UNIVERSIDADE FEDERAL DE MINAS GERAIS

Relatório Final

Nome: Alice Aguiar, Breno Miguel, Lucas Augusto, Lucas Martins
Subsistema: Eletrônica
Data:09/01/2021

1. RESUMO

Esse relatório é referente ao projeto proposto aos candidatos que chegaram ao trainee do processo seletivo da Fórmula Tesla UFMG. Para nossa equipe foi proposto a confecção de uma Placa de Circuito Impresso (PCB) em forma de um shield para arduino que tem como função o teste de 4 protocolos de comunicação, UART, I2C, SPI e CAN. Inicialmente esse projeto era para ser executado presencialmente, entretanto, com a pandemia fomos forçados a fazer cada um em sua casa e utilizar ferramentas online que simularam o comportamento de um arduino para assim conseguir finalizarmos o projeto. Todos os sites e softwares utilizados serão descritos ao longo deste relatório para um melhor entendimento de como foi todo o processo da confecção do projeto da PCB.

2. INTRODUÇÃO

A definição de protocolos de comunicação segundo o Grupo de Teleinformática e Automação (GTA) da UFRJ [1]:

> "Conjunto de regras, padrões e especificações técnicas que regulam a transmissão de dados entre computadores por meio de programas específicos, permitindo a detecção e correção de erros; protocolo de transmissão de dados".

Os protocolos que iremos tratar neste relatório serão, UART, I2C, SPI e CAN. Cada um desses protocolos tem um desempenho melhor para um determinado objetivo, iremos descrever o funcionamento de cada um desses protocolos de forma breve logo a baixo.

2.1. UART

UART é uma abreviação de Universal Asynchronous Receiver / Transmitter. É um protocolo *full-duplex* e possui uma comunicação serial, ou seja, envia a informação sequencialmente, consequentemente é necessário um menor número de fios para fazer esse tipo de comunicação. Como o próprio nome diz é uma comunicação assíncrona, não precisa de um clock para ocorrer a comunicação, é necessário apenas que os dois sistemas

computacionais que estão se comunicando estejam configurados com a mesma taxa de transferência de bits. Pode ser aplicado em receptores GPS, Arduino, Módulos Bluetooth, Modems GSM e GPRS, Sistemas de Comunicação sem fio, dentre outros.

2.2. I2C

O protocolo de comunicação I2C é half-duplex, pois contém apenas um pino para envio de dados, e síncrono, pois usa um pino de clock. Seu funcionamento é dado pelo dispositivo mestre que deve informar aos dispositivos escravos o início da comunicação ou *start condition*, em seguida, todos os escravos estarão prontos para receber a primeira informação que é o endereço do escravo que comunicará com o mestre, junto com a operação que este escravo desempenhará. A comunicação se dará até que o dispositivo mestre envie uma *stop condition* para interromper a comunicação. Suas aplicações é a comunicação entre circuitos integrados, sejam eles microcontroladores, sensores (acelerômetros, GPS, etc…).

2.3. SPI

Serial Peripheral Interface ou SPI é um protocolo de comunicação full-duplex e síncrono. Ele funciona de modo que o mestre gera um clock e seleciona através do pino SS com qual dispositivo será efetuada a comunicação. Em seguida os dados são enviados para o dispositivo de destino pelo pino MOSI e então o dispositivo escravo envia uma resposta (se necessário) ao mestre pelo pino MISO. É utilizado em LCD e LED, memórias flash e EEPROM, para a comunicação entre diferentes periféricos como conversor analógico digital (ADC) e conversor digital analógico (DAC).

2.4. CAN

CAN é um acrônimo para Controller Area Network e é um protocolo half-duplex e síncrono. Seu funcionamento é baseado no conceito multi-mestre, no qual todos os módulos podem se tornar mestre em determinado momento e escravo em outro e suas mensagens são enviadas em regime multicast, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede. Além disso o CAN também exercita o conceito CSMA/CD with NDA (*Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration*), isto é: todos os módulos verificam o estado do barramento, analisando se outro módulo está ou não enviando mensagens com maior prioridade, para que o módulo de maior prioridade continue enviando sua mensagem deste ponto, sem ter que reiniciá-la. No CAN, os dados não são representados por bits em nível “0” ou nível “1”, ao

invés disso, são representados por bits Dominantes e bits Recessivos, criados em função da condição presente nos fios CAN_H e CAN_L. O CAN é utilizado em metrôs, ferrovias, máquinas raio X, além de diversas aplicações em veículos, tendo em vista que o CAN foi inicialmente criado com esse propósito.

3. PROJETO INICIAL

O projeto consiste em um testador de protocolos de comunicação que é um dispositivo que identifica falha ou sucesso na comunicação entre dois sistemas eletrônicos ou computacionais. Esse é subdividido em duas partes, a parte eletrônica, que consiste em uma placa de circuito impresso (PCB), projetada como um shield, que será acoplado no arduino uno e o código que será responsável por fazer o teste de comunicação dos arduinos por meio dos protocolos: UART, I2C, SPI e CAN.

Iniciamos o projeto desenvolvendo o protocolo UART pela sua simplicidade. Em seguida, foi desenvolvido o sistema de botões, que permitem escolher qual protocolo a ser testado e LEDs, as quais indicam qual protocolo está sendo testado, em prol de uma melhor interface para o usuário.

Durante a elaboração do projeto houveram mudanças no planejamento do produto final, as quais serão citadas a seguir. Uma LED verde seria acesa no caso de um sucesso na comunicação e uma LED vermelha seria acesa no caso do fracasso, essa ideia foi abandonada, pois a falha ou sucesso da comunicação depende da necessidade de cada usuário. Um exemplo desse caso dá-se se colocássemos como sucesso na comunicação se 70% dos dados fossem transmitidos corretamente, porém um possível usuário precisa de uma comunicação com, no mínimo, 90% de acerto. Por esse motivo, decidiu-se mostrar apenas essa taxa e deixar para o usuário decidir se a comunicação entre os arduinos foi efetiva.

4. CÓDIGOS

4.1. BOTÕES

Dentro da estrutura do código foram definidas algumas portas de input que correspondem aos botões, quando um botão é apertado ele ativa um protocolo, cada botão acionará um protocolo específico, e será feita a comunicação por meio deste protocolo.

4.2. UART

O código do UART foi o primeiro a ser desenvolvido e o objetivo inicial foi a

comunicação utilizando apenas um caractere. Posteriormente, a equipe decidiu que, para os códigos dos demais protocolos, o propósito primário era a comunicação de apenas um caractere. Para isso, no código do UART, decidiu-se utilizar LEDs para certificar o funcionamento do envio do caractere para o receptor e o envio de volta para o transmissor e a impressão no monitor serial para conferir com exatidão as mensagens enviadas. Depois de várias falhas no teste, um dos membros da equipe percebeu que a função utilizada para conferir a transmissão, *Serial.println,* sujava o buffer e impedia o funcionamento correto do programa. Para solucionar esse problema sem deixar de conferir a transmissão dos dados, utilizamos uma função para limpar o buffer sempre que usássemos a função *Serial.println* durante a comunicação.

Depois do sucesso da comunicação com um caractere, iniciou-se o desenvolvimento do código para *strings*, o que apresentou mais dificuldades que o previsto, pois a lógica que foi utilizada de conferir a transmissão no transmissor e no receptor, na fase anterior do código, foi bem complexo e desnecessário, visto que a checagem do sucesso da transmissão só precisa ser feita no transmissor. Essa foi a alteração mais significativa na lógica do programa nas duas diferentes fases de desenvolvimento.

4.3. I2C

O código correspondente ao arduino mestre tem como uma característica fundamental a utilização da biblioteca Wire.h, todas as funções utilizadas para a comunicação dependem dessa biblioteca. A comunicação tem início com este arduino enviando no caminho de dados o endereço correspondente ao escravo o qual ele quer se comunica, isso é de suma importância, visto que, pode haver mais de um mestre e mais de um escravo na comunicação I2C, logo em seguida ele fala se ele irá enviar ou receber uma mensagem. No nosso projeto ele envia uma mensagem, essa mensagem é lida pelo escravo por uma função que diferentemente dos outros protocolos não está definida dentro, mas fora do loop do código. Após o escravo ler a mensagem o mestre pede para que o escravo envie a mensagem recebida para ele e faz a comparação se a mensagem é igual ou diferente.

Inicialmente, a mensagem era apenas um caractere, com o desenvolvimento do código fizemos o envio de uma string. Durante esse desenvolvimento inicial a mensagem enviada e recebida era mostrada no monitor serial para vermos se a comunicação estava sendo efetiva. No código final o que será mostrado no monitor serial será a porcentagem correspondente ao sucesso da mensagem enviada e recebida.

Para destacar alguns elementos importantes do código, para o começo da transmissão é utilizada a função *Wire.beginTransmission(*endereço do escravo*)* e para o fim da *transmissão Wire.endTransmission(),* para o mestre pedir uma mensagem ao escravo *Wire.requestFrom()* essa função possui dois argumentos, o primeiro o endereço do escravo e o segundo o número de bits da mensagem e para enviar *Wire.write(*mensagem a ser enviada*)*. Agora para o escravo receber e enviar uma mensagem tivemos que criar essas funções, mas apenas lendo do caminho de dados com a função *Wire.read()* e enviando como a função *Wire.write(*mensagem a ser enviada*)* a mensagem como no mestre.

4.4. SPI

O desenvolvimento do código do SPI foi relativamente simples, depois que um dos membros da equipe adquiriu dois arduinos para o teste do programa, pois a plataforma utilizada para a elaboração dos outros códigos apresentou problemas com alguns elementos da biblioteca do SPI. Depois do estabelecimento da comunicação entre o mestre e o escravo, utilizou-se a função *SPI.transfer* para a troca de informação, a única diferença da primeira parte do projeto (transmissão de 1 caractere) para a segunda (transmissão de um string), foi o acréscimo de um loop para o envio sequencial dos caracteres da string.

4.5. CAN

O código do protocolo CAN foi o último a ser desenvolvido pela equipe, já que o consideramos mais complexo em diversos aspectos, sendo eles os principais: não obtenção do módulo MCP2515, que foi utilizado para implementar o CAN no arduino; procura das bibliotecas necessárias no GitHub; impossibilidade de compilar no TinkerCad. Visto isso, utilizamos a IDE do próprio arduino para fazer o código, o qual compila sem erros, entretanto não pudemos testar a lógica.

Neste protocolo, é possível enviar muitas mensagens ao mesmo tempo, logo o padrão é criar uma struct, que armazena as várias possibilidades de mensagens. Logo, tivemos que atribuir a cada posição da struct um endereço diferente, para que o escravo leia apenas a mensagem que lhe foi endereçada. Então, a função utilizada para o envio foi a *MCP2515.sendMessage(struc can)*, enviando a struct como parâmetro, e para leitura a função *MCP2515.readMessage(struc can)*, recebendo toda a struct. Primeiramente, como nos outros módulos, foi feito o código para o envio de apenas 1 caractere, mas posteriormente foi adicionado um loop para enviar uma string.

4.6. PORCENTAGEM

O cálculo da porcentagem de sucesso é muito simples pois é a mesma para todos os códigos, o código basicamente compara a string recebida com a string estabelecida no código do transmissor, caractere por caractere. Ao final dessa comparação, obtemos a porcentagem de sucesso com operações matemáticas triviais.

5. CIRCUITO

Inicialmente o circuito foi desenvolvido no tinkercad, projetado com 2 arduinos unos e uma protoboard de forma limitada. Não foi feita a ligação para a comunicação usando o CAN por três problemas. O programa não suportava as bibliotecas utilizadas por esse protocolo e não tinha o módulo necessário para a comunicação. Outro problema era que as portas do arduino utilizadas pelo SPI, também eram usadas pelo CAN, de forma que não era possível simular um único circuito. Mas a partir desse circuito preliminar foi montado o circuito final que seria usado para o projeto da PCB.

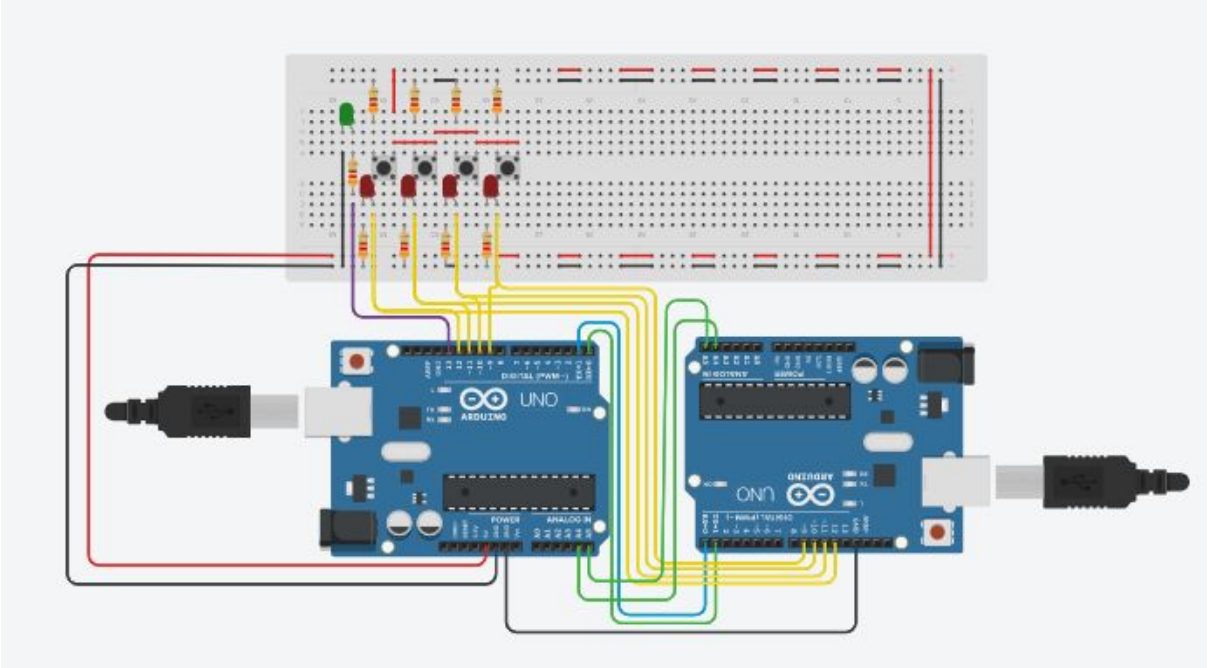

Imagem **1**: Circuito preliminar, feito no Tinkercad, utilizado para fazer testes.

5.1. TINKERCAD

O TinkerCad, inicialmente, foi o programa utilizado para simular os arduinos com os códigos de todos os protocolos. Porém, ao longo do projeto, descobriu-se alguns gargalos com a plataforma que atrasaram o desenvolvimento do testador de protocolos.

O primeiro desses problemas foi com o CAN, integrantes do grupo, enquanto pesquisavam sobre esse protocolo para facilitar o desenvolvimento do código posteriormente, perceberam que o TinkerCad não suporta o módulo MCP2515, o qual é necessário para o teste do CAN. Outras plataformas foram testadas, mas sem sucesso.

O problema seguinte foi bem inesperado, o que fez com que os integrantes que trabalhavam com o código SPI perdessem um tempo significativo para descobrir que o problema estava na plataforma que, teoricamente, suportava toda a biblioteca desse protocolo. O TinkerCad possui um problema com a função *SPI.transfer(),* de forma que o código sempre travava assim que chegava nessa função, não executando nada depois dela. Diversos códigos prontos foram testados, sem sucesso, até que foi achado em um fórum do próprio TinkerCad várias pessoas relatando o mesmo problema. A partir daí, o desenvolvimento do código do SPI foi interrompido, mas graças a um integrante que conseguiu dois Arduinos Uno, foi possível o teste deste protocolo.

5.2. FRITZING

O FRITZING é uma plataforma de desenvolvimento semelhante ao TinkerCad e foi utilizada a fim de resolver os problemas encontrados nesta. Contudo, depois de vários testes e estudos sobre a plataforma, descobriu-se que ela não simula um Arduino como o Tinkercad, mas permite que o usuário carregue o código em um arduino ligado ao seu computador. Portanto, a equipe não conseguiu tirar proveito desse programa.

6. PCB

Após todo o desenvolvimento do circuito, já mostrado neste relatório, ele foi adaptado para ser uma PCB utilizando a plataforma Easyeda.

6.1. EASYEDA

Esse programa tem como finalidade facilitar a execução de um projeto de uma PCB. Primeiramente é necessário montar o circuito da PCB a ser projetado e logo em seguida montar o circuito impresso em si. Para isso ele conta com vários recursos, bibliotecas e componentes. É possível consultar preço, datasheet dos componentes e até fazer a compra dos componentes e mandar fabricar a placa projetada por esse programa.

6.2. PROJETO DO CIRCUITO

Basicamente o projeto do circuito estava pronto no Tinkercad, foi feito apenas uma formalização desse circuito e a adição do módulo MCP2515 para a comunicação pelo CAN.

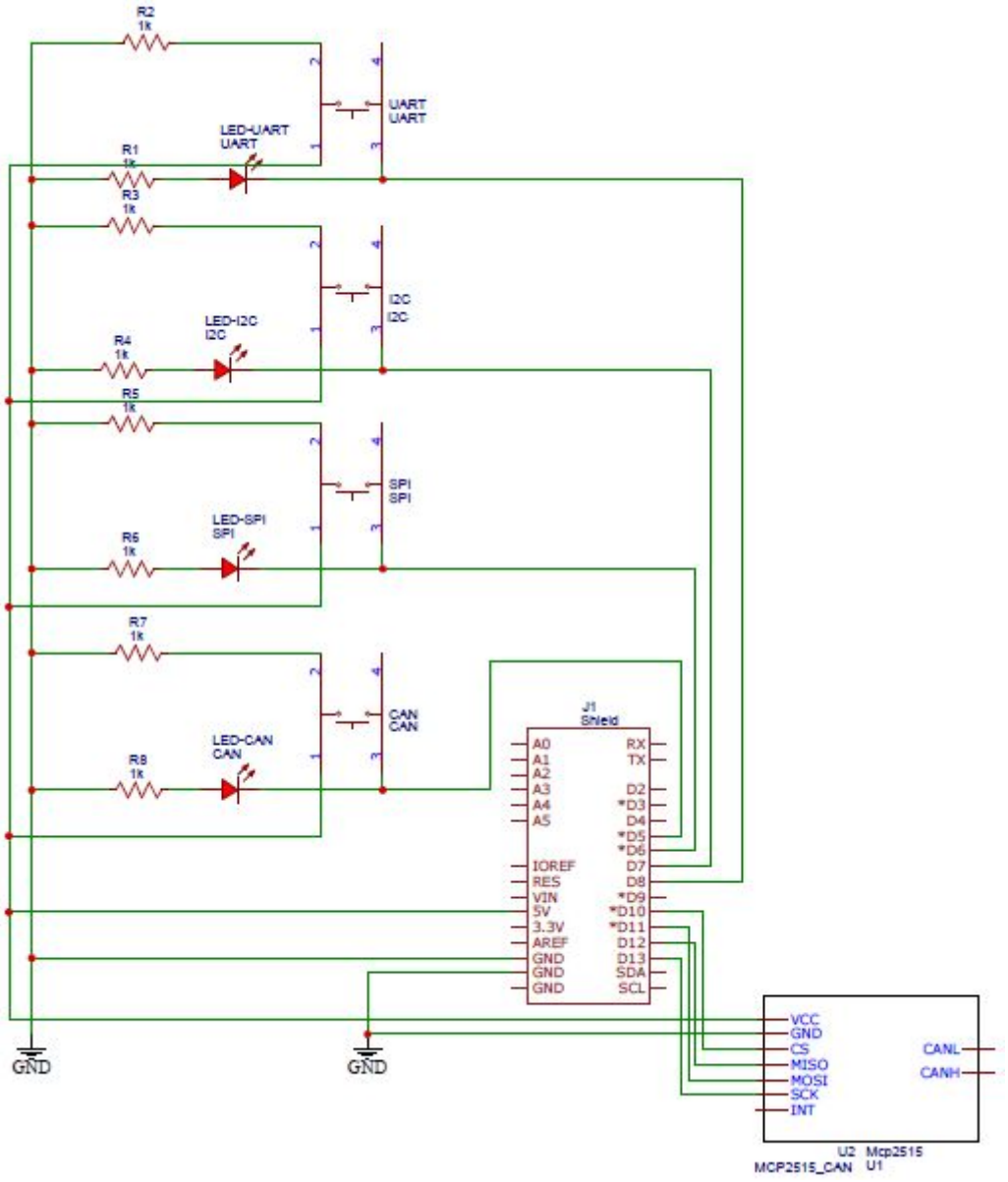

Imagem 2: Circuito utilizado para a montagem da PCB.

Para um correto funcionamento desse circuito, na PCB foram adicionadas algumas coisas que não estão representadas na imagem acima, como a conexão SPI, que foi feita colocando pinos do tipo fêmea, assim as portas de comunicação desse protocolos serão conectadas do mestre ao shield por um jumper. Mas esses detalhes serão tratados mais a frente.

6.3. PROJETO DA PCB

A PCB é a placa onde os componentes eletrônicos são organizados e conectados de acordo com o especificado no circuito, ela possui duas partes, a parte superior e inferior. Alguns componentes atravessam a placa de modo que eles conseguem ser conectados em ambas as partes, entretanto outros podem ser fixados apenas em uma das partes - não atravessa a placa -. Outra coisa importante a ser observada na placa são as trilhas, em momento algum elas podem se cruzar e esses cruzamentos podem ser evitados pela disposição dos componentes e pela utilização hora da parte superior da placa e hora da inferior no traçado delas. Também é importante lembrar que um cuidado a ser tomado é evitar um ângulo reto na trilha.

O shield será encaixado em um arduino uno e por meio de jumpers um segundo arduino uno se conectará a ele. Para essa conexão foram colocados pinos fêmea, um conjunto de pinos para cada protocolo, visto que as conexões para cada protocolo de comunicação é diferente. Esses pinos fêmeas têm trilhas que levam aos pinos machos que correspondem a entrada correta no arduino que está com a PCB acoplada.

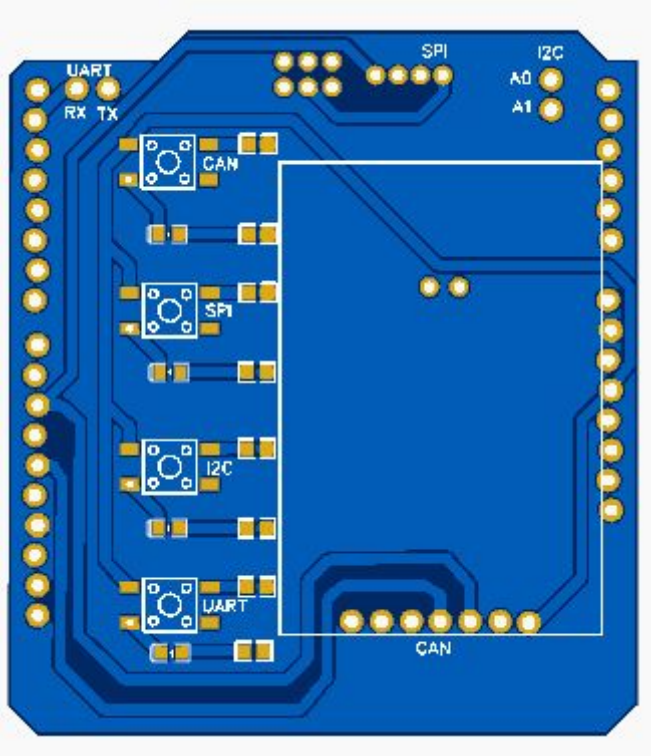

Imagem 3: PCB, parte superior e inferior.

A imagem acima representa a parte superior, vermelho, e inferior, azul, da placa. O grande retângulo amarelo no meio da placa representa o módulo MCP2515 que será conectado à placa. Todos os componentes estão conectados na parte superior da placa, como

mostrado na imagem abaixo que contém apenas essa parte da placa.

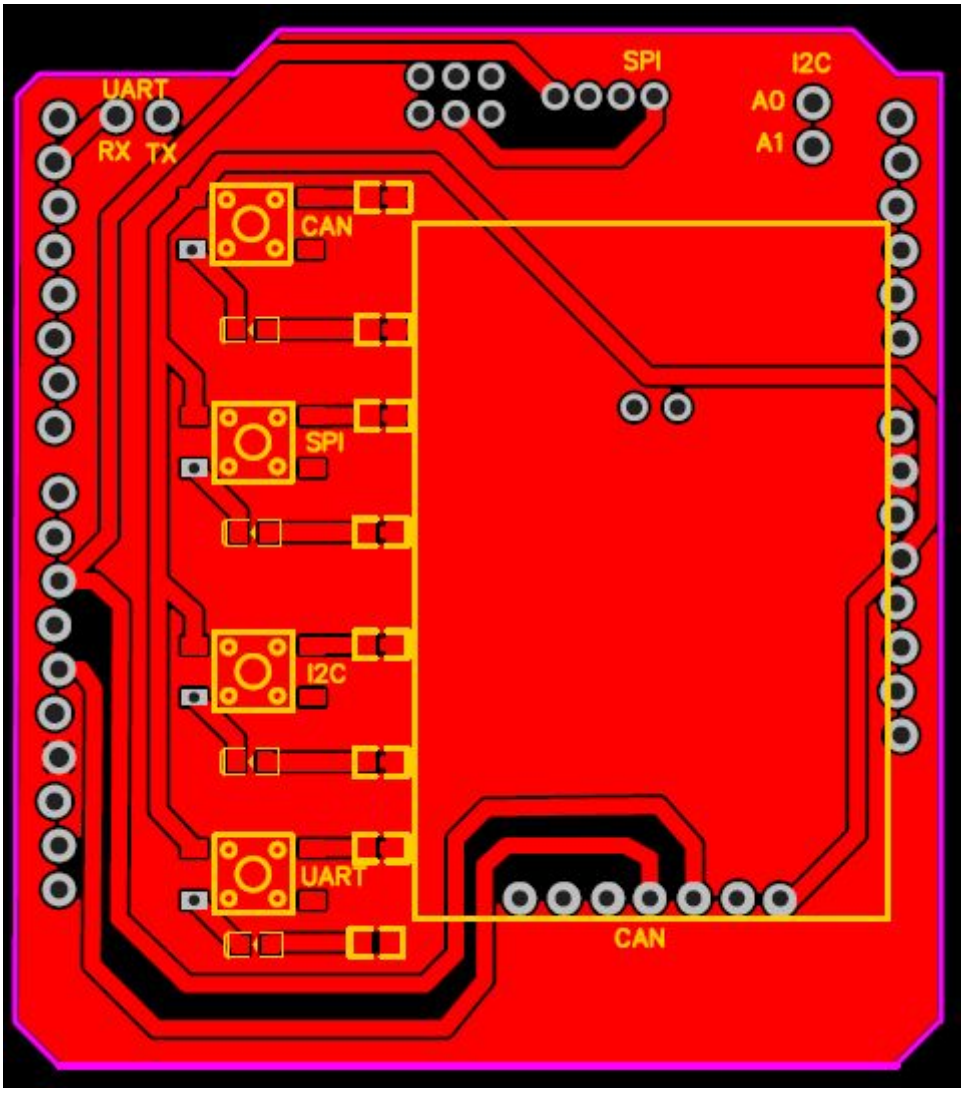

Imagem 4: PCB, parte superior.

Já a parte inferior tem apenas algumas trilhas que não foram feitas na parte superior para evitar o cruzamento.

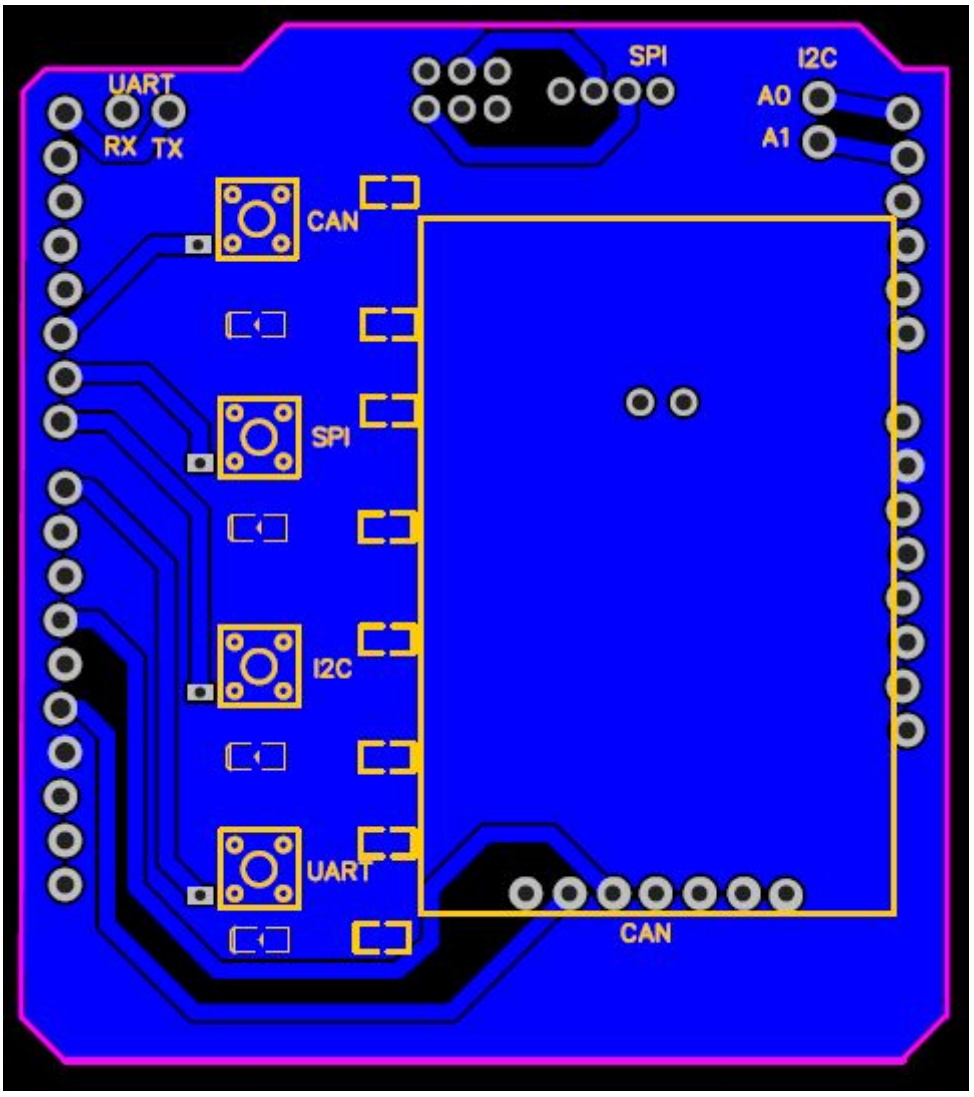

Imagem 5: PCB, parte inferior.

Toda essa parte colorida na placa, que não é uma trilha que leva um ponto a outro, tanto na parte superior, quanto na inferior da placa, é o ground. Ele serve para diminuir os ruídos na placa.

Geralmente o protocolo SPI tem como inputs as entradas 13 a 10 do arduino, mas isso é conflitante com as entradas do CAN, que usa as mesmas portas. Para resolvermos esse problema, colocamos as entradas de 13 a 11 nas portas ICSP e manter a entrada 10, responsável por chamar o escravo. Não vimos problema no CAN e o SPI compartilharem essa porta, visto que, ela tem a mesma função em ambos os protocolos.

6.4. PREÇOS E COMPONENTES

O shield projetado apresenta poucos componentes eletrônicos, em sua maioria, esses

componentes tem como função melhorar a usabilidade da PCB. Os componentes são:

- Resistor 1k ohm: 8 peças, no site baú da eletrônica, são encontrados a 0,04R$ a unidade, totalizando 0,24R$;
- Led vermelho: 4 peças, também encontrado no site baú da eletrônica por 0,24R$ a unidade, totalizando 0,96R$;
- Switch: 4 peças, encontrado a 0,40R$, totalizando 1,60R$;
- As barras de pinos, foram encontradas na faixa de 0,40R$ a 0,58R$, sendo 11 barras de pinos isso totaliza um preço máximo de 6,38R$;
- A PCB encontramos locais que fabricam a 5$, 5 placas, convertendo para reais fica 26,89R$.

Com isso o custo total estimado do projeto, desprezando os valores de frete fica a 36,07R$.

7. FABRICAÇÃO

Com o projeto pronto no Easyeda, é feita a exportação do projeto para imprimir ele em papel fotográfico e assim fixar as trilhas projetadas em uma placa de fenolite. Essa placa é revestida de cobre e quando mergulhada em um ácido todo o cobre se solta da placa, menos o que está tampado pelo papel fotográfico. Nesse momento, apenas com as trilhas de cobre na placa são feitos os furos para a fixação dos componentes e por fim a soldagem deles.

8. CONCLUSÃO

Com a finalização do projeto é nítido como são importantes os protocolos de comunicação, a conferência se eles estão atuando de forma adequada e que cada um deles tem suas especificidades fazendo que cada um deles seja melhor para situações distintas. Ademais, foi percebido também, que a melhor forma de desenvolver um projeto que atenda as necessidades da situação proposta neste relatório é o teste físico com os arduinos em mãos, já que existiram vários empecilhos das plataformas online e de desktop para desenvolvimentos de projetos em arduino. Para contornarmos esses problemas fizemos o teste físico para um dos protocolos (SPI), mas não conseguimos fazer o teste para o CAN por não termos o módulo MPC 2515 disponível.

Outra coisa percebida pela equipe, foi que um projeto de um circuito simples pode ser transformado facilmente em uma PCB e ela pode ser desenvolvida com um baixo custo.

9. REFERÊNCIAS

[1]https://www.gta.ufrj.br/grad/03_1/http/protoc.html#:~:text=Mas%20o%20que%20nos%20interessa,protocolo%20de%20transmiss%C3%A3o%20de%20dados%22.

[2]https://www.baudaeletronica.com.br/resistor-1k-5-1-4w.html?gclid=Cj0KCQiAlsv_BRDtARIsAHMGVSYK-88bNmkZmxf1i10XKirWnKFWVsGyCHvM706686zflj_dAwXBwt0aApriEALw_wcB

[3]https://www.baudaeletronica.com.br/led-difuso-5mm-vermelho.html?gclid=Cj0KCQiAlsv_BRDtARIsAHMGVSZjuec8URPJfB6RYK2zr5SO0TyO4B9xuq8cRZCf9EvH6OmFOWX7HEEaAjlTEALw_wcB

[4]https://proesi.com.br/conector-barra-pino-femea-pci-1x2-mci-2-54mm-180.html

[5]https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html

[6]http://www.alexag.com.br/CAN_Bus_Parte_2.html

[7]https://www.circuitbasics.com/how-to-set-up-uart-communication-for-arduin/

[8]https://circuitdigest.com/microcontroller-projects/arduino-spi-communication-tutorial

[9]https://www.embarcados.com.br/comunicacao-spi-parte-2/

[10]http://www.alexag.com.br/Artigos/SAE2002.pdf

[11]https://www.electronicshub.org/arduino-mcp2515-can-bus-tutorial/

[12]https://www.embarcados.com.br/barramento-can-entre-arduinos-uno/

[13]https://www.youtube.com/watch?v=GS7iTWTRtOI

[14]https://www.youtube.com/watch?v=fr5kALgcW0o