Desenvolvimento de uma Plataforma para Competição Automotiva Autônoma - PCAA

Lucas Medeiros de Queiroz

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

2018

Lucas Medeiros de Queiroz

Desenvolvimento de uma Plataforma para Competição Automotiva Autônoma - PCAA

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Engenharia de Computação do Centro de Informática, da Universidade Federal da Paraíba, como requisito para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Dr. Mardson de Freitas Amorim

Novembro de 2018

Catalogação na publicação
Seção de Catalogação e Classificação

Q3d Queiroz, Lucas Medeiros de.
Desenvolvimento de uma Plataforma para Competição Automotiva Autônoma - PCAA / Lucas Medeiros de Queiroz. - João Pessoa, 2018.
65 f. : il.

Orientação: Mardson Amorim.
Monografia (Graduação) - UFPB/CI.

1. telemetria. 2. microcontrolador. 3. L3. 4. I2C. 5. DCP. 6. veículo autônomo. I. Amorim, Mardson. II. Título.

UFPB/BC

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos nove dias do mês de novembro de dois mil e dezoito, às oito horas e trinta minutos, em sessão pública no auditório do Centro de Informática do Campus I da Universidade Federal da Paraíba, na presença da Banca Examinadora presidida pelo professor orientador **Mardson Freitas de Amorim**, e pelo Eng. **Euler Bakke Duarte de Souza** e pelo professor **José Antônio Gomes de Lima**, o aluno **LUCAS MEDEIROS DE QUEIROZ**, apresentou o trabalho de conclusão de curso intitulado: **DESENVOLVIMENTO DE UMA PLATAFORMA PARA COMPETIÇÃO AUTOMOTIVA AUTÔNOMA - PCAA**, como requisito curricular indispensável para a integralização do Curso de Engenharia de Computação.

Após a exposição oral, o candidato foi arguido pelos componentes da Banca que se reuniram reservadamente e decidiram APROVAR a monografia com nota 10 (dez). Divulgando o resultado formalmente ao aluno e demais presentes, eu, na qualidade de Presidente da Banca, lavrei a presente ata que está assinada por mim, pelos demais examinadores e pelo aluno.

Prof. Mardson Freitas de Amorim (orientador)

Eng. Euler Bakke Duarte de Souza

Prof. José Antônio Gomes de Lima

Aluno: Lucas Medeiros de Queiroz

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

AGRADECIMENTOS

A Deus por ter me dado saúde e força para concluir o curso. A minha mãe Judenice, meu pai Leomax, minhã irmã Lívia, minha madrinha Consuelo, minha tia Conceição, tia Alcinda, tia Carmen, Rute, tio José, tia Judilede, tia Maria de Fátima, tia Zulmira, minhas avós Conceição e Sabina, meus avôs Manoel (em memória) e João Vieira, todos os meus tios e tias e toda a minha família que sempre me apoiaram desde o início para que eu possa chegar a onde estou.

A meus amigos ao longo do curso, Leandro, Renno, Janislley, Luiz, Kened, Ewerton, Emanuella, Natália, Marília, Jeferson, Raul, Caio, Wilter, Lira, Gustavo, Jordy, Smith, Lucas, Diego, Diego Ramon, Elias e aos demais que esteve comigo ao longo do curso.

Ao meu orientador Mardson e ao professor Ruy que acreditaram no meu potencial e sempre me motivaram no curso. E aos demais professores que fizeram parte da minha caminhada na universidade.

"Envelheci dez anos ou mais nesse último mês"
Engenheiros do Hawaii

RESUMO

Neste trabalho é desenvolvido um veículo que contará com sensores e subsistemas de controle que controlarão o veículo e gerarão dados referentes ao seu estado atual com objetivo de deixa-lo autônomo. Para ter acesso a esses dados, uma requisição é feita via I2C com o protocolo L3 para o veículo pelo microcontrolador externo, para assim, os dados requeridos serem transmitidos de volta para o controlador externo. Para essa comunicação entre o veículo e o controle externo, utiliza-se o sistema de telemetria para a obtenção desses dados provenientes do veículo. Para isso será utilizado o protocolo L3 que atua na camada de aplicação, e o protocolo I2C para a comunicação na camada física. A troca de informações na grande maioria dos veículos convencionais ocorre por meio do protocolo de comunicação CAN, pois consiste em um sistema robusto e de alta confiabilidade. Baseado neste protocolo, é utilizado um protocolo de comunicação chamado Devices Communication Protocol (DCP) em desenvolvimento no LMI que consiste na utilização de apenas um fio para a transmissão de dados, que atenderá aos propósitos do trabalho.

O veículo seguirá uma trajetória representada por uma pista. Ela contará com símbolos que servirá para indicação de posicionamento e para informar sinalizações a serem seguidas. Esse trabalho mostrou que o veículo responde bem às requisições com os protocolos DCP, L3 e I2C, e consegue ser gerenciado de forma satisfatória dentre os protocolos.

Palavras-chave: telemetria, microcontrolador, L3, I2C, DCP, veículo autônomo
.

ABSTRACT

In this work is developed a vehicle that will have sensors and control subsystems that will control the vehicle and generate data recurrent to its current state in order to make it autonomous. In order to access this data, a request is made via I2C with the L3 protocol to the vehicle by the external microcontroller, so that the required data is transmitted back to the external controller. For this communication between the vehicle and external control, the telemetry system is used to obtain this data from the vehicle. For this, the L3 protocol that acts in the application layer will be used, and the I2C protocol for the communication in the physical layer. The internal information exchange in the vast majority of conventional vehicles occurs through the CAN communication protocol, since it consists of a robust and high reliability system. Based on this protocol, a communication protocol called Devices Communication Protocol (DCP) is being developed in the LMI, which consists in the use of only one wire for data transmission, which will serve the purposes of the work.

The vehicle will follow a path represented by a lane. It will have symbols that will serve to indicate positioning and to report signs to be followed. This work showed that the vehicle responds well to requests with the DCP, L3 and I2C protocols, and can be managed satisfactorily among the protocols.

Key-words: telemetry, microcontroller, L3, I2C, DCP, autonomous car.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

LMI – Laboratório de Medidas e Instrumentação

DCP – Devices Communication Protocol

I2C – Inter-Integrated Circuit

CAN – Controller Area Network

CSMA/CD – Carrier Sense Multiple Access with Collision Detection

SDA – Serial Data

SCL – Serial Clock

CNC – Comando numérico computadorizado

Sumário

1 INTRODUÇÃO

O conhecimento da programação de computadores está sendo cada vez mais necessário no dia a dia e no mercado de trabalho. Basicamente tudo que fazemos envolve o uso softwares de alguma forma, seja na substituição de trabalhos antigos, antes manuais, até o surgimento de novos trabalhos onde sua existência se dá pelo fato do avanço da tecnologia (Crossley, 2014).

Com o intuito de estimular pessoas de todas as idades a ingressar no universo da programação e da robótica através de algo mais divertido, é proposto um trabalho que motive-os a estudar e desenvolver programas para controle de um automóvel autônomo em escala reduzida em uma pista de competição.

1.1 Definição do Problema

Com o crescente aumento do número de carros autônomos, é iminente que eles se tornarão algo presente no nosso cotidiano em alguns anos (G1, 2018). Buscando uma forma para entender melhor como funciona essa tecnologia e termos um primeiro contato ainda na formação acadêmica, é proposto uma plataforma de competição em que um modelo reduzido de um automóvel é totalmente programável a fim de torná-lo autônomo para competições. Com isso, é possível ter um entendimento básico de como um carro autônomo funciona por meios de disputas de competições em que o objetivo é que o veículo apresente a melhor autonomia para terminar o trajeto.

1.2 Objetivo geral

Desenvolvimento de uma plataforma de competição com um modelo reduzido de automóvel programável onde se é possível torná-lo autônomo para competições.

1.3 Objetivos específicos

São estabelecidos os seguintes objetivos específicos:

- **Estabelecer a comunicação entre o meio externo e o veículo.**
- **Desenvolver o subsistema de direção, subsistema de detecção de faixa do veículo, subsistema de sinalização e iluminação, subsistema de tração e subsistema de detecção de colisão**
- **Estabelecer a comunicação entre todos os subsistemas.**
- **Elaborar os requisitos para a pista de competição para o modelo**

2 CONCEITOS GERAIS E REVISÃO DA LITERATURA

Neste capítulo são explicadas os conceitos que são utilizados nesse trabalho, além dos trabalhos relacionados.

2.1 Telemetria

Telemetria é a ciência de obter dados de algum local e transmitir esses dados de um ponto a outro para serem analisados (LOZANO-NIETO, 1999).

2.1.1 Métodos de Telemetria

A telemetria pode ser executada por diferentes meios: ópticos, mecânicos, hidráulicos, elétricos entre outros. O mais usado no nosso contexto é o método baseado em sinais elétricos, levando vantagens sobre os outros por conseguir um alto alcance nas áreas a serem analisadas, e também podem ser facilmente adaptadas em estruturas já existentes. (LOZANO-NIETO, 1999). A telemetria eletrônica pode ser utilizada tanto via cabo ou sem fio, sendo a sem fio mais complexa, pois requer a utilização de radiofrequência. Apesar da sua complexidade, ela é amplamente utilizada pois transmite informações por longas distâncias e em alta velocidade, permitindo a utilização em locais de difíceis acessos (LOZANO-NIETO, 1999).

2.1.2 Telemetria Automotiva

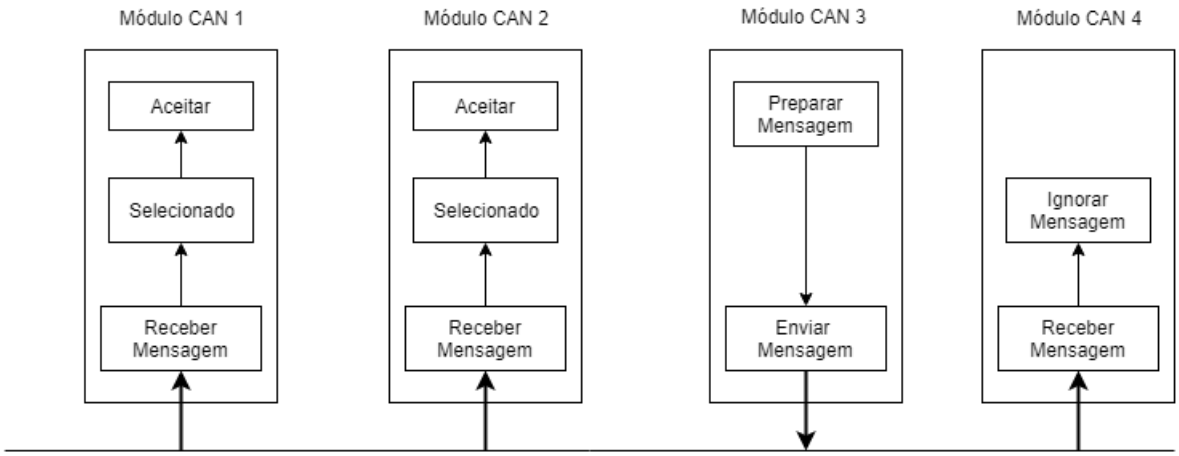
A telemetria automotiva é utilizada por engenheiros em competições e treinos para obter informações sobre temperatura do motor, velocidade, aceleração, entre outros parâmetros a partir da utilização de sensores presentes no veículo. Com esses dados a equipe é capaz de fazer melhorias nos veículos, além de em casos de acidentes, serem capazes de descobrirem a sua causa (POLESE, 2017).

2.2 Protocolo CAN

O CAN é um protocolo de comunicação serial síncrono amplamente utilizado no meio automotivo, que é conhecido por ser um sistema robusto e de alta confiabilidade de transmissão (BALDISSERA, 2011). O CAN reduz o número de fios e a complexidade do cabeamento utilizado nos automóveis para operar, utilizando apenas um par de cabos trançados. Outra característica interessante é o fato de ser multi-mestre, onde os módulos podem ser mestres em um momento e no outro escravos, e trabalhar com mensagens multicast, em que todos os módulos da rede recebem as mensagens e a partir da

identificação reconhecer se a mensagem é ou não de seu interesse (BALDISSERA, 2011). A exemplificação do envio e recebimento das mensagens pode ser visto na figura 1.

Figura 1: Mensagem transmitida via barramento CAN



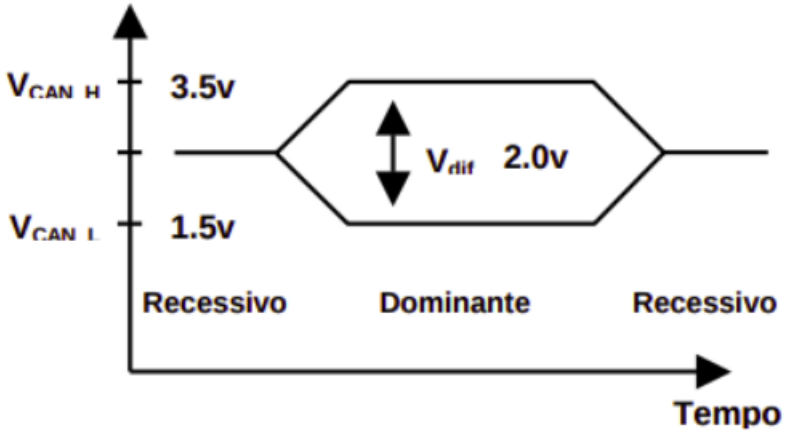
Fonte: Adaptado de (BALDISSERA, 2011)

2.2.1 Transmissão

A transmissão dos dados não ocorrem por meio de nível lógico alto e baixo, e sim pela representação de bits dominantes e recessivos. Como podemos observar na figura 2, os bits dominantes são representados por uma diferença de potencial de 2 *Volts* entre os fios *CAN High* e *CAN Low*, enquanto os recessivos não apresentam diferença de potencial (BALDISSERA, 2011).

Para que ocorra a transmissão dos bits de maneira com que a diferença de potencial se mantenha constante, é utilizado um par de fios trançados que faz com que a interferência presente em um dos fios também afete ao outro fio de maneira igual por meio do efeito de cancelamento (Cassiolato, 2014), mantendo assim a diferença de potencial entre eles constantes, evitando a necessidade da utilização de fios blindados (GUIMARÃES and SARAIVA, 2003).

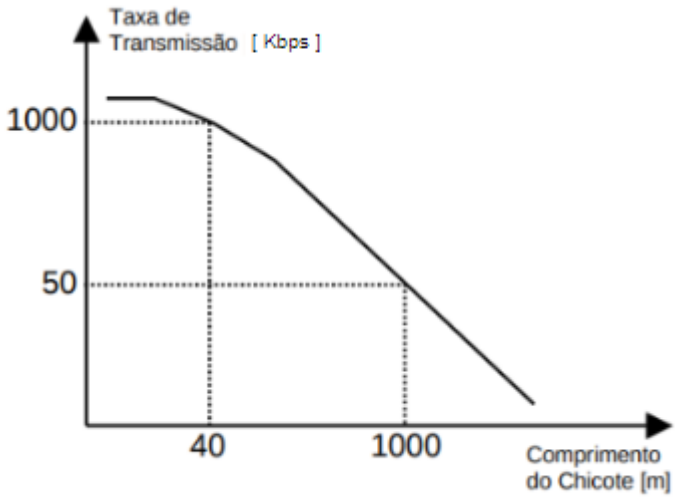
Figura 2: Níveis de Tensão no CAN e os bits Dominantes e Recessivos.



Fonte: (GUIMARÃES and SARAIVA, 2003)

Assim como podemos analisar na figura 3, a velocidade de transmissão é inversamente proporcional ao comprimento do barramento. A taxa máxima é de 1Mbps levando em consideração um barramento de 40 metros (BALDISSERA, 2011). Essa representação entre o comprimento do barramento a taxa de transmissão pode ser melhor entendida na imagem a seguir.

Figura 3: Relação entre tamanho dos fios e taxa de transmissão.

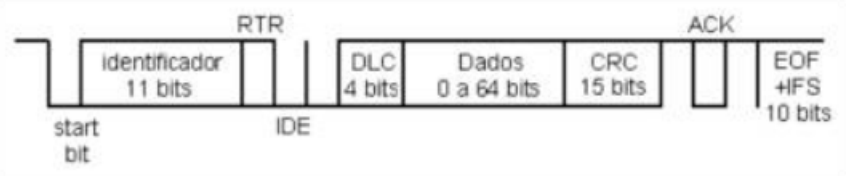


Fonte: (GUIMARÃES and SARAIVA, 2003)

2.2.2 Pacote CAN

No protocolo CAN, o pacote pode possuir dois formatos: CAN 2.0A (*standard*) e o CAN 2.0B (formato estendido). No formato 2.0A, ele possui um identificador de 11 bits possibilitando a utilização de 2048 mensagens. Já no CAN 2.0B foi aumentado para 29 bits de identificação, possibilitando aproximadamente 537 milhões de mensagens. Esse aumento acabou com a limitação de mensagens do formato 2.0A, mas por outro lado, em alguns casos os 18 bits adicionais no identificador aumentam o tempo de transmissão, podendo ser um problema para aplicações que trabalham em tempo real (BALDISSERA (2011)). O formato do pacote CAN 2.0A é visto na figura 4 e do formato CAN 2.0B é visto na figura 5.

Figura 4: Formato do Pacote CAN 2.0A



Fonte: (BALDISSERA, 2011)

Figura 5: Formato do Pacote CAN 2.0B

Fonte: (BALDISSERA, 2011)

O significado dos campos em ambos os formatos é descrito na tabela 1.

Tabela 1: Descrição dos Campos do Pacote CAN

Campo	Formato CAN	Descrição
SOH	2.0 A/B	Marca o início da mensagem
IDENTIFICADOR	2.0 A/B	Estabelece a prioridade mensagem
SRR	2.0 B	Substitui o RTR do formato 2.0A, fica sempre em estado dominante
IDE	2.0 B	Identifica se está usando a extensão,
RTR	2.0 A/B	Quando acionado, indica que a informação está sendo solicitada
DLC	2.0 A/B	Informa ao receptor quantos bytes de dados a mensagem possui
DATA	2.0 A/B	Até 64 bits de dados podem ser transmitidos
CRC	2.0 A/B	Checksum para detecção de erros
ACK	2.0 A/B	Receptor sobrescreve o bit recessivo caso a mensagem seja recebida sem erros
EOF	2.0 A/B	Indica o fim do pacote
IFS	2.0 A/B	Tempo para o controlador mover os dados do buffer

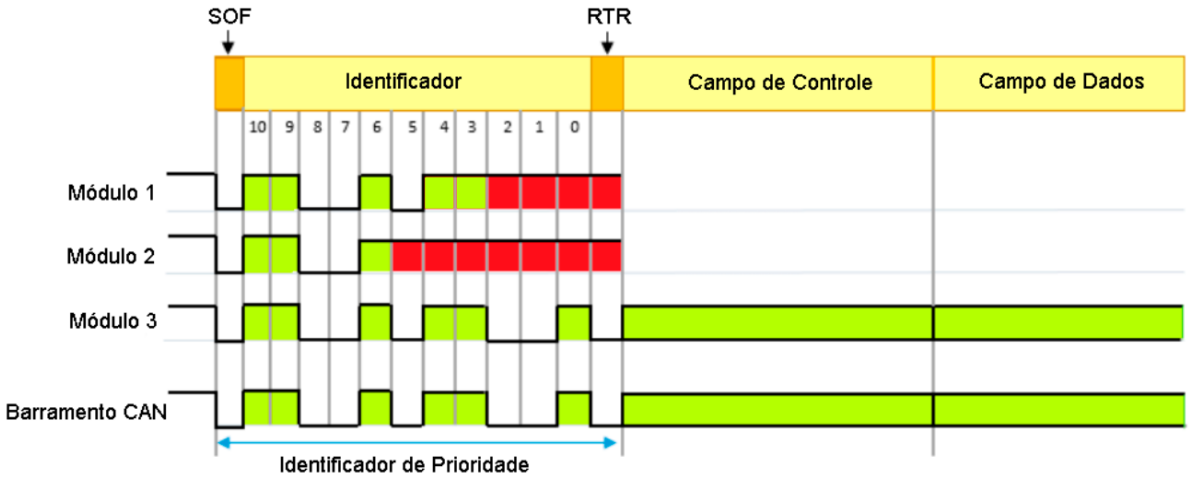
Fonte: Adaptado de (BALDISSERA, 2011)

2.2.3 Tratamento de Colisão

O protocolo CAN é do tipo CSMA/CD+AMP (BALDISSERA (2011)). O CSMA significa que cada módulo da rede CAN deve aguardar o barramento livre para iniciar a transmissão. O CD+AMP indica que a colisão será tratada pela arbitração de bits, ou seja, o bit dominante (bit "0") terá mais alta prioridade do que o bit recessivo (bit "1"). Para essa arbitração, o protocolo CAN utiliza um identificador de 11 bits que é utilizado como marcador de prioridade. Quanto menor esse identificador, maior será a prioridade que o módulo eletrônico possui sobre o envio ao barramento (GUIMARÃES and SARAIVA, 2003).

Para o envio de dados, o módulo escuta o barramento para identificar se está livre ou não. Se caso mais de um módulo escute o barramento ao mesmo tempo e identifiquem ele livre, ambos iniciarão o envio dos dados no barramento e escutarão de volta para verificarem se os dados lidos correspondem ao enviado. Caso não seja, o envio é colocado em espera, o dado com maior prioridade é lido e logo após volta a escutar o barramento para tentar enviar o dado (GUIMARÃES and SARAIVA, 2003). Esse processo de tratamento é ilustrado na figura 6.

Figura 6: Tratamento de Colisão

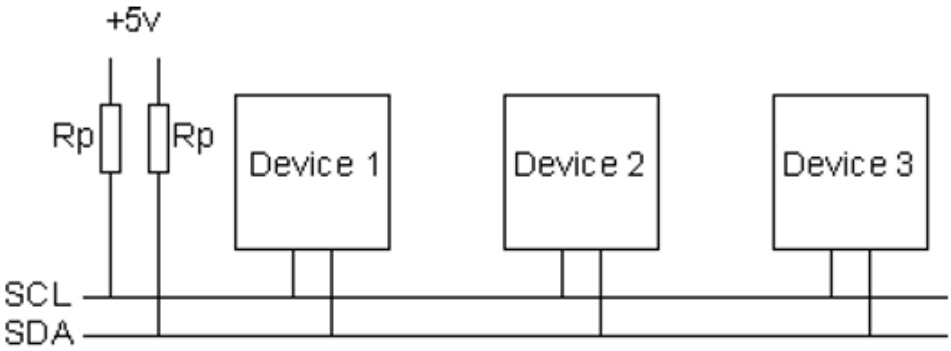


Fonte: Adaptado de (Avatefipour and Malik (2017))

2.3 Protocolo I2C

O protocolo I2C descreve o funcionamento de um barramento de comunicação de apenas dois fios para transmissão dos dados, SDA e SCL. O barramento é mantido em High por resistores de pull-ups e funciona com alimentação tipicamente de 3.3V ou 5V (CAMARA). O esquema dos fios e dos resistores são mostrados na figura 7.

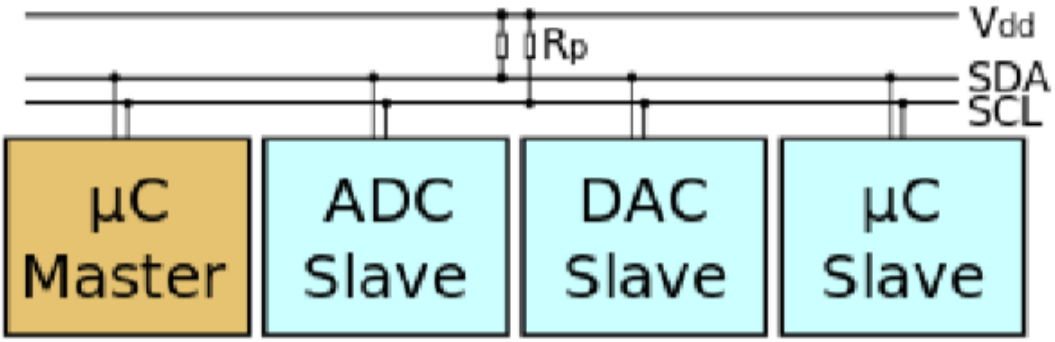
Figura 7: Barramento I2C



Fonte: (CAMARA)

O I2C possui dois tipos de dispositivos, o *Master*, que é responsável para coordenar todos os periféricos, e o *Slave* (figura 8).

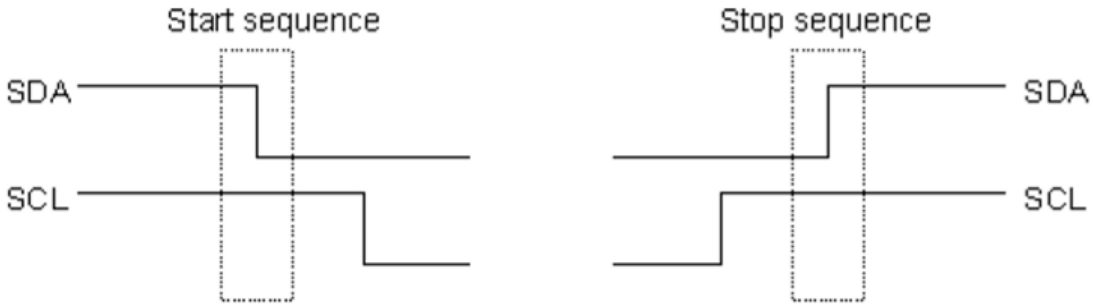
Figura 8: Master e Slave do I2C



Fonte: (CAMARA)

O inicio da transmissão no barramento é imposto pelo *Master*, que é caracterizado pelo Start Bit, em que consiste no SDA em Low enquanto o SCL está em High e então é iniciado o envio dos bits no barramento. O SCL irá pulsar sinalizando o *clock* e então o valor de SDA é lido como o valor do bit. A transmissão se encerra quando é lido o Stop bit, que consiste na subida do SDA enquanto o SCL esta em High (figura 9) (CAMARA).

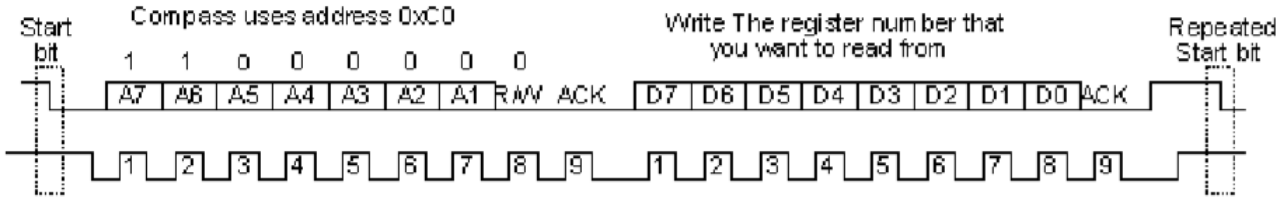
Figura 9: Start e Stop bit do I2C



Fonte: (CAMARA)

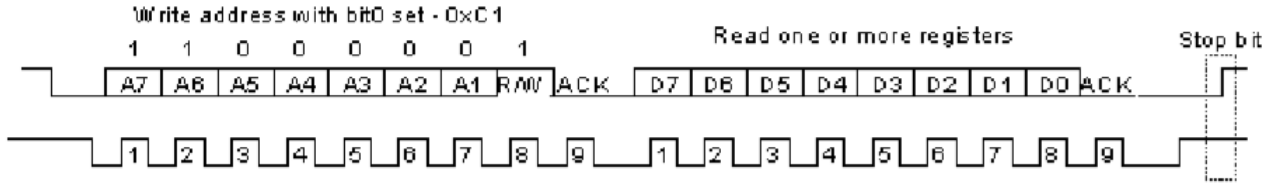
Após o Start bit, o *Master* envia o endereço do dispositivo que deseja se comunicar junto com o comando de escrita ou leitura (bit R/W). Caso o dispositivo exista no barramento, ele irá confirmar enviando um ACK para o *Master*, iniciando assim a transmissão (CAMARA). As operações de escrita e leitura são ilustrados na figura 10 e 11 respectivamente.

Figura 10: Processo de Escrita I2C



Fonte: (CAMARA)

Figura 11: Processo de Leitura I2C



Fonte: (CAMARA)

O I2C alcança 100 Kbit/s na sua velocidade padrão, podendo chegar a 400 Kbit/s no modo rápido, e ate 3,4 Mbit/s no modo alta velocidade (NXP, 2014).

2.4 Protocolo L3

O L3 é um protocolo da camada de aplicação em desenvolvimento pelo LMI para a telemetria. O L3 é formado por um pacote de 12 bytes, subdivididos em 7 campos. Cada campo possui uma quantidade de bytes pré-determinadas. A tabela 2 mostra o formato do pacote L3 e a 3 descrição dos campos (Amorim, 2005).

Tabela 2: Pacote L3

SOH	IDS	IDD	COD	DATA/PADDING	PAD	CRC-8

Fonte: Protocolo L3

Tabela 3: Campos do Pacote L3

Campo	Número de Bytes	Descrição
SOH	1	Início do Pacote
IDS	1	Identificador de Origem
IDD	1	Identificador de Destino
DATA/PADDING	6	Código de Instrução
PAD	1	Informação Enchimento
CRC-8	1	Código de checagem de redundância cíclica

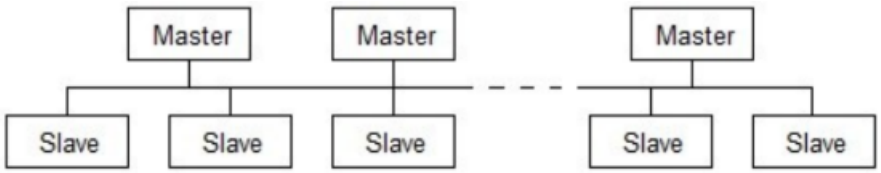
Fonte: Protocolo L3

A tabela de instrução e a de códigos de erros do protocolo se encontram em anexo.

2.5 Devices Communication Protocol (DCP)

O DCP é um protocolo da camada física em desenvolvimento no LMI de comunicação bi-direcional. Como mostrado na figura 12 o protocolo adota o conceito de *Master* e *Slave* em que toda a sua comunicação ocorre por meio de um barramento em nível lógico *High* (pull-up) de um único fio. Para o dispositivo enviar dados no barramento, ele apenas deve impor uma sequência de níveis lógicos *Low* (Amorim, 2018).

Figura 12: Topologia do DCP



Fonte: Protocolo DCP

A velocidade da transmissão de dados depende do dispositivo usado. Para isso é atribuído um tempo base para o protocolo que irá determinar o tempo dos pulsos do DCP.

2.5.1 Start Sync

Para identificar o início do pacote e para diferenciar no barramento o *Master* dos *Slaves* é atribuído um pulso inicial em *Low* chamado de *Start Sync*, em que para o *Master*

esse tempo do pulso em *Low* equivale a duração de 25 vezes o tempo base do DCP para a aplicação e 12,5 vezes para o *Slave*.

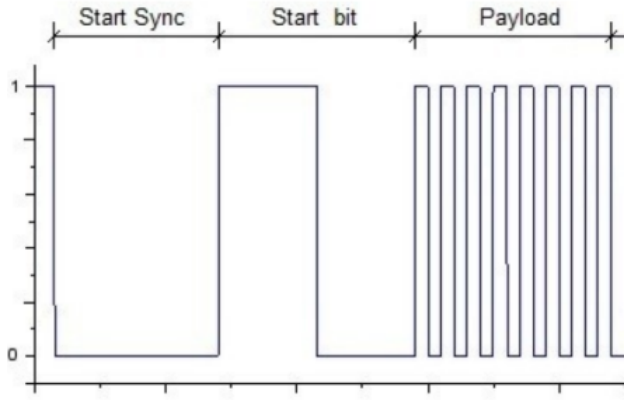
2.5.2 Start Bit

Após o *Start Sync*, acontece o pulso *Start Bit* que consiste no barramento um tempo em *Low* e um tempo em *High* iguais. A duração desses tempos são igual a 7,5 vezes o tempo base para cada um.

2.5.3 Payload

Após o *Start bit*, são enviados os bits dos pacotes de dados que correspondem ao *payload*. A identificação dos bits "0"e "1"são a partir do tempo em *High* que passam no barramento. Para o tempo em *High* do bit "0", é atribuído a duração igual ao tempo base, e para o bit "1", é atribuído duas vezes o tempo base, e para separar os bits é usado um tempo em low igual ao tempo base. O formato do pacote para o DCP é visto na figura 13.

Figura 13: Pacote DCP



Fonte: Protocolo DCP

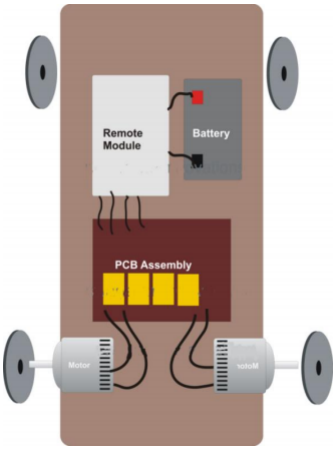
2.6 Trabalhos Relacionados

A partir do estudo dos trabalhos relacionados, pode-se obter uma ideia do cenário atual sobre o tema proposto, e com isso utilizar os resultados obtidos por outros autores para seguir como referência.

Segundo (Simatupang and Yosua, 2016) um carrinho de brinquedo controlado por Rádio Frequência foi adaptado para ser controlado por um ESP8266 (Systems, 2018) via *WiFi*, em que se conectam a um roteador e são controlados por uma página html

pelo navegador que exibe os comandos a serem executados. O uso do roteador pode gerar latência no recebimento dos pacotes pelo fato de estarem roteando mais de um dispositivo conectado à rede, gerando assim atrasos no recebimento dos comandos. Outro ponto para destacar é que os comandos do carro são controlados manualmente e não apresenta nenhum sensor ou autonomia. Na figura 14 é ilustrado o resultado odo trabalho, que consiste de um módulo *wireless* com bateria que recebe os comandos para a atuação dos motores.

Figura 14: Ilustração de um Carro Controlado sem Fio

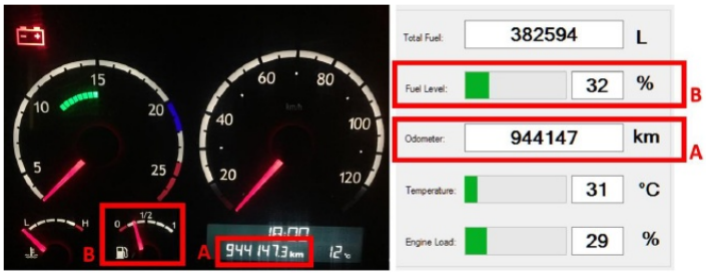


Fonte: (Simatupang and Yosua, 2016)

Já no trabalho (Fracarolli et al., 2012) é construído um carro com peças Lego e sua comunicação é realizada através da tecnologia *ZigBee* (Inc.) por ser uma alternativa energeticamente econômica em comparação aos demais. Sua interface com o usuário é efetuada através do uso de um *software* programado na linguagem Python que recebe e envia os comandos por uma porta serial. Neste trabalho a conexão entre o servidor e o carro acontece de forma direta, ou seja, não necessita do uso de roteadores para encaminhar os dados, diminuindo assim o tempo de resposta aos comandos. Assim como em (Simatupang and Yosua, 2016), o controle do carro ainda necessita dos comandos manuais para sua operação, não utilizando sensores.

No trabalho de (POLESE, 2017) o objetivo é o diagnóstico veicular com transmissão pela internet em tempo real, que consiste na simulação dos sensores presentes no carro, e o resultado obtido é enviado pela internet e analisado por um técnico capacitado para o diagnóstico. Na figura 15 é ilustrado o resultado obtido através da simulação dos sensores.

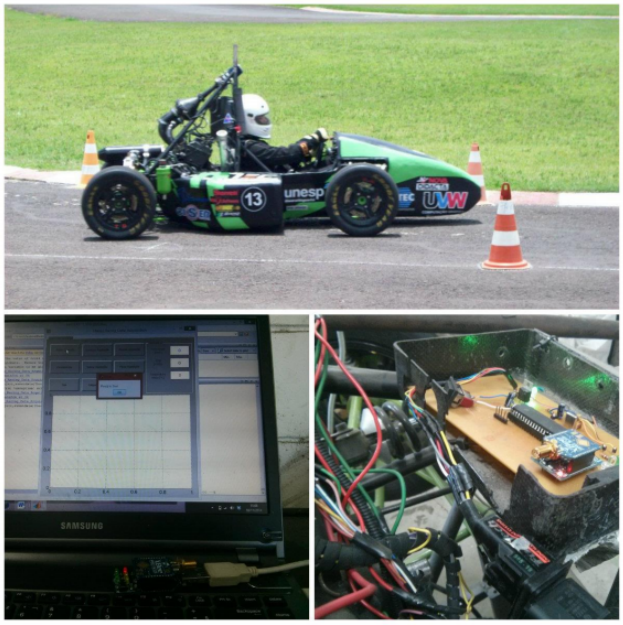
Figura 15: Aferição da Telemetria



Fonte: (POLESE, 2017)

No trabalho (Queirós et al., 2011) e em (Siqueira, 2014), os autores utilizam da telemetria para o sensoriamento de um veículo. Esses dados são obtidos também pelo uso da tecnologia *Zigbee* em que seu objetivo é o melhoramento de um veículo de competição a partir da análise dos dados obtidos. Na figura 16 é mostrado o veículo e o seu sensoriamento com os módulos de comunicação utilizados.

Figura 16: Carro de competição



Fonte: (Siqueira, 2014)

O trabalho de (Martins et al.) utiliza um veículo autônomo para participar de uma competição patrocinada pela empresa Freescale, que distribui os kits prontos para a montagem e programação dos veículos para a competição. O que se destaca na competição é a equipe conseguir uma melhor autonomia e velocidade do seu veículo, uma vez que todas as equipes competidoras também receberam o mesmo kit. O carro utiliza uma câmera

que capta a pista e responde com um determinado sinal, esse sinal será tratado e voltado em forma de comando para o veículo. Vence a competição o carro que fizer o trajeto mais rápido, sendo assim, o que conseguir manter o melhor controle da velocidade e conseguir permanecer na pista até o final do trajeto. O controle ocorre dentro do veículo, não obtendo os dados dos sensores remotamente.

3 METODOLOGIA

O modelo utilizado para o veículo será reaproveitado de um carro de controle remoto. Tendo em mãos o chassi do veículo, os motores e as pontes H, serão aproveitados para a reconstituição e simulação de um automóvel com maior representação possível de suas funcionalidades e ações de um veículo real. Essa representação será possível com a utilização de sinalizadores, sensores e atuadores que serão controlados por comandos recebidos externamente, utilizando o protocolo da camada de aplicação L3. A comunicação com o veículo ocorrerá através de um microcontrolador conectado ao veículo trocando dados via o protocolo de camada física I2C. Esse microcontrolador poderá receber e enviar os comandos pela rede Wi-Fi ou está programado para controlar o veículo offline.

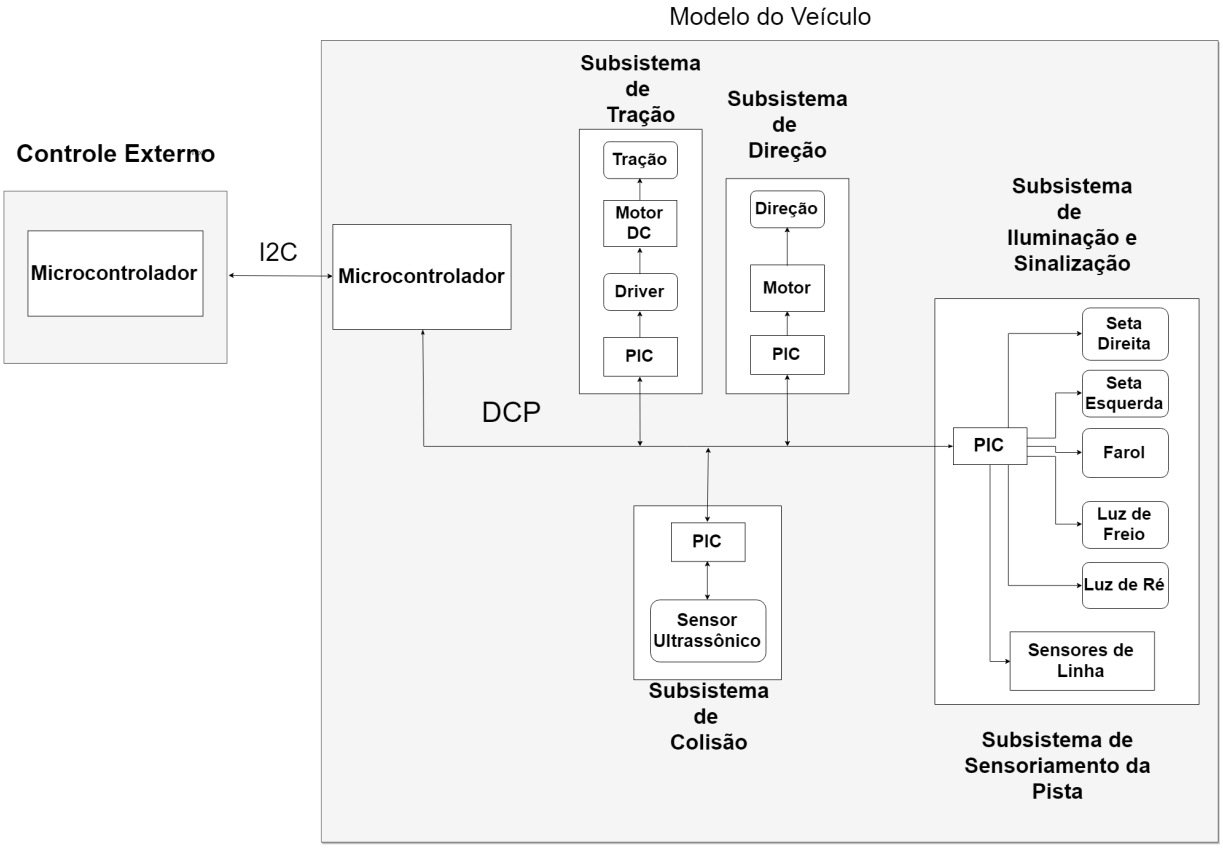
Inicialmente, toda a parte de sinalização, sensores e atuadores do modelo se encontram desligados. A partir do momento em que ligar e se conectar ao programa externo responsável pelo controle, cabe ao programa enviar uma requisição ao modelo e ele enviar uma resposta (pacote) em que conterá as informações referentes a todos os subsistemas presente no modelo. Após o programa ter acesso a essas informações, ele tratará os dados, e logo após enviará um pacote informando ao veículo os novos estados dos subsistemas. Esse ciclo de requisição - resposta - comando , permanecerá por todo o tempo em que a conexão permanecer ativa.

O modelo do veículo será capaz de comportar todos os sensores e atuadores que serão utilizados para se localizar na plataforma, além do uso de leds utilizados para a sinalização e iluminação.

O sistema de controle do veículo será divido em cinco subsistemas, como se pode ver na figura 17. Esses subsistemas serão responsáveis pelo recebimento e envio de informações referentes as suas funções. Serão divididos em subsistema de direção, subsistema de tração, subsistema de iluminação e sinalização, subsistema de detecção de faixa e por último o subsistema de colisão. Todos esses subsistemas serão controlados por um microcontrolador *master* que se comunicará por meio do protocolo DCP. Uma explicação mais detalhada referente aos subsistemas e ao protocolo utilizado para a comunicação entre os subsistemas será explicado mais adiante.

A pista de competição contará com sinalizadores que serão capazes de informar ao modelo, através do sensoriamento presente no mesmo, a sinalização necessária para que ele consiga continuar na pista e chegar ao final. Cabe ao programa externo de controle solicitar as informações dos subsistemas e definir a ação dos mesmos.

Figura 17: Diagrama



Fonte: Autoria Própria

3.1 Controle Externo

O controle externo mostrado na figura 17 , será responsável pelo modo de como o veículo será controlado. A opção será por meio de um dispositivo com comunicação I2C que contenha a programação de controle para o veículo.

3.2 Comunicação com o Veículo Modelo

A troca de dados do controle externo com o modelo será através do protocolo da camada física I2C com o protocolo da camada de aplicação L3, que permitirá que qualquer microcontrolador compatível consiga obter os dados dos subsistemas e enviar comandos para controlar o veículo.

Os comandos do protocolo L3 foram adaptados para a necessidade do trabalho. Sendo assim, os comandos são mostrados na tabela 4, em que o nome do subsistema equivale a instrução referente a ela.

Tabela 4: Comandos L3

Mnemônico	Código	Byte de Dados						Descrição
SSP	83	0	0xFF	0xFF	0xFF	0xFF	0xFF	Programa requisita o estado dos subsistemas
RSP	80	0	TRAÇÃO	DIREÇÃO	SINALIZAÇÃO	SENSORES	SINALIZAÇÃO	Veículo responde o estado dos subsistemas
ASP	81	0	TRAÇÃO	DIREÇÃO	SINALIZAÇÃO	0XFF	0XFF	Programa escreve para o veículo os comandos para os subsistemas
ACK	6	CÓDIGO	0XFF	0XFF	0XFF	0XFF	0XFF	Veículo confirma o código recebido
ENQ	5	DISPOSITIVO	0xFF	0xFF	0xFF	0xFF	0xFF	Programa requisita se os subsistemas estão operacionais
NAK	21	COD_ERRO	CÓDIGO	0xFF	0xFF	0xFF	0xFF	Carro informa erro no código recebido
E1B	51	ENDEREÇO	DADO	0xFF	0xFF	0xFF	0xFF	Mudar parâmetros dos subsistemas

Fonte: Autoria Própria

Cada subsistema tem seus conjuntos de instruções de tamanho de um byte, ou seja, o máximo de instruções utilizadas para essa aplicação é de 255 instruções referente as suas funções.

3.3 Microcontroladores Utilizados

Para cada subsistema foi escolhido um microcontrolador PIC que satisfaça os requisitos de funcionamento de cada subsistema.

3.3.1 PIC12F675

O PIC12F675 é utilizado para os subsistemas de tração, direção e colisão. Pois apresenta os seguintes recursos:

- 2 Timers
- Oscilador interno de 4 MHz
- 128 bytes na memória EEPROM

- 64 Bytes de SRAM
- 6 pinos de entrada/saída digital
- Interrupção na mudança de estado dos pinos digitais

Esses recursos (Technology (2010)) foram suficientes para suprir os requisitos dos subsistemas em questão.

3.3.2 PIC16F628A

O PIC16F628A além de apresentar as características já mostradas no PIC12F675, ele apresenta os seguintes recursos adicionais (Technology (2009)):

- Periférico para gerar PWM
- 224 Bytes de SRAM
- 16 pinos de entrada/saída digital

A presença de um periférico para gerar o PWM foi escolhido para controle da luz de freio do subsistema de sinalização, pois no PIC12F675 para essa aplicação o PWM gerado tem uma frequência baixa, fazendo com que não seja uma boa escolha para essa função.

Esses recursos a mais do PIC16F628A atende aos requisitos do subsistema de sinalização e o subsistema de sensoreamento da pista. A utilização de um PIC para cada um dos subsistemas em questão iria deixar o PIC subutilizado. Então para otimizar, foi utilizado os dois subsistemas no mesmo PIC, visto que um subsistema não atrapalharia o outro.

3.3.3 PIC16F877A

Todo o gerenciamento dos subsistemas ocorrerá por meio do PIC16F877A por via do protocolo DCP e do protocolo L3 em conjunto com o I2C. A escolha desse microcontrolador foi pela alta frequência de trabalho (20 MHz) e pelo fato de existir o protocolo I2C já implementado em hardware (Technology (2003b)).

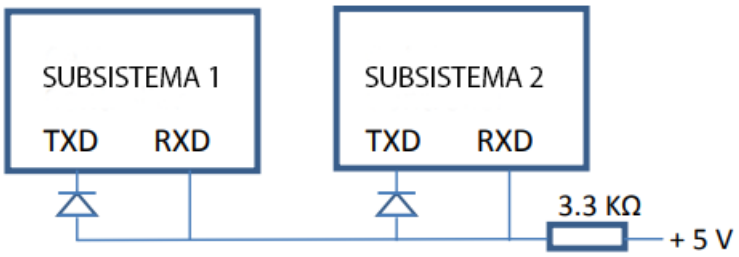
Esse PIC para esse trabalho funcionará como *Master* para a comunicação com os subsistemas pelo protocolo DCP, e como *Slave* na comunicação I2C com o dispositivo externo.

3.4 Utilização do Devices Communication Protocol - DCP

O protocolo DCP apresenta características semelhantes ao CAN (utilização de poucos fios e tratamento de colisão BALDISSERA (2011)), sendo que o DCP utiliza apenas um fio para a transmissão de dados ao invés de dois como no protocolo CAN e I2C. Com isso, perde a proteção contra interferências no barramento, mas facilita pelo fato de não precisar de um transceptor para ler e enviar os dados.

Para a leitura e escrita dos dados no barramento, é utilizado o mesmo processo utilizado para ler o protocolo CAN sem transceptores (Boys (2012)). O esquemático para leitura e escrita do protocolo DCP é vista na figura 18.

Figura 18: Esquemático DCP

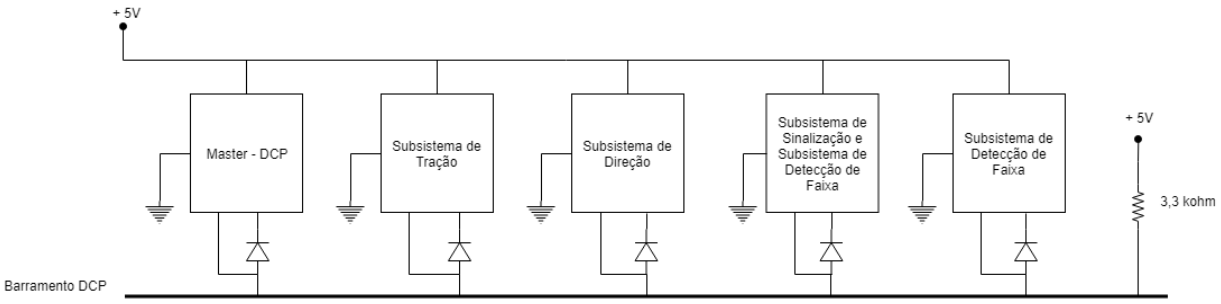


Fonte: Adaptado de Boys (2012)

O TXD será utilizado para a transmissão de dado no barramento colocando o barramento em Low, e o RXD irá ler o barramento

Para essa aplicação, o DCP é utilizado para a comunicação entre 5 microcontroladores PIC (19), quatro microcontroladores trabalharão em 4MHz e o *Master* em 20MHz, sendo assim é atribuído o tempo base de 20 microssegundos baseado nos microcontroladores de 4MHz. Então, os tempos dos pulsos do DCP ficará como na tabela 5.

Figura 19: Barramento DCP



Fonte: Autoria Própria

Tabela 5: Duração Pulsos DCP

PULSOS DCP	DURAÇÃO
START SYNC RECALL	37,5 * TEMPO BASE = 37,5 * 20 us = **750 us**
START SYNC MASTER	25 * TEMPO BASE = 25 * 20 us = **500 us**
START SYNC SLAVE	12,5 * TEMPO BASE = 12,5 * 20 us = **250 us**
START BIT HIGH	7,5 * TEMPO BASE = 7,5 * 20 us = **150 us**
START BIT LOW	7,5 * TEMPO BASE = 7,5 * 20 us = **150 us**
BIT 1	2 * TEMPO BASE = 2 * 20 us = **40 us**
BIT 0	1 * TEMPO BASE = 1 * 20 us = **20 us**
TEMPO EM LOW ENTRE OS BITS	1 * TEMPO BASE = 1 * 20 us = **20 us**

Fonte: Autoria Própria

O formato do pacote consiste em 3 bytes para instruções normais e 4 bytes para o recall. O primeiro byte, chamado de Byte 0, irá indicar a quantidade de bytes que virá no barramento, incluindo o Byte 0. O segundo byte irá indicar o endereço de origem do pacote, e por fim o último byte irá conter a dado (instrução). O formato do pacote é vista na tabela 6.

Tabela 6: Pacote DCP Utilizado

BYTE 0	ENDEREÇO DE ORIGEM	INSTRUÇÃO

Fonte: Autoria Própria

Para o Recall, o formato do pacote muda. O Byte 0 continua indicando quantos bytes serão enviados no pacote, mas agora no segundo byte é enviado o endereço de destino da instrução, no terceiro byte é indicado o endereço da memória EEPROM em que o dado será gravado e por fim no quarto byte é enviado o dado. Na tabela 7 é mostrado o formato do pacote para o Recall.

Tabela 7: Pacote DCP Recall

BYTE 0	ENDEREÇO DE DESTINO	ENDEREÇO DA EEPROM	DADO

Fonte: Autoria Própria

3.4.1 Tratamento de colisão

Quando mais de um subsistema quiser enviar dados no barramento DCP sem que ocorra colisão, é atribuído um tempo (chamado de anti-colisão - t_{ac}), em que cada subsistema irá escutar o barramento por esse tempo, que tem relação com seu endereço, antes de iniciar o envio no barramento. Ou seja, quanto menor o seu endereço, mais prioridade de envio no barramento ele tem. O cálculo do tempo de leitura do barramento está na figura 20.

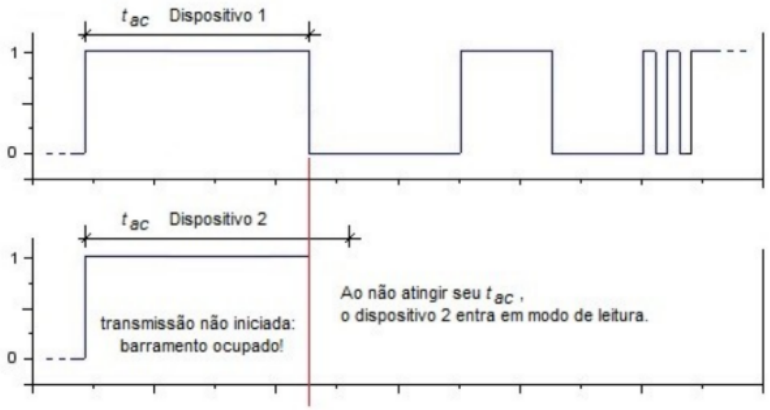
Figura 20: Cálculo do Tempo Anti-Colisão

$$t_{ac} = \frac{(ENDEREÇO + 6) * TEMPO\ BASE}{4}$$

Fonte: DCP

Dessa forma, o quando mais de um dispositivo decidir enviar dados no barramento, o t_{ac} irá seguir como ilustra a figura 21.

Figura 21: Dispositivo 2 reconhece barramento ocupado e entra em modo de leitura.

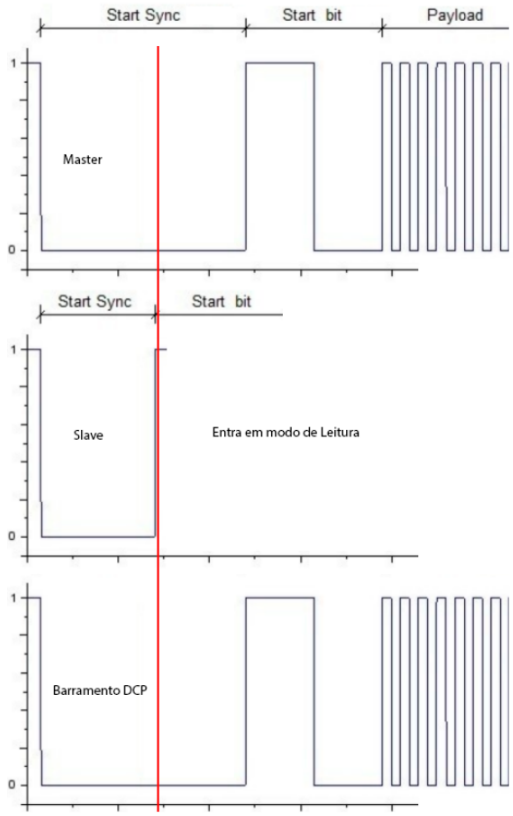


Fonte: DCP

Mesmo com o t_{ac}, diferentes subsistemas podem iniciar a contagem do t_{ac} em momentos diferentes, com isso, pode coincidir de começarem a enviar no barramento no mesmo instante. Para tratar isso, é verificado a ocorrência da colisão ao iniciar o *Start Bit* e durante o envio do bit "1".

A colisão verificada no *Start Bit* ocorre quando um *master* e um *slave* iniciam a transmissão no barramento no mesmo instante, portanto, o que diferencia eles é a duração do *Start Sync*. Ou seja, quando o *slave* terminar de enviar o seu *Start Sync*, o *master* ainda estará enviando o seu, e como o barramento tem como prioridade o tempo em *Low*, ao iniciar o *Start Bit* do *slave*, que deveria está em *High* o barramento, é lido o barramento e verificado que está em *Low*, então o *slave* irá sair do modo de envio e entrar no modo de leitura para receber o pacote do *master*. Na figura 22 ilustra a ocorrência de colisão no *Start Bit*.

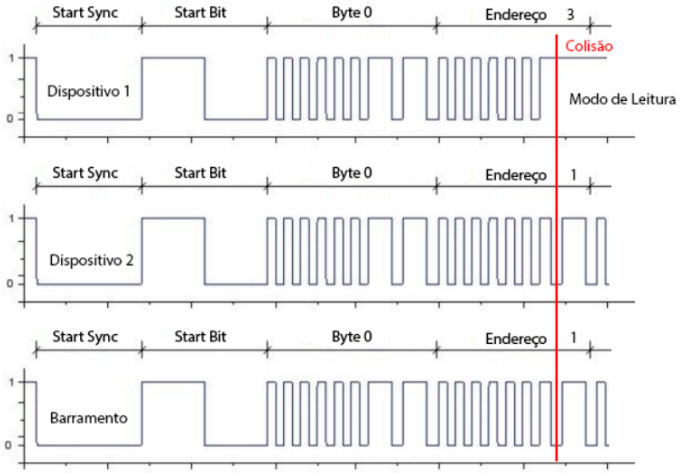
Figura 22: Slave identifica colisão e entra em modo de leitura.



Fonte: DCP

A colisão também pode ocorrer quando dois *slaves* iniciam a transmissão no mesmo instante. Como ambos utilizam o mesmo tempo para o *Start Sync* e *Start Bit*, o que diferenciará eles serão os dados enviados. Como o DCP prioriza o envio do bit "0", pois ele passa menos tempo em *High*, será verificado o estado do barramento durante o envio do bit 1. Quando iniciar o envio do bit 1 e passar 75% do tempo em *High*, é verificado o barramento, se estiver em *Low*, indica que está sendo enviado um bit 0 no barramento, a figura 23 ilustra essa situação. Sendo assim, como o envio dos bits começam do mais significativos bits até os menos significativos, o subsistema que possuir o menor endereço terá prioridade no barramento.

Figura 23: Slaves identificam colisão e o de maior endereço entra em modo de leitura.



Fonte: DCP

3.5 Modelo do Veículo

O modelo do veículo será o reaproveitamento do chassi de um carro de controle remoto em que consiste na representação do modelo Fiat UNO mostrado na figura 24.

Figura 24: Fiat Uno

Fonte: Editada de (FIAT, 2012)

Para seguir a mesma representação do modelo Fiat UNO, será utilizado a sinalização e iluminação nos mesmos locais do modelo original (FIAT, 2012). A representação da iluminação e da sinalização no modelo utilizado pode-se ver na figura 25.

Figura 25: Sinalização e Iluminação

Fonte: Editada de (FIAT, 2012)

Os subsistemas apresentados na figura 17 serão detalhados a seguir.

3.5.1 Subsistema de Iluminação e Sinalização

Com o objetivo de aliviar o microcontrolador *master* das tarefas de controle da iluminação e sinalização do veículo, mostrado na figura 17, essa função será atribuída a um PIC que receberá e responderá aos comandos do *master* através do protocolo DCP, detalhado no item 3.6.

A sinalização e iluminação do veículo ocorrerá de maneira semelhante aos automóveis convencionais (figura 25), que consiste no uso de setas, farol, luz de freio e luz de ré, e sua ativação será proveniente dos comandos recebidos do programa de controle externo.

A codificação do subsistema de iluminação e sinalização esta na faixa entre os valores 64 e 127. Esses valores são identificados pelo bit mais significativo acionado, o bit 6, e o restante dos bits menos significativos são responsáveis pelos faróis, luzes de seta, luzes de freio e luz de ré como mostra na tabela 8.

Tabela 8: Formato das Instruções de Sinalização

X	BIT DE SINALIZAÇÃO	REQUISIÇÃO	SETA DIREITA	SETA ESQUERDA	LUZ DE FREIO	LUZ DE RÉ	FAROL
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Fonte: Autoria Própria

Quando o bit de maior significância acionado for o bit equivalente ao bit de sinalização, o subsistema irá reconhecer como sua instrução e irá tratar o dado para acionar suas funções de acordo com cada bit acionado.

3.5.2 Subsistema de Tração

Partindo da ideia de aliviar o microcontrolador *Master* das tarefas, é utilizado um subsistema para controle da tração do veículo, que ocorrerá por meio um motor DC e uma ponte H (Braga, 2008a).

O motor DC traseiro e a ponte H utilizados para a tração serão reaproveitados do chassi do carro de controle remoto utilizado para esse trabalho. Ele está acoplado a um sistema redutor de velocidade por engrenagens e fornece torque as rodas traseiras (conjunto fixo) em que ambas as rodas se movimentam na mesma velocidade, sem tração diferencial.

Em testes feitos com o carrinho foi alcançado uma velocidade aproximadamente de 2 metros por segundo. Como essa velocidade pode ser alta para essa aplicação, essa velocidade será controlada por meio de PWM (Braga, 2008b), e seu sentido de rotação será controlado por meio da ponte H. O veículo contará com três velocidades, simulando as marchas de um carro convencional, e a marcha ré. Para a marcha ré, será adotada uma velocidade baixa, referente a primeira velocidade dentre as três para o veículo prosseguir.

Como o que é alterado no motor é apenas a velocidade de rotação e não o torque, para o veículo conseguir se movimentar é necessário testes para que se obtenha o melhor do *duty cycle* para cada uma das três velocidades e a ré do veículo.

O conjunto de possíveis instruções do subsistema de tração vai de 8 a 15, mas é formado basicamente por seis instruções, que são os comandos de cada uma das três marchas, a marcha ré, o motor desligado e a instrução de requisição do estado do subsistema pedido pelo *master*. As instruções estão na tabela 9.

Tabela 9: Formato das Instruções de Tração

Instrução	Descrição
8	Parar Motor
9	Marcha 1
10	Marcha 2
11	Marcha 3
12	Marcha Ré
13	Requisição do Estado do Subsistema

Fonte: Autoria Própria

Ao receber essas instruções do *master*, o veículo irá imediatamente executar esse comando, e ao ser requisitado sobre o seu estado, o subsistema irá enviar a instrução referente ao seu estado atual no barramento DCP, onde o *master* irá armazenar esse estado e enviar para controlador externo a informação.

3.5.3 Subsistema de Direção

Aproveitando o sistema de controle de direção do carrinho de controle remoto utilizado no trabalho, a direção do veículo será controlada por um motor DC controlado por uma ponte H. Para que se obtenha uma direção, o motor é acionado para girar em uma direção e travar seu eixo, fazendo com que se tenha um alto consumo de corrente. Para amenizar esse consumo, é utilizado o *Duty Cycle* de 70% para que consiga obter a força necessária para o deslocamento dos pneus dianteiros para um dos lados acionados.

Para o subsistema de direção, é utilizado apenas quatro instruções para o seu controle. Elas são para virar para a direita, virar para a esquerda, seguir em frente (motor desligado) e a requisição. As possíveis codificações de instruções estão entre os valores 0 e 7, mas as implementadas são mostradas na tabela 10 em que detalha as instruções.

Tabela 10: Formato das Instruções da Direção

Instrução	Descrição
4	Seguir em Frente (Motor Parado)
5	Esquerda
6	Direita
7	Requisição do Estado do Subsistema

Fonte: Autoria Própria

Como a velocidade do veículo pode ser alta, o tempo que demoraria para o controle externo fazer a requisição de como está a direção, e depois enviar o novo comando, pode fazer com que o veículo tenha se deslocado mais do que o necessário. Sendo assim, o tempo em que o motor de direção permanecerá acionado será de acordo com qual velocidade o subsistema de tração esteja. Ou seja, quando for enviado um comando de virar para uns dos lados, o veículo virará por um tempo estimado a partir da velocidade e voltará a ficar com o motor de direção desativado.

3.5.4 Subsistema de Colisão

Para a detecção de obstáculos será utilizado um sensor de distância HC-SR04 (FilipeFlop, 2011) que ficará na parte dianteira do veículo (figura 26), ele irá fazer uma nova medida a cada 30 ms, esse tempo poderá ser alterado utilizado o sistema de recall explicado no tópico 3.6.1, e ao fim da medição, enviará no barramento DCP a distância. Sendo assim, o *Master* não precisa enviar uma requisição ao subsistema para saber a distância do obstáculo.

Figura 26: Indicação do Sensor de Distância

Fonte: Editada de (FIAT, 2012)

Como medida de segurança, é escolhida uma distância mínima em que quando o sensor detectar que a distância lida é igual ou inferior a mínima, é enviado uma instrução no barramento DCP em que todos os subsistemas irão ler e executar o comando de alerta, que consiste no veículo parar o motor de tração e ligar as luzes de setas e o freio sinalizando o modo de alerta, sendo capaz de executar somente o comando de ré e de direção para que o veículo saia da distância mínima estabelecida.

Como o envio da medida é de apenas um byte, a medida é limitada a 127 centímetros, preenchendo os sete bits menos significativos, sobrando assim o bit mais significativo para sinalizar uma colisão como mostrado na tabela 11. Como o valor da medida pode coincidir com alguma instrução de outro subsistema fazendo com que ocorra uma leitura de instrução errada, os outros subsistemas apenas verificam o bit mais significativo de dado recebido no barramento DCP é igual a "1". Se for, é caracterizado colisão, se não, é ignorado o endereço do subsistema de colisão e verificado se a instrução é destinada a eles.

Tabela 11: Formato da Instrução de Colisão

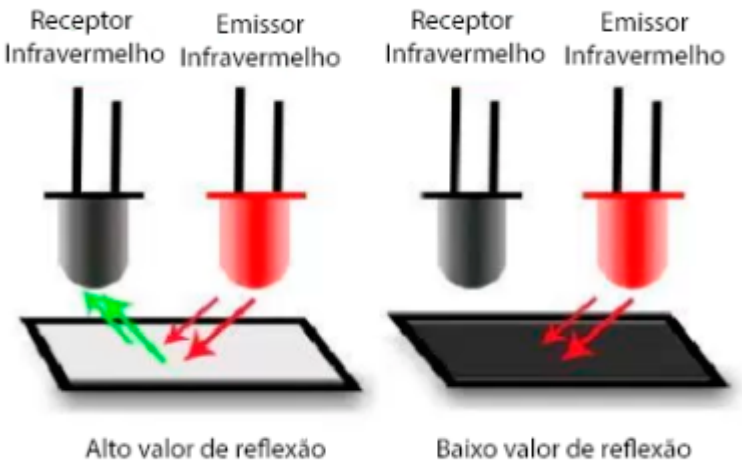
BIT DE COLISÃO	DISTÂNCIA						
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Fonte: Autoria Própria

3.5.5 Subsistema de Detecção de Faixa

O subsistema de detecção de faixa mostrado na figura 17, será composto por um PIC e quatro sensores detectores de faixa. Os detectores utilizarão de leds e sensores infravermelhos que irão detectar uma superfície refletora ou não. Serão utilizados as cores preta (baixo valor de reflexão) e branca (alto valor de reflexão) (Castro et al., 2003) para compor a pista, em que será uma linha preta no fundo branco. Na figura 27 é mostrado como ocorre a reflexão da luz infravermelha na cor branca e na cor preta. Na cor branca, o alto valor de reflexão faz com que a luz emitida seja refletida e detectada pelo receptor infravermelho, o que não acontece com a cor preta, que absorve a luz e não reflete para que o receptor detecte.

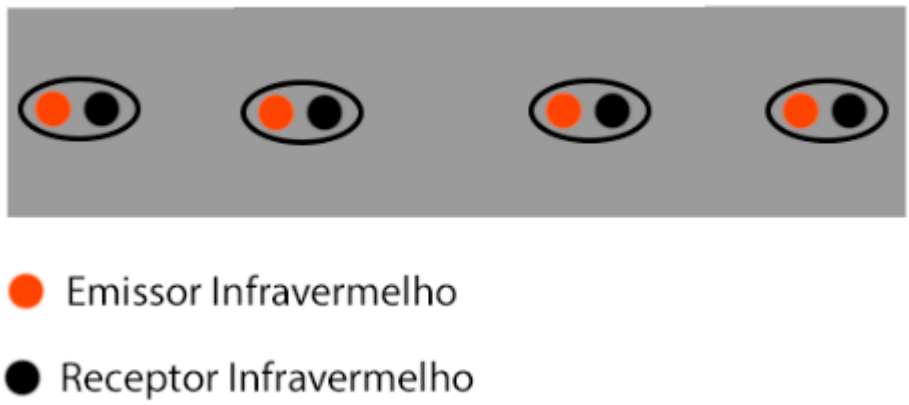
Figura 27: Detecção de faixa



Fonte: Autoria Própria

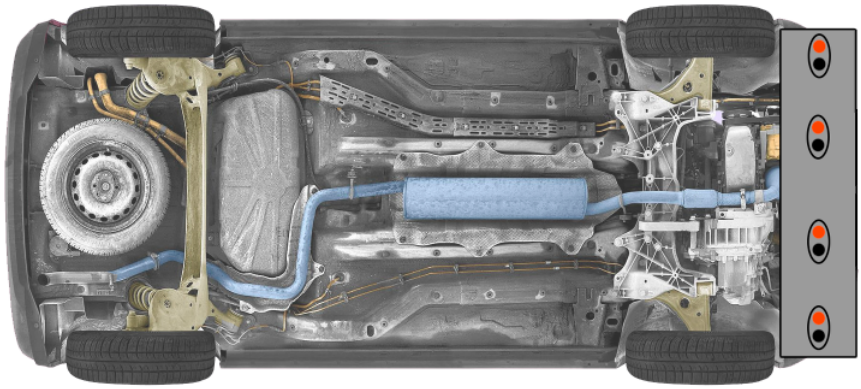
Serão utilizados quatro leds emissores de infravermelho e quatro receptores infravermelho, em que a junção de um emissor com um receptor resultará em um sensor capaz de identificar se há superfície reflexiva ou não. Com isso, serão utilizados dois sensores no centro para a identificação da faixa contínua preta, que indicará o trajeto ao veículo, e os dois sensores restantes serão utilizados na extremidade de cada lado para captar os marcadores presentes na pista. Na figura 28 é representado como será o detector de faixas e na figura 29 onde ele ficará presente no veículo.

Figura 28: Detector de faixa



Fonte: Autoria Própria

Figura 29: Detector de faixa no Carro



Fonte: Autoria Própria

O formato da codificação para o protocolo DCP desse subsistema é visto na tabela 12, em que basicamente cada um dos 4 primeiros bits equivale ao estado de cada sensor respectivamente. Ou seja, quando ocorre a alteração da leitura do sensor, o bit equivalente é alterado e assim que ocorre essa alteração, por meio de interrupção, é enviando no barramento o estado atual de cada sensor. A codificação vai dos valores de 16 até 31.

Tabela 12: Formato da Instrução de Detecção de Faixa

0	0	0	Bit de identificação dos Sensores	Sensor 4	Sensor 3	Sensor 2	Sensor 1
bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Fonte: Autoria Própria

3.5.6 Microcontrolador Master - DCP

Para que se possa melhor aproveitar das vantagens no microcontrolador *Master*, é necessário otimizar o gerenciamento de informações. Para isso, os subsistemas precisam ter seus endereços atribuídos por ordem de prioridade no barramento DCP, em que o menor valor equivale a maior prioridade no barramento, pois a velocidade do veículo junto com a baixa taxa de amostragens de dados da distância de colisão e da pista podem fazer com que ocorra a perda de informações importantes. Tentando amenizar esse problema, foi atribuído por ordem de prioridade os endereçamentos presentes na tabela 13.

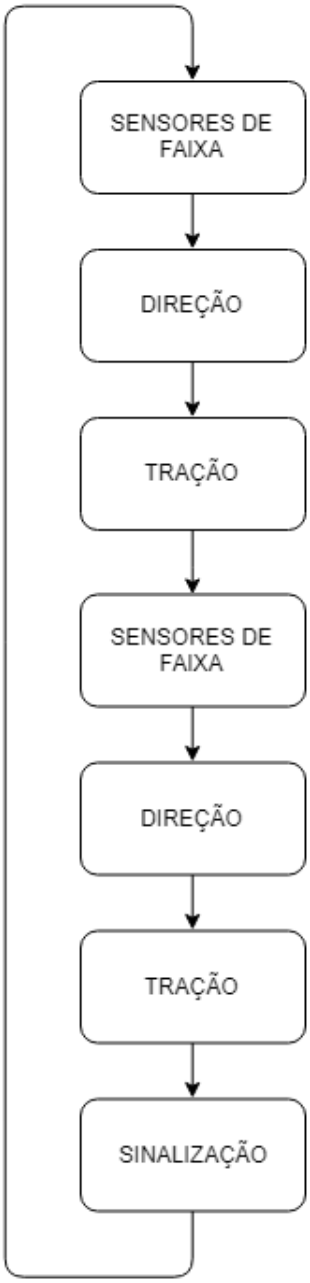
Tabela 13: Endereços dos Subsistemas

Endereço	Subsistema
1	Master
2	Colisão
3	Detecção de Faixa
4	Direção
5	Tração
6	Sinalização

Fonte: Autoria Própria

Para que o *Master* monitore os subsistemas e mantenha o mais atualizado possível os seus dados, é atribuído uma sequência de requisições ao barramento dos estados dos subsistemas. Para isso, os que apresentam os menores endereços (maior prioridade) são requisitados mais vezes durante o ciclo de requisições. Na figura 30 é possível ver a ordem das requisições.

Figura 30: Requisições do Master



Fonte: Autoria Própria

3.5.7 Microcontrolador Slave - I2C

Durante o ciclo da figura 30, o microcontrolador funciona no modo *Slave* com o protocolo I2C para trocar informações com o controlador externo (*Master*) pelo protocolo L3 combinado com o I2C. Os comandos trocados entre eles são mostrados na figura 4,

em que o *Slave* pode enviar as informações dos subsistemas para o controlador externo, como também pode receber os comandos que serão enviados para os subsistemas veículo. Fazendo assim, o controle do veículo será efetuado de acordo com a programação existente no microcontrolador conectado.

3.6 Simbologia para a Pista

A pista de competição utilizará de marcação preta em um fundo claro para a utilização de símbolos e de faixas contínuas para o uso do detector de faixa. O detector de faixa será responsável pela leitura do ambiente e as sinalizações presentes na pista.

Os símbolos utilizados para esse trabalho tiveram como referências simbologias utilizadas em (Rovai, 2016a).

3.6.1 Lado Direito e Esquerdo

A pista contará com duas faixas pretas contínuas em que o veículo poderá se mover e alternar entre as faixas quando necessário, as faixas serão a da direita e da esquerda. Para que o veículo consiga se localizar em qual faixa da pista ele está, é atribuído marcações ao redor das faixas pretas que sinalizam a sua posição. Para a sinalização da faixa direita, é colocado uma marcação do lado direito da faixa a cada 10 cm, enquanto o outro lado da faixa direita continua com o fundo claro, como mostrado na figura 31.

Figura 31: Sinalização Lado Direito

Fonte: Autoria Própria

O detector de faixa mostrado na figura 28 irá ler a faixa preta continua com os 2 sensores centrais, enquanto os sensores nos extremos do detector irão ler as marcações ao redor da faixa continua para as a sinalização presente na pista. Com isso é possível através do programa controlador, a detecção de qual faixa da pista o veículo se encontra.

Para a sinalização da faixa esquerda, é seguida a mesma ideia da sinalização da faixa direita, sendo invertida a sinalização para a faixa continua do lado esquerdo. Podemos ver a sinalização referente ao lado esquerdo na figura 32.

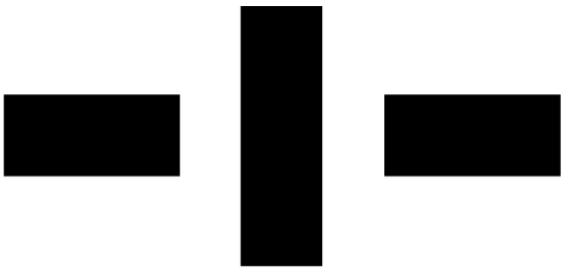
Figura 32: Sinalização Lado Esquerdo

Fonte: Autoria Própria

3.6.2 Parada

Para que o programa controlador consiga detectar o início, fim e uma sinalização que indica parada do veículo, é usado também símbolos que serão lidos pelo detector de faixas. Para a sinalização de parada, é utilizado duas marcações ao redor das faixas continuas (figura 33). Diferente da indicação do lado esquerdo e direito em que as marcações ocorrem apenas em um dos lados de cada faixa, a de parada é utilizado duas marcações ao redor de cada faixa continua, tanto na faixa do lado esquerdo, como na faixa do lado direito, que indica para o programa controlador que o veículo deve parar no momento da leitura.

Figura 33: Sinalização de Parada



Fonte: Autoria Própria

3.7 Função Recall

Durante os testes, é necessário mudar alguns parâmetros dos subsistemas. Para isso, ao invés de regravar o PIC, é implementada a função Recall nos subsistemas que necessitam dessas mudanças. Ela consiste na alteração dos valores gravados na EEPROM dos microcontroladores e reiniciá-los para funcionarem com os novos valores. O comando L3 para o Recall é o E1B mostrado na tabela 4, em que a identificação do subsistema que será submetido a mudança dos parâmetros é identificado no campo IDS do pacote L3 (tabela 4).

3.7.1 Recall no Subsistema de Colisão

Os parâmetros mudados no subsistema de colisão consiste na alteração do valor minimo de que será declarado como colisão, e a sua taxa de amostragem, ou seja, o tempo entre as medidas a serem feitas. Os endereços da EEPROM destinados para esses valores são mostrados na tabela 14

Tabela 14: Endereço EEPROM do Subsistema de Colisão

ENDEREÇO EEPROM	DESCRIÇÃO
2	Limite da Colisão
3	Taxa de Amostragem

Fonte: Autoria Própria

A taxa de amostragem é calculada multiplicando o valor gravado por 10 milissegundos.

Como o *master* não irá fazer requisições ao subsistema de colisão, para que se possa utilizar o recall nesse subsistema, ele ficará os dois primeiros segundos logo após ligar o veículo em modo de espera para o recall e assim alterar os valores. Após os dois segundos, a interrupção é desabilitada e ele irá fazer medidas a cada tempo estimado pela taxa de amostragem alterada.

3.7.2 Recall no Subsistema de Tração

Para o subsistema de tração, os parâmetros mudados são os equivalentes ao duty cycle do PWM do motor, o que muda a velocidade do veículo. Os endereços e as descrições estão na tabela 15.

Tabela 15: Endereço EEPROM do Subsistema de Tração

ENDEREÇO EEPROM	DESCRIÇÃO
2	Duty Cycle Marcha 1
3	Duty Cycle Marcha 2
4	Duty Cycle Marcha 3
5	Duty Cycle Marcha Ré

Fonte: Autoria Própria

3.7.3 Recall no Subsistema de Direção

Para o subsistema de direção, os valores alterados são os tempos em que a direção ficará acionada referente a marcha do veículo. Os endereços e as descrições estão na tabela 15.

Tabela 16: Endereço EEPROM do Subsistema de Direção

ENDEREÇO EEPROM	DESCRIÇÃO
2	Tempo de Duração para a Marcha 1
3	Tempo de Duração para a Marcha 2
4	Tempo de Duração para a Marcha 3

Fonte: Autoria Própria

O valor colocado será multiplicado por 15 milissegundos, esse valor é equivalente ao período do PWM utilizado. Ou seja, para um valor de 10 gravado no endereço 2 da EEPROM, será equivalente a 10 vezes o período do PWM (15 milissegundos) que é igual a 150 milissegundos acionado.

4 APRESENTACÃO DOS RESULTADOS

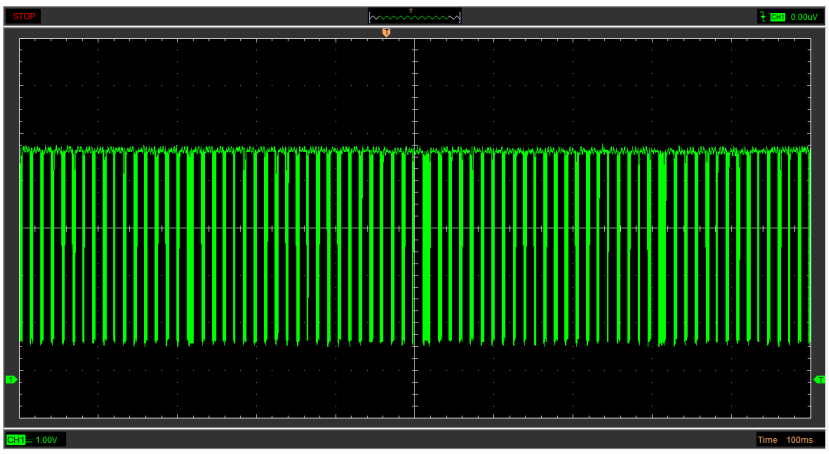
Nessa seção será mostrado o resultado obtido na implementação e utilização do protocolo DCP em conjunto com os subsistemas, a comunicação do protocolo L3 com o I2C e a resposta do veículo aos comandos. Toda a programação utilizada para a implementação dos subsistemas e dos protocolos foi a linguagem assembly para PIC's. O ambiente de desenvolvimento e simulação foi o MPLAB X, programa da própria Microchip. E por fim para a gravação foi utilizado o gravador PicKit 3.

4.1 Aplicação do DCP

Os resultados obtidos com o uso do protocolo DCP se mostraram eficazes, pois ele se mostrou capaz de coordenar os envios de mensagens no barramento mesmo quando ele se encontrava em estresse.

Na figura 34 podemos ver o protocolo funcionando em tempo real no momento em que o *Master* faz requisições a cada 10 milissegundos aos subsistemas, envia os comandos recebidos do I2C E recebe a resposta dos estados dos subsistemas seguindo a ordem mostrada na figura 30.

Figura 34: Master fazendo requisições no barramento DCP e subsistemas respondendo



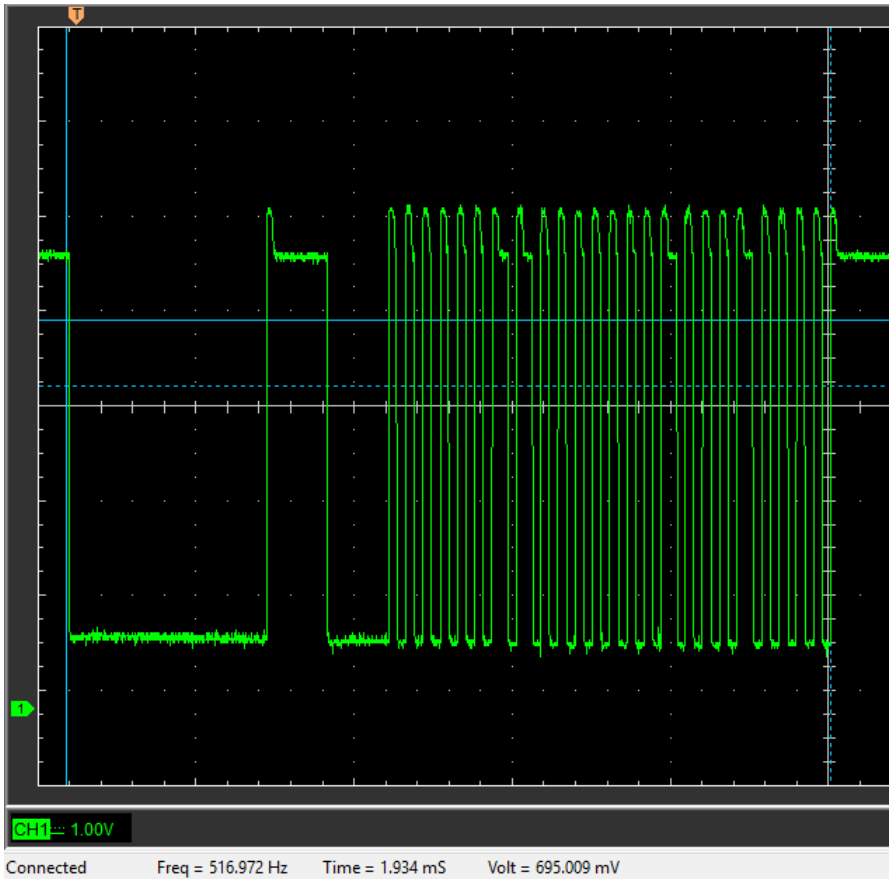
Fonte: Autoria Própria

4.1.1 Velocidade do DCP

Para o envio do *Master*, considerando o pior caso em que todos os bits enviados são "1", o pacote dura em torno de 2240 microssegundos e contém três bytes, ou seja,

o DCP utilizando o tempo base de 20 microssegundos consegue chegar a uma taxa de transmissão de aproximadamente 10,7 kbits/s. Na figura 35 é possível ver um pacote real de requisição ao subsistema de sensoriamento do protocolo e sua duração de 1934 microssegundos.

Figura 35: Duração de um pacote DCP de 1,93 milissegundos



Fonte: Autoria Própria

4.2 Subsistemas

Foram implementados cinco subsistemas mais o *Master*, e em que cada um foi testado valores para se conseguir resultados satisfatórios.

4.2.1 Subsistema de Colisão

Inicialmente, o subsistema de colisão estava pensado com o objetivo de sinalizar para o veículo parar quando atingisse uma distância bem pequena, algo em torno de 5 centímetros de um obstáculo caso o controlador externo não tenha reagido a sua aproximação. Mas com testes feitos com o veículo, foi visto que ele alcança 2 m/s em sua

velocidade máxima, e que quando os motores são parados, ele ainda se move pela inércia em torno de 1 metro aproximadamente. Então, como o veículo não apresentava um sistema de freagem, essa distância de colisão tem que ser maior, algo em torno de uns 40 centímetros, o que já alivia o impacto com o obstáculo, preservando assim o veículo. Essa distância pode ser alterada utilizando o sistema de recall explicado no tópico 3.7.1.

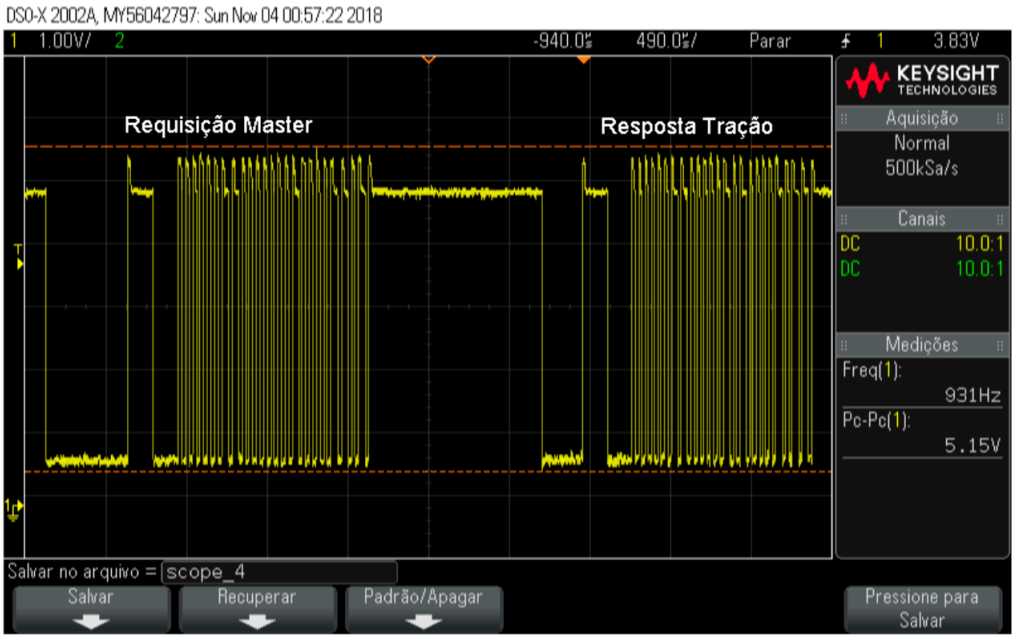
4.2.2 Subsistema de Tração

Para amenizar o fato do veículo não apresentar um sistema de freagem, é possível alterar a velocidade através do PWM alterando o *duty cycle*.

O problema encontrado é que com a mudança da velocidade, também é perdido o torque, e na simulação da primeira marcha, a que deveria ser a de maior força, é a que apresenta o menor torque, ou seja, para que se consiga subir um plano inclinado ou se locomover com seu próprio peso, é atribuído um *duty cycle* mínimo para ser a velocidade 1 e a ré. O melhor valor encontrado para a velocidade 1 foi de 60%. Visando o tipo de terreno e o peso de algum microcontrolador a mais conectado ao veículo, é possível modificar esses valores de velocidade pelo sistema de recall, podendo escolher os melhores valores para cada situação.

A requisição do *master* para o subsistema é mostrado na figura 36.

Figura 36: Resposta do subsistema de Tração para o *master*



Fonte: Autoria Própria

4.2.3 Subsistema de Direção

Como o veículo é um seguidor de faixa, a direção deve responder imediatamente a mudança de estado do sensor, mas como nesse caso o objetivo é um carro controlado por um programa externo, esse controle apresenta um atraso, em que velocidades mais altas pode fazer com que o controlador envie um comando de mudar a direção quando detectar que está saindo da faixa, e quando conseguir enviar o próximo comando de direção, o veículo tenha se locomovido mais que o esperado e saia de rota.

Para tentar amenizar esse problema, quando o programa aciona uma direção, seja ela direita ou esquerda, o veículo vira por um intervalo de tempo baseado na sua velocidade atual, por exemplo, na velocidade 1, o tempo em que os pneus são virados dura em torno de 400 milissegundos por instrução. Então, quando se quiser manter a direção por um período de tempo maior, é só enviar esse comando repetidas vezes antes de que o tempo da direção acabe.

Para cada velocidade é atribuído um tempo para a direção em que pode ser mudada também através do recall. A atribuição de tempo para cada velocidade é mostrada na tabela 17.

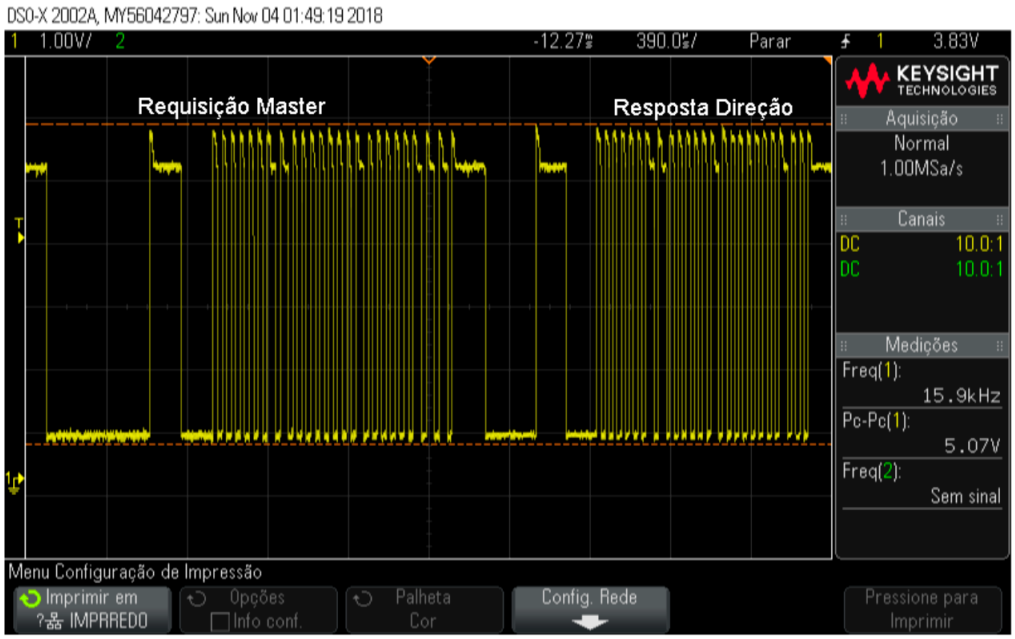
Tabela 17: Duração da mudança de direção de acordo com a velocidade

Tempo da direção	Velocidade
400 milissegundos	1
200 milissegundos	2
100 milissegundos	3
400 milissegundos	Ré

Fonte: Autoria Própria

A requisição do *master* e a resposta do subsistema de direção é visto na figura 37.

Figura 37: Resposta do subsistema de Direção para o *master*



Fonte: Autoria Própria

4.2.4 Subsistema de Iluminação e Sinalização

Para esse subsistema são utilizados cinco pares de leds que indicam o farol, freio, luz de ré, seta esquerda e seta direita. Pare ele não é necessário a alteração de parâmetros como ocorre nos de direção e tração, pois o tempo em que as setas piscam e o *duty cycle* utilizado para o acionamento do freio uma vez configurado, não é mais necessário a sua alteração.

A sua resposta ao *master* é mostrado na figura 38 e o acionamento dos leds se encontra na figura 40.

Figura 38: Resposta do subsistema de Sinalização para o *master*

Fonte: Autoria Própria

4.2.5 Subsistema de Detecção de Faixa

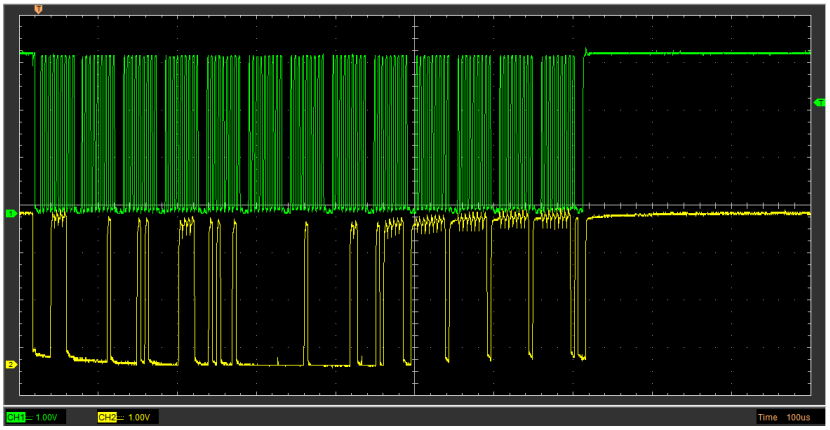
São utilizados quatro sensores de faixa comuns comprados no mercado local. Esses sensores apresentam saída digital, em que é *low* quando não é encontrada a faixa, e *high* na presença dela. Com isso, quando ocorre a mudança do estado desses sensores, é jogado no barramento essa informação e o *master* é encarregado de passar o controlador externo. Para acomodar os sensores, é feita uma placa que também suportará o sensor de faixa do subsistema de colisão, essa placa e as demais são mostradas no tópico 4.4.2.

4.3 Utilização do I2C e L3

Apesar do PIC16F877A ter implementado em hardware o protocolo I2C, ele gera interrupção para que o dado recebido seja tratado. Como o protocolo DCP também está sendo executado nesse PIC, os dois ficam disputando a rotina de interrupção, fazendo com que possa haver conflitos e perda de pacotes. Para que isso seja amenizado, o controlador externo tenta se comunicar com o veículo, caso esteja havendo uma transmissão DCP no momento, o clock do I2C é mantido em *low*, fazendo com que o controlador externo aguarde o pacote DCP terminar e por fim conseguir se comunicar. Mas foi detectado que o uso com um microcontrolador que apresenta o I2C implementado via software, pode apresentar erros de perda de pacotes e até travamento no barramento. Esse problema pode ser resolvido implementando o I2C via software de modo que esteja preparado para essa situação.

Portanto, para essa aplicação é utilizado um microcontrolador que apresenta a sua implementação em hardware, um Arduino Uno, que utiliza o ATmega328 (Technology (2003a)). A exibição do I2C utilizando o L3 é visto na figura 39.

Figura 39: Protocolo I2C com L3



Fonte: Autoria Própria

Na figura anterior é mostrado o pacote de I2C com 13 divisões, em que a primeira é o endereço do veiculo, e os restante são os 12 bytes do protocolo L3.

4.4 Modelo do Veículo

Para a colocação dos subsistemas, sensores e leds, foi necessário fazer algumas furações no veículo. A seguir será detalhado o que foi feito.

4.4.1 Colocação dos LED's

Para facilitar, foi feito quatro placas de PCB com a utilização de uma CNC, duas para a parte da frente, e duas para a traseira.

Após feita a furação, foi encaixado os LED's junto com as placas no veículo, como mostrado na imagem 40.

Figura 40: Frente e Traseira do veículo com os LEDS

4.4.2 Placas Utilizadas

Foram feitas três placas principais, uma para comportar o PIC16F877A e os reguladores de tensão que se encaixam com a outra que comportar os quatro PIC's com os subsistemas (figura 41), e a ultima que é onde se encontram os sensores de faixa e sensor de distância. Na figura 43 é visto o carro com todos os sensores e subsistemas.

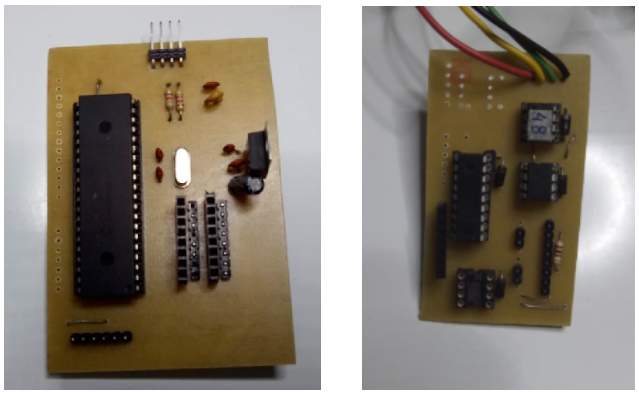
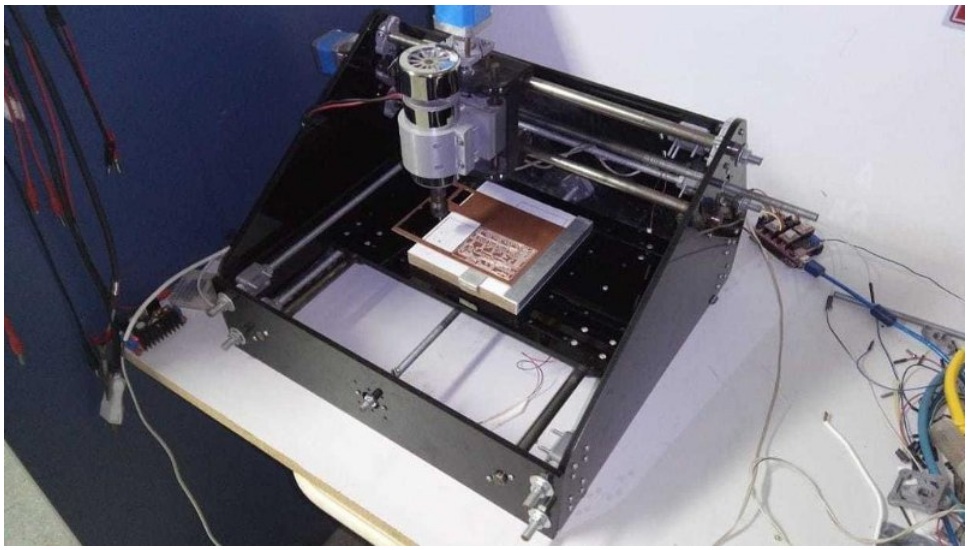


Figura 41: Placas produzidas para o veículo

Todas as placas foram feitas com o uso de uma CNC feita no LMI (figura 42).

Figura 42: CNC



Fonte: Autoria Própria

Após a confecção das placas, foi colocado tudo encapsulado no veiculo (figura 43)

Figura 43: Modelo final do Veículo

CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho teve como objetivo a criação de um veículo capaz de receber comandos externos e gerencia-los internamente através dos protocolos L3, I2C e DCP. Tendo isso, o veículo foi capaz de responder aos comandos recebidos de maneira rápida e se auto gerenciar com a utilização dos protocolos. A utilização do DCP utilizando um fio como barramento, simplifica a conectividade entre os subsistemas e diminui seu custo, pois não necessita de hardware adicional (um transceptor).

Para trabalhos futuros, a utilização de um servo motor no lugar de um motor DC para a utilização de controle da direção, diminui o consumo de energia e se tem um maior controle sobre ele. Também é recomendado a utilização de placas dupla face para blindagem do circuito para evitar interferências. E o isolamento da alimentação entre o circuito e os motores também amenizam as interferências.

É interessante também o uso de um subsistema responsável pelo monitoramento da bateria do veículo, informando a sua carga atual. E também a utilização de um subsistema para medir a velocidade em tempo real.

REFERÊNCIAS

Amorim, Mardson F.; Junior, A. S. M. R. G. S. A. A. A. R. D. W. S. (2005). **Protocolo de comunicação para telemetria v2.0.4**. Protocolo L3.

Amorim, M. F. (2018). **Devices Communication Protocol**. Protocolo DCP.

Aosong, E. (2010). **Temperature and humidity module DHT11 Product Manual**.

Avatefipour, O. and Malik, H. (2017). **State-of-the-Art Survey on In-Vehicle Network Communication "CAN-Bus"Security and Vulnerabilities**. 6:720–727.

BALDISSERA, E. N. (2011). **Decodificador De Dados Usando Protocolo CAN – Padrão SAE J1939**. Universidade de Passo Fundo.

Boys, R. (2012). **CAN Primer: Creating Your Own Network** . Disponível em: <http://www.keil.com/download/files/canprimer_v2.pdf>, Acesso em: 20 de Agosto de 2018.

Braga, N. C. (2008a). **Conheça as pontes H (MEC068a)**. Disponível em: <http://www.newtoncbraga.com.br/index.php/robotica/5166-mec068a>, Acesso em: 23 de maio de 2018.

Braga, N. C. (2008b). **O que é PWM?** Disponível em: <http://www.newtoncbraga.com.br/index.php/robotica/5169-mec071a>, Acesso em: 23 de maio de 2018.

CAMARA, R. C. **Barramento e Protocolo I2C**. Disponível em: <http://www.univasf.edu.br/ romulo.camara/novo/wp-content/uploads/2013/11/Barramentoe-Protocolo-I2C.pdf>, Acesso em: 31 de Outubro de 2018.

Cassiolato, C. (2014). **EMI–Interferência Eletromagnética**. Disponível em: <http://www.profibus.org.br/artigos/EMI_Interferencia_Eletromagnetica.pdf>, Acesso em: 23 de Junho de 2018.

Castro, A. P. d. A. S., Labaki, L. C., Caram, R. M., Basso, A., and Fernandes, M. R. (2003). **Medidas de refletância de cores de tintas através de análise ectral**. Associação Nacional de Tecnologia do Ambiente Construído.

Crossley, R. (2014). **Robôs x empregos: a automação vai fechar mais vagas do que criar?** Disponível em:< http://www.bbc.com/portuguese/noticias/2014/06/140630_robos_empregos_lab>, Acesso em: 23 de maio de 2018.

FIAT (2012). **Manual de Uso e Manutenção Uno - Fiat**. Disponível em: <https://www.fiat.com.br/content/dam/fiat-brasil/manuais-carros/1951520.pdf>, Acesso em: 23 de maio de 2018.

FilipeFlop (2011). **Como conectar o Sensor Ultrassônico HC-SR04 ao Arduino**. Disponível em: <https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/>, Acesso em: 23 de maio de 2018.

Fracarolli, J. P. V. et al. (2012). **Implementação do controle remoto de uma miniatura de carro operado via computador utilizando comunicação wireless.** PhD thesis, UNIVERSIDADE DE SÃO PAULO.

G1 (2018). **O futuro é o carro sem motorista?** Disponível em: <http://g1.globo.com/carros/noticia/2015/11/o-futuro-e-o-carro-sem-motorista.html>, Acesso em: 23 de Junho de 2018.

GUIMARÃES, A. and SARAIVA, A. (2003). **Um Roteiro De Implementação De Uma Rede CAN (Controller Area Network)** . Conferência Internacional de Engenharia Automotiva - SIMEA.

Inc., D. I. Product Manual v1.xEx - 802.15.4 Protocol.

LOZANO-NIETO, A. (1999). **A. Telemetry. In: WEBSTER, J. G. The Measurement, Instrumentation, and Sensors Handbook**. CRC Press.

Martins, R., Koba, F., Neves, J., Lemos, M. A., and Marques, M. A. Um veículo autônomo baseado no modelo freescale.

NXP, S. (2014). I2C-bus specification and user manual. NXP Semiconductors.

POLESE, B. (2017). **Dispositivo De Telemetria Veicular Multiprotocolo Com Transmissão Via Internet**. Universidade de Passo Fundo.

Popova, Y. (2016). **EXPOSING THE SEEDY UNDERBELLIES OF BMWS, MOTORCYCLES, AND LOCOMOTIVES**. Disponível em: <https://www.wired.com/2016/02/urban-insects-yasena-popova/>, Acesso em: 23 de maio de 2018.

Queirós, J. M. R. et al. (2011). **Sistema de sensorização e telemetria de um VEC (veículo eléctrico de competição)**. Faculdade de Engenharia da Universidade do Porto.

Rovai, M. (2016a). **Robô explorador de labirintos, utilizando Inteligência Artificial com Arduino**. Disponível em: <https://mjrobot.org/2016/04/28/robo-explorador-de-labirintos-utilizando-inteligencia-artificial-com-arduino/>, Acesso em: 23 de Junho de 2018.

Rovai, M. J. (2016b). **"Raqueando" o carrinho de controle remoto**. Disponível em: <https://mjrobot.org/2016/05/07/raqueando-o-carrinho-de-controle-remoto/>, Acesso em: 23 de maio de 2018.

Simatupang, J. W. and Yosua, M. (2016). **A Remote Controlled Car Using Wireless Technology**. *Journal of Electrical And Electronics Engineering*, 1(2):pages 56–61.

Siqueira, G. B. (2014). **Telemetria automotiva usando módulo de transmissão Wireless Zigbee**. Universidade Estadual Paulista (UNESP).

Systems, E. (2018). **ESP8266EX Datasheet**. <https://www.espressif.com/sites/default/files/docu esp8266ex_datasheet_en.pdf>.

Technology, M. (2003a). **ATmega48A/PA/88A/PA/168A/PA/328/P**. <http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf>.

Technology, M. (2003b). **PIC16F87XA Data Sheet**. <http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf>.

Technology, M. (2009). **PIC16F627A/628A/648A Data Sheet**. <http://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf>.

Technology, M. (2010). **PIC12F629/675 Data Sheet**. <http://ww1.microchip.com/downloads/en/DeviceDoc/41190G.pdf>.