e estruturamos o veículo no *blender* e este foi o seu resultado final:

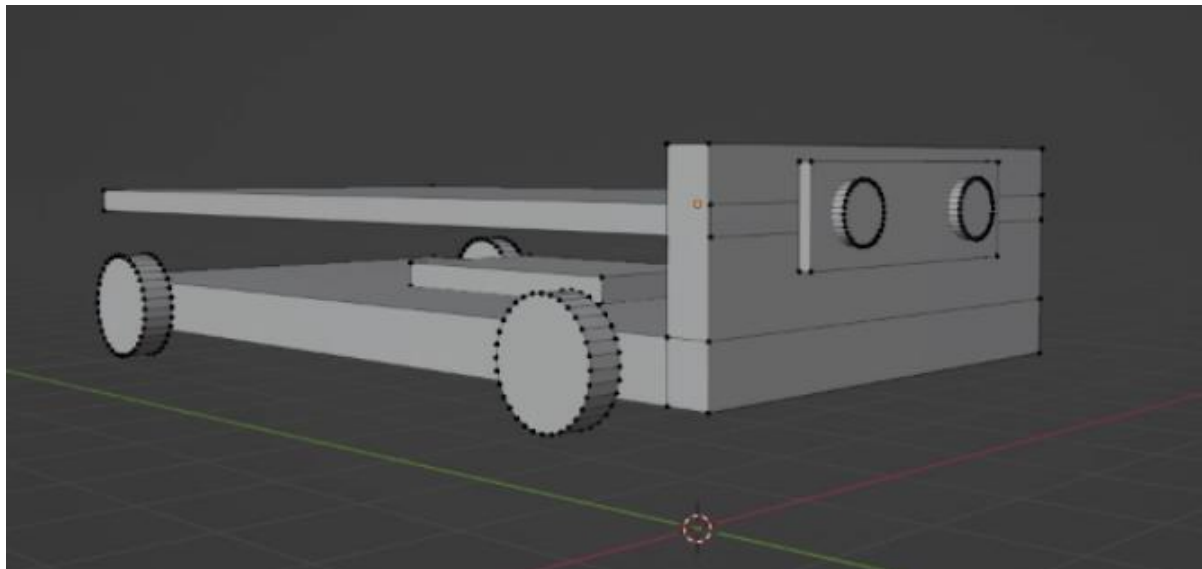


Imagem 15 - esboço do veículo 3D em *blender*

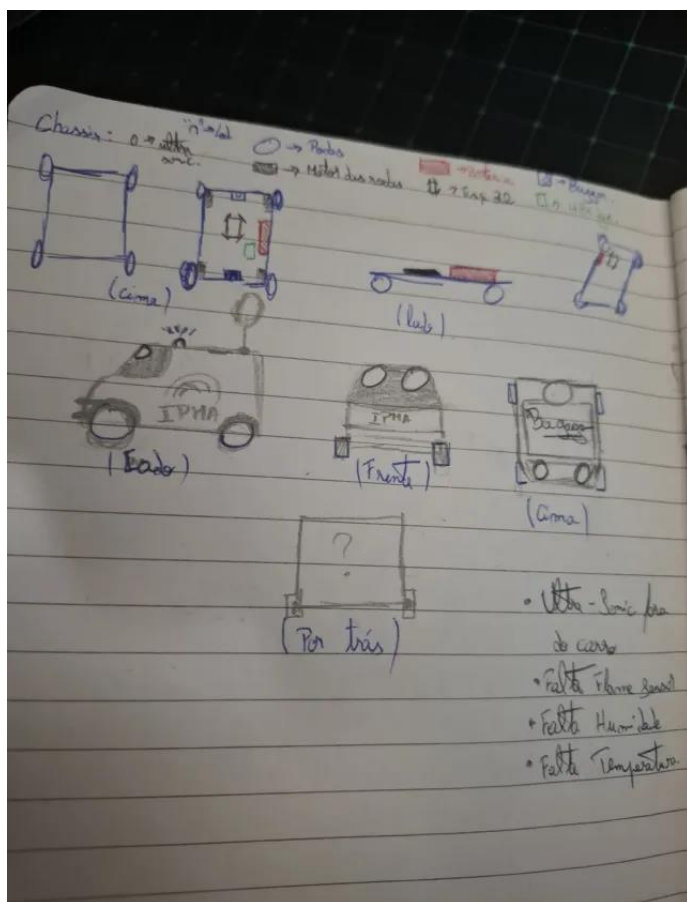


Imagem 16 - esboços em papel

Nas duas imagens dá para ver o nosso veículo idealizado, com os vários componentes e os vários ângulos.

Temos ideia de meter um *led* em cima, um *ultrasonic* em frente para detetar obstáculos, na parte de trás o *ESP32*, num dos lados a bateria. Num dos lados ter sensor de temperatura e sensor de humidade.

Como é obvio, está imagem não é o resultado final do carro pretendido, mas é uma pequena ideia do que poderá aparecer no nosso projeto final. Irá depender do orçamento, dificuldades que encontremos ao desenvolver o projeto *PBL* e ideia do grupo de *designers*.

Arquitetura do Programa

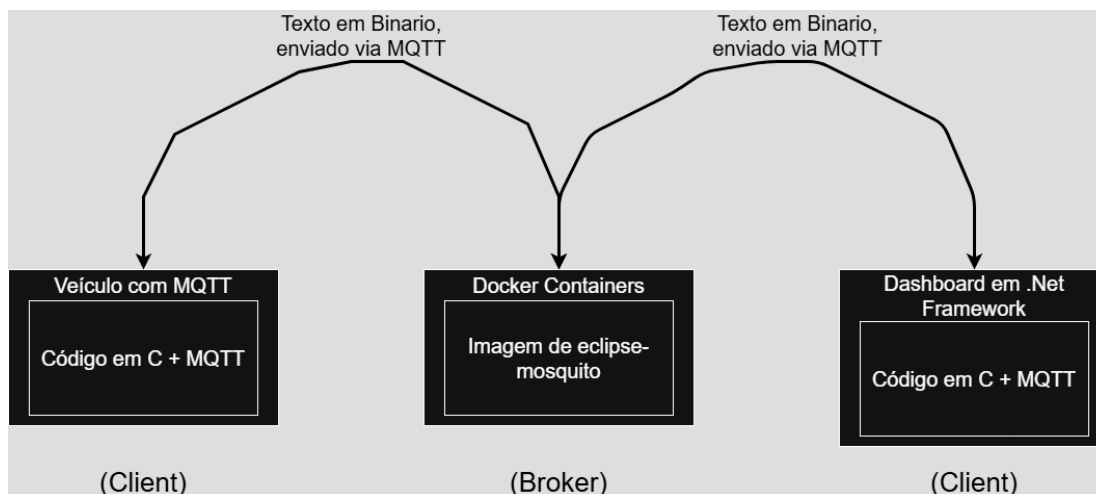


Imagem 17 - Arquitetura do projeto S.T.A.R

A arquitetura utilizada no nosso projeto, esta dividida em dois componentes:

- ❖ *Broker*
- ❖ *Cliente*

Na componente de *broker* é usado para comunicar com o cliente do *C#* e do cliente do *ESP32*. O broker está a usar uma imagem do “*eclipse-mosquitto*”, em *Docker-Container*.

Tanto o veículo como a *dashboard* comunicação via *MQTT* via wireless na internet local e os dados são enviados em binário. Devido a estarmos a falar de um hardware mais fraco, relativamente ao veículo, teríamos de então optar com esse formato.

O código do *ESP32* é enviado via flash programado em *C* no *Arduino IDE*, enquanto o cliente *c#* é feito no *IDE Visual Studio 2022* com a Framework *.Net Framework*!

Imagens antigas do veículo



Imagem 18 – Imagens antigas do Veículo do PBL

Imagens atuais do veículo

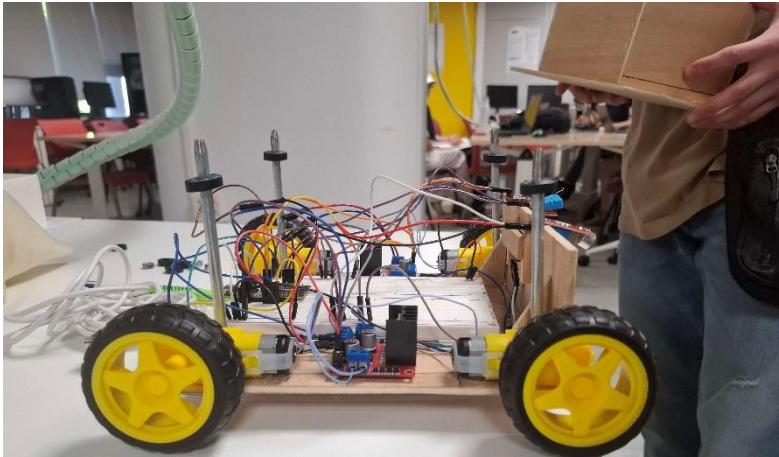


Imagem 19 - Imagem atual do veículo visto de lado, sem caixa

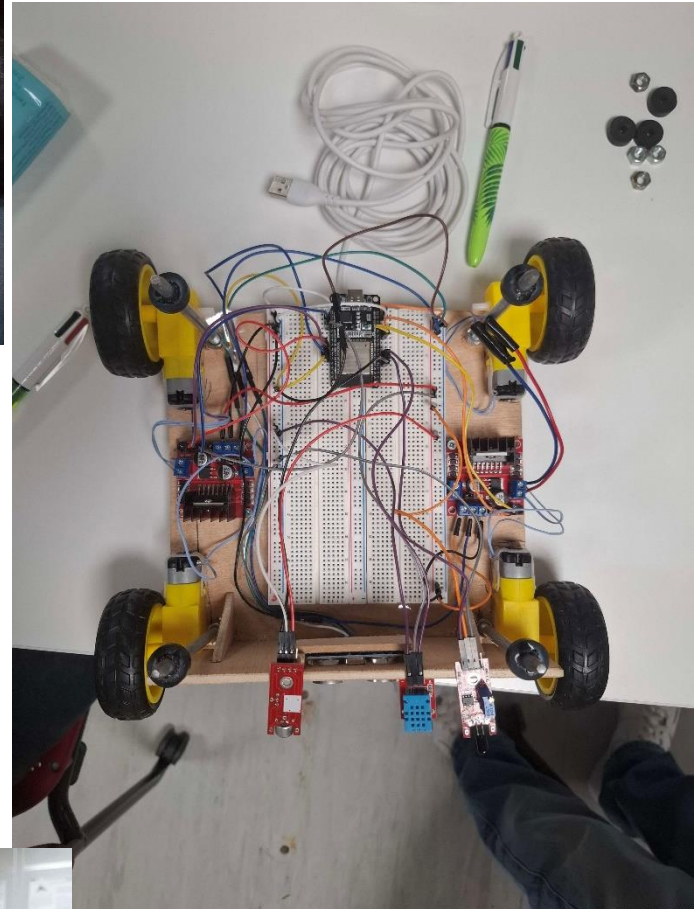


Imagem 20 - Imagem atual do veículo vista de cima, sem caixa

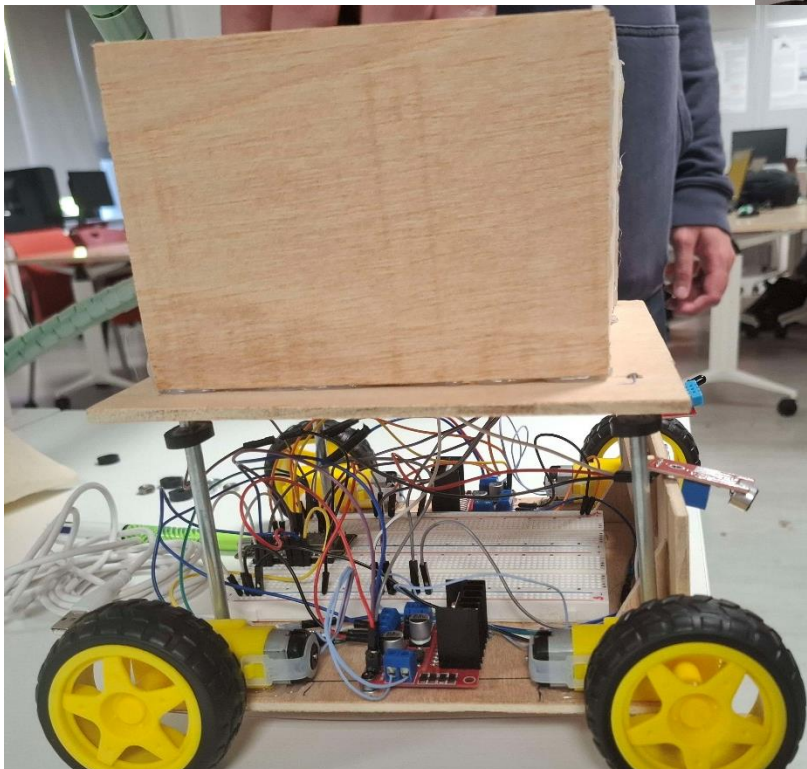


Imagem 21 - Imagem atual do veículo vista de lado com caixa



Imagem 22 - Imagem atual do veículo visto de cima, com caixa e um cubo.

Atualmente o veículo está estruturado para aceitar a nossa parte de cima para transportar os objetos, mas também para receber a parte de cima dos designers. Removendo só as borrachas.



Imagem 23 - Sistema de suporte do chassi e parte de cima do chassi

Na imagem acima, isso é bem explicado. A parte do lado direito fica agarrado ao nosso chassi enquanto a parte do lado esquerdo é onde ficará a placa que permite transportar os objetos definidos no *briefing*.

Estas borrachas do lado esquerdo são totalmente fáceis de remover, permitindo que o nosso veículo tenha suporte a alterar como os objetos são transportados.

Diagrama de Circuitos Necessários

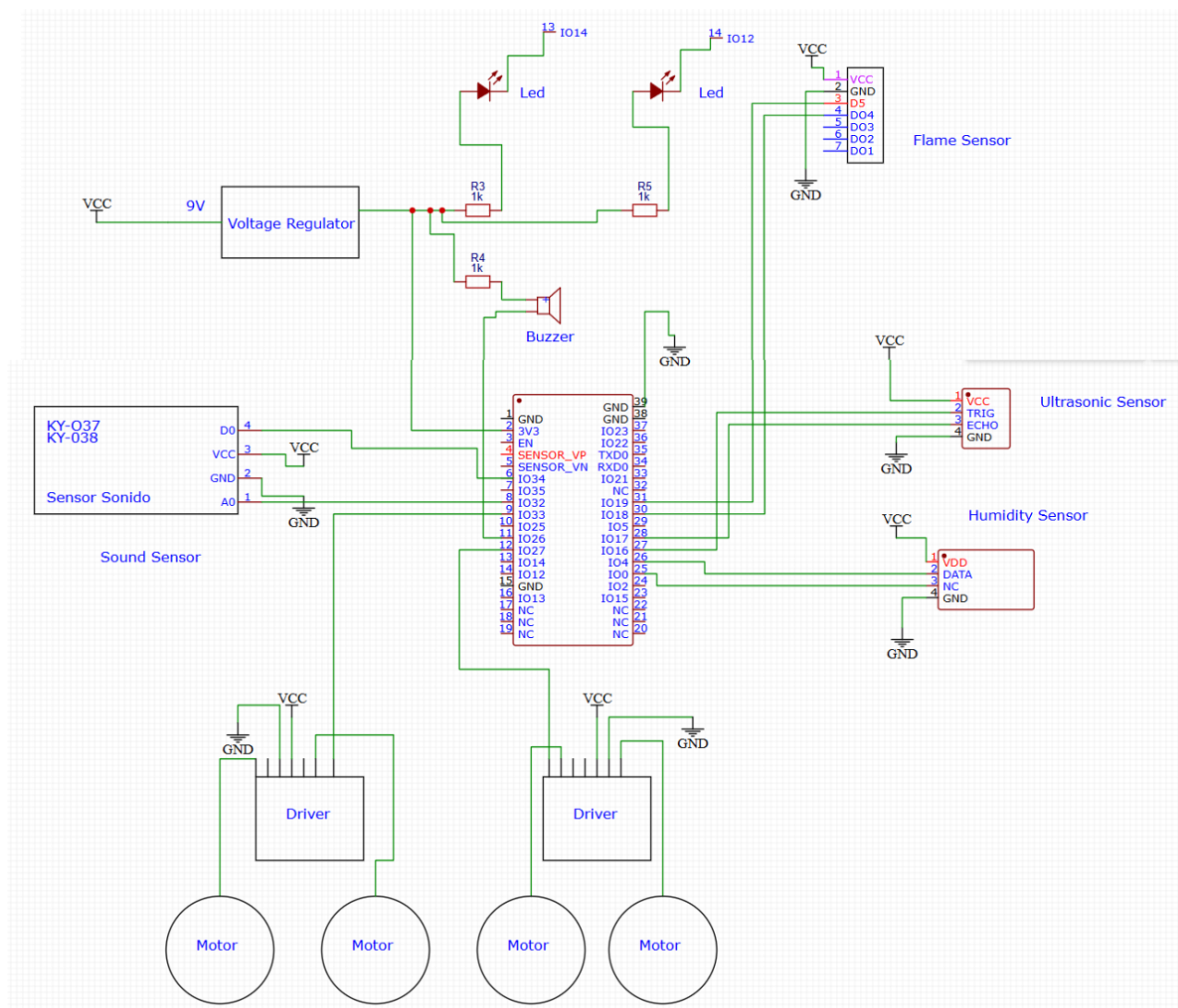


Imagem 24 - Diagrama de Circuito do Veículo

Como mostrado acima o circuito do veículo é complexo. O mesmo contém um *ESP32* no centro, que está ligado ao *Sound Sensor* na *GPIOs 8* está ligado o *Sound Sensor*, no *GPIOs 25 e 26* está ligado o *Humidity Sensor*. O *UltraSonic* está ligado ao *GPIOs 27 e 28*, *Flame Sensor* na *GPIOs 30 e 31*. Ambos os componentes *IOT* têm de ter um *Power Ground (GND)* e *Power Input (VCC)*. Para ser considerado um veículo é necessário ter rodas, então iremos ter duas drivers que estão ligados cada um a 2 rodas.

Dashboard

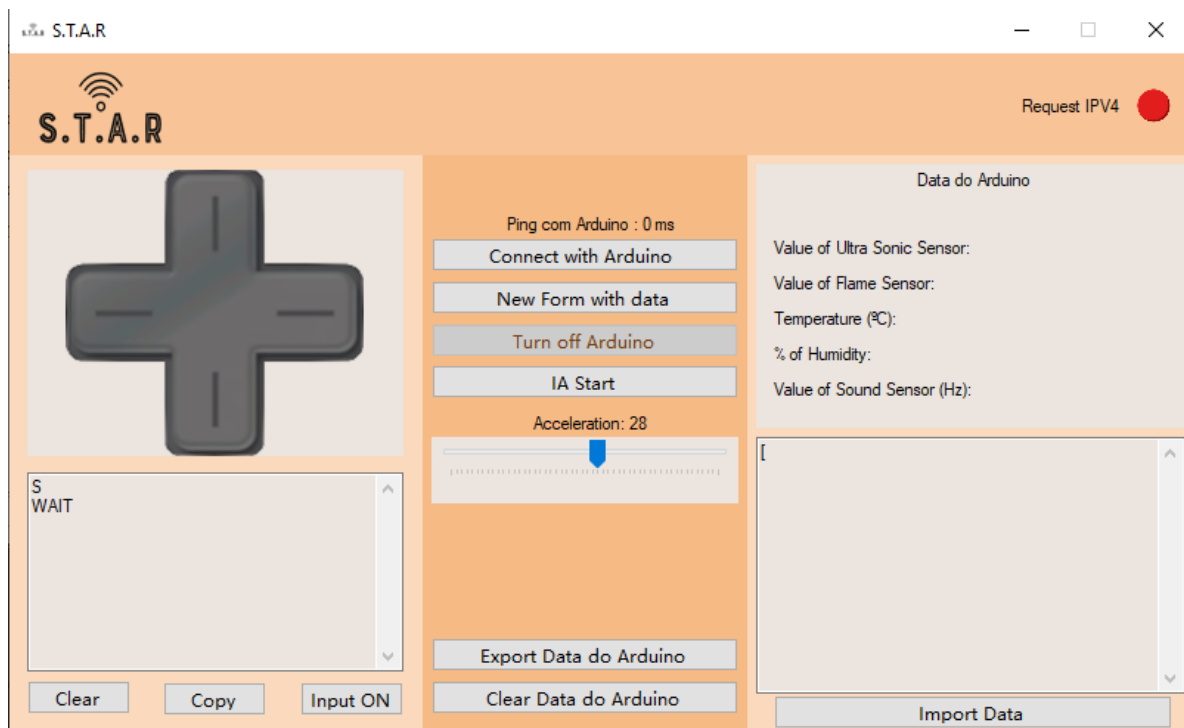


Imagem 25 - Dashboard do PBL

Como dito acima iremos ter um projeto em *.Net Framework* em C#. Nesta *dashboard* irá ter um botão para ligar-se ao *arduino* (*Connect with Arduino*), botão para ver os dados ao longo do tempo num *Form* (*New Form with data*), botão para desligar-se ao *arduino* (*Turn off Arduino*) que somente irá estar disponível quando estiver ligado ao *Arduino*, um botão de condução autónoma (*IA Start*), botão para exportar os dados do *arduino* (*Export Data do Arduino*), apagar os dados do *arduino* (*Clear Data do Arduino*), um botão de *Clear* que irá apagar os últimos *inputs* do cliente, botão de *Copy* que irá copiar os *inputs* apresentados na *TextBox* para o *Clipboard* e *Input OFF* para não permitir ao programa detectar os *inputs*. Caso o utilizador quiser importar os dados basta só clicar *Import Data* e seleccionar que os dados serão metidos na aplicação. Estes dados, são exportados para um ficheiro **.json* ou **.xlsx* ou **.json* ou **.dat*.

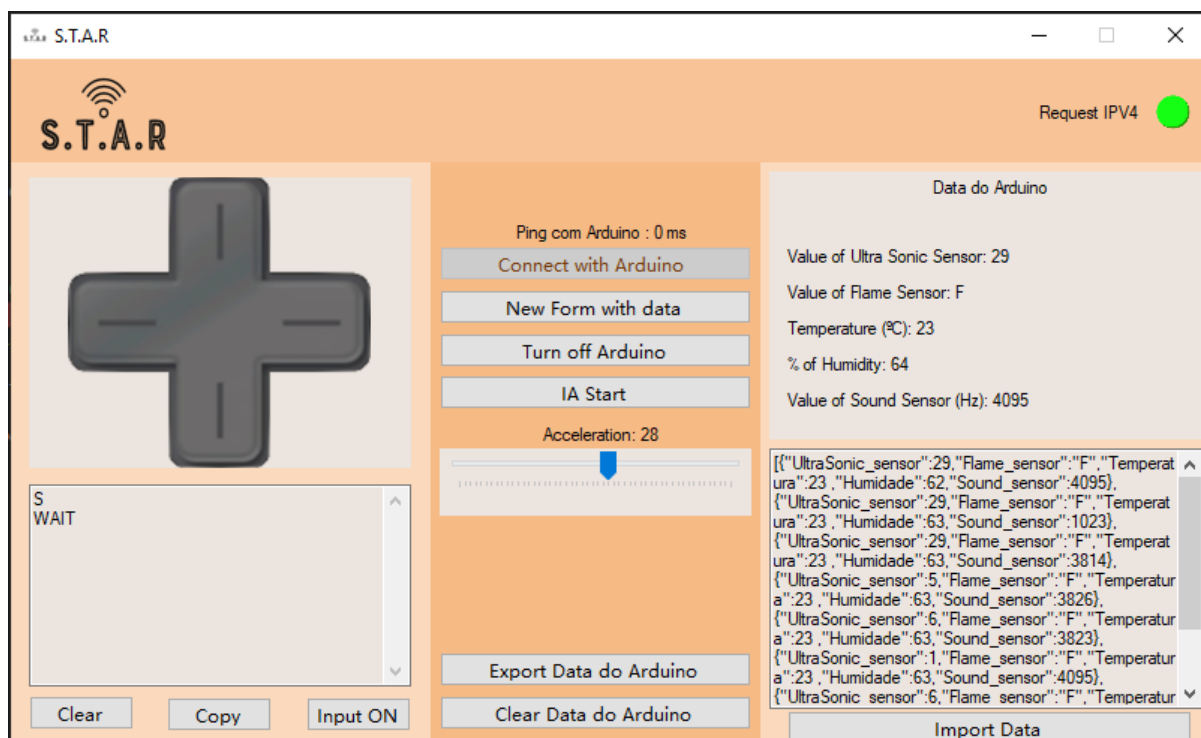


Imagem 26 - Dashboard com alguns dados vindos do ESP32

Após o cliente fazer uma ligação com sucesso ao *ESP32* clicando no botão “*Connect with Arduino*” e meter o *IpAdress* do *Broker* e tiver sucesso com a conexão do *ESP32* a bolinha vermelha irá passar a verde a demonstrar que a comunicação foi efectuada com sucesso.

Caso assim seja, o cliente poderá telecomandar o veículo e receber os dados dos componentes *IOT* em tempo real. Para depois desligar-se do *Arduino* basta só clicar no botão “*Turn off Arduino*”.

Planeamento do Projeto

- Estudo do *C#*
- Estudo do *C*
- Criação da Aplicação:
 - Implementação do Design da Aplicação
 - Ligação ao Veículo via cabo ou wireless

Distribuição de Tarefas

A tarefa de trabalhar no *C* é o Steve Vilas, o André Mendes fica responsável pela parte do *arduíno* perceber como os componentes e o circuito funciona e o André Custódio fica responsável pela execução do código em *C#*. Ambos os elementos ficam responsáveis pelo relatório, Planeamento semanal e ajuste no desenvolvimento da estrutura do veículo!

Protobuf e *MQTT* foi executado pelo André Custódio

Relatório de Ética, soldagem de componentes, apoio moral, contribuição na codificação do projeto atual ficou responsável pelo Steve Vilas.

Codificação *MQTT* e construção do veículo foi efectuado pelo André Mendes.

Proposta de plataformas padrão, funcionalidades a implementar

Ainda tem de ser efectuado a comunicação *MQTT* entre o *arduíno* e o código *c#*, fazer o carro mover-se, criar a *IA*, fazer uma *dashboard* atrativa, corrigir *bugs*, importar dados de um ficheiro para a *dashboard*, veículo ligado e funcional!

Análise Ética do PBL

A Deontologia de *Immanuel Kant* assenta no pressuposto de que "é impossível pensar em qualquer coisa no mundo, ou mesmo além dele, que possa ser considerada boa ilimitadamente, exceto uma boa vontade." Em relação ao nosso projeto a Deontologia de Kant diz que S.T.A.R. é eticamente correto pois a nossa intenção é o benefício de meteorologistas.

Segundo o consequencialismo, o nosso projeto é eticamente correto pois, como foi referido anteriormente, o nosso fim é sempre o benefício dos meteorologistas.

Há sempre pessoas que estão a utilizar esta tecnologia para espiar pessoas, mas o nosso objetivo é e será sempre a análise de dados atmosféricos e nunca a invasão da privacidade das pessoas. E com isto o ratio de pessoas que são prejudicadas para as pessoas que não são está claramente inclinado para o bem da população, pois o dado fornecido pelo nosso veículo ajuda muito mais a sociedade do que o pouco da população que está a utilizar esta tecnologia para fins errados. Uma solução que encontramos para este problema é vender o nosso produto a companhias confiáveis e conhecidas para que estes "leaks" deixem de existir. Outra solução para este problema também seria caso um dos nossos veículos sejam roubados de uma das companhias que nós oferecemos parceria fosse bloqueado o acesso aos próprios sensores e transmissão de dados. Segundo as Virtudes: Nós não conseguimos aplicar virtudes a um veículo, mas a intenção de ajudar pessoas em perigo e outras situações benéficas à polícia e aos meteorologistas é uma ação altruísta que beneficia várias pessoas.


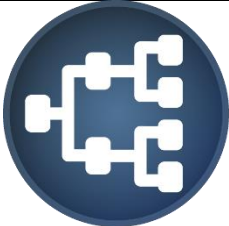


Imagem 27 - Imagem de Logo

Recursos Utilizados

 <p>Imagem 28 - Visual Studio Code</p>	<p>Visual Code– Ambiente de desenvolvimento da <i>Microsoft</i> para o desenvolvimento de software. Foi utilizado para desenvolver uma parte do projeto usando a linguagem <i>Javascript</i> no ambiente <i>nodejs</i>.</p>
 <p>Imagem 29 - Visual Studio 2022</p>	<p>Visual Studio 2022 – Ambiente de desenvolvimento da <i>Microsoft</i> para o desenvolvimento de software. Foi utilizado para desenvolver uma parte do projeto usando a linguagem <i>c#</i> com a <i>.Net Framework</i></p>
 <p>Imagem 30 - Arduino IDE</p>	<p>Arduino IDE – Ambiente de desenvolvimento da <i>Arduino Software</i>, para o desenvolvimento e configuração do nosso veículo. Foi utilizado para desenvolver uma parte do nosso projeto usando a linguagem <i>c</i>.</p>
 <p>Imagem 31 - Microsoft Word</p>	<p>Word – Foi utilizado para realizar este relatório e relatório da cadeira de <i>Sistemas Operativos</i>.</p>

 <p><i>Imagem 32 - Microsoft Excel</i></p>	<p>Excel – Foi utilizado para fazer o TODO list e usar o registo das tarefas semanais.</p>
 <p><i>Imagem 33 - Microsoft Power Point</i></p>	<p>Power Point – Utilizado para conceção das apresentações Referentes ao PBL.</p>
 <p><i>Imagem 34 - Brave</i></p>	<p>Brave – browser que permitiu navegar na internet que utilizamos para pesquisar informação e esclarecer dúvidas.</p>
 <p><i>Imagem 35 - Discord</i></p>	<p>Discord - Foi utilizado para comunicar com os colegas de projeto e comunicar com os docentes das cadeiras envolvidos no PBL.</p>
 <p><i>Imagem 36 - GitHub</i></p>	<p>GitHub – Utilizado para que o projeto possa ser acedido por qualquer programador que tenha acesso ao repositório para que possa consultar ou contribuir no mesmo.</p>

 <p>Imagem 37 - Docker</p>	<p>Docker – Aplicação para criar um container e conseguir que haja uma imagem do eclipse mosquito para comunicar entre dispositivos via MQTT.</p>
 <p>Imagem 38 - MQTT Explorer</p>	<p>MQTT Explorer – Aplicação para ver a gestão de tópicos MQTT em tempo real.</p>

Ferramentas de Desenvolvimento:

- *Visual Studio Code;*
- *Visual Studio 2022;*
- *Arduino IDE;*
- *MQTT Explorer*
- *Docker*

Browser:

- *Brave;*
- *Opera;*
- *Google Chrome*

Ferramentas para desenvolvimento de apresentação e relatório:

- *Microsoft Office Power Point 365*
- *Microsoft Office Word 365;*
- *Microsoft Office Excel 365;*
- *Office Timeline;*
- *Draw.io;*

Aplicação/Site de Comunicação:

- *Discord;*
- *Whatsapp;*
- *Git;*
- *GitHub;*
- *Gmail;*

Capítulo IV – Cronograma Final e Justificação de desvios

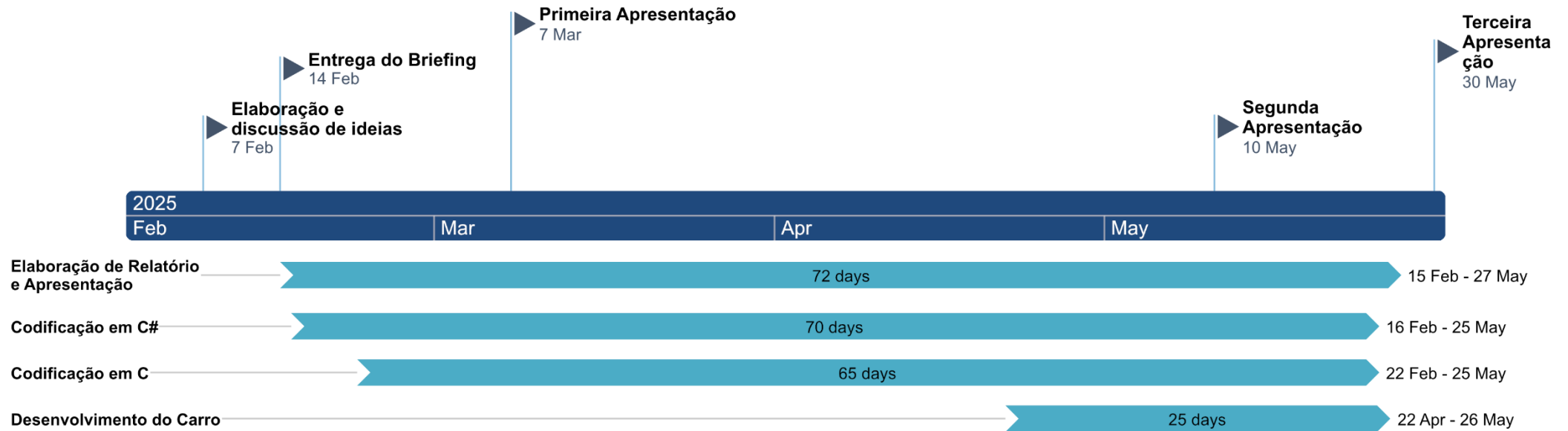


Imagem 39 - Cronograma Final

Justificação

O Cronograma final teve desvios.

Na primeira semana tivemos a espera do *briefing* do projeto, entretanto tivemos a planear e discutir acerca do projeto, no desenvolvimento do veículo com tecnologias como *C#* e *C*.

A 15 de fevereiro começamos o desenvolvimento do relatório e acabamos a 27 de maio e a apresentação foi começada a 7 de dezembro e finalizada a 7 de julho.

No dia 16 de fevereiro começamos o desenvolvimento do projeto e iria ser desenvolvimento em *.Net* na *dashboard* e o veículo em *C*.

O desenvolvimento da estrutura do veículo começou no dia 22 de abril até dia 26 de maio.

Capítulo V – Análise do percurso pessoal

Neste capítulo vamos falar sobre o nosso percurso pessoal durante o período em que realizamos o *PBL*. Esta foi a primeira vez que tivemos um projeto em contexto real e profissional, tendo que fazer um veículo em concílio com os alunos do IADE do curso de Licenciatura em Design de segundo ano. Para nós, foi um trabalho em contexto real, muito complexo e difícil de conciliar com os horários e tentar ter um projeto bom em conciliação com o grupo de Designs. Apesar de não termos atingido essa meta, foi bom o nosso trabalho de equipa. Sempre que um elemento do grupo tivesse em apuros ou atrapalhado, apoiávamos uns aos outros.

A linguagem que foi usada para programar a aplicação, foi a linguagem *C#* em conjunto com o *C*, ao início tivemos algumas dificuldades em perceber como programava e conciliar a parte de *IOT* e a *dashboard*. No entanto, com alguma pratica, conseguimos entender como ligar as coisas e podendo trabalhar com sucesso. Esta experiência superou as vossas expetativas, pois aprendemos como soldar os componentes *IOT*, novos métodos de comunicação, existe muitos componentes *IOT*.

Gostamos muito de realizar este projeto, pois foi a primeira vez que realizamos um projeto tão grande e complexo para um possível cliente. Com esta análise concluímos o nosso último relatório de *PBL*, agradecendo ao professor Nathan Campos por ajudar e disponibilizar o seu tempo. Agradecemos também ao professor André Sabino por ser uma grande ajuda neste projeto e por proporcionar grandes desafios neste projeto!

Conclusão

Após terminarmos este relatório podemos concluir que o desenvolvimento *Project Base Learning* correu muito bem e que a experiência que tiramos desta formação vai ser muito enriquecedora tanto no nosso percurso escolar, como profissional. É claro que cometemos muitos erros e isso faz parte do desenvolvimento de um profissional. É com os erros que aprendemos e nos tornamos melhores pessoas e melhores profissionais. Adquirimos muitos conhecimentos e métodos de trabalho que nunca iremos esquecer. Este desenvolvimento de contexto real, permitiu-nos adquirir experiência no desenvolvimento *IOT* e desenvolvimento de uma *dashboard*. Sentimos que conseguimos atingir e até superar os objetivos do desenvolvimento *PBL* e esperamos que este relatório também demonstre isso.

Bibliografia e Web Grafia

Projeto 1:

- <https://www.instructables.com/Carro-Telecomandado-Arduino/>

Projeto 2:

- <https://pplware.sapo.pt/tutoriais/arduino-robot-controlado-por-movimentos-do-telemovel/>

Projeto 3:

- <https://www.linkedin.com/pulse/what-programming-languages-does-nasa-use-analytics-insight-p5nrc>

- <https://science.nasa.gov/mission/mars-2020-perseverance/rover-components/>