**UNIVERSIDADE FEDERAL DE OURO PRETO**

**ESCOLA DE MINAS**

**COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU**

**Rafael Pinati Nascimento**

# Estudo e desenvolvimento de interface gráfica aliada às comunicações de dados entre dispositivos

**MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

**Ouro Preto, 2019**

Rafael Pinati Nascimento

# Estudo e desenvolvimento de interface gráfica aliada às comunicações de dados entre dispositivos

Monografia apresentada ao Curso Engenharia de Controle e Automação, como parte dos requisitos para a obtenção de título de graduação em Engenharia de Controle e Automação.

Orientador: Prof. Ms. Diógenes Viegas Mendes Ferreira

Ouro Preto

Escola de Minas – UFOP

Junho/2019

Monografia intitulada "Estudo e desenvolvimento de interface gráfica aliada às comunicações de dados entre dispositivos" defendida e aprovada, em 3 de julho de 2019, pela comissão avaliadora constituída pelos professores:

Prof. M. Sc. Diógenes Viegas Mendes Ferreira - Orientador

Prof. Dr. Paulo Marcos de Barros Monteiro – Professor Convidado

André Almeida Santos

Prof. André Almeida Santos – Professor Substituto Convidado

**AGRADECIMENTOS**

Agradeço primeiramente a Deus pela força e orientação ao longo desta caminhada. Aos meus pais por me amarem incondicionalmente e por estarem sempre presentes. À minha irmã, Raissa, que mesmo na distância me apoiou e meus amigos de faculdade que acabaram por se tornarem minha segunda família. Agradeço a universidade pelo ensino de qualidade e a todos os professores pela paciência e conhecimentos, a todos meu muito obrigado.

*“Na vida, não vale tanto o que temos, nem tanto importa o que somos. Vale o que realizamos com aquilo que possuímos e, acima de tudo, importa o que fazemos de nós.”* Chico Xavier

## Resumo

Internet das Coisas é o modo como os objetos físicos estão conectados e se comunicando entre si e com o usuário, através de sensores inteligentes e softwares que transmitem dados para uma rede. Este conceito está impulsionando a inovação de negócios digitais e reformulando experiências com interfaces gráficas. Neste novo mundo conectado profissionais que dominarem as tendências inovadoras de IoT terão a oportunidade de liderar a inovação digital. Para estar à frente, junto a outros nesta liderança, é importante entender o conceito de interfaces, pois elas que criam as interações com os usuários e mais do que isso, desenvolver uma aplicação, que interaja também com outros dispositivos, permite entender as dificuldades e enxergar as possibilidades da aplicação, sempre visando maior qualidade e menor custo, para estar apto a este tipo de serviço que será de suma importância para a revolução IOT.



Palavras-chaves: Internet das coisas, Interfaces gráficas, Comunicação de dados, Inovação digital.

## Abstract

Internet of Things is the way physical objects are connected and communicating with each other and with the user through intelligent sensors and software that transmit data to a network. This concept is driving digital business innovation and reshaping experiences with graphical interfaces. In this new connected world professionals who master the innovative trends of IoT will have the opportunity to lead the digital innovation. To be ahead, along with others, in this leadership, it is important to understand the concept of interfaces, since they create interactions with users and more, developing an application, which also interacts with other devices, allows us to understand the difficulties and see the possibilities of this application, always aiming at higher quality and lower cost, to be able to this type of service that will be of very important to the revolution IoT.

**Keywords:** Internet of Things, Graphic Interfaces, Data Communication, Digital Innovation.

## Lista de abreviaturas e siglas

| | |
|---|---|
| Bit | Binary digit (Dígito binário) |
| Bps | Bits por segundo |
| DMI | Direct Media Interface (Interface direta de mídia) |
| USB | Universal Serial Bus (Porta serial universal) |
| GPIO | General Purpose Input/Output (Portas de entrada/sáida gerais) |
| EIA | Electronic Industries Alliance (Aliança das industrias eletrônicas) |
| TIA | Telecommunications Industries Association (Associação das Indústrias de Telecomunicações) |
| ETA | Eletronics Industries Association (Associação das Indústrias de Eletrônica) |
| GUI | Graphical User Interface (Interface gráfica do usuário) |
| IOS | Sistema operacional IOS 7ª geração |
| IoT | Internet of things (Internet das coisas) |
| ISO | Organização Internacional de Normalização |
| LAN | Local Area Network (Rede local) |
| LED | Light Emitting Diode (Diodo Emissor de Luz) |
| OpenGL | Open Graphics Library (Biblioteca gráfica livre) |
| RS232 | Recommend Standard – 232 (padrão recomendado 232) |
| RS485 | Recommend Standard – 485 (padrão recomendado 485) |
| TCP | Transmission Control Protocol (Protocolo de Controle de Transmissão) |
| IP | Internet Protocol (Protocolo de Internet) |
| SPI | Serial Peripheral Interface (Interface Periférica Serial) |
| WAN | Wide Area Network (Rede de longa distância) |

# Lista de tabelas

# Lista de figuras

# Sumário

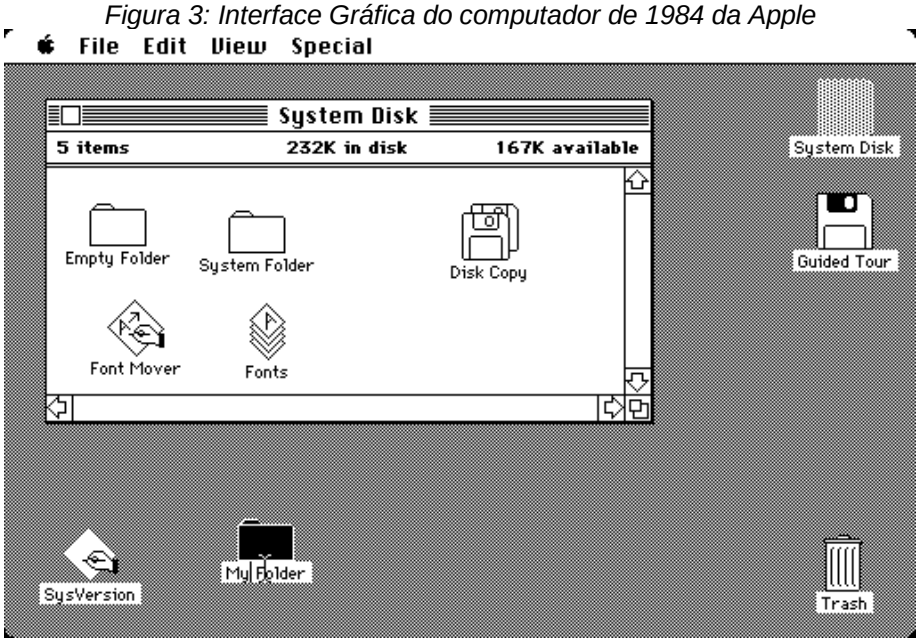
## 1. Introdução:

"O território que vemos através da janela ampliada em nosso novo veículo não é a paisagem habitual de planícies, árvores e oceanos, mas uma paisagem de informação cujos marcos são palavras, números, gráficos, imagens, conceitos, parágrafos, raciocínios, fórmulas, diagramas, provas, corpos de literatura e escolas de crítica" (HOWARD RHEINGOLD,1985 apud STEVEN JOHNSON, 1997).

A citação de Howard refere-se à integração do mouse as telas de computador. Nos primórdios da era digital, o mouse fazia com que o usuário entrasse naquele espaço de dados, operando o ponteiro na tela, e assim permitindo criar janelas, abrir e fechar ferramentas, reorganizar o espaço-informação, entre outras coisas. Nascia aí a interface homem máquina.

A GUI, *Graphical User Interface* (Interface gráfica do usuário), começou a ganhar aspectos mais modernos, em 1974, com o surgimento da "Smalltalk*".* A grande inovação foram as janelas, que possuíam bordas e barras de títulos que permitiam a identificação e o reposicionamento delas, juntamente com o conceito de ícones e menu de tarefas. Já em 1984, com a chegada da Apple, um passo importante na história das interfaces gráficas foi dado. A equipe de desenvolvedores trabalhou em uma interface baseada em ícones, em que cada um deles indicava um documento ou uma aplicação. Além disso, a equipe criou a primeira barra de menu desdobrável (*pull-down*), que hospedava todos os menus logo nas primeiras linhas da tela.

*Figura 3: Interface Gráfica do computador de 1984 da Apple*




*Fonte: LOURENÇO, L. (2011)*

Atualmente quem dita as regras de design das interfaces são as grandes empresas do ramo. O layout do Windows 8 revolucionou as interfaces gráficas e trouxe um novo conceito, conhecido como “flat design”. Este conceito traz para o usuário cores vibrantes, ícones simples e elementares, aumento da opacidade e suavização dos objetos. O resultado é leveza e simplicidade para interface fazendo que tudo fique mais intuitivo. Este conceito foi melhorado com o lançamento do iOS7, sistema operacional 7ª geração dos aparelhos da Apple; e continua sempre se atualizando.

Figura 4:Interface gráfica Windows 8


Fonte: LOURENÇO, W. (2012)

Segundo dicionário Aurélio, interface é um dispositivo (material e lógico) graças ao qual se efetuam as trocas de informações entre dois sistemas. É a parte “visível” do projeto ou sistema em qual o usuário interage, podendo ser entendida como um meio para um diálogo entre homem e máquina através de ícones, menus e janelas em alternativa às complexas linhas de comando.

Embora fáceis de usar, aplicativos com interfaces gráficas são difíceis de desenvolver, pois envolve uma linguagem de programação mais avançada e difícil manutenção do código. Portanto deve ser desenvolvida por projetistas mais experientes e com uma estrutura de desenvolvimento de alta produtividade que facilite a visão do código para futuras manutenções e atualizações.

Johnson (p.28, 2001) reforça a importância do design de interface:

> A importância do *design* de interface gira em torno deste aparente paradoxo: vivemos numa sociedade cada vez mais moldada por eventos que se produzem no ciberespaço, e apesar disso o ciberespaço continua, para todos os propósitos, invisível, fora de nossa apreensão perceptiva. Nosso único acesso a esse universo paralelo de zeros e uns se dá através do conduto da interface do computador, o que significa que a região mais dinâmica e mais inovadora do mundo contemporâneo só se revela para nós através dos intermediários anônimos do design de interface.

Importante deixar claro que, para um desenvolvedor, a interface deve ir muito além da estética do projeto, é como será realizada a interação do usuário com o sistema. Segundo Oliveira (2004) uma interface não só deve criar as estratégias de uso como deve orientar, interagir e auxiliar o usuário durante as interações. É necessário antecipar as necessidades do usuário e garantir que a interface seja de fácil acesso. Quanto mais portas forem evitadas e as dificuldades minimizadas maior será a experiência de interação do usuário.

A norma ISO 9421-11 (norma da Organização Internacional de Normalização) define usabilidade como critério para sistemas interativos, levando em conta a efetividade, eficiência e satisfação dos usuários para com os sistemas (GARBIN, 2010). Com isso o padrão ISO diz que a usabilidade deve ser mais que os usuários saberem usar e sim se eles podem executar a tarefa com facilidade. A usabilidade é importante porque, se um usuário não consegue alcançar seus objetivos de forma eficaz, eles buscarão outra solução, já que há tantas opções disponíveis no mercado. Portanto usabilidade se refere à satisfação do usuário.

Atualmente existem inúmeras ferramentas para desenvolvimento de aplicações gráficas, cada qual com suas particularidades. A escolha da ferramenta para projeto é de extrema importância, não só na fase de desenvolvimento, como também nas fases de manutenção e atualização do código, uma vez que algumas ferramentas permitem o desenvolvimento de código fonte portátil, isto é, que pode ser executado em várias plataformas, como Windows, Linux, mobile, entre outros sistemas operacionais, sem a necessidade de modificações no código.

Com o decorrer dos anos a interface gráfica evoluiu atendendo às necessidades de cada período. Atualmente, é usada em diversas áreas, principalmente aliada a sistemas embarcados. Enquanto um computador usual serve aos mais diversos propósitos: checar seu *e-mail*, usar a *internet*, criar e editar textos; um sistema embarcado é especialmente desenvolvido para concretizar pequenas

ações da forma mais eficiente possível, independente ou parcialmente independente de intervenção humana.

Micro controladores estão cada dia mais populares e acessíveis devido ao seu custo e poder de processamento. Essa popularidade está conectando não só os computadores como também pessoas, objetos e qualquer tarefa que possa ser realizada automaticamente. Suas propriedades e funcionalidades são peça chave para que outros dispositivos comuniquem com interfaces gráficas. São dispositivos que aliados a interfaces gráficas permitem criar uma comunicação entre homem e outros dispositivos, com alta usabilidade e com custo baixo.

Estes dispositivos comunicam–se por interfaces de comunicação, via portas seriais, protocolos RS232 e RS485, comunicações sem fios, protocolo TCP/IP (Protocolo de Controle de Transmissão) e protocolos SPI (Serial Peripheral Interface), entre outras. O envio e recebimento desses dados é feito nos periféricos dos micro controladores através de módulos eletrônicos que possuem a capacidade de converter sinais analógicos em sinais digitais e de um protocolo para outro (sinal digital para outro protocolo de sinal digital).

## 1.1 Justificativa

*Internet of Things* (IoT) é parte integrante da Internet do Futuro e pode ser definida como uma infraestrutura de rede global dinâmica com recursos de autoconfiguração baseados em protocolos de comunicação padrão e interoperáveis onde os recursos físicos e virtuais têm identidades, atributos físicos e personalidades virtuais, usam interfaces inteligentes e são perfeitamente integradas na rede de informação (Traduzido e Editado CERP IoT, 2009)

Não é nenhuma novidade que estamos vivendo em uma era digital em que soluções tecnológicas surgem e ganham mais espaço a cada dia. Neste segmento tecnológico que as interfaces gráficas se encaixam. Os usuários para se comunicarem com esses dispositivos precisam de uma interface gráfica e esta, para acessar os dados, necessitam dos protocolos de comunicação.

Esse trabalho busca adquirir experiência em interfaces gráficas e comunicação com outros dispositivos, pois essa é uma aplicação estimulante, uma vez que trabalha com diversas áreas do curso de forma interativa, demanda lógica e criatividade, além de que, no ponto de vista empresarial, é um ramo que está em processo de

crescimento. Portanto, será desenvolvida uma aplicação nesta área para poder enxergar as vantagens, compreender as dificuldades e superar barreiras do desenvolvimento deste tipo de aplicação.

## 1.2 Objetivo:

Estudar os conceitos de interfaces gráficas e comunicações de dados para desenvolver uma aplicação, de maneira coerente e bem articulada. Será desenvolvido uma interface de um autoatendimento de uma lanchonete enquanto a mesma realiza comunicação com uma máquina de cartão para o pagamento e um servidor para recebimento do pedido em uma suposta cozinha, sempre buscando ferramentas que possam agregar no desenvolvimento.

## 1.3 Objetivos específicos:

- Coletar informações sobre softwares e bibliotecas utilizadas no mercado para desenvolvimento destas aplicações;
- Comparar as vantagens e desvantagens das ferramentas;
- Projetar uma interface gráfica para um autoatendimento de uma suposta lanchonete utilizando a linguagem de programação python.
- Realizar a comunicação entre interface e periféricos para transferência de dados utilizando o Raspberry Pi.

# 2 Revisão bibliográfica

## 2.1 Raspberry Pi:

Microcontroladores são circuitos integrados desenvolvidos para controle de processos lógicos. São dotados memória interna, entradas e saídas digitais e analógicas, *timers*, contadores, periféricos para comunicação serial e sem fio, geradores de pulsos, entre outros (NETO, MONTEIRO, QUEIROGA, 2012).

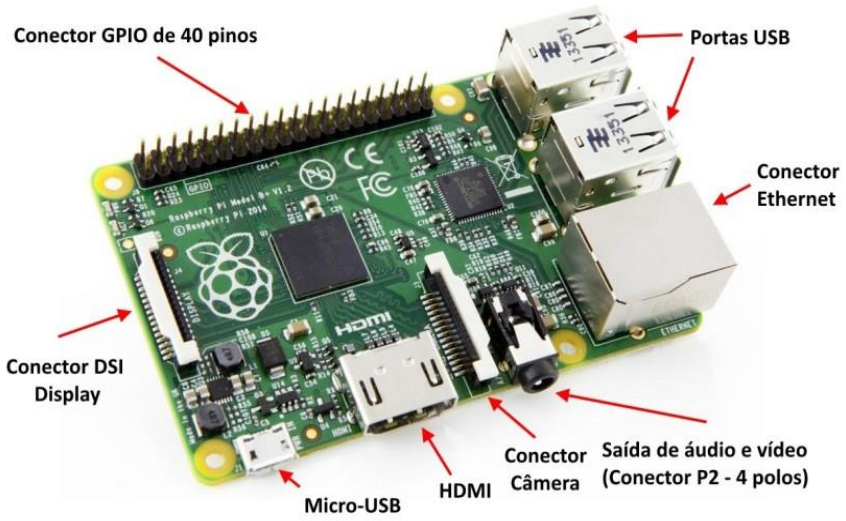
Logo se tornou muito popular devido ao seu alto poder de processamento, tamanho reduzido e custo baixo. Popularidade essa que só vem aumentando com suas atualizações.

Sua aplicabilidade está diretamente ligada às instruções nela inseridas e os dispositivos nela comunicados. Diversos tipos de microcontroladores estão disponíveis no mercado, os quais os mesmos diferem no poder de processamento, disponibilidade de portas e número de linhas de entradas e saídas, propriedades estas que devem ser levadas em consideração no desenvolvimento de um projeto (NETO, MONTEIRO, QUEIROGA, 2012).

Um dos mais famosos, atualmente, é o Raspberry Pi. Raspberry PI é uma mistura de computador com um micro controlador de baixo custo. É uma placa pequena, porém muito poderosa e foi idealizada e desenvolvida no Reino Unido em 2006 pela Fundação Raspberry Pi inicialmente localizada na Universidade de Cambridge (EBERMAN, PESENTE, RIOS, PULINI, 2017).

A placa está equipada com conectores GPIO, portas USB, conector Ethernet, conector DSI Display, conector para câmera, entrada micro-USB, saída HDMI e de áudio e vídeo, responsáveis pelas comunicações de dados, conforme mostrado na Figura 3.

Figura 3: Portas do Raspberry Pi3




Fonte: ARDUINO E CIA (2014)

Upton e Halfacree (2013) detalham essas conexões:

- Conector GPIO de 40 pinos: Conjunto de 40 pinos responsáveis por enviar e receber sinais digitais. São utilizados para incorporar componentes eletrônicos ao sistema, como a utilização de botões, relés, LED (*Light Emitting Diode*), entre outros.
- Portas USB: São portas usadas para conectar dispositivos periféricos, como teclado, mouse e monitor sem há necessidade de desligar ou reiniciar o sistema.
- Conector Ethernet: Mesmo a maioria de suas placas possuir um chip para conexões Wi-Fi, está embutida nela uma porta para conexões de rede via cabo.
- Pino A/V: Saída utilizada para transmissões de áudio e vídeo.
- Conector Câmera: Entrada do modulo de câmera da Raspberry Pi.
- HDMI: Saída de áudio e vídeo digital de monitores e televisões com as características HDMI.
- Micro USB: Na versão Raspberry Pi 3, o micro USB é utilizado para alimentação da placa.

- Conector DSI Display: Também conhecido como *display serial interface,* é utilizado para pequenas telas de LCD ou LED. Recentemente a *Raspberry Foundation* liberou um módulo display próprio da marca.

Eberman, Pesente, Rios e Pulini (2017) citam e descrevem algumas linguagens que acompanham o Raspberry: Scratch, Pygames e Python. Scratch é uma linguagem inicial para aprendizado da lógica de programação, desenvolvida para que crianças se familiarizem com o conceito desde novos. Pygames é uma biblioteca em Python, desenvolvida para facilitar a criação de jogos. E Python é uma linguagem mais elaborada, utilizada para fins da robótica, conjunto de computadores que processam mesma informação, e outras aplicações, dependendo da biblioteca utilizada. Porém o Raspberry não se limita a essas linguagens, C, Ruby, Java e Perl, são exemplos de outras linguagens de programação famosas que também são trabalhadas com o Raspberry.

## 2.2 Linguagem Python

Segundo Borges (2014) a linguagem Python foi desenvolvida pelo matemático e programador Guido Van Rossum no instituto Nacional de Pesquisa para Matemática e Ciência da Computação da Holanda a partir de uma linguagem existente, a linguagem ABC.

Com decorrer dos anos a linguagem não parou de receber melhorias e diversas versões foram criadas a partir daí. O Python 3.0 ou Python 3000 é o seguimento mais usados hoje em dia. Foi lançado em dezembro de 2008, também chamada Python3. O próprio idealizador da linguagem, Guido Van Rossum (2007), em uma entrevista ressaltou a quebra de compatibilidade com a família 2.x para corrigir falhas que foram descobertas neste padrão, e para limpar os excessos das versões anteriores.

A linguagem tinha como foco engenheiros e físicos, mas atualmente é muito utilizada por diversas empresas como um dos componentes padrões de seus sistemas, como por exemplo Google, Yahoo, Microsoft, NASA, Disney, e muitas outras (BORGES, 2014). Diversas faculdades também a adotaram como linguagem padrão em cursos de computação e engenharia.

O Python possui uma sintaxe clara e concisa, que favorece a legibilidade do código fonte, tornando a linguagem mais produtiva. A linguagem possui uma vasta coleção de módulos prontos para uso, além de outros frameworks desenvolvidos por terceiros, pois a mesma é *Open Source*, ou seja, é uma linguagem com código aberto

onde todo programador pode fazer atualizações e liberar para a comunidade. A mesma também inclui diversas estruturas de alto nível, como listas, dicionários, threads, entre outras.

As Threads é uma destas estruturas essenciais para nossa aplicação. As Threads são linhas de execução que compartilham um espaço de memória com outras execuções (BORGES, 2014). É uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente. Isso permite que uma tarefa não seja interrompida enquanto outra está sendo executada.

Além de ser utilizado como linguagem principal no desenvolvimento de sistemas, o Python também é muito utilizado como linguagem script em vários softwares, permitindo automatizar tarefas e adicionar novas funcionalidades, entre eles: BrOffice.org, PostgreSQL, Blender, GIMP e Inkscape (BORGES, 2014).

Borges (2014) também cita e descreve as principais características da linguagem Python, sendo ela, uma linguagem de altíssimo nível, orientada a objeto, de tipagem dinâmica e forte, interpretada e interativa.

Tipagem dinâmica significa que o próprio interpretador do Python infere o tipo dos dados que uma variável recebe, sem a necessidade que o usuário da linguagem diga de que tipo determinada variável é. Forte porque o interpretador do Python avalia as expressões e não faz coerções automáticas, resultando em erro, assim você terá a certeza que o seu resultado é mais consistente.

É interativa porque as linhas de código podem ser digitadas em um prompt (linha de comando) semelhante ao *shell* do sistema operacional. E, por fim, interpretada porque têm seus códigos fontes transformados em uma linguagem intermediária (específica de cada linguagem), que será interpretada pela máquina virtual da linguagem quando o programa for executado.

Uma linguagem quando é dita interpretada significa que ela é multiplataforma e, por sua vez, pode ser distribuída e executada sem fonte original. Isso significa que um mesmo programa pode ser compilado em diversos sistemas operacionais como Windows, Linux e Apple.

**2.3 Biblioteca Kivy:**

A portabilidade é uma palavra-chave para desenvolvimento de softwares, a mesma se encontra equacionada nas áreas de linguagens de programação, leitura e escrita de dados e comunicações padronizadas como ANSI C, SQL, TCP, ISO, entre outras (KIVY, 2016).

No caso de interfaces gráficas, o conceito de portabilidade é mais restrito, pois a maioria das ferramentas para tal possui um conjunto de recursos básicos para sua implementação, conhecidos como *toolkit*, todas desenvolvidas de forma independente. A falta de padronização implica na necessidade de reprogramar os módulos da interface quando esta necessitar transferir de um ambiente para outro. Essa reprogramação pode ser extremamente cara pelo fato de que a maior parte do código se trata de sua interface gráfica.

Kivy é uma ampla biblioteca open-source do Python, o que significa que todo mundo pode oferecer melhorias e atualizações no código. Ela foi criada para desenvolvimento gráfico e pode ser utilizada no desenvolvimento de utilitários e jogos. Aplicações desenvolvidas com essa biblioteca aceitam até cinco toques simultâneos, o que a torna interessante para o desenvolvimento de jogos e podem ser executados em diversas plataformas como: Linux, Windows, OS X, Android e iOS (ULLOA, 2013).

Uma aplicação desenvolvida pela biblioteca Kivy será executada diretamente na OpenGL, *Open Graphics Library* (Biblioteca gráfica livre), que por sua vez é executada na placa gráfica, assim toda plataforma que oferece suporte a OpenGL poderá executar o código (ULLOA, 2013). Em outras palavras, todo o código será executado em cima da placa de vídeo, e é por essa razão, que em muitos casos, as aplicações Kivy têm performance igual, ou então superior a aplicações da própria plataforma.

A OpenGL está presente na maioria das plataformas, isso significa que ela é multiplataforma e como o Kivy é construída em cima dessa biblioteca gráfica esta pode ser executada em qualquer dispositivo que oferece suporte a mesma.

Abreviação de Kivy language, KV é uma linguagem de marcação do tipo QML, XML, JSON para construção de interfaces gráficas (KIVY, 2016). As linguagens de marcação permitem criar documentos com uma estrutura de representação que seja compreendida por diversos sistemas de software, independe da máquina onde estão sendo executados. Isto significa que um mesmo documento pode ser manipulado em um computador pessoal ou mesmo um de grande porte sem que haja dependência

de um determinado sistema operacional, tornando assim uma linguagem independente do sistema ou software onde são visualizados.

Seu objetivo é separar o código de interface visual do código da lógica de negócio, ou seja, para separar nosso código python do nosso código que é utilizado para construção da parte visual. A própria linguagem já fornece uma primeira camada de separação do código da tela, que interage com o usuário, do código python que implementa as regras de negócio. A linguagem Kivy seria o V do padrão MVC (*Model-View-Controler*) até porque com a biblioteca nós implementamos a parte visual e é possível criar interface instanciando as classes que desejamos e adicionando ao layout.

Também chamada apenas de KV, a biblioteca é bastante poderosa pois nos permite criar as mais variadas interfaces utilizando uma linguagem simplista e bastante funcional. A mesma foi projetada para construção de janelas gráficas, da mesma forma que o python foi projetado para implementação das regras de negócio, ou seja, se determinado campo está correto, se determinado arquivo existe, se há conexão com internet, transações com banco de dados e etc. Podemos dizer então que o Python é de uso geral, pode até mesmo ser usado para construção da parte gráfica, mas principalmente as regras de negócio, enquanto a biblioteca Kivy é focada num único objetivo: criar a interface de interação com o usuário.

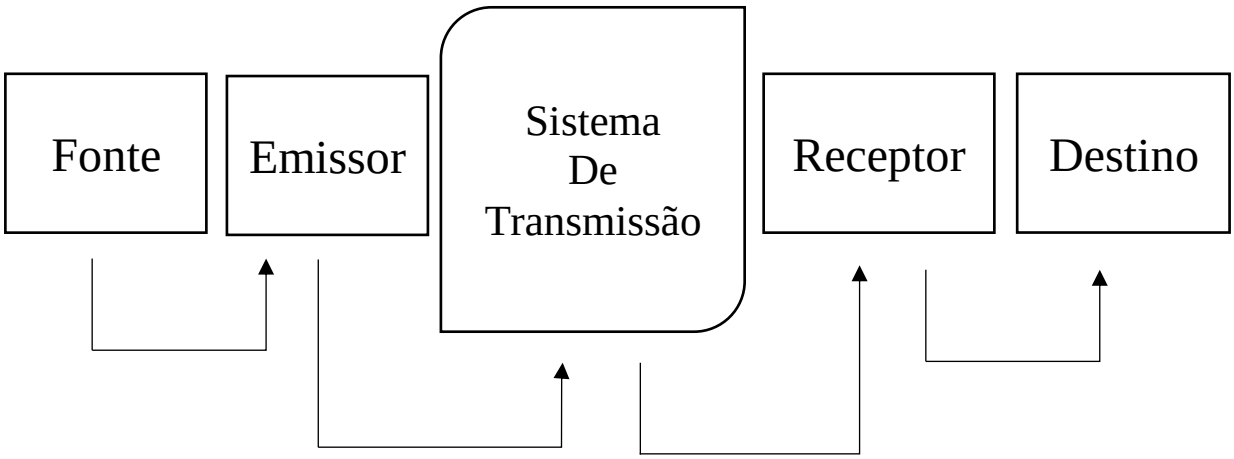
O desenvolvimento de interfaces gráficas é um trabalho que exige alto nível de programação e orientação a objetos, portanto alguns conceitos como classes, objetos, instâncias e widgets se fazem necessários. Uma classe descreve as variáveis, as propriedades, os procedimentos e os eventos de um objeto. Ao instanciar uma classe, ou seja, ao definir o conjunto de atributos e métodos dela, criasse um objeto que nada mais é que um exemplo ou um caso daquela classe. Os widgets nada mais são que componentes gráficos da biblioteca kivy que herdam suas propriedades de alguma classe pré-definida, como a classe Button, por exemplo. Assim podemos definir que um widget é uma instancia de uma classe, ou seja, um objeto.

Quando a linguagem kivy é executada o que acontece de fato é instanciação de várias classes como se estivesse executando um código python de forma normal, como resposta a execução do código kivy será um conjunto de novas classes ou novas instancias, isto porque muitos dos componentes são criados de duas maneiras: Uma parte com a linguagem kivy e outra parte com o Python.

**2.4 Comunicação de Dados:**

Comunicação de dados é a troca de informações entre dois ou mais dispositivos e um meio de transmissão, como a comunicação USB mais comumente usada hoje em dia. Esta troca de informações parte de um indivíduo que cria esse dado e vai para aquele que processa ou exibe este dado. Para que as comunicações de dados ocorram, os dispositivos de comunicação devem fazer parte de um sistema de comunicações, formado por 5 partes, como é mostrado na figura abaixo:

Figura 4: Estrutura de comunicação de dados




Fonte: Do próprio autor.

A fonte gera os dados que são convertidos, pelo emissor, em sinais adequados ao sistema de transmissão que transporta a informação sob forma de sinais até o receptor. O receptor transforma os sinais em dados novamente para que o destino processe ou exiba estes dados. Para que a transmissão seja correta a mesma deve seguir um protocolo (FOROUZAN, 2010).

O protocolo é quem dita as regras para que a comunicação de dados seja realizada da maneira correta, é como se fosse o meio no qual os dispositivos se comunicam (FOROUZAN, 2010). Sem o protocolo, mesmo que os dispositivos estejam conectados a transmissão de dados não ocorrerá de forma correta. É como se pessoas de nacionalidades diferentes tentassem se comunicar, elas iriam ouvir, porém não se entenderiam. A fonte e o destino podem enviar e receber estes dados de diversas formas, tais como por texto, números, imagens, áudio e vídeo.

O texto é representado por padrões de bits, ou seja, uma sequência de zeros ou um, que é interpretado como caracteres seja ela letra, dígito ou sinal de pontuação

(FOROUZAN, 2010). Essa interpretação tem por base a tabela ASCII, uma tabela arbitrária mas universalmente aceita que associa um caractere a cada byte (conjunto de 8 bits).

As imagens também são um conjunto de bits. Ela é composta por uma matriz de pixels, em que cada pixel é um pequeno ponto. A resolução das imagens depende da quantidade pixels que a imagem é dividida e quanto maior sua quantidade maior é a resolução da imagem (FOROUZAN, 2010).

Forouzan (2010) ainda nos diz que o áudio é, por natureza, diferente de texto, números ou imagens, ele é contínuo, não discreto, ou seja o sinal é analisado continuamente, enquanto o vídeo se refere ao registro ou à transmissão de uma imagem ou filme, que pode ser de natureza discreta ou continua, por exemplo em uma combinação de imagens é de característica discreta e são dispostas em série para transmitir a ideia de movimento.

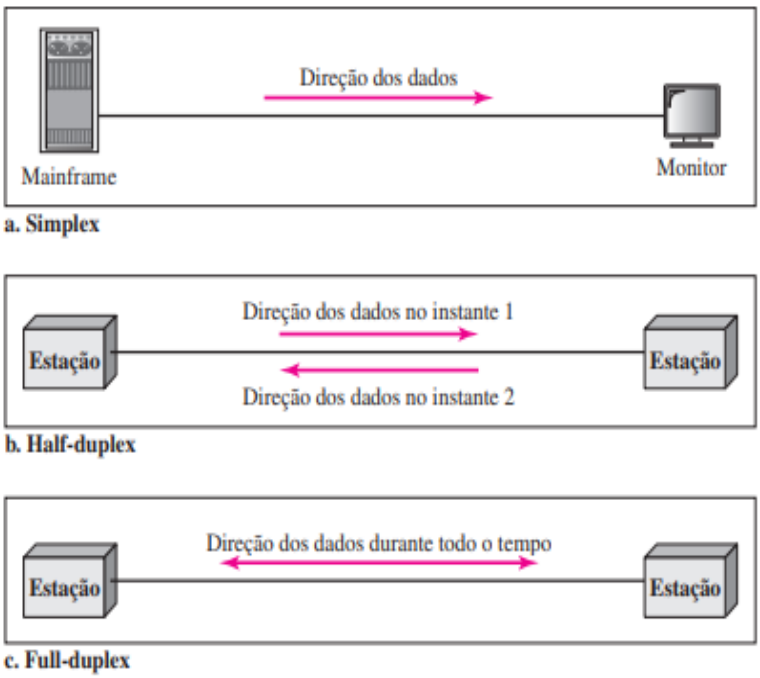
A eficácia de um sistema de comunicações de dados depende de quatro características fundamentais: entrega, precisão, sincronização e *jitter*. Forouzan (2010) defini essas características como:

- Entrega: O sistema deve receber o dado no destino certo e apenas os destinatários devem receber a informação, caso este falhe o sistema não termina de executar a tarefa ou segue por um caminho não desejado.
- Precisão: O sistema deve receber com precisão o mesmo dado que foi enviado. Porém perturbações, como as de ambiente, são frequentes e estas podem gerar sinais distorcidos, levando a erros no sistema ou até mesmo danificando componentes. Para isso é importante que o sistema crie estratégias para fiscalizar a confiabilidade dos dados, como por exemplo checksum, que nada mais é que uma soma dos bytes para conferir a integridade dos arquivos.
- Sincronização: O sistema deve entregar os dados na hora que foi pedido para não haver atrasos na execução. No caso de sistemas com tempos de respostas rápidos, como por exemplo sistema de controle de misseis, a sincronização é impressindível.
- Jitter é uma variação estatística do atraso na entrega de dados em uma rede, ou seja, pode ser definida como a medida de variação do atraso entre os

pacotes sucessivos de dados. Observa-se ainda que uma variação de atraso elevada produz uma recepção não regular dos pacotes.

A comunicação entre dois dispositivos depende dos seus canais, e existem três tipos de canais de comunicação: "*simplex", "half-duplex"* ou *"full-duplex"*, conforme mostrado na Figura 5.

Figura 5: Tipos de canais de comunicação




Fonte: BEHROUZ A. FOROUZAN (2010, pg. 6)

Uma comunicação é dita simplex quando há um transmissor e um receptor, sendo que este papel não se inverte nunca no período de transmissão, ou seja, a comunicação simplex tem sentido unidirecional (FOROUZAN, 2010). Pode-se ter um transmissor para vários receptores, e o receptor não tem a possibilidade de sinalizar se os dados foram recebidos. Como por exemplo, teclados, que só são capazes de introduzir informações e monitores, que só são capazes de receber informações.

Uma comunicação é dita *half duplex* quando temos um Transmissor e um Receptor, sendo que ambos podem transmitir e receber dados, porém nunca simultaneamente, quando um estiver enviando mensagem o outro terá de esperar a transmissão ser concluída, por isso ela tem de sentido bidirecional, mas nunca trafega nos dois sentidos simultaneamente (FOROUZAN, 2010).

Uma comunicação é chamada full duplex (ou simplesmente duplex) quando temos o transmissor e o receptor, podendo os dois transmitir dados simultaneamente em ambos os sentidos (transmissão bidirecional) (FOROUZAN, 2010). Poderíamos entender uma linha full-duplex como equivalente a duas linhas simplex, uma em cada direção. O modo full-duplex é usado quando é necessária a comunicação em ambas as direções durante o tempo todo. Entretanto, a capacidade total do canal tem de ser dividida entre as duas direções.

### 2.4.1 Comunicação RS 232

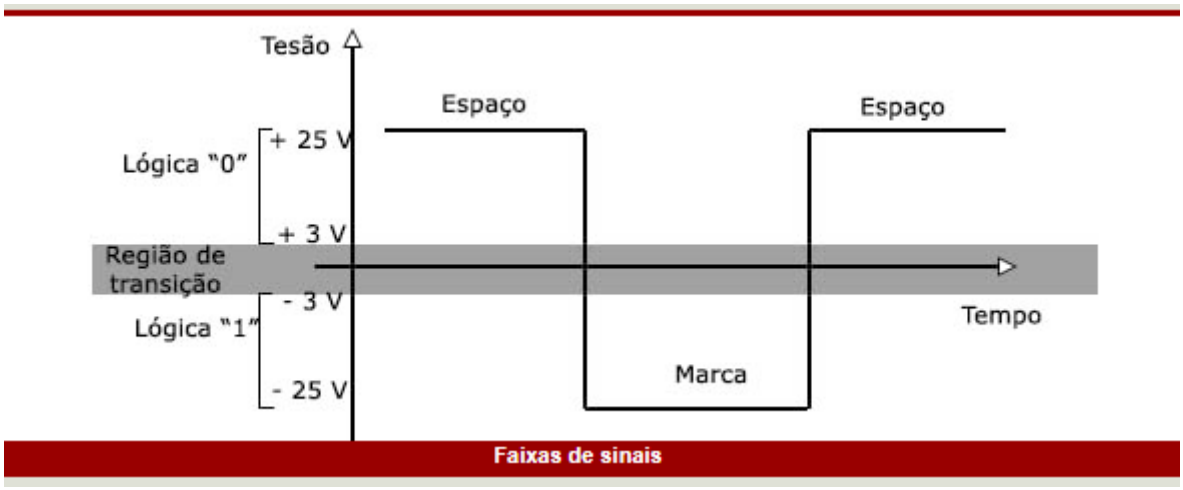
RS232 é uma abreviação de Recommend Standard – 232, ou em português, Padrão Recomendado – 232. É um protocolo de comunicação binária entre um equipamento terminal de dados (DTE), que pode ser um computador, e um equipamento de comunicação de dados (DCE), que é o equipamento que recebe os dados, como um modem, por exemplo. É este protocolo quem dita as regras para os dispositivos conversarem corretamente.

O padrão ETA-232 (Associação das Indústrias de Eletrônica) e EIA/TIA-232 (conjunto de padrões de telecomunicações da Associação das Indústrias de Telecomunicações) foram incluídos no protocolo 232 a fim de padronizar características elétricas e pinagem dos conectores. Atualmente o padrão mais compatível com o RS-232 é o internacional V.24 ITU (*International Telecommunication Union*) (traduzido e adaptado de TEXAS INSTRUMENTS, 2002). A preocupação de tornar o protocolo mais compatível com a norma internacional é a compatibilidade dos produtos do mercado com a porta serial RS-232.

Com o passar dos anos, a porta de comunicação serial RS-232 veio sendo substituída pela porta USB, pois ela é mais rápida, possui conectores mais simples e tem um melhor suporte para os drivers (BARBOSA, 2018). Devido a isso muitos computadores vem sem a porta serial de fábrica, porém essa comunicação ainda é usada em pontos de venda, como leitores de códigos de barras, e na área industrial. Quando o assunto é micro controladores a comunicação ainda é muito usada, devido a sua simplicidade.

No protocolo padrão de comunicação via RS-232, caracteres são enviados um a um como um conjunto de bits. Os sinais enviados estão na forma serial, o que significa que os bits são enviados sequencialmente, um a um, através de uma linha única de comunicações. Eles são representados por níveis de tensão e a comunicação é bilateral, ou seja, “full-duplex”. Segundo a EIA (1996) para interpretação do bit 0 utilizamos uma faixa de tensão entre +3V a +25Volts e para interpretação do bit 1 utilizamos uma faixa de tensão entre -3 a -25Volts, de acordo com a figura 6:

Figura 6: Faixas de sinais RS-232




Fonte: INSTITUTO NEWTON BRAGA (2003)

A lógica negativa garante que maior segurança na transmissão de dados. Note também que a faixa entre -3 V e +3 V consiste numa região “indeterminada” em que não há reconhecimento de nível lógico.

Para que os sinais sejam enviados corretamente, deve escolher os conectores corretos para aplicação, o que dependem da entrada do dispositivo. Os conectores mais famosos são o DB-9 e DB-25:

Figura 7: Conectores RS232 DB-9 e DB-25


Fonte: Cerne e Tecnologia (2006)

Tabela 1: Descrição dos pinos DB9

| Pino | Descrição |
|---|---|
| 1 | DCD |
| 2 | RX |
| 3 | TX |
| 4 | DTR |
| 5 | GND |
| 6 | DSR |
| 7 | RTS |
| 8 | CTS |
| 9 | NC |

Fonte: Cerne e Tecnologia (2006)

Segundo Barbosa (2018), uma comunicação serial mínima deve utilizar os pinos TD (Transmitted Data), RD (Received Data) e o terra, ou Signal Ground (SG) como é representado na tabela 1. As voltagens na serial RS-232 possuem o terra (GND) como referência, uma vez que o RS-232 não está equipotencializado, ou seja, o sistema não está balanceado, a comunicação se torna altamente suscetível a erros. Para evitar esse tipo de erro é importante o correto balanceamento do sistema.

Uma vez balanceado, o software deve ser configurado, e para esta existem diversas configurações que dizem respeito à velocidade de transmissão, também conhecida como Baud Rate, e os bits de paridade e parada. Barbosa (2018) explica essas configurações:

- A velocidade é medida em bps que é a quantidade de bits transmitidas por segundo. As velocidades mais comuns são 300bps, 1200bps, 2400bps, 9600bps, 19200bps, etc.
- Bits de paridade é uma configuração que, quando selecionada, permite detectar erros na transmissão. Normalmente na posição nula, porém pode ser par ou ímpar.
- Bits de parada são enviados no fim de cada mensagem para fins de sincronização. Estes podem ter 1 ou 2 bits de parada.
- Outras configurações especiais são definidas quando pinos enviam sinais de "*handshake*", ou outras checagens de integridade dos dados. Combinações comuns são RTS/CTS, DTR/DSR, ou XON/XOFF, porém não são muito usadas hoje em dia.

### 2.4.2 Comunicação TCP/IP

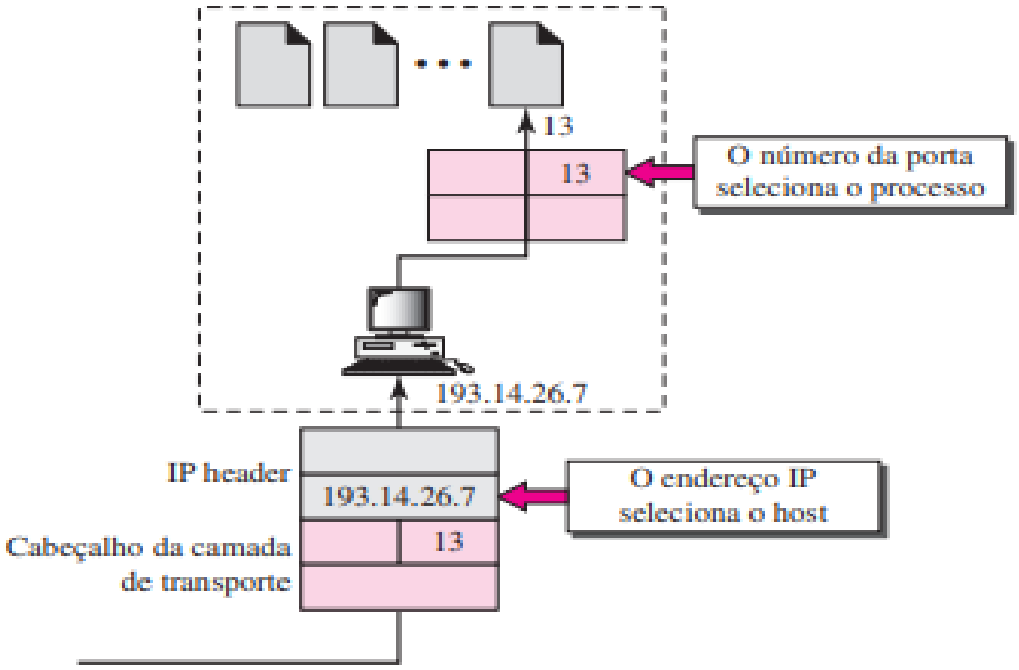
Criada na década de 60 nos EUA, a comunicação TCP/IP surgiu como uma rede acadêmica a fim de interligar os centros computacionais das universidades do país. A proposta foi tão inovadora que os conceitos originais de rede perduram até hoje (FONTINELLE, 2015).

TCP/IP são dois protocolos usados nas redes de computadores e por serem os mais famosos o termo TCP/IP se tornou comum quando se quer referir à família inteira. O protocolo TCP/IP permite tanto a conexão de redes locais (LAN) como de redes globais (WAN), sendo o maior exemplo disso a própria internet que adota este protocolo para a maior parte da transmissão de dados, sem contar que os principais sistemas operacionais oferecem suporte para o TCP/IP (PALMA, PRATES, 2000).

Segundo Forouzan (2008) para transmissão de dados é necessário escolher um endereço de IP e um endereço de transporte. Um endereço de IP, chamado também de *host*, é escolhido para o destino enquanto outro de origem fica

encarregado da resposta do destino. O endereço de transporte, também denominado número de porta, seleciona um processo dentre os que estão em execução, esta porta fica encargo tanto da entrega quanto da resposta. A seleção dos endereços de IP e transporte estão representados na figura 8:

Figura 8: Seleção do endereço de IP (host) e o número da porta




Fonte: Behrouz A. Forouzan (2008, pg. 706)

A combinação destes endereços é conhecida como *socket*. Este *socket* é um mecanismo de comunicação de um cliente para um servidor, onde o cliente seria a origem e o servidor o destino. O protocolo exige um par de endereços *socket*, informações estas que fazem parte do cabeçalho IP, onde os endereços IP estão alocados, e do cabeçalho TCP, que contém as portas.

O TCP é um protocolo orientado a conexão e confiável. Utiliza as portas para implementar essa comunicação entre processos. Algumas portas já estão pré-definidas e representam funcionalidades específicas. A tabela 2 apresenta alguns exemplos destas portas pré-definidas:

*Tabela 2: Portas padrão usadas pelo TCP*

| *Porta* | *Protocolo* | *Descrição* |
|---|---|---|
| 7 | Echo | Ecoa um datagrama recebido de volta para o emissor |
| 9 | Discard | Descarta qualquer datagrama recebido |
| 11 | Users | Usuários ativos |
| 13 | Daytime | Retorna a data e a hora |
| 17 | Quote | Retorna o comentário do dia |
| 19 | Chargen | Retorna uma string de caracteres |
| 20 | FTP, Dados | File Transfer Protocol (conexão de dados) |
| 21 | FTP, Control | File Transfer Protocol (conexão de controle) |
| 23 | TELNET | Telnet |
| 25 | SMTP | Simple Mail Transfer Protocol |
| 53 | DNS | Domain Name Services |
| 67 | BOOTP | Bootstrap protocol |
| 79 | Finger | Finger |
| 80 | HTTP | HyperText transfer protocol |
| 111 | RPC | Remote Procedure Call |

Fonte: Behrouz A. Forouzan (2008, pg. 716)

Forouzan (2008) também ressalta que existem diversas maneiras de realizar a comunicação TCP/IP, e que a mais comum delas é a cliente/servidor. No caso o processo cliente requisita um dado de um servidor que, se bem-sucedida, responde ao cliente, enviando os dados pedidos. Tanto o processo cliente quanto o processo servidor precisam estar sincronizadas à mesma porta para comunicação ser bem-sucedida.

Este protocolo é orientado ao fluxo de dados, o que significa que ele envia e recebe dados através de um fluxo de bytes. O TCP também é do tipo full-duplex, como visto anteriormente, permite que esse fluxo de bytes trafegue em ambas direções simultaneamente, assim seus processos (cliente e servidor) implementa um buffer, que é uma área da memória de armazenamento temporária durante a transferência de dados, de transmissão e recepção para os dados trafegarem simultaneamente.

Para uma transmissão orientada a conexão, o TCP resume-se em três fases: estabelecimento da conexão, transferência de dados e encerramento da conexão, no qual Forouzan (2008) descreve:

- Estabelecimento da conexão: No TCP, por ser uma comunicação full-duplex, ambas as partes transmitem dados simultaneamente, isso implica que para estarem aptas a transmissão cada parte deve aprovar a comunicação que só é permitida após a inicialização da mesma.
- Transferência de dados: Após aprovada a comunicação, inicia a troca de mensagens, no qual, na maioria das vezes, o cliente requisita um dado e o servidor, em resposta, devolve o argumento pedido.
- Encerramento de uma conexão: Concluída a requisição, qualquer parte pode encerrar esta comunicação, mas que na maioria das vezes, vem do cliente, que a iniciou.

A camada de transporte oferece duas opções de serviços de transporte de dados: confiável ou não confiável (FOROUZAN, p. 708, 2008).

Se a confiabilidade de dados for de extrema importância, este serviço se mostra necessário. O então protocolo implementa mecanismos de controle de fluxos, erros e ordenação de pacotes, porém ao exigir este tipo de serviço o fluxo de dados se torna mais lento. Da forma quando este não exigir a mesma confiabilidade ou mesmo quando um sistema exigir respostas rápidas é interessante não optar por este serviço e implementar mecanismos próprios para confiabilidade de dados.

A confiabilidade de dados, como já citado, se resume em ordenação de pacotes de dados, na qual a comunicação deve garantir a entrega destes o fluxo inteiro na ordem, e controle de erros, na qual através do protocolo ou de mecanismos próprios verificam se há dados corrompidos ou perdidos. Este último utiliza de ferramentas simples tais como: *checksum*, confirmação de recebimento e *time-out*. Forouzan explica essas ferramentas.

- *Checksum*: Uma ferramenta que utiliza um algoritmo para validação de dados. O mesmo realiza uma série de operações com os bytes que são enviados e inclui nesta sequência um ou mais bytes que são os bytes verificadores. O ponto que recebe, realiza as mesmas operações e verifica se este corresponde ao byte verificador. Não correspondendo a mensagem é descartada.

- Confirmação: O TCP utiliza esta ferramenta para confirmar se o dado enviado foi recebido pelo destino. Quando um dado chega ao destino, o mesmo envia um sinal para a origem informando que a mensagem foi recebida.

- *Time-out*: É o tempo pré-definido para uma conexão ser satisfeita, caso ela não ocorra, a mesma automaticamente encerra a conexão.

# 3. Desenvolvimento

O desenvolvimento da interface gráfica terá como ferramentas principais a biblioteca KIVY e o software PyCharm, que tem como linguagem principal o Python. Os conceitos de orientação a objetos, usabilidade e flat design deverão ser levados em consideração no desenvolvimento para que a interface atenda as expectativas desejadas, como fácil manutenção, elevada funcionalidade e design chamativo. As comunicações de dados utilizadas serão a comunicação serial e a TCP/IP que também farão uso de bibliotecas já existentes para a linguagem, pyserial e socket; bibliotecas estas que trazem funções já implementadas para aplicação. As mesmas devem ser estruturadas de tal forma que os conceitos como entrega de dados correta, precisão e sincronização sejam atendidos.

A interface do autoatendimento será desenvolvida para que um usuário possa escolher os produtos/itens de forma interativa, pagar e receber o pedido. O desenvolvimento do programa colocará todos os conceitos a prova, pois demanda uma interface usual e bonita que interaja não só com o usuário também com outros equipamentos, através dos protocolos de comunicação utilizados, criando uma interação homem-máquina.

## 3.1 Desenvolvimento da interface gráfica

Importante entender que toda a aplicação possuirá uma única janela e esta será vinculada a um único widget. Isto se deve a principal característica desta linguagem, a operação multiplataforma, ou seja uma aplicação desenvolvida em Kivy deverá ser executada tanto em um Windows, que trabalha com várias janelas quanto em sistemas operacionais utilizados em dispostivos móveis. Este ultimo não criam várias janelas, o usuário se move pela aplicação através de telas, desta forma uma interface para atender os dois conceitos deve trabalhar com apenas uma única janela.

A execução do código kivy resultara numa estrutura hierárquica de classes e instancias, isto porque os objetos sempre estarão contidos dentro de outros. Da mesma forma acontecerá na definição de *widgets*, o nível mais alto da hierarquia sempre será a raiz da arvore ou como podemos chamar o topo da hierarquia e este

conterá todos os outros *widgets*. Geralmente, o *widget* raiz será um gerenciador de layout. Um gerenciador aloca todos os widgets e outros gerenciadores de *layouts*, que também são *widgets*.

Na parte gráfica foi criada uma classe chamada Mainscreen, esta classe contém um *widget BoxLayout* no topo da hierarquia, que é um gerenciador de *layouts*. Este *BoxLayout* tem como atributos principais orientação vertical, onde tudo que estiver nele será disposto verticalmente e fundo vinculado a uma imagem, sem cores para representação apenas da imagem. É neste gerenciador que todos *widgets* serão dispostos. A tabela 3 apresenta os *widgets* utilizados para a construção da interface gráfica:

Tabela 3: Classes usadas no desenvolvimento da interface gráfica

| **Classe** | **Descrição** |
|---|---|
| Label | O *widget* da classe *Label* é utilizado simplesmente para renderizar texto |
| Button | O *widget* da classe *Button* é uma Label associado a uma série de ações que são disparadas quando pressionado |
| ScrollView | O *widget ScrollView* fornece uma *viewport* de rolagem/panorâmica que é cortada na caixa delimitadora do *scrollview*. |
| BoxLayout | O *widget* da classe *BoxLayout* arranja *Widgets* de maneira adjacente (vertical ou horizontalmente), para preencher todo o espaço. |
| GridLayout | O *widget* da classe *GridLayout* organiza os componentes em uma matriz, calculando o espaço disponível na tela do aparelho utilizado. Dessa forma, o conteúdo é apresentado em linhas e colunas. |
| Popup | A classe *Popup* é um *widget* cuja função é criar *popups* modais. Por padrão, esse *popup* irá cobrir a janela principal. |
| Animation | A classe *Animation* é usada para animar as propriedades *Widget*. Você deve especificar pelo menos o nome da propriedade e o valor do objeto. |

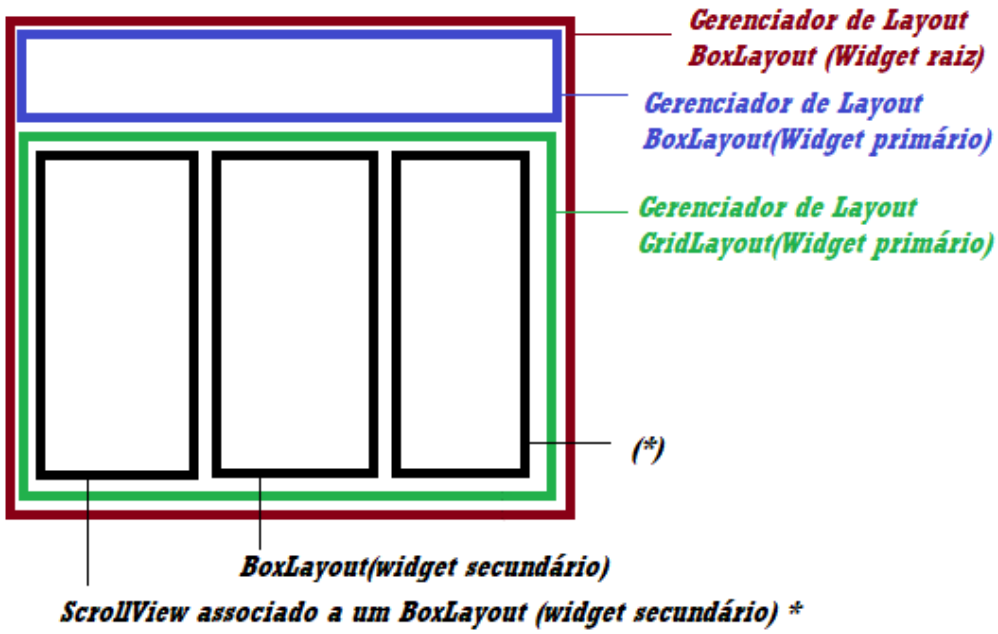
Fonte: Do próprio autor.

Descendo a hierarquia temos outros dois gerenciadores de *layout*, um da classe *BoxLayout* e outro da classe *GridLayout*, este segundo tem como principal característica a divisão de sua estrutura em colunas. O *BoxLayout* desta hierarquia conterá dois *widgets* da classe *Button*, inseridos em um retângulo que representa uma faixa dentro do *widget* raiz. Os botões dispostos neste gerenciador são responsáveis

por mostrar e limpar a lista de objetos selecionados. Enquanto o *GridLayout* será dividido em 3 colunas, terá orientação vertical e armazenará três *widgtes* do tipo *BoxLayout*, dois com propriedades *ScroolView*, que permite que os *widgets* nela inseridos disponham de uma barra de rolagem.

Uma vez definida a estrutura hierárquica dos gerenciadores de layout, basta adicionar os outros componentes que representam imagens e botões, e estes *widgtes*, por sua vez, serão filhos dos gerenciadores a qual estão dispostos. A representação da hierarquia é vista na figura abaixo:

*Figura 9: Estrutura hierárquica dos gerenciadores de layout*



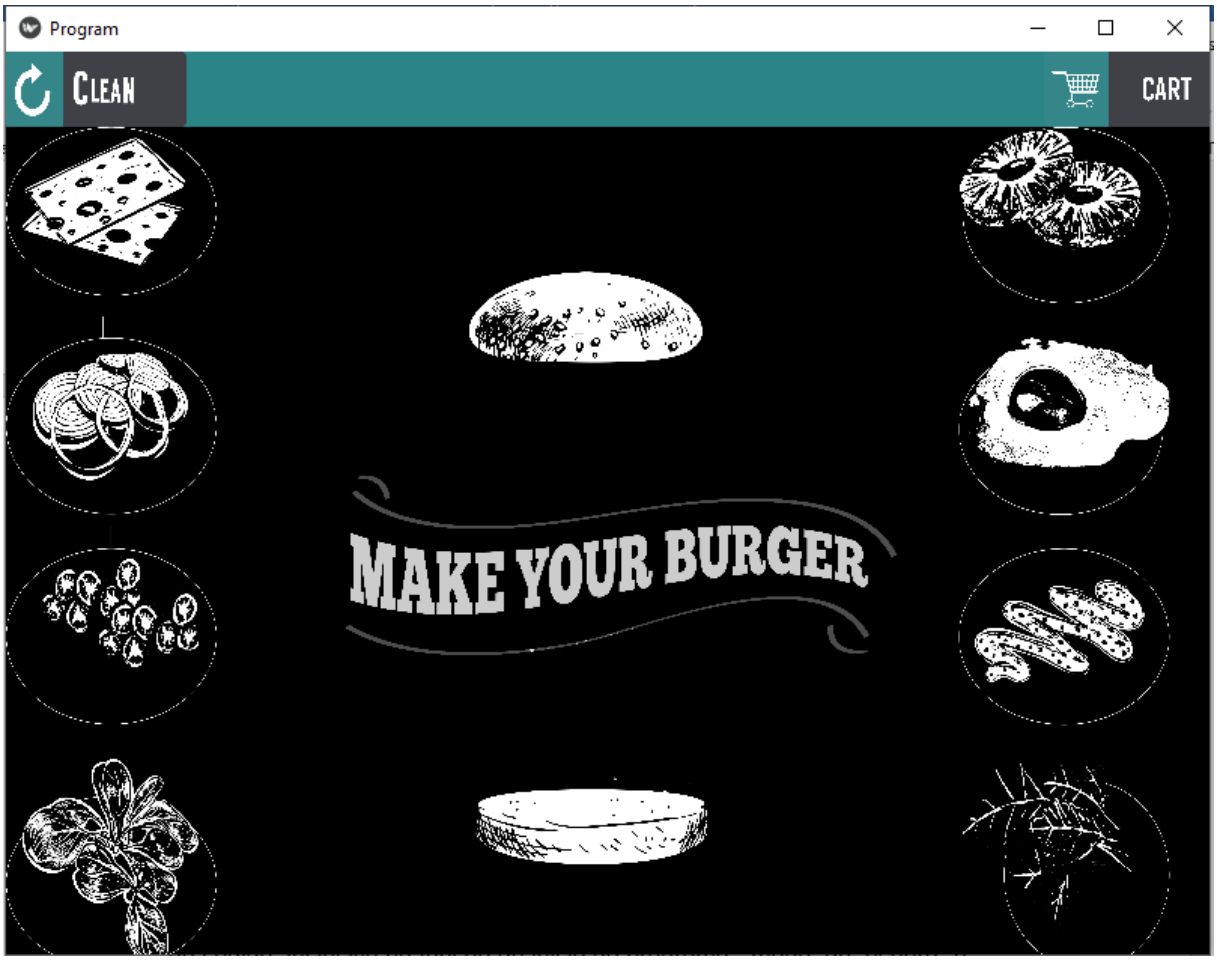

*Fonte: Do próprio autor.*

Nos dois *BoxLayout* associados a um *ScrollView* serão inseridas *widgets Button* com fundo associado a imagens e sem borda, para que o botão tome a forma da imagem carregada, mantendo suas propriedades de botão. No *BoxLayout* do centro serão inseridas três imagens para *design*, onde uma será animada no início do programa.

Estes *widgets* que são alocados nos *BoxLayouts* secundários representam a interação com o usuário, eles são responsáveis por adicionar os itens na lista, mostrar

os itens selecionados e limpar a lista, interagindo com o usuário, por meio de animações e representações dinâmicas.

Figura 10: Tela inicial da interface gráfica




Fonte: Do próprio autor

Na figura acima podemos observar a hierarquia dos componentes da interface do mesmo jeito que foi representada na figura 9. No topo encontra se o *BoxLayout* raiz que tem como filhos o *BoxLayout* que aloca uma faixa onde os botões Clean e Cart estão, e, abaixo deste, está o *GridLayout* onde outros três gerenciadores estão alocados e por fim as imagens e botões, filhos dos gerenciadores anteriores.

Partindo para a parte funcional do código encontramos a mesma classe Mainscreen que recebe a parte gráfica do código .kv. Nesta classe encontramos 7

funções, sendo que uma delas provém do método *“__init__”*, que nada mais faz que definir as condições iniciais das variáveis e dos *widgets* ao iniciar o programa.

Em seguida temos uma função nomeada como "modo_de_espera", uma função criada para resumir o código já que seu bloco de instruções é necessário em várias partes do código, inclusive na função de início do programa. “modo_de_espera" é responsável por verificar se nenhuma imagem está selecionada, caso a mesma seja falsa executa uma sequência de instruções que retira a imagem animada, caso seja verdadeira adicionasse a imagem animada novamente.

As outras funções são chamadas quando eventos "*click*" são disparados. Ao clicar nos botões-imagens a função "click_itens" é chamada e traz consigo dois parâmetros, uma variável do tipo ponteiro, que aponta qual botão-imagem foi clicado, e o dicionário do conjunto de imagens daquele *BoxLayout*, para verificar a qual chave do dicionário o botão-imagem pertence. A função é responsável por verificar se o botão-imagem clicado já está selecionado, se falso, a mesma informa para um dicionário que a determinada imagem está sendo selecionado e altera a cor de fundo do botão-imagem para a cor padrão do item selecionado, em seguida remove a imagem animada do *BoxLayout* central e adiciona um *widget Image* com a mesma figura do botão, para trazer interação com o usuário. Da mesma forma, caso a verdadeiro, a mesma informa para aquele dicionário que a determinada imagem está sendo deselecionada e altera a cor de fundo do botão-imagem para a cor padrão, remove o *widget* imagem adicionado com a ação do disparo anterior e verifica se algum item está selecionado com a função "modo_de_espera”.

Figura 11: Representação de eventos click da função Click_itens

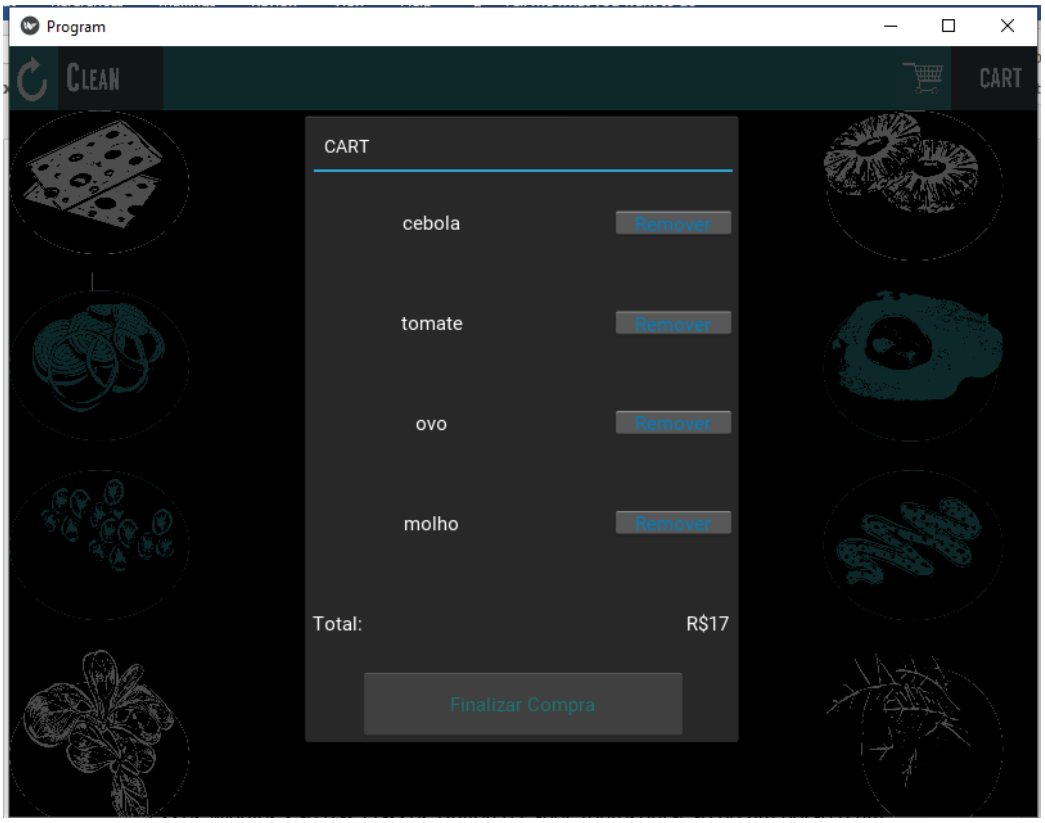

Fonte: Do próprio autor

A função “Clean” tem como objetivo limpar a lista de objetos selecionados, voltando para as condições iniciais do programa. A função realiza uma varredura pelo dicionário que aponta para os objetos selecionados e aqueles que estiverem selecionados terá sua condição alterada para falsa, a cor de fundo da imagem-botão será alterada, os *widgtes* que foram adicionados dinamicamente, no evento click da imagem-botão, serão removidos e as condições iniciais do programa retomam com a função “modo_de_espera”.

A função que se segue é responsável por representar a lista de itens selecionados pelo usuário. Definida como “Cart”, a função cria um *popup* que é uma janela que se abre no navegador para apresentar uma informação extra. Nesta popup será criado um *BoxLayout* onde um botão com a finalidade de concluir o programa e outros *widgtes* serão adicionados. Esses *widgtes* são das classes *Button* e *Label*, representação de um botão e um texto. Eles serão criados dinamicamente a partir dos objetos selecionados e são dispostos juntos em cada iteração. O *popup*, então, irá

apresentar quais foram os itens selecionados e, caso algum não estiver correto, um botão para remove-los diretamente no *popup* estará representado.

Figura 12: Representação evento click da função Cart




Fonte: Do próprio autor.

Este botão criado dinamicamente, quando clicado aciona a função "click_menos" que é responsável por remover os objetos do *popup*, ao mesmo tempo que, orientado, interage com navegador desselecionando os itens e removendo os *widgtes* do próprio navegador que foram apontados pelo botão, além da verificação “modo_de_espera”.

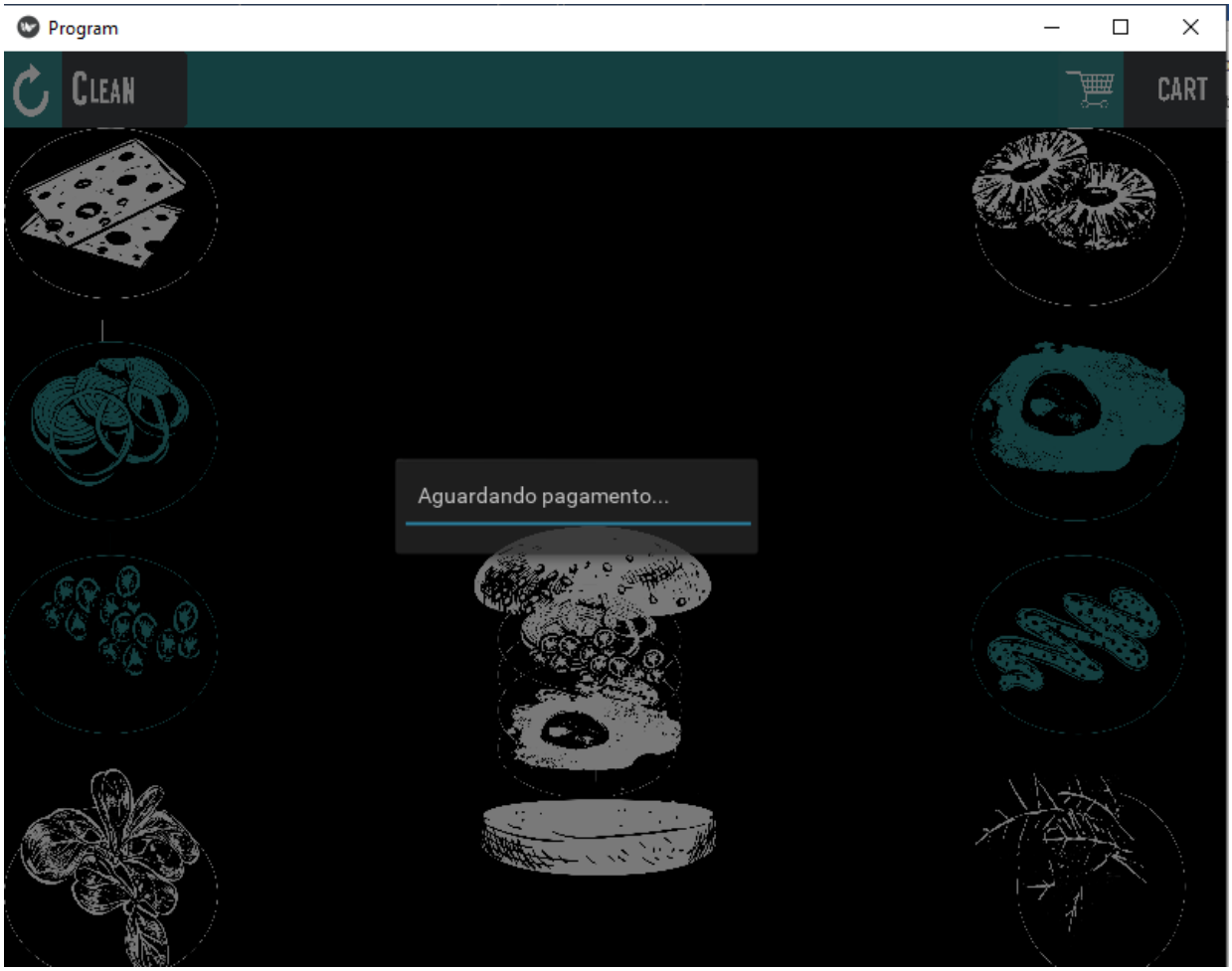
As funções que se seguem se referem as comunicações de dados e para melhor organização, serão tratados no tópico a seguir.

### 3.2 Protocolos de comunicação

Para simulação das comunicações seriais e TCP/IP foram utilizados um emulador de comunicação serial, Virtual Serial Ports Emulator, que simula uma entrada serial, e o Hercules que é uma ferramenta que exibe esses dados, seriais e TCP/IP.

O botão concluir fecha o *popup* da lista e verifica se algum item está adicionado, caso a mesma não esteja vazia cria um *popup* e uma animação para informar, e interagir, que o equipamento está aguardando o pagamento, como é visto na figura 13. Enquanto a interface gráfica interage com o usuário, uma nova *thread é iniciada*. A mesma está vinculada a função “pagamento” onde serão realizadas as transferências de dados.

Figura 13: Representação evento click da função concluir

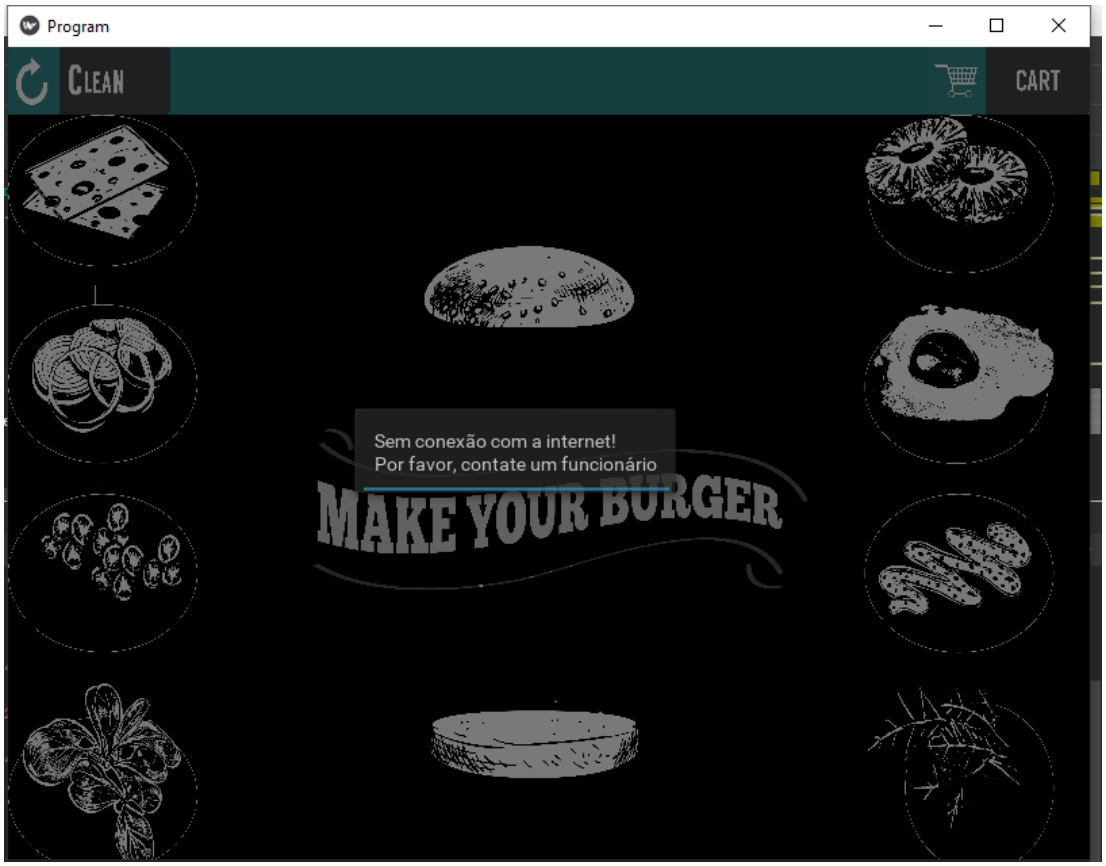



Fonte: Do próprio autor

A importância de se dividir o processo em duas ou mais vias está na necessidade do programa executar as representações gráficas enquanto realiza as transmissões de dados. No caso do programa desenvolvido, caso não fosse dividido o processamento em duas *threads*, nossa interface não reagiria enquanto nosso controlador estivesse esperando uma resposta de algum equipamento.

A função pagamento, primeiramente, tenta estabelecer uma conexão com o servidor através dos comandos *try/except.* Com este recurso é possível prever um erro antes que o mesmo ocorra, como uma conexão não bem-sucedida. Caso o erro ocorra o programa tomará uma série de decisões a partir desta informação. Na interface desenvolvida caso o equipamento não consiga se comunicar com o servidor, um *popup* será criado para informar para o usuário que a conexão não foi estabelecida e que não será possível realizar a operação, limpando a lista de itens selecionados, conforme a figura 14.

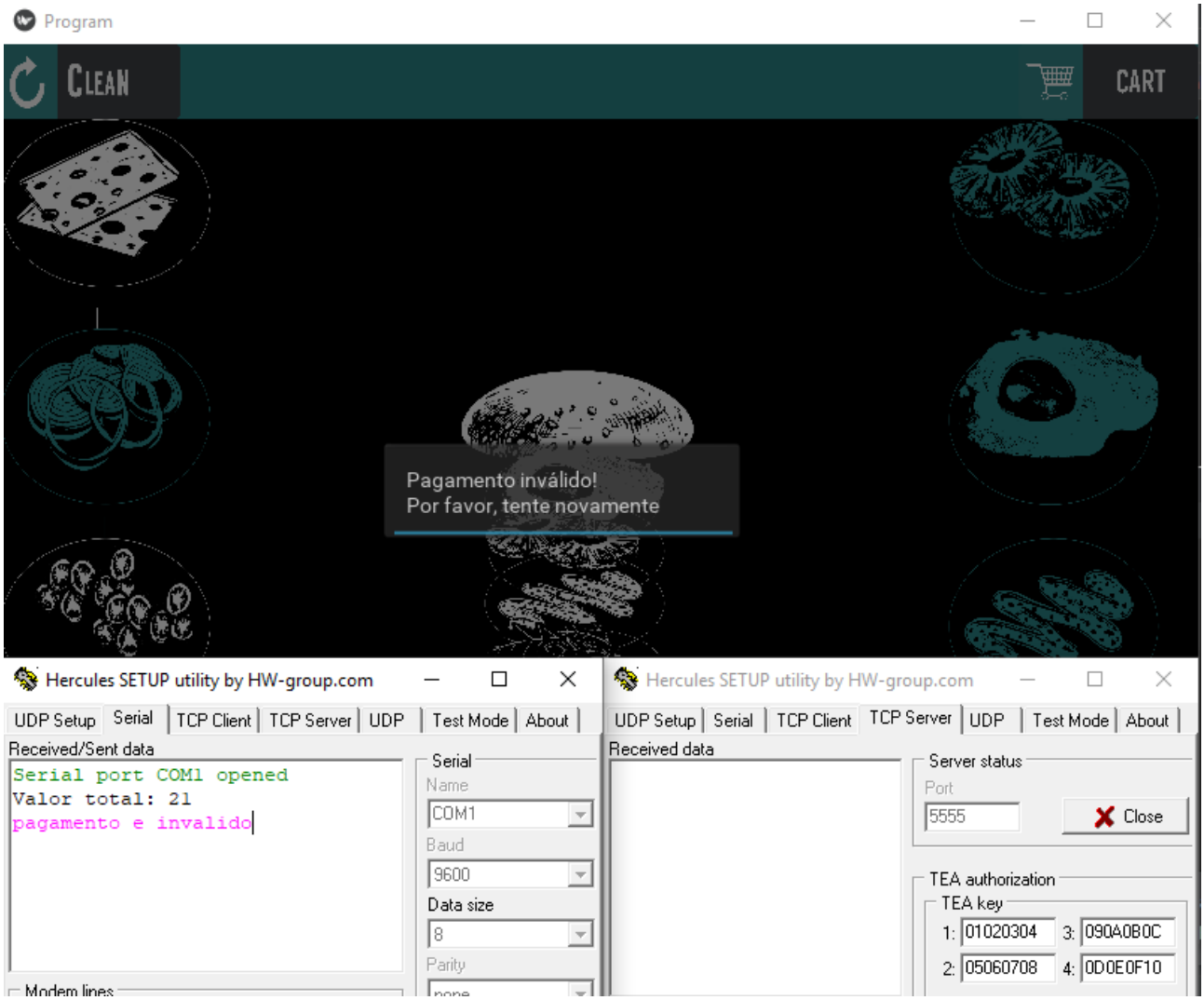
Figura 14: Mensagem avisando falha na conexão com o servidor




Fonte: Do próprio autor.

Caso a conexão seja bem-sucedida, inicia-se uma conexão serial com a porta serial emulada, nas quais suas configurações são estabelecidas como *BaudRate* igual a 9600bps, paridade nula, 8 bits de parada e *timeout* de 20 segundos, que será o tempo limite para concluir a operação. O micro controlador, então, envia via porta serial o valor total da compra e aguarda por 20 segundos uma resposta, caso receba uma resposta inválida ou o tempo de execução for excedido, uma outra mensagem é exibida para o usuário informando que o pagamento não foi concluído. Neste caso, a lista não é enviada para o servidor e, por questões de usabilidade, os itens da lista não são retirados, assim agilizando uma nova tentativa de pagamento, de acordo com a figura 15.

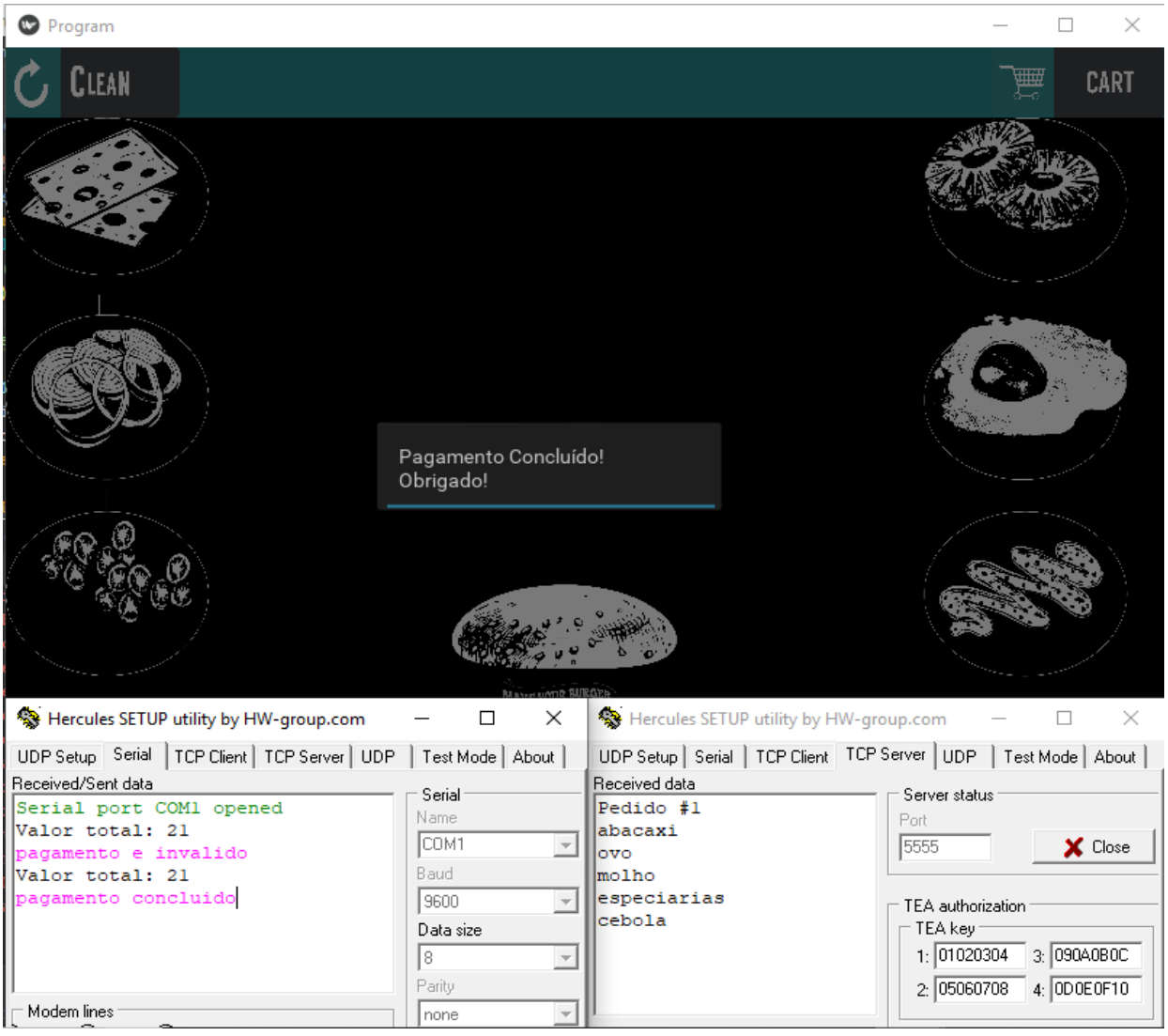
Figura 15: Mensagem informando que o pagamento foi inválido e mensagem serial exibida na ferramenta Hercules




Fonte: Do próprio autor

Uma vez o pagamento concluído, a interface informa o usuário da compra e envia, via TCP/IP, o número do pedido junto a lista de itens selecionados para o servidor. Neste caso nosso servidor foi definido por um IP fixo na nossa rede e os dados enviados para este socket foram visualizados na ferramenta Hércules, visto na figura 16.

Figura 16: Mensagem informando que o pagamento foi concluído e dados seriais e TCP exibidos na ferramenta Hercules




Fonte: Do próprio autor.

Encerrada a operação, a interface retorna para as condições iniciais para que assim um próximo usuário possa escolher seus itens a sua maneira. Outros equipamentos poderiam fazer parte deste sistema, como por exemplo uma impressora para imprimir os comprovantes, onde os dados chegariam para ela

através do RS-232, já desenvolvida. Qualquer tipo de equipamento pode ser adicionado ao sistema, basta a correta comunicação com da interface com os equipamentos.

### 3.3 Comunicação com os periféricos

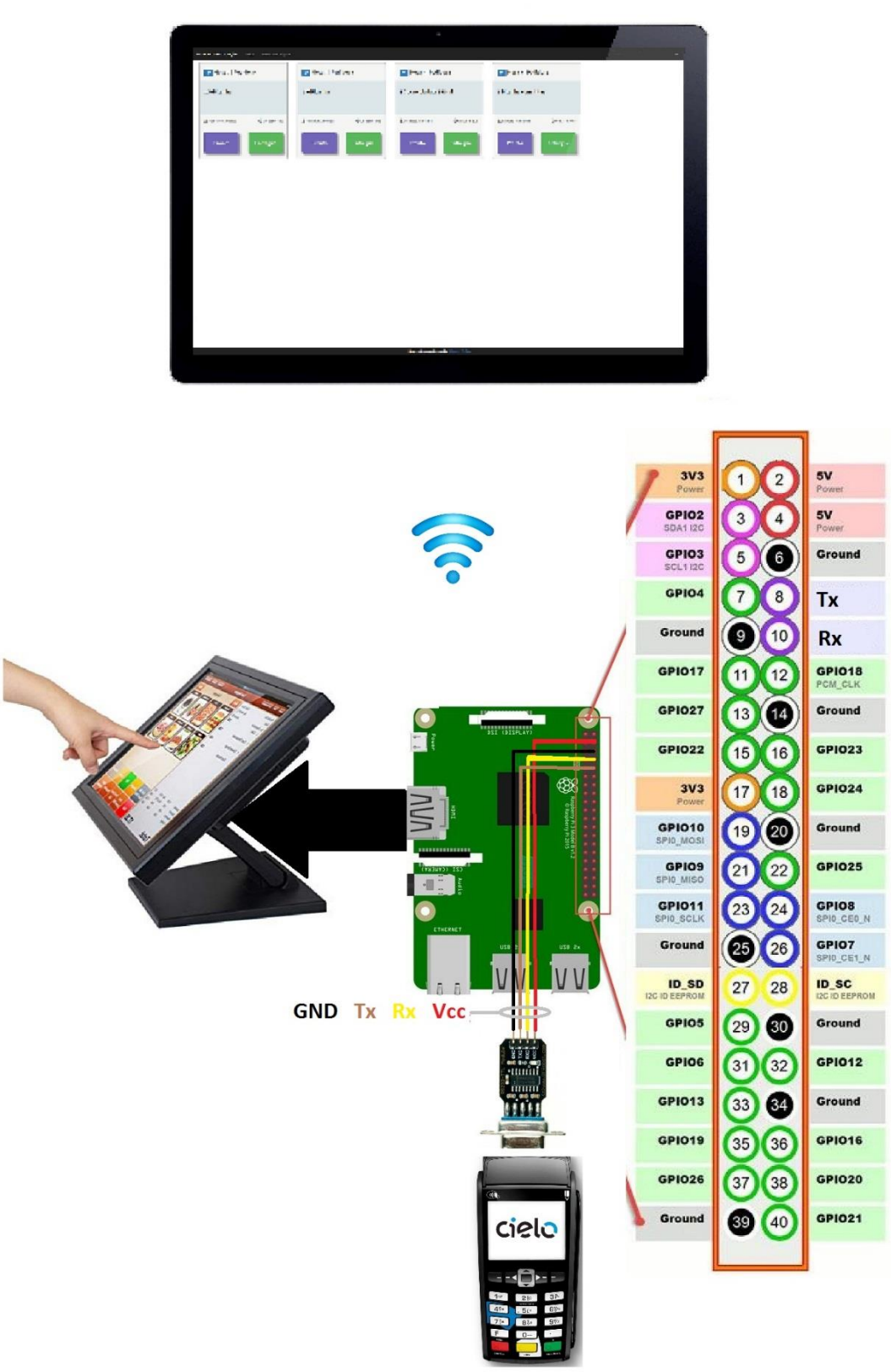
Para realização da comunicação entre a interface e os periféricos utilizou-se o microcontrolador Raspberry Pi3. A interface seria exibida em um monitor *touch* com comunicação HDMI, porta esta oferecida pelo Raspberry. Ao iniciar o sistema operacional, um script pode ser adicionado, podendo este ser em Python, para que o mesmo inicie o software quando o sistema fosse iniciado.

Para a comunicação entre a máquina de cartão e o microcontrolador, faz uso de um módulo DB9-TTL que permite que as saídas e entradas digitais do Raspberry interligue com o dispositivo. Os pinos Rx e Tx interligam com o módulo de forma inversa, ou seja, o pino Rx do microcontrolador faz ligação com o Tx do módulo, enquanto o Tx do Raspberry faz ligação com o Rx do módulo, para que assim a entrada de um comunique-se com a saída do outro. É necessário também equipotencializar o sistema, interligando os Terras do módulo com o do microcontrolador. O módulo também necessita de uma alimentação 5V, o qual também pode ser realizada interligando com a saída de tensão do micro.

O Raspberry Pi3 também possui um chip para comunicação sem fio, o que dispensa o uso de um módulo para a transferência de dados TCP. Porém para que a cozinha receba as mensagens enviadas pelo autoatendimento seria necessário que o software do servidor estivesse endereçado para o mesmo IP e porta que nosso sistema, além de estar sempre com a conexão aberta, para que a cozinha sempre possa receber mensagens do autoatendimento.

As comunicações e ligações dos periféricos estão representadas na figura 17.

Figura 17: Comunicação da interface com os periféricos




Fonte: Do próprio autor

## 4. Conclusões:

O desenvolvimento de uma interface que não só interage com os usuários como também é capaz de se comunicar com os equipamentos trouxe os resultados esperados. Durante o desenvolvimento não foi possível só enxergar as vantagens desse sistema e as barreiras que deveriam ser enfrentadas, como também foi possível ver as possibilidades que uma interface desta pode trazer.

Os critérios de interface se mostraram satisfatórios. A interface tem alta usabilidade, pois é uma ferramenta simples, objetiva e rápida. A hierarquia está bem definida e os *widgets* estão dispostos para que se tenha uma interação rápida. A programação está coerente, tendo nenhum evento atrapalhando a ação de outro. A interface atende os conceitos de “FlatDesing”, pois a mesma está corretamente alinhada, imagens sem fundo, animações, graus de leveza e opacidade, propriedades que uma interface moderna de ter.

O envio e recebimento de dados ocorreu de forma correta. A comunicação de dados pelos protocolos entregou as mensagens nos destinos certos e apenas para o destinatário solicitado, a entrega do pacote foi precisa, não sofrendo alterações por interferências e houve sincronização na troca de mensagens, um dispositivo que enviava a mensagem, recebia a resposta sincronizada e correta.

As barreiras encontradas foram as esperadas. Exigiu elevado conhecimento em programação orientada a objetos. O desenvolvimento gráfico exigiu que todos os componentes fossem apontados e assim poder alterar suas propriedades dinamicamente criando uma interação com o usuário a cada clique.

Para realização das comunicações se mostrou necessária a execução de tarefas em paralelo, pois a interface gráfica não pode parar enquanto outra operação está sendo realizada. Portanto também se fez necessário conhecimento em threads, semáforos, em geral habilidades com programação em tempo real.

A comunicação do homem com a máquina através de interfaces com transmissão de dados não se limita a aplicação proposta no trabalho. Uma aplicabilidade está na automação residencial, onde é possível controlar e monitorar todos os dispositivos IoT de sua residência, utilizando apenas uma interface e um protocolo sem fio. Um outro exemplo está nos sistemas de autoatendimento para bebidas, na qual uma interface está ligada a um sensor de vazão e uma máquina de cartão por meio de um microcontrolador que cobra do cliente os ml consumidos.

Uma muita usada está nos transportes públicos que por meio de um sensor RFID coleta os dados do cartão que confere com um banco de dados e exibe para o usuário. As aplicações para esta tecnologia são gigantes. Uma gama de dispositivos possui a capacidade de comunicação com microcontroladores, número este que só vai aumentar com os conceitos de IoT; com uma interface interessante, milhares de produtos e serviços podem ser desenvolvidos e oferecidos para população.

## 5. Referências

ARDUINO E CIA. **Raspberry Pi**: Instale o Raspbian e crie seu primeiro programa em Python. 26/novembro/2014. Disponível em: <https://www.arduinoecia.com.br/2014/11/raspberry-pi-b-raspbian-python.html> Acessado em 11/04/2019

BARBOSA, André Sarmento. **Interfaces e periféricos**. 2018. Disponível em: < http://www.andresarmento.com/estacio/interfaces/>. Acesso em: 10 abr. 2019

BORGES, Luis Eduardo. **Python para desenvolvedores**. 1ª edição. São Paulo: Novatec Editora Ltda., 2014

**CERNE TECNOLOGIA**. 2006. Disponível em < http://www.cerne-tec.com.br/Artigo_07_ComunicacaoSerial.pdf> Acesso em: 10 jul. 2019

**CERP IoT**. 2009. Disponível em: <https://pdfs.semanticscholar.org/c0db/f1420620f9444082aad54e98c634e0ae438f.pdf> Acesso em: 5 jun. 2019.

EBERMAM, Elivelto; PESENTE, Guilherme; RIOS, Renan Osório; REBELO, Igor Irla Bocianoski. **Interação Entre Homem e Computador**. 2009. Disponível em <https://pt.scribd.com/document/19653938/IHC-Interacao-entre-Homem-e-Computador-Apostila-TASI-IHC-2009-2>. Acesso em: 7 abr. 2019

EIA – **EIA Information**, 1996. Disponível em: <ftp1.digi.com/support/cabling/eia_232_info.pdf> Acesso em: 12 abr. 2019

FONTINELLE, Carlos Gomes. **Fundamentos de Redes de Computadores.** 2015. Disponível em: <livrosdigitais.org.br/baixar-livro/1478APHFFQWAR> Acesso em: 10 abr. 2019

FOROUZAN, Behrouz A. **Comunicação de dados e redes de computadores.** 4ºed. Porto Alegre: Editora AMGH Ltda., 2010

JOHNSON, Steve. **Cultura da interface**. Rio de Janeiro: Jorge Zahar Editor, 2001

**KIVY DOCUMENTAÇÃO**. Versão 1.9.2. dev0. Disponível em: <https://buildmedia.readthedocs.org/media/pdf/kivy/latest/kivy.pdf>. Acesso em: 9 abr. 2019

LOURENÇO, L. **A História da informática hardware.com**. 10/agosto/2011. Disponível em: <https://www.hardware.com.br/guias/historia-informatica/lisa-macintosh.html> Acessado em 10/04/2019

LOURENÇO, W. **Windows 8**: Visão geral e preparação do ambiente de desenvolvimento. 2012. Disponível em: <https://www.devmedia.com.br/windows-8-visao-geral-e-preparacao-do-ambiente-de-desenvolvimento/26170> Acessado em 10/04/2019

NETO, Benjamin Batista de Oliveira; MONTEIRO, Priscila de França; QUEIROGA, Sandro Lino Moreira. **Aplicabilidade dos Microcontroladores em Inovações Tecnológicas**. 2012. Disponível em: < http://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/2433/2526>. Acesso em: 10 JUN. 2019

OLIVEIRA, Joaquim Francisco Cavalcante. **Avaliação das qualidades ergonômicas de interfaces interativas de sistemas de informação**. 2004. Disponível em: <https://www.aedb.br/seget/arquivos/.../125_INTERATIVIDADEJoaquimFrancisco.doc>. Acesso em: 5 abr. 2019

PALMA, Luciano; PRATES, Rubens. **Guia de Consulta Rápida TCP/IP**. Novatec Editora Ltda., 2000

PULINI, Carlos - **Programação para leigos com RASPBERRY PI**. 1ºed. João Pessoa: Editora do Ifes, 2017

ULLOA, Roberto - **Kivy: Interactive Applications in Python**. 2ª edição. UK: Packt Publishing Ltd., 2013

UPTON, Eben; HALFACREE, Gareth - **Raspberry PI. Guia do usuário**, Editora Alta Books, 2017

TEXAS INSTRUMENTS - **Interface circuits for TIA/EIA-232-F**. 2002. Disponível em: <http://www.ti.com/lit/an/slla037a/slla037a.pdf>. Acesso em: 10 abr. 2019