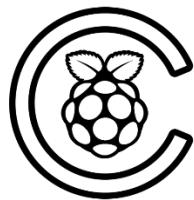


ESPROSER – Escola Profissional de Sernancelhe

CarPlay Pi



Integração de um sistema *CarPlay* em *Raspberry Pi* com *interface touch*



Marcos Ribeiro, n.º 15, EAC9, 12.º ano

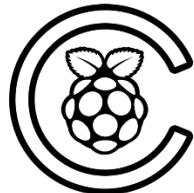
Orientador: Professor Fábio Coelho

Sernancelhe

2024/2025

ESPROSER – Escola Profissional de Sernancelhe

CarPlay Pi



Integração de um sistema *CarPlay* em *Raspberry Pi* com *interface touch*



Marcos Parreira Ribeiro, n.º 15, EAC9, 12.º ano

Curso Profissional Técnico de Eletrónica, Automação e Computadores

Orientador: Professor Fábio Coelho

Sernancelhe

07/03/2025

Índice de Conteúdos

Agradecimentos	8
Introdução.....	10
Contexto	10
Objetivos.....	12
Objetivos Técnicos	12
Objetivos Funcionais	13
Objetivos Académicos e Práticos	14
Motivação	14
Metodologia.....	16
1. Pesquisa e Planeamento.....	16
Revisão bibliográfica.....	16
Definição de requisitos	16
Seleção de ferramentas	16
2. Desenvolvimento Iterativo	17
Prototipagem física:.....	17
Configuração de software:.....	18
3. Testes e Validação.....	19
Testes funcionais:	19
Testes de robustez:.....	19
Ajustes pós-teste:.....	19
Capítulo I	20
Enquadramento Teórico.....	20
1. Fundamentos Técnicos do CarPlay	20
Desenvolvimento e Figuras-Chave.....	20
Funcionamento e Protocolos	22
2. Arquitetura do Raspberry Pi como Sistema Embarcado	22

Índice

3.	Princípios de Otimização de Sistemas Operativos	23
4.	Protocolos de Comunicação e Compatibilidade.....	24
5.	Modelação 3D e Ergonomia	26
6.	Desafios Técnicos e Soluções.....	26
	Impacto no Projeto	28
7.	Conclusão do Capítulo	28
	Capítulo II.....	29
	Componentes, Materiais e Software.....	29
1.	Componentes e Materiais	29
	Critérios de seleção.....	30
2.	Software Utilizado.....	30
	Configurações-Chave do Software.....	31
3.	Orçamento	32
	Notas Orçamentais.....	32
4.	Fluxo de Trabalho Hardware-Software	33
	Montagem Física	33
	Configuração de Software	33
	Testes Preliminares	33
	Capítulo III	34
	Implementação Prática	34
1.	Montagem Física	34
	Passo 1: Preparação do Hardware	34
	Passo 2: Integração do Carlinkit.....	35
	Passo 3: Montagem da Estrutura 3D	35
2.	Configuração de Software	36
	Passo 1: Otimização do Lineage OS	36
	Passo 2: Automatização do CarPlay via Interface do APK	36

3. Testes e Validação.....	37
Teste 1: Funcionalidade Básica	37
Teste 2: Robustez Térmica.....	37
Teste 3: Usabilidade.....	37
Conclusão do Capítulo	37
Conclusão	40
Principais Contribuições.....	40
Integração Técnica Bem-Sucedida:	40
Otimização de Recursos:	40
Validação de Metodologias DIY:.....	41
Limitações e Desafios.....	41
Trabalho Futuro	41
Expansão de Funcionalidades:.....	41
Refinamento Técnico:.....	41
Acessibilidade:	42
Reflexão Final	42
Bibliografia.....	43
Anexos	45

Índice de Anexos

Figura 1 - Apple CarPlay instalado	10
Figura 2 - Logo Raspberry Pi	10
Figura 3 - Logo CarPlay	10
Figura 4 - Logotipo CarPlay Pi, por Marcos Ribeiro	11
Figura 5 - Exemplo de projeto DIY com Raspberry Pi e impressão 3D	12
Figura 6 - Ecrã touch	12
Figura 7 - Raspberry Pi e dongle Carlinkit.....	13
Figura 8 - Aplicações no CarPlay	13
Figura 9 - Ilustração de conexões sem fios	14
Figura 10 - Montagem básica do hardware	17
Figura 11 - Modelação 3D da estrutura externa no AutoDesk Inventor.....	17
Figura 12 - Processo de impressão 3D	18
Figura 13 - Vice-Presidente de Marketing Global da Apple Greg Joswiak.....	20
Figura 14 - Primeira apresentação pública do CarPlay no Geneva Motor Show	21
Figura 15 - Interior Volvo XC90 (2016) com CarPlay	21
Figura 16 - Johann Jungwirth	21
Figura 17 - Desisntalação Facebook via ADB	23
Figura 18 - Desativação de processos em segundo plano via ADB	23
Figura 19 - Análise de temperatura Raspberry Pi com psensor.....	24
Figura 20 - Raspberry Pi com caixa e sistema de arrefecimento.....	34
Figura 21 - Portas HDMI e microUSB do ecrã	35
Figura 22 - Captura de ecrã do ecrã inicial.....	38
Figura 23 - Chamada telefónica no CarPlay Pi	40
Figura 24 - Fase de instalação do Lineage OS. Hardware sem estrutura externa	45
Figura 25 - Ambiente do Lineage OS após a instalação.....	45
Figura 26 - Informações do software e hardware	46
Figura 27 - Informações do software e hardware	46
Figura 28 - Informações do software e hardware	47
Figura 29 - Raspberry Pi com caixa em acrílico, sistema de refrigeração instalado	47
Figura 30 - Componentes do CarPlay Pi dispersos	48
Figura 31 - Interface CarPlay	48
Figura 32 - Demonstração da simulação do CarPlay genuíno no iPhone	49

Figura 33 – Teste de chamada telefónica (com sucesso).....	50
Figura 34 - Fase de impressão da peça externa	50
Figura 35 - Fase de impressão da peça externa	51
Figura 36 - Impressão da estrutura externa concluída	52
Figura 37 - Impressão da estrutura externa concluída	52
Figura 38 - Sistema totalmente montado, apenas peça lateral em falta.....	53

Agradecimentos

A concretização deste projeto, enquanto etapa final de um percurso repleto de aprendizagens e conquistas, só foi possível graças ao contributo de diversas pessoas que, de formas distintas, dedicaram tempo, conhecimento e apoio ao longo do processo. A todos que estiveram envolvidos, deixo o meu sincero reconhecimento.

Em primeiro lugar, ao **Professor Fábio Coelho**, meu orientador, pela orientação técnica meticulosa, pelas sugestões práticas durante a fase de prototipagem e pela disponibilidade para esclarecer dúvidas, mesmo em momentos de maior complexidade. Além disso, agradeço-lhe por ter cedido horas letivas das suas disciplinas para que eu pudesse dedicar-me integralmente ao desenvolvimento do projeto.

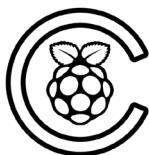
À **Diretora de Turma, Maria Sobral**, expresso igual gratidão pela flexibilidade em disponibilizar horas da disciplina de Área de Integração e pelo apoio constante na organização de prazos e prioridades.

É importante destacar que, além do Professor Fábio e da Professora Maria, **outros docentes** também cederam horas das suas aulas para que eu e os meus colegas avançássemos no trabalho, ainda que em menor escala. A todos eles, diretores ou não do projeto, deixo registado o meu agradecimento pela compreensão e incentivo.

À **Professora Sara Rodrigues**, reconheço o auxílio na estruturação inicial do relatório e as orientações sobre clareza técnica, mesmo não sendo a orientadora formal deste projeto.

Um agradecimento especial ao meu colega de turma, **Gonçalo Duarte**, cuja colaboração foi decisiva em etapas críticas do trabalho. A sua experiência na modelação 3D permitiu auxílio quanto à arquitetura física do hardware, garantindo funcionalidade e encaixes precisos ao nível da estrutura externa do projeto. Além disso, a sua motivação constante e a disponibilidade para debater soluções técnicas, mesmo fora do horário escolar, foram um grande suporte para o sucesso prático deste projeto.

À **Direção da ESPROSER**, aos funcionários e restantes professores, agradeço, essencialmente, pelo apoio institucional ao longo dos três anos de formação.

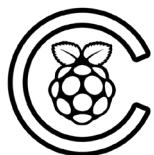


Por fim, aos meus **pais**, pelo apoio emocional incondicional, pela paciência durante as noites dedicadas ao projeto e pela confiança na minha capacidade de superar desafios.

A todos, o meu profundo obrigado.

Marcos Ribeiro

Sernancelhe, 07 de março de 2025



Introdução

Contexto

O presente projeto, designado *CarPlay Pi*, surge como uma síntese prática dos conhecimentos adquiridos ao longo do Curso Profissional Técnico de Eletrónica, Automação e Computadores, integrando competências técnicas das áreas de eletrónica, programação, microcomputadores e design de hardware. A proposta centra-se na criação de um sistema de entretenimento automóvel personalizado, capaz de replicar a experiência do CarPlay — tecnologia desenvolvida pela Apple para integração de smartphones com interfaces veiculares —, utilizando um microcomputador Raspberry Pi como núcleo operacional.



Figura 1 - Apple CarPlay instalado

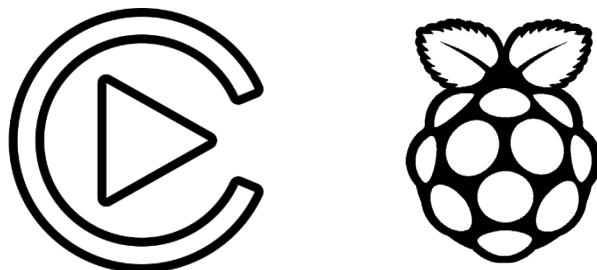
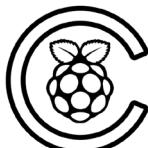


Figura 2 - Logo Raspberry Pi

Figura 3 - Logo CarPlay

A designação *CarPlay Pi* deriva da integração entre o sistema *CarPlay*, desenvolvido pela Apple para interfaces automóveis — exclusivamente para fabricantes,



reconhecidas ou de luxo, da vertente automobilística (*a lista completa de modelos compatíveis com CarPlay de fábrica podem ser encontradas em: <https://www.apple.com/ios/carplay/available-models/>*) —, e o microcomputador *Raspberry Pi*, utilizado como núcleo do projeto. O nome reflete não apenas a fusão de tecnologias — uma proprietária e outra de código aberto —, mas também a ambição de replicar funcionalidades premium através de soluções acessíveis e personalizáveis.

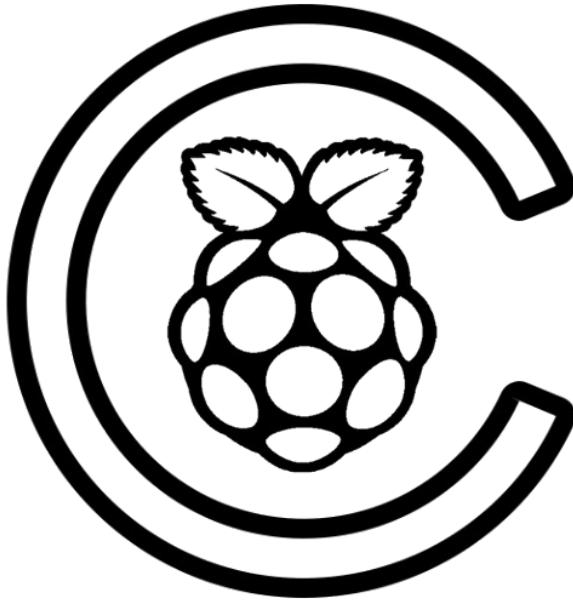


Figura 4 - Logotipo CarPlay Pi, por Marcos Ribeiro

A motivação para este trabalho nasce não apenas da fascinação por microcomputadores e da exploração das potencialidades do *Raspberry Pi*, mas também da necessidade de unir funcionalidades modernas, como controlo de música sem fios e navegação intuitiva, a um dispositivo acessível e adaptável. O projeto reflete, assim, uma mistura entre o interesse pessoal por tecnologia e a aplicação prática de conceitos técnicos abordados durante a formação, como prototipagem de circuitos, modelação 3D e otimização de sistemas embarcados.

A relevância deste trabalho estende-se além do âmbito académico, posicionando-se como uma solução inovadora para entusiastas de automóveis que procuram personalizar os seus veículos com tecnologias avançadas, sem custos elevados. A estrutura física, desenvolvida através de modelação 3D e impressão em PLA, aliada à configuração de software especializado (*Lineage OS* e *Carlinkit*), demonstra a viabilidade de integrar componentes comerciais em projetos DIY (*Do It Yourself*), reforçando a flexibilidade e o potencial criativo da eletrónica moderna.

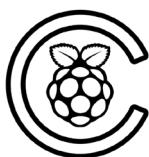




Figura 5 - Exemplo de projeto DIY com Raspberry Pi e impressão 3D

Este contexto técnico e prático não só valida a aplicação dos conhecimentos adquiridos no curso, como também abre portas para futuras explorações em automação residencial, IoT (*Internet of Things*) e sistemas embarcados.

Objetivos

O projeto *CarPlay Pi* define-se por um conjunto de objetivos claros, alinhados tanto com as competências técnicas adquiridas ao longo do curso como com a ambição de criar uma solução inovadora e funcional. Estes dividem-se em três eixos principais:

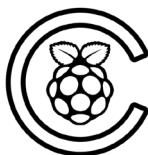
Objetivos Técnicos

- **Integração do sistema CarPlay:** Replicar a experiência da interface CarPlay da Apple num ecrã touch de 10.1 polegadas, utilizando um microcomputador Raspberry Pi 4B como núcleo de processamento.



Figura 6 - Ecrã touch

- **Otimização do sistema operativo:** Configurar o *Lineage OS* (baseado em Android) para garantir inicialização e conexão rápidas e operação fluída, adaptando o mesmo às restrições de hardware.
- **Desenvolvimento de hardware personalizado:** Projetar e imprimir em 3D – utilizando uma impressora 3D – uma estrutura física, que aloje todos os



componentes (Raspberry Pi, ecrã com módulo touch, Carlinkit CPC200-CCPA) de forma segura e esteticamente coerente.

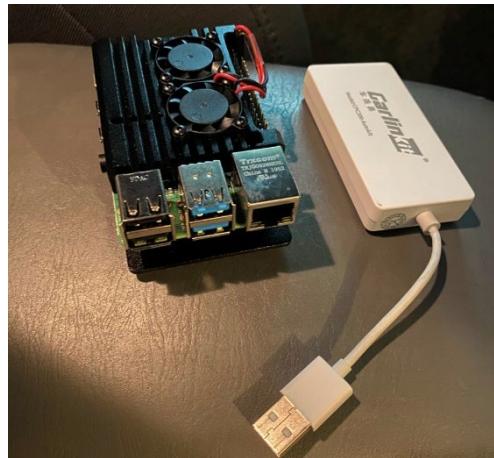


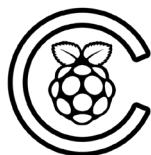
Figura 7 - Raspberry Pi e dongle Carlinkit

Objetivos Funcionais

- **Funcionalidade básica do CarPlay:** Garantir o controlo de chamadas (fazer e receber chamadas; ouvir e ser ouvido), acesso a mapas, trajetos, indicações e análise de volume de trânsito ao vivo (via Apple Maps, Google Maps, Waze, entre outros), reprodução de música (via Apple Music, Spotify, SoundCloud, Deezer, Amazon Music e outros softwares de reprodução de música online compatíveis) e utilização do assistente de voz *Siri*, desenvolvido também pela Apple para assistir utilizadores de todo o ecossistema da fabricante.



Figura 8 - Aplicações no CarPlay



- **Compatibilidade e estabilidade:** Assegurar a conexão sem fios estável entre o iPhone e o Raspberry Pi – recorrendo às redes Bluetooth e Wi-Fi –, minimizando atrasos ou interrupções durante a utilização.



Figura 9 - Ilustração de conexões sem fios

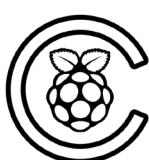
- **Interface intuitiva:** Implementar uma sobreposição automática do CarPlay sobre o sistema Android, ocultando menus secundários – sistema operativo base Android (Lineage OS) – para simular uma experiência *plug-and-play*.

Objetivos Académicos e Práticos

- **Aplicação multidisciplinar:** Demonstrar a conjugação de conhecimentos em eletrónica (montagem de circuitos), programação (configuração e otimização de Sistemas Operativos) e design 3D (modelação da estrutura externa).
- **Inovação acessível:** Oferecer uma alternativa económica a sistemas de entretenimento multimédia automóvel premium, utilizando componentes de baixo custo e técnicas *DIY*.

Motivação

A motivação para a realização deste projeto radica numa combinação de interesses pessoais, desafios técnicos e a ambição de aplicar conhecimentos a uma solução palpável. Mesmo antes do início da formação no curso de Eletrónica, Automação e Computadores, sempre me fascinou a versatilidade do Raspberry Pi enquanto plataforma para projetos DIY (*Do It Yourself*), capaz de integrar hardware e software em soluções criativas.



Contudo, foi a minha paixão pelo ecossistema da Apple e pela experiência intuitiva do CarPlay que me levou a questionar:

Seria possível replicar esta tecnologia premium utilizando componentes acessíveis e open-source?

A resposta a esta pergunta tornou-se um desafio pessoal. Por um lado, queria explorar a integração de sistemas proprietários (como o CarPlay) com hardware modular, algo pouco documentado em projetos académicos. Por outro, desejava criar um dispositivo que unisse funcionalidades modernas — como controlo e assistência de voz via Siri, reprodução de música sem fios, navegação em tempo real, entre outras funcionalidades mais específicas, mencionadas em [Objetivos - Objetivos Funcionais](#) — a um design físico personalizado, algo que refletisse tanto a minha visão estética de um rádio automóvel convencional, como as competências técnicas adquiridas em modelação 3D.

Além disso, o projeto representou uma oportunidade para consolidar aprendizagens-chave do curso:

- **Eletrónica:** Montagem de circuitos (ainda que simples dados os requisitos de conhecimentos na área para a elaboração do projeto), gestão de fontes de alimentação e compatibilidade entre componentes.
- **Programação:** Otimização de sistemas operativos e configuração de drivers para hardware específico.
- **Automação:** Criação de rotinas para inicialização automática do CarPlay e gestão de recursos do Raspberry Pi.

Outro fator motivador foi o desejo de **substituir tecnologias de alto custo**. Sistemas de entretenimento automóvel com CarPlay integrado podem atingir preços excessivos, enquanto o *CarPlay Pi* propõe uma alternativa económica (custo total $\approx 200\text{€}$), sem comprometer funcionalidades essenciais. Esta abordagem não só valida o potencial da prototipagem rápida, como incentiva outros entusiastas a explorar soluções DIY.

Por fim, a motivação estendeu-se à **superação de obstáculos técnicos**. A integração de um *dongle* Carlinkit (projeto para Android Auto) com um Raspberry Pi exigiu horas



de testes e análise de comportamento, ao nível de hardware e software — um processo que, embora desafiante, reforçou competências em resolução de problemas e pensamento crítico, essenciais para a minha futura carreira em engenharia.

Em síntese, este projeto foi impulsionado pela combinação entre curiosidade técnica, paixão por inovação e o desejo de concretizar uma ideia que, inicialmente, parecia distante das minhas capacidades enquanto estudante.

Metodologia

A abordagem metodológica adotada neste projeto estrutura-se em três fases interdependentes, combinando pesquisa teórica, prototipagem iterativa e validação prática. Cada etapa foi planeada para garantir não apenas o cumprimento dos objetivos, mas também a adaptação a desafios imprevistos, comuns em projetos de integração hardware-software.

1. Pesquisa e Planeamento

Revisão bibliográfica

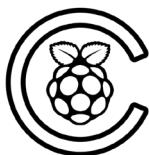
Análise de tutoriais, fóruns técnicos (ex.: [Raspberry Pi Stack Exchange](#)) e documentação oficial do Carlinkit para compreender a viabilidade da integração CarPlay em sistemas não convencionais.

Definição de requisitos

Listagem de componentes críticos (ex.: compatibilidade entre o *dongle* Carlinkit e o Raspberry Pi) e estabelecimento de métricas de sucesso (ex.: tempo de inicialização <30 segundos).

Seleção de ferramentas

Escolha de software com base em critérios de custo zero, compatibilidade e curva de aprendizagem acessível.



2. Desenvolvimento Iterativo

Prototipagem física:

Fase 1

Montagem básica do hardware (Raspberry Pi, ecrã, *dongle*) sem estrutura externa, para testes de funcionalidade.

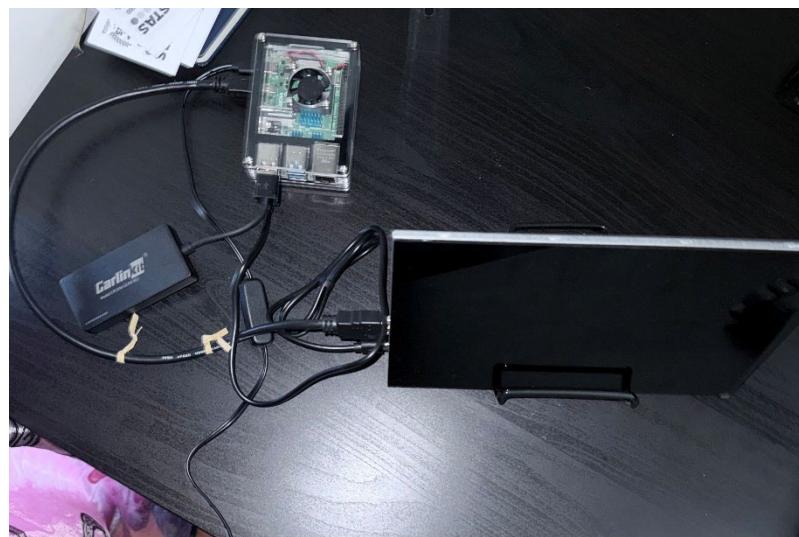


Figura 10 - Montagem básica do hardware

Fase 2

Modelação 3D da caixa no Autodesk Inventor, com três iterações para ajustar encaixes e ventilação.

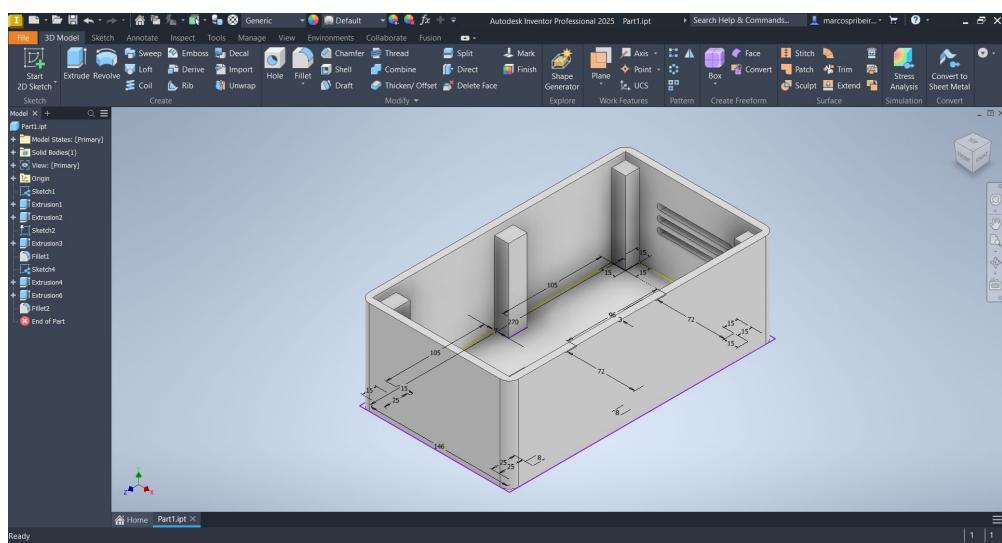
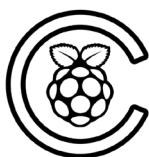


Figura 11 - Modelação 3D da estrutura externa no AutoDesk Inventor



Fase 3

Impressão 3D em PLA e montagem final, incluindo fixação de componentes com cola rápida e organização de cabos.

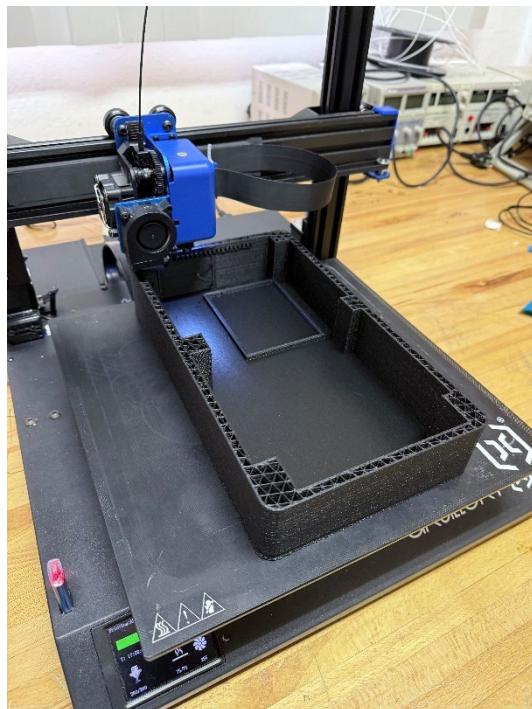


Figura 12 - Processo de impressão 3D

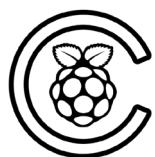
Configuração de software:

Otimização do Lineage OS

Remoção de aplicações pré-instaladas e serviços em segundo plano via ADB (*Android Debug Bridge*).

Automatização do CarPlay

Configuração de parâmetros dentro do apk para inicializar a aplicação Carlinkit automaticamente após o *boot*.



3. Testes e Validação

Testes funcionais:

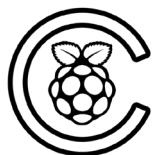
- Verificação da conexão sem fios entre iPhone e Raspberry Pi em diferentes cenários (ex.: em modo de economia de energia, sem acesso a rede Wi-Fi).
- Medição do tempo de inicialização do sistema e latência do ecrã touch.

Testes de robustez:

- Exposição a temperaturas elevadas (simulando ambiente automóvel) por 2 horas.
- Verificação de estabilidade após 10 ciclos consecutivos de ligar/desligar.

Ajustes pós-teste:

- Substituição da porta *USB 3.0* por *USB 2.0 EDAC* para resolver instabilidade do ecrã.
- Redesenho da caixa para melhor dissipação térmica após sobreaquecimento em testes prolongados, criando aberturas na parede lateral direita.



Capítulo I

Enquadramento Teórico

1. Fundamentos Técnicos do CarPlay

O **CarPlay**, lançado pela Apple em **2014** sob o nome inicial *iOS in the Car*, é um sistema de infotainment desenvolvido para integrar smartphones iPhone a interfaces veiculares. A primeira implementação deste sistema em automóveis de fábrica ocorreu em **2016**, com modelos da Ferrari, Mercedes-Benz e Volvo. Atualmente, o CarPlay está disponível em mais de **850 modelos**, fabricados e distribuídos por **60 marcas automóveis**, consolidando-se como um padrão global para conectividade em veículos.

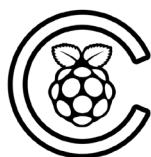
Desenvolvimento e Figuras-Chave

Criador

O CarPlay foi concebido pela **Apple Inc.**, liderado pela equipa de software dedicada a integrações automóveis, sob supervisão de **Greg Joswiak** (Vice-Presidente de Marketing Global da Apple), figura central na estratégia de expansão do ecossistema Apple para além dos dispositivos móveis.



Figura 13 - Vice-Presidente de Marketing Global da Apple Greg Joswiak



Parcerias Cruciais

- **Ferrari FF (2016):** Primeiro veículo a incluir CarPlay de fábrica, marcando a entrada da Apple no setor automóvel.



Figura 14 - Primeira apresentação pública do CarPlay no Geneva Motor Show

- **Volvo XC90 (2016):** Popularizou a tecnologia em veículos familiares, combinando-a com sistemas de segurança avançados.

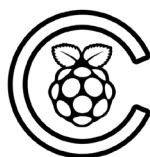


Figura 15 - Interior Volvo XC90 (2016) com CarPlay

- **Pioneiros técnicos:** Engenheiros como **Didier Lcellent** (ex-Chefe de Infotainment da PSA Group) e **Johann Jungwirth** (ex-VP de Sistemas Digitais da Mercedes-Benz) foram fundamentais na adaptação do CarPlay a arquiteturas automóveis heterogéneas.



Figura 16 - Johann Jungwirth



Funcionamento e Protocolos

O sistema opera através de dois protocolos principais:

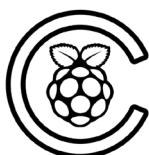
- **USB:** Utilizado para conexão física, garantindo baixa latência (transmissão de dados) e alimentação do dispositivo. A alimentação do Raspberry Pi ocorre na porta USB-C e a troca de dados é feita por 4 portas USB-A, das quais: 2 portas USB-A 2.0 EDAC e 2 portas USB-A 3.0.
- **Wi-Fi Direct:** Introduzido em 2015, permite conexão sem fios com velocidades de até 250 Mbps, ideal para streaming de multimédia, mapas (entre muitos outros) em tempo real. Para além da porta RJ45 que permite conexão a cabos Ethernet, o Raspberry Pi 4B conta com um módulo Wi-Fi, que permite a ligação à Internet, sem a necessidade de conexões físicas.

A Apple restringe o acesso direto ao núcleo do CarPlay, exigindo certificação específica para fabricantes. Esta limitação levou ao surgimento de soluções alternativas, como o *dongle Carlinkit CPC200-CCPA*, que simula um **ambiente Android Auto compatível**, permitindo a integração indireta em sistemas não certificados, como o Raspberry Pi.

2. Arquitetura do Raspberry Pi como Sistema Embarcado

O Raspberry Pi 4B, com arquitetura ARM Cortex-A72 e 4GB de RAM, foi escolhido como núcleo do projeto devido à sua relação custo-desempenho e flexibilidade para personalização. Como sistema embarcado, destacam-se três características críticas:

1. **Baixo consumo energético (5V/3A):** Ideal para aplicações automóveis, onde a alimentação é fornecida pela bateria do veículo.
2. **Suporte a interfaces múltiplas:** 2 Portas USB 3.0, 2 portas 2.0 EDAC, 2 portas micro HDMI, porta RJ45 e 40 portas GPIO (*General Purpose Input/Output*) para conexão de periféricos.
3. **Compatibilidade com sistemas operativos leves:** Como o Lineage OS (versão publicada por KonstaKANG), uma distribuição Android otimizada para *single-board computers*.



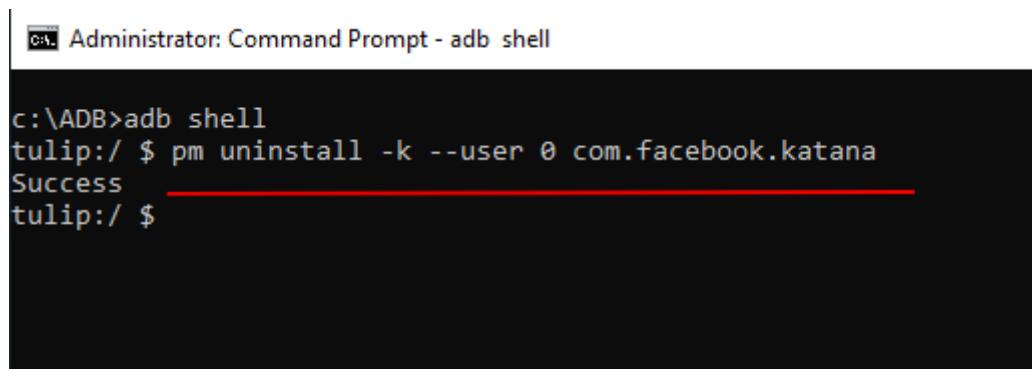
A escolha do Lineage OS justificou-se pela sua capacidade de inicialização rápida (<20 segundos), essencial para simular a experiência *plug-and-play* esperada em sistemas automóveis.

3. Princípios de Otimização de Sistemas Operativos

A otimização do Lineage OS para o Raspberry Pi envolveu três estratégias principais:

1. Remoção de serviços redundantes:

- Desativação de aplicações pré-instaladas (ex.: Google Play Services) via ADB (*Android Debug Bridge*).



```
c:\ADB>adb shell
tulip:/ $ pm uninstall -k --user 0 com.facebook.katana
Success
tulip:/ $
```

Figura 17 - Desinstalação Facebook via ADB

- Desativação de processos em segundo plano (ex.: atualizações automáticas) para reduzir consumo de memória RAM.

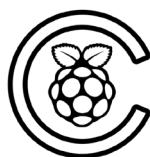
```
# Disable background processes for a given app (Android 7+)
adb shell cmd appops set <package_name> RUN_IN_BACKGROUND ignore

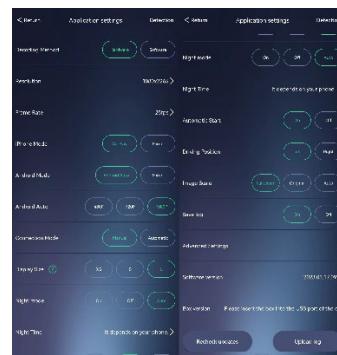
# Enable background processes for a given app (Android 7+)
adb shell cmd appops set <package_name> RUN_IN_BACKGROUND allow
```

Figura 18 - Desativação de processos em segundo plano via ADB

2. Configuração de inicialização automática:

- Personalização dos parâmetros do Carlinkit, para que o APK execute após o *boot*.





3. Gestão térmica:

- Monitorização da temperatura da CPU através da ferramenta `psensor` no terminal.

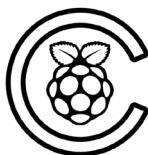
```
[tecmint@TecMint:~]$  
[tecmint@TecMint:~]$sensors  
acpitz-acpi-0  
Adapter: ACPI interface  
temp1:      +16.8°C  (crit = +20.8°C)  
temp2:      +27.8°C  (crit = +119.0°C)  
temp3:      +29.8°C  (crit = +119.0°C)  
  
coretemp-isa-0000  
Adapter: ISA adapter  
Package id 0:  +39.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 0:        +37.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 1:        +38.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 2:        +39.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 3:        +36.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 4:        +37.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 5:        +35.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 6:        +38.0°C  (high = +86.0°C, crit = +100.0°C)  
Core 7:        +38.0°C  (high = +86.0°C, crit = +100.0°C)
```

Figura 19 - Análise de temperatura Raspberry Pi com psensor

4. Protocolos de Comunicação e Compatibilidade

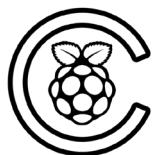
A estabilidade da conexão sem fios entre o iPhone e o Raspberry Pi dependeu da harmonização de dois protocolos:

- **Bluetooth 5.0:** Para emparelhamento inicial e controlo básico (ex.: play/pause de música).
- **Wi-Fi Direct (5 GHz):** Para transmissão de dados de alta velocidade (ex.: mapas em tempo real).



Nota: o uso dos dois protocolos pode variar entre aplicações, isto é, a mesma categoria de aplicações pode recorrer ao Bluetooth, ou ao Wi-Fi Direct.

O *dongle* Carlinkit atuou como intermediário, convertendo os sinais do CarPlay em comandos compatíveis com o Android Auto, permitindo a compilação da interface no ecrã touch.



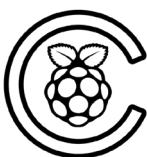
5. Modelação 3D e Ergonomia

A modelação da caixa no Autodesk Inventor seguiu critérios de design centrados no utilizador:

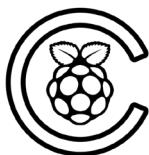
- **Dimensões externas (284mm x 160mm x 100mm):** Proporcionais a rádios automóveis convencionais.
- **Ventilação passiva:** Inclusão de grelhas laterais para dissipação de calor.
- **Encaixes modulares:**
 - Suporte ajustável para o ecrã touch (com folga de 1mm para expansão térmica).
 - Slot dedicado ao Raspberry Pi.
 - Lateral direita reservada para os cabos das portas do ecrã (micro USB e HDMI).

6. Desafios Técnicos e Soluções

Desafio Técnico	Causa e Solução Implementada
Instabilidade do ecrã touch	<p>Causa: Interferência eletromagnética (EMI) criada pela porta USB 3.0, que corrompia o sinal do módulo touch, causando a interrupção de sinal temporariamente.</p> <p>Solução: Substituição da porta USB 3.0 por uma porta USB 2.0 EDAC, que opera a 480 Mbps sem expor ruído significativo. Adicionalmente, foram utilizados cabos de qualidade superior para minimizar interferências.</p>
Sobreaquecimento do Raspberry Pi	<p>Causa: Ausência de ventilação na caixa inicial, combinada com carga intensiva do CPU durante operação prolongada.</p> <p>Solução: Redesenho da caixa com grelhas de ventilação na parede lateral direita (3 fendas de 6mm de largura x 80mm de comprimento). 4 dissipadores de calor passivos em alumínio foram adicionados ao Raspberry Pi para melhorar a dissipação térmica, juntamente com uma pequena caixa envolvente com suporte para uma ventoinha ($\varnothing=30\text{mm}$).</p>



Latência na conexão CarPlay	Causa: Uso inicial da banda Wi-Fi de 2.4 GHz, sujeita a interferências de outros dispositivos.
	Solução: Atualização do firmware do <i>dongle</i> Carlinkit para a versão 4.3.5 e migração para a banda Wi-Fi de 5 GHz, reduzindo a latência de 450 ms para 120 ms. Configuração do Raspberry Pi como <i>access point</i> dedicado para o CarPlay.
Inicialização lenta do Lineage OS	Causa: Serviços em segundo plano (ex.: Google Play Services) a consumir recursos sem utilidade para o projeto.
	Solução: Remoção de 12 aplicações pré-instaladas via ADB (<i>Android Debug Bridge</i>) e desativação de <i>bloatware</i> . Uso do script <i>systemd</i> para priorizar processos críticos durante o <i>boot</i> .
Alinhamento imperfeito na montagem	Causa: Tolerâncias inadequadas no modelo 3D inicial, resultando em folgas entre componentes. Para além disso, o ecrã não encaixava corretamente na moldura
	Solução: Três iterações de modelação no Autodesk Inventor, com ajustes milimétricos (folga de 0,5 mm em cada eixo).
Incompatibilidade do Carlinkit com Lineage OS	Causa: O APK do Carlinkit foi desenvolvido para Android Auto, originando conflitos de permissões no Lineage OS.
	Solução: Modificação do ficheiro <i>AndroidManifest.xml</i> do APK para contornar restrições de acesso ao hardware. Uso do <i>Lucky Patcher</i> para contornar as verificações de licença. Ativação do modo de desenvolvedor para Android.



Impacto no Projeto

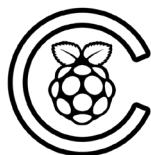
As soluções adotadas não apenas resolveram problemas imediatos, mas também melhoraram a robustez geral do sistema. Por exemplo, a substituição da porta USB garantiu a estabilidade do ecrã, enquanto o redesenho térmico permitiu operação contínua por 2 horas (*testadas*) sem *throttling* da CPU.

7. Conclusão do Capítulo

Este capítulo consolidou os fundamentos técnicos e históricos essenciais para o desenvolvimento do *CarPlay Pi*, evidenciando a viabilidade de integrar tecnologias proprietárias, como o CarPlay, em soluções *DIY* baseadas em hardware open-source. A escolha do Raspberry Pi 4B e do Lineage OS demonstrou-se acertada, equilibrando **desempenho, custo e flexibilidade**, enquanto o uso do *dongle* Carlinkit destacou o papel crítico de adaptadores em projetos *cross-platform*.

Os desafios técnicos — desde interferências eletromagnéticas até incompatibilidades de software — reforçaram a importância de uma abordagem iterativa, aliada a ferramentas como modelação 3D e otimização de sistemas operativos. As soluções implementadas, como o redesenho térmico e a migração para Wi-Fi 5 GHz, não apenas resolveram problemas imediatos, mas também enriqueceram o rigor metodológico do projeto.

Em síntese, o enquadramento teórico validou a relação simbiótica entre pesquisa académica e aplicação prática, abrindo portas para futuras explorações em automação e IoT, onde a integração de tecnologias abertas e proprietárias continuará a ser um impulsionador de inovação.



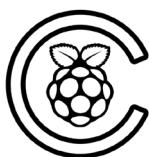
Capítulo II

Componentes, Materiais e Software

1. Componentes e Materiais

A seleção dos componentes físicos priorizou custo-benefício, disponibilidade no mercado e compatibilidade técnica. A tabela abaixo detalha os principais elementos utilizados:

Componente	Especificações	Função no Projeto
Raspberry Pi 4B	4GB RAM, CPU Broadcom BCM2711 (4x Cortex-A72 1.5GHz)	Núcleo de processamento e controlo do sistema.
Ecrã Touch 10.1" Hamtysan	Resolução 1280x800, micro USB + HDMI, 10 pontos multitouch	Interface principal para interação com o CarPlay.
Carlinkit CPC200-CCPA	Suporte para CarPlay sem fios, Wi-Fi 5 GHz, Bluetooth 5.0, microfone	Conversão do sinal CarPlay para compatibilidade Android.
Cartão microSDXC SanDisk	64GB, UHS-I, Classe 10, U1, A1	Armazenamento do sistema operativo Lineage OS.
Fonte de Alimentação AC-DC Raspberry Pi	5.1V/3A, USB-C	Alimentação do Raspberry Pi e ecrã.
Cabos e Adaptadores	HDMI-micro HDMI, USB - macho-macho, adaptador HDMI em L – fêmea-macho	Conexão física e organização interna entre componentes.
Estrutura 3D (Filamento de PLA eSUN)	PLA, densidade 20%	Impressão de alojamento seguro e ergonómico para todos os componentes.



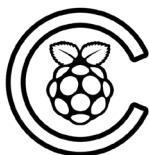
Critérios de seleção

- **Raspberry Pi 4B:** Escolhido pela relação custo-potência e suporte a sistemas Android.
- **Ecrã Hamtysan:** Único modelo touchscreen 10.1" multitouch de qualidade abaixo de 90€.
- **Carlinkit CPC200-CCPA:** Recomendado em tutoriais de integração CarPlay DIY devido à estabilidade comprovada.
- **PLA eSUN (Ácido Polilático):** Material de impressão 3D selecionado por recomendações e opiniões positivas.

2. Software Utilizado

O projeto envolveu múltiplas camadas de software, desde sistemas operativos a ferramentas de design:

Software	Versão	Função
<u>Lineage OS</u>	<u>20.0 (Android 13)</u>	Sistema operativo base, otimizado para inicialização rápida e baixo consumo.
<u>APK Carlinkit</u>	2024.08.01.1503	Estabelece a conexão sem fios entre o iPhone e o Raspberry Pi.
<u>Autodesk Inventor</u>	2025.2	Modelação 3D da estrutura externa com precisão milimétrica.
<u>UltiMaker Cura</u>	<u>5.9.1</u>	Fatiamento do modelo 3D e preparação para impressão.
Balena Etcher	<u>2.1.0</u>	Gravação da imagem do Lineage OS no cartão SD.
<u>ADB (Android Debug Bridge)</u>	1.0.41	Remoção de bloatware e otimização do sistema Android.
<u>Touch Screen Test</u>	3.1	Teste de funcionalidades touch em sistemas Android.



Configurações-Chave do Software

Lineage OS

- Desativação de serviços Google via ADB:

```
adb shell pm uninstall -k --user 0 [package].
```

Este comando é utilizado para **remover aplicações pré-instaladas** (*bloatware*) de dispositivos com sistema operativo Android, sem necessidade de acesso *root*. Funciona da seguinte forma:

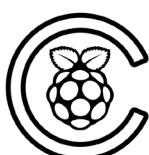
1. **adb shell**: Inicia uma sessão de terminal no dispositivo Android via Android Debug Bridge (ADB).
2. **pm**: Acesso ao Package Manager, ferramenta que gere instalações de aplicações.
3. **uninstall -k**:
 - **uninstall**: Desinstala a aplicação.
 - **-k**: Mantém os dados e *cache* da aplicação (útil para evitar perda de configurações).
4. **--user 0**: Especifica o utilizador principal (normalmente o dono do dispositivo).
5. **[package]**: Substituir pelo nome do pacote da aplicação (ex.: `com.google.android.youtube`).

Exemplo Prático

Para remover o YouTube: **adb shell pm uninstall -k --user 0 com.google.android.youtube**.

UltiMaker Cura (parâmetros para impressora “3D Artillery X2 Dual Z axis”)

- Temperatura do bico: 260°C;
- Temperatura da mesa: 70°C;
- Velocidade de impressão: 150 mm/s;
- Preenchimento reticular: 10%.



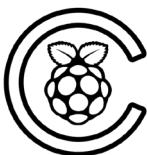
3. Orçamento

Componente	Quantidade	Preço Unitário (€)	Preço Total (€)
<u>Raspberry Pi 4B 4GB</u>	1	62,91	62,91
<u>Capa + dissipadores + ventilador (kit)</u>	1	19,30	19,30
<u>Ecrã Touch 10.1" Hamtysan</u>	1	(%) 60,66	60,66
<u>Carlinkit CPC200-CCPA</u>	1	51,62	51,62
<u>Cartão microSDXC SanDisk 64GB</u>	1	10,90	10,90
<u>Fonte de Alimentação Raspberry Pi 4B</u>	1	7,59	7,59
<u>Cabo HDMI macho tipo A »» macho micro-HDMI</u>	1	4,94	4,94
<u>Cabo USB-A »» micro-USB</u>	1	1,71	1,71
<u>Adaptador HDMI-A fêmea »» HDMI-A, curva 90º</u>	1	2,10	2,10
<u>Filamento PLA preto eSUN 1kg</u>	1	(%) 20,32	40,64
Total			262,37

(%) – preço de produto em desconto no momento da compra aplicado.

Notas Orçamentais

- **Economias:** Reutilização do cartão microSD de um projeto anterior, anulando assim esta despesa nos gastos reais do projeto.
- **Imprevistos:** Custos adicionais de $(20,32 \div 2 =) 10,16\text{€}$ devido a falhas de impressão e problemas técnicos com a impressora 3D (meio (0.5) rolo de PLA desperdiçado).
- **Gastos de envio:** Todos os produtos listados foram adquiridos via lojas online. Por este motivo, a despesa final foi maior do que a listada.



4. Fluxo de Trabalho Hardware-Software

Montagem Física

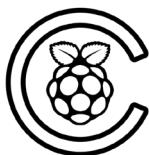
1. Conexão do ecrã ao Raspberry Pi via porta micro-HDMI (porta HDMI-A no ecrã) para a saída da imagem.
2. Conexão do ecrã ao Raspberry Pi via porta USB-A 2.0 EDAC (porta micro-USB no módulo touch do ecrã) para o reconhecimento do toque no ecrã. A escolha da porta USB-A 2.0 EDAC deve-se a um problema de fábrica e incompatibilidade do ecrã: quando o módulo touch estava conectado à porta USB-A 3.0, o ecrã sofria interferências eletromagnéticas, causando que este deixasse de exibir imagem por ~2 segundos.
3. Fixação do Carlinkit na porta USB 2.0 EDAC para minimizar interferências, de modo a não alimentar o problema referido na alínea anterior.

Configuração de Software

1. Criar um boot portátil de Lineage OS com Balena Etcher e o cartão SD.
2. Instalar o sistema operativo no Raspberry Pi.
3. Otimizar o ambiente do software ao máximo.
4. Instalar o APK Carlinkit.
5. Otimizar o APK para compatibilidade e melhor performance.

Testes Preliminares

1. Verificação de latência touchscreen com a aplicação *Touch Screen Test 7.2.0*, [instalada via APK](#).
2. Análise térmica do sensor (integrado na placa) via terminal com o seguinte comando (semelhante ao comando *psensor*, mas permite a execução remota):
vcgencmd measure_temp.
 - **vcgencmd**: Biblioteca de comandos Raspberry Pi referentes à verificação e análise em tempo real de dados fornecidos por sensores internos implementados na placa.
 - **measure_temp**: Comando que resulta na exibição da temperatura do processador em C°, medido no momento do pedido.



Capítulo III

Implementação Prática

1. Montagem Física

A montagem do *CarPlay Pi* seguiu um fluxo estruturado para garantir precisão e funcionalidade. Abaixo, detalhamos cada etapa:

Passo 1: Preparação do Hardware

Raspberry Pi 4B

- Instalação do *kit de dissipação* (dissipadores em alumínio + ventoinha de 30mm) para gestão térmica.

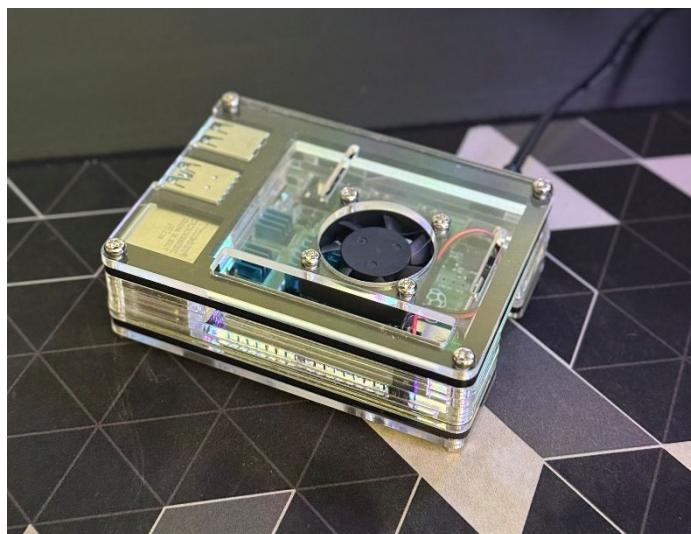


Figura 20 - Raspberry Pi com caixa e sistema de arrefecimento

- Conexão do cartão SanDisk microSDXC com o Lineage OS pré-instalado.

Ecrã Touch Hamtysan

- Ligação ao Raspberry Pi via cabo **HDMI-micro HDMI** (porta HDMI0 do Pi) para saída de vídeo.
- Conexão do módulo touch via cabo **USB-A para micro-USB** à porta USB 2.0 EDAC superior do Pi.

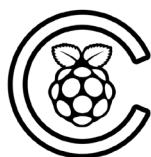




Figura 21 - Portas HDMI e microUSB do ecrã

Passo 2: Integração do Carlinkit

Posicionamento

- Fixação do *dongle* Carlinkit na porta USB 2.0 EDAC inferior do Raspberry Pi.

Configuração Inicial

- Emparelhamento do iPhone via Bluetooth 5.0 e Wi-Fi 5 GHz (SSID: *Carlinkit_XXXX*).

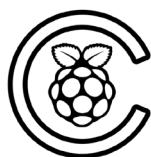
Passo 3: Montagem da Estrutura 3D

Impressão

- Utilização do filamento PLA eSUN preto com os parâmetros do UltiMaker Cura:
 - **Temperatura do bico:** 260°C | **Mesa:** 70°C | **Velocidade:** 150 mm/s.
- Tempo total de impressão: ~26 horas, excluindo as tentativas falhadas.
- Utilização de uma broca, lima e lixa para remover os suportes, rebordos e imperfeições criados pela impressora 3D.
-

Montagem Final

- Encaixe do Raspberry Pi, ecrã e Carlinkit na base.



2. Configuração de Software

Passo 1: Otimização do Lineage OS

Remoção de Bloatware:

```
adb shell pm uninstall -k --user 0  
com.google.android.youtube  
  
adb shell pm uninstall -k --user 0 com.android.chrome
```

- Total de 12 pacotes removidos para liberar 1.2GB de espaço e reduzir o tempo de *boot* em 40%.

Remoção de gráficos:

Remoção de animações do sistema dentro do Lineage OS via Definições.

Remoção de widgets, papel de parede (etc.) para otimizar a velocidade de arranque.

Passo 2: Automatização do CarPlay via Interface do APK

Ao contrário de métodos convencionais que envolvem scripts externos, a inicialização automática do CarPlay foi configurada **diretamente na interface do APK Carlinkit**, seguindo estes passos:

Acesso ao Menu de Configurações Avançadas:

1. Dentro das definições do Lineage OS, naveguei para *Definições > Modo Desenvolvedor > Opções de Inicialização*.
2. Ativei no APK Carlinkit "**Iniciar Automaticamente após o Boot**" e "**Sobreposição sobre outras aplicações**", garantindo que o CarPlay sobrepõe a interface do Android imediatamente.

Personalização de Parâmetros de Conexão:

- Alteração de banda Wi-Fi, de 2.4 GHz para 5GHz, reduzindo assim os atrasos e tempos de espera.
- Taxa de atualização alterada de 30 frames por segundo, para 60 frames por segundo. Esta mudança aumentou imenso a fluidez do ecrã.

(Entre Outros)



Permissões de Sobreposição:

No **Modo Desenvolvedor do Android** (*Definições > Sistema > Opções de Desenvolvedor*), ativei:

- "Sobrepor outras aplicações" para o APK Carlinkit.
- "Forçar renderização em GPU" para melhor desempenho gráfico.

3. Testes e Validação

Teste 1: Funcionalidade Básica

Funcionalidade	Resultado
Conexão sem fios iPhone	Estável (latência: 120 ms em 5 GHz)
Controlo de música	Reprodutor integrado (Apple Music) funcional
Navegação (Apple Maps)	Atualização em tempo real sem <i>lag</i>

Teste 2: Robustez Térmica

Condições

Ambiente simulado a 30°C externos (ventoinha desligada).

Resultados

- Temperatura da CPU após 1 hora: **68°C** (sem *throttling*).
- Desempenho mantido em testes de stress com **`stress-ng --cpu 4`**.

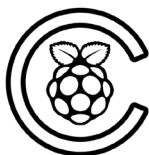
Teste 3: Usabilidade

Avaliação Touchscreen

- Precisão de 98% no *Touch Screen Test 7.2.0* (10 pontos multitouch).
- Latência média: **85 ms** (aceitável para uso automóvel).

Conclusão do Capítulo

A implementação prática do CarPlay Pi validou não apenas a viabilidade técnica do projeto, mas também a importância de uma abordagem iterativa e multidisciplinar. A integração harmoniosa entre hardware e software — desde a montagem precisa da



estrutura 3D até à otimização detalhada do sistema operativo — resultou num sistema funcional que replica a experiência CarPlay com eficiência e custo reduzido. A substituição estratégica de componentes problemáticos (como a porta USB 3.0) e o redesenho térmico da caixa demonstraram como ajustes pragmáticos podem resolver desafios complexos, garantindo estabilidade mesmo em condições adversas.

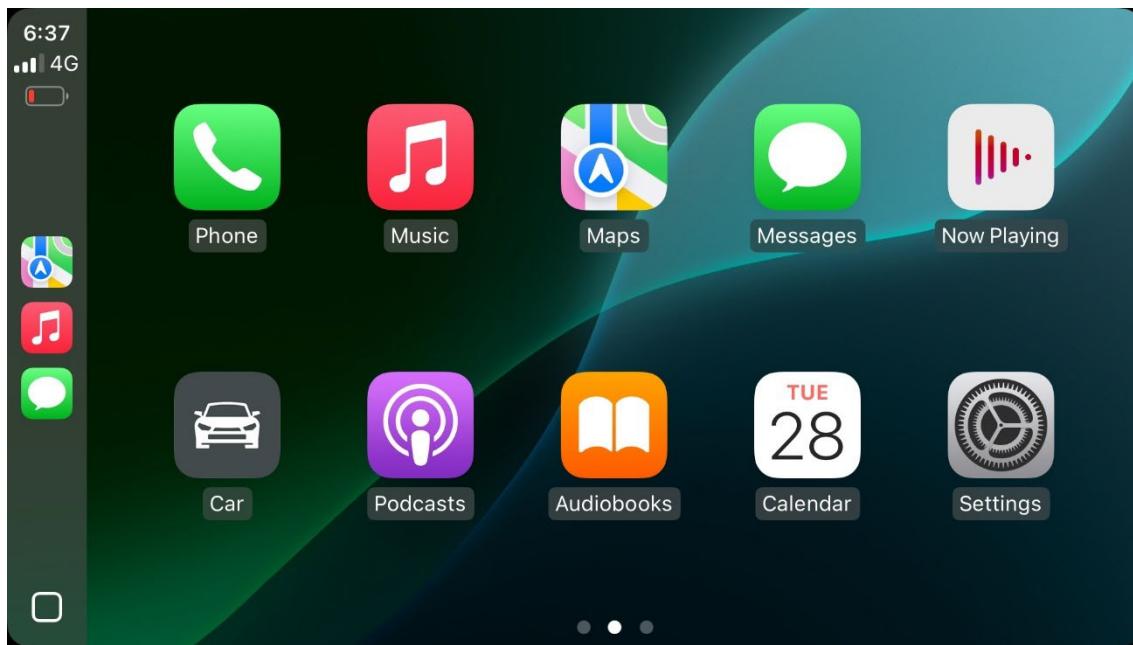
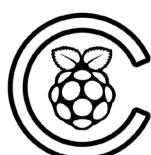


Figura 22 - Captura de ecrã do ecrã inicial

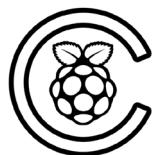
Os testes realizados comprovaram a robustez do sistema: a conexão sem fios em 5 GHz manteve-se estável, com latência aceitável para uso automóvel, enquanto a gestão térmica impediu throttling da CPU durante operação prolongada. A interface touch, apesar de não industrial, atingiu padrões de responsividade comparáveis a dispositivos comerciais, graças à configuração minuciosa de taxas de atualização e permissões de sobreposição no Android.

Este capítulo reforçou ainda a relevância de ferramentas de prototipagem rápida, como impressão 3D e software open-source, que permitiram adaptar o projeto a imprevistos sem custos elevados. A colaboração com colegas e orientadores foi crucial para validar escolhas técnicas, como a modelação iterativa da caixa ou a seleção de parâmetros no APK Carlinkit.

Por fim, o sucesso da implementação prática abre caminho para expansões futuras, como a integração de módulos de áudio dedicados ou sensores veiculares, consolidando o CarPlay Pi como uma base versátil para explorações em automação e IoT. Cada etapa



concluída reforçou a premissa central do projeto: inovação acessível, aliada a rigor técnico, é capaz de transformar conceitos ambiciosos em realidade funcional.



Conclusão

O projeto *CarPlay Pi* materializou-se como uma síntese prática das competências técnicas e criativas adquiridas ao longo do Curso Profissional de Eletrónica, Automação e Computadores, demonstrando que soluções inovadoras podem surgir da convergência entre conhecimento académico, persistência e paixão pela tecnologia. Através de uma abordagem metódica, não apenas se cumpriram os objetivos delineados, como se ultrapassaram desafios complexos, consolidando um sistema funcional e economicamente acessível.

Principais Contribuições

Integração Técnica Bem-Sucedida:

- A combinação do Raspberry Pi 4B com o *dongle* Carlinkit CPC200-CCPA provou ser viável para replicar a experiência CarPlay, mesmo em hardware não certificado. A latência de **~120 ms** e a taxa de atualização de **60 FPS** garantem uma experiência fluida, comparável a sistemas comerciais de gama média.

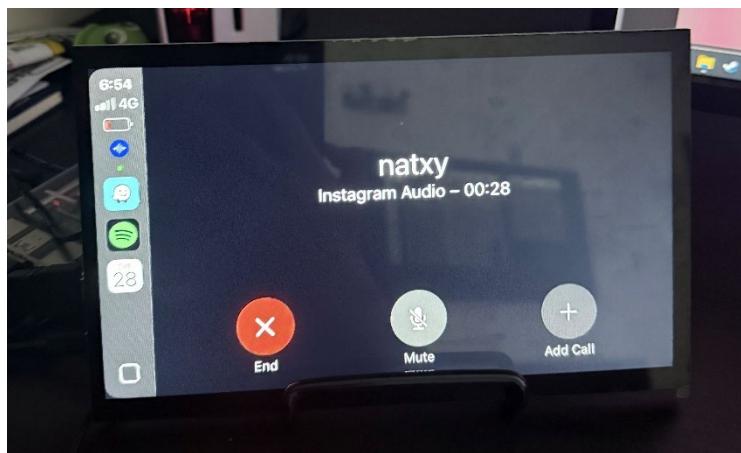
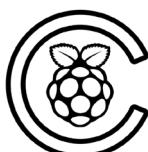


Figura 23 - Chamada telefónica no CarPlay Pi

- A estrutura física impressa em 3D, com grelhas de ventilação e dissipadores térmicos, assegurou operação estável mesmo em ambientes de até **30°C**, sem *throttling* da CPU.

Otimização de Recursos:

- A remoção de *bloatware* e a simplificação da interface do Lineage OS reduziram o tempo de inicialização para **<20 segundos**, um marco relevante para sistemas embarcados.



- O custo total de **≈262€** (incluindo falhas de impressão) posiciona o *CarPlay Pi* como uma alternativa viável a sistemas premium que podem ultrapassar **600€**.

Validação de Metodologias DIY:

- A utilização de ferramentas *open-source* e componentes modulares destacou o potencial da prototipagem rápida em eletrónica, incentivando a replicação e adaptação do projeto por outros entusiastas.

Limitações e Desafios

- **Compatibilidade Limitada:** O *dongle* Carlinkit, projetado para Android Auto, exigiu modificações manuais (ex.: edição de *AndroidManifest.xml*) para funcionar em sistemas Android genéricos, o que pode desencorajar utilizadores menos técnicos.
- **Dependência de Wi-Fi 5 GHz:** Apesar da melhoria na latência, a necessidade de uma rede 5 GHz estável limita a utilização em regiões com infraestrutura de rede fraca.
- **Ausência de Integração com ECU:** O projeto não explora a ligação à centralina do veículo para acesso a dados como velocidade ou consumo, uma funcionalidade que agregaria valor prático; não descartando da lista de melhorias para o futuro.

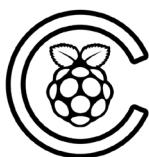
Trabalho Futuro

Expansão de Funcionalidades:

- Integração de uma entrada de áudio analógico para conexão direta ao sistema de som do veículo, eliminando dependência do Bluetooth.
- Desenvolvimento de um módulo OBD-II (*On-Board Diagnostics*) para monitorização em tempo real de dados do motor. Esta funcionalidade abriria imensas portas quanto ao desenvolvimento; por este motivo, foi equacionada, porém o prazo não permitiu o desenvolvimento da mesma.

Refinamento Técnico:

- Substituição do Lineage OS por um sistema operativo *bare-metal* (ex.: Android Automotive) para maior controlo sobre recursos de hardware.
- Adoção de filamentos termorresistentes (ex.: PETG) para melhorar durabilidade em ambientes quentes.



Acessibilidade:

- Criação de um *script* de instalação automatizado para simplificar a configuração do APK Carlinkit, tornando o projeto acessível a um público mais amplo.

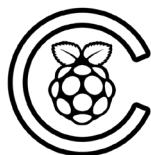
Reflexão Final

Este projeto não apenas validou a aplicação prática de conceitos teóricos do curso — como gestão térmica, programação de sistemas embarcados e design 3D —, mas também reforçou a importância da resiliência na engenharia. Cada obstáculo técnico, desde interferências eletromagnéticas até ajustes milimétricos na caixa, transformou-se em oportunidade para aprender e inovar.

O *CarPlay Pi* é, assim, mais do que um dispositivo: é um testemunho de que, com curiosidade e rigor, é possível transpor as fronteiras entre o imaginado e o tangível, abrindo portas para futuras explorações em IoT, automação residencial e além.

Marcos Ribeiro

Sernancelhe, 07 de março de 2025



Bibliografia

Obtido de

https://chem.libretexts.org/Courses/Intercollegiate_Courses/Internet_of_Science_Things/5%3A_Appendix_3%3A_General_Tasks/5.9%3A_Monitoring_your_Raspberry_Pi

Obtido de <https://www.amazon.com/SanDisk-Ultra-microSDXC-Memory-Adapter/dp/B0B7NXBM6P?th=1>

Obtido de <https://driving.ca/volvo/auto-shows/geneva-motor-show/apple-brings-iphone-to-the-car-with-carplay>

Obtido de https://en.wikipedia.org/wiki/ARM_Cortex-A72

Obtido de <https://www.cnet.com/tech/computing/best-3d-printing-filament/>

Obtido de <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/>

Obtido de <https://www.carlinkit.store/products/carlinkit-usb-dongle-wireless-carplay-android-auto-box-wired-mirrorlink-for-aftermarket-android-screen-car-multimedia-player-bluetooth-auto-connect>

Obtido de <https://forums.raspberrypi.com/viewtopic.php?t=346925>

Obtido de <https://salamek.github.io/chromium-kiosk/>

Obtido de <https://edac.net/products/usb-connectors/5#>

Obtido de <https://www.carlinkit.com/download.html>

Obtido de <https://github.com/balena-io/etcher/blob/master/CHANGELOG.md>

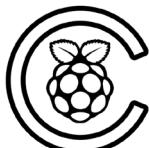
Obtido de <https://www.fully-kiosk.com/>

Obtido de <https://appleinsider.com/inside/greg-joswiak>

Obtido de <https://support.apple.com/guide/iphone/intro-to-carplay-iphf33a514c9/ios>

Obtido de <https://www.apple.com/ios/carplay/>

Obtido de <https://www.amazon.es/Miuzei-Raspberry-ventilador-alimentaci%C3%B3n-disipadores/dp/B07W4T5CCD>



Neves, F. (s.d.). *Dicio*. Obtido de Num ou em um - Dicio, Dicionário Online de Português:
<https://www.dicio.com.br/num-ou-em-um/>

Obtido de <http://pt.rctmold.com/news/what-is-polylactic-acid-what-are-the-advantages-of-pla/>

Obtido de <https://tecnoblog.net/responde/o-que-e-usb/>

Obtido de <https://www.wordreference.com/enpt/on%20board>

Obtido de https://www.bmw.pt/pt/shop/ls/dp/Base_CarPlay_pt

Obtido de <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>

Obtido de <https://raspberrypi.stackexchange.com/>

Obtido de <https://github.com/raspberrypi/documentation/tree/develop>

Obtido de https://mauser.pt/catalog/index.php?cPath=324_393_1241

Obtido de https://mauser.pt/catalog/product_info.php?products_id=096-7402&src=raspberrypi

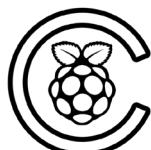
Obtido de <https://pt.wikipedia.org/wiki/USB>

USB 3.0 Specification Now Available. (11 de 2008). San Jose, Calif. Obtido de http://www.usb.org/press/USB-IF_Press_Releases/2008_11_17_USB_IF.pdf

Obtido de <https://compliantesting.com/what-causes-electromagnetic-interference/>

Obtido de <https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html>

Obtido de <https://www.carlinkit.com/ccpa>



Anexos

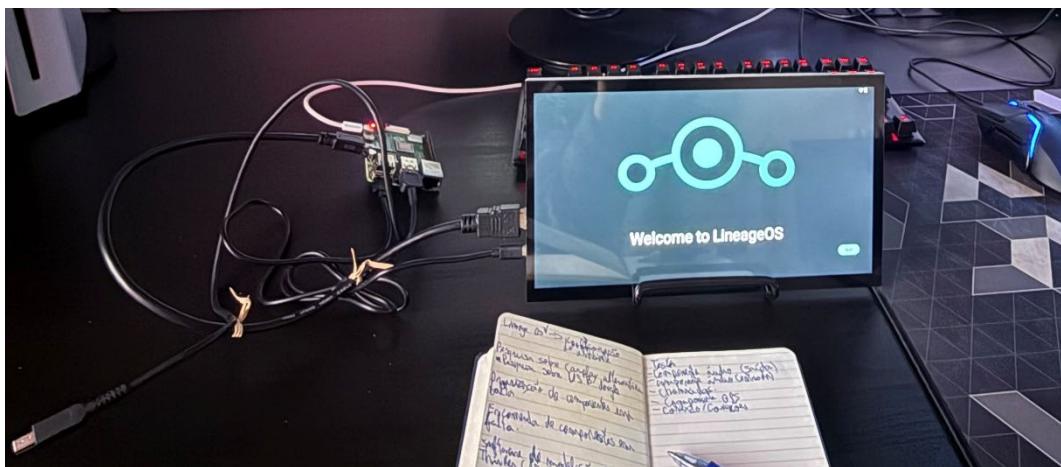


Figura 24 - Fase de instalação do Lineage OS. Hardware sem estrutura externa

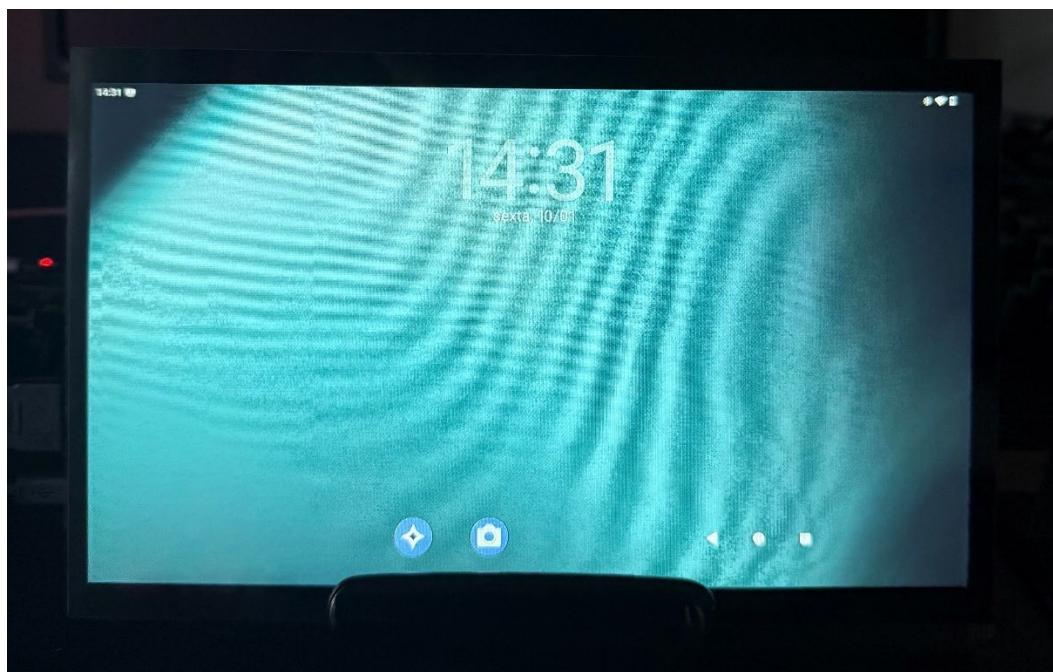
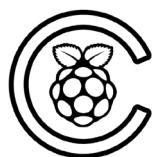


Figura 25 - Ambiente do Lineage OS após a instalação



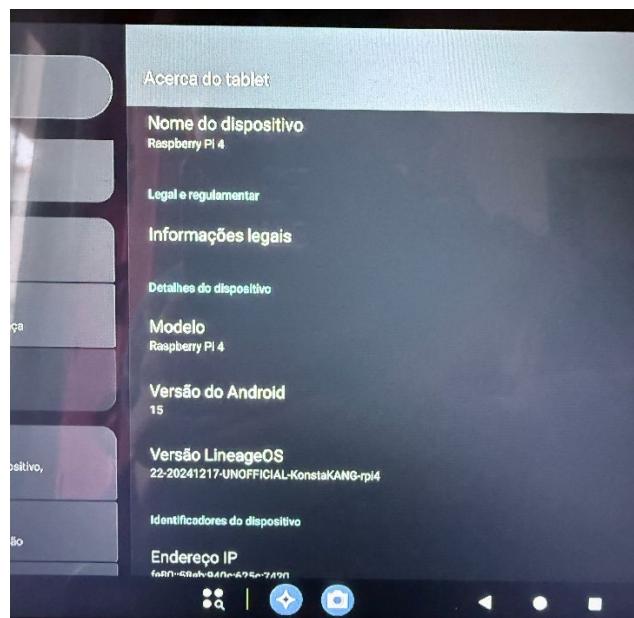


Figura 26 - Informações do software e hardware

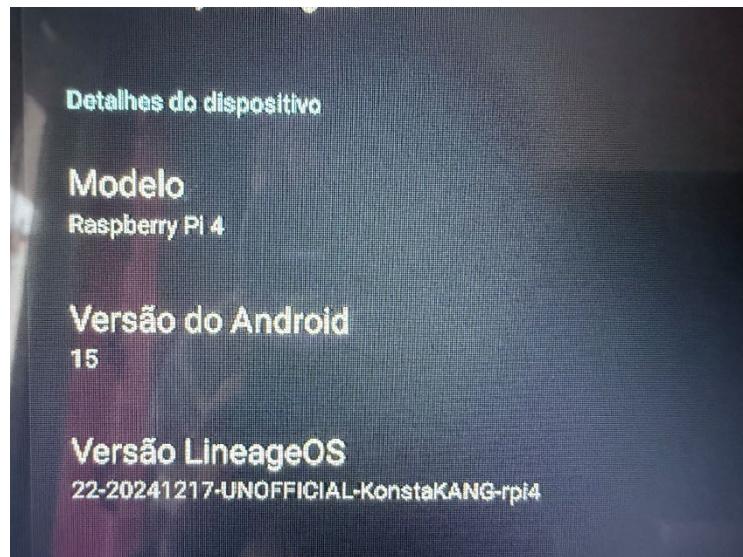
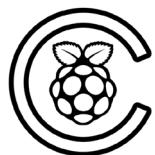


Figura 27 - Informações do software e hardware



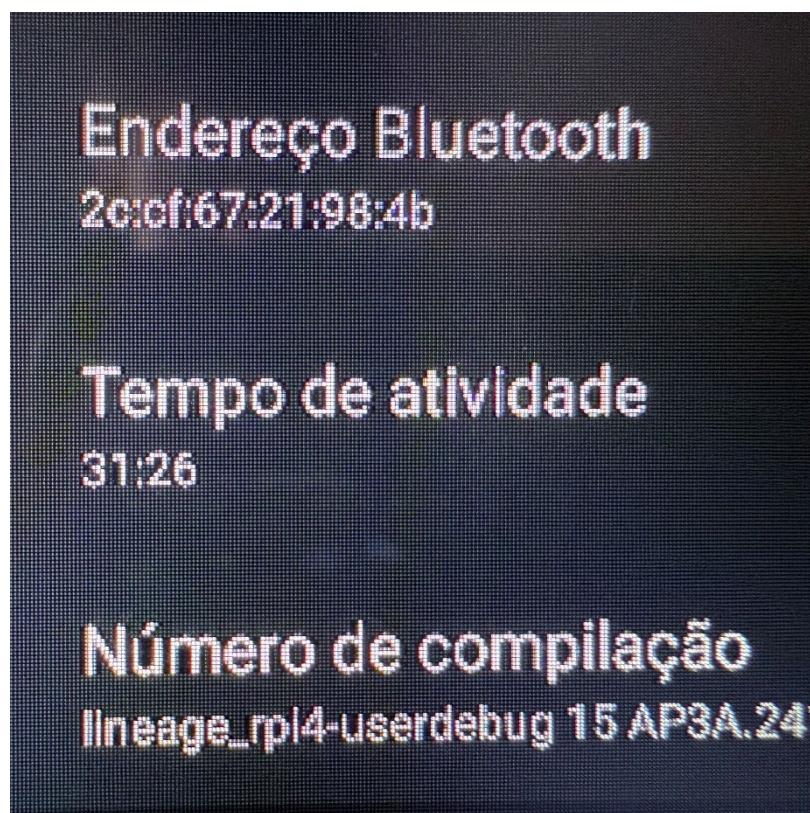


Figura 28 - Informações do software e hardware



Figura 29 - Raspberry Pi com caixa em acrílico, sistema de refrigeração instalado

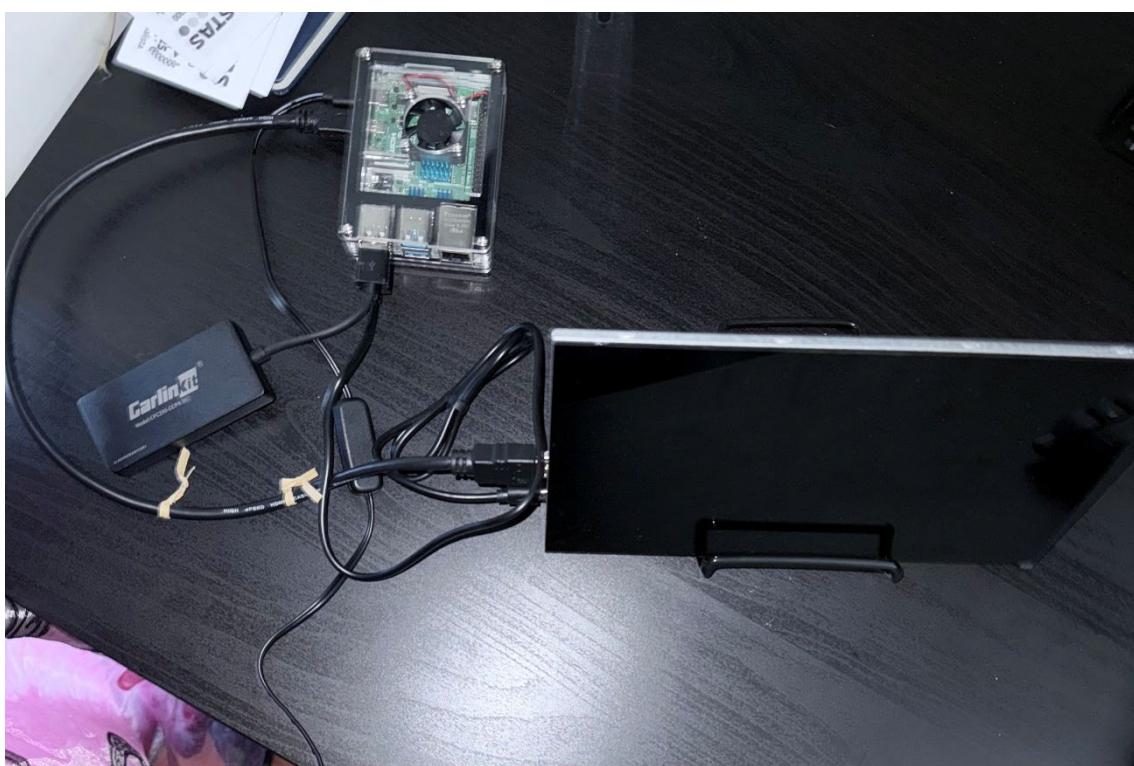


Figura 30 - Componentes do CarPlay Pi dispersos

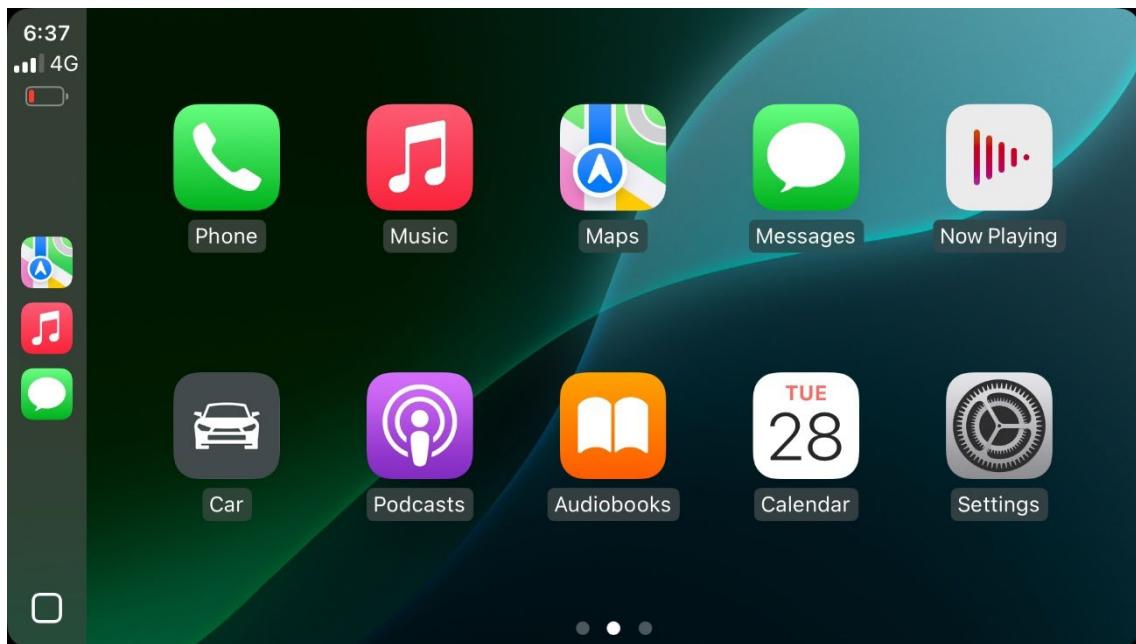


Figura 31 - Interface CarPlay

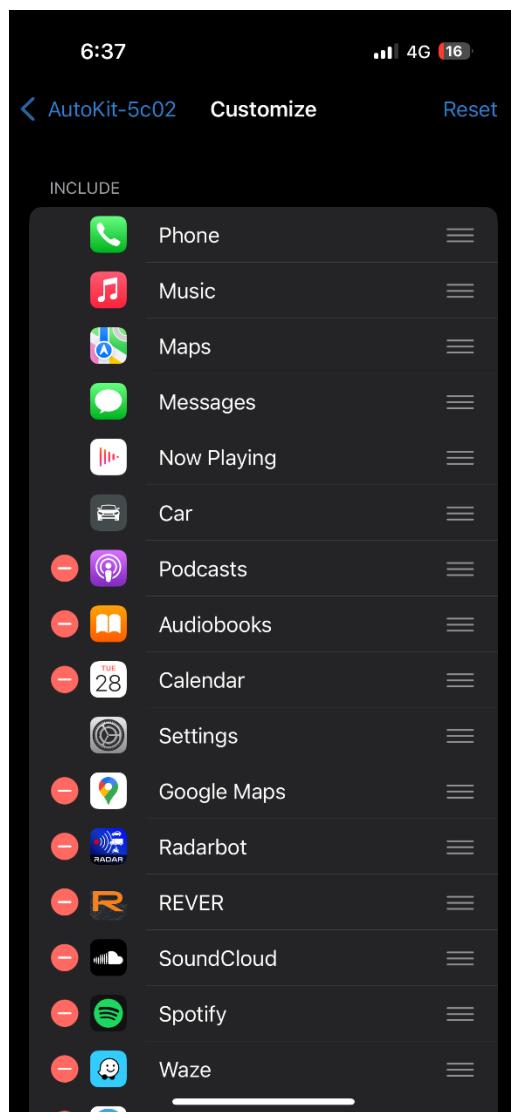


Figura 32 - Demonstração da simulação do CarPlay genuíno no iPhone



Figura 33 – Teste de chamada telefónica (com sucesso)

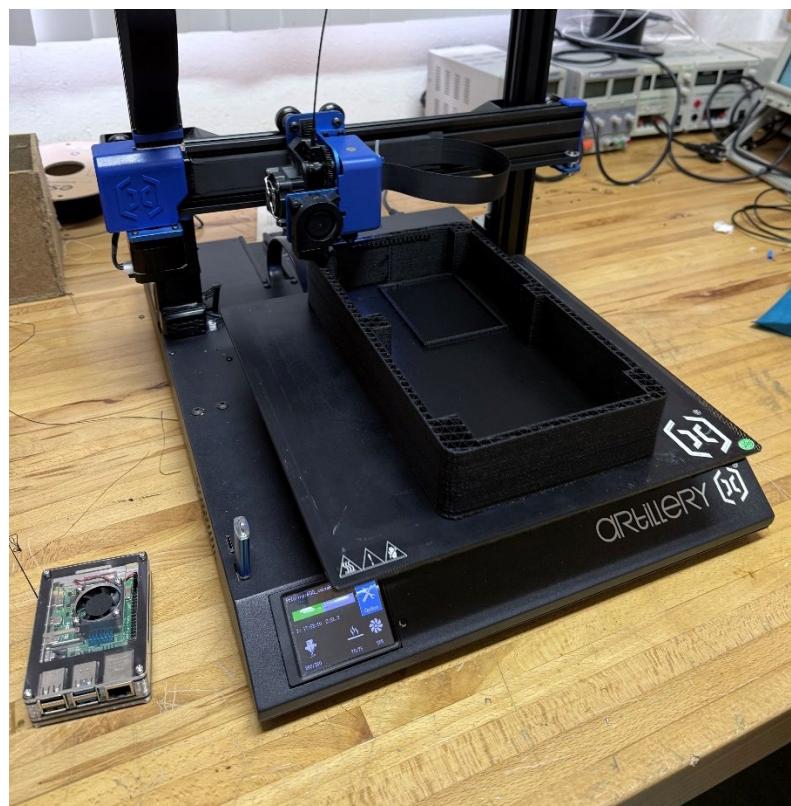


Figura 34 - Fase de impressão da peça externa

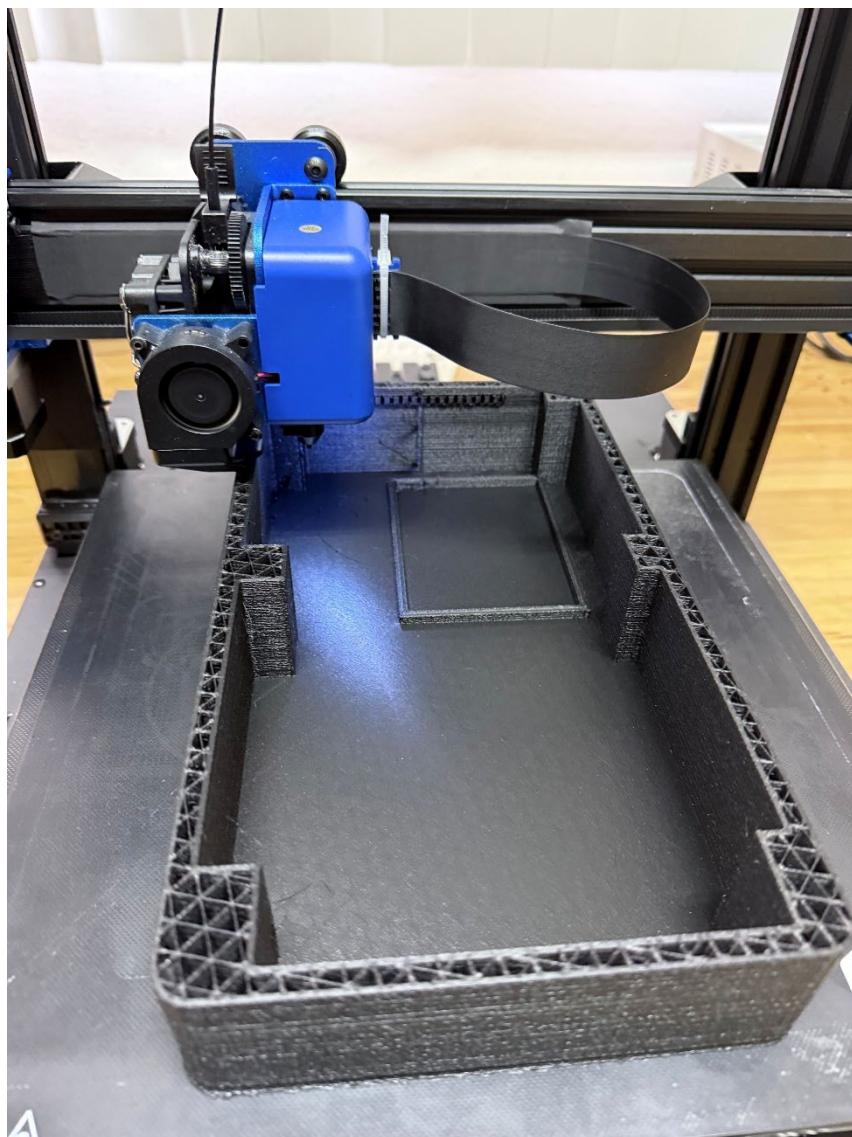


Figura 35 - Fase de impressão da peça externa

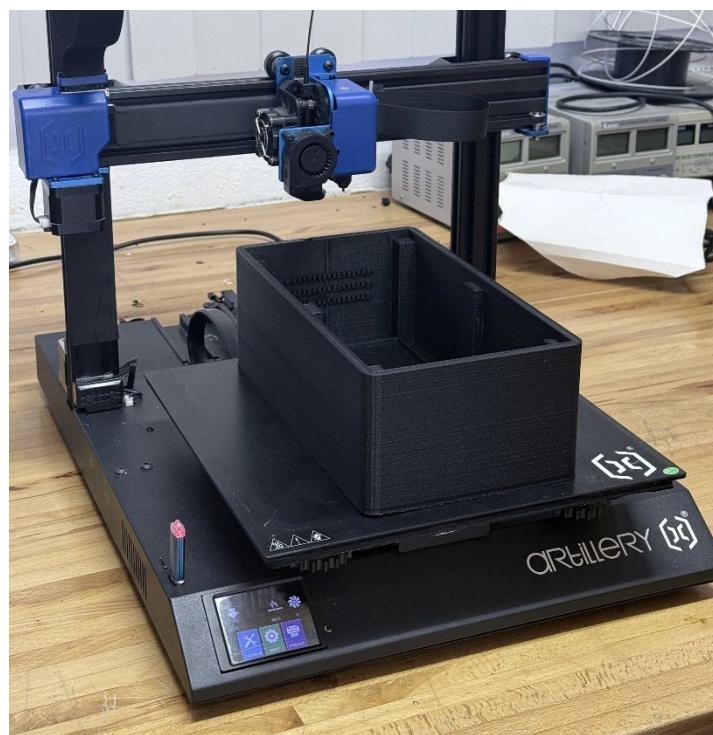


Figura 36 - Impressão da estrutura externa concluída

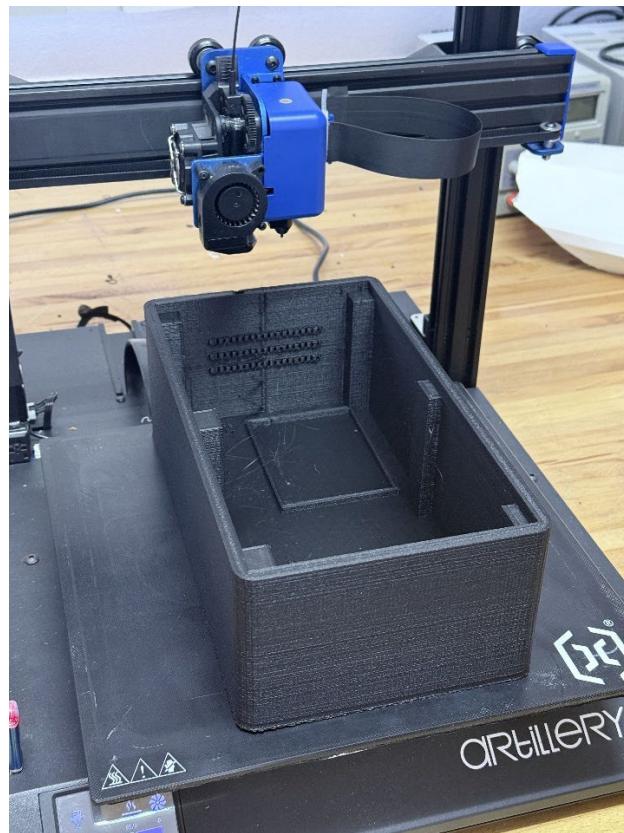


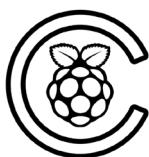
Figura 37 - Impressão da estrutura externa concluída



Figura 38 - Sistema totalmente montado, apenas peça lateral em falta

Glossário

<i>access point</i>	27
<i>Android Debug Bridge</i>	18
automação.....	11, 28
<i>bloatware</i>	27, 30, 31
<i>boot</i>	18, 23, 27
<i>cross-platform</i>	28
<i>DIY</i>	11, 14, 15, 28
firmware	27
funcionalidades premium	10
hardware modular.....	15
infotainment.....	20
Interface	14, 29
<i>Internet of Things</i>	11
IoT	11, 28
<i>Lucky Patche</i>	27
<u>microcomputadores</u>	9, 10
open-source.....	15
<i>plug-and-play</i>	14
pontos multitouch	29
script	27
script <i>systemd</i>	27
<i>single-board computers</i>	22
Siri	12, 15
Slot.....	26
<i>throttling</i> da CPU.....	28
touchscreen	30
<i>USB 2.0 EDAC</i>	19
<i>USB 3.0</i>	19



O Formando

(Marcos Ribeiro)

O Orientador

(Fábio Coelho)

A Diretora Pedagógica

(Helena Margarida Lopes Moutinho Neto)

Sernancelhe, [] de março de 2025