FRANCIS BENJAMIN ZAVALETA CASTRO

Projeto de Sistemas Computacionais: Análise e Controle do Fluxo de Pessoas em Ambientes Fechados

São Paulo

2021

FRANCIS BENJAMIN ZAVALETA CASTRO

Projeto de Sistemas Computacionais: Análise e Controle do Fluxo de Pessoas em Ambientes Fechados

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em *Internet of Things*

Orientador: Prof. Livre-Docente Carlos Eduardo Cugnasca

São Paulo
2021

Catalogação-na-publicação

Zavaleta Castro, Francis Benjamin
Projeto de Sistemas Computacionais: Análise e Controle do Fluxo de Pessoas em Ambientes Fechados / F. B. Zavaleta Castro -- São Paulo, 2020.
50 p.

Monografia (MBA em MBA em Internet of Things) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.internet of things 2.sistemas embarcados 3.projeto de sistemas digitais I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Nome: CASTRO, Francis Benjamin Zavaleta

Título: Projeto de Sistemas Computacionais: Análise e Controle do Fluxo de Pessoas em Ambientes Fechados

Monografia apresentada ao PECE – Programa de Educação Continuada em Engenharia da Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do curso de MBA em *Internet of Things*.

Aprovado em: 09/02/2021

Banca Examinadora

Prof(a). Dr(a). Carlos Eduardo Cugnasca

Instituição: Departamento de Enga. de Computação e Sistemas Digitais - EPUSP

Julgamento: Aprovado

Prof(a). Dr(a). Eduardo Ferreira Franco

Instituição: PCS/POLI - USP

Julgamento: Aprovado

Prof(a). Dr(a). VIDAL AUGUSTO ZAPPAROLI CASTRO MELO

Instituição: PECE / POLI / USP

Julgamento: APROVADO

DEDICATÓRIA

Dedico este trabalho aos meus avôs,
pelo seu constante apoio e dedicação
em cada etapa da minha vida.

AGRADECIMENTOS

À escola politécnica da Universidade de São Paulo - EPUSP e, principalmente, ao Prof. Carlos Eduardo Cugnasca cuja paciência e suporte na orientação foi fundamental para o melhor desenvolvimento deste trabalho.

A todos meus colegas do MBA em Internet of things, a nossa convivência e troca de experiência na sala de aula, tornaram este curso uma experiência valiosa e gratificante.

À minha família, que são o motor da minha vida, pelo constante apoio em todos meus projetos empreendidos.

RESUMO

CASTRO, F.B.Z. **Projeto de Sistemas Computacionais: Análise e Controle do Fluxo de Pessoas em Ambientes Fechados**. 2021. 53 p. Monografia (MBA em *Internet of Things*) - Programa de Educação Continuada em Engenharia, Escola Politécnica, Universidade de São Paulo. São Paulo, 2021.

Este trabalho tem como objetivo o desenvolvimento de um sistema embarcado para a análise e o controle do fluxo de pessoas em ambientes comerciais fechados. Devido aos acontecimentos provenientes da última pandemia que enfrenta a humanidade (Covid-19), os protocolos de segurança em relação à quantidade de pessoas em ambientes fechados têm sido mudados e a tendência é a continuidade dos mesmos. O conhecimento do volume de pessoas em tempo real auxilia na definição das estratégias de negócio para a tomada de decisões baseadas em dados. As soluções atuais existentes no mercado utilizam visão computacional para determinar o fluxo de pessoas mas exigem que o usuário adquira sistemas com arquitetura fechada, definidas por seus fabricantes, não favorecendo a interoperabilidade, além de apresentarem interfaces primitivas em relação à experiência do usuário. Neste trabalho pretende-se integrar a visão computacional com tecnologias de comunicação sem fio e serviços de análise de dados, com a finalidade de gerar uma inteligência embutida num dispositivo que seja capaz de interagir com o usuário final gerando alertas parametrizáveis, Foi realizada uma prova de conceito utilizando um módulo com microcontrolador e módulos sem fio GSM. Pretendeu-se demonstrar a capacidade do dispositivo de realizar a contagem das saídas e entradas de pessoas num ponto específico.

Palavras-Chave: sistemas embarcados, inteligência de máquina, visão robótica, controle de pessoas em ambiente fechado.

ABSTRACT

This investigation has as objective the development of an embedded system that analyzes and controls the movement of people in commercial environments. Prior to the events derived from the pandemic that humanity is currently facing (Covid-19), the security protocols for the movement of people in closed environments have already been changing and there is a tendency for them to continue doing so. Knowing the amount of people in real time helps in defining business strategies for data decision making. The current solutions on the market that the user acquires are systems with closed architecture, defined by their manufacturer and they do not favor interoperability. They also present primitive interfaces in relation to user experience. Designing a device that meets these new market demands involves customizing the operating system kernel, integrating peripherals, developing computer vision algorithms, and incorporating embedded intelligence. In this reaserch it is expected to carry out a proof of concept, that uses a module based on a microprocessor as a peripheral for the constitution of a user interface and wireless communication modules. It is intended to demonstrate that using software engineering concepts oriented to the development of embedded systems it is possible to generate a device with high performance and low cost.

Keywords: embedded systems, machine intelligence, computer vision, people control in a closed environment.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE DIAGRAMAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ARM	Advanced RISC Machine
CISC	*Complex Instruction Set Computer*
CPU	*Central Processing Unit*
EC2	*Elastic Compute Cloud*
GPIO	*General Purpose Input/Output*
GPRS	General Packet Radio Services
GSM	*Global System for Mobile Communications*
GUI	Interface Gráfica de Usuário
I^2C	*Inter - Integrated Circuit*
IoT	*Internet of Things* (Internet das Coisas)
OT	*Object Tracking*
RDS	*Relational Database Service*
RISC	*Reduced Instruction Set Computer*
RTC	*Real Time Clock*
SDK	*Software Development Kit*
SIM	*Subscriber Identity Module*
SoC	*System on Chip*
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchronous Receiver/Transmiter*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS E MOTIVAÇÃO

No final do ano de 2019, na cidade de Wuhan na China, foi reportada a aparição de um novo vírus de tipo coronário, causador de infecções respiratórias agudas. “No Abril de 2020, foram reportados mais de 1,5 milhão de casos e 85 mil mortes no mundo, e espera-se que um número ainda maior de casos e óbitos venha a ocorrer nos próximos meses” (BARRETO et al., 2020).

De acordo com Zucco et al. (2020), “A transmissão do patógeno pode ocorrer por inalação de gotículas respiratórias infectadas, em particular se a exposição de gotículas for muito próxima (cerca de 2 metros) e também inclui contato com membranas mucosas”. A evidência científica aponta que o principal meio de prevenção é evitar contato próximo, mas isso é um grande desafio considerando os fatores culturais e infraestruturais das grandes cidades latino-americanas.

Entre as medidas de contenção dos governos, o isolamento social foi um denominador comum. Escolas e centros comerciais foram fechados para reduzir os contágios e evitar o colapso do sistema de saúde. Em países menos desenvolvidos as medidas de isolamento tiveram impacto socioeconômico devido à fragilidade dos serviços públicos e renda familiar (LEMOS; ALMEIDA-FILHO, 2020).

Segundo a ABRASCE (2019), existem 577 *shoppings* no Brasil, tendo uma média de 502 milhões de visitantes por mês. Os dados do fluxo de pessoas aponta uma movimentação muito dinâmica em ambientes fechados ,como lojas, cinema e espaços de refeição. Considerando a significância do setor em termos de faturamento e geração de empregos, é importante estabelecer protocolos que permitam a operabilidade segura.

Entre as soluções utilizadas pelo mercado estão as câmeras IP inteligentes, que realizam a contagem de pessoas dentro do ambiente monitorado, fornecendo dados críticos por meio de gráficos acessados via *login*. A principal desvantagem dessa solução é que a plataforma depende do fornecedor, além de não ter a capacidade de interagir com seu entorno.

Nesse contexto de restrições e medidas de distanciamento social é necessário ter um controle sobre a quantidade de pessoas que ingressam nos estabelecimentos

em tempo real, com uma arquitetura de dados que permita emitir alertas quando os parâmetros (definidos previamente pelo usuário) do fluxo são atingidos.

1.2 OBJETIVO

Este trabalho teve como objetivo a aplicação de conceitos de engenharia de software, *Data Science* e sistemas embarcados, para o projeto de um sistema computacional para a análise e controle da movimentação de pessoas em ambientes fechados, segundo a visão de Internet das Coisas (*Internet of Things* – IoT). Considerou-se integração com módulos de comunicação sem fios e a experiência do usuário.

Além disso, buscaram-se outros objetivos específicos, como:

1. Implementar algoritmos de classificação para controle de fluxo de pessoas por imagens em tempo real.
2. Desenvolver métodos de alertas para a interação do dispositivo com o usuário baseados em dados e operados por módulos de comunicação sem fio.
3. Criar um protótipo funcional que permita validar a modelagem do sistema proposto, aplicando conceitos de IoT.

1.3 JUSTIFICATIVA

No contexto de restrições e medidas de distanciamento social, o presente trabalho surge como uma contribuição para a mitigação desse problema, propondo o desenvolvimento de um dispositivo desenvolvido no conceito de IoT para a aquisição de dados e gestão do fluxo de pessoas em ambientes fechados. O dispositivo proposto atua como um sensor sem fio, um *gateway* de pré-processamento para a nuvem e um atuador, interagindo dinamicamente com os usuários finais.

A principal diferença com as soluções existentes de câmeras inteligentes, é a integração com módulo de comunicação sem fio, o que dota o dispositivo de atuação com os usuários finais por meio de alertas em tempo real, parametrizáveis pelo usuário por meio de interfaces humano-computador como *Dashboard* e *Chatbots*.

Entre as funcionalidades do dispositivo, o controle de pessoas permite informar ao usuário quando a capacidade do ambiente fechado está sendo atingido. Isso atinge os requisitos dos protocolos sanitários de limite máximo de pessoas dentro do

ambiente segundo o sector empresarial sem a necessidade de ter uma pessoa designada. Dependendo da fase da cidade em questão, este parâmetro de limite máximo poderá ser variado pelo usuário diretamente no dispositivo.

1.4 MÉTODO DE PESQUISA

O método de pesquisa utilizado no presente trabalho foi a solução de problema de engenharia. Foi identificado um problema no contexto atual de pandemia por Covid-19, analisadas as suas características do problema, e levantamento dos requisitos funcionais e não funcionais, que serviram como base para a elaboração da especificação do produto proposto.

A implementação do protótipo do produto considerou os conceitos de sistemas embarcados, inteligência artificial e computação na nuvem, com o objetivo de desenvolver um protótipo de baixo custo para o controle automático do fluxo de pessoas em ambientes fechados, fornecendo informações que permitam a tomada de decisões baseadas em dados.

A principal técnica do método de pesquisa utilizado neste trabalho foi de prototipagem, que possibilitou a validação da solução pela realização de uma prova de conceito.

Após a definição dos requisitos funcionais e não funcionais do projeto, foi proposta uma arquitetura para o sistema embarcado, selecionando-se os componentes de hardware e software, dentre as opções do mercado. A integração dos componentes correspondeu à etapa a de maior complexidade, devido às múltiplas abordagens tecnológicas utilizadas neste trabalho, como visão computacional, visualização de dados, desenvolvimento de *firmware* e utilização de diversos periféricos.

A abordagem do método de pesquisa de prototipagem de sistemas embarcados considerado neste trabalho foi a *top-down*, utilizando-se um processo de decomposição em funções menos complexas, com a finalidade de refinar as funcionalidades dos componentes e a interoperabilidade entre as suas partes.

1.4 ORGANIZAÇÃO

O Capítulo 1 apresentou as motivações, o objetivo, as justificativas, método de pesquisa e a estrutura do trabalho.

O Capítulo 2 apresenta uma revisão sobre os fundamentos da arquitetura dos processadores, desde as arquiteturas clássicas até o hardware baseado em *System on Chip* (SoC). Por fim, apresenta-se as características do módulo utilizado na construção do protótipo funcional.

O Capítulo 3 apresenta as características do *kernel* do Linux utilizado no sistema embarcado projetado, e discute metodologias e ferramentas para o desenvolvimento de uma distribuição Linux customizada.

O Capítulo 4 destaca tecnologias de comunicação sem fio, aprofundando-se a tecnologia *Global System for Mobile Communications* (GSM). São apresentadas as características do módulo, utilizado na construção do protótipo funcional e suas aplicações.

O Capítulo 5 apresenta os elementos da interface com o usuário, sob as visões de hardware e software.

O Capítulo 6 destaca os fundamentos teóricos dos algoritmos de visão computacional aplicados nesta pesquisa. São apresentados os algoritmos de detecção de imagens e *object tracking*.

O Capítulo 7 corresponde à parte principal do trabalho. São apresentadas as arquiteturas e metodologias para o desenvolvimento de um dispositivo para o controle de pessoas, segundo a visão de integração de hardware e software.

O Capítulo 8 apresenta as conclusões da pesquisa, assim como as potenciais melhorias a serem tratadas em trabalhos de pesquisa posteriores.

Finalizando o trabalho encontram-se relacionadas as referências utilizadas no trabalho.

2 FUNDAMENTAÇÃO TÉCNICA

Para projetar um sistema computacional de IoT é preciso analisar as diversas arquiteturas de processadores disponíveis . A infraestrutura de comunicação para o envio de dados e os serviços de software que os processam e geram valor.

No presente capítulo são abordados conceitos teóricos sobre organização e tipos de processadores utilizados em sistemas embarcados, módulos de comunicação sem fio GSM, organização de interfaces Humano-Computador e algoritmos de visão computacional utilizados neste trabalho .

2.1 Fundamentos de Processadores Embarcados

Na era da tecnologia da informação diversos dispositivos são responsáveis pela coleta, processamento de dados e tomada de decisões. Esses dispositivos têm como elemento principal uma unidade de processamento do inglês *Central Processing Unit* (CPU), ou simplesmente processador. Segundo Heath (2003) muitos dispositivos atuais de uso cotidiano não seriam eficientes nem econômicos sem a integração do processador. Para Heath (2003) um sistema embarcado é um sistema que tem como elemento principal o microprocessador, programado para cumprir um determinado objetivo, sem a possibilidade de ser reprogramado pelo usuário.

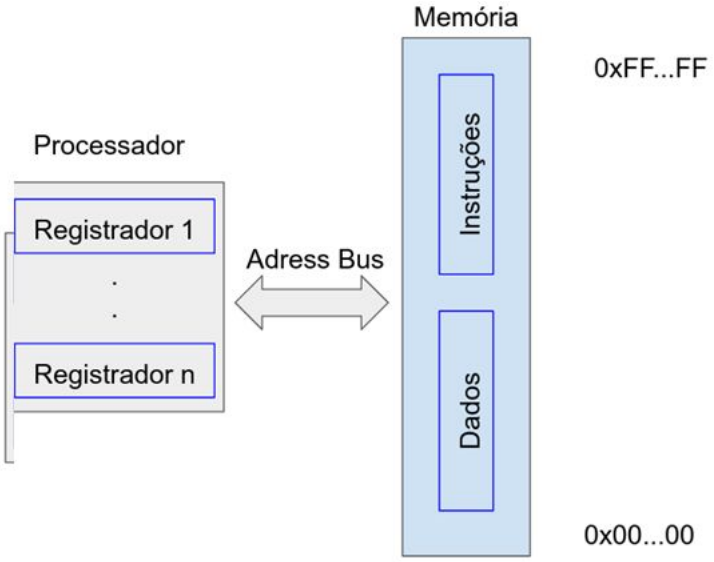
Segundo Furber (2000) um processador de uso geral é um autômato de estado finito que executa instruções mantidas em memória.

Os processadores seguem um modelo de armazenamento de programa, sendo que os seus registradores interagem diretamente com as unidades da memória por meio das instruções (região de instruções e dados), como mostra a Figura 1, na qual as instruções e dados se mantêm na mesma unidade de memória, permitindo que as instruções sejam tratadas como dados quando for preciso, possibilitando, assim, a autocriação de instruções (FURBER, 2020).

Historicamente dois tipos de arquitetura foram predominantes: a *Reduced Instruction Set Computer* (RISC) e a *Complex Instruction Set Computer* (CISC). A arquitetura RISC é posterior à CISC, produto de uma redução de instruções e outros aprimoramentos.

Segundo Heath (2003), as análises feitas em computadores e *mainframes* CISC concluíram que a maior parte dos programas utilizavam unicamente 20% das instruções do processador. O objetivo da arquitetura RISC foi implementar um processador com melhor desempenho e instruções menos complexas em relação aos processadores CISC, consequentemente a um custo menor.

Figura 1 – Modelo de armazenamento de programa



Fonte: o autor.

Para Heath (2003), as principais características da arquitetura RISC são:

- Todas as instruções são executadas num ciclo único de *clock*.
- Possuem a característica *register-based manipulation*, na qual a unidade principal de memória é manipulada diretamente pelos registradores e não diretamente pelas instruções.
- Unidades de execução sem microcódigo.

2.1.1 Procesadores ARM

Os processadores *Advanced RISC Machine* (ARM) são baseados na arquitetura RISC, e segundo Furber (2020), foram os primeiros em serem desenvolvidos para uso comercial. Suas principais diferenças em relação à arquitetura RISC são:

- *Delay Branches*: devido aos problemas causados pelas interrupções do fluxo das informações, processadores RISC utilizam *branches* atrasados, que são

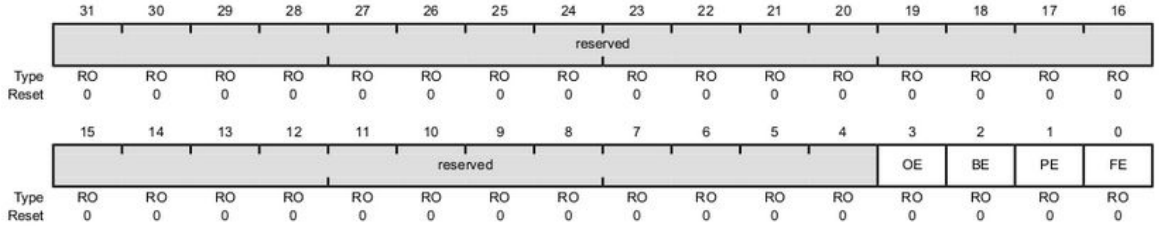
efetivados depois da instrução ter sido concluída. Na arquitetura ARM não são considerados por comprometer a atomicidade das instruções individuais.

- Único ciclo de execução para todas as instruções: como possuem um único módulo de memória para os dados e as instruções, cada operação precisa pelo menos dois ciclos de *clock*. O ARM foi projetado para utilizar a menor quantidade possível de ciclos de *clock* utilizando um *clock* extra.

O modelo de programação do ARM permite mudar os estados do sistema por operações definidas, sendo considerada cada instrução como uma transformação do estado anterior dos dados nos registradores do processador. Existem registradores visíveis (disponibilizados para o programador) e invisíveis (reservados unicamente para o uso do sistema).

Como exemplo, a Figura 2 apresenta os endereços do registrador UARTRSR para a comunicação serial UART de um processador Cortex M4 da Texas Instruments, que segue a arquitetura ARM (TEXAS INSTRUMENTS, 2014).

Figura 2 - Exemplo de registradores de 32 bits



Fonte: Tiva TM4C123GH6PM Data Sheet (2014).

Nessa figura é possível observar os 32 bits, dos quais são disponibilizados unicamente quatro para o programador.

Os periféricos, como o controlador de disco, a interface de rede e os demais dispositivos, são tratados como direções de memória com suporte para interrupções. Segundo (Furber (2020) os periféricos alocam a atenção do processador realizando interrupções de dois tipos: IRQ (*normal interrupt*) e FIQ (*fast interrupt*).

2.1.2 Raspberry PI

O Raspberry Pi é um computador em placa simples, desenvolvido pela Raspberry Pi Foundation. Baseada num processador BCM2837, *Advanced Risc Machine* (ARM) Cortex A53 de 64 bits quad-core (Raspberry Pi Foundation, 2021). A arquitetura usada neste processador é a Armv8, sendo assim um processador multicore com capacidades *Symmetric Multiprocessing* (SMP) e múltiplas SMP *clusters* (ARM, 2021).

Neste trabalho foi escolhido o Raspberry PI B 3 pelo fato de ter um processador BCM2837 que suporta multiparalelismo de processos, considerando a complexidade do sistema a ser projetado e integração de aplicações de visão computacional e comunicação, foi a característica predominante. Os núcleos do processador ARM trabalham a 1,2 GHz, resultando uma velocidade de processamento aceitável para a construção do protótipo.

Entre os principais recursos disponíveis da placa:

- Soc para câmera.
- 1 GB de RAM.
- Wi-Fi e Bluetooth 4.1BLE.
- 40 pinos de GPIO.
- Protocolos *Serial Peripheral Interface* (SPI) e *Inter-Integrated Circuit* (I^2C).
- GPIOs.

Como a Raspberry Pi Foundation tem iniciativas de hardware livre, é possível obter o projeto do hardware e criar distribuições customizadas em Linux para a placa.

2.2 Sistema operacional Linux

O sistema operacional é um elemento fundamental de um sistema computacional. Nesta seção são abordados os conceitos básicos do funcionamento e estrutura do *kernel* do Linux e a sua utilização em sistemas embarcados focando na comunicação entre a camada física e de aplicação. Uma ferramenta para o desenvolvimento de um sistema operacional embarcado também é apresentada.

2.2.1 Conceitos de Sistemas Operacionais

Um sistema operacional pode ser definido como um software que gerencia recursos do hardware e aplicações de software de um sistema computacional. Para Ward (2015) um sistema operacional é produto de um processo de abstração, dividindo o sistema em componentes, as quais são chamadas *layers* ou níveis, que são um conjunto de componentes que ficam entre a interação do usuário com o hardware.

2.2.2 Organização do Sistema Operacional Linux

O sistema Linux é organizado em três grandes *layers*:

- Interface de Usuário, que integra os componentes da Interface Gráfica de Usuário (GUI), servidores e *shell*.
- Linux *Kernel*, que é composta pela chamada do sistema, administração de processos, administração de memória e *drivers* de dispositivos.
- Hardware, que integra o processador (CPU), memória principal (RAM), discos e portos de rede.

O *layer* de mais baixo nível é o Hardware, o *kernel* fica entre as requisições solicitadas pelos serviços da *layer* de interface de usuário e as operações de processamento e leitura/escrita de memória do hardware.

Segundo Ward (2015) a principal diferença entre a *layer* de interface de usuário e o *kernel* é que este tem acesso irrestrito ao hardware, especificamente ao processador e memória. O modo usuário, por sua vez, tem acesso a pequenos espaços de memória e operações seguras do CPU.

Para o presente trabalho de pesquisa o *kernel* é o principal componente de estudo, já que é responsável e de alocar os processos e determinar as regiões de memória para interagir com os dispositivos e *drivers*, atuando como o operador do hardware do sistema.

2.2.3 *Drivers* e Integração de Periféricos

Os *drivers* são programas que contêm a informação e descrição de um dispositivo para o seu controle e interação pelo *kernel*. Segundo Ward (2015) um dispositivo é acessível unicamente se o *kernel* permite sua ativação.

Dispositivos típicos de um computador, como discos e dispositivos de rede, já tem integrados os *drivers* no *kernel*.

As descrições dos dispositivos e componentes do hardware recebe o nome de *device tree,* ou árvore de dispositivos, e o *kernel* utiliza essa estrutura de dados para gerenciar os periféricos.

Caso seja preciso integrar um novo periférico ao sistema, é necessário modificar o *device tree* e compilar novamente o *kernel.*

2.2.4 Sistema Operacional embarcado Escolhido.

Considerando a plataforma de hardware escolhida, foi considerado a utilização de um sistema operacional embarcado GNU/Linux, que suporte protocolos de comunicação SPI e I^2C e recursos para aplicações que contemplam multiprocessamento em paralelo.

O sistema operacional oficial do Raspberry PI B 3 é o Raspbian, baseado em Debian GNU/Linux. Por ser uma distribuição Linux, já vem com algumas ferramentas básicas utilizadas neste trabalho como: biblioteca para câmera, ferramentas de build GNU (gcc e makefile) e Python. A organização do *filesystem* também é genérica, o que facilita a navegação dentro do sistema.

Neste trabalho foram desenvolvidas aplicações nas layers do *kernel* e Interface de Usuário. Foi preciso integrar no sistema operacional drivers para o *Real Time Clock* (RTC) e configurações para a comunicação serial e ISP.

2.3 Módulos de comunicação sem fio

Os dispositivos de comunicação sem fio têm a capacidade de transmitir dados e informações sem a utilização de cabos. As distâncias de transmissão dependem da tecnologia utilizada. Nesta seção é apresentada a tecnologia *Global System for Mobile Communications* (GSM), assim como as características do módulo utilizado neste trabalho que a suporta.

2.3.1 *Global System For Mobile Communication*

Segundo Stuber (2017) GSM foi desenvolvido no ano 1992 como o primeiro sistema digital para celulares. Ele fornece serviços de dados assíncronos e síncronos, e o sinal e os canais de voz são completamente digitais, sendo considerado como um sistema de segunda geração (2G) de telefonia celular.

2.3.2 Módulo SIM900A

Neste trabalho o módulo SIM900A foi escolhido para integrar o sistema embarcado projetado pelas suas interfaces e por suportar microfone e caixa de som. A interface de comunicação serial foi fundamental na escolha do módulo, já que permite uma integração com o microcontrolador do módulo de processamento.

Um módulo SIM900A funciona com dupla banda, o GSM e o *General Packet Radio Services* (GPRS), e foi desenvolvido pela companhia SIMCOM (LIGANG, 2009).

A comunicação com plataformas embarcadas é via serial e contém uma interface SPI para integração e customização com outros sistemas. O *chip* tem integrado os protocolos TCP/IP e TCP/IP AT, que permitem a recepção de comandos AT pela interface serial, facilitando a sua programação.

No que corresponde ao consumo elétrico, o SIM900A foi projetado para ter um baixo consumo de energia, chegando a consumir 1,5 mA em modo *sleep* (LIGANG, 2009).

A Figura 4 apresenta uma fotografia do módulo SIM900A.

Figura 3- Módulo SIM900A

Fonte: extraído de (LIGANG, 2009).

A pinagem desse módulo é apresentada na Tabela 1.

Tabela 1 -Pinagem SIM900A

pino	**label**
1	dcc
2	dtr
3	txd
4	rxd
5	speaker
6	microphone
7	reset
8	gnd

Fonte: extraído de (LIGANG, 2009).

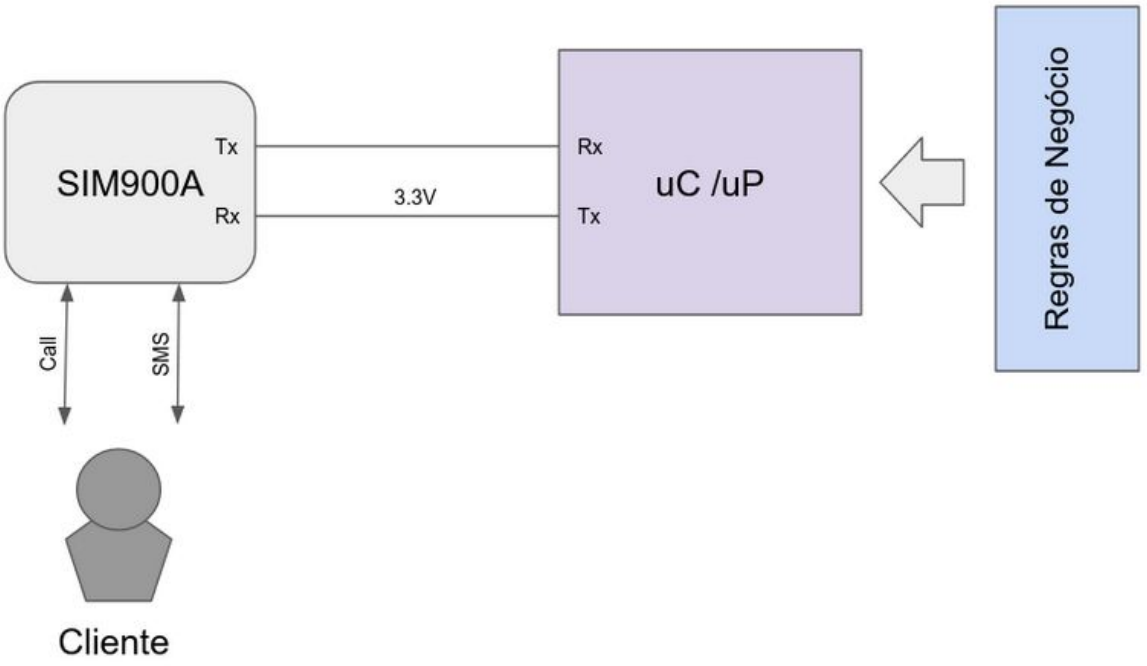
Esse módulo requer de 4,5 V a 5 V como tensão de alimentação, mas o nível lógico do circuito é 3,3 V, de forma que o valor enviado da plataforma embarcada ao pino de recepção não pode exceder esse valor.

Para estabelecer comunicação com a rede de telefonia é necessário inserir um cartão *Subscriber Identity Module* (SIM) de qualquer operadora no módulo.

2.3.3 Aplicações

Para utilizar as funções GSM do módulo é preciso uma comunicação serial com uma plataforma microcontrolada ou microprocessada. A interação mediante ligações ou envio de SMS podem ser definidas em função das regras do negócio do projeto, como mostra a Figura 5.

Figura 4 - Aplicação do SIM900A



Fonte: o autor.

2.4 Componentes de interface de usuário

No presente trabalho de pesquisa, os componentes da interface com o usuário são classificados como elementos de hardware ou software controlados pelo sistema, interagindo diretamente com os usuários finais.

2.4.1 Periféricos de Hardware

Os periféricos são dispositivos que se conectam nas portas de entrada/saída para ter acesso aos barramentos do computador. Segundo Floyd (2006) o computador envia os dados a um dispositivo por uma porta de saída do dispositivo periférico, e recebe a informação mediante uma porta de entrada.

Para Floyd (2006), as vias são caminhos para os sinais digitais, sendo que um dos mais comuns é o *Universal Serial Bus* (USB).

No projeto deste trabalho de pesquisa foram integrados quatro dispositivos periféricos conforme são apresentados na tabela 2, em função das necessidades levantadas na justificação do projeto.

Tabela 2 - Periféricos e interfaces de usuário

Periférico	**Interação com o usuário**
Real Time Clock	Noção e medição do tempo e data atual
Módulo GSM	Envio/recepção de SMS e ligações
Câmera	Detecção de pessoas
LCD	Configuração de funcionalidades do sistema

Fonte: o autor.

A continuação são detalhados os principais usos dos periféricos:

- Real Time Clock (RTC) conectado no barramento I^2C. Ele foi escolhido pois o Raspberry Pi não possui dispositivos para mensurar o tempo.
- Módulo GSM na interface Serial. Ele foi escolhido pela necessidade de se enviar SMS como interação com o mundo externo.

- Câmera conectada na *Camera Serial Interface* (CSI). Trata-se de um dispositivo periférico importante no sistema, pois realiza a coleta dos dados necessários para a determinação do fluxo de pessoas.
- Display de cristal líquido (LCD) conectado na interface USB, destinado à interação entre o usuário e o dispositivo.

Pela quantidade de dispositivos periféricos utilizados foi necessário realizar a configuração dos *drivers* no *device tree* do sistema operacional.

Na Tabela 2 são apresentados os dispositivos periféricos e suas respectivas interações com o usuário.

2.4.2 Projeto de *Dashboards*

Os *dashboards* são instrumentos para a comunicação efetiva de dados, e para Few (2006), eles oferecem uma única e poderosa forma de comunicação, mas raramente atingem seu potencial, não pela tecnologia utilizada, e sim pelo projeto e implementação.

Neste trabalho os *dashboards* foram apresentados como um valor agregado, com o objetivo de apresentar os dados coletados e análises para a tomada de decisões de gerenciamento de operações.

Sendo o seu objetivo principal a comunicação com o usuário, é difícil padronizar a construção de *dashboards*, mas existem boas práticas que podem auxiliar o seu desenvolvimento. Algumas dessas práticas identificadas por Few (2006) são:

- contraste entre uma medida principal e as medidas derivativas ou secundárias;
- categorização da visualização dos elementos de análises;
- apresentação de medidas comparativas de dados históricos;
- variedade de tipos de gráficos para enfatizar medidas heterogêneas.

2.5 Algoritmos de visão computacional

A visão computacional trata das metodologias e técnicas para a extração de informação, para Solem (2012) a informação extraída pela visão computacional pode ser modelos 3D, detecção de objetos ou reconhecimentos de grupos.

Algumas vezes a visão computacional tenta imitar a visão humana. As técnicas para atingir esses objetivos envolvem modelos matemáticos, programação e estatística.

2.5.1 Software utilizado

A implementação de soluções de visão computacional implica na manipulação de imagens, vídeo *streaming* e matrizes.

Python é uma linguagem de programação que contém conjuntos de APIs para facilitar as implementações de soluções de visão computacional, sendo os módulos mais importantes:

- Pilow (*Python Imaging Library*), módulo para manipulação de imagens que permite a sua conversão para diferentes formatos, editar segmentos e dimensões.
- Numpy, módulo de álgebra linear para a manipulação e criação de *arrays* e matrizes, que contém classes para a geração de números aleatórios e *ranges*.
- Matplotlib, módulo para a construção de gráficos em 2D que contém classes para a criação de gráficos estatísticos.

Além da linguagem de programação e seus módulos para a manipulação de imagens, é preciso utilizar uma biblioteca que contenha a implementação dos principais algoritmos matemáticos para visão computacional.

O *OpenCV* é uma biblioteca desenvolvida pela Intel, *open source,* com a finalidade de auxiliar o desenvolvimento de programas para visão computacional em tempo real, e possui uma interface para a linguagem Python. Segundo Solem (2012), os módulos do *OpenCV* abrangem grandes áreas da visão computacional, contendo funções para a leitura e escrita de imagens, assim como operações matriciais.

2.6 Algoritmos de Visão Computacional

Neste item são apresentados os algoritmos para a detecção e *tracking* de objetos, sob uma perspectiva computacional do *OpenCV*.

Segundo SOO (2013), uma imagem é geralmente tratada em baixo nível para a remoção de ruídos. A maioria de recursos computacionais realiza a análise das imagens ou *frames* em cores da escala de cinza, por exemplo, para reduzir o custo computacional.

2.6.1 Detecção de Objetos

Nos casos em que é aplicada a detecção de um objeto, o objetivo é determinar em que imagem, ou *frame,* se encontra o objeto. Para Rosebrock (2017), o custo computacional desses algoritmos é bastante elevado. Entre os algoritmos de detecção estão os *Haar Cascades*, HOG + Linear SVM e os algoritmos baseados em *deep learning:* Faster R-CNNs, YOLO e *Shot Detectors* (SSDs).

O principal objetivo da técnica é classificar se um determinado objeto pertence ou não à classe de interesse, em função das suas de uma imagem ou *frame* digital.

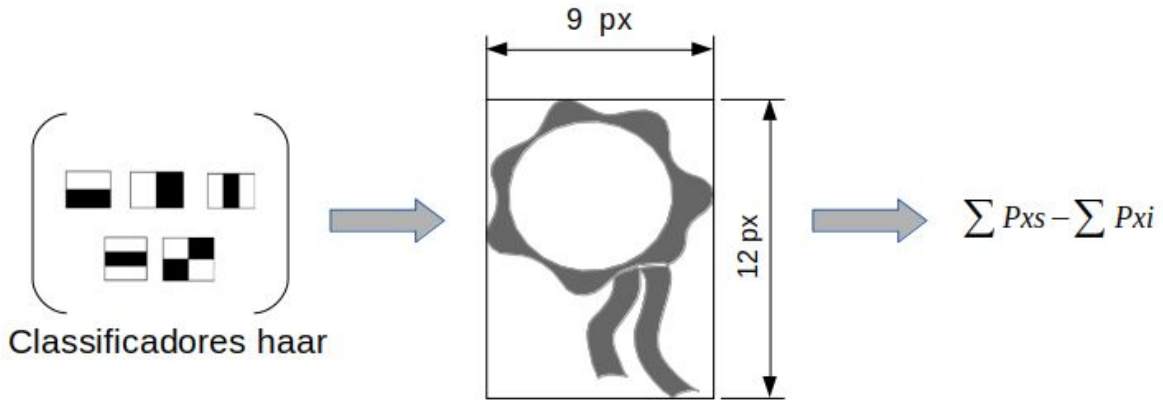
2.6.1.1 Algoritmos *Haar Cascade*

Os algoritmos *Haar Cascade* são muito comuns nas aplicações de detecção, e algumas de suas características principais são as regiões retangulares dos *bounding boxes* (identificador de objeto detectado em função de altura e comprimento). Segundo Rosebrock (2017), o algoritmo realiza uma análise de todos os *pixels* da imagem, sendo considerado um algoritmo computacionalmente custoso.

O classificador *Haar Cascade se* destina à classificação de aprendizado supervisionado, e utiliza diversos estágios (ou *stages*) para o seu treinamento, nos quais a sua qualidade é diretamente proporcional ao número de *stages*. O treinamento consiste em sobreposicionar (rotacionalmente) a imagem do objeto que se pretende detectar, e uma grande quantidade de imagens que não contêm as características do objeto (denominadas imagens negativas). Cada *stage* realiza o reconhecimento das imagens positivas até ter um resultado favorável (máximo número de detecções positivas e menor número de falsos positivos).

O mecanismo de detecção dos algoritmos *Haar Cascade* consiste na interação de classificadores geométricos para a extração das características principais das imagens positivas. A Figura 7 apresenta esse processo de interação.

Figura 5- Representação do algoritmo de classificação Haar Cascade



Fonte: o autor.

O resultado de cada iteração consiste na subtração da soma dos *pixels* acima da área branca e embaixo da área preta de cada retângulo. O processo de cálculo pode ser muito extenso, considerando imagens com tamanhos maiores, mas pode ser otimizado pela aplicação de um conceito introduzido por Viola & Jones (ZHANG, 2010) denominado *Integral Imagen*, construída pela seguinte fórmula:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

onde ii(x,y) é a imagem integral em *pixel* na posição x e y, as medidas originais da imagem são representadas por x´ e y´.

Utilizando a Fórmula 1 é possível obter a área em *pixels* de qualquer região retangular (Fórmula 2):

$$\sum_{(x,y) \in ABCD}^{n} i(x,y) = ii(D) + ii(A) - ii(B) - ii(C) \quad (2)$$

considerando o seguinte *dataset* de treinamento (Fórmula 3):

$$S = \{ (x1, z1),\ i = 1, \ldots, N \}\ \forall\ x1 \in X,\ Z1 \in Z \quad (3)$$

sendo X o domínio do “instance space”, ou seja, o possível espaço de características (ou *features*) obtido das imagens positivas a serem treinadas. O “label space” corresponde ao domínio da variável que será predita, representada por Z.

O algoritmo de aprendizado de máquina *Adaboost* produz um modelo aditivo, para prever o *label* em função de cada *input feature* (Fórmula 4):

$$F^T(x) = \sum_{t=1}^{T} f_t(x) \qquad \forall \quad F^T(x) \ : X \rightarrow R \tag{4}$$

A expressão do membro esquerdo da fórmula indica que é um valor real obtido do vetor das *features*. O total de “weak classificicators” (classificadores que individualmente são melhores que qualquer randômico) é representado pelo T. Finalmente, o algoritmo realiza uma regressão logística na função aditiva (Fórmula 5):

$$L^T = \sum_{i=1}^{N} exp\left\{ - ZiF^T(Xi) \right\}. \tag{5}$$

O algoritmo tenta achar a melhor função aditiva possível, sempre que um vetor de *features* é dado como entrada. Na Fórmula 2 é possível observar que para obter a área em *pixel* total será preciso a utilização de quatro *arrays*. Considerando a forma geométrica retangular dos classificadores, que por natureza os retângulos compartilham esquinas, é possível reduzir a quantidade de *arrays* conforme é realizada a interação dos classificadores.

Identificadas as principais características de forma otimizada pelo conceito de *integral image*, é preciso moderar essas características em relevantes e irrelevantes para a detecção do objeto, e isso pode ser atingido pela aplicação do algoritmo de *AdaBoost*. Segundo Zhang (2010), o algoritmo realiza a seleção das melhores características em função de um *dataset* de treinamento, realizando assim um *score* entre as imagens negativas e positivas no *dataset*.

2.6.1.2 *Deteção de objetos*

Os algoritmos de *Object Tracking* (OT) realizam predições com o objetivo de calcular a posição-deslocamento em função do estado inicial do objeto num determinado *frame* de vídeo.

Para Wu, Li e Yang (2013), o principal objetivo do OT é estimar os estados do objeto nos *frames*. Existem vários fatores que podem afetar essa predição, principalmente os físicos, como iluminação, fundo, etc.

Existem *dataset* genéricos treinados para facilitar o desenvolvimento de OT, mas segundo Wu, Lim e Yang (2013), na maioria o objeto de estudo são humanos ou automóveis em background estáticos.

2.6.2.1 Componentes

Para Wu, Lim e Yang (2013) o **esquema de representação** (*Representation Scheme*) é o principal componente visual do OT. Entre os principais estão as representações por histogramas de cor e gradientes de orientação.

O **mecanismo de busca** é um componente que se destina a realizar a estimativa dos estados do objeto. Segundo Wu, Lim e Yang (2013) são utilizados métodos estocásticos e determinísticos. Geralmente as funções objetivo são não lineares e contém muitos mínimos locais. Para otimizar esse problema são utilizados algoritmos estocásticos como os filtros de partículas, aumentando a eficiência computacional.

2.6.2.2 Metodologias

Segundo Wang et al. (2019), é possível aumentar a operacionalidade e velocidade adotando o *fully-convolutional siamese framework*, utilizando como bloco principal do *tracking system* uma função. A seguir é apresentado o modelo **SiamFC** (Fórmula 6):

$$g_\theta(z,\ x)\ = f_\theta(z)\ *\ f_\theta(x). \qquad (6)$$

Neste exemplo tem-se a variável z e a variável x como a imagem de busca. A função realiza uma comparação entre as duas variáveis com o objetivo de obter o mapa dos centroides de massa das imagens é o maior centroide na última posição estimada. É assim que as similitudes entre os as entradas x e z são avaliados.

Segundo Wang et al. (2019), existe uma grande importância em produzir por *frame* uma máscara binária de segmentações (*binary segmentation masks*) com o propósito de ir mais além das pontuações de similitudes e coordenadas de *bounding box*, produzindo uma máscara binária por *pixel*, conseguindo, assim, a predição da máscara binária por *frame* (Fórmula 7):

$$m_n = h_\phi(g^n_\Theta(z, x)) \quad (7)$$

A Fórmula 2 apresenta a função de predição de máscara (m_n), que é a relação entre duas variáveis: z e x, a imagem a segmentar e o objeto a realizar o *tracking* respectivamente. Com essa nova relação é possível, em função de uma imagem de referência, a rede neural do *framework* retornar uma máscara de segmentação diferente.

3. Análise e controle do fluxo de pessoas em ambientes fechados

3.1 Engenharia de Requisitos

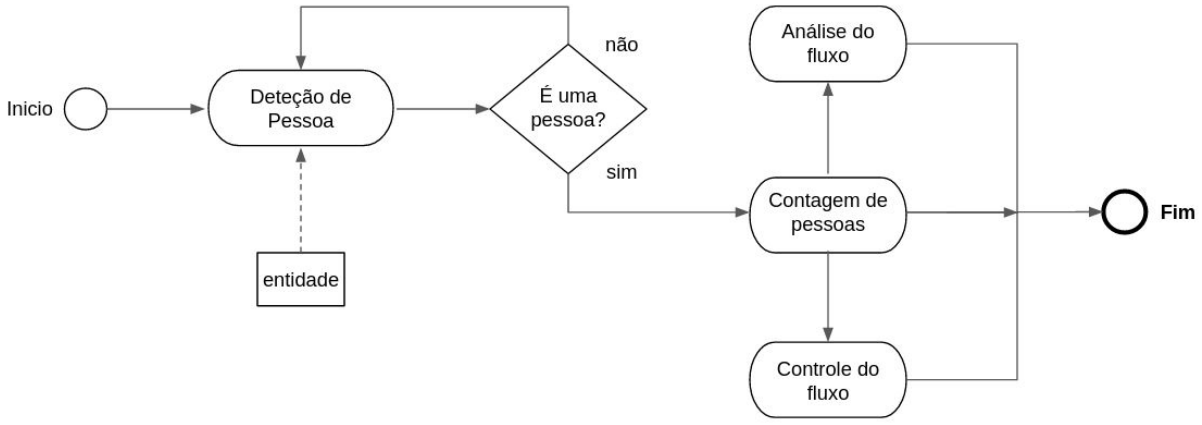
Para a definição da arquitetura do sistema foi necessário identificar os requisitos funcionais e não funcionais do sistema, visando atingir os objetivos planejados.

Para a identificação dos requisitos funcionais foram determinados os casos de uso do sistema.

3.1.1 Casos de uso Gerais do sistema

O sistema foi projetado com a finalidade de controlar a passagem de pessoas num ponto específico, trocando informação constantemente com o ambiente externo e gerando inteligência para fins estratégicos. O Diagrama 1 descreve as funcionalidades gerais do sistema.

Diagrama 1 - Funcionalidades gerais do sistema



Fonte: o autor.

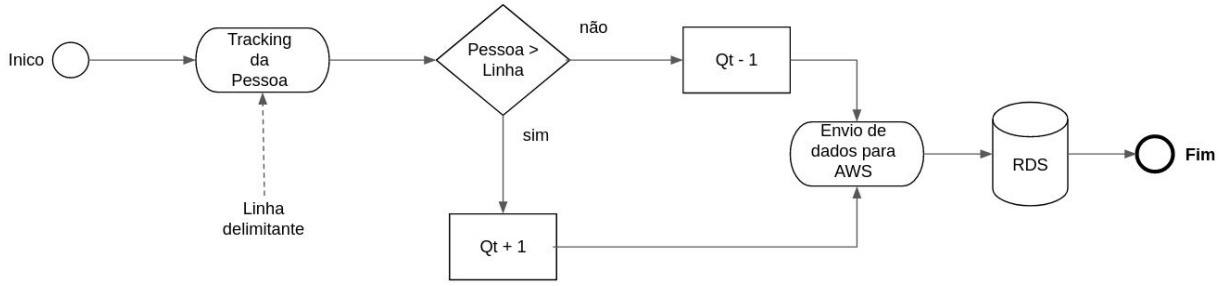
Conforme apresentado no Diagrama 2, o sistema tem quatro principais casos de uso, iniciando com a detecção de pessoas como *trigger* principal, incluindo o *setup* inicial que é a calibração da câmera. O sistema realizará a detecção da entidade no range configurado, resultando numa decisão. Se a entidade for uma pessoa ou grupo de pessoas será realizado o processo de contagem e na sequência realizados dois processos em paralelo de análise e controle do fluxo.

Os casos de uso de contagem, análise e fluxo de pessoas, possuem uma complexidade maior, pelo que serão detalhados separadamente.

3.1.2 Contagem de Pessoas

É o macroprocesso principal, já que deriva em dois processos paralelos (análise e controle do fluxo). Ele é detalhado no Diagrama 2.

Diagrama 2 - Caso de uso contagem de pessoas



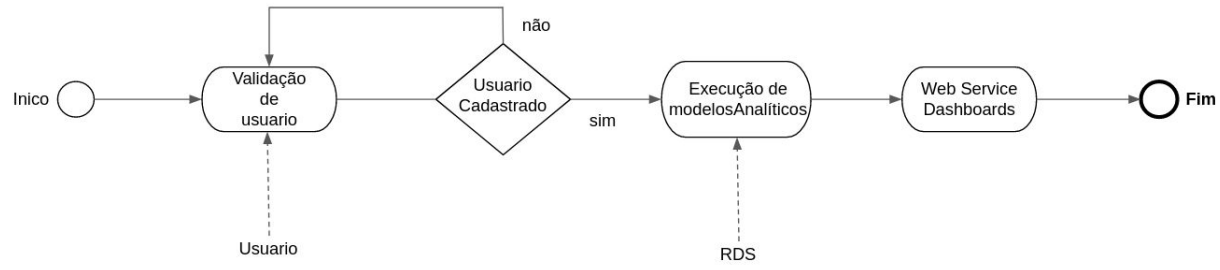
Fonte: o autor.

Detectada uma pessoa, é preciso realizar um acompanhamento do deslocamento da mesma para determinar se está entrando ou saindo do ponto de controle, para isso é definida uma linha de referência no setup do sistema. Os dados são enviados para a nuvem e escritos no serviço RDS.

3.1.3 Análise do Fluxo de Pessoas

A análise do fluxo, consiste em apresentar para o usuário, mediante uma ferramenta visual, estatísticas em tempo real (Diagrama 3).

Diagrama 3 - Caso de uso de Análise de pessoas



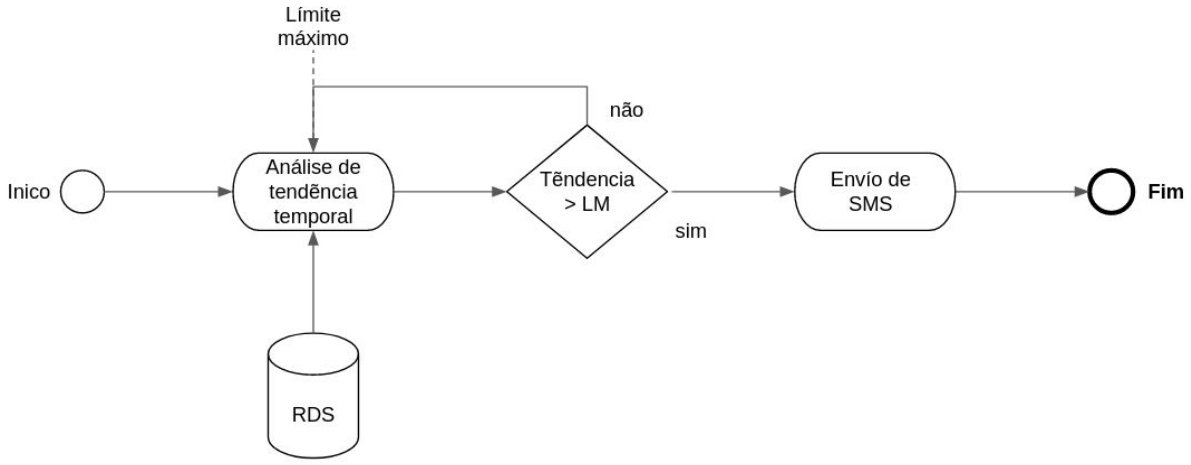
Fonte: o autor.

O *web service* de *dashboards* é a interface homem-máquina. O acesso é fornecido pelo *chatbot*. Todo o processo é executado na nuvem, no serviço EC2 *Elastic Compute Cloud* (EC2).

3.1.4 Controle do Fluxo de Pessoas

O controle do fluxo de pessoas envolve a interação entre o sistema computacional e os indivíduos de tomada de decisão (Diagrama 4).

Diagrama 4 - Caso de uso de Controle de pessoas



Fonte: o autor.

O parâmetro de entrada é o limite máximo, ingressado pelo usuário no *setup* do sistema. Os dados em tempo real são analisados em séries temporais para estimar o tempo que irá estourar o limite máximo do fluxo de pessoas num determinado ponto.

3.1.5 Requisitos Funcionais

Em função dos casos de uso do sistema foram definidos os requisitos funcionais de acordo com as necessidades do projeto. A tabela 3 apresenta os requisitos funcionais propostos.

Tabela 3 - Requisitos funcionais do projeto.

REQUISITO FUNCIONAL	**NECESSIDADE**
Contagem da entrada e saída de pessoas num ambiente fechado por visão computacional.	Calcular o fluxo de pessoas

Controle parametrizado do fluxo de pessoas por comunicação SMS em tempo real.	Alertar ao usuário de incidentes acontecidos
Apresentação visual de dados por *Dashboards*.	Comunicar efetivamente dados do processo.
Armazenamento e processamento dos dados em plataforma na nuvem.	Disponibilizar informação eficazmente
Exibição dos dados de estado do hardware por via eletrônica.	Informar o estado físico do sistema

Fonte: o autor.

Os requisitos não funcionais do projeto, apresentados na Tabela 4, foram definidos em função das características técnicas indiretas que garantem desempenho ao projeto.

Tabela 4 - Requisitos não funcionais do projeto.

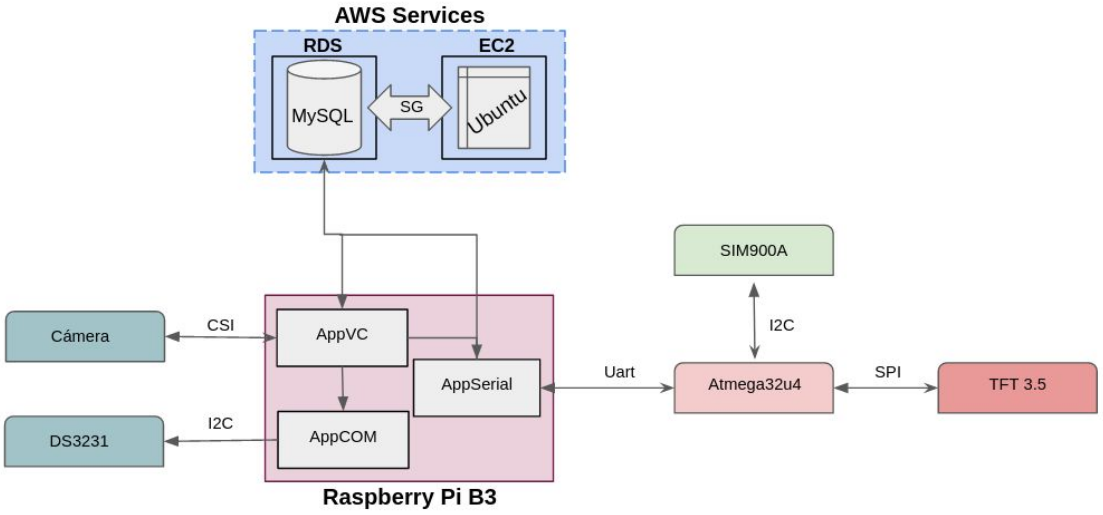
REQUISITOS NÃO FUNCIONAIS	**NECESSIDADE**
Alta escalabilidade das aplicações.	Crescimento dinâmico da solução
Utilização de linguagens de programação *multithread* na construção do software.	Paralelismo de aplicações
Interface de configuração simples e interativa.	Facilidade de interagir com o sistema

Fonte: o autor.

3.2 Arquitetura Proposta

A Figura 8 apresenta a arquitetura do sistema proposto para o controle do fluxo de pessoas num ambiente fechado.

Figura 6- Arquitetura proposta



Fonte: o autor.

A arquitetura contempla três elementos principais:

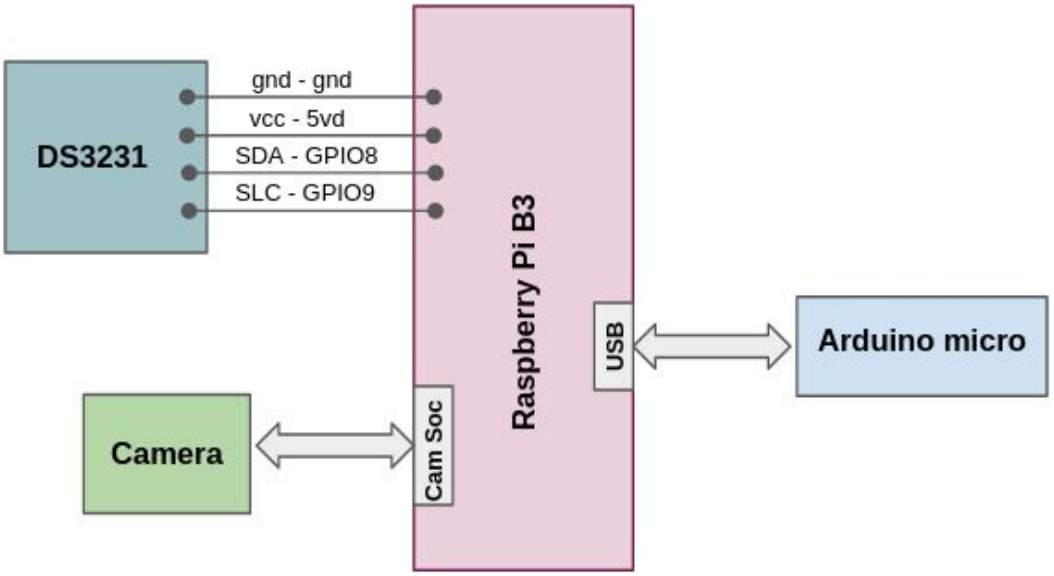
- **Módulo com microprocessador:** o Raspberry Pi B3 é o elemento principal do sistema, hospedando as três aplicações de software que realizam o controle de fluxo das pessoas no ambiente, interagindo também com o hardware periférico. O sistema operacional é o Raspbian, que é uma distribuição otimizada para a plataforma baseada no Debian.
- **Periféricos:** são dispositivos eletrônicos com a capacidade de interagir com o mundo físico e o usuário. A câmera, módulo de relógio de tempo real e o microcontrolador Atmega32u4 têm comunicação direta com o Raspberry Pi nos seus respectivos protocolos de comunicação. O microcontrolador é responsável pelo controle do módulo GSM e do *Display* LCD.
- **Computação na Nuvem:** o dispositivo físico realiza a troca de dados com o *Amazon Web Services*, especificamente para o serviço do banco de dados relacional (RDS). Foram criadas regras de entrada configuradas num *Security Group* (SG) para a comunicação com uma instância *Ubuntu Server* no serviço

de EC2, onde os dados são processados e no qual fica hospedado o *dashboard* de controle em tempo real, acessado pelo IP pública pelo usuário.

3.3 Detalhe dos Componentes de Hardware

A placa de desenvolvimento Raspberry Pi tem as conexões com os periféricos de hardware apresentados na Figura 9.

Figura 7 - Integração de componentes de hardware



Fonte: o autor.

A Figura 9 apresenta as conexões entre os periféricos principais e o Raspberry Pi. As conexões com o módulo RTC estão diretamente nos GPIOs da placa especificamente no GPIO I^2C.

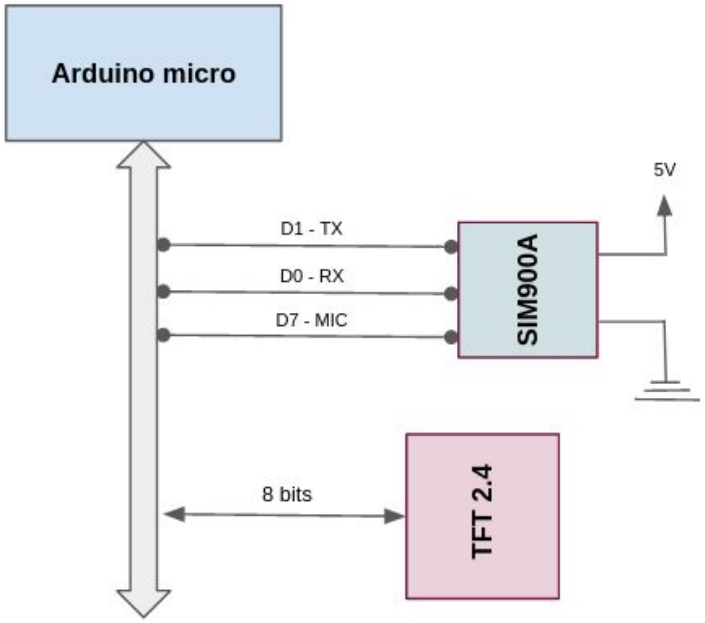
A câmara tem um cabo de conexão *flat* para o SoC do Raspberry Pi. A comunicação é sinalizada por um LED na câmera.

O Arduino micro contém o microcontrolador Atmega32u4. A placa de desenvolvimento contém uma entrada para micro USB e um *chip* conversor USB-serial.

Na placa Raspberry Pi é preciso realizar a configuração no *kernel* para habilitar a comunicação. Existe um utilitário presente no Raspbian, o *Raspi-config*, que permite habilitar as entradas da câmera e a comunicação I^2C.

Para o controle do módulo GSM e do *display LCD* foi utilizada a placa de desenvolvimento Arduino micro, cujas conexões no barramento de entradas e saídas são detalhadas na Figura 10.

Figura 8 - Conexões de periféricos no microcontrolador



Fonte: o autor.

O módulo GSM com o *chip* SIM900A funciona com alimentação constante de até 5,5 V e pode ser alimentado pelo Arduino, mas para tanto é preciso realizar uma divisão de tensão na sinal enviada da placa para o módulo, devido ao nível lógico de 3.3V . Para este trabalho não foi adicionado microfone nem caixa de som, sendo utilizadas unicamente as funcionalidades SMS.

O *display LCD* é controlado na funcionalidade de 8 bits, permitindo ao usuário olhar as configurações do sistema.

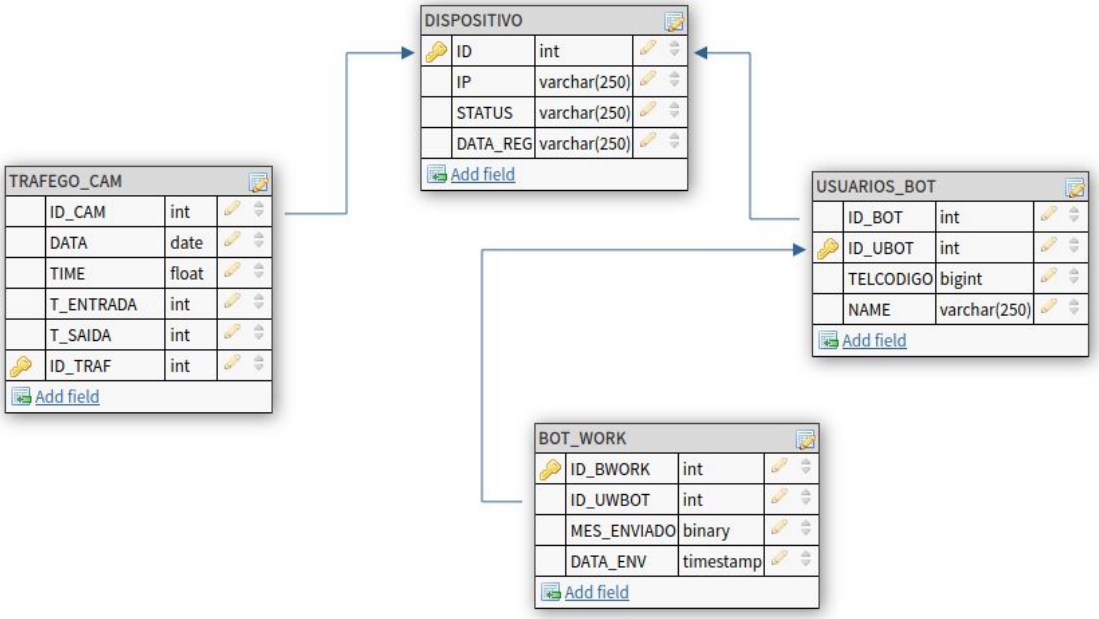
3.4 Detalhe dos Componentes de Software

Os componentes de software foram hospedados no módulo Raspberry Pi (Aplicação visão computacional, aplicação de comunicação e aplicação serial) e em Serviços da Nuvem da AWS (*Dashboard* e *Database*).

3.4.1 Banco de Dados

A base de dados do sistema é baseada no *Relational Database Service* (RDS), numa instância MySQL. A Figura 11 apresenta o modelo relacional da aplicação.

Figura 9- Modelo Relacional da database do sistema



Fonte: o autor.

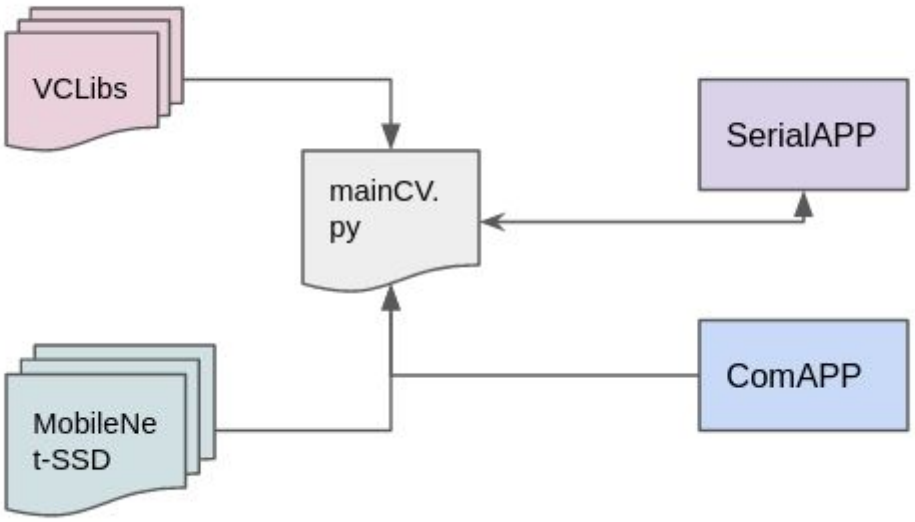
O modelo contempla quatro tabelas:

- **Dispositivo**: é a tabela principal do modelo, e contempla as informações básicas do dispositivo.
- **Trafego_Cam**: contém as informações da aplicação de IA, monitorando a passagem de pessoas.
- **Usuários_Bot:** é uma tabela que registra os dados dos usuários cadastrados na aplicação.
- **Bot_Work**: tabela de detalhe da tabela Usuários_Bot, que contém os registros do *triggers* do sistema.

3.4.2 Aplicação de visão computacional (AppVC)

A aplicação principal do sistema é de visão computacional, sendo que esta aplicação tem a seguinte estrutura detalhada na Figura 12.

Figura 10- Organização do software da aplicação principal



Fonte: o autor.

Toda a aplicação foi escrita na linguagem Python, e o arquivo **main** foi desenvolvido considerando técnicas de programação *multi-thread* para garantir a funcionalidade em tempo real e comunicação com as outras aplicações.

A pasta MobileNet-SSD contém os modelos treinados (**cafe deep learning)**, utilizando o *MobileNet Single Shot Detector* (SSD) para identificar os movimentos das pessoas.

A pasta VClibs contém arquivos desenvolvidos numa visão de programação orientada a objetos (POO), as classes criadas são utilizadas para consulta no banco de dados, algoritmos de *centroid tracking*, comunicação com as aplicações serial e COM.

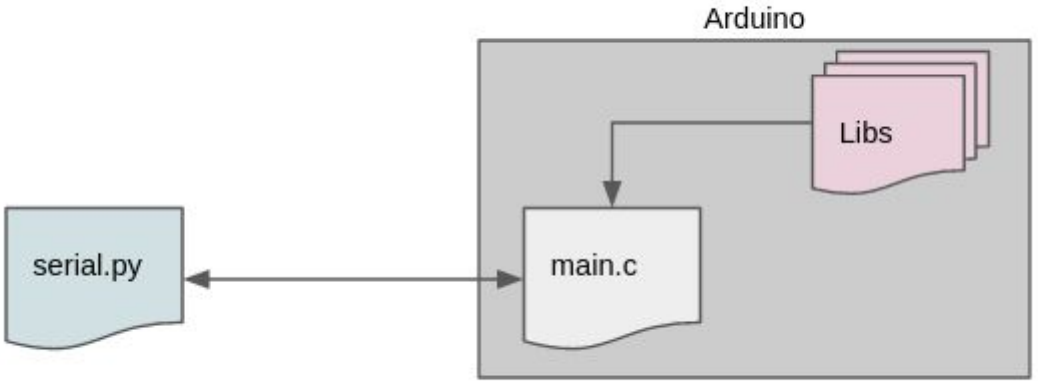
A comunicação com a aplicação serial (SerialAPP) é bidirecional. Os dados de configuração de sistema coletados pelo *display Touch* são processados, assim também são enviados os comandos AT para o módulo GSM.

Os dados de tempo são recebidos pela aplicação COM, a comunicação é unidirecional.

3.4.3 Aplicação serial (App Serial)

Aplicação para a comunicação com a placa Arduino micro, a estrutura da aplicação contempla integração entre o microcontrolador e Raspberry Pi. Foi desenvolvido um *firmware* na linguagem C e uma interface na linguagem Python. A Figura 13 apresenta essa estrutura.

Figura 11 - Software para Comunicação entre aplicação serial e microcontrolador



Fonte: o autor.

O arquivo **main** contém a implementação principal para o controle dos periféricos pelo microcontrolador, e foi desenvolvido segundo o paradigma de programação estruturada. Na pasta **libs** ficam os *headers* das bibliotecas para acesso ao TFT e SIM900A. É preciso ter uma interface entre o barramento serial e o microcontrolador, para isso foi desenvolvido o arquivo **serial.py**, realizando a interface e enviando os dados para a aplicação de software principal.

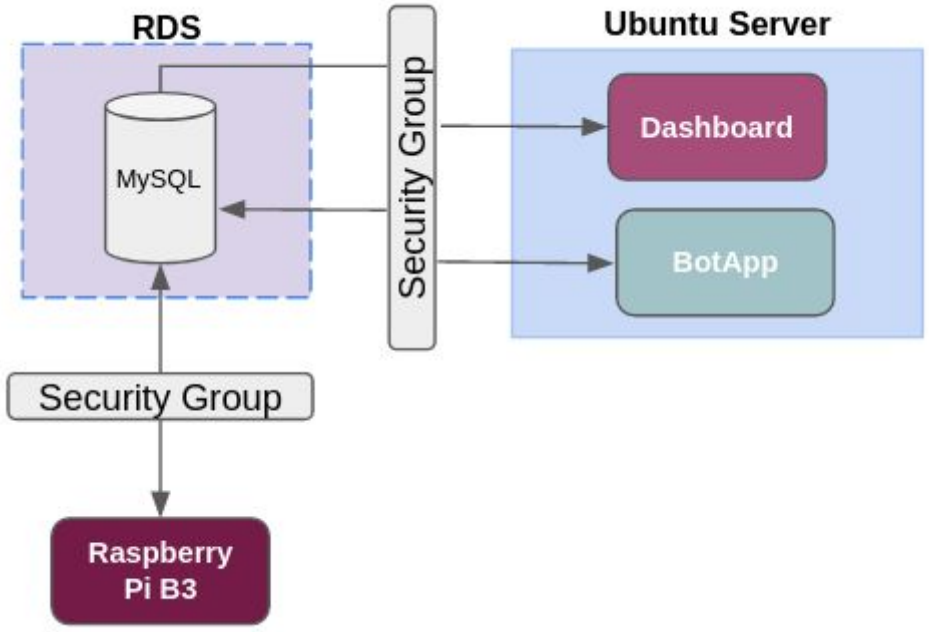
3.4.4 Aplicação de comunicação (AppCom)

A aplicação COM contém um arquivo desenvolvido na Linguagem Python, que realiza requisições para o módulo RTC via protocolo de comunicação I^2C. As informações de data e hora são integradas na aplicação principal.

3.4.5 EC2

O *Elastic Compute Cloud* (EC2) é um serviço da AWS que permite criar instâncias com imagens de diferentes sistemas operacionais. Neste trabalho foi proposta a criação de uma instância *Ubuntu Server*, sendo que a Figura 13 detalha os serviços hospedados no servidor.

Figura 12- Integração de aplicações na AWS



Fonte: o autor.

Existem duas aplicações principais, que são executadas simultaneamente e interagem com o serviço RDS.

A aplicação *Dashboard* foi desenvolvida em *Dash framework* para Python baseado no framework web para Python *flask*, gerado um servidor *web*, utilizando um IP público para visualizar a informação. A comunicação com o serviço RDS é unidirecional, unicamente para a consulta de informação.

Existe um serviço que gerencia as mensagens do *Telegram* em função dos dados coletados pelo sistema. A comunicação com o serviço RDS é bidirecional, já que precisa ler os dados e escrever na tabela de registro.

3.5 Resultados do Protótipo

A Tabela 5 apresenta os resultados da prova de conceito em função dos requisitos funcionais do projeto. A última coluna apresenta um *link* para um filme no Google Drive que demonstra o funcionamento do respectivo requisito funcional do protótipo.

Tabela 5 - Resultados do Protótipo do Projeto

Objetivo	Requisito Funcional	Protótipo	Link
Controle de fluxo de pessoas por imagens em tempo real.	-Contagem da entrada e saída de pessoas num ambiente fechado por visão computacional. -Apresentação visual de dados .	-Serviço AppVC para a detecção de saídas/entradas de pessoas. -Envio dos dados para a AWS. -*Deploy* de *dashboards* num serviço web na EC2.	https://drive.google.com/drive/u/0/folders/1qfFQ1XtG2OEayTW_oOe0vyYwab4cHFje
Alertas operados por módulos de comunicação sem fio.	Controle parametrizado do fluxo de pessoas por comunicação SMS em tempo real.	-Serviço AppVC para o controle e análise temporal do fluxo. -Serviço AppCOM para a comunicação com o módulo GSM pelo microcontrolador.	https://drive.google.com/drive/u/0/folders/1kk3-13G4ixnuHUMs-avTAOyfB_oNYb0F
Interação do dispositivo com o usuário.	Exibição dos dados de estado do hardware por via eletrônica.	-Serviço AppCOM para envio de dados do status do hardware e operação para o LCD controlado pelo microcontrolador	https://drive.google.com/drive/u/0/folders/19hmDDewhs05mHPoEvhD_HKR_J8lds6a6

Fonte: o autor.

Os requisitos funcionais foram implementados no protótipo, permitindo avaliar o cumprimento dos objetivos.

No controle do fluxo por visão computacional, o serviço principal AppVC envia os dados do fluxo de pessoas para a nuvem num ponto específico em tempo real. O serviço de *Dashboards* foi colocado em produção no serviço EC2, garantindo uma alta disponibilidade da visualização das informações.

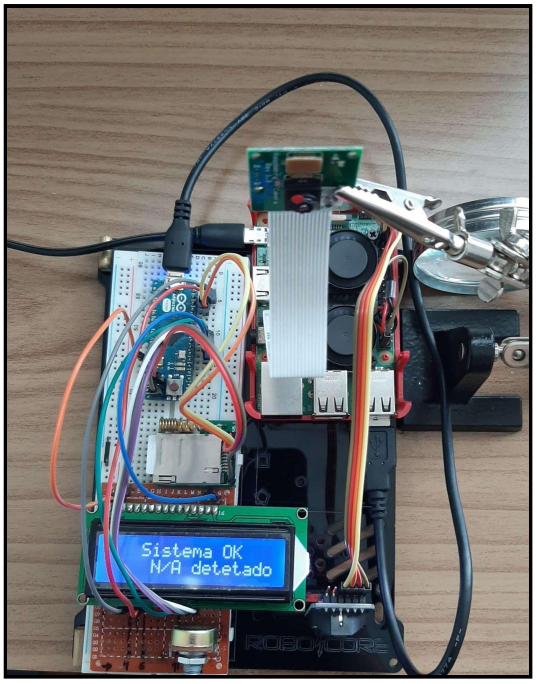
O serviço AppCOM centralizou a troca de informações entre o microprocessador com o módulo GSM e LCD, que atuam como interfaces de usuário. Na apresentação das informações do *status* de sistema apresentou um bom desempenho, mas no envio de SMS foi identificada uma latência, o mesmo ocorrendo no envio dos comandos AT pelo protocolo serial, pois não foi implementado um sistema de tempo real no microcontrolador.

Os serviços desenvolvidos neste protótipo funcional, foram projetados numa visão de multithread, devido a complexidade do processo e a quantidade de periféricos, sendo executados num sistema operacional debian no módulo com microprocessador.

Os periféricos de interação com o usuário (LCD e GSM) foram centralizados no microcontrolador, isso facilitou a comunicação com o Raspberry Pi, utilizando uma porta USB do módulo.

Na Figura 15 é apresentada uma fotografia do protótipo. Depois da realização do *boot* do sistema operacional é realizada a conferência da comunicação com o microcontrolador.

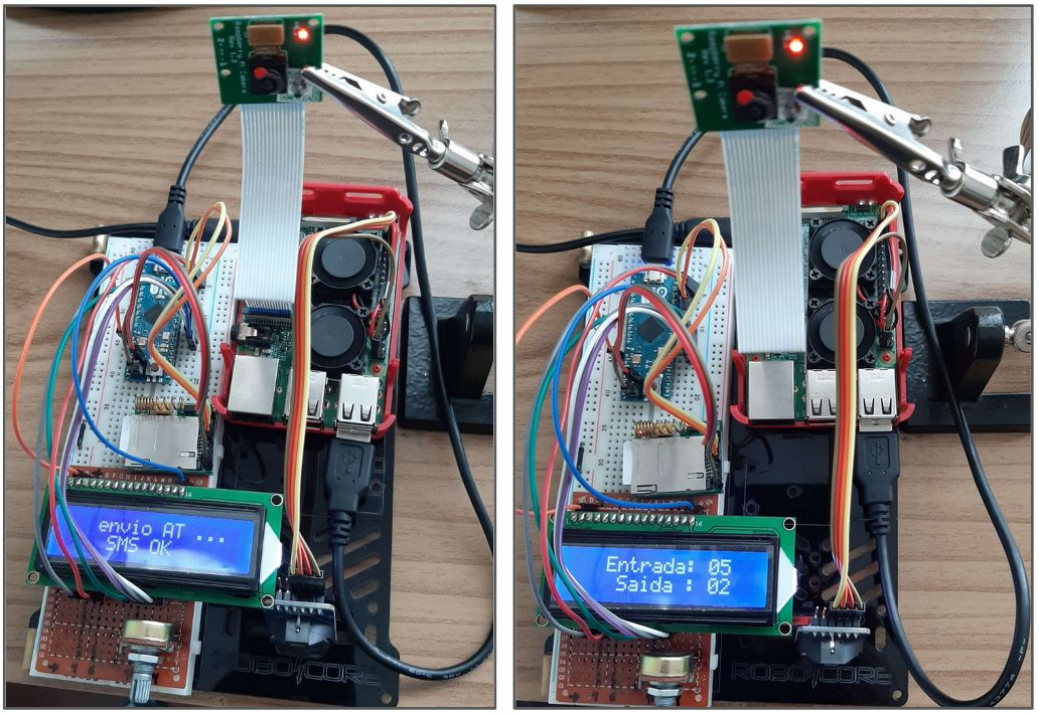
Figura 13- Protótipo em Operação



Fonte: o autor.

A Figura 16 apresenta a interação entre o protótipo com o usuário pelo envio de SMS e visualização da passagem de pessoas no LCD.

Figura 14- Envio de SMS e Visualização da Passagem de Pessoas



Fonte: o autor.

3.6 Considerações sobre o Capítulo

Neste capítulo foi apresentada a arquitetura do projeto em função dos requisitos funcionais e não funcionais. A mesma que foi validada pela elaboração de um protótipo funcional. Os resultados da implementação foram documentados visando o cumprimento dos objetivos da pesquisa.

4. CONSIDERAÇÕES FINAIS

4.1 Conclusões

Da presente pesquisa aplicada é possível concluir a viabilidade funcional de um sistema de controle de fluxo de pessoas com a capacidade de interagir com os usuários finais, provendo alertas mediante a comunicação GSM, além do fornecimento das análises feitas pelo dispositivo mediante *dashboards*.

Funcionalmente, o sistema proposto e desenvolvido identifica automaticamente a entrada e saída de pessoas num ponto específico, determinando assim o fluxo de pessoas. Os dados são enviados para uma plataforma na nuvem, onde é realizado o processamento final e apresentado em *dashboards* para o usuário final. O dispositivo está dotado de uma inteligência embutida que permite enviar alertas para o usuário quando a capacidade máxima de pessoas dentro do ambiente fechado é atingida.

Do ponto de vista estratégico, o sistema permite conhecer melhor os hábitos das pessoas que frequentam um determinado ambiente fechado, o que é uma vantagem competitiva para a realização de propostas comerciais que precisam de dados de volumetria de clientes.

No contexto em que o presente trabalho foi desenvolvido, a pandemia da Covid-19 gerou mudanças significativas na capacidade dos estabelecimentos comerciais, reduzindo em 40% na fase amarela (Prefeitura SP, 2020). O dispositivo projetado tem a capacidade de controlar o parâmetro de capacidade de pessoas dentro do estabelecimento, adaptando-se dinamicamente conforme a fase da pandemia na cidade.

Considerando que o presente trabalho é fundamentado num protótipo como prova de conceito, a metodologia de desenvolvimento foi de integração de hardware e software, alocando recursos computacionais do processador na aplicação principal, e centralizando a gestão dos periféricos pelo microcontrolador, foi possível reduzir os custos dos módulos de hardware, obtendo-se um protótipo do sistema com um custo inferior a US$ 100.

4.2 Trabalhos Futuros

A aplicação de conceitos de visão computacional com sistemas embarcados, computação na nuvem e inteligência de negócios proporcionam aplicações com um grande potencial. Trabalhos futuros podem ser desenvolvidos nas linhas de hardware e software em continuidade ao presente trabalho.

Existem dois protocolos de comunicação utilizados nesta pesquisa que não são padrão do Linux: o I^2C e o SPI, e dependem exclusivamente do módulo com microprocessador, neste caso o Raspberry Pi B3. A integração exclusiva de todos os periféricos no microcontrolador com conexão serial para a placa pode deixar a arquitetura mais dinâmica, com a possibilidade de ser usada por diversas plataformas microprocessadas com o Linux como sistema operacional.

O modelo de dados proposto é relacional, o que permite uma rápida injeção de dados pelo dispositivo em operação. Porém, em termos de escalabilidade pode ser pouco eficiente, com um alto volume de dispositivos e leituras em simultâneo. O estudo de aplicação de uma abordagem NOSQL (banco de dados não relacional) poderia melhorar o desempenho do sistema.

O sistema operacional que foi utilizado neste trabalho é o Raspbian, uma distribuição customizada do Debian para o Raspberry Pi. Porém, por ser uma versão oficial, contém múltiplos módulos desnecessários para a aplicação, como o de áudio e o edição de textos. A geração de um *filesystem* específico e mais otimizado pode aumentar a capacidade do módulo e disponibilidade do poder computacional.

No presente trabalho não foi considerada a parte mecânica, como a caixa para proteção do hardware. O desenvolvimento de uma caixa considerando uma ventilação passiva pode garantir o melhor desempenho do sistema. A inclusão de servomotores para a movimentação da câmera pode potencializar as funcionalidades do sistema, como a calibração automática ou seguimento de objetivos.

REFERÊNCIAS

ABRASCE. **Números do setor**. Abrasce, 2019. Disponível em:<https://abrasce.com.br/numeros/setor/>. Acesso em: 22 Mai. de 2020.

ALMEIDA, M. Uma introdução ao XML, sua utilização na internet e alguns conceitos complementares. **Ci. inf.** vol.31 no.2 Brasília May/Aug. 2002.

BARRETO, M.L. et al. O que é urgente e necessário para subsidiar as políticas de enfrentamento da pandemia de COVID-19 no Brasil?. **Revista Brasileira de Epidemiologia.** Rio de Janeiro, vol.23, Apr 22, 2020.

CAHN, J. **CHATBOT: Architecture, Design, & Development**. 2017. 46. (Senior Thesis Department of Computer and Information Science) – University of Pennsylvania, Pennsylvania, 2017.

FEW, S. **Information Dashboard Design The Effective Visual Communication Of Data**. Usa: O´Reilly, 2006.

FLOYD, T. **Fundamentos de Sistemas Digitales**. 9ed. MADRID: Pearson, 2006.

FURBER, S. **ARM System-on-Chip Architecture**. 2ed. UK: Pearson, 2000.

HEATH, S. **Embedded Systems Design**. 2ed. UK: Newnes, 2003.

LEMOS P.; ALMEIDA-FILHO, N.. **COVID-19, desastre do sistema de saúde no presente e tragédia da economia em um futuro bem próximo**. Brazilian Journal of Implantology and Health Sciences (BJIHS), v.2, n.4, p. 39-50, April 29, 2020.

LIGANG. **SIM900A Hardware Design**. Shanghai: Shanghai, SIMCom Wireless Solutions, december. 2009. 57p.

MALAKHOV, Composable Multi-Threading for Python Libraries. **Scipy**.Austin, July.2016.

ROSEBROCK, A. **Deep Learning For Computer Vision With Python**. 1ed. PENSILVANIA: PyImageSearch, 2017.

SOLEM, E. **Computer Vision with Python**. USA: O´Reilly, 2012.

SOO, S. **Object detection using Haar-cascade Classifier**. Institute of Computer Science, University of Tartu, 2014. [online], semanticscholar.org, 2013. Disponível em:

(https://www.semanticscholar.org/paper/Object-detection-using-Haar-cascade-Classifier-Soo/0f1e866c3acb8a10f96b432e86f8a61be5eb6799). Acesso em: (21/10/2020).

STUBER, G. **Principles of Mobile Communication**. Switzerland: Springer, Cham, 2017.

TEXAS INSTRUMENTS. **Tiva TC123GH6PM Microcontroller**. Austin: Texas Instrument Incorporated, June. 2014. 1404p.

WANG, Q. et al. Fast Online Object Tracking and Segmentation: A Unifying Approach. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). **Proceedings ...**, 2019 CVPR, 2019.

WARD, B. **How Linux Works**. 3ed. USA: No Starch Press, 2021.

WU,Y; LIM, J.; YANG, M.H. Online Object Tracking: A Benchmark. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, **Proceedings ...**, CVPR, 2013, pp. 2411-2418, 2013.

YOCTOPROJECT. **Getting Started: The Yocto Project Overview**. Yoctoproject, 2020. Disponível em :<https://www.yoctoproject.org/software-overview/>. Acesso em: 12 Mai. de 2020.

ZHANG, C.; Zhengyou, Z. **A survey of recent Advances in Face Detection**. researchgate.net, 2010. Technical Report, MSR-TR-2010-66, Microsoft Research, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, June 2010. Disponível em: (https://www.researchgate.net/publication/235616690). Acesso em: (21/10/2020).

ZUCCO, L. et al. Consideraciones perioperatorias para el nuevo coronavirus 2019 (COVID-19). **Anesthesia Patient Safety Foundation APSF**. marzo 26, 2020..

PREFEITURA SP. **Cidade de São Paula na Fase Amarela**. Prefeitura 2020. Disponível em:<https://www.prefeitura.sp.gov.br/cidade/secretarias/saude/> Acesso em: 03 Mar. de 2021.

ARM. **ARM Cortex a-53.** ARM 2021. Disponível em :<https://developer.arm.com/ip-products/processors/cortex-a/cortex-a53> Acesso em: 03 Mar. de 2021.

RASPBERRYPI. **Bradcom BCM2837**. RASPBERRYPI 2021. Disponível em :<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md>Acesso em: 03 Mar. de 2021.