SICOMBO: sistema universal de IoT com computação de borda para seguro monitoramento clínico

Rafael Marinho e Silva

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Rafael Marinho e Silva

SICOMBO: sistema universal de IoT com computação de borda para seguro monitoramento clínico

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Shigueo Nomura, Ph. D
Coorientador: Prof. Flávio de Oliveira Silva, Ph. D

Uberlândia
2020

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S586 Silva, Rafael Marinho e, 1991-
2020 SICOMBO: sistema universal de IoT com computação de borda para seguro monitoramento clínico [recurso eletrônico] / Rafael Marinho e Silva. - 2020.

Orientador: Shigueo Nomura.
Coorientador: Flávio de Oliveira Silva.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.9
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Nomura, Shigueo,1968-, (Orient.). II. Silva, Flávio de Oliveira,1970-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 43/2020, PPGCO				
Data:	30 de dezembro de 2020	Hora de início:	15:15	Hora de encerramento:	17:15
Matrícula do Discente:	11812CCP026				
Nome do Discente	Rafael Marinho e Silva				
Título do Trabalho:	SICOMBO: sistema universal de IoT com computação de borda para seguro monitoramento clínico				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Pedro Frosi Rosa - FACOM/UFU; Fábio Carneiro Mokarzel - ITA; Flávio de Oliveira Silva - FACOM/UFU (coorientador) e Shigueo Nomura - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Fábio Carneiro Mokarzel - São José dos Campos/SP; Pedro Frosi Rosa, Flávio de Oliveira Silva e Shigueo Nomura - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Shigueo Nomura, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Pedro Frosi Rosa**, **Professor(a) do Magistério Superior**, em 11/01/2021, às 11:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Flávio de Oliveira Silva**, **Professor(a) do Magistério Superior**, em 11/01/2021, às 17:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Fabio Carneiro Mokarzel**, **Usuário Externo**, em 11/01/2021, às 21:35,

Agradecimentos

Agradeço primeiramente a DEUS! Por me sustentar nos momentos difíceis e me fortalecer para superar todos os meus desafios. Obrigado Senhor por todas as graças que Tu me concedes, por guiar meus passos, me abençoar com a dose necessária de sabedoria, inteligência e discernimento para concluir meus objetivos.

Aos meus pais Waldir e Maria Célia, meus irmãos Danielly e Gabriel, e meus sobrinhos Juninho, Davi e Helena por me amarem incondicionalmente e me apoiarem em todos os momentos, para que os meus objetivos fossem alcançados. Amo vocês!

À minha noiva Lara, agradeço pelo amor, pela compreensão e por ser meu porto seguro. Não consigo imaginar tudo que passei sem ter você ao meu lado. Te amo! Agradeço o incentivo da sua família, em especial aos meus sogros José Wilson e Eulália.

À Universidade Federal de Uberlândia, à Faculdade de Computação (FACOM), os servidores Erisvaldo e Sônia e todos os professores, que de forma significativa fizeram parte desta conquista em minha vida.

Ao meu orientador professor Dr. Shigueo Nomura pela orientação, auxílio e dedicação para realização deste trabalho. Por todo conhecimento e experiência de vida compartilhados comigo em diversas conversas, sempre me auxiliando e disposto a me ajudar sem medir esforços. Obrigado por acreditar no meu potencial e pela confiança depositada.

Ao meu coorientador professor Dr. Flávio de Oliveira Silva pela sua disponibilidade em contribuir e ajudar nos momentos críticos do meu trabalho.

Aos professores da banca examinadora, professor Dr. Fábio Carneiro Mokarzel e professor Dr. Pedro Frosi Rosa, por terem aceitado o convite e por ajudarem a engrandecer o meu trabalho, pela paciência e pelo tempo dedicado.

A todos os colegas que dividiram momentos únicos e importantes no laboratório e durante as aulas, em especial ao colega Rodrigo Moreira, pelo incentivo, apoio e parceria.

De forma geral, agradeço a todos os meus amigos e familiares, que direta ou indiretamente fizeram parte dessa etapa de minha vida. A contribuição de cada um foi única e insubstituível.

Resumo

Tradicionalmente, o monitoramento integral do quadro clínico de pacientes ocorre em ambientes hospitalares com equipamentos e profissionais especializados, exigindo altos investimentos financeiros. Isso limita o acesso de pessoas de baixa renda a tratamentos básicos em hospitais do SUS no Brasil. Contudo, os avanços na Internet das Coisas (IoT) têm impulsionado inovações na área da saúde, incluindo o monitoramento remoto de pacientes. Neste contexto, este trabalho propõe um sistema de baixo custo, utilizando computação de borda e dispositivos IoT, para monitorar remotamente dados vitais específicos dos pacientes e e ainda, controlar de forma precisa a administração de medicamentos. Durante o desenvolvimento do sistema, foram criadas duas arquiteturas. A primeira inclui microcontroladores ESP32 conectados à Internet para coletar e gerenciar dados vitais, como frequência cardíaca, saturação periférica de oxigênio no sangue e temperatura corporal do paciente, além da temperatura ambiente. Ela também possui a capacidade de identificar o profissional que realiza procedimentos de medicação no paciente. A segunda arquitetura foi desenvolvida com o objetivo de aprimorar a primeira, adicionando os seguintes recursos: controle preciso da prescrição e administração de medicamentos, visando reduzir riscos à saúde do paciente por erros nesses processos; armazenar localmente os dados coletados pelos sensores e processados pelos microcontroladores ESP32 em cartões microSD *(datalogger)*; enviar estes dados por meio de um protocolo ($MQTT$) para um minicomputador (*Raspberry Pi*) que é gerenciado pelo sistema desenvolvido no *Node-RED*, que por sua vez armazena (via protocolo ($HTTP$) todas as informações geradas em uma planilha no *Google Drive*. Os resultados experimentais, das simulações das duas arquiteturas com um voluntário, foram devidamente validados (*benchmarking*) comparando-se com dispositivos usados em ambientes hospitalares. Também, os dados foram coletados, transmitidos e armazenados pelos respectivos módulos do sistema desenvolvido, conforme o esperado nos experimentos. Pode-se concluir que o sistema desenvolvido, visando acessibilidade (baixo custo) para todos os pacientes, atingiu os objetivos propostos e satisfez às hipóteses sobre monitoramento remoto do quadro clínico e segurança na prescrição e administração precisas de medicamentos dos pacientes.

Palavras-chave: Acessibilidade de pacientes, computação de borda, dispositivos de *IoT*, monitoramento remoto de pacientes, segurança de pacientes, sistema de baixo custo.

Abstract

Traditionally, the comprehensive monitoring of patients' clinical conditions occurs in hospital environments with specialized equipment and professionals, requiring high financial investments. This limits the access of low-income individuals to basic treatments in SUS hospitals in Brazil. However, advances in the Internet of Things (IoT) have propelled innovations in the healthcare sector, including remote patient monitoring. In this context, this work proposes a low-cost system utilizing edge computing and IoT devices to remotely monitor specific vital patient data and precisely control medication administration. Two architectures were created during the development of the system; the first one has ESP32 microcontrollers connected to the Internet for collecting and managing vital data such as heart rate, blood oxygen peripheral saturation, a patient's body temperature, and the temperature of his or her environment. It also has the feature of identifying the professional who performs the medication procedure on the patient. The second architecture, on the other hand, was developed to improve the first, adding the following features: precise control of prescription and administration of medications, to reduce the risks to the patient's health due to errors in these processes; locally store the data collected by the sensors and processed by the ESP32 microcontrollers on microSD cards *(datalogger)*; send this data through a protocol ($MQTT$) to a minicomputer (*Raspberry Pi*) that is managed by the system developed in *Node-RED*, which in turn stores it (via protocol ($HTTP$) all information generated in a spreadsheet on *Google Drive*. The experimental results, from the simulations of the two architectures with a volunteer, were duly validated (*benchmarking*) compared to devices used in hospital environments. Also, data were collected, transmitted and stored by the respective modules of the developed system, as expected in the experiments. One can conclude that the developed system, aiming for accessibility (low cost) for all patients, achieved the proposed objectives and fulfilled the hypotheses regarding remote monitoring of patients' clinical conditions and safety in precise prescription and administration of medications.

Keywords: Patient affordability, edge computing, IoT devices, patient remote monitoring, patient safety, low cost system.

Lista de ilustrações

Lista de siglas

APIs *Application Programming Interface*

FC Frequência Cardíaca

FCNs *Fog Computing Nodes*

GND *Graduated Neutral Density filter*

GPIO *General Purpose Input/Output*

HIS *Hospital Information System*

HTTP *Hypertext Transfer Protocol*

I^2C *Inter-Integrated Circuit*

INPI Instituto Nacional da Propriedade Industrial

IA Inteligência Artificial

IDC *International Data Corporation*

IoT *Internet of Things*

IP *Internet Protocol*

LCD *Liquid Crystal Display*

LED *Light Emitting Diode*

LOD *Linked Open Data*

MQTT *Message Queuing Telemetry Transport*

NTP *Network Time Protocol*

OASIS *Organization for the Advancement of Structured Information Standards*

OLED *Organic Light-Emitting Diode*

OMS Organização Mundial da Saúde

PWM *Pulse-Width Modulation*

QR Code *Quick Response Code*

RFID *Radio-Frequency IDentification*

SCL *Serial Clock Line*

SDA *Serial Data Line*

SPI *Serial Peripheral Interface*

SPO Saturação Periféria de Oxigênio no Sangue

SUS Sistema Único de Saúde

TC Temperatura Corporal

TCP *Transmission Control Protocol*

TLS *Transport Layer Security*

UART *Universal Asynchronous Receiver Transmitter*

UDP *User Datagram Protocol*

UI *User Interface*

VCC Voltagem em Corrente Contínua

VDC *Voltage Direct Current*

Sumário

Capítulo 1

Introdução

A produção de dispositivos cada vez menores e com um poder de processamento cada vez maior tem permitido o acesso de pessoas ou instituições a tecnologias avançadas e com preço muito mais convidativo. Sem dúvida, muitas áreas têm sido beneficiadas, principalmente a da saúde que tem continuamente acompanhado e aproveitado a evolução dos seus equipamentos. Com essas inovações, dispositivos que trabalhavam isoladamente *(offline)* tiveram melhorias a ponto de se tornarem conectáveis à rede mundial de computadores, fazendo surgir uma nova área de pesquisa chamada Internet das Coisas (*Internet of Things (IoT)*) (MADAKAM et al., 2015).

A capacidade da *IoT* para atender a uma variedade de demandas com soluções específicas, contribuiu para o desenvolvimento de pesquisas e inovações com aplicações em muitas áreas importantes como no agronegócio, na saúde e na indústria (SAHA; MANDAL; SINHA, 2017).

A utilização da *IoT* segue aquecida no mundo todo, e segundo a previsão da *International Data Corporation (IDC)* estima que haverá 41,6 bilhões de dispositivos conectados à *IoT*, gerando 79,4 zettabytes (ZB) de dados em 2025 (MACGILLIVRAY; REINSEL, b) (MACGILLIVRAY; REINSEL, a). No Brasil, o aporte para o setor em 2019 foi da ordem de *US$9* bilhões, dos quais os principais destinos foram para as aplicações no agronegócio, na saúde e na prestação de serviços públicos (IDC,).

1.1 Problematização

A saúde é uma das áreas que mais evoluiu nas últimas décadas por causa dos avanços tecnológicos. Os exemplos desta evolução estão na facilidade de comunicação entre os profissionais, nos diagnósticos mais precisos auxiliados pelas técnicas de inteligência computacional, exames mais avançados auxiliados por dispositivos computadorizados e até na utilização de robôs em cirurgias complexas. No entanto, esses recursos tecnológicos não estão disponíveis e acessíveis para todos. A maioria dos países enfrentam desafios que impactam diretamente nessa área, como o envelhecimento da população, pobreza,

mortalidade infantil, surtos frequentes de epidemias e a falta de saneamento básico, o que contribuiu para aumentar a demanda por serviços médicos no mundo nos últimos anos (FARAHANI et al., 2018).

Mesmo com as adversidades enfrentadas, o Sistema Único de Saúde (SUS) brasileiro é considerado um modelo de referência, devido à sua complexidade de atuação sobre a diversidade e abrangência territorial do país, o qual cobre 75% da população brasileira. Mas, o país está passando por uma grave crise econômica e a saúde pública tem sofrido muito nos últimos anos com cortes no orçamento, agravando problemas como falta de profissionais, superlotação nos hospitais, infraestrutura defasada, tecnologia de baixa qualidade (CEEN,).

Por outro lado, muitos estudos estão sendo realizados para analisar o impacto e as consequências causadas pela crescente utilização de recursos tecnológicos na área da saúde. Segundo o relatório do *Digital Health Technology Vision* publicado em 2019 pela *Accenture*, o "digital não é mais um diferencial", para eles os meios tecnológicos tornaram-se parte de tudo que as pessoas e instituições fazem (SAFAVI; KALIS; THOMPSON, 2019). Com isso, surgem principalmente na saúde pública, desafios de se produzir dados com qualidade e analisá-los por meio de tecnologias computacionais como de IoT de maneira ética e com segurança (SALERNO et al., 2017).

Baseado nisso, a Organização Mundial da Saúde (OMS) lançou um boletim para a submissão de *papers*, com o intuito de incentivar pesquisadores a produzirem trabalhos que contribuam com esses desafios (ZANDI et al., 2019). E, durante a crise gerada pelo novo coronavírus, a OMS tem se responsabilizado em acompanhar e gerenciar as informações desde que o classificou como pandemia no dia 30 de janeiro de 2020 (GAYTHORPE et al., 2020). Sua competência abrange a utilização de meios tecnológicos para gerar informação diária para o mundo sobre o número de pessoas infectadas, o número de mortes causadas pelo vírus, surgimento de novas pesquisas, os principais cuidados que se devem tomar, dentre várias outras informações fundamentais no combate à pandemia (WORLD HEALTH ORGANIZATION, b).

Com a pandemia da Covid-19 que atingiu o Brasil, os problemas já existentes tornaram-se ainda mais críticos, sendo registrados até o momento 4.915.289 de casos confirmados, 4.263.208 de casos recuperados e um total de 146.352 óbitos (MINISTÉRIO DA SAÚDE, a). Com estes números, segundo a plataforma *Worldometers*, o Brasil é o país com mais infecções na América do Sul e o terceiro a nível mundial, atrás apenas dos Estados Unidos e Índia (WORLDOMETERS,).

Outros problemas que tendem a aumentar com a falta de processos bem definidos e sistemas informatizados eficientes, são os riscos à segurança dos pacientes causados por uma variedade de falhas, denominadas como eventos adversos graves relacionados à assistência hospitalar. De acordo com a pesquisa do II Anuário de Segurança Assistencial Hospitalar no Brasil, em 2017 cerca de 54.769 pessoas perderam suas vidas devido a estes

eventos, uma média de 150 mortes por dia no território brasileiro (COUTO et al., 2018). Dentro desta margem, existem erros evitáveis durante o processo de medicação, como dosagens ou infusão incorretas, instruções incertas, uso de abreviações com interpretações equivocadas e prescrições inadequadas. No mundo, o custo relacionado a erros de medicação foi estimado em *US$42* bilhões anualmente, quase 1% da despesa global em saúde (WORLD HEALTH ORGANIZATION, a).

A área da saúde no Brasil, assim como na maioria das regiões no mundo, enfrenta graves desafios para atender à demanda dos seus pacientes. Diante do contexto, é necessário investimento financeiro, mas também se torna importante a implementação de projetos que sejam de baixo custo e eficientes no atendimento aos enfermos (FARAHANI et al., 2018).

1.2 Desafios e objetivos da pesquisa

Diante da problematização levantada, os desafios desta pesquisa foram:

- ❏ Oferecer uma solução de baixo custo que seja passível de ser implementada em sistemas de saúde como o SUS no Brasil e que os recursos sejam acessíveis aos pacientes de forma universal;
- ❏ Oferecer segurança aos pacientes em termos de não ocorrência de falhas humanas no tratamento clínico ou mais especificamente, na prescrição ou administração de medicamentos.

Com os desafios a serem superados, o objetivo da pesquisa foi desenvolver um sistema eficiente e de baixo custo que possa aproveitar os recursos da *(IoT)* associados à arquitetura da computação de borda *(edge computing)* e os recursos de *software open source* como o *Google Drive*.

Mais precisamente, os objetivos específicos do trabalho foram:

- ❏ Possibilitar diagnósticos e tratamentos mais eficientes, ainda que de forma remota, através de um sistema de coleta, transmissão e armazenamentos de dados clínicos como temperatura corporal ou do recinto onde o paciente se encontra, frequência cardíaca ou saturação periférica de oxigênio no sangue;
- ❏ Reduzir erros na prescrição ou administração de medicamentos aos pacientes através do acompanhamento ou registro preciso de todas as ações relacionadas ao processo de medicação como horário, dosagem, profissional responsável, modo de aplicação;
- ❏ Possibilitar um controle mais preciso da evolução do quadro clínico dos pacientes, por meio de relatórios e históricos dos dados monitorados e registrados e consequentemente, levar à elaboração de prontuários eletrônicos fundamentados em dados clínicos mais precisos obtidos pelo sistema desenvolvido.

1.3 Hipóteses e contribuições esperadas

As questões a serem respondidas mediante a proposta do trabalho e a simulação através de experimentos do protótipo desenvolvido para o sistema proposto foram:

1. É possível construir um sistema dotado de arquitetura com dispositivos de *IoT* que funcione adequadamente em um ambiente com baixíssima tolerância a falhas como num hospital?

2. O sistema a ser desenvolvido seria capaz de coletar e transmitir os dados clínicos vitais e sigilosos medidos nos pacientes e ainda armazená-los, garantindo a integridade e confidencialidade?

3. O sistema de monitoramento seria capaz de proporcionar tomadas de decisão eficaz por parte de uma equipe médica, mesmo que o paciente esteja em tratamento à distância (até fora do ambiente hospitalar), quando surgir alguma anomalia nos dados de temperatura corporal, frequência cardíaca ou taxa de oxigênio no sangue?

4. Com a implementação do sistema proposto, seria possível reduzir significativamente ou até a zero, os erros na administração de medicamentos a um paciente através da identificação das causas destes erros de forma a tomar medidas para que os mesmos não se repitam?

Com o desenvolvimento do trabalho, esperou-se que cada uma das hipóteses levantadas anteriormente fossem devidamente satisfeitas e os objetivos propostos fossem alcançados.

Para que a expectativa fosse concretizada, o sistema consistiu de duas arquiteturas cujas propostas que se encontram pormenorizadas nas Seções 3.1 e 3.2 e os respectivos desenvolvimentos que estão apresentados nas Seções 4.1 e 4.2. Os resultados obtidos com a simulação e os experimentos envolvendo voluntário submetido ao sistema desenvolvido em funcionamento estão descritos nas Seções 5.2 e 5.3.

1.4 Organização da dissertação

Esta dissertação se encontra organizada em seis capítulos. O Capítulo 2 apresenta o levantamento bibliográfico realizado sobre os tópicos relacionados ao desenvolvimento do sistema e trabalhos correlacionados ao tema da dissertação. O Capítulo 3 descreve a proposta das duas arquiteturas que compõem o sistema desenvolvido. O Capítulo 4 detalha todas as etapas de desenvolvimento das duas arquiteturas. O Capítulo 5 apresenta e discute todos os resultados obtidos dos dois experimentos (um para cada arquitetura) com o sistema desenvolvido em funcionamento. O Capítulo 6 contém a conclusão do trabalho, as contribuições realizadas com o desenvolvimento do trabalho e algumas sugestões para trabalhos futuros.

Capítulo 2

Revisão da literatura correlata

2.1 Internet das coisas (*IoT*)

O termo *IoT* foi apresentado pela primeira vez em 1999 por Kevin Ashton, como título de uma apresentação ministrada na *Procter & Gamble (P&G)*, para se referir ao conceito de permitir que um computador, por meio dos seus dispositivos, detecte informações sem a intervenção humana (ASHTON et al., 2009). Desde então, com o progresso da tecnologia tornou-se possível, através da Internet, a comunicação entre diversos tipos de dispositivos, popularizando a *IoT* (SHI; DUSTDAR, 2016).

O vasto campo de aplicação da *IoT* tornou sua demanda aquecida no mundo todo e segundo a previsão da *IDC* estima-se que haverá 41,6 bilhões de dispositivos conectados à *IoT*, gerando 79,4 zettabytes (ZB) de dados em 2025 (FRAMINGHAM,). No Brasil, o aporte para o ano de 2019 foi de *US$9* bilhões, sendo que os principais estímulos foram das aplicações no agronegócio, na saúde e na prestação de serviços públicos (IDC,).

2.1.1 Arquitetura multicamadas da *IoT*

A *IoT* possui uma arquitetura multicamadas, e de forma geral ela pode ser dividida em quatro níveis. Como exemplo, em (XU; HE; LI, 2014) foram analisados artigos para evidenciar os desafios e as promessas da *IoT* na medicina e na saúde e apresentaram essas camadas voltadas para os cuidados dos pacientes de forma descentralizada aos ambientes hospitalares, como ilustrado na Figura 1.

No sentido de baixo para cima, o primeiro nível representa a camada de detecção e atuação *(Sensing)*, onde os diferentes tipos de dispositivos são capazes de detectar e trocar informações automaticamente com o meio físico e entre si, como sensores, leitores de código de barras, etiquetas e gravadores *Radio-Frequency IDentification (RFID)*, câmera, etc. A camada de rede *(Networking)* é responsável por transmitir os dados trafegados entre os dispositivos e as centrais, por meio de redes cabeadas, redes *Wi-Fi*, *bluetooth* etc. Na camada de serviço, *(Service)* são processadas as informações recebidas pelos dispositivos,

incluindo troca e armazenamento de informações, gerenciamento de dados, mecanismos de busca e comunicação. Nela atuam tecnologias como sistemas de Inteligência Artificial (IA) hospedadas em servidores na nuvem *(cloud)*. Já a camada de *Interface* é responsável por oferecer métodos de interação entre usuários e aplicativos, pois é nela que os resultados dos serviços anteriores são apresentados e analisados. Grande parte da contribuição da *IoT* se encontra nessa camada, devida à possibilidade de visualizar e monitorar dados de dispositivos, antes isolados e sem processamento (XU; HE; LI, 2014). O aplicativo *Blynk* é um exemplo que cabe nesta camada, possibilitando criar interfaces de comunicação com dispositivos conectados na Internet, por meio de um desenvolvimento simples e intuitivo (BLYNK,).

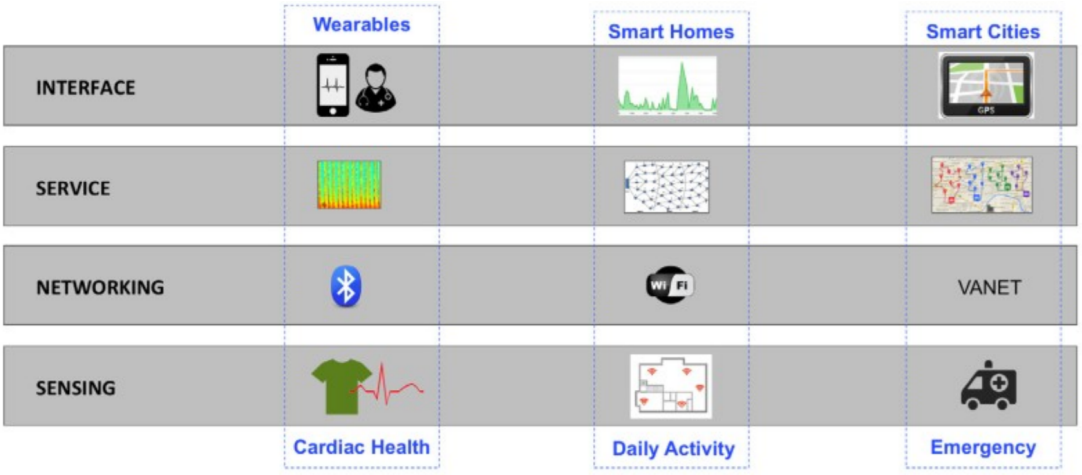

Figura 1 – Camadas da arquitetura *IoT* aplicada no ambiente da saúde.

Fonte: XU, HE e LI (2014)

Já em (XIE; YANG; YANG, 2018), foi desenvolvido um método para enriquecer os registros médicos baseados em *IoT*, usando como referência o conceito de *Linked Open Data (LOD)*, que basicamente são técnicas para vincular dados e disponibilizá-los no formato da arquitetura *web*. O método foi aplicado em um sistema de informação hospitalar (*Hospital Information System (HIS)*) real, contendo dados fornecidos por dispositivos *IoT*. Para eles, um sistema como este, coleta os dados de maneira dispersa devida à distribuição dos dispositivos, o que gera um grande volume de dados com a semântica fraca. Portanto, isso prejudica a qualidade das análises para a tomada de decisões. Segundo os autores, os resultados após o experimento mostraram que os registros médicos foram enriquecidos significativamente, tornando o acesso às informações mais eficiente. O trabalho mostrou que além de uma boa infraestrutura, são necessários processos eficazes durante a coleta e armazenamento dos dados gerados pelos dispositivos *IoT*.

2.1.2 *IoT* na nuvem

A computação em nuvem *(cloud computing)* permanece avançando e fornecendo serviços ainda mais eficientes, mantendo as vantagens de migrar as tarefas locais, para servidores com poder computacional elevado. No entanto, a largura de banda das redes não evoluiu da mesma forma. Portanto, com o rápido progresso dos dispositivos móveis e aplicações baseadas na *IoT*, surgiram vários desafios para o uso destes serviços em computação em nuvem. Dentre esses desafios, destacam-se a transferência de grandes volumes de dados e a possibilidade de muita latência de comunicação, podendo inviabilizar procedimentos em ambientes que exigem rápida reação a eventos como no ambiente hospitalar, principalmente de emergência (SHI; DUSTDAR, 2016).

Outro desafio importante é a segurança dos dados transmitidos para a nuvem, como descrito em (HOSSAIN; MUHAMMAD, 2016). Os autores desenvolveram um sistema para prevenir situações de risco à saúde dos pacientes que tinham seus dados vitais monitorados e enviados para a nuvem. Para isso, uma equipe de profissionais da saúde tinham acesso aos dados na nuvem para analisá-los e tomar decisões. Os autores também enfatizaram a importância de ter um método de segurança para transmitir os dados dos pacientes. E por isso eles alertam sobre os riscos que os dados correm durante a transferência e sincronização dos dispositivos de *IoT* com a nuvem.

Segundo (SHI; DUSTDAR, 2016), com a implementação da *IoT* cada vez mais difundida, iniciou-se a era pós-nuvem em que os dispositivos geram muitos dados na borda da rede, tendo também o aumento de aplicativos para processá-los. Portanto, a computação em nuvem não tem sido eficiente o bastante para lidar com grandes volumes de dados e com a privacidade das informações transferidas.

2.1.3 *IoT* na névoa

Como alternativa aos desafios da computação em nuvem, o paradigma da computação em névoa *(fog computing)* tem-se destacado. Há ainda a possibilidade de se combinar as duas arquiteturas, extraindo o melhor serviço de cada uma (SHI; DUSTDAR, 2016). O termo "computação em névoa" foi descrito primeiramente por (BONOMI et al., 2012), para tratar de uma infraestrutura de computação descentralizada em que as aplicações, a comunicação e os dados armazenados podem ser distribuídos entre a nuvem e a borda da rede onde as informações são geradas. Neste sentido, os autores destacam a importância do posicionamento dos *gateways* na rede, como roteadores, *switches* e outros dispositivos responsáveis por processar e transmitir os dados dessa arquitetura. Eles possuem um papel fundamental neste paradigma e são chamados de nós de computação em névoa (*Fog Computing Nodes (FCNs)).*

Em seu trabalho, (RAHMANI et al., 2018) aplicaram a Figura 1 para beneficiar um sistema de saúde com monitoramento remoto por meio da combinação dos recursos da

computação em névoa e da computação em nuvem. Os autores exploraram a camada de névoa por meio das posições estratégicas dos *gateways* na rede para contribuir com a solução de problemas do sistema como a mobilidade dos sensores e usuários. Como resultado, eles concluíram que a aplicação dos conceitos da computação em névoa aprimora diferentes características da arquitetura *IoT* usada para aplicativos de assistência médica, citando a eficiência energética, o desempenho, a interoperabilidade, dentre outras.

2.1.4 *IoT* na borda

Assim como ocorreu com a computação em névoa, a computação de borda *(edge computing)* aparece para atender às necessidades que surgiram após a popularização da *IoT*. Elas ficam próximas dentro da arquitetura de rede, mas possuem funções distintas. Segundo (SHI et al., 2016), os dois paradigmas atuam processando dados na extremidade da rede, mas a computação em névoa é responsável pela infraestrutura dos nós de comunicação, enquanto a computação de borda se concentra na atuação das coisas (dispositivos, sensores, etc.).

O aumento dos dispositivos móveis e das coisas conectadas à Internet fez com que suas funções mudassem de consumidores para também produtores de informação, e com isso um grande fluxo de dados passou a ser gerado na borda da rede. Logo, esses dados precisam ser processados na borda para proteger a privacidade das informações, obter menor tempo de resposta com processamento mais eficiente e com menor consumo da largura de banda (DOLUI; DATTA, 2017). Com isso, dispositivos criados para funções diversas, passaram a ter novas utilidades na *IoT*, como a família do *Raspberry Pi* e os microcontroladores, tendo como exemplo o ESP32.

2.2 Componentes e recursos para dispositivos de *IoT*

2.2.1 *Raspberry Pi*

O *Raspberry Pi* é conhecido como um computador de placa única por executar funções como os de *desktops*, *notebooks* e *smartphones* e por ser construído em uma pequena placa de circuito impresso com baixo custo (UPTON; HALFACREE, 2014).

A primeira versão da família *Raspberry Pi*, o modelo 1B, foi lançado pela Fundação *Raspberry Pi* em fevereiro de 2012, com o intuito de promover conhecimento sobre a computação (HALFACREE, 2018).

No entanto, devido à sua grande procura, o *Raspberry Pi* passou a atuar em diferentes cenários e com a *IoT*, ficou conhecido mundialmente. Seu atual modelo é o *Raspberry Pi 4* mostrado na Figura 2 e suas principais especificações são: o processador de 4 núcleos de 1,5 GHz, até 4GB de memória *RAM*, conexões sem fio de 2,4 GHz e 5,0 GHz, *Bluetooth 5.0* e a conexão *gigabit Ethernet* (RASPBERRY,). Com todo esse poder, ele pode ser

usado como um *gateway* na computação em névoa e ter sensores acoplados a ele na borda da rede, características consideradas vantajosas dentro da *IoT*.

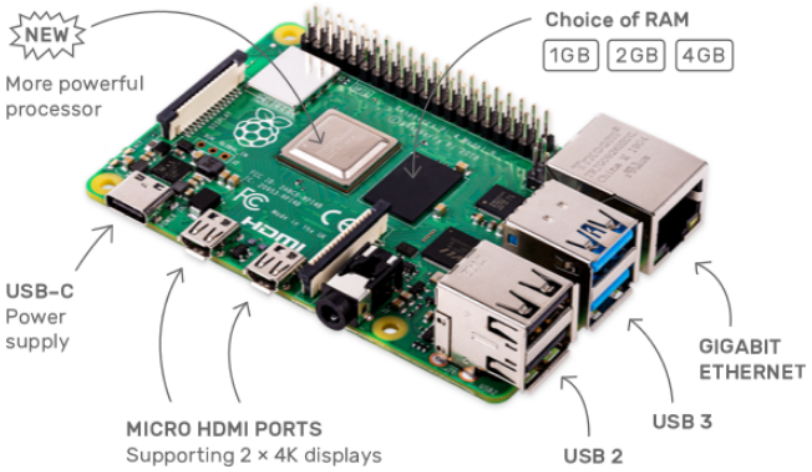

Figura 2 – *Raspberry Pi 4*.

Fonte: RASPBERRY ()

2.2.2 Microcontrolador ESP32

Os microcontroladores também foram beneficiados com a evolução da microeletrônica em conjunto com a *IoT*. Os sistemas embarcados, por sua vez, tiveram uma ascensão com a possibilidade de conexão com a Internet. O microcontrolador ESP32 é um exemplo dessa realidade, pois ele possui um elevado desempenho quando comparado com os microcontroladores tradicionais, como os modelos do Arduino e o ESP8266 (CIRCUITO,).

A Figura 3 ilustra a tabela de comparação dos principais recursos entre os três microcontroladores citados: ESP32, ESP8266 e o Arduino Uno R3. Vale destacar os dois núcleos de processamento de 32 bits, conexão *Wi-Fi* e *Bluetooth* e as memórias *RAM* e *Flash* que o microcontrolador ESP32 possui (SYSTEMS,).

Um dos modelos mais comercializados de microcontrolador ESP32 está mostrado na Figura 4 com destaque para os seus pinos *General Purpose Input/Output (GPIO)*21 (a) e *GPIO*22 (b) que são responsáveis por trabalhar com o protocolo *Inter-Integrated Circuit (*I^2C*)* (SANTOS; SANTOS, a).

Protocolo I^2C:

Basicamente, o protocolo I^2C é responsável pelo funcionamento de um barramento de comunicação serial que utiliza apenas os dois pinos. Ele foi criado pela *Philips* no

RECURSOS	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16Mhz
Wifi	Sim	Sim	Não*
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	4MB	4MB	32KB
GPIO	30	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

Figura 3 – Tabela de comparação de recursos entre microcontroladores.

Fonte: OLIVEIRA ()

início da década de 90 e é muito utilizado para conectar periféricos de baixa velocidade a placas-mãe, microcontroladores e muitos tipos de sensores. Os dois fios que compõem o barramento I^2C são o *Serial Data Line (SDA)* e o *Serial Clock Line (SCL)*, sendo o *SCL* responsável pelo *clock* do barramento e o *SDA* pela transmissão de dados. Um único barramento pode atender vários dispositivos, por meio dos seus endereços I^2C, em uma comunicação *half-duplex* (CIRCUIT BASIC,).

ESP32-CAM:

O ESP32-CAM é um módulo baseado no ESP32, mas que possui uma câmera OV2640 e um *slot* para cartão microSD integrado, conforme está apresentado na Figura 5. Esses recursos combinados com o baixo consumo de energia permitem que o ESP32-CAM seja amplamente usado em aplicações de *IoT* inteligentes, identificadores de *Quick Response Code (QR Code)*, dentre vários outros projetos. Os recursos adicionais são conexões *Wi-Fi* e *Bluetooth* para transferência de imagem e vídeos sem fio. Diferentemente do ESP32 tradicional, o ESP32-CAM conta com apenas 16 pinos (*GPIOs*) disponíveis dotados de funções de *Pulse-Width Modulation (PWM)*, I^2C, *Serial Peripheral Interface (SPI)*, e *Universal Asynchronous Receiver Transmitter (UART)*, sendo que 10 deles são de entrada e saída e 6 estão relacionados à energia, conforme se verifica na Figura 6. Outra diferença do tradicional é que o ESP32-CAM não possui entrada USB, necessitando de um conversor USB para realizar sua programação (AI-THINKER,).

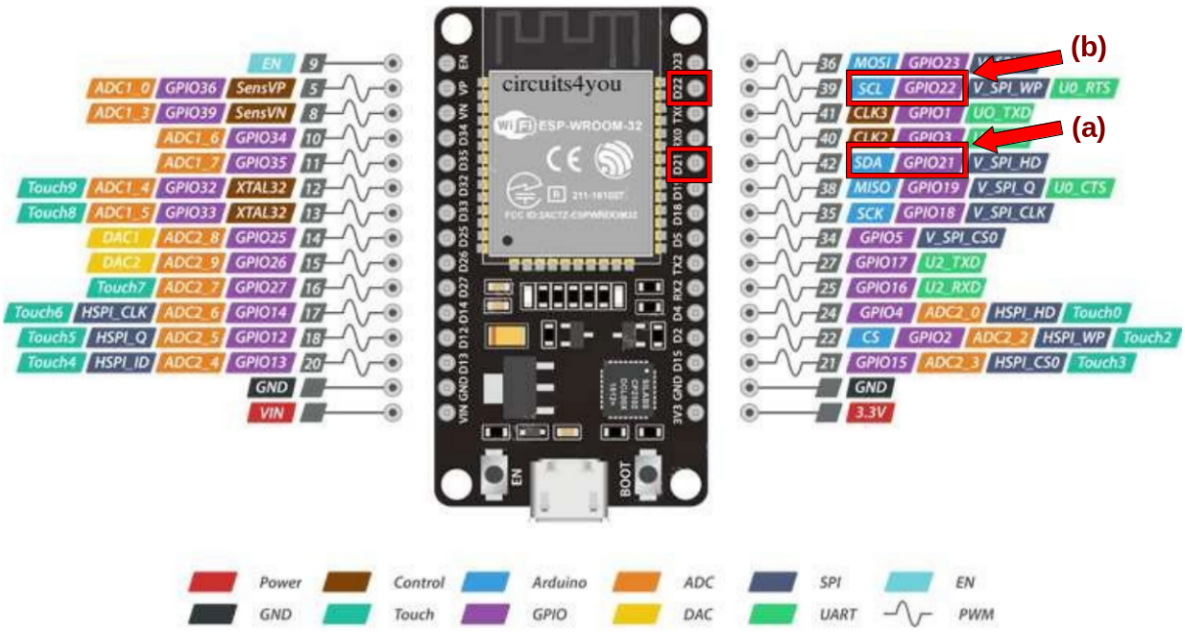

Figura 4 – Pinagem do ESP32. (a) *GPIO*21; (b) *GPIO*22

Fonte: SANTOS e SANTOS (a)

.

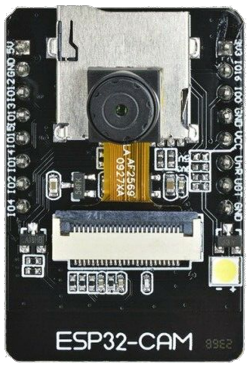

Figura 5 – Módulo ESP32-CAM

Fonte: SANTOS e SANTOS ()

.

2.2.3 Sensores de temperatura

A medição de temperatura é uma função executada nas aplicações de *IoT* por meio de diversos tipos de sensores. Neste trabalho, foram utilizados sensores para medir a temperatura corporal de um paciente e a temperatura do ambiente onde ele se encontra, conforme descritos a seguir.

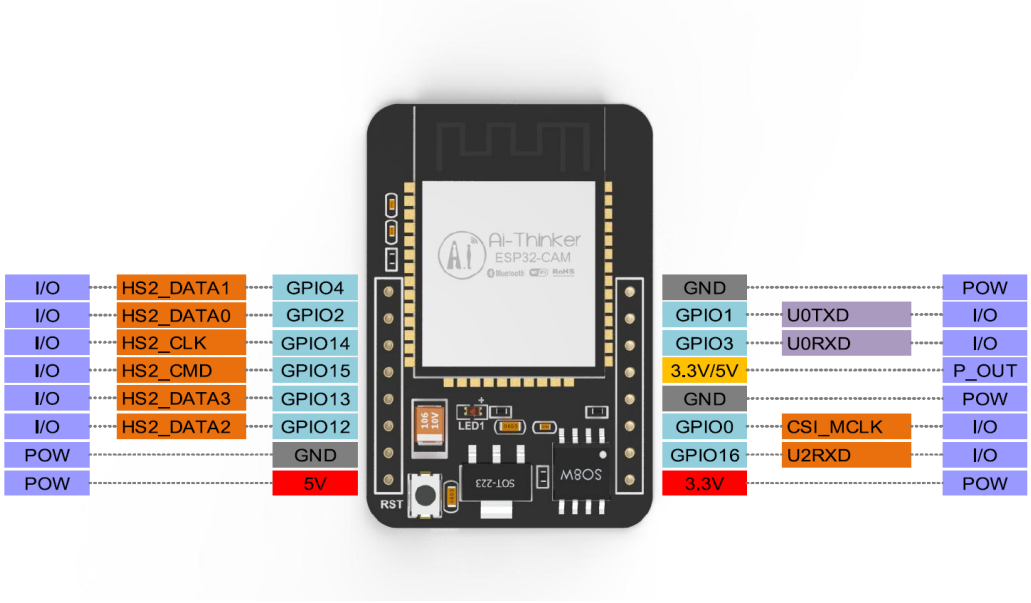

Figura 6 – Pinagem do módulo ESP32-CAM

Fonte: SANTOS e SANTOS ()

.

Sensor de temperatura MLX90614:

O sensor de temperatura infravermelho MLX90614 possui dois chips que medem a temperatura ambiente e a temperatura corporal ou de objetos, sem necessidade do contato físico. Sua calibração é padrão de fábrica, mas ela pode ser alterada dependendo da necessidade do ambiente onde ele for utilizado. Sua faixa de medição da temperatura ambiente está entre $-40\,°C$ e $125\,°C$ e da temperatura corporal ou de objetos está entre $-70\,°C$ e $380\,°C$, com variação de $0,5\,°C$. Sua comunicação com o microcontrolador é realizada via interface I^2C (MELEXIS INSPIRED ENGINEERING,). Neste trabalho, o sensor é utilizado para medir temperatura corporal e está ilustrado na Figura 7.

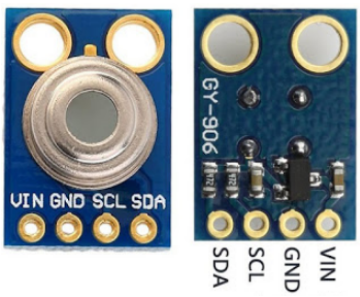

Figura 7 – Sensor de temperatura MLX90614.

Fonte: TECNODOMOS ()

Sensor de temperatura DS18B20:

O sensor DS18B20 é também muito utilizado em projetos de *IoT*, por causa da sua qualidade durante a medição da temperatura e da facilidade de operação com apenas três fios (de alimentação 3.3 V ou 5 V, negativo *Graduated Neutral Density filter (GND)* e de dados). O fio de dados comporta até 12 bits de precisão a partir do conversor digital-analógico integrado para repassar as medições de temperatura em graus Celsius. Sua temperatura de operação varia de -55°C a + 125°C com uma precisão de ± 0,5°C ao longo de quase toda sua faixa. Cada sensor DS18B20 possui um código de série exclusivo de 64 bits, permitindo que vários sensores sejam usados no mesmo barramento, utilizando apenas um único pino *GPIO* de dados do microcontrolador. E ainda, o sensor possui uma versão à prova de água, mostrada na Figura 8. Essa versão é muito útil para várias aplicações e necessidades de projetos que devem considerar a existência de umidade durante a medição. Mas, nessa versão, não é recomendado pelo fabricante ultrapassar os 100°C, devida à capa de PVC que impermeabiliza o sensor (RESOLUTION, 2019).

Figura 8 – Sensor de temperatura DS18B20.

Fonte: ROZAQ e DS (2017)

Sensor de temperatura DHT22:

O sensor de umidade e temperatura conhecido como DHT22 possui em seu módulo um sensor de umidade do ar que mede de 0% até 100% com precisão de até 5% e um sensor de temperatura que mede o ar do ambiente nas escalas de -40ºC a +80ºC com precisão 0,5ºC. Ambos estão conectados a um controlador de 8 bits que produz um sinal digital no pino de dados (data). A sua estrutura está ilustrada na Figura 9 com quatro pinos, dos quais três (pino de alimentação Voltagem em Corrente Contínua (VCC) (3.3 *Voltage Direct Current (VDC)* ou máximo 5 *VDC*) (1), negativo (*GND*) (4) e pino de dados (2))

são fundamentais para sua utilização. A qualidade de medição instantânea da umidade e da temperatura ambiente, o baixo consumo de energia e a facilidade de manipulação e configuração tornam o sensor DHT22 muito popular para o desenvolvimento de projetos de *IoT* e de automação (LIU, 2013).

Figura 9 – Estrutura do sensor de temperatura e umidade DHT22.

Fonte: SANTOS e SANTOS (c)

2.2.4 Sensor de frequência cardíaca e de saturação de oxigênio no sangue MAX30100

O MAX3010 é um sensor para detectar sinais de frequência cardíaca e de saturação de oxigênio no sangue, ideal para projetos de dispositivos vestíveis, de atividade física e de monitoramento médico. As medições são baseadas na variação do fluxo sanguíneo identificada por dois *Light Emitting Diode (LED)s* em seu módulo apresentado na Figura 10 (MAXIM INTEGRATED PRODUCTS,). As leituras são armazenadas em uma estrutura de dados *FIFO* e compartilhadas em um barramento conectado a um microcontrolador via protocolo I^2C.

2.2.5 Módulo *RFID* RC522

O módulo RC522, ilustrado na Figura 11, é composto por um transceptor com decodificador, uma antena e um transponder. Os componentes são responsáveis por energizar com um sinal de radiofrequência (13,56 MHz) uma bobina contida nos dispositivos de proximidade para gravar ou ler informações. Exemplos de dispositivos de proximidade são cartão branco e tag azul que possuem tecnologia *Mifare* de 1 kb de memória (NXP SEMICONDUCTORS,). Assim como os sensores anteriores, o módulo *RFID* também se comunica com o microcontrolador via protocolo I^2C, através dos pinos *SCL* e *SDA*.

Figura 10 – Sensor de frequência cardíaca e taxa de oxigênio no sangue, MAX30100

Fonte: PEKO'S LIBRARY ().

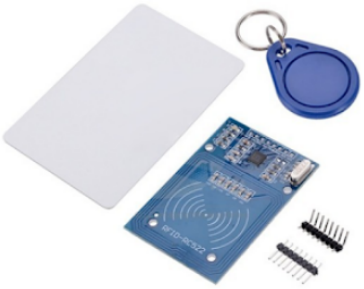

Figura 11 – Módulo *RFID* RC522

Fonte: KOYANAGI ().

2.2.6 Display *OLED* SSD1306

O SSD1306 é um pequeno display *Organic Light-Emitting Diode (OLED)* gráfico, de chip único com resolução de 128x64 e 0,96" e está mostrado na Figura 12. O seu baixo consumo de energia e a sua comunicação também feita pelo protocolo I^2C facilitam e tornam amigável a interação dos usuários com a arquitetura desenvolvida (SYSTECH, 2005).

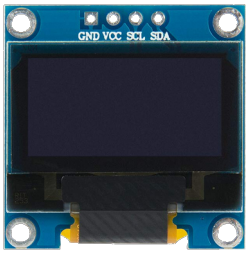

Figura 12 – Display *OLED* SSD1306.

Fonte: SANTOS e SANTOS (b)

2.2.7 Módulo de microSD

Um recurso importante para os projetos de *IoT* são os registros (*datalogger*) das informações geradas pelos dispositivos e sensores. Baseado nisso o módulo Adaptador de Cartão MicroSD (*MicroSD Card Adapter*) realiza leituras e gravações em cartões microSD a comando de um microcontrolador (SANTOS; SANTOS,). A comunicação entre eles é feita através do protocolo *SPI*, que utiliza 6 pinos, sendo dois para alimentação do circuito (5V) e os outros para comunicação. Com esse recurso, é possível criar cópias de segurança dos dados medidos e/ou gerados pelos sensores e outros dispositivos de uma arquitetura *IoT*, prevenindo que as informações geradas sejam perdidas, caso ocorra alguma falha na rede de comunicação (AIEA,). O módulo adaptador de cartão microSD é ilustrado na na Figura 13.

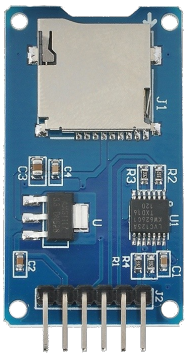

Figura 13 – Módulo Adaptador de Cartão MicroSD.

Fonte: AIEA ()

2.2.8 Protocolo *MQTT*

Uma condição para que um dispositivo faça parte de um sistema baseado em *IoT* é a sua capacidade de conexão com a Internet. Geralmente, o cenário padrão deste ambiente de *IoT* envolve redes com recursos limitados e dispositivos com baixo poder de processamento. Logo, os protocolos de rede tradicionais, como o *Hypertext Transfer Protocol (HTTP)*, não são adequados para o mundo da *IoT*, devida à necessidade de comunicação síncrona entre cliente e servidor. A comunicação escalada em grande quantidade de dispositivos tende a não ser confiável em redes de alta latência (YUAN,).

O protocolo *Message Queuing Telemetry Transport (MQTT)* por sua vez, foi desenvolvido pela IBM no final dos anos 90, com o intuito de ser simples, leve e que permitisse a interação de várias máquinas com poucos recursos de processamento. Desde então, ele evoluiu significativamente ao ponto de se tornar um padrão *Organization for the Advancement of Structured Information Standards (OASIS)*, com suporte para softwares lives e linguagens de programação populares, as quais evidenciam-se as linguagens C, C++,

C#, Java, Python e Lua, sendo considerado como referência para a comunicações de dispositivos de *IoT* (COPPEN, Richard,).

O sucesso do *MQTT* é devido ao modelo de publicação e assinatura que o torna um protocolo seguro, com vários níveis de serviço e possibilidade de conexão de *1* para *N*. Diferentemente da relação cliente-servidor, o modelo de publicação e assinatura possui dois tipos de entidade, um *message Broker* e inúmeros clientes como ilustrado na Figura 14 (OASIS,).

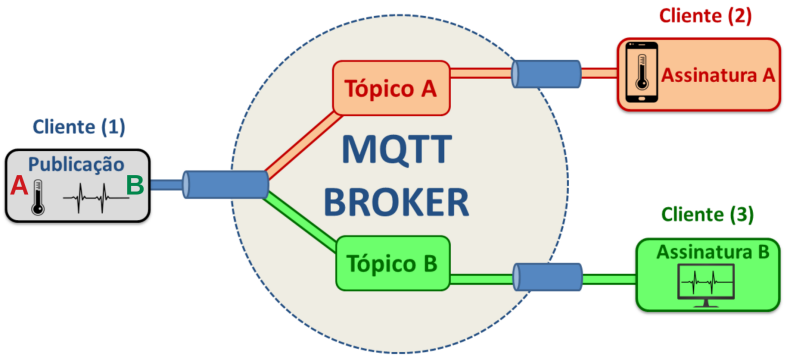

Figura 14 – Arquitetura *MQTT* Broker (HIVEMQ TEAM,).

Neste modelo os clientes se conectam ao *broker* através de uma conexão *Transmission Control Protocol (TCP)/Internet Protocol (IP)* simples ou uma conexão *Transport Layer Security (TLS)* criptografada para mensagens sensíveis. O *Broker* tem um papel fundamental neste cenário, ele é responsável por receber todas as informações publicadas pelos clientes e organizá-las em "tópicos", os quais serão assinados pelos clientes interessados na informação. Ou seja, após a mensagem ser publicada, o *Broker* irá encaminhá-la para os clientes que assinam o tópico referente a ela (MICROSOFT AZURE,). Um *Broker* muito utilizado nos projetos *IoT* e de código aberto é o *Eclipse Mosquitto*. Atualmente ele atende as versões 5.0, 3.1.1 e 3.1 do protocolo *MQTT* (MOSQUITTO,).

2.2.9 *Node-RED*

O *Node-RED* é uma ferramenta usada para criação de aplicativos de *IoT*, painéis de gestão (*Dashboard*) e projetos de automações. A ferramenta permite a conexão de dispositivos de *hardware*, *Application Programming Interface (APIs)* e serviços *online*. É um software de código aberto desenvolvido pela IBM e tem como objetivo simplificar a criação de projetos complexos por meio da programação visual orientada por eventos. Através de um navegador, uma ampla variedade de blocos de código (nós) podem ser conectados para gerar um fluxo (*flow*), que irá executar uma ou várias tarefas. Por ser leve para sua execução, o *Node-RED* é ideal para ser instalado na borda da rede em dispositivos de baixo custo como o *Raspberry Pi* e também em servidores de nuvem. Sua comunidade

é muito grande devido à popularização da *IoT* e pela facilidade de compartilhamento dos fluxos criados pelos usuários. O repositório de pacotes conta com mais de 225.000 módulos, disponíveis para serem editados e adicionados a novos projetos (NODE-RED, a).

A interação com o *Node-RED* num navegador é feita através do endereço (*IP*) da máquina que ele está instalado, seguido da porta 1880 (a). Dessa forma, a página de acesso ao *Node-RED* irá carregar, conforme reproduzida na Figura 15. No lado esquerdo se encontra a lista dos nós (nodes) (b) instalados no *Node-RED* e que estão disponíveis para serem usados. No centro está a área onde os fluxos *(flow)* (c) são criados através da conexão dos nós. No lado direito da tela, se apresentam as informações dos nós, as descrições das execuções do sistema, dentre outras configurações (d). Também do lado direito, na parte superior da tela, se encontra o botão de *"Deploys"* (e), responsável por executar e ativar os fluxos construídos (NODE-RED, c).

Como exemplo, a Figura 16 apresenta dois fluxos do módulo *"Dashboard"*, que possui vários nós utilizados para criar interfaces gráficas com a usabilidade de monitorar e/ou interagir em tempo real com dispositivos *IoT* e outras tecnologias como protocolos de rede (*MQTT*, *HTTP*, *TCP*, *User Datagram Protocol (UDP)*, etc). O acesso à interface de usuário (*User Interface (UI)*) é feito pelo mesmo endereço anterior incluindo o "\ui" (h) como na Figura 16. A interface apresentada na Figura 16 é referente aos fluxos criados com os dois elementos de interação *(widgets)* (f) e (g) da Figura 15. O primeiro elemento (f) é um medidor de interface do usuário que pode ser usado para diversas finalidades como por exemplo, apresentar os dados de temperatura de um ambiente que está sendo medido em tempo real. O segundo elemento (g) é um gráfico de linha que também pode ser usado com diversas funcionalidades, inclusive representar a medição da temperatura ao longo do tempo. Todos os nós contidos no módulo *"Dashboard"* do *Node-RED* podem ser editados e personalizados de acordo com o projeto que eles forem usados (NODE-RED, b).

2.2.10 Formulários e planilhas do *Google Drive*

O *Google* disponibiliza diversos produtos e serviços que atendem pessoas físicas e jurídicas com uma variedade de recursos. Um deles é o *Google Drive* que serve como uma plataforma de armazenamento em nuvem com a vantagem de ser gratuita, um aspecto importante para o propósito do sistema deste trabalho. São disponibilizados serviços *online* de editores de formulários, planilhas, textos e slides de forma interativa e compartilhada. Todos esses recursos podem ser aproveitados dentro do limite de armazenamento de 15 GB (GOOGLE DRIVE, b).

Uma facilidade do *Google Drive* a destacar é o fato de permitir o compartilhamento e a edição simultânea por vários usuários até de ambientes empresariais de maneira colaborativa. (GOOGLE DRIVE, a)

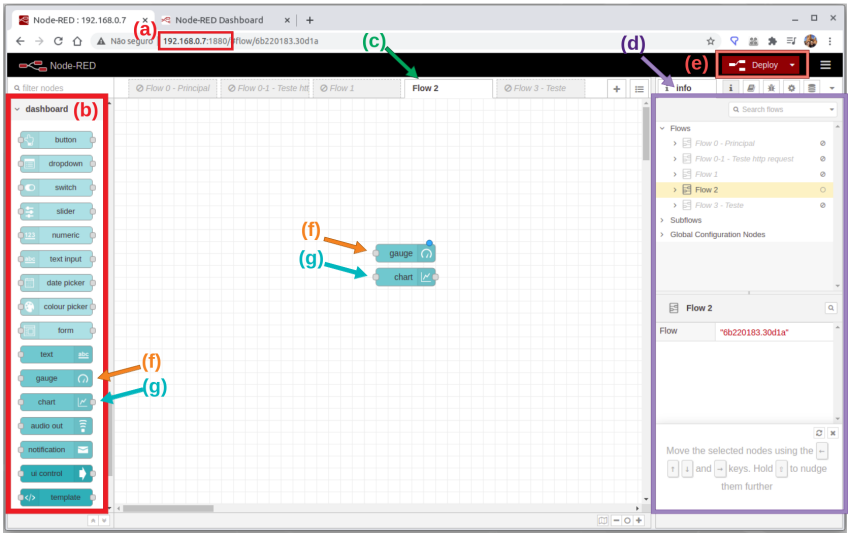

Figura 15 – Página de acesso ao *Node-RED*

Legenda: (a) endereço *IP* da página do *Node-RED*; (b) conjuntos de nós *dashboard* (c) página de edição de fluxo do *Node-RED*; (d) informações dos nós e fluxos do *Node-RED*; (e) botão de *Deploy*; (f) nós *gauge*; (g) nós *chart*

Outro destaque é a possibilidade de combinar os vários recursos. Por exemplo, é possível registrar nas planilhas do *Google* Planilhas, as respostas de um determinado formulário do Formulários *Google* disponibilizado na Internet (LASKOWSKI, 2016). Este tipo de facilidade permite que outras aplicações possam registar informações nas planilhas de maneira automática e em tempo real, como é o caso deste trabalho em que os dispositivos de *IoT* geram dados e devem registrá-los nas referidas planilhas (KODALI; RAJANARAYANAN, 2019).

2.3 Monitoramento de dados vitais usando IoT

A *IoT* é considerada promissora para enfrentar os desafios existentes na saúde, devida à possibilidade de soluções para situações específicas e com baixo custo. Seu potencial está relacionado à capacidade de automação e mobilidade dos dispositivos conectados à Internet, sendo ótimos recursos para auxiliar nos tratamentos e diagnósticos mais eficientes (FARAHANI et al., 2018). Este trabalho deverá contribuir para superar os desafios na saúde através de um sistema de baixo custo voltado à solução de problemas locais.

Os dados gerados pelos dispositivos de *IoT* poderão beneficiar todo o ambiente hospitalar como os prontuários e também contribuir para a diminuição dos riscos à segurança dos pacientes, pelo controle preciso da administração de medicamentos (XIE; YANG; YANG, 2018). Essa segurança aos pacientes é uma contribuição esperada com a implementação do sistema desenvolvido neste trabalho.

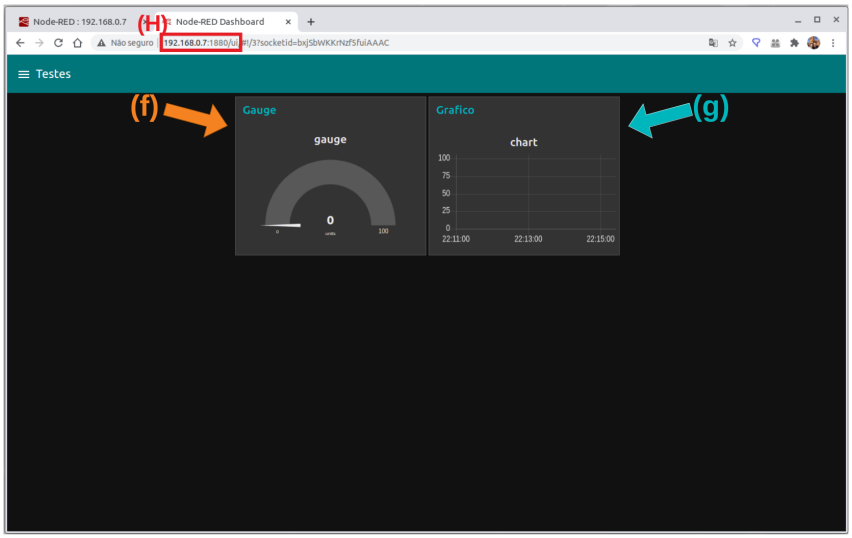

Figura 16 – Interface *Dashboard Node-RED*.

Legenda: (f) nó *gauge* e (g) nó *chart* na interface gráfica do *Node-RED*; (h) endereço *IP* da página de interface.

O projeto do Ministério da Saúde brasileiro chamado *Lean* nas Emergências, obteve resultados significativos na redução da superlotação dos setores de emergências de mais de vinte hospitais do SUS (MINISTÉRIO DA SAÚDE, b). No terceiro ciclo de sua implantação, foi reduzido em média 43% do indicador de superlotação, 39% do tempo médio do paciente no pronto-socorro e 37% do tempo de passagem pela urgência até a alta. Este projeto é uma metodologia japonesa adaptada para utilização na área da saúde e seu principal fundamento é a utilização de indicadores sobre os dados do hospital para tomada de decisões ágeis. Para isso, é necessária a existência de um sistema informatizado e o monitoramento de tempo e dos dados dos pacientes (GOVERNO DO BRASIL,). O presente trabalho espera contribuir suprindo essa necessidade.

Outro projeto brasileiro é o Robô Laura, criado pelo paranaense Jacson Fressato. Ele é um robô cognitivo gerenciador de riscos com IA que monitora os dados dos prontuários do hospital, para detectar e alertar a equipe médica sobre possíveis sinais de sepse nos pacientes (KALIL, 2017). Segundo os dados da empresa responsável pelo robô, foram monitorados em 13 hospitais cerca de 2,5 milhões de pacientes e 12.289 deles foram beneficiados pelos seus métodos de análise. Foi realizada uma pesquisa com 55 mil pacientes, em um período de seis meses antes e depois do uso da tecnologia, no qual apresentaram redução de mortalidade de 25% dos casos e o tempo de internação caiu para 10% (ESTADÃO CONTEÚDO,).

O sistema descrito por (TARYUDI PRASETYO; NUGRAHA; AMMAR, 2019) foi desenvolvido para monitorar e enviar sinais vitais por meio do microcontrolador ESP32, conectado a uma rede Wi-Fi. Segundo os autores, esse sistema permite que os profissionais

de saúde monitorem os dados do paciente em tempo real por meio do aplicativo Blynk, instalado em um smartphone. No projeto, foram utilizados o sensor MAX30100 para medir a frequência cardíaca e a oximetria, o sensor MLX90614 para medir a temperatura corporal e do ambiente, um display OLED para visualização local dos dados e um módulo GPS para determinar a localização do paciente. Embora o sensor MLX90614 seja capaz de medir tanto a temperatura corporal quanto a do ambiente, os autores não apresentam os valores medidos da temperatura ambiente no artigo. Isso é relevante, visto que a temperatura ambiente desempenha um papel crucial no tratamento dos pacientes, como discutido em (SALGADO et al., 2015). Além disso, é importante considerar a quantidade de dados monitorados e a capacidade do display OLED para exibi-los de maneira legível. A sobrecarga de informações pode dificultar a interpretação dos profissionais de saúde que utilizam o sistema. Uma possível solução seria a inclusão de um segundo display.

O sistema desenvolvido por (RIBEIRO, 2020) visa monitorar os sinais vitais de uma pessoa, oferecendo flexibilidade para uso em diferentes ambientes e situações. Baseado na arquitetura, composta principalmente pelo microcontrolador ESP32 e pelo sensor de dados cardíacos MAX30100, o autor propõe que o sistema seja utilizado não apenas em ambientes hospitalares, mas também em ambientes externos, permitindo que o paciente se movimente. Isso é viabilizado pela capacidade energética da arquitetura, fornecida por uma bateria de 1000 mAh. No entanto, o autor ressalta que, para aumentar o tempo de vida útil da bateria, é necessário interromper a execução do ESP32 periodicamente. Embora essa estratégia seja eficaz para economizar energia, o autor reconhece que pode não ser ideal para ambientes que exigem monitoramento contínuo dos sinais vitais, pois a vida útil da bateria pode se tornar uma preocupação adicional em momentos críticos. Assim, embora o sistema tenha sido elogiado pela sua capacidade de medir com precisão os dados de frequência cardíaca e oximetria, é importante considerar suas limitações em relação à autonomia da bateria, conforme observado pelo autor.

No trabalho de (YUSOF; HAU, 2018), foi projetado um mini monitor doméstico portátil para monitorar os sinais vitais, incluindo frequência cardíaca, oximetria e temperatura corporal. Esse monitor utiliza o sensor AD8232 para capturar o sinal de ECG, o sensor MAX30100 para oximetria e o sensor DS18B20 para medir a temperatura corporal. O microcontrolador Arduino Nano é responsável por gerenciar os sensores e exibir os sinais medidos em um LCD. Além disso, o Arduino Nano também é responsável por alimentar um aplicativo móvel por meio do módulo Bluetooth HC-05. Esse aplicativo exibe as medições dos sinais vitais e envia notificações em caso de detecção de alguma anomalia. No entanto, os próprios autores reconhecem que o sistema desenvolvido apresenta algumas limitações, como a voltagem utilizada para alimentar o microcontrolador Arduino Nano. Esta limitação pode ser considerada quando comparada aos recursos de processamento e conexões de rede Wi-Fi disponíveis em outros microcontroladores, como o ESP32.

Capítulo 3

Proposta

A proposta do trabalho consiste no desenvolvimento de um sistema baseado em dispositivos *IoT* com duas arquiteturas para atingir os objetivos descritos na Seção 1.2.

Para isso, foi projetada e desenvolvida uma primeira arquitetura com três microcontroladores ESP32, responsáveis por controlar os sensores e enviar os dados pela Internet (via rede *Wi-Fi*) até a interface do aplicativo *Blynk* que, por sua vez, foi configurado em um *smartphone* conectado à rede 4G. Este aplicativo foi configurado para exibir, em tempo real, os dados medidos pelos sensores e enviados pelos microcontroladores. Para validar o sistema, assegurando a exatidão ou precisão dos dados medidos, foram instalados *displays*, tanto no local da medição (arquitetura de microcontroladores), quanto no local da exibição (aplicativo *Blynk*).

Após a validação da primeira arquitetura, foi projetada e desenvolvida a segunda arquitetura de forma a tornar o sistema eficaz e completo. Além de utilizar três microcontroladores ESP32, esta arquitetura é dotada de um minicomputador (*Raspberry Pi*) onde estão instalados um servidor (*Broker Eclipse Mosquitto*) e uma ferramenta de programação orientada a eventos (*Node-RED*). O servidor (*Broker Mosquitto*) realiza a comunicação (através do protocolo *MQTT*) dos três microcontroladores ESP32 com o sistema desenvolvido no *Node-RED*. Dois dos microcontroladores gerenciam os sensores dos dados vitais e do ambiente e o terceiro gerencia o processo de controle da administração precisa de medicamentos. Há ainda um microcontrolador com recurso de câmera (ESP32-CAM) para visualizar pela interface do sistema do *Node-RED* todas as ações realizadas por um profissional. Além disso, o sistema, ao receber os dados, envia-os através do protocolo *HTTP* às planilhas do *Google Drive* para armazenamento. O armazenamento dos dados vitais e do ambiente também é feito localmente, através de dois módulos que se utilizam de cartões de memória microSD. Para a visualização e controle da exatidão das informações processadas pelo sistema, foram instalados *displays* em pontos estratégicos (onde se localizam os sensores e o módulo *RFID*). Vale ressaltar que todos os dispositivos desta arquitetura encontram-se conectados localmente de forma a assegurar o monitoramento do paciente mesmo que ocorra uma eventual falha de conexão com a Internet.

O detalhamento das duas arquiteturas que compõem a proposta se encontra nas seções a seguir.

3.1 Arquitetura I

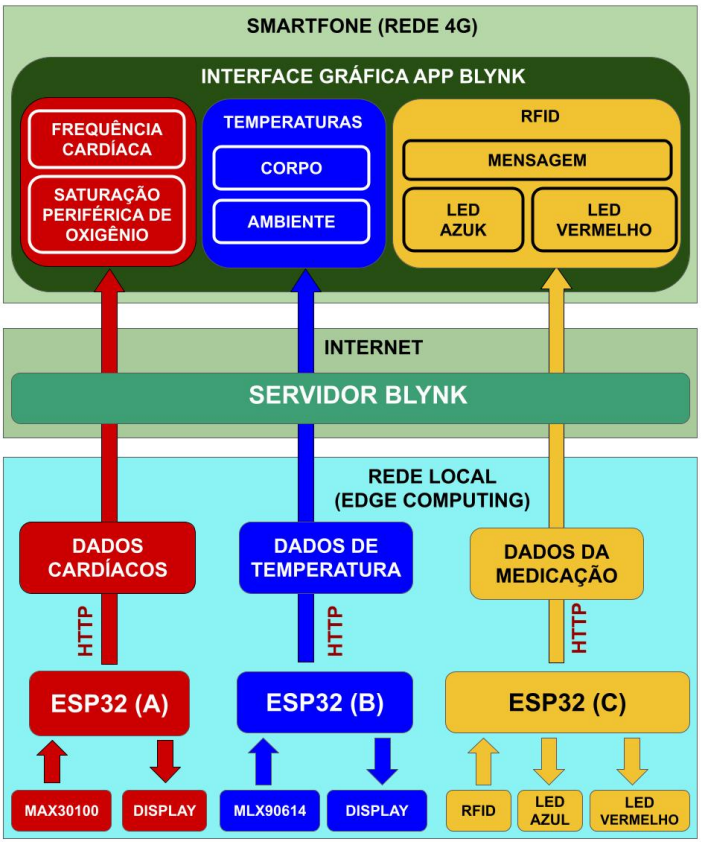

Figura 17 – Arquitetura I em camadas.

Basicamente, como ilustrado na 17 esta arquitetura é composta pelos seguintes processos:

- ❏ Processo para medição das temperaturas do corpo de um paciente e do ambiente. O esquema da Figura 20 apresenta os passos necessários para a execução deste processo envolvendo os componentes de hardware para os dispositivos.
- ❏ Processo de medição da frequência cardíaca e a saturação periférica de oxigênio no sangue de um paciente. O esquema da Figura 21 apresenta os passos necessários para a execução deste processo envolvendo os componentes de hardware para os dispositivos.
- ❏ Processo de controle de precisão na administração de medicamentos a pacientes. O esquema da Figura 22 apresenta os passos necessários para a execução deste processo envolvendo os componentes de hardware para os dispositivos.

A Figura 18 apresenta o cenário da comunicação via Internet entre os dispositivos *IoT* e o aplicativo *Blynk*.

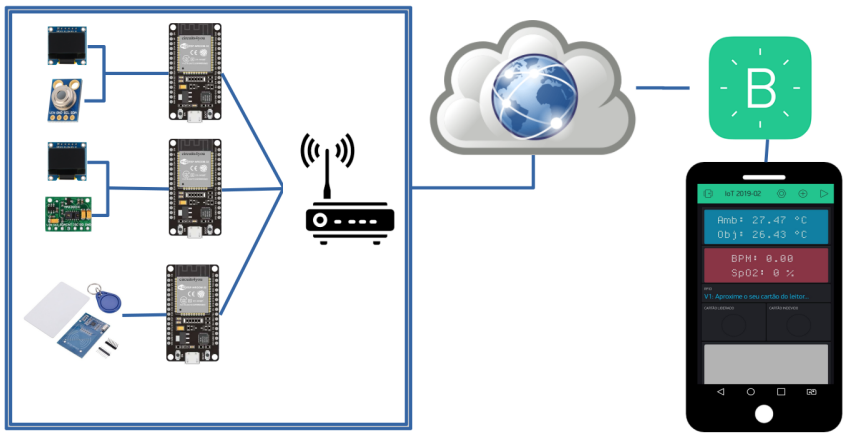

Figura 18 – Cenário de implementação e funcionamento do sistema proposto

Nesse cenário, os principais componentes são os três microcontroladores ESP32 e o aplicativo *Blynk*. A interface do aplicativo está ilustrada na Figura 19.

Figura 19 – Interface do aplicativo *Blynk*

Esta arquitetura utiliza os conceitos da computação de borda pois os três ESP32 atuam de forma distribuída e independentes, processando os dados medidos pelos sensores e exibindo-os em *displays* locais, sem a necessidade de transmissão de dados para processamento. Os microcontroladores também podem enviar os dados já processados para serem exibidos na interface do aplicativo *Blynk*, servindo como um meio de detecção de eventuais problemas no processo.

Dentre outras funções, os ESP32 gerenciam os sensores, os *displays* e o módulo *RFID*, realizam a conexão via rede *Wi-Fi* e enviam os dados desses dispositivos para o servidor do *App Blynk* que alimentou a interface criada em um *smartphone* com Android 7.0.

A arquitetura em questão possibilita o acompanhamento à distância das informações sobre o quadro clínico dos pacientes através dos dispositivos de *IoT* capazes de funcionarem em tempo real.

O esquema apresentado na Figura 20 constitui os passos para monitorar a medição ou coleta das temperaturas do corpo de um paciente e do ambiente (recinto) em que este se encontra. Um microcontrolador ESP32 realiza a conexão com a rede *Wi-Fi* e interage com o servidor do aplicativo *Blynk*. Em seguida, o ESP32 inicializa o *display* e o sensor de temperatura MLX90614. A programação é que, a cada repetição do laço do ESP32, o sensor faça a coleta das temperaturas (corpo e ambiente). A cada coleta realizada, o ESP32 processa e envia as informações para serem exibidas no *display* instalado nele e também na interface do aplicativo *Blynk*.

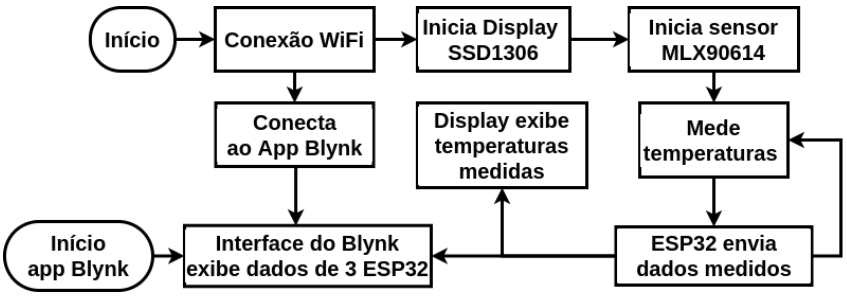

Figura 20 – Esquema do processo de medição das temperaturas do corpo e do ambiente.

O esquema apresentado na Figura 21 mostra os passos que permitem a medição da frequência cardíaca e da saturação periférica de oxigênio no sangue medidos pelo sensor MAX30100. O segundo microcontrolador ESP32 da arquitetura em questão, após realizar a conexão da rede *Wi-Fi* e do servidor de aplicativo *Blynk*, inicializa o *display* e o sensor (MAX30100) para medição de frequência cardíaca e oximetria. A cada repetição do laço programado no ESP32, o referido sensor coleta os dados de um paciente. Em seguida, o ESP32 processa os dados coletados e envia as informações para serem exibidas no seu *display* e na interface do aplicativo *Blynk*.

O esquema apresentado na Figura 22 mostra os passos referentes à execução do processo de controle para a administração segura de medicamentos aos pacientes. O terceiro microcontrolador ESP32 da arquitetura em questão inicia o processo se conectando à rede *Wi-Fi* e ao servidor do aplicativo *Blynk*. Em seguida, o microcontrolador inicia o módulo *RFID* para que este possa ler os dados registrados no dispositivo (cartão ou tag) de acesso por proximidade. Quando ocorre a aproximação de um dispositivo (cartão ou tag), o ESP32 está programado para comparar os dados lidos pelo módulo *RFID* com os programados nele. Dessa forma, o sistema somente libera o procedimento de medicação

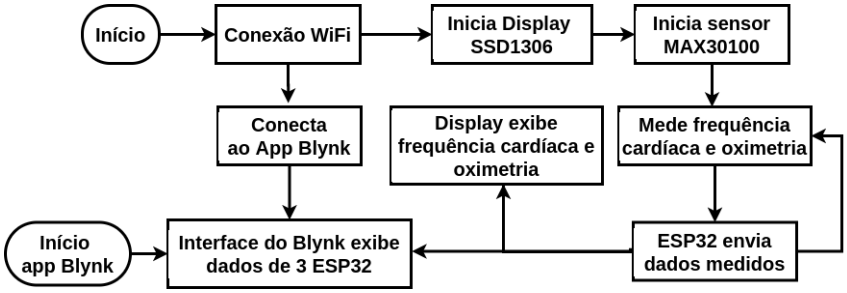

Figura 21 – Esquema do processo de medição da frequência cardíaca e a saturação periférica de oxigênio no sangue.

ao paciente quando o acesso for realizado por um profissional autorizado. O ESP32 indica a liberação através de um *LED* aceso na cor verde e a negação ao acesso através de um *LED* aceso na cor vermelha.

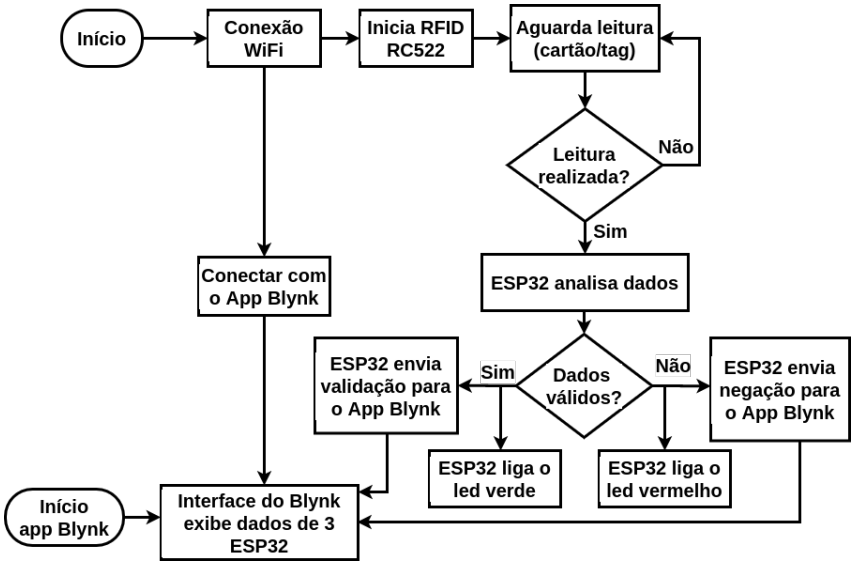

Figura 22 – Esquema do controle de precisão na administração de medicamentos.

3.2 Arquitetura II

Basicamente, como ilustrado na 23 a segunda arquitetura é constituída das seguintes funcionalidades:

- ❏ Uso do minicomputador *Raspberry Pi 3* para hospedar servidor (*Broker Eclipse Mosquitto*) e a ferramenta de programação orientada a eventos conhecida como *Node-RED*;
- ❏ Sistema desenvolvido no *Node-RED* para gerenciar os dados gerados pela arquitetura dos dispositivos *IoT* e apresentá-los em sua interface gráfica;

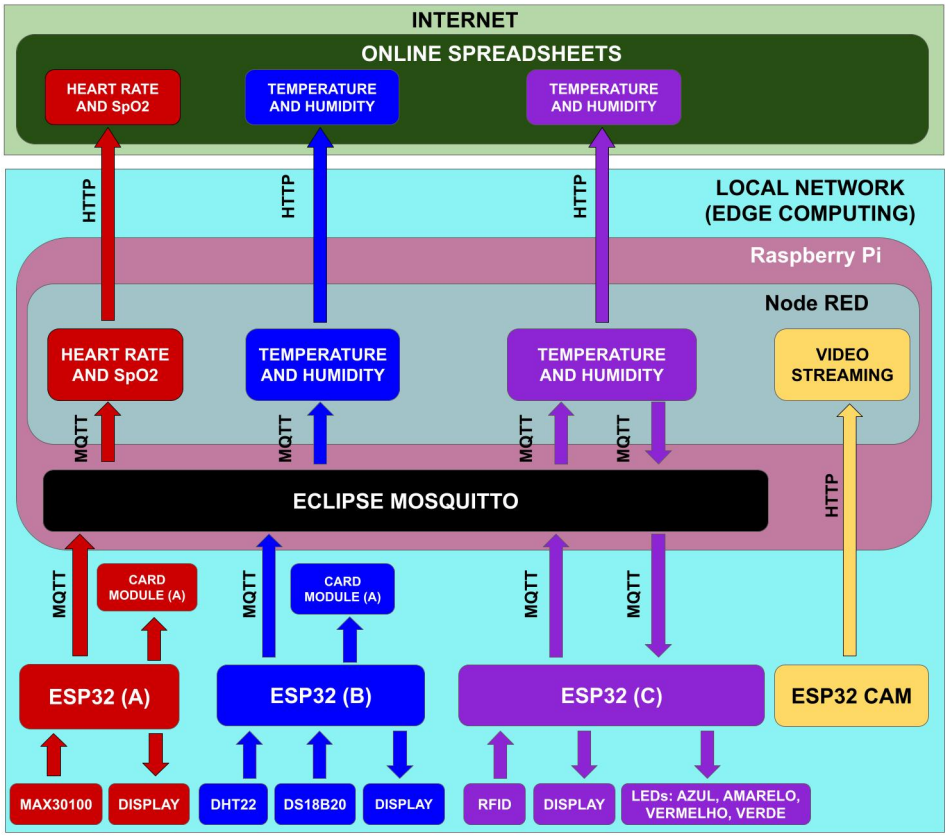

Figura 23 – Arquitetura II em camadas.

- ❏ Envio dos dados coletados pelos sensores e dos dados de controle da administração de medicamentos e armazenamento destes dados nas planilhas do *Google Drive*;
- ❏ Captura do procedimento da administração de medicamentos através da câmera do microcontrolador ESP32-CAM;
- ❏ Captura dos processos de coleta dos dados vitais de um paciente através da câmera do ESP32-CAM;
- ❏ Exibição das imagens capturadas pela câmera do microcontrolador na interface gráfica do sistema no *Node-RED*;
- ❏ Armazenamento dos dados de temperatura corporal, temperatura ambiente, umidade do ar, frequência cardíaca, saturação periférica de oxigênio no sangue e do horário da medição em cartões do tipo microSD;
- ❏ Sincronização dos relógios dos microcontroladores ESP32 com o horário oficial brasileiro através do protocolo de tempo para redes *Network Time Protocol (NTP)*;

O cenário da segunda arquitetura proposta está ilustrado na Figura 24. A importância do minicomputador *Raspberry Pi* nesta arquitetura está relacionada à sua função para

proporcionar, em conjunto aos três microcontroladores ESP32, a computação de borda e preencher as lacunas da arquitetura anterior.

O servidor (*Broker Eclipse Mosquitto*) foi instalado no minicomputador *Raspberry Pi* para gerenciar a transferência de dados entre os três microcontroladores ESP32 e o sistema orientado a eventos criado na ferramenta *Node-RED*. Basicamente, essa comunicação está ilustrada na Figura 24 através dos fluxos representados pelas setas azuis e vermelhas, mostrando a transferência de dados pelos processos de publicação e assinatura no *Broker*, via protocolo *MQTT*.

Vale destacar que todos os dispositivos desta arquitetura encontram-se conectados numa rede local doméstica. O sistema orientado a eventos que foi desenvolvido no *Node-RED* permite receber os dados e apresentá-los em sua interface sem depender de conexão com a Internet. Assim, é possível que o monitoramento do paciente seja feito numa rede local, evitando interrupções indesejáveis neste monitoramento por causa de falhas na Internet. Um recurso adicional e importante do sistema no *Node-RED* é o do envio de dados às planilhas do *Google Drive* e o respectivo armazenamento na nuvem para permitir que os profissionais da saúde tenham acesso ao histórico dos dados à distância e devidamente atualizados, pois estas planilhas são alimentadas em tempo real com dados novos à medida que o monitoramento é feito. Enfim, a arquitetura em questão aproveita os recursos e as vantagens da computação de borda e da computação em nuvem para satisfazer as hipóteses levantadas no trabalho.

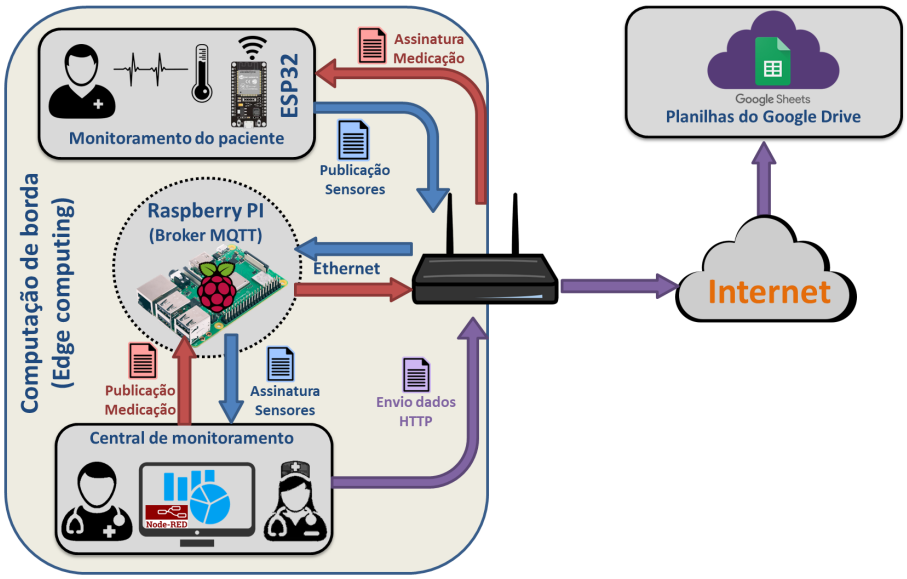

Figura 24 – Cenário da arquitetura II do sistema desenvolvido.

Processo de medição das temperaturas do corpo e do ambiente bem como de medição da umidade do ar

O esquema apresentado na Figura 25 refere-se aos passos para monitorar a medição dos valores de temperaturas do corpo de um paciente e do ambiente (recinto) em que este se encontra bem como da umidade do ar deste recinto e armazená-los localmente e na nuvem da Internet. Um microcontrolador ESP32 realiza a conexão com a rede *Wi-Fi* e com o servidor (*Broker*) instalado no *Raspberry Pi*. Em seguida, o ESP32 inicializa o módulo de armazenamento na memória do tipo microSD, o *display* SSD1306 e consulta o horário oficial brasileiro no servidor *NTP* e armazena-o no microSD.

O ESP32 também inicializa os sensores DS18B20 e DHT22 para medir, respectivamente a temperatura corporal e a temperatura e a umidade do ar do recinto.

Conforme o laço de repetição programado no ESP32, foi definida a medição a cada 2 segundos para observar a variação dos valores medidos pelos sensores.

As funções programadas no ESP32 são: coletar e processar os dados das temperaturas (corpo e ambiente) e da umidade do ar; enviar as informações para serem exibidas no *display* SSD1306 instalado nele; armazenar as informações no cartão microSD instalado no módulo adaptador de cartão microSD; publicar os dados medidos em um tópico de comunicação do *Broker*, que encaminha estas informações para o sistema no *Node-RED*, onde elas serão exibidas em sua interface gráfica e também serão enviadas para a planilha do *Google Drive*.

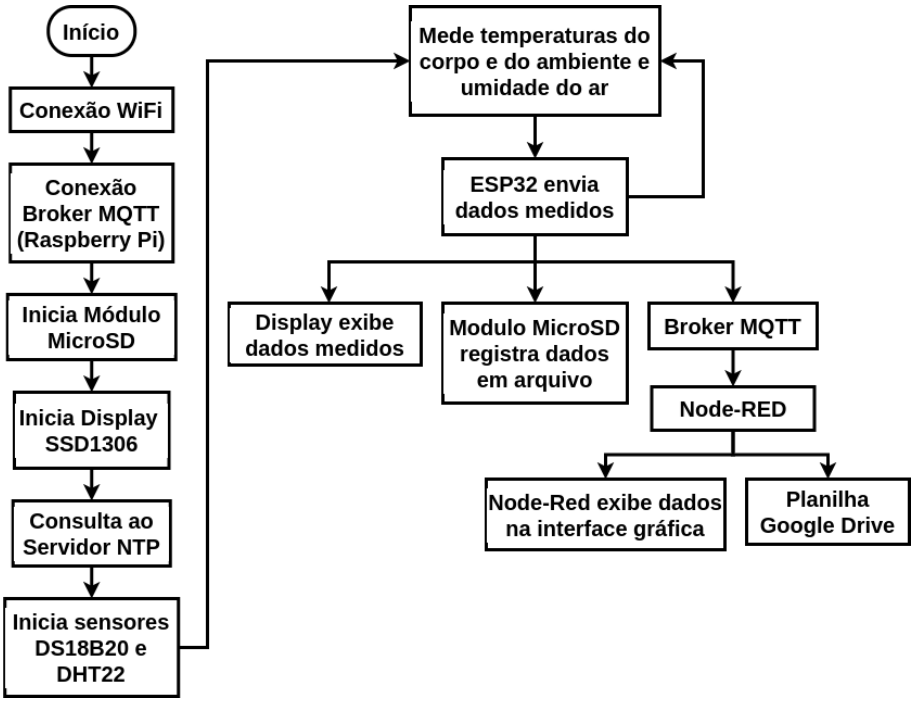

Figura 25 – Esquema do processo de medição das temperaturas do corpo e do ambiente bem como de medição da umidade do ar.

Processo de medição da frequência cardíaca e da saturação periférica de oxigênio no sangue

O esquema da Figura 26 apresenta os passos que permitem a medição e o armazenamento da frequência cardíaca e da saturação periférica de oxigênio no sangue de um paciente. Esse processo é muito similar ao apresentado anteriormente, mudando apenas os parâmetros medidos. Ao invés das temperaturas e da umidade do ar, neste processo medem-se frequência cardíaca e saturação periférica de oxigênio no sangue.

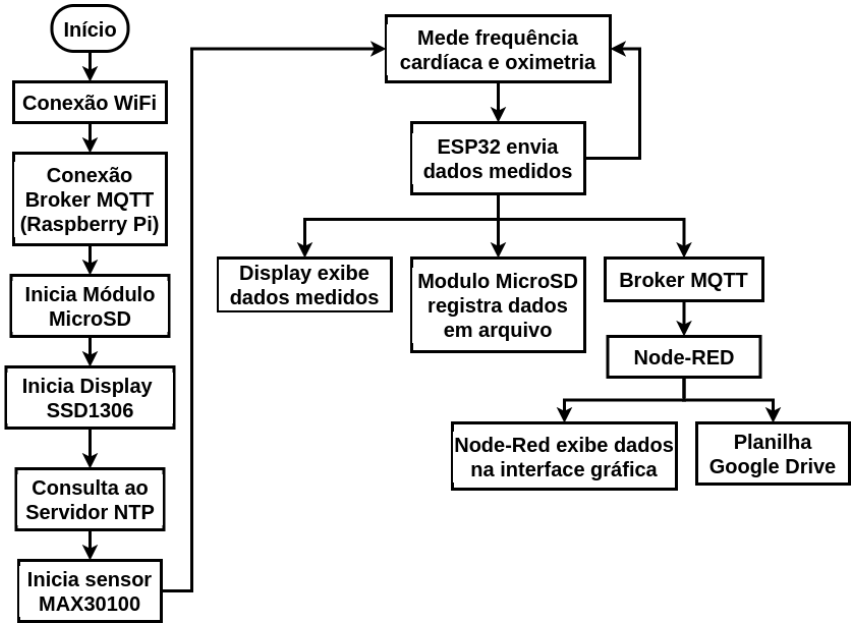

Figura 26 – Esquema do processo de medição da frequência cardíaca e a saturação periférica de oxigênio no sangue.

Processo de controle da administração de medicamentos ao paciente

A Figura 27 é uma representação esquemática dos passos envolvidos no controle preciso da administração de medicamentos de forma a oferecer um tratamento mais seguro ao paciente. Um microcontrolador ESP32 realiza a conexão com a rede *Wi-Fi* e com o servidor (*Broker*) instalado no minicomputador *Raspberry Pi*. Em seguida, o ESP32 inicializa o *display*, consulta e registra o horário oficial brasileiro (servidor *NTP*). Depois disso, o ESP32 inicializa o módulo *RFID*, acende o *LED* amarelo que sinaliza a espera pela prescrição do medicamento e exibe a mensagem "Aguardando Receita" no *display* SSD1306 instalado nele.

Concomitantemente, o ESP32 passa a aguardar o envio da prescrição do medicamento que é realizada na interface gráfica do sistema (*Node-RED*) por um médico (profissional da saúde), quando ele seleciona dentro de uma lista, o medicamento a ser prescrito.

Após o medicamento ser selecionado, o ESP32 apaga o *LED* amarelo, acende o *LED* azul e exibe o nome do medicamento no *display* SSD1306, sinalizando que está aguardando a aproximação do cartão ou tag referente ao medicamento prescrito. Caso o cartão não contenha o registro do medicamento prescrito, o ESP32 acenderá o *LED* vermelho, exibirá no *display* SSD1306 conectado a ele, a mensagem "Erro", e enviará para a interface do sistema uma mensagem de negação ao acesso contendo o nome do medicamento prescrito, o nome do medicamento contido no cartão. Todas essas informações, inclusive o horário (h:m:s) da ocorrência, são armazenadas na planilha do *Google Drive*.

Se o cartão de aproximação for o adequado (com o registro do medicamento prescrito), o ESP32 acenderá o *LED* verde, exibirá no *display* SSD1306 conectado a ele, o nome do medicamento e a palavra "Autorizando" e enviará uma mensagem de autorização para a interface do sistema, com o nome do medicamento prescrito, o nome do medicamento do cartão e a mensagem de autorização. Da mesma forma que no caso anterior, as informações acrescidas do horário da autorização são enviadas à planilha do *Google Drive*.

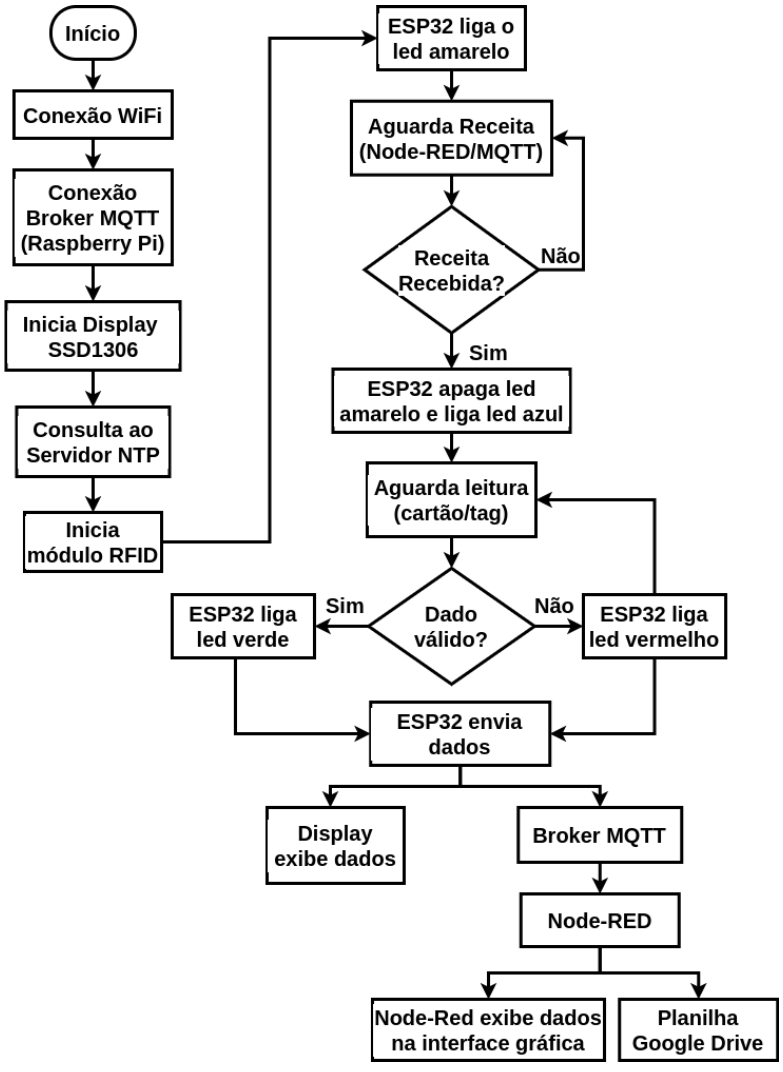

Figura 27 – Esquema do controle de precisão para o procedimento de medicação.

Capítulo 4

Desenvolvimento do sistema

4.1 Desenvolvimento da arquitetura I

4.1.1 Interface do aplicativo *Blynk*

O ambiente utilizado para o desenvolvimento foi o Arduino IDE. Para realizar a comunicação pela Internet, a biblioteca do *Blynk* "BlynkSimpleEsp32.h" foi incluída no código da programação dos microcontroladores ESP32.

Uma interface composta por dois "*Liquid Crystal Display (LCD) Settings*", um "*Value Display Setting*" e dois "*LED Settings*" foi criada no aplicativo *Blynk*. As configurações estão mostradas na Figura 28. O primeiro "*LCD Settings*" (a) foi configurado para exibir os valores das temperaturas de ambiente "Amb:" (a1) e do objeto "Obj:" (a2) medidos pelo sensor MLX90614. O segundo "*LCD Settings*" (b) foi configurado para exibir os valores da frequência cardíaca "BPM:" (b1) e da saturação de oxigênio no sangue "SpO2:" (b2), medidos pelo sensor MAX30100. O "*Value Display Setting*" (c) foi configurado para exibir no campo "*RFID*" (c1), as mensagens enviadas pelo ESP32 durante o processo de leitura do cartão branco e da tag azul pelo módulo *RFID*. O primeiro "*LED Settings*" (d) foi configurado para acender o *LED* verde (d1) quando o *LED* verde equivalente na *protoboard* for acessado, indicando "CARTÃO LIBERADO" (d2). Já o segundo "*LED Settings*" (e) foi configurado para acender o *LED* vermelho (e1) quando o *LED* vermelho correspondente na *protoboard* for acessado, indicando "CARTÃO INDEVIDO" (e2).

4.1.2 Processo de medição das temperaturas do corpo e do ambiente

O *display* SSD1306 foi utilizado para a leitura local dos valores medidos pelo sensor MLX90614. Esta medida é uma segurança para o caso de queda na conexão à Internet. Dessa forma, os valores do *display* e do aplicativo *Blynk* podem ser comparados para garantir a confiabilidade na transmissão das informações. A comunicação do ESP32 com

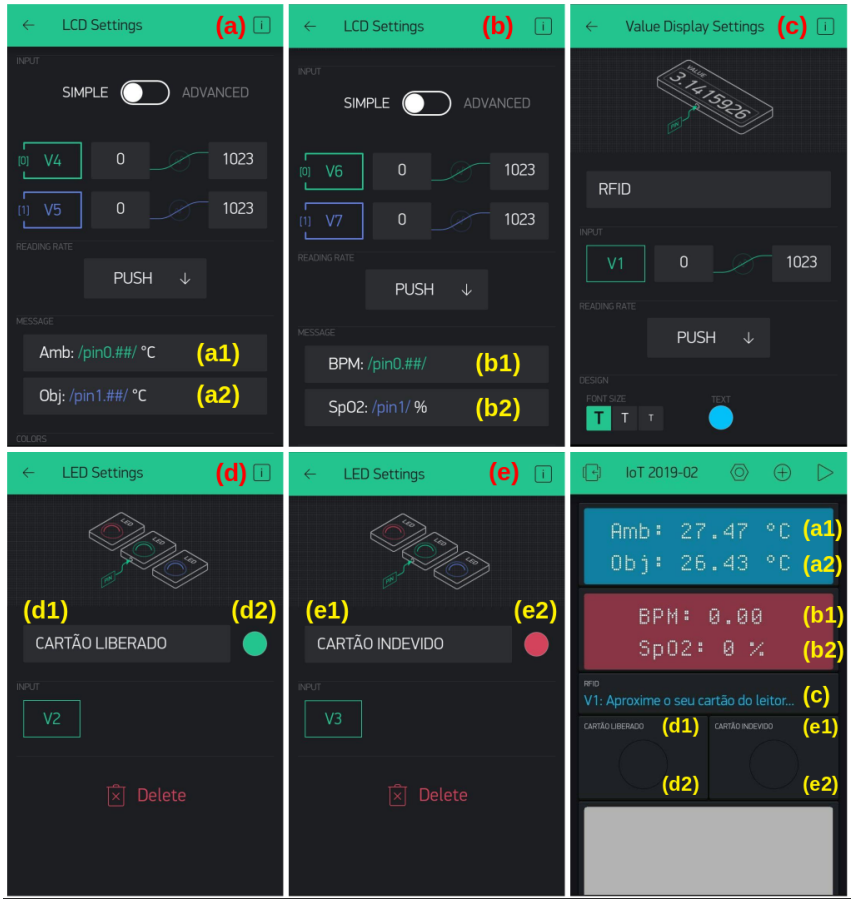

Figura 28 – Configurações realizadas no aplicativo *Blynk*.

o sensor e com o *display* é realizada via interface I^2C e está ilustrada na Figura 29.

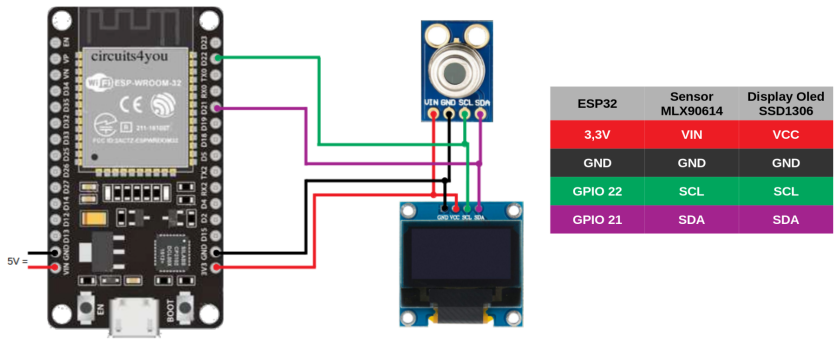

Figura 29 – Conexões entre o sensor MLX90614, ESP32 e o *display* SSD1306.

4.1.3 Processo de medição da frequência cardíaca e da saturação periférica de oxigênio no sangue

O sensor MAX30100 é conectado ao segundo ESP32 para realizar as medições do processo em questão. Dois resistores *pullup* com valor de 4,7 KOhms foram usados para proteger o sensor contra fluxos de correntes indevidas. Também um *display* SSD1306 para a leitura local dos dados foi utilizado. A conexão dos pinos e as imagens do sensor MAX30100, do *display* SSD1306 e do microcontrolador ESP32 estão apresentadas na Figura 30.

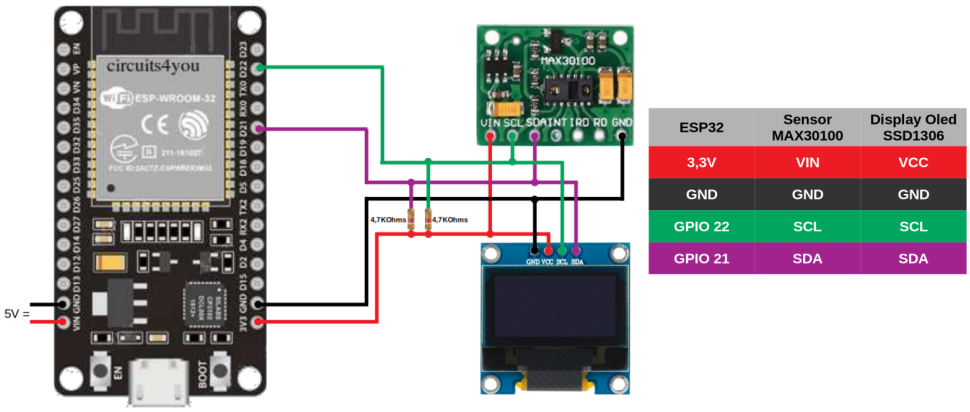

Figura 30 – Conexões entre o sensor MAX30100, ESP32 e o *display OLED*.

4.1.4 Processo de controle de precisão na administração de medicamentos para os pacientes

Para minimizar ou evitar erros na administração de medicamentos, o terceiro ESP32 é responsável pelo controle ou pela autorização dos profissionais escalados para realizar a medicação nos pacientes. Essa parte da arquitetura é composta pelo ESP32, um módulo *RFID* e dois *LED*s (um vermelho e um verde) conectados a resistores de 330 Ohms.

Dessa forma, a autorização dos profissionais ocorre de acordo com as informações contidas nos dispositivos de proximidade (cartão branco ou tag azul). Quando esses dispositivos são aproximados do módulo *RFID*, os dados são lidos e confrontados com a programação do ESP32. Se o profissional for autorizado, o ESP32 enviará uma mensagem com os dados do profissional e o *LED* verde se acenderá na placa. Se a autorização for negada, o ESP32 enviará uma mensagem de "Acesso negado" e o *LED* vermelho será ativado.

O módulo *RFID* é responsável por ler as informações contidas no cartão branco e na tag azul bem como decodificar os dados que são enviados pelo ESP32 para o aplicativo *Blynk*.

As imagens e os pinos de conexão entre o módulo *RFID*, o ESP32 e os *LED*s vermelho e verde, estão apresentados na Figura 31.

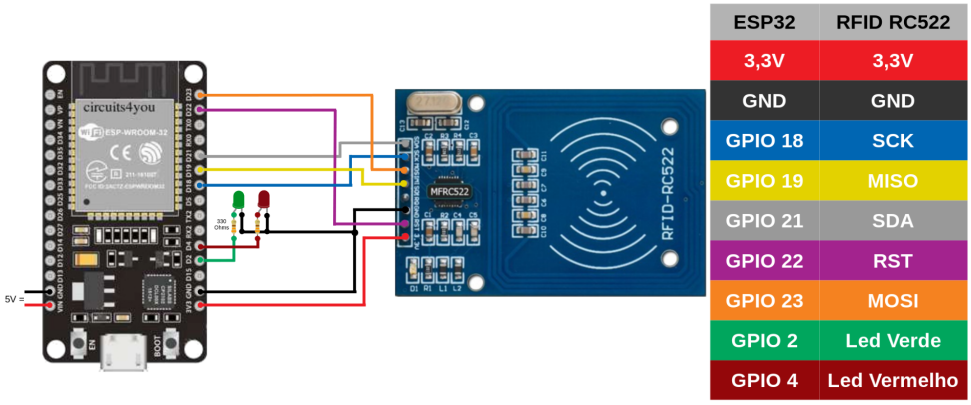

Figura 31 – Conexões entre o módulo *RFID* RC522, o microcontrolador ESP32 e os *LED*s vermelho e verde.

4.1.5 Fonte de alimentação da arquitetura do projeto

Para a alimentação dos dispositivos desta arquitetura *IoT* foi utilizada uma fonte de celular. A referida fonte possui valores nominais de entrada para tensão de 127 V a 240 V e valores de saída de 5 V e 2 A. A Figura 32 mostra a fonte usada na montagem do protótipo de sistema.

Figura 32 – Fonte de alimentação do protótipo para o sistema desenvolvido

4.2 Desenvolvimento da arquitetura II

Da mesma forma que a arquitetura I, esta arquitetura também utiliza três microcontroladores ESP32, mas de maneira mais robusta e eficiente.

A arquitetura foi desenvolvida para monitorar, armazenar em cartões microSD *(data-logger)* e enviar para planilhas do *Google Drive*, os dados vitais de temperatura corporal, de frequência cardíaca e de saturação periférica de oxigênio no sangue, assim como os dados complementares de temperatura e de umidade relativa do ar do ambiente em que o paciente se encontra.

Outra função aprimorada nesta arquitetura é o controle mais preciso da prescrição e administração de medicamentos através do armazenamento de dados nas planilhas do *Google Drive*.

Outro recurso resultante do desenvolvimento desta arquitetura, é a utilização de um microcontrolador dotado de câmera (ESP32-CAM) para permitir que os profissionais da saúde possam acompanhar por vídeo e em tempo real, todas as ações realizadas com o paciente monitorado.

Para dar visibilidade aos dados processados, foi desenvolvido um sistema com interface gráfica através da linguagem de programação orientada a eventos *Node-RED*, onde o seu *framework* foi instalado em um *Raspberry Pi* junto com o *Broker* (servidor) *Eclipse Mosquitto*. Este *Broker* foi o responsável por fazer a comunicação entre os três ESP32 e o sistema desenvolvido no *Node-RED*, através do protocolo *MQTT*. O ESP32-CAM se comunicou com o sistema *Node-RED* de forma independente através do protocolo *HTTP* para exibir em sua interface gráfica (*Node-RED*) os vídeos capturados pela câmera do ESP32-CAM.

4.2.1 Programação dos microcontroladores ESP32

O ambiente utilizado para a programação dos microcontroladores ESP32 nesta arquitetura foi o Arduino IDE. No código da programação dos ESP32 foi incluída a biblioteca "AsyncMqttClient.h" para realizar a comunicação deles com o *Broker MQTT*. Foi definida na programação dos microcontroladores ESP32 que a maior prioridade seria estabelecer a conexão com a rede *Wi-Fi* para se manterem conectados ao *Broker* instalado no *Raspberry Pi*. Se a conexão fosse perdida os ESP32 tentariam uma nova conexão de imediato até que ela fosse estabelecida. Este requisito é fundamental para que todos os dados produzidos pelos ESP32 sejam enviados para o sistema *(Node-RED)* através do *Broker*, via protocolo *MQTT*.

Os dados enviados pelo ESP32 são formatados como *strings* onde estes dados são separados por ";". O sistema do *Node-RED* por sua vez, manipula os dados recebidos através de três nós (*split*, *join* e *function*) e envia os dados para as planilhas do *Google Drive* através dos nós *http request*.

4.2.2 Sistema desenvolvido no *Node-RED*

O sistema em questão foi desenvolvido no ambiente do *Node-RED* com suporte do navegador do *Google Chrome*, endereçado no IP "192.168.0.7:1880". Também o endereço "192.168.0.7:1880/ui" foi usado para acessar a interface gráfica do sistema.

O servidor (*Broker*) se comunica com seus clientes através de tópicos que podem ser por mensagem de publicação, quando o cliente envia dados para o tópico, ou por mensagem de assinatura, quando o cliente recebe os dados.

A Figura 33 apresenta o ambiente de desenvolvimento onde os nós de cor lilás se comunicam com o *Broker* através do protocolo *MQTT*. Os nós indicados pelas setas (laranja, vermelha e azul) são referentes às mensagens de assinaturas no *Broker*, e o nó indicado pela seta verde se comunica com o *Broker* através de mensagens de publicações.

Na Figura 33, o nó *split* separa os dados da mensagem (*string*) para torná-los acessíveis de maneira independente, nos dois fluxos que iniciam com nós lilás indicados pelas setas laranja e vermelha. O nó *join* que sucede o nó *split* em cada fluxo, manipula esses dados com eficiência, armazenando-os em um objeto, como se fosse um vetor de elementos.

Na sequência de cada fluxo, o nó *function* por sua vez distribui os dados para cada nó específico da interface gráfica do sistema do *Node-RED* para que, finalmente, as informações possam ser exibidas.

Ainda nos três primeiros fluxos da Figura 33, verifica-se o nó *function* seguido do nó *http request* para enviar adequadamente os dados para as planilhas do *Google Drive*. Particularmente, o nó *"http request"* serve para encaminhar os dados enviados pelo ESP32 via protocolo *MQTT* formatados para o endereço *web* dos formulários que preenchem as planilhas de forma iterativa.

Por fim, os nós identificados pelas letras (a) até (n) na interface gráfica da Figura 33 também exercem uma função importante no programa do *Node-RED*. Esses nós permitem a exibição de forma organizada e intuitiva dos dados monitorados pelos sensores e dos registros em vídeos capturados pelo ESP32-CAM das ações relacionadas à execução dos processos representados pelos fluxos. As representações gráficas das exibições (saídas) associadas aos referidos nós estão devidamente identificadas por letras na Figura 34.

4.2.3 Criação de formulários e planilhas do *Google Drive*

Primeiramente, uma base de dados *online*, gratuita e relativamente simples, foi criada através do *Google* Planilhas, sem necessidade de um servidor dedicado para executar o serviço. É necessária somente uma conta de *Gmail* para ter acesso a essas aplicações. A base de dados é para possibilitar o acesso, a visualização e a edição por vários usuários de forma simultânea e colaborativa.

Em seguida, três formulários relacionados a uma planilha com três tabelas foram criados para registro de dados nas planilhas do *Google Drive* conforme relacionados a

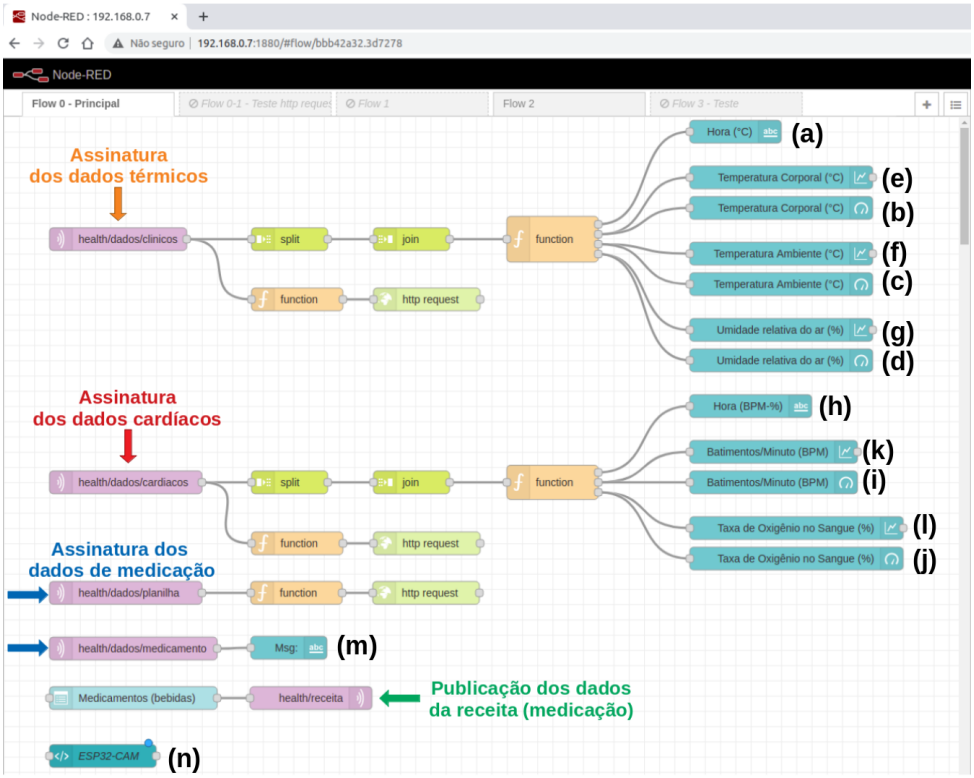

Figura 33 – Ambiente de desenvolvimento do *Node-RED*.

seguir:

- ❏ O primeiro formulário registra na primeira tabela conforme a Figura 35:
  - Os horários (formato h:m:s) de medição de dados pelos sensores (DS18B20 e DHT22), indicados pelas setas amarelas;
  - Os dados da temperatura corporal, indicados pelas setas vermelhas;
  - Os dados da temperatura ambiente, indicados pelas setas azuis;
  - Os dados da umidade relativa do ar, indicados pelas setas verdes.
- ❏ O segundo formulário registra na segunda tabela conforme a Figura 36:
  - Os horários (formato h:m:s) de medição de dados pelo sensor MAX30100, indicados pelas setas amarelas;
  - Os dados da frequência cardíaca, indicados pelas setas vermelhas;
  - Os dados da saturação periférica de oxigênio no sangue, indicados pelas setas azuis.
- ❏ O terceiro formulário registra na terceira tabela conforme a Figura 37:
  - Os horários (formato h:m:s) referentes à administração de medicamentos, indicados pelas setas amarelas;

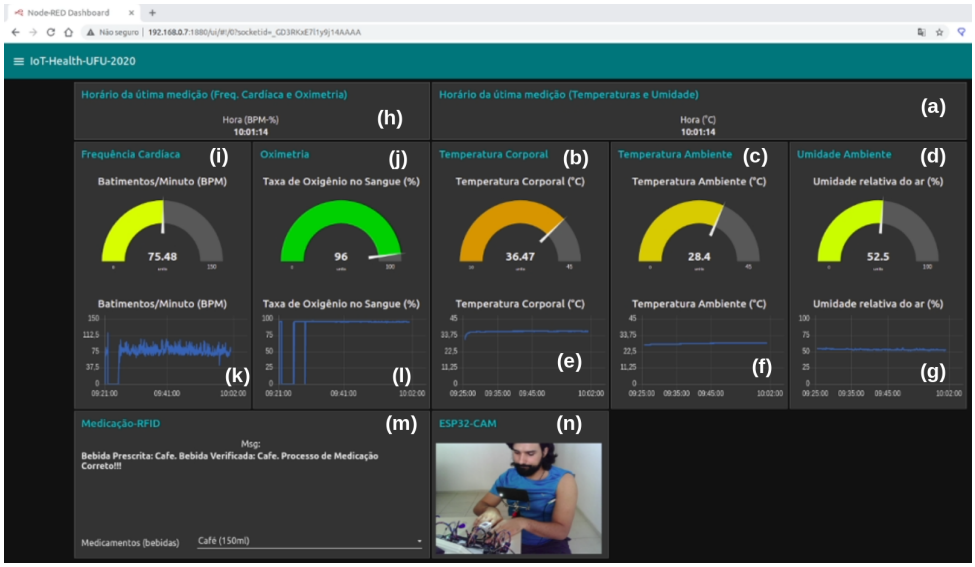

Figura 34 – Interface gráfica do sistema desenvolvido no *Node-RED*.

- A identificação do medicamento prescrito, indicada pelas setas vermelhas;
- O nome do medicamento no cartão de proximidade lido pelo módulo *RFID*, indicado pelas setas azuis;
- O status da validação (“Correta” ou “Errada”) para prosseguimento ou não com a administração do medicamento, indicado pelas setas verdes.

Os processos de controle e envio dos dados pelos ESP32 até o registro na planilha do *Google Drive*, são detalhados nas próximas seções.

4.2.4 Monitoramento dos dados das temperaturas e da umidade

O *display* SSD1306 foi utilizado para que os valores medidos pelos sensores DS18B20 e DHT22 fossem visualizados no local da medição em tempo real. O sensor DS18B20 mede a temperatura corporal do paciente e o sensor DHT22 mede a temperatura e a umidade relativa do ar do ambiente em que o paciente se encontra.

O módulo adaptador de cartão microSD foi usado para armazenar estes dados medidos localmente em um cartão de memória com a capacidade de armazenamento de 2 GB. A conexão entre o microcontrolador ESP32 e os dispositivos mencionados está mostrada na Figura 38.

Quando se inicia o monitoramento, o ESP32 realiza uma consulta no servidor *NTP* para atualizar seu relógio com o horário oficial brasileiro. A prioridade do ESP32 é se manter conectado a rede *Wi-Fi* e também com o *Broker*. A cada laço de repetição, o ESP32 recebe pelos *GPIOs* de dados, os valores medidos pelos dois sensores mencionados e os envia para o *Broker* através de uma mensagem do tipo publicação *(publisher)*. Em

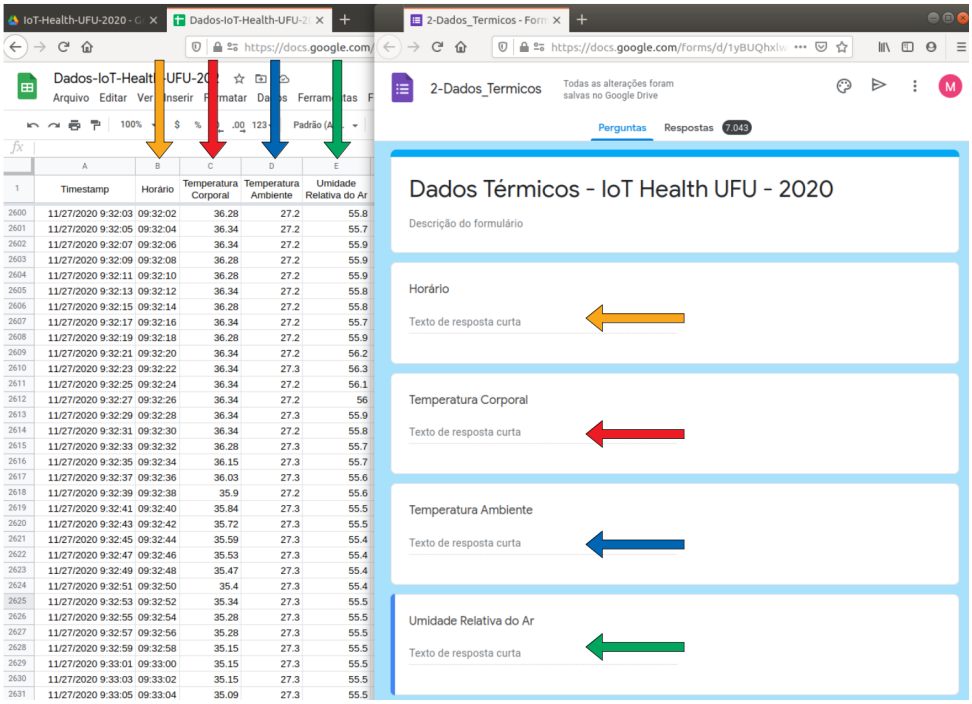

Figura 35 – Formulário e tabela do processo de medição das temperaturas do corpo e do ambiente e da umidade relativa do ar.

seguida, o ESP32 envia os dados para serem exibidos no *display* e registra-os no cartão microSD.

Definiu-se que o formato da mensagem do tipo *string* a ser enviada pelo microcontrolador ESP32 que está conectado aos sensores DS18B20 e DHT22, deve ser da seguinte forma:

$$\text{“}hora; temperaturaCorporal; temperaturaAmbiente; umidade\text{”}$$

O processo do envio desta mensagem de publicação do ESP32 para o *Broker* está ilustrado na Figura 39.

O fluxo apresentado na Figura 40 serve para processar a mensagem em questão e distribuir os dados para cada nó correspondente aos componentes da interface gráfica ilustrada na Figura 34.

A distribuição dos dados da mensagem aos respectivos nós ocorre da seguinte maneira:

- ❏ O horário (h:m:s) para o nó “*text*” chamado “Hora” (a);
- ❏ A temperatura corporal para o nó “*gauge*” (b) e para o nó “*chart*” (e), ambos chamados “Temperatura Corporal (°C)”;
- ❏ A temperatura ambiente para o nó “*gauge*” (c) e para o nó “*chart*” (f), ambos chamados “Temperatura Ambiente (°C)”;
- ❏ A umidade para o nó “*gauge*” (d) e para o nó “*chart*” (g), ambos chamados “Umidade relativa do ar (%)”.

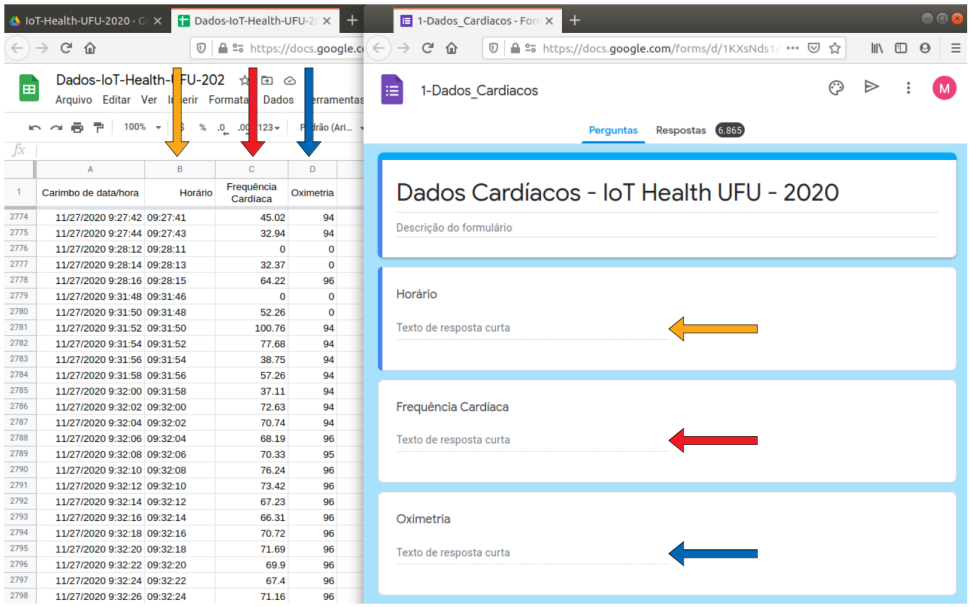

Figura 36 – Formulário e tabela do processo de medição da frequência cardíaca e da taxa de oxigênio no sangue.

Para preencher a planilha da Figura 35, o nó *function* conectado com o nó *http request* responde cada questão do formulário correspondente, através dos *Input Text Box* de cada pergunta. Cada *Input Text Box* tem um número de identificação (ID) descrito no código da página *web* do formulário.

Na Figura 41, é apresentado o formato do *link* contido dentro do nó *http request*, com o endereço *web* do formulário, os códigos de identificação (ID) de cada *Input Text Box* e os dados correspondentes no formato de um objeto do tipo *payload*, criado pelo nó *function*.

4.2.5 Monitoramento dos dados de frequência cardíaca e de saturação periférica de oxigênio no sangue

Um segundo *display* SSD1306 foi utilizado para permitir que os valores de frequência cardíaca e saturação periférica de oxigênio no sangue, medidos pelo sensor MAX30100 fossem visualizados no local e em tempo real. Também para o monitoramento em questão, foi utilizado o recurso para armazenar localmente os dados medidos em um cartão de memória com capacidade de armazenamento de 2 GB, usando o segundo módulo adaptador de cartão microSD. A conexão entre o ESP32 e os referidos dispositivos está mostrada na Figura 42.

O monitoramento se inicia com o ESP32 atualizando o seu relógio com o horário oficial brasileiro consultado no servidor *NTP* para que os dados monitorados possam ser mapeados pelo horário (h:m:s) em que eles foram medidos. Este ESP32 em questão também possui como prioridade, se manter conectado à rede *Wi-Fi* e com o *Broker*.

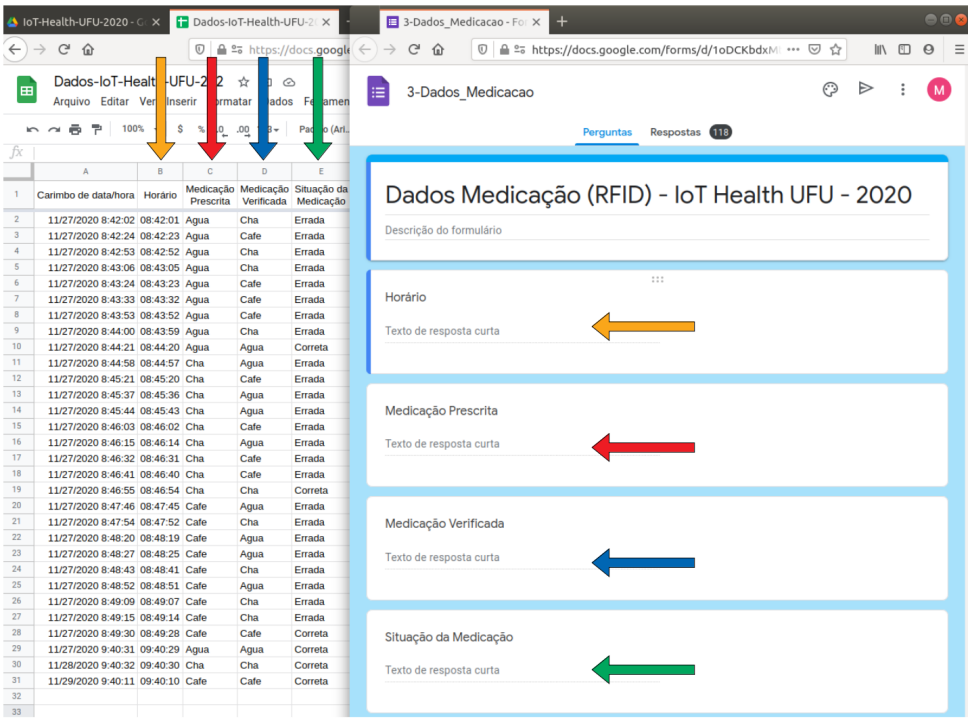

Figura 37 – Formulário e tabela do processo de medicação.

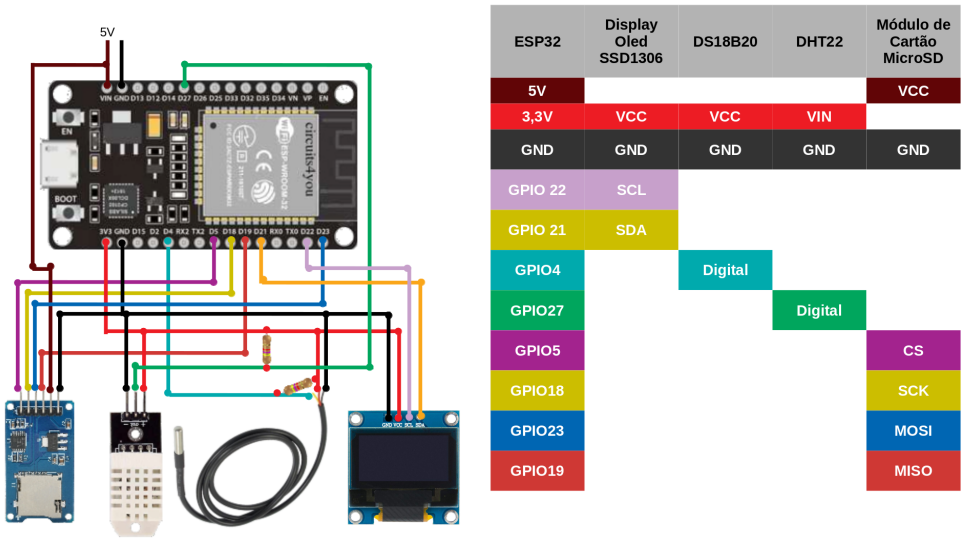

Figura 38 – Conexão entre o ESP32 e os dispositivos para monitoramento térmico e da umidade do ar.

Assim, a cada laço de repetição, irá receber através do protocolo I^2C os dados medidos pelo sensor MAX30100 e os enviará para o *Broker* através de uma mensagem do tipo de publicação *(publisher)*. Em seguida, exibirá os dados medidos no *display* e fará o registro no cartão microSD.

Também, estabeleceu-se que a mensagem a ser enviada pelo microcontrolador ESP32

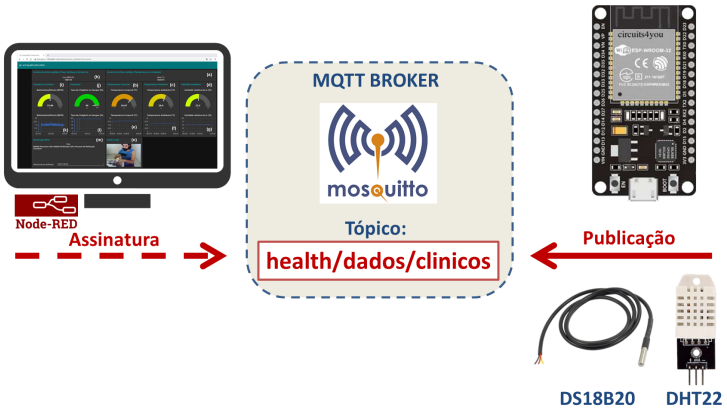

Figura 39 – Publicação do ESP32 no tópico "health/dados/clinicos" do *Broker*.

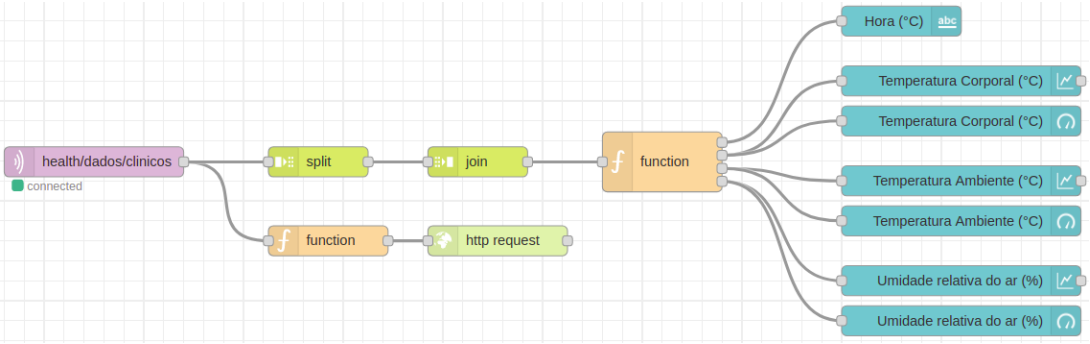

Figura 40 – Fluxo dos dados das temperaturas e da umidade.

```
https://docs.google.com/forms/d/e/<Endereço web do Formulário>/formResponse?ifq\
&entry.(ID hora)={{payload.hora}}\
&entry.(ID Temperatura-Corporal)={{payload.Temperatura-Corpo}}\
&entry.(ID Temperatura-Ambiente)={{payload.Temperatura-Ambiente}}\
&entry.(ID Umidade)={{payload.Umidade}}
```

Figura 41 – Endereço *web* do formulário das temperaturas do corpo e do ambiente e da umidade relativa do ar, referente ao nó *http request*.

conectado ao sensor MAX30100, deve ser formatada como tipo *string* da seguinte maneira:

$$"hora; frequenciaCardiaca; oximetria"$$

O processo do envio desta mensagem de publicação do ESP32 para o *Broker* está ilustrado na Figura 43.

Assim, o fluxo apresentado na Figura 44 serve para processar a mensagem em questão e distribuir os seus dados para cado nó correspondente aos componentes da interface gráfica que se encontra ilustrada na Figura 34.

Os dados da mensagem se distribuem da seguinte forma para cada um dos nós:

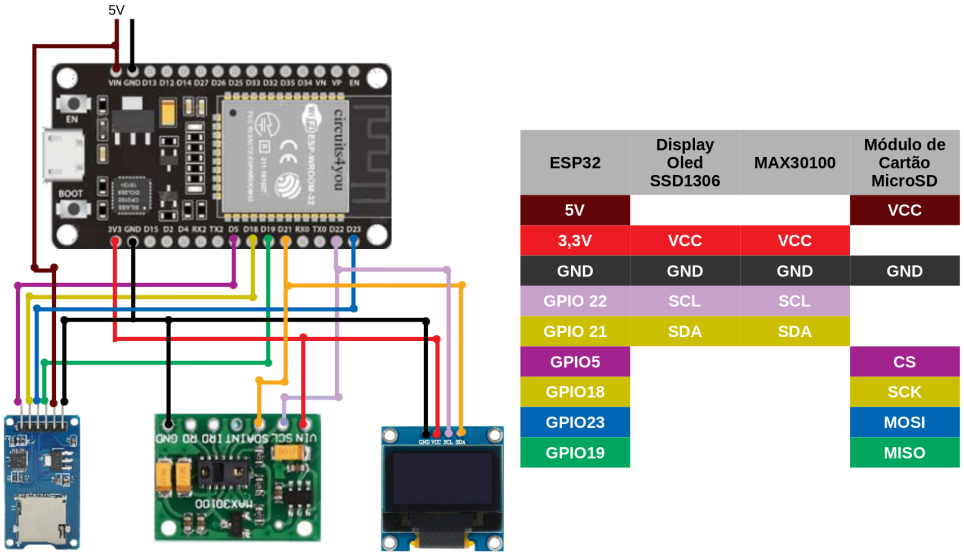

Figura 42 – Conexão entre o ESP32 e os dispositivos para monitoramento dos dados de frequência cardíaca e de saturação periférica de oxigênio no sangue

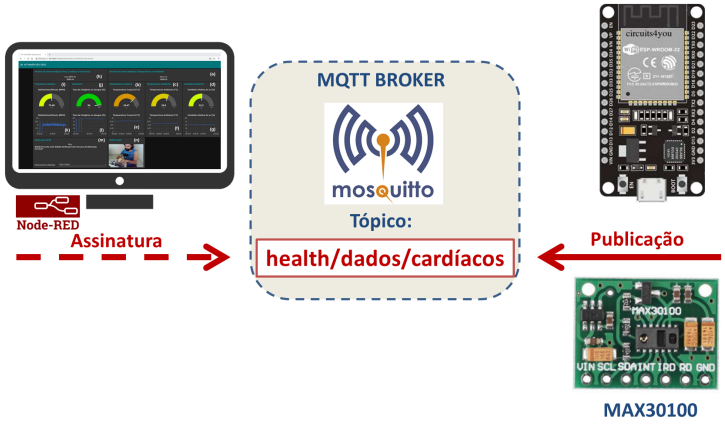

Figura 43 – Publicação do ESP32 no tópico "health/dados/cardiacos" do *Broker*.

- ❑ A hora para o nó "*text*" chamado "Hora (BPM-%)" (h);
- ❑ A frequência cardíaca para o nó "*gauge*" (i) e para o nó "*chart*" (k), ambos chamados "Batimentos/Minuto (BPM)";
- ❑ A saturação periférica de oxigênio no sangue para o nó "*gauge*" (j) e para o nó "*chart*" (l), ambos chamados "Taxa de Oxigênio no Sangue (%)".

Para preencher a planilha da Figura 36, o nó *function* conectado com o nó *http request* responde as questões do formulário correspondente.

Na Figura 45, verifica-se o formato do *link* contido dentro do nó *http request* com o endereço *web* do formulário, os códigos de identificação (ID) de cada *Input Text Box* e os dados correspondentes no formato de um objeto do tipo *payload*, criado pelo nó *function*.

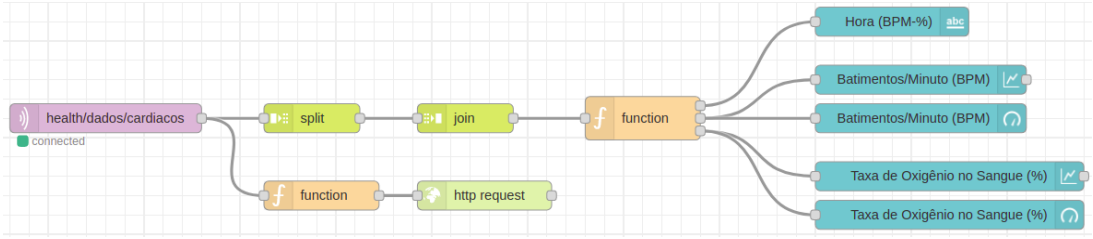

Figura 44 – Fluxo dos dados de frequência cardíaca e saturação periférica de oxigênio no sangue.

```
https://docs.google.com/forms/d/e/<Endereço web do Formulário>/formResponse?ifq\
&entry.(ID hora)={{payload.hora}}\
&entry.(ID Frequência-Cardíaca)={{payload.Frequencia-Cardíaca}}\
&entry.(ID Oximetria)={{payload.Oximetria}}\
```

Figura 45 – Endereço *web* do formulário da frequência cardíaca e saturação periférica de oxigênio no sangue, referente ao nó *http request*.

4.2.6 Controle para a administração precisa de medicamentos

O terceiro microcontrolador ESP32 desta arquitetura serve para controlar a administração precisa de medicamentos funcionando juntamente com o sistema desenvolvido no *Node-RED*. Além do ESP32, compõem a arquitetura, um módulo *RFID*, quatro *LEDs* (amarelo, azul, vermelho, verde) e um *display* conforme se verificam na Figura 46.

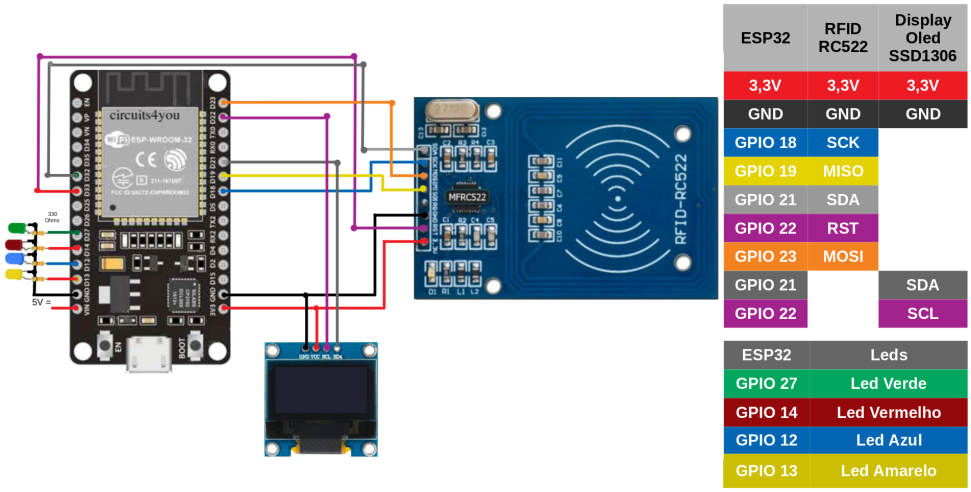

Figura 46 – Conexões dos dispositivos do processo de medicação.

Este ESP32 é o único programado para se conectar ao *Broker* através de três tópicos. A interação com estes tópicos por parte do sistema (*Node-RED*) está ilustrada pelos três fluxos (a), (b) e (c) da Figura 47.

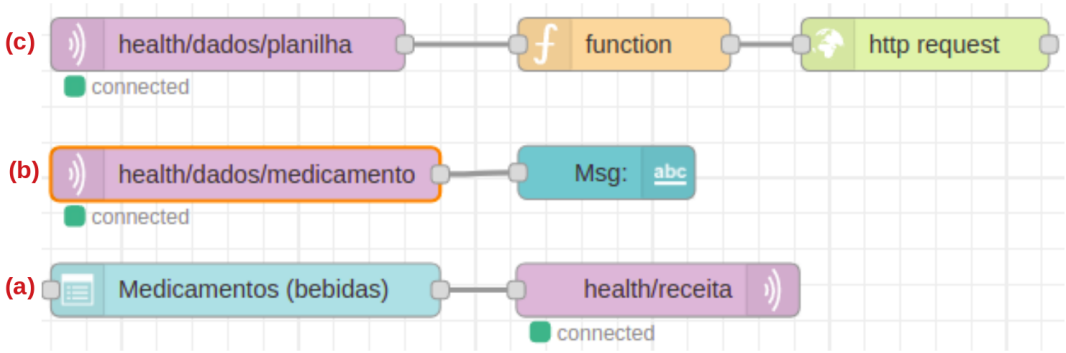

Figura 47 – Fluxos do processo de medicação.

Quando se inicia o controle em questão, o ESP32 se conecta com a rede *Wi-Fi* e também ao *Broker*, através da assinatura do tópico "health/receita" ilustrada na Figura 48. Com essa interação, o ESP32 acende o *LED* amarelo e envia para ser exibida no *display*, a mensagem de "Aguardando Receita", até que o médico (profissional responsável) interaja com a interface gráfica do sistema (*Node-RED*), selecionando em uma lista, o medicamento a ser prescrito, como ilustrada na Figura 49. Nesse processo, o sistema envia o código referente ao medicamento para o microcontrolador ESP32 que está conectado ao módulo *RFID*. Desta forma, o ESP32 exibe no *display* o nome do medicamento que foi selecionado (prescrito) pelo médico e acende o *LED* azul. O referido *LED* indica que o sistema desenvolvido no *Node-RED* está aguardando a confirmação do medicamento por meio da aproximação do cartão de validação (correspondente ao medicamento prescrito) no módulo *RFID*. Esta função se repete todas as vezes que um medicamento é prescrito através da interface do sistema (*Node-RED*).

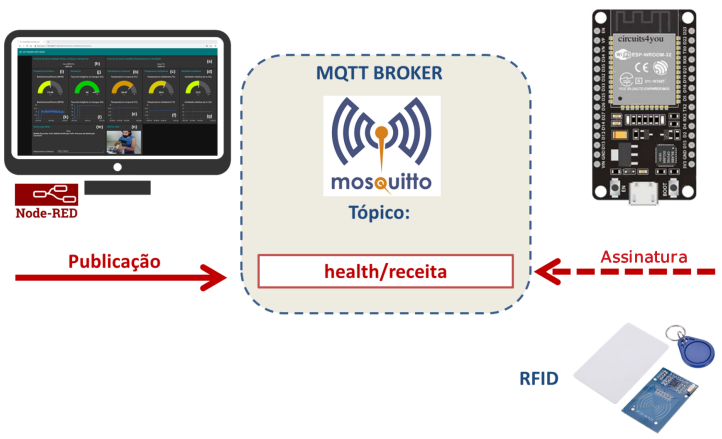

Figura 48 – Publicação do sistema (*Node-RED*) e assinatura do ESP32 no tópico "health/receita" do *Broker*.

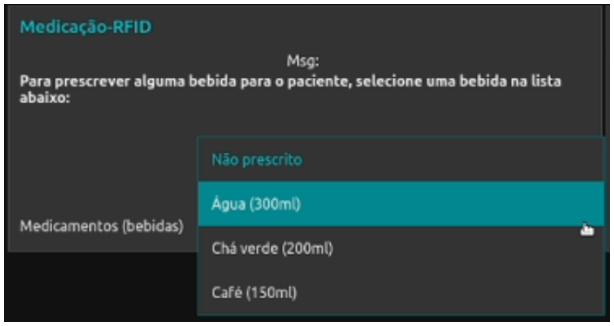

Figura 49 – Lista de medicamentos a serem prescritos na interface gráfica do *Node-RED*.

O sistema *Node-RED* se comunica com o *Broker* pelos fluxos (b) e (c) da Figura 47, através de mensagens do tipo de assinatura. E o ESP32 se comunica com esses fluxos, pelo *Broker*, através de mensagens do tipo de publicação, conforme mostra a Figura 50.

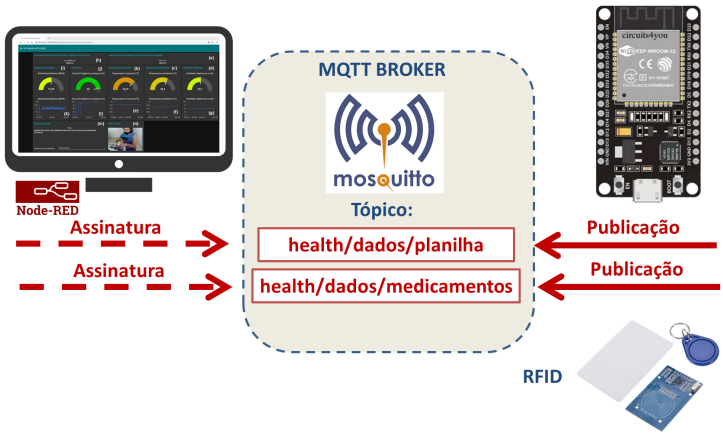

Figura 50 – Publicação do sistema (*Node-RED*) e assinatura do ESP32 no tópico "health/receita" do *Broker*.

Como nos casos anteriores, o sistema *Node-RED* é responsável por registrar na planilha do *Google Drive* (Figura 37) o nome do medicamento prescrito, o nome do medicamento vinculado ao cartão de confirmação e se a administração do medicamento foi autorizada ou não. O fluxo (a) da Figura 47 é o responsável por essa tarefa.

Quando o cartão é lido pelo módulo *RFID*, o ESP32 publica de maneira simultânea nos tópicos "health/dados/planilha" e "health/dados/medicamento" do *Broker*, conforme ilustração da Figura 52. O fluxo (a) é o responsável por receber os dados do tópico correspondente e enviar para a planilha através do nó "*http request*".

A mensagem (*string*) recebida possui o seguinte formato:

$$"hora; bebidaPrescrita; bebidaCartao"$$

A "hora" é referente ao instante em que o cartão foi aproximado do módulo *RFID* para a leitura; a "bebidaPrescrita" é o nome da bebida selecionada na interface gráfica como

mostra a Figura 49. Se o cartão for correspondente à prescrição realizada na interface do sistema, o ESP32 acende o *LED* verde e imprime no *display* a palavra "Autorizado". Se o cartão não corresponder com a bebida prescrita, o ESP32 acende o *LED* vermelho e imprime no *display* a palavra "Erro!!!".

Para preencher a planilha da Figura 37, o nó *function* conectado com o nó *http request*, responde as questões do formulário correspondente. Na Figura 51, é apresentado o formato do *link* dentro do nó *http request* com o endereço *web* do formulário, os códigos de identificação (ID) de cada *Input Text Box* e os dados correspondentes no formato de um objeto do tipo *payload*, criado pelo nó *function*.

```
https://docs.google.com/forms/d/e/<Endereço web do Formulário>/formResponse?ifq\
&entry.(ID hora)={{payload.hora}}\
&entry.(ID Medicacao-Prescrita)={{payload.Medicacao-Prescrita}}\
&entry.(ID Medicacao-Cartão)={{payload.Medicacao-Cartão}}\
&entry.(ID Situacao-da-Medicação)={{payload.Situacao-da-Medicação}}
```

Figura 51 – Endereço *web* do formulário do processo de medicação, referente ao nó *http request*.

Além das mensagens enviadas para alimentar a planilha do *Google Drive*, as mensagens exibidas no *display* são enviadas pelo ESP32 para o fluxo (b), através do *Broker*, como mensagens de texto corrido para serem impressas na interface gráfica sistema. Estas mensagem servem para informar remotamente ao médico (profissional da saúde), os procedimentos realizados durante as leituras dos cartões no módulo *RFID*. Ou seja, quando um medicamento é prescrito e depois verificado pela leitura do cartão, é exibida uma mensagem na interface gráfica com o nome do medicamento prescrito, o nome do medicamento referente ao cartão e se o procedimento foi autorizado. Isso proporciona mais segurança ao paciente.

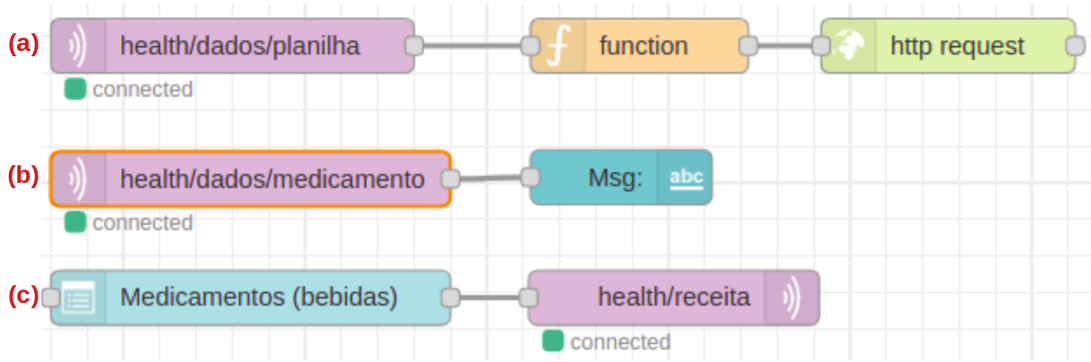

Figura 52 – Fluxo de dados para o controle da administração de medicamentos.

4.2.7 ESP32-CAM no sistema *(Node-RED)*

O ESP32-CAM foi usado com o sistema *(Node-RED)* para permitir aos profissionais da saúde o acompanhamento visual de toda a interação realizada com o paciente monitorado, através da interface gráfica do sistema. O ESP32-CAM foi programado para se conectar com a rede *Wi-Fi* e disponibilizar suas imagens capturadas através de um endereço IP.

O fluxo do sistema *(Node-RED)* com apenas um nó da Figura 53 é o responsável por obter as imagens capturadas do ESP32-CAM e exibi-las na interface gráfica do sistema.

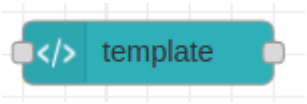

Figura 53 – Fluxo do ESP32-CAM.

4.2.8 Fontes de alimentação da arquitetura II

Para garantir que todos os dispositivos fossem energizados de maneira adequada, foram usados 5 carregadores de *smartphone* como fontes de energia:

- ❏ (a) Fonte do ESP32 conectado aos sensores DS18B20 e DHT22 ;
- ❏ (b)Fonte do ESP32 conectado ao sensor MAX30100;
- ❏ (c)Fonte do ESP32 conectado ao *RFID*;
- ❏ (d)Fonte do *Raspberry Pi*;
- ❏ (e)Fonte do ESP32-CAM.

Figura 54 – Fontes de alimentação da arquitetura II.

Capítulo 5

Experimentos e resultados

5.1 Comprovação científica das medidas realizadas

Ressalta-se que a escolha dos sensores usados para a obtenção dos resultados foi definida pelo processo de atendimento hospitalar ao paciente por meio da triagem em que se medem Temperatura Corporal (TC), Frequência Cardíaca (FC) e Saturação Periféria de Oxigênio no Sangue (SPO).

O estado térmico do paciente é uma referência importante para direcionar a conduta do profissional da saúde. É de conhecimento que a TC varia de acordo com a pessoa, idade, ambiente e atividade e a hora do dia. Segundo o "Guia de prática clínica: Febre", a TC interna normal está entre $36,5\,°C$ e $37,5\,°C$. E as formas mais eficientes para realizar a medição são via retal, oral, axilar, temporal e timpânica, como estão apresentadas na tabela da Figura 55 (CONSELHO FEDERAL DE FARMÁCIA, 2018).

Procedimento Mensuração	Normotermia
Retal	**36,6 °C e 38 °C**
Oral	**35,5 °C e 37,5 °C**
Axilar	**34,7 °C e 37,4 °C**
Temporal	**36,6 °C e 37,8 °C**
Timpânica	**35,7 °C e 37,8 °C**

Figura 55 – Tabela das áreas para medir a temperatura corporal.

A medição da FC é considerada complexa por causa de diversos fatores que podem alterar a quantidade de batimentos por minuto do coração, como a idade e o peso. E ainda, depende se a pessoa, mesmo estando em repouso no momento da medição, fez alguma atividade ou não, está estressada ou não. Por isso, os batimentos considerados normais para adultos em repouso vão de 50 até 100 bpm (MASON et al., 2007). Sabe-se

que um fator importante de um coração saudável é a capacidade dele conseguir acelerar e desacelerar para atender à necessidade de oxigênio conforme as atividades que variam ao longo do dia. Estudos indicam que o nível de SPO de uma pessoa saudável varia de 95% até 100% (WORLD HEALTH ORGANIZATION, c).

5.2 Experimento I

5.2.1 Resultados do experimento I

Os resultados obtidos neste experimento servem para certificar o funcionamento da interação entre a interface do *Blynk*, os sensores e os microcontroladores ESP32 para coletar os dados clínicos de um paciente e transferi-los através da Internet para mostrá-los no *smartphone* do profissional da saúde que é responsável pelo acompanhamento.

A Figura 56 mostra uma instalação dos dispositivos em funcionamento. Os dispositivos se encontram alimentados apenas por uma fonte com valores nominais de entrada *bivolt* (127 V ou 240 V) e saída de 5 V e 2 A.

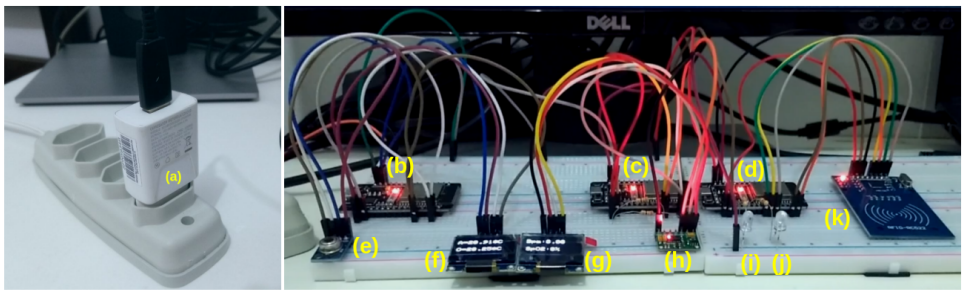

Figura 56 – (a) Fonte de alimentação; (b) Microcontrolador ESP32 para dados de temperatura; (c) Microcontrolador ESP32 para dados de frequência cardíaca e de saturação de oxigênio no sangue; (d) Microcontrolador ESP32 para controle da administração de medicamentos; (e) Sensor MLX90614; (f) *Display* para dados de temperatura; (g) *Display* para dados de frequência cardíaca e de saturação de oxigênio no sangue; (h) Sensor MAX30100; (i) *LED* verde; (j) *LED* vermelho; (k) Módulo *RFID* RC522.

5.2.1.1 Medição das temperaturas do corpo e do ambiente

A Figura 57 mostra como o sensor MLX90614 (a), o ESP32 (b) e o *display OLED* (c) foram dispostos na *protoboard*. Pela característica do funcionamento do laço de repetição do microcontrolador ESP32, foi possível obter a cada segundo, a medição da temperatura do ambiente e também a temperatura do corpo quando aproximado do sensor. Como era esperado, quando não há nenhum corpo a ser medido, a temperatura medida é igual à do ambiente.

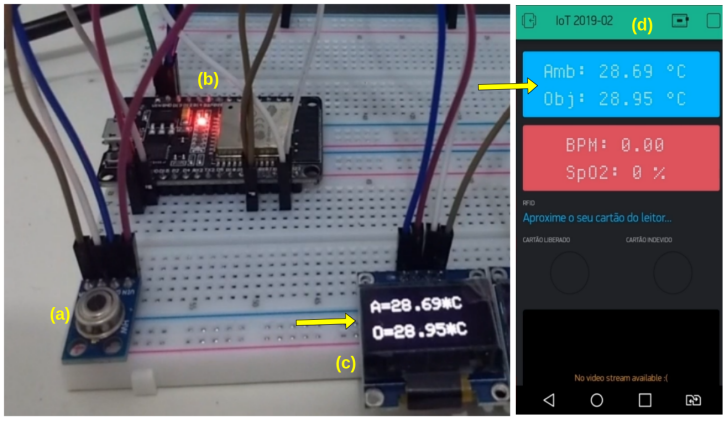

Figura 57 – (a) Sensor MLX90614; (b) Microcontrolador ESP32; (c) *Display OLED*; (d) Interface do *Blynk*.

A Figura 58 mostra o funcionamento do sensor MLX90614 (b) quando o dedo (a) de uma pessoa foi aproximado medindo a sua temperatura sem contato físico e o ESP32 (c) atualizou os dados no aplicativo *Blynk* (e) em poucos milissegundos. Conforme indicado pelas setas na Figura 58, o *display* (d) e o aplicativo (e) exibem os mesmos valores medidos, garantindo a qualidade da comunicação entre o ESP32 (c) e o *Blynk* (e).

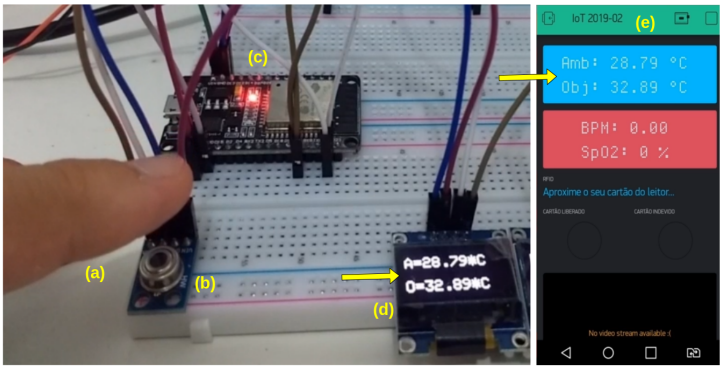

Figura 58 – (a) Dedo de uma pessoa sem febre; (b) Sensor MLX90614; (c) Microcontrolador ESP32; (d) *Display OLED*; (e) Interface do *Blynk*.

5.2.1.2 Medição da frequência cardíaca e da saturação periférica de oxigênio no sangue

O procedimento para medir os dados de frequência cardíaca e da taxa de oxigênio no sangue foi o mesmo do passo anterior. A Figura 59 apresenta a maneira pela qual o sensor MAX30100 (a), o ESP32 (b) e o *display OLED* (c) foram dispostos na *protoboard*. As setas nas imagens indicam que, quando não há contato do dedo do paciente com o sensor, os valores dos dados mostrados são nulos, tanto pelo *display* (c) quanto pelo aplicativo *Blynk* (d).

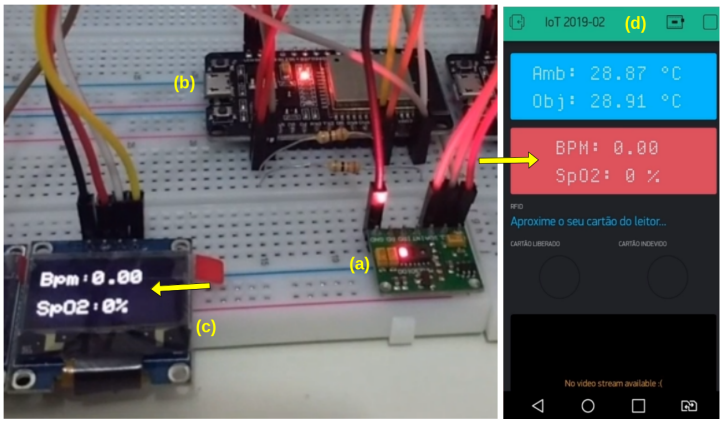

Figura 59 – (a) Sensor MAX30100; (b) Microcontrolador ESP32; (c) *Display OLED*; (d) Interface do *Blynk*.

A Figura 60, por sua vez, mostra a medição da frequência cardíaca e da saturação periférica de oxigênio no sangue de uma pessoa, pelo contato do seu dedo (a) com o sensor MAX30100 (b). O microcontrolador ESP32 (c) atualizou a saída no aplicativo (e) a cada medição feita.

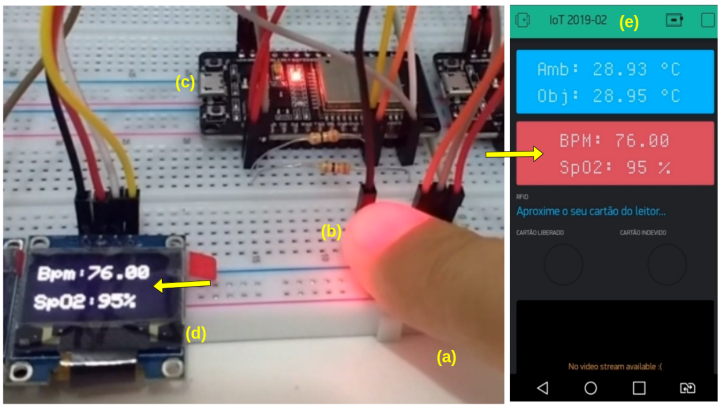

Figura 60 – (a) Dedo de uma pessoa sem problemas cardíacos; (b) Sensor MAX30100; (c) Microcontrolador ESP32; (d) *Display OLED*; (e) Interface do *Blynk*.

5.2.1.3 Simulação do controle de precisão na administração de medicamentos

Para se obter os resultados da simulação do controle em questão, a Figura 61 mostra como o módulo *RFID* (a), o microcontrolador ESP32 (b) e os *LED*s verde (c) e vermelho (d) foram dispostos na *protoboard*, assim como a tela do aplicativo *Blynk* (e) com a área reservada para essa interação. O módulo *RFID* levou ao maior atraso de atualização do aplicativo, próximo a meio segundo, mas ainda dentro da tolerância estabelecida. Esse atraso ocorre por causa da sequência de leitura das informações contidas no cartão branco e na tag azul pelo módulo *RFID* e da atualização do *Blynk* pelo ESP32.

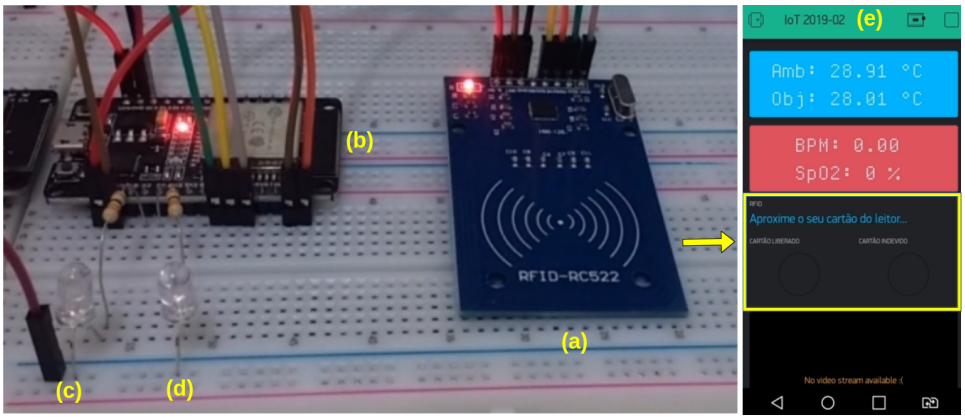

Figura 61 – (a) Módulo *RFID* RC522; (b) Microcontrolador ESP32; (c) *LED* verde; (d) *LED* vermelho; (e) Interface do *Blynk*.

Para preparar a simulação, foram registrados no cartão branco, os dados de uma pessoa autorizada e na tag azul, os dados de uma outra pessoa.

Na Figura 62, o módulo *RFID* (a) entrou em ação para ler os dados do cartão branco e decodificá-los, quando este cartão foi aproximado do módulo. Os dados decodificados por sua vez, foram enviados por meio do protocolo I^2C para o ESP32 (c). Os dados registrados no cartão branco (b) foram definidos como sendo válidos na programação do ESP32. Assim, quando o cartão branco foi posicionado próximo ao módulo *RFID*, o *LED* verde (d) na *protoboard* acendeu, juntamente com o *LED* verde (f) da interface do aplicativo *Blynk* (e). Durante um período de 3 segundos, a mensagem "Acessado por Rafael Marinho" (g) foi exibida também nessa interface com o nome da pessoa autorizada a realizar o procedimento e que estava registrado no cartão.

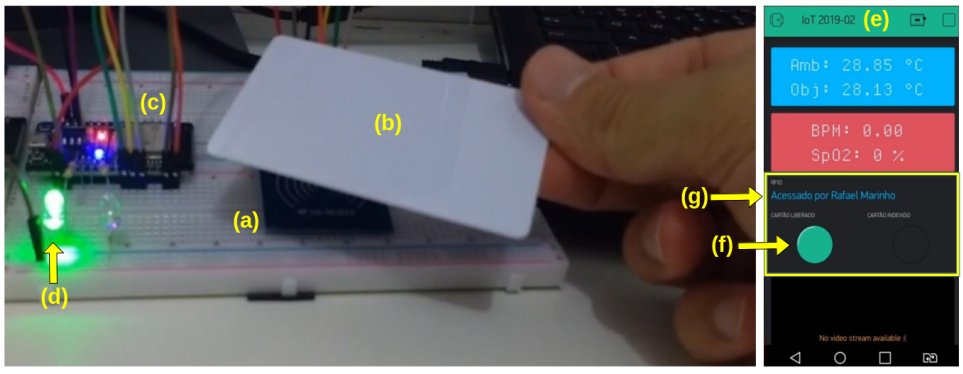

Figura 62 – (a) Módulo *RFID* RC522; (b) Cartão branco; (c) Microcontrolador ESP32; (d) *LED* verde; (e) Interface do *Blynk*; (f) *LED* verde no *Blynk*; (g) Mensagem de autorização sucedida.

A Figura 63 mostra por sua vez o acesso negado pelo ESP32 (c), quando o módulo *RFID* (a) fez a leitura da informação contida na tag azul (b). Como foi definido na

programação do ESP32 que os únicos dados válidos eram os registrados no cartão branco, ao aproximar a tag azul do módulo *RFID*, não ocorreu autorização à pessoa portadora desta tag para realizar o procedimento. Assim, o *LED* vermelho (d) da *protoboard* e o *LED* vermelho (f) da interface do *Blynk* (e) se acenderam. E ainda, a mensagem "Acesso Negado! Cartão Indevido!" (g) foi mostrada nessa interface por 3 segundos.

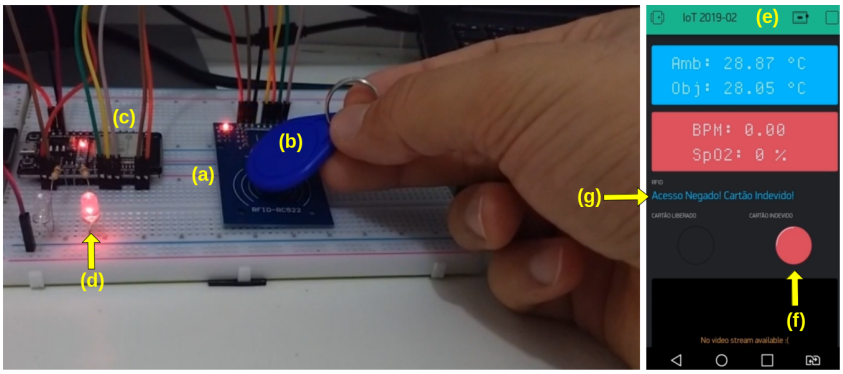

Figura 63 – (a) Módulo *RFID* RC522; (b) Tag azul; (c) Microcontrolador ESP32; (d) *LED* vermelho; (e) Interface do *Blynk*; (f) *LED* vermelho no *Blynk*; (g) Mensagem de autorização negada.

5.2.2 Discussão dos resultados do experimento I

A temperatura do dedo de uma pessoal saudável e sem febre não foi constante por causa da variação da distância entre o dedo e o sensor. O que se verificou é que se manteve estável em $32,9\,°C$ para uma distância de 3 cm. Então, detectou-se a necessidade de calibração da temperatura medida. De acordo com (COSTANZO; FLORES, 2020) é necessário realizar uma calibração no sensor MLX90614 e, para garantir a precisão da medição da temperatura de um objeto, ele deve cobrir totalmente o campo de visão do sensor infravermelho. Mediante este conhecimento e devido ao problema na variação da posição do dedo entre uma medição e outra, decidiu-se pela montagem de um gabarito simples (um pedaço de tubo) onde o objeto cobre totalmente o sensor (b), o dedo do paciente seria apoiado (b) para garantir que a distância seja fixa de 3 cm (a), conforme se verifica na Figura 64.

Há ainda o problema da temperatura do dedo não corresponder de maneira fidedigna à TC interna, por ser um membro periférico e ter a temperatura mais baixa do que as outras partes do corpo indicadas na tabela da Figura 55. Neste caso, justifica-se a necessidade de se medir a temperatura do ambiente pelo sensor em questão, uma vez que a variação da temperatura do ambiente poderá influenciar na temperatura do dedo. O valor da temperatura ambiente medido pelo sensor MLX90614 foi de $28,8\,°C$.

No experimento da medição de FC e de SPO, o sensor MAX30100 foi pressionado pelo dedo de uma pessoa saudável e em repouso e forneceu valores de FC de 76 bpm

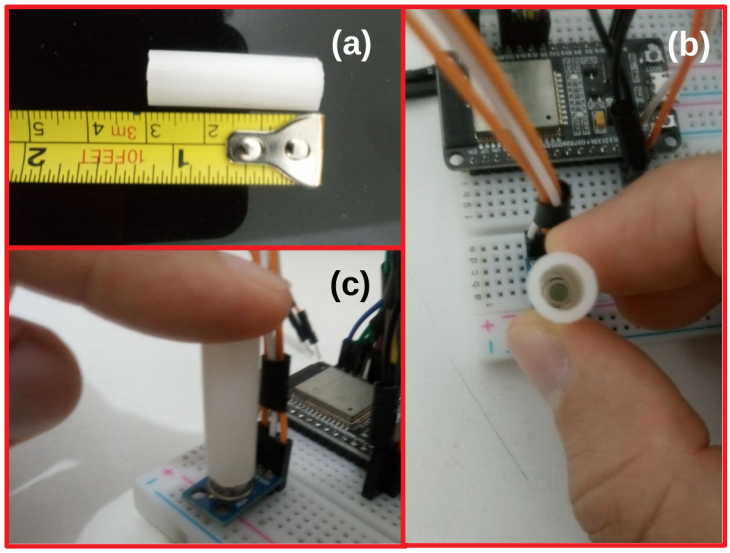

Figura 64 – Apoio de 3 cm para medição da temperatura do dedo de uma pessoa.

e de SPO de 95%, ou seja, os valores estão estritamente dentro do intervalo atestado cientificamente por (MASON et al., 2007), onde os batimentos considerados normais para adultos em repouso vão de 50 até 100 bpm.

Para atestar o funcionamento do controle da precisão na administração de medicamentos, foi simulado o processo de autorização dos profissionais responsáveis pela medicação dos pacientes. Valendo-se da tecnologia *RFID*, dois dispositivos de aproximação receberam dados de pessoas diferentes para simular a capacidade do sistema de autorizar ou não um determinado profissional a fazer a administração do medicamento.

Na primeira simulação, os dados contidos no cartão branco foram definidos na programação do ESP32 como sendo do profissional autorizado. Então, quando o cartão branco foi aproximado do módulo *RFID*, o *LED* verde se acendeu indicando que houve a validação e uma mensagem foi enviada ao aplicativo *Blynk*.

Na segunda simulação, os dados inseridos na tag azul foram definidos na programação do ESP32 como sendo de um profissional não autorizado. Desta forma, quando o cartão azul foi aproximado do módulo *RFID*, o *LED* vermelho se acendeu e uma mensagem foi enviada ao aplicativo *Blynk* avisando que não houve autorização.

Essas simulações resultaram em 100% de acertos nas autorizações ou não dos procedimentos às pessoas registradas nos respectivos dispositivos (cartão branco ou tag azul). Os dispositivos foram posicionados próximos ao módulo por diversas vezes, numa sequência totalmente aleatória, atestando a precisão do sistema para o controle em questão.

Enfim, os sucessos dos experimentos e das simulações comprovados pela geração de resultados que estão em conformidade com os valores atestados pelas referências científicas, evidenciam que o sistema pode ser implementado e adaptado para atender a diferentes situações e necessidades dentro dos objetivos para os quais foi proposto.

5.3 Experimento II

Dentre os objetivos deste experimento, destacam-se:

- ❏ Verificar a operacionalidade e a eficácia do sistema dotado da arquitetura II através de uma simulação experimental com voluntário ingerindo líquidos que representam medicamentos;
- ❏ Verificar a capacidade de armazenamento dos dados (temperaturas do corpo e do ambiente, umidade relativa do ar, frequência cardíaca e saturação periférica de oxigênio) e horários da coleta pelos sensores, nos dois cartões microSD e nas planilhas do *Google Drive*;
- ❏ Validar o sistema quanto ao controle da administração de medicamentos e à capacidade de registro das ações relacionadas a este controle nas planilhas do *Google Drive*;
- ❏ Validar as medidas realizadas pelos sensores de temperatura (DS18B20), de umidade (DHT22) e de frequência cardíaca e taxa de oxigênio no sangue (MAX30100) utilizando como referências os dispositivos de medição aprovados pelos órgãos reguladores;
- ❏ Verificar a confiabilidade do microcontrolador (ESP32-CAM) dotado de câmera para registrar as imagens das ações tomadas durante o processo de administração de medicamentos a um paciente.

Este experimento foi realizado durante 3 dias consecutivos para validar a arquitetura II com a finalidade de atestar a eficácia do sistema desenvolvido para monitorar o quadro clínico de um paciente, coletando, enviando e armazenando os dados. Estabeleceu-se como duração prevista de cada sessão do experimento por dia de 30 minutos, com horário inicial às 9:30 e de término às 10:00 da manhã. Para simular a ingestão por via oral de um medicamento e analisar o seu efeito no paciente, foi elaborado o seguinte protocolo a ser seguido pelo voluntário para cada uma das sessões do experimento:

- ❏ No primeiro dia e no horário estabelecido, ingerir 300 ml de água;
- ❏ No segundo dia e no horário estabelecido, ingerir 200 ml de chá verde;
- ❏ No terceiro dia e no horário estabelecido, ingerir 150 ml de café.

As bebidas acima e suas respectivas dosagens foram escolhidas apenas para simular a ingestão de diferentes medicamentos em diferentes dosagens.

Antes do início de cada sessão, o líquido que representa o medicamento prescrito pelo médico foi devidamente registrado através da interface gráfica do sistema.

Vale destacar que a ingestão do líquido ocorreu após aproximar o cartão correspondente ao medicamento prescrito do módulo *RFID* e receber autorização.

Para satisfazer a exigência de um controle preciso na administração de medicamentos, realizaram-se simulações utilizando dois cartões que não possuem o registro do medicamento prescrito e um cartão com o registro esperado. Nas simulações, o voluntário tenta validar o processo através dos dois cartões inadequados, alternando a ordem deles. Após realizar as tentativas sem sucesso na validação, o voluntário consegue validar o processo com o uso do cartão adequado. Vale lembrar que todas as tentativas têm sido registradas nas planilhas do *Google Drive*, possibilitando o levantamento de causas de eventuais erros na administração de medicamentos. E ainda, objetivando oferecer segurança ao paciente que está em tratamento com medicamentos, todas as ações tomadas pelos profissionais podem ser filmadas através da câmera do ESP32-CAM e as imagens exibidas na interface gráfica do sistema. Essas imagens são apenas capturadas e usadas para acompanhar as ações que envolvem a pessoa monitorada, ou seja, elas não são armazenadas. Mas, para se ter maior controle sobre as análises das ações que foram realizadas, seria interessante gravar em pelo menos duas perspectivas. A primeira pela tela do computador responsável por interagir com a interface gráfica do sistema (*Node-RED*) e a segunda pela câmera de um *smartphone* que filmará a interação da pessoa monitorada com os sensores da arquitetura.

Para atestar a qualidade ou a precisão dos valores medidos pelos sensores da arquitetura, tais valores foram confrontados com os obtidos pelos dispositivos de medição devidamente aprovados pelos órgãos reguladores. Listam-se os referidos dispositivos:

- ❏ Termômetro clínico digital da marca “Bioland”, modelo “T104”, com o número de registro da Anvisa “10410130019”;

- ❏ Termohigrômetro digital do fabricante “Bilbos / Akrom”, modelo “KR42”, com o número do certificado de calibração “2262/2020”;

- ❏ Oxímetro de pulso de dedo da marca“DellaMed”, modelo “MD300C1”, com o número de registro da Anvisa “80795950001”.

5.3.1 Resultados do experimento II

Os resultados obtidos neste experimento tiveram o propósito de atestar o funcionamento esperado e correto da arquitetura II.

5.3.1.1 Medição das temperaturas do corpo e do ambiente

A Figura 65 apresenta a exibição dos valores na interface gráfica do sistema *(Node-RED)* (a) e no *display* (e) conectado ao ESP32 (h). As setas amarelas indicam os dados da temperatura corporal, as setas azuis indicam os dados da temperatura ambiente e as setas vermelhas indicam os dados da umidade relativa do ar. Os valores são exibidos simultaneamente no *display* e na interface gráfica (a). O termômetro clinico digital (c) foi usado para comparar com os dados medidos pelo sensor DS18B20 (b). Já o termohigrômetro digital (i) foi usado para comparar com os dados da temperatura ambiente e da umidade relativa do ar medidos pelo sensor DHT22 (f). Todo o processo foi capturado pela câmera do ESP32-CAM (d) e exibido na interface do sistema (*Node-RED*).

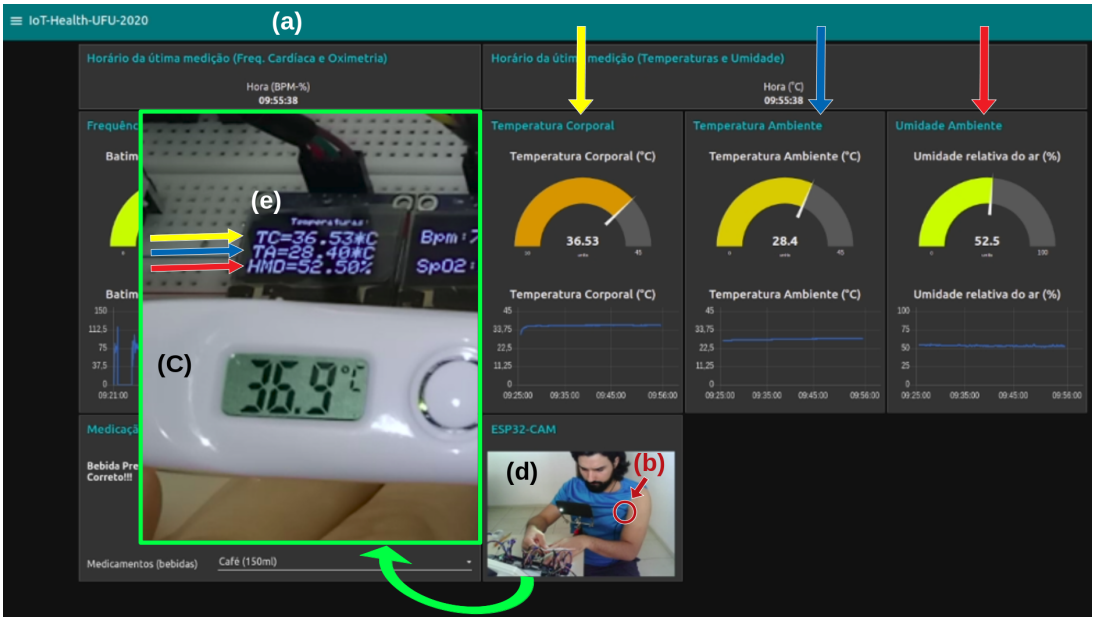

Figura 65 – (a) Interface do sistema (*Node-RED*); (b) Sensor DS18B20; (c) Termômetro externo; (d) Vídeo ESP32-CAM; (e) *display*

As Figuras 66, 67 e 68 apresentam em forma de gráficos, os resultados de todas as medidas das temperaturas do corpo e ambiente e da umidade realizadas durante os três dias da simulação de administração de medicamentos.

5.3.1.2 Medição da frequência cardíaca e da saturação periférica de oxigênio no sangue

A Figura 69 mostra os valores da referida medição, exibidos na interface gráfica do sistema *(Node-RED)* (a) e no *display* (e) conectado ao ESP32 (f). As setas vermelhas indicam os dados da frequência cardíaca e as setas azuis indicam a saturação periférica de oxigênio no sangue. Assim como na medição anterior, os valores foram exibidos simultaneamente no *display* e na interface gráfica (a). O oxímetro (c) foi usado para comparar com

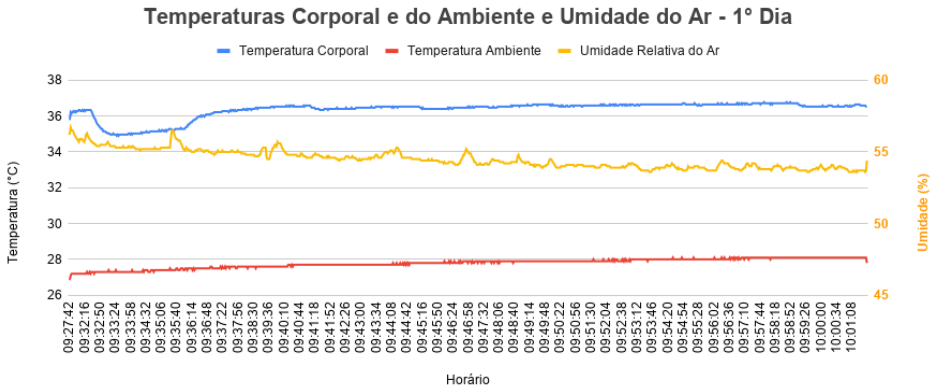

Figura 66 – Medição das temperaturas do corpo e do ambiente dia 1

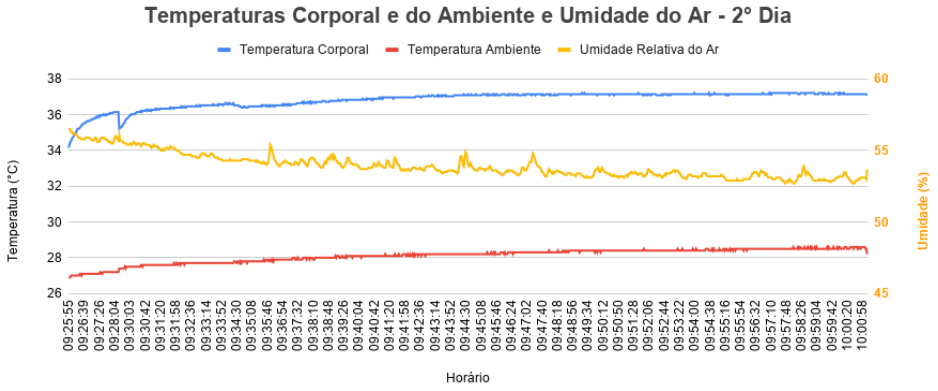

Figura 67 – Medição das temperaturas do corpo e do ambiente dia 2

os dados medidos pelo sensor MAX30100 (b) conectado ao dedo da pessoa monitorada. Todo o processo foi capturado pela câmera do ESP32-CAM (d) e exibido na interface do sistema (*Node-RED*).

As Figuras 70, 71 e 72 apresentam os gráficos referentes aos resultados de todas as medidas das frequências cardíacas e das taxas de oxigênio no sangue realizadas durante os três dias de simulação da administração de medicamentos.

5.3.1.3 Simulação do controle de precisão na administração de medicamentos

Esta simulação gerou resultados através do teste do sistema desenvolvido no *Node-RED* e da arquitetura II com um voluntário. Ao iniciar a simulação, o voluntário selecionou na interface gráfica do sistema (d) um dos nomes da lista de bebidas que representam medicamentos conforme mostrada na Figura 73. Em seguida, uma mensagem é enviada pelo sistema (via protocolo *MQTT*) para o microcontrolador ESP32 (a) contendo o nome da bebida representando o medicamento prescrito. O ESP32 acende o *LED* azul (b),

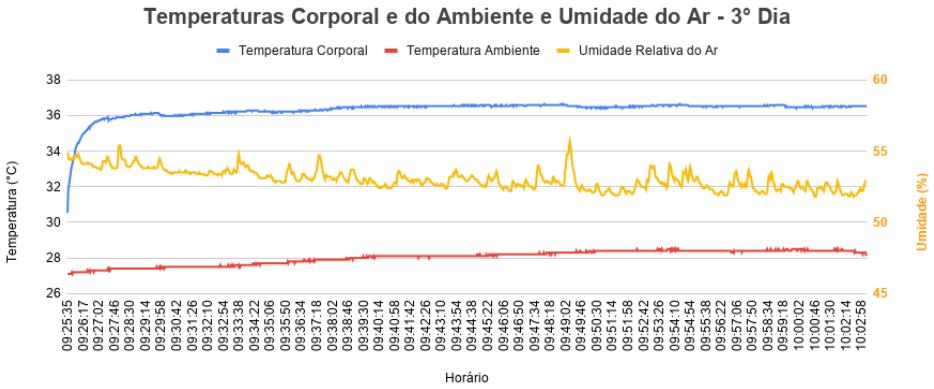

Figura 68 – Medição das temperaturas do corpo e do ambiente dia 3

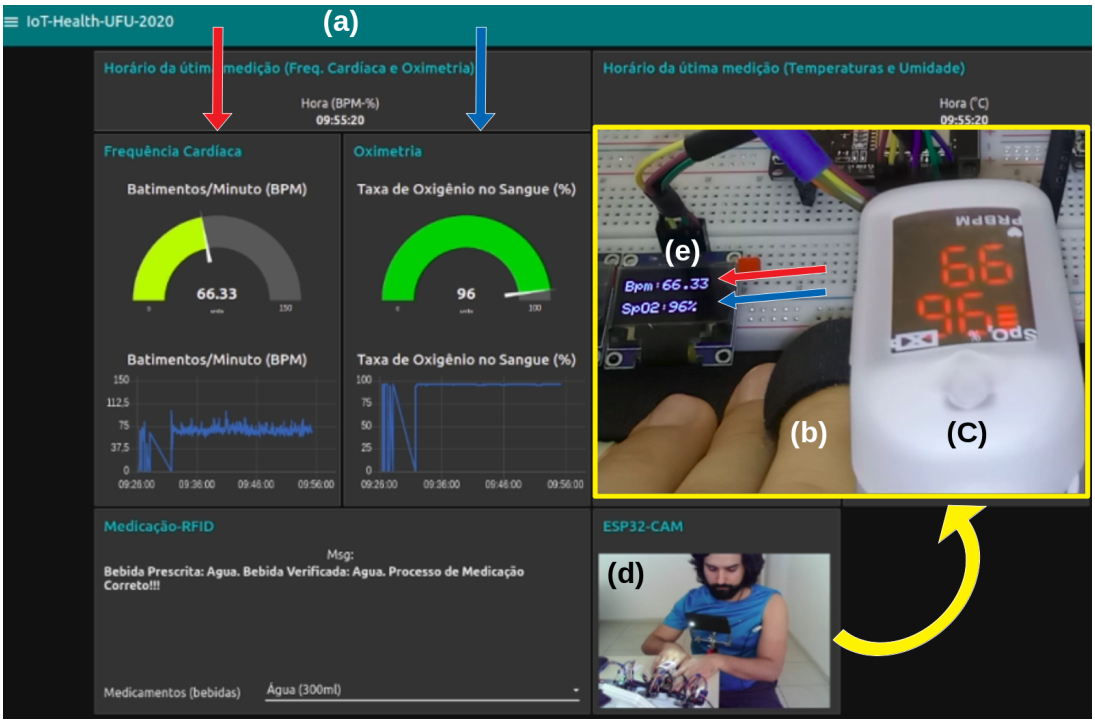

Figura 69 – (a) Interface do sistema (*Node-RED*); (b) Dedo de uma pessoa conectado ao Sensor MAX30100; (c) Oxímetro externo; (d) Vídeo ESP32-CAM; (e) *display*

imprime no *display* (c) uma mensagem com o nome da bebida selecionada. Quando um cartão de proximidade com o registro da bebida a ser validada foi aproximada do módulo de *RFID* (e), houve a autorização para a ingestão da bebida.

Detalhamento da simulação

A bebida "água" foi selecionada na interface gráfica do sistema (d) da Figura 73 e os cartões de proximidade inválidos por hipótese foram apresentados na seguinte sequência:

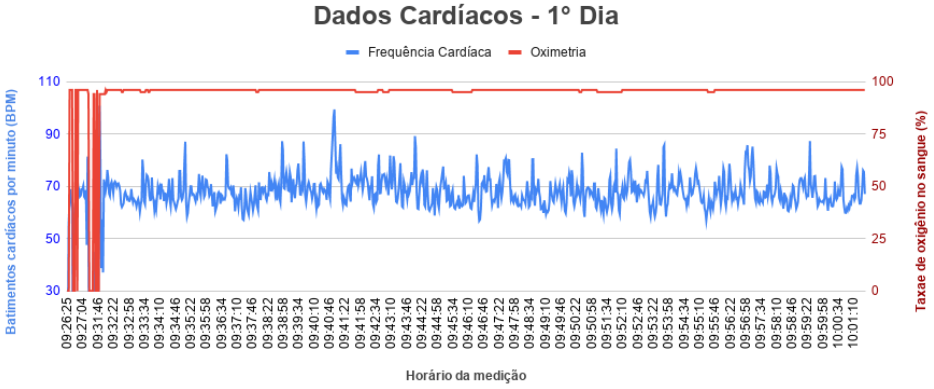

Figura 70 – Medição da frequência cardíaca e a saturação periférica de oxigênio no sangue dia 1

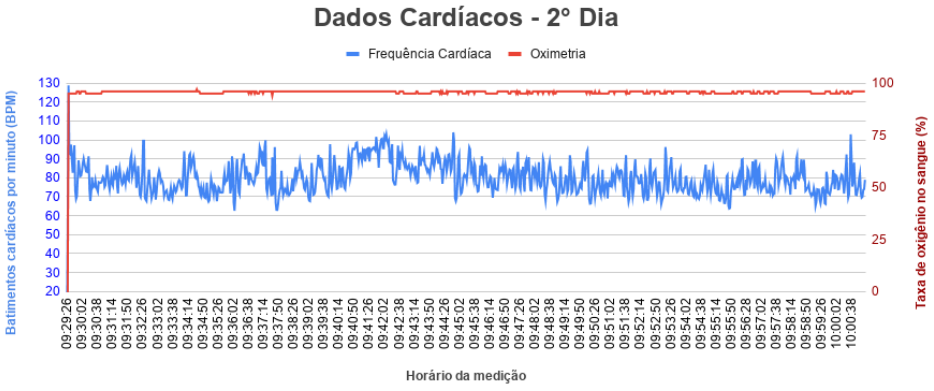

Figura 71 – Medição da frequência cardíaca e a saturação periférica de oxigênio no sangue dia 2

“cha”, “cafe”, “cha”, “cha”, “cafe”, “cafe”, “cafe”, “cha”. Para cada tentativa de validação com esses cartões, o ESP32 acendeu o *LED* vermelho e exibiu no *display* uma mensagem de acesso negado. A mensagem também foi enviada via protocolo *MQTT* para o sistema do *Node-RED* que por sua vez, exibiu a mensagem na interface gráfica e enviou-a à nuvem para ser armazenada na planilha do *Google Drive*. As informações carregadas foram o horário que o cartão foi lido, o nome da bebida prescrita, o nome da bebida registrado no cartão e o resultado da validação que foi negativo para todas as tentativas. Quando o voluntário aproximou o cartão onde estava registrada a bebida “água”, o ESP32 apagou o *LED* vermelho e acendeu o *LED* verde, imprimiu a mensagem de acesso autorizado no *display* e enviou esta mensagem via protocolo *MQTT* para o sistema do *Node-RED*. Este sistema por sua vez, realizou a mesma ação que no caso anterior, apenas alterando a mensagem para o resultado da validação positivo.

O protocolo descrito anteriormente foi repetido para as duas outras bebidas, obtendo-

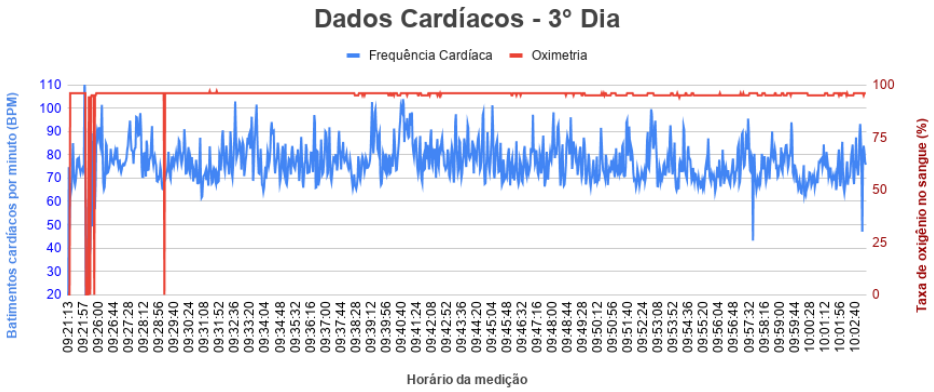

Figura 72 – Medição da frequência cardíaca e a saturação periférica de oxigênio no sangue dia 3

se 9 resultados para cada tipo de bebida e um total de 27 resultados conforme se verificam na tabela da Figura 74.

A tabela da Figura 75 é referente à simulação da administração de medicamentos, realizada durante três dias. Nesta simulação, o objetivo era realizar a prescrição da bebida (representando um medicamento) no sistema (*Node-RED*), realizar a leitura do cartão de maneira correta no módulo *RFID* e depois ingerir a bebida. Os procedimentos foram gerenciados pelo ESP32 em conjunto com o sistema (*Node-RED*) e todas as informações foram devidamente registradas na planilha do *Google Drive*.

5.3.1.4 Validação dos dados transmitidos e armazenados

Após a execução dos experimentos, todos os dados armazenados localmente nos cartões microSD e os enviados para serem armazenados na nuvem (planilhas do *Google Drive*) foram verificados. A validação consistiu em detectar possíveis dados que não foram registrados nos cartões microSD ou dados que não foram enviados para as planilhas do *Google Drive*. O resultado da validação foi que todos os dados coletados foram devidamente armazenados nos cartões e também nas planilhas do *Google Drive* conforme a tabela da Figura 76.

5.3.2 Discussão dos resultados do experimento II

Um parâmetro importante que foi utilizado neste experimento é o horário (o relógio do sistema foi atualizado pelo microcontrolador ESP32 de acordo com o horário oficial brasileiro obtido do servidor *NTP*). Este parâmetro serviu de referência para verificar os tempos envolvidos na medição e no armazenamento dos dados referentes ao monitoramento remoto de um paciente. Assim, durante o experimento, tanto o horário de medição por um sensor, quanto o horário de gravação da coleta na planilha do *Google Drive* foram

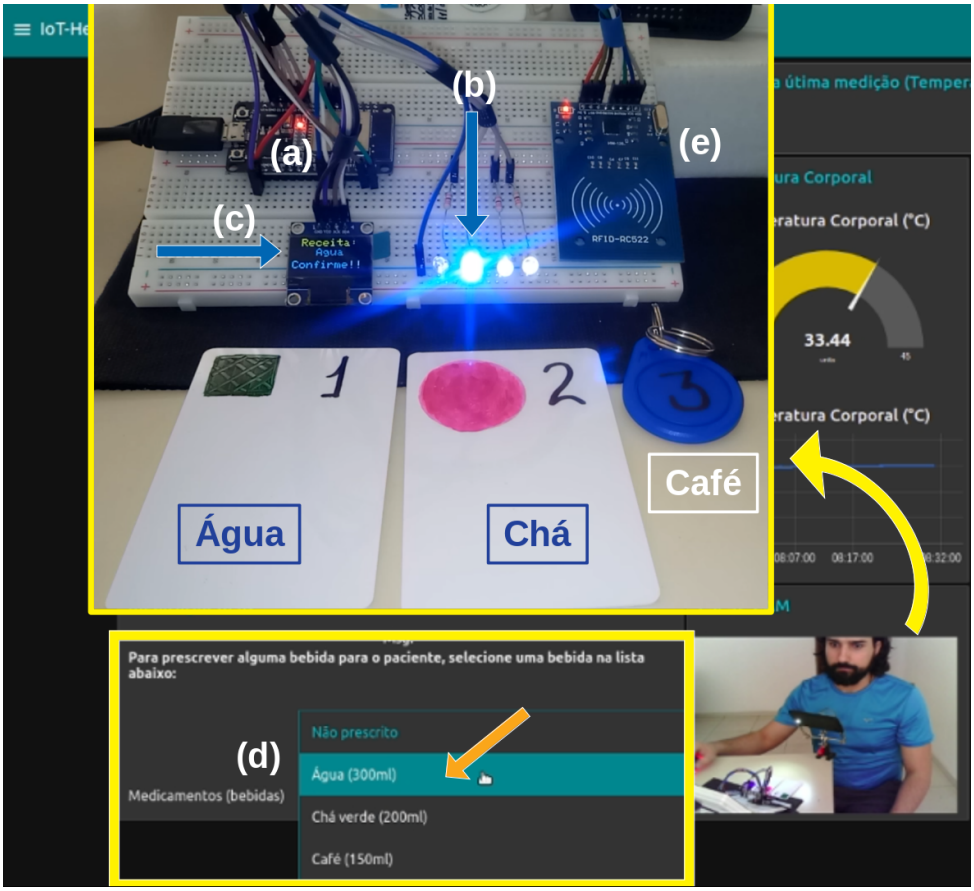

Figura 73 – ESP32 (a); *LED* azul (b), *display* (c); interface gráfica do sistema (*Node-RED*); módulo *RFID* (e).

registrados. Analisando-se os registros, é possível verificar um atraso entre a medição e a gravação de aproximadamente um segundo conforme atestam as duas primeiras colunas da tabela da Figura 77.

No início da sessão do primeiro dia do experimento, detectou-se que havia uma falha no contato entre a fonte de alimentação e o microcontrolador ESP32 conectado aos sensores DS18B20 e DHT22. Isso justifica a falha apresentada no trecho inicial do gráfico da Figura 70, entre os horários de 09h26m25 até 09h31m46s. Assim, o registro de dados do monitoramento no primeiro dia do experimento para a medição das temperaturas do corpo e do ambiente e da umidade ocorreu de forma estável a partir de 09h31min48s.

Uma outra análise foi verificar se o microcontrolador ESP32 conectado aos sensores DS18B20 e DHT22 e o microcontrolador ESP32 conectado ao sensor MAX30100 conseguem realizar a quantidade estipulada de medições necessárias. Devido à falha na alimentação de energia, mencionada anteriormente, consideraram-se válidas para análise, as medições entre os horários de 09h32m00s e 10h01m00s. De acordo com a programação dos referidos microcontroladores, as medições deveriam ter ocorrido a cada dois segundos. Assim, no período considerado de 29 minutos da sessão experimental, eram esperadas 871

	ID Teste	Horário planilhas google	Horário arquitetura Iot	Medicação Prescrita	Medicação Verificada	Situação da Medicação
2	1	11/27/2020 8:42:02	08:42:01	Agua	Cha	Errada
3	2	11/27/2020 8:42:24	08:42:23	Agua	Cafe	Errada
4	3	11/27/2020 8:42:53	08:42:52	Agua	Cha	Errada
5	4	11/27/2020 8:43:06	08:43:05	Agua	Cha	Errada
6	5	11/27/2020 8:43:24	08:43:23	Agua	Cafe	Errada
7	6	11/27/2020 8:43:33	08:43:32	Agua	Cafe	Errada
8	7	11/27/2020 8:43:53	08:43:52	Agua	Cafe	Errada
9	8	11/27/2020 8:44:00	08:43:59	Agua	Cha	Errada
10	9	11/27/2020 8:44:21	08:44:20	Agua	Agua	Correta
11	10	11/27/2020 8:44:58	08:44:57	Cha	Agua	Errada
12	11	11/27/2020 8:45:21	08:45:20	Cha	Cafe	Errada
13	12	11/27/2020 8:45:37	08:45:36	Cha	Agua	Errada
14	13	11/27/2020 8:45:44	08:45:43	Cha	Agua	Errada
15	14	11/27/2020 8:46:03	08:46:02	Cha	Cafe	Errada
16	15	11/27/2020 8:46:15	08:46:14	Cha	Agua	Errada
17	16	11/27/2020 8:46:32	08:46:31	Cha	Cafe	Errada
18	17	11/27/2020 8:46:41	08:46:40	Cha	Cafe	Errada
19	18	11/27/2020 8:46:55	08:46:54	Cha	Cha	Correta
20	19	11/27/2020 8:47:46	08:47:45	Cafe	Agua	Errada
21	20	11/27/2020 8:47:54	08:47:52	Cafe	Cha	Errada
22	21	11/27/2020 8:48:20	08:48:19	Cafe	Agua	Errada
23	22	11/27/2020 8:48:27	08:48:25	Cafe	Agua	Errada
24	23	11/27/2020 8:48:43	08:48:41	Cafe	Cha	Errada
25	24	11/27/2020 8:48:52	08:48:51	Cafe	Agua	Errada
26	25	11/27/2020 8:49:09	08:49:07	Cafe	Cha	Errada
27	26	11/27/2020 8:49:15	08:49:14	Cafe	Cha	Errada
28	27	11/27/2020 8:49:30	08:49:28	Cafe	Cafe	Correta

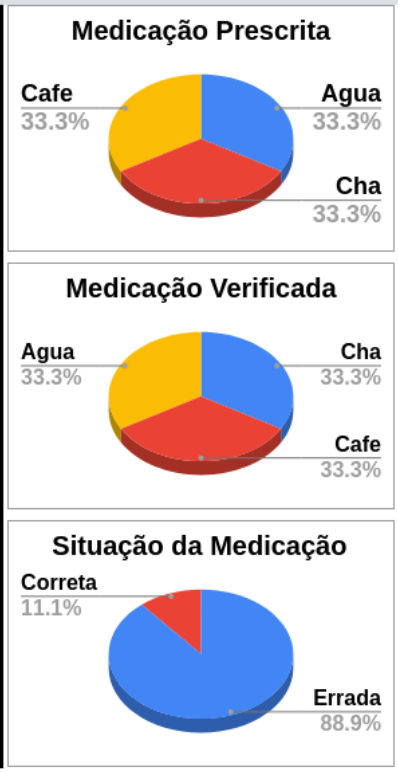

Figura 74 – Tabela com os valores obtidos da simulação da administração de medicamentos.

28	11/27/2020 9:40:31	09:40:29	Agua	Agua	Correta
29	11/28/2020 9:40:32	09:40:30	Cha	Cha	Correta
30	11/29/2020 9:40:11	09:40:10	Cafe	Cafe	Correta

Figura 75 – Tabela com os valores obtidos da simulação da administração de medicamentos durante os três dias.

medições armazenadas nos cartões microSD e nas planilhas do *Google Drive*. Na tabela da Figura 78 verifica-se que somente alguns dados referentes às medições das temperaturas do corpo e do ambiente e à medição da umidade, deixaram de ser armazenados no primeiro e segundo dias do experimento. No primeiro dia, 16 medições em sequência não foram registradas dentro do período de aproximadamente 30 segundos, entre os seguintes horários 09h39m00s e 09h39m30s. No segundo dia, 15 medições não foram registradas dentro do período de aproximadamente 30 segundos, entre os seguintes horários 09h36m20s e 09h36m50s, com a ocorrência de uma medição no horário 09h36m40s. Ainda para contextualizar melhor a análise, o gráfico da Figura 79 destaca através da seta azul (a) a falha no primeiro dia e da seta vermelha (b) a falha no segundo dia.

Dados armazenados nos cartões microSD e nas planilhas (Google drive)				
Dia das medições	Temp. Corporal / Temp. Ambiente/ Umidade		Frequência Cardíaca / Oximetria	
	Micro SD	Planilha Google	Micro SD	Planilha Google
27/11/2020	884	884	931	931
28/11/2020	994	994	952	952
29/11/2020	1129	1129	1160	1160

Figura 76 – Relação de dados coletados e armazenados localmente nos cartões microSD e na nuvem (planilhas do *Google Drive*).

	A	B	C	D	E	F
1			Planilha Google		Micro SD	
2	Horário de registro na planilha	Horário de medição do sensor (cartão microSD)	Frequência Cardíaca	Oximetria	Frequência Cardíaca	Oximetria
933	11/28/2020 10:00:27	10:00:26	92.35	95	92.35	95
934	11/28/2020 10:00:29	10:00:28	76.15	96	76.15	96
935	11/28/2020 10:00:31	10:00:30	69.54	96	69.54	96
936	11/28/2020 10:00:33	10:00:32	71.3	95	71.3	95
937	11/28/2020 10:00:35	10:00:34	82.21	95	82.21	95
938	11/28/2020 10:00:37	10:00:36	102.98	95	102.98	95
939	11/28/2020 10:00:39	10:00:38	90.35	95	90.35	95
940	11/28/2020 10:00:41	10:00:40	75.39	96	75.39	96
941	11/28/2020 10:00:43	10:00:42	78.63	96	78.63	96
942	11/28/2020 10:00:45	10:00:44	88.07	96	88.07	96
943	11/28/2020 10:00:47	10:00:46	78.8	96	78.8	96
944	11/28/2020 10:00:49	10:00:48	71.12	96	71.12	96
945	11/28/2020 10:00:51	10:00:50	70.93	96	70.93	96
946	11/28/2020 10:00:53	10:00:52	73.38	96	73.38	96
947	11/28/2020 10:00:55	10:00:54	75.52	96	75.52	96
948	11/28/2020 10:00:57	10:00:56	79.81	96	79.81	96
949	11/28/2020 10:00:59	10:00:58	82.31	96	82.31	96
950	11/28/2020 10:01:01	10:01:00	72.09	96	72.09	96
951	11/28/2020 10:01:03	10:01:02	69.55	96	69.55	96
952	11/28/2020 10:01:05	10:01:04	69.97	96	69.97	96
953	11/28/2020 10:01:07	10:01:06	73.67	96	73.67	96
954	11/28/2020 10:01:09	10:01:08	74.04	96	74.04	96

Figura 77 – Comparação entre os horários de armazenamento dos dados nos cartões microSD e nas planilhas do *Google Drive*.

Com relação à análise das gravações de dados, verificou-se que não houve nenhuma falha no funcionamento dos microcontroladores. No entanto, o *display* e a parte da interface gráfica do sistema relacionados às temperaturas do corpo e do ambiente e à umidade não foram atualizados até passar os dois segundos programados. Também nesta análise, vale destacar a importância da utilização do horário para registrar os momentos em que as medições foram realizadas. Nesse caso, o sistema apresentou uma pequena falha dentre todas as medições realizadas. Uma provável causa desta falha foi a perda de comunicação do microcontrolador ESP32 com a rede *Wi-Fi*. o que leva ele a tentar se conectar novamente. A análise revelou que foram 31 perdas dentre 2582 medições realizadas, levando a uma taxa de falha de 1,2%. Uma possível solução para prevenir ou diagnosticar a ocorrência deste tipo de falhas é, por exemplo, através do uso do servidor *Zabbix* que possui

recursos que podem atender esta situação.

Quantidade de dados monitorados entre 09h32m00s e 10h01m00s		
Dia das medições	**Temp. Corporal / Temp. Ambiente/ Umidade**	**Frequência Cardíaca / Oximetria**
27/11/2020	855	871
28/11/2020	856	871
29/11/2020	871	871

Figura 78 – Quantidade de dados medidos nos três dias de prática entre o período de 09h32m00s e 10h01m00s.

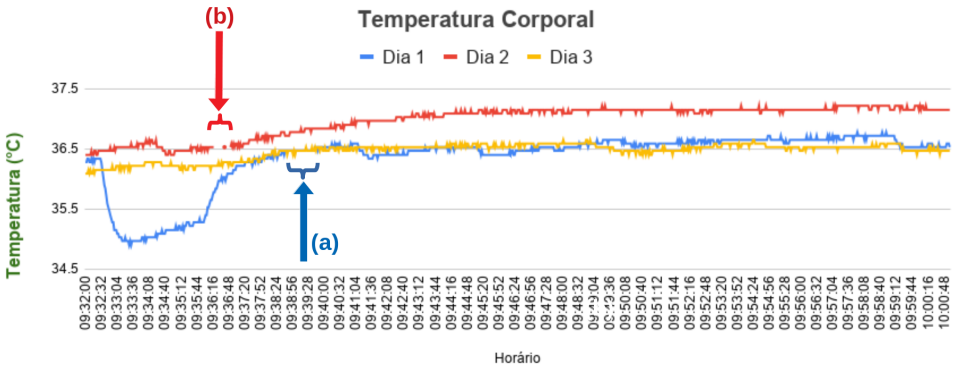

Figura 79 – Falha no registro dos dados (a) dia 1 e (b) dia 2.

Observou-se que as medidas de umidade coletadas pelo sensor DHT22 apresentaram muitas variações durante a execução do experimento conforme se verificam nos gráficos. Segundo (KOESTOER et al., 2019), este sensor é muito sensível e deve ser calibrado de acordo com o ambiente onde ele está sendo usado para evitar erros na medição. No experimento de (KOYANAGI, 2017), constata-se que o sensor é soprado por uma pessoa e logo depois, os valores referentes à umidade sofrem uma variação significativa. A sensibilidade é tão alta que o simples posicionamento do sensor próximo ao corpo de uma pessoa, faz com que ocorra uma interferência considerável nos valores de umidade a serem medidos.

No caso deste experimento, constata-se pela análise das imagens do vídeo que foi gravado durante as sessões experimentais que o sensor se encontra posicionado praticamente junto ao corpo do voluntário e ocorrem variações nos valores da umidade à medida em que este voluntário se movimenta. Essa constatação foi importante para perceber a importância de se calibrar o sensor DHT22 baseando-se no trabalho existente (KOESTOER et al., 2019), assim como tomar um cuidado relevante para posicionar ou instalar este sensor de forma a evitar interferências indesejadas.

Ao final deste capítulo, encontra-se a Figura 80, que apresenta uma tabela detalhada contendo as comparações das medições realizadas pelos sensores e dispositivos usados em

ambientes hospitalares. A tabela destaca os registros de medição ao longo do tempo, indicando o dia, a hora, os minutos e os segundos das aferições. A análise comparativa desses dados foi validada pelas imagens do vídeo gravado durante todo o experimento.

Tipos de Medições	1° Dia 27/11/2020			2° Dia 28/11/2020			3° Dia 29/11/2020		
	Horário das Medições	Dispositivos Externo	Sensor IoT	Horário das Medições	Dispositivos Externo	Sensor IoT	Horário das Medições	Dispositivos Externo	Sensor IoT
Temp. Corporal	**09h34m46s**	36.4	35.15	**09h32m02s**	36.6	36.4	**09h32m06s**	36.6	36.15
Temp. Ambiente in		27.7	27.3		27.8	27.6		27.8	27.5
Temp. Ambiente out		27.6			28.6			28.3	
Umidade		52	55.2		52	55		49	53.3
Freq. Cardíaca	**09h32m56s**	66	65.75	**09h30m44**	77	77.48	**09h30m30s**	76	76.70
Oximetria		97	96		96	95		96	96
Temp. Corporal	**09h38m16s**	36.5	36.4	**09h36m18s**	36.7	36.53	**09h35m28s**	36.6	36.22
Temp. Ambiente in		27.7	27.6		27.8	27.9		27.8	27.7
Temp. Ambiente out		27.6			29.1			28.1	
Umidade		52	54.8		52	53.9		49	52.9
Freq. Cardíaca	**09h37m04s**	67	67.04	**09h35m12s**	77	77.09	**09h35m10s**	70	70.04
Oximetria		97	96		96	95		96	96
Temp. Corporal	**09h43m18s**	36.5	36.47	**09h44m38s**	37	37.09	**09h44m28s**	36.9	36.53
Temp. Ambiente in		27.8	27.7		28	28.20		28	28.1
Temp. Ambiente out		27.6			29.4			27.9	
Umidade		52	54.6		51	53.9		50	53
Freq. Cardíaca	**09h41m46s**	72	72.49	**09h42m54s**	87	88.57	**09h42m36s**	81	81.94
Oximetria		97	96		97	95		97	95
Temp. Corporal	**09h47m08**	36.5	36.53	**09h47m24s**	37	37.09	**09h47m**	36.9	36.59
Temp. Ambiente in		27.8	27.9		28.1	28.30		28.0	28.2
Temp. Ambiente out		27.6			29.6			27.9	
Umidade		52	54.6		51	54.3		49	52.7
Freq. Cardíaca	**09h46m12s**	66	66.55	**09h46m18**	80	80.18	**09h46m14**	77	77.27
Oximetria		97	95		97	95		96	96
Temp. Corporal	**09h53m22s**	36.6	36.5	**09h51m42s**	37	37.15	**09h50m26s**	36.7	36.47
Temp. Ambiente in		27.8	28		28.1	28.4		28.1	28.4
Temp. Ambiente out		27.6			29.6			27.9	
Umidade		52	53.7		51	53.4		49	52.2
Freq. Cardíaca	**09h50m26s**	67	67.56	**09h50m42s**	77	76.91	**09h49m48s**	77	77.50
Oximetria		96	96		96	95		96	95
Temp. Corporal	**09h56m28s**	36.7	36.65	**09h56m56s**	37.1	37.15	**09h55m38s**	36.9	36.53
Temp. Ambiente in		27.8	28		28.2	28.5		28.2	28.4
Temp. Ambiente out		27.6			29.9			28.1	
Umidade		52	54.2		51	53.3		50	52.5
Freq. Cardíaca	**09h55m20s**	66	66.33	**09h55m30s**	71	71.08	**09h59m44s**	75	75.57
Oximetria		96	96		97	95		96	96

Figura 80 – Comparação entre medições dos sensores *IoT* e dispositivos de referência

Capítulo 6

Conclusão

Os resultados obtidos com a simulação do sistema de baixo custo desenvolvido neste trabalho, levam a concluir que o monitoramento remoto do quadro clínico de pacientes e o controle da precisão na administração de medicamentos aos pacientes são plenamente possíveis usando os recursos e conceitos da computação de borda.

A simulação do funcionamento do sistema proporcionou resultados de medição com valores precisos e o armazenamento dos dados de forma eficiente. Observou-se que nenhum dado medido deixou de ser armazenado localmente na memória e à distância na planilha do *Google Drive*. O que reforça a vantagem do sistema em armazenar os dados e também disponibilizá-los para serem acessados pela Internet para consultas do histórico registrado.

Com a integração entre os sensores e os *displays*, o módulo *RFID*, os *LED* (amarelo, azul, vermelho e verde) e, principalmente, o módulo de cartões microSD, foi possível comparar os resultados e o tempo de resposta do momento que os sensores fizeram a medição até o momento em que os dados foram registrados na planilha. E o sistema desenvolvido no *Node-RED* se mostrou eficaz para satisfazer as hipóteses levantadas.

Todas as funcionalidades do sistema foram projetadas e se tornaram viáveis devido à versatilidade de recursos que o microcontrolador ESP32 proporciona. O seu poder de processamento, a sua conectividade por rede *Wi-Fi* e a sua capacidade de dispor diversos pinos *GPIO* permitiram que os dados medidos pelos sensores fossem disponibilizados nos *displays* e na interface do sistema (*Node-RED*) e também enviados para serem armazenados nas planilhas do *Google Drive*, podendo também ser acompanhados em tempo real.

A constatação de que é possível monitorar o estado clínico de um paciente através da medição em tempo real dos seus dados vitais e a visualização dos parâmetros relacionados a estes dados, mesmo que o paciente e o profissional da saúde estejam em quaisquer pontos diferentes do planeta conectados à Internet, traz inúmeras possibilidades de aplicação do sistema desenvolvido na área da saúde.

6.1 Principais Contribuições

As duas arquiteturas desenvolvidas neste trabalho incentivam a criação e desenvolvimento de novas tecnologias usando recursos da *IoT* aplicados a computação de borda para atuar na área da saúde. Os sólidos resultados obtidos com as duas arquiteturas comprovam que é possível contribuir com a área da saúde com dispositivos de baixo custo. A eficiência apresentada pelos microcontroladores ESP32 de gerenciar, armazenar, receber e enviar dados através da rede de computadores demonstram que o aprimoramento destas arquiteturas é promissor dentro da área da saúde, principalmente pelo período crítico vivenciado devido a pandemia mundial da Covid-19.

A qualidade do desenvolvimento da arquitetura I deste trabalho motivou a decisão de protocolá-la como um programa de computador junto ao Instituto Nacional da Propriedade Industrial (INPI) com o número de registro BR 51 202002656-7. Já a arquitetura II será aprimorada para que seja patenteada.

Todas as informações relacionadas com a arquitetura I deste trabalho foi submetida no formato de um artigo com o título "An affordable remote patient monitoring system based on commodity IoT devices", na renomada conferência *CLOSER (International Conference on Cloud Computing and Services Science)*

6.2 Trabalhos Futuros

Pretende-se, como trabalhos futuros:

- ❏ Projetar e criar uma placa de circuito impressa com todos os componentes da segunda arquitetura;
- ❏ Desenvolver recursos de segurança da informação e aplicá-los no sistema desenvolvido neste trabalho;
- ❏ Desenvolver recursos de inteligência computacional para o microcontrolador ESP32, para que ele reconheça e realize alertas sobre variações suspeitas dos dados medidos pelos sensores, para ajudar a realizar diagnósticos precoces e mais eficientes.

Referências

AI-THINKER. **ESP32-CAM Module**. Ai-Thinker. Disponível em: <https://loboris.eu/ESP32/ESP32-CAM\%20Product\%20Specification.pdf>. Acesso em: 10/04/2020.

AIEA, V. M. **Interfacing Catalex Micro SD Card Module with Arduino**. MY PROJECT WEBSITE. Disponível em: <https://www.vishnumaiea.in/projects/hardware/interfacing-catalex-micro-sd-card-module>. Acesso em: 05/10/2020.

ASHTON, K. et al. That 'internet of things' thing. **RFID journal**, Hauppauge, New York, v. 22, n. 7, p. 97–114, 2009.

BLYNK. **Blynk documentation**. Blynk. Disponível em: <http://docs.blynk.cc/>. Acesso em: 04/02/2020.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: **Proceedings of the first edition of the MCC workshop on Mobile cloud computing**. [S.l.: s.n.], 2012. p. 13–16. https://doi.org/10.1145/2342509.2342513.

CEEN. **Os 6 maiores desafios que a saúde pública do Brasil vem enfrentando**. CEEN. Disponível em: <https://www.ceen.com.br/os-6-maiores-desafios-que-a-saude-publica-do-brasil-vem-enfrentando/>. Acesso em: 20/08/2020.

CIRCUIT BASIC. **BASICS OF THE I2C COMMUNICATION PROTOCOL**. Circuit Basic. Disponível em: <http://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>. Acesso em: 02/09/2020.

CIRCUITO, C. **Tutorial: conhecendo o ESP32, publisher=Blog Curto Circuito**. Disponível em: <https://www.curtocircuito.com.br/blog/conhecendo-esp32/>. Acesso em: 05/02/2020.

CONSELHO FEDERAL DE FARMÁCIA. Anamnese farmacêutica e verificação de parâmetros clínicos. In: **Guia de prática clínica: sinais e sintomas não específicos: febre.** [S.l.: s.n.], 2018. v. 3, p. 34–91.

COPPEN, Richard. **OASIS Message Queuing Telemetry Transport (MQTT) TC**. OASIS Open standards. Open source. Disponível em: <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt>. Acesso em: 03/02/2020.

COSTANZO, S.; FLORES, A. A non-contact integrated body-ambient temperature sensors platform to contrast covid-19. **Electronics**, Multidisciplinary Digital Publishing Institute, v. 9, n. 10, p. 1658, 2020. https://doi.org/10.3390/electronics9101658.

COUTO, R. C. et al. II Anuário da segurança assistencial hospitalar no brasil. In: **Instituto de Estudos de Saúde Suplementar**. [S.l.: s.n.], 2018.

DOLUI, K.; DATTA, S. K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: IEEE. **2017 Global Internet of Things Summit (GIoTS)**. [S.l.], 2017. p. 1–6. https://doi.org/10.1109/GIOTS.2017.8016213.

ESTADÃO CONTEÚDO. **Hospitais brasileiros testam inteligência artificial contra sepse**. Estadão Conteúdo. Disponível em: <https://exame.abril.com.br/ciencia/hospitais-brasileiros-testam-inteligencia-artificial-contra-sepse/>. Acesso em: 04/02/2020.

FARAHANI, B. et al. Towards fog-driven iot ehealth: Promises and challenges of iot in medicine and healthcare. **Future Generation Computer Systems**, Elsevier, v. 78, p. 659–676, 2018. https://doi.org/10.1016/j.future.2017.04.036.

FRAMINGHAM, M. **The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast**. International Data Corporation (IDC). Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS45213219>. Acesso em: 02/02/2020.

GAYTHORPE, K. et al. Report 8: Symptom progression of covid-19. **Imperial College London**, March 2020. https://doi.org/10.25561/77344.

GOOGLE DRIVE. **Compartilhar arquivos do Google Drive**. Google Drive. Disponível em: <https://support.google.com/docs/answer/2494822?co=GENIE.Platform\%3DDesktop&hl=pt-BR>. Acesso em: 06/09/2020.

_____. **Explore the Storage Features of Drive**. Google Drive. Disponível em: <https://www.google.com/intl/en-GB_ALL/drive/using-drive/>. Acesso em: 10/09/2020.

GOVERNO DO BRASIL. **Novos hospitais participam de projeto para reduzir a lotação nas emergências**. Governo do Brasil. Disponível em: <https://www.gov.br/pt-br/noticias/saude-e-vigilancia-sanitaria/2020/01/novos-hospitais-participam-de-projeto-para-reduzir-a-lotacao-nas-emergencias>. Acesso em: 02/02/2020.

HALFACREE, G. **The Official Raspberry Pi Beginner's Guide: How to Use Your New Computer**. [S.l.]: Raspberry Pi PRESS, 2018.

HIVEMQ TEAM. **Publish & Subscribe - MQTT Essentials: Part 2**. Disponível em: <https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/>. Acesso em: 28/08/2020.

HOSSAIN, M. S.; MUHAMMAD, G. Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring. **Computer Networks**, Elsevier, v. 101, p. 192–202, 2016. https://doi.org/10.1016/j.comnet.2016.01.009.

IDC. **IDC Brasil prevê crescimento de 4,9% no mercado de TIC em 2019**. International Data Corporation Brasil. Disponível em: <http://br.idclatin.com/releases/news.aspx?id=2462>. Acesso em: 04/02/2020.

KALIL, A. J. **Avaliação do impacto na identificação de pacientes com risco de sepse após implantação de um robô cognitivo gerenciador de riso (Robô Laura)**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2017.

KODALI, R. K.; RAJANARAYANAN, S. C. Iot based indoor air quality monitoring system. In: IEEE. **2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)**. [S.l.], 2019. p. 1–5. https://doi.org/10.1109/WiSPNET45539.2019.9032855.

KOESTOER, R. et al. A simple calibration methods of relative humidity sensor dht22 for tropical climates based on arduino data acquisition system. In: AIP PUBLISHING LLC. **AIP Conference Proceedings**. [S.l.], 2019. v. 2062, n. 1, p. 020009. https://doi.org/10.1063/1.5086556.

KOYANAGI, F. **ESP32 com RFID: Controle de Acesso**. Fernando K Tecnologias. Disponível em: <https://www.fernandok.com/2018/02/esp32-com-rfid-controle-de-acesso.html>. Acesso em: 05/02/2020.

_____. **Gráfico de temperatura e umidade com display OLED**. Fernando K Tecnologias, 2017. Disponível em: <https://www.fernandok.com/2017/11/grafico-de-temperatura-e-umidade-com.html>. Acesso em: 05/10/2020.

LASKOWSKI, L. Google forms and sheets for library gate counts. **Journal of Access Services**, Taylor & Francis, v. 13, n. 3, p. 151–158, 2016. https://doi.org/10.1080/15367967.2016.1184577.

LIU, T. Digital-output relative humidity & temperature sensor/module dht22 (dht22 also named as am2302). **Aosong Electronics Co.,Ltd**, 2013.

MACGILLIVRAY, C.; REINSEL, D. **41.6 billion IoT devices will be generating 79.4 zettabytes of data in 2025**. Help Net Security. Disponível em: <https://www.helpnetsecurity.com/2019/06/21/connected-iot-devices-forecast/>. Acesso em: 15 jan 2020.

_____. **Worldwide Global DataSphere IoT Device and Data Forecast, 2019–2023**. International Data Corporation (IDC). Disponível em: <https://www.idc.com/getdoc.jsp?containerId=US45066919>. Acesso em: 15 jan 2020.

MADAKAM, S. et al. Internet of things (iot): A literature review. **Journal of Computer and Communications**, Scientific Research Publishing, v. 3, n. 05, p. 164, 2015. http://dx.doi.org/10.4236/jcc.2015.35021.

MASON, J. W. et al. Electrocardiographic reference ranges derived from 79,743 ambulatory subjects. **Journal of electrocardiology**, Elsevier, v. 40, n. 3, p. 228–234, 2007. https://doi.org/10.1016/j.jelectrocard.2006.09.003.

MAXIM INTEGRATED PRODUCTS. **Pulse Oximeter and Heart-Rate Sensor IC for Wearable Health**. Maxim Integrated Products. Disponível em: <https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf>. Acesso em: 02/02/2020.

MELEXIS INSPIRED ENGINEERING. **MLX90614 family, single and dual zone infrared thermometer in TO-39**. Melexis Inspired Engineering. Disponível em: <https://www.melexis.com/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf>. Acesso em: 03/02/2020.

MICROSOFT AZURE. **Publisher-Subscriber pattern**. Microsoft Azure. Disponível em: <https://docs.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber>. Acesso em: 05/02/2020.

MINISTÉRIO DA SAÚDE. **Painel Coronavírus**. Ministério da Saúde Brasileiro. Disponível em: <https://covid.saude.gov.br/>. Acesso em: 20/10/2020.

_____. **Projeto Lean nas Emergências: redução das superlotações hospitalares**. Ministério da Saúde. Disponível em: <https://www.saude.gov.br/saude-de-a-z/projeto-lean-nas-emergencias>. Acesso em: 02/02/2020.

MOSQUITTO. **Eclipse Mosquitto An open source MQTT broker**. Mosquitto. Disponível em: <https://mosquitto.org/>. Acesso em: 05/02/2020.

NODE-RED. **About**. Node-RED. Disponível em: <https://nodered.org/about/>. Acesso em: 04/09/2020.

_____. **node-red-dashboard 2.25.0**. Node-RED. Disponível em: <https://flows.nodered.org/node/node-red-dashboard>. Acesso em: 12/09/2020.

_____. **User Guide**. Node-RED. Disponível em: <https://nodered.org/docs/user-guide/>. Acesso em: 10/09/2020.

NXP SEMICONDUCTORS. **MFRC522 Standard performance MIFARE and NTAG frontend**. NXP Semiconductors N.V. Disponível em: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>. Acesso em: 05/09/2020.

OASIS. **MQTT Version 5.0**. OASIS. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>. Acesso em: 03/02/2020.

OLIVEIRA, J. **Arduino, ESP32 e ESP8266 – Comparação**. xProjetos.net. Disponível em: <http://xprojetos.net/arduino-esp32-e-esp8266-comparacao/>. Acesso em: 02/02/2020.

PEKO'S LIBRARY. **RCWL-0530 MAX30100 an oximetry heart rate temperature sensor**. Peko's Library. Disponível em: <https://pekolibrary.wordpress.com/2018/05/27/rcwl-0530-max30100/>. Acesso em: 04/02/2020.

RAHMANI, A. M. et al. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. **Future Generation Computer Systems**, Elsevier, v. 78, p. 641–658, 2018. https://doi.org/10.1016/j.future.2017.02.014.

RASPBERRY. **Raspberry Pi 4**. Raspberry Pi Foundation. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-4-model-b/>. Acesso em: 05/02/2020.

RESOLUTION, D. P. Programmable resolution 1-wire digital thermometer. **Maxim integrated, USA**, 2019.

RIBEIRO, M. M. M. S. **Wearable sensor for continuous monitoring of physiological parameters**. Dissertação (Mestrado) — Universidade do Porto, 2020. https://doi.org/10.1038/s41746-019-0150-9.

ROZAQ, I. A.; DS, N. Y. Uji karakterisasi sensor suhu ds18b20 waterproof berbasis arduino uno sebagai salah satu parameter kualitas air. **Prosiding SNATIF**, p. 303–309, 2017.

SAFAVI, K.; KALIS, B.; THOMPSON, A. The post digital era is upon us digital health tech vision. In: **Accenture**. [S.l.: s.n.], 2019.

SAHA, H. N.; MANDAL, A.; SINHA, A. Recent trends in the internet of things. In: IEEE. **2017 IEEE 7th annual computing and communication workshop and conference (CCWC)**. [S.l.], 2017. p. 1–4. https://doi.org/10.1109/CCWC.2017.7868439.

SALERNO, J. et al. Ethics, big data and computing in epidemiology and public health. **Annals of epidemiology**, Elsevier, v. 27, n. 5, p. 297–301, 2017. https://doi.org/10.1016/j.annepidem.2017.05.002.

SALGADO, P. d. O. et al. Cuidados de enfermagem a pacientes com temperatura corporal elevada: revisão integrativa. **Revista Mineira de Enfermagem**, v. 19, n. 1, p. 212–226, 2015. https://doi.org/10.5935/1415-2762.20150017.

SANTOS, R.; SANTOS, S. **ESP32-CAM Video Streaming and Face Recognition with Arduino IDE**. Random nerd tutorials. Disponível em: <https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/>. Acesso em: 02/04/2020.

_____. **ESP32 Data Logging Temperature to MicroSD Card**. Disponível em: <https://randomnerdtutorials.com/esp32-data-logging-temperature-to-microsd-card/>. Acesso em: 04/10/2020.

_____. **ESP32 I2C Communication: Set Pins, Multiple Bus Interfaces and Peripherals (Arduino IDE)**. Random nerd tutorials. Disponível em: <https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/>. Acesso em: 02/02/2020.

_____. **ESP32 OLED Display with Arduino IDE**. Random nerd tutorials. Disponível em: <https://randomnerdtutorials.com/esp32-ssd1306-oled-display-arduino-ide/>. Acesso em: 18/06/2020.

_____. **ESP32 with DHT11/DHT22 Temperature and Humidity Sensor using Arduino IDE**. Ai-Thinker. Disponível em: <https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>. Acesso em: 10/09/2020.

SHI, W. et al. Edge computing: Vision and challenges. **IEEE internet of things journal**, IEEE, v. 3, n. 5, p. 637–646, 2016. https://doi.org/10.1109/JIOT.2016.2579198.

SHI, W.; DUSTDAR, S. The promise of edge computing. **Computer**, IEEE, v. 49, n. 5, p. 78–81, 2016. https://doi.org/10.1109/MC.2016.145.

SYSTECH, S. **SSD1306 Datasheet**. [S.l.]: Rivers, 2005.

SYSTEMS, E. **ESP32 Technical Reference Manual**. Espressif Systems. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf>. Acesso em: 04/02/2020.

TARYUDI PRASETYO, I.; NUGRAHA, A.; AMMAR, R. Health Care Monitoring System Based-on Internet of Things. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2019. v. 1413, n. 1, p. 012008. http://dx.doi.org/10.1088/1742-6596/1413/1/012008.

TECNODOMOS. **GY-906 - Sensor de Temperatura por Infravermelho**. TecnoDomos. Disponível em: <http://tecnodomos.blogspot.com/2018/06/gy-906-sensor-de-temperatura-por.html>. Acesso em: 04/02/2020.

UPTON, E.; HALFACREE, G. **Raspberry Pi user guide**. [S.l.]: John Wiley & Sons, 2014.

WORLD HEALTH ORGANIZATION. **10 facts on patient safety**. World Health Organization. Disponível em: <https://www.who.int/features/factfiles/patient_safety/en/>. Acesso em: 03/02/2020.

_____. **Coronavirus disease (COVID-19) pandemic**. World Health Organization. Disponível em: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019>. Acesso em: 17/05/2020.

_____. **Pulse oximetry training manual.** World Health Organization. Disponível em: <https://cdn.who.int/media/docs/default-source/patient-safety/pulse-oximetry/who-ps-pulse-oxymetry-training-manual-en.pdf?sfvrsn=322cb7ae_6>. Acesso em: 17/05/2020.

WORLDOMETERS. **Reported Cases and Deaths by Country, Territory, or Conveyance**. Worldometers. Disponível em: <https://www.worldometers.info/coronavirus/>. Acesso em: 23/10/2020.

XIE, C.; YANG, P.; YANG, Y. Open knowledge accessing method in iot-based hospital information system for medical record enrichment. **IEEE Access**, IEEE, v. 6, p. 15202–15211, 2018. https://doi.org/10.1109/ACCESS.2018.2810837.

XU, L. D.; HE, W.; LI, S. Internet of things in industries: A survey. **IEEE Transactions on industrial informatics**, IEEE, v. 10, n. 4, p. 2233–2243, 2014. https://doi.org/10.1109/TII.2014.2300753.

YUAN, M. **Getting to know MQTT**. Disponível em: <https://developer.ibm.com/technologies/messaging/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 05/09/2020.

YUSOF, M. A.; HAU, Y. W. Mini home-based vital sign monitor with Android mobile application (myVitalGear). In: IEEE. **2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)**. [S.l.], 2018. p. 150–155. https://doi.org/10.1109/IECBES.2018.8626639.

ZANDI, D. et al. New ethical challenges of digital technologies, machine learning and artificial intelligence in public health: a call for papers. **Bulletin of the World Health Organization**, World Health Organization, v. 97, n. 1, p. 2–2, 2019. http://dx.doi.org/10.2471/BLT.18.227686.