UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ESPECIALIZAÇÃO EM SISTEMAS EMBARCADOS PARA INDÚSTRIA AUTOMOTIVA

ANDRÉ LUIZ LIMA DO COUTO

SOFTWARE PARA MONITORAMENTO DE ABERTURA E FECHAMENTO DO VIDRO AUTOMOTIVO

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2021

ANDRÉ LUIZ LIMA DO COUTO

SOFTWARE PARA MONITORAMENTO DE ABERTURA E FECHAMENTO DO VIDRO AUTOMOTIVO

Monografia de Especialização, apresentada ao Curso de Especialização em Sistemas Embarcados para Indústria Automotiva, do Departamento Acadêmico de Eletrônica – DAELN, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Especialista.

Orientador: Prof. M. Sc. Luiz Fernando Copetti

CURITIBA
2021

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Curitiba

Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Eletrônica
Curso de Especialização em Sistemas Embarcados para Indústria Automotiva

TERMO DE APROVAÇÃO

SOFTWARE PARA MONITORAMENTO DE ABERTURA E FECHAMENTO DO VIDRO AUTOMOTIVO

por

André Luiz Lima do Couto

Esta monografia foi apresentada em 08 de Dezembro de 2021 como requisito parcial para a obtenção do título de Especialista em Sistemas Embarcados para Indústria Automotiva. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. M.Sc. Luiz Fernando Copetti
Orientador

_____
Prof. Dr. Kleber Kendy Horikawa Nabas
Membro titular

_____
Prof. M. Sc. Omero Francisco Bertol
Membro titular

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

DEDICATÓRIA

Dedico esse trabalho de conclusão de curso de especialização a minha família, que me apoiou na dedicação a todos os esforços necessários para ter sucesso em minhas buscas e conquistas.

AGRADECIMENTOS

Agradeço a todos que me ajudaram diretamente e indiretamente para que esse trabalho fosse concluído, em especial a minha família que sempre me apoio em seguir com os estudos, e aos professores que ministraram o Curso de Especialização em Sistemas Embarcados para Industria Automotiva, devido a atenção dada durante o curso e o canal aberto para sanar possíveis dúvidas.

RESUMO

COUTO, André Luiz Lima do. **Software para Monitoramento de Abertura e Fechamento do Vidro Automotivo**. 2021. 33 páginas. Monografia de Especialização em Sistemas Embarcados para Indústria Automotiva, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

Esse trabalho demonstrar uma tecnologia de programação por Sistema Operacional de Tempo Real, para ser possível essa demonstração foi utilizado o exemplo de uma necessidade de monitorar a abertura e fechamento de vidro automotivo. Além de aplicado conceitos de programação, nesse trabalho aplica-se conhecimento de arquitetura de microcontrolador e exemplos de alguns componentes eletrônicos para a validação do software como LEDs, buzzers, resistores, protoboards e jumpers. A tecnologia de programação usada nesse trabalho é uma as mais utilizadas nos dias atuais, sendo amplamente utilizada por inúmeros desenvolvedores de programas.

Palavras-chave: Monitoramento. Microcontrolador. Sistema Operacional. Indústria Automotiva.

ABSTRACT

COUTO, André Luiz Lima do. **Automotive Glass Opening and Closing Monitoring Software**. 2021. 33 páginas. Monografia de Especialização em Sistemas Embarcados para Indústria Automotiva, Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2021.

This work demonstrates a programming technology for Real Time Operating System, to be possible this demonstration is used the example of a need to monitor the opening and closing of automotive glass. In addition to applying programming concepts, this work applies knowledge of microcontroller architecture and examples of some electronic components for software validation such as LEDs, buzzers, resistors, protoboards and jumpers. The programming technology used in this work is one of the most used nowadays, being widely used by numerous program developers.

Keywords: Monitoring. Microcontroller. Operational system. Automotive industry.

LISTA DE FIGURAS

sumário

1 INTRODUÇÃO

Este trabalho é sobre um software de monitoramento de abertura e fechamento de vidro automotivo, neste trabalho será apresentado os conhecimentos adquiridos no Curso Especialização em Sistemas Embarcados para a Indústria Automotiva, vale citar as disciplinas Métodos de Desenvolvimento de Software Automotivo, Estrutura de Microcontroladores e Processamento de Sinais como base para o desenvolvimento desse trabalho.

A importância desse trabalho é exemplificar como a tecnologia cada vez está mais acessível, e pode ser aplicada em várias situações para resolver dos mais simples até os mais complexos dos problemas. Será explorado o desenvolvimento estruturado de software baseado na linguagem C/C++, e componentes eletrônicos necessários para teste prático do software.

1.1 PROBLEMA

A ideia deste trabalho surgiu de um problema pessoal que tive em meu veículo onde meu filho de 1 ano e 4 meses de idade abria o vidro traseiro esquerdo do meu carro sem eu perceber.

1.2 OBJETIVOS

Objetivo geral do trabalho é desenvolver um software de monitoramento de abertura e fechamento do vidro automotivo. O objetivo especifico é pôr em prática os conhecimentos adquiridos no Curso de Especialização em Sistemas Embarcados para Industria Automotiva.

1.2.1 Objetivo Geral

Desenvolver um software que através de periféricos mostre de modo intuitivo a o usuário a porcentagem de abertura e fechamento dos vidros automotivos.

1.2.2 Objetivos Específicos

- Colocar em prática os conhecimentos adquiridos ao longo do curso;

- Informar a movimentação de abertura e fechamento de vidro automotivo;
- Alertar por meios visuais e auditivos caso os vidros abram além do que o usuário deseja.
- Evitar que o usuário tenha que desviar os olhos da estrada para verificar se os vidros estão na condição de abertura ou fechamento que ele deseja.

1.3 JUSTIFICATIVA

A importância de evitar tirar os olhos da estrada, vias, ruas, etc., ... quando se está com o carro em movimento.

Alertar que o vidro do carro está aberto o suficiente para que uma criança, por exemplo, consiga jogar algum objeto (brinquedo, chupeta, mamadeira, etc., ...) de dentro para fora do veículo.

Como objetivo principal desse trabalho é criar um software de fácil manutenção, e que possa ser atribuído novas características de funcionamento com pouca mudança no programa já desenvolvido.

1.4 ESTRUTURA DO TRABALHO

Esta monografia de especialização está dividida em 5 (cinco) seções, a primeira seção foi introduzido o assunto tema do trabalho e também foram abordados a motivação e os objetivos geral e específicos da pesquisa, a justificativa e a estrutura geral do trabalho.

Já na segunda seção será abordada as áreas de conhecimento chave para a elaboração do trabalho na revisão bibliográfica.

A seguir na terceira seção será abordado todo o desenvolvimento do software e o desenvolvimento de hardware para validação do software.

Na quarta seção será apresentado os resultados obtidos com a elaboração do trabalho relacionado com os conhecimentos adquiridos e testes práticos.

Por ultimo na quinta seção: Citações de fontes do trabalho, os objetivos atingidos e proposta de trabalho futuros.

2 REVISÃO BIBLIOGRÁFICA

2.1 TECNOLOGIA AUTOMOTIVA

Os primeiros automóveis tinham como prioridade facilitar o transporte de carga (caminhões) e de pessoas (ônibus e carros), porém com o avanço da tecnologia, o conforto no interior dos automóveis pode ser um diferencial na disputa na venda de veículos pelas montadoras, um fator importante é que nos grandes centros urbanos, as vezes pode se levar até 2 horas para ir de casa até o trabalho, e o conforto dentro dos veículos pode melhorar a qualidade de vida do usuário.

2.2 MICROCONTROLADORES

Segundo Arduino.CC os microcontroladores são componentes eletrônicos mais usados atualmente, pois se trata de um circuito integrado de multifunções, assim um único microcontrolador pode fazer a função de vários componentes eletrônicos. Algumas de suas funções são:

- Processar cálculos aritméticos e lógicos;
- Memória de leitura e escrita para armazenamento de dados;
- Entrada e saída de sinais analógicos e digitais.

Os microcontroladores podem ser usados para automação e controle de produtos e seus periféricos, como carros, computadores, motores industriais, sistemas de iluminação, controle remotos, brinquedos entre outros.

Por ter seu tamanho relativamente pequeno, consumir pouca energia, baixo custo e formato de fácil aplicação em circuitos impressos, os microcontroladores são uma alternativa eficiente para controlar muitos processos.

2.3 PLATAFORMA ARDUINO

A plataforma Arduino segundo Arduino.CC começou no ano 2005 com o objetivo de facilitar a vida de estudante que tivesse o intuito de criar projetos que necessitasse de controle integrado e interação com o usuário. Trata-se de uma placa com circuito eletrônico impresso desenvolvido na Itália, seus desenvolvedores

Massimo Banzi, David Cuatielles, Tom Igoe, Gianluca Martino e David Mellis se reunião em um bar da cidade italiana de Ivrea após o expediente de trabalho, esse Bar tinha o nome de Arduino, em referência a um nobre italiano do século X e XI que se dominou rei de toda Itália que mais tarde acabou morto por rivais.

Hoje em dia a família Arduino está muito grande, além de PCB (placa circuito impresso) para gravação de programas em microcontroladores, dispõem de varias soluções para os mais diversos tipos de projetos.

2.4 PROGRAMAÇÃO EM C++

A linguagem de programação C++ segundo Casavella é derivada de outras linguagens: ALGOL 1968, BCPL 1969, B 1970, C 1972 até C++ 1986. A linguagem C teve como foco o desenvolvimento de sistemas operacionais e compiladores, ela teve grande aceitação devido sua versatilidade e características de linguagem de “baixo nível” e “alto nível”. A linguagem C++ foi a evolução da linguagem C com o intuito de incorporar comandos para facilitar a programação de orientação a objetos.

Apesar de ser uma linguagem genérica utilizadas para vários propósitos a linguagem C++ exige grande dedicação e atenção do programador, pois ela é “Case Sensitive”, ou seja, comando em letras maiúscula e minúsculas possuem diferentes sentidos, assim o compilador do programa pode interpretar uma palavra como variável ao invés de um comando. Exemplo: mostre “Soma”, comando “sOmA”. A linguagem C/C++ foi padronizada e revisada pela *American National Standard Institute* (ANSI), assim foi feito para evitar a incompatibilidade da programação entre os semelhantes compiladores.

2.5 O “FREERTOS”

Por volta do ano de 2003 o desenvolvedor de software e empresário Richard Barry desenvolveu um código aberto de bibliotecas de programas baseado no RTOS (“*Real Time Operation System*” – Sistema Operacional de Tempo Real) esse código aberto, denominado “FreeRTOS”, foi um grande sucesso muito devido a sua compatibilidade com várias plataformas de microcontroladores. Em 2017 *a Amazon Web Services* (AWS) adquiriu o projeto “FreeRTOS”, porém Richard continua trabalhando agora como parte de uma equipe da AWS. No ano de 2018 o

“FreeRTOS” se consagrou como Sistema Operacional de Tempo Real mais baixado no mundo. Alcançando a marca de um download a cada 175 segundos.

Um ponto bem interessante do “FreeRTOS” que além do download ser gratuito, também o seu uso é gratuito, ou seja, pode ser usado em projetos comerciais sem a necessidade de pagar uma quantia pelo direito de uso a o proprietário (no caso a AWS). No site onde é possível baixar o “FreeRTOS”, possui várias documentações que auxiliam no uso da plataforma assim como exemplos de códigos já prontos.

3 DESENVOLVIMENTO

3.1 CONCEITO DE ELABORAÇÃO DO SOFTWARE

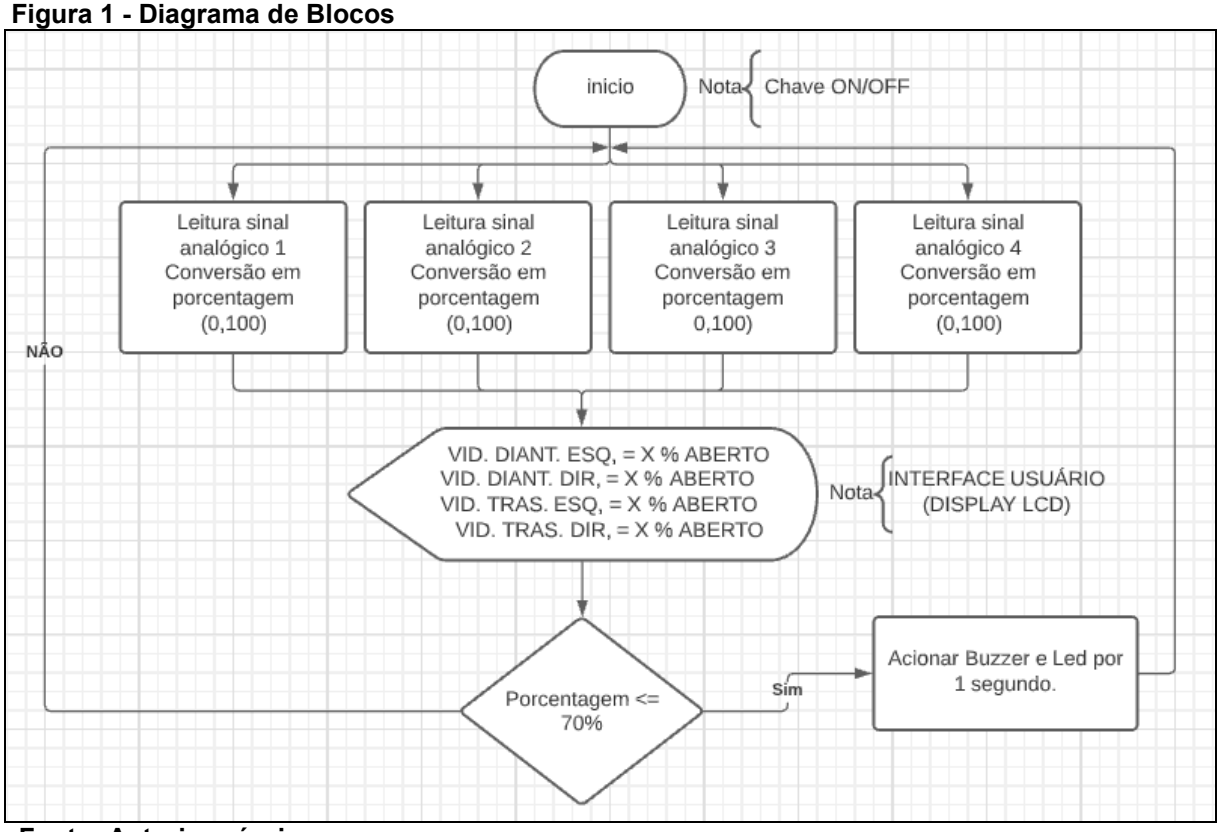
A ideia para a elaboração de um software de monitoramento de abertura e fechamento de vidro automotivo, surgiu de um problema pessoal, que era não saber se os vidros, principalmente os traseiros, estavam fechados ou abertos sem que eu desvia-se o olhar da via que eu estava conduzindo. Esse tipo de monitoramento ficou mais coerente, no meu ponto de vista, quando meu filho estava na idade de 1 ano e 6 meses, as vezes só percebia que ele tinha aberto o vidro traseiro direito do carro, quando uma forte rajada de vento pegava na lateral do carro, ocasionando em um barulho interno dentro do automóvel. Assim tive a ideia de elaborar um software que através de um display de LCD mostra-se a o condutor em porcentagem a posição dos vidros de 0 a 100%, onde 0% os vidros estão totalmente abertos e 100% os vidros estão totalmente fechados, também o software controlará quatro “LED” (diodo emissor de luz) e quatro “Buzzer” (componente eletrônico emissor de som). O LED e o Buzzer serão acionados todas vezes que o vidro tiver fechamento igual ou menor a 70%.

3.2 DIAGRAMAS DE BLOCOS

O Diagrama de Blocos, é um meio muito utilizado por desenvolvedores para organizar a lógica de funcionamento de um algoritmo, algoritmo é basicamente todos os passos da rotina de funcionamento de um programa. Assim na Figura 1 o diagrama de blocos mostra a ideia de funcionamento do software.

No retângulo de arestas arredondadas demonstra o início do sistema, os retângulos regulares demostram o processamento das condições de funcionamento, no retângulo de duas arestas arredondadas e um lado pontiagudo exibe as informações apresentadas de interesse do usuário, no losango indica as condições de funcionamento levando em conta os dados apresentados à o usuário.

Figura 1 - Diagrama de Blocos



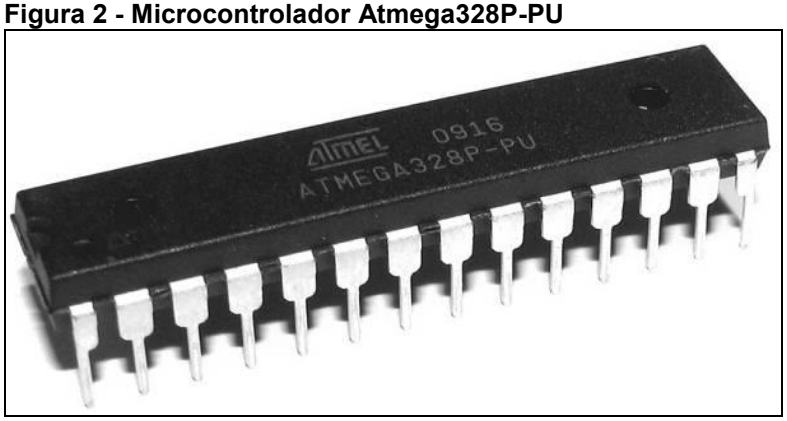
Fonte: Autoria própria.

Determinado os passos de funcionamento do Software Monitoramento de Abertura e Fechamento Vidro Automotivo, foi iniciado a pesquisa do Hardware para teste do Software.

3.3 ESCOLHA DO MICROCONTROLADOR

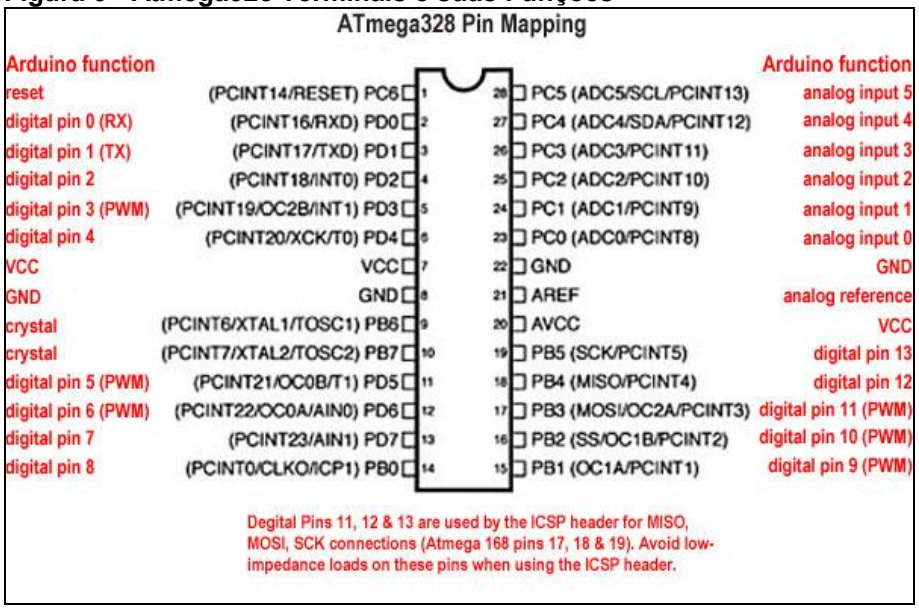
Assim que estabelecido a lógica de funcionamento do software de monitoramento, o próximo passo foi fazer pesquisas para encontrara um meio de fazer a leitura do posicionamento dos vidros. Depois de muitas horas de pesquisas e estudo, foi escolhido o microcontrolador Atmega328P-PU com seus aspectos físico e terminais apresentados, respectivamente, na Figura 2 e na Figura 3. Esse microcontrolador possui multifuncionalidades, o importante para a confecção do projeto foi a função de se poder inserir em sua memória informações que podem ser lidas para efetuar uma ação. Para inserir essas informações precisa-se de uma interface de programação do microcontrolador, e em base de pesquisas como melhor opção foi escolhido a plataforma Arduino Uno R3.

Figura 2 - Microcontrolador Atmega328P-PU



Fonte: Arduino. CC (2021).

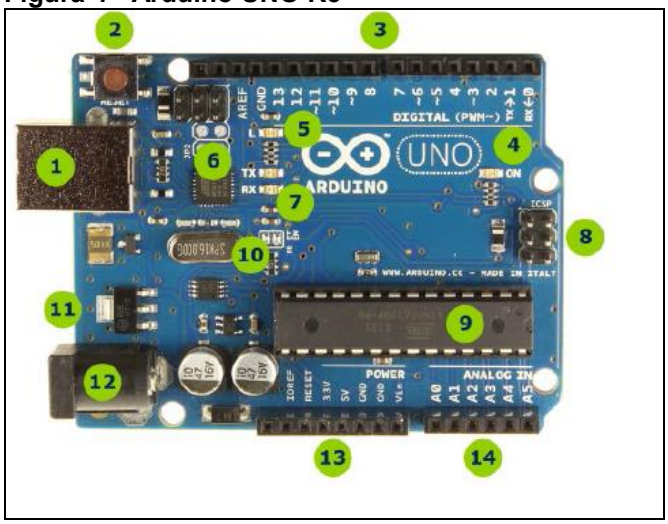
Figura 3 - Atmega328 Terminais e suas Funções



Fonte: Arduino. CC (2021).

A plataforma Arduino Uno R3 com seu aspecto físico e sua estrutura demonstrados, respectivamente, na Figura 4 e na Figura 5, é uma ferramenta muito potente pra programação de microcontroladores, seu fabricante dispõe de um site na internet informando tutorial de seu funcionamento, e sua instalação em computadores é muito fácil. Junto com o software de programação o Arduino tem uma pasta com biblioteca de exemplos de programas.

Figura 4 - Arduino UNO R3



Fonte: Arduino. CC (2021).

Figura 5 - Estrutura Arduino UNO R3

1 - Conector USB para o cabo tipo AB
2 - Botão de reset
3 - Pinos de entrada e saída digital e PWM
4 - LED verde de placa ligada
5 - LED laranja conectado ao pin13
6 - ATmega encarregado da comunicação com o computador
7 - LED TX (transmissor) e RX (receptor) da comunicação serial
8 - Porta ICSP para programação serial
9 - Microcontrolador ATmega 328, cérebro do Arduino
10 - Cristal de quartzo 16Mhz
11 - Regulador de voltagem
12 - Conector fêmea 2,1mm com centro positivo
13 - Pinos de voltagem e terra
14 - Entradas analógicas

Fonte: Arduino. CC (2021).

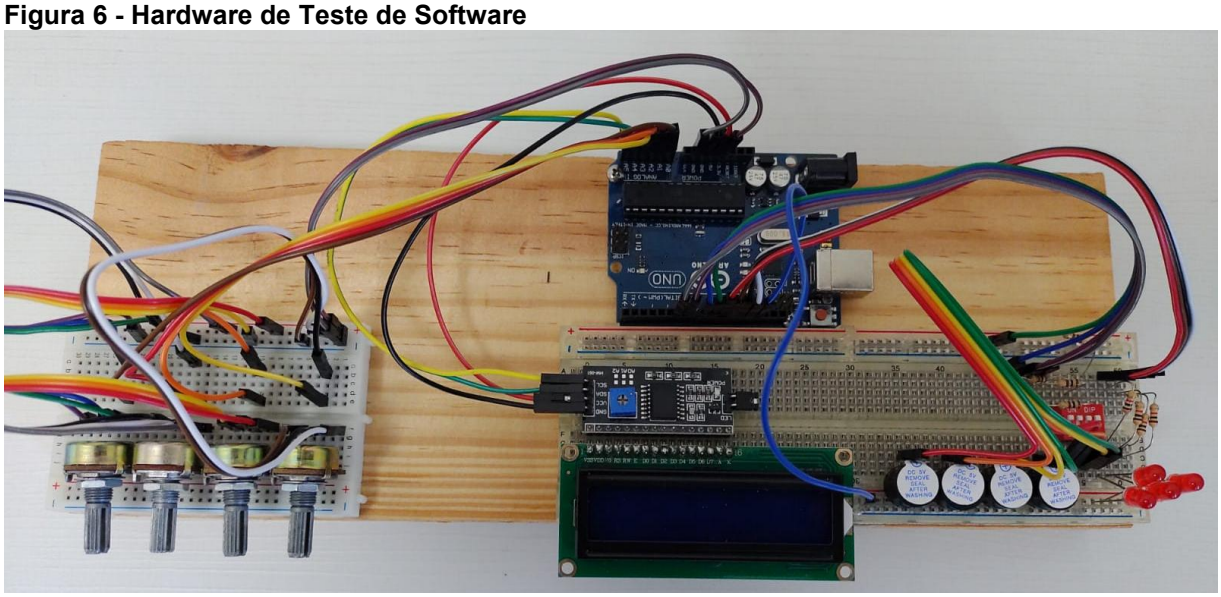
3.4 COMPONENTES PARA TESTE DO SOFTWARE

Foi feito uma pesquisa para verificar um meio de validar se o software realmente funcionaria, assim foi adquirido os seguintes componentes:

- 4 (quatro) potenciômetros de 10k ohms (responsáveis por simular a posição de abertura e fechamento dos vidros automotivos);
- 4 (quatro) LEDs (responsáveis por emissão de sinal de luz);
- 4 (quatro) Buzzers (responsáveis por emissão de sinal sonoro);
- 1 (hum) Display de LCD de 16 colunas por 2 linhas (responsável em informar o usuário a posição dos vidros de 0 a 100%);

- 1 (hum) Módulo Serial I2C para Display LCD Arduino (dispositivo que permite controlar o Display utilizando somente dois terminais);
- 4 (quatro) resistores de 1k ohms (para limitar corrente elétrica para proteção dos LEDs);
- 4 (quatro) resistores de 100 ohms (para limitar corrente elétrica para proteção dos LEDs);
- 1 (hum) Chave On/Off de 4 posições (permitir o usuário desligar o sinal sonoro);
- 1 (hum) Protobord de 400 pontos (montagem dos quatro potenciômetros);
- 1 (hum) Protobord de 830 pontos (montagem dos 4 Buzzers, 4 LEDs, 1 Chave On/Off, 4 Resistores de 1k ohms, 4 Resistores de 100 ohms, Display LCD 16x2 e Módulo Serial I2C para Display LCD Arduino;
- 30 (quarenta) Jumpers macho x macho (ligar os terminais dos componentes nos pinos do Arduino);
- 4 (quatro) Jumpers macho x fêmea (ligar pinos de alimentação e comunicação do Módulo I2C no Arduino;
- 1 (hum) Corte de Madeira Pinus na dimensão 300 mm de comprimento x 100 mm de largura x 20 mm de altura (Base de montagem de todos os componentes adquiridos para facilitar o transporte como mostra a Figura 6).

Figura 6 - Hardware de Teste de Software



Fonte: Autoria própria.

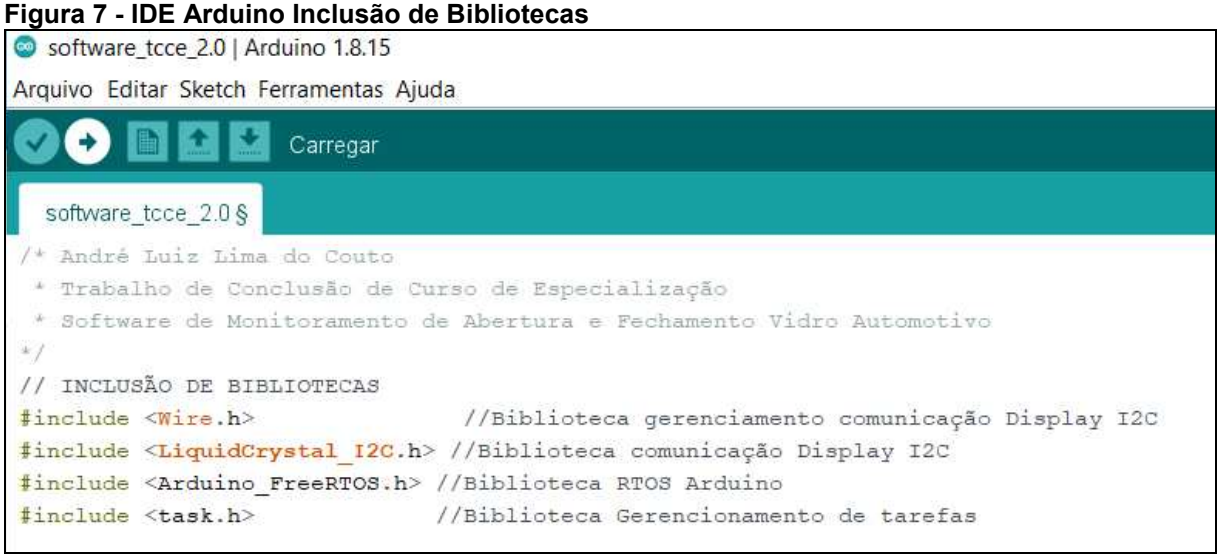
3.5 PROGRAMAÇÃO DO MICROCONTROLADOR NA LINGUAGEM DE PROGRAMAÇÃO C/C++

3.5.1 Apresentação de Ambiente de Programação e Inclusão de Bibliotecas

Para escrever o programa na linguagem C/C++ foi utilizado um Ambiente de Desenvolvimento Integrado (sigla em inglês IDE "*Integrated Development Enviroment")* que é possível ser baixado no site da Arduino. CC. Esse ambiente é característico de varias plataformas de elaboração de programas, e facilita muito a escrita da linguagem de programação, pois tem varias ferramentas que auxiliam o desenvolvedor a escrever. Uma ferramenta muito utilizada na IDE do Arduino é a inclusão de bibliotecas, essas bibliotecas são código já feitos disponíveis quando se baixa o software de instalação da IDE do Arduino e/ou pode ser baixado uma biblioteca da internet se for compatível com a interface do Arduino, esse o caso da biblioteca do FreeRTOS.

Muitos acessórios do Arduino e itens compatíveis, também chamados de periféricos, são vendidos pela internet e geralmente o vendedor já indica uma biblioteca para ser baixada onde se tem o código com funções que são responsáveis pelo correto funcionamento do periférico. Para elaboração do Software de Monitoramento de Abertura e Fechamento de Vidro Automotivo foi utilizado 4 bibliotecas, sendo duas responsáveis pelo funcionamento do Display de LCD com protocolo de comunicação I2C, e outras duas bibliotecas de funcionamento do FreeRTOS. O comando utilizado na IDE do Arduino para incluir bibliotecas é o "#include" demonstrado na Figura 7.

Figura 7 - IDE Arduino Inclusão de Bibliotecas



Fonte: Autoria própria.

3.5.2 Definir na Programação as Funções dos Pinos do Arduino

3.5.2.1 Configuração display LCD e módulo I2C

Como alguns pinos do Arduino podem ter mais de uma função, é necessário definir as funções dos pinos usados. Para a utilizar o Display de LCD as bibliotecas Wire e LiquidCrystal_I2C já fazem boa parte a configuração dos pinos de comunicação do Módulo I2C e com o Arduino, esses pinos são:

- A4 SDA (pino 27 do ATMEGA328);
- A5 SCL (pino 28 do ATMEGA328).

Assim utilizou-se o comando “#define”, demonstrado na Figura 8, para definir para o Arduino a quantidade de colunas e linhas que o Display que estamos usando tem, e o endereço da memória do Módulo I2C para comunicação com o Arduino. Esse endereço geralmente é informado pelo fabricante do Módulo I2C.

Figura 8 - Definição do Display no Código de Programação

```
/*define LCD*/
#define LCD_16X2_I2C_ADDRESS          0x3F
#define LCD_16X2_COLS                 16
#define LCD_16X2_ROWS                 2
```

Fonte: Autoria própria.

3.5.2.2 Configuração entradas analógicas

Nos pinos 23, 24, 25 e 26 do processador ATMEGA328 estão localizados respectivamente as entradas digitais A0, A1, A2 e A3. Essas entradas foram configuradas no código do programa para receber o sinal vindo do potenciômetro como demonstrado na Figura 9, essa função simula um sensor que identifica a posição de abertura e fechamento dos vidros.

Figura 9 - Definição Entradas Analógicas

```
/*define entradas analógicas*/
#define ENTRADA_ANALOGICA_A0          0 //SENSOR VIDRO DIANTEIRO ESQUERDO
#define ENTRADA_ANALOGICA_A1          1 //SENSOR VIDRO DIANTEIRO DIREITO
#define ENTRADA_ANALOGICA_A2          2 //SENSOR VIDRO TRASEIRO ESQUERDO
#define ENTRADA_ANALOGICA_A3          3 //SENSOR VIDRO TRASEIRO DIREITO
```

Fonte: Autoria própria.

3.5.2.3 Configuração saídas digitais

Nos pinos de saídas digitais do 6 a o 12 do Arduino, foi configurado na programação o controle dos buzzers e leds, como de mostrado na Figura 10, que são responsáveis por emitir sinal sonoro e luminoso a o usuário alertando que os vidros podem estar iguais ou menores que 70% fechado.

Figura 10 - Definição Pinos de Saída Digital

```
/*define sáidas digitais*/
#define LED_DD        10              //sáida digital 10
#define BUZZER_DD     6              // sáida digital  6
#define LED_DE        11             //sáida digital 11
#define BUZZER_DE     7              // sáida digital  7
#define LED_TD        12             //sáida digital 12
#define BUZZER_TD     8              // sáida digital  8
#define LED_TE        13              //sáida digital 13
#define BUZZER_TE     9              // sáida digital  9
```

Fonte: Autoria própria.

3.5.3 Definir as Variáveis do Sistemas e Tarefas

Os valores variáveis do sistema devem ser configurados para poderem serem utilizadas pelo processador, assim foi configurados os valores que são apresentados na biblioteca do Display de LCD, e a criação de tarefas e parâmetros que pertencem a biblioteca do FreeRTOS demonstrado na Figura 11.

Figura 11 - Definir Variáveis do Sistema e Tarefas

```
/*variaiveis do sistema*/
LiquidCrystal_I2C lcd(LCD_16X2_I2C_ADDRESS,
                      LCD_16X2_COLS,
                      LCD_16X2_ROWS);
/*Tarefas*/
//void task_lcd(void *pvParamters);
void taks_Vidro_Dianteiro_Direito(void *pvParameters);
void task_Vidro_Dianteiro_Esquerdo(void *pvParameters);
void task_Vidro_Traseiro_Direito(void *pvParameters);
void task_Vidro_Traseiro_Esquerdo(void *pvParameters);

TaskHandle_t Vidro_Dianteiro_Direito;
TaskHandle_t Vidro_Dianteiro_Esquerdo;
TaskHandle_t Vidro_Traseiro_Direito;
TaskHandle_t Vidro_Traseiro_Esquerdo;
```

Fonte: Autoria própria.

Para processar as informações entregues pelos potenciômetros, foi necessário definir 4 parâmetros e 4 tarefas com os comandos sugeridos pela API do FreeRTOS.

3.5.4 Configuração do Setup

Na configuração do setup, deixamos tudo pronto para as configurações iniciais do Sistema, ou seja, quando o sistema é ligado todo comando que estiver dentro do setup vai ser executado apenas uma vez. Na programação do Sistema de Monitoramento no setup configurou-se inicialmente a taxa de envio de bits por segundo pelo comando “Serial.begin()”, informará também o usuário que o sistema está sendo inicializado. No setup também foi criado as tarefas, como exemplo, segue explicação da primeira tarefa que segue o mesmo modelo para as demais:

- xTaskCreate – comando para criação de tarefa;
- Task_Vidro_Dianteiro_Direito – lugar da locação da variável criada;
- Task1 – identificador da task;
- 156 – Locação de espaço na memória para manipulação da variável.
- NULL - Local para identificar parâmetros (nesse programa sem parâmetros a declarar);
- 1 – Prioridade de execução da tarefa (quanto maior o número maior a prioridade da tarefa);
- NULL - Local para definir identificador para tratar a tarefa (nesse programa a tarefa funcionara do mesmo modo por toda a rotina do software, assim não precisa ser declarada.

Na Figura 12 mostra toda a escrita do Código acima no Setup, e também mostras o comando “void loop()”, esse comando é muito comum em sistemas que as tarefas podem ser executadas uma de cada vez, porém como nesse sistemas as tarefas ocorrem de modo quase que simultaneamente, esse comando “void loop” fica em branco, pois tudo será feito pela “void Task”.

Figura 12 - Configuração do Setup

```
void setup()
{
  /*Inicializa serial*/
  Serial.begin(9600);

  /*Inicializa serial o LCD, liga black light e limpa o LCD */
  lcd.init();
  lcd.backlight();
  lcd.setCursor (1,0);
  lcd.print("INICIALIZANDO");
  lcd.setCursor (1,1);
  lcd.print("MONITORAMENTO");
  delay(5000);
  lcd.clear();

  xTaskCreate(Task_Vidro_Dianteiro_Direito, "Task1", 156, NULL, 4, NULL);
  xTaskCreate(Task_Vidro_Dianteiro_Esquerdo, "Task2", 156, NULL, 3, NULL);
  xTaskCreate(Task_Vidro_Traseiro_Direito, "Task3", 156, NULL, 2, NULL);
  xTaskCreate(Task_Vidro_Traseiro_Esquerdo, "Task4", 156, NULL, 1, NULL);

}
void loop() {
  /*Tudo é feito pelas tarefas, logo não precisa ter o programa aqui*/

}
```

Fonte: Autoria própria.

3.5.5 Criação da Rotina de Funcionamento

Nessa parte coloca-se em prática tudo que foi descrito no Diagrama de Blocos, porém agora em linguagem de programação na IDE do Arduino, como cada tarefa funciona de modo independente, elas são descritas separadas mais de modo semelhante seguindo os tópicos:

- void Task_Vidro_Dianteiro_Esquerdo(void *param) – chamar a tarefa a ser executada;
- (void) param; - definir as variáveis a serem processadas na tarefa;
- while (1) – definir todas as condições de processamento de dados e funcionamento da tarefa.

Na Figura 13 está demonstrado toda a rotina de funcionamento da tarefa, desde a leitura do sinal do potenciômetro, até a parte de interação com o usuário, que é a impressão de informação no Display de LCD, e a lógica de emissão de sinal sonoro e luminoso emitido pelo buzzer e pelo LED respectivamente.

Figura 13 - Código da Tarefa Monitoramento Vidro Dianteiro Esquerdo

```
/*TAREFAS DESCRITAS ABAIXO*/

void Task_Vidro_Dianteiro_Esquerdo(void *param)
{
  (void) param;
  int leitura_analogica_1 = 0;
  int abertura_porcento_1 = 0;
  pinMode(LED_DE, OUTPUT);
  pinMode(BUZZER_DE, OUTPUT);
  while (1)
  {
    leitura_analogica_1 = analogRead(ENTRADA_ANALOGICA_A0);
    abertura_porcento_1 = (leitura_analogica_1 / 6.8);

    lcd.setCursor (5, 0);
    lcd.print ("   ");
    lcd.setCursor (0, 0);
    lcd.print ("DE=");
    lcd.setCursor (3, 0);
    lcd.print(abertura_porcento_1);
    lcd.print ("%");
    if (abertura_porcento_1 <= 70)
    {
      digitalWrite(LED_DE, HIGH);
      digitalWrite(BUZZER_DE, HIGH);
      vTaskDelay(1000 / portTICK_PERIOD_MS);
      digitalWrite(LED_DE, LOW);
      digitalWrite(BUZZER_DE, LOW);
      vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    else {
      digitalWrite(LED_DE, LOW);
      digitalWrite(BUZZER_DE, LOW);
      vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
```

Fonte: Autoria própria.

3.5.6 Calibração do Sensor de Posicionamento dos Vidros.

Para fazer o cálculo de porcentagem de abertura dos vidros, foi utilizado uma biblioteca do Arduino que tem um programa já pronto, demonstrado na Figura 14,

que faz a leitura de sinal analógico A0 e mostra o valor no monitor serial do Arduino demonstrado na Figura 15.

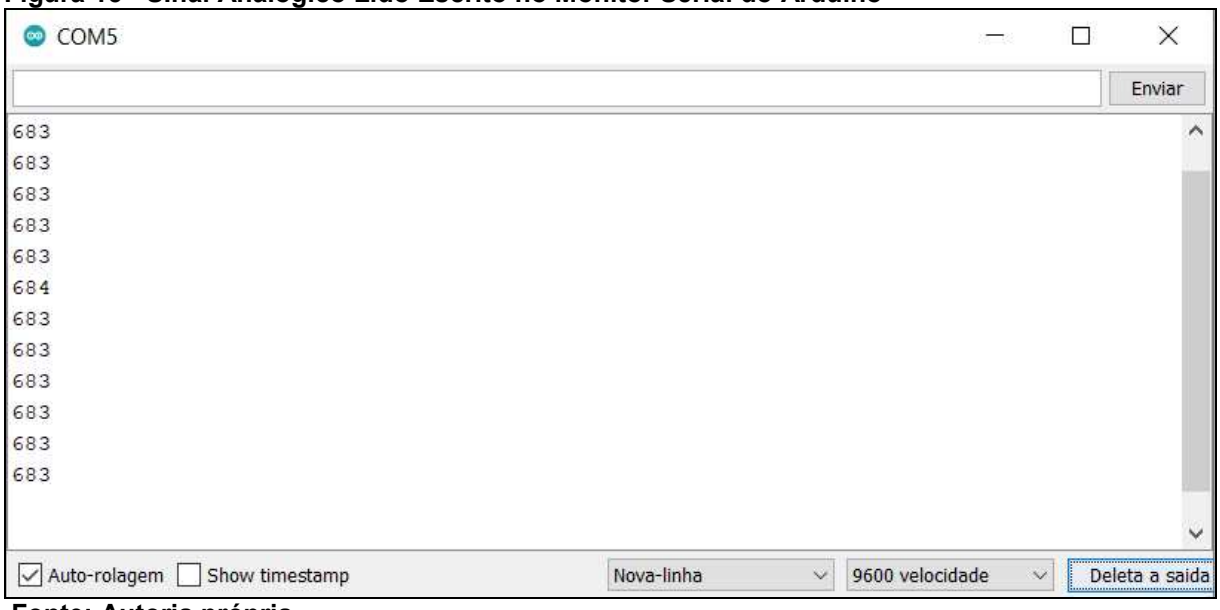
Figura 14 - Programa de Leitura Sinal Analógico da Biblioteca Arduino

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1000);        // delay in between reads for stability
}
```

Fonte: Arduino. CC (2021).

Figura 15 - SInal Analógico Lido Escrito no Monitor Serial do Arduino



Fonte: Autoria própria.

Verificando que o valor da leitura do sinal analógico e aproximadamente 680 para o cursor do potenciômetro na posição exemplificando o vidro do carro 100% fechado, no programa do Software foi criado a variável “abertura_porcento”, que é a o valor da leitura da variável “leitura_analogica” dividida por 6,8. Na figura 16 demonstra no detalhe a função de conversão de sinal analógico em porcentagem.

Figura 16 - Processamento do Sinal Analógico e Conversão em Porcentagem

```
leitura_analogica_1 = analogRead(ENTRADA_ANALOGICA_A0);
abertura_porcento_1 = (leitura_analogica_1 / 6.8);
```

Fonte: Autoria própria.

Com a calibração do sensor foi encerrado a elaboração do software que ficou pronto para teste de hardwares

3.6 VALIDAÇÃO DO PROGRAMA EM TESTE DE HARDWARE

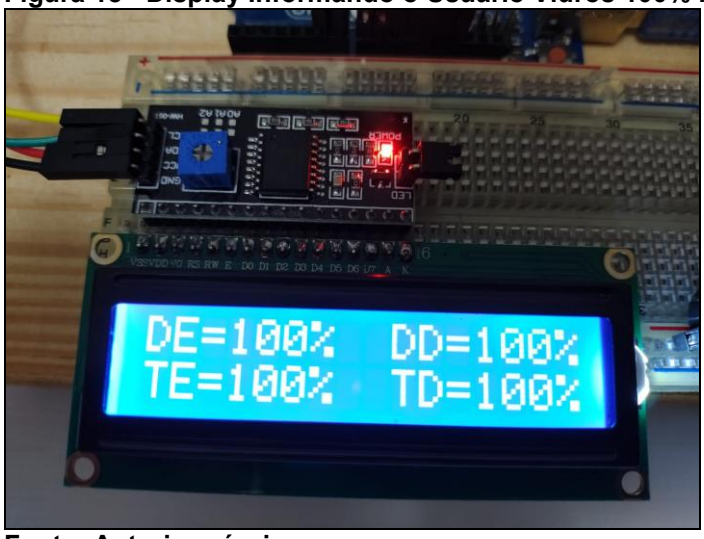
Ao alimentar a placa do Arduino o sistema começa a inicializar como mostrado na Figura 17.

Figura 17 - Display Informando ao Usuário Sistema Inicializando

Fonte: Autoria própria.

Após 5 segundos, o sistema mostra no display a atual posição dos vidros em porcentagem de fechamento como mostra a Figura 18.

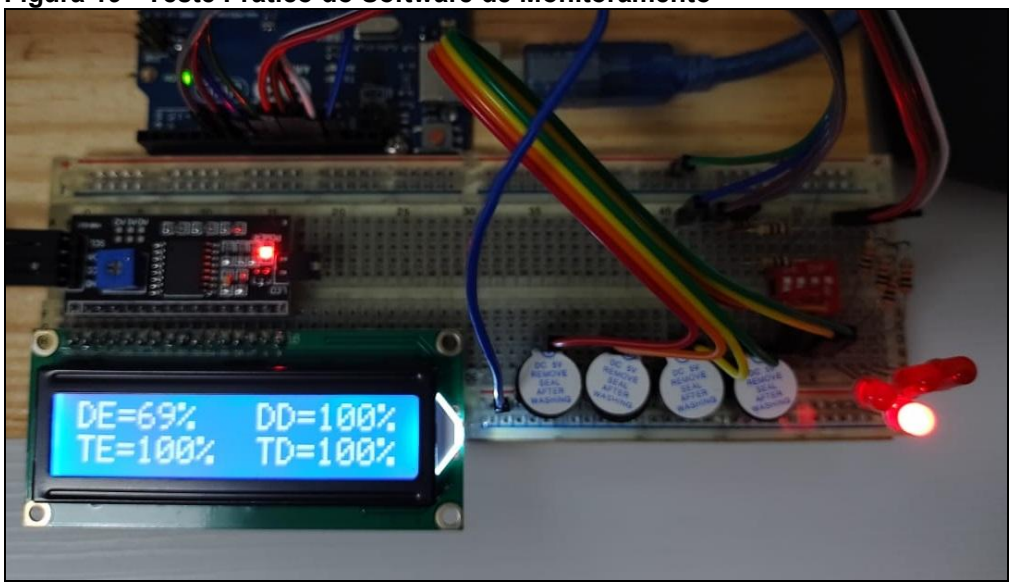
Figura 18 - Display Informando o Usuário Vidros 100% Fechados



Fonte: Autoria própria.

Para fazer o teste simulando o usuário abrindo o vidro dianteiro esquerdo, foi movimentado o cursor do potenciômetro ligado a entrada analógica A0 do Arduino, até o display informar um valor abaixo de 70% e o buzzer emitir um sinal sonoro como programado no software, na figura 19 demonstra que o LED emite um sinal luminoso informando a o usuário que o vidro dianteiro esquerdo já tem uma abertura razoável.

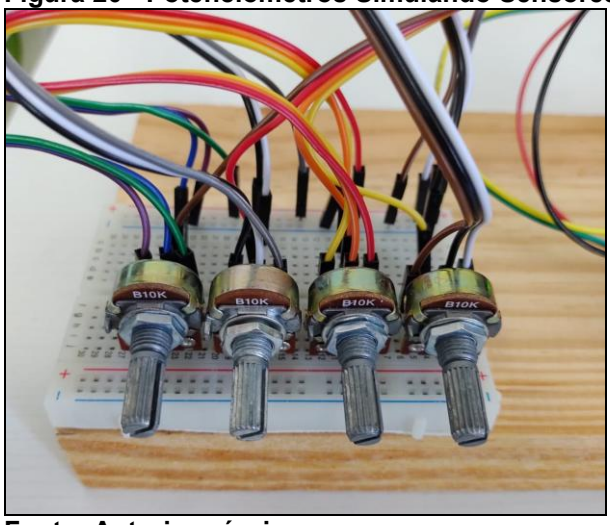
Figura 19 - Teste Prático do Software de Monitoramento



Fonte: Autoria própria.

Foram feitos testes atuando sobre todos os potenciômetros, representados na Figura 20 e o sistema teve funcionamento com forme o esperado.

Figura 20 - Potenciômetros Simulando Sensores



Fonte: Autoria própria.

Como teste final foi atuado o cursor de todos os potenciômetros até o display informar o valor de 0%, o sistema acendeu todos os LEDs e acionou todos os buzzers como era esperado, como mostra a Figura 21, assim finalizando a validação do Software Monitoramento de Abertura e Fechamento Vidro Automotivo.

Figura 21 - Teste Prático Final do Sistema

Fonte: Autoria própria.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Como resultados desse trabalho apresentados na seção 3 três citada anteriormente, os dados foram aplicados de maneira prática, assim do ponto de vista de validação do sistema, os resultados foram os melhores possíveis. As pesquisas executadas vieram de encontro a todo a carga de conhecimento para a satisfatória elaboração do projeto, e pode se dizer que depois de a construção dessa monografia estou bem mais preparado para atuar profissionalmente na área de sistemas embarcados.

5 CITAÇÕES

O FreeRTOS é ideal para aplicativos em tempo real profundamente integrados que usam microcontroladores ou pequenos microprocessadores. Este tipo de aplicação normalmente inclui uma mistura de requisitos de tempo real de hardware e software (BARRY, 2016, p. 2).

O Arduino UNO é a melhor placa para começar com eletrônica e codificação. Se esta é sua primeira experiência mexendo na plataforma, o UNO é a prancha mais robusta com a qual você pode começar a jogar. A UNO é a placa mais usada e documentada de toda a família Arduino (ARDUINO.CC, 2021, <<https://store.arduino.cc/products/arduino-uno-rev3/>>).

Não utilize mais a função delay() do Arduino e sim a função vTaskDelay() do freeRTOS, pois esta função trabalha em harmonia com o controle das tarefas, aproveitando esta espera para permitir que o código de outra task possa voltar a executar (BACK, 2019, p.36).

A linguagem C é uma linguagem de propósito geral, o que quer dizer que se adapta a praticamente qualquer tipo de projeto, altamente portável e extremamente rápida em tempo de execução. A linguagem C++ é uma evolução da linguagem C que incorpora orientação a objetos (CASAVELLA, 2021, <<http://linguagemc.com.br/breve-historia-da-linguagem-c/>>).

O conhecimento sobre teoria de sistemas de tempo real e sobre mecanismos de funcionamento de um sistema operacional de tempo real (RTOS) permitem ao projetista compreender com mais clareza como empregar corretamente o RTOS no projeto de sistemas embarcados multitarefas (DENARDIN, BARRICHELO, 2019, p. 370).

6 CONCLUSÃO

O problema levantado nesse trabalho foi concluído em partes, já que o foco dessa monografia era elaborar um Software de Monitoramento de Abertura e Fechamento Vidro Automotivo. O Objetivo geral e especifico desse trabalho que são elaborar um software de fácil manutenção e colocar em prática os conhecimentos adquiridos no curso de especialização foram compridos em sua grande parte, assim do ponto de vista de elaboração de software os resultados foram satisfatórios, já que em teste de bancada pode-se interagir com o sistema.

Para projetos futuros, poderá ser complementado esse projeto com novos periféricos, como sensor de temperatura. Também terá a possibilidade de em projetos futuros, fazer a pesquisa de parceiros pra contribuir com a instalação do sistema em um automóvel de fato.

REFERÊNCIAS

ARDUINO.CC. **Arduino UNO REV3.** Disponível em: <https://www.arduino.cc/>. Acesso em: 8 set. 2021.

BACK, Max. **Programando Multi-Tarefa na Prática:** Ultilizando linguagem C/C++ FREERTOS E ARDUINO. 2. ed, sem data.

BARRY, Richard. **Dominio do Kernel do FreeRTOS.** Guia prático. Disponível em: <https://www.freertos.org/Documentation/RTOS_book.html>. Acesso em: 10 out. 2021.

CASAVELLA, Eduardo. **Breve história da linguagem C**. Disponível em http://linguagemc.com.br/breve-historia-da-linguagem-c/. Acesso em: 28 out. 2021.

DENARDIN, Gustavo Weber; BARRIQUELLO, Carlos Henrique. **Sistemas Operacionais de Tempo Real e Sua Aplicação em Sistemas Embarcados**. 1. ed. São Paulo: Blucher, 2019.