

Documentação do software do sistema embarcado - requisitos e modelagem

Modelo machine learning (TinyML) de reconhecimento de movimento usando Raspberry Pi Pico

Jonas Lendzion Schultz II

link GitHub:

<https://github.com/jonas-ii/jonasii-pico-motion.git>

Mestrando em Engenharia Elétrica
202304977
jonasschultz1309@gmail.com

1 Introdução

O presente relatório tem como objetivo documentar o software utilizado no sistema embarcado e no hospedeiro (PC), bem como os requisitos de modelagem do projeto final da disciplina EEL510265 - Programação de Sistemas Embarcados ministrada pelo Prof. Eduardo Augusto Bezerra.

Fazendo um breve resumo do projeto final, a ideia é implementar um modelo de machine learning embarcado (TinyML) em uma Raspberry Pi Pico para a identificação de movimentos.

Para se captar esses movimentos, utiliza-se um acelerômetro MMA845X. Diferentemente da proposição original, esse é um acelerômetro digital (e não analógico), logo, para o software embarcado na Raspberry Pi Pico, foi adicionalmente necessário implementar a comunicação via i2C aumentando um pouco mais a complexidade do projeto.

2 Coleta de dados - 1^a versão do software embarcado (sem TinyML)

A própria empresa Raspberry Pi possui um repositório no GitHub, no qual existe um exemplo de utilização de um acelerômetro MMA8451. Ao implementar o exemplo '.c' em conjunto com uma 'CMakeLists', percebeu-se que alterações são necessárias nesses arquivos com o intuito de proporcionar o funcionamento do acelerômetro digital em específico - com a árvore de projeto inicialmente proposta. Percebe-se que a maioria do exemplo está escrito em *C*, no entanto, esta versão é somente uma preliminar e a utilizada no projeto está em *C++* como requisitado.

Finalmente, a estrutura em árvore do arquivo do projeto de coleta de dados é a seguinte:

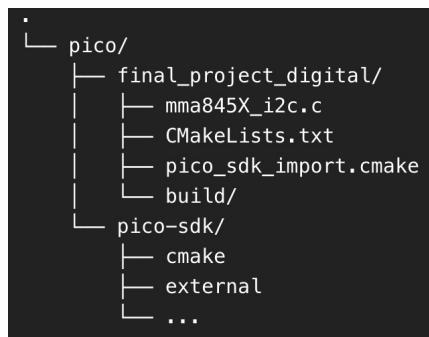


Figure 1: Árvore do projeto (sem ML)

Uma vez feito o 'cross compiler', envia-se o arquivo com a extensão '.uf2' para a Raspberry Pi Pico.

O set-up utilizado é o seguinte:

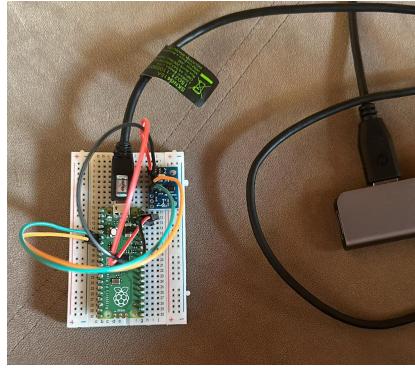


Figure 2: Set-up do projeto

Alguns movimentos foram testados utilizando recursos do Arduino (Serial Monitor e Serial Plotter) para verificar o bom funcionamento do sistema embarcado. Aqui está um exemplo ao realizar alguns movimentos aleatórios somente com o intuito de analisar se o sensor acelerômetro estava variando seus eixos (x,y,z) de acordo com os movimentos realizados:

	x	y	z
13:36:00.000	-2.14407220	0.00000000	2.022320
13:36:05.005	-1.111329	-0.574895	0.656719
13:36:06.588	-1.168770	-0.593965	0.637559
13:36:07.091	-1.302891	-0.689766	0.675879
13:36:07.578	-1.494492	-1.187930	0.177715
13:36:08.083	-1.475332	-1.130449	0.407638
13:36:08.557	-> -9.771680	1.362891	0.421523
13:36:09.070	-> -10.212363	4.100274	-1.034649
13:36:09.570	-> 3.487149	-7.740704	9.350157
13:36:10.065	-> 0.019160	7.664065	8.181387
13:36:10.591	-> 0.249082	10.866329	1.379531
13:36:11.092	-> 1.045227	7.803899	-1.437012
13:36:11.577	-> 1.443157	10.901331	0.401229
13:36:12.087	-> 2.126777	11.649376	-1.382891
13:36:12.593	-> 7.683223	-7.893985	1.724414
13:36:13.090	-> 7.836590	-1.130449	-0.862207
13:36:13.595	-> -22.896387	-5.384904	12.319981
13:36:14.062	-> -20.003204	2.337530	11.323653
13:36:14.593	-> -4.778879	-2.375859	12.319981
13:36:15.094	-> -1.705254	0.823887	16.995060
13:36:15.572	-> -0.306563	-1.034649	9.733359
13:36:16.082	-> -0.728086	-1.267090	8.813672
13:36:16.588	-> -0.287400	-1.130449	9.500078
13:36:17.089	-> -0.102333	-1.096009	9.400000
13:36:17.567	-> -0.249982	-1.149609	9.829161
13:36:18.095	-> -0.268242	-1.092129	9.637559
13:36:18.591	-> -0.268242	-1.168770	9.637559
13:36:19.095	-> -0.287402	-1.092129	9.714200
13:36:19.605	-> -0.268242	-1.149609	9.695049
13:36:20.072	-> -0.344883	-1.130449	9.790840
13:36:20.573	-> -0.268242	-1.168770	9.695040

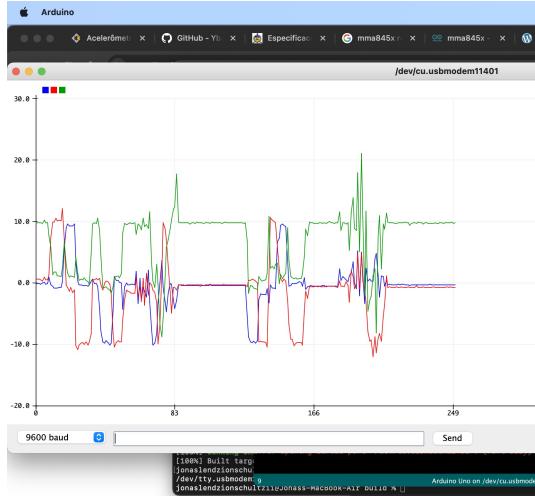


Figure 3: Monitor serial e plotter ao realizar movimentos com o set-up

Após a validação dessa etapa, parte-se para a realização do modelo de machine learning embarcado (TinyML).

3 Software embarcado versão final - com TinyML

Para tanto, será utilizado o programa 'Edge Impulse Studio'.

Depois que o projeto for criado e a CLI instalada, a maneira mais fácil de obter dados do Pico é usando o encaminhador Edge Impulse Data. Isso permite encaminhar dados coletados

por uma interface serial para o estúdio. Este método só funciona perfeitamente em sensores com frequências de amostragem mais baixas, como no caso do projeto em questão (gestos humanos).

O encaminhador de dados é usado para retransmitir facilmente dados de qualquer dispositivo para o Edge Impulse via serial. Os dispositivos gravam os valores dos sensores por meio de uma conexão serial e o encaminhador de dados coleta os dados, assina-os e envia-os ao serviço de ingestão.

Dentro desse programa, pode-se plotar os dados (movimentos humanos realizados) para verificar se eles estão bem definidos e com isso será realizada separação do dataset em treino/validação e teste.

A imagem abaixo mostra a divisão dos clusters dentro da plataforma:



Figure 4: Divisão dos clusters - movimentos

Uma rede neural simples foi implementada para prever o movimento humano realizado. Ela possui 4 neurônios de saída - já que temos 4 classes possíveis de movimentos.

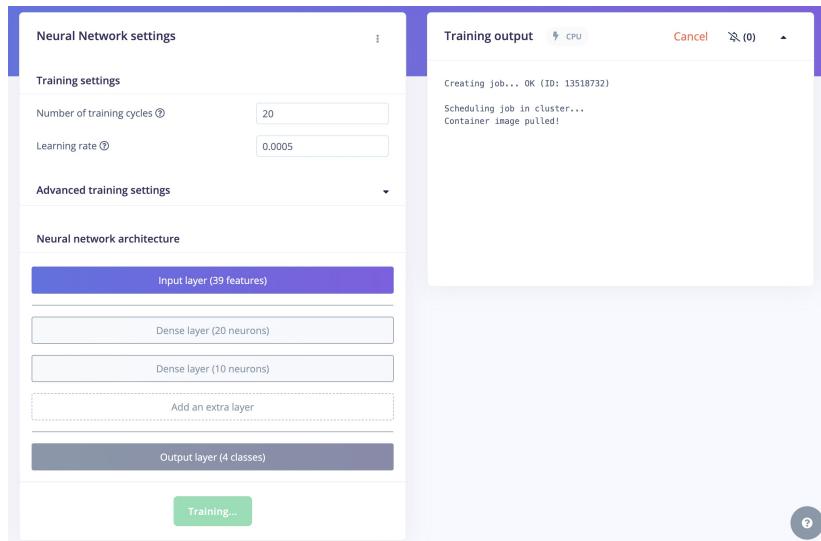


Figure 5: Rede neural implementada no treinamento do modelo

Após o seu treinamento, o seguinte desempenho foi alcançado:



Figure 6: Desempenho da rede neural com o conjunto de treinamento

E ao aplicar o conjunto de teste, o resultado obtido foi o seguinte:

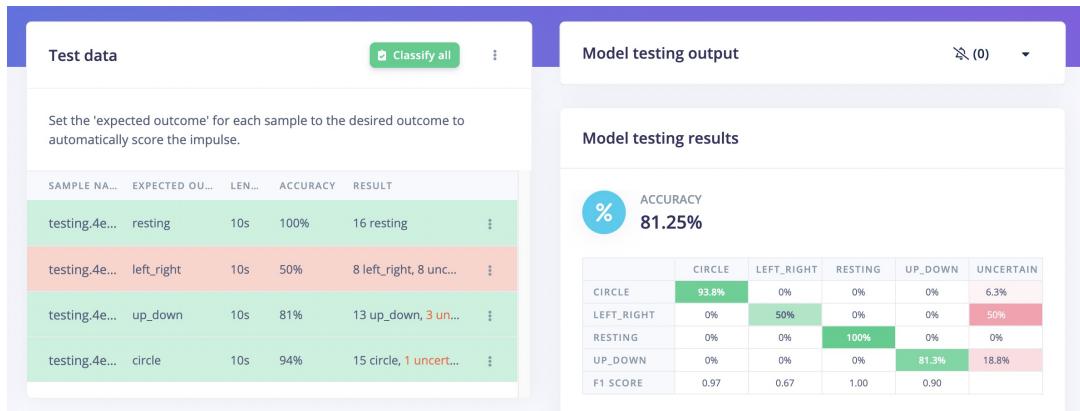


Figure 7: Desempenho com o conjunto de teste

Ressalta-se que os resultados obtidos poderiam ainda ser melhorados ao aumentar o conjunto de treinamento. No caso do projeto em questão, capturou-se 60 segundos de cada movimento para o conjunto de treinamento.

Ao final de toda a previsão e desenvolvimento do modelo, deve-se realizar a conversão e modificação dos arquivos gerados para o envio à Pico. O Edge Impulse possui uma seção de Deployment para auxiliar em que pode ser escolhida a extensão C++ para os arquivos.

O requisito principal dessas modificações é atender a proposição de utilizar uma fila de eventos (nesse caso um buffer com um ID específico para cada evento) dos gestos humanos e adicionar a função de leitura do rtc interno para a adição da data/hora do evento.

Finalmente, a estrutura final do projeto é a seguinte:

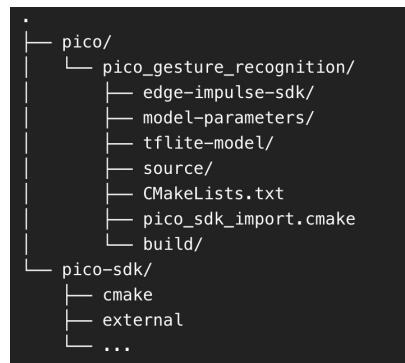
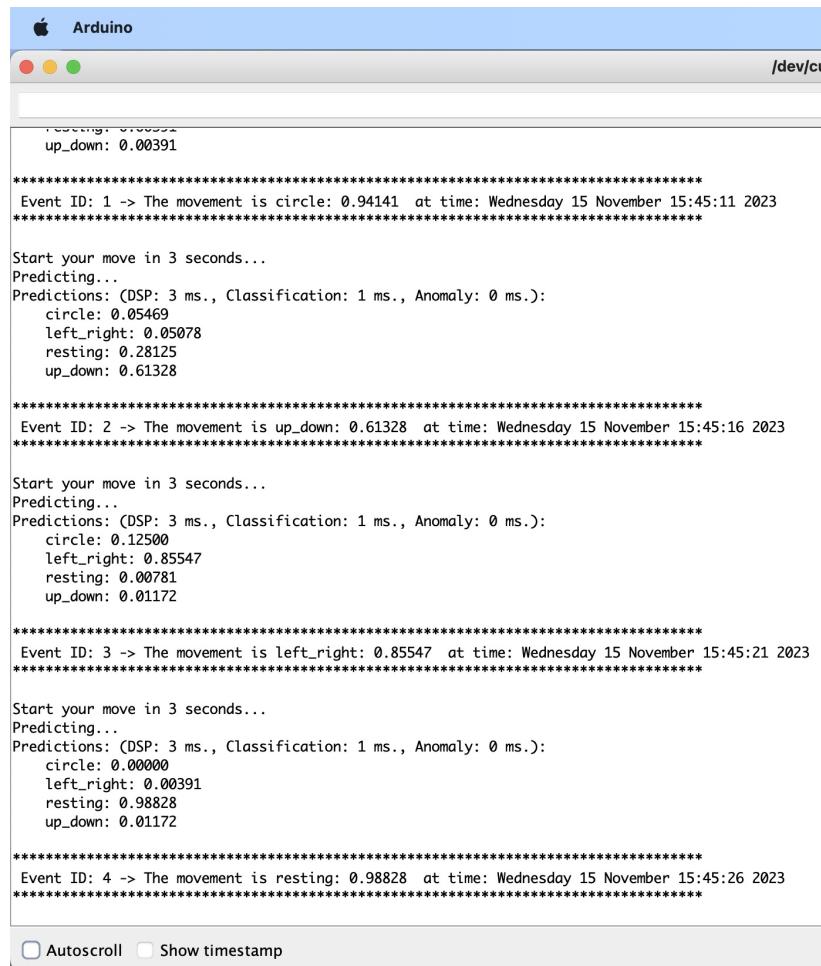


Figure 8: Árvore do projeto (com ML)

Após a criação de um arquivo 'main.cpp' dentro da pasta *sources/* e das respectivas mudanças para o envio da mensagem via USB/Serial, tem-se o seguinte resultado:



The screenshot shows the Arduino Serial Plotter window. The title bar says "Arduino". The top right corner shows the port name "/dev/cu". The main area displays a series of movement events. Each event includes an ID, movement type, time, and a timestamp. Below each event, there is a prediction section showing DSP, Classification, and Anomaly values for four categories: circle, left_right, resting, and up_down. The events are as follows:

- Event ID: 1 -> The movement is circle: 0.94141 at time: Wednesday 15 November 15:45:11 2023
- Event ID: 2 -> The movement is up_down: 0.61328 at time: Wednesday 15 November 15:45:16 2023
- Event ID: 3 -> The movement is left_right: 0.85547 at time: Wednesday 15 November 15:45:21 2023
- Event ID: 4 -> The movement is resting: 0.98828 at time: Wednesday 15 November 15:45:26 2023

At the bottom of the plotter window, there are two checkboxes: "Autoscroll" and "Show timestamp".

Figure 9: Serial Plotter do software embarcado

O evento, inicialmente presente em um buffer do sistema embarcado, é enviado via USB/Serial para o hospedeiro (já que, conforme relatado pelo professor, não há mais a necessidade de se usar uma UART). Nessa etapa utilizou-se somente o Serial Plotter da interface Arduino, porém, para a próxima seção, essa fila (buffer) será enviada para uma outra fila no hospedeiro de acordo com as especificações propostas.

Observa-se que esses eventos possuem um ID e uma data/hora específico. Após a validação dessa etapa, parte-se para o desenvolvimento do software do hospedeiro.

4 Software do hospedeiro (PC)

A primeira modificação foi feita no software embarcado na Pico, pois visa-se reduzir o tamanho da mensagem já que a mesma será copiada para um 'log' em forma de fila no host (PC).

Dante disso, somente a mensagem presente entre os asteriscos (*) da Figura 10 é enviada via USB/Serial para o host.

O host possui um sistema operacional MacOS, logo, foi utilizado o 'Xcode' em uma forma de *Xcode project* para a realização do programa em *C++*.

O programa acessa a porta USB correspondente à conexão da Raspberry Pi Pico - consequentemente também seta a *Baud Rate* necessária e o caminho até essa porta - e define as funções correspondentes para realizar a leitura dos eventos enviados pelo setup embarcado. Cabe, então, ao usuário escolher entre as diversas opções disponibilizadas o que ele deseja executar, conforme a imagem abaixo:

```
*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option:
```

Figure 10: Menu do Software do Host (PC)

Alguns casos do funcionamento desse software do host serão mostrados em seguida.

Inicialmente, mostra-se a funcionalidade de adquirir um evento (movimento) do sistema embarcado e adicioná-lo a uma fila, além disso, como requisitado, é possível ao 'admin' visualizar os eventos presentes nessa fila:

```
*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 1

The event is: Event ID: 444 -> The movement is resting: 0.97266 at time:
Wednesday 15 November 16:22:12 2023

*****  

Welcome 'admin'  

Press 1 if you want to read a movement (5 seconds)  

Press 2 if you want to check the log (queue)  

Press 3 if you want to clear the log (queue)  

Press 4 if you want to see the active time of Pico and close the serial port  

*****  

Option: 2

The event 0 in the queue of the host is:  

Event ID: 411 -> The movement is resting: 0.97266 at time:  

Wednesday 15 November 16:19:26 2023  

The event 1 in the queue of the host is:  

Event ID: 430 -> The movement is resting: 0.98047 at time:  

Wednesday 15 November 16:21:01 2023  

The event 2 in the queue of the host is:  

Event ID: 432 -> The movement is resting: 0.98438 at time:  

Wednesday 15 November 16:21:11 2023  

The event 3 in the queue of the host is:  

Event ID: 444 -> The movement is resting: 0.97266 at time:  

Wednesday 15 November 16:22:12 2023

*****  

Welcome 'admin'  

Press 1 if you want to read a movement (5 seconds)  

Press 2 if you want to check the log (queue)  

Press 3 if you want to clear the log (queue)  

Press 4 if you want to see the active time of Pico and close the serial port  

*****  

Option: 3

The queue is now empty
```

Figure 11: Caso de funcionamento do software do host para adicionar eventos na fila e visualizá-la

Em seguida, mostra-se que o usuário 'admin' também pode reiniciar a fila ao apagar todos os eventos previamente listados nela:

```
*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 2

The event 0 in the queue of the host is:
Event ID: 472 -> The movement is resting: 0.95703 at time:
Wednesday 15 November 16:24:32 2023
The event 1 in the queue of the host is:
Event ID: 475 -> The movement is resting: 0.98047 at time:
Wednesday 15 November 16:24:47 2023
The event 2 in the queue of the host is:
Event ID: 484 -> The movement is resting: 0.97656 at time:
Wednesday 15 November 16:25:32 2023

*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 3

The queue is now empty

*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 2

*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
```

Figure 12: Caso de funcionamento do software do host ao esvaziar a fila

Além disso, conforme a especificação do projeto, o usuário 'admin' também pode acessar o tempo em que a Raspberry Pi Pico esteve ativa, nesse caso, o tempo é dado em μ s.

```
*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 1

The event is: Event ID: 502 -> The movement is resting: 0.91016 at time:
Wednesday 15 November 16:27:02 2023

*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 1

The event is: Event ID: 504 -> The movement is resting: 0.95703 at time:
Wednesday 15 November 16:27:12 2023

*****
Welcome 'admin'
Press 1 if you want to read a movement (5 seconds)
Press 2 if you want to check the log (queue)
Press 3 if you want to clear the log (queue)
Press 4 if you want to see the active time of Pico and close the serial port
*****
Option: 4

The active time of Pico is: 2537015234 (us)

Serial port closed

Program ended with exit code: 0
```

Figure 13: Caso de funcionamento do software do host ao requisitar o tempo em que o micro-controlador esteve ativo

Finalmente, após as explicações do funcionamento tanto do software embarcado e o software do host (PC), tem-se o seguinte diagrama de classes:

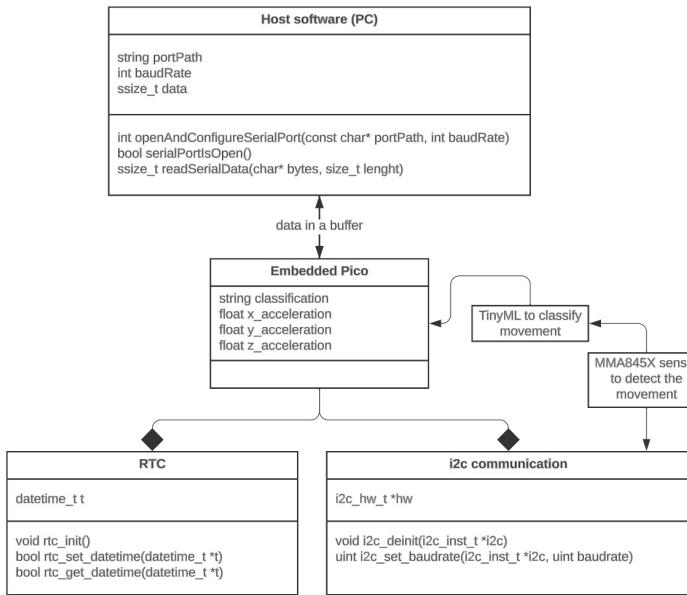


Figure 14: Diagrama de classes simplificado do projeto

5 Software do smartphone (versão preliminar)

Desenvolver um software para smartphone requer, inicialmente, uma breve pesquisa para analisar quais são as melhores linguagens de programação de acordo com sistema operacional do smartphone alvo.

Visto que possuo um iPhone e, como já relatado anteriormente, também possui um computador macOS, foi feita a escolha pela linguagem Swift, pois ela possui compatibilidade perfeita com o ecossistema iOS.

A linguagem Swift, criada pela Apple, oferece um ambiente altamente intuitivo e eficiente para a criação de aplicativos nativos, garantindo um desempenho otimizado. Além disso, ela possui inspirações na programação orientada a objetos - como foi requisitado pela disciplina para a realização do software na linguagem C++.

Swift, embora tenha sua própria sintaxe e abordagem, herda muitos conceitos fundamentais da orientação a objetos presentes em linguagens como C++: classes, estruturas, protocolos e métodos, permitindo a modelagem de entidades e o encapsulamento de dados e comportamentos.

Após essa breve introdução da linguagem e de apresentar as suas inúmeras semelhanças com C++ orientado à objetos, visto que esse é meu primeiro contato com um software realizado para celular, procurei realizar um simples exemplo de 'HelloWorld' como proposto na especificação da entrega 3.

Detalharei brevemente o código empregado nessa aplicação. O projeto se divide em dois arquivos principais, 'HelloWorldApp.swift' e 'ContentView.swift'.

O 'HelloWorldApp.swift' é o arquivo principal. Ele serve como ponto de entrada para a execução do código em um aplicativo. Quando o programa é executado, o sistema começa a execução a partir desse arquivo.

Com relação ao 'ContentView.swift', temos a declaração das seguintes **structs** - como em C++.

1. **ContentView**: Essa struct define a view principal (View) do aplicativo. Ela cria uma estrutura de visualização que consiste em um VStack (um recipiente vertical que organiza as visualizações nele) contendo uma imagem e um texto.
 - 'VStack': Empilha visualizações verticalmente. No código fornecido, contém uma imagem e um texto.
 - 'Image': Exibe uma imagem usando o sistema de ícones dos Symbols do iOS, nesse caso, utilizou-se o símbolo "globe".
 - 'imageScale': Define o tamanho da imagem como grande (large).
 - 'foregroundColor': Define a cor da imagem como a cor de destaque (accentColor) definida no tema do aplicativo.

- (e) '*Text*': Exibe um texto com o conteúdo "Hello world from the smartphone app!". O padding() aplicado no VStack adiciona um preenchimento ao redor do conteúdo dentro do recipiente.
2. ContentView_Previews: Esta struct é um PreviewProvider, que oferece uma visualização de pré-visualização do ContentView durante o desenvolvimento no ambiente de desenvolvimento da Apple (o Xcode nesse caso). A função previews retorna uma instância de ContentView para mostrar uma representação visual do layout e aparência do aplicativo enquanto você o desenvolve.
- (a) '*previews*': Retorna uma instância de ContentView() para exibição na interface de pré-visualização do Xcode.

A utilização da struct logo acima é muito interessante pois permite ao programador emular o funcionamento do código (feito para um smartphone iOS) em seu computador (macOS).

Com isso, tem-se o seguinte resultado:

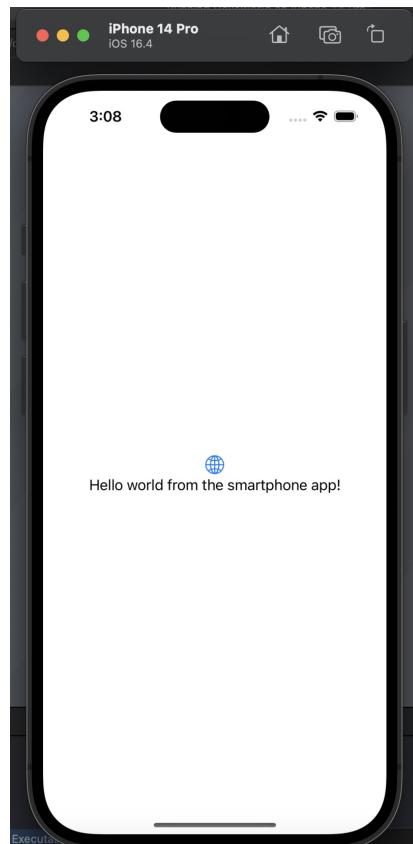


Figure 15: Software 'Hello world' do smartphone

Apesar de ser muito semelhante à linguagem *C++*, na Swift as visualizações são construídas como estruturas que estão conforme ao protocolo chamado View.

Diante disso, efetuou-se um diagrama de classes conceitual, indicando as relações entre as estruturas:

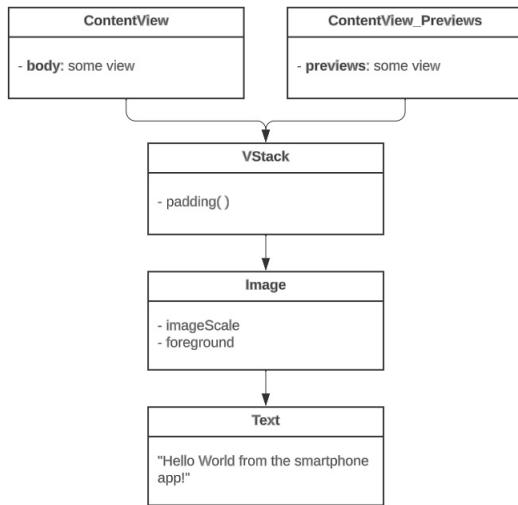


Figure 16: Diagrama das estruturas presentes no software do smartphone

Com relação ao próximos passos e plano de testes, com a finalidade de facilitar o desenvolvimento do software, será suposto que o smartphone já está recebendo a mensagem recebida entre (*) da figura 9 (via bluetooth, usb, etc - à definir). Será realizada uma tentativa de programar a interface desse recebimento e, com isso, pode-se realizar a exibição da mensagem do movimento que o modelo de TinyML está prevendo e praticar alguma ação com ela (mudança de cor do ecran, inserir um emoji, etc).