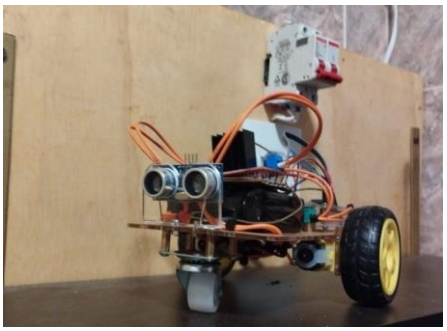

CIÊNCIA TECNOLOGIA E CULTURA NA PRAÇA

O USO DA TÉCNOLOGIA ARDUINO E O EMPREGO DE SENSORES EM UM DISPOSITIVO CONTROLADO PELO USUÁRIO

Cássio Antônio Prestes Galafassi
Denis Gonzaga
Fernando Martos Rezende de Campos Aguirre
Giovana Alves Pereira
Giovanni de Souza Gonçalves
José Aldair Alves dos Santos
Matheus Viacava Sigoli

1 Cássio A.P. Galafassi Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, cassio.galafassi@fatec.sp.gov.br
2 Denis Gonzaga Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, denis.gonzaga@fatec.sp.gov.br
3 Fernando M.RC. Aguirre Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, fernando.aguirre01@fatec.sp.gov.br
4 Giovana A. Pereira Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, giovana.santos29@fatec.sp.gov.br
5 Giovanni S. Gonçalves Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, giovanni.goncalves01@fatec.sp.gov.br
6 José A.A.Santos Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, jose.santos375@fatec.sp.gov.br
7 Matheus V. Sigoli Graduando em Tecnologia em Sistemas Embarcados, Fatec Jundiaí, matheus.sigoli@fatec.sp.gov.br

RESUMO:

Este projeto consiste na criação de um carrinho controlado remotamente por meio de um dispositivo móvel via Bluetooth, utilizando uma placa Arduino como o cérebro do sistema. O objetivo é criar um veículo de pequeno porte que possa ser controlado de forma precisa e conveniente a partir de um smartphone ou tablet.Podem ser adicionados outros componentes ao carrinho, como sensores ultrassônicos para evitar obstáculos ou verificadores de linha para seguir uma trajetória. Como um controle será utilizado o aplicativo Dabble que é próprio para a interpretação do modelo UNO e possui várias funcionalidades como console e controle.Será introduzido conceitos como Arduino e sistemas embarcados, além de mostrar quais foram os materiais utilizados na confecção do carrinho, a sua montagem, a sua programação e os problemas encontrados durante os testes realizados.

PALAVRAS-CHAVE: Carrinho; controle remoto; Arduino; componentes eletrônicos; programação.

INTRODUÇÃO

Inicialmente precisa-se entender o que é Arduino UNO, uma placa de desenvolvimento baseada em um microcontrolador que combina entradas e saídas digitais e analógicas para permitir a criação de projetos eletrônicos. Ele é programado usando principalmente a linguagem C/C++ fornecendo uma plataforma versátil para a prototipagem e desenvolvimento de sistemas embarcados de forma acessível.

Utilizando o Arduino, juntamente com componentes eletrônicos adicionais que serão detalhados posteriormente neste artigo, é possível conceber um sistema embarcado. Este sistema é caracterizado por sua capacidade de executar uma função específica.

MATERIAL E MÉTODOS

Nessa seção, serão descritos os materiais, procedimentos e métodos utilizados para conduzir o projeto com o objetivo de possibilitar a replicação do mesmo por outros.

Componentes utilizados no carrinho:

- 1 Kit Chassi Acrílico de Carro com 2 motores DC, 3 Rodas e 1 Interruptor
- 1 Módulo Duplo Ponte H L298n
- 1 Arduino UNO
- 1 Bateria 12V
- Terminais Tubulares 0,5mm
- Cabos 0,5mm
- Jumpers Femea
- Jumpers Macho
- 3 Conectores Macho P4 Cftv
- 2 Conectores Femea P4 Cftv
- 1 Módulo Bluetooth HM 10

Itens necessários:

- SmartPhone
- Aplicativo Arduino ESP Bluetooth – Dabble
- IDE Arduino
- Notebook ou computador pessoal
- Ferramentas gerais

Imagens dos componentes:

Imagem 1- Chassi

Imagem 2 – Ponte H

Imagem 3 – Arduino

Imagem 4 – Bateria

Imagem 5 - Terminal tubular

Imagem 6 - Conector P4 Cftv Macho

Imagem 7 - Conector P4 Cftv Femea

Imagem 8 - Módulo Bluetooth H10

Imagem 9 - Jumper Macho

Imagem 10 - Jumper Femea

Imagem 11 – Cabo 0,5mm

Confecção do carrinho:

Inicia-se a montagem encaixando as rodas ligadas aos motores e o botão para interromper o fluxo de energia nos encaixes específicos recortados na placa de acrílico de acordo com a imagem abaixo:

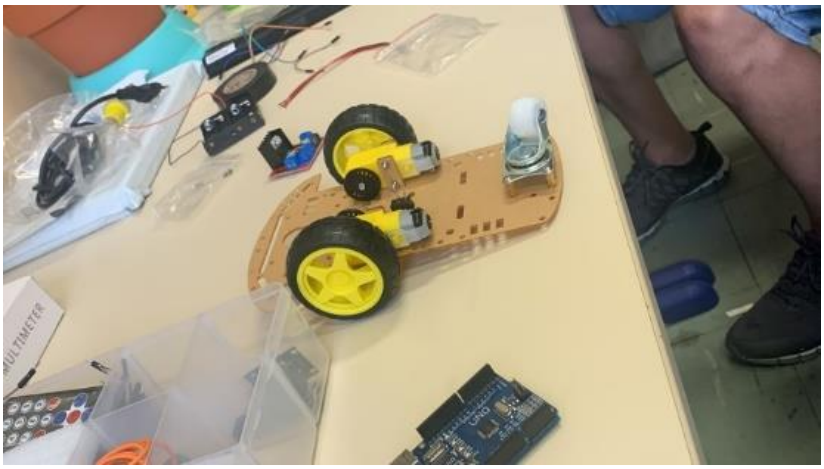

Imagem 12 – Chassi montado

Após a montagem na base de acrilico foi implementado o esquema abaixo para a ligação da Ponte H aos motores, foi utilizado o componente joystick para Arduino a fim de calibrar a direção das rodas.

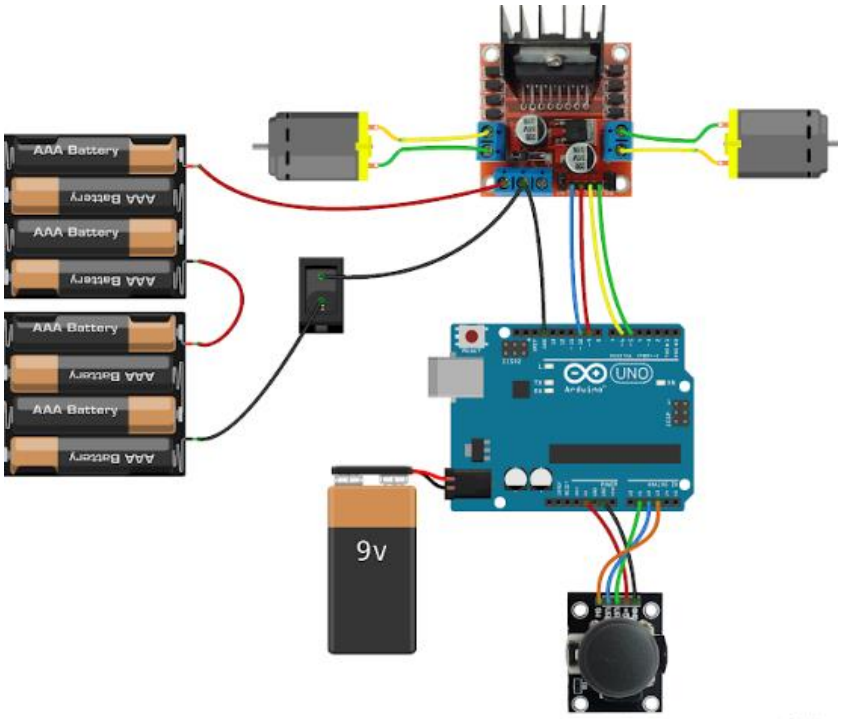

Imagem 13 - Esquema para controlar o carrinho com Joystyck

Foi utilizado o módulo “Ponte H” - para realizar o controle de tensão e a movimentação de “para frente e para trás”, também foi instalada uma fonte de alimentação de 12V no carrinho para a realização dos testes. Após a calibragem das rodas foi realizada uma tentativa de utilizar o componente Wireless Nrf24l01, porém o modelo adquirido se mostrou ser um componente “arcaico” e após vários problemas de conexão encontrados ele foi substituído pelo componente Bluetooth H10.

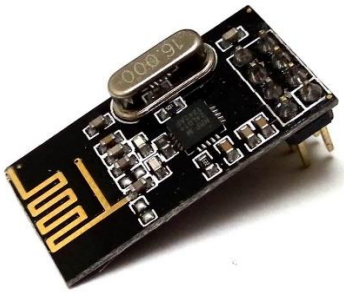

Imagem 14 – Placa Wireless Nrf24l01

Imagem 15 – Componente Bluetooth H10

Foi utilizado o esquema de montagem abaixo para a realização da conexão Bluetooth com o Arduino:

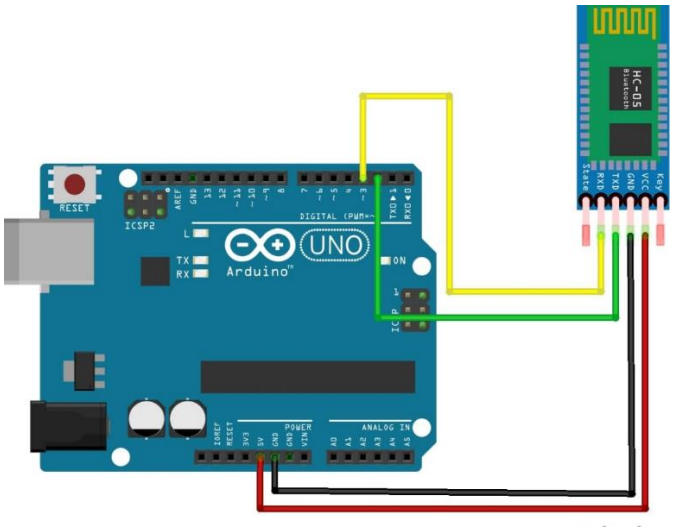

Imagem 16 – Esquema Módulo Bluetooth H10

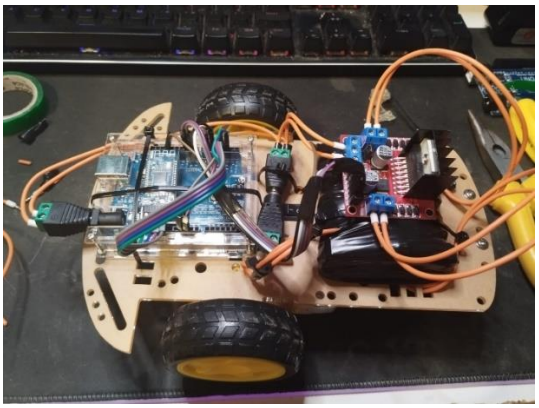

Imagem 17 – Carrinho já montado

Com o carrinho pronto e a placa de conexão instalada demos início aos testes utilizando o controle no modo Joystick do aplicativo Dabble, utilizando a Biblioteca <Dabble.h> para a comunicação.

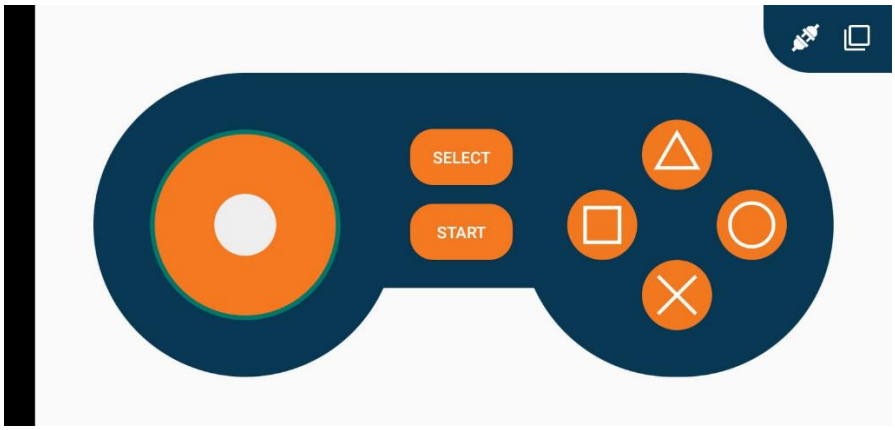

Imagem 18 – Tela do controle com Joystick via celular

Após a montagem do carrinho e com o resultado satisfatório dos testes realizados foi dado início a instalação dos componentes periféricos com diversas funcionalidades. O primeiro componente a ser instalado foi o Buzzer, instalado em uma entrada digital no Arduino a fim de funcionar como uma buzina do carrinho respondendo a o botão triangulo do controle utilizado.

Imagem 19 – Componente Buzzer

O segundo componente instalado foi o sensor ultrassônico, o qual consegue verificar pequenas distancias, para que o próprio carrinho evite uma colisão frontal ao detectar um objeto na sua frente.

Imagem 20 – Sensor ultrassônico

O último componente a ser adicionado foi o módulo de sensor de gás inflamável para demostrar um exemplo de utilidade em situação real na qual o carrinho pode ser utilizado em dutos estreitos a fim de se verificar a presença de gás.

Imagem 21 – Sensor de gas inflamavel

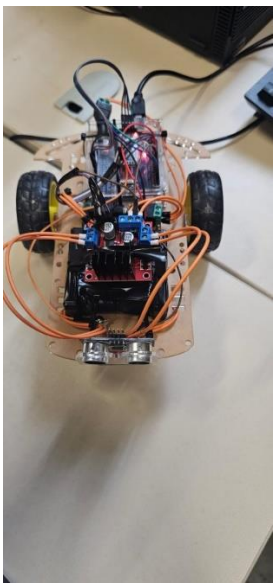

Imagem 22 – Carrinho Completo

Codificação do carrinho

Com o nosso projeto já montado vamos para a parte de codificação. Para fazer o carrinho rodar, criamos esse código dentro da IDE do Arduino.

Esse código inclui a biblioteca Dabble e a biblioteca ultrassônico no código. A primeira permite a comunicação entre um dispositivo móvel e o Arduino via Bluetooth. A segunda é usada para controlar sensores ultrassônicos, que são dispositivos que emitem pulsos sonoros e medem o tempo que leva para esses pulsos serem refletidos de volta para o sensor. Isso permite medir distâncias com precisão.

```
1   #include <Dabble.h>
2
3   //Sensor Sonico
4   #include <Ultrasonic.h>
5
6   //baixe e instale a biblioteca
7   //https://github.com/MakerHero/Ultrasonic
8
```

Imagem 23 – Código para instalar e incluir as bibliotecas necessárias

Este código é um exemplo de um programa Arduino que configura algumas constantes, define pinos para sensores, motores e outros componentes, além de incluir módulos da biblioteca Dabble para facilitar a comunicação entre o Arduino e um dispositivo móvel, como um smartphone ou tablet.

```
9   #define pino_trigger 11
10  #define pino_echo 7
11  Ultrasonic ultrasonic(pino_trigger, pino_echo);
12  #define pinIN1 10
13  #define pinIN2 9
14  #define pinIN3 6
15  #define pinIN4 5
16  #define Buzina 8
17  #define CUSTOM_SETTINGS
18  #define INCLUDE_GAMEPAD_MODULE
19  #define SensorGas A0
20
```

Imagem 24 – Código para definir os pinos que serão utilizados

Este código em Arduino defini algumas variáveis e constantes para serem usadas posteriormente no programa.

```
21  int StatusLed = 0;
22  int pDireita  = 100;
23  int pEsquerda = 100;
24  boolean SensorON = false;
25  boolean botaoAntSensor = HIGH;
26  boolean botaoSensor   = HIGH;
27  boolean botaoAnt = HIGH;
28  boolean botao    = HIGH;
29  boolean botao1 = false;
```

Imagem 25 – Código para declarar as variaveis que vamos usar

Esse código faz parte do bloco setup() em um programa Arduino. A função setup() é executada uma vez quando o Arduino é inicializado e é usada para configurar os pinos, portas e outros componentes de saída ligados ao Arduino.

```
31  void setup() {
32    pinMode(pinIN1, OUTPUT);
33    pinMode(pinIN2, OUTPUT);
34    pinMode(pinIN3, OUTPUT);
35    pinMode(pinIN4, OUTPUT);
36    pinMode(Buzina, OUTPUT);
37    pinMode(SensorGas, INPUT);
38    Dabble.begin(9600);
39
40  }
41
```

Imagem 26 – Código para configurar os pinos no arduino

Este código faz parte da função loop() em um programa Arduino. A função loop() é executada continuamente após a inicialização do Arduino e é usada para implementar o comportamento principal do programa.

```
43  void loop()
44  {
45      Dabble.processInput();
46      if (GamePad.isStartPressed())
47      {
48        botao = 1;
49        digitalWrite(Buzina, HIGH);
50        delay(100);
51        digitalWrite(Buzina, LOW);
52      }
53      else
54      {
55        botao = 0;
56      }
57      if (GamePad.isSquarePressed())
58      {
59      }
60
```

Imagem 27 – Condição para iniciar o GamePad no aplicativo

Esse trecho de código é usado para detectar mudanças no valor da variável botão em comparação com a iteração anterior. Se botão for verdadeiro e diferente do valor anterior (botaoAnt), a variável botao1 é alternada (inverte o valor) e a variável botaoAnt é atualizada. Isso é útil para identificar quando um botão foi pressionado ou liberado e executar ações com base nessa detecção de mudança de estado.

```
61    if (botao && (botao != botaoAnt))
62    {
63      botao1 = !botao1;
64    }
65    botaoAnt = botao;
```

Imagem 28 – Condição para salvar o estado do botão

Em resumo, esse trecho de código ativa (liga) quatro dispositivos ou componentes conectados aos pinos pinIN1, pinIN2, pinIN3 e pinIN4 quando a variável botao1 é verdadeira. O que esses dispositivos ou componentes fazem depende da aplicação específica do código e do hardware ao qual estão conectados. Por exemplo, esses pinos podem estar ligados a motores, LEDs ou outras cargas que devem ser ativadas quando botao1 é verdadeira.

```
67    if (botao1)
68    {
69        digitalWrite(pinIN1, HIGH);
70        digitalWrite(pinIN2, HIGH);
71        digitalWrite(pinIN3, HIGH);
72        digitalWrite(pinIN4, HIGH);
73    }
```

Imagem 29 – Condição para ativar os botões

Esse código está relacionado ao processamento do ângulo ou orientação obtidos do controle do jogo do aplicativo Dabble e faz ajustes para garantir que o valor resultante esteja dentro de um intervalo específico. O valor ajustado é então armazenado na variável pot2 para uso posterior no programa.

```
74    else
75    {
76      int CALCULO = GamePad.getAngle();
77      CALCULO = CALCULO - 270;
78      if (CALCULO < 0)
79      {
80        CALCULO = CALCULO + 360;
81      }
82      int pot2 = CALCULO;
```

Imagem 30 – Condição para pegar o ângulo do Joystick

Este código responde às ações do controle do jogo no aplicativo Dabble. Ele ajusta a variável pot1 com base na posição do joystick, gera tons no dispositivo conectado ao pino Buzina quando os botões "Triangle" e "Circle" são pressionados e atualiza a variável botaoSensor de acordo com o estado do botão "Circle". As ações específicas que ocorrem quando esses eventos são acionados dependem da aplicação e do uso pretendido do código.

```
83      int pot1 = GamePad.getRadius();
84      if (GamePad.isTrianglePressed())
85      {
86        tone(Buzina, 3000, 100);
87      }
88      if (GamePad.isCirclePressed())
89      {
90        botaoSensor = 1;
91        tone(Buzina, 3000, 500);
92      }
93      else
94      {
95        botaoSensor = 0;
96      }
```

Imagem 31 – Código para ativar o buzzer

Esse código controla a ativação do sensor de gás e a geração de um tom sonoro com base nas mudanças no valor de botaoSensor. Se botaoSensor for pressionado e diferente do valor anterior (botaoAntSensor), a variável SensorON é alternada. Se SensorON for verdadeira, ele lê o sensor de gás e, se o valor da leitura estiver acima de 60, gera um tom na buzina. Esse código é usado para ativar o sensor de gás e emitir um alerta sonoro quando uma condição específica é atendida.

```
97      if (botaoSensor && (botaoSensor != botaoAntSensor))
98      {
99        SensorON = !SensorON;
100     }
101     botaoAntSensor = botaoSensor;
102     if(SensorON == true)
103     {
104       int AnalogSensorGas = analogRead(SensorGas);
105
106       if (AnalogSensorGas > 60)
107       {
108         tone(Buzina, 3000, 100);
109       }
110     }
111
```

Imagem 32 – Código para saber se o sensor de gas é ligado

Este código ajusta as variáveis pDireita e pEsquerda para controlar a velocidade dos motores de um dispositivo (como um veículo) com base no valor de pot2. Quando pot2 está na faixa de 0 a 180, a esquerda é controlada, reduzindo a velocidade à medida que pot2 diminui. Quando pot2 está na faixa de 180 a 360, a direita é controlada, reduzindo a velocidade à medida que pot2 aumenta. Isso pode ser usado para controlar o movimento do dispositivo para a esquerda e para a direita com base na posição do potenciômetro ou sensor que fornece o valor de pot2.

```
113     if (pot2 <= 180) {
114       //Esquerda
115       pDireita  = 100;
116       pEsquerda = map(pot2, 180, 0, 100, 0);
117     }
118     else {
119       //Direita
120       pDireita  = map(pot2, 180, 360, 100, 0);
121       pEsquerda = 100;
122     }
```

Imagem 33 – Código para capturar a posição do dedo do Joystick para controlar o carrinho

Se o botão "Cross" é pressionado, este código ajusta a velocidade dos motores com base na variável pot1 e inverte o sentido de rotação dos motores conectados aos pinos pinIN2 e pinIN4. Isso pode ser usado para controlar o movimento de um dispositivo, como um veículo, com base na posição de pot1 e ação do botão "Cross".

```
124     if (GamePad.isCrossPressed())
125     {
126       //inverte os motores
127       int velocidade = map(pot1, 0, 7, 0, 255);
128
129       analogWrite(pinIN1, LOW);
130       analogWrite(pinIN2, velocidade * pDireita / 100);
131
132       analogWrite(pinIN3, LOW);
133       analogWrite(pinIN4, velocidade * pEsquerda / 100);
134     }
```

Imagem 34 – Código para converter a velocidade e a direção do carrinho na direção correta

Este código controla o movimento do dispositivo com base na distância medida pelo sensor ultrassônico e na posição do potenciômetro pot1. Se a distância medida for menor que 25 cm, o dispositivo avança. Caso contrário, a velocidade dos motores é controlada pelo potenciômetro. O código é usado para permitir que o dispositivo se mova para a frente quando não há obstáculos próximos e controle manualmente a velocidade com base no potenciômetro quando necessário.

```
135       else
136       {
137         //Sensor Sonico
138         float cmMsec;
139         long microsec = ultrasonic.timing();
140         cmMsec = ultrasonic.convert(microsec, Ultrasonic::CM);
141
142          //Para frente
143          if(cmMsec < 25)
144          {
145            digitalWrite(pinIN1, HIGH);
146            digitalWrite(pinIN2, HIGH);
147            digitalWrite(pinIN3, HIGH);
148            digitalWrite(pinIN4, HIGH);
149          }
150          else{
151            int velocidade = map(pot1, 0, 7, 0, 255);
152
153            analogWrite(pinIN1, velocidade * pDireita / 100);
154            analogWrite(pinIN2, LOW);
155
156            analogWrite(pinIN3, velocidade * pEsquerda / 100);
157            analogWrite(pinIN4, LOW);
158          }
159       }
160     }
161   }
```

Imagem 35 – Código do sensor ultrassônico

O código completo atualizado se encontra disponível a no seguinte link no GitHub:

https://github.com/yokoshen/Arduino-Carrinho-Code

Para carregar o código no Arduino é necessário configurar a placa e a porta de comunicação. Desse modo, conecte o Arduino à USB do seu computador, em seguida acesse o menu Ferramentas e busque o modelo da sua placa Arduino. Em nosso exemplo usaremos a placa Arduino UNO, conforme a imagem.

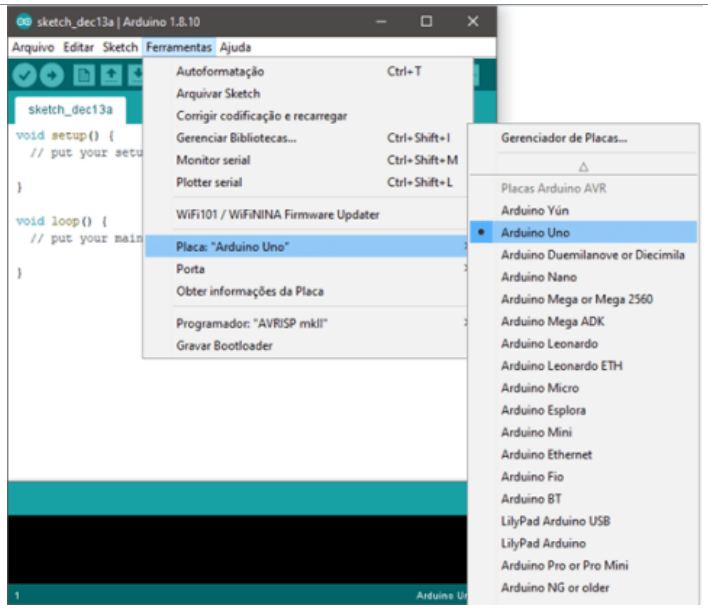

Imagem 36 – Seleção da placa Arduino.

No exemplo abaixo a placa Arduino UNO foi reconhecida com sucesso pela porta COM de número 5.

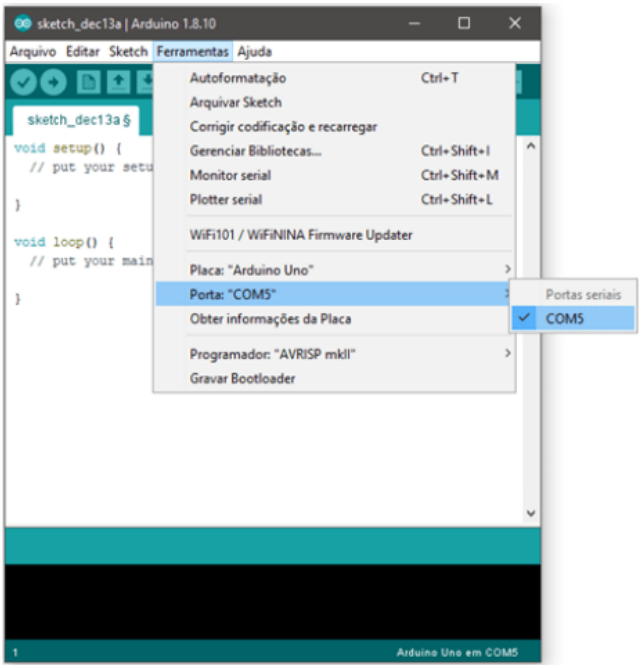

Imagem 37 – Seleção da porta COM.

Feito isso, é necessário baixar todas as bibliotecas utilizadas no código seguindo os seguintes passos:

Clicar em Sketch, incluir biblioteca e clicar em gerenciar bibliotecas.

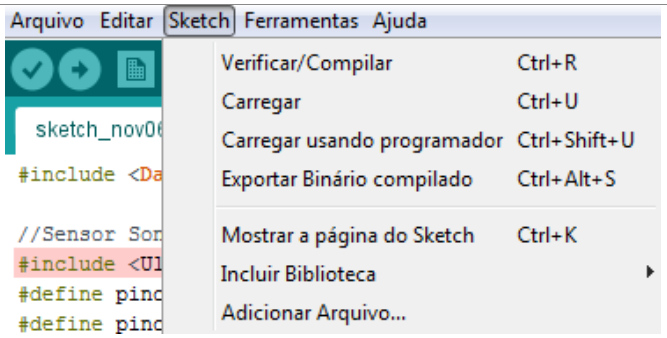

Imagem 38 – Sketch

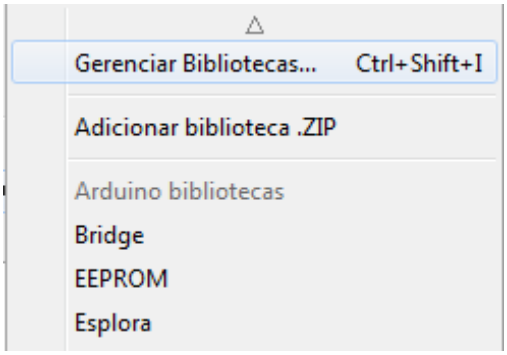

Imagem 39 – Gerenciar bibliotecas

Vai assim abrir uma janela para pesquisar e instalar as bibliotecas:

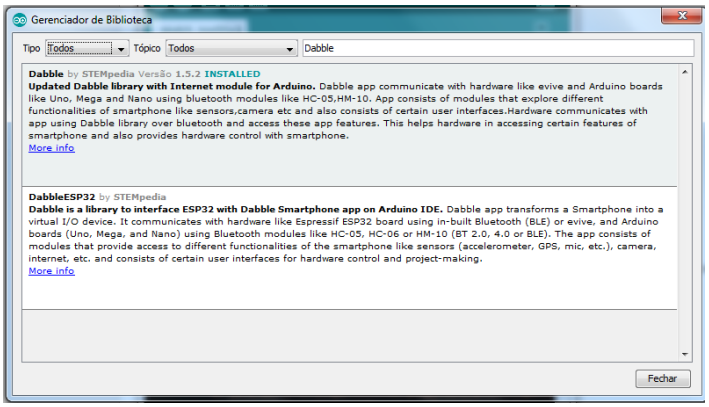

Imagem 40 – Bibliotecas

Em seguida carregue o código no Arduino, conectando o Arduino no computador utilizando um cabo USB, conforme a imagem abaixo.

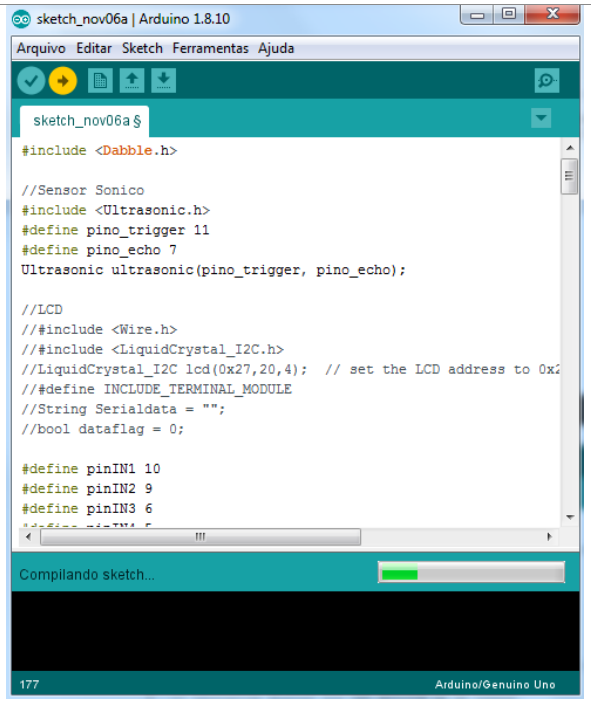

Imagem 41 – Carregar o arquivo

A transferência do código demorará alguns segundos, mas, logo em seguida, o LED ligado ao pino 13 começará a piscar em intervalos de 1 segundo.

RESULTADOS E DISCUSSÃO

Será aprofundado os detalhes de testes realizados:

Carrinho ligado com o componente Joystick

Imagem 42 – Componente Joystick

Teste: Utilização do Componente Joystick no Carrinho

A aplicação do joystick mostrou-se eficaz, permitindo assim identificação e ajuste da calibração do motor. O desempenho do mesmo atendeu a expectativa esperada, validando assim a decisão de incluir essa opção no aplicativo.

Carrinho com Componente Wi-Fi

Imagem 43 – Componente Wi-fi

Teste: Implementação do Componente Wireless

A tentativa de integração da peça Wi-Fi não obteve sucesso, resultando na incapacidade do carrinho em receber comandos.
Para identificar as possíveis causas deste problema, foi realizado testes em ambas as peças WiFi, verificando as antenas e testando a conectividade com a rede de internet. Além disso, a aplicação do mesmo código em ambos os componentes para verificar paridade.
No entanto, após uma análise aprofundada, foi identificado que a placa Wi-Fi em uso era obsoleta e apresentava limitações em termos de programação e capacidade de conectividade, o que explicou a falha na comunicação apresentada anteriormente.
Diante disso, foi feito a substituição para o módulo Bluetooth, devido à sua praticidade e à capacidade de atender as necessidades do projeto de forma eficiente.

Carrinho com o módulo Bluetooth

Imagem 44 – Componente Bluetooth

Teste: Componente Bluetooth.

Após substituir o módulo Wi-Fi pelo Bluetooth, foi observado um desempenho satisfatório, que permitiu iniciar os testes com o aplicativo móvel Dabble, tornando o controle do carrinho mais otimizado.
O sucesso na operação com o Bluetooth possibilitou uma transição facilitada na integração de outros componentes, incluindo o sensor de proximidade e o sensor de detecção de gás, evidenciando a eficácia da solução adotada.

CONCLUSÕES

Em resumo, a realização deste projeto proporcionou uma valiosa experiência de aprendizado e inovação. Integrando de um Arduino, um módulo Bluetooth e motores de corrente contínua, foi capaz de criar um veículo de pequeno porte controlado remotamente por meio de um aplicativo, possibilitando a exploração de conceitos fundamentais de eletrônica, programação e comunicação sem fio.

Embora enfrentando desafios na tentativa de implementar um componente Wi-Fi, o sucesso na utilização do módulo Bluetooth permitiu alcançar resultados satisfatórios. O carrinho se moveu de forma consistente, realizando curvas e movimentos para a frente e para trás, com o celular atuando como o controle remoto eficiente.

Este projeto ilustra a importância da flexibilidade e adaptação na engenharia e na tecnologia. Muitas vezes é necessário revisar a abordagem e adotar soluções alternativas para alcançar os objetivos. Em última análise, essa experiência nos auxiliou na aprimoração de novas habilidades e técnicas e a compreender a importância da resiliência e da criatividade na resolução de desafios tecnológicos.

REFERÊNCIAS

CÓDIGO ATUALIZADO. **GitHub**. Disponível em: https://github.com/yokoshen/Arduino-Carrinho-Code. Acesso em: 08 nov. 2023.

ESQUEMA DE LIGAÇÃO. **Google Drive**. Disponível em: https://drive.google.com/file/d/0B91L9po09u35TVozeDlfcUI0Mms/view?resourcekey=0-DuZB_ajdXDhw2pM8jK8uDw. Acesso em: 15 out. 2023.

MATERIAL PARA CONFIGURAÇÃO DO ARDUINO. **Arduino.** Disponível em: https://www.arduino.cc/reference. Acesso em: 08 nov. 2023.

VIANA, Carol Correia. Como programar o Arduino. **Blog da robótica**. Disponível em:https://www.blogdarobotica.com/2020/09/22/como-programar-o-arduino/#:~:text=Para%20carregar%20o%20c%C3%B3digo%20no,UNO%2C%20conforme%20a%20Figura%202.. Acesso em: 10 out. 2023.