IAN ELMÔR LANG

Periférico para Interação Gestual em Terminais Públicos

São Paulo
2018

IAN ELMÔR LANG

Periférico para Interação Gestual em Terminais Públicos

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Engenharia Elétrica – Ênfase em Computação.

São Paulo
2018

IAN ELMÔR LANG

Periférico para Interação Gestual em Terminais Públicos

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Engenharia Elétrica – Ênfase em Computação.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. André Riyuiti Hirakawa

São Paulo
2018

Prof. Dr. André Riyuiti Hirakawa

AGRADECIMENTOS

Agradeço à minha família, aos meus amigos e à comunidade da Universidade de São Paulo pelo apoio ao longo da minha formação.

Ian

RESUMO

Terminais computacionais públicos, com os quais utilizadores travam breves interações, têm uma presença importante no cotidiano moderno: exemplos incluem caixas automáticos, armários de entregas inteligentes, máquinas de vendas e terminais de informação.

Este projeto propõe uma arquitetura para um periférico que possibilita a interação com tais terminais através de gestos: um sensor de imagem captura um fluxo de vídeo da mão do usuário, que é interpretado por um microcontrolador de alta performance que determina quais dedos da mão estão levantados e abaixados. Um identificador do gesto reconhecido é então enviado, através de uma conexão USB, para o computador principal do terminal, que o interpreta como uma entrada de dados pelo usuário.

Entre as razões que tornam interessante explorar esta modalidade de interação gestual com terminais públicos, estão a maior largura de banda dos dedos para transmissão de informação, em relação ao punho e braços (que fazem parte da interação com teclados e telas de toque) e o menor risco de que a interação possa ser um vetor de transmissão de doenças, pois pois não há contato físico direto com o terminal. Desse modo, a adoção do periférico por terminais públicos poderia permitir que usuários realizem suas operações mais rapidamente e estejam expostos a menos riscos durante épocas de epidemia.

Um protótipo da arquitetura proposta for implementado, incluindo um sistema embarcado que desempenha as funções do periférico e uma interface gráfica que comunica-se com os o sistema embarcado e permite que um utilizador controle um teclado numérico realizando gestos. O protótipo foi testado com o autor e dois voluntários: tais testes envolveram acionar todas as teclas do teclado numérico em uma sequência conhecida, e realizar todos os gestos reconhecidos pelo sistema para avaliar sua taxa de acerto. Tais testes foram bem-sucedidos com o autor e um dos voluntários (interação correta com o teclado, reconhecimento da maioria dos gestos com taxa de acerto maior ou igual a 80%), mas o outro voluntário encontrou erros sistemáticos no reconhecimento de gestos. Isso sugere que a arquitetura adotada permite a interação gestual pretendida, mas o método de reconhecimento de gestos deve ser refinado.

Palavras-Chave – Terminais Públicos, Sistemas Embarcados, Interação Gestual, Visão Computacional.

ABSTRACT

Public computing terminals, with which users establish brief interactions, have an important presence in modern day-to-day life: examples include ATMs, intelligent delivery lockers, sales machines, and information terminals.

This project proposes an architecture for a peripheral that enables interaction with such terminals through gestures: an image sensor captures a video stream from the user's hand, which is interpreted by a high performance microcontroller that determines which fingers of the hand are raised and lowered. A recognized gesture identifier is then sent via a USB connection to the terminal's main computer, which interprets it as a data input by the user.

Among the reasons that make it interesting to explore this modality of gestural interaction with public terminals, are the greater bandwidth of the fingers for information transmission, in relation to the wrist and arms (that are part of the interaction with keyboards and touch screens) and the lower risk that the interaction may be a vector of disease transmission, since there is no direct physical contact with the terminal. In this way, the adoption of peripherals by public terminals could allow users to carry out their operations more quickly and be exposed to fewer risks during disease outbreaks.

A prototype of the proposed architecture is implemented, including an embedded system that performs the functions of the peripheral and a graphical interface that communicates with the embedded system and allows a user to control a numerical keyboard by performing gestures. The prototype was tested with the author and two volunteers: such tests involved triggering all the keys from the numerical keyboard in a known sequence, and performing all gestures recognized by the system to evaluate their hit rate. Such tests were successful with the author and one of the volunteers (correct interaction with the keyboard, recognition of most gestures with hit rate greater than or equal to 80%), but the other volunteer found systematic errors in gesture recognition. This suggests that the adopted architecture allows for the desired gestural interaction, but the gesture recognition method should be refined.

Keywords – Public Terminais, Embedded Systems, Gesture Interaction, Computer Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

BSP	*Board Support Package*
DCMI	*Digital Camera Interface*
DMA	*Direct Memory Access*
FPS	*Frames per Second*
FIFO	*First-in-First-out*
HAL	*Hardware Abstraction Layer*
HID	*Human Interface Device*
I2C	*Inter-Integrated Circuit*
IHC	*Interação Humano-Computador*
ISP	*Image Signal Processor*
ISR	*Interrupt Service Routine*
LCD	*Liquid Crystal Display*
PID	*Product Identifier*
RAM	*Random-access Memory*
RGB	*Red Green Blue*
USB	*Universal Serial Bus*
UVC	*USB Video Class*
VID	*Vendor Identifier*

SUMÁRIO

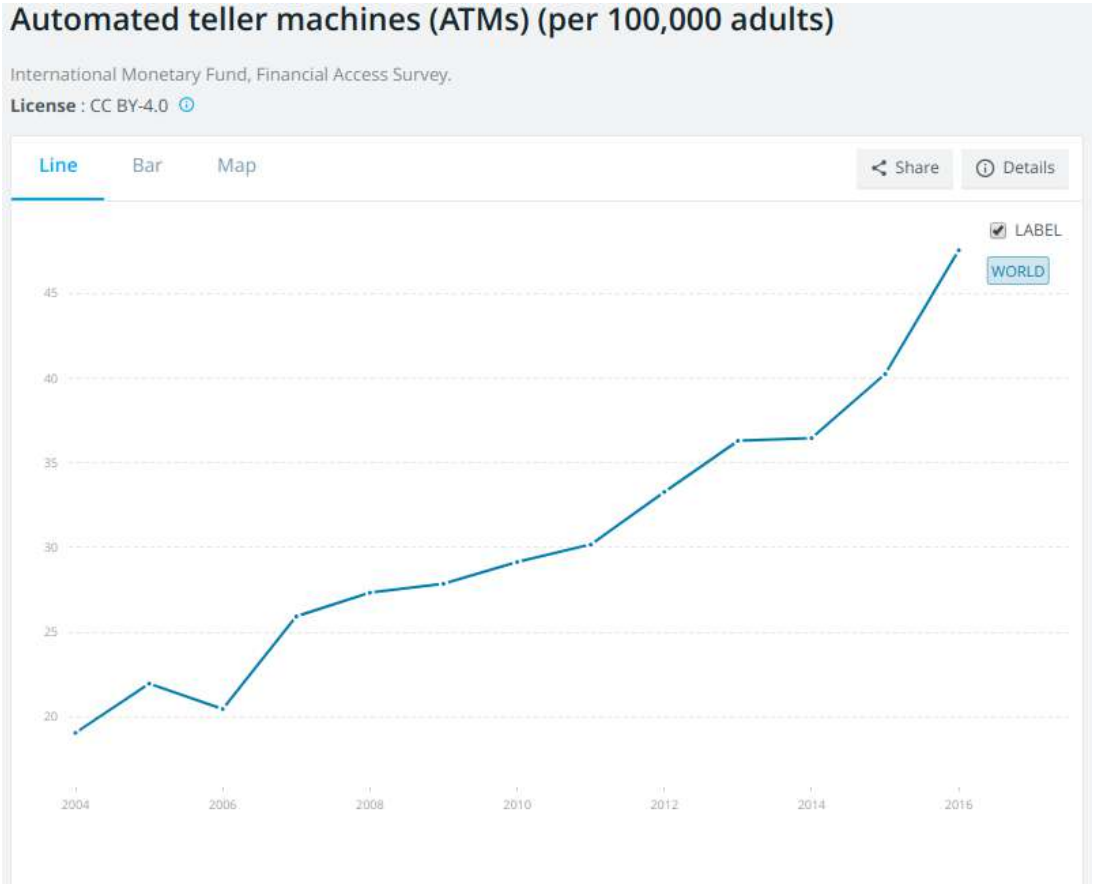
1 Introdução

1.1 Motivação

A realização deste projeto foi motivada pela importante presença de diversos tipos terminais computacionais públicos, com os quais os utilizadores travam interações rápidas, por exemplo:

- A despeito de inovações como o internet banking, a quantidade de caixas automáticos (*Automated Teller Machines* - ATM) continua a crescer ao redor do mundo, como evidenciado pelo gráfico da figura 1, levantado pelo Fundo Monetário Internacional.
- Armários de entregas inteligentes, nos quais o usuário pode inserir uma senha para recuperar uma encomenda depositada anteriormente por um serviço de entregas. Um dos modelos mais notáveis, o Amazon Locker, está presente em mais de duas mil localizações em mais de 50 cidades (AMAZON, 2015).
- Máquinas automáticas de vendas.
- Terminais de informação em museus, centros de compras e outros locais de grande circulação.

Figura 1: Crescimento na quantidade de caixas automáticos ao redor do mundo, entre 2004 e 2017.



Fonte: (WORLD BANK, 2018)

Figura 2: Dois exemplos de terminais computacionais públicos: Um armário de entregas e um caixa automático.

Fonte: (BUSINESS INSIDER, 2012)

O presente trabalho foi motivado pela ideia de explorar uma nova modalidade de interação com tais terminais, a interação gestual, e como implementá-la de um modo econômico e prático.

1.2 Justificativa

Cabe justificar o presente trabalho em duas dimensões: o interesse de se explorar a interação gestual em terminais públicos, e as qualidades da arquitetura adotada no projeto como uma solução para tal tipo de interação gestual. Alguns dos fatores que tornam interessante a ideia de explorar a interação como opção para terminais públicos:

- Em relação tanto à interação através de um teclado físico, quanto à interação através de telas de toque, que envolvem a movimentação dos braços e/ou punho, a modalidade de interação gestual abordada envolve apenas levantar e abaixar dedos. Assim, pode-se aproveitar a maior largura de banda dos dedos para transmissão de

informação, em relação ao punho e braços. O trabalho (LANGOLF; CHAFFIN; FOULKE, 1976) avaliou a largura de banda destas três alternativas sobre uma amplitude de deslocamentos de 0,25cm a 30cm, e determinou larguras de banda 38 bits/s, 23 bits/s e 10 bits/s para dedos, punho e braços, respectivamente.

- Em relação tanto à interação através de um teclado físico, quanto à interação através de telas de toque, evita-se a preocupação de que a interação com os terminais computacionais públicos possa ser um vetor de transmissão de doenças, pois pois não há contato físico direto com o terminal. Trabalhos de caracterização do microbioma na superfície de caixas automáticos já identificaram a presença da bactéria *Escherichia coli*, que pode causar intoxicações alimentares, em terminais em Vellore, na Índia (NACHIMUTHU et al., 2015), em Malatya, na Turquia (TEKEREKOğLU et al., 2013), e em Ilesa e Ijebu Seja, na Nigéria (AWOPETU et al., 2017).

- Em relação à interação através de teclas físicas, evita-se a preocupação com o desgaste mecânico ao longo do tempo.

Desse modo, a adoção do periférico por terminais públicos poderia permitir que usuários realizem suas operações mais rapidamente e estejam expostos a menos riscos durante épocas de epidemia.

A arquitetura adotada no projeto é apresentada e comparada com outras alternativas de IHC gestual no capítulo 2. Suas principais qualidades são:

- Do ponto de vista do computador que recebe os gestos detectados, o periférico age de maneira equivalente a um teclado físico capaz de informar qual tecla foi detectada através da classe HID de comunicação sobre USB, a mesma utilizada pour *mouses* e teclados USB. Dessa maneira, não ha necessidade de adaptar o software embarcado dos terminais públicos para realizar o processamento das imagens dos gestos realizados pelo utilizador, o que simplifica a adoção do periférico.

- O periférico é baseado em um microcontrolador, de custo e consumo de energia mais baixos em comparação com computadores de placa única como o Raspberry Pi.

1.3 Objetivo

Em vista da motivação e da justificativa apresentadas acima, o projeto teve por objetivo a produção de um protótipo de um periférico que permite a interação de usuários

com computadores de uso público, como caixas automáticos e terminais de informação, através gestos realizados com uma mão. Pelo restante deste texto, a ideia de "gesto" corresponde a combinações de dedos levantados e abaixados de uma mão. Para possibilitar uma avaliação do sistema embarcado, o projeto também abrangeu a produção de um programa com interface gráfica, que aproveita os gestos reconhecidos pelo periférico para permitir que o utilizador controle um teclado numérico.

1.4 Organização do Trabalho

Abaixo encontram-se descrições do conteúdo dos demais capítulos presentes nesta dissertação.

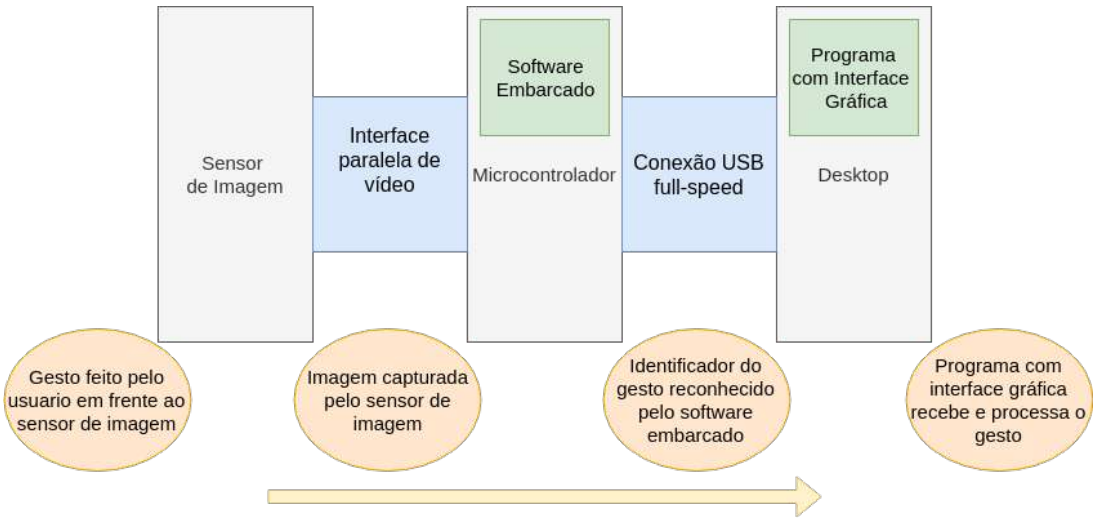
- O **capítulo 2** apresenta a arquitetura do protótipo desenvolvido e discute os aspectos conceituais que permitem sua localização no universo dos sistemas de interação gestual.
- O **capítulo 3** lista a sequência de etapas em que foi organizado o trabalho no projeto.
- O **capítulo 4** apresenta o conjunto de requisitos funcionais e não funcionais que foram elencados para o projeto.
- O **capítulo 5** apresenta as tecnologias, em termos de hardware e software, que foram selecionadas para o desenvolvimento do sistema embarcado e da interface gráfica.
- O **capítulo 6** relata o processo de implementação do sistema embarcado e da interface gráfica.
- O **capítulo 7** descreve testes que foram utilizados para validar o funcionamento de diferentes aspectos do projeto e avaliar seu desempenho.
- O **capítulo 8** apresenta considerações finais a respeito do projeto e das possibilidades de expandi-lo no futuro.
- O **Apêndice A** mostra detalhes das bibliotecas que foram utilizadas na elaboração do software embarcado.

2 Aspectos Conceituais

Este capítulo tem por objetivo localizar a arquitetura que foi escolhida para o projeto no universo de seu domínio de aplicação: sistemas embarcados utilizados para a IHC através de gestos, em particular aqueles que baseiam-se na filmagem do usuário.

A figura 3 mostra uma visão geral de tal arquitetura, com os componentes de hardware escolhidos (cinza), interfaces de comunicação (azul), software de aplicação elaborado (verde) e o fluxo de informações (laranja). Nesta arquitetura, o periférico de reconhecimento de gestos consiste em um microcontrolador de alta performance, associada a um sensor de imagem. Um software embarcado permitie ao microcontrolador recuperar imagens do sensor, processá-las para identificar realizados pelo usuário e enviar os resultados obtidos para um computador através de uma conexão USB 2.0. Tal transmissão de informação ocorre de acordo com o protocolo HID (Human Interface Device), para comunicação de dispositivos de interação humano-computador (IHC) através de barramentos USB, que é utilizado, por exemplo, por *mouses* e teclados.

Figura 3: Visão global do sistema desenvolvido.



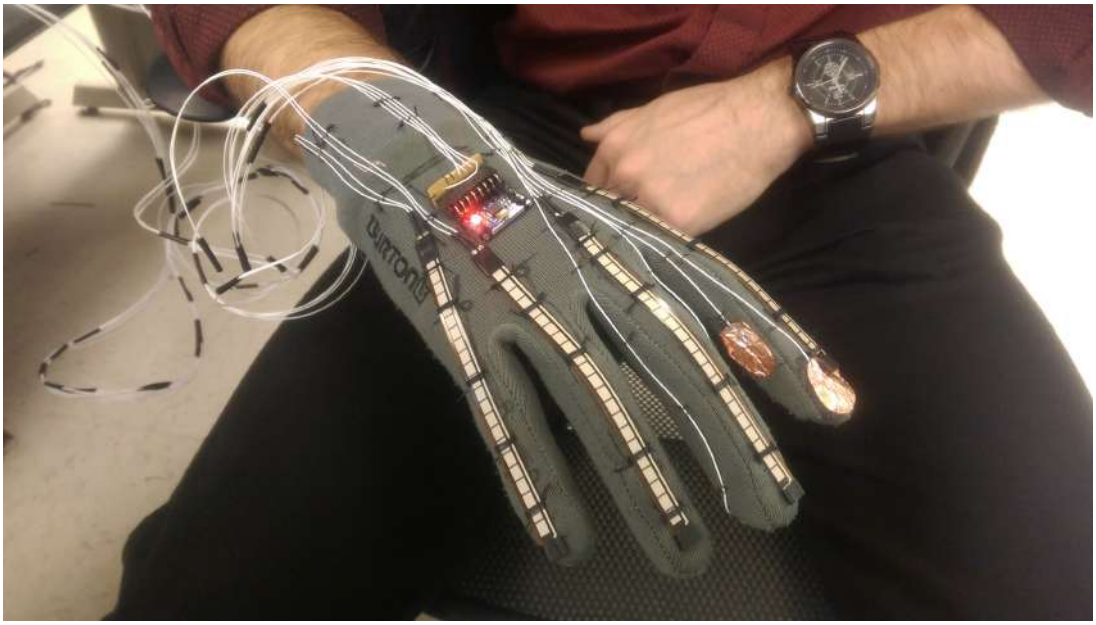
Fonte: Autor

2.1 Estratégias de Reconhecimento de Gestos

Uma revisão da literatura apresenta três grandes estratégias para o desenvolvimento de sistemas de interação humano computador:

- Sistemas que incluem uma luva vestida pelo usuário, equipada com sensores flex (componentes flexíveis capazes de informar o grau em que estão sendo deformados) e inerciais (acelerômetros, giroscópios). À medida que o usuário movimenta seus dedos, mãos e braços, a leitura destes sensores permite a um microcontrolador quantificar o movimento realizado. Exemplos desta abordagem incluem (LIN; VILLALBA, 2014), (FLORES et al., 2014) (DEKATE; KAMAL; SUREKHA, 2014).

Figura 4: Exemplo de uma luva com sensores utilizada para a IHC através de gestos.



Fonte: (LIN; VILLALBA, 2014)

- Sistemas que utilizam um mecanismo de sonar para acompanhar a posição de elementos da mão do utilizador no espaço. Exemplos desta abordagem incluem (KALGAONKAR; RAJ, 2009) e (NANDAKUMAR et al., 2016).

Figura 5: Exemplo de um mecanismo de sonar utilizado para a IHC através de gestos.

Fonte: (FINGERIO, 2017)

- Sistemas que filmam parte do corpo do utilizador com uma ou mais câmeras, e analisam o fluxo de imagens obtidas para identificar os gestos realizados. Esta estratégia será o foco das próximas seções.

Cabe observar que tais estratégias não são mutuamente excludentes: o trabalho (SANTOS et al., 2018), por exemplo, baseou-se em uma câmera montada sobre uma luva vestida pelo usuário.

A opção pela utilização da estratégia baseada em vídeo no projeto que foi desenvolvido, em detrimento das outras duas, pode ser justificada pois:

- A estratégia baseada em uma luva pode ser considerada demasiado invasiva para ser utilizada na interação com terminais públicos.
- A estratégia baseada em tecnologias de sonar exige equipamentos e conhecimentos menos disseminados do que a estratégia baseada na filmagem do usuário.

2.2 Reconhecimento de Gestos por Vídeo

Nesta seção, a arquitetura adotada para o projeto será comparada com outras abordagens para o reconhecimento de gestos através da filmagem das mãos do usuário, sob duas óticas: a da escolha da plataforma de computação utilizada para a análise de imagem, e sob a ótica da escolha do protocolo de comunicação sobre USB utilizado, se algum.

2.2.1 Plataforma de Computação

O dispositivo LeapMotion (LEAPMOTION, 2018), mostrado na figura 6, e o dispositivo Kinect (MICROSOFT, 2018) são periféricos que possibilitam a interação humano computador através da movimentação de partes do corpo. Ambos usam o principio de filmar o usuário com uma ou mais câmeras e enviar o fluxo de imagens a um computador principal (desktop ou console de jogos) mais potente, capaz de de executar um sistema operacional como Windows, Linux ou macOS, o qual processa as imagens para determinar as entradas de dados a partir da movimentação do usuário.

Figura 6: IHC através do dispositivo LeapMotion.

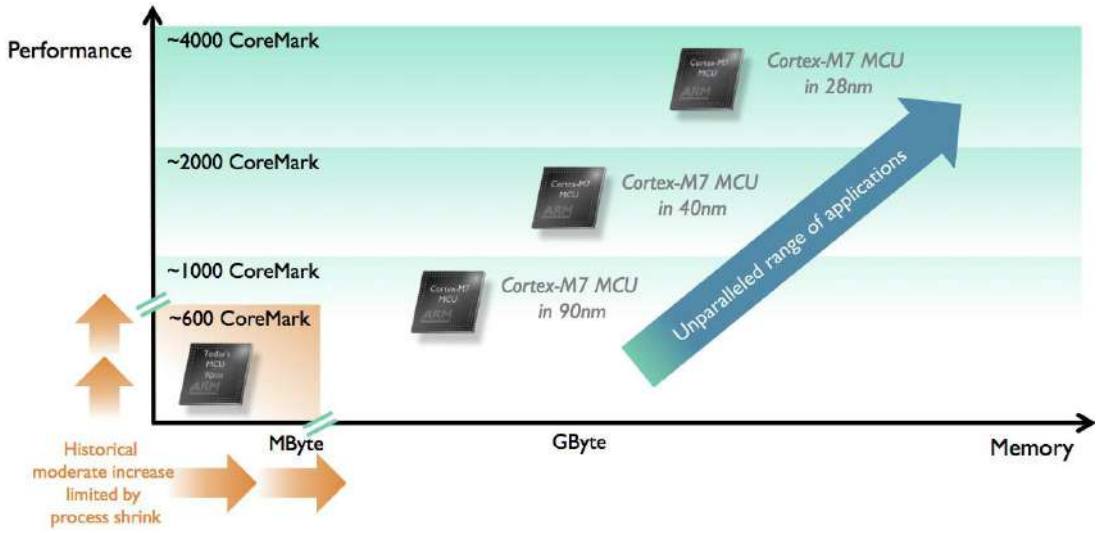
Fonte: (WESTOVER, 2013)

Sistemas de interação gestual como os explorados em (LEI et al., 2014) e (BEGALINOVA; SHINTEMIROV, 2014), também recorrem à estratégia de processar as imagens capturadas em um computador que executa um sistema operacional completo e é suportado pela biblioteca de processamento de imagens openCV (OPENCV, 2018b): no caso, computadores de placa única como o raspberry pi (RASPBERRY..., 2015).

O presente trabalho, por sua vez, procura implementar um periférico de reconhecimento de gestos que realiza o processamento de imagens em um microcontrolador: um dispositivo computacional de performance, preço e consumo de energia mais reduzido em relação às plataformas de computação utilizadas pelos projetos supracitados, e desprovido de alguns dos mecanismos necessários para a execução de sistemas operacionais como Linux (por exemplo, a *Memory Mapping Unit*, necessária para utilização de memória virtual).

A idéia de utilizar um microcontrolador é inspirada pela evolução observada no desempenho deste tipo de plataforma computacional nos últimos anos, como retratado por exemplo no gráfico da figura da figura 7. O microcontrolador STM32F746 utilizado neste projeto, equipado com um núcleo ARM Cortex-M7, alcança uma pontuação de 1082 no benchmark computacional CoreMark, o que está próximo da pontuação de 2060 alcançada pelo Raspberry Pi Zero (EMBC, 2018).

Figura 7: Evolução da linha de processadores Cortex-M, utilizada em microcontroladores.



Fonte: (WILLIAMS, 2014)

Neste contexto, o produto comercial que mais se assemelha ao protótipo desenvolvido neste trabalho é a plataforma openMV (OPENMV, 2018), retratada na figura 8, que também é composta por microcontrolador STM32F7 e um sensor de imagem Omnivision. A plataforma openMV também realiza processamento de imagens em seu microcontrolador STM32F7, mas nao tem o reconhecimento de gestos como alvo, sendo uma plataforma mais geral de aprendizado e prototipação de aplicações de visão computacional em sistemas embarcados.

Figura 8: A plataforma OpenMV.

Fonte: (OPENMV, 2018)

2.2.2 Conexão USB

A especificação do barramento de comunicação *Universal Serial Bus* (USB IMPLEMENTERS' FORUM, 2001b), comumente denominado USB, prevê um conjunto de "classes de comunicação", as quais estabelecem os protocolos de trocas de dados sobre USB a ser utilizados por diferentes tipos de dispositivos. Tais classes incluem, entre outras:

- Audio – Exemplos de utilização:-falantes, microfones, cartas de som, etc.
- Human Interface Device (HID) – Exemplos de utilização: teclados, *mouses*, e *joysticks*.

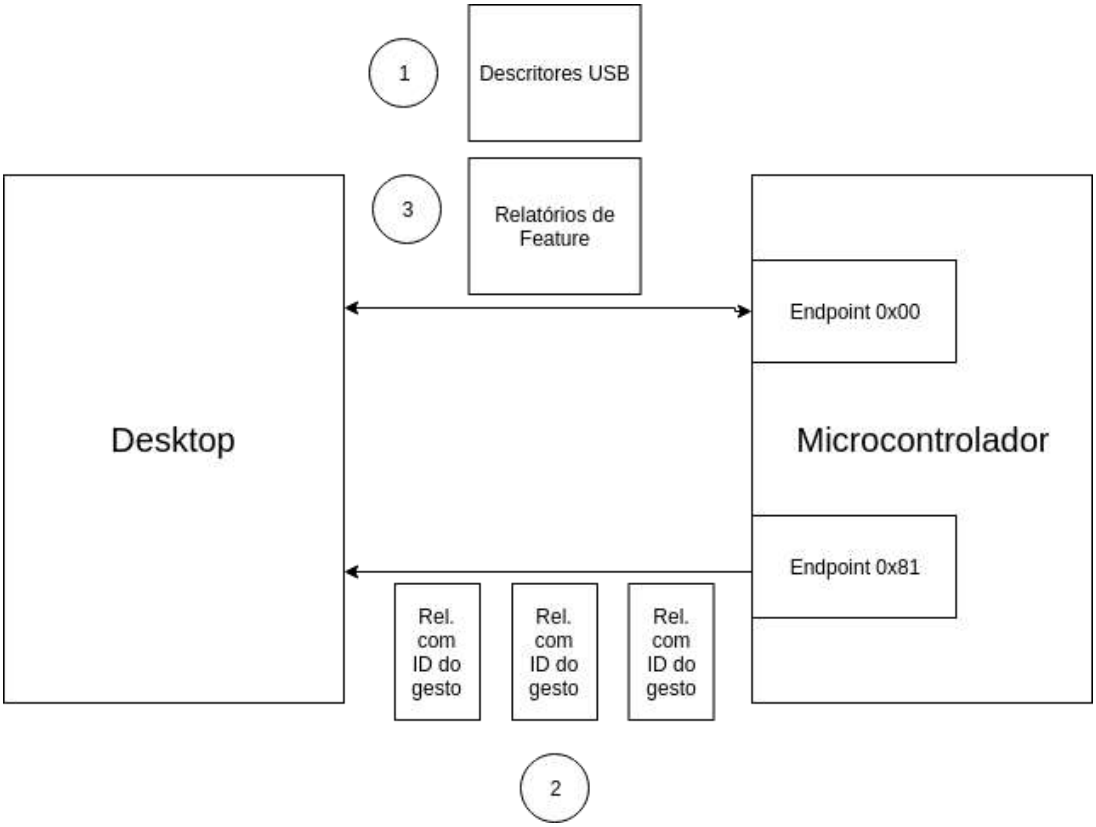
- USB Video Class (UVC) – Exemplos de utilização: *webcams*.
- Printer – Exemplos de utilização: impressoras e máquinas CNC.
- Mass storage – Exemplos de utilização: drives de memória.
- Communications Class Device (CDC) – Exemplos de utilização: Modems e adaptadores Ethernet e Wi-Fi.

Dispositivos como o LeapMotion e o Kinect utilizam a classe UVC (USB IMPLEMENTERS' FORUM, 2012) para transmitir um fluxo de vídeo para o computador principal, que é responsável tanto por processá-las para determinar os gestos realizados, quanto por atualizar a lógica da aplicação em função dos gestos reconhecidos. Caso terminais públicos adotassem este tipo de periférico, seria necessário modificar seu software de aplicação para adicionar funcionalidades de visão computacional.

O periférico desenvolvido para o presente trabalho, por sua vez, utiliza a classe HID (USB IMPLEMENTERS' FORUM, 2001a), apresentando-se ao computador principal de maneira similar a um teclado, *mouses*, ou *joysticks*. Assim, pode-se esperar que sua adoção por terminais públicos implique em modificações menos drásticas no software destes.

A figura 9 esquematiza a transmissão de informações de gestos através da classe HID. No momento em que a conexão é estabelecida, o computador principal lê um conjunto de estruturas de dados (denominadas descritores USB) através do *endpoint* número 0x00 (1 na figura). Em comunicação USB, *endpoints* são buffers de dados endereçáveis da interface USB de um dispositivo – o *endpoint* 0x00 é especial por estar presente em todos os dispositivos, e portanto é utilizado para a leitura dos descritores durante a inicialização. Os descritores informam que trata-se de um dispositivo da classe HID, que enviará relatórios com 1 byte de tamanho com uma periodicidade de 30 Hz através do *endpoint* número 0x81. A partir deste ponto, o computador principal utiliza as chamadas "interrupt transfers" do barramento USB para receber, periodicamente, tais relatórios, que contém uma máscara de bits que representa quais dedos do usuário estão levantados (2 na figura). A classe HID também permite ao computador principal ler e escrever relatórios contendo informações de configuração do dispositivo, através do *endpoint* 0x00 (3 na figura), mas esta funcionalidade não é empregada neste projeto.

Figura 9: Diagrama da transmissão de gestos detectado pela classe HID.



Fonte: Autor

3 Metodologia do Trabalho

O trabalho no projeto foi consistiu nas seguintes tarefas:

1. Definição da problemática a ser abordada, e da arquitetura geral do sistema (capítulos 1 e 2).
2. Levantamento dos requisitos do sistema (capítulo 4).
3. Seleção e aquisição dos componentes de hardware, escolha das bibliotecas de software (capítulo 5).
4. Implementação do sistema embarcado, com funcionamento verificado através de sua tela LCD (capítulo 6 – Seção 1).
5. Implementação da interface gráfica (capítulo 6 – Seção 2).
6. Realização dos testes finais (capítulo 7).
7. Conclusão da monografia, elaborada em paralelo com as tarefas anteriores (capítulo 8).

O autor foi o responsável pela execução das tarefas listadas, consultando periodicamente seu orientador.

4 Especificação de Requisitos do Sistema

Para guiar a implementação e os testes do sistema embarcado e da interface gráfica, foram levantados os requisitos funcionais e não-funcionais elencados nas próximas sessões.

4.1 Requisitos do Sistema Embarcado

4.1.1 Requisitos Functionais

- O periférico deve reconhecer todas as todas as combinações possíveis de dedos levantados e abaixados.
- Ao ser alimentado por um cabo USB, o sistema deve apresentar-se ao host USB como um dispositivo da classe HID (Human Interface Device).
- Após a conclusão das rotinas de inicialização, o sistema deve capturar imagens através do sensor, analisá-las para identificar o gesto (conjunto de dedos da mão levantados) realizado pelo usuário e enviar um identificador do gesto identificado na forma de um relatório da classe HID de comunicação sobre USB.

4.1.2 Requisitos Não-funcionais

- Um único cabo USB 2.0 deve ser suficiente para a transmissão de dados e alimentação do periférico.
- O sistema deve reconhecer e reportar gestos à uma taxa maior ou igual a 30 atualizações por segundo (menor taxa na qual opera o menor taxa na qual opera o periférico LeapMotion (LEAPMOTION, 2018)).

4.2 Requisitos da Interface Gráfica

4.2.1 Requisitos Funcionais

- A interface gráfica deve esperar que o usuário mantenha um mesmo gesto por um segundo antes de considerá-lo como uma entrada de dados válida, de modo a evitar entradas espúrias.
- A interface gráfica deve oferecer um usuário um *feedback* em tempo real do qual gesto está sendo detectado.
- Deve ser igualmente oferecido ao usuário um *feedback* em tempo real do transcorrer do tempo de espera até que o gestos seja considerado válido.
- O sistema deve reconhecer ao menos as seguintes teclas tradicionais de teclados numérics: dígitos de 0 a 9, “Confirma” e “Deleta”.
- A cada tecla deve corresponder um ou mais gestos, e tal correspondência deve ser mostrada ao usuário na interface gráfica.

4.2.2 Requisitos Não-Funcionais

- O sistema deve recolher relatorios com o gesto detectado pelo sistema embarcado à uma taxa maior ou igual a 30 atualizações por segundo.

5 Tecnologias Utilizadas

A escolha das tecnologias utilizadas no projeto teve três dimensões:

- As componentes de hardware do sistema embarcado.
- As ferramentas utilizadas para compilar o software embarcado e registrá-lo na memória não-volátil do sistema embarcado, e as bibliotecas utilizadas na programação de tal embarcado.
- As bibliotecas utilizadas na programação da interface gráfica.

Tais dimensões são exploradas nas próximas seções do presente capítulo.

5.1 Sistema Embarcado – Hardware

A figura 10 mostra as duas placas de desenvolvimento que foram escolhidas para a constituição do hardware do sistema embarcado: a 32F746G-DISCOVERY (no topo) e STM32F4DIS-CAM (abaixo) conectadas por um cabo tipo "ribbon"

Figura 10: Placas de desenvolvimento utilizadas no projeto.

Fonte: (STMICROELECTRONICS, 2017a)

5.1.1 32F746G-DISCOVERY

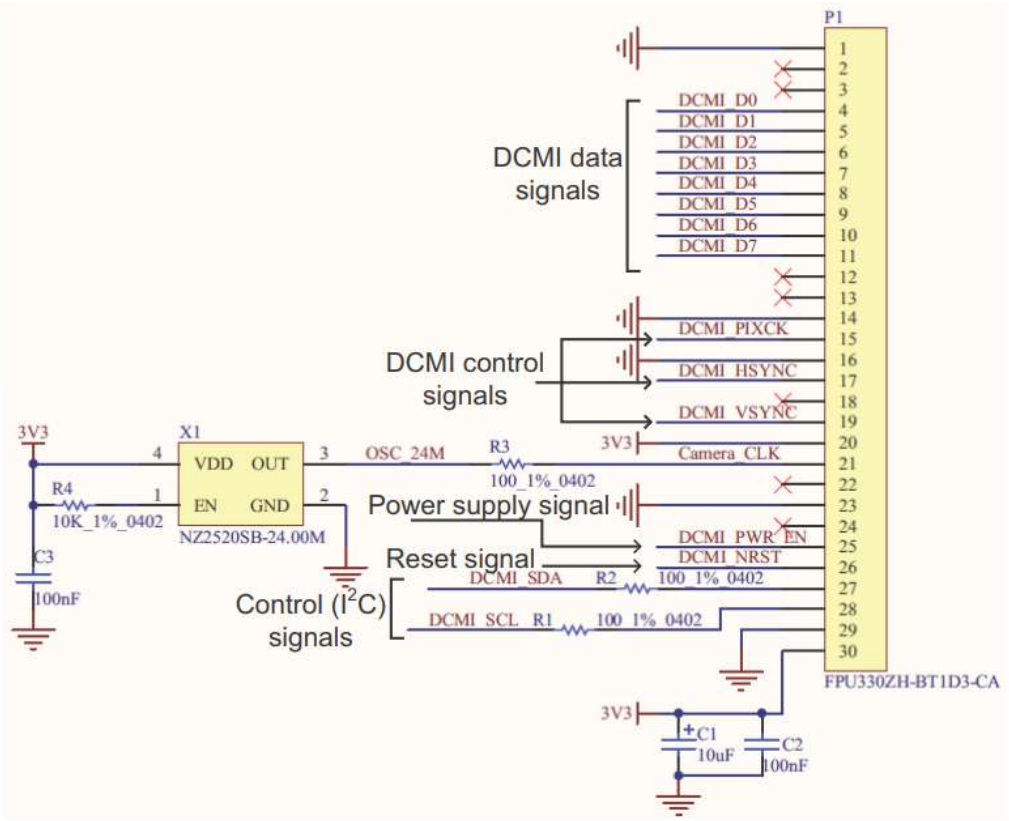
A placa STM32F746G-DISCOVERY (STMICROELECTRONICS, 2015) contém um microcontrolador STM32F746NGH6, com, entre outras características:

- Um núcleo ARM Cortex-M7 com uma unidade de ponto flutuante, cache L1 com 4 KB para dados e 4 KB para instruções, frequência de operação de até 216 MHz.
- 1 Mbyte de memória Flash para armazenamento não-volátil de dados e instruções.
- 340 Kbytes de RAM.
- 4 interface I2C.
- 1 interface de de câmera denominada DCMI, com um barramento de dados paralelo de 8 e os sinais usuais de sincronização (relógio de pixel, VSYNC, HSYNC).
- 1 interface USB Full-Speed e 1 interface USB High-Speed.

A placa contém também diversas interfaces e periféricos, de especial interesse para este projeto são:

- Um conector de câmera que permite acessar tanto a interface DCMI (para transporte de dados de vídeo) quanto uma das interfaces I2C do microcontrolador (para permitir que este leia e escreva registradores do sensor de imagem para configurá-lo). A imagem 11 mostra os sinais presentes no conector.
- Um conector USB que permite acessar a interface Full-Speed.
- Uma tela de cristal líquido (Liquid Crystal Display - LCD) com 272 x 480 pixels.

Figura 11: O conector de câmera da placa STM32F746G-DISCOVERY.



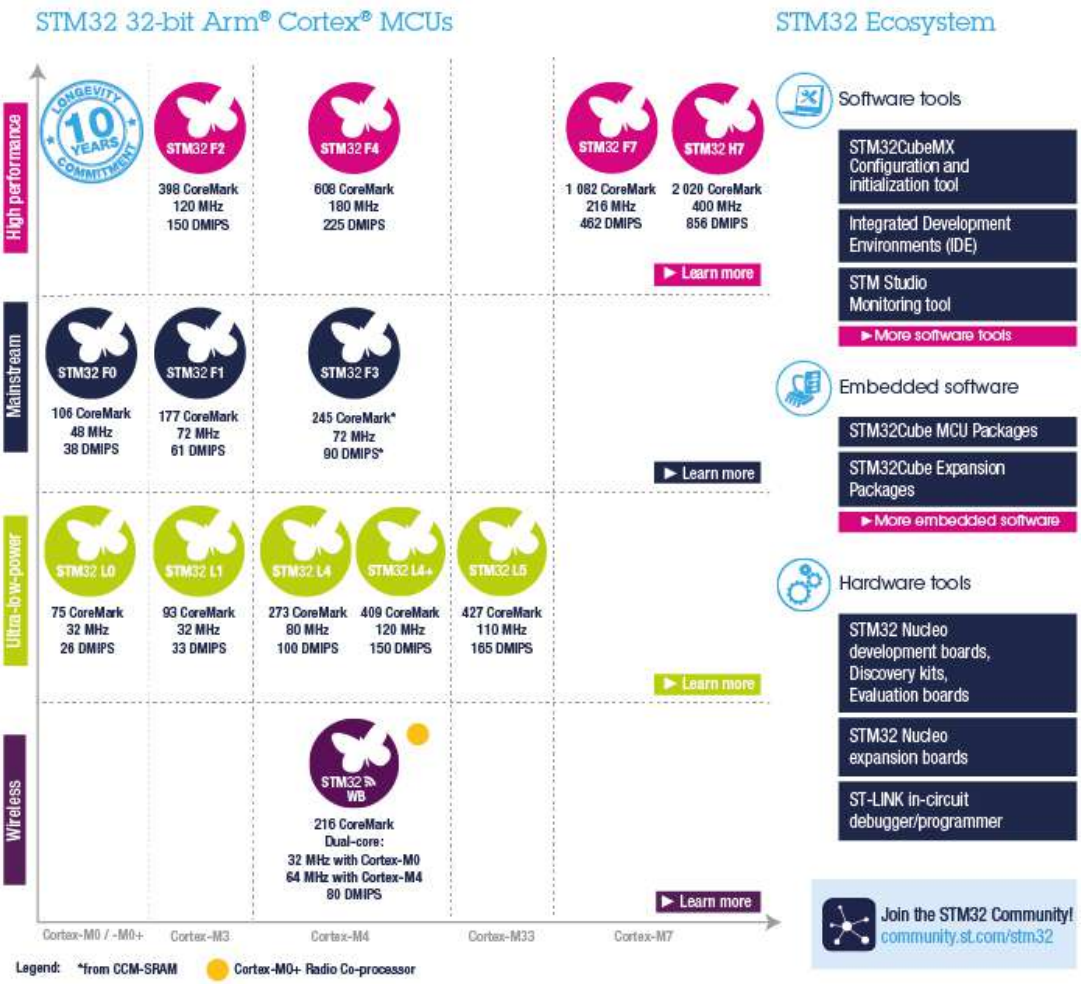
Fonte: (STMICROELECTRONICS, 2017a)

Alguns atores que influenciaram a escolha desta placa foram:

- Presença da interface de câmera, da interface USB (necessários para o funcionamento da arquitetura escolhida) e da tela LCD (útil para depuração).

- Alta performance do microcontrolador, tornando mais provável que o sistema final tenha uma boa performance. Ao mesmo tempo, a linha STM32 abrange uma vasta gama de microcontroladores com características diversas de preço, consumo de energia e performance (Figura 12). A portabilidade de código entre estes modelos é facilitada pelo pacote de STM32Cube, explorado na seção que trata do software embarcado. Assim, caso o projeto seja bem-sucedido, pode-se explorar mais facilmente sua portabilidade para um microcontrolador mais econômico e com menor consumo de energia no futuro.
- Familiaridade do autor com o ecossistema STM32.

Figura 12: Uma visão geral da linha de microcontroladores STM32.



Fonte: (STMICROELECTRONICS, 2018a)

5.1.2 STM32F4DIS-CAM

A placa STM32F4DIS-CAM contém um sensor de imagem OV9656, desenvolvido pela Omnivision Technologies. O sensor é utilizado para capturar um fluxo de vídeo de baixa resolução da mão do usuário, e enviá-lo para o microcontrolador. A escolha do sensor de imagem foi guiada pelo seguintes requisitos:

- Poder ser interfaceado de modo simples à placa STM32F746-DISCO através de seu conector de câmera.
- Ser capaz de capturar imagens em baixas resoluções (menores ou iguais à resolução QQVGA de 120 * 160) a 30 quadros por segundos, pois a quantidade de dados a processar aumenta sensivelmente em resoluções mais altas.
- Ser capaz de emitir imagens no formato de cor YUV, com uma componente de luminosidade e duas de crominância, e não apenas em formatos como RGB. Desse modo, o microcontrolador não precisaria realizar uma operação de conversão para recuperar a luminância de cada pixel, que é a informação utilizada em algoritmos que separam a frente e o fundo de uma imagem.

Uma revisão da documentação disponível para a placa STM32F4DIS-CAM (STMICROELECTRONICS, 2017b) e para o sensor OV9656 (OMNIVISION, 2006) mostra que o componente atende a estes requisitos.

5.2 Sistema Embarcado – Software

5.2.1 Compilação do Software Embarcado e Programação em Memória

Para a compilação do software embarcado, foi escolhido o compilador de linguagem C da *toolchain* (pacote de ferramentas de desenvolvimento) *arm-none-eabi-gcc* (ARM, 2018) – a versao da *toolchain* gcc (Gnu Compiler Collection) utilizada para o trabalho com microcontroladores baseados em processadores com arquitetura ARM.

Para o registro do programa compilado na memória do dispositivo foi utilizada a ferramenta st-flash, parte do pacote st-util (STLINK, 2018). Tal ferramenta é capaz de comunicar-se com subsistema ST-Link presente na placa STM32F746-DISCO através de uma conexão USB, e transmitir o arquivo binário a ser programado na memória de acordo com o protocolo esperado pelo dito subsistema.

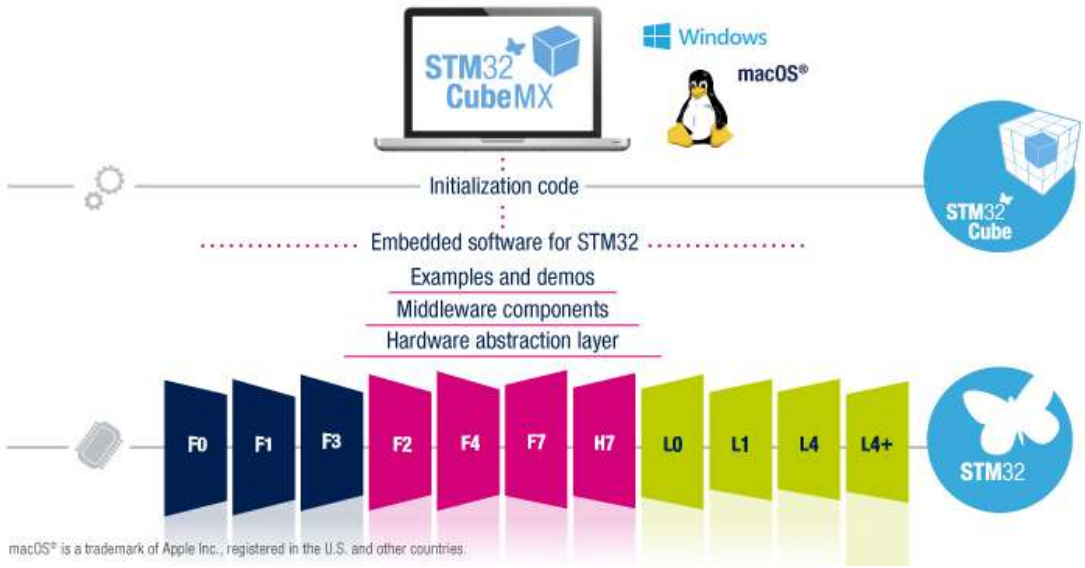
5.2.2 STM32CubeF7

A iniciativa STM32Cube, da empresa STMicroelectronics, oferece para cada microcontrolador da linha STM32 uma coleção de bibliotecas em linguagem C que inclui:

- Uma camada de abstração de hardware (denominada HAL, ou *Hardware Abstraction Layer*), que permitem configurar a operação dos diferentes periféricos do microcontrolador (timers, I2C, DCMI, etc.) através de estrturas de dados e funções padronizadas, como alternativa ao acesso direto a registradores.
- Um pacote de apoio à placas (denominado BSP, ou *Board Support Package*), que contém para cada placa de desenvolvimento (por exemplo, o componente STM32F746-DISCOVERY) um conjunto de *drivers* para os periféricos presentes na placa e para alguns dos periféricos que podem ser conectados a ela.
- Um conjunto de componentes de middleware adaptados para a produção de aplicações complexas com o microcontrolador, incluindo um sistema operacional em tempo real (FreeRTOS), uma pilha TCP/IP (lwip), um sistema de arquivos (FAT), uma biblioteca de gráficos (STEMWin), e uma biblioteca que implementa a pilha de protocolo para dispositivos USB e suas diversas classes de operação (HID, CDC, etc.).
- Exemplos de utilização das bibliotecas presentes no pacote.

A figura 13 esquematiza o ecossistema previsto pela iniciativa STM32Cube.

Figura 13: O ecossistema de hardware e software previsto pela iniciativa STM32Cube.



Fonte: (STMICROELECTRONICS, 2018b)

O pacote STM32CubeF7, o componente da iniciativa STM32Cube que tem como alvo os microcontroladores da linha STM32F7, foi utilizado para o densenvolvimento do projto. A utilização do pacote foi motivada pelas seguintes razões:

- Diminuir a complexidade da tarefa de inicializar e interagir com os periféricos da placa, especialmente a interface USB. Assim, torna-se possível partir mais rapidamente para a implementação da lógica de aplicação do sistema embarcado.
- Facilitar a portabilidade do software embarcado para outros microcontroladores da linha STM32.

5.3 Interface Gráfica

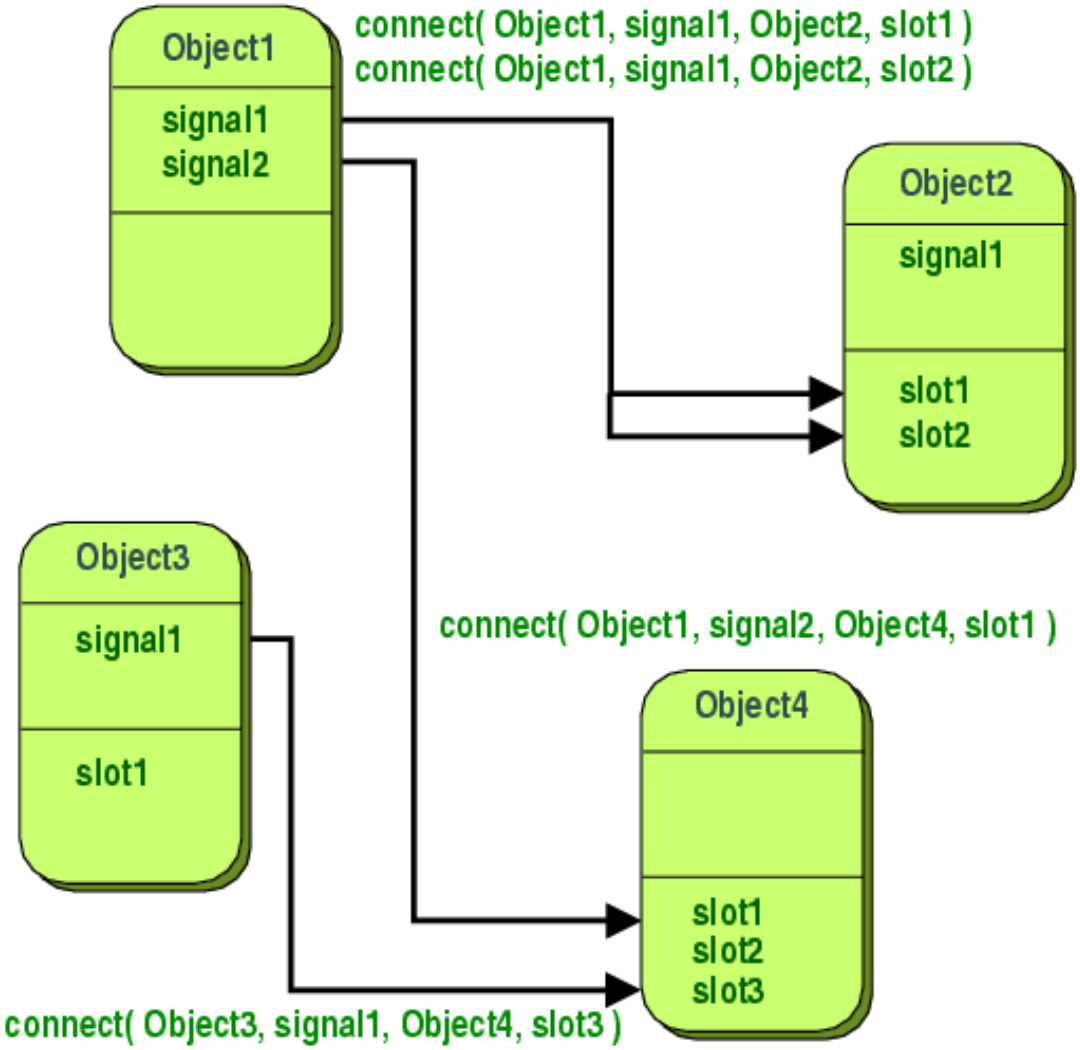
5.3.1 Framework Qt

A interface gráfica foi desenvolvida em linguagem C++ com o *framework* QT (QT COMPANY, 2018a), que oferece ferramentas para o desenvolvimento de aplicações gráficas nos sistemas operacionais Linux, Windows e macOS.

A utilização do framework foi motivada pelas seguintes razões:

- A presença do mecanismo de "signals" e "slots" do framework, o qual permite a implementação de interfaces gráficas movidas a eventos através da definição de métodos "signal", que podem ser conectados a um ou mais métodos "slot". Quando um "signal" é chamado com este argumento, os "slots" aos quais ele foi conectado são chamados com o mesmo argumento. Com este mecanismo, esquematizado na figura 14, apresenta-se uma implementação natural para uma interface gráfica movida pelo periférico de interação gestual: um objeto leitor recebe periodicamente qual gesto foi lido, e ao recebê-lo usa um "signal" para transmitir esta informação a "slots" de outros objetos do programa.
- Familiaridade do autor com o framework.

Figura 14: Diagrama do mecanismo de signal e slot do framework Qt.



Fonte: (QT COMPANY, 2018b)

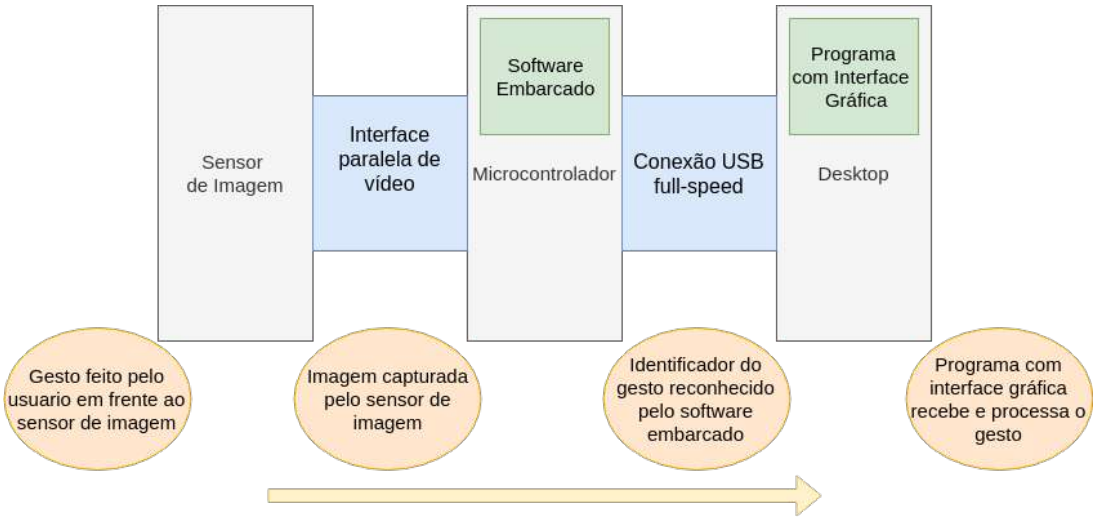
5.3.2 Libusb e HidAPI

A biblioteca HidAPI (OTT, 2018), elaborada em linguagem C, foi utilizada para a comunicação com o sistema embarcado, através do protocolo Human Interface Device (HID) de comunicação sobre USB (USB IMPLEMENTERS' FORUM, 2001a). A biblioteca admite múltiplos backends para a comunicação USB, no caso do projeto foi utilizada a biblioteca Libusb (LIBUSB, 2018).

6 Projeto e Implementação

A figura 3 do capítulo 2, reproduzida abaixo como a figura 3 é um retrato global do sistema que foi desenvolvido. O presente capítulo explora o projeto e implementação de seus dois grandes componentes: o sistema embarcado (duas colunas à esquerda) na seção 6.1, e a interface gráfica (coluna à direita) na seção 6.2.

Figura 15: Visão global do sistema desenvolvido.



Fonte: Autor

6.1 Projeto e Implementação do Sistema Embarcado

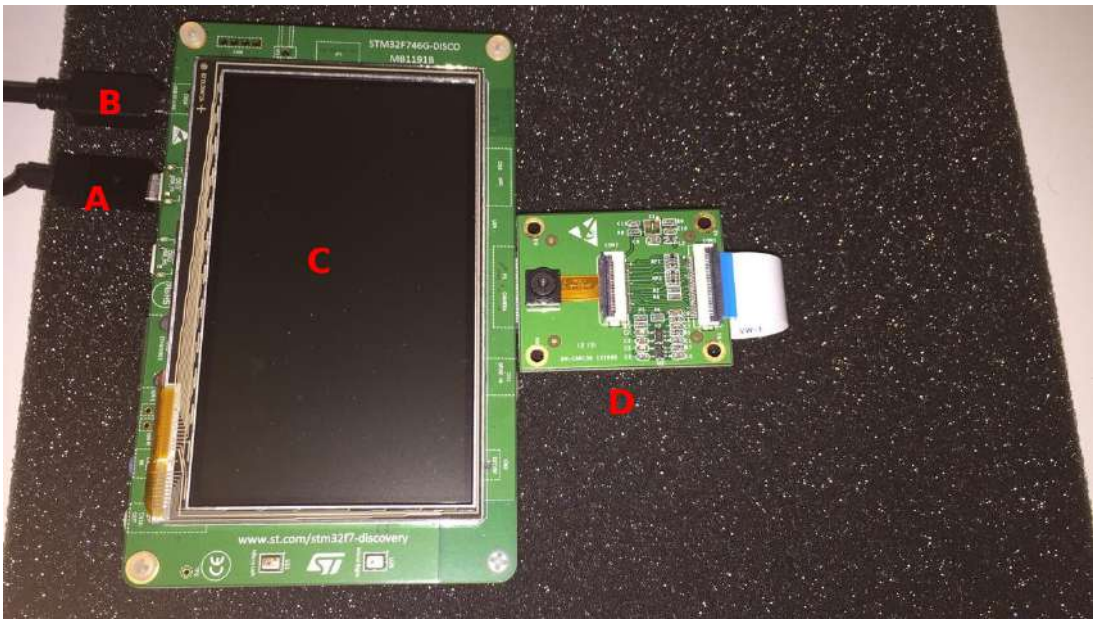
A figura 16 mostra a plataforma embarcada que foi utilizada ao longo do desenvolvimento do projeto, com seus principais componentes marcados com letras:

- **A:** Cabo USB 2.0 utilizado para transmitir relatórios periódicos com o identificador do gesto reconhecido pelo sistema. Também é responsável pela alimentação elétrica do sistema.
- **B:** Cabo USB 2.0 utilizado para acessar o sub-sistema STLink V2 da placa principal,

que permite realizar o *debug* do software embarcado e programar a memória não-volátil da placa principal com o arquivo binário contendo o software embarcado compilado.

- **C:** A placa principal do sistema, denominada 32F746G-DISCOVERY. Como discutido no capítulo 3, trata-se de uma placa de desenvolvimento desenvolvido pela companhia STMicroelectronics, que contém como componente principal o microcontrolador STM32F746NGH6 da mesma companhia. A tela de LCD da placa foi utilizada para depurar as rotinas de processamento de imagens ao longo do desenvolvimento do projeto.
- **D:** A placa STM32F4DIS-CAM, que contém o sensor de imagem OV9656 utilizado para filmar a mão do usuário. A conexão com a placa principal do sistema embarcado é realizada através de um cabo tipo "ribbon".

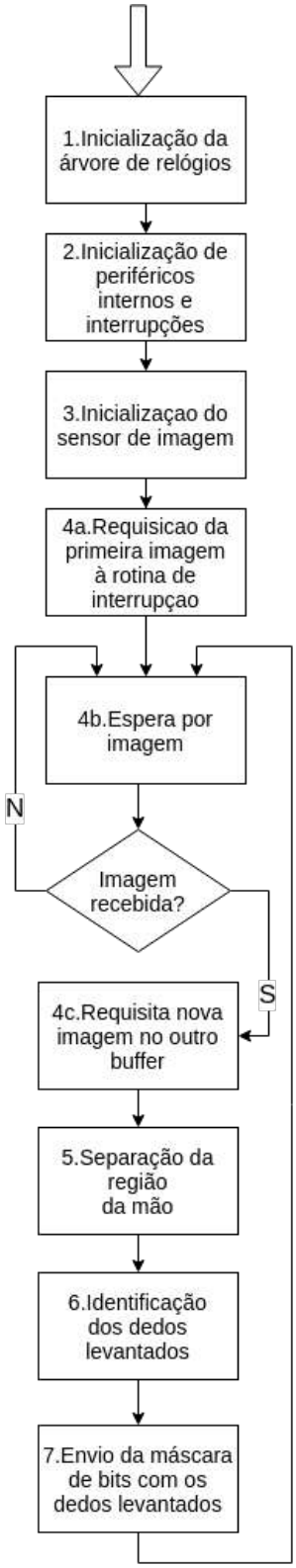
Figura 16: Foto do sistema embarcado utilizado no projeto.



Fonte: Autor.

Com a atividade de desenvolvimento do software embarcado, buscou-se implementar a sequência de passos retratada na figura 17. Tais etapas são exploradas nas próximas seções.

Figura 17: Visão geral da sequência de passos executada pelo software embarcado.

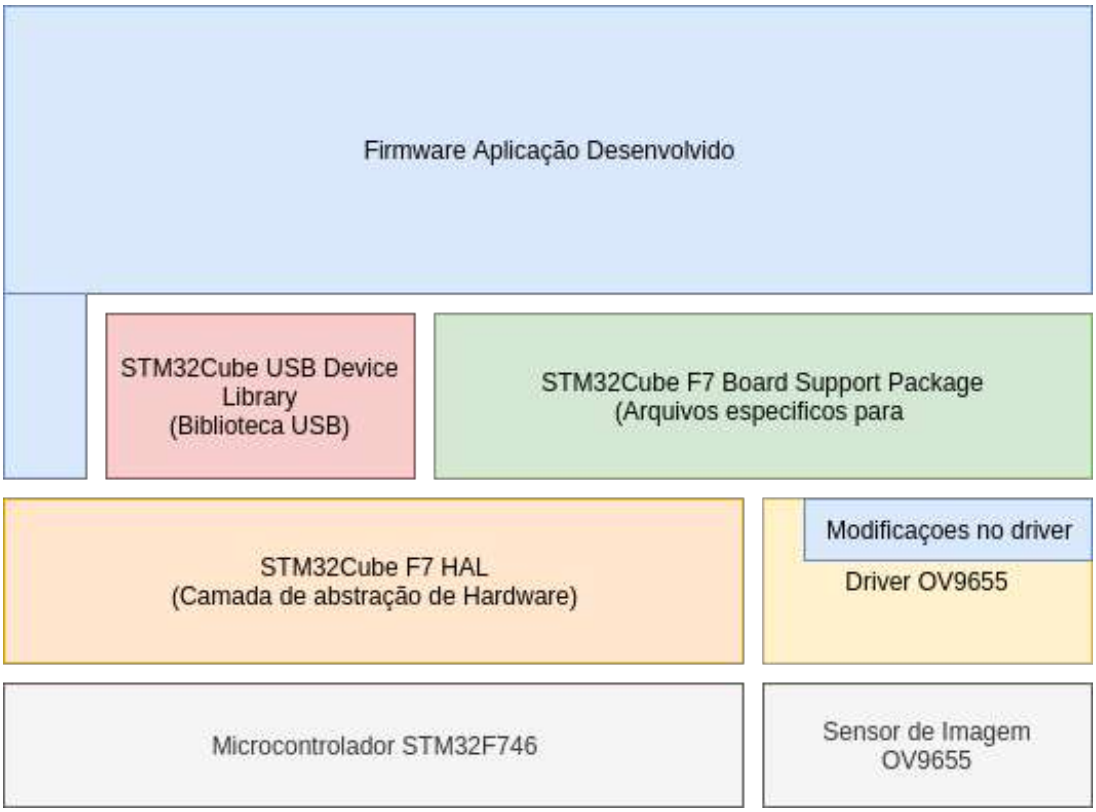


Fonte: Autor.

6.1.1 Inicialização e Configuração do Hardware

Os passos 1, 2 e 3 do diagrama da figura 17 correspondem à configuração e inicialização do hardware do sistema. A figura 18 mostra a pilha de software mostra a pilha de software utilizada para a interação em baixo nível com o hardware do sistema embarcado: as bibliotecas Board Support Package (BSP), Hardware Abstraction Layer (HAL) e USB Device Library, assim como o driver do sensor de imagem OV9656, são componentes do pacote STM32CubeF7.

Figura 18: Pilha de software utilizada na concepção do sistema embarcado.



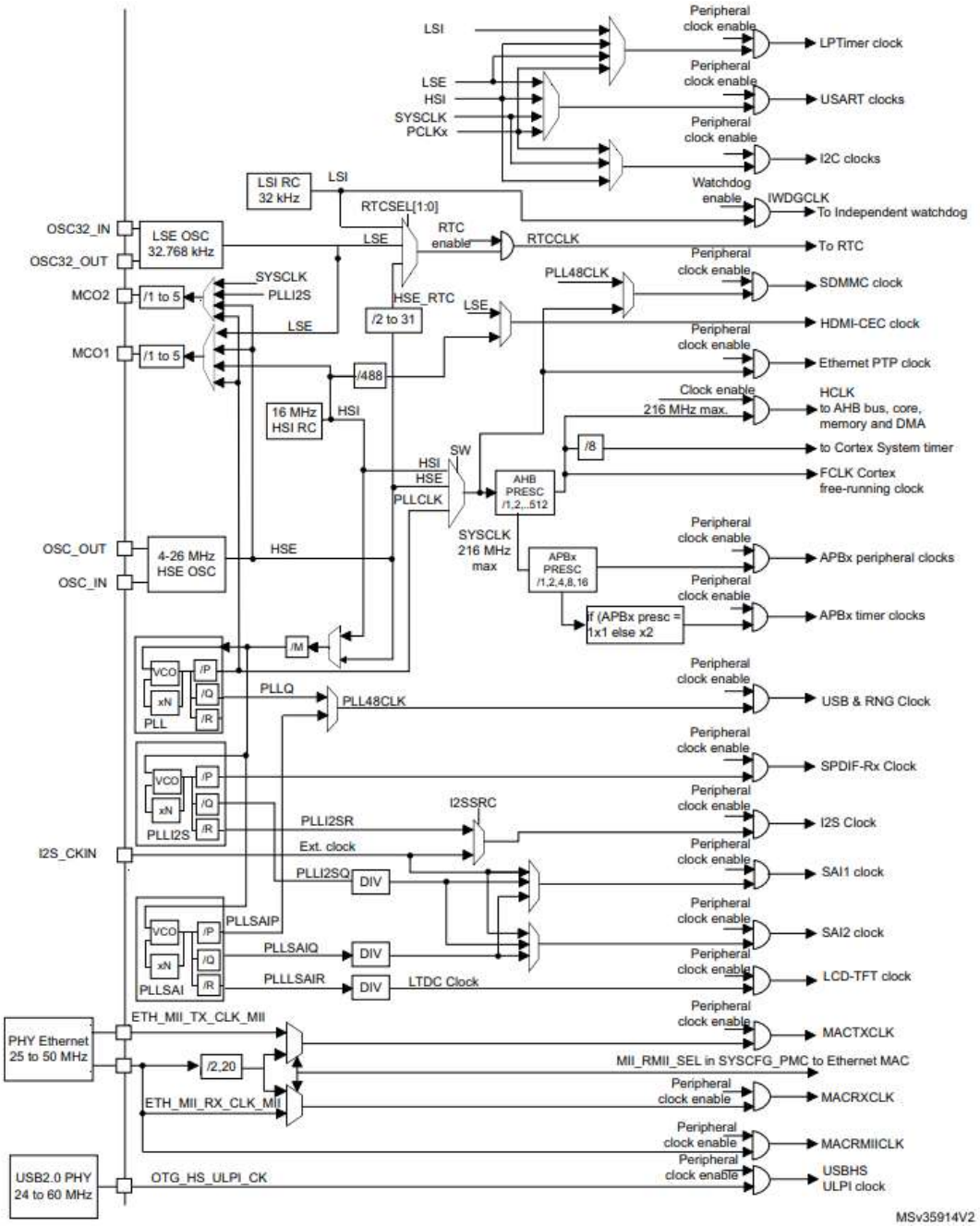
Fonte: Autor.

6.1.1.1 Inicialização da Árvore de Relógios

Microcontroladores modernos contém complexos sistemas de geração e distribuição de sinais de relógio, denominados "árvores de relógios" (clock tree). A figura 19 mostra um diagrama da árvore de relógio do microcontrolador STM32F746. Um passo primordial da inicialização do microcontrolador é a configuração desta estrutura, eventualmente uma tarefa complexa. A ferramenta (STMICROELECTRONICS, 2018c) permite, entre outras funções, gerar uma função de inicialização da árvore de relógios de microcontroladores STM32 em função de quais periféricos serão utilizados e das frequências de operação

desejadas.

Figura 19: Árvore de relógios do microcontrolador STM32F746.

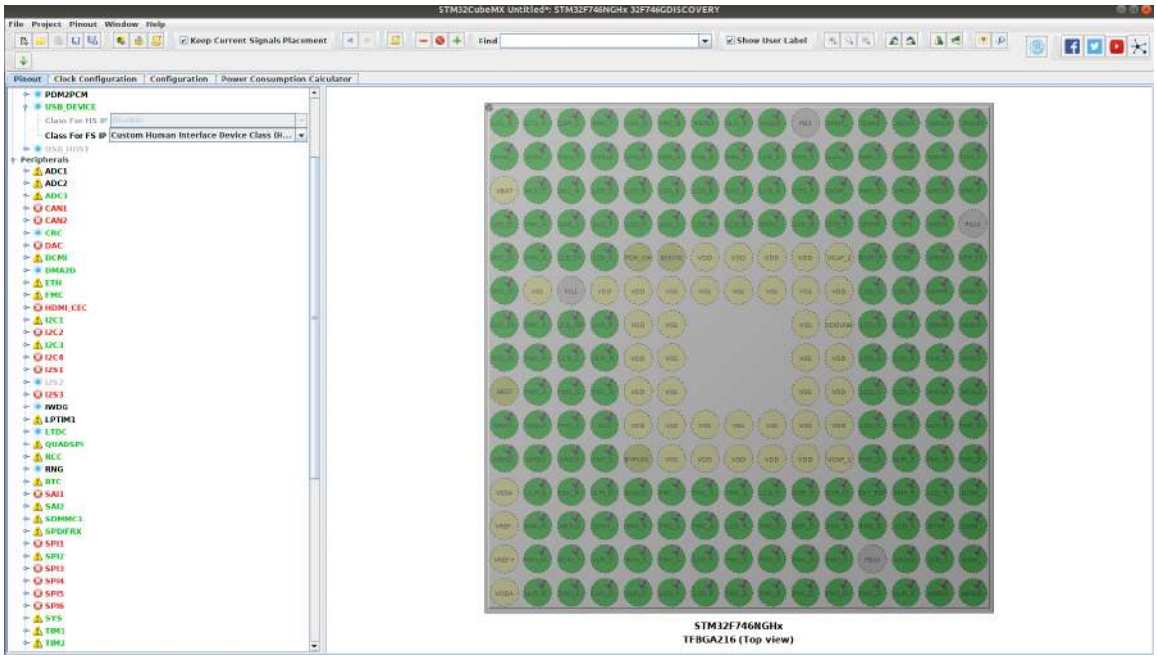


Fonte: (STMICROELECTRONICS, 2018d).

Tais opções são selecionadas através de uma interface gráfica: a figura 20 mostra a tela que permite selecionar os periféricos a ser utilizados, enquanto a figura 21 mostra uma representação da árvore de relógio, que permite selecionar as frequências dos diferentes sinais de relógio distribuídos para os periféricos. Ambas as imagens mostram as configurações que foram utilizadas neste projeto. Após concluídas as configurações, foi ge-

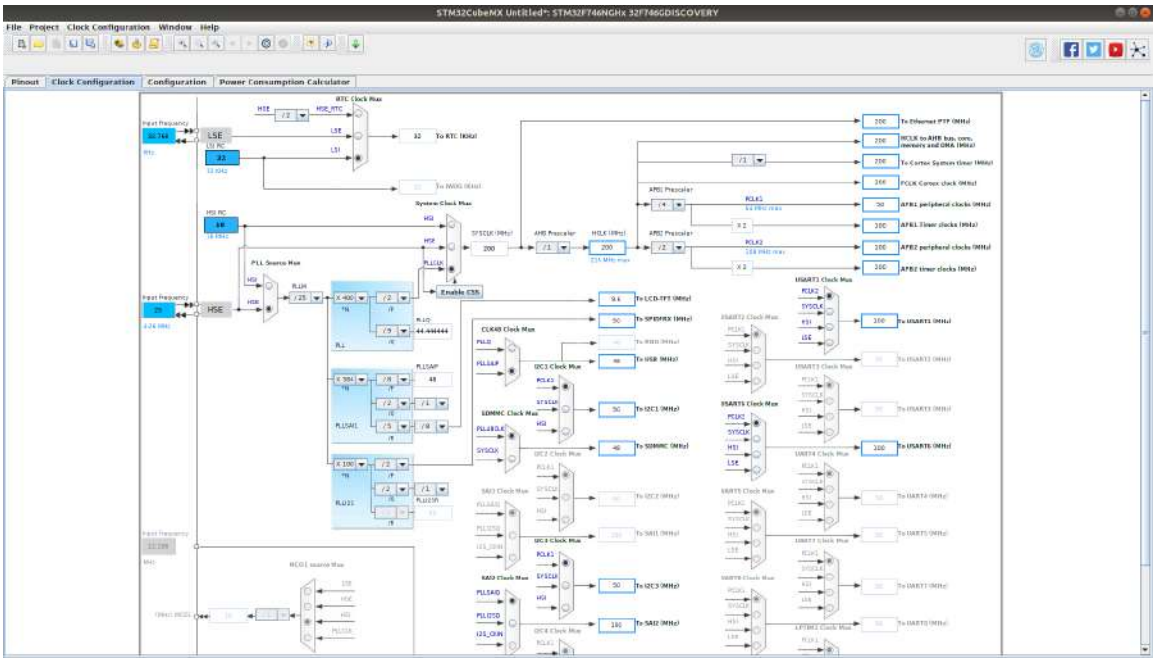
rada uma função de inicialização que chama as funções necessárias da biblioteca HAL para fornecer os sinais de relógio desejados para os diferentes componentes do microcontrolador utilizados no projeto.

Figura 20: Tela do software STM32CubeMX que permite escolher quais periféricos serão utilizados.



Fonte: Autor

Figura 21: Tela do software STM32CubeMX que permite configurar a árvore de relógios do projeto.



Fonte: Autor

6.1.1.2 Inicialização dos Periféricos e Interrupções

Após passar reber sinais de relógio, os periféricos do microcontrolador STM32F746G devem ser configurados. Os periféricos utilizados neste projeto incluem:

- A interface de câmera DCMI do microcontrolador, utilizada para receber imagens do sensor OV9656, como discutido na seção 6.1.2.
- Uma das interfaces I2C do microcontrolador, utilizada para configurar o sensor de imagem através da leitura e escrita de registradores, como discutido na seção 6.1.1.3.
- A interface USB Full-Speed do microcontrolador, utilizada para a comunicação com o computador principal, como discutido no capítulo 2 e na seção 6.1.4.
- O mecanismo de acesso direto à memória (DMA) do microcontrolador, utilizada para transportar dados de imagem do periférico DCMI para a memória RAM, em paralelo com o processamento de imagem realizado pelo processador, como discutido na seção 6.1.2.
- A interface de memória SDRAM do microcontrolador, utilizada para acessar o buffer de imagem da tela LCD que é utilizada para inspecionar o funcionamento das rotinas

de tratamento de imagens na seção 6.1.3.

As funções utilizadas para realizar tais configurações são detalhadas no Apêndice A.

Durante a inicialização do sistema também é necessário ativar as linhas de interrupção que serão utilizadas pelo periféricos, que incluem:

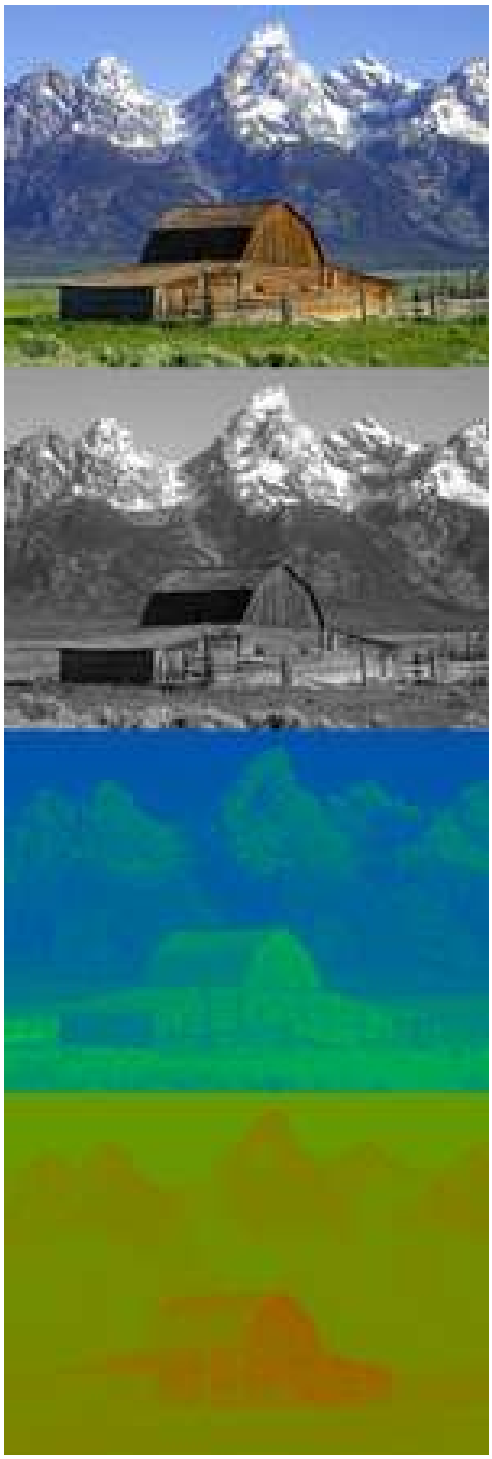
- A interrupção que indica a ocorrência de eventos na interface USB.
- A interrupção do canal da interface DMA utilizado para transportar dados de vídeo para a memória RAM.
- A interrupção IT_LINE da interface DCMI, ativada sempre que se encerra a recepção de uma linha da imagem, evento indicado pela mudança do sinal HSYNC para o valor lógico baixo.

6.1.1.3 Inicialização do Sensor de Imagem

O sensor de imagem OV9565 é configurado através da interface I2C que o conecta ao microcontrolador STM32F746. O driver "ov9656.c/.h" que integra a biblioteca BSP do pacote STM32CubeF7 foi utilizada como base para tal configuração neste projeto.

Uma dificuldade que se manifestou foi o fato de que o driver inicializa o sensor de modo que este emita imagens no formato RGB 5:6:5, em que cada pixel é codificado em 16 bits, com 5 dedicados à componente vermelho, 6 à verde e 5 à azul. Para a aplicação dos algoritmos de processamento de imagem utilizados para diferenciar o primeiro plano e o plano de fundo de uma imagem, é preferível que a imagem esteja em um formato de cor que armazene em componentes separadas a luminosidade e a cor do pixel. Um formato com essa característica, o YUV 4:2:2 (ilustrado na figura 22) é suportado pelo sensor de imagem OV9656. Assim, o datasheet do sensor de imagem OV9565 (OMNIVISION, 2006) foi estudado, e o driver foi modificado para inicializar o sensor de imagem para capturar imagens de tamanho 160x120 pixels a 30 quadros por segundo, no formato de cor YUV 4:2:2.

Figura 22: De cima para baixo: uma imagem colorida, sua componente Y, sua componente U e sua componente V.



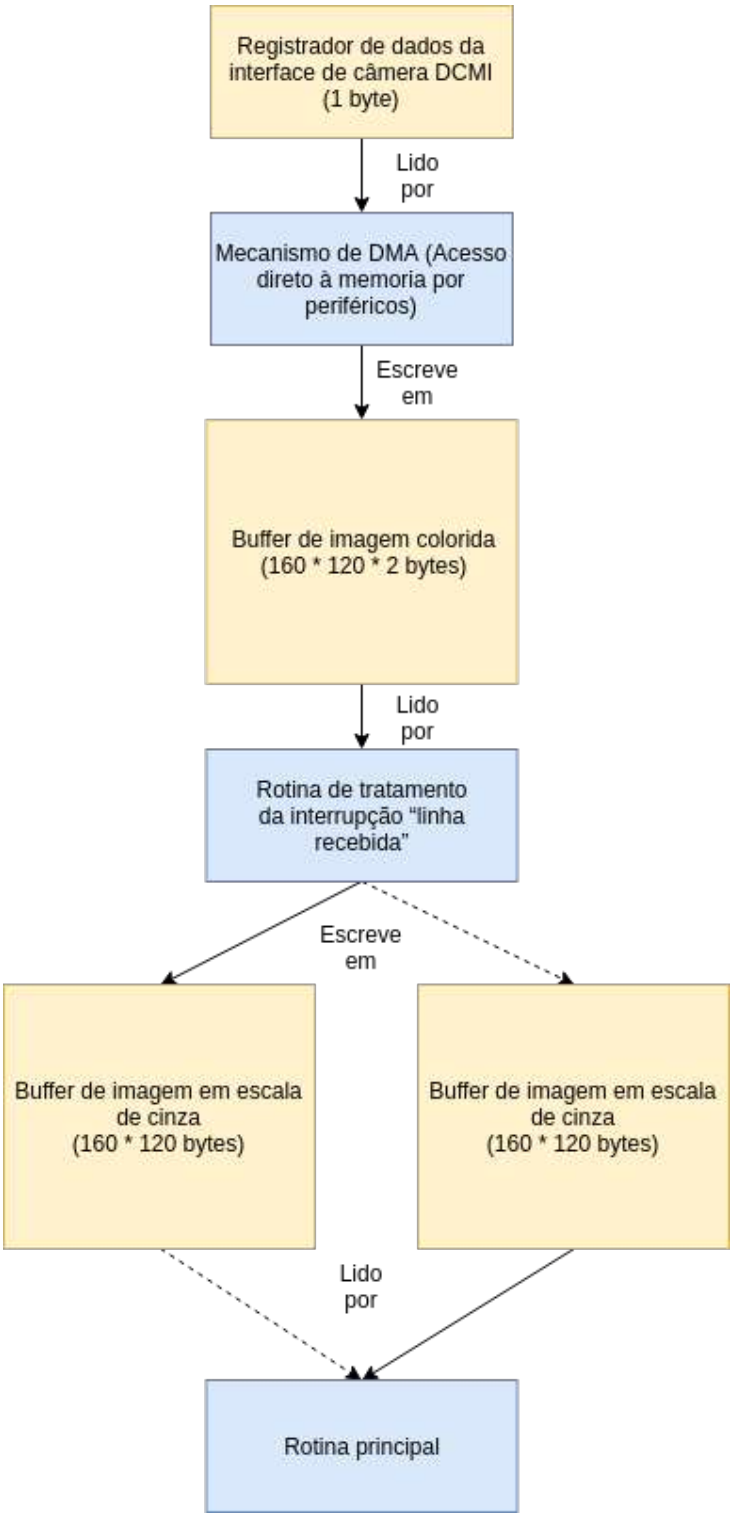
Fonte: (WIKIPEDIA, 2015)

6.1.2 Recepção de Imagens

Os passos 4a a 4c do diagrama da figura 17 correspondem à recepção de imagems do sensor OV9656 através da interface de câmera DCMI.

A figura 23 fornece uma visão global de como os dados de imagem são transportados para a rotina principal do software embarcado. A interface de câmera DCMI (STMICROELECTRONICS, 2017a) recebe um byte de imagem por vez, que é armazenado em seu registrador de dados. O mecanismo de acesso direto à memória (DMA) do microcontrolador STM32F746G foi configurado anteriormente para transportar cada byte recebido neste registrador para um buffer na memória RAM. A posição para qual o mecanismo de DMA transporta o conteúdo do registrador é incrementada a cada byte transportado, e resetada ao endereço inicial do buffer ao final da recepção de um quadro de vídeo.

Figura 23: Estratégia com dois *buffers* para transporte dos dados de vídeo à rotina principal do sistema embarcado.



Fonte: Autor

Ao final da recepção de cada linha de imagem, a interface DCMI aciona sua interrupção IT_LINE. Foi definida uma rotina de tratamento para esta interrupção, que recu-

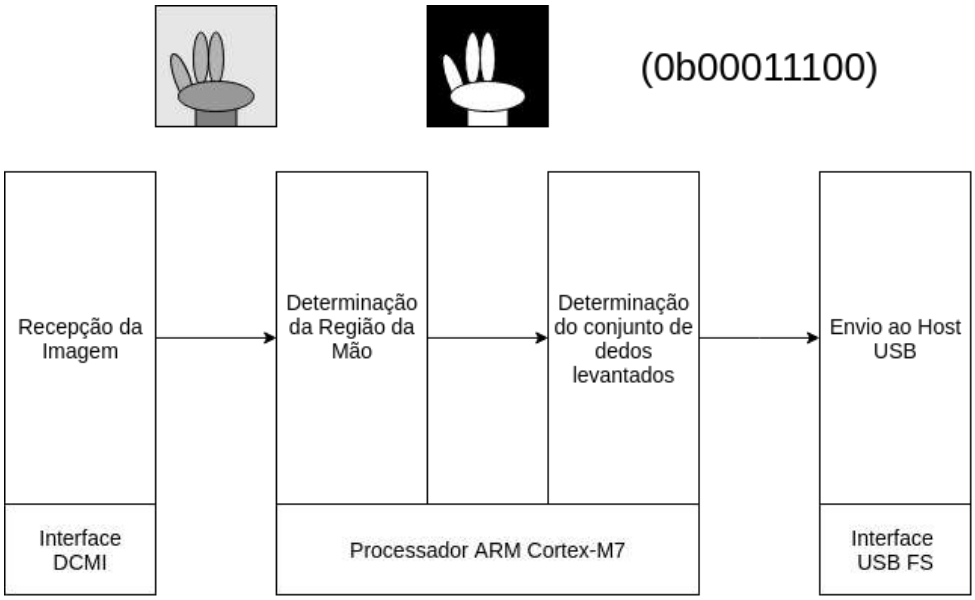
pera as informações de luminosidade presentes na linha recebida e as transporta para um buffer. Ao mesmo tempo, a rotina principal processa a imagem recebida anteriormente em um segundo buffer.

Assim que a rotina de interrupção termina de preencher o primeiro buffer e a rotina principal termina de processar a imagem no segundo buffer, as duas rotinas comunicam-se para que os dois buffers troquem de papéis. Tal comunicação é realizada através de variáveis compartilhadas, que recebem o modificador volatile da linguagem C.

6.1.3 Processamento de Video

Os passos 5 e 6 do diagrama da figura 17 correspondem ao processamento da imagem recebida do sensor de imagem para determinar o conjunto de dedos levantados e abaixados. A figura 24 é um diagrama de como se passa tal processamento de imagens desde a recepção de informações do sensor OV9656 até o envio da máscara de bits representando o gesto identificado.

Figura 24: Etapas de processamento de video realizadas pelo sistema embarcado.



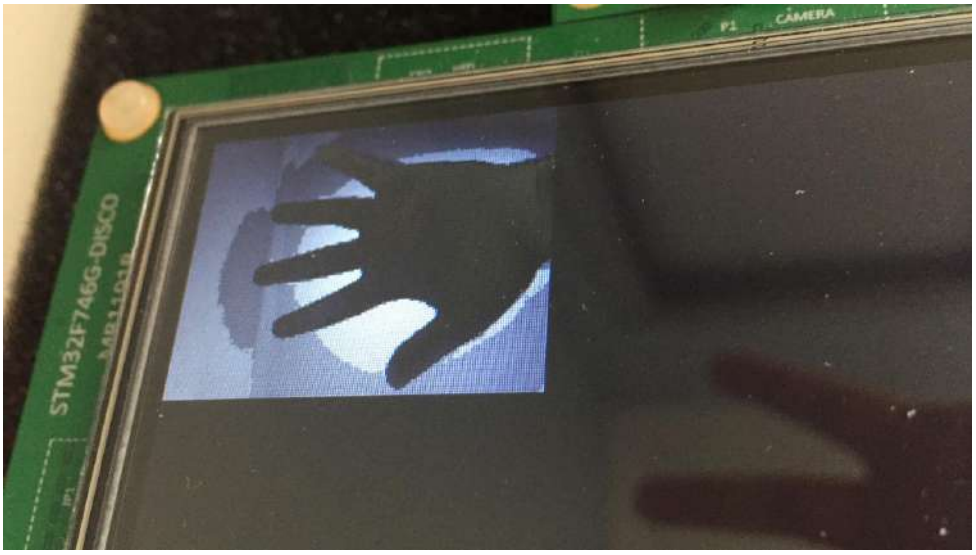
Fonte: Autor

6.1.3.1 Separação da Região da Mão

A imagem recebida do sensor de imagem é composta de pixels no formato YUV 4:2:2, com uma componente de 8 bits de luminosidade para cada pixel (Y) e duas componentes de cor de 8 bits cada para cada par de pixels (U e V). Para o processamento da imagem,

as componentes U e V são descartadas, e obtém-se uma imagem em escala de cinza, como mostrado na figura 25, exibida na tela LCD da placa 32F746G-DISCOVERY. Como a componente Y é codificada em 8 bits, existem $2^8 = 256$ níveis de cinza que os pixels da imagem podem assumir (0 a 255).

Figura 25: Imagem em escala de cinza capturada pelo sensor de imagem.



Fonte: Autor

Neste projeto, supomos que o sensor de imagem capturou uma imagem da mão em frente a um fundo uniforme em termos de iluminação e com uma maior incidência de luz do que a palma da mão do utilizador. Nesta situação, podemos empregar o algoritmo de Otsu (OTSU, 1979) para determinar um nível de cinza t, tal que todos os pixels com luminosidade maior ou igual a t são considerados como parte do plano de fundo da imagem, e todos os outros pixels são considerados como parte do primeiro plano da imagem (portanto, a região da mão).

O algoritmo baseia-se em procurar um valor de t que minimize a variância ${\sigma_w}^2(t)$ de valores de luminosidade no interior das classes de pixel "0"(luminosidade inferior a t) e "1"(luminosidade superior ou igual t), calculada como:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{6.1}$$

Onde ${\sigma_0}^2$ e ${\sigma_1}^2$ são as variâncias internas às duas classes, e os pesos ω_0 e ω_1 são calculados a partir da função $p(i)$ (a proporção de pixels com intensidade i no total de pixels da imagem) da seguinte maneira:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i) \tag{6.2}$$

$$\omega_1(t) = \sum_{i=t}^{255} p(i) \tag{6.3}$$

Como explicado em (TURKEL, 2012), minimizar a variância intra-classes ${\sigma_w}^2(t)$ equivale a maximizar a variância inter-classes ${\sigma_b}^2(t)$ (pois a variância total, igual a somas das variâncias intra- e inter-classes é constante e independe de t):

$$\sigma_b^2(t) = \omega_0(t)[1 - \omega_0(t)][\mu_0(t) - \mu_1(t)]^2 \tag{6.4}$$

Onde $\mu_0(t)$ e $\mu_1(t)$ são os valores médios das duas classes, calculados como:

$$\mu_0(t) = \sum_{i=0}^{t-1} \frac{ip(i)}{\omega_0(t)} \tag{6.5}$$

$$\mu_1(t) = \sum_{i=t}^{255} \frac{ip(i)}{\omega_1(t)} \tag{6.6}$$

Em vista do discutido acima, existe uma maneira iterativa eficiente para calcular a equação 6.4 para os 256 valores possíveis de t, em busca do valor de t que maximiza esta variável. Para tanto, basta calcular $\mu_0(t)$, $\mu_1(t)$, $\omega_0(t)$ e $\omega_1(t)$ segundo as definições destas variáveis, e para valores maiores de t utilizar os passos indutivos:

$$\omega_0(t+1) = \omega_0(t) + p(i) \tag{6.7}$$

$$\mu_0(t+1) = \frac{\omega_0(t)\mu_0(t) + (t+1)p(t+1)}{\omega_0(t+1)} \tag{6.8}$$

$$\mu_1(t+1) = \frac{\mu_T + \omega_0(t+1)\mu_0(t+1)}{1 - \omega_0(t+1)} \tag{6.9}$$

$$\mu_T = \sum_{i=0}^{255} p(i) \tag{6.10}$$

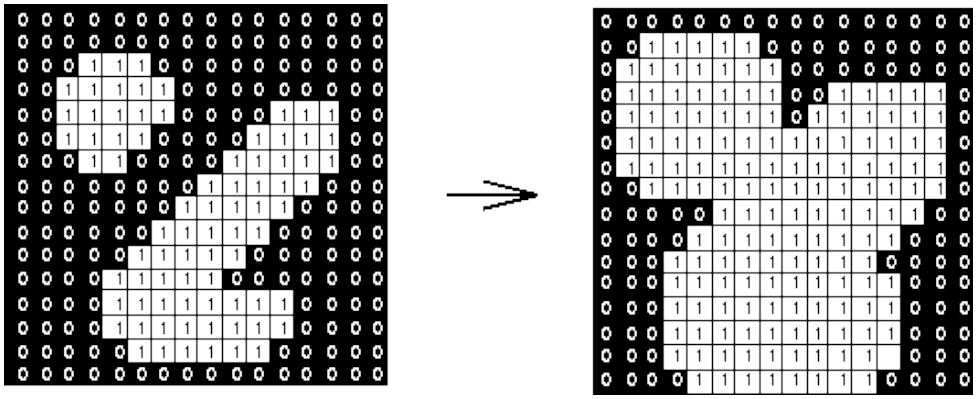
A figura 26 mostra o resultado da aplicação do algoritmo de Otsu a uma imagem capturada pelo sensor OV9656. Podemos ver que a região da mão é determinada, mas nota-se que há uma quantidade de pixels no interior da mão que erroneamente considerados como parte do fundo. O fato de que o conjunto de pixels erroneamente classificados varia de imagem para a imagem, e o isolamento de tais pixels, indica-se que se trata de um fenômeno de "pepper noise", pixels isolado que são capturados pelo sensor de imagem com uma luminosidade anormalmente alta, devido aos ruídos inerentes à captura de imagem.

Figura 26: Imagem binarizada através do algoritmo de Otsu, exibindo "Pepper noise"

Fonte: Autor

Textos didáticos como (FISHER et al., 2004) mencionam que a operação de dilatação morfológica pode ser utilizada para eliminar tal "pepper noise". A figura 27 mostra um exemplo de aplicação de uma variedade desta operação, na qual a matriz binária à esquerda é a entrada e a matriz à direita é a saída. Os células marcados com '1' na saída correspondem à união dos células marcados com '1' na entrada com os células marcados com '0' na entrada que são vizinhos (horizontalmente, verticalmente, ou pelas diagonais) de alguma célula marcada com um '1' na entrada. Outras variedades da operação consideram outros conjuntos de células vizinhos para determinar quais células passam de '0' a '1'.

Figura 27: Exemplo de aplicação de uma operação de dilatação sobre uma imagem binária.



Fonte: (FISHER et al., 2004)

Optou-se por empregar esta mesma variedade da operação de dilatação morfológica para eliminar o "pepper noise" na matriz binária resultante da aplicação do algoritmo de Otsu. A figura 28 mostra que a aplicação da operação de dilatação efetivamente elimina o "pepper noise". No entanto, ainda resta uma questão a ser confrontada: existem regiões conexas de pixels externas à mão que são consideradas pelo algoritmo de Otsu como parte do plano de frente, pode ser visto nos cantos da tela LCD na figura.

Figura 28: Resultado da aplicação de uma operação de dilatação sobre a saída do algoritmo de Otsu.

Fonte: Autor

Para isolar a componente conexa de pixels da região da mão, foi utilizado o algoritmo de busca em largura (FEOFILOFF, 2010), da seguinte maneira:

1. Verificar se o pixel central da imagem faz parte do primeiro plano da imagem. Se não, considera-se que a mão do usuário não esta presente na imagem, e o processo se encerra. Se sim, o pixel é marcado como parte da região da mão, e suas coordenadas são inseridas em uma fila (FIFO) First-in-First-out Q.

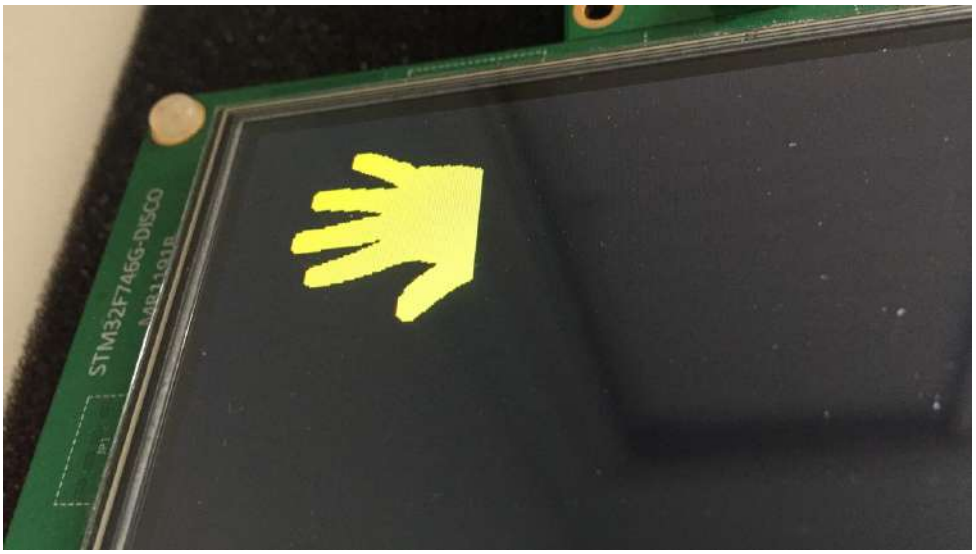
2. Se a fila Q estiver vazia, o processo se encerra, e considera-se o conjunto de pixels que foram marcados como parte da região da mão como a saída do algoritmo. Se não, as coordenadas do próximo pixel P são recuperadas da frente da fila.

3. Para cada vizinho horizontal e vertical de P que faz parte do plano de frente da imagem mas ainda não foi marcado como parte da região da mão, o vizinho é marcado como parte da região da mao e suas coordenadas são inseridas em P.

4. O passo 2 é executado novamente.

A figura 29 mostra a região da mão isolada através da aplicação em sequência do algoritmo de Otsu, da operação de dilatação e da busca em largura.

Figura 29: Resultado da seleção da componente conexa de pixels do primeiro plano no centro da imagem.



Fonte: Autor

6.1.3.2 Identificação dos Dedos Levantados

Sobre a imagem binária produzida pela etapa de separação da região da mão, aplica-se um algoritmo de identificação do conjunto de dedos levantados. A abordagem adotada

para identificar o conjunto de dedos levantados foi baseado no método de contagem de dedos levantados apresentando no artigo (MALIMA; OZGUR; CETIN, 2006).

O método do artigo começa por calcular o centróide $(\bar{x}, \bar{y})$ da região da mão. Para tanto, supondo que foram determinados k pixels com coordenadas (x_i, y_i) como parte da região da mão do usuário, calculamos:

$$\bar{x} = \frac{\sum_{i=1}^{k} x_i}{k} \tag{6.11}$$

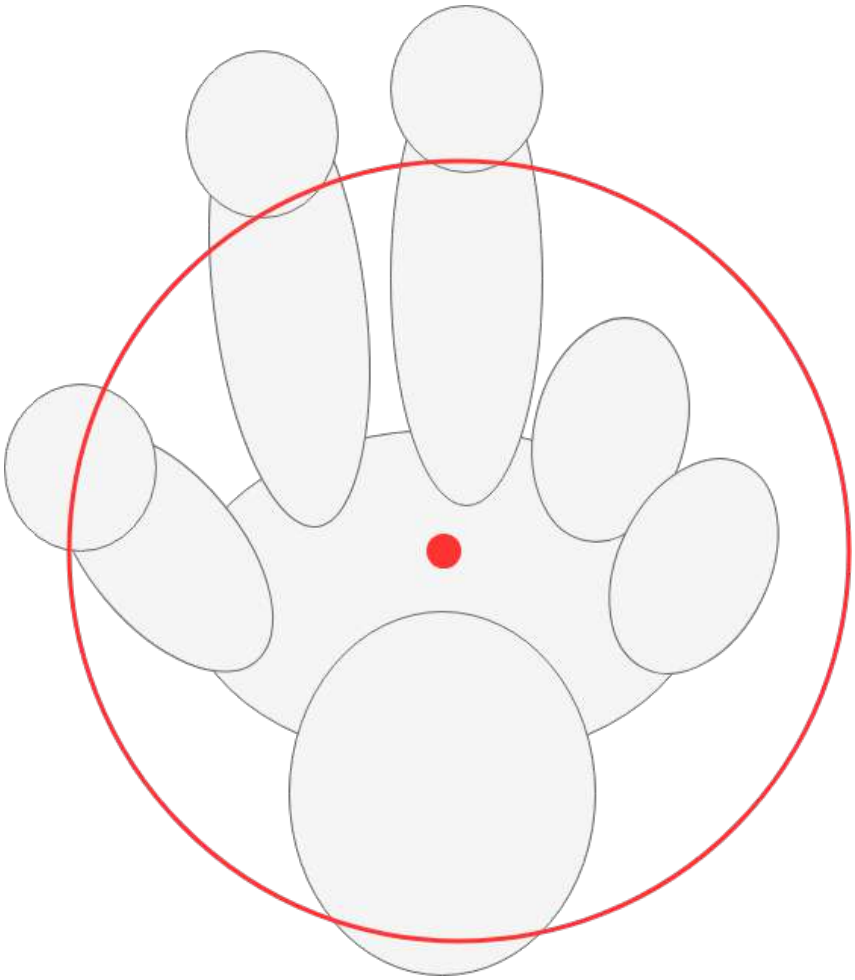
$$\bar{y} = \frac{\sum_{i=1}^{k} y_i}{k} \tag{6.12}$$

Em seguida, calculamos a maior distância d de um ponto da mão para o centróide, ou seja, percorremos todos os pixels (x_i, y_i) e escolhemos um índice j que maximize:

$$d = \sqrt{(x_j - \bar{x})^2 + (x_y - \bar{y})^2} \tag{6.13}$$

O artigo então indica que se trace um círculo de raio $0.7 \times d$ centrado em $(\bar{x}, \bar{y})$, e que a contagem de dedos será igual à quantidade de vezes que o círculo passa da região externa à mão à região interna à mão, menos um (devido ao cruzamento com o pulso). Esta abordagem é retratada na figura 30, no caso da figura, são contados três dedos.

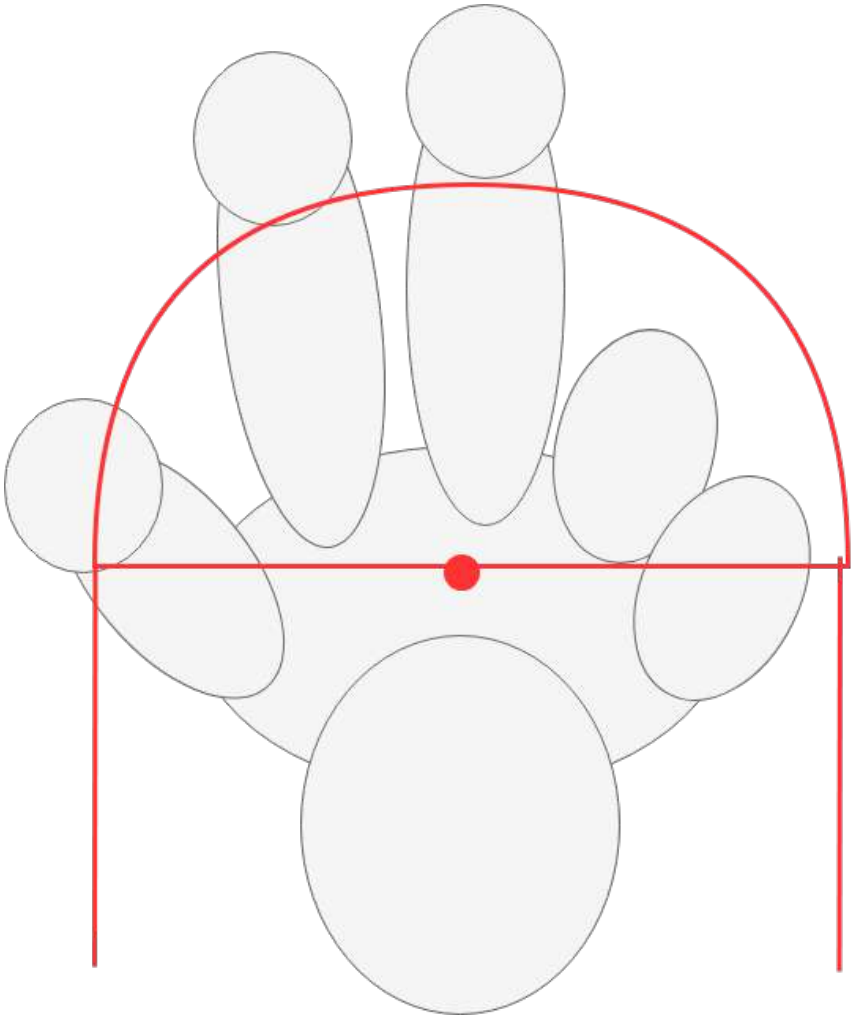
Figura 30: Utilização de um círculo ao redor do centróide da mão para contar os dedos levantados.



Fonte: Autor

A partir deste ponto, esta seção abordará como o método do artigo para o presente trabalho. Como no presente trabalho supõe-se que a mão estará orientada no mesmo eixo que o sensor de imagem, explorou-se traçar apenas um semicírculo, com a parte do círculo que esteja acima do centróide na imagem. Tal abordagem mostrou-se eficiente em geral, mas eventualmente deixava de capturar o polegar quando este estivesse estendido para baixo do centróide. Para resolver esse defeito, decidiu-se estender os limites do semicírculo para a base da imagem, de modo que se cruze um polegar estendido mesmo nesta situação. A figura 31 mostra esta nova abordagem.

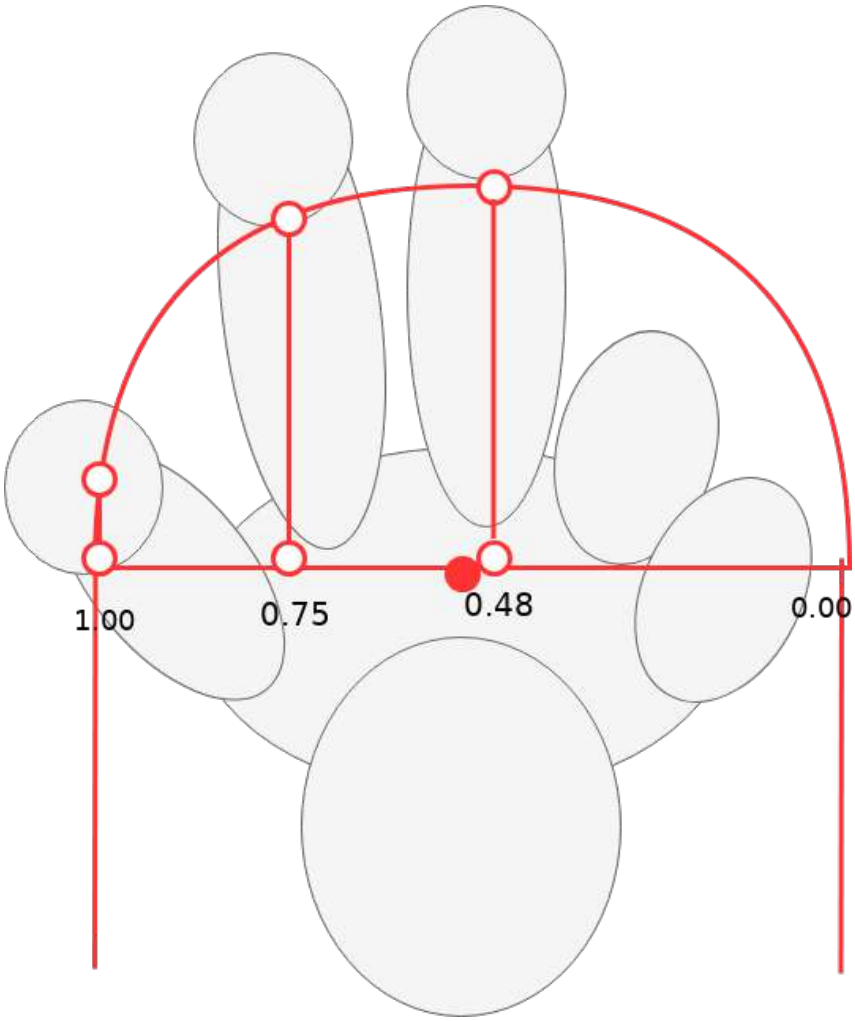
Figura 31: Semicírculo e retas utilizados para a localização dos dedos levantados.



Fonte: Autor

Para ir além da contagem e dedos e determinar quais os dedos levantados, optou-se por um algoritmo baseado na projeção, em uma reta horizontal que passa pelo centróide, do ponto médio das intersecções dos dedos com o círculo utilizado para a contagem. As entradas do algoritmo são o número N de dedos levantados que foram contados, e as coordenadas de suas projeções na linha horizontal, em um sistema de coordenadas que tem a extremidade direita do semicírculo como 0 e a extremidade esquerda como 1, como ilustrado na figura 32.

Figura 32: Exemplo da projeção, em uma linha horizontal que passa pelo centróide da mão, das intersecções dos dedos o com círculo de contagem.



Fonte: Autor

Durante a execução da algoritmo, para cada conjunto possível E de dedos levantados que tenha N dedos levantados:

1. Elabora-se uma lista crescente dos valores esperados $e_i, 0 \leq i \leq N-1$ para as coordenadas das projeções dos dedos do conjunto E.
2. Elabora-se uma lista crescente dos valores c_i medidos na prática.
3. A pontuação do conjunto E é calculada como a soma dos valores absolutos das diferenças $c_i - e_i$, para $0 \leq i \leq N-1$.

O conjunto de dedos que obtenha a menor pontuação e selecionado como o que mais provavelmente está sendo realizado pelo usuário. Cabe notar que os conjuntos de dedos são manipulados na forma de máscaras de bits.

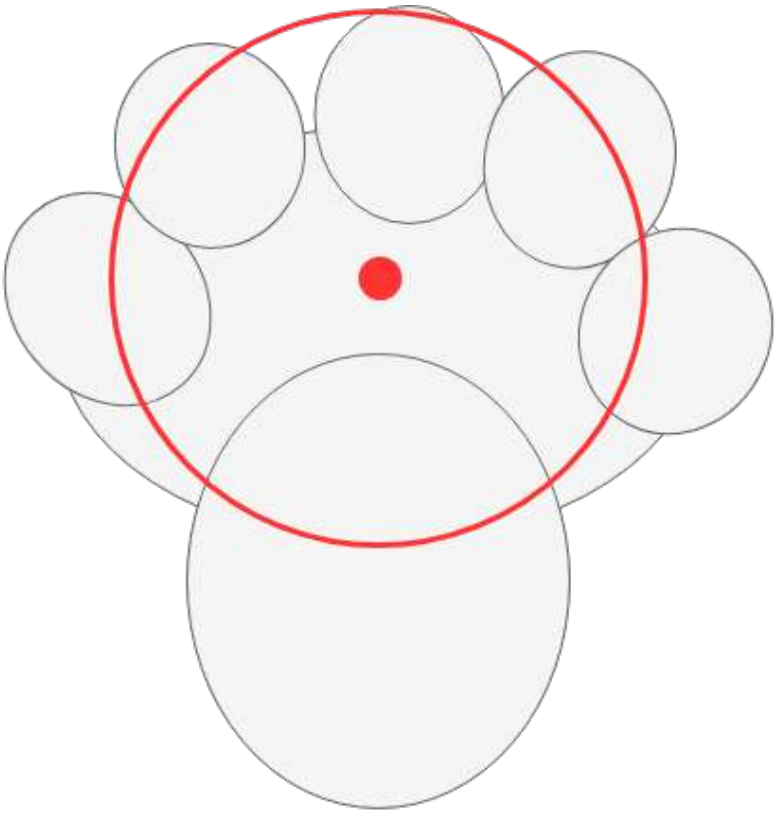
Tabela 1: Valores esperados para as coordenadas das projeções dos dedos.

	Coordenada esperada
mínimo	0,2
anular	0,35
médio	0.5
indicador	0,70
polegar	0,95

Os valores esperados e_i utilizados no algoritmo foram determinados a partir de medidas com as mãos do autor, e podem ser consultados na tabela 1.

Como um caso à parte, considera-se que nenhum dedo está levantado se uma grande proporção (mais de 90%) da área do semicírculo está preenchida por pixels pertencentes à mão, como visto na figura 33.

Figura 33: Determinação da situação sem nenhum dedo levantado.

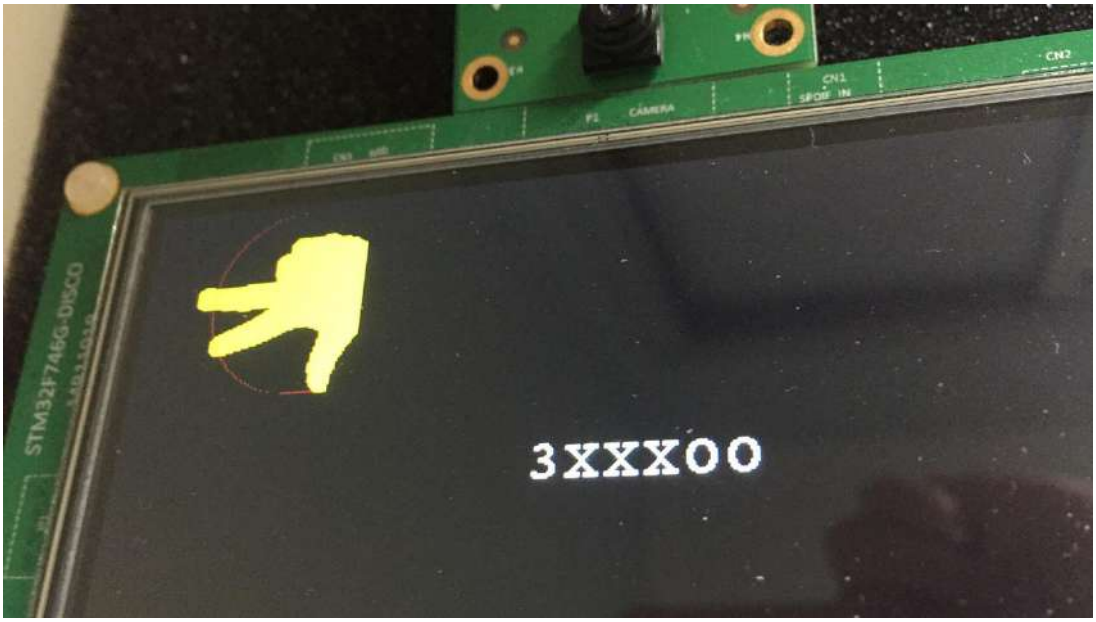


Fonte: Autor

A tela LCD foi utilizada avaliar em tempo real a determinação do conjunto de dedos levantados, através da impressão, no centro da tela, a quantidade de dedos que foi contada, seguida por uma representação da máscara de bits a ser enviado à interface gráfica (5 caracteres, onde 'O' significa dedo abaixado e 'X' dedo levantado). Este modo de avaliação

pode ser vista na figura 39.

Figura 34: Visualização da determinação do conjunto de dedos levantados na tela LCD do sistema embarcado.



Fonte: Autor

6.1.4 Envio dos Resultados da Análise de Imagem

Os passo 7 do diagrama da figura 17 corresponde ao envio da máscara de bits representando o conjunto de dedos levantados e abaixados ao computador principal, através da interface USB Full-Speed.

Para tanto, a máscara de bits deve ser enviada ao endpoint 0x81, para ser lida pelo computador principal através de transferências USB tipo "interrupção", como visto na seção 2.2.2. A função USBD_CUSTOM_HID_SendReport_FS(), da biblioteca para dispositivos USB do pacote STM32CubeF7, é utilizada para realizar esta operação.

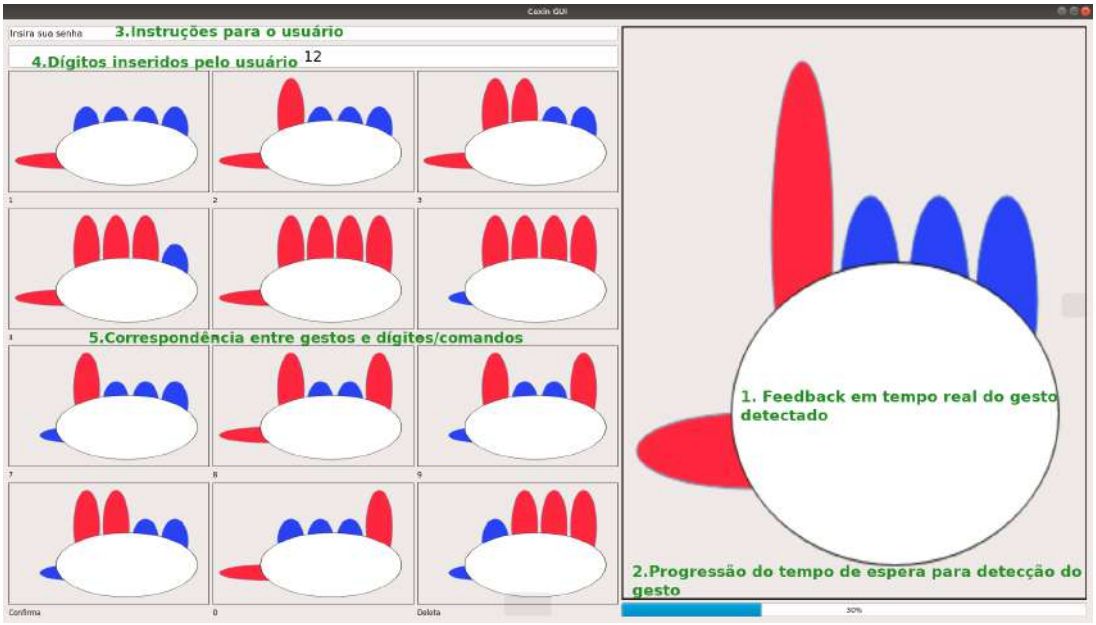
6.2 Projeto e Implementação da Interface Gráfica

A figura 35 mostra a tela que é apresentada ao usuário pela interface gráfica que foi desenvolvida, com seus principais elementos marcados textualmente:

- **1.** À direita, um ícone mostra ao usuário em tempo real qual gesto está sendo reconhecido pelo sistema embarcado.

- **2.** Para evitar entradas espúrias de dados, a interface gráfica espera que o usuário mantenha um gesto por um segundo antes de considerá-lo como uma entrada de dados válida. Esta barra de progresso mostra ao usuário o avanço deste tempo de espera.
- **3.** Uma caixa de texto é reservada para fornecer instruções ao usuário.
- **4.** Uma caixa de texto mostra a sequência de dígitos que foram inseridos pelo usuário através de gestos.
- **5.** Uma matriz de ícones mostra a equivalência de gestos com comandos (entrada de dígitos, deleção e confirmação).

Figura 35: Imagem da interface gráfica que foi desenvolvida.

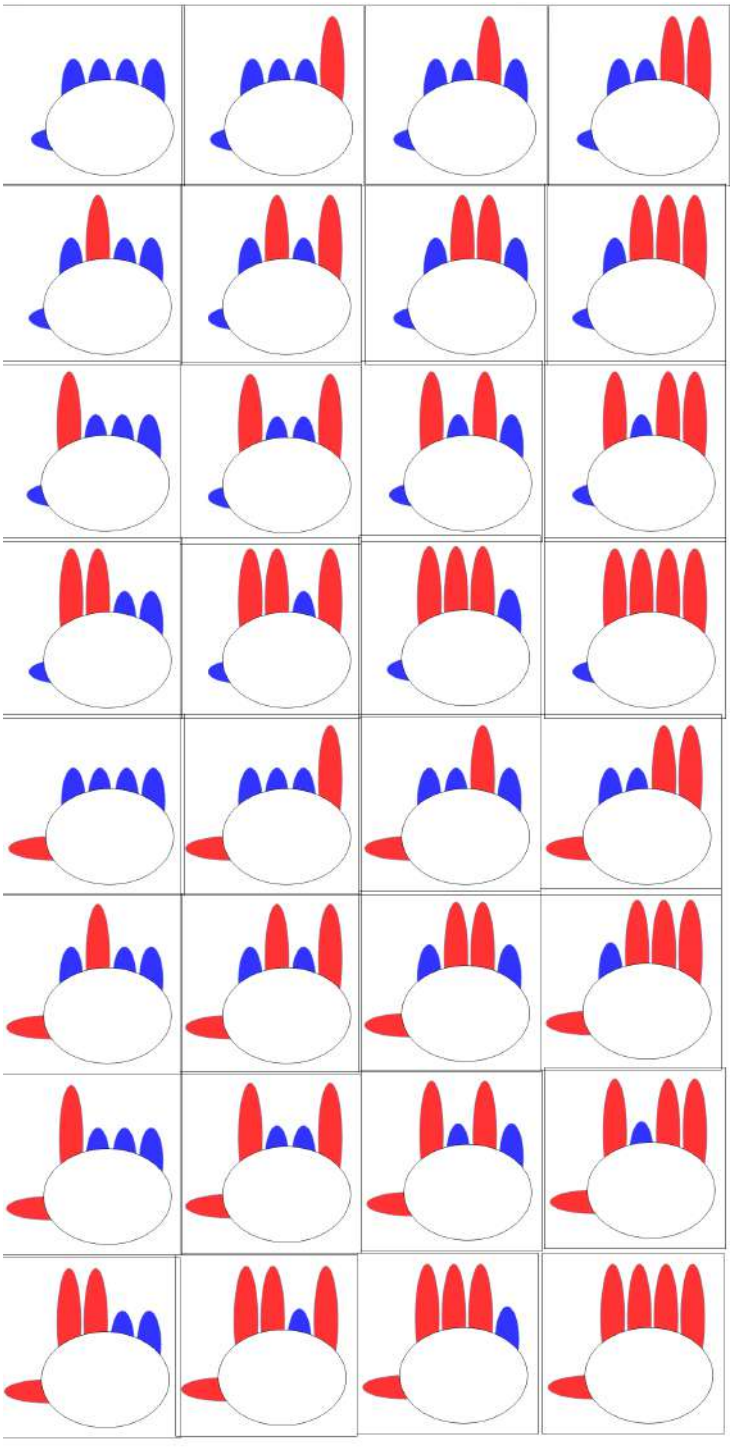


Fonte: Autor

6.2.1 Ícones de Gesto

Para que a interface gráfica possa comunicar ao usuário qual gesto é esperado para cada ação, e qual gesto está sendo interpretado pelo periférico de reconhecimento de gestos, foram desenvolvidos 32 ícones representando os gestos que o periférico pode reconhecer. Os icones foram elaborados com a ferramenta Draw.io (DRAW.IO, 2018) e estão retratados na figura 36. Os ícones representam dedos abaixados em azul e dedos levantandos em vermelho.

Figura 36: Ícones desenvolvidos para a interface gráfica.



Fonte: Autor

6.2.2 Hierarquia de Classes

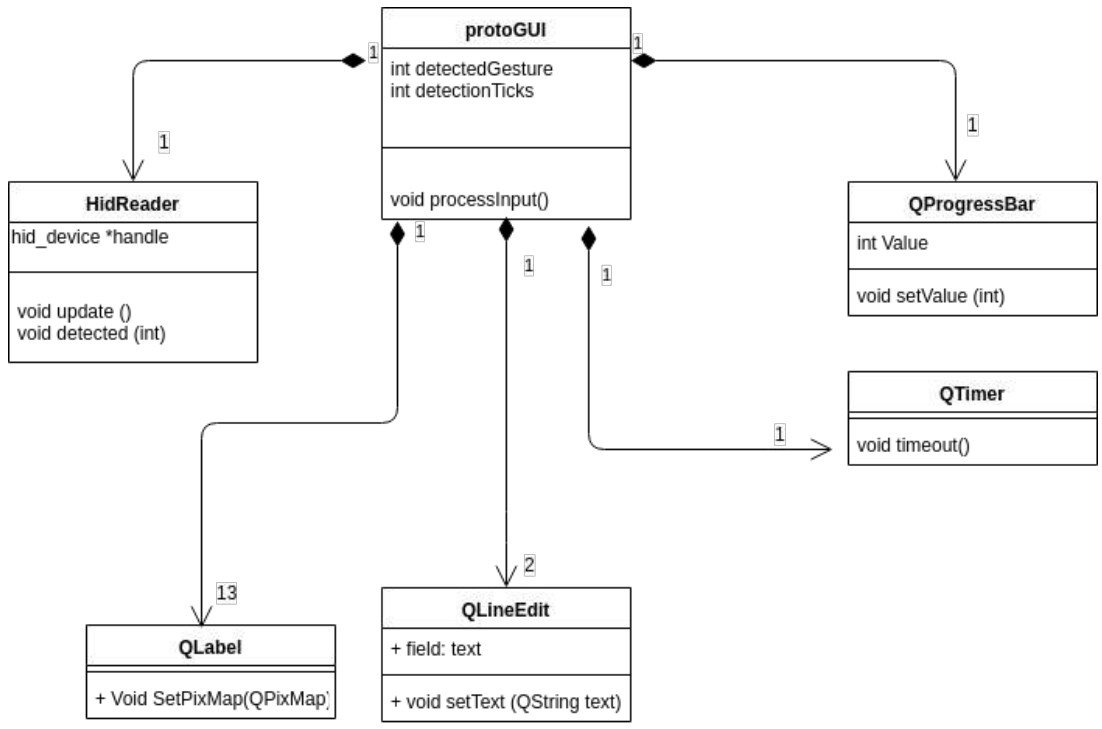
A figura 37 mostra a classe principal que foi desenvolvida para a interface gráfica, e as classes associadas a ela por composição. As classes QLabel, QLineEdit, QProgressBar

e QTimer integram o framework Qt, e são utilizadas para:

- QLabel: Exibir os ícones que representam gestos.
- QProgressBar: Exibir o tempo de espera para que o gesto atualmente reconhecido seja considerado uma entrada válida.
- QLineEdit: Exibir as caixas de texto.
- QTimer: Emitir, com uma frequência de 60 Hz, um "signal" timeout(), que é conectado ao "slot" update() do objeto da classe HidReader.

A classe HidReader, também desenvolvida para o projeto, é discutida na próxima seção. Esta classe lê periodicamente relatórios com o gesto reconhecido pelo sistema embarcado, e utiliza seu "signal" detected() para informar ao resto da interface qual gesto foi recebido.

Figura 37: Diagrama de classes mostrando os componentes de protoGUI, a classe principal da aplicação gráfica.



Fonte: Autor

6.2.3 Comunicação com o Sistema Embarcado

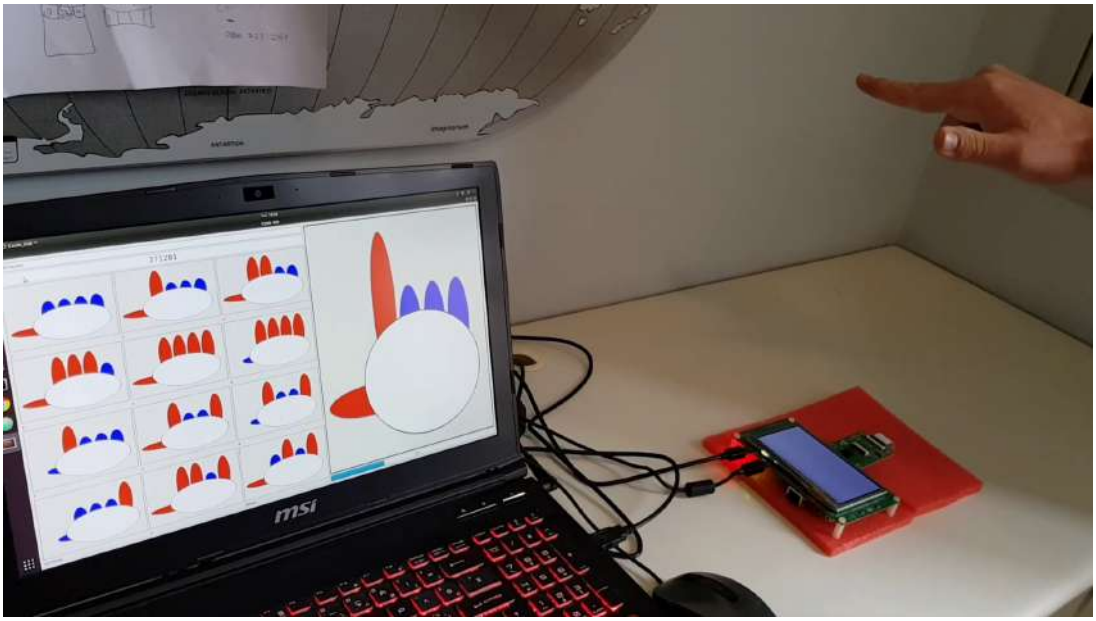
A classe HIDReader, desenvolvida para a interface gráfica, é responsável pela comunicação com o sistema embarcado através do protocolo HID. Em sua função de inicia-

lização, a classe utiliza a função hid_open() da biblioteca HIDapi para obter um “handle” que permite acessar o sistema embarcado. Já em seu método “slot” update(), chamado periodicamente devido à sua conexão com o sinal “timeout” do timer utilizado na interface gráfica, a classe utiliza a funçao hid_read() da biblioteca, utilizando o “handle” como argumento, para ler um relatório HID do sistema embarcado, contendo a mascara de bits representando o conjunto de dedos levantados.

6.3 Funcionamento do Sistema Completo

A figura 38 mostra a Interação de um usuário com o protótipo completo. Mantendo sua mão direita aproximadamente 30 centímetros acima do sensor de imagem, o utilizador controla o teclado da interface gráfica levantando e abaixando dedos da dita mão.

Figura 38: Interação com o sistema completo.



Fonte: Autor

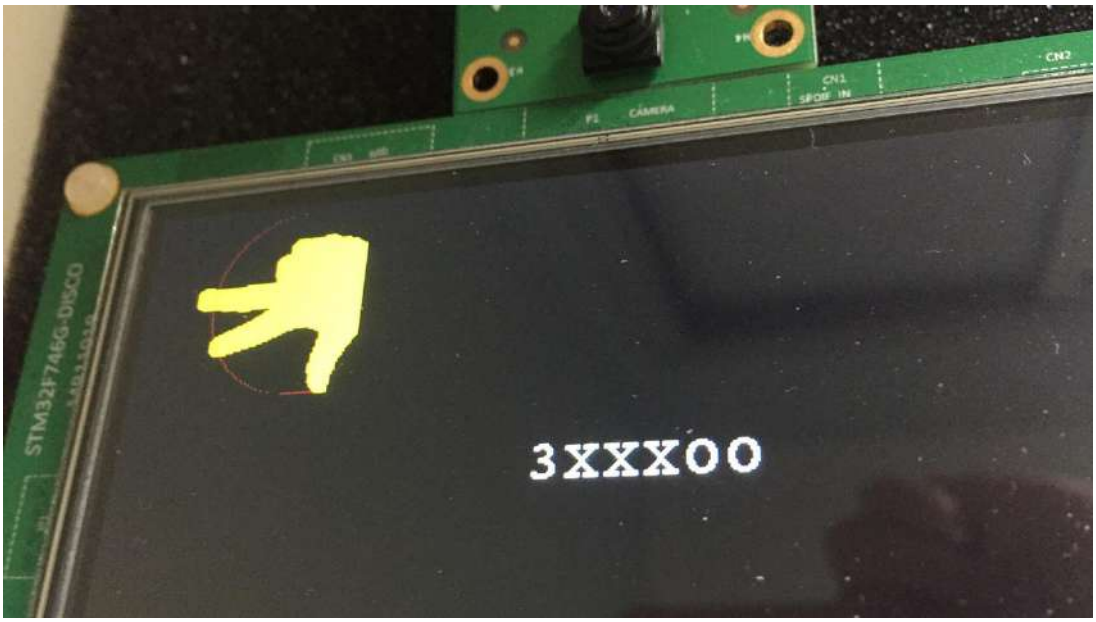
7 Testes e Avaliação

Este capítulo apresenta as estratégias utilizadas para testar diferentes aspectos do projeto.

7.1 Teste com a Tela LCD

Como relatado ao longo do capítulo 6, a tela LCD da placa 32F746G-DISCOVERY foi sistematicamente utilizada para inspecionar o funcionamento das operações de separação da região e de detecção dos dedos levantados. A figura 39 apresentada no capítulo 6 e reproduzida aqui, exibe as diferentes técnicas utilizadas para esta depuração:

Figura 39: Depuração do processamento de imagem pelo software embarcado com a tela LCD.



Fonte: Autor

- Gerar uma imagem com pixels amarelos e pretos, indicando quais pixels estão ou não sendo considerados parte da mão do usuário.

- Inserir sobre esta imagem, em vermelho, o semicírculo utilizado para localizar os dedos.

- Imprimir a quantidade de dedos que foi contada, seguida por uma representação da máscara de bits a ser enviado à interface gráfica (5 caracteres, onde 'O' significa dedo abaixado e 'X' dedo levantado).

7.2 Teste de Algoritmos

Considerou-se que dois algoritmos utilizados no projeto poderiam ser testados individualmente: a implementação do algoritmo de Otsu e a implementação da operação de dilatação.

- Para a implementação do algoritmo de Otsu, verificou-se que esta produz a saída esperada ao receber como entrada a imagem em escala de cinza do exemplo de aplicação do algoritmo que é apresentado em (GREENSTED, 2010).

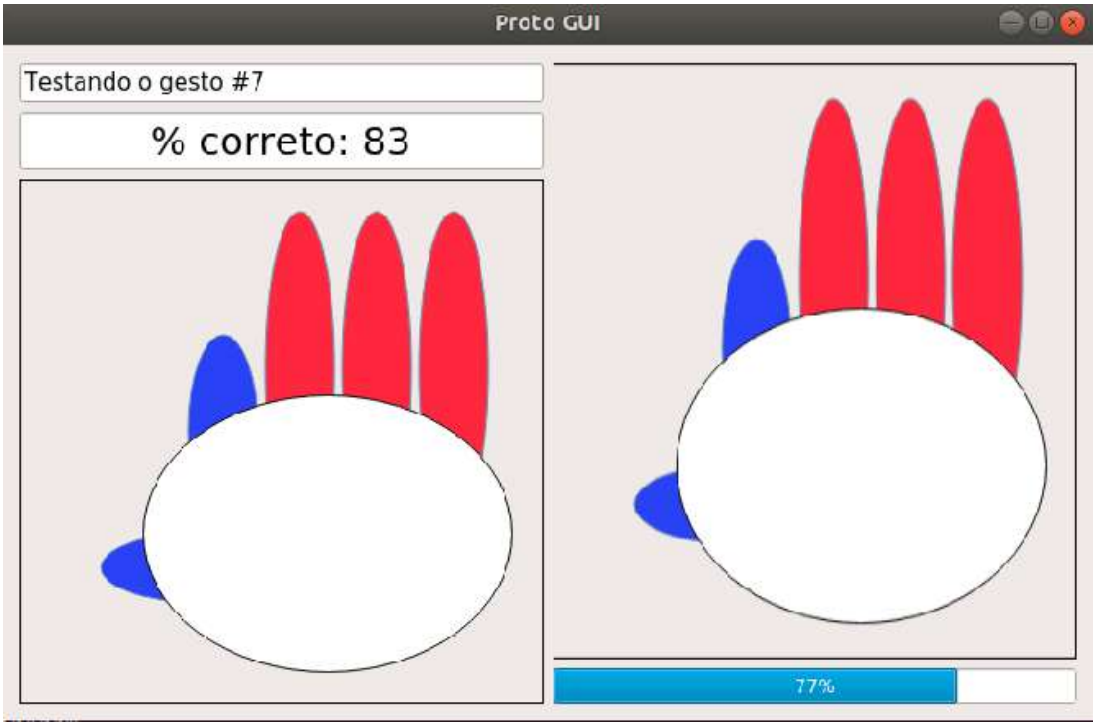
- Para a implementação da operação de Dilatação, verificou-se que, ao receber uma matriz binária com as dimensões da imagem capturada pelo sensor de imagem (160x120) preenchida quase completamente com '1's, com 120 '0's inseridos aleatoriamente, mas garantindo-se que cada um tenha ao menos um '1' como vizinho, a função devolve uma matriz preenchida completamente com '1's.

7.3 Teste com a Interface Gráfica com Teclado

Para verificar a funcionalidade básica tanto do sistema embarcado quanto da interface gráfica com teclado apresentada ao final do capítulo 6, o autor e dois voluntários realizaram o teste de digitar os números de 1 a 9 em sequência realizar o comando confirma. O autor e o voluntário 1 foram capazes de realizar esta tarefa, enquanto que o voluntário 2 não foi devido à altas instabilidades no reconhecimento dos gestos correspondentes aos números 4 e 9.

7.4 Avaliação da taxa de acerto do sistema

Figura 40: Interface utilizada no teste.



Fonte: Autor.

Para avaliar a taxa de acerto da sistema de reconhecimento de gestos, foi desenvolvida a interface gráfica mostrada na figura 40. A interface percorre os 31 gestos possíveis que envolvem ao menos um dedo levantado, mostrando o ícone de cada uma deles na esquerda da interface por cinco segundos, e então aguardando cinco segundos sem mostrar nenhum ícone antes de passar para o próximo. Para cada gesto mostrado na esquerda da interface, o usuário deve reproduzir o dado gesto na direita da tela utilizando o periférico de reconhecimento de gestos. O teste produz como saída, para cada gesto, a proporção dos cinco segundos que o gesto esperado (mostrado na esquerda) foi detectado pelo sistema embarcado (mostrado na direita).

O autor e os dois voluntários mencionados no seção anterior realizaram este teste os resultados obtidos estão presentes nas tabelas 2, 3 e 4, e no gráfico da figura 41. Podemos considerar que o teste foi bem sucedido com o autor e o voluntário 1 (reconhecendo 30 e 16 gestos com taxa de acerto maior ou igual a 80% de precisão, respectivamente), mas não com o voluntário 2.

Tabela 2: Resultado da avaliação da taxa de acerto do sistema com o autor.

ID do gesto	Dedos Levantados	Taxa de Acerto (%)
1	Mínimo	90,6667
2	Anelar	100,0000
3	Mínimo, Anelar	100,0000
4	Médio	100,0000
5	Mínimo, Médio	100,0000
6	Anelar, Médio	100,0000
7	Mínimo, Anelar, Médio	100,0000
8	Indicador	100,0000
9	Mínimo, Indicador	100,0000
10	Anelar, Indicador	100,0000
11	Mínimo, Anelar, Indicador	100,0000
12	Médio, Indicador	100,0000
13	Mínimo, Médio, Indicador	100,0000
14	Anelar, Médio, Indicador	100,0000
15	Mínimo, Anelar, Médio, Indicador	80,0000
16	Polegar	74,0000
17	Mínimo, Polegar	100,0000
18	Anelar, Polegar	100,0000
19	Mínimo, Anelar, Polegar	80,6667
20	Médio, Polegar	100,0000
21	Mínimo, Médio, Polegar	100,0000
22	Anelar, Médio, Polegar	100,0000
23	Mínimo, Anelar, Médio, Polegar	100,0000
24	Indicador, Polegar	100,0000
25	Mínimo, Indicador, Polegar	100,0000
26	Anelar, Indicador, Polegar	92,0000
27	Mínimo, Anelar, Indicador, Polegar	86,6667
28	Mínimo, Anelar, Indicador, Polegar	100,0000
29	Mínimo, Médio, Indicador, Polegar	94,0000
30	Anelar, Médio, Indicador, Polegar	93,3333
31	Mínimo, Anelar, Médio, Indicador, Polegar	86,6667

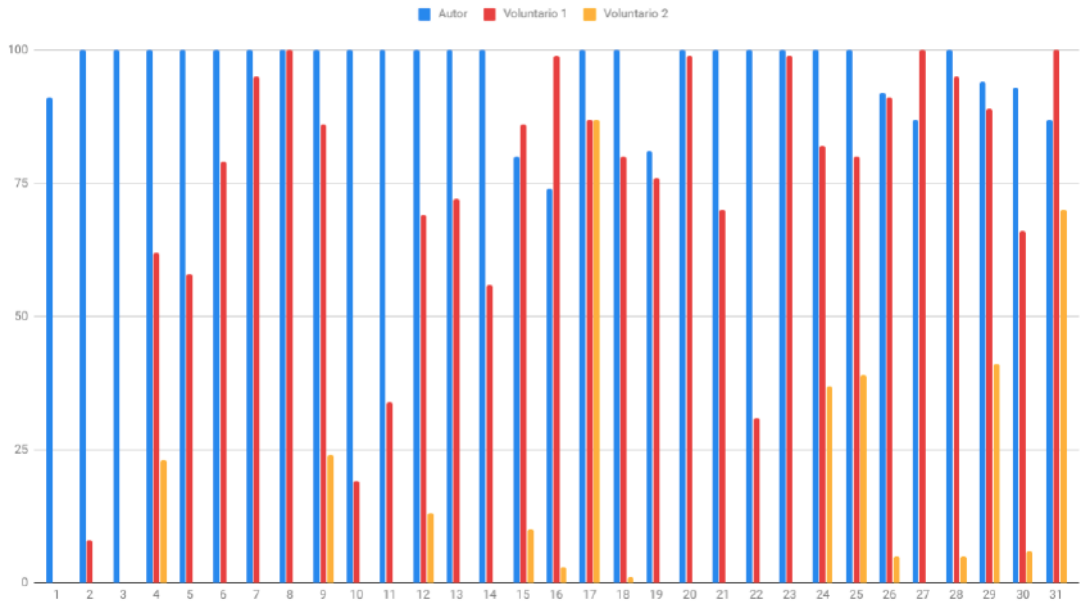
Tabela 3: Resultado da avaliação da taxa de acerto do sistema com o voluntário 1.

ID do gesto	Dedos Levantados	Taxa de Acerto (%)
1	Mínimo	00,0000
2	Anelar	08,0000
3	Mínimo, Anelar	00,0000
4	Médio	62,0000
5	Mínimo, Médio	58,0000
6	Anelar, Médio	79,3333
7	Mínimo, Anelar, Médio	94,6667
8	Indicador	100,0000
9	Mínimo, Indicador	86,0000
10	Anelar, Indicador	19,3333
11	Mínimo, Anelar, Indicador	34,0000
12	Médio, Indicador	69,3333
13	Mínimo, Médio, Indicador	72,6667
14	Anelar, Médio, Indicador	56,6667
15	Mínimo, Anelar, Médio, Indicador	86,0000
16	Polegar	99,3333
17	Mínimo, Polegar	86,6667
18	Anelar, Polegar	80,0000
19	Mínimo, Anelar, Polegar	76,0000
20	Médio, Polegar	99,3333
21	Mínimo, Médio, Polegar	70,0000
22	Anelar, Médio, Polegar	31,3333
23	Mínimo, Anelar, Médio, Polegar	99,3333
24	Indicador, Polegar	82,0000
25	Mínimo, Indicador, Polegar	80,0000
26	Anelar, Indicador, Polegar	91,3333
27	Mínimo, Anelar, Indicador, Polegar	100,0000
28	Mínimo, Anelar, Indicador, Polegar	94,6667
29	Mínimo, Médio, Indicador, Polegar	88,6667
30	Anelar, Médio, Indicador, Polegar	66,0000
31	Mínimo, Anelar, Médio, Indicador, Polegar	100,0000

Tabela 4: Resultado da avaliação da taxa de acerto do sistema com o voluntário 2.

ID do gesto	Dedos Levantados	Taxa de Acerto (%)
1	Mínimo	00,0000
2	Anelar	00,0000
3	Mínimo, Anelar	00,0000
4	Médio	22,6667
5	Mínimo, Médio	00,0000
6	Anelar, Médio	00,0000
7	Mínimo, Anelar, Médio	00,0000
8	Indicador	00,0000
9	Mínimo, Indicador	24,0000
10	Anelar, Indicador	00,0000
11	Mínimo, Anelar, Indicador	00,0000
12	Médio, Indicador	12,6667
13	Mínimo, Médio, Indicador	00,0000
14	Anelar, Médio, Indicador	00,0000
15	Mínimo, Anelar, Médio, Indicador	10,0000
16	Polegar	02,6667
17	Mínimo, Polegar	87,3333
18	Anelar, Polegar	01,3333
19	Mínimo, Anelar, Polegar	00,0000
20	Médio, Polegar	00,0000
21	Mínimo, Médio, Polegar	00,0000
22	Anelar, Médio, Polegar	00,0000
23	Mínimo, Anelar, Médio, Polegar	00,0000
24	Indicador, Polegar	37,3333
25	Mínimo, Indicador, Polegar	39,3333
26	Anelar, Indicador, Polegar	04,6667
27	Mínimo, Anelar, Indicador, Polegar	00,0000
28	Mínimo, Anelar, Indicador, Polegar	08,0000
29	Mínimo, Médio, Indicador, Polegar	41,3333
30	Anelar, Médio, Indicador, Polegar	06,0000
31	Mínimo, Anelar, Médio, Indicador, Polegar	70,0000

Figura 41: Gráfico dos resultados da avaliação da taxa de acerto (taxa de acerto em função do ID do gesto).



Fonte: Autor

7.5 Avaliação da Taxa de Quadros por Segundo

A função HAL_GetTick() da camada de abstração de hardware STM32CubeF7 HAL permite recuperar a quantidade de milissegundos transcorrida desde a chamada da função HAL_Init(), através de um timer configurado por esta segunda função. Chamando-se a HAL_GetTick() ao final de cada ciclo de recepção e processamento de uma imagem, pode-se mensurar a duração destes ao ciclos calculando-se a diferença entre os valores retornados por chamadas consecutivas da função.

Através dessa técnica, a taxa de imagens processada por segundo pelo sistema embarcado foi mensurada. A tabela 5 mostra o resultado desta avaliação, tanto com a configuração com dois buffers de recepção de imagens apresentada na figura 23, quanto com uma configuração com apenas um buffer. A análise da tabela permite concluir que o uso de dois buffers traz ganhos reais de performance ao sistema, permitindo que ele processe todas as imagens produzidas pelo sensor OV9656.

Tabela 5: Frequência de funcionamento de diferentes configurações do software embarcado.

Configuração	**Duração média**	**Frequência**
Processamento com 1 buffer	66ms	15Hz
Processamento com 2 buffers	33ms	30Hz

8 Considerações Finais

8.1 Conclusões do Projeto de Formatura

As seguintes considerações finais podem ser feitas a respeito do protótipo que foi desenvolvido ao longo do projeto:

- O projeto global sistema embarcado foi bem sucedido à medida que o sistema é capaz de inicializar seus periféricos internos e o sensor de imagem, recuperar e processar imagens à taxa de 30 quadros por segundo, e enviar máscaras de bits com o gesto reconhecido para um computador principal.
- O projeto da interface gráfica foi bem-sucedido, à medida que a interface é capaz de fornecer um feedback em tempo real ao usuário de qual gesto está sendo reconhecido pelo sistema embarcado, e aproveitar os gestos reconhecidos para o controle de um teclado numérico.
- O sucesso do processo de tratamento de imagem é incerto: testes com o autor e dois voluntários foram bem sucedidos para dois dos participantes, mas não para um terceiro. Caberia então realizar mais testes no futuro.
- O arquivo binário do software embarcado compilado tem um tamanho de 129 KiB, cabendo confortavelmente nos 340 KiB de memória RAM disponíveis no microcontrolador. Assim, há margem para portar o código para microcontroladores com uma menor quantidade de memória disponível. Por outro lado, uma comparação com o menor tamanho que alcança uma versão compilada da biblioteca de processamento de imagens OpenCV (130 MiB) (OPENCV, 2018a) fornece algum tipo de perspectiva do minimalismo e especialização do software embarcado que foi desenvolvido.

8.2 Contribuições

As contribuições deste projeto são:

- Demonstrar que um microcontrolador de alto desempenho, como o STM32F746G, pode ser utilizado para realizar uma tarefa de reconhecimento de gestos em tempo real, a uma taxa de atualizações relativamente elevada (30/seg).
- Produzir um periférico USB de interação gestual com componentes largamente disponíveis e bem-documentados, e que portanto pode servir de base para explorações futuras deste tipo de arquitetura por equipes interessadas.

8.3 Perspectivas de Continuidade

Algumas perspectivas de continuidade imediatas para o trabalho são realizar mais testes com usuários variados, e adicionar funcionalidades como identificar se o usuário está apresentando sua mão direita ou sua mão esquerda, e adaptar o processamento em função desta variável.

Em um segundo momento, um próximo passo relevante para este projeto seria adaptar o firmware que foi produzido para um modelo de microcontrolador de menor preço e com menor consumo de energia. As linhas STM32F2 e STM32F4 da ST Microelectronics seriam opções naturais para tal, pois apresentam arquiteturas similares à linha STM32F7, da qual faz parte o controlador STM32F746 que foi utilizado.

REFERÊNCIAS

AMAZON INC. *Amazon Locker Delivery*. 2015. Disponível em: ⟨https://www.amazon.com/b?ie=UTF8&node=6442600011⟩. Acesso em: 25 nov. 2018.

ARM HOLDINGS. *GNU Toolchain — GNU-RM Downloads*. 2018. Disponível em: ⟨https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads⟩. Acesso em: 25 nov. 2018.

AWOPETU, B. et al. Microorganisms on humancomputer interfaces: An examination of automated teller machines and cybercafes in ilesa and ijebujesa, nigeria. v. 5, Janeiro 2017.

BEGALINOVA, A.; SHINTEMIROV, A. Design of embedded gesture recognition system for robotic applications. In: *2014 IEEE 8th International Conference on Application of Information and Communication Technologies (AICT)*. [S.l.: s.n.], 2014. p. 1–4.

BUSINESS INSIDER. *Here's A Picture Of Amazon Locker, The New Delivery Box Amazon Is Using To Take Over The World*. 2012. Disponível em: ⟨https://www.businessinsider.com/amazon-locker-2012-8⟩. Acesso em: 25 nov. 2018.

DEKATE, A.; KAMAL, A.; SUREKHA, K. S. Magic glove - wireless hand gesture hardware controller. In: *2014 International Conference on Electronics and Communication Systems (ICECS)*. [S.l.: s.n.], 2014. p. 1–4.

DRAW.IO. *Draw.io*. 2018. Disponível em: ⟨https://www.draw.io/⟩. Acesso em: 25 nov. 2018.

EMBEDDED MICROPROCESSOR BENCHMARK CONSORTIUM. *CoreMark — an EEMBC Benchmark*. 2018. Disponível em: ⟨https://www.eembc.org/coremark/scores.php⟩. Acesso em: 25 nov. 2018.

FEOFILOFF, P. *Busca em largura (BFS)*. 2010. Disponível em: ⟨https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/bfs.html⟩. Acesso em: 25 nov. 2018.

FINGERIO. *FingerIO - Using Active Sonar for Fine-Grained Finger Tracking*. 2017. Disponível em: ⟨http://fingerio.cs.washington.edu/⟩. Acesso em: 25 nov. 2018.

FISHER, R. et al. *Image Processing Learning Resources - Dilation*. 2004. Disponível em: ⟨https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm⟩. Acesso em: 25 nov. 2018.

FLORES, M. B. H. et al. User-oriented finger-gesture glove controller with hand movement virtualization using flex sensors and a digital accelerometer. In: *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. [S.l.: s.n.], 2014. p. 1–4.

GREENSTED, A. *Otsu Thresholding*. 2010. Disponível em: ⟨http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html⟩. Acesso em: 25 nov. 2018.

KALGAONKAR, K.; RAJ, B. One-handed gesture recognition using ultrasonic doppler sonar. In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. [S.l.: s.n.], 2009. p. 1889–1892. ISSN 1520-6149.

LANGOLF, G. D.; CHAFFIN, D.; FOULKE, J. A. An investigation of fitts' law using a wide range of movement amplitudes. *Journal of motor behavior*, v. 8, p. 113–28, Junho 1976.

LEAPMOTION, INC. *LeapMotion*. 2018. Disponível em: ⟨https://www.leapmotion.com/⟩. Acesso em: 25 nov. 2018.

LEI, Y. et al. A real-time hand gesture recognition algorithm for an embedded system. In: *2014 IEEE International Conference on Mechatronics and Automation*. [S.l.: s.n.], 2014. p. 901–905. ISSN 2152-7431.

LIBUSB. *A cross-platform library to access USB devices*. 2018. Disponível em: ⟨https://github.com/libusb/libusb⟩. Acesso em: 25 nov. 2018.

LIN, M.; VILLALBA, R. *Sign Language Glove*. 2014. Disponível em: ⟨http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2014/rdv28_mjl256/webpage/⟩. Acesso em: 25 nov. 2018.

MALIMA, A.; OZGUR, E.; CETIN, M. A fast algorithm for vision-based hand gesture recognition for robot control. In: *2006 IEEE 14th Signal Processing and Communications Applications*. [S.l.: s.n.], 2006. p. 1–4. ISSN 2165-0608.

MICROSOFT, INC. *Kinect para Windows*. 2018. Disponível em: ⟨https://developer.microsoft.com/pt-br/windows/kinect/⟩. Acesso em: 25 nov. 2018.

NACHIMUTHU, R. et al. Prevalence of multi drug resistant strains on touch screen of automated teller machine. *Asian Journal of Pharmaceutical and Clinical Research*, v. 8, Março 2015.

NANDAKUMAR, R. et al. Fingerio : Using sonar for fine-grained finger tracking. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. [S.l.: s.n.], 2016. p. 1515–1525.

OMNIVISION TECHNOLOGIES. *OV9656 Color CMOS SXGA (1.3 MegaPixel) CAMERACHIP™ Sensor with OmniPixel® Technology with OmniPixel® Technology*. 2006.

OPENCV. *How to build more compact OpenCV applications on Linux*. 2018. Disponível em: ⟨https://github.com/opencv/opencv/wiki/Compact-build-advice⟩. Acesso em: 25 nov. 2018.

OPENCV. *OpenCV Library*. 2018. Disponível em: ⟨https://opencv.org/⟩. Acesso em: 25 nov. 2018.

OPENMV, LLC. *OpenMV — Small-Affordable-Expandable*. 2018. Disponível em: ⟨https://openmv.io/⟩. Acesso em: 25 nov. 2018.

OTSU, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 9, n. 1, p. 62–66, Janeiro 1979. ISSN 0018-9472.

OTT, A. *HIDAPI library for Windows, Linux, FreeBSD and Mac OS X*. 2018. Disponível em: ⟨https://github.com/signal11/hidapi⟩. Acesso em: 25 nov. 2018.

QT COMPANY. *Qt — Cross-platform Software Development for Embedded and Desktop*. 2018. Disponível em: ⟨https://www.qt.io/⟩. Acesso em: 25 nov. 2018.

QT COMPANY. *Signals and Slots — Qt 4.8*. 2018. Disponível em: ⟨http://doc.qt.io/archives/qt-4.8/signalsandslots.html⟩. Acesso em: 25 nov. 2018.

RASPBERRY pi - Teach, Learn and Make with Raspberry Pi. 2015. Disponível em: ⟨https://www.raspberrypi.org/⟩. Acesso em: 25 nov. 2018.

SANTOS, L. et al. Dynamic gesture recognition using a smart glove in hand-assisted laparoscopic surgery. *Technologies*, Janeiro 2018. ISSN 2227-7080.

STMICROELECTRONICS N.V. *Discovery kit with STM32F746NG MCU*. 2015. Disponível em: ⟨https://www.st.com/resource/en/data_brief/32f746gdiscovery.pdf⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *Digital camera interface (DCMI) for STM32 MCUs*. 2017. Disponível em: ⟨https://www.st.com/content/ccc/resource/technical/document/application_note/group0/c0/ef/15/38/d1/d6/49/88/DM00373474/files/DM00373474.pdf/jcr:content/translations/en.DM00373474.pdf⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *Expansion boards for the STM32F4 Discovery kit*. 2017. Disponível em: ⟨https://www.st.com/resource/en/data_brief/stm32f4dis-ext.pdf⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *STM32 32-bit Arm Cortex MCUs*. 2018. Disponível em: ⟨https://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *STM32CubeF7*. 2018. Disponível em: ⟨https://www.st.com/en/embedded-software/stm32cubef7.html⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *STM32CubeMX*. 2018. Disponível em: ⟨https://www.st.com/en/development-tools/stm32cubemx.html⟩. Acesso em: 25 nov. 2018.

STMICROELECTRONICS N.V. *STM32F75xxx and STM32F74xxx advanced Arm®-based 32-bit MCUs*. [S.l.], 2018. Acesso em: 25 nov. 2018.

TEKEREKOğLU, M. S. et al. Bacteria found on banks' automated teller machines (atms). *African journal of microbiology research*, Abril 2013. ISSN 1619-1621.

THE STLINK PROJECT. *Open source version of the STMicroelectronics Stlink Tools*. 2018. Disponível em: ⟨https://github.com/texane/stlink⟩. Acesso em: 25 nov. 2018.

THE WORLD BANK. *Automated teller machines (ATMs) (per 100,000 adults)*. 2018. Disponível em: ⟨https://data.worldbank.org/indicator/FB.ATM.TOTL.P5?end=2017&start=2004⟩. Acesso em: 25 nov. 2018.

TURKEL, E. *Automatic Thresholding*. 2012. Disponível em: ⟨http://www.math.tau.ac.il/~turkel/notes/otsu.pdf⟩. Acesso em: 25 nov. 2018.

USB IMPLEMENTERS' FORUM. *Device Class Definition for Human Interface Devices (HID)*. [S.l.], 2001. Acesso em: 25 nov. 2018.

USB IMPLEMENTERS' FORUM. *Universal Serial Bus Specification*. [S.l.], 2001. Acesso em: 25 nov. 2018.

USB IMPLEMENTERS' FORUM. *Universal Serial Bus Device Class Definition for Video Devices*. [S.l.], 2012.

WESTOVER, B. *Leap Motion Controller*. 2013. Disponível em: ⟨https://www.pcmag.com/article2/0,2817,2422047,00.asp⟩. Acesso em: 25 nov. 2018.

WIKIPEDIA. *YUV*. 2015. Disponível em: ⟨https://en.wikipedia.org/wiki/YUV⟩. Acesso em: 25 nov. 2018.

WILLIAMS, C. *ARM gives Internet of Things a piece of its mind – the Cortex-M7*. 2014. Disponível em: ⟨https://www.theregister.co.uk/2014/09/24/arm_cortex_m7/⟩. Acesso em: 25 nov. 2018.

APÊNDICE A – BIBLIOTECAS UTILIZADAS

Neste apêndice, são discutidas com mais detalhes funções do pacote de software STM32CubeF7 que foram empregadas no desenvolvimento do projeto.

A.1 Funções do Pacote BSP Utilizadas

Os drivers stm32746g_discovery_camera.c/.h e ov9656.c/.h da biblioteca BSP foram utilizado para configurar a interface de câmera DCMI e o sensor de imagem OV9656. Suas funções mais importantes são:

- BSP_CAMERA_Init(): para inicializar as interfaces I2C e DCMI, e utilizar a interface I2C para configurar o sensor de imagem com uma determinada resoluçao.
- BSP_CAMERA_ContinuousStart(): Para fazer com que a interface DCMI passe a receber dados de video continuamente.

Os drivers stm32746g_discovery_lcd.c/.h e stm32746g_discovery_sdram.c/.h da biblioteca BSP foram utilizado para configurar a interface de câmera DCMI e o sensor de imagem OV9656. Suas funções mais importantes são:

- BSP_LCD_Init() e BSP_LCD_DisplayOn() para configurar e ligar a tela.
- BSP_LCD_DrawPixel(): Para desenhar um pixel de determinada cor, em um determinada posição da tela.
- BSP_LCD_DisplayStringAt(): Para escrever uma string na tela.

A.2 Funções do Pacote HAL Utilizadas

A funções mais importantes da biblioteca HAL utilizadas diretamente pelo código de aplicação são:

- HAL_Init(): Para inicializar a camada de abstração de hardware.
- HAL_Delay(): Para aguardar um dada quantidade de milisegundos.
- HAL_NVIC_SetPriority(): Para configurar uma interrupção.

A.3 Funções do Pacote Middlewares Utilizadas

A única biblioteca do pacote de middlewares utilizada no projeto foi a biblioteca de dispositivos USB (STM32_USB_Device_Library). Os arquivos da biblioteca são divididos em dois grupos: Core (os arquivos utilizados para implementar as funcionalidades básicas da comunicação sobre USB) e Class (arquivos contendo descritores e código de aplicação, que permitem implementar as funcionalidades de diferentes classes de comunicação USB, como CDC, HID ou Áudio).

No caso do projeto, foram utilizados os arquivos "Core"e os arquivos "usbd_customhid.ch", que podem ser empregados em conjunto para implementar um dispositivo HID diferente de um teclado ou mouse. Suas funções mais importantes são:

- USBD_Init(): Para para inicializar a interface USB.
- USBD_RegisterClass() e USBD_CUSTOM_HID_RegisterInterface(): Para configurar a classe do dispositivo (atrvés do registro de funções callback para diversos eventos).
- USBD_Start(): Para fazer com que a interface USB comece a comunicar-se com o computador principal.
- USBD_CUSTOM_HID_SendReport_FS(): Para enviar um relatório da classe HID através da interface Full-speed.