UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
ENGENHARIA ELÉTRICA

ROBSON DE SOUZA DONATO

DESENVOLVIMENTO DE UM SISTEMA RFID DE ACESSO DISTRIBUÍDO SOB A PERSPECTIVA DE IoT

CAMPINA GRANDE
2019

ROBSON DE SOUZA DONATO

DESENVOLVIMENTO DE UM SISTEMA RFID DE ACESSO DISTRIBUÍDO SOB A PERSPECTIVA DE IoT

Trabalho de conclusão de curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Orientador: Rafael Bezerra Correia Lima

CAMPINA GRANDE

2019

ROBSON DE SOUZA DONATO

DESENVOLVIMENTO DE UM SISTEMA RFID DE ACESSO DISTRIBUÍDO SOB A PERSPECTIVA DE IoT

Trabalho de conclusão de curso submetido à Coordenação do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

APROVADA EM: 12/07/2019

Rafael Bezerra Correia Lima
Orientador
UFCG

Gutemberg Gonçalves dos Santos Júnior
UFCG

CAMPINA GRANDE
2019

Agradecimentos

Prioritariamente a **Deus** e minha família por tudo, em especial, meu exemplo de vida: minha mãe e pai **Rosadalha Donato**.

Além disso, não posso deixar de citar o melhor grupo que integrei em todos esses anos de minha vida: Armário. Sou grato a todos pelo auxílio em meu crescimento pessoal, por me fazer expandir minha visão de mundo, reavaliar alguns conceitos e interpretações relacionados à vida em sociedade. Há, no entanto, um subgrupo do Armário, o qual pela mistura do efeito do tempo, do acaso e das ações recorrentes de consideração a mim (expressas tão particularmente durante os últimos anos), que me sinto na obrigação de expô-lo registrando o nome de cada membro. Listá-los-ei a seguir sem indicação de prioridades, da mesma forma que fiz para melhor identificá-los assim que os conheci: pela naturalidade. Portanto, têm-se:

- Núcleo potiguar: João Jales, João Pedro, Kaio Nikelisson, Lucas Oliveira.
- Núcleo cearence: Paulo Roberto, Victor Germano, Goldofredo Feitosa.
- Núcleo paraibano: Giuseppe, Ítalo, José Adeilmo, João Victor, Matheus Ferreira, Osmar Lucas, Raphael Victor, *Ravi Helon* (uma das pessoas mais sinceras que eu conheci), Safire Torres, Ulisses Gomes, Vandilson, Walter Guedes.
- Núcleo baiano: Paulo Victor.

Também agradeço ao professor Rafael Bezerra Correia Lima pelo auxílio neste trabalho.

Por fim, porém, absolutamente não menos importante que os registros anteriores, sou grato por ter conhecido, por intermédio do curso, uma pessoa muito importante para mim: minha namorada Darlanny Diniz.

—

“Nada façais por partidarismo ou vanglória, mas por humildade”
(Filipenses 2:3)

Resumo

Este projeto tem por principal objetivo apresentar o desenvolvimento de um um módulo de acesso RFID que funciona, sob a perspectiva de um dispositivo IoT, junto a um sistema remoto que o monitora e o administra além de processar e armazenar informações dos usuários que o utilizam. Para isso, foram utilizadas algumas soluções de hardware, tais como: placa de desenvolvimento baseada no chip ESP-32 (Node MCU 32S) e leitor RFID fabricado com o chip MFRC522. A linguagem de programação e IDE *open-source* *Processing* foi escolhida para implementação da interface gráfica associada ao software administrador, o qual, assim como outros dispositivos do sistema, utiliza protocolo de comunicação MQTT para troca de informação.

Palavras-chaves: IoT, ESP-32, RFID, *Processing*, MQTT.

Abstract

This project has as main objective to present the development of an RFID access module that works, from the perspective of an IoT device, next to a remote system that monitors and manages it besides processing and storing information of the users that use it. For this, some hardware solutions were used, such as: development board based on the ESP-32 chip (Node MCU 32S) and RFID card made with MFRC522 chip. The open-source programming language and IDE Processing was chosen to implement the graphic interface associated with the software administrator, which, like other system devices, uses MQTT communication for information exchange.

Key-words: IoT, ESP-32, RFID, Processing, MQTT.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Controle de acesso, de modo geral, significa autorizar, restringir ou negar a entrada ou saída de pessoas ou veículos em um determinado local. Seja em uma residência comum, laboratórios ou departamentos de desenvolvimento tecnológico, é razoável inferir que sempre existirão zonas em muitos destes ambientes que demandam preocupações maiores em relação ao método, à gerência e à segurança da propriedade, das pessoas, de invenções, equipamentos e outros ativos.

Embora muitas soluções de controle de acesso de pessoas existam no mercado, ainda não é comum e relativamente barato aquelas cujo conjunto eletro-mecânico (responsável pela a ação física final de liberar ou não o acesso) integre um sistema cliente-servidor de modo que sua possibilidade de conexão à internet o habilite a receber de um administrador remoto comandos para constantes reconfigurações e também enviar dados importantes que ajudem no monitoramento das pessoas e dos acessos aos ambientes. À luz da crescente perspectiva de ser possível integrar qualquer objeto à rede de internet, aplicações desse tipo em diversos setores tendem a ser cada vez mais comuns.

De modo geral, imaginar a integração de qualquer objeto à internet não é mais um conceito abstrato, pois devido a aceitação cada vez crescente do conceito de *internet of things - IoT*, muitas soluções de hardware vêm sendo testadas, bem como a solução de comunicação baseada no protocolo MQTT, acrônimo para *Message Queuing Telemetry Transport*, para esta finalidade (BUYYA; DASTJERDI, 2016).

IoT pode ser entendido como a possibilidade de integração de dispositivos físicos que podem possuir sensores, software, microcontroladores etc. em suas construções, de modo que seja puramente viável estabelecer a troca de informações entre os mesmos e entre a rede mundial de computadores. Atualmente, muitas aplicações com IoT estão surgindo devido a sinergia entre empresários, indústria, clientes e a internet, o que justifica a atenção especial da comunidade acadêmica em entender, aprimorar e tentar inovar cada vez mais o propósito de integração do mundo físico ao mundo virtual (MAVROMOUSTAKIS; MASTORAKIS; BATALLA, 2016).

MQTT, por sua vez, é um protocolo de comunicação simples originalmente criado em 1999 para comunicação entre máquinas (M2M - *machine-to-machine*) em aplicações industriais. Por ser simples, leve e também atender bem ao conceito de IoT, MQTT tem se tornado o principal protocolo para essa tecnologia. Há três elementos básicos na topologia deste protocolo: *Subscribe*, *Publish*, *Broker*. O *Broker* é o servidor MQTT conectado à nuvem que pode receber informações de qualquer lugar do planeta (que possua internet). Os *Publishers* disponibilizam informações para serem entregues aos *Subscribers* e estas

são sempre intermediadas pelo *Broker* (HILLAR, 2017).

Face aos conceitos apresentados e entendendo-se, portanto, o hardware de um módulo de acesso como um objeto capaz de receber um endereço IP, enviar e receber dados trabalhando em conjunto com um software administrador, a proposta principal deste trabalho é desenvolver um módulo de acesso RFID e um software administrador capazes de trocar informações usando-se o protocolo MQTT. Essas tecnologias estão em ênfase dentro das perspectivas oferecidas por Internet das Coisas.

Buscou-se cumprir os seguintes objetivos geral e específicos:

1.1 Objetivo Geral

Desenvolver um módulo de acesso RFID de modo a integrá-lo, sob a perspectiva de um dispositivo IoT, a um sistema remoto de monitoramento e administração do mesmo e dos usuários que o utilizam.

1.2 Objetivos Específicos

O módulo de acesso deve ser um cliente MQTT com as seguintes características:

- Identificar usuários a partir da leitura de tags RFID que operam na frequência 13,56 MHz
- Receber informações de um servidor de retaguarda para constante reconfiguração e ser capaz de operar tanto online quanto offline
- Possuir LED e display indicadores de status de acesso bem como um sistema alternativo de energia (bateria de lítio)

Faz-se necessário neste sistema um servidor de retaguarda como cliente MQTT com as seguintes funcionalidades:

- Possuir interface gráfica desenvolvida a partir da linguagem de programação *Processing*
- Possuir um estágio de autenticação para garantir certo grau de segurança na comunicação com o *broker*.
- Configurar inicialmente novos dispositivos IoT via comunicação serial
- Oferecer opções de cadastramento/descadastramento de usuários e associação dos mesmos às tags RFID

- Enviar e receber periodicamente dados dos dispositivos IoT que, junto ao software de interface gráfica, fazem parte do sistema.

2 Referencial Teórico

Nesta seção serão abordados os principais aspectos teóricos relacionados ao aprendizado necessário para a concepção do projeto proposto neste trabalho.

2.1 Sistemas Distribuídos

Um sistema distribuído é um sistema que consiste de uma coleção de máquinas autônomas conectadas por uma rede de comunicação e equipadas com sistemas de softwares designados para produzir um ambiente computacional consistente e integrado (JIA; ZHOU, 2005). Do ponto de vista do usuário um serviço pode ser percebido como uma simples tarefa executada por uma única máquina, no entanto, em muitos casos, muitas máquinas são utilizadas para prover o funcionamento do sistema que a torna possível.

O mais bem conhecido sistema distribuído é a coleção de servidores web que juntos fornecem o banco de dados para documentos de hipertexto e multimídia conhecido como *World-Wide-Web*. Portanto, um sistema cliente-servidor é um sistema distribuído em que o servidor compartilha recursos com vários clientes e, em geral, a comunicação entre os nós da rede é feita com auxílio de vários outros nós que integram o sistema. Algumas das principais características dos sistemas distribuídos são:

- **Compartilhamento de recursos** - Software, hardware e dados podem ser compartilhados entre os usuários. Por exemplo, uma impressora pode ser compartilhada entre vários usuários;
- **Escalabilidade** - A possibilidade de expandir o número de dispositivos do sistema;
- **Tolerância a falha** - Os dispositivos conectados à rede podem ser vistos como recursos de redundância, um sistema de software pode ser instalado em múltiplas máquinas de modo que frente a uma falha de hardware e/ou software possa ser detectada e tolerada por outros dispositivos;

Entre algumas desvantagens pode-se citar:

- **Complexidade** - Coordenar tudo pode não ser uma tarefa simples, custa caro, além de problemas relacionados a segurança.
- **Segurança** - A facilidade oferecida pelos sistemas distribuídos no acesso aos dados, pode criar uma dificuldade na garantia da segurança dos dados existentes ou em relação à privacidade dos dados secretos.

- **Custo** - Existe um alto custo para a implementação de aplicações colaborativas, devido ao fato de os recursos estarem fisicamente separados.

2.2 IoT - *Internet of Things* (Internet das coisas)

Faz-se importante, antes de apresentar uma definição de IoT, entender o propósito de um objeto/dispositivo conectado à rede de internet. A ideia central por trás dessa concepção é tornar possível a independência do objeto em sua interação com o mundo físico, ou seja, sem intervenção humana. Como exemplo, pode-se pensar em um aparelho de ar condicionado projetado para possuir conexão com a internet e, dessa forma, ser capaz de se comunicar com outros objetos dentro de um mesmo ecossistema. Essa possibilidade pode aumentar as funcionalidades desse aparelho e a forma com a qual o mesmo interage com o ambiente. Então, com essa nova possibilidade de receber informações de outros objetos em relação à presença ou não de pessoas dentro do ambiente e/ou qual pessoa está dentro do ambiente e suas preferências, algum algoritmo pode tratar esses dados e em seguida realizar algumas decisões automáticas, como: realizar o desligamento ou não do aparelho, colocar-se no melhor ponto de operação frente às condições instantâneas do ecossistema. Essas atribuições que, em certa medida, podem ser denominadas de "atribuições cognitivas" garantem outro nível de relevância para as coisas e, neste exemplo, vão muito além do simples objetivo de condicionar a temperatura dado um ponto de operação determinado por um usuário.

Há, no entanto, algumas limitações relacionadas à esse propósito como quantidade de memória do objeto, largura de banda, uso de energia, ser adaptado para atividades bem definidas e ter alguma forma de inteligência, que é, em geral, a habilidade de receber e transmitir dados a partir de sensores embarcados. Em resumo, entre os elementos principais de um objeto conectado, pode-se listar: receber, gerar, transmitir e armazenar dados; algoritmo para processar dados; o ecossistema com o qual irá reagir e integrar (fonte).

Do ponto de vista da comunicação entre objetos, cada um deve ter um ou mais ID para ser reconhecido por outro e, dessa forma, estabelecer conexão. O sistema GS1 propõe, por exemplo, uma tecnologia baseada em tags RFID que associam unicamente uma URL a uma informação logística relacionada a um objeto (GS1, 2019). A empresa Google tem proposto o projeto *"Physical Web"*[1] que associa uma URL única a um objeto. Por trás dessas tecnologias existe a luta por normas e padrões para IoT entre grandes companias de internet, pois cada uma deseja impor suas próprias tecnologias.

[1] https://google.github.io/physical-web/

2.2.1 Definição de Internet das Coisas

O termo IoT foi usado pela primeira vez em 1999 por Kevin Ahston, cofundador de MIT's Auto-ID Center e importante na criação do padrão RFID. O termo em si faz menção à possibilidade de quaisquer objetos, dispositivos e sensores estarem conectados à internet.

O CERP-IoT (*Cluster of European Research Projects on the Internet of Things*) define internet das coisas como:

> Uma infraestrutura dinâmica de uma rede global. Essa rede global possui capacidade de auto configuração baseada em padrões e protocolos de comunicação de interoperabilidade. Nesta rede, objetos físicos e virtuais tem identidades, atributos físicos, personalidades virtuais, interfaces inteligentes e são integrados à rede de uma forma transparente.

A definição anterior apresenta uma característica temporal e outra espacial relacionada à IoT, que permite que pessoas se conectem de qualquer lugar em qualquer momento a partir de objetos.

Entre as aplicações com capacidades de afetar diretamente o dia a dia das pessoas, têm-se:

- Saúde e sistemas de monitoramento remoto para ajudar pessoas;
- Otimização do uso da água na agricultura;
- Veículo conectado para ajudar o gerenciamento do trânsito
- Relógios conectados podem ajudar na promoção do bem estar das pessoas;
- Otimização de consumo e distribuição de energia elétrica

Diante da expectativa dos rumos que esse novo paradigma pode tomar, há também perspectivas relacionadas a um novo tipo de mercado capaz de criar novas tendências e novos tipos de trabalho.

2.2.2 IoT e a "nuvem"

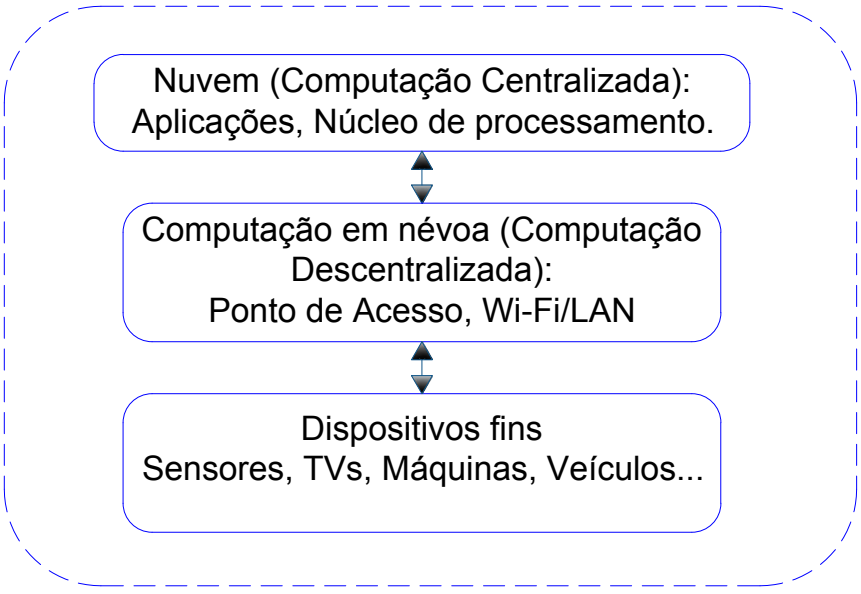
A computação em nuvem, devido às suas capacidades de processamento e armazenamento sob demanda, pode ser usada para analisar dados gerados por objetos IoT. O modelo "pague conforme o uso", adotado por muitas empresas oferecem serviços em nuvem, reduziu consideravelmente o preço do armazenamento de dados e processamento dos mesmos, estimulando, em certa medida, o aumento de aplicações com IoT. Com a elasticidade da nuvem, os mecanismos de processamento de fluxo distribuídos (SPEs) podem implementar recursos importantes, como tolerância a falhas e escalonamento automático para cargas de trabalho em rajadas (BUYYA; DASTJERDI, 2016).

2.2.3 Análise em tempo real com IoT e "computação em névoa" (*fog computing*)

Diante do paradigma de IoT, muitos dos dados coletados precisam ser analisados o mais próximo possível do dispositivo para então se gerar ações rápidas de processamento. Em muitas aplicações não se pode esperar por uma conexão de rede para acessar a nuvem, e a partir da resposta da mesma executar alguma tarefa. Por exemplo, para um veículo conectado identificar um sinal de trânsito indicado "pare"ou um pedestre prestes a atravessar uma faixa de segurança exige uma ação praticamente instantânea, é interessante que esse processamento aconteça no carro, não na nuvem. A depender da aplicação, muitos dados precisam ser processados em roteadores, *switches*, pontos de acesso sem fio, câmeras de segurança e servidores (CISCO, 2015).

Por isso, assim como a ideia de *edge computing*, surge o conceito de "computação em névoa", que distribui os recursos e os serviços de processamento, comunicação, controle e armazenamento para mais perto dos dispositivos. A Figura 1 ilustra em forma de blocos a arquitetura típica para computação em névoa.

Figura 1 – Arquitetura típica de *fog computing*



Fonte: Próprio Autor

2.2.3.1 Benefícios da computação em névoa (*fog computing*)

De acordo com a *Cisco* (CISCO, 2015), a extensão do conceito de nuvem para mais próximo das coisas que geram e atuam nos dados traz em si alguns benefícios:

- Maior agilidade nos negócios: com as ferramentas certas, os desenvolvedores podem desenvolver rapidamente aplicativos de névoa e implantar-los quando necessário.
- Melhor segurança: proteção dos nós em névoa podem usar as mesmas soluções de segurança física e segurança cibernética já bem definidas para outras soluções de TI.
- Controle de privacidade: a análise dos dados confidenciais pode ser feita localmente, em vez de enviá-los para a nuvem. Uma equipe ou responsável técnico de TI, de confiança do proprietário, pode monitorar e controlar os dispositivos que coletam, analisam e armazenam dados.
- Menor custo operacional: Menor uso da largura de banda da rede processando os dados selecionados localmente.

A Tabela 1 mostra as características da computação em névoa e em nuvem para aplicações com Internet das Coisas.

Tabela 1 – Características de computação em névoa e em nuvem para IoT.

	Nós próximos de dispositivos IoT (*fog computing*)	Nuvem
Tempo de resposta	Milesegundos	Minutos, dias, semanas
Exemplo de aplicações	Comunicação M2M; Telemedicina	Análise massiva de dados
Quanto tempo dados IoT ficam armazenados	Passageiro (horas ou dias)	Meses ou anos
Cobertura Geográfica	Local (Exemplo: uma cidade)	Global

Fonte: Adaptado de (CISCO, 2015)

2.2.4 Comunicação entre dispositivos IoT

O modelo "*publish/subscribe*" é uma maneira simples de trocar mensagens em ambientes distribuídos e, devido a simplicidade, tem sido adotado o popular protocolo de comunicação baseado em M2M: MQTT. Pois, em um cenário dinâmico, onde nós se conectam e se desconectam da rede frequentemente e, mesmo assim, as transferências de mensagens precisam ser bem sucedidas, esse modelo mostra-se eficiente (BUYYA; DASTJERDI, 2016).

Em certa medida, há uma forte tendência do protocolo MQTT se consolidar, entre outras alternativas, como o mais viável para melhor prover comunicação entre dispositivos IoT. Essa possível fatalidade ocorre porque MQTT é mais leve, por exemplo, que o protocolo HTTP 1.1 e, portanto, é uma opção muito interessante quando é preciso enviar e receber dados em tempo quase real. Além de já ser popular em IoT e projetos embarcados, também está ganhando espaço em aplicação web e aplicativos móveis (HILLAR, 2017).

Na Tabela 2 há uma comparação entre os principais protocolos de comunicação para IoT.

Tabela 2 – Principais protocolos de comunicação para IoT

Comparação Entre Protocolos					
Protocolo	Protocolo de Transporte	Modelo da Mensagem	Segurança	Caso de melhor experiência	Arquitetura
AMPQ	TCP	Publicar/Assinar	Alta / Opcional	Integração de empresas	P2P
CoAP	UDP	Requisição/ Resposta	Média / Opcional		Árvore
DDS	UDP	Publicar/Assinar e Requisição/Resposta	Alta / Opcional	Forças Armadas	Barramento
MQTT	TCP	Publicar/Assinar e Requisição/Resposta	Média / Opcional	Mensagens IoT	Árvore
UPnP	-	Publicar/Assinar e Requisição/Resposta	-	Consumidor	P2P
XMPP	TCP	Publicar/Assinar e Requisição/Resposta	Alta / Opcional	Gerenciamento remoto	Cliente-Servidor
ZeroMQ	UDP	Publicar/Assinar e Requisição/Resposta	Alta / Opcional	CERN	P2P

Fonte: Adaptado de (BUYYA; DASTJERDI, 2016)

2.3 Protocolo MQTT

Embora já mencionado nas seções anteriores, nesta seção serão apresentados com mais detalhes algumas definições e esclarecimentos acerca do protocolo MQTT. Em sua concepção, buscou-se atingir alguns objetivos, tai como: ser de fácil implementação; fornecer qualidade de serviço; ser leve e usar a largura de banda eficientemente; lidar com questões de segurança; trabalhar com dados de diferentes. Na página oficial do organização

[2] há o seguinte conceito sumarizado do MQTT:

> MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging "machine-to-machine" (M2M) or "Internet of Things" world of connected devices, and for mobile applications where bandwidth and battery power are at a premium

O MQTT foi um protocolo interno e proprietário da IBM por muitos anos até ser lançado na versão 3.1 em 2010 como um produto livre de *royalties*. Em 2013, MQTT foi padronizado e aceito no consórcio OASIS[3]. Em 2014, o OASIS lançou-a publicamente como versão MQTT 3.1.1. O MQTT também é um padrão ISO (ISO / IEC PRF 20922).

2.3.1 Modelo de mensagem "Publicar/Assinar" (*Publish/Subscribe*)

Como já visto na Tabela 2, os modelos de publicação e assinatura representam uma alternativa útil para IoT. Também conhecido como modelo *pub/sub*, é uma maneira de dissociar um cliente transmitindo uma mensagem de outro cliente que recebe uma mensagem. MQTT possui arquitetura *pub/sub*, mas não é uma fila de mensagens. No MQTT, se ninguém está inscrito em um tópico, as mensagens destinadas para ele são simplesmente ignoradas e perdidas.

Um cliente transmitindo uma mensagem é chamado de *publisher* (publicador), o que recebe uma mensagem é chamado de *subscriber* (Assinante). No centro há um servidor MQTT (*Broker*) que tem a responsabilidade de conectar clientes e filtrar dados. Esses filtros fornecem:

Filtragem de assunto: os clientes assinam tópicos. Cada mensagem publicada deve conter um tópico e o broker é responsável pela retransmissão da mensagem para clientes inscritos nesse tópico.

Filtragem de conteúdo: os brokers têm a capacidade de inspecionar e filtrar os dados publicados. Assim, quaisquer dados não criptografados podem ser gerenciados pelo intermediário antes de serem armazenados ou publicado para outros clientes.

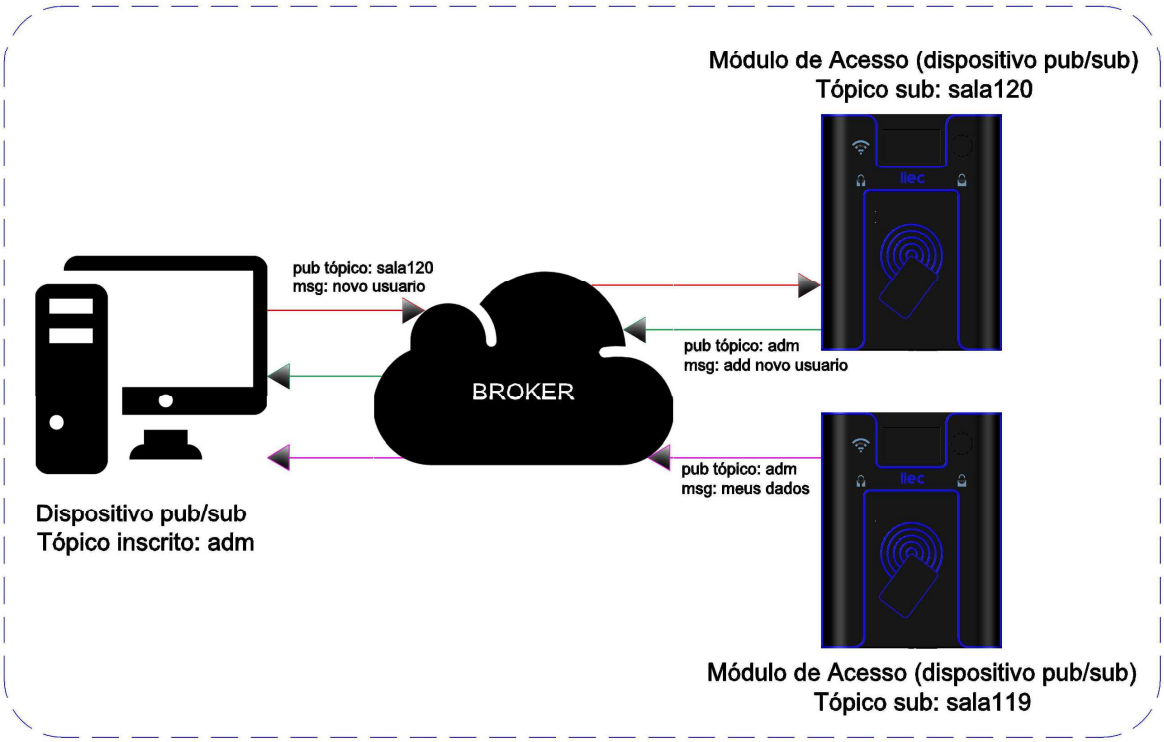
Tipo de filtragem: um cliente que está esperando um fluxo de dados também pode aplicar seus próprios filtros. Os dados recebidos podem ser analisados e o fluxo de dados podem ter um processamento a mais ou pode ser ignorado.

[2] http://mqtt.org/

[3] OASIS é um consórcio sem fins lucrativos que impulsiona o desenvolvimento, a convergência e a adoção de padrões abertos para a sociedade global da informação.

A Figura 2 ilustra como funciona o modelo *pub/sub*. Nessa figura, há três dispositivos conectados ao *broker* MQTT e cada um assina um tópico. O dispositivo à esquerda (representado por um PC) publica a mensagem "novo usuário" para qualquer dispositivo que assina o tópico "sala120". As setas vermelhas indicam o fluxo dessa mensagem do dispositivo origem até o dispositivo final. De modo similar, caso haja necessidade de qualquer módulo de acesso enviar mensagens para o dispositivo à esquerda, eles devem publicá-las para o topíco "adm", conforme indicam o fluxo das mensagens das setas em verde e em magenta..

Figura 2 – Comunicação MQTT.



Fonte: Próprio Autor

Nota-se, portanto, que esse protocolo separa os clientes que publicam mensagens para um tópico dos que assinam para o mesmo tópico. Como o *broker* gerencia a troca de mensagens, não há necessidade de um cliente identificar diretamente outro cliente baseado em aspectos físicos como, por exemplo, via endereço IP. MQTT e outros modelos *pub/sub* também são invariantes ao tempo. Isso implica que uma mensagem publicada por um cliente pode ser lida e respondida a qualquer momento por um assinante. Pois um assinante pode estar em um estado de baixa energia ou largura de banda limitada e publicar minutos ou horas depois para um tópico. Por causa da falta de interdependência físicas e temporais, os modelos *pub/sub* são muito bem escaláveis.

2.3.2 Estrutura de pacotes MQTT

Um pacote MQTT fica na parte superior da camada TCP da pilha de rede do modelo OSI. O protocolo funciona trocando uma série de pacotes de controle de forma bem definida, os quais são formados por um cabeçalho fixo de 2 bytes, que deve estar sempre presente, um cabeçalho de tamanho variável (opcional) e conclui com a mensagem (*payload*) (novamente, opcional). A Figura 3 mostra a estrutura do cabeçalho fixo que deve estar sempre presente no envio dos pacotes.

Figura 3 – Formato do cabeçalho fixo em um pacote de dados MQTT.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			

Name	**Value**	**Direction of flow**	**Description**
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

MQTT Control Packet	**Fixed Header flags**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT v5.0	DUP	QoS		RETAIN
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0
AUTH	Reserved	0	0	0	0

Fonte: Próprio Autor

2.4 Sistema RFID

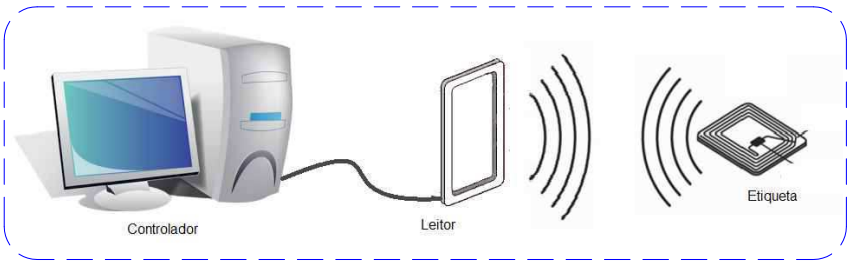
RFID (*Radio-Frequency IDentification*) ou identificação por radiofrequência é um sistema composto basicamente por dois componentes:

- **Transponder** (ou tag/etiqueta). Localizado no objeto a ser identificado;
- **Leitor**. Este dispositivo, a depender do projeto ou da tecnologia usada, pode ser um dispositivo de leitura ou escrita/leitura.

Existem dois tipos básicos de etiquetas ou tags RFID. O primeiro tipo não possui fonte de alimentação própria, ditas etiquetas "passivas". A energia necessária para envio de seus dados é obtida a partir do campo eletromagnético do leitor RFID. Em geral, possuem informações gravadas de forma permanente, mas alguns modelos permitem gravação ou regravação de dados. O segundo tipo, por sua vez, são etiquetas "ativas", isto é, possuem alimentação própria e isso possibilita a transmissão de seus dados a uma distância maior, quando comparadas às etiquetas passivas, do leitor RFID.

Um exemplo básico de um sistema RFID é mostrado na Figura 4. Um leitor normalmente contém um módulo de radiofrequência (transmissor e receptor), uma unidade de comando e um elemento de acoplamento ao transponder. Além disso, muitos leitores são equipado com uma interface adicional para permitir o encaminhamento os dados recebidos a outro sistema (PC, microcontroladores, etc.). O transponder, que representa o dispositivo de transporte de dados real de um sistema RFID, normalmente consiste em um elemento de acoplamento e um microchip eletrônico (FINKENZELLER; MÜLLER, 2010).

Figura 4 – Sitema RFID.



Fonte: https://www.gta.ufrj.br/grad/15_1/rfid/tecnologia.html

Assim como qualquer outra tecnologia de telecomunicações, apenas algumas faixas do espectro de frequência são liberadas para uso da tecnologia RFID. A Tabela 3 sumariza alguns tipos de aplicações para as frequências correspondentes permitidas para aplicação da identificação por radiofrequência.

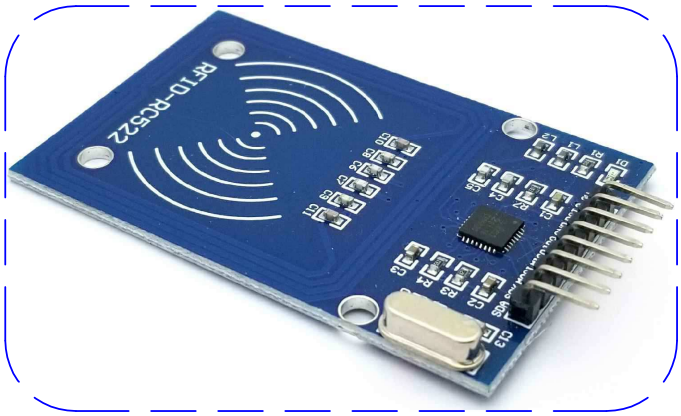
Tabela 3 – Faixas de frequências utilizadas em RFID.

Frequência	Aplicações
125 - 148 kHz	Identificação de animais (de estimação ou em pecuária)
13,56 MHz	Identificação de livros em bibliotecas, roupas
915 MHz	Cadeias de suprimentos: caixas, paletes, contêineres
2,45 GHz	Pedágios em rodovia, frota de veículos

Fonte: Adaptado de (LEHPAMER, 2008)

Entre os dispositivos eletrônicos já disponíveis no mercado para aplicações de RFID na frequência de 13,56 MHz, o circuito integrado MFRC522 fabricado pela *NXP Semiconductors* tem se destacado. Possui alimentação típica de 3,3 V, saída para antena e suporta interface de comunicação via SPI, I_2C e UART (NXP SEMICONDUCTORS, 2016). Essas funcionalidades facilitam os desenvolvimentos de projetos que integram esse leitor RFID com plataformas microprocessadas como Arduino, NodeMCU, Raspberry Pi, etc. A Figura 5 mostra uma placa típica construída com esse leitor e com o respectivo projeto de antena transceptora.

Figura 5 – Placa RFID - RC522.



Fonte: Próprio Autor

2.5 Hardware e soluções para IoT

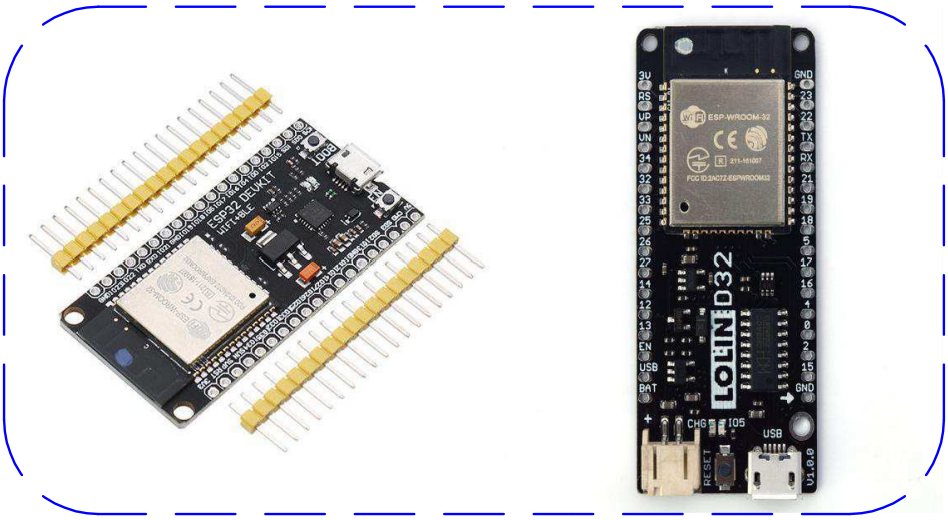
Há muitas frentes de pesquisas aprimorando soluções de rede para IoT. Entre elas, existe a associação para desenvolvimento de protocolos para soluções baseadas em LoRa (*LoRa Alliance*), que usa técnica de modulação por espalhamento espectral para se atingir longo alcance e baixo consumo de energia[4] (SEMTECH, 2019).

[4] LoRa é um acrônimo para *LOng RAnge*. Técnicas de espalhamento espectral são utilizadas em redes locais sem fios Wi-Fi 802.11b, 802.11g 802.11n e na tecnologia 3G.

A operadora de rede SigFox também atua no mercado construindo redes para conectar dispositivos de baixa potência e vem se mostrando forte competidora nesse novo modelo de mercado oriundo das perspectivas de IoT. Ela emprega a técnica de modulação por chaveamento por deslocamento de fase diferencial binário (DBPSK) e a gaussiana por chaveamento de frequência (GFSK) que permite a comunicação usando a banda de rádio Industrial que usa 868 MHz na Europa e 902 MHz nos EUA. Ela utiliza um sinal de longo alcance que passa livremente através de objetos sólidos, chamado de "Ultra Banda Estreita" e requer pouca energia, sendo denominado "Rede de área ampla de baixa potência (LPWAN)"(DREGVAITE; DAMASEVICIUS, 2016). O sinal também pode ser usado para cobrir facilmente grandes áreas e alcançar objetos subterrâneos (AGHA; PUJOLLE; YAHIYA, 2016).

Por serem, em geral, *open-source* e mais acessíveis para qualquer classe de desenvolvedor, algumas plataformas de desenvolvimento para IoT rapidamente tornaram-se populares. Como é o caso do kit de desenvolvimento de software baseado no chip ESP8266[5], NodeMCU. Em geral as placas de desenvolvimento são *open-hardware* baseadas em chip da família ESP e integram GPIO, PWM, ADC, comunicação I_2C, SPI entre outras funcionalidades em uma única placa. Empresas como Wemos[6] também oferece uma série de placas para IoT baseadas em chips da família ESP. A Figura 6 mostra placas da empresa NodeMCU e da empresa Wemos.

Figura 6 – À esquerda: placa NodeMCU32S. À direita: placa Wemos LOLIND32



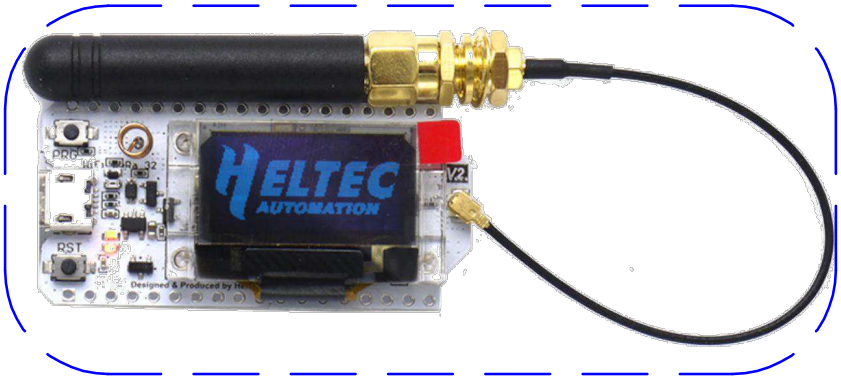
Fontes: (NODEMCU, 2019), (WEMOS, 2019)

A empresa *Heltec Automation*[7] uniu as potencialidades do ESP-32 com a tecnologia de rede LoRa e oferece, além de outras soluções, uma placa para aplicações IoT com

[5] Chip produzido pela empresa Espressif, teve a produção foi iniciada no fim de 2013
[6] https://wiki.wemos.cc/doku.php
[7] https://docs.heltec.cn/#/en/

conexãoWi-Fi, Bluetooth e LoRa. Essa placa está iluistrada na Figura 7.

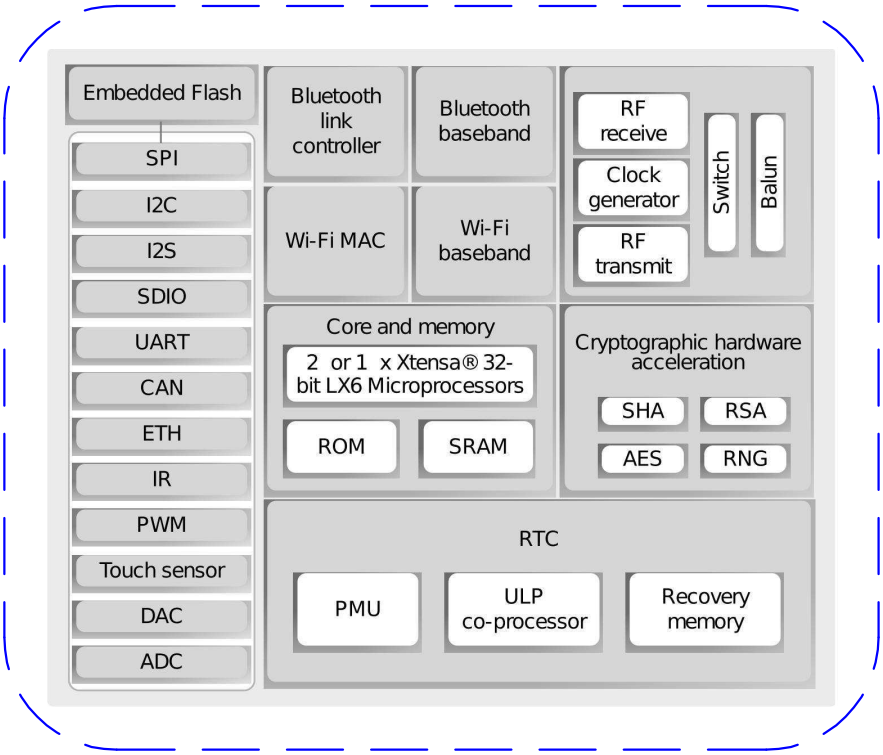
Figura 7 – Placa de desenvolvimento LoRa32.



Fonte: (HELTEC, 2019)

A Figura 8 mostra o diagrama de blocos funcional da arquitetura do ESP-32. A partir dessa imagem é possível compreender os tipos e a quantidade de recursos que garantem a potencialidade desse chip para desenvolvimentos de projetos no contexto de IoT.

Figura 8 – Arquitetura do chip ESP32.



Fonte: (ESPRESSIF, 2019)

3 Medodologia

O foco inicial do trabalho foi a revisão bibliográfica apresentada na seção anterior e, partindo-se disso, conseguiu-se pesquisar e definir algumas ferramentas que foram utilizadas para o desenvolvimento deste projeto. Sob um ponto de vista mais panorâmico, as atividades foram divididas em três principais vertentes: o desenvolvimento do software administrador, a eletrônica e a definição e modelagem da estrutura mecânica do produto. A Figura 9 ilustra de forma sistêmicas as subetapas para cada vertente supracitada.

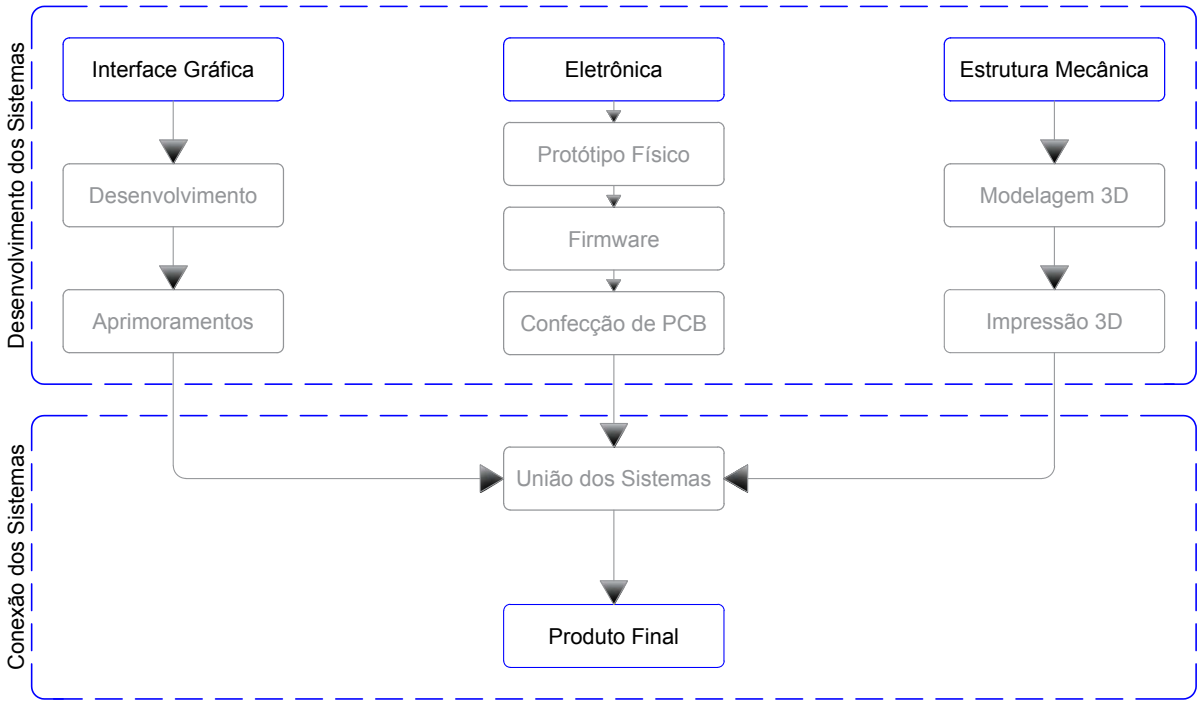


Figura 9 – Divisão sistêmica do projeto para a sua execução.

Fonte: Próprio Autor

3.1 Interface Gráfica

Estabeleceu-se previamente que o desenvolvimento do software administrador seria dividido em duas etapas principais de acordo com a seguinte ordem de prioridade:

1. Janela para configuração inicial de um dispositivo de acesso IoT;
2. Janela para adicionar e remover usuários (e suas respectivas TAGs RFID), bem como visualizar detalhes e a quantidade de usuários já inseridos no sistema;

A solução adotada para o desenvolvimento do software administrador baseia-se na linguagem de programação *Processing*, *open source*, adequada para o desenvolvimento de interfaces gráficas junto à agregação de muitas tecnologias(REAS; FRY, 2014). Além das bibliotecas nativas para desenhos, que são baseadas em Java, utilizou-se mais duas importantes bibliotecas: uma para comunicação serial e outra para comunicação com o servidor.

Em *Processing*, escrevendo-se os programas em modo Java, ao se utilizar o comando *"import processing.serial.*;"* carrega-se a biblioteca que viabiliza a escrita e a leitura de dados via porta serial entre o computador que executa o programa e eventuais dispositivos externos. De modo similar, o comando *"import mqtt.*;"* carrega a biblioteca cujas funções permitem estabelecer a comunicação com base no protocolo MQTT entre o programa em execução e um servidor (Broker). Neste último caso, faz-se necessário instalar a biblioteca previamente pelo gerenciador de bibliotecas do programa. A Figura 10 mostra a primeira janela do software administrador desenvolvida ao longo do trabalho. A partir dela configura-se inicialmente o módulo de acesso IoT a ser instalado para acionar ou não a abertura de uma porta. O software encarrega-se de identifica os canais de comunicação disponíveis (portas COM), lista-os para que o administrador selecione a porta correta e, dessa forma, estabelecer a comunicação serial entre o computador e o módulo de acesso. Em seguida, faz-se necessário digitar dados de configuração e enviá-los para o dispositivo, conforme mostra a Figura 11. Após essa etapa de configuração, toda a comunicação entre os módulos de acesso e o software é feita via protocolo MQTT.

Cada fechadura IoT irá se inscrever (ou assinar) em um único tópico que, de certa forma, neste projeto, isso também o identifica dentro da rede. O software administrador também é um cliente MQTT e irá assinar um único tópico. Os diferentes tipos de mensagens que são trocadas e que indicam diferentes tipos de tarefas são diferenciadas por identificadores - embutidos na própria mensagem - pré determinados e conhecidos por todos os dispositivos que integram o sistema.

Então, por exemplo, se a fechadura da sala 101 assina o tópico "*sala101*", logo, qualquer mensagem enviada pelo *broker* para essa fechadura específica deve ser endereçada para esse tópico. A comunicação no sentido inverso acontece de forma análoga. O corpo da mensagem, por sua vez, definido para os propósitos deste projeto, tem a seguinte formatação: "*identificador+mensagem*". O sinal "+" é utilizado na composição da mensagem para separar as diferentes strings que a compõe e isso facilita a forma como o algoritmo quebra essa mensagem para posterior tratamento de dados. Por isso, vale ressaltar que não é aconselhável configurar dispositivos para este sistema utilizando-se o caractere "+". Os identificadores servem para determinar um tipo de atividade a ser executada tanto pelo hardware terminal quanto pelo software administrador em execução em algum PC. A Tabela 4 explicita as definições de identificadores e dos tópicos com os quais os dispositivos desse sistema estão necessariamente configurados.

Figura 10 – Interface para configuração do módulo de acesso IoT.

Fonte: Próprio Autor

Tabela 4 – Identificadores utilizados para o algoritmo selecionar tarefas a serem executas.

Identificador	Função a ser realizada	Mensagem formatada para envio
adds	Adiciona um novo usuário	adds+nome+CPF+tag
del1	Deleta um usuário	del1+CPF
delA	Deleta todos os usuários	delA+
fix1	Envia comando para abertura de emergência de alguma porta	fix1+sala/101

Fonte: Próprio Autor

A Figura 12 mostra a tela do sofrware para adicionar um novo usuário. Todos os dados de cadastro, bem como constantes informações enviadas pelos dispositivos são salvos no HD do computador, em arquivos com formato JSON. Esse formato é interessante por ser relativamente fácil de manipular usando-se as devidas bibliotecas em Java e possui um arranjo que facilita a leitura dos dados. Pode-se colocar em um único arquivo JSON uma lista com todos os usuários e há liberdade para incrementar qualquer item de interesse no escopo das variáveis ou da própria lista. No inicio cada um deles tem a seguinte configuração:

```
{
    "CPF": "00014000410",
    "nome": "Ravi Helon",
    "inativo": false,
    "id": 2,
    "tag": "U2 F4 G5 D8",
    "salas": ["liec/salaAAA", ..., "liec/salaXXX"],
    "registros": [{"data":
                   "hora":
                   },
                   {"data":
                    "hora":
                   },
                   .
                   .
                   .

                  ]
}
```

Figura 11 – Interface com exemplo de configuração do módulo de acesso IoT.

Fonte: Próprio Autor

Figura 12 – Interface para adicionar um novo usuário

Fonte: Próprio Autor

Em todo o período de projeto o software administrador foi desenvolvido e utilizado em ambiente Windows. No entanto, *Processing* é uma linguagem multiplataforma e, muito provavelmente, esta aplicação funcionará bem em ambiente Linux ou MacOS.

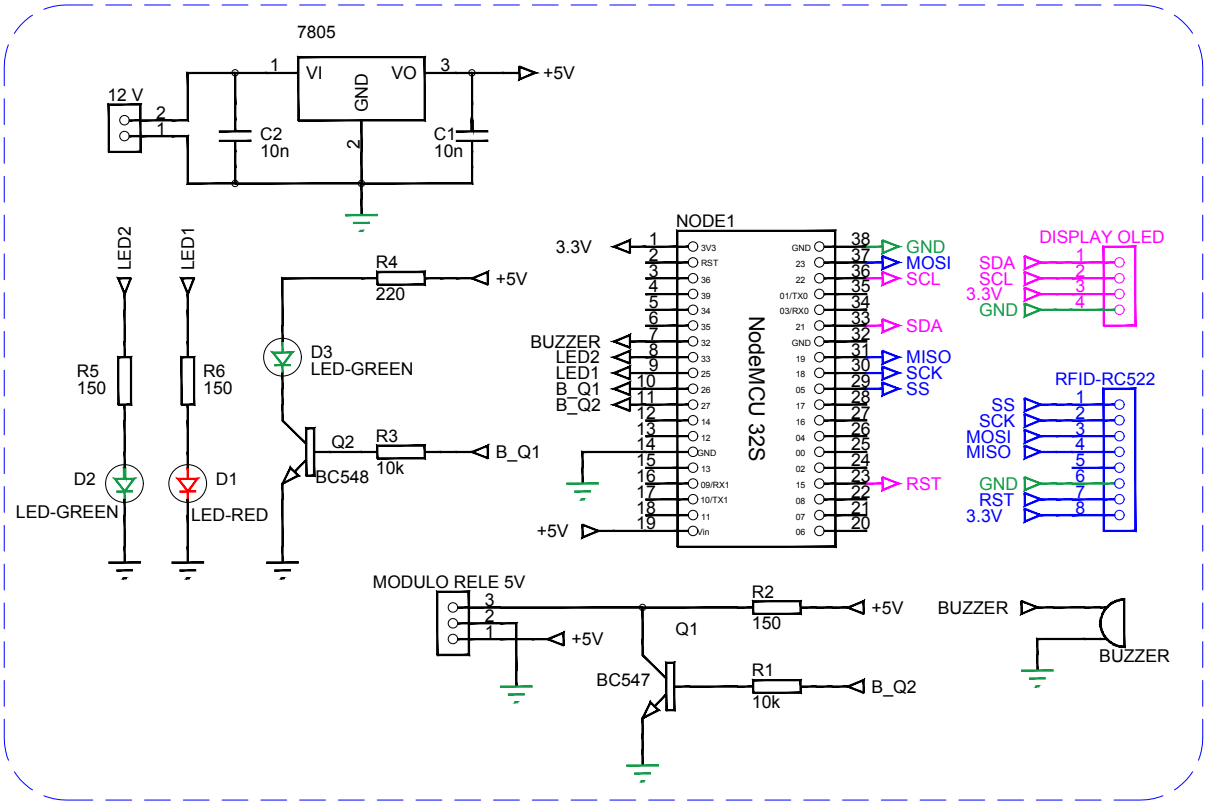
3.2 Eletrônica

O núcleo principal da solução de hardware para o módulo de acesso que aciona fisicamente a fechadura da porta é formado por uma placa da plataforma Node MCU, baseada no chip ESP-32 (ESPRESSIF, 2019), pelo *shield* RFID - RC522 e por um relé para atuar fechando ou não o circuito do solenoide da trava eletrônica.

Para interface homem-máquina do dispositivo terminal optou-se pelo uso de um display OLED de 0.9"além de LED indicadores de status para oferecerem validação visual opcional relacionada à conexão entre o dispositivo e a rede local e à permissão ou não de acesso ao ambiente na ocasião da leitura de alguma tag. Além desses dispositivos, há um buzzer para alerta sonoro. Todos esses elementos eletrônicos em conjunto servem para transmitir informações imediatas aos usuários do sistema.

O circuito formado por todos os dispositivos e módulos citados anteriormente bem como suas respectivas conexões pode ser visualizado em detalhes na Figura 13.

Figura 13 – Esquemático do Circuito.



Fonte: Próprio Autor

O relé é acionado a partir da aplicação de uma tensão de 5 V em seu terminal de *trigger*. A placa Node MCU é alimentada com 5 V, no entanto, o nível de tensão de trabalho do chip ESP-32 é de 3,3 V, logo, essa é a tensão máxima de saída de suas portas digitais. Por isso, como pode ser visto na Figura 13, faz-se necessário o uso de um transistor funcionando como chave para garantir o sinal de *trigger* do relé bem definido de 0 ou 5 V. Utilizou-se também um transistor para excitar o LED indicador de status de conexão Wi-Fi (LED D3). Essa escolha deveu-se ao fato de evitar ao máximo a drenagem da corrente através do chip ESP-32, pois, esse LED, diferentemente dos demais, muito provavelmente emitirá luz de maneira quase não intermitente.

De forma geral, não há muita complexidade associada à montagem do circuito eletrônico, pois, pode-se resumi-lo como a junção de soluções prontas que executam isoladamente tarefas, estas sim, de fato, não triviais. No entanto, de acordo com o que se venha propor para o funcionamento do dispositivo, principalmente no contexto de internet das coisas, o software embarcado pode ganhar níveis cada vez maiores de complexidade.

O *firmware* desenvolvido para este projeto está representado por um fluxograma

que pode ser visualizado na Figura 14. Baseia-se em instruções para que o ESP-32 permaneça em constante conexão com a rede de internet, integrando-se a um sistema distribuído cliente-servidor, do qual este dispositivo terminal é um cliente MQTT. Dentre os principais aspectos preocupantes que influenciaram o desenvolvimento do software embarcado para se ter o funcionamento contínuo do produto, têm-se:

1. A necessidade de configurá-lo previamente para o primeiro uso ou primeira conexão Wi-Fi. Ou seja, há um conjunto de instruções que deve ser executado uma vez quando algum administrador fizer essa pré configuração, que consiste basicamente em enviar dados para o ESP-32 com as informações acerca da rede local que ele deve se conectar (SSID e senha), do broker MQTT, do tópico que ele deve se inscrever bem como dados de autenticação para comunicação com o broker. Essa etapa é feita uma única vez via comunicação serial entre o dispositivo e o computador que executa o software administrador;

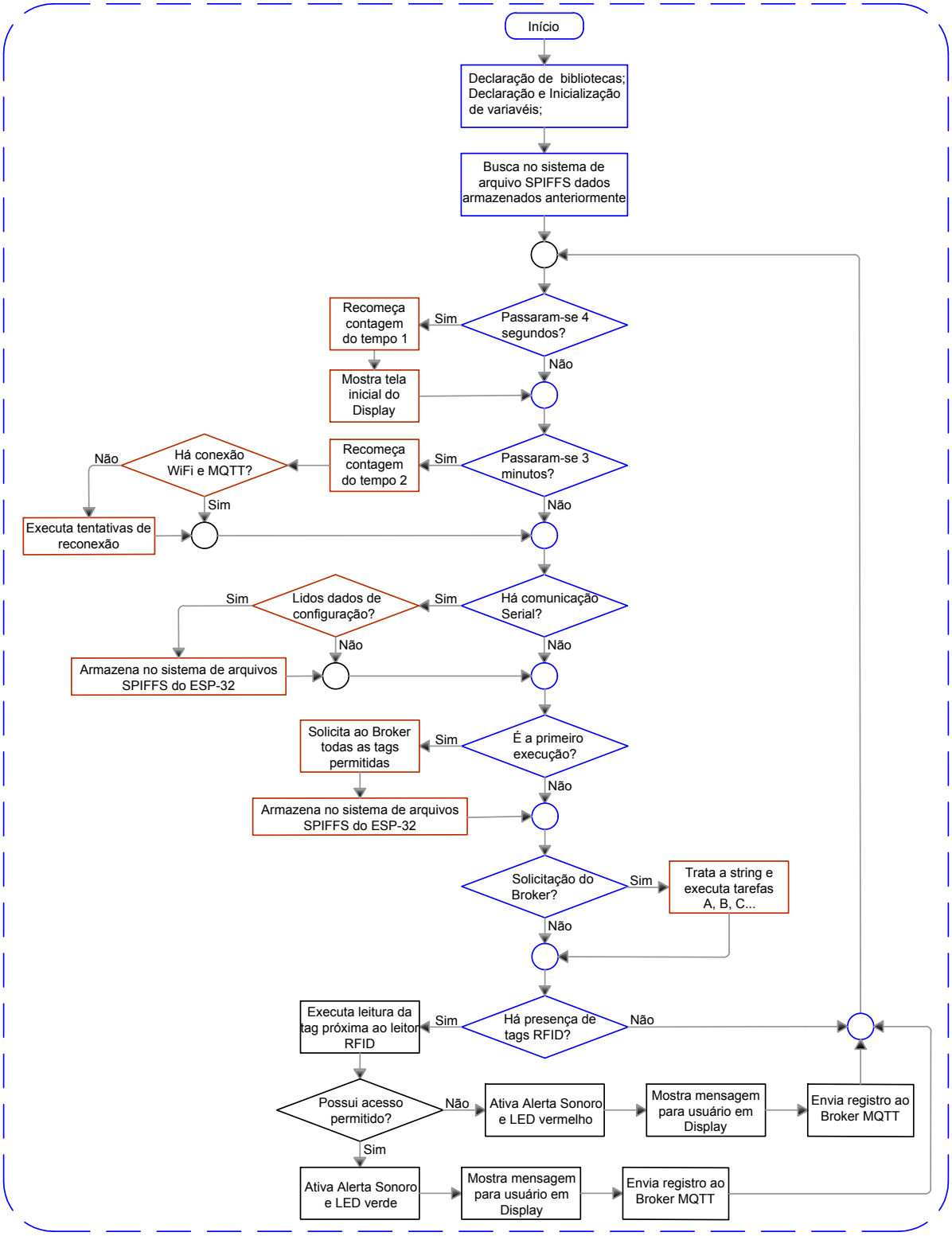
2. A manutenção do funcionamento do módulo de acesso quando o mesmo ficar temporariamente offline. Caso isso ocorra, ele deve utilizar o núcleo mínimo de dados (TAGs que possuem permissão de acesso, por exemplo) outrora salvos na memória permanente do ESP-32;

3. A necessidade de verificar a conexão com a internet de forma periódica (no intervalo estabelecido de 3 em 3 minutos) e sempre tentar reestabelecê-la automaticamente caso ocorra alguma falha. Durante a falha deve-se acionar um ou mais alertas visuais (LED e mensagem em display);

4. A necessidade de comunicação visual e sonora caso a tag lida pelo leitor RFID possua ou não permissão de acesso. No primeiro caso o acionamento sonoro deve ser realizado em uma frequência diferente, preferencialmente mais aguda, em relação ao mesmo tipo de alerta para o segundo caso;

A escolha pelo período de verificação ser a cada 3 minutos foi empírica. Em uma tentativa de reconexão com a rede de internet, o módulo de acesso permanece inativo para outras tarefas por aproximadamente 10 segundos. Caso a falha persista, ele volta a funcionar offline valendo-se dos dados previamente salvos em sua memória permanente e depois de 3 minutos fará outra tentativa de conexão. Durante esses 10 segundos não há lançamentos de exceções para leitura e verificação de qualquer tag que se aproxime do leitor RFID. É por isso que escolheu-se um período relativamente longo para tentativa de reestabelecer a conexão com a internet, pois busca-se diminuir a chance de algum usuário precisar usar o serviço de acesso exatamente no instante em que o módulo encontra-se inativo.

Figura 14 – Firmware para ESP-32.



Fonte: Próprio Autor

Tabela 5 – Bibliotecas utilizadas para escrita do software embarcado.

BIBLIOTECA	DESCRIÇÃO
SPI.h	Permite comunicação via protocolo SPI (*Serial Peripheral Interface*)
Wire.h	Permite comunicação via protocolo I2C/TWI
MFRC522.h	Biblioteca RFID para MFRC522 (SPI)
WiFi.h	Permite a conexão com a intenet
PubSubClient.h	Biblioteca para cliente MQTT
ArduinoJson.h	Permite manipulação de dados em uma estrutura JSON
SSD1306Wire.h	Prover API para uso de display monocromático/colorido OLED 128x64 pixels
FS.h e SPIFFS	Permite utilizar o sistema de arquivos SPIFFS em que torna possível manipulações básicas de arquivos (*Read, Write, Open e Close*)

Fonte: Próprio Autor

As bibliotecas mais importantes que foram usadas no desenvolvimento do *firmware* estão apresentadas na Tabela 5

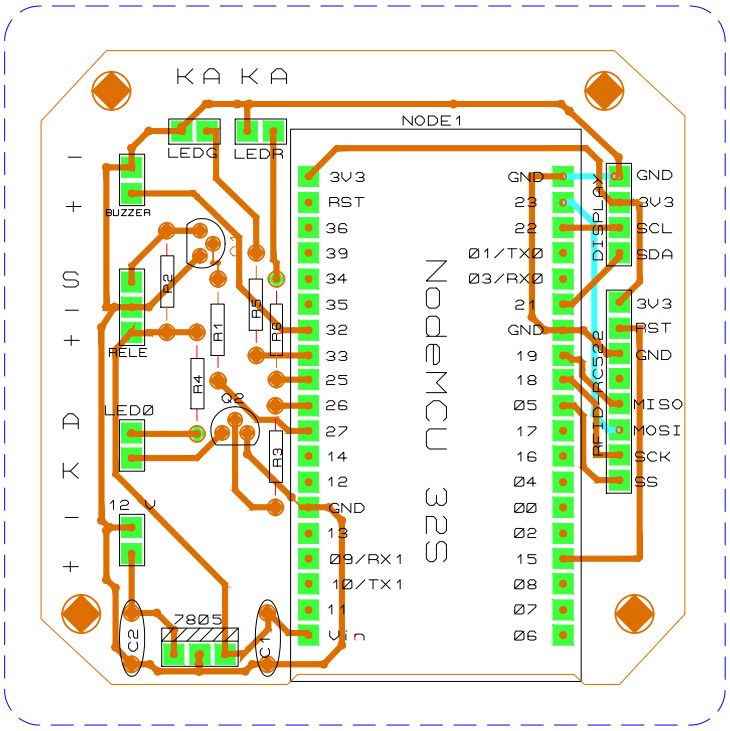
Escolheu-se o programa Proteus ARES para confecção da placa de circuito impresso (PCI). Quase todos os componentes eletrônicos propostos para este projeto possuem esquemáticos e *footprint* prontos nas bibliotecas padrões deste software. A Figura 15 ilustra o resultado final dessa etapa do projeto.

3.3 Modelagem da estrutura mecânica (carcaça do produto)

A primeira versão do produto proposto neste trabalho, dentro da fase de testes e posteriormente para uso permanente, será instalada para controle de acesso da sala 119 do prédio Embedded localizado no interior da Universidade Federal de Campina Grande - UFCG. Considerando-se, portanto, que usuários terão constantes experiências de uso com o dispositivo, buscou-se idealizar um design que não possibilite apenas um sistema funcional, mas que seu aspecto físico transmita a ideia de qualidade, isto é, ofereça algum impacto para quem olha devido a boa aparência além da usabilidade associada.

A partir disso, definiu-se algumas características para concepção dessa estrutura como cores, ergonomia, tipo de material a ser utilizado. O propósito é que, para além das características externas associadas ao design, em seu interior haja melhor acomodação das partes que compõem o circuito eletrônico além de blindá-los contra sujeira, poeira, umidade etc.

Figura 15 – Projeto da Placa de Cicuito Impresso - PCI

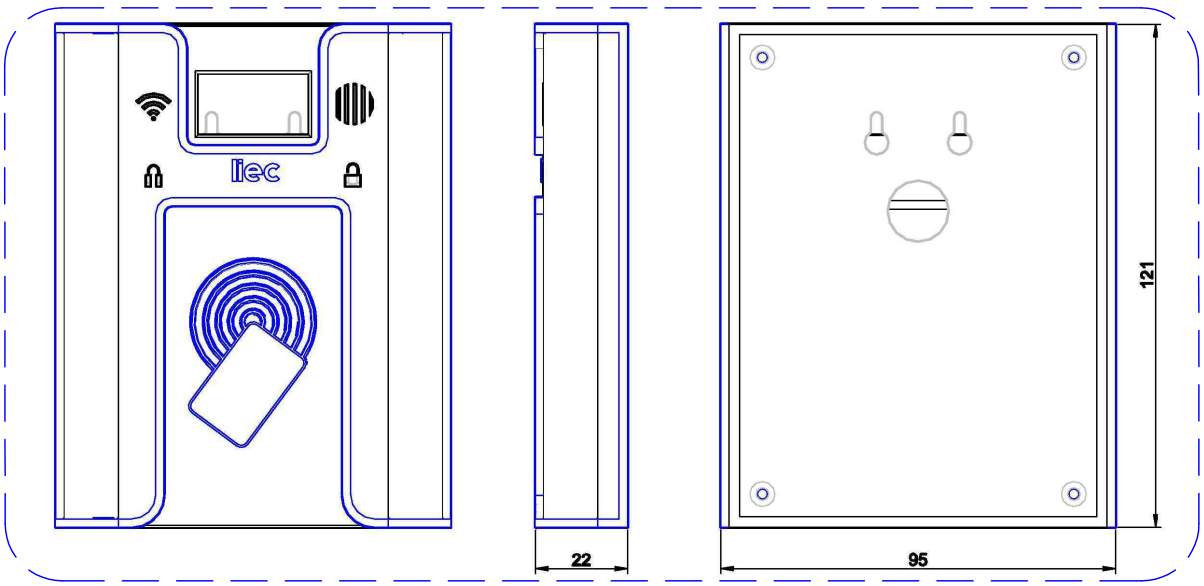


Fonte: Próprio Autor

Para o trabalho de modelagem utilizou-se o software AutoCAD 2018. O resultado deste item do projeto com suas devidas dimensões em milímetros pode ser visualizado na Figura 16.

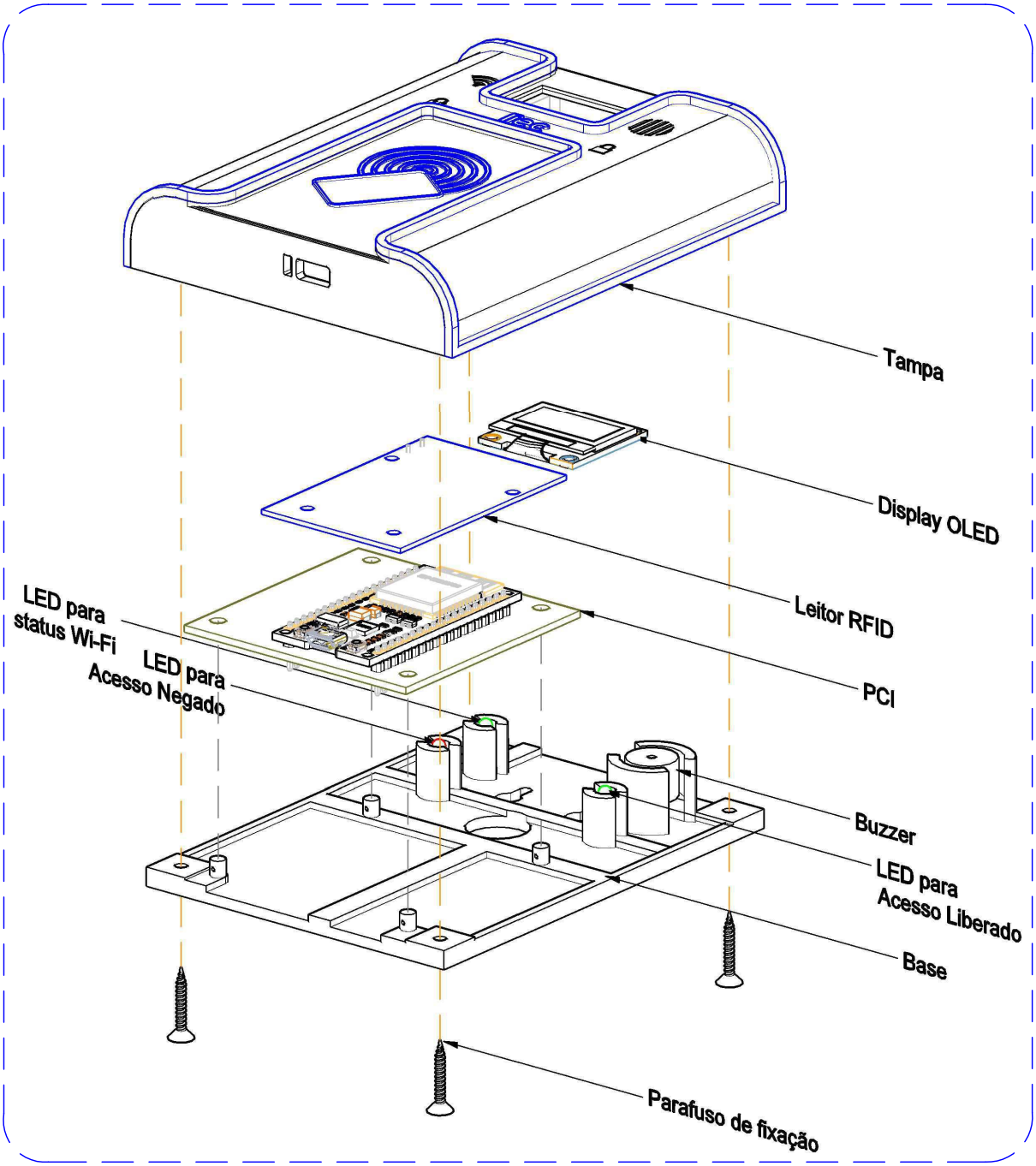
A Figura 17 apresenta com mais detalhes a disposição dos componentes eletrônicos no interior da estrutura externa. O leitor RFID e o diplay OLED possuem base de fixação na parte interna da tampa (não visível na imagem) e os outros componentes, conforme visto na imagem, têm elementos de apoio sobre a base.

Figura 16 – Projeto da carcaça do produto. Da esquerda para direita: visão frontal, lateral e traseira.



Fonte: Próprio Autor

Figura 17 – Partes integrantes do produto físico.



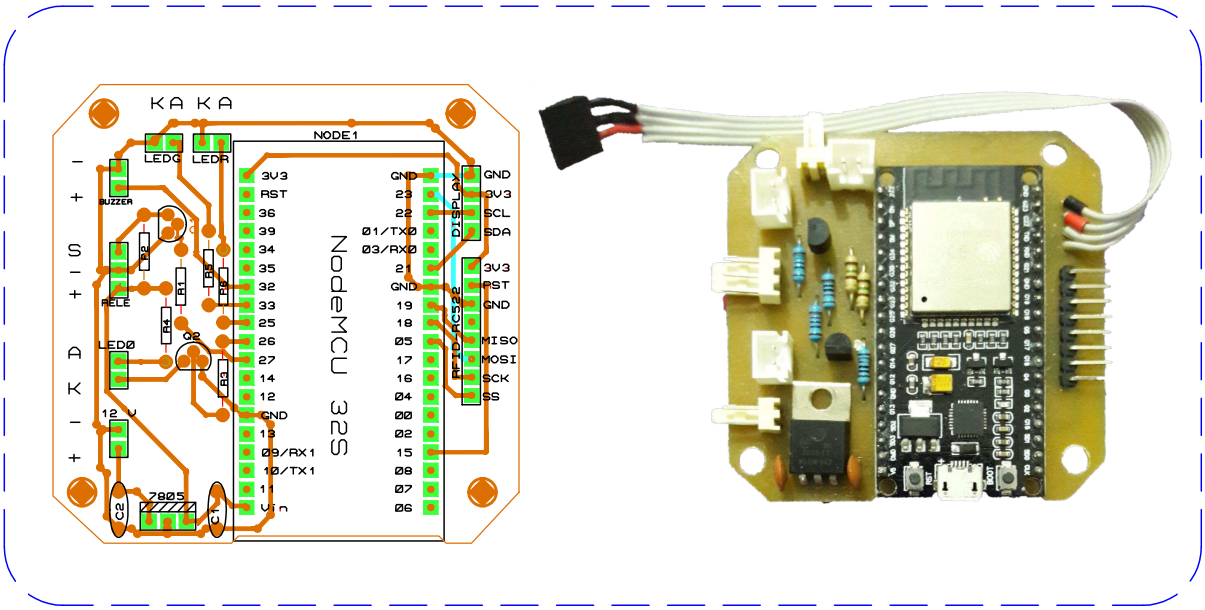
Fonte: Próprio Autor

4 Resultados Práticos

O projeto como um todo foi executado e colocado em prática no Laboratório do Departamento de Engenharia Elétrica - LIEC, localizado na UFCG. Para o cumprimento dos objetivos propostos, o *broker* utilizado não está na nuvem, e sim, instalado em um servidor localizado no mesmo prédio onde o módulo de acesso será instalado definitivamente. Utilizou-se algumas vezes, apenas para efeitos de testes, um *broker* público localizado na nuvem, acessível a partir do endereço *https://iot.eclipse.org/*. Embora alguns resultados já tenham sido apresentados na seção anterior, nesta parte será discorrido de forma objetiva tudo que foi possível tirar da fase de projeto e ser posto em prática. O que foi explicado e mostrado em relação ao software administrador já é em si o resultado do mesmo.

Em relação ao hardware, a Figura 18 ilustra lado a lado o projeto e a PCI confeccionada.

Figura 18 – Placa de Circuito Impresso Confeccionada.



Fonte: Próprio Autor

De modo similar, a Figura 19 mostra lado a lado a modelagem da carcaça do produto e sua respectiva impressão 3D (já com a inserção dos elementos do projeto eletrônico), feita em plástico PLA (Ácido Poliático). Apesar da impressão ter sido procedida com depósitos de material em camadas de 0.1 mm, ainda assim, elas se fazem bastante visíveis e comprometem, em certa medida, a sensação de produto "bem acabado"da peça. E para dar o encaminhamento final desta seção dedicada a apresentar os resultados práti-

cos, têm-se a Figura 20 que mostra a estrutura impressa após fase de pintura na tentativa de deixá-la similar ao modelo idealizado no projeto.

Voltando-se às vertentes iniciais que delinearam o modo como todo o projeto foi conduzido, representado por um organograma na Figura 9, vale enfatizar, a partir dessa mesma imagem reproduzida na Figura 21, quais subetapas podem ser consideradas "fechadas" e quais estão "abertas" para futuros projetos de melhoria e acréscimos de recursos.

Na Figura 9 todas as subetapas foram representadas por retângulos com tonalidade de cor cinza. Para efeito de comparação, aquelas que não possuem larga margem para melhorias, continuação ou modificação do projeto original estão, na Figura 21, contornadas de verde, caso contrário, estão contornadas de vermelho.

Nota-se a partir da Figura 21 que o software administrador é o único subproduto com muitas possibilidades de receber mais recursos e ser continuado melhorando este projeto como um todo. Os seus recursos atuais foram desenvolvidos para garantir o funcionamento básico do sistema. Como descrito anteriormente, é a partir dele que configura-se inicialmente o módulo de acesso terminal, cadastra-se ou não usuários via comunicação remota utilizando-se protocolo MQTT e se gerencia arquivos JSON onde ficam salvos informações gerais dos usuários do sistema. Ou seja, o software administrador atende aos requisitos mínimos para deixar todo o sistema funcional.

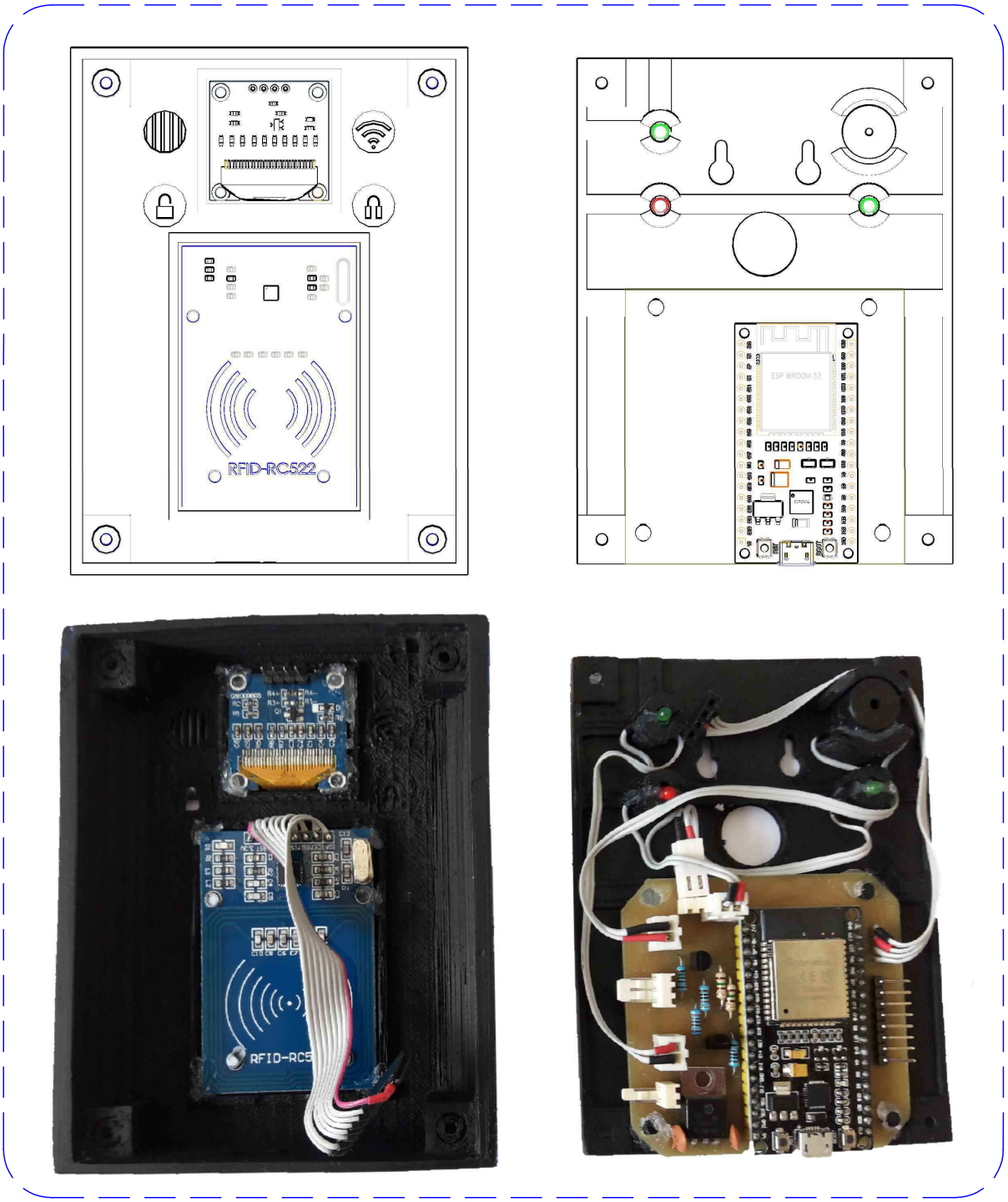
4.1 Custo financeiro associado ao projeto do módulo de acesso

Os custos financeiros associados a concepção física do módulo de acesso, usando-se como referência o ano de 2019 e considerando-se que todos os componentes foram comprados na cidade de Campina Grande - PB, estão apresentados na Tabela 6.

Tabela 6 – Custos associados à construção física do dispositivo de acesso.

Placas/Componentes	Preço Médio (ano de referência: 2019)
NodeMCU32S	R$ 45,00
Display OLED 128x64	R$ 29,00
RFID RC522	R$ 19,00
Módulo Relé 5V	R$ 6,00
Placa para PCI	R$ 10,00
Fio de solda	R$ 10,00
Confecção da PCI	R$ 40,00
Transistores/Conectores Resistores, LED etc.	R$ 3,00
Impressão 3D	R$ 40,00
TOTAL	**R$ 202,00**

Figura 19 – Impressão 3D da carcaça do dispositivo.

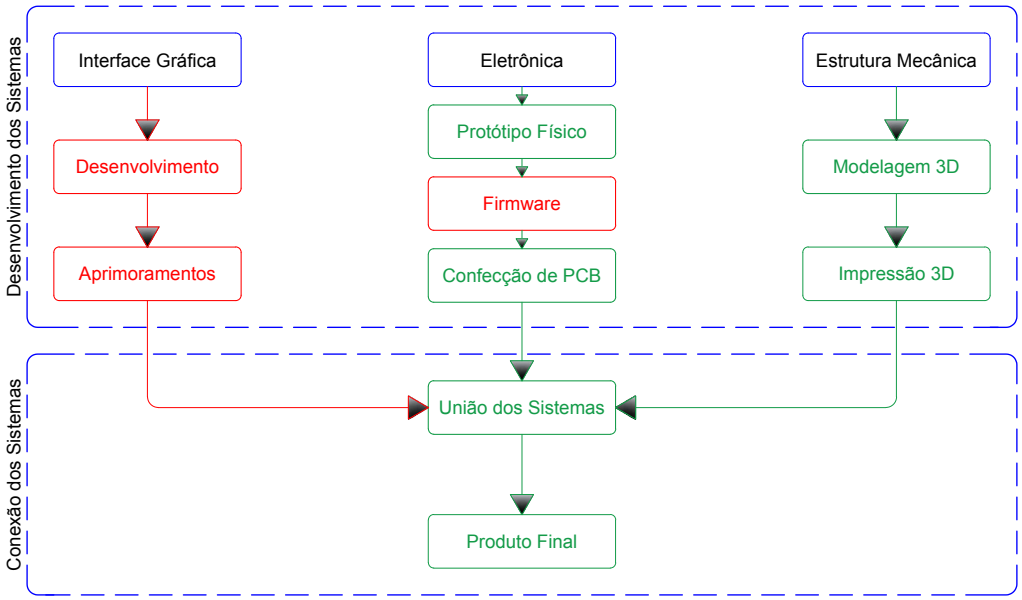


Fonte: Próprio Autor

Figura 20 – Visão frontal do dispositivos de acesso.

Fonte: Próprio Autor

Figura 21 – Etapas totalmente finalizadas: contornadas em cor verde. Etapas a serem continuadas: contornadas em cor vermelho.



Fonte: Próprio Autor

5 Conclusão

De forma geral, "êxito" é a palavra que resume razoavelmente bem os resultados deste projeto. Êxito por ter sido uma experiência enriquecedora, útil para a formação profissional na área de Engenharia Elétrica; por, ao fim, o produto desenvolvido ter passado pela agregação bem-sucedida de diferentes sistemas, ter utilidade prática e não ter ficado aprisionado no escopo da simulação.

Há, no entanto, coisas que podem ser melhoradas ou, mais apropriadamente, continuadas, principalmente no software administrador que também assume papel de servidor dentro do sistema, possui um banco de dados associado e enorme potencial para processar informações às margens dos dispositivos IoT (neste trabalho, os módulos de acesso). Por isso, essa aplicação gráfica foi estruturada para tornar viável o acréscimo modular de algoritmos caso surja o objetivo de expandi-lo sob qualquer perspectiva. Isso implica que o desenvolvimento deste software a partir da linguagem de programação e IDE *Processing* foi só um ponto de partida em relação a seu potencial inserido no contexto de Internet das Coisas e de "*fog computing*". Sobre esse horizonte, não há limites a não ser a própria criatividade aliada às necessidades que o produto se propõe a suprir. Em relação à experiência com o usuário do software de interface gráfica, pode ser interessante acrescentar a mesma, com intuito de melhorá-la, uma janela para mostrar a topologia do sistema e a planta baixa do prédio e outra aba de boas vindas a ser exibida no momento de inicialização do programa, além de mecanismos de busca, área para apresentação de dados e estatísticas.

O hardware, por sua vez, é um subsistema "fechado", isto é, para quaisquer perspectivas de alterações deve-se levar em consideração que o projeto eletrônico como um todo e, muito provavelmente, a estrutura mecânica precisarão ser refeitos. Em contrapartida, a ideia do *firmware* empregada na programação do modulo de acesso pode ser utilizada para outros dispositivos IoT que utilizam ESP-32. Neste contexto, de certo modo, o reuso do software embarcado facilitaria a integração de outros dispositivos IoT com outros tipos de demandas ao sistema do software administrador deste projeto, abrindo uma margem maior para mais trabalhos e desenvolvimentos em análise e processamento de dados.

É interessante e talvez importante relacionar os conhecimentos utilizados (já descritos ao longo das seções anteriores) para desenvolvimento do projeto com as principais disciplinas técnicas que foram estudadas durante a graduação em Engenharia Elétrica, a saber: Expressão Gráfica, Circuitos Elétricos I e II, Eletrônica, Dispositivos Eletrônicos, Introdução à programação, Técnicas de Programação.

Referências

AGHA, K.; PUJOLLE, G.; YAHIYA, T. **Mobile and Wireless Networks**. Wiley, 2016. 241 p. ISBN 9781119007555. Disponível em: <https://books.google.com.br/books?id=_BzfDAAAQBAJ>.

BUYYA, R.; DASTJERDI, A. **Internet of Things: Principles and Paradigms**. [S.l.: s.n.], 2016. 1-354 p.

CISCO. **White Paper: Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are**. [S.l.], 2015. Disponível em: <https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf>. Acesso em: 30 de maio de 2019.

DREGVAITE, G.; DAMASEVICIUS, R. **Information and Software Technologies: 22nd International Conference, ICIST 2016, Druskininkai, Lithuania, October 13-15, 2016, Proceedings**. Springer International Publishing, 2016. 665 p. (Communications in Computer and Information Science). ISBN 9783319462547. Disponível em: <https://books.google.com.br/books?id=ApcoDQAAQBAJ>.

ESPRESSIF. **ESP32**. 2019. Disponível em: <https://www.espressif.com/en/products/hardware/esp32/overview>.

FINKENZELLER, K.; MÜLLER, D. **RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication**. Wiley, 2010. ISBN 9781119991878. Disponível em: <https://books.google.com.br/books?id=jAszZEQYa9wC>.

GS1. **The Value and Benefits of the GS1 System of Standards**. 2019. Disponível em: <www.gs1.org>. Acesso em: 30 de maio de 2019.

HELTEC. 2019. Disponível em: <https://heltec.org/>. Acesso em: 30 de maio de 2019.

HILLAR, G. C. **MQTT Essentials - A Lightweight IoT Protocol**. [S.l.]: Packt Publishing, 2017. ISBN 9781787287815.

JIA, W.; ZHOU, W. **Distributed Network Systems: From Concepts to implementations**. [S.l.]: Springer International Publishing, 2005. (Network Theory and Applications). ISBN 0387238409.

LEHPAMER, H. **RFID Design Principles**. Artech House, 2008. 100 p. (Artech House microwave library). ISBN 9781596931947. Disponível em: <https://books.google.com.br/books?id=yg0fAQAAIAAJ>.

MAVROMOUSTAKIS, C.; MASTORAKIS, G.; BATALLA, J. **Internet of Things (IoT) in 5G Mobile Technologies**. Springer International Publishing, 2016. (Modeling and Optimization in Science and Technologies). ISBN 9783319309132. Disponível em: <https://books.google.com.br/books?id=Wj8GDAAAQBAJ>.

NODEMCU. 2019. Disponível em: <https://www.nodemcu.com/index_en.html>. Acesso em: 30 de maio de 2019.

NXP SEMICONDUCTORS. **MFRC522. Standard performance MIFARE and NTAG frontend**. 2016. Disponível em: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>. Acesso em: 30 de maio de 2019.

REAS, C.; FRY, B. **Processing: A Programming Handbook for Visual Designers and Artists**. The MIT Press; second edition edition, 2014. (The MIT Press). ISBN 9780262028288. Disponível em: <https://www.amazon.com/Processing-Programming-Handbook-Designers-Artists/dp/026202828X/ref=sr_1_6?s=books&ie=UTF8&qid=1406934187&sr=1-6&keywords=processing>.

SEMTECH. **LoRa Modulation Basics**. 2019. Disponível em: <https://www.semtech.com/uploads/documents/an1200.22.pdf>. Acesso em: 30 de maio de 2019.

WEMOS. 2019. Disponível em: <https://wiki.wemos.cc/start>. Acesso em: 30 de maio de 2019.