Estruturação da Atualização do Reh@Panel

Relatório do Projeto

Eurico Diogo Rodrigues Teixeira Nº 203014
Orientado por Drº Sergi Bermúdez i Badia
Coorientado por Yuri Almeida
Integrado no NeuRehabLab

1. Resumo

Este relatório aborda todas as fases de desenvolvimento de um projeto final de Licenciatura de Engenharia Informática, que teve como principal objetivo a estruturação e desenvolvimento de uma nova versão do "Reh@Panel"(também conhecido como ControlPanel(CP)), que por sua vez tem por objetivo de funcionar como HUB que apartir do mesmo se liga módulos de devices(Kinematics, Electrophysiological, Eye Tracking, Head Tracking) .

É exposta a problemática da antiga versão do CP e a motivação da escolha para a utilização de um novo paradigma de programação.

É apresentado todo o processo, desde o estudo e aprendizagem deste novo paradigma de programação para a Unity, até o desenvolvido e implementação do mesmo.

É explicada toda a implementação, funcionalidades e implementações futura para a continuação deste projeto.

Finalmente é feita a ponderação de todo o trabalho feito até agora. Vantagens e desvantagens, dificuldades e barreiras no projeto e avaliação do cumprimento dos objetivos serão discutidos, tendo em conta o estado presente do projeto.

Índice

Índice das Ilustrações

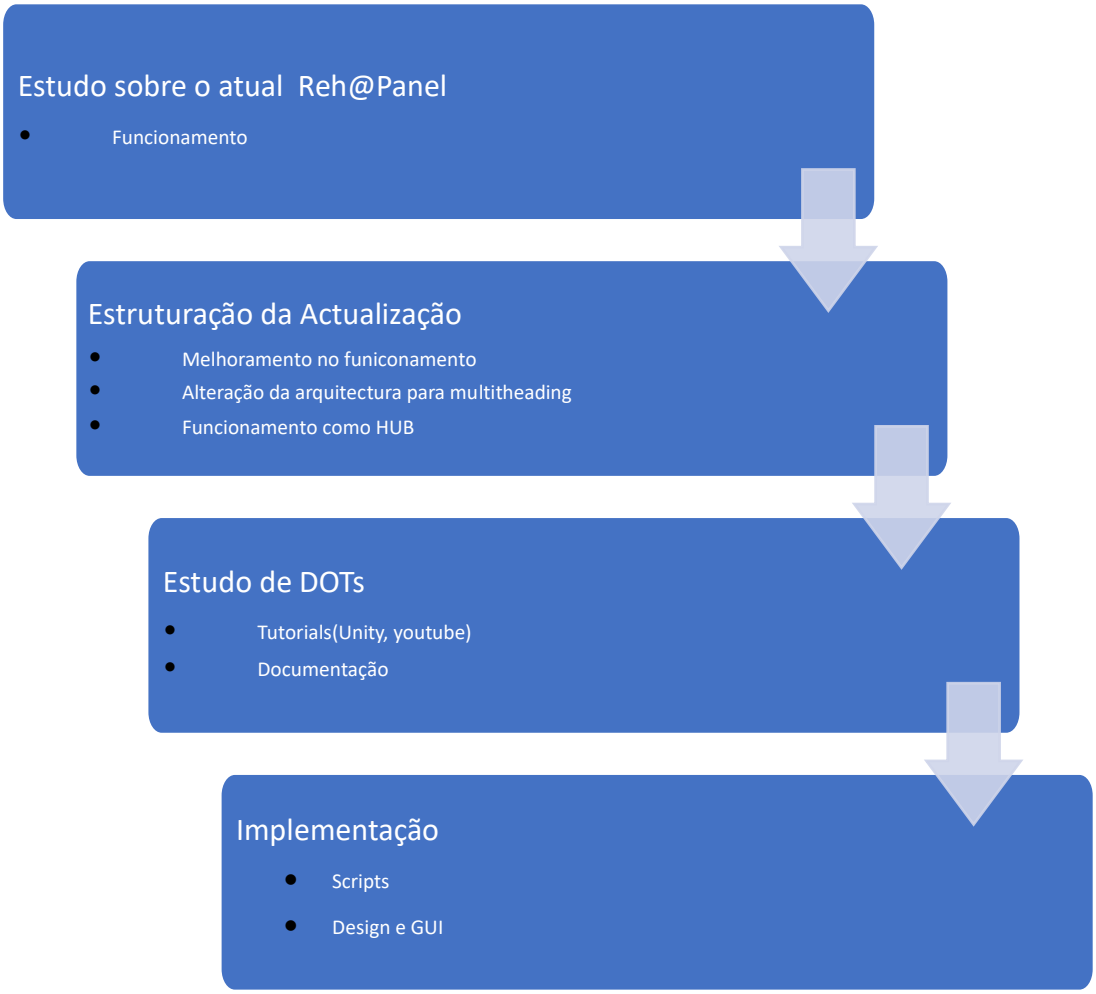
2. .Introdução

O projeto abordado neste relatório irá ser realizado no âmbito de projeto final de Licenciatura em Engenharia Informática. Foi desenvolvido pelo o discente Eurico Teixeira durante o ano letivo 2019/2020, orientado pelo o Professor Sergi Bermúdez i Badia, integrado no grupo de investigação NeuroRehablab, presente no ITI.

O projeto tem como contexto o desenvolvimento e estruturação das atualizações a serem feitas no Reh@Panel para que este se torne num software mais simples a utilizar e mais dinâmico.

Neste relatório vão ser descritas todas as fases de desenvolvimento, bem como todos os scripts e objetos principais presentes no projeto. Irá ser feita uma análise das funcionalidades relacionadas com estes. Finalmente será apresentada uma crítica ao projeto.
Todo o desenvolvimento será realizado utilizado o software principal, motor de de jogo Unity, todos os scripts escritos na linguagem de programação C#.

3. Etapas de desenvolvimento



a. Estudo sobre atual Reh@Panel

O estudo serviu como motivação e forma ideal para conhecer esta ferramenta do NeuroRehabLab que até a data não tinha qualquer conhecimento sobre ela, nem a finalidade que a mesma tinha.

Este estudo teve como principal objetivo perceber principais utilidades do CP(Reh@Panel) e que aspetos poderiam ser melhorados para que este tivesse um desempenho mais atual e adequado no âmbito que costuma ser usado.

b. Estruturação da Actualização

O projeto teve por base o software Reh@Panel. É um software que atua como um router de dispositivos, conectando um grande número de dispositivos de rastreamento e outro hardware aos RehabNet Training Games para o paciente interagir.

Durante a estruturação foi detetado os principais problemas que o atual CP(Reh@Panel) possui e que era fundamental os restruturar para um melhor desempenho quando o mesmo é utilizado, alguns destes problemas tem haver de todos os módulos do diferentes dispositivos estarem integrados no Reh@Panel e não com módulos externos, a GUI do mesmo estava toda implementada no código o que leva a utilização desnecessária do CPU porque sempre que o software é aberto a primeira parte de código a ser executada no CPU é o código da GUI.

O Reh@panel como possui todos os módulos integrados o torna num software mais pesado do que o necessário, um solução para isso é a conversão uma “hub”, que assim ele fará apenas a conexão entre o modulo do dispositivo e do RehabNet Training Games.

c. Estudo de DOTS

DOTs (Data-Oriented-Tech-Stack) é um paradigma relativamente recente(2 anos) implementado na Unity, que este paradigma teve a sua apresentação oficial durante a a Unite Copenhagen 2019, como sendo um paradigma recente este estudo foi realizado a partir da documentação e exemplos fornecidos por a própria Unity, além da documentação utilizada foi utlizado tutorials fornecidos por o Orientador ao discente.

A maior novidade de DOTs é que este é um paradigma onde a programação é orientada a dados ao invés de objetos que é mais comum na Unity, este paradigma é composto por três elementos fundamentais Job System, Entity Component System(ECS) e por Burst Compiler.

d. Implementação

Na implementação foi desenvolvido scripts com as funções do Reh@Panel utilizado DOTs, a implementação da GUI foi realizada através das ferramentas que a Unity possui no seu software para a criação dos diferente componentes que fazem parte de uma GUI(panel, button, etc).

4. Metodologia

Nesta secção será explicado resumidamente em que consiste DOTs e feita uma comparação entre os benefícios entre DOTs e OOP (sigla de programação orientada a objectos (POO)).

1. Job System

O principal objetivo do Job System é facilitar a escrita de código multi-thread, em um cenário comum, o código é executado em cascata no thread principal i.e um novo bloco é executado quando o anterior termina. Como os atuais processadores possuem multicores não faz qualquer sentido que tudo seja executado no main thread e os outros threads ficam ociosos. Outro aspeto da utilização deste sistema, é divisão do código e faz com ele seja multi-encadeado, ou seja as várias partes do nosso códigos são executadas ao mesmo tempo em diferentes threads, desta forma é-nos proporcionado uma programação multi-thread de forma muito simplificada e outro apeto fulcral é que todo o código compartilha internamente os threads do job, evitando automaticamente a criação excessiva de threads.

2. Brust Compiler

O Brust Compiler é um compilador de alto desempenho que foi desenvolvido com o principal objetivo de melhorar a eficiência do Job System, este aumento drástico no desempenho se deve ao fato de o Brust converter o código em uma versão nativa otimizada utilizado LLVM(low level virtual machine, é uma infraestrutura do compilador de escrita de linguagem C++, desenvolvida para otimizar os tempos de compilação, ligações e execução de programas escritos em linguagens de programação).

3. ECS (Entity Component System)

O ECS (Entity Component System) é constituído por três elementos, este são as Entity (entidades), Component (componentes)e por o System(sistema).

i. Entity

A entity é um identificador no mundo ECS, ela serve como um ID, ou mesmo como um apontador, a entity é considerando o GameObject mas na programa orientada a dados, este id pode ser usado para armazenar uma referencia a outras entidades ou componentes, uma entidade filha pode precisar a entidade pai em uma hierarquia. Sempre que uma entidade é criada esta recebe um ou mais componente e as componentes é que define como as entidades serão organizadas no sistema ECS, i.e o EntityManager agrupa as entidades que possuem o

mesmo componentes juntos na memória, este conjuntos de componentes é chamado de Archetype, desta forma torna a execução mais eficaz.

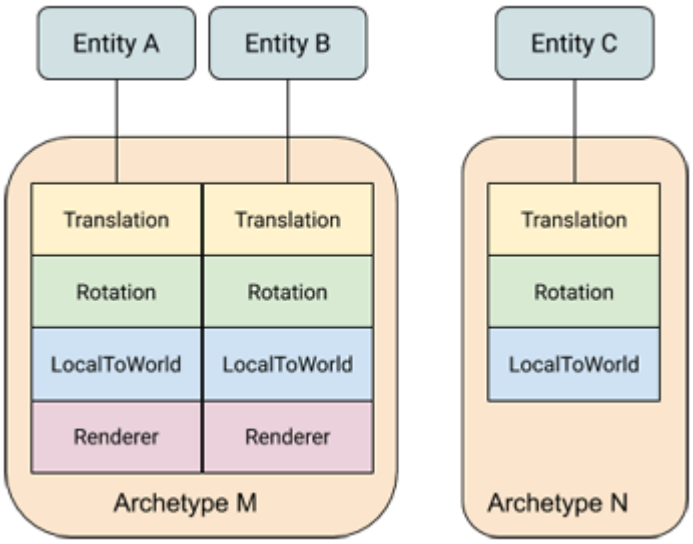


Figura 1Diagrama de Archetype

Como se pode ver no diagrama na Figura 1, as entidade A e B compartilham o archetype M, enquanto a entidade C possui o archetype N, desta forma podemos alterar fluidamente o archetype de um entity adicionando ou removendo componentes em tempo de execução.

ii. Component

Como se viu no ponto anterior(i), que as entidades têm uma função similar a um ID ou a um apontador, as components são as estruturas que representa os dados de um programa, para definir as components por norma usa-se structs para estruturar os dados.

iii. Chunks

Como já referido anteriormente, o EntityManager agrupa todas as entidades que possuem o mesmo tipo de compomentes em Archetypes, Chunks é os blocos de memória onde residem todas as entidades com o mesmo Archetype.

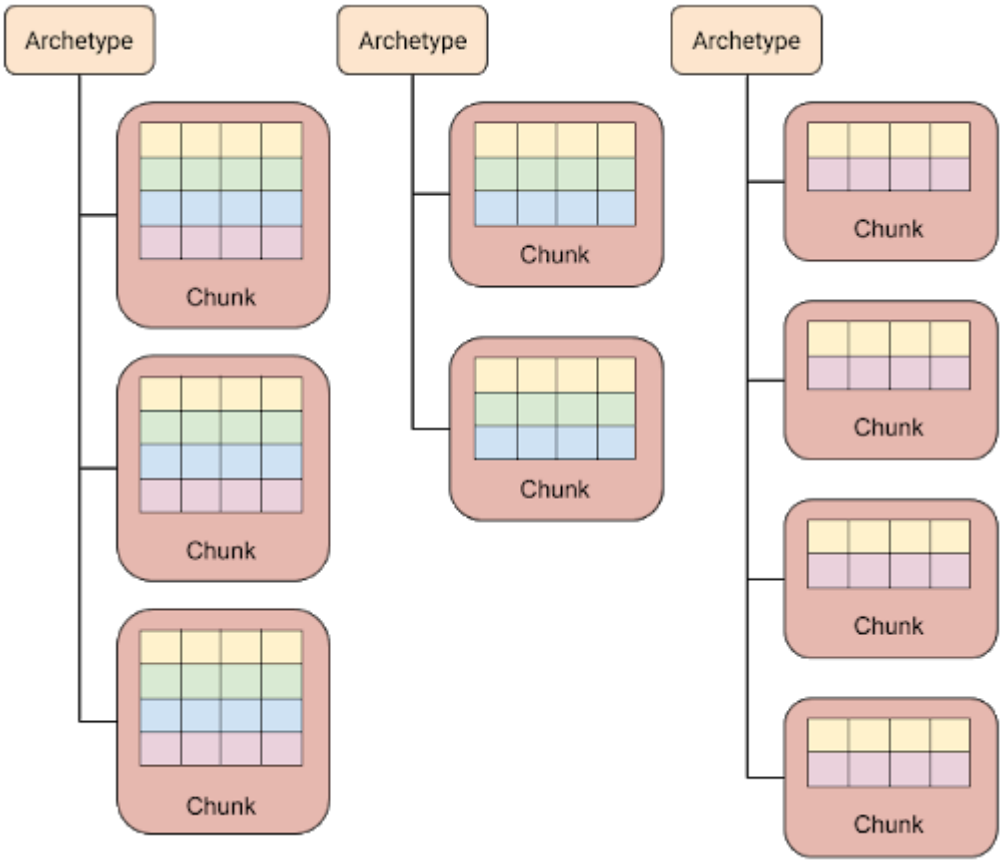


Figura 2 Diagrama Chunks

iv. System

O Systens são a lógica do programa em si, são neles onde os dados das componentes são manipulados, um exemplo bastante simples que pode ser referindo para entender do que se trata dos sistemas em ECS, se houver uma entidade em cena e esta possuir a componente rotação o system é que modifica os valores da rotação da entidade que está em cena.

4. Benefícios

DOTs aumenta substancialmente o desempenho do código e resolve de uma maneira eficaz os problemas de codificação Multithread, além destas melhorias também otimiza a utilização da memória, como é uma programação orientada a data, sempre que uma entidade(em OOP (POO) conhecida como objeto) já não é necessária a memoria que esta estava alocada é libertada e pode ser ocupada por uma outra entidade algo que não acontece quanto se utiliza OPP(POO), os principais benefícios de usar este novo paradigma, é a possibilidade de obter um alto desempenho sem possuir hardware de elevado custo e a simplicidade da reutilização do código, otimiza o desempenho do software no hardware, não é necessário nos preocupar em programar como os dados são organizados ou armazenados na memória porque a ECS é responsável por otimizar o armazenamento dos dados na memória e como estes serão processados.

5. Projeto

a. Estruturação

Na primeira parte de desenvolvimento deste projeto foi identificado os principais problemas que o ReH@panel possui, este são o uso excessivo de CPU, ser um software pesado quando é executado, por isso foi necessário fazer uma restruturação do software.

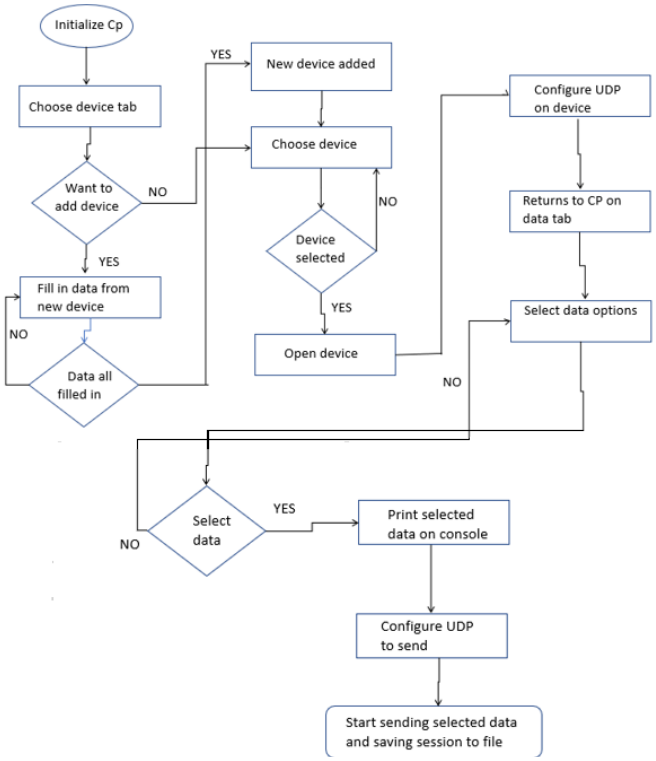


Figura 3 Fluxograma principal da atualização Reh@panel

A figura 3 está representado o novo funcionamento do Reh@panel, este irá funcionar como um “hub” que a partir do CP podemos abrir os módulos do dispositivo que se pretende utilizar ou até mesmo adicionar novos módulos de dispositivos que não estejam na interface do CP, apos da seleção o device este deve ser configurado (port, id) para ser conectar ao CP, apos esta conexão feita no CP, é selecionados os dados que se pretende enviar para o Rehanet Training Games , apos desta seleção configura-se a conexão do Rehanet Training Games para este receber os dados que se pretende enviar a partir do CP.

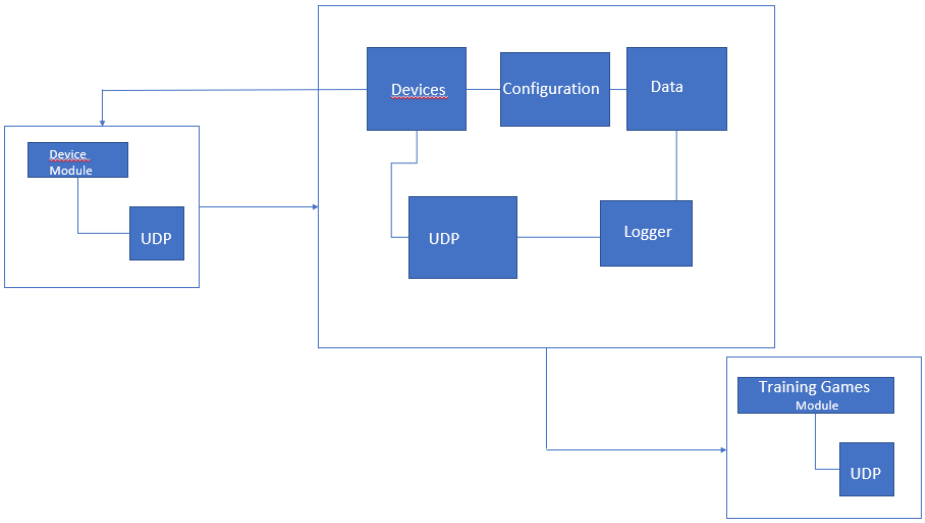


Figura 4 Arquitetura da atualização Reh@panel

Na figura acima encontra-se a arquitetura da atualização do CP, a criação desta arquitetura é a segunda parte da estruturação desta atualização por ser constituída por 2 blocos principais que anteriormente se encontravam juntos que é o bloco do modulo do device e o bloco do CP, o bloco do device terá um modulo para configuração do mesmo i.e selecionar o dados que o próprio device pode enviar para o CP e um outro modulo para configurar a comunicação do mesmo. O bloco CP será responsável por inicial apenas o device que o utilizador pretender utilizar ou adicionar ao próprio CP, será possível selecionar no CP os dados que se pretende enviar para o Training Game essa função pertence ao modulo Data do CP, o modulo UDP funcionará de forma similar ao modulo UDP do device a diferença será que enquanto o modulo do device apenas envia dados o CP irá receber e enviar para o Training Game.

b. Implementação

Foi desenvolvido scripts utilizado DOTs para o desenvolvimento da atualização para o Reh@panel, e esses sripts foram o de adicionar dispositivos, os scripts para a comunicação, no scripts de adicionar os dipositivos foi utilizado a paradigma de DOTs e de OPP(POO) a necessidade de utilizar ambos deve-se ao fato de todas as packages que temos de instalar na Unity para utilizar DOTs estam todas numa versão preview(um versão que ainda não está finalizada), é aqui que a primeira barreira para a implementação total em DOTs aparece, pois por não ser packages finalizadas, ao tentar programar um botão no OnClick()(atribuição a um botão, uma função em algum script) com uma função utlizado DOTs ele não reconhece essas funções por este não ser um paradigma completamente implementado.

O scprit responsável por a criação de botões é o ***OpenFile,*** neste script temos a função ***OpenExplorer*** que serve para abrir o OpenExplorer da Windows para selecionar a partição do executável se pretende fixar a um botão. A função ***ValueDropdown*** serve para selecionar a categoria do dispositivo que se está adicionar ao CP, a ***CreateError*** serve para a criação das mensagens de erros ao inserir os dados do botão i.e caso algum campo não seja preenchido é

criado uma mensagem de aviso para o user para o preencher o campo em falta, para instanciar é utilizado a função ***Ins_Button*** e a ultima função deste script é a ***ButtonClicked***, esta função serve para quando se clicar num botão criado este vai executar que lhe foi atribuído, esta função é necessária porque os botões são criados dinamicamente não é possível usa o recurso OnClick da Unity.

Na parte da comunicação não foi possível terminar a sua implementação, esta é constituída por dois scripts que são o JobClient e o JobServer, como os nomes indicam um é o Client que envia dados e o é o Server que recebe os dados.

Começado por o JobServer, o primeiro aspeto que se deve salientar é a utilização de struct e dentro dessas mesmo struct temos funções nativas como Onpdate, Execute algo que difere com a forma tradicional de programar na unity, a primeira struct que temos no JobServer é a ServerUpdateConnectios, é a estrutura principal do server pois a entidade com esta estrutura terá uma networkdriver e uma nativelist que servirá para armazenar todas as conexões realizadas. A outra struct é a ServerUpdateJob esta estrutura necessita de uma atribuição diferente da primeira, porque esta estrutura serve para atualizar os dados recebidos de forma constante e de os reencaminhar (não implementado a seção de reencaminhar os dados), essa atribuição chama-se IJobParalleForDefer, enquanto a primeira é uma struct genérica do JobSystem e as struct genéricas por norma têm a atribuição IJob, e por final o script possui um MonoBehaviour para que se possa atribuir este script a algum GameObject.

O JobClient funciona de forma similar ao JobServer mas este apenas envia os dados e não recebe nenhum, a nível das struct foi necessário utilizar a struct genérica do JobSystem (IJob) e a struct da atribuição IJobParalleForDefer, e no final a necessidade de colocar um MonoBehaviour para ser possível colocar este script num GameObject.

c. Produto Final

Nesta nova versão do CP(Control Panel), possuímos 4 tabs figura 4, cada uma está responsável a mostra ao utilizado um segundo menu.

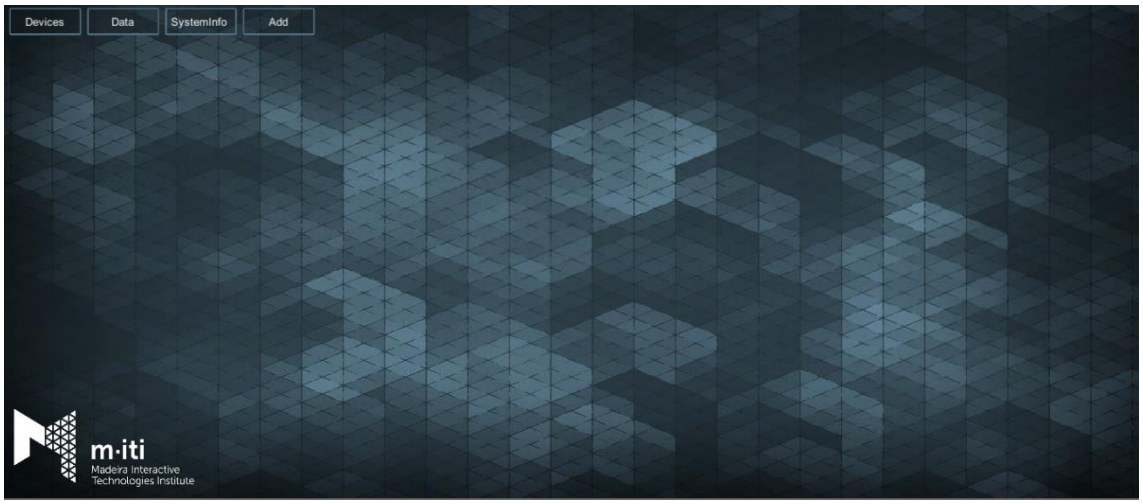


Figura 5 Janela principal

A tab Devices mostra ao utilizador todos os devices que se encontram instalados no CP, inicialmente esse menu se encontra por sem nenhum device pois o utilizador apos de instalar o CP necessita de adicionar os devices que pretende possuir no seu CP como se pode ver pela a figura 6.

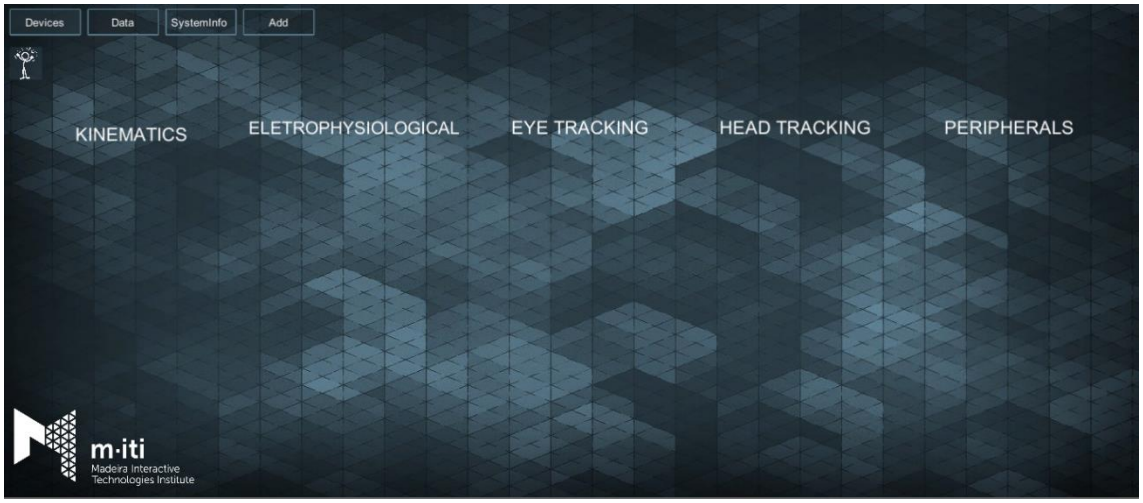


Figura 6 janela dos devices

Como é possível visualizar pela a figura 6, existem 5 categoria de devices, para que seja possível cada botão ser criado na posição correta foi definido 5 reacttransform(coordenadas onde se encontra um objeto (X,Y,Z)), cada reacttransform corresponde as coordenadas de cada categoria, para que os mesmo sejam criados junto ao objeto que indica a sua categoria, o objecto que possui o nome da categoria possui um componente designado por Vertical Layout

Group que este componente serve para que seja possível agrupar diverso objetos numa coluna com o mesmo espaçamento entre os mesmos.

Na figura abaixo encontra-se o menu apresentado quando o utilizador seleciona o a tab para adicionar um novo device ao seu CP, o utilizador necessidade de colocar um nome que este é guardado num variável do tipo texto(texto), que que mais tarde esta seja utilizada quando o botão é instanciando, o campo Path é o campo o utilizador colocar a localização do modulo executável do device, inicialmente a localização fica guardada numa variável do tipo text no mesmo instante é realizado um casting de text para string, porque a função StartInfo(função predefinida por a biblioteca System para que seja possível a execução de ficheiros externos ao software), apenas aceita variáveis do tipo string, por fim temos um dropdown que possui cada categoria dos devices nesta secção que o reacttraform é escolhido.

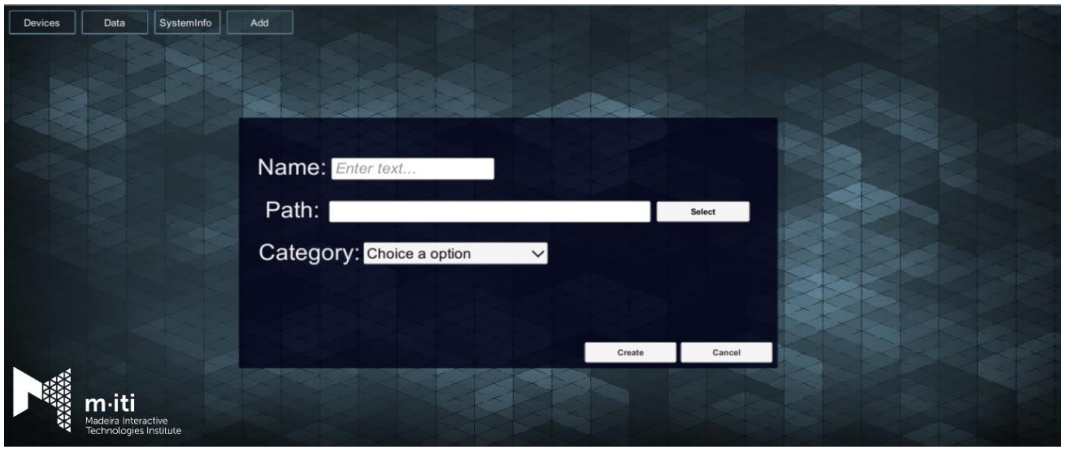


Figura 7 Janela adicionar Devices

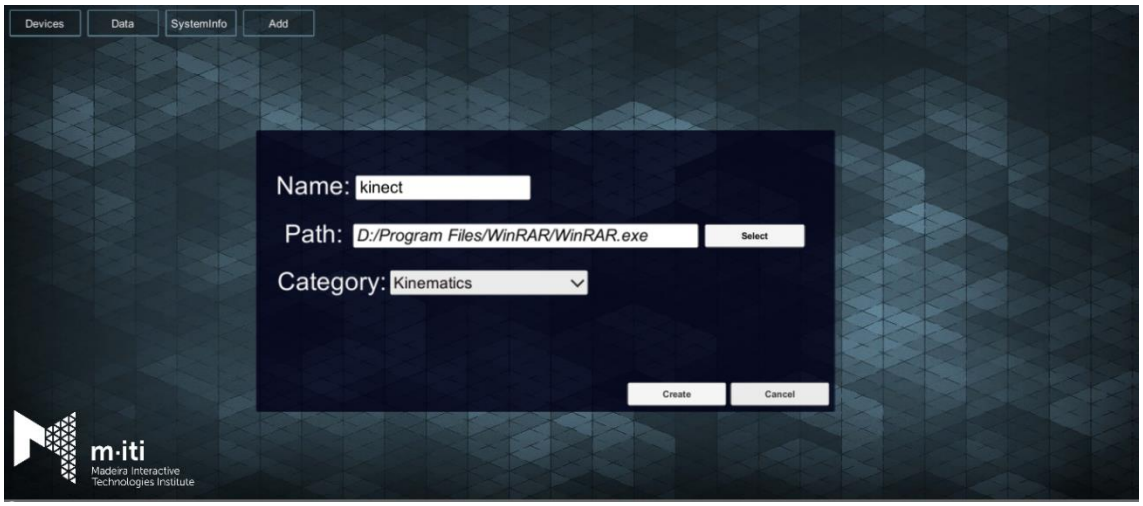


Figura 8 Janela preenchida para o botão

Quando o a criação do botão está concluída ele aparece no menu dos devices, o botão criado é um prefab(um objeto genericamente criado com a finalidade de tornar o mesmo reutilizável o tempo todo) que anteriormente criado. Cada butão possui a função ButtonClicked e a mesma

requer uma string que é a localização do executável e um id único para que cada um funcione de forma independente.

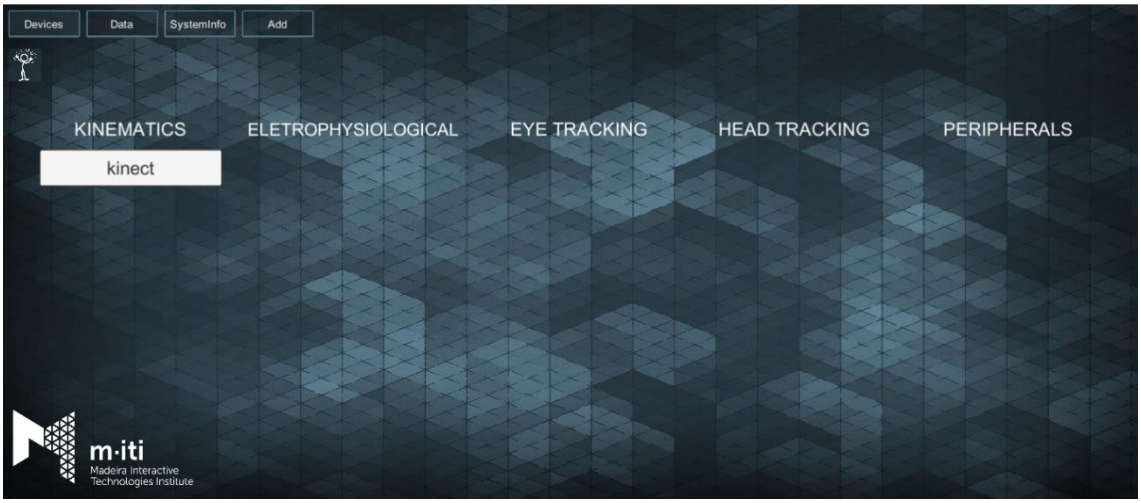


Figura 9 Botão criado com sucesso

6. Resultados

Os resultados obtidos comprovam que a utilização de DOTs será a melhor opção para otimizar ao máximo o Reh@panel, pois o atual feito em OOP(POO) tem um uso excessivo do processador a realização de nenhuma tarefa como se pode conferir na figura seguinte,

Figura 10 Uso do atual Reh@Panel

Como se confere na figura o atual Reh@Panel utiliza 37% do processador apenas estando aberto sem executar nenhum ligação a um dispositivo nem qualquer outra função.

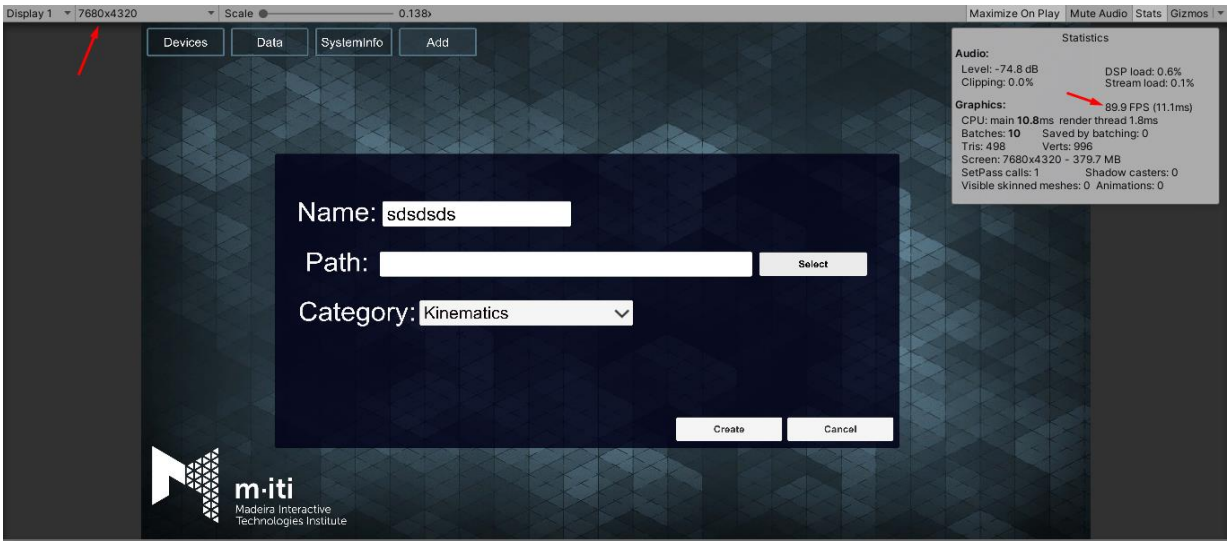


Figura 11 Actualização do Reh@panel

Passado para DOTs é possível no mesmo hardware rodar a atualização numa resolução 4K(7680X4320) e obter em média 80fps(frame per second), é uma comparação justa para afirmar que DOTs é a melhor opção para a otimização do software de certo modo é possível deduzir que é, porque o atual está feito em 720p e em média está nos 60fps quando não realizar qualquer tarefa e ao compilar o software na Unity, usar uma resolução altíssima e mesmo assim obter uma média de fps ótima (80), mostra como DOTs aproveita todo o potencial do hardware, pois sempre que se compila qualquer software ou jogo na Unity é criado um género de maquina virtual, e o hardware fica divido para a maquina virtual e a Unity, visto isto posso de certo modo afirmar que é o caminho certo o único contra é as packages necessárias estarem num versão preview e não estarem instáveis.

Quando à comunicação foi possível estabelecer permanente uma ligação entre cliente e servidor no qual o servidor receber o dado enviado pelo o cliente e confirma como recebeu o dado com sucesso, porém está a ocorrer uma perda no envio, no qual não é possível ainda justificar a perda, pois quando se faz uma busca pelo o mesmo apenas existe dois resultados e os resultados são antigos e não resolvem o atuar problema.

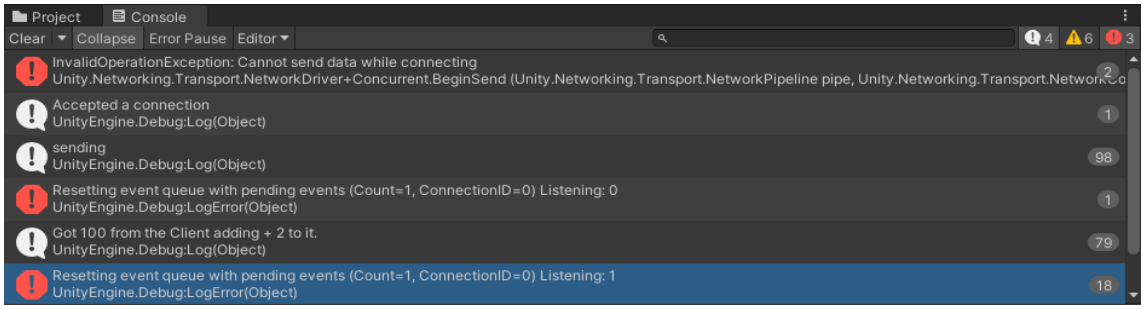


Figura12 Erros no envio de dados

Como se pode conferir na imagem acima o primeiro erro que ocorre é insignificante pois o motivo de ocorrer é por a ligação entre Cliente e Servidor não estar ainda realizada, após da mesma estar realizada o mesmo erro não ocorre mais nenhuma vez. Como se pode ver o cliente enviou 98 pacotes, enquanto o Servidor confirmar apenas 79, há uma perda de 19 pacotes e cada pacote corresponde a um erro de “Resetting event queue with pending events”.

```
#if ENABLE_UNITY_COLLECTIONS_CHECKS
            for (int i = 0; i < m_ConnectionList.Length; ++i)
            {
                int conCount = m_EventQueue.GetCountForConnection(i);
                if (conCount != 0 && m_ConnectionList[i].State != NetworkConnection.State.Disconnected)
                {
                    UnityEngine.Debug.LogError("Resetting event queue with pending events (Count=" +
                                               conCount +
                                               ", ConnectionID="+i+") Listening: " + Listening);
                }
            }
#endif
            int free;
            while (m_PendingFree.TryDequeue(out free))
            {
                int ver = m_ConnectionList[free].Version + 1;
                if (ver == 0)
                    ver = 1;
                m_ConnectionList[free] = new Connection {Id = free, Version = ver};
                m_FreeList.Enqueue(free);
            }

            m_EventQueue.Clear();
            m_DataStream.Clear();
            CheckTimeouts();
        }
```

Figura13 Função InternalUpdate

Como se pode conferir na figura 12, o erro que ocorre 19 vezes, é nada mais que uma espécie de warnings, pode-se afinal porque a figura 13 possui a função InternalUpdate que pertence ao ficheiro **NetworkDriver.cs,** ficheiro este que pertence a biblioteca da Unity Transport, como anteriormente dito é uma espécie de warnings porque é utilizado o UnityEngine.Debug.LogError, como se pode conferir na figura 13, este debug possui o símbolo de erro os motivos para este possível erro pode ser devido a TimeOuts mas neste momento é algo que não se tem certezas, porque a documentação existente é escassa e incompleta devido as packages estarem em estado de desenvolvimento.

7. Conclusão

Nesta fase final sinto-me um pouco insatisfeito com o trabalho, por este se ter tornado um pouco mais teórico do que prático, e por não ter sido possível todos os objetivos por barreiras externas que foram exposta, as packages utilizadas iriam ser finalizadas durante o presente ano e serem lançadas oficialmente e todos os recursos que existem na Unity para atribuir alguma função a um elemento da GUI iriam reconhecer as funções escritas em DOTs, o desenvolvimento dessas packages seriam oficialmente apresentadas na versão 2020.1 da Unity e por motivos pandémicos vividos em 2020, estas packages continuaram numa versão preview mas com uma atualização nos seus próprios métodos e nesta atualização as mesmas ficaram mais estáveis, mas sem ser possível as associar a um botão.

Devido as packages estarem numa versão preview e por não estarem completamente implementas na Unity, numa primeira vista pode parecer um grande erro em substituir OOP(POO) por DOTs na programação na Unity porque os recursos da Unity apenas reconhecem por enquanto a programação em OOP(POO), mas não é, porque os dois principais objetivos para esta atualização é a redução do uso excessivo do CPU e tornar num software mais simples, a atual versão Reh@panel foi implementado em OOP(POO) e como se viu na figura ele sem fazer qualquer conexão com um Rehanet Training Games, usa em média 38% CPU, o que é um uso excessivo, para reduzir este uso seria necessário escrever o código de forma diferente em OOP e uma possível forma de reduzir esse uso seria, obter uma paralelização(uso de vários cores do processador para executar uma tarefa) e para obter isto em OOP é difícil e no caso mal programado levará a lentidão do software, já em DOTs não temos esse problema pois a paralelização é “feita automaticamente” porque este paradigma é focado para a utilização mais eficiente dos recursos do hardware, e o grande problema de DOTs é a escassez de exemplos de como utilizar e os exemplos que existem são todos fornecidos pela a Unity.

Desde o inicio que apareci às Group Meetings, mas por o meio deixei de aparecer, nesta reuniões consegui compreender o foco do grupo e o trabalho de todos os membros do grupo, agradeço imenso a todo o grupo pois desde o inicio me senti 100% integrado, inclusive fiz uma apresentação numa dessas reuniões sobre o projeto e ao elaborar a apresentação cometi o erro de fazer uma apresentação técnica a mais pois me esqueci que nem todos os membros do grupo são formados em IT.

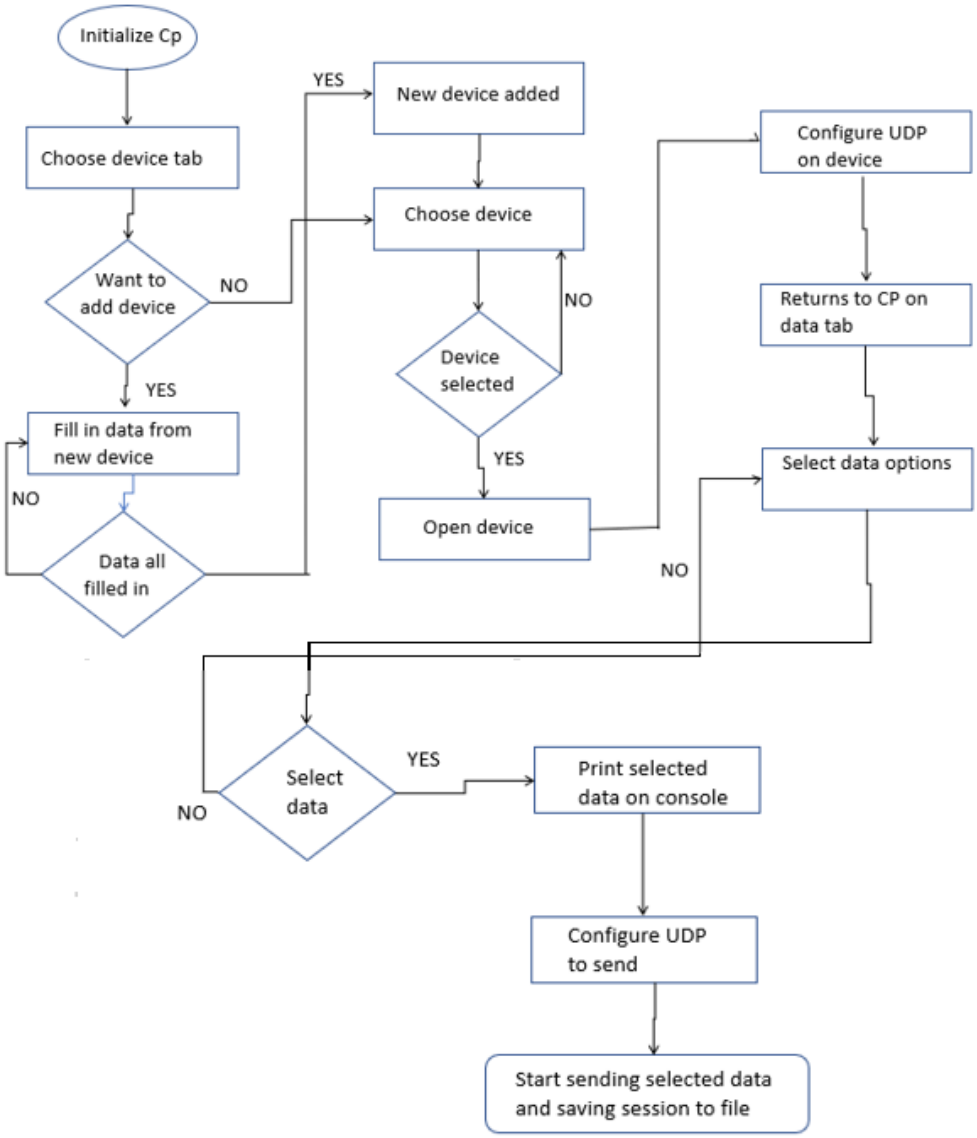
Por fim projeto tem imenso potencial, pois a utilização de DOTs será revolucionário pois a otimização do hardware ao “limite”, é algo que antigamente era extremamente difícil e raramente feita quando o paradigma usado é OOP(POO) por ser algo extremamente difícil programar, mas com esta nova forma toda tudo isso mais simples, a transição de OOP(POO) para DOTs é complicado pois uma forma não tem nada haver com a outra e este paradigma e a falta de exemplos para facilitar essa transição ainda complica mais um pouco.

No trabalho futuro seria terminar a comunicação e apos desta terminar, terminar as outras funções do CP e por a criação de um modulo genérico para novos dispositivos, com este modulo genérico sempre que houvesse um novo dipositivo seria “quase automatico” a criação do modulo para esse dispositivo.

No fim, queria agradecer a todo NeuroRehaLab, por a forma com me recebeu e por toda a ajuda que deram deste o inicio deste projeto, um agradecimento especial ao membro Yuri Almeida por toda ajuda que me deu, ao ajudar o Prof Sergi Bermúdez a me orientar neste projeto e o um grande agradecimento ao orientador Prof Sergi Bermudex por ter aceitado me orientar e por esta oportunidade de trabalhar num projeto desafiante.

1. Anexos

I. Fluxograma principal do funcionamento do CP



II. Arquitetura do Controlpanel