



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

THIAGO LIRA SOUZA SANTOS

RELATÓRIO DE ESTÁGIO SUPERVISIONADO
LACRA - LABORATÓRIO DE CONSTRUÇÕES RURAIS E AMBIÊNCIA

Campina Grande, julho de 2022

THIAGO LIRA SOUZA SANTOS

RELATÓRIO DE ESTÁGIO SUPERVISIONADO
LACRA - LABORATÓRIO DE CONSTRUÇÕES RURAIS E AMBIÊNCIA

*Relatório de Estágio Supervisionado submetido
à Coordenação do Curso de Engenharia
Elétrica da Universidade Federal de Campina
Grande – campus de Campina Grande como
parte dos requisitos necessários para a
obtenção do grau de Bacharel em Ciências no
Domínio da Engenharia Elétrica.*

Área de Concentração: Eletrônica

Orientador:

Edmar Candeia Gurjão, Dr.

Campina Grande, julho de 2022

THIAGO LIRA SOUZA SANTOS

RELATÓRIO DE ESTÁGIO SUPERVISIONADO
LACRA - LABORATÓRIO DE CONSTRUÇÕES RURAIS E AMBIÊNCIA

*Relatório de Estágio Supervisionado submetido
à Coordenação do Curso de Engenharia
Elétrica da Universidade Federal de Campina
Grande – campus de Campina Grande como
parte dos requisitos necessários para a
obtenção do grau de Bacharel em Ciências no
Domínio da Engenharia Elétrica.*

Área de Concentração: Eletrônica

Aprovado em / /

Professor Dr. Jaidilson Jó da Silva
Universidade Federal de Campina Grande
Avaliador

Professor Dr. Edmar Candeia Gurjão
Universidade Federal de Campina Grande - UFCG
Orientador

Dedico este trabalho às minhas filhas Fernanda e Florence pois sem elas eu não teria sido tão resiliente durante esta minha trajetória.

AGRADECIMENTOS

Agradeço a Deus por tudo.

Agradeço aos meus pais pelos ensinamentos de vida, sempre me incentivando a estudar, me ensinando valores e mostrando quais caminhos seguir para ser uma pessoa em busca do bem.

Agradeço a todos que direta e indiretamente contribuíram para a minha longa trajetória nesse curso e na realização desse trabalho.

“The Show Must Go On”

Queen.

RESUMO

Neste relatório são descritas as atividades realizadas pelo estagiário Thiago Lira Souza Santos, graduando em Engenharia Elétrica pela Universidade Federal de Campina Grande (UFCG), durante o estágio no Laboratório de Construções Rurais e Ambiência (LaCRA) da Unidade Acadêmica de Engenharia Agrícola da UFCG, no período de 04 de maio de 2022 a 07 de julho de 2022, totalizando 185 horas. As atividades desenvolvidas tiveram como base as demandas das pesquisas desenvolvidas naquele laboratório, que em maior parte foi no desenvolvimento de sistemas de aquisição de dados. *Dataloggers* foram desenvolvidos para uso com sensores de temperatura, umidade relativa, luminosidade e cor. A leitura dos sensores é realizada por microcontroladores do tipo ESP32 que enviam os dados para uma planilha do *Google Sheet* através da internet para que possam ser armazenados e analisados pelos pesquisadores daquele laboratório.

Palavras-chave: Sistemas de aquisição de dados; Datalogger; Sensores; Microcontrolador.

ABSTRACT

This report describes the activities carried out by the intern Thiago Lira Souza Santos, graduating in Electrical Engineering from the Universidade Federal de Campina Grande (UFCG), during his internship at the Laboratório de Construções Rurais e Ambiente (LaCRA) of the Unidade Acadêmica de Engenharia Agrícola at UFCG, from May 4, 2022 to July 7, 2022, totaling 185 hours. The activities developed were based on the demands of the research developed in that laboratory, which mostly involved the development of data acquisition systems. Dataloggers were developed for use with temperature, relative humidity, brightness and color sensors. The sensors are read by ESP32 microcontrollers that send the data to a Google Sheet via the internet so that they can be stored and analyzed by the researchers of that laboratory.

Keywords: Data acquisition system; Datalogger; Sensor; Microcontroller.

SUMÁRIO

1	Introdução	10
1.1	Objetivos	10
1.2	Organização do Relatório	11
2	Local do Estágio	12
3	Fundamentação Teórica	13
3.1	Sistemas de Aquisição de Dados	13
3.2	Conversor Analógico Digital	13
3.3	Protocolo de Comunicação I2C	14
3.4	Protocolo de Comunicação 1-Wire	16
4	Material Utilizado	18
4.1	Microcontrolador ESP32	18
4.2	Sensor HTU21D	19
4.3	Sensor TCS34725	20
4.4	Sensor DS18B20	21
5	Atividades Desenvolvidas	22
5.1	Demandas do laboratório	22
5.1.1	Pesquisa A – Cultivo de pimenta “biquinho”	22
5.1.2	Pesquisa B – Materiais alternativos para revestimento de paredes	23
5.2	Integração com uma planilha eletrônica do <i>Google Sheet</i>	24
5.2.1	Passo 1 – Criando a planilha	24
5.2.2	Passo 2 – Inserindo script na planilha	27
5.2.3	Implantando como aplicativo <i>web</i>	28
5.3	Desenvolvimento e funcionamento do sistema	32
5.3.1	Ambiente de Desenvolvimento	32
5.3.2	Funcionamento do sistema	32
5.3.3	Códigos e bibliotecas utilizadas	34
5.3.4	Esquema Elétrico	34
5.3.5	Diagrama de conexões	35
5.3.6	Montagem do circuito e teste	37
6	Considerações Finais	41
7	Referências	42
	Anexo A - Código do Datalogger da Pesquisa A:	43
	Anexo B - Código do Datalogger da Pesquisa B:	47

1 INTRODUÇÃO

O presente relatório descreve as atividades desenvolvidas durante o Estágio Supervisionado do aluno Thiago Lira Souza Santos, do curso de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG), no Laboratório de Construções Rurais e Ambiente (LaCRA), sob orientação do professor Edmar Candeia Gurjão e supervisão do professor José Pinheiro Lopes Neto.

A disciplina de Estágio Supervisionado faz parte da grade curricular do curso de Engenharia Elétrica da UFCG e o cumprimento da carga horária mínima de 180 horas na qual o aluno esteve matriculado é requisito para obtenção do grau de bacharel em Engenharia Elétrica.

O estágio teve início no dia 04 de maio de 2022 e teve sua conclusão no dia 07 de julho do mesmo ano, com carga horária de 20 horas semanais, totalizando 185 horas, sendo cumpridas assim as exigências da disciplina.

O programa de estágios é fundamental pois dá oportunidade de o aluno aplicar e consolidar, através das atividades práticas, os conceitos, teorias e conhecimentos adquiridos durante sua trajetória no curso.

1.1 OBJETIVOS

O estágio realizado teve como objetivo principal aplicar os conhecimentos teóricos e práticos adquiridos durante o curso de Engenharia Elétrica, mais especificamente na área de sistemas embarcados e instrumentação eletrônica, servindo como base para o desenvolvimento de sistemas de aquisição de dados.

Para que o objetivo principal pudesse ser realizado, os seguintes objetivos específicos foram tomados:

- Estudo de sensores de temperatura, umidade e luminosidade;
- Implementação de leitura de sensores com microcontrolador;
- Montagem de circuitos com microcontrolador em placa de prototipagem;
- Estudo e implementação de um ambiente de coleta de dados na nuvem.

1.2 ORGANIZAÇÃO DO RELATÓRIO

No Capítulo 2 têm-se a descrição do local de estágio bem como sua localização e as principais atividades desenvolvidas naquele local.

No Capítulo 3 é apresentada a fundamentação teórica necessária para dar embasamento às atividades desenvolvidas durante o estágio, levantando conceitos sobre sistemas de aquisição de dados, conversor analógico-digital e os protocolos de comunicação I2C e 1-Wire.

No Capítulo 4 é descrito o material eletrônico utilizado na montagem dos sistemas de aquisição de dados como o microcontrolador utilizado e os sensores de temperatura, umidade relativa, luminosidade e cor.

No capítulo 5 são relatadas as atividades desenvolvidas no estágio, partindo da identificação da demanda do laboratório até chegar na montagem e teste dos sistemas de aquisição de dados desenvolvidos.

Por fim, no Capítulo 6, são feitas algumas considerações sobre as atividades desenvolvidas durante o período de estágio.

2 LOCAL DO ESTÁGIO

O estágio foi realizado nas dependências do Laboratório de Construções Rurais e Ambiente que faz parte da Unidade Acadêmica de Engenharia Agrícola (UAEA) da UFCG, localizado na Rua Aprígio Veloso, 882, Bairro Universitário, situada na cidade de Campina Grande – PB.

Figura 1 - Laboratório de Construções Rurais e Ambiente (LaCRA)



Fonte: Autoria própria.

As principais atividades desenvolvidas no LaCRA são pesquisas científicas no âmbito das construções rurais e ambiente para esse tipo de construção, bem como o estudo de silos para armazenamento de grãos.

Dentre os diversos campos de pesquisa realizado nesse laboratório, a atividade realizada durante este estágio consistiu no desenvolvimento de dois sistemas de aquisição de dados (*datalogger*), sendo um destes para uso na agricultura, fazendo a aquisição de dados de sensores de temperatura, umidade relativa, cor e luminosidade, e outro *datalogger* para utilização numa pesquisa sobre isolamento térmico de paredes com uso de material alternativo, fazendo a leitura e de doze sensores de temperatura.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 SISTEMAS DE AQUISIÇÃO DE DADOS

Os sistemas de aquisição de dados têm como finalidade medir grandezas físicas (tais como temperatura, umidade, luminosidade etc.) para coletar e documentar valores de forma automática [1].

Sensores de diversos tipos são utilizados para mensurar grandezas relacionadas a fenômenos físicos. Muitas vezes o sinal de saída de alguns sensores precisa passar por uma etapa de condicionamento, sendo amplificados para apresentarem níveis suficientes para sua leitura bem como filtrados para atenuar ruídos indesejáveis que possam estar somados ao sinal original de interesse.

Sistemas digitais baseados em microcontrolador podem ser utilizados para realizar a aquisição e armazenamento desses dados de forma prática, possibilitando inclusive a reprogramação de algoritmos para se adequar às necessidades do usuário.

3.2 CONVERSOR ANALÓGICO DIGITAL

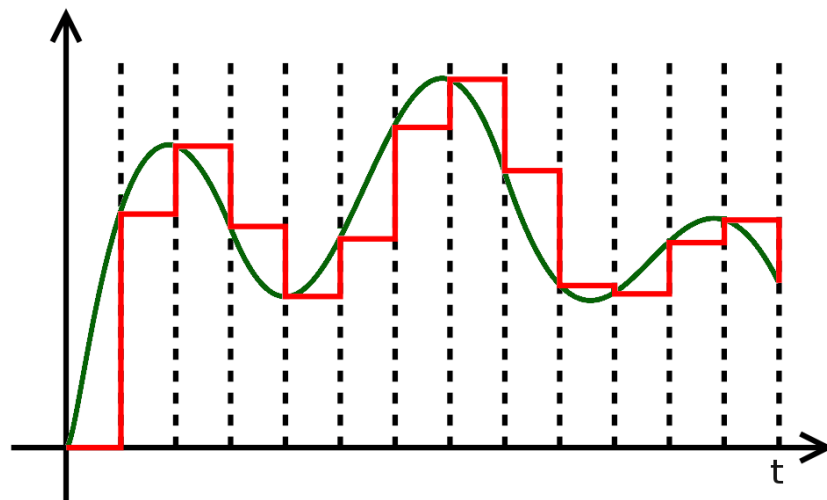
A conversão de um sinal analógico em sua representação digital equivalente é realizada por um Conversor Analógico Digital (ADC, do inglês *Analog to Digital to Converter*) e se faz necessária como uma interface entre os sensores analógicos (transdutores) e o sistema digital.

Os transdutores são dispositivos sensíveis à variação de alguma grandeza física que geralmente apresentam como sinal de saída um sinal elétrico (tensão ou corrente) [2].

Um conversor A/D transforma um sinal analógico, contínuo no tempo, em um sinal amostrado, discreto no tempo, quantizado dentro de um número finito de valores inteiros para sua variação de amplitude determinado pela resolução do conversor em bits [3]. Quanto mais bits de saída tiver o conversor A/D melhor ele vai ser [4].

São diversas as técnicas para realizar a conversão de um sinal analógico para sua representação digital: conversão paralela, rampa simples, rampa dupla, sigma-delta etc. Mais detalhes sobre como são os tipos de conversores A/D podem ser encontrados em [4].

Figura 2 - Representação de um sinal analógico (verde) em sua representação digital equivalente (vermelho)



Fonte: Adaptado de [3].

3.3 PROTOCOLO DE COMUNICAÇÃO I2C

Alguns sensores comerciais podem ser encontrados apresentando seu sinal de saída já na forma digital, como é o caso dos sensores de temperatura e umidade (HTU21D) e o sensor de cores e luminosidade (TCS34725), utilizados na realização desse trabalho.

Desenvolvido pela Philips em meados dos anos 80, o barramento de comunicação serial I2C foi desenvolvido para facilitar a comunicação entre componentes que estivessem montados na mesma placa de circuito [5]. O nome I2C vem do inglês *Inter IC* (IC é a abreviatura para *Integrated Circuit*).

O barramento I2C apresenta algumas características interessantes, dentre elas:

- Uso de apenas dois pinos em seu barramento, SDA (dados) e SCK (*clock*);
- Velocidade de transmissão de dados que vai de 100 kbps no modo *Standard* podendo chegar até 5 Mbps no modo *Ultra Fast*;
- Transmissão de dados de forma serial síncrona;
- Máximos 1008 dispositivos escravos conectados ao mesmo barramento;
- Número ilimitado de dispositivos mestre conectados ao barramento.

O protocolo de comunicação I2C funciona com a transferência de dados por mensagens onde essa é dividida em pedaços de dados.

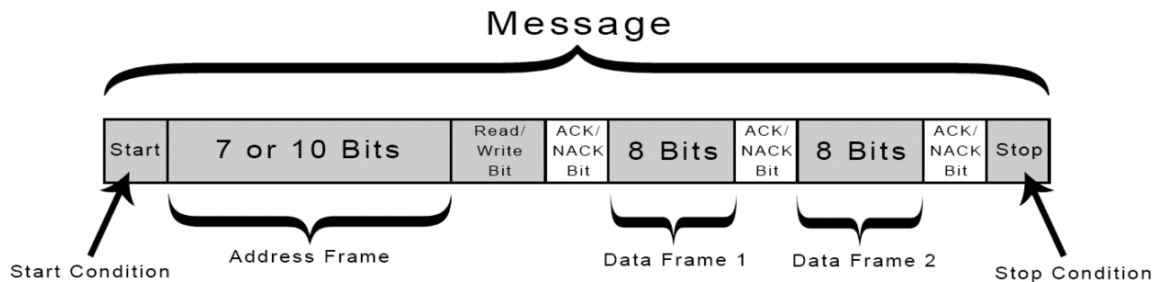
Cada mensagem enviada ou recebida contém o endereço do dispositivo escravo precedido de uma condição que sinaliza o início (*Start*). Após os bits de endereço temos um bit (*Read/Write Bit*)

que sinaliza se o dispositivo mestre está enviando dados ao dispositivo escravo ou se está requisitando dados deste.

Cada pedaço dos dados enviados é seguido por um bit que identifica se os dados foram recebidos ou não com sucesso (ACK/NACK Bit).

Por fim, um bit de parada (*Stop*) é enviado informando que a mensagem chegou ao fim.

Figura 3 - Estrutura da mensagem do protocolo I2S

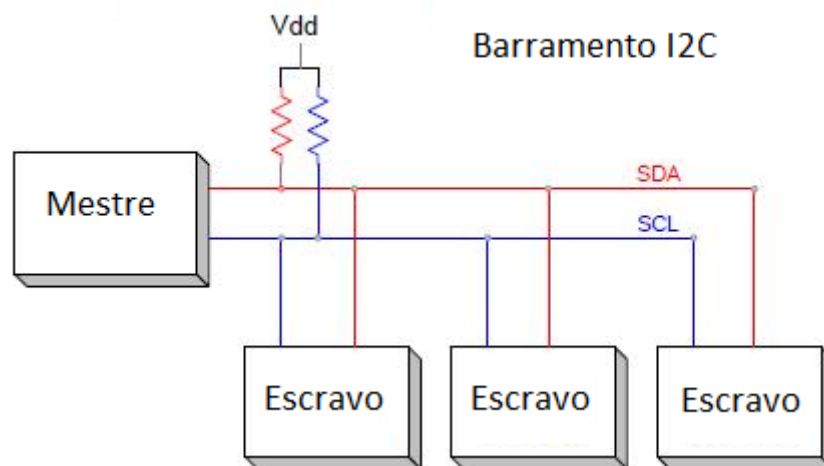


Fonte: [6]

A transmissão de dados via I2C se inicia pelo envio da condição de *Start* pelo dispositivo mestre para todos os dispositivos escravos conectados (nível lógico baixo no pino SDA com o pino SCL inativo).

O dispositivo mestre envia o endereço binário do dispositivo escravo no qual ele irá se comunicar seguido do bit de identificação de leitura ou escrita (*Read/Write Bit*). Cada dispositivo escravo compara o seu próprio endereço com o endereço enviado pelo mestre. Caso o endereço enviado corresponda ao endereço do dispositivo escravo, este último responde com um bit ACK colocando o pino SDA em nível lógico baixo. Ao receber o bit ACK, o dispositivo mestre envia um pedaço dos dados e ao fim do envio de cada pedaço o dispositivo escravo responde com o bit de ACK informando que os dados daquele pedaço foram recebidos. Ao fim do envio da mensagem, o dispositivo mestre sinaliza com a condição de parada (*Stop*) mantendo o pino SCL em nível lógico alto antes do pino SDA também ficar em nível lógico alto [6].

Figura 4 - Barramento I2C típico



Fonte: Adaptado de [7].

3.4 PROTOCOLO DE COMUNICAÇÃO 1-WIRE

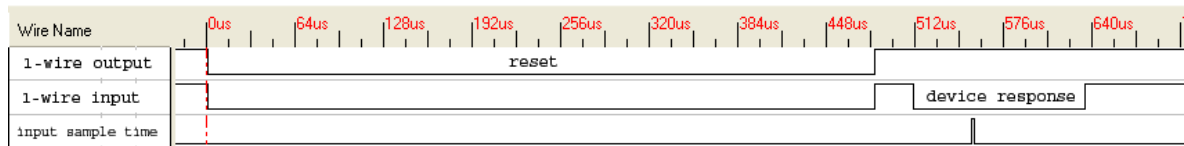
O barramento de comunicação 1-Wire foi desenvolvido pela Dallas Semiconductor Corporation projetado para transmissão de dados em baixa velocidade (16,3 kbit/s) em um único condutor [8]. Outra característica desse barramento é a possibilidade da utilização de apenas dois fios para seu funcionamento (terra e dados). Um único pino para dados de entrada e saída é possível devido à implementação com dreno aberto tanto no dispositivo mestre quanto nos dispositivos escravos e um único resistor de *pull-up* é necessário para todos os dispositivos conectados à rede [9].

Semelhante ao protocolo I2C, o barramento 1-Wire consegue maior alcance na transmissão dos dados mas com menor velocidade [8]. Geralmente é utilizado na comunicação de pequenos dispositivos como o sensor digital de temperatura DS18B20 também produzido pela Dallas Semiconductor Corporation e utilizado em um dos sistemas de aquisição de dados desse trabalho.

Os dispositivos que utilizam do barramento 1-Wire fazem uso de um sistema de endereçamento com uma palavra de identificação (ID) de 64 bits. Cada ID é única e traz consigo informações como o tipo de dispositivo e sua funcionalidade [10].

Para iniciar o uso do barramento é necessário que o dispositivo mestre envie a informação de *reset* colocando o pino de dados em nível lógico baixo por 480 μ s. Em seguida, o dispositivo mestre espera por um pulso em nível lógico baixo de no mínimo 60 μ s enviado pelo dispositivo escravo conectado.

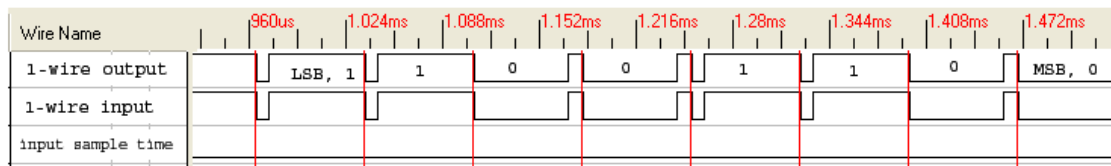
Figura 5 - Procedimento de reset do barramento 1-Wire



Fonte: Adaptado de [9].

Após o reset, o dispositivo mestre envia um comando de 8 bits com o bit menos significativo primeiro (LSB, do inglês *Least Significant Bit*). Existem diversos comandos como comandos de leitura, escrita etc. A cada comando enviado um código de detecção de erro CRC (do inglês *Cyclic Redundancy Check*) de 8 bits é também enviado.

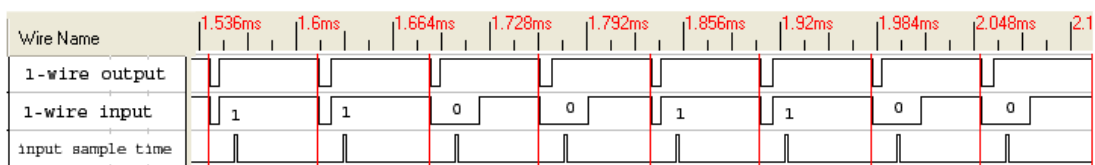
Figura 6 - Envio do comando 00110011 pelo dispositivo mestre



Fonte: Adaptado de [9].

Para leitura dos bits recebidos, o dispositivo mestre realiza amostras periódicas de 8 bits do pino de dados e ao fim de cada conjunto também é realizada a leitura do CRC para detecção de possíveis erros durante a transmissão. Assim como no envio do comando, a organização dos dados é dada com o envio do bit menos significativo primeiro.

Figura 7 - Leitura do byte de dados 00110011



Fonte: Adaptado de [9].

Comandos direcionados a um determinado dispositivo também podem ser enviados. Dessa forma, apenas o dispositivo de interesse é solicitado e somente ele executará os comandos subsequentes.

4 MATERIAL UTILIZADO

Foram desenvolvidos dois sistemas de aquisição de dados utilizando microcontrolador e alguns sensores de temperatura, umidade relativa e sensor de cor e luminosidade.

Durante o processo de escolha dos sensores optou-se por dispositivos com resposta digital, evitando assim circuitos externos de condicionamento de sinais e possíveis calibrações necessárias aos sensores analógicos. Os sensores de temperatura e umidade bem como o de luminosidade e cor utilizam do barramento I2C para comunicação e foram adquiridos em forma de módulos (placa de circuito com os respectivos sensores e outros componentes necessários para o seu funcionamento), facilitando a montagem do hardware do sistema.

4.1 MICROCONTROLADOR ESP32

Foram desenvolvidos dois *dataloggers* e para isso optou-se pelo uso do microcontrolador ESP32 da Espressif. Este dispõe de recursos suficientes para a realização desse trabalho, que é a conexão com a internet por rede sem fio (Wi-Fi), possibilitando enviar os dados para uma planilha na nuvem sem a necessidade de um armazenamento externo para os dados, bem como a disposição do barramento I2C por hardware, facilitando a comunicação com diversos sensores do mercado.

Além de dispor de um barramento I2C e Wi-Fi, o ESP32 apresenta algumas outras características relevantes:

- Processador Dual Core de 32 bit;
- 512 kB de memória RAM;
- Frequência base 160 MHz (máxima de 240 MHz);
- 36 GPIO;
- Hardware de comunicação I2S, I2C, SPI, UART entre outros;
- Wi-Fi 802.11 b/g/n;
- Bluetooth 4.2.

Figura 8 – Microcontrolador ESP32



Fonte: Autoria própria.

4.2 SENSOR HTU21D

Para leitura da temperatura ambiente e umidade relativa do ar foi escolhido o sensor HTU21D desenvolvido pela *Measurement Specialties™* que apresenta as seguintes características relevantes retiradas do seu *datasheet* disponível em [11] :

- Saída digital por barramento I2C;
- Resposta linear para temperaturas na faixa de 5 a 60 °C;
- Resposta linear para valores de umidade relativa entre 20 e 80 por cento;
- Resolução de até 14 bits para valores de temperatura;
- Resolução de até 12 bits para valores de umidade relativa;
- Tensão de alimentação típica de 3,3 volts;
- Consumo de energia típico de 2,7 μ W.

Figura 9 - Módulo sensor de temperatura e umidade relativa HTU21D



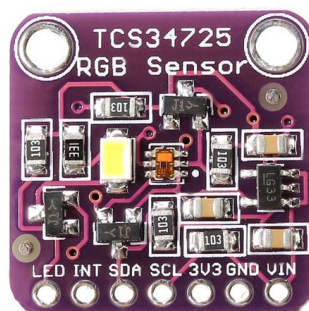
Fonte: [12]

4.3 SENSOR TCS34725

O sensor de cor e luminosidade utilizado em um dos sistemas de aquisição de dados foi o TCS34725 desenvolvido pela *Texas Advanced Optoelectronic Solutions - TAOS®*. Este sensor retorna valores digitais para os níveis de luminosidade para as cores vermelho, verde, azul e branco e apresenta um filtro bloqueador para a faixa infra-vermelha. Este sensor apresenta as seguintes características relevantes retiradas do seu *datasheet* disponível em [13]:

- Saída digital por barramento I2C;
- Sensores de luz vermelha, verde, azul e branca com filtro de infra-vermelho;
- Ganho programável;
- Tensão de alimentação típica de 3 volts;
- Corrente de consumo máxima de 330 μ A.

Figura 10 - Módulo sensor de cor e luminosidade TCS34725



Fonte: [14]

4.4 SENSOR DS18B20

Para leitura das temperaturas em um dos sistemas foram utilizados doze sensores DS18B20. Produzido pela *Maxim Integrated™*, esse sensor é um termômetro digital com resolução configurável de 9 a 12 bits. Sua comunicação é feita através do protocolo 1-Wire que requer de apenas um pino para transmissão dos dados. O mesmo também pode funcionar no modo de alimentação parasita, utilizando apenas o pino de dados e o pino de referência (GND), eliminando a necessidade de uma fonte de alimentação externa adicional. Cada dispositivo apresenta um código de série único de 64 bits para sua identificação, permitindo assim o uso de múltiplos sensores no mesmo barramento 1-Wire [15].

O modelo utilizado foi do tipo encapsulado em tubo metálico e à prova d'água pois é o que apresenta maior resistência mecânica para o manuseio.

Figura 11 - Sensor de temperatura digital DS18B20



Fonte: [16].

A seguir podemos ver algumas características retiradas do *datasheet* desse sensor em [17]:

- Faixa de tensão de alimentação de +3,0 a +5,5 volts;
- Corrente de consumo máxima de 1,5mA durante o uso;
- Erro de $\pm 1^{\circ}\text{C}$ para valores de temperatura entre -30°C e $+100^{\circ}\text{C}$;
- Resolução de temperatura configurável de 9 a 12 bits;
- Uso de apenas 1 pino para comunicação através do barramento 1-Wire.

5 ATIVIDADES DESENVOLVIDAS

5.1 DEMANDAS DO LABORATÓRIO

Será feita uma breve descrição sobre as pesquisas que foram atendidas pelas atividades do estágio, uma voltada para o cultivo de uma planta e outra voltada para construção. Para diferenciá-las no texto, a pesquisa voltada para o cultivo de uma planta será chamada de Pesquisa A enquanto que a pesquisa com foco na construção será chamada de Pesquisa B.

5.1.1 PESQUISA A – CULTIVO DE PIMENTA “BIQUINHO”

Durante o período do estágio, estavam sendo realizadas duas pesquisas que necessitavam de sistemas de aquisição de dados para diversos sensores.

Uma das pesquisas é sobre o cultivo de pimenta do tipo “biquinho” onde há a necessidade de mensurar grandezas como temperatura ambiente, umidade relativa do ar e intensidade luminosa do ambiente do cultivo. Quatro diferentes ambientes foram montados na área externa do laboratório de modo que pudessem ser avaliados os resultados do cultivo daquela espécie em diferentes condições de ambiência.

Figura 12 - Ambientes preparados para o cultivo

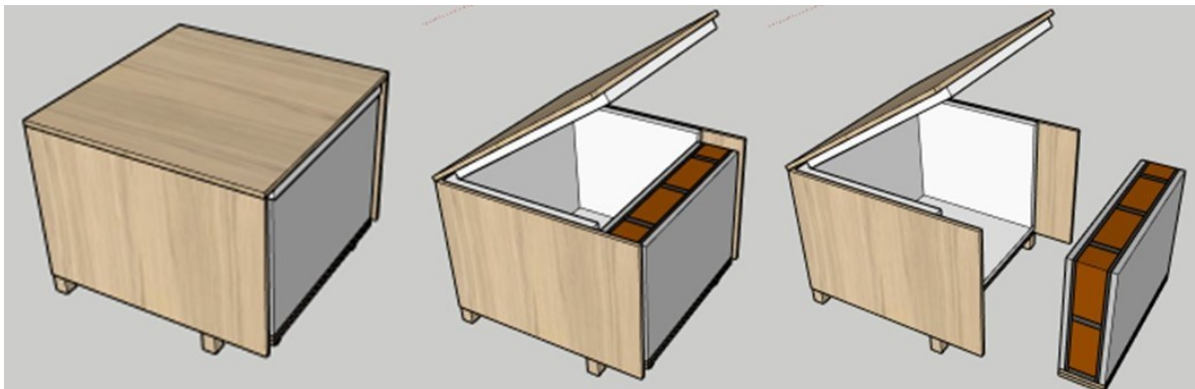


Fonte: Autoria própria.

5.1.2 PESQUISA B – MATERIAIS ALTERNATIVOS PARA REVESTIMENTO DE PAREDES

A outra pesquisa que está em desenvolvimento naquele laboratório é sobre o estudo de materiais alternativos na construção de edificações. Uma caixa térmica para realização de estudos de transferência de calor em paredes de alvenaria estava sendo desenvolvida. A mesma foi produzida em mdf e revestida internamente com placas de poliestireno para uma melhor vedação. Em uma de suas faces haverá uma abertura para entrada/saída do painel de alvenaria (simulando uma parede), tornando a caixa fixa e os painéis móveis.

Figura 13 - Modelo da caixa térmica



Fonte: LaCRA.

Uma fonte radiante de calor, composta por seis lâmpadas incandescentes de 100W cada, estará presente no espaço interno da caixa que representa o ambiente externo da edificação.

Para ambas as pesquisas foram desenvolvidos sistemas de aquisição de dados para os sensores utilizado.

O sistema de aquisição de dados da Pesquisa A foi desenvolvido para leitura de um sensor do tipo HTU21D (temperatura e umidade) e outro do tipo TCS34725 (luminosidade e cor). Já o sistema desenvolvido para a Pesquisa B é composto por doze sensores de temperatura DS18B20 no total. Ambos os sistemas foram desenvolvidos utilizando um microcontrolador do tipo ESP32 e fazem a leitura dos sensores e coleta dos dados a cada 10 minutos.

5.2 INTEGRAÇÃO COM UMA PLANILHA ELETRÔNICA DO *GOOGLE SHEET*

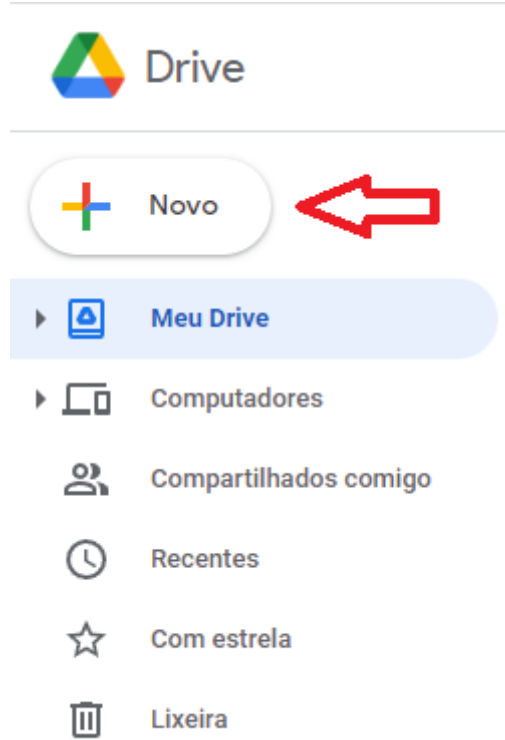
Para armazenar os dados dos sensores optou-se pelo uso de uma planilha eletrônica na nuvem. A escolha do *Google Sheet* se deu por ser um serviço gratuito e de fácil integração ao projeto.

O procedimento para integrar o *Google Sheet* para receber dados e armazená-los na planilha foi simples e seguiu os mesmos passos para os dois sistemas de aquisição de dados desenvolvidos. Dito isto, será demonstrado a configuração de apenas uma das planilhas, pois a outra seguiu o mesmo procedimento mudando apenas os links das mesmas.

5.2.1 PASSO 1 – CRIANDO A PLANILHA

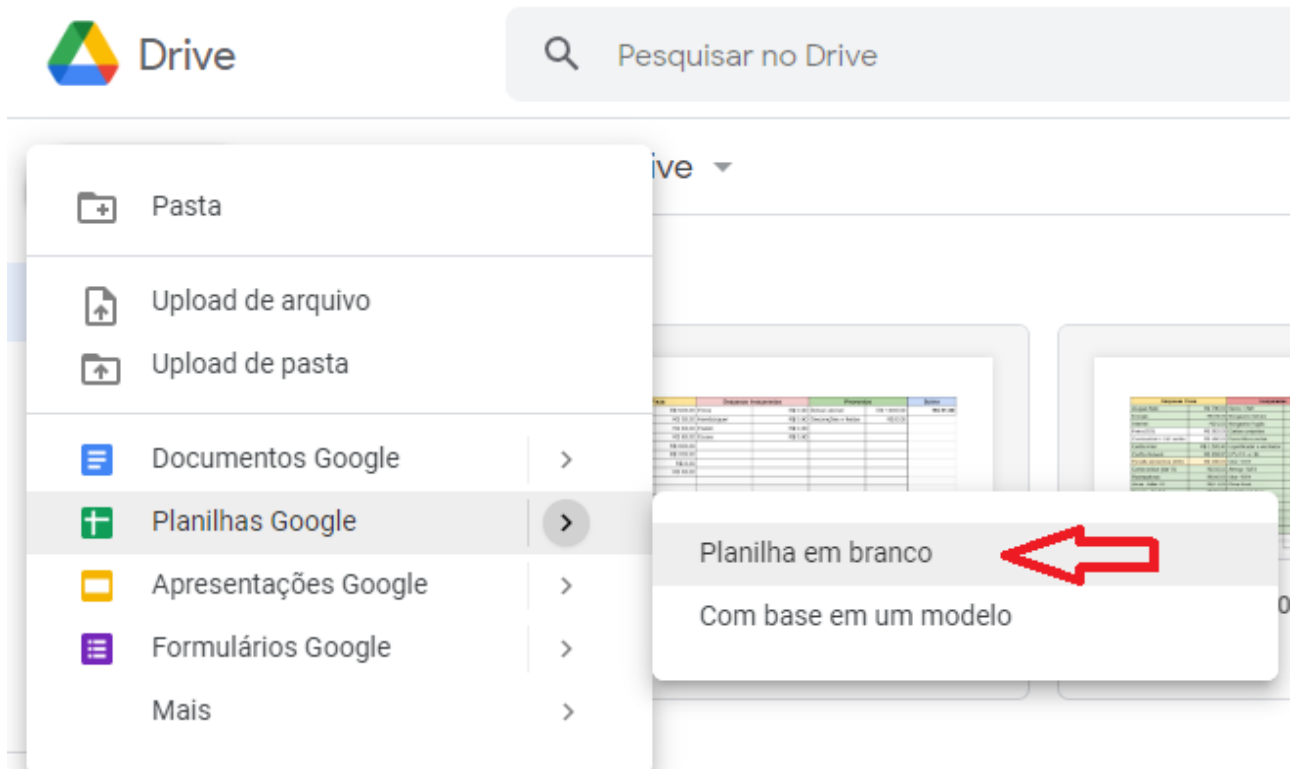
Primeiramente, o usuário da planilha precisa estar logado com sua conta do *Google* e clicar no ícone do *Google Drive*. Estando logado, o usuário deve clicar no botão “Novo” como ilustrado:

Figura 14 - Criando uma nova planilha no *Google Drive* (1)



Fonte: Autoria própria.

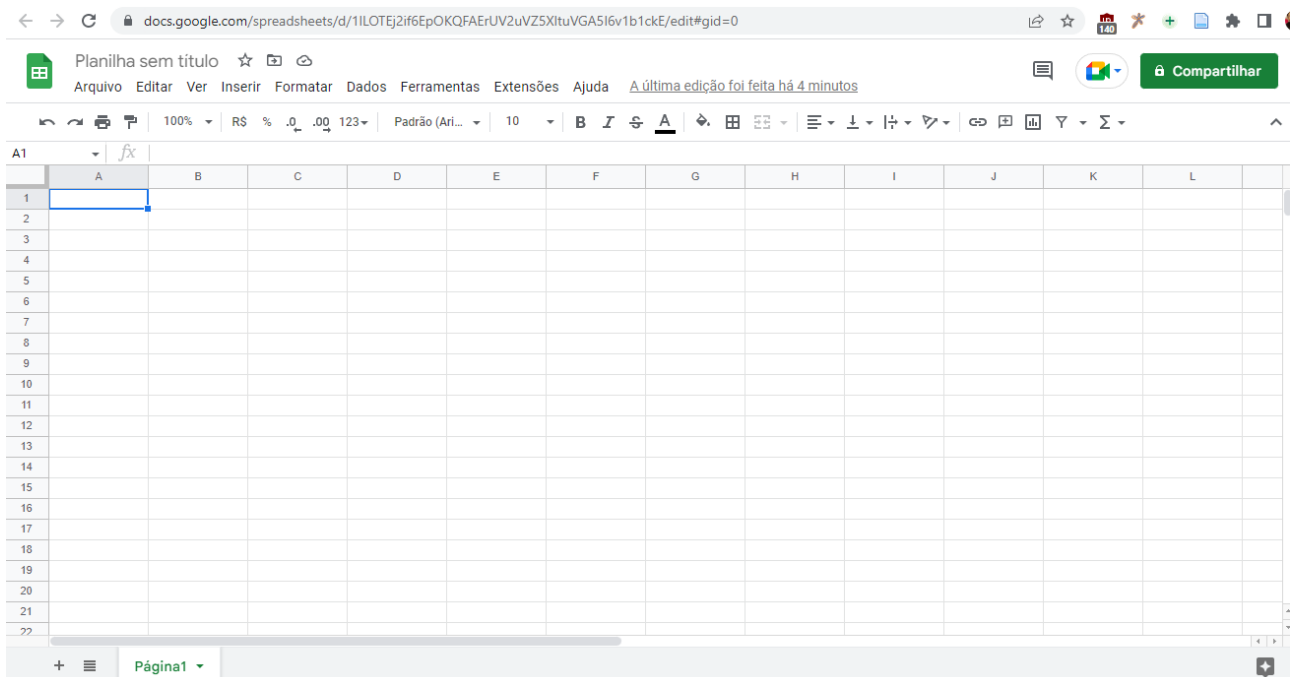
Ao clicar no botão novo, algumas opções irão surgir na tela. Em “Planilhas Google”, clicar em “Planilha em branco”.

Figura 15 - Criando uma nova planilha no *Google Drive* (2)

Fonte: Autoria própria.

Feito isso, uma nova planilha em branco é gerada.

Figura 16 - Planilha em branco gerada



Fonte: Autoria própria.

5.2.2 PASSO 2 – INSERINDO SCRIPT NA PLANILHA

Uma vez gerada a planilha, é necessário inserir um *script* que fará a planilha receber dados através de um link na internet.

Copie o link identificador da planilha:

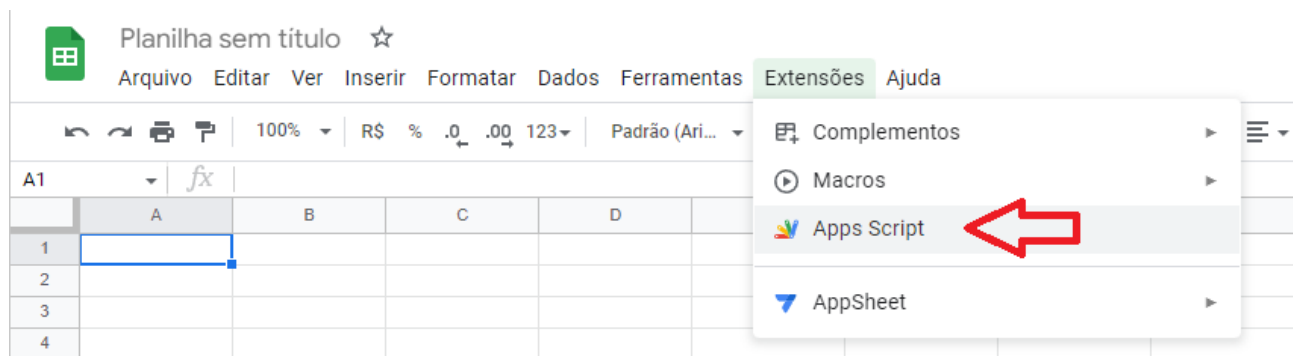
Figura 17 - Identificador da planilha



Fonte: Autoria própria.

Em seguida, clique em “*Apps Script*” no menu “Extensões”.

Figura 18 - Criando um *script* (1)



Fonte: Autoria própria.

Uma nova aba irá abrir no navegador onde será possível digitar o *script* desejado.

Foi utilizado um *script* modelo disponível em [18] e adaptado para as necessidades do projeto. Os *scripts* completos estão disponíveis nos Anexos.

Com a aba do editor de *scripts* aberta, modifique o endereço identificador para o da planilha que irá receber os dados:

Figura 19 - *Script* com identificador da planilha

Fonte: Autoria própria.

5.2.3 IMPLANTANDO COMO APLICATIVO WEB

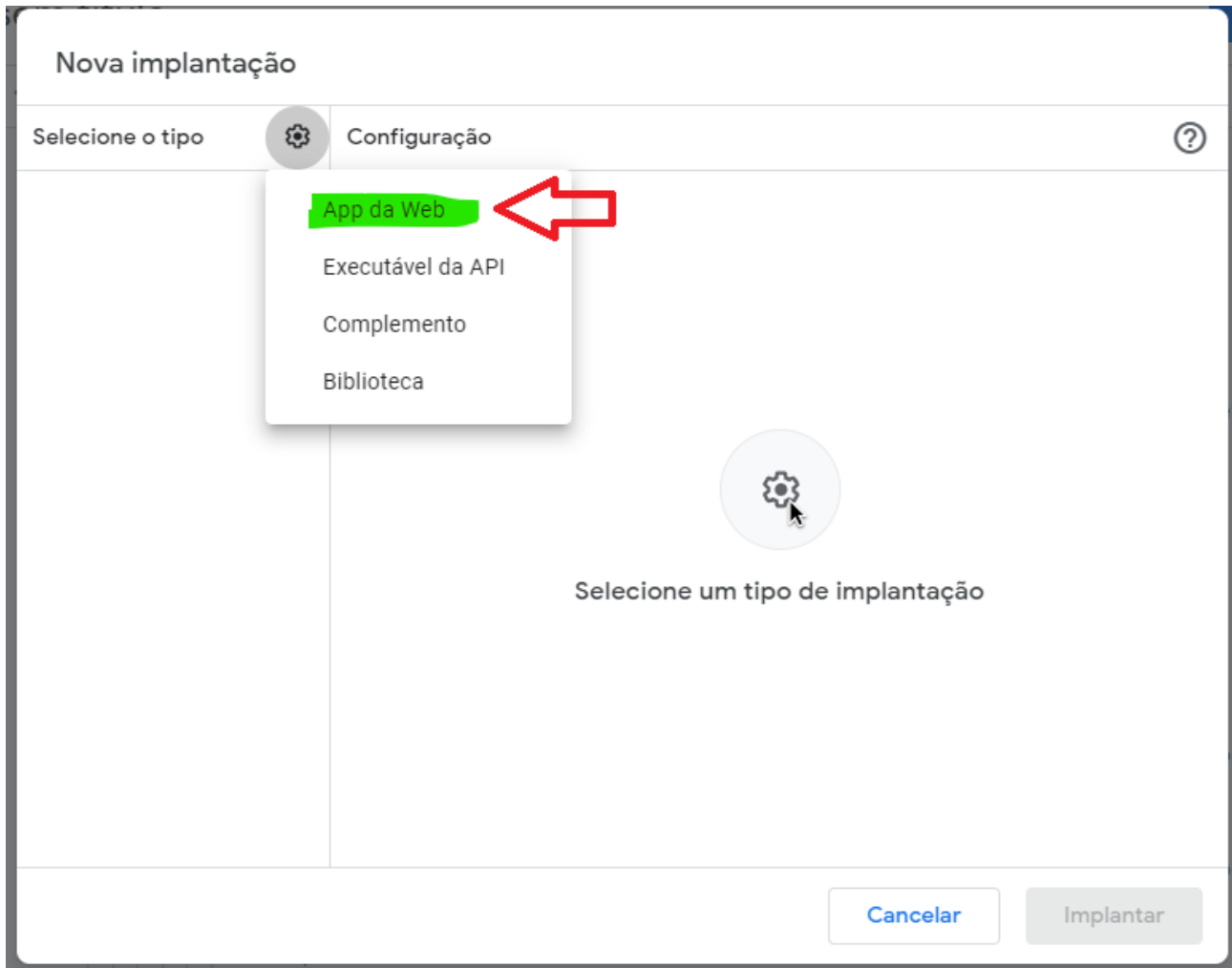
Com o *script* feito, é hora de implantar e tornar a planilha acessível como aplicativo *web*. Clique no botão “Implantar” e em seguida “Nova implantação”:

Figura 20 – Implantação como aplicativo *web* (1)

Fonte: Autoria própria.

Uma nova janela irá se abrir. Clique no botão para selecionar o tipo “App da Web”:

Figura 21 - Implantação como aplicativo web (2)





Fonte: Autoria própria.

Modifique o campo “Quem pode acessar” para “Qualquer pessoa” e em seguida clique em “Implantar”:

Figura 22 - Implantação como aplicativo web (3)

Nova implantação

Selecione o tipo  Configuração 

App da Web

Descrição

Nova descrição

App da Web

Executar como

Eu (thiagolirasouzasantos@gmail.com) ▼

O app da Web será autorizado a usar os dados da sua conta.

Quem pode acessar

Qualquer pessoa ▼

Isto também pode ser usado como uma biblioteca. [Saiba mais](#)

Cancelar Implantar

Fonte: Autoria própria.

Realize a permissão de acesso e por fim copie o link da URL da aplicação gerada:

Figura 23 – Implantação concluída e link de aplicação gerado


Nova implantação

Implantação atualizada.

Versão 1 em 5 de jul., 15:16

Código de implantação


AKfycbwO6yUT5gFxzLd19w56FLOOD836jCXI2smAdGJTqNhxgalmJ7phz0E2p3iyzfyhRBBDOQ


 Copiar

App da Web

URL

<https://script.google.com/macros/s/AKfycbwO6yUT5gFxzLd19w56FLOOD836jCXI2smAdGJTqNhxgalmJ7phz0E2p3iyzfyhRBBDOQ>

 Copiar



Concluído

Fonte: Autoria própria.

5.3 DESENVOLVIMENTO E FUNCIONAMENTO DO SISTEMA

5.3.1 AMBIENTE DE DESENVOLVIMENTO

Para o desenvolvimento do código do sistema foi utilizado o software Visual Studio Code (VS Code) da Microsoft. Este possui inúmeras ferramentas e extensões que tornam o desenvolvimento mais rápido, indo do complemento automático de funções à diferenciação e destaque por cores no texto [19].

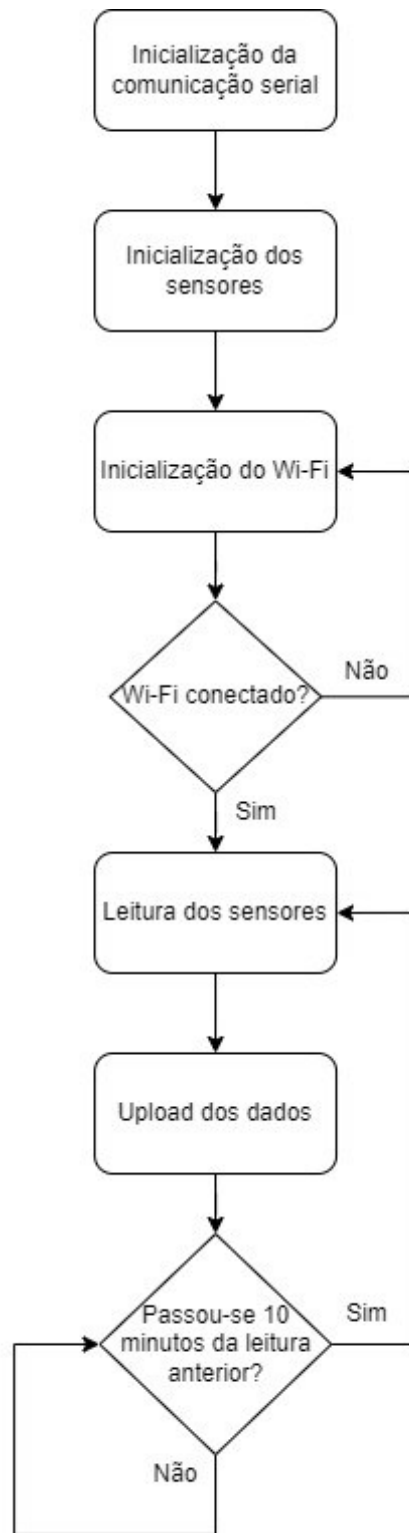
A linguagem de programação adotada para o desenvolvimento foi a do Arduino por sua vasta popularidade e comunidade de apoio, além de dispor das bibliotecas de funções utilizadas para comunicação com os sensores utilizados nesses projetos. A linguagem Arduino é baseada em C++, o que dá suporte à programação orientada à objetos. Vale ressaltar que a própria Espressif (fabricante do ESP32) tem suporte à linguagem Arduino, portando várias bibliotecas do seu framework nativo (ESP-IDF) para esta linguagem [19].

Para fazer a integração do Arduino com o VS Code utilizou-se da extensão PlatformIO que reúne vários frameworks de diversos fabricantes de microcontroladores e dispositivos de sistemas embarcados [19]. O PlatformIO é uma extensão gratuita e está disponível para baixar no próprio VS Code.

5.3.2 FUNCIONAMENTO DO SISTEMA

Um fluxograma foi desenvolvido para melhor entendimento do funcionamento dos sistemas de aquisição de dados bem como para servir de ferramenta de apoio para o desenvolvimento dos códigos. Por terem funcionamentos semelhantes, o mesmo fluxograma é válido para os dois sistemas de aquisição de dados.

Figura 24 - Fluxograma de funcionamento dos sistemas



Fonte: Autoria própria.

Ao ser energizado, o microcontrolador realiza inicialização da comunicação serial, dos sensores e da conexão com a internet por meio de uma rede sem fio Wi-Fi. O mesmo ficará tentando se conectar à internet até que a conexão seja estabelecida.

Uma vez conectado à internet, o microcontrolador realiza a leitura dos sensores e em seguida envia os dados coletados para uma planilha eletrônica do *Google Sheet* onde ficarão armazenados e disponíveis para consulta e análise. A leitura dos sensores e o envio dos dados são feitos a cada 10 minutos.

5.3.3 CÓDIGOS E BIBLIOTECAS UTILIZADAS

A escolha do *framework* Arduino se deu pela vasta disponibilidade de códigos, funções e bibliotecas já desenvolvidas pela comunidade. O próprio ambiente de desenvolvimento integrado (IDE do inglês, *Integrated Development Environment*) do Arduino apresenta um recurso de busca de bibliotecas de funções no qual foi possível encontrar bibliotecas para uso dos sensores utilizados. As bibliotecas utilizadas foram:

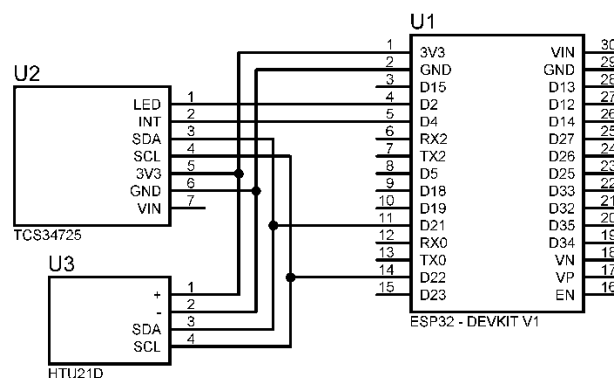
- SparkFunHTU21D.h – utilizada para o sensor de temperatura e umidade relativa HTU21D;
- Adafruit_TCS34725.h – utilizada para o sensor de cor e luminosidade TCS34725;
- DallasTemperature.h – utilizado para o sensor de temperatura DS18B20.

Os códigos completos desenvolvidos para os dois sistemas de aquisição de dados podem ser encontrados no Anexo desse relatório.

5.3.4 ESQUEMA ELÉTRICO

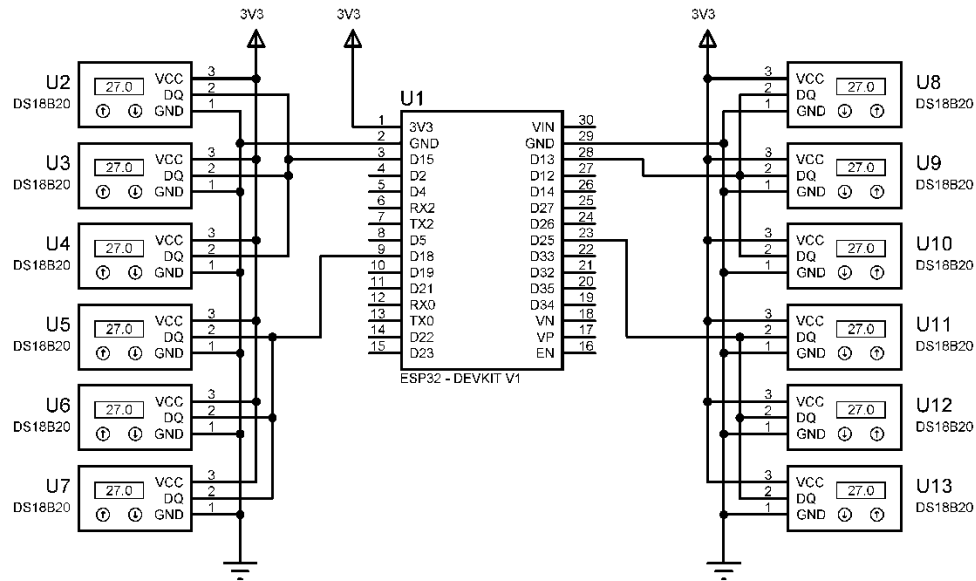
Os esquemas elétricos foram desenvolvidos utilizando o software Proteus 8.6 e serviram para documentação dos projetos.

Figura 25 - Esquema elétrico do datalogger da pesquisa A



Fonte: Autoria própria.

Figura 26 - Esquema elétrico do datalogger da pesquisa B

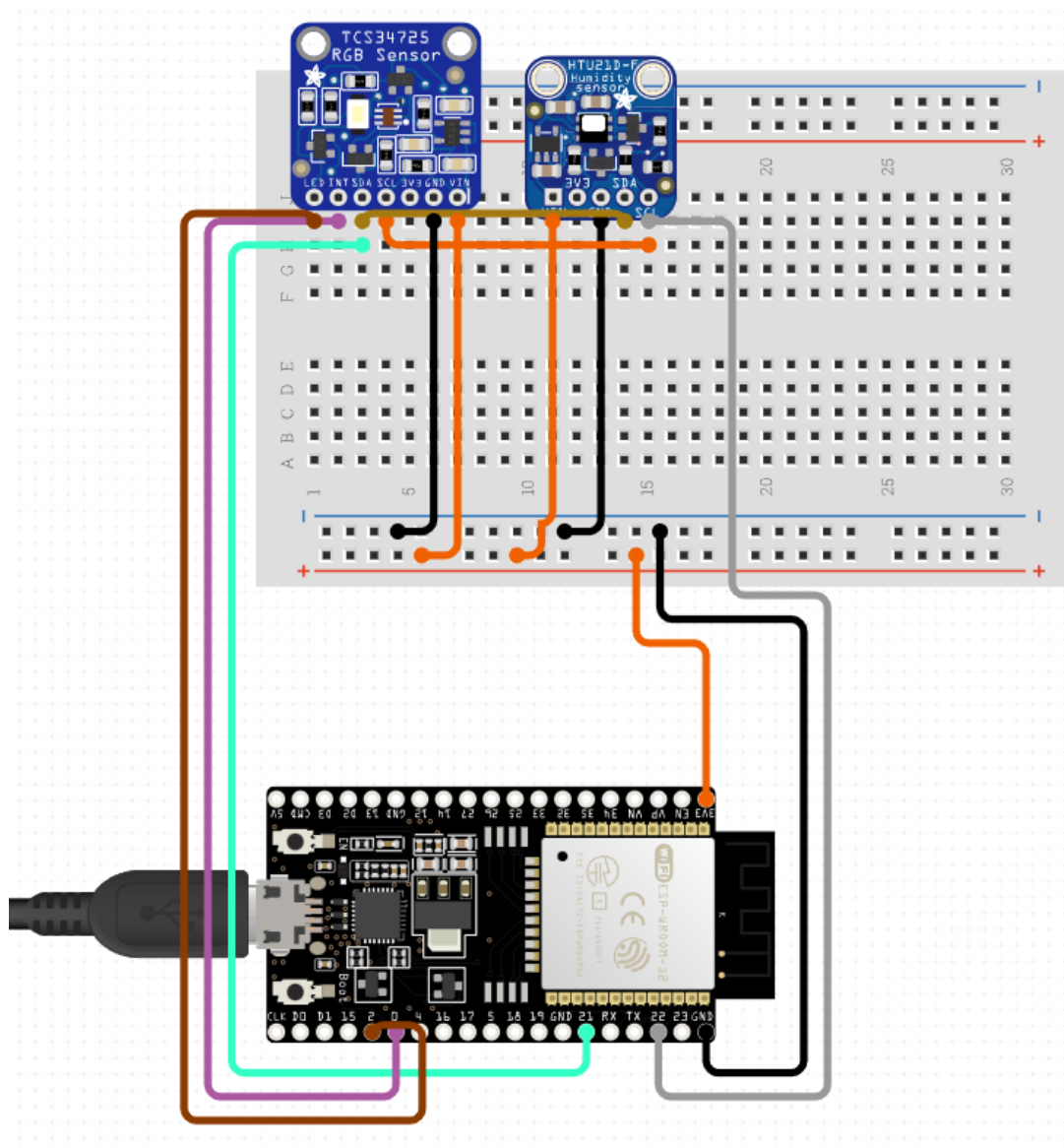


Fonte: Autoria própria.

5.3.5 DIAGRAMA DE CONEXÕES

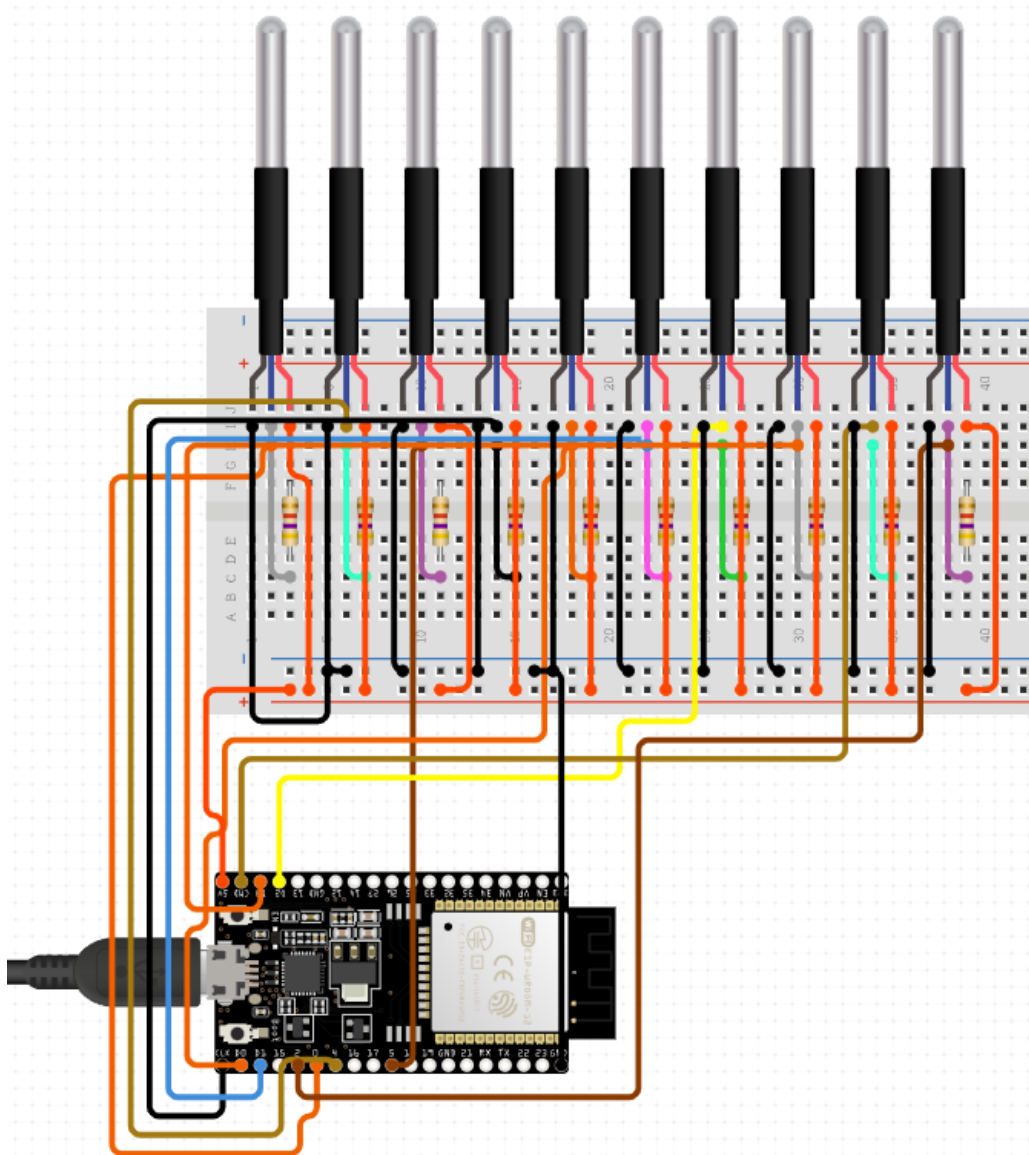
Para validar o fluxograma desenvolvido para o sistema, inicialmente foram realizadas montagens em *protoboard*. Os diagramas de conexões foram desenvolvidos com o auxílio de um *software online* disponível no link www.circuito.io.

Figura 27 - Esquema de conexões do *datalogger* para a Pesquisa A



Fonte: Autoria própria.

Figura 28 - Esquema de conexões do *datalogger* para a Pesquisa B

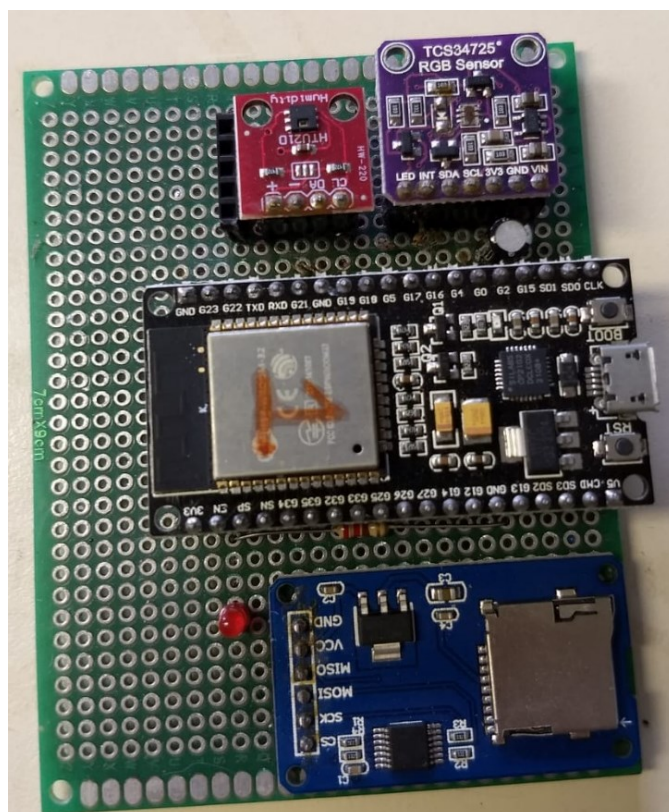


Fonte: Autoria própria.

5.3.6 MONTAGEM DO CIRCUITO E TESTE

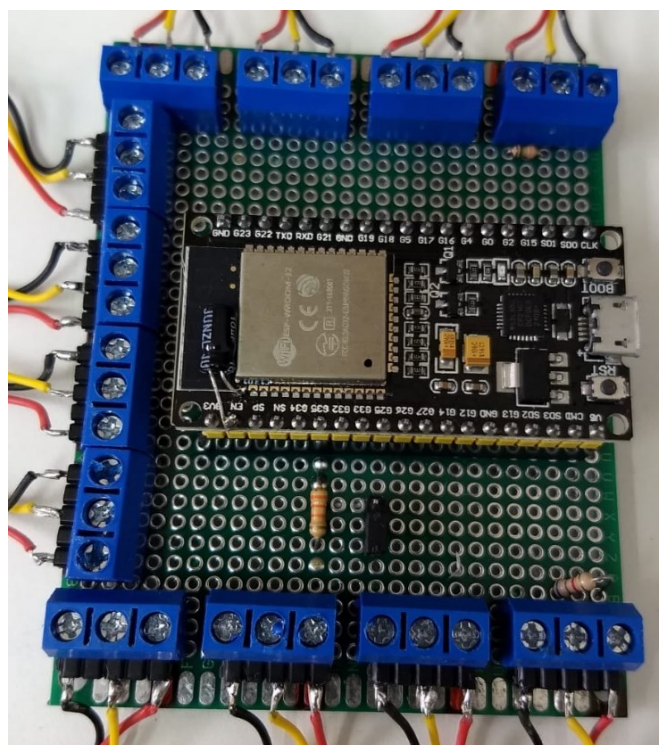
A proposta inicial para montagem dos circuitos era de ser desenvolvido o layout das placas via software bem como a confecção das mesmas em uma máquina CNC ou até mesmo por processo industrial, porém, por motivos de tempo para entrega dos sistemas funcionando, essa proposta foi deixada de lado e os circuitos foram montados em placas de circuito perfurada para prototipagem.

Figura 29 - Placa do *datalogger* para a Pesquisa A



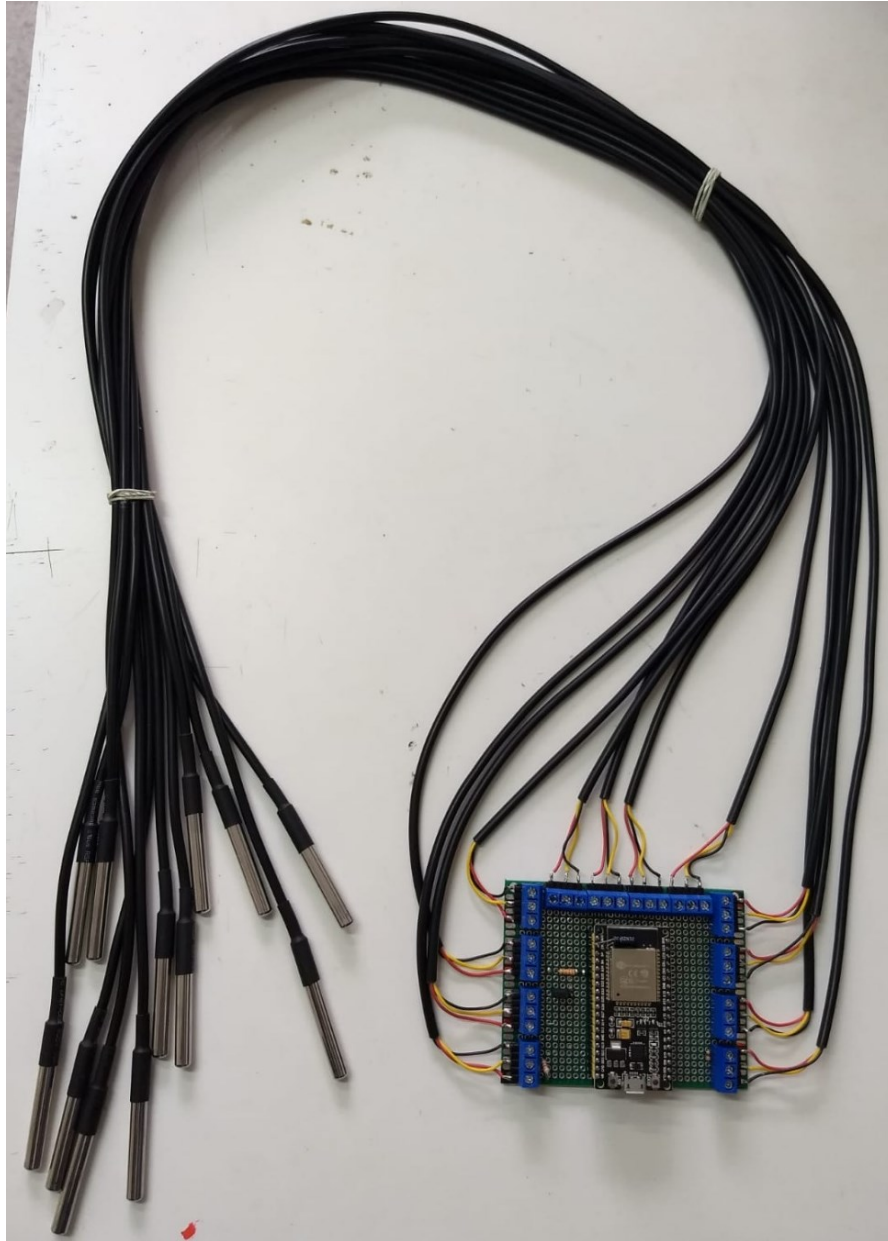
Fonte: Autoria própria.

Figura 30 - Placa do *datalogger* para a Pesquisa B



Fonte: Autoria própria.


Figura 31 - Datalogger com os sensores DS18B20 para a Pesquisa B



Fonte: Autoria própria.

Ao ligar, os sistemas de aquisição de dados irão se conectar à rede de internet e enviar os dados dos sensores a cada 10 minutos, permitindo serem acessados nas planilhas criadas.

Figura 32 - Planilha com os dados

 Daniele Melo - Ambiente 1 ☆ 📁 ☁

Arquivo Editar Ver Inserir Formatar Dados Ferramentas Extensões Ajuda [A última edição foi feita](#)

100% | R\$ % .0 .00 123 | Arial | 10 | **B** *I* U **A** 📄 🗑️ 🔄

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	DATA	HORA	Temperatura	Umidade	Temp_Cor	Lux	Red	Green	Blue	Clear	PAR	PAR2	
95	22/06/2022	05:26:13	999.00	113.93	6471	36	48	49	59	126	0	302	
96	22/06/2022	05:36:13	20.77	115.05	8116	262	294	356	457	944	3	2141	
97	22/06/2022	05:46:13	20.89	115.01	6274	1384	1509	1584	1800	3940	20	9563	
98	22/06/2022	05:56:13	21.07	113.88	6039	2072	2179	2269	2511	5619	31	13677	
99	22/06/2022	06:06:13	999.00	115.19	6068	2974	3057	3217	3549	8030	44	19376	
100	22/06/2022	06:16:13	21.15	115.22	6121	5502	5571	5927	6551	15017	82	35753	
101	22/06/2022	06:26:13	21.15	114.94	6040	4100	4132	4376	4803	11136	61	26442	
102	22/06/2022	06:36:13	21.61	113.88	6014	8050	8129	8583	9404	21806	120	51871	
103	22/06/2022	06:46:12	22.01	112.39	5997	7042	7093	7478	8168	18841	105	45094	
104	22/06/2022	06:56:13	21.78	112.74	5849	9450	9520	9909	10689	24810	141	59778	
105	22/06/2022	07:06:15	22.52	109.46	5833	13826	13748	14346	15393	35821	207	20847	
106	22/06/2022	07:16:13	22.52	109.55	5809	14400	14282	14888	15933	37196	216	24128	
107	22/06/2022	07:26:13	23.28	106.79	5795	19572	19443	20236	21643	50656	293	56444	
108	22/06/2022	07:36:23	999.00	104.76	5770	18030	17857	18567	19801	46542	270	46426	
109	22/06/2022	07:46:22	23.70	104.38	5797	25272	24890	25998	27759	65525	379	25768	
110	22/06/2022	07:56:22	23.55	105.56	5791	21952	21608	22552	24051	56498	329	4767	
111	22/06/2022	08:06:23	23.55	105.08	5891	18706	18224	19247	20643	48366	280	50202	
112	22/06/2022	08:16:22	23.98	104.02	5781	23000	22662	23621	25174	59125	345	11212	
113	22/06/2022	08:26:22	24.33	103.06	5854	23774	23513	24633	26449	61811	356	17314	
114	22/06/2022	08:36:23	23.49	106.88	0	27916	27016	28506	30419	65535	418	36292	
115	22/06/2022	08:46:24	24.48	106.88	0	44506	44204	45548	45864	65535	633	33402	

+ ☰ **Página1** ▼

Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

A disciplina de Estágio Obrigatório contida na grade curricular do curso de Engenharia Elétrica da UFCG é fundamental para que o graduando possa colocar em prática conteúdos abordados durante o curso.

Neste trabalho foi apresentado uma sequência de procedimentos utilizados durante a atividade do estágio, englobando desde a fundamentação teórica para execução do mesmo bem como os testes e os resultados apresentados com o sistema desenvolvido.

Durante a realização do estágio foi possível aplicar conceitos e técnicas sobre eletrônica e instrumentação no desenvolvimento de dois tipos de sistemas de aquisição de dados para leitura de sensores de temperatura, umidade relativa, cor e luminosidade.

7 REFERÊNCIAS

- [1] “Aquisição de Dados”. <https://br.omega.com/prodinfo/aquisicao-de-dados.html> (acessado 27 de junho de 2022).
- [2] “Sensores e Transdutores | Render Blog”, 5 de dezembro de 2014. <https://blog.render.com.br/diversos/sensores-e-transdutores/> (acessado 27 de junho de 2022).
- [3] “Trazendo o mundo real para dentro do processador - Conversor A/D”, *Embarcados - Sua fonte de informações sobre Sistemas Embarcados*, 18 de setembro de 2015. <https://www.embarcados.com.br/conversor-a-d/> (acessado 27 de junho de 2022).
- [4] N. C. Braga, “Como funcionam os Conversores A/D - parte 1 (ART224)”, *Instituto Newton C. Braga*. <https://www.newtoncbraga.com.br/index.php/como-funciona/1508-conversores-ad> (acessado 27 de junho de 2022).
- [5] “I2C Bus”. <https://www.i2c-bus.org/> (acessado 27 de junho de 2022).
- [6] S. C. | D. Electronics | 55, “Basics of the I2C Communication Protocol”, *Circuit Basics*, 13 de fevereiro de 2016. <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/> (acessado 27 de junho de 2022).
- [7] E. Teixeira, “Biblioteca I2C para FRDM-KL25Z”, *Embarcados - Sua fonte de informações sobre Sistemas Embarcados*, 20 de abril de 2017. <https://embarcados.com.br/biblioteca-i2c-para-frdm-kl25z/> (acessado 8 de julho de 2022).
- [8] “1 fio”. <http://stringfixer.com/pt/1-Wire> (acessado 28 de junho de 2022).
- [9] “One wire”, *Wikipédia, a enciclopédia livre*. 8 de abril de 2022. Acessado: 28 de junho de 2022. [Online]. Disponível em: https://pt.wikipedia.org/w/index.php?title=One_wire&oldid=63353022
- [10] “1-Wire Communication Protocol”. <http://ptcomputador.com/Ferragens/network-equipment/46080.html> (acessado 28 de junho de 2022).
- [11] “1899_HTU21D.pdf”. Acessado: 29 de junho de 2022. [Online]. Disponível em: https://cdn-shop.adafruit.com/datasheets/1899_HTU21D.pdf
- [12] T. Tecnologia, “HTU21D Sensor de Temperatura e Umidade”. <https://www.easytronics.com.br/htu21d-sensor-de-temperatura-e-umidade> (acessado 28 de junho de 2022).
- [13] “TCS34725.pdf”. Acessado: 29 de junho de 2022. [Online]. Disponível em: <https://cdn-shop.adafruit.com/datasheets/TCS34725.pdf>
- [14] T. Tecnologia, “Sensor RGB TCS34725 | Reconhecimento de Cores”. <https://www.easytronics.com.br/sensor-rgb-tcs34725> (acessado 28 de junho de 2022).
- [15] “DS18B20 - Programmable Resolution 1-Wire Digital Thermometer”, p. 20.
- [16] T. Tecnologia, “Sensor de Temperatura DS18B20 à Prova d’Água Waterproof”. <https://www.casadarobotica.com/sensores-e-modulos/sensores/temperatura/sensor-de-temperatura-ds18b20-a-prova-d-agua-waterproof> (acessado 1º de julho de 2022).
- [17] “DS18B20 - Programmable Resolution 1-Wire Digital T.pdf”. Acessado: 1º de julho de 2022. [Online]. Disponível em: <https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>
- [18] C. Z. Nazário, “Aprenda como usar o ESP32 para publicar dados no Google Sheets”, *Crescer*, 12 de abril de 2021. <https://www.crescerengenharia.com/post/aprenda-como-usar-o-esp32-para-publicar-dados-no-google-sheets> (acessado 5 de julho de 2022).
- [19] T. L. S. Santos, “Implementação de Filtros FIR pelo Método Overlap-Save com Microcontrolador”.
- [20] “esp32_technical_reference_manual_en.pdf”. Acessado: 24 de maio de 2022. [Online]. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

ANEXO A - CÓDIGO DO DATALOGGER DA

PESQUISA A:

```
#include <Arduino.h>
#include <HTTPClient.h>
#include <WiFi.h>
#include "string.h"
#include <SPI.h>
#include <Wire.h>
#include "SparkFunHTU21D.h"
#include "Adafruit_TCS34725.h"

#define LED 13
#define TEMPO_10_S 10000
#define TEMPO_10_M 600000

/*****
 * Constantes e variáveis globais
 * *****/
const char* ssid = "LACRA Estudos"; // Identificação da rede
const char* password = "";
char *server = "script.google.com"; // Server URL
char *GScriptId = "AKfycbw-eDqzdUiPCmKvnf6gvCc1fUblDHzKfjc5aRiiPwrew7xaG0z6ELnd-2ygqFhAPBFs"; //Link da planilha Ambiente 4

const int httpsPort = 443;
const float cf_par = 0.015;
const float cf_lux = 2.0;
int counter;
long tempo1, tempo2, espera;
float umidade = 0.0;
float temperatura = 0.0;
uint16_t r, g, b, c, colorTemp, luminance, lux, par, par2;

/*****
 * Instâncias de objetos
 * *****/
WiFiClientSecure client;
HTU21D htu21d;
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_614MS,
TCS34725_GAIN_1X);
```

```

/*****
 * Protótipo de funções
 * *****/
void connect_wifi(void);
void enviarMedicao(void);

/*****
 * Setup
 * *****/
void setup() {
    counter = 0;                // Zera o contador para reset

    Serial.begin(115200);        // Inicializa a comunicação serial

    /*****
     * Inicialização de pinos de entrada e saída
     * *****/
    pinMode(LED, OUTPUT);
    digitalWrite(LED, LOW);

    /*****
     * Inicialização do sensor de temperatura e umidade
     * *****/
    htu21d.begin();

    /*****
     * Inicialização do sensor de cor e luminosidade
     * *****/
    if (tcs.begin()){
        Serial.println("\n\nSensor de cor detectado!");
    } else{
        Serial.println("Sensor de cor TCS34725 não detectado!");
    }

    connect_wifi();              // Inicializa o wi-fi
}

/*****
 * Loop
 * *****/
void loop() {
    tempo1 = millis();

    /*****
     * Medições
     * *****/
    if(tempo1 >= (tempo2 + TEMPO_10_M)){
        tempo2 = tempo1;
        counter++;
    }
}

```

```

    // Leitura do sensor de cores e luminosidade
    tcs.getRawData(&r, &g, &b, &c);
    colorTemp = tcs.calculateColorTemperature_dn40(r, g, b, c);
    luminance = tcs.calculateLux(r, g, b);
    lux = luminance * cf_lux;
    par = lux * cf_par;
    par2 = (0.65847 * c) + (1.60537 * r) + (2.30216 * g) + (0.50019 * b);

    // Leitura do sensor de umidade e temperatura
    umidade = htu21d.readHumidity();
    temperatura = htu21d.readTemperature();

    enviarMedicao();
}
}

/*****
 * Função para conectar ao WiFi
 * *****/
void connect_wifi(void){
    Serial.print("Connecting to wifi: ");
    Serial.println(ssid);
    Serial.flush();
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        digitalWrite(LED, !digitalRead(LED));
        Serial.print(WiFi.status());
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    digitalWrite(LED, HIGH);
}

/*****
 * Função para enviar os dados
 * *****/
void enviarMedicao(){
    HTTPClient http;

    String url = String("https://script.google.com") + "/macros/s/" + GScriptId +
"/exec?" + "value1=" + String(temperatura) + "&value2=" + String(umidade)
+ "&value3=" + String(colorTemp) + "&value4=" + String(lux) + "&value5=" + String(r)

```

```
+ "&value6=" + String(g) + "&value7=" + String(b) + "&value8=" + String(c) +  
"&value9=" + String(par) + "&value10=" + String(par2);  
  
Serial.print("Making a request");  
http.begin(url.c_str());           //Specify the URL and certificate  
http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);  
int httpCode = http.GET();  
String payload;  
if (httpCode > 0) {                //Check for the returning code  
    payload = http.getString();  
    Serial.println(httpCode);  
    Serial.println(payload);  
}  
else {  
    Serial.println("Error on HTTP request");  
}  
http.end();  
}
```

ANEXO B - CÓDIGO DO DATALOGGER DA

PESQUISA B:

```
#include <Arduino.h>
#include <HTTPClient.h>
#include <WiFi.h>
#include "string.h"
#include "OneWire.h"
#include "DallasTemperature.h"

#define N_SENSORES 12
#define ONE_WIRE_BUS_1 16
#define ONE_WIRE_BUS_2 17
#define ONE_WIRE_BUS_3 25
#define ONE_WIRE_BUS_4 13
#define TEMPERATURE_PRECISION 9
#define TEMPO_10_S 10000
#define TEMPO_10_M 600000
#define TEMPO_1_H 3600000
#define PIN_TEMPO 35

/*****
 * Constantes e variáveis globais
 * *****/

const char* ssid = "LACRA Estudos";          // Identificação da rede
const char* password = "";
char *server = "script.google.com";          // Server URL
char *GScriptId =
"AKfycbwuPhmrir65VQBHSmSQeG0xniQurBf1s3BaChEC5NiXGKR1j092fxbgT6w5Xbbz3QvyNg"; //Link
da planilha
const int httpsPort = 443;
int counter;
float Temperatura[N_SENSORES];
long tempo1, tempo2, espera;

/*****
 * Instâncias de objetos
 * *****/

WiFiClientSecure client;
OneWire oneWire_1(ONE_WIRE_BUS_1);
OneWire oneWire_2(ONE_WIRE_BUS_2);
OneWire oneWire_3(ONE_WIRE_BUS_3);
OneWire oneWire_4(ONE_WIRE_BUS_4);
```

```

DallasTemperature sensors_1(&oneWire_1);
DallasTemperature sensors_2(&oneWire_2);
DallasTemperature sensors_3(&oneWire_3);
DallasTemperature sensors_4(&oneWire_4);

/*****
 * Protótipo de funções
 * *****/

void connect_wifi(void);
void enviarMedicao(void);
void leituraTemp1(void);
void leituraTemp2(void);
void leituraTemp3(void);
void leituraTemp4(void);

/*****
 * Setup
 * *****/
void setup() {
    counter = 0;                // Zera o contador para reset

    Serial.begin(115200);        // Inicializa a comunicação serial

    pinMode(PIN_TEMPO, INPUT); // Pino de seleção para escolha do tempo de coleta de
    dados

    // Inicialização dos sensores DS18B20
    sensors_1.begin();
    sensors_1.begin();
    sensors_2.begin();
    sensors_3.begin();
    sensors_4.begin();

    connect_wifi();              // Inicializa o wi-fi
}

/*****
 * Loop
 * *****/
void loop() {
    tempo1 = millis();
    // Testa o pino de seleção de temporização (10 segundos ou 10 minutos)
    if(digitalRead(PIN_TEMPO) == HIGH){
        espera = TEMPO_10_M;
    }
    else{
        espera = TEMPO_10_S;
    }
}

```



```

    }

    leituraTemp1();
    leituraTemp2();
    leituraTemp3();
    leituraTemp4();

    if(tempo1 >= (tempo2 + espera)){
        tempo2 = tempo1;
        counter++;

        enviarMedicao();
    }
}

/*****
 * Função para conectar ao WiFi
 * *****/
void connect_wifi(void){
    Serial.print("Connecting to wifi: ");
    Serial.println(ssid);
    Serial.flush();
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(WiFi.status());
        Serial.print(".");
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

/*****
 * Função para enviar os dados
 * *****/
void enviarMedicao(){
    HTTPClient http;

    String url = String("https://script.google.com") + "/macros/s/" + GScriptId +
"/exec?" + "value1=" + String(Temperatura[0]) + "&value2=" + String(Temperatura[1]) +
"&value3=" + String(Temperatura[2]) + "&value4=" + String(Temperatura[3]) +
"&value5=" + String(Temperatura[4]) + "&value6=" + String(Temperatura[5]) +
"&value7=" + String(Temperatura[6]) + "&value8=" + String(Temperatura[7]) +
"&value9=" + String(Temperatura[8]) + "&value10=" + String(Temperatura[9]) +
"&value11=" + String(Temperatura[10]) + "&value12=" + String(Temperatura[11]);

```

```

Serial.print("Making a request");
http.begin(url.c_str()); //Specify the URL and certificate
http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);
int httpCode = http.GET();
String payload;
if (httpCode > 0) { //Check for the returning code
    payload = http.getString();
    Serial.println(httpCode);
    Serial.println(payload);
}
else {
    Serial.println("Error on HTTP request");
}
http.end();
}

/*****
 * Função de leitura dos sensores no barramento 1
 * *****/
void leituraTemp1(void){
    sensors_1.requestTemperatures();
    Temperatura[0] = sensors_1.getTempCByIndex(0);
    Temperatura[1] = sensors_1.getTempCByIndex(1);
    Temperatura[2] = sensors_1.getTempCByIndex(2);
}

/*****
 * Função de leitura dos sensores no barramento 2
 * *****/
void leituraTemp2(void){
    sensors_2.requestTemperatures();
    Temperatura[3] = sensors_2.getTempCByIndex(0);
    Temperatura[4] = sensors_2.getTempCByIndex(1);
    Temperatura[5] = sensors_2.getTempCByIndex(2);
}

/*****
 * Função de leitura dos sensores no barramento 3
 * *****/
void leituraTemp3(void){
    sensors_3.requestTemperatures();
    Temperatura[6] = sensors_3.getTempCByIndex(0);
    Temperatura[7] = sensors_3.getTempCByIndex(1);
    Temperatura[8] = sensors_3.getTempCByIndex(2);
}

/*****
 * Função de leitura dos sensores no barramento 4

```

```
* **** */
void leituraTemp4(void){
    sensors_4.requestTemperatures();
    Temperatura[9] = sensors_4.getTempCByIndex(0);
    Temperatura[10] = sensors_4.getTempCByIndex(1);
    Temperatura[11] = sensors_4.getTempCByIndex(2);
}
```