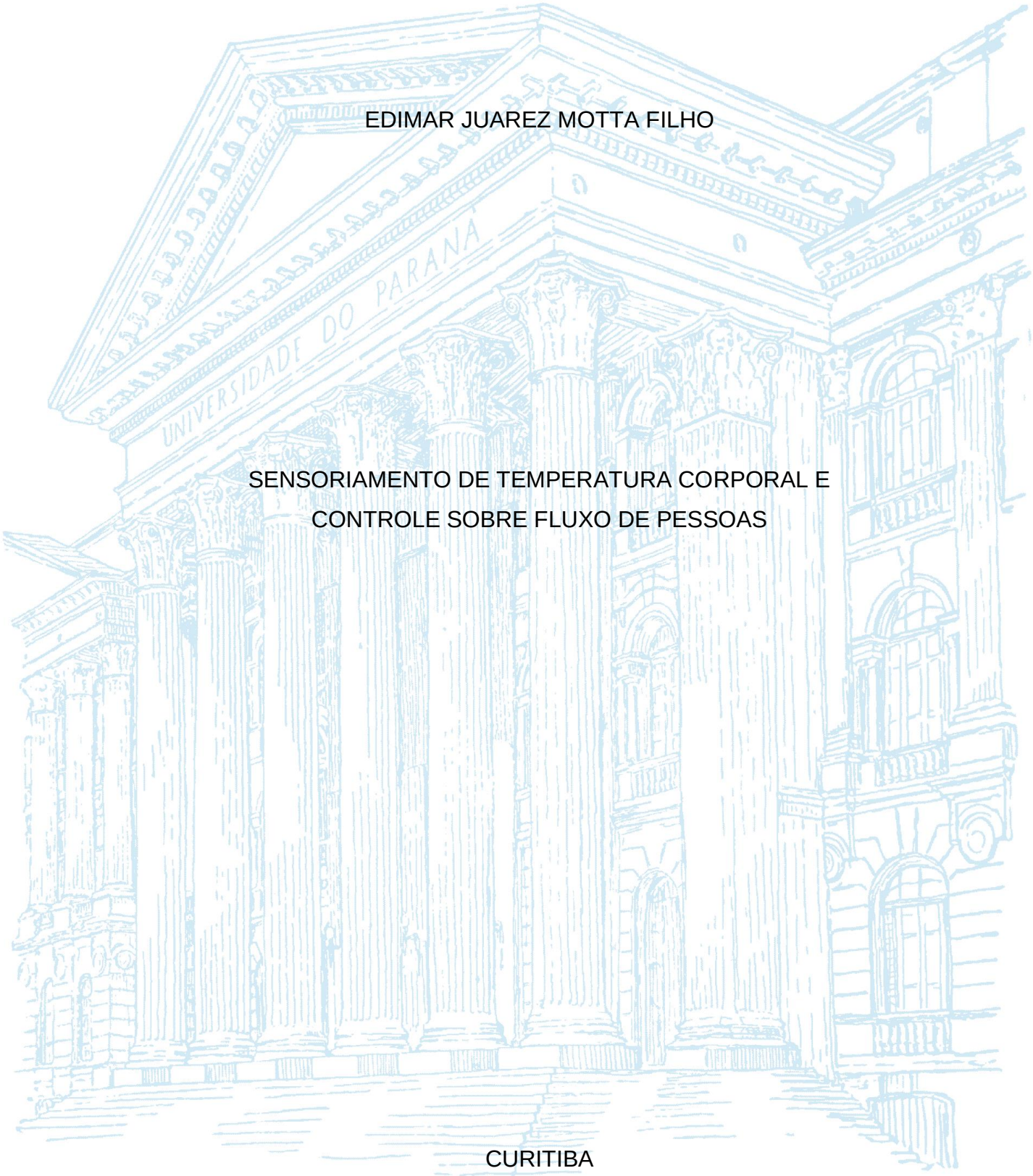
UNIVERSIDADE FEDERAL DO PARANÁ
ENGENHARIA ELÉTRICA



EDIMAR JUAREZ MOTTA FILHO

SENSORIAMENTO DE TEMPERATURA CORPORAL E CONTROLE SOBRE FLUXO DE PESSOAS

CURITIBA
2021

EDIMAR JUAREZ MOTTA FILHO

SENSORIAMENTO DE TEMPERATURA CORPORAL E CONTROLE SOBRE FLUXO DE PESSOAS

Trabalho de Conclusão de Curso de Engenharia Elétrica, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná.

Orientador: Prof. Ph.D. André Augusto Mariano

CURITIBA

2021

EDIMAR JUAREZ MOTTA FILHO

SENSORIAMENTO DE TEMPERATURA CORPORAL E CONTROLE SOBRE FLUXO DE PESSOAS

TRABALHO APRESENTADO AO CURSO DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARANÁ, COMO REQUISITO À OBTENÇÃO DO TÍTULO DE GRADUAÇÃO.

COMISSÃO EXAMINADORA

_____

Prof. PH.D ANDRÉ AUGUSTO MARIANO

_____

Prof. Dr. GIDEON VILLAR LEANDRO

_____

Prof. Dr. MARCOS VINICIO HAAS RAMBO

CURITIBA, FEVEREIRO DE 2021

AGRADECIMENTOS

Gostaria de agradecer aos meus familiares e amigos por estarem sempre dispostos e presentes para me auxiliar nos momentos de dificuldade.

A todos os professores do DELT pelos ensinamentos ao longo de todos os anos da graduação.

Ao meu professor orientador pela disposição, instrução e tempo aplicados no trabalho.

RESUMO

A pandemia global de Covid-19 fez com que todas as pessoas mudassem hábitos e rotinas. A preocupação com a contaminação trouxe o isolamento social e a restrição de circulação de pessoas nos ambientes. Nesse trabalho é desenvolvido uma solução para a área de gestão de estabelecimentos e ambientes que inerentemente circulam pessoas. O projeto tem como objetivo a elaboração de um sistema com protótipo que mede a temperatura do usuário e automatiza a higienização das mãos com álcool em gel, bem como a implementação de um sistema de telemetria para o gestor local visualizar os dados obtidos. O projeto também visa alcançar um baixo custo para que se torne um produto viável no mercado. Os resultados mostram que as soluções propostas no trabalho foram satisfatórias e pode-se concluir que o projeto possui potencial para ajudar a combater a pandemia global do coronavírus.

Palavras-chave: temperatura, sensoriamento, microcontrolador, internet.

ABSTRACT

The global pandemic of Covid-19 caused all people to change habits and routines. The concern with contamination brought social isolation and restriction of movement in environments. In this work, a solution is developed for the area of management of establishments and environments that inherently circulate people. The project aims to develop a system and a prototype that measures a user's temperature and automates hand hygiene with alcohol gel, as well as the implementation of a telemetry system for the local manager to view data. The project also aims to achieve a low cost so that it may become a viable product on the market. The results show that the solutions proposed in the work were satisfactory and it is concluded that the project has the potential to help combat the global coronavirus pandemic.

Keywords: temperature, sensing, microcontroller, internet.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

µC – microcontrolador
AC – Alternating Current
ASTM – American Society for Testing and Materials
BJT – Bipolar Junction Transistor
CSS – Cascading Style Sheet
DC – Direct Current
EEPROM – Electrically-Erasable Programmable Read-Only Memory
HTML – Hypertext Markup Language
I2C – Inter-Integrated Circuit
IDE – Integrated Development Environment
JSON – JavaScript Object Notation
LCD – Liquid Crystal *Display*
LED – Light-Emitting Diode
MDF – *Medium-density fiberboard*
MOSFET – Metal-Oxide-Semiconductor Field Effect Transistor)
MW – Microwave
OMS – Organização Mundial da Saúde
PCI – Placa de Circuito Impresso)
PIR – Passive infrared
SCL – Serial Clock
SDA – Serial Data
SPI – Serial Peripheral Interface
SQL – Structured Query Language

SUMÁRIO

1 INTRODUÇÃO

Desde os tempos antigos, na época de Platão, Aristóteles e Hipócrates (século V a.C.), já se entendia a relação entre temperatura corporal e vida. Hipócrates notou que o corpo humano possuía variações de temperatura nas diferentes partes do corpo, logo, concluiu que se uma região está mais quente que o restante, pode estar presente uma doença nessa área.

Nesse período, para mensurar a temperatura, os médicos gregos se dispunham apenas de suas mãos, mas foi quando Hipócrates criou a primeira técnica para diferenciar temperaturas no corpo humano. Ele esfregava lama em seus pacientes e observava em quais regiões ela secava primeiro (INFRAREDMED, 2017). Esse feito deu origem a termografia, que atualmente significa “a técnica de registro gráfico das temperaturas de diversos pontos do corpo por detecção da radiação infravermelha por ele emitida” de acordo com o Dicionário Online de Português (TERMOGRAFIA, 2020). Entretanto, os primeiros sensores de temperatura surgiram no Renascimento.

Em 1592, Galileu Galilei construiu um dispositivo que usava ar em um vaso que alterava o nível de água de uma coluna, cuja altura indicava a variação do resfriamento. Mais tarde, em 1612, na Itália, Santorio Santorio inventou o primeiro termômetro. O aparelho era constituído por um tubo que continha um líquido que se expandia conforme o aumento de temperatura, porém, não haviam unidades exatas para a medição (OMEGA, 2015). O primeiro a usar mercúrio, elemento cuja a dilatação térmica é bastante linear, e padronizar uma escala que é usada até hoje, foi o físico alemão Daniel Gabriel Fahrenheit, em 1714 (FRAZÃO, 2019).

Com a evolução da tecnologia, diversos tipos de termômetros foram surgindo, entre eles está o termômetro infravermelho, que possui a vantagem de conseguir aferir a temperatura sem o contato com a superfície a ser medida. (INTRUSUL, 2018). Graças a esse benefício, ele se torna o melhor candidato para ser usado na pandemia de Covid-19 nos anos de 2019 e 2020.

1.1 PROBLEMA E MOTIVAÇÃO

Durante a época de pandemia global do novo Coronavírus (Sars-Cov-2), que teve início em 11 de março de 2020 pela OMS e se mantém até hoje, o controle sobre

o fluxo de pessoas em ambientes comuns vem se tornando cada vez mais importante para que se possa evitar o contágio do vírus, impedindo aglomerações desnecessárias e incentivando o distanciamento social (BBC NEWS, 2020).

Além disso, a medição de temperatura corporal das pessoas que transitam determinado lugar, além de servir como uma “triagem simples”, visto que é um dos sintomas do vírus, (HOSPITAL OSWALDO CRUZ, 2020), é também um importante indicador de como está a disseminação da doença, uma vez que a febre “é um fator comum à maioria dos pacientes hospitalizados com COVID-19. O grau de elevação da temperatura pode refletir a gravidade da inflamação” (THARAKAN, S., NOMOTO, K., MIYASHITA, S e ISHIKAWA, K. 2020).

Este projeto visa aliar essas ideias com o uso de álcool em gel e criar um *hardware* de baixo custo junto a um sistema de telemetria que possam auxiliar no combate ao novo coronavírus. Desse modo, ao analisar esse segmento no mercado, são encontradas soluções comuns e baratas, tais como as pistolas de medição de temperatura e os totens de álcool em gel com pedal; e soluções completas e caras, com sistema de telemetria e *software* próprio. A motivação do trabalho é criar uma solução completa e acessível para este nicho de mercado.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Este trabalho possui como objetivo desenvolver um sistema distribuidor de álcool em gel que realiza medições de temperatura das pessoas que transitam em determinado ambiente, bem como disponibilizar tais valores para o gestor local.

1.2.2 Objetivos específicos

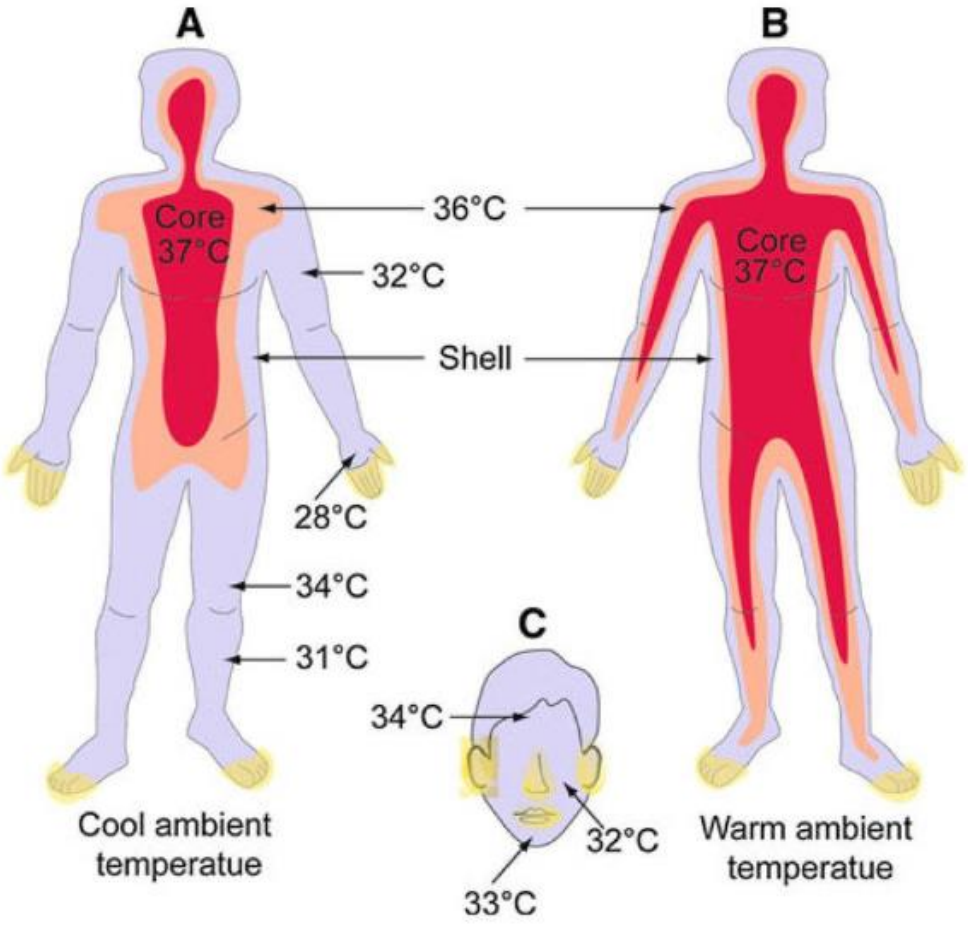
a) Automatizar a distribuição de álcool em gel;
b) Coletar informações de temperatura sem contato;
c) Implementar *LCD* de 32 (16x2) caracteres para interface com usuário;
d) Implementar sensor de presença;
e) Armazenar os valores e quantidades de temperaturas mensuradas;
f) Conectar sistema com a internet;
g) Criar uma página *web* para visualização dos dados.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 TEMPERATURA CORPORAL

A temperatura corporal habitual que aferimos e trabalhamos é a temperatura média de todas as regiões do corpo, que geralmente está na faixa de 36,5 a 37,5 ºC, entretanto, a temperatura que os sensores térmicos detectam é a temperatura superficial da pele humana, que geralmente é mais baixa e que sofre maior influência do meio ambiente. (SERVERSCHECK, 2021)

Figura 1: Diferenças nas temperaturas do corpo humano



Fonte: ServersCheck, 2021.

Apesar dessas diferenças, um instrumento calibrado corretamente pode ser capaz de avaliar a presença de febre nos indivíduos de forma confiável através das medições pelo pulso ou pela testa (CHEN, G., XIE, J., DAI, G., ZHENG, P., HU, X., LU, H., XU, L., CHEN, Xuequin., CHEN, Xiaomin, 2020).

2.2 DISTRIBUIDOR DE ÁLCOOL EM GEL

Para dispensar o álcool em gel de forma automatizada, foi necessário procurar uma bomba capaz de transportar o fluido de forma constante. Em um primeiro momento, para fim de testes, foi usado água ao invés de gel. Também foi escolhida uma bomba de água simples, modelo de nome JT100 (ELETRODEX). Entretanto, ao refinar o protótipo, percebeu-se que não era possível utilizar a mesma bomba para álcool em gel, pois a viscosidade do fluido não permitia o correto funcionamento da bomba. Dessa forma, a escolha foi migrada para uma bomba peristáltica, capaz de transportar líquidos de maior viscosidade.

Figura 2: Bomba peristáltica.

Fonte: RoboCore, 2021.

O princípio de funcionamento desse tipo de bomba é bastante simples e se assemelha ao sistema digestivo dos seres humanos. No corpo humano, o alimento é deslocado ao longo do tubo digestivo pela ação dos músculos, através de movimentos consecutivos de contração e relaxamento. Na bomba peristáltica, a mangueira fica fixa no cabeçote e é pressionada por roletes que realizam movimentos circulares. Esses movimentos criam um vácuo que “puxa” o fluido. Após a passagem do rolete a mangueira retoma seu diâmetro original (SPLABOR, 2017).

Figura 3: Funcionamento de uma bomba peristáltica.

Fonte: Adaptado de SPLabor, 2017.

2.3 SENSOR DE TEMPERATURA

O sensor é um dispositivo capaz de traduzir informações do meio para outros sistemas. No caso desse projeto, o sensor terá de ser capaz de ler a temperatura corporal do indivíduo e comunicar ao microcontrolador essa leitura. Como medida restritiva devido ao coronavírus, o sensor deve ser capaz de aferir a temperatura sem contato, com objetivo de evitar contaminação tópica. Para tal, é necessário o uso de termômetros infravermelhos.

Esse tipo de termômetro tem seu funcionamento baseado na radiação infravermelha emitida por corpos quentes. Todo objeto com temperatura acima de zero absoluto (0 Kelvin) emite radiação eletromagnética em sua superfície, proporcional a sua temperatura. Parte dessa radiação é infravermelha, e com ajuda de lentes, pode ser captada por um elemento detector (como termopilhas, detectores piroelétricos, bolômetros, e detectores fotônicos). Esse por sua vez gera um sinal elétrico proporcional à radiação que será processado digitalmente até se tornar o valor de temperatura do objeto (OPTRIS, 2019).

Para efeitos práticos, buscou-se um sensor já inserido em um módulo para prototipagem com interface de comunicação já aplicada, para que não fosse necessário realizar condicionamentos nem processamentos de sinais inerentes a aquisição da temperatura no microcontrolador, com o objetivo de facilitar a implementação.

Ao realizar a pesquisa de mercado, não foram encontradas muitas ofertas. Alguns sensores industriais custavam na casa de milhares de reais e/ou não possuíam uma precisão adequada para medição de temperaturas corporais. A melhor opção encontrada foi o módulo do componente MLX90614, e foi comprado o modelo BAA.

Figura 4: Sensor MLX90614.

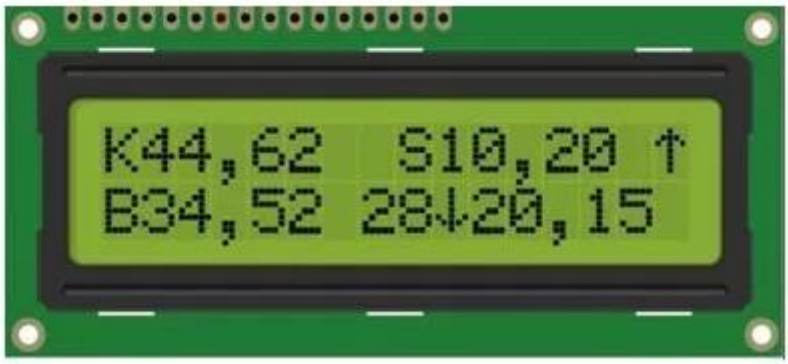
Fonte: Melexis, 2019.

Porém, após testes, o mesmo foi substituído pelo modelo DCC, pelo campo de visão mais estreito (antes 90º, agora 35º) e pela maior precisão na faixa de 22 a 40 ºC, além de cumprir com as normas da ASTM (*American Society for Testing and Materials*) nas especificações de termômetros infravermelhos para determinação de temperaturas em pessoas. (ANEXO A; MELEXIS, 2019, pg 36).

2.4 *DISPLAY*

Os *display*s são ótimas soluções para quando se necessita mostrar informações ao usuário. No trabalho, será usado tanto para cumprimentar quanto para mostrar as temperaturas em forma numérica. No mercado, existem diversos modelos com várias tecnologias, como *LCD*, *OLED*, *AMOLED*, entre outras. Entretanto, devido a pouca quantidade de informações a serem mostradas e ao desejo de manter um baixo orçamento, é fácil optar pela opção mais popular e mais usada no campo da eletrônica, o HD44780, que é um *display LCD* de 16 por 2 caracteres.

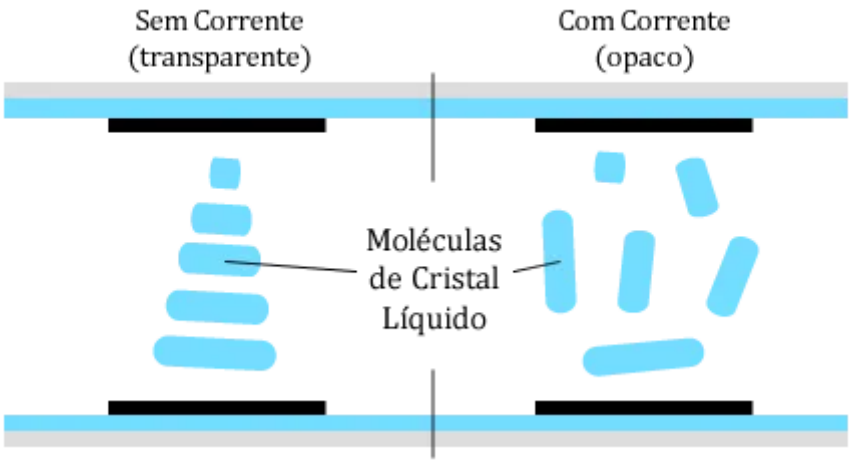
Figura 5: *Display* HD44780.



Fonte: Vida de Silício, 2017.

O funcionamento do *LCD* é baseado na propriedade que o cristal líquido tem de alterar seu estado natural de transparente para opaco quando submetido a uma carga elétrica. O *display* é formado por uma fina camada de cristal líquido entre duas placas de vidro com uma fonte de luz abaixo da estrutura (LARGURA, 2017).

Figura 6: Funcionamento do *LCD*.



Fonte: Vida de Silício, 2017.

2.5 SENSOR DE PRESENÇA

O sensor de presença será necessário no trabalho para que se possa detectar a mão da pessoa e então liberar o álcool em gel no momento certo. Existem vários tipos de sensor de presença/movimento, como os *PIR* (Passive infrared), os *MW* (Microwave), os de refletância, os ultrassônicos e os de vibração. Pelo mesmo argumento do *display*, visando a simplicidade, facilidade de acesso e o baixo custo, foi escolhido o sensor de movimento PIR DYP-ME003.

Figura 7: Sensor de presença PIR DYP-ME003.

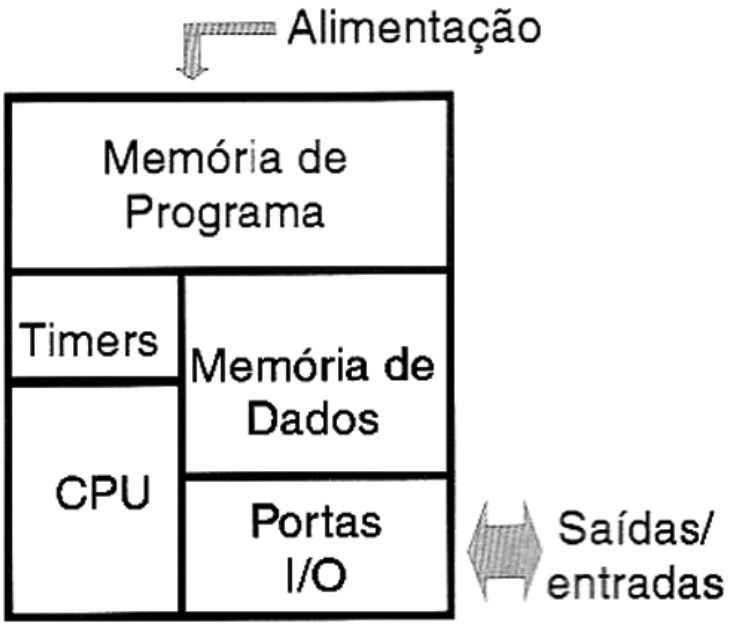
Fonte: Adafruit, 2014.

Esse sensor é feito a partir de um outro sensor chamado de piroelétrico, que funcionam baseados em um material chamado de eletreto. O eletreto apresenta carga elétrica natural em suas faces, mas essa carga varia sensivelmente na presença de radiação infravermelha. O circuito mostrado na imagem do módulo do sensor serve para amplificar o fraco sinal produzido pelo eletreto, regulação da tensão de alimentação, ajuste de sensibilidade e implementar os modos de funcionamento com *trigger* repetível e não-repetível (INSTITUTO NEWTON C. BRAGA, 2008).

2.6 MICROCONTROLADOR

Um microcontrolador é um circuito integrado que é capaz de realizar operações lógicas, armazenar dados, temporizar e se comunicar com o meio exterior pelas portas de entrada e saída. Ele pode ser interpretado como um computador muito simplificado e miniaturizado. Seu principal papel é realizar funções básicas de controle sobre circuitos a partir de informações que fluem por ele, como por exemplo, controlar um ventilador, um sistema de abertura de portas ou os movimentos de um robô (BRAGA, s.d.).

Figura 8: Estrutura interna de um microcontrolador.



Fonte: Instituto Newton C. Braga, s.d.

No caso do projeto, o microcontrolador será o elemento responsável por processar os eventos de entrada de dados, que são detecção da mão e leitura de temperatura, e os eventos de saída, que são comandar o acionamento do motor e enviar dados ao *display* e ao servidor. Então, analisando o mercado, chegou-se na escolha do ESP8266 como microcontrolador principal, pois possui conectividade *Wi-Fi* embutida e tem poder de processamento suficiente para as ações desejadas.

Os microcontroladores PIC foram descartados pois não possuíam conexão *Wi-Fi* nativa. Já a plataforma Arduino, apesar de possuir modelos com *Wi-Fi* embutido, possuem preços ainda muito elevados. Também não foi escolhido o sucessor do ESP8266, o ESP32, pois, apesar de atender todos os requisitos e possuir melhorias em várias características, ele seria um superdimensionamento desnecessário no projeto, perdendo contra seu antecessor na questão de custo.

Além disso, também foram encontradas alternativas ao microcontrolador escolhido, como o chip da *Realtek* RTL8710, que possui algumas características mais interessantes e pelo mesmo custo. Porém, ainda assim não foi a escolha preferida devido à grande comunidade do ESP8266, o que torna o processo de aprendizado e, principalmente, de *debugging*, muito mais fácil de se realizar.

Para fins de prototipagem, foi escolhido um módulo NodeMCU v3 Lolin, que contém ESP8266 como chip principal. Sua grande diferença é a maior facilidade na disposição dos pinos e a conexão USB embutida, facilitando a programação do microcontrolador.

Figura 9: Módulo NodeMCU v3 Lolin.

Fonte: Autocore Robótica, 2020.

2.7 BANCO DE DADOS

Segundo a Oracle, “Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador.” (ORACLE, 2021)

Ainda, existem duas categorias de banco de dados, os bancos relacionais e os não-relacionais. Os relacionais são organizados por estruturas de tabela, e são compostos por colunas, tuplas ou registros. Utilizam uma linguagem chamada de SQL e são muito utilizados no mundo inteiro. Os não-relacionais podem trabalhar com dados mistos (imagens, mapas e tabelas) que não podem ser tabulados em linhas e colunas, sendo muito utilizado em grandes soluções baseadas em nuvem. (GOMES, 2019)

No projeto, foi decidido usar o banco de dados não-relacional da Google chamado *Firebase Realtime Database* (GOOGLE, 2021), devido aos serviços disponíveis pela empresa no plano gratuito, como atualização em tempo real dos dados, fácil visualização, gerenciamento e aquisição dos dados, e ser acessível

diretamente pelo navegador web, o que o tornou um bom ambiente para a prototipagem do trabalho.

2.8 SERVIÇO DE HOSPEDAGEM DA PÁGINA WEB

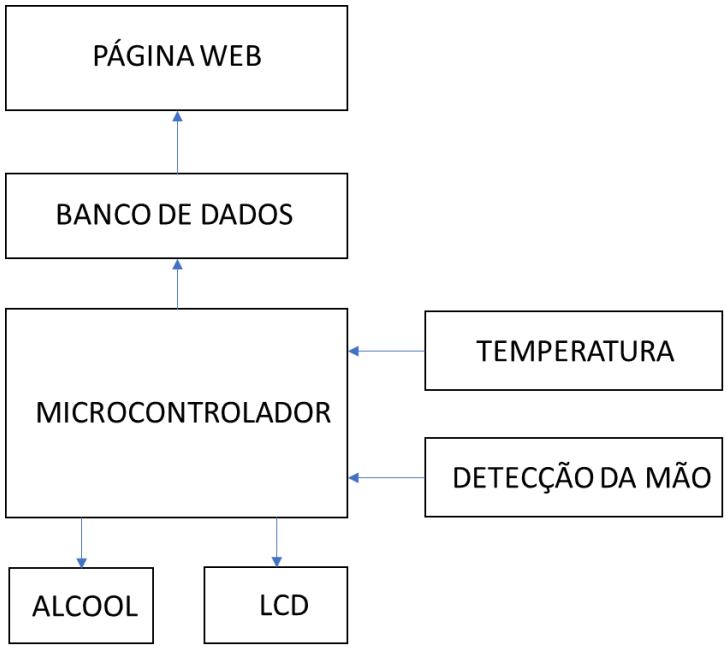
Hospedagem de sites é o serviço de alugar um espaço digital dentro de um servidor para armazenar todos os arquivos e dados necessários para o funcionamento de um site ou uma aplicação (LONGEN, 2020).

No caso do projeto, foi usado o serviço da Google de *Firebase Hosting* (GOOGLE, 2021), dentro da mesma plataforma do *Firebase* utilizada no banco de dados, pois gerir ambos no mesmo ambiente facilitou o desenvolvimento do trabalho. Além disso, o serviço possui um plano gratuito ótimo para testes, com direito a 5GB de armazenamento e 1GB de download por dia, 20 mil operações de upload por dia e 50 mil operações de download por dia.

3 MATERIAL E MÉTODOS

A estrutura do projeto está mostrada no diagrama de blocos abaixo, que mostra o microcontrolador como elemento central. Em relação ao µC, o distribuidor de álcool em gel e o *LCD* são dispositivos de saída e o sensor de temperatura e o de presença são dispositivos de entrada. O envio de informações ao banco de dados também é feito pelo µC, já a aplicação de página da *web* é um serviço separado, que usa o banco gerado por ele.

Figura 10: Diagrama geral do projeto



Fonte: O autor, 2020.

O desenvolvimento do projeto iniciou com a calibração do sensor de temperatura, com a meta de equiparar-se às pistolas infravermelhas do mercado. Depois, a primeira versão do protótipo foi elaborada em software e em seguida trazida para o hardware. Posteriormente, foi realizada a programação do microcontrolador em C/C++ e da página web em HTML, CSS e JavaScript.

3.1 CALIBRAÇÃO SENSOR DE TEMPERATURA

O primeiro passo do desenvolvimento envolveu o uso de um Arduino UNO e do sensor MLX90614. A programação foi realizada usando a IDE do Arduino. Segundo o *datasheet* do MLX90614 (MELEXIS, 2019), o sensor já vem calibrado de fábrica e com emissividade definida como 1 (de corpo negro). Esse valor foi alterado para 0,98 (que é a emissividade da pele humana) (BIOMEC, 2020) mudando o valor no endereço de memória 0x04 da EEPROM (MELEXIS, 2019, pg.14) o código de teste está na figura 11:

Figura 11: Código usado para calibração do sensor.

```
#include <Wire.h> // I2C library, required for MLX90614
#include <SparkFunMLX90614.h> // SparkFunMLX90614 Arduino library

IRTherm therm; // Create an IRTherm object to interact with throughout

void setup()
{
  Serial.begin(9600); // Initialize Serial to log output
  delay(200);
  therm.begin(); // Initialize thermal IR sensor
  therm.setUnit(TEMP_C);
  therm.setEmissivity(0.98);
}
void loop() {
  if (therm.read()){
    Serial.print("Object: " + String(therm.object(), 1));
    Serial.write(' o'); // Degree Symbol
    Serial.println("C");
    Serial.print("Ambient: " + String(therm.ambient(), 1));
    Serial.write(' o'); // Degree Symbol
    Serial.println("C");
    Serial.println();
  }
  delay(600);
}
```

Fonte: O autor, 2020.

A biblioteca da *SparkFun* foi utilizada com a finalidade de simplificar o código, automatizando a leitura e escrita de dados nos registradores do sensor.

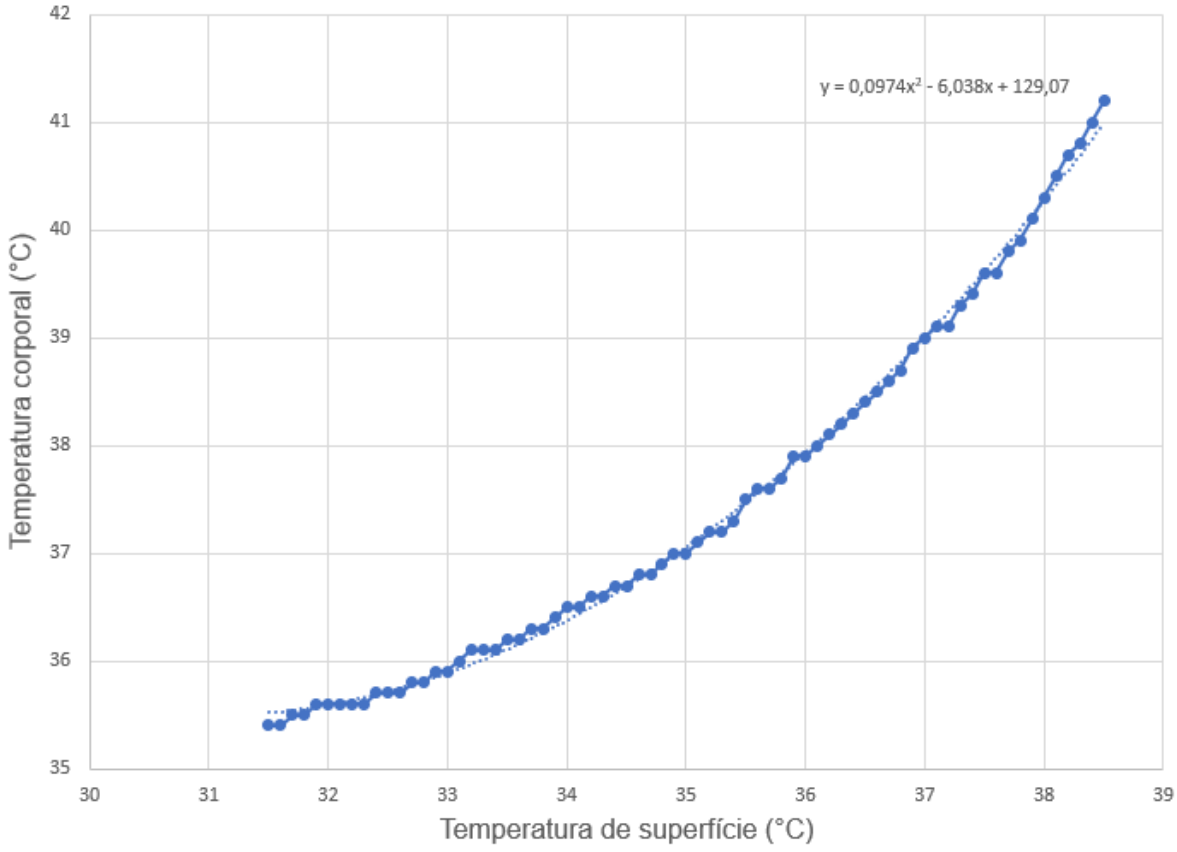
Usando um termômetro de mão FR1DZ1 (ACCUMED, 2011) no modo “objetos” como ponto de controle, notou-se que as leituras se assemelhavam, comprovando que a calibração de fábrica estava correta. Entretanto, é mais interessante trabalhar com o termômetro no modo “corpo humano”, pois esse valor é o mais parecido com os 36-37ºC da temperatura corporal normal. Isso acontece porque a temperatura superficial da pele é geralmente mais baixa que a temperatura

interna do corpo, e os termômetros comerciais possuem um ajuste interno para que realiza essa compensação (COLEMAN, 2020). Infelizmente, os algoritmos utilizados não são disponibilizados ao público, mas pode-se observar na patente EP1680652A1 (MICROLIFE, 2006) um exemplo do método utilizado, onde a compensação é feita a partir da criação de uma tabela baseada em números substanciais de testes clínicos.

Para contornar esse problema, foi realizado o seguinte experimento: num pote grande de vidro, foi colocada água aquecida a 40ºC. Esperados alguns minutos, usou-se o termômetro de controle para aferir a temperatura do pote nos 2 modos de operação e os valores foram anotados. À medida que a água esfriava, esse processo foi repetido para todas as temperaturas entre 38.5 e 31.5ºC em passos de 0.1ºC. Dessa maneira, foi criado um gráfico no Excel e escolhida uma curva de tendência quadrática para relacionar as duas temperaturas. No fim, encontramos a relação:

$$T_{corpo} = 0{,}0974 \cdot T_{superfície}^2 - 6{,}038 \cdot T_{superfície} + 129{,}07 \qquad (1)$$

Figura 12: Gráfico de temperatura corporal x temperatura superficial.



Fonte: O autor, 2020.

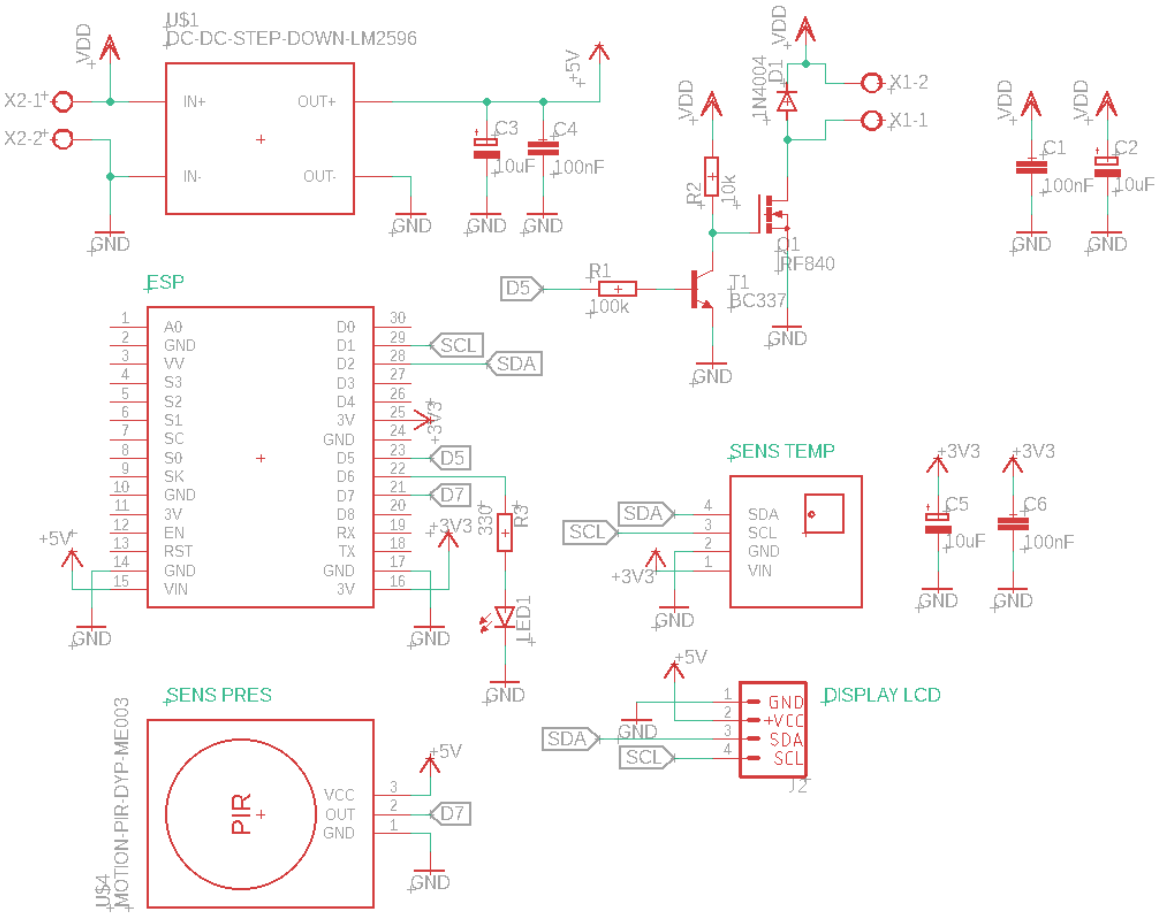
Entretanto, apesar da fórmula encontrada ser uma boa estimativa para os testes do protótipo, deve-se lembrar que a temperatura superficial da pele depende não só do local onde é medido, mas também depende da temperatura ambiente (SUAREZ F., NOZARIASBMARZ A., VASHAEE D e ÖZTÜRK M.C., 2016), portanto, em dias muito frios a temperatura esperada da pele humana também cai e a fórmula encontrada não compensa esse aspecto. No dia do experimento, fazia-se 26,6 ºC.

3.2 PROTÓTIPO

3.2.1 CIRCUITO ELETRÔNICO

O projeto de circuito eletrônico foi realizado no software Eagle, a primeira versão do trabalho não possuía *display*, sensor de presença e bomba peristáltica. Já a segunda versão, mostrada a seguir, implementou esses aspectos.

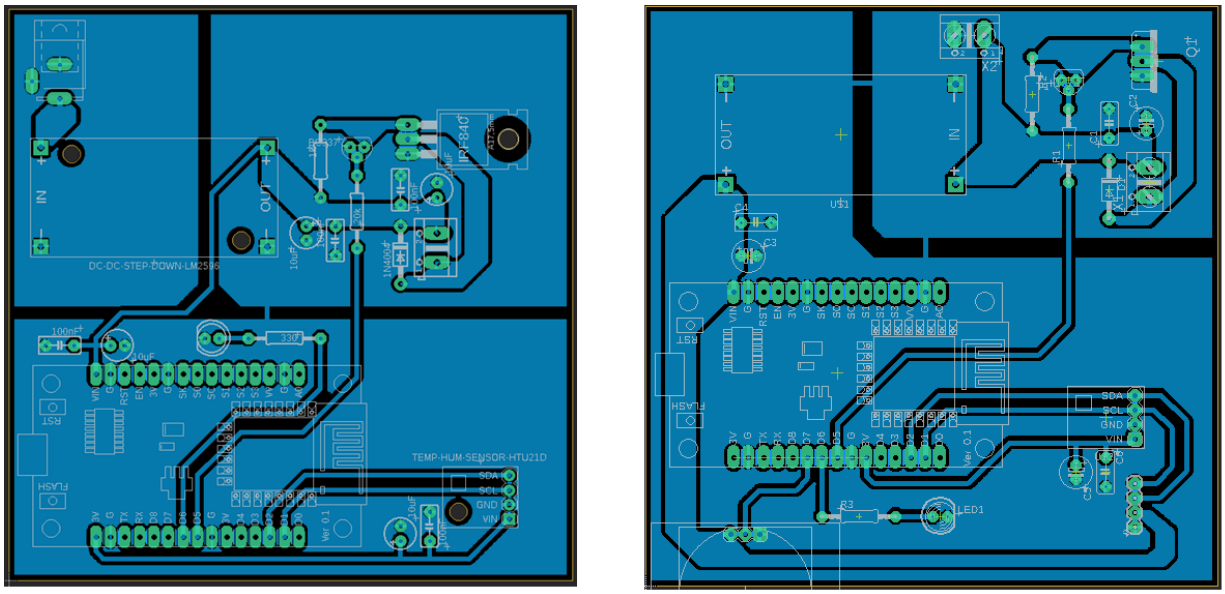
Figura 13: Esquemático completo do protótipo.



Fonte: O autor, 2021.

Na sequência, foi criado o *layout* do protótipo. Para tal, foram adotadas boas práticas no desenho das trilhas, como evitar ângulos de 90º, separar referenciais de tensão entre a parte de potência (driver da bomba) e a parte lógica (µC e sensores), e alargar trilhas que conduziam uma corrente maior. Além disso, uma vez que o método utilizado para passar o desenho para a PCI seria o método térmico, com o agravante de ser produzido dentro de casa, respeitando o isolamento social, foi também aumentado o distanciamento entre trilhas e *pads* dos componentes, para evitar os erros gerados pelo processo.

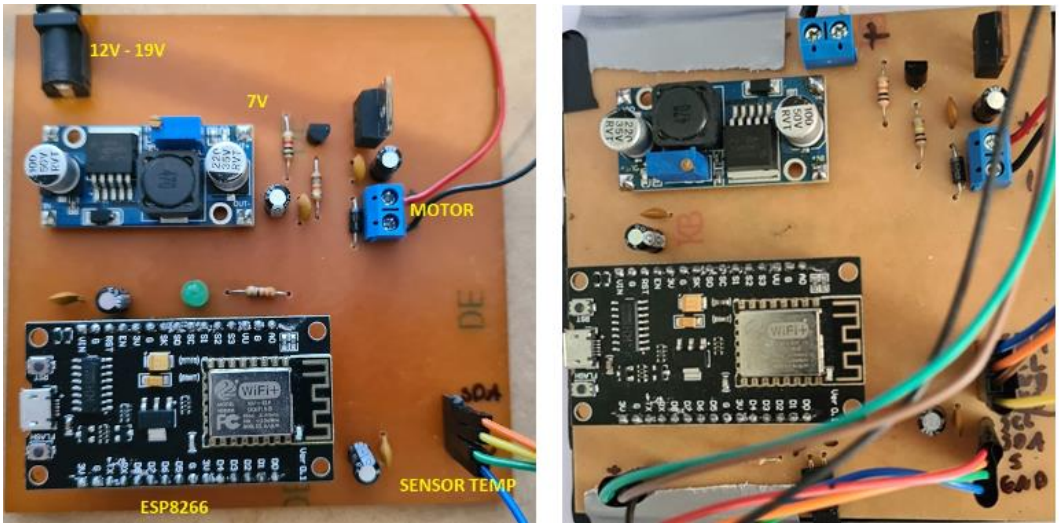
Figura 14: *Layout* da primeira versão (esquerda) e da segunda versão (direita) do protótipo.



Fonte: O autor, 2021.

O *layout* foi então impresso em papel fotográfico por uma impressora a laser e colocado sobre a parte de cobre de uma placa de fenolite. Usando um ferro de passar roupas, esquentou-se a superfície com a impressão e depois, usando água corrente, retirou-se o papel, com o desenho transferido para a placa. Mergulhou-se a PCI numa solução de percloreto de ferro com a finalidade de corroer todo o cobre não marcado pela tinta. Em seguida a placa foi furada com um furador de placas e envernizada. Depois, os componentes foram posicionados e soldados, com resultado final na próxima figura.

Figura 15: Placa finalizada da primeira versão (esquerda) e da segunda versão (direita).



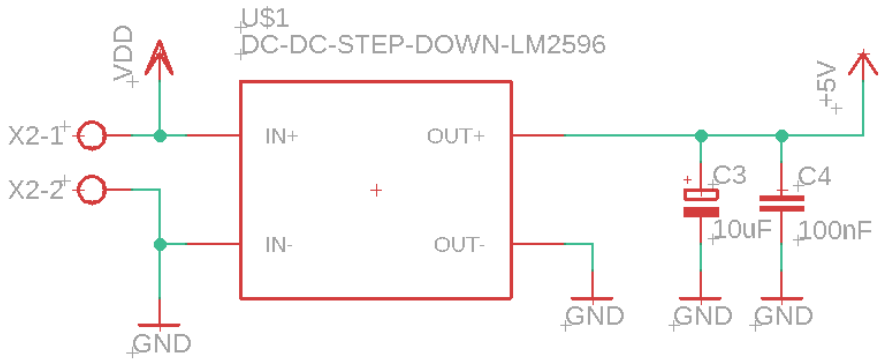
Fonte: O autor, 2021.

Cada parte do circuito será explicada adiante.

3.2.1.1 ALIMENTAÇÃO

O circuito de alimentação é aquele cuja função é mandar energia elétrica a todos os demais componentes da PCI. Foi definido que o projeto fará uso de uma fonte de notebook de 19V para converter a entrada AC da tomada para *DC*. Na placa, entretanto, essa tensão terá que ser abaixada novamente, uma vez que o módulo do ESP8266, do *display LCD* e do sensor de presença precisam de 5V e o sensor de temperatura precisa de 3,3V. Para tal, será usado mais um módulo, dessa vez o conversor *DC*/*DC* baseado no circuito integrado LM2596. Esse módulo consiste basicamente de um conversor do tipo *buck* que recebe uma tensão *DC* de entrada e a abaixa na saída, baseada no rápido chaveamento de um transistor aliado a elementos de armazenamento de energia. Sua eficiência energética é muito maior que dos reguladores lineares, que abaixam a tensão passivamente, dissipando a energia como calor.

Figura 16: Esquemático do circuito de alimentação.



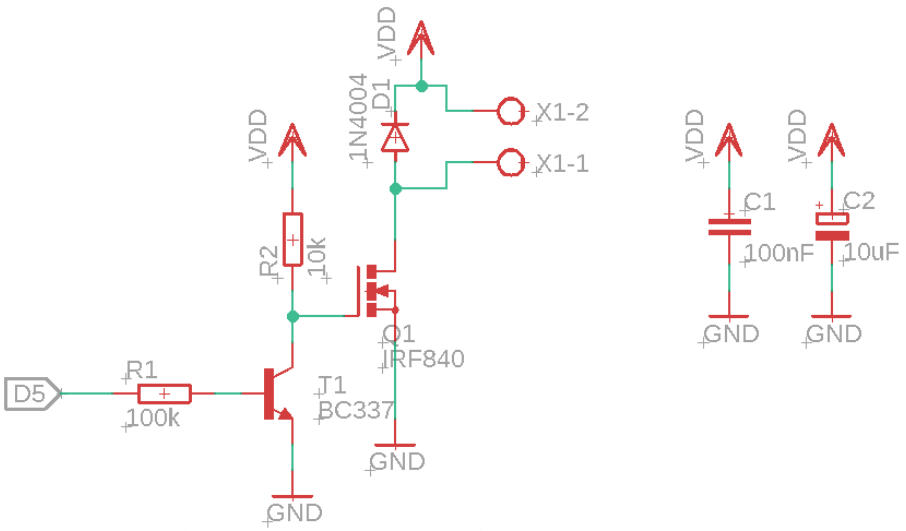
Fonte: O autor, 2021.

Na figura 15, X2 representa o borne que recebe os 2 fios provenientes da fonte externa, VDD representa 19V e GND o referencial de tensão. C3 e C4 são capacitores de desacoplamento que estão localizados perto da saída do módulo conversor e também perto da entrada do módulo do µC. A tensão de saída é de 5V.

3.2.1.2 DRIVER

O driver é o circuito responsável pela ativação do motor da bomba peristáltica. Primeiro, verificou-se que o motor possuía corrente nominal de 450mA, portanto foi decidido usar um transistor *MOSFET* por sua maior capacidade que o *BJT* de conduzir corrente. Foi escolhido o modelo IRF840, por ser um modelo facilmente encontrado no mercado e possuir capacidade para suprir uma corrente de dreno até 5,1A (VISHAY SILICONIX, 2016). Entretanto, por sua tensão V_{GS} de *threshold* ser entre 2 e 4V e as portas digitais do µC possuírem como nível alto apenas 3,3V, decidiu-se usar um *BJT* para chavear o *MOSFET*. Foi escolhido o transistor NPN BC337-25 por ser também facilmente encontrado e por seu baixo custo.

Figura 17: Esquemático do circuito de driver.



Fonte: O autor, 2021.

C1 e C2 são capacitores de desacoplamento cujos valores foram escolhidos sendo 100nF (para filtrar frequências mais altas, como a frequência de chaveamento do módulo conversor) e 10uF (para filtrar frequências mais baixas, como da rede elétrica). X1 representa o conector borne que é conectado aos terminais do motor. Uma vez que o motor é uma carga indutiva, é necessário um diodo de roda livre conectado em antiparalelo com ela, para evitar picos elevados de tensão quando a carga é desligada. Para tal, foi escolhido o diodo 1N4004, também pelo motivo de ser barato e fácil de encontrar.

Os resistores R1 e R2 possuem a função de polarizar o *BJT* para que opere entre as regiões de corte e saturação, seus valores foram calculados da seguinte forma: primeiro, escolheu-se o valor arbitrário de 10kΩ para R2 e verificou-se que a máxima corrente que passaria sobre ele seria de 1,9mA (19V da fonte dividido por 10kΩ), que é bem menor que a corrente máxima de coletor do BC337, que é de 800mA (ONSEMICONDUCTOR, 2013). Em seguida, para descobrir a mínima corrente de base, usamos o valor máximo da corrente (1,9mA) e dividimos pelo valor mínimo de ganho de corrente (β) do transistor utilizado, que vale 160, obtendo um valor de 11,87µA. Como margem de segurança, para garantir a saturação, dobramos a corrente encontrada para então chegar na corrente mínima de base de 23,7µA.

$$I_B > 2 \cdot \frac{I_C}{\beta} \quad (2)$$

Considerando a saída do µC (V_O) como 3,3V e a queda de tensão V_{BE} do *BJT* como 0,7V, temos que:

$$R_B < \frac{V_O - V_{BE}}{I_B} \quad (3)$$

$$R_B < \frac{3,3 - 0,7}{23,7\mu} \quad (4)$$

$$R_B < 109,7k\Omega \quad (5)$$

Portanto, foi escolhido o valor de R_B como 100kΩ. Para averiguar se o *BJT* estava operando em saturação, foi montado na protoboard e testado. Para V_O = 3,3V, os seguintes valores foram encontrados:

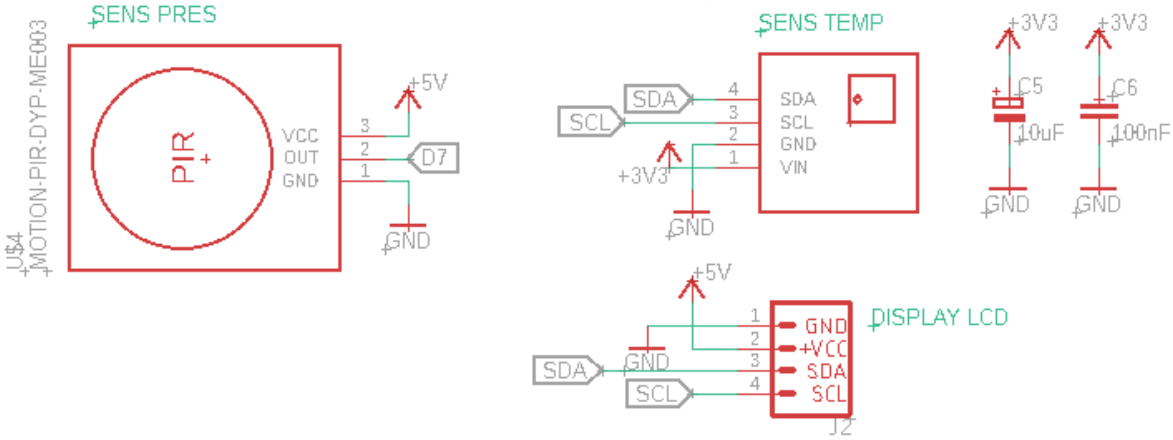
I_C = 1,93mA
I_B = 25µA
V_{BE} = 0,63V
V_{CE} = 0,2V

Verifica-se que o transistor está no modo de saturação porque, utilizando o ganho β mínimo do transistor usado, que é de 160 de acordo com o datasheet, para a corrente I_B medida de 25µA a corrente I_C esperada seria de 4mA, ao contrário do valor inferior medido de 1,93mA.

3.2.1.3 SENSORES E *DISPLAY*

Os sensores e o *display* são componentes que se comunicam com o µC pelas portas de entrada e saída. O sensor de presença se comunica pela porta D7 enviando pulso de alguns segundos em nível alto se for detectada a presença da mão e nível baixo se não. O sensor de temperatura e o *display LCD* se comunicam com o µC usando o protocolo I2C, sendo o µC o *master* e os 2 dispositivos os *slaves*. Eles compartilham as trilhas sem conflito pois possuem diferentes endereços, assim, só reagem à comunicação com o master se seu endereço de memória for detectado. Os capacitores C5 e C6 são de desacoplamento e ficam perto do sensor de temperatura.

Figura 18: Esquemático dos sensores e *display*

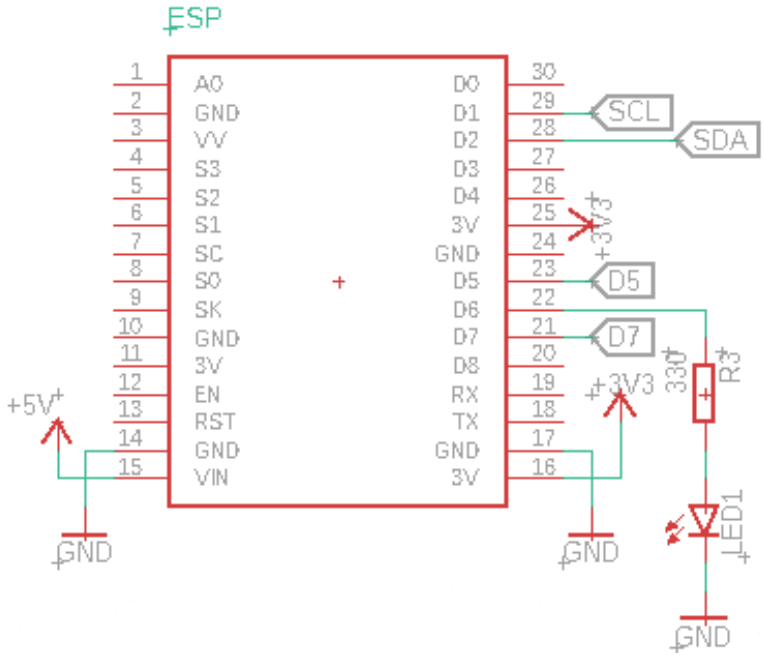


Fonte: O autor, 2021.

3.2.1.4 MICROCONTROLADOR

O microcontrolador é alimentado pelo pino de V_{IN} com os 5V recebidos do módulo conversor. Ele usa as portas D1 e D2 para comunicação I2C com o *display* e o sensor de temperatura, D5 como saída para acionar o motor da bomba, D6 como saída para o *LED* de indicação, e D7 como entrada para a comunicação com o sensor de presença.

Figura 19: Esquemático do módulo do microcontrolador.



Fonte: O autor, 2021.

Levando em consideração a escolha dos pinos, os pinos SCL e SDA só tinha conexão nas portas D1 e D2, portanto não houve flexibilidade. A escolha de D5, no entanto, foi decidida analisando o comportamento da porta no momento de boot do ESP8266. Uma vez que se desejava que o motor não iniciasse acionado, necessitava-se que a tensão V_O do circuito do driver estivesse em nível alto. Verificou-se que os pinos D0, D5, D6 E D7 apresentavam essa característica no momento de *boot* (KOYANAGI, 2018). Para desempate, foi considerado as funções extras de cada pino: D0 serve para “acordar” o µC caso colocado em modo *sleep*, D7 é usado na comunicação UART, e D5 e D6 são usados na comunicação SPI (EXPRESSIF, 2020). Foi decidido que os pinos com menor possibilidade de serem usados no futuro são D5 e D6. Para facilitar o desenho do *layout* da PCI, foi escolhido por fim, o pino D5 para o motor. Já para o *LED* e para o sensor de presença, não havia esse tipo de regra, e foram escolhidos arbitrariamente.

3.2.2 PROGRAMAÇÃO

A programação do ESP8266 foi escrita utilizando a linguagem C/C++ no ambiente de desenvolvimento *Visual Studio Code*. O código está dividido em quatro partes, a primeira consiste na inclusão das bibliotecas, definição de constantes e declaração de variáveis; a segunda nas definições das funções auxiliares; a terceira na função setup e a quarta na função loop.

Figura 20: Primeira parte do código do µC.

```
#include <LiquidCrystal_I2C.h>
#include <SparkFunMLX90614.h>  //Biblioteca SparkFunMLX90614
#include <Wire.h>              //Biblioteca para I2C
#include "FirebaseESP8266.h"

#define FIREBASE_HOST "teste-54db0.firebaseio.com"
#define FIREBASE_AUTH "WDZLHuUjXwpFJKkvHB[redacted]"
#define WIFI_SSID "Evolution1403"
#define WIFI_PASSWORD "[redacted]"

#define LED_EXTERN 12  //D6
#define MOTOR_PIN 14   //D5
#define PIR_PIN 13     //D7

IRTherm therm;
LiquidCrystal_I2C lcd(0x27, 16, 2);
FirebaseData firebaseData;
String path = "/measures/PKG";
FirebaseJson json1;
FirebaseJson json2;

const int numberOfReads = 5;
const double presenceTHold = 30;

int pkgCounter = 0;
int readsCounter = 0;
double Tobj = 25;
double Tamb = 25;
int lastTimestamp = 0;
```

Fonte: O autor, 2021.

As três primeiras bibliotecas mostradas na figura servem para facilitar a comunicação I2C usada entre o µC e o sensor de temperatura e o *display LCD*. Elas possuem em seus arquivos internos os endereços de memória de cada dispositivo e também funções pré-definidas que usam esses endereços para leitura e escrita dos mesmos. A última biblioteca carrega as funções que são usadas na comunicação com o banco de dados.

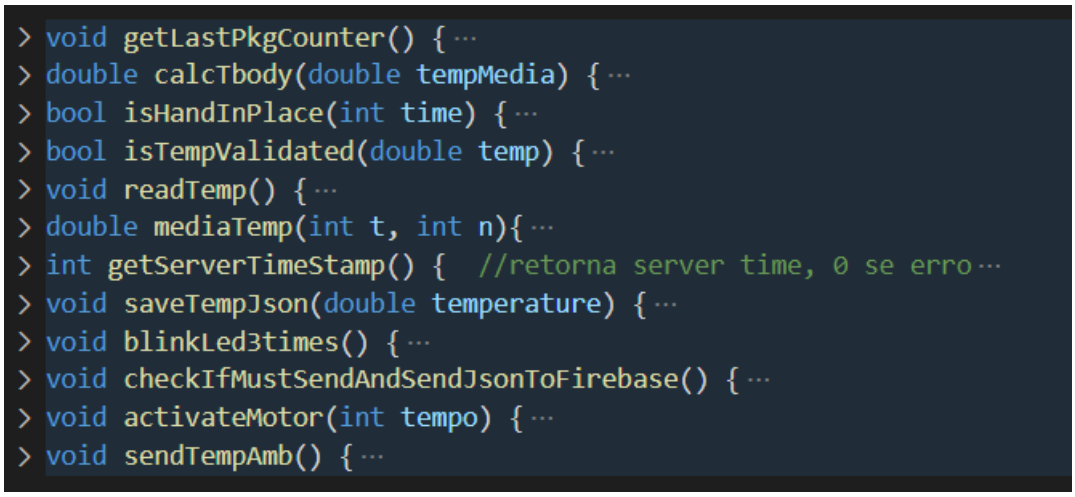
Logo em seguida, são definidas as constantes que possuem, em ordem: o endereço web do banco de dados, o token de segurança gerado pelo serviço de hospedagem do banco de dados, o nome da rede wireless a se conectar e a senha da mesma. Após, são definidos os pinos onde são conectados o *LED*, o motor e o sensor de presença.

Também é definido o número mínimo de temperaturas para que se possa enviar ao banco de dados, a temperatura limiar para detecção de presença, e as

variáveis globais que serão usadas pelas funções para ler e gravar as informações dentro do ESP8266.

Então, foram declaradas e definidas 12 funções auxiliares, mostradas na figura 21, que serão usadas pelas 2 funções principais, a setup e a loop, mostradas mais à frente.

Figura 21: Funções auxiliares.



Fonte: O autor, 2021.

Em ordem, as funções atendem as respectivas necessidades:

- a primeira é responsável por fazer uma conexão com o servidor para receber o número do último pacote enviado;
- a segunda faz o cálculo da temperatura corporal seguindo a fórmula (1);
- a terceira verifica se a mão está posicionada sobre o sensor;
- a quarta verifica se a temperatura está de acordo com o que se espera de uma temperatura corporal;
- a quinta lê a temperatura; a sexta realiza a amostragem e calcula a temperatura média;
- a sétima faz conexão com o servidor e recebe a data e horário atual na formatação *timestamp*;
- a oitava salva as temperaturas registradas (corporal, superficial e do ambiente) como um objeto *JSON*;
- a nona foi usada somente para fins de debug, piscando o *LED* 3 vezes;

- a décima verifica se o número de leituras é maior que o mínimo necessário para tentar enviar o objeto *JSON* ao banco de dados, caso positivo, também realiza esse envio;
- a décima primeira aciona o motor da bomba peristáltica;
- a décima segunda função auxiliar não foi implementada no final, mas possui o papel de enviar somente as temperaturas ambiente ao longo do dia.

Adiante, a função setup realiza a configuração básica do µC toda vez que é ligado, preparando-se para garantir o bom funcionamento da função loop que vem em seguida.

Figura 22: Função setup.

```
void setup() {
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(MOTOR_PIN, HIGH);
    digitalWrite(LED_EXTERN, LOW);
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(MOTOR_PIN, OUTPUT);
    pinMode(LED_EXTERN, OUTPUT);
    pinMode(PIR_PIN, INPUT);
    delay(200);
    lcd.init();  // initialize the lcd
    lcd.backlight();
    lcd.setBacklight(HIGH);
    lcd.setCursor(0, 0);
    lcd.print("Conectando Wifi");
    lcd.setCursor(0, 1);
    lcd.print(WIFI_SSID);
    therm.begin();          //Inicializa sensor de temperatura infravermelho
    therm.setUnit(TEMP_C);  //Seleciona temperatura em Celsius
    therm.setEmissivity(0.98);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
        delay(300);
    }
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
    firebaseData.setBSSLBufferSize(1024, 1024);
    firebaseData.setResponseSize(1024);
    getLastPkgCounter();
    lastTimestamp = getServerTimeStamp();

    lcd.setCursor(0, 0);
    lcd.print("Calibrando      ");
    lcd.setCursor(0, 1);
    lcd.print("sensor presenca ");
    delay(60000);
}
```

Fonte: O autor, 2021.

A primeira parte do setup consiste em definir o pino de entrada digital (sensor de presença) e os pinos das saídas digitais (*LED* e motor). O pino para o *LED* interno da placa também foi definido e foi usado para fins de debug. Ao analisar em conjunto com o esquemático mostrado anteriormente, percebe-se que configuração define o estado inicial dos *LED*s e do motor como desligados. Também é iniciado o *display LCD*, com a luz de fundo ligada e a mensagem de conectando à rede; o sensor de temperatura, trabalhando em celsius e com emissividade de 0,98; a conexão Wi-Fi, com o nome da rede e a senha; e a conexão com o banco de dados, usando o endereço web e o token de identificação e também limitando os tamanhos dos buffers para 1024 bytes. Depois, são chamadas duas funções auxiliares, para pegar o número do último pacote enviado e o último *timestamp*. Por fim o *LCD* mostra uma mensagem de calibração do sensor de presença e o código entra em um *delay* de 1 min, pois foi percebido nos testes que o sensor só funciona corretamente depois desse tempo.

Logo, a função loop é chamada e é nela que o µC vai se manter na maior parte do tempo.

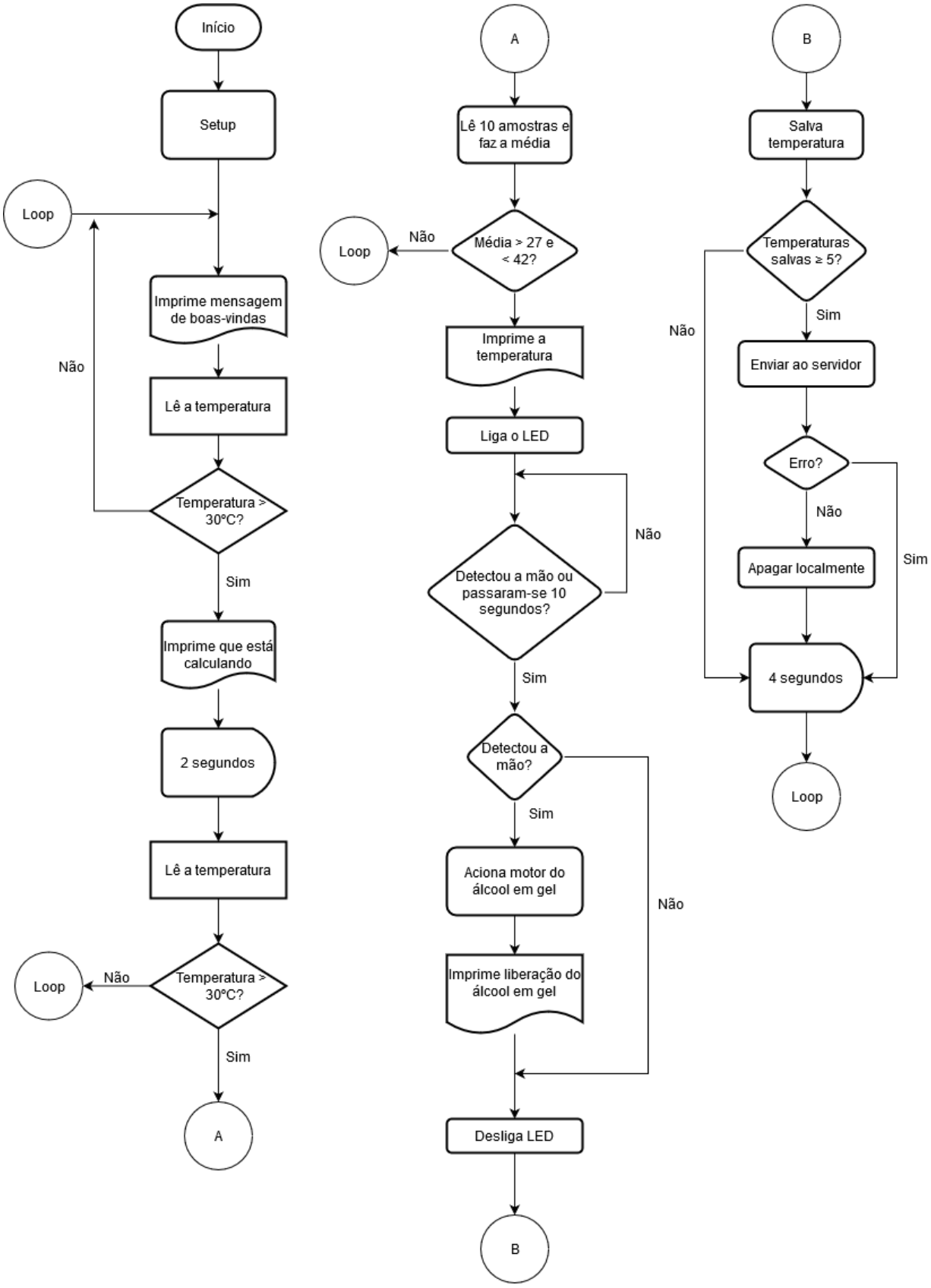
Figura 23: Função loop.

```
void loop() {
    lcd.setCursor(0, 0);
    lcd.print("Bem-vindo! Veja ");
    lcd.setCursor(0, 1);
    lcd.print("sua temperatura");
    readTemp();                  //le temp constantemente
    if (Tobj > presenceTHold) {  //se detectar pessoa
        digitalWrite(LED_BUILTIN, LOW);
        lcd.setCursor(0, 0);
        lcd.print("Calculando...   ");
        lcd.setCursor(0, 1);
        lcd.print("                ");
        if (isHandInPlace(2000)) {  //2s para pessoa ajustar mao abaixo do sensor
            double tmedia = mediaTemp(100,10);
            if (isTempValidated(tmedia)) {···
            delay(4000);                    //4s até a proxima leitura
            while (Tobj > presenceTHold) {  //só deixa ler dnv se tirar a mão···
        }
    } else {
        digitalWrite(LED_BUILTIN, HIGH);
    }
    delay(200);
}
```

Fonte: O autor, 2021.

Essa função segue a seguinte rotina, mostrada na figura 24 na forma de fluxograma:

Figura 24: Fluxograma do funcionamento detalhado da função loop.



Fonte: O autor, 2021.

A função loop faz uso de todas as funções auxiliares para que as ações mostradas na figura 24 possam ser realizadas. A função inteira pode ser encontrada no apêndice A.

Além disso, foi realizado um teste de memória para descobrir qual o número máximo de medidas que podem ser armazenadas internamente durante o funcionamento do protótipo. Para isso, foi inserido o trecho de código mostrado na figura 25 no início da função loop.

Figura 25: Teste de memória.

```
//################################
//##        TESTING SIZE MEMORY##
//################################
lcd.setCursor(0, 0);
lcd.print("INICIANDO TESTE ");
lcd.setCursor(0, 1);
lcd.print("DE MEMORIA      ");
delay(10000);
lcd.setCursor(0, 0);
lcd.print("                ");
lcd.setCursor(0, 1);
lcd.print("                ");
while(readsCounter < 1000){
lcd.setCursor(0, 0);
lcd.print("c: "  + String(readsCounter));
lcd.setCursor(0, 1);
lcd.print(String(ESP.getFreeHeap(),DEC));
saveTempJson(36.1);
delay(250);
```

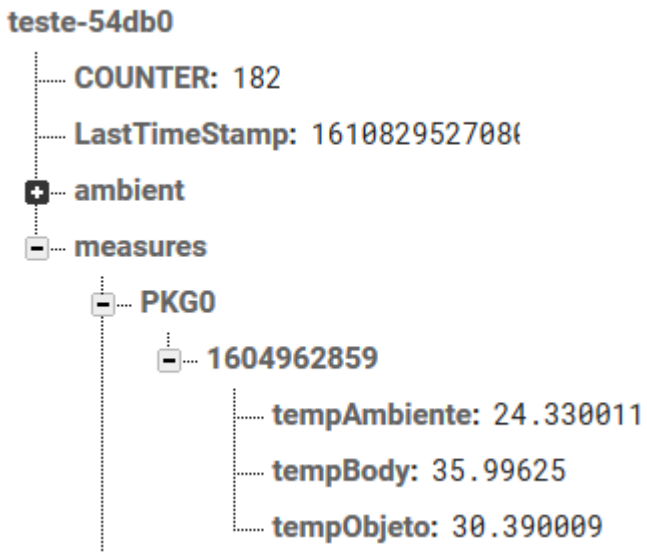
Fonte: O autor, 2021.

Foi descoberto que o número máximo para armazenamento interno é de 42 medidas. Após isso, usou-se a função auxiliar para enviar as 42 medidas ao servidor com sucesso e concluir o teste de memória.

3.3 BANCO DE DADOS

A organização do banco de dados ficou da seguinte maneira:

Figura 26: Organização do banco de dados no site de hospedagem.



Fonte: O autor, 2021.

No caminho raiz “/” ficam dois valores inteiros, “Counter” que é usar para contar quantos pacotes o ESP8266 enviou, e “LastTimeStamp”, que mostra a data e horário no formato de *timestamp* da última conexão realizada. No caminho “ambient” ficam as temperaturas ambientes (não usadas) e no caminho “measures” ficam os dados de temperaturas enviados. Dentro de “measures”, seus filhos tem como nome o número do pacote (o número do “counter” atual) e dentro de cada pacote, seus filhos tem como nome o *timestamp* da data e horário que a medida foi realizada.

3.4 PÁGINA WEB

A página web foi desenvolvida usando o software *Visual Studio Code*, a partir de um modelo de página disponível no site educacional W3Schools (TUTORIA W3SCHOOLS, 2017). Seu princípio de organização (HTML) e *layout* (CSS) foram mantidos, apenas adicionando-se os scripts (JavaScript) de comunicação com o banco de dados, de geração de gráficos e de criação do calendário. O site está hospedado no Firebase no endereço web “https://edimartccplotly.web.app/”.

O HTML começa com o *head*, referenciando os arquivos de estilos CSS e implementando os scripts.

Figura 27: Head do arquivo HTML.

```
<!-- https://www.w3schools.com/css/css_website_layout.asp -->
<!DOCTYPE html>
<html>

<head>
  <link rel="shortcut icon" href="#" />
  <link rel="stylesheet" href="css/styles.css"/>
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.9.0/themes/base/jquery-ui.css" />
  <script src="https://code.jquery.com/jquery-1.8.2.js"></script>
  <script src="https://code.jquery.com/ui/1.9.0/jquery-ui.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.0.0/firebase-app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.0.0/firebase-database.js"></script>
  <script src=js/moment.js></script>
  <script src=js/Chart.min.js></script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels@0.7.0"></script>
  <script src=js/utils.js></script>
  <script src=js/firebase.js></script>
</head>
```

Fonte: O autor, 2021.

Os arquivos de CSS são o styles.css, que dá formatação à página em si, e o jquery-ui.css, que dá formatação ao calendário. Os arquivos de JavaScript são: jquery-1.8.2.js e jquery-ui.js, que estão sendo usados para o calendário (apesar da biblioteca jQuery ter um enorme potencial, foi escolhido se ater apenas a esse recurso, pois não era necessário adicionar interatividades e dinamismo muito mais complexos à página do projeto); firebase-app.js, firebase-database.js e firebase.js, que são usados para usar as funções de conexão com o banco de dados; moment.js, que é usado para se trabalhar com conversão de *timestamps* e datas. Chart.js, junto com seu plugin e utils.js, que são usados para a construção dos gráficos.

Depois, vem o *body* da página, contendo os elementos de texto e a organização das *divs*.

Figura 28: Body do arquivo HTML.

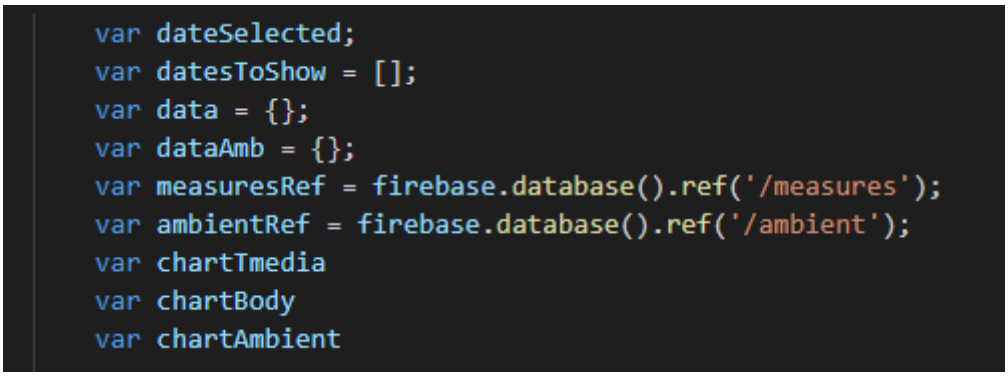
```
<body>
  <div class="header">
    <h1>Edimar - TCC</h1>
    <p>Engenharia Elétrica - UFPR</p>
  </div>
  <div class="topnav">···
  </div>
  <div class="row">
    <div class="leftcolumn">
      <div class="card">
        <h3>Selecione a data para análise</h3>
          <p>Data: <input type="text" id="calendar"></p>
      </div>
      <div class="card">
        <h2>Temperatura média por dia</h2>
          <div id="canvasgraph2">
            <canvas id="tmediagraph"></canvas>
          </div>
          <h2>Quantidade de pessoas por temperatura por dia</h2>
          <div id="canvasgraph1">
            <canvas id="bodygraph"></canvas>
          </div>
        </div>
    </div>
    <div class="rightcolumn">···
    </div>
  </div>

  <div class="footer">···
  </div>
  <script src="js/index.js"></script>
</body>
```

Fonte: O autor, 2021.

Ao fim do *body*, é adicionado o arquivo JavaScript principal, que constrói os gráficos baseados na data escolhida e no banco de dados, chamado de index.js. Ele inicia-se declarando as variáveis globais que serão usadas pelo Chart.js.

Figura 29: Variáveis globais do arquivo JavaScript.

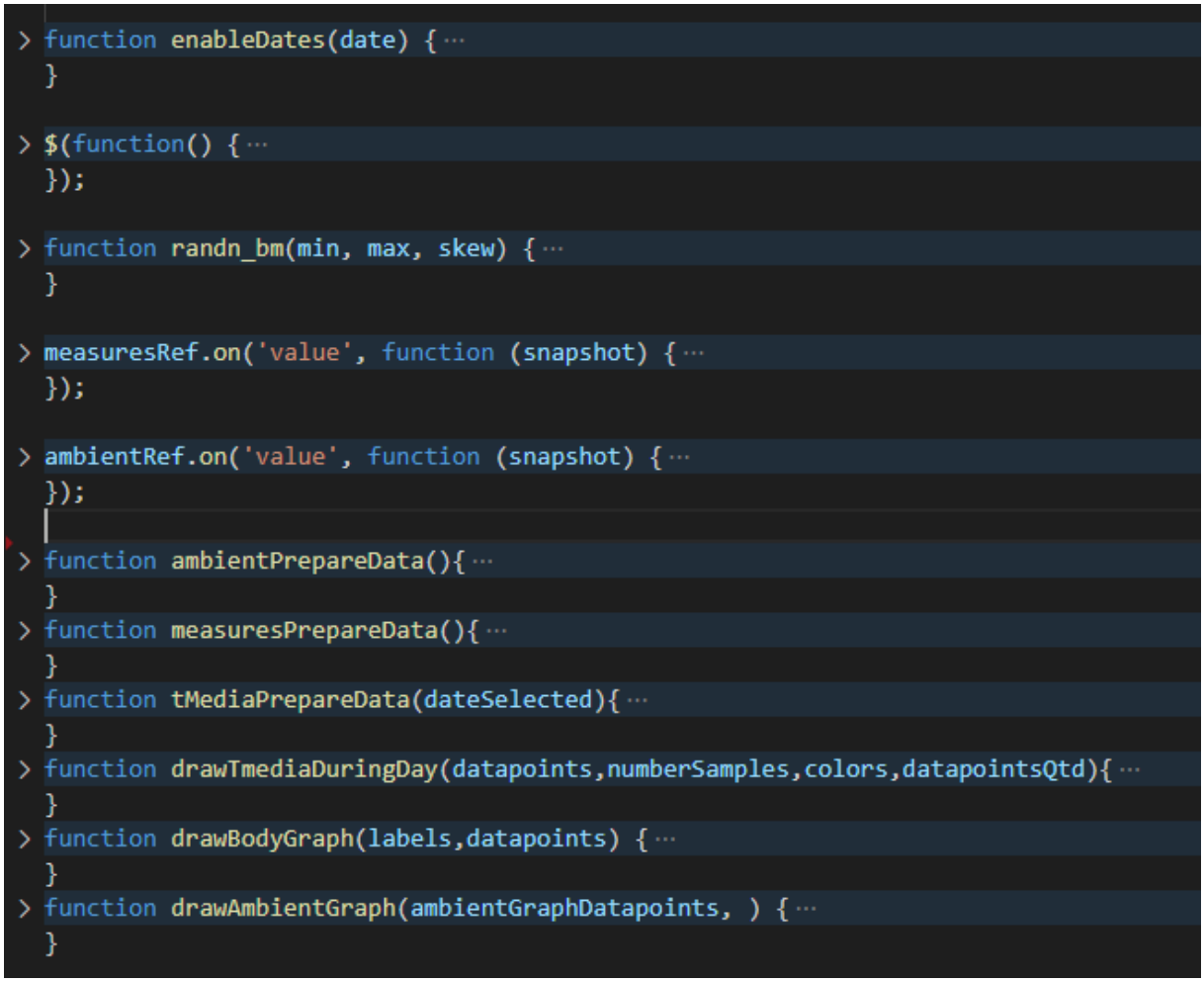


```
var dateSelected;
var datesToShow = [];
var data = {};
var dataAmb = {};
var measuresRef = firebase.database().ref('/measures');
var ambientRef = firebase.database().ref('/ambient');
var chartTmedia
var chartBody
var chartAmbient
```

Fonte: O autor, 2021.

Logo após, são definidas as funções que irão implementar o calendário, capturar os dados do Firebase e desenhar os gráficos na tela.

Figura 30: Funções do arquivo JavaScript.



Fonte: O autor, 2021.

O código flui conforme o fluxograma da figura 31 e funciona assim: primeiro, assim que a página é carregada, há uma requisição do banco de dados pelas temperaturas corporais (também há pelas temperaturas ambiente, mas não foi implementada na versão final da página) pela variável “measuresRef”, que chama a função “measuresPrepareData”, que por sua vez extrai os dados importantes, os organiza e os manda para “drawBodyGraph” e “tMediaPrepareData”, a primeira é responsável por desenhar o gráfico de “quantidade de pessoas por temperatura por dia” e a segunda organiza os dados para construir o gráfico “temperatura média por dia”. A construção em si do gráfico é automatizada pelo JavaScript incluído de nome “Chart.js”.

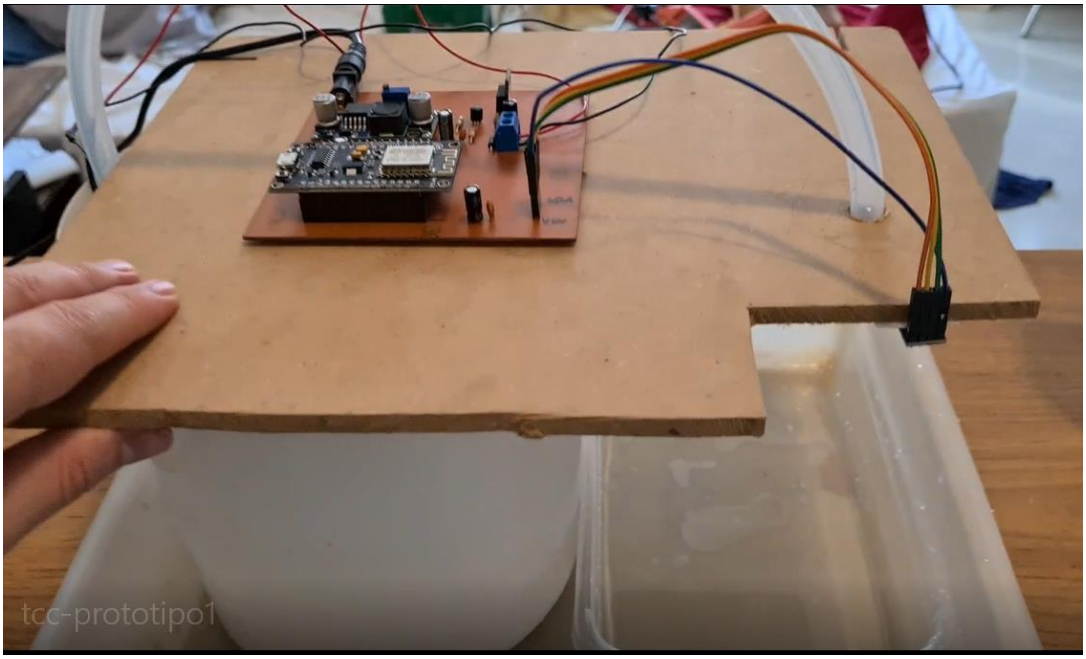
Figura 31: Fluxograma da construção dos gráficos da página web

Fonte: O autor, 2021.

3.5 HARDWARE

O desenvolvimento da parte física do protótipo começou com testes usando potes de plástico e um pedaço de MDF, mostrado na figura 32:

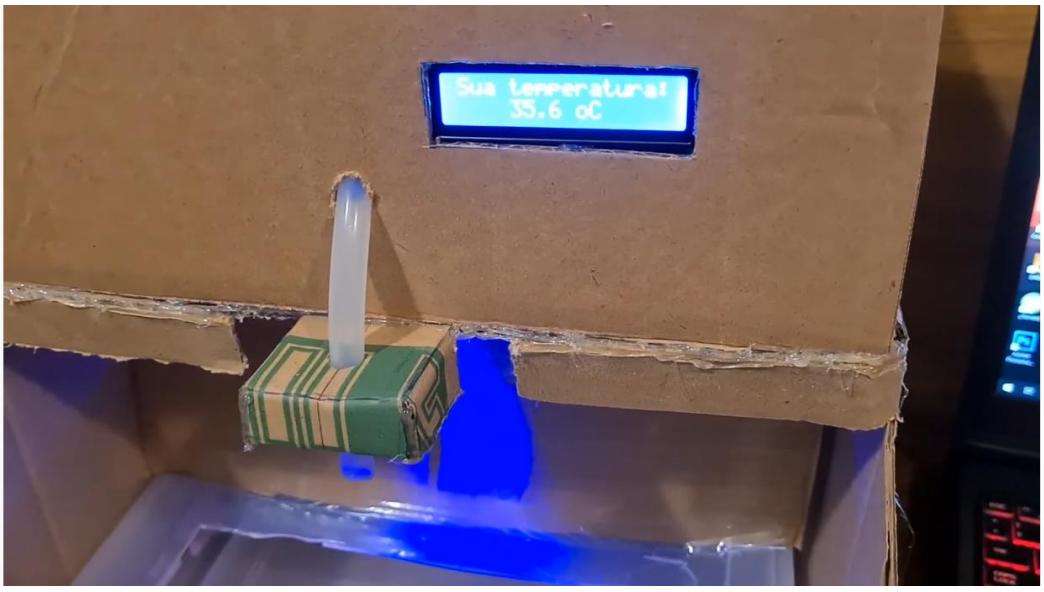
Figura 32: Protótipo inicial de testes



Fonte: O autor, 2021.

Nesse protótipo, foram testadas as funções básicas do projeto, como a medição de temperatura, a conexão com o servidor e o acionamento do motor do álcool em gel. Tudo isso serviu para confirmar o funcionamento da PCI e do código do µC. Logo depois, o segundo protótipo foi realizado, já com a implementação do *display LCD* e do sensor de presença. Esse protótipo foi feito com papelão e cola quente e serviu para testar todas as funcionalidades do projeto.

Figura 33: Segundo protótipo



Fonte: O autor, 2021.

Por fim, o protótipo final foi realizado utilizando como estrutura um *dispenser* de papel toalha, com auxílio de parafusos, fitas dupla face e abraçadeiras para fixação dos elementos.

Figura 34: Dispenser de papel toalha

Fonte: Leroy Merlin, 2021

O resultado final será apresentado mais adiante no trabalho.

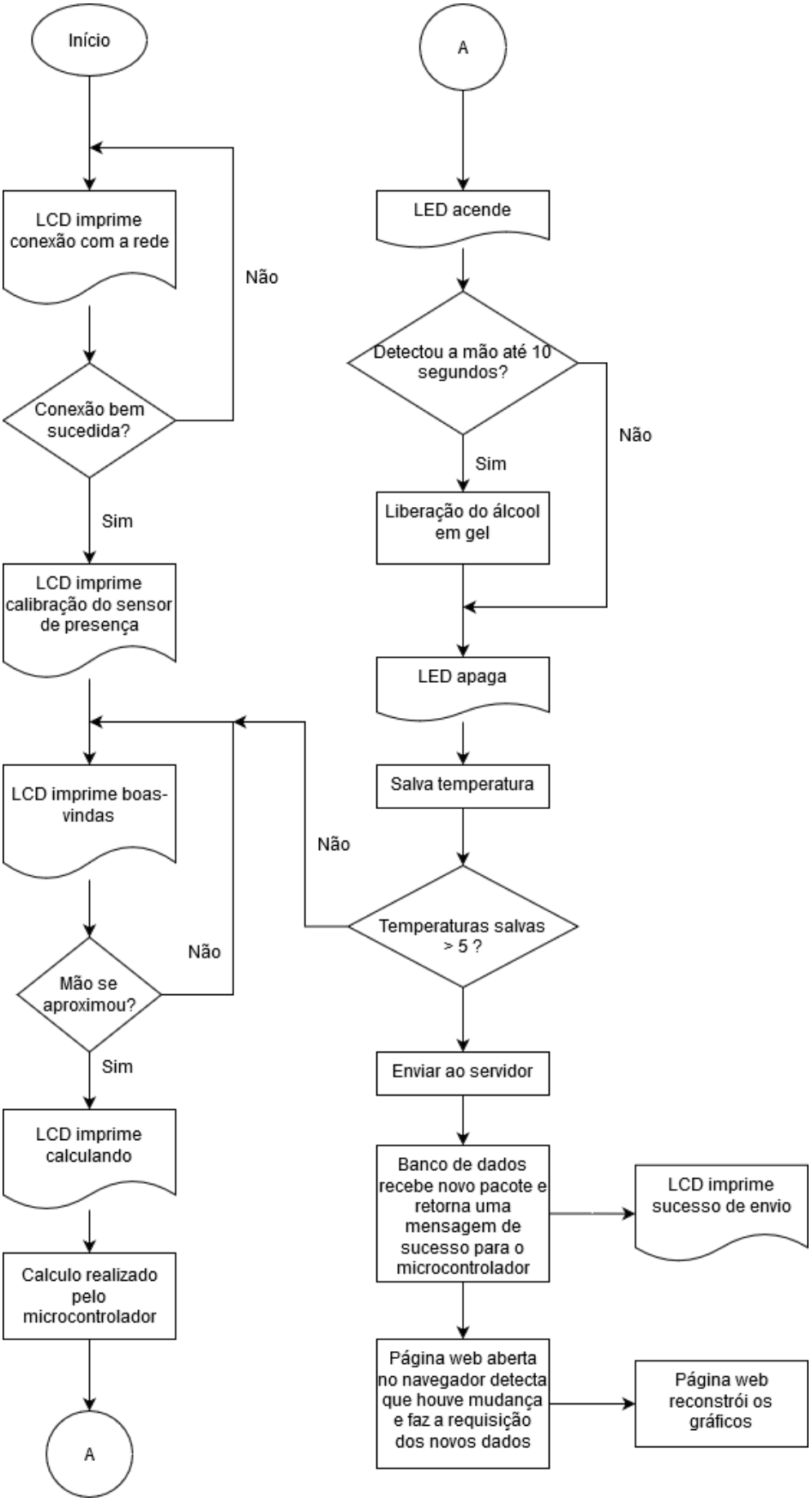
3.6 DESCRIÇÃO DO FUNCIONAMENTO DO PROTÓTIPO FINAL

O protótipo é ligado na tomada por meio de um cabo de energia tripolar para notebooks, mas somente após apertar o botão em sua lateral que é realmente ligado. Ao ligar, o *display LCD* mostra na tela uma mensagem que está se conectando à rede configurada em seu firmware. Se a rede não for encontrada ou se a senha estiver incorreta, ele permanecerá com a mesma mensagem na tela até que a rede seja encontrada ou seu firmware modificado para configurar outra rede. Após o sucesso, aparece uma mensagem de calibração do sensor de presença, e após um minuto, que é quando o sensor está pronto, a mensagem é trocada para uma mensagem de boas-vindas e convida o usuário e medir sua temperatura.

Ao se aproximar do sensor, o *display* imprime a mensagem de que está calculando a temperatura, mas na verdade ele só calcula 2 segundos depois. Esse

delay é proposital para que a pessoa posicione sua mão corretamente sobre o sensor antes na temperatura ser medida. Após o cálculo, a tela mostra a temperatura resultante e o *led* no centro do protótipo acende, indicando que o usuário pode colocar a mão abaixo do equipamento para receber o álcool em gel. Se a pessoa optar por receber o álcool em gel, aparece no *display* a mensagem de que ele está sendo liberado. Se não, após 10 segundos a ação é cancelada e o *LED* apaga. Após isso, para fins de debug, o protótipo mostra no *display* que a temperatura está sendo salva internamente e volta a mensagem de boas-vindas do início. Se 5 temperaturas foram feitas, o protótipo tenta enviá-las para o banco de dados. Se houver erro, ele tentará novamente quando a próxima leitura for realizada. Ele só apagará a memória interna quando o pacote for enviado com sucesso para o servidor. Conforme o teste mostrado anteriormente, é possível armazenar 42 medidas no armazenamento interno do microcontrolador.

Figura 35: Fluxograma do funcionamento do projeto.



Fonte: O autor, 2021.

3.7 MATERIAIS

Para avaliar os custos de materiais para a confecção do protótipo, foi criada seguinte tabela:

Tabela 1: Materiais e custos do protótipo.

Material	Custo/Un.	Qtd.	Unidade	Custo final
Borne 2 vias	R$2,40	2	Unitário	R$4,80
Módulo *DC-DC* Step-Down-LM2596	R$11,90	1	Unitário	R$11,90
Transistor *BJT* NPN BC337-25	R$0,19	1	Unitário	R$0,19
Transistor *MOSFET*-N IRF840	R$3,50	1	Unitário	R$3,50
Diodo IN4004	R$0,18	1	Unitário	R$0,18
Módulo NodeMCU v3 Lolin	R$42,66	1	Unitário	R$42,66
Módulo MLX90614-*DCC*	R$89,19	1	Unitário	R$89,19
Módulo *display LCD* 16 X 2 HD44780	R$17,90	1	Unitário	R$17,90
Módulo PIR DYP-ME003	R$12,90	1	Unitário	R$12,90
Resistores e capacitores diversos	R$2,50	1	Unitário	R$2,50
Bomba peristáltica	R$59,00	1	Unitário	R$59,00
Placa de Fenolite Virgem Simples	R$4,90	1	Unitário	R$4,90
LED Azul 5mm	R$0,24	1	Unitário	R$0,24
Porta Papel Toalha Interfolhado Dispenser Invoq Premisse	R$44,99	1	Unitário	R$44,99
Mangueira PVC 4mm	R$3,30	1	m	R$3,30
Botão liga-desliga	R$2,50	1	Unitário	R$2,50
Gel Antisséptico 70º INPM - Mãos COPERALCOOL 400g	R$13,99	1	Unitário	R$13,99
Fonte Carregador 19v 2.1a Notebook	R$49,99	1	Unitário	R$49,99

Fonte: O autor, 2021.

Dessa forma, o custo do protótipo do projeto foi de R$ 364,63 no total. Pode-se avaliar que o custo foi considerado satisfatório comparando com as soluções existentes no mercado: O *dispenser* automático S20-08 LP-E (LP DO BRASIL, c2020)

possui valor de venda de 385 reais, mas possui somente a liberação de álcool automatizada, já o totem TD PRO (ACTIVA-ID, c2021) é vendido por 1099 reais, mas, apesar de medir a temperatura e automatizar a liberação do álcool, não possui nenhum sistema de telemetria.

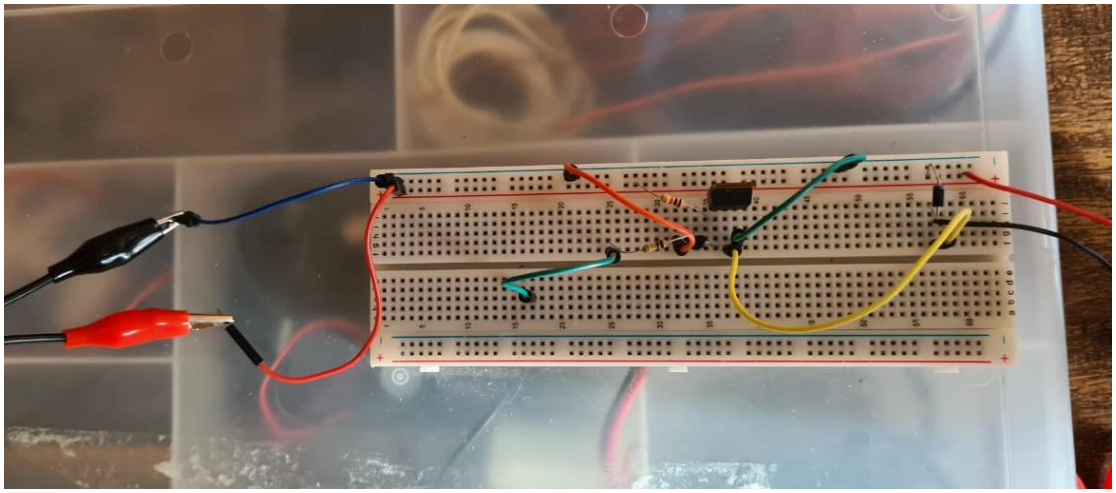
Também é válido salientar que o custo total seria muito inferior a esse no caso de produção em massa, onde os módulos separados poderiam ser reprojetados em um único circuito impresso e os materiais e componentes pedidos no setor atacadista.

4 APRESENTAÇÃO DOS RESULTADOS

4.1 VALIDAÇÃO DRIVER DO MOTOR

Para validação do driver de motor realizado, foi feito o circuito em protoboard mostrado na figura 36:

Figura 36: Teste driver do motor.



Fonte: O autor, 2021.

A tensão da fonte foi medida e foi encontrado o valor de 19,22V.

Para V_O = 3,3V, os seguintes valores foram encontrados para o *BJT*:

I_C = 1,93mA
I_B = 25µA
V_{BE} = 0,63V

V_{CE} = 0,2 V

Uma vez que o emissor está aterrado, $V_{CE} = V_C$ e pelo esquemático vê-se que $V_C = V_G$, portanto, para V_G = 0,02V, o *MOSFET* está em corte. Já para V_O = 0V, foi encontrado para o *MOSFET*:

V_G = 19,02V
I_D = 500mA

Com o *MOSFET* conduzindo a corrente necessária, conclui-se que está operando corretamente como chave.

4.2 CALIBRAÇÃO SENSOR DE TEMPERATURA

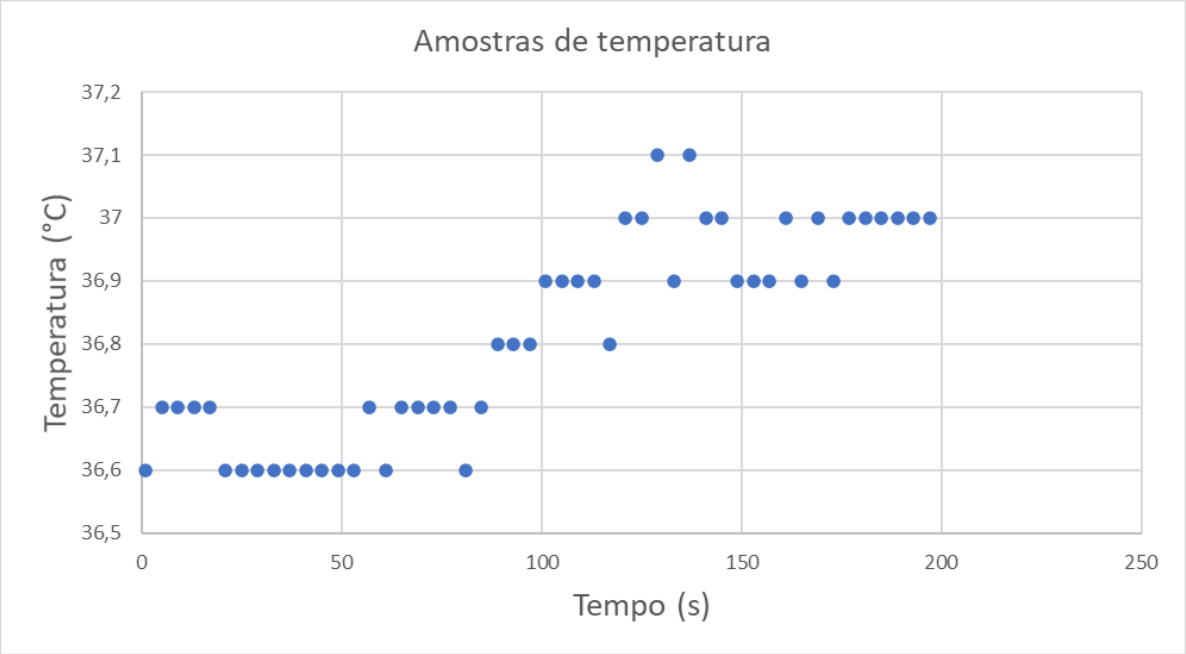
Os resultados da calibração do sensor de temperatura foram satisfatórios, pois nos testes realizados, o valor calculado pelo microcontrolador sempre apresenta um valor próximo ao termômetro clínico, com a maioria dos testes resultando em uma diferença nula ou de 1 ou 2 décimos de grau.

Figura 37: Comparação medição de temperatura do protótipo e do termômetro clínico.

Fonte: O autor, 2021.

Para melhor análise, foi realizado 50 leituras seguidas no pulso do autor com intervalo de 4 segundos entre cada leitura, resultando no seguinte gráfico:

Figura 38: Amostragem de temperaturas



Fonte: O autor, 2021.

No início, no meio e ao final do experimento, foi realizada também a medição pelo instrumento de controle, as 3 medições resultaram em um valor de 36,8°C.

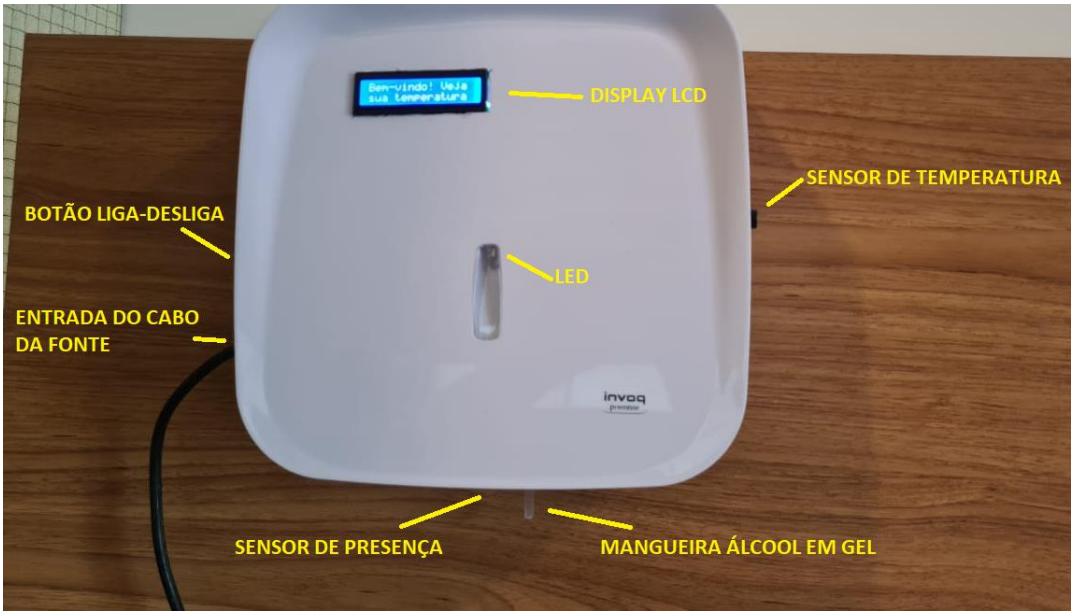
Ao analisar o gráfico, é possível perceber que houve um aumento gradual do valor exibido, isso pode ser explicado pelo fato de que as temperaturas foram medidas sucessivamente e o calor emitido pelo pulso pode ter afetado o sensor, pois ele não possui proteção térmica do meio ambiente, cabendo somente ao algoritmo do fabricante para realizar a compensação.

A média dos valores amostrados é de 36,812 ºC e o desvio padrão é de 0,165 ºC, que foram considerados satisfatórios para o projeto.

4.3 HARDWARE FINAL

O protótipo final foi desenvolvido dentro de um toalheiro, que abrigou todos os elementos de forma compacta, como mostra a figura 39, e de fácil manutenção, como mostra a figura 40.

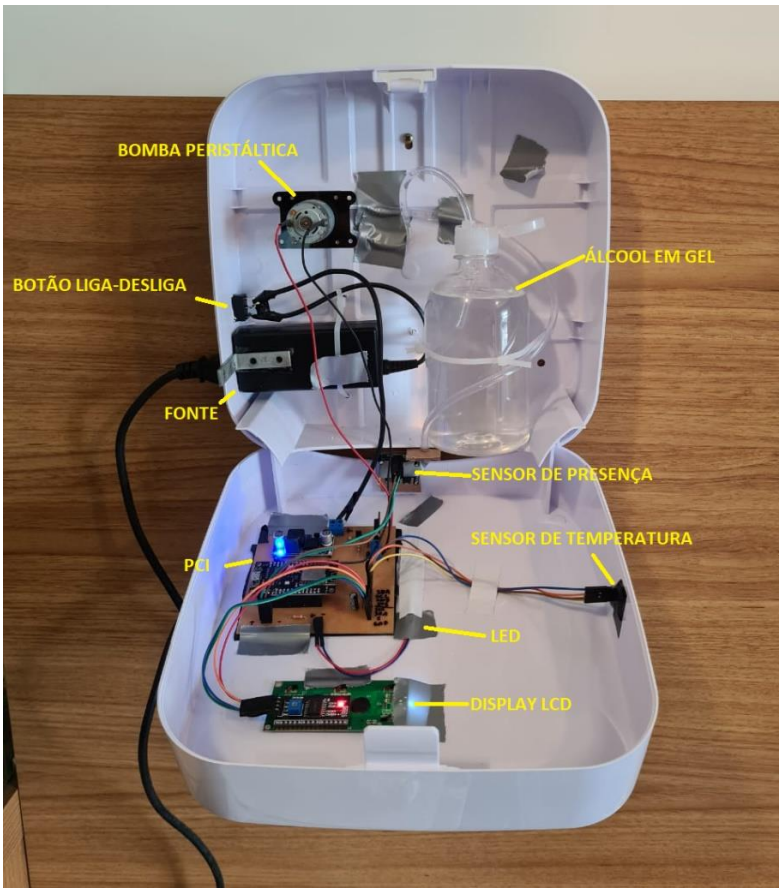
Figura 39: Protótipo fechado e instalado em uma parede por meio de parafusos.



Fonte: O autor, 2021.

O toalheiro possui uma trava na parte de cima que permite a fácil abertura do protótipo.

Figura 40: Protótipo aberto.

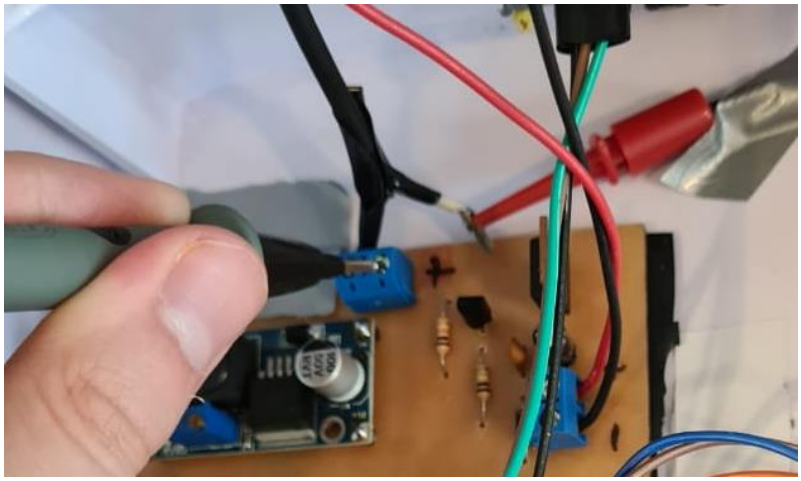


Fonte: O autor, 2021.

4.3.1 CONSUMO ELÉTRICO

Para medir o consumo elétrico do protótipo, foi colocado o multímetro em série com a fonte de energia, como mostra a figura 41:

Figura 41: Medição da corrente do protótipo.



Fonte: O autor, 2021.

Durante o uso normal do equipamento, excetuando-se quando o motor da bomba é ligado, a corrente consumida manteve a média de 41,8mA. Quando o motor é ativado, ela passa a ter a média de 640mA durante o período de acionamento. Ao medir o tempo de uma operação completa, vê-se que o motor é acionado durante 1 segundo entre os 13 segundos totais da operação. Portanto, para calcular o gasto mensal se o aparelho permanecer ligado 24 horas por dia e com uma fonte de eficiência de 80%, é feito o seguinte cálculo:

$$CONSUMO_{kWh} = \frac{POTÊNCIA_W \cdot TEMPO_h}{1000} \qquad (6)$$

$$CONSUMO_{kWh} = \left(\frac{19{,}22 \cdot 41{,}8 \cdot 10^{-3} \cdot 24 \cdot 30}{1000} \cdot \frac{12}{13} \cdot \frac{1}{0{,}8}\right) + \left(\frac{19{,}22 \cdot 640 \cdot 10^{-3} \cdot 24 \cdot 30}{1000} \cdot \frac{1}{13} \cdot \frac{1}{0{,}8}\right) \qquad (7)$$

$$CONSUMO_{kWh} = 1{,}52kWh \qquad (6)$$

Supondo o valor do kWh como sendo de 51 centavos, o valor para a situação de uso ininterrupto do equipamento é de 77,5 centavos mensais, considerado um custo satisfatório para o projeto.

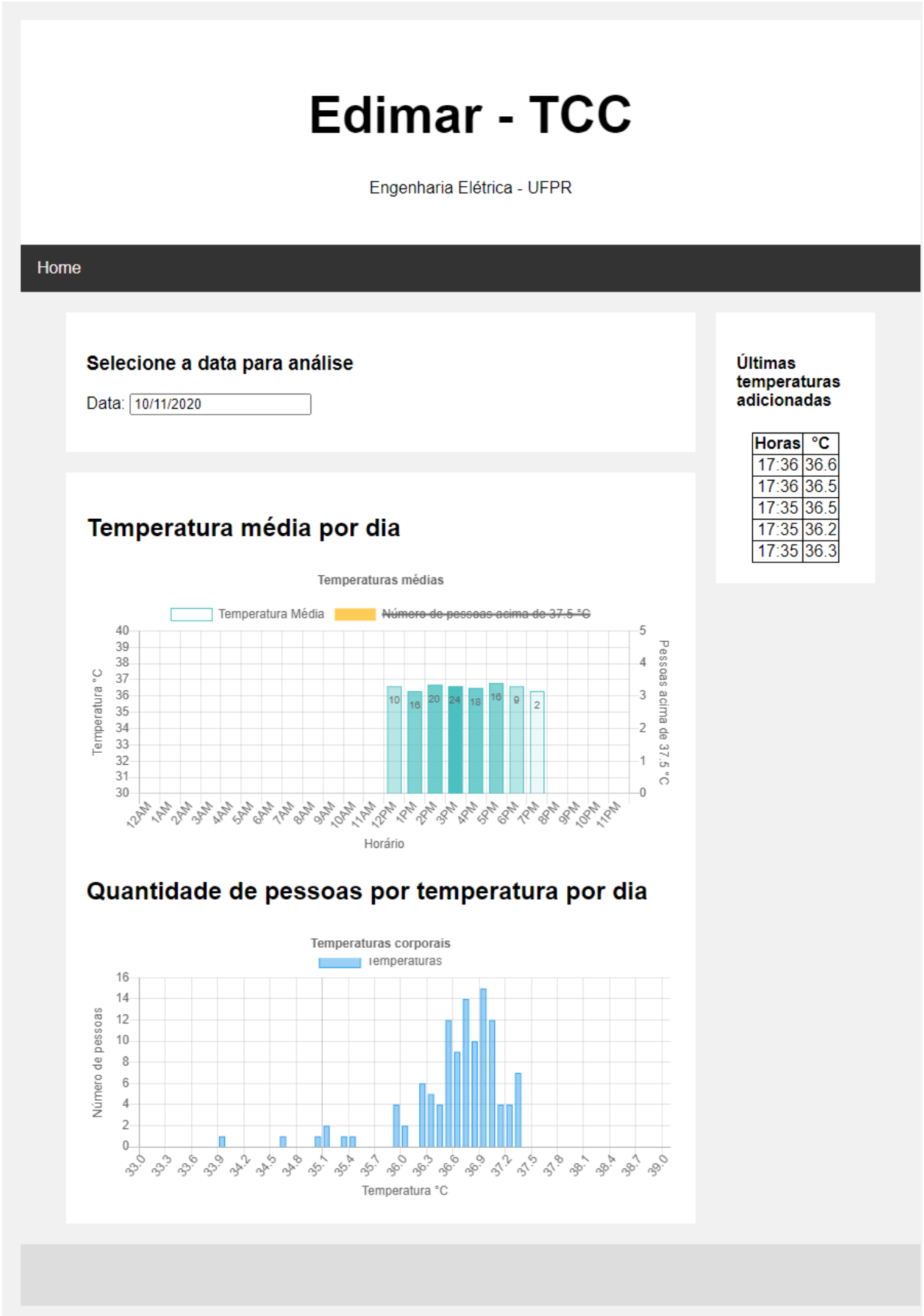
4.4 BANCO DE DADOS

Ao enviar valores ao banco de dados por meio do microcontrolador e ao ler esses por meio da página web, verifica-se que os tempos de acesso e de atualização são muito rápidos, em questão de 1 ou 2 segundos no máximo, o que é satisfatório para o projeto.

4.5 PÁGINA WEB

A primeira parte da página web consiste na seleção da data para separar as temperaturas daquele dia e formar os 2 gráficos desenvolvidos com elas. A próxima parte é uma que foi usada para debug, onde aparecem no canto direito da tela as 5 últimas temperaturas adicionadas.

Figura 42: Webpage construída para o projeto.



Fonte: O autor, 2021.

A parte mais importante são os gráficos. O primeiro mostra qual a distribuição das temperaturas médias ao longo do período de 24 horas do dia selecionado. O eixo horizontal é o horário, dividido de hora em hora; o eixo vertical é a temperatura corporal, com precisão de 0,1ºC; o número e as cores dentro das barras verticais indicam a quantidade de amostras para aquele horário, sendo as cores mais escuras conforme maior o número; o segundo eixo vertical (da direita) mostra a quantidade de pessoas com temperatura acima de 37.5ºC (que é o limiar para febre determinado pelo termômetro clínico em mãos) por meio de barras amarelas, mas no caso não está sendo mostrado pois não houveram tais medidas nesse dia.

O segundo gráfico mostra a quantidade de pessoas para cada temperatura entre 33 e 39 ºC em intervalos de 0,1 ºC para a data selecionada, sendo o eixo vertical a quantidade e o eixo horizontal a temperatura.

5 CONSIDERAÇÕES FINAIS

Esse trabalho foi desenvolvido de modo a criar um sistema que facilite o gestor, de um determinado negócio/comércio, a entender como o fluxo de pessoas dado no ambiente observado está ligado aos casos de coronavírus, por meio da detecção de altas temperaturas corporais. Dessa maneira, diversos conhecimentos foram consolidados, principalmente nas áreas de eletrônica, instrumentação e programação, tratando de assuntos como sensoriamento, protocolos de comunicação, banco de dados e desenvolvimento de aplicativo.

O *hardware* atingiu o objetivo proposto de medir a temperatura corporal com precisão similar a um termômetro clínico infravermelho e realizar a liberação de álcool em gel para higienização do usuário.

O *website* também atingiu seu objetivo, pois os gráficos gerados sobre os dados de temperatura são atualizados em tempo real e são de clara interpretação, para que o gestor tenha facilidade para analisar e tomar decisões sobre seu negócio. Além disso, o *website* também é expansível, por ser facilmente editável e capaz de criar novos gráficos/tabelas/imagens.

5.1 MELHORIAS FUTURAS

a) Criar modelo em software do totem e imprimir em impressora 3D, para obter um melhor acabamento.
b) Calibrar o sensor de temperatura usando um conjunto grande de amostras, em diferentes temperaturas, ambientes e regiões do corpo, para que as medidas sejam as mais fiéis possíveis.
c) Criar nova placa substituindo os módulos por circuitos usando os mesmos componentes e terceirizar a fabricação da PCI.
d) Melhorar conectores e cabos de conexão elétrica.
e) Substituir o pequeno cabo de silicone dentro da bomba peristáltica por um maior, para evitar usar o cabo de PVC que apresentou problemas de conexão.

REFERÊNCIAS

ACCUMED. **Termômetro Clínico Digital Sem Contato.** Accumed Produtos Médico Hospitalares, 2011. Disponível em: < https://accumed.com.br/wp-content/uploads/2018/08/IM_THGTSC1_REV02_130212.pdf >. Acesso em 09 de fev de 2021.

ACTIVA-ID. **Termômetro Digital TD PRO 2 em 1 com dispenser automático de Álcool Gel.** Activa-ID Especialistas em tecnologia de identificação, c2021. Disponível em: < https://www.activashop.com.br/termometro/termometro-k9 >. Acesso em 09 de fev de 2021.

BBC NEWS, Redação. **Coronavírus: OMS declara pandemia.** BBC NEWS BRASIL, 2020. Disponível em: < https://www.bbc.com/portuguese/geral-51842518 > Acesso em: 18 de set de 2020.

BIOMEC. **Termografia.** BioMEC, 2020. Disponível em: < https://biomec.paginas.ufsc.br/?page_id=166 > Acesso em: 10 de jan de 2021.

BRAGA, Newton C. **Microcontroladores.** Instituto Newton C. Braga, s.d. Disponível em: < https://www.newtoncbraga.com.br/index.php/eletronica/52-artigos-diversos/13263-o-basico-sobre-os-microcontroladoresparte-1-mic139 >. Acesso em: 10 de dez de 2020.

COLEMAN, Mike. **Accurate Fever Scanning with Infrared Forehead Thermometers: Issues, Solutions and How to Calibrate.** Fluke, 2020. Disponível em: < https://br.flukecal.com/blog/accurate-fever-scanning-infrared-forehead-thermometers-issues-solutions-and-how-calibrate > Acesso em 23 de out 2020.

ELETRODEX. **Mini Bomba de Água Submersa para Aquário Arduino.** Eletrodex Eletrônica, s.d. Disponível em: < https://www.eletrodex.com.br/mini-bomba-de-agua-submersa-para-aquario-arduino-3-6v.html >. Acesso em: 09 de fev de 2021.

EXPRESSIF. Datasheet: ESP8266EX, 2020. Disponível em: < https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf >. Acesso em 16 de jan de 2021.

FRAZÃO, Dilva. **Biografia de Daniel Gabriel Fahrenheit.** eBiografia, 2019. Disponível em: < https://www.ebiografia.com/daniel_fahrenheit/ > Acesso em: 5 de dez de 2020.

CHEN, G., XIE, J., DAI, G., ZHENG, P., HU, X., LU, H., XU, L., CHEN, Xuequin, CHEN, Xiaomin. **Validity of Wrist and Forehead Temperature in Temperature Screening**

in the General Population During the Outbreak of 2019 Novel Coronavirus: a prospective real-world study. medRxiv, 2020. Disponível em: < https://doi.org/10.1101/2020.03.02.20030148 >. Acesso em 22 de mar de 2021.

GOMES, Pedro César T. **Quais os principais bancos de dados e quais suas diferenças.** OP Services, 2019. Disponível em: < https://www.opservices.com.br/banco-de-dados/ > Acesso em: 10 de jan de 2021.

GOOGLE. **Firebase**, c2021. Página inicial. Disponível em: < https://firebase.google.com/ > Acesso em: 09 de fev de 2021.

HOSPITAL OSWALDO CRUZ. **Muita atenção à febre.** Notícias Hospital Oswaldo Cruz, 2020. Disponível em: < https://www.hospitaloswaldocruz.org.br/imprensa/noticias/muita-atencao-a-febre >. Acesso em: 18 de set de 2020.

INFRAREDMED. **A História da Temperatura Média.** InfraRedMed, 2017. Disponível em: < https://www.infraredmed.com/historia-da-termografia-medica/ > Acesso em 15 de set de 2020.

INSTITUTO NEWTON C. BRAGA. **Sensor piroelétrico de presença (ART590).** Instituto Newton C. Braga, 2008. Disponível em: < https://www.newtoncbraga.com.br/index.php/artigos/54-dicas/4276-art590.html >. Acesso em: 5 de dez de 2020.

INTRUSUL. **Tipos de termômetros e suas utilidades.** Intrusul, instrumentos de medição, 2018. Disponível em: < http://blog.instrusul.com.br/tipos-de-termometros-e-suas-utilidades/ >. Acesso em: 18 de set de 2020.

KOYANAGI, Fernando. **NodeMCU ESP8266: Detalhes e Pinagem.** Fernando K Tecnologia, 2018. Disponível em: < https://www.fernandok.com/2018/05/nodemcu-esp8266-detalhes-e-pinagem.html > Acesso em 17 de nov de 2020.

LARGURA, Ronan. **Display LCD 16x2 com Arduino.** Vida de Silício, 2017. Disponível em: < https://portal.vidadesilicio.com.br/*display-lcd*-16x2-com-arduino/ >. Acesso em 5 de dez de 2020.

LONGEN, Andrei. **O Que é Hospedagem de Site? Guia Para Iniciantes.** Hostinger Tutoriais, 2020. Disponível em: < https://www.hostinger.com.br/tutoriais/o-que-e-hospedagem-de-site#Diferenca-Entre-Hospedagem-de-Sites-e-Dominio > Acesso em 23 de nov de 2020.

LP DO BRASIL. **S20-08 LP-E: Dispenser Automático.** LP do Brasil Exportação e Importação, c2020. Disponível em: < https://lpdobrasil.com.br/produto/dispenser-

para-sabonete-espuma-alcool-espuma-espuma-antisseptica/ >. Acesso em 09 de fev de 2021.

MELEXIS. Datasheet: MLX90614 family, 2019. Disponível em: < https://mel-prd-cdn.azureedge.net/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf >. Acesso em 21 de jan de 2021.

MICROLIFE. **An infrared thermometer and a method for determining a temperature.** European Patent Office, 2006. Disponível em: < https://patents.google.com/patent/EP1680652A1/en >. Acesso em 09 de fev de 2021.

OMEGA Engineering, Inc. **Breve Histórico do Sensor de Temperatura.** Omega, 2015. Disponível em: < https://br.omega.com/artigos-tecnicos/historia-do-sensor-de-temperatura.html >. Acesso em 17 de set de 2020.

ONSEMICONDUCTOR. Datasheet: BC337, BC337-25, BC337-40 Amplifier Transistors, 2013. Disponível em: < https://www.onsemi.com/pub/Collateral/BC337-D.PDF >. Acesso em 16 de jan de 2021.

OPTRIS. **Basic principles of non-contact temperature measurement.** Optris infrared measurements, 2019. Disponível em: < https://www.optris.fr/tl_files/pdf/Downloads/Zubehoer/IR-Basics.pdf > Acesso em: 18 de jan de 2021.

ORACLE. **Banco de dados.** Oracle, 2021. Disponível em: < https://www.oracle.com/br/database/what-is-database/ >. Acesso em 10 de jan 2021.

SERVERSCHECK. **Difference between Body and Skin temperature**, c2021. Disponível em: < http://manuals.serverscheck.com/EST-Difference_between_core_and_skin_temperature.pdf > Acesso em: 22 de mar de 2021.

SPLABOR. **Como funciona uma bomba peristáltica.** SPLabor, equipamentos para laboratórios, 2017. Disponível em: < http://www.splabor.com.br/blog/bombas-peristalticas-2/aprendendo-mais-o-que-e-e-como-funciona-uma-bomba-peristaltica/ >. Acesso em: 15 de jan de 2021.

SUAREZ, F., NOZARIASBMARZ, A., VASHAEE, D. e ÖZTÜRK, M.C. **Designing thermoelectric generators for self-powered wearable electronics.** Energy & Environmental Science, 2016. Disponível em: < https://doi.org/10.1039/C6EE00456C >. Acesso em 16 de dez de 2020.

THARAKAN, S., NOMOTO, K., MIYASHITA, S e ISHIKAWA, K. **Body temperature correlates with mortality in COVID-19 patients.** Critical Care, 2020. Disponível em:

< https://ccforum.biomedcentral.com/articles/10.1186/s13054-020-03045-8#citeas >. Acesso em 18 de set de 2020.

TERMOGRAFIA. *In:* DICIO, Dicionário Online de Português. Portal: 7Graus, 2020. Disponível em: < https://www.dicio.com.br/termografia/ >. Acesso em: 15 de set de 2020.

TUTORIA W3SCHOOLS. **How To – Blog Layout.** W3school.com, 2017. Disponível em: < https://www.w3schools.com/howto/howto_css_blog_*layout*.asp >. Acesso em 1 de nov de 2020.

VISHAY SILICONIX. Datasheet: IRF840, SiHF840 Power *MOSFET*, 2016. Disponível em: < https://www.vishay.com/docs/91070/sihf840.pdf >. Acesso em 16 de jan de 2021.

APÊNDICE 1 – FIRMWARE ESP8266

```
#include <Arduino.h>
#include <LiquidCrystal_I2C.h>
#include <SparkFunMLX90614.h>  //Biblioteca SparkFunMLX90614
#include <Wire.h>              //Biblioteca para I2C
#include "FirebaseESP8266.h"

#define FIREBASE_HOST "teste-54db0.firebaseio.com"
#define FIREBASE_AUTH "WDZLHuUjXwpFJKkvHBOzr0**********"
#define WIFI_SSID "Evolution1403"
#define WIFI_PASSWORD "**********"

#define LED_EXTERN 12  //D6
#define MOTOR_PIN 14   //D5
#define PIR_PIN 13     //D7

IRTherm therm;
LiquidCrystal_I2C lcd(0x27, 16, 2);
FirebaseData firebaseData;
String path = "/measures/PKG";
FirebaseJson json1;
FirebaseJson json2;

const int numberOfReads = 5;
const double presenceTHold = 30;

int pkgCounter = 0;
int readsCounter = 0;
double Tobj = 25;
double Tamb = 25;
int lastTimestamp = 0;

void getLastPkgCounter() {
    if (Firebase.getInt(firebaseData, "/COUNTER"))
        pkgCounter = firebaseData.intData();
}
double calcTbody(double tempMedia) {
    return round((0.0974*tempMedia*tempMedia - 6.038*tempMedia + 129.07)*10)/1
0;

}
bool isHandInPlace(int time) {
    delay(time);
    if (therm.read()) {
        if (therm.object() > presenceTHold)
            return true;
    }
```

```
    return false;
}
bool isTempValidated(double temp) {
    if (temp < 27 || temp > 42)
        return false;
    return true;
}
void readTemp() {
    if (therm.read()) {
        Tobj = therm.object();
        Tamb = therm.ambient();
        //Serial.println("Object: " + String(Tobj, 2));
    }
}
double mediaTemp(int t, int n){
    double soma = 0;
    for(int i = 0;i<n;i++){
        if (therm.read()) {
            soma += therm.object();
            delay(t);
        } else i--;
    }
    return soma/n;
}
int getServerTimeStamp() {  //retorna server time, 0 se erro
    if (Firebase.setTimestamp(firebaseData, "/LastTimeStamp")) {
        int serverTime = firebaseData.intData();
        if (serverTime > 0)
            return serverTime;
    }
    return 0;
}
void saveTempJson(double temperature) {
    double tbody = calcTbody(temperature);
    int timestamp = readsCounter+1000000;
    json1.set(String(timestamp) + "/tempObjeto", Tobj);
    json1.set(String(timestamp) + "/tempAmbiente", Tamb);
    json1.set(String(timestamp) + "/tempBody", tbody);
    readsCounter++;
}
void blinkLed3times() {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(150);
    digitalWrite(LED_BUILTIN, LOW);
    delay(150);
    digitalWrite(LED_BUILTIN, HIGH);
    delay(150);
    digitalWrite(LED_BUILTIN, LOW);
    delay(150);
```

```
    digitalWrite(LED_BUILTIN, HIGH);
    delay(150);
    digitalWrite(LED_BUILTIN, LOW);
}
void checkIfMustSendAndSendJsonToFirebase() {
    if (readsCounter >= numberOfReads) {
        lcd.setCursor(0, 0);
        lcd.print("Enviando ao     ");
        lcd.setCursor(0, 1);
        lcd.print("servidor        ");
        if (Firebase.set(firebaseData, path + String(pkgCounter), json1) && Fi
rebase.setInt(firebaseData, "/COUNTER", pkgCounter) ) {
            lcd.setCursor(0, 0);
            lcd.print("Enviado com     ");
            lcd.setCursor(0, 1);
            lcd.print("sucesso         ");

            json1.clear();
            readsCounter = 0;
            pkgCounter++;
            // blinkLed3times();
        } else {
            // Serial.println("FAILED -
> REASON: " + firebaseData.errorReason());
            lcd.setCursor(0, 0);
            lcd.print(firebaseData.errorReason().substring(0,15));
            lcd.setCursor(0, 1);
            lcd.print(firebaseData.errorReason().substring(15,31));
        }

    }
}
void activateMotor(int tempo) {
    digitalWrite(MOTOR_PIN, LOW);
    delay(tempo);
    digitalWrite(MOTOR_PIN, HIGH);
}
void sendTempAmb() {
    int timestamp = getServerTimeStamp();
    json2.set("/tempAmbiente", Tamb);
    if (Firebase.set(firebaseData, "/ambient/" + String(timestamp), json2)) {
        // Serial.println("PASSED");
    } else {
        // Serial.println("FAILED -> REASON: " + firebaseData.errorReason());
    }
    json2.clear();
}

void setup() {
```

```
    digitalWrite(LED_BUILTIN, LOW);
    digitalWrite(MOTOR_PIN, HIGH);
    digitalWrite(LED_EXTERN, LOW);

    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(MOTOR_PIN, OUTPUT);
    pinMode(LED_EXTERN, OUTPUT);
    pinMode(PIR_PIN, INPUT);

    // Serial.begin(9600);  //Inicializa comunicação serial em 9600 de baud ra
te
    delay(200);
    lcd.init();  // initialize the lcd
    lcd.backlight();
    lcd.setBacklight(HIGH);
    lcd.setCursor(0, 0);
    lcd.print("Conectando Wifi");
    lcd.setCursor(0, 1);
    lcd.print(WIFI_SSID);
    // Serial.println();
    therm.begin();          //Inicializa sensor de temperatura infravermelho
    therm.setUnit(TEMP_C);  //Seleciona temperatura em Celsius
    therm.setEmissivity(0.98);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    // Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        // Serial.print(".");
        delay(300);
    }
    // Serial.println();
    // Serial.print("Connected with IP: ");
    // Serial.println(WiFi.localIP());
    // Serial.println();
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    Firebase.reconnectWiFi(true);
    firebaseData.setBSSLBufferSize(1024, 1024);
    firebaseData.setResponseSize(1024);
    getLastPkgCounter();
    lastTimestamp = getServerTimeStamp();

    lcd.setCursor(0, 0);
    lcd.print("Calibrando      ");
    lcd.setCursor(0, 1);
    lcd.print("sensor presenca ");
    delay(60000);

}
```

```
void loop() {

    //################################
    //##        TESTING SIZE MEMORY##
    //################################
    lcd.setCursor(0, 0);
    lcd.print("INICIANDO TESTE ");
    lcd.setCursor(0, 1);
    lcd.print("DE MEMORIA      ");
    delay(10000);
    lcd.setCursor(0, 0);
    lcd.print("                ");
    lcd.setCursor(0, 1);
    lcd.print("                ");
    // while(readsCounter < 1000){
    // lcd.setCursor(0, 0);
    // lcd.print("c: "  + String(readsCounter));
    // lcd.setCursor(0, 1);
    // lcd.print(String(ESP.getFreeHeap(),DEC));
    // saveTempJson(36.1);
    // delay(250);
// lcd.setCursor(0, 0);
//     lcd.print("criando 43 temp ");
//     lcd.setCursor(0, 1);
//     lcd.print("                ");
//     delay(1000);
//     for(int i = 0; i < 42; i++){
//         saveTempJson(36.1);
//     }

//     lcd.setCursor(0, 0);
//     lcd.print("criado          ");
//     lcd.setCursor(0, 1);
//     lcd.print(String(readsCounter));
//     delay(1000);
//     checkIfMustSendAndSendJsonToFirebase();
//     delay(10000);

    lcd.setCursor(0, 0);
    lcd.print("Bem-vindo! Veja ");
    lcd.setCursor(0, 1);
    lcd.print("sua temperatura");

    // time = millis();
    // if (time - lastTime > 1000 * 60 * 60) {  //1h
    //     sendTempAmb();
    //     lastTime = time;
    // }
```

```
     readTemp();                  //le temp constantemente
    if (Tobj > presenceTHold) {  //se detectar pessoa
        digitalWrite(LED_BUILTIN, LOW);
        lcd.setCursor(0, 0);
        lcd.print("Calculando...   ");
        lcd.setCursor(0, 1);
        lcd.print("                ");
        if (isHandInPlace(2000)) {  //2s para pessoa ajustar mao abaixo do sen
sor
            double tmedia = mediaTemp(100,10);
            if (isTempValidated(tmedia)) {
                lcd.setCursor(0, 0);
                lcd.print("Sua temperatura:");
                lcd.setCursor(0, 1);
                lcd.print("    " + String(calcTbody(tmedia), 1) + " oC    ");
                digitalWrite(LED_EXTERN, HIGH);
                bool isHandInPlace = false;
                long int t = millis();
                while (!isHandInPlace && millis() - t < 10000) {
                    if (digitalRead(PIR_PIN) == HIGH) {
                        isHandInPlace = true;
                        digitalWrite(LED_EXTERN, LOW);
                        lcd.setCursor(0, 0);
                        lcd.print("Liberando alcool");
                        lcd.setCursor(0, 1);
                        lcd.print("                ");
                    }
                    yield();  //reset wdt
                }
                digitalWrite(LED_EXTERN, LOW);
                if (isHandInPlace) {
                    activateMotor(1000);
                    delay(500);
                }

                lcd.setCursor(0, 0);
                lcd.print("Salvando...     ");
                lcd.setCursor(0, 1);
                lcd.print("leitura n: " + String(readsCounter+1) + "/" + Strin
g(numberOfReads));
                saveTempJson(tmedia);
                delay(400);

                checkIfMustSendAndSendJsonToFirebase();
            }
            delay(4000);                    //4s até a proxima leitura
            while (Tobj > presenceTHold) {  //só deixa ler dnv se tirar a mão
                readTemp();
                delay(200);
```

```
            }
        }
    } else {
        digitalWrite(LED_BUILTIN, HIGH);
    }
    delay(200);
}
```

APÊNDICE 2 – CÓDIGO PÁGINA WEB (HTML, CSS, JAVASCRIPT)

```
<!-- https://www.w3schools.com/css/css_website_layout.asp -->
<!DOCTYPE html>
<html>

<head>
  <link rel="shortcut icon" href="#" />
  <link rel="stylesheet" href="css/styles.css"/>
  <link rel="stylesheet" href="https://code.jquery.com/ui/1.9.0/themes/base/jq
uery-ui.css" />
  <script src="https://code.jquery.com/jquery-1.8.2.js"></script>
  <script src="https://code.jquery.com/ui/1.9.0/jquery-ui.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.0.0/firebase-
app.js"></script>
  <script src="https://www.gstatic.com/firebasejs/8.0.0/firebase-
database.js"></script>
  <script src=js/moment.js></script>
  <script src=js/Chart.min.js></script>
  <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-
datalabels@0.7.0"></script>
  <script src=js/utils.js></script>
  <script src=js/firebase.js></script>

</head>

<body>

  <div class="header">
    <h1>Edimar - TCC</h1>
    <p>Engenharia Elétrica - UFPR</p>
  </div>

  <div class="topnav">
    <a href="#">Home</a>
    <!-- <a href="#">Link</a> -->
    <!-- <a href="#">Link</a> -->
    <!-- <a href="#" style="float:right">Link</a> -->
  </div>

  <div class="row">
    <div class="leftcolumn">
      <div class="card">
        <h3>Selecione a data para análise</h3>
          <p>Data: <input type="text" id="calendar"></p>
      </div>
      <div class="card">
        <h2>Temperatura média por dia</h2>
          <div id="canvasgraph2">
```

```
            <canvas id="tmediagraph"></canvas>
          </div>
          <h2>Quantidade de pessoas por temperatura por dia</h2>
          <div id="canvasgraph1">
            <canvas id="bodygraph"></canvas>
          </div>
        </div>
    </div>
    <div class="rightcolumn">
      <div class="card">
        <h4>Últimas temperaturas adicionadas</h4>
        <table>
          <tr>
            <th>Horas</th>
            <th>°C</th>
          </tr>
          <tr>
            <td id="d1">-</td>
            <td id="t1">-</td>
          </tr>
          <tr>
            <td id="d2">-</td>
            <td id="t2">-</td>
          </tr>
          <tr>
            <td id="d3">-</td>
            <td id="t3">-</td>
          </tr>
          <tr>
            <td id="d4">-</td>
            <td id="t4">-</td>
          </tr>
          <tr>
            <td id="d5">-</td>
            <td id="t5">-</td>
          </tr>
        </table>
      </div>
    </div>
  </div>

  <div class="footer">
    <h2></h2>
  </div>
  <script src="js/index.js"></script>
</body>

</html>
```

```
* {
    box-sizing: border-box;
  }

  #cmDPuR5O {
    position: absolute; top: 20%; left: 43%; display: none;
  }

  body {
    font-family: Arial;
    padding: 10px;
    background: #f1f1f1;
  }

  /* Header/Blog Title */
  .header {
    padding: 30px;
    text-align: center;
    background: white;
  }

  .header h1 {
    font-size: 50px;
  }

  /* Style the top navigation bar */
  .topnav {
    overflow: hidden;
    background-color: #333;
  }

  /* Style the topnav links */
  .topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
  }

  /* Change color on hover */
  .topnav a:hover {
    background-color: #ddd;
    color: black;
  }

  /* Create two unequal columns that floats next to each other */
  /* Left column */
```

```
.leftcolumn {
  float: left;
  width: 70%;
  margin-left: 5%;
}

/* Right column */
.rightcolumn {
  float: left;
  width: 20%;
  background-color: #f1f1f1;
  padding-left: 20px;
  margin-right: 5%;
}

/* Fake image */
.fakeimg {
  background-color: #aaa;
  width: 100%;
  padding: 20px;
}

/* Add a card effect for articles */
.card {
  background-color: white;
  padding: 20px;
  margin-top: 20px;
}

/* Clear floats after the columns */
.row:after {
  content: "";
  display: table;
  clear: both;
}

/* Footer */
.footer {
  padding: 20px;
  text-align: center;
  background: #ddd;
  margin-top: 20px;
}

/* Responsive layout - when the screen is less than 800px wide, make the two
columns stack on top of each other instead of next to each other */
 @media screen and (max-width: 800px) {

   .leftcolumn,
```

```
    .rightcolumn {
      width: 100%;
      padding: 0;
    }
  }

  /* Responsive layout - when the screen is less than 400px wide, make the nav
igation links stack on top of each other instead of next to each other */
  @media screen and (max-width: 400px) {
    .topnav a {
      float: none;
      width: 100%;
    }
  }




  table,
  td,
  th {
    border: 1px solid black;
  }

  table {
    margin-left: auto;
    margin-right: auto;
    width: 50%;
    border-collapse: collapse;
    text-align: center;
  }

  canvas{
    -moz-user-select: none;
    -webkit-user-select: none;
    -ms-user-select: none;
  }

  #canvasgraph1 {
    width: 75%;
    margin-left: 12%;
  }

  @media screen and (max-width: 1200px) {
    #canvasgraph1 {
      width: 100%;
      margin-left: 0%;
    }
  }
```

```
console.log('Debugging');

    var dateSelected;
    var datesToShow = [];
    var data = {};
    var dataAmb = {};
    var measuresRef = firebase.database().ref('/measures');
    var ambientRef = firebase.database().ref('/ambient');
    var chartTmedia
    var chartBody
    var chartAmbient

function enableDates(date) {
    return [datesToShow.includes(date.valueOf())];
}

$(function() {
    $("#calendar").datepicker({
        beforeShowDay: enableDates,
        onSelect: function() {
            dateSelected = $(this).datepicker('getDate');
            tMediaPrepareData(dateSelected);
            measuresPrepareData();
        },
        dateFormat: 'dd/mm/yy',
        dayNames: ['Domingo','Segunda','Terça','Quarta','Quinta','Sexta','Sába
do','Domingo'],
        dayNamesMin: ['D','S','T','Q','Q','S','S','D'],
        dayNamesShort: ['Dom','Seg','Ter','Qua','Qui','Sex','Sáb','Dom'],
        monthNames: ['Janeiro','Fevereiro','Março','Abril','Maio','Junho','Jul
ho','Agosto','Setembro','Outubro','Novembro','Dezembro'],
        monthNamesShort: ['Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago','Set
','Out','Nov','Dez']
    });
});

function randn_bm(min, max, skew) {
    let u = 0, v = 0;
    while(u === 0) u = Math.random(); //Converting [0,1) to (0,1)
    while(v === 0) v = Math.random();
    let num = Math.sqrt( -
2.0 * Math.log( u ) ) * Math.cos( 2.0 * Math.PI * v );

    num = num / 10.0 + 0.5; // Translate to 0 -> 1
    if (num > 1 || num < 0) num = randn_bm(min, max, skew); // resample betwee
n 0 and 1 if out of range
    num = Math.pow(num, skew); // Skew
    num *= max - min; // Stretch to fill range
    num += min; // offset to min
```

```
    return num;
}

measuresRef.on('value', function (snapshot) {
    data = snapshot.val();
    measuresPrepareData();
});

ambientRef.on('value', function (snapshot) {
    dataAmb = snapshot.val();
    ambientPrepareData();
});

function ambientPrepareData(){
    let ambientGraphDatapoints = [];
    let offset = -1;
    let i = 0;
    for (let time in dataAmb) {
        var timestamp = parseInt(time) * 1000;
        let number = dataAmb[time].tempAmbiente;
        // let rounded = Math.round(number * 10) / 10;
        let date = moment(timestamp).toDate();

        let now = moment();
        // let teste = moment(1604600864000);
        if (now.diff(date, 'hours') < 24){
            if(offset == -1)
                offset = i;
            ambientGraphDatapoints[i - offset] = {
                x: date,
                y: number
            }
        }
        i++;
    }
    drawAmbientGraph(ambientGraphDatapoints);
}
function measuresPrepareData(){
    let datapoints = [];
    let bodyTemp = [];
    let dayBodyTemp = [];
    let allTS = {};
    let numberOfPKGs = Object.keys(data).length;
    let labels = ['33.0', '33.1', '33.2', '33.3', '33.4', '33.5', '33.6', '33.
7', '33.8', '33.9', '34.0', '34.1', '34.2', '34.3',
        '34.4', '34.5', '34.6', '34.7', '34.8', '34.9', '35.0', '35.1', '35.2'
, '35.3', '35.4', '35.5', '35.6', '35.7', '35.8', '35.9', '36.0',
        '36.1', '36.2', '36.3', '36.4', '36.5', '36.6', '36.7', '36.8', '36.9'
, '37.0', '37.1', '37.2', '37.3', '37.4', '37.5', '37.6', '37.7', '37.8',
```

```
        '37.9', '38.0', '38.1', '38.2', '38.3', '38.4', '38.5', '38.6', '38.7'
, '38.8', '38.9', '39.0'
    ];

    for (let i = 0; i < 61; i++) {
        datapoints[i] = 0;
    }
    for (let i = 0; i < numberOfPKGs; i++) {
        let pkg = 'PKG' + i;
        for (let timestamp in data[pkg]) {
            allTS[timestamp] = data[pkg][timestamp]
        }
    }
    for (let i = 0; i < numberOfPKGs; i++) {
        let pkg = 'PKG' + i;
        for (let timestamp in data[pkg]) {
            let dateaux = moment(parseInt(timestamp)*1000).toDate();
            dateaux.setHours(0,0,0,0)
            if(i==0)
                datesToShow.push(dateaux.valueOf())
            else if (dateaux.valueOf() != datesToShow[datesToShow.length-
1].valueOf()){
                datesToShow.push(dateaux.valueOf())
            }
            let number = data[pkg][timestamp].tempBody;
            let rounded = Math.round(number * 10) / 10;
            bodyTemp.push(rounded);
        }
    }
    if(!dateSelected){
        dateSelected = moment(datesToShow[datesToShow.length-1]).toDate();
    }
    for(let ts in allTS){
        let dateaux = moment(parseInt(ts*1000)).toDate();
        if(dateaux.getDate().valueOf() === dateSelected.getDate().valueOf()){
            let number = allTS[ts].tempBody;
            let rounded = Math.round(number * 10) / 10;
            dayBodyTemp.push(rounded);
        }
    }

    for (let i = 0; i < dayBodyTemp.length; i++) {
        let index = Math.round((dayBodyTemp[i] - 33) * 10);
        datapoints[index]++;
    }
    // for (let i = 0; i < bodyTemp.length; i++) {
    //     let index = Math.round((bodyTemp[i] - 33) * 10);
    //     datapoints[index]++;
    // }
```

```
    for (let i = 0; i < 5; i++) {
        let lastDates = []
        let allTSLength = Object.keys(allTS).length;
        // lastDates[i] = new Date(parseInt(Object.keys(data['PKG'+ (numberOfP
KGs-1)])[i])*1000)
        lastDates[i] = new Date(parseInt(Object.keys(allTS)[allTSLength - 1 -
i])*1000)
        let h = lastDates[i].getHours();
        let m = lastDates[i].getMinutes();
        h = (h<10) ? '0' + h : h;
        m = (m<10) ? '0' + m : m;
        // console.log(h + ':' + m)
        document.getElementById("d"+(i+1)).innerHTML = h + ':' + m;
        document.getElementById("t"+(i+1)).innerHTML = bodyTemp[bodyTemp.lengt
h - (i+1)];
    }

    drawBodyGraph(labels,datapoints);


    if(!dateSelected){
        tMediaPrepareData(datesToShow[datesToShow.length-1]);
    } else {
        tMediaPrepareData(dateSelected);
    }
}
function tMediaPrepareData(dateSelected){
    let datapoints = [];
    let datapointsQtd = [];
    let bodyTempPerHourSum = [];
    let numberOfPKGs = Object.keys(data).length;
    let numberSamples = [];
    for (let i = 0; i < 24; i++) {
        datapoints[i] = 0;
        numberSamples[i] = 0;
        datapointsQtd[i] = 0;
        bodyTempPerHourSum[i] = 0;
    }
    for (let i = 0; i < numberOfPKGs; i++) {
        let pkg = 'PKG' + i;
        for (let timestamp in data[pkg]) {
            let ts = parseInt(timestamp)*1000;
            let dateaux = moment(ts).toDate();
            dateaux.setHours(0,0,0,0)
            if(dateaux.valueOf()===dateSelected.valueOf()){
                let day = moment(ts).toDate();
                numberSamples[day.getHours()]++;
```

```
                bodyTempPerHourSum[day.getHours()] += data[pkg][timestamp].tem
pBody
                if(data[pkg][timestamp].tempBody >=37.5)
                    datapointsQtd[day.getHours()]++
            }
        }
    }
    for (let i = 0; i < 24; i++) {
        if(numberSamples[i]!=0)
            datapoints[i] = Math.round(bodyTempPerHourSum[i]/numberSamples[i]
* 10) / 10
        else
            datapoints[i]= 0
    }
    var color = Chart.helpers.color;
    let max = 1;
    for (let i = 0; i < 24; i++) {
        if(numberSamples[i]>max)
            max = numberSamples[i]
    }
    var coresAlpha = Array(24).fill().map((_, i) => Math.round(numberSamples[i
]/max * 10) / 10);
    let colors = Array(24).fill().map((_, i) => color(window.chartColors.green
).alpha(coresAlpha[i]).rgbString());
    drawTmediaDuringDay(datapoints, numberSamples,colors,datapointsQtd);
}
function drawTmediaDuringDay(datapoints,numberSamples,colors,datapointsQtd){

    let ctx = document.getElementById('tmediagraph').getContext('2d');
    if(chartTmedia){
        chartTmedia.destroy();
    }
    chartTmedia = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: ['12AM','1AM','2AM','3AM','4AM','5AM','6AM','7AM','8AM','9
AM','10AM','11AM','12PM',
            '1PM', '2PM','3PM','4PM','5PM','6PM','7PM','8PM','9PM','10PM','11P
M',],
            datasets: [{
                yAxisID: 'y-axis-1',
                label: 'Temperatura Média',
                backgroundColor: colors,
                borderColor: window.chartColors.green,
                borderWidth: 1,
                data: datapoints,
                datalabels:  {
                    anchor: 'end',
                    align: 'bottom',
```

```
                font:{
                    size: 10
                },
                formatter: function(value, context) {
                    if(numberSamples[context.dataIndex])
                        return numberSamples[context.dataIndex];
                    return null
                }
            }
            },{
                hidden: true,
                label: 'Número de pessoas acima de 37.5 °C',
                backgroundColor: window.chartColors.yellow,
                yAxisID: 'y-axis-2',
                data: datapointsQtd,
                datalabels: {
                    labels: {
                        title: null
                    }
                }
            }]
        },
        options: {
            responsive: true,
            legend: {
                position: 'top',

            },
            title: {
                display: true,
                text: 'Temperaturas médias'
            },
            scales: {
                xAxes: [{
                    display: true,
                    scaleLabel: {
                        labelString: 'Horário',
                        display: true
                    }
                }],
                yAxes: [{
                    id: 'y-axis-1',
                    stacked:false,
                    display: true,
                    scaleLabel: {
                        display: true,
                        labelString: 'Temperatura °C'
                    },
                    ticks: {
```

```
                        beginAtZero: true,
                        // callback: function (value) { if (value % 1 === 0) {
 return value; } },
                        min: 30,
                        suggestedMax: 40,
                    }
                },{
                    type: 'linear', // only linear but allow scale type regist
ration. This allows extensions to exist solely for log scale for instance
                    display: true,
                    scaleLabel: {
                        display: true,
                        labelString: 'Pessoas acima de 37.5 °C'
                    },
                    position: 'right',
                    id: 'y-axis-2',
                    gridLines: {
                        drawOnChartArea: false
                    },
                    ticks: {
                        beginAtZero: true,
                        callback: function (value) { if (value % 1 === 0) { re
turn value; } },
                        suggestedMin: 0,
                        suggestedMax: 5,
                    }
                }]
            }

        }
    });
}
function drawBodyGraph(labels,datapoints) {
    var color = Chart.helpers.color;
    let ctx = document.getElementById('bodygraph').getContext('2d');
    if(chartBody){
        chartBody.destroy();
    }
    chartBody = new Chart(ctx, {
        type: 'bar',
        data: {
                labels: labels,
                datasets: [{
                    label: 'Temperaturas',
                    backgroundColor: color(window.chartColors.blue).alpha(0.5)
.rgbString(),
                    borderColor: window.chartColors.blue,
                    borderWidth: 1,
                    data: datapoints
```

```
                }]
            },
        options: {
            plugins: {
                // Change options for ALL labels of THIS CHART
                datalabels: {
                    labels: {
                        title: null
                    }
                }
            },
            responsive: true,
            legend: {
                position: 'top',
            },
            title: {
                display: true,
                text: 'Temperaturas corporais'
            },
            scales: {
                xAxes: [{
                    display: true,
                    scaleLabel: {
                        labelString: 'Temperatura °C',
                        display: true
                    }
                }],
                yAxes: [{
                    display: true,
                    scaleLabel: {
                        display: true,
                        labelString: 'Número de pessoas'
                    },
                    ticks: {
                        beginAtZero: true,
                        callback: function (value) { if (value % 1 === 0) { re
turn value; } },
                        suggestedMin: 0,
                        suggestedMax: 5,
                    }
                }]
            }
        }
    });
}
function drawAmbientGraph(ambientGraphDatapoints, ) {
    var color = Chart.helpers.color;
    var ctx = document.getElementById('ambgraph').getContext('2d');
    if(chartAmbient){
```

```
        chartAmbient.destroy();
    }
    chartAmbient = new Chart(ctx, {
        type: 'line',
        data: {
            datasets: [{
                label: 'Temperaturas',
                backgroundColor: color(window.chartColors.red).alpha(0.5).rgbS
tring(),
                borderColor: window.chartColors.red,
                fill: false,
                data: ambientGraphDatapoints
            }]
        },
        options: {
            plugins: {
                // Change options for ALL labels of THIS CHART
                datalabels: {
                    labels: {
                        title: null
                    }
                }
            },
            responsive: true,
            title: {
                display: true,
                text: 'Temperatura Ambiente'
            },
            scales: {
                xAxes: [{
                    type: 'time',
                    display: true,
                    scaleLabel: {
                        display: true,
                        labelString: 'Date'
                    },
                    ticks: {
                        major: {
                            fontStyle: 'bold',
                            fontColor: '#FF0000'
                        }
                    }
                }],
                yAxes: [{
                    display: true,
                    scaleLabel: {
                        display: true,
                        labelString: 'Temperatura °C'
                    }
```

```
                }]
            }
        }
    });
}
```