



**RELATÓRIO DO PROJETO DE
PROJECT FACTORY**

LICENCIATURA EM ENGENHARIA INFORMÁTICA

Professor Orientador: André Sabino

André Custódio, 20220112

André Mendes, 20220355

Steve Vilas, 20200856

15/02/2024

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.

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.

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
MQTT.....	11
Código inicial de MQTT	11
Peças para o Arduino	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
Bibliotecas instaladas (DLLs).....	16
Capítulo III – O Projeto	17
Projetos que nos inspiraram	17
Ideia 1.....	17
Ideia 2.....	18
Ideia do nosso projeto	19
Arquitetura do Programa	20
Diagrama de Circuitos Necessários.....	21

Planeamento do Projeto	22
Distribuição de Tarefas	22
Proposta de plataformas padrão, funcionalidades a implementar	22
Recursos Utilizados	23
Ferramentas de Desenvolvimento:.....	25
Ferramentas para desenvolvimento de apresentação e relatório:	25
Aplicação/Site de Comunicação:.....	25
Capítulo IV – Cronograma Final e Justificação de desvios	26
Justificação	27
Capítulo V – Análise do percurso pessoal	28
Conclusão	29
Bibliografia e Web Grafia	30

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 - Carro ideia 1.....	17
Imagem 12 - Ideia carro 2	18
Imagem 13 - Carro ideia 3.....	18
Imagem 14 - Exemplo de veículo	19
Imagem 15 - Arquitetura do projeto S.T.A.R	20
Imagem 16 - Comparação de ProtoBuf e JSON	21
Imagem 17 - Diagrama de Circuito do Veículo	21
Imagem 18 - Visual Studio Code	23
Imagem 19 - <i>Visual Studio 2022</i>	23
Imagem 20 - Arduino IDE	23
Imagem 21 - Microsoft Word.....	23
Imagem 22 - Microsoft Excel	24
Imagem 23 - Microsoft Power Point	24
Imagem 24 - Brave	24
Imagem 25 - Discord	24
Imagem 26 - GitHub	24

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

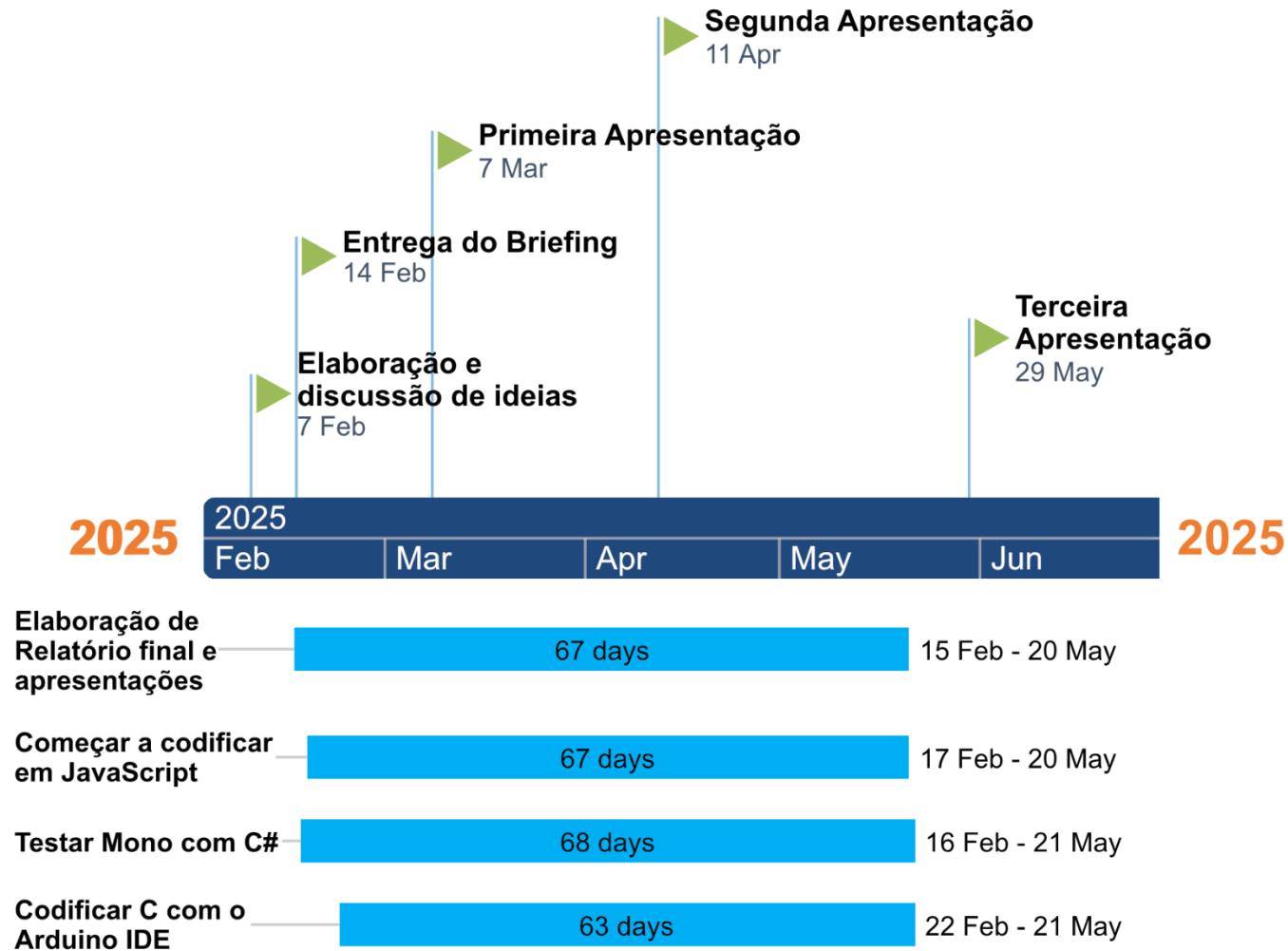


Imagem 1 - Cronograma inicial

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;
}
```

Diagrama de anotações no código:

- Uma seta vermelha aponta da expressão `(IPAddress.Parse(IPV4))` para o texto "Ip da Máquina".
- Uma seta vermelha aponta da expressão `new string[] { Path }` para o texto "Topic do MQTT".
- Uma seta vermelha aponta da linha `client.Connect(clientId);` para o texto "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:

- *Microphone Sensor;*
- *Temperature Sensor;*
- *Clock sensor;*
- *Sound Sensor;*
- *Flame Sensor;*
- *Lazer sensor;*
- *Common Cathode Led;*
- *Seven-Color Automatic Flashing LED;*
- *Passive Buzzer;*
- Suporte de Bateria;
- Converter modulo Blinghe;
- Sensor de Humidade;
- 4 peças de *Geared Motoro DC3V-12V;*
- *DC Motor Driver Board Drive;*
- *ESP32;*
- *BreadBoard;*
- *Active Buzzer;*

Ao longo do tempo está lista poderá ser modificada. Mas para já irá ser esta a nossa lista de peças ligadas ao Arduíno necessárias.

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.

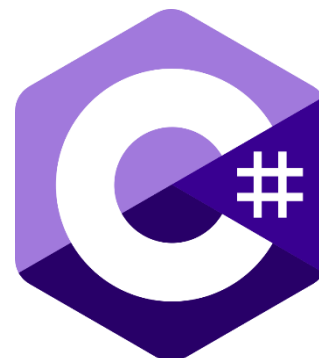


Imagem 6 - C#

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.

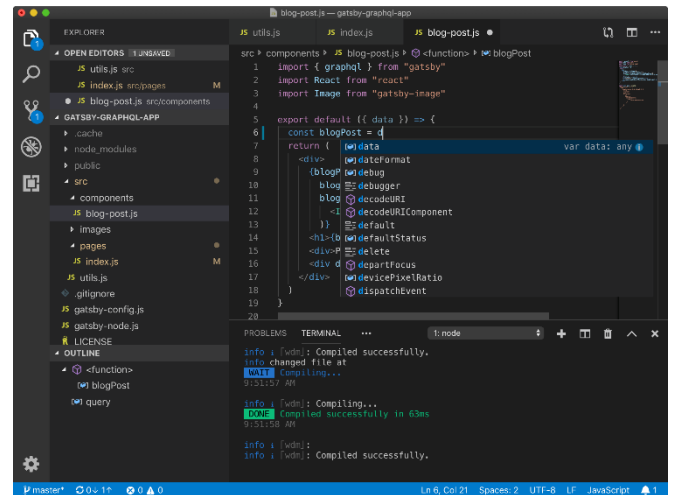


Imagem 8 - IDE Visual Studio Code

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.

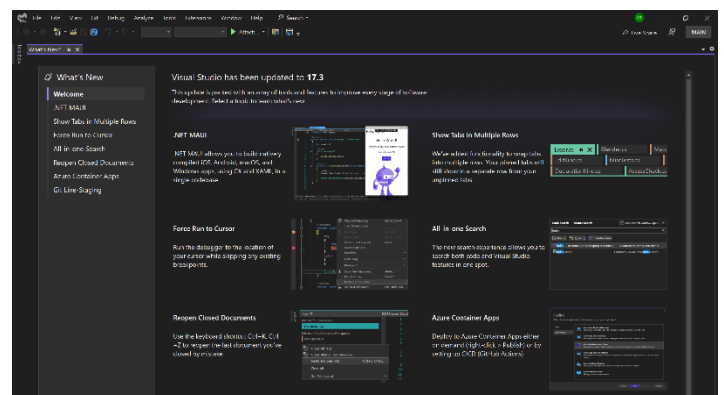


Imagem 9 - IDE Visual Studio 2022

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.

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

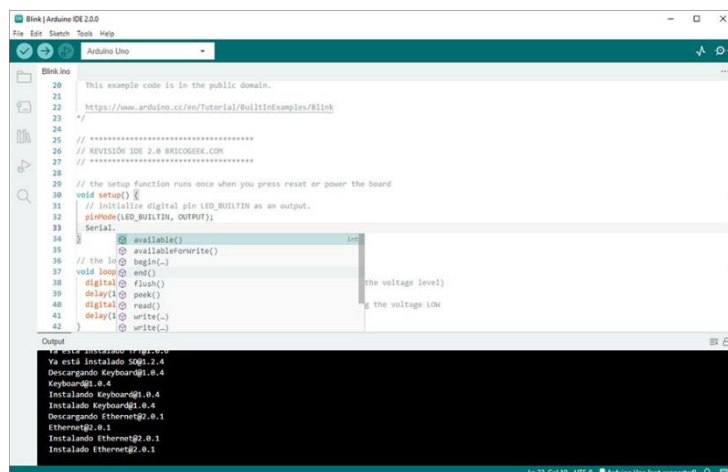


Imagem 10 - IDE Arduino IDE

Bibliotecas instaladas (DLLs)

...

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 *arduíno* 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

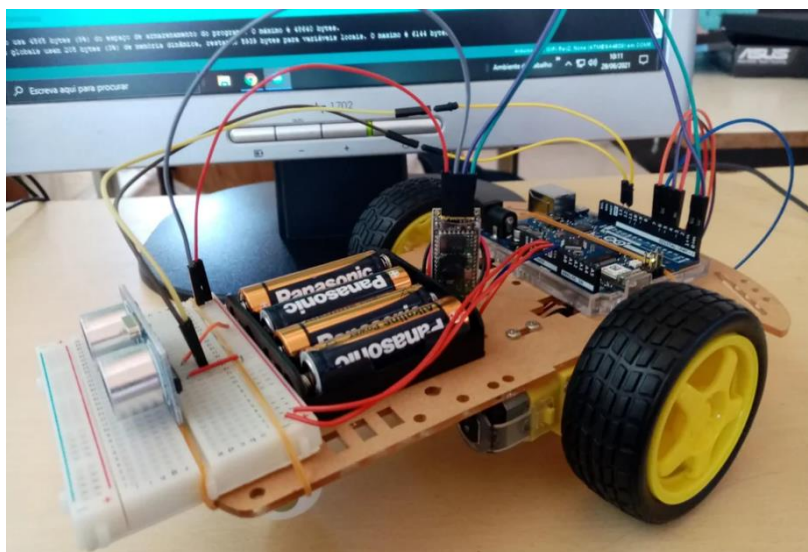


Imagem 11 - Carro ideia 1

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 *Arduíno*, *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

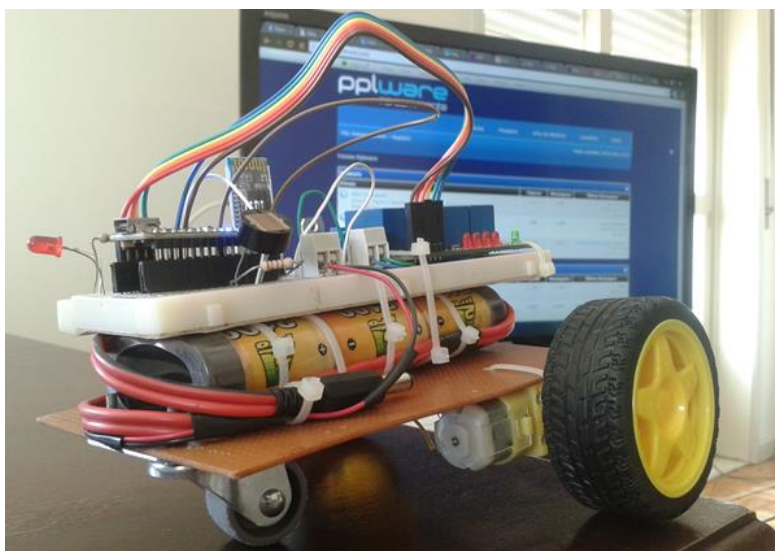


Imagem 12 - Ideia carro 2

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.

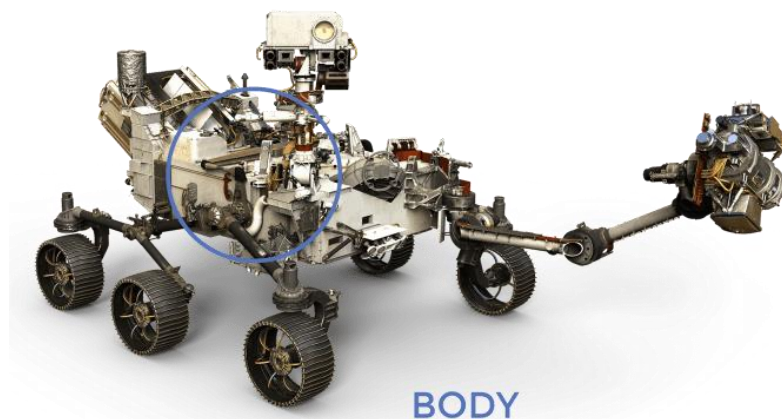


Imagem 13 - Carro ideia 3

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 como por exemplo este veículo:



Imagem 14 - Exemplo de veículo

O veículo iria ter 2 sensores de distância para calcular a distância (número 1), uma zona onde ligar o *arduino* ao computador (número 3), sensor de calcular a temperatura do espaço (número 4) e entre o número 1 e 2 iria haver uma caixa para transportar os objetos.

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

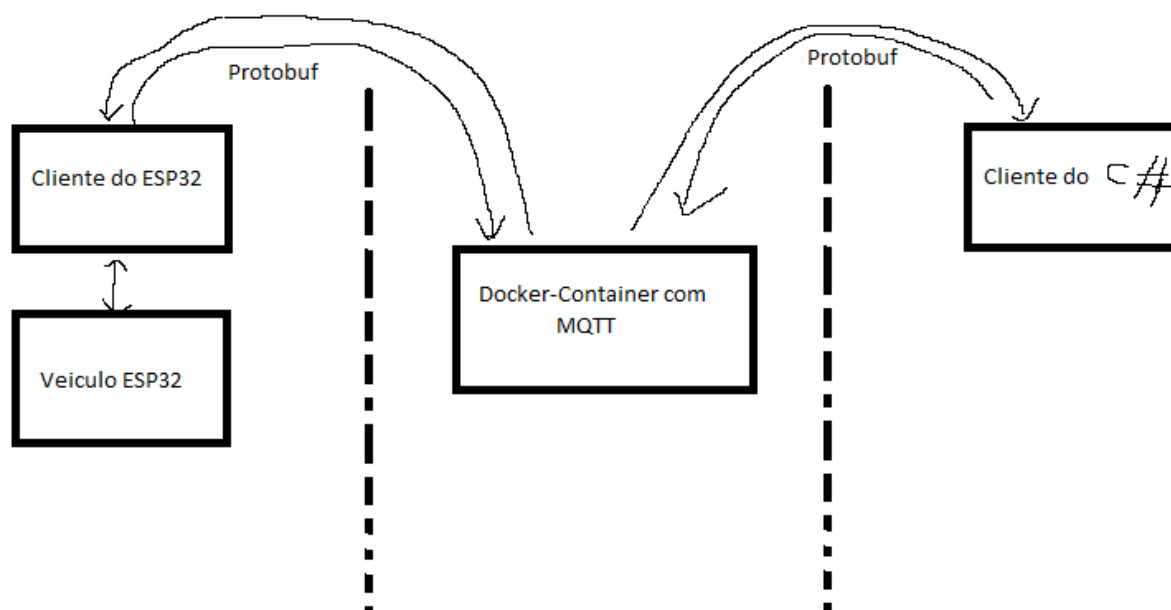


Imagem 15 - Arquitetura do projeto S.T.A.R

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

- ❖ Servidor
- ❖ Cliente

Na componente de servidor é usado uma imagem do “*eclipse-mosquitto*”, que permite criar um pequeno servidor para que haja uma comunicação entre o cliente do C# e do cliente do ESP32.

Ambos os clientes enviam dados em *Protobuf*, novo data-format muito mais eficiente e melhor que o *json*. Devido a estarmos a falar de um hardware mais fraco, relativamente ao veículo, teríamos de então optar com esse formato.

O lado do cliente do ESP32, como indicado acima irá ser um veículo que envia os dados dos componentes para a *dashboard*. Para enviar os dados de um sítio para outro usa-se o *MQTT* via *wireless* na *internet* local.

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*!

Poderão ver abaixo a latência de um JSON em comparação com um *ProtoBuf*:

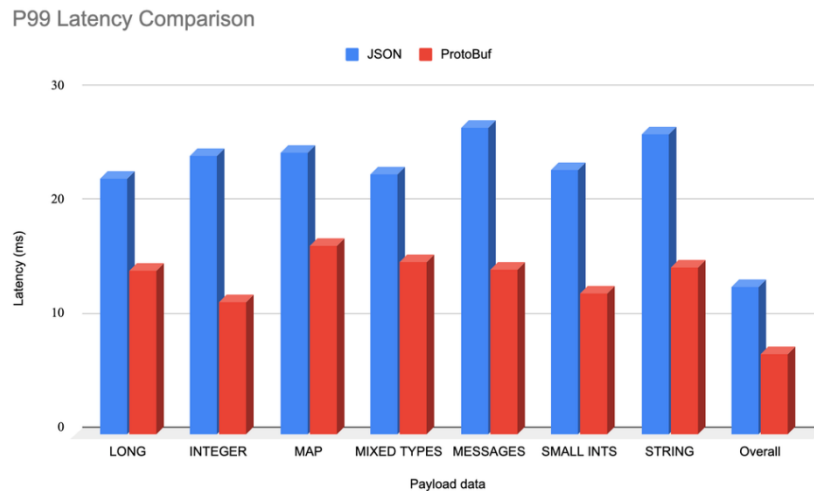


Imagem 16 - Comparação de *ProtoBuf* e *JSON*

Diagrama de Circuitos Necessários

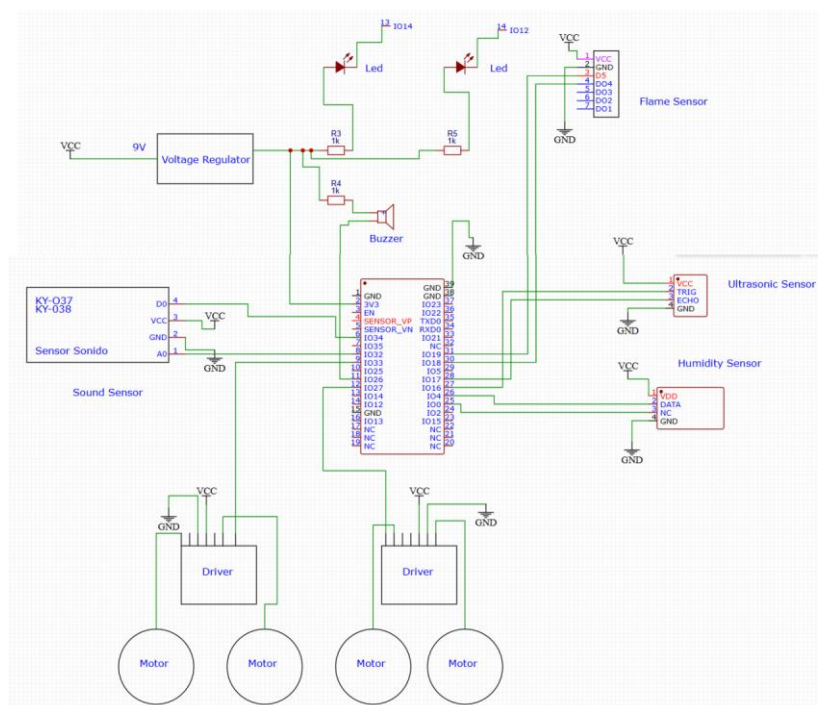


Imagem 17 - Diagrama de Circuito do Veículo

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.

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!

Proton e *MQTT* foi executado pelo André Custódio

Relatório de Ética, soldagem de componentes e apoio moral 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!

Recursos Utilizados



 <p>Imagem 18 - 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 19 - 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 20 - 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 21 - Microsoft Word</p>	<p>Word – Foi utilizado para realizar este relatório e relatório da cadeira de Sistemas Operativos.</p>



Imagem 22 - Microsoft Excel

Excel – Foi utilizado para fazer o TODO list e usar o registo das tarefas semanais.



Imagem 23 - Microsoft Power Point

Power Point – Utilizado para conceção das apresentações Referentes ao PBL.



Imagem 24 - Brave

Brave – browser que permitiu navegar na internet que utilizamos para pesquisar informação e esclarecer dúvidas.



Imagem 25 - Discord

Discord - Foi utilizado para comunicar com os colegas de projeto e comunicar com os docentes das cadeiras envolvidos no PBL.



Imagem 26 - GitHub

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.

Ferramentas de Desenvolvimento:

- *Visual Studio Code;*
- *Visual Studio 2022;*
- *Arduino IDE;*

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

Justificação

Capítulo V – Análise do percurso pessoal

Conclusão

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/>