UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Implementação de um sistema de controle automático de velocidade para veículos

Guilherme Augusto Bileki

Implementação de um sistema de controle automático de velocidade para veículos

Guilherme Augusto Bileki

Orientador: Eduardo do Valle Simões

Monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP - para obtenção do título de Bacharel em Ciências de Computação.

Área de Concentração: Controle e Automação

USP – São Carlos
Outubro de 2015

“A imaginação é mais importante que a ciência,
porque a ciência é limitada,
ao passo que a imaginação
abrange o mundo inteiro”

Albert Einstein

Agradecimentos

A Deus por minha vida, família e amigos.

Aos meus pais, João e Célia, que sempre me incentivaram, apoiaram e me ajudaram na medida do possível.

Aos meus amigos, que me ajudaram a solucionar problemas encontrados neste projeto e por estarem comigo nos bons momentos.

Ao meu orientador Prof. Eduardo do Valle Simões, que me ensinou, motivou e orientou ao longo deste projeto.

Aos professores, pelos ensinamentos técnicos e de vida.

Resumo

O objetivo deste trabalho é desenvolver um sistema de controle automático de velocidade para veículos. Esse sistema deve ser capaz de ler a velocidade do veículo e controlar a aceleração do mesmo para manter essa velocidade de acordo com a especificação do motorista. Para isso, o sistema deve contar com um motor capaz de acionar o cabo do acelerador do veículo.

Esse sistema deve obter a velocidade atual e o estado do pedal do acelerador quando o usuário mantém uma velocidade constante por 10 segundos. A partir daí o sistema entende que deve ficar responsável por manter a velocidade constante até que o pedal do acelerador seja novamente acionado pelo motorista.

O desenvolvimento do sistema é focado em utilizar ferramentas acessíveis de baixo custo. E com elas consegue cumprir o objetivo de manter a velocidade do veículo constante.

Palavras-chave: Controle embarcado, automação, veículo autônomo, piloto automático, controle de velocidade.

Sumário

Lista de Figuras

Lista de Gráficos

Lista de Abreviaturas e Siglas

CaRINA - Carro Robótico Inteligente para Navegação Autônoma

ICMC – Instituto de Ciências Matemáticas e de Computação

DARPA - Defense Advanced Research Projects Agency

OBD-II - On-Board Diagnostics

GPS - Global Positioning System

IDE - Integrated Development Environment

ECU - Engine Control Unit

CAN - Controller Area Network

PWM - Pulse Width Modulation

IoT - Internet of Things

IA - Inteligência Artificial

CAPÍTULO 1: INTRODUÇÃO

1.1. Contextualização e Motivação

A tecnologia tem sido aplicada cada vez mais ao conforto e acessibilidade no transporte, cujo objetivo é oferecer ajuda ao motorista para enfrentar o trânsito e problemas de mobilidade. Por isso, tem aumentado o número de pessoas que procuram veículos com sistemas mais sofisticados em prol de seu conforto, como faróis adaptáveis, advertência de proximidade, sensoriamento que monitora pontos cegos, piloto automático, entre outros (TECNOLOGIA, 2012).

Neste trabalho será abordado um dos itens mais antigos criados com o objetivo de facilitar a vida do motorista, o *cruise control*, comumente traduzido como "piloto automático", que nada mais é do que um sistema que mantém a velocidade do veículo constante. O *cruise control* atualmente desenvolvido para veículos comerciais é mais sofisticado do que o inicialmente concebido na década de 1950 (TEETOR, 1950), contando com o uso de sensores para garantir uma distância segura de outros veículos. E não pode ser comparado aos projetos de veículos autônomos, por exemplo o *Google Self-Driving Car* (GOOGLE, 2015), o CaRINA (Carro Robótico Inteligente para Navegação Autônoma) do ICMC-USP (Instituto de Ciências Matemáticas e de Computação) (CARINA2, 2015) ou os veículos que competem nas competições do DARPA (*Defense Advanced Research Projects Agency*), pois estes são veículos modificados estruturalmente ou construídos com o intuito de serem autônomos.

Itens como o *cruise control* moderno, que além de escolher a velocidade desejada, o motorista deve selecionar também a distância de segurança a ser observada, têm sido implementados de fábrica nos veículos mais novos e luxuosos, e nos modelos mais econômicos, o *cruise control* mais simples (RODAS, 2015). Entretanto, nos modelos mais antigos, este item não vem incluso de fábrica, mas é possível ser adicionado por um valor próximo dos R$ 2.000,00, dependendo do modelo e marca do veículo. Dado o alto custo para a adição do *cruise control* em um veículo, este trabalho visa a construção de um sistema similar, usando itens de baixo custo e com o mínimo de modificações no veículo.

1.2. Objetivo

1.2.1. Objetivo Geral

O objetivo deste trabalho é desenvolver um sistema de controle automático de velocidade para veículos, usando ferramentas acessíveis de baixo custo. Esse sistema deve ser capaz de ler a velocidade do veículo por meio da interface de diagnóstico OBD-II (*On-Board Diagnostics*) e controlar a aceleração do mesmo para manter essa velocidade de acordo com a especificação do motorista. Para isso, o sistema deve contar com um motor capaz de acionar o cabo do acelerador do veículo. As ferramentas selecionadas para o projeto são: um Arduino Pro Mini (ARDUINO, 2015), um módulo Bluetooth (BTH-07, 2015), um motor DC CEP 12V (BOSH, 2015) e um Módulo de diagnóstico OBD-II Bluetooth (OBD, 2015).

O veículo escolhido para esse projeto foi um Mitsubishi Pagero TR4, pois essa marca possui acelerador mecânico controlado por cabo. Isso permite que se controle a aceleração do veículo diretamente no cabo do acelerador, evitando que seja necessário enviar comandos específicos para a ECU (*Engine Control Unit*). Esse sistema obtém a velocidade atual e o estado do pedal do acelerador da interface de diagnóstico do veículo OBD-II. Essas informações são obtidas por meio de um módulo adaptador com Bluetooth (ELM327) conectado à interface OBD-II. O sistema de controle é conectado ao adaptador ELM327 por meio de um módulo Bluetooth HC-05 (GRCBYTE, 2014).

Quanto à operação do sistema, o usuário deve levar o veículo até a velocidade que deseja conduzir, mantendo-a constante (com variação de mais ou menos 4 Km/h) por 10 segundos. A partir daí o sistema emite um sinal sonoro e o pedal do acelerador deve ser liberado. A partir deste momento o sistema entende que deve ficar responsável por manter a velocidade constante até que o pedal do acelerador seja novamente acionado pelo motorista. Desta forma, o sistema de controle automático de velocidade proposto se diferencia dos já existentes, que utilizam botões para se ajustar manualmente a velocidade que o veículo deve seguir. Isso faz com que o controle do veículo seja realizado de forma mais transparente ao usuário (UBÍQUA, 2015).

1.2.2. Objetivos específicos

O sistema de controle automático de velocidade para veículos deve apresentar as seguintes características:

- Ler dados de velocidade do módulo de diagnóstico OBDII Bluetooth ELM327;
- Acessar os dados de velocidade com um micro controlador Atmega328 (ATMEGA328, 2015) programado como Arduino, via módulo Bluetooth BTH-07;
- Controlar a aceleração do motor do veículo exercendo tração no cabo do acelerador por meio de um motor DC 12V;
- Usar PID (*Proportional-Integral-Derivative Controller*) para controlar a velocidade do veículo manipulando a aceleração do mesmo;
- Possuir um sistema de abortar a função de controle de velocidade por meio de um botão de "pânico" que pode ser acionado pelo motorista;
- Ser implementado e calibrado para um veículo Mitsubishi Pajero TR4 Flex modelo 2010;
- Utilizar itens de baixo custo e que exijam o mínimo de modificações no veículo.

1.3. Organização da Monografia

Este trabalho está estruturado em três capítulos, divididos da seguinte forma:

- Revisão bibliográfica: apresentação dos conceitos, ferramentas e técnicas para o entendimento do trabalho;
- Desenvolvimento: apresentação do desenvolvimento dos controladores de velocidade e aceleração, além dos resultados obtidos com os testes;
- Conclusão: validação dos objetivos do trabalho e considerações finais.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

O desenvolvimento desse trabalho envolve a utilização de diferentes tecnologias e componentes utilizados ao longo da implementação do sistema de controle de velocidade. As subseções seguintes apresentam um embasamento teórico sobre essas tecnologias, caracterizando os aspectos mais importantes para a execução deste trabalho.

2.2. Conceitos e Técnicas Relevantes

2.2.1. Controlador proporcional integral derivativo

Um controlador proporcional integral derivativo ou PID calcula continuamente um "valor de erro" como a diferença entre uma medida da variável de processo e um desejado ponto de ajuste. O controlador tenta minimizar o erro ao longo do tempo por ajuste de uma variável de controle que une as ações proporcional, integral e derivativa (PID, 2015). O processo de controle PID é mostrado na Figura 1, na qual o processo gera um valor *y(t)*, adicionando um valor esperado *r(t)* gera-se um erro *e(t)* que passa pelas equações proporcional, integrativa e derivativa, resultando em um valor de ajuste *u(t)* para realimentar o processo, até a estabilização do sistema no valor esperado.

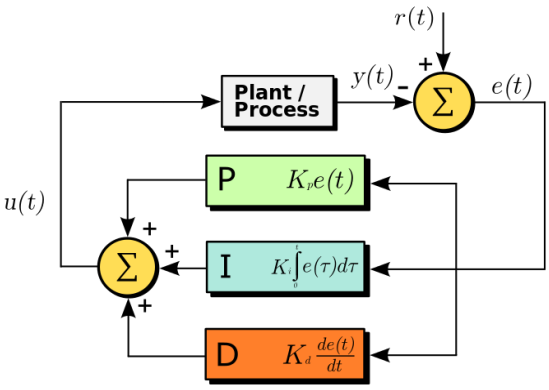

Figura 1: Diagrama de um PID

No caso deste trabalho, o valor *y(t)* é a posição do pedal do acelerador lida em tempo real e o *r(t)* é a posição do pedal do acelerador definida pelo motorista como a que deve ser a mantida constante.

2.2.2. *Cruise Control*

Na década de 1940, Ralph Teetor, inventor cego, construiu o primeiro protótipo de controlador de velocidade (TEETOR, 1950), em Hagerstown, Indiana (Estados Unidos). Como presidente da Perfect Circle Corporation passou os 30 anos seguintes desenvolvendo, testando e abrindo o mercado para controladores de velocidade. Na década de 1960 a Perfect Circle é comprada pela Dana Corporation e posteriormente por divisões da empresa é fundada a Precision Controls Division que é comprada pela Rostra Technologies, tornando-se a Rostra Precision Controls.

No Brasil, em 1997 a Dalgas Precision Equipments inicia suas atividades no mercado brasileiro como representante exclusiva da Rostra Precision Controls para implantar a comercialização do piloto automático e outros produtos fabricados por esta empresa. Com uma pesquisa recente dos usuários do piloto automático da empresa, descobriu-se que os principais motivos para a aquisição do equipamento são: conforto, saúde, economia de combustível e evitar multas de trânsito. (DALGAS, 2015)

2.2.3. Arduino

O Arduino foi desenvolvido com intuito de ser uma plataforma de fácil entendimento, e de fácil programação, sendo multi plataforma e tendo diversos módulos a parte que podem ser acoplados para aumentar suas funções básicas. Uma vantagem desta plataforma é sua grande comunidade que trabalha com a filosofia *open source* (OPENSOURCE, 2015), divulgando projetos gratuitamente, desde os mais simples até projetos bem complexos. De acordo com o site oficial (ARDUINO, 2015), o Arduino é uma plataforma de prototipagem eletrônica de hardware livre, com suporte de entrada e saída embutido e uma linguagem de programação padrão. O objetivo desse projeto de hardware livre é criar ferramentas que são acessíveis, com baixo custo, flexíveis e de fácil utilização por artistas e amadores.

O Arduino consiste de um microprocessador Atmega programado com um software específico desenvolvido pelo usuário em uma IDE (*Integrated Development Environment*) simples que contém as configurações do tipo de *hardware* (modelo do Arduino), porta do computador usada, bibliotecas utilizadas, etc.

2.2.4. Protocolo OBD-II

O OBD (*On-Board Diagnostics*) (OBD, 2015), ou diagnóstico de bordo, é uma interface padrão criada pela indústria automotiva na década de 1990, que permite que qualquer computador acesse e leia as informações processadas pela central eletrônica do veículo. Essas informações podem variar de acordo com o veículo; modelos mais simples terão menos informações que os modelos mais completos. (FLATOUT, 2015)

O protocolo OBD-II, é um aperfeiçoamento do OBD tanto em capacidade quanto padronização. A norma OBD-II especifica o tipo de conector de diagnóstico e sua pinagem, os protocolos de sinalização elétricos disponíveis e o formato de mensagens. A mensagem é baseada em um código de requisição de 4 dígitos hexadecimais precedido de uma letra: P para o motor e transmissão, B para a carroceria do veículo, C para chassis e U para a rede CAN (*Controller Area Network*) (OBD, 2015). Dos 4 dígitos, os dois primeiros definem o modo, descrito na Tabela I do Anexo A. Enquanto os dois últimos dígitos se referem a informação específica que se quer obter, e sua devida resposta, em hexadecimal, é diferente para cada código. Como pode ser visto na Tabela II do Anexo A.

2.2.5. Módulo de diagnóstico *OBD-II Bluetooth* ELM 327

O ELM327 (ELM327, 2014) funciona como uma ponte entre as portas OBD-II e uma interface RS232 padrão. O ELM327 é baseado em outros circuitos integrados, o ELM320, o LM322 e o ELM323 e foram adicionados a ele 7 protocolos CAN. O resultado é um circuito integrado que pode automaticamente perceber e converter a maioria dos protocolos que estão em uso atualmente. (CERQUEIRA, BEZERRA, *et al.*, 2009)

Há várias opções de Módulos com o circuito ELM327, com interfaces via cabos, Módulos WiFi e Módulos Bluetooth. Entre as opções de menor custo está o Módulo

ELM327 Bluetooth, o que o torna muito popular. Após conectado ao veículo, o módulo começa a emitir as informações que o veículo dispõe, e essas informações podem ser resgatadas pelo computador ou pelo celular, com auxílio de softwares que compreendam as informações do protocolo OBD-II ou apenas um monitor serial.

2.3. Trabalhos Relacionados

O documento de título "*Remote Exploitation of an Unaltered Passenger Vehicle*" ("A exploração Remota de um Veículo de Passageiros Inalterado") (MILLER e VALASEK, 2015) é um guia de como os autores Dr. Charlie Miller e Chris Valasek, "hackearam" um Jeep Cherokee 2014, sem modificar a parte mecânica ou arquitetural do veículo, utilizando-se apenas de falhas de software da central multimídia do veículo, que atualmente já foram corrigidas.

O projeto de título "*OBD-II Arduino Car Information Display*" (KONCHA, 2014) propõe um display de diagnóstico em tempo real, mostrando as informações que da ECU do veículo. Porém, a conexão entre o Arduino e o OBD-II é feita via cabo serial, facilitando o processo de comunicação.

O projeto de título "*Adding a bit of Arduino to my old Toyota RAV4*" (BOUGAKOV, 2013), visa a comunicação do do módulo de diagnóstico OBD-II Bluetooth com o Arduino, adicionando algumas funções por meio de dispositivos adicionais, como GPS (*Global Positioning System*).

O projeto de título "OBDII HC-05" (GRCBYTE, 2014) é um guia de comunicação entre o módulo ELM327 e o módulo Bluetooth HC-05 conectado ao Arduino. Nele é descrito passo-a-passo como fazer a comunicação entre os módulos, como interpretar os dados provindos do ELM327 e exemplos de código para Arduino que tratam essas informações.

2.3.1. Projeto do Carro Autônomo da Google

O "*Google self-driving car Project*" (GOOGLE, 2015), como mostrado na Figura 2 é um dos exemplos mais famosos de direção autônoma, pois seu grupo de desenvolvimento

conta com engenheiros que participam dos desafios da DARPA (uma série de corridas de veículos autônomos organizada pelo Governo dos EUA).

Figura 2: Carro Autônomo da Google

O projeto aborda mais que uma central multimídia ou piloto automático, pois o veículo é construído sob medida para atender aos requisitos de segurança e autonomia, além de ter sensores de alto nível de complexidade. Por exemplo, o sistema de transmissão elétrica é limitado a uma velocidade máxima de 40 km/h, o para-brisa é flexível e a parte frontal é feita de espuma para amortecer o choque no caso de uma colisão com pedestre ou ciclista e há dois sistemas diferentes que controlam direção e freio, mas ainda falta um controle manual. (GIZMODO, 2015)

2.3.2. CaRINA 2

O CaRINA é um projeto com uma proposta parecida com a da Google, entretanto o veículo utilizado é um modelo padrão e menos modificado, ou seja, um veículo já comercial com modificações para que seja autônomo. O CaRINA 2 (CARINA2, 2015) é o segundo protótipo do projeto, mostrado na Figura 3, e conta com mais sensores e tecnologia mais nova. Ainda está em fase de desenvolvimento, mas tem sido testado em situações controladas e obtido bons resultados (OLIVEIRA, 2013).

Figura 3: CaRINA 2

2.4. Considerações Finais

Cada um dos projetos relacionados citados contribuiu para o desenvolvimento deste trabalho, tanto como fonte de informação sobre novas tecnologias e componentes, como base de codificação para o software do Arduino. Segue o capítulo de desenvolvimento propriamente dito.

CAPÍTULO 3: DESENVOLVIMENTO

Para desenvolver este trabalho, dois sistemas distintos foram desenvolvidos: sistema de controle de aceleração e sistemas de controle de velocidade. Ambos unidos após testados separadamente. O diagrama geral pode ser visto na Figura 4.

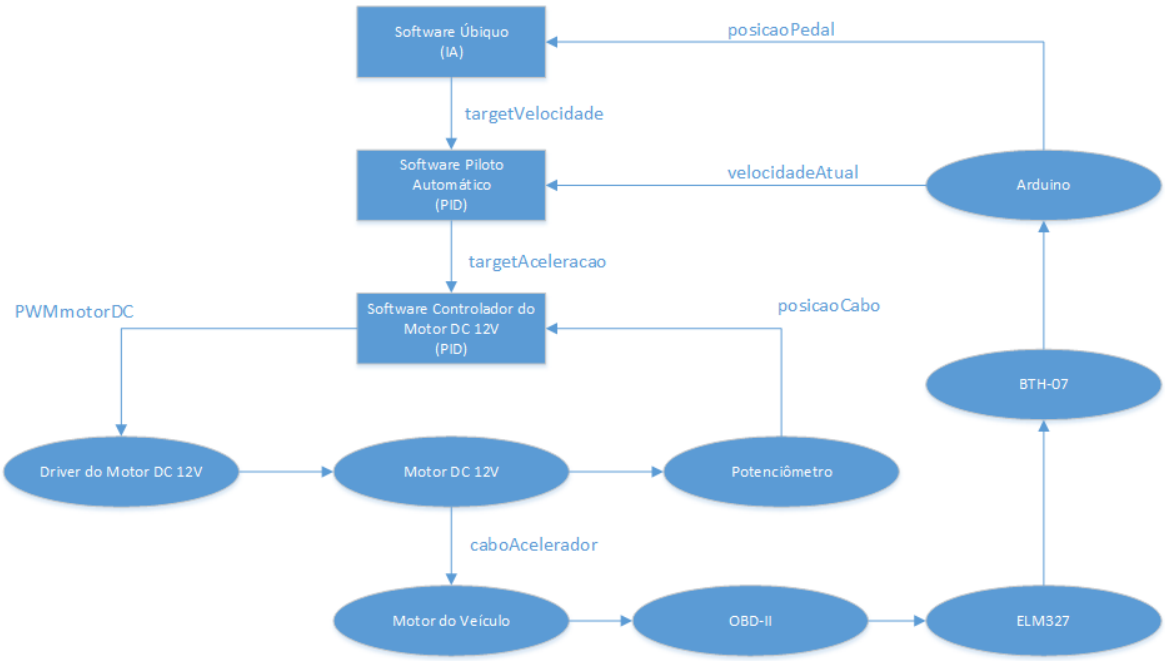

Figura 4: Diagrama do Projeto

O diagrama mostra o fluxo de informações dentro do sistema geral, cada retângulo representa um sistema desenvolvido neste trabalho, enquanto as elipses representam os componentes físicos utilizados.

Para descrever o desenvolvimento do trabalho como um todo, esta seção será dividida em 2 subseções: Desenvolvimento do sistema do controlador de aceleração e Desenvolvimento do sistema do controlador de velocidade.

3.2. Desenvolvimento do sistema do controlador de aceleração

Como não foi possível adquirir um servo-motor com torque suficiente para acionar o cabo do acelerador do veículo, foi utilizado um motor DC com maior potência, controlado por um software capaz de acioná-lo como se fosse um servo, ou seja, especificando-se uma determinada posição para o eixo. O software e a montagem física para simular o servo-motor, estão descritos nas próximas subseções.

3.2.1. Componentes para simular o servo-motor

Para simular um servo-motor, foi utilizado um Motor DC CEP 12V (BOSH, 2015), mostrado na Figura 5. Para acioná-lo foi utilizado o circuito de um *driver* de motor DC desenvolvido em outro Trabalho de Conclusão de Curso (LIMA, 2015). Ele é constituído de uma ponte H, que é um circuito eletrônico que permite que o micro controlador forneça a corrente necessária para o funcionamento do motor de corrente contínua nos dois sentidos com velocidade variável. Este *driver* pode ser visto na Figura 6.

Figura 5: Motor CEP 12V da BOSCH

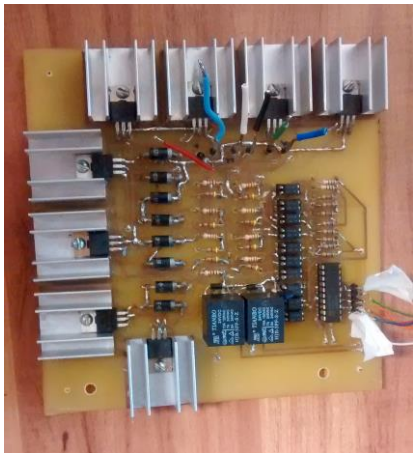

Figura 6: Controlador de Motor 12V

No motor foi fixado um potenciômetro para obter leituras da posição do eixo do motor e este sitema fixado no veículo ao lado do cabo do acelerador. Foi conectado ao eixo do motor um cabo preso na alavanca do cabo do acelerador, como mostrado na Figura 7.

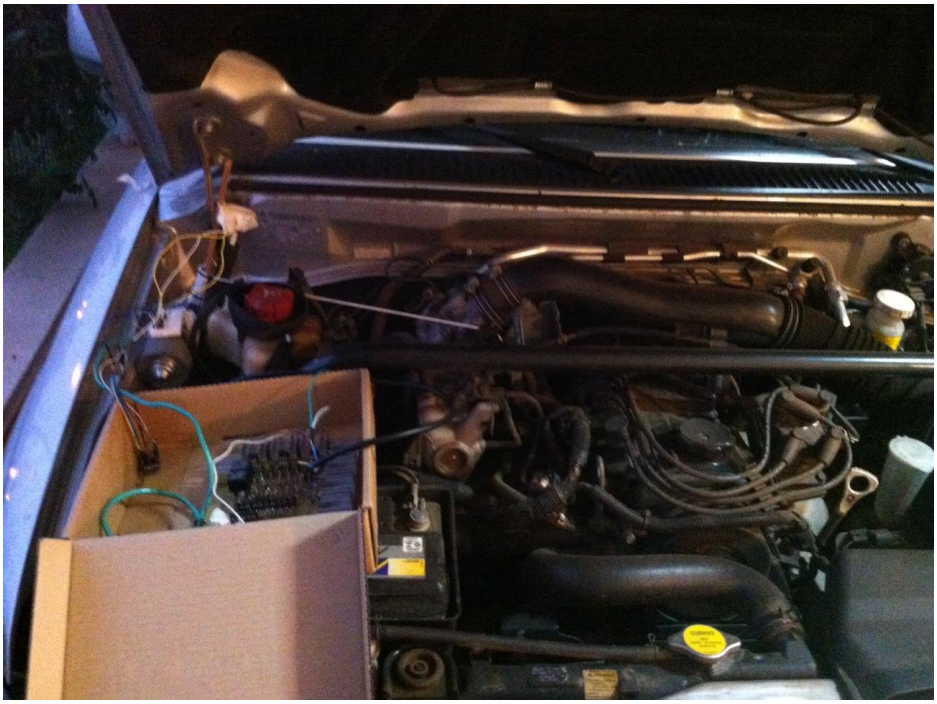

Figura 7: Montagem física dos componentes

3.2.2. Software de controle

O software para simular o "servo-motor" foi feito em Arduino e consiste de uma função PID (no diagrama, o bloco "Software Controlador do Motor DC 12V") que recebe os parâmetros "targetAceleracao" e "posicaoCabo", que são o valor da posição desejada do cabo do acelerador e a posição efetiva do cabo lida pelo potenciômetro, respectivamente. Esta função foi modificada para usar apenas as constantes *Kp* e *Ki*, pois são utilizadas faixas de tolerância, em que o fator proporcional controla o valor de rotação do eixo até entrar na faixa, e dentro dela o fator integrativo estabiliza o valor da posição do eixo. Esses valores são enviados ao *driver* do motor como PWM (*Pulse Width Modulation*) ao Motor

DC 12V, que rotacional seu eixo e realimenta o sistema PID. O código pode ser visto no Apêndice A.

3.3. Desenvolvimento do sistema do controlador de velocidade

O bloco “Software Piloto Automático” do diagrama geral, pode ser visto na Figura 8. Seu desenvolvimento e software serão descritos nas subseções a seguir.

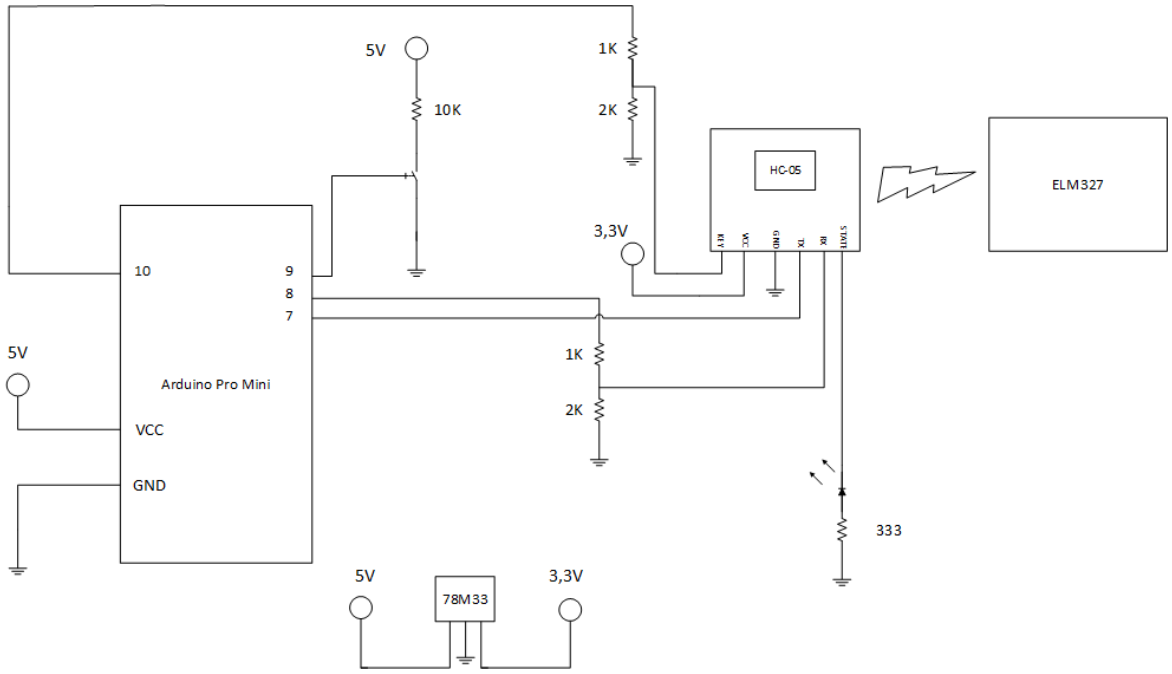

Figura 8: Circuito do sistema de piloto automático

3.3.2. Pesquisa de viabilidade

A pesquisa começou com o propósito de comunicar a ECU sem modificações estruturais no veículo, o que levou ao diagnóstico OBD-II. Dos diversos tipos de conexão OBD-II, a mais viável e de baixo custo encontrada foi o Módulo Bluetooth 2.0 ELM327, que pode ser adquirido no Brasil e por aproximadamente R$ 25,00 (LIVRE, 2015), ao contrário dos cabos Serial/OBD-II ou os Módulos WiFi/OBD-II de maior custo e mais difíceis de serem encontrados.

Com a compra do módulo e um aplicativo para smartphone gratuito, o Torque (PLAYSTORE, 2015), que mostra um monitor de informações do veículo de forma gráfica e personalizável, foi descoberto que o projeto era viável, pois sem alterações no veículo é possível obter informações sobre velocidade, posição do pedal do acelerador, níveis de combustível, entre outros, entretanto havia limitações, que no caso do ELM327 eram de ter acesso a apenas algumas informações do veículo, sem a possibilidade de enviar ou alterar qualquer uma delas. Entretanto a impossibilidade de enviar comandos para a ECU do veículo pode ser considerada como uma vantagem, pois impede comandos que possam alterar o comportamento do veículo em movimento e causar acidentes.

3.3.3. Testes de comunicação entre o ELM327 e o veículo

Com o aplicativo Torque os dados eram capturados em quando o veículo era ligado, mas ao desligá-lo as informações eram perdidas, sendo recuperadas apenas ao ligar o motor, o que se mostrou um problema.

Em paralelo, outros aplicativos foram testados, dentre eles, o Terminal ELM327 (PLAYSTORE, 2013), que é um console serial, em que depois de pareado com o Módulo ELM327, pode-se enviar códigos de requisição de dados. Este aplicativo teve os mesmos problemas de pareamento que o Torque, com o adendo de quando o veículo era desligado, ao religá-lo, não se comunicava mais ao módulo. Este problema foi resolvido posteriormente com a definição do protocolo correto do veículo, ao invés de defini-lo como automático.

3.3.4. Comunicação entre o Arduino e ELM327

A pesquisa para esta etapa foi muito problemática, pois apesar da grande quantidade de material disponível de projetos envolvendo a comunicação entre Arduino e Bluetooth, é escassa a documentação sobre o Módulo Bluetooth HM-10 (JNHUAMAO, 2014), que era o módulo incialmente disponível para o projeto.

Com a montagem de um circuito simples para regular a tensão da fonte de alimentação para o módulo, foi possível testá-lo. Entretanto, ao testar o pareamento com o ELM327, não foi obtido sucesso, levando a pesquisas sobre o problema.

Foi descoberto que o módulo Bluetooth 4.0 não conseguia se comunicar com um dispositivo ELM327 Bluetooth 2.0 e, portanto, foi necessária a troca do módulo HM-10 pelo BTH-07, que é composto pelo HC-05, um módulo Bluetooth 2.0. Após a troca e uma nova implementação de comandos para o novo módulo, como segue no Apêndice A o problema de comunicação entre o ELM327 e o Arduino foi solucionado.

3.3.5. Testes de comunicação entre o Arduino e ELM327

Após muitos testes e pesquisa nos manuais do ELM327, na composição do protocolo OBD-II e fóruns de eletrônica, foi descoberto que o problema da perda de comunicação era devido ao modo de identificar o protocolo de comunicação do veículo, que é definido como “automático” por padrão, e assim só consegue identificar o protocolo correto na primeira vez que o veículo é ligado. Portanto, foi definido forçadamente o protocolo ATSP5 (Protocolo 5, válido para o modelo Mitsubishi Pajero TR4 Flex), o que resolveu os problemas de comunicação e pareamento.

3.4. Resultados Obtidos

Com a integração dos sistemas de controle de aceleração e de velocidade, o ciclo geral foi fechado e pode-se testar o sistema geral. Para fins de teste, o “Software Ubíquo” do diagrama geral foi simulado entrando com parâmetro “targetVelocidade” de forma manual pelo terminal serial. Este parâmetro indica a velocidade que o motorista quer manter constante servindo de valor objetivo para o potenciômetro.

A entrada via serial foi testada com diversos valores e produziu os 4 gráficos a seguir. No Gráfico 1 pode-se analisar 2 curvas, que são respectivamente os valores: Posição relativa do pedal do acelerador e Posição desejada do pedal. O gráfico mostra um teste de aproximadamente 4 minutos, em que foram dados alvos entre 20% e 100%. Como pode ser vista nas duas curvas, o controlador de posição tenta adequar a curva azul

seguindo a laranja, usando uma estratégia linear, para uma melhor visualização, o Gráfico 2 mostra um zoom do processo de estabilização, que no caso não necessita de muita rapidez, por se tratar de uma tarefa que demanda muito torque.

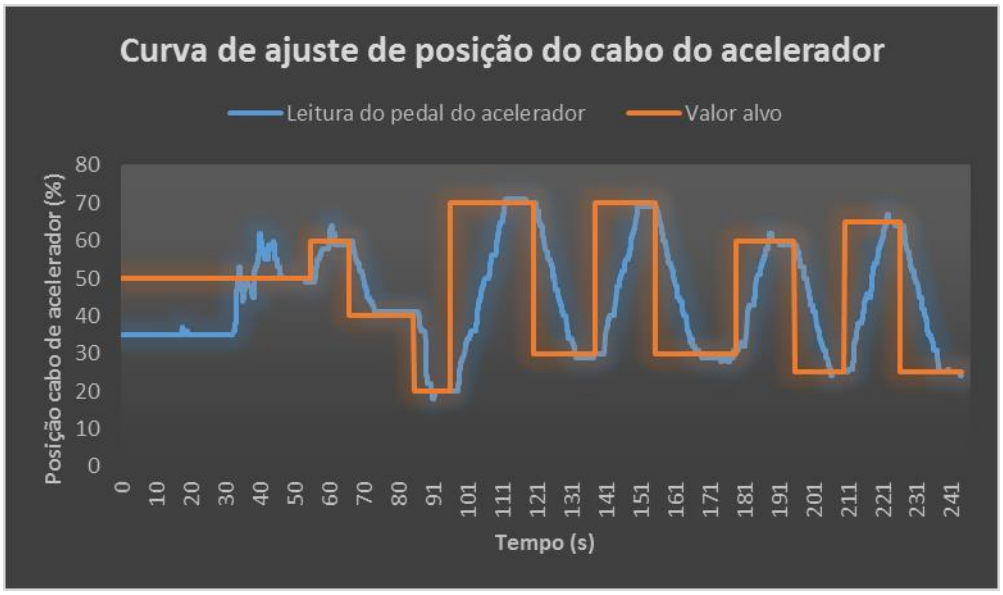

Gráfico 1: Curva de ajuste de posição do eixo do cabo do acelerador

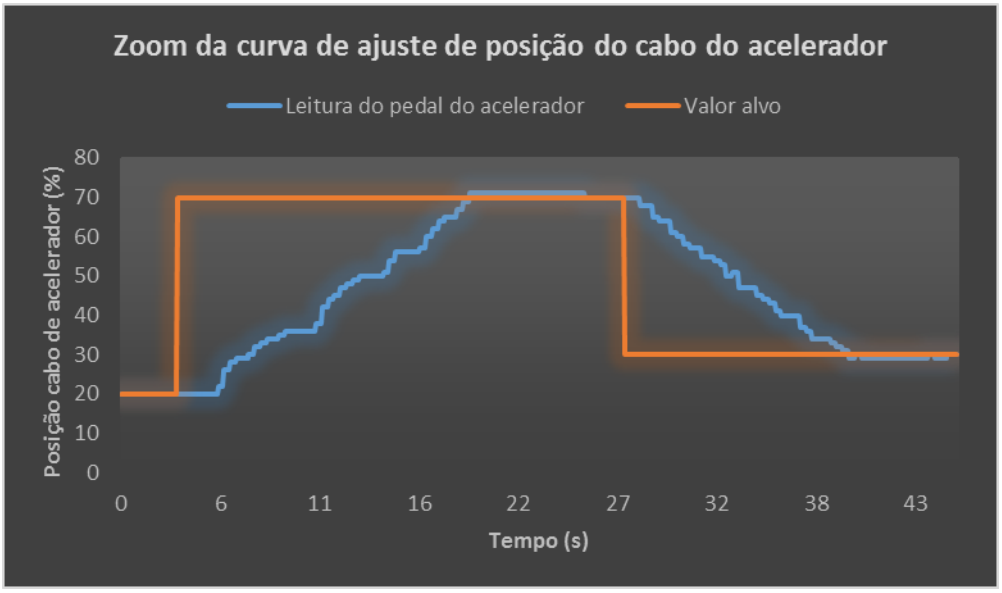

Gráfico 2: Zoom da curva de ajuste de posição do cabo do acelerador

O valor da posição do pedal alimenta a função de PID que controla a rotação do eixo do motor DC 12V, sendo o alvo da mesma, no caso, o parâmetro “targetAceleracao”.

É visível no Gráfico 3, o comportamento do controlador de PID, que tem uma curva característica, em que o valor lido vai se aproximando do valor alvo e quando chega, ultrapassa a curva fazendo com que o fator integrativo entre em ação, estabilizando a curva.

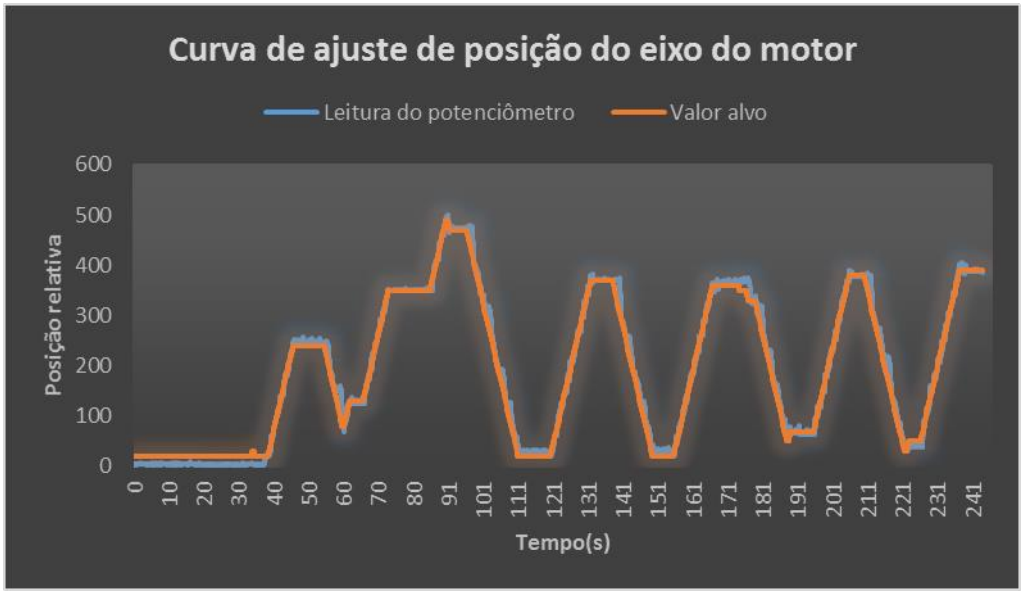

Gráfico 3: Curva de ajuste de posição do eixo do motor

3.5. Dificuldades, Limitações e Trabalhos Futuros

Das dificuldades do desenvolvimento do trabalho, pode-se considerar a aquisição de componentes não comercializados, ou de difícil acesso no Brasil, o que fez com que parte do projeto ficasse parada até a chegada dos componentes necessários. Além da falta de componentes, a documentação dos mesmos é precária ou até mesmo incorreta, como no caso do manual dos módulos Bluetooth que continham comandos que não existiam ou com muito pouca documentação.

Das limitações para o desenvolvimento, pode-se considerar a falta de recursos para a aquisição de alguns componentes, como o servo-motor e o cabo serial/OBD-II que facilitariam o desenvolvimento do projeto, entretanto o seu alto custo limitou o desenvolvimento do trabalho.

Como trabalho futuro, pode-se aprimorar o sistema com os componentes citados (o servo-motor e o cabo serial/OBD-II) transformando a implementação do sistema para: servo-motor, Arduino e cabo serial/OBD-II. Assim as vantagens dessa implementação seriam: a diminuição do ruído causado tanto na leitura do potenciômetro, como no circuito eletrônico; diminuição do risco de danificação do sistema; e estética. Outra possibilidade é integrar as informações do sistema com a Internet, ou seja, integrando o veículo ao IoT (Internet of Things) e as aplicações disso são incontáveis, por exemplo: integrar informações de GPS, mapas e aplicativos de trânsito como o Waze (WAZE, 2015), utilizando recursos de trânsito para regular a velocidade do veículo e evitar infrações, considerando a sinalização de trânsito; ou usar as informações de taxa de combustível do veículo para alertar o motorista de um posto de combustível próximo com preços mais baixos.

3.6. Considerações Finais

Dos módulos descritos no diagrama geral, apenas o sistema ubíquo não foi implementado, pois sua implementação é um vislumbre do sistema final e para fins de teste foi facilmente substituído por comandos diretos. Portanto o ciclo dos sistemas de controle foi fechado com a integração dos sistemas de controle de aceleração e velocidade, com isso os testes foram possíveis. Mesmo com algumas limitações e problemas no desenvolvimento, o sistema final proposto foi concebido e funcionou como esperado. O próximo capítulo aborda a conclusão do trabalho.

CAPÍTULO 4: CONCLUSÃO

O *cruise control* é um sistema simples e viável de ser reproduzido com materiais de baixo custo. Contudo esses materiais demandam mais cuidado e mais componentes de suporte para exercer as mesmas tarefas. No caso desse trabalho, a falta de um servo-motor e um cabo serial/OBD-II, demandaram um conhecimento adicional sobre funcionamento de motores e comunicação Bluetooth. Entretanto, a utilização desses materiais mais caros para o desenvolvimento do sistema de piloto automático ainda é mais viável economicamente do que a implantação do sistema oficial instalados pela concessionária.

Com os testes mostrados nos resultados obtidos, foi possível a calibração das constantes do PID, das tolerâncias e dos tempos de espera otimizados para o veículo de teste, dado que os tempos de resposta da comunicação entre o Módulo ELM327 e o Arduino fossem respeitados. Para que o sistema seja implementado em outro veículo, pequenas modificações devem ser feitas, pois para cada veículo pode haver alteração nos tempos de resposta, força de torque da alavanca do cabo do acelerador, entre outras.

Para o desenvolvimento completo do trabalho, não foram o bastante os conhecimentos inicialmente previstos, foi necessária muita pesquisa para resolução de problemas encontrados sobre os equipamentos utilizados e formas de contornar problemas de comunicação entre os dispositivos Bluetooth, por falta de documentação dos mesmos.

Após a realização dessas considerações, pode-se afirmar que o sistema cumpriu o objetivo proposto, além de ser uma opção mais viável economicamente do que o sistema disponível na concessionária, tendo seu custo menor até com os materiais mais caros: o servo-motor e o cabo serial/OBD-II. Mas por questões de tempo e dificuldades de testes com o veículo em movimento, os objetivos que visavam o "*panic button*" e o acionamento por meio de IA (Inteligência Artificial) não puderam ser implementados neste trabalho.

Houveram algumas contribuições de pessoas externas ao projeto, como colegas de turma, que ajudaram nas pesquisas iniciais sobre os equipamentos usados, alunos de pós-graduação com maiores conhecimentos em controladores PID e manuseio de motores, colegas de outros cursos que cederam alguns materiais para testes e/ou para o projeto final.

4.1. Contribuições

Este projeto produziu um protótipo de sistema de controle automático de velocidade de veículos, caracterizando-se por ter fechado o laço de controle responsável por ler a velocidade atual do veículo a partir da interface de diagnóstico OBD-II e controlar o cabo do acelerador do motor do veículo para manter a velocidade desejada. Apesar de não terem sido realizado testes de campo, conduzindo o veículo em movimento em estrada por questões de segurança, este projeto deixa documentado todo um processo de solução dos problemas encontrados para que todos os módulos do sistema pudessem se comunicar e trocar informação. As informações contidas neste trabalho devem servir de ponto de partida para que em trabalhos futuros uma versão mais robusta das partes eletrônicas e mecânicas do sistema possam ser implementadas e então testadas com mais segurança no veículo em movimento.

4.2. Considerações sobre o Curso de Graduação

Este trabalho não aborda assuntos restritos à Ciência da Computação, por se tratar de um projeto mecânico, eletrônico e de computação aplicado ao controle de veículos automotivos. O uso do micro controlador Arduino necessita da programação em linguagem C, que é uma especialidade da abordagem do curso de Bacharelado em Ciências de Computação e, portanto, sem esse conhecimento, o processo de desenvolvimento do sistema seria mais complexo. Apesar de o foco do curso não ser a eletrônica, algumas matérias de início do curso abordaram o tema, algumas delas ministradas pelo orientador deste trabalho, que foram de essencial importância para o embasamento teórico e prático deste trabalho.

Considerando que um Bacharel em Ciências de Computação tem uma gama de possibilidades de trabalho nas mais diversas áreas, o foco maior em desenvolvimento de software desanima alguns alunos que tem afinidade por hardware. Além de que o desenvolvimento voltado a hardware tem sido cada vez menor, devido à facilidade e ao baixo custo de se obter circuitos complexos prontos do exterior. Uma universidade como a USP, ainda mais no campus de São Carlos, com tanta produção científica voltada à

tecnologia, deveria investir mais em manter o aprendizado voltado também à área de hardware.

REFERÊNCIAS

(78M33, 2003) 78M33. 78M33 Datasheet. **Site da All Datasheet**, 2003. Disponivel em: <http://www.alldatasheet.com/view.jsp?Searchword=78M33>. Acesso em: 4 Novembro 2015.

(ALIEXPRESS, 2015) ALIEXPRESS. Arduino Pro Mini. **Site do Aliexpress**, 2015. Disponivel em: <http://pt.aliexpress.com/item/Free-Shipping-1pcs-lot-ATMEGA328P-Pro-Mini-328-Mini-ATMEGA328-5V-16MHz-for-Arduino/32340811597.html?spm=2114.02020208.3.1.02O5s6&ws_ab_test=searchweb201556_3_79_78_77_91_80,searchweb201644_5,searchweb201560_9>. Acesso em: 4 Novembro 2015.

(ARDUINO, 2015) ARDUINO. Arduino. **Site do Arduino**, 2015. Disponivel em: <https://www.arduino.cc/>. Acesso em: 4 Novembro 2015.

(ATMEGA328, 2015) ATMEGA328. ATmega328. **Site da Atmel**, 2015. Disponivel em: <http://www.atmel.com/pt/br/devices/ATMEGA328.aspx>. Acesso em: 4 Novembro 2015.

(BOSH, 2015) BOSH. CATÁLOGO 2004 | 2005 de Motores Elétricos. **Site da Bosh**, 2015. Disponivel em: <http://www.bosch.com.br/br/negociosindustriais/produtos/peqPorte/pg/pdf/catalogomt.pdf>. Acesso em: 4 Novembro 2015.

(BOUGAKOV, 2013) BOUGAKOV. Adding a bit of Arduino to my old Toyota RAV4. **Site do Bougakov**, 2013. Disponivel em: <http://arduino.bougakov.com/post/43975903095/inventory-bluetooth-module-bth-07>. Acesso em: 4 Novembro 2015.

(BTH-07, 2015) BTH-07, M. Bluetooth Modem - Minimum pass-through module BTH-07. **Elecfreaks**, 2015. Disponivel em: <http://www.elecfreaks.com/store/bluetooth-modem-minimum-passthrough-module-bth07-p-229.html>. Acesso em: 4 Novembro 2015.

(CARINA2, 2015) CARINA2. Projeto CaRINA 2. **Site do LRM**, 2015. Disponivel em: <http://lrm.icmc.usp.br/web/index.php?n=Port.ProjCarina2Info>. Acesso em: 4 Novembro 2015.

(CERQUEIRA, BEZERRA, *et al.*, 2009) CERQUEIRA, A. D. et al. Sistema De Diagnóstico Para Veículos que Utilizam Os Protocolos ISO 9141 e ISO 14230 Através De Uma Plataforma em Labview. **Site da FATEC SAnto André**, 2009. Disponivel em: <http://www.fatecsantoandre.com.br/sdpv.pdf>. Acesso em: 4 Novembro 2015.

(DALGAS, 2015) DALGAS. História do piloto automático (Cruise Control) ou controle de velocidade. **Site da Dalgas**, 2015. Disponivel em: <http://www.dalgas.com.br/4/40-historia-do-piloto-automatico-cruise-control-ou-controlador-de-velocidade>. Acesso em: 4 Novembro 2015.

(ELM327, 2014) ELM327. ELM327 OBD to RS232 Interpreter. **Elmelectronics**, 2014. Disponivel em: <http://elmelectronics.com/DSheets/ELM327DS.pdf>. Acesso em: 4 Novembro 2015.

(FLATOUT, 2015) FLATOUT. Como transfromar seu smartphone em um computador de bordo. **Site do FlatOut!**, 2015. Disponivel em: <http://www.flatout.com.br/como-transformar-seu-smartphone-em-um-computador-de-bordo/>. Acesso em: 4 Novembro 2015.

(GIZMODO, 2015) GIZMODO. Novo carro do Google não tem volante e dirige sozinho. **Site do Gizmodo**, 2015. Disponivel em: <http://gizmodo.uol.com.br/prototipo-carro-google/>. Acesso em: 4 Novembro 2015.

(GOOGLE, 2015) GOOGLE. Google Self-Driving Car Project. **Site do Google Self-Driving Car Project**, 2015. Disponivel em: <https://www.google.com/selfdrivingcar/>. Acesso em: 4 Novembro 2015.

(GRCBYTE, 2014) GRCBYTE. OBDII HC-05. **Site do Grcbyte**, 2014. Disponivel em: <https://sites.google.com/site/grcbyte/electronica/arduino/obdii-bluetooth>. Acesso em: 4 Novembro 2015.

(JNHUAMAO, 2014) JNHUAMAO. Bluetooth 4.0 BLE module. **Site da SeeedStudio**, 2014. Disponivel em: <https://www.seeedstudio.com/wiki/images/c/cd/Bluetooth4_en.pdf>. Acesso em: 4 Novembro` 2015.

(KONCHA, 2014) KONCHA. http://konchatech.blogspot.com.br/2014/02/obd-ii-arduino-car-information-display.html. **Site do Koncha Tech**, 2014. Disponivel em: <http://konchatech.blogspot.com.br/2014/02/obd-ii-arduino-car-information-display.html>. Acesso em: 4 Novembro 2015.

(LIMA, 2015) LIMA, F. M. B. **Carro Inspetor Rádio Controlado Com Transmissão De Imagem Em Tempo Real Para Inspeção De Locais De Difícil Acesso**. São Carlos: [s.n.], 2015.

(LIVRE, 2015) LIVRE, M. Mini Scanner Elm327 Automotivo Obd2 Bluetooth. **Site do Mercado Livre**, 2015. Disponivel em: <http://produto.mercadolivre.com.br/MLB-714500185-mini-scanner-elm327-automotivo-obd2-bluetooth-_JM>. Acesso em: 4 Novembro 2015.

(MILLER e VALASEK, 2015) MILLER, C.; VALASEK, C. Guide to Remote Car Hacking by Charlie Miller and Chris Valasek. **Site do SecurityZap**, 2015. Disponivel em: <http://securityzap.com/remote-car-hacking-charlie-miller-chris-valasek/>. Acesso em: 4 Novembro 2015.

(MITSUBISHI, 2015) MITSUBISHI. Mitsubishi Cruise Controls. **Site The Cruise Control Store**, 2015. Disponivel em: <http://www.thecruisecontrolstore.com/mitsubishi/>. Acesso em: 4 Novembro 2015.

(OBD, 2015) OBD. On-board diagnostics. **Site da Wikipedia**, 2015. Disponivel em: <https://en.wikipedia.org/wiki/On-board_diagnostics>. Acesso em: 4 Novembro 2015.

(OLIVEIRA, 2013) OLIVEIRA, M. Carro sem motorista. **Site da Revista pesquisa FAPESP**, 2013. Disponivel em: <http://revistapesquisa.fapesp.br/2013/11/18/carro-sem-motorista/>. Acesso em: 4 Novembro 2015.

(OPENSOURCE, 2015) OPENSOURCE. Open Source Initiative. **Site da Open Source Initiative**, 2015. Disponivel em: <http://opensource.org/>. Acesso em: 4 Novembro 2015.

(PID, 2015) PID. PID controller. **Site da Wikipedia**, 2015. Disponivel em: <https://en.wikipedia.org/wiki/PID_controller>. Acesso em: 4 Novembro 2015.

(PLAYSTORE, 2013) PLAYSTORE. ELM327 Terminal. **Site da Playstore**, 2013. Disponivel em: <https://play.google.com/store/apps/details?id=Scantech.Terminal&hl=pt_BR>. Acesso em: 4 Novembro 2015.

(PLAYSTORE, 2015) PLAYSTORE. Aplicativo Torque. **Site da Playstore**, 2015. Disponivel em: <https://play.google.com/store/apps/details?id=org.prowl.torque&hl=pt_BR>. Acesso em: 4 Novembro 2015.

(RODAS, 2015) RODAS. Relaxe.O piloto Assumiu. **Site da Revista Quatro Rodas**, 2015. Disponivel em: <http://quatrorodas.abril.com.br/reportagens/novastecnologias/conteudo_143946.shtml>. Acesso em: 4 Novembro 2015.

(TECNOLOGIA, 2012) TECNOLOGIA. A tecnologia a serviço da eficiência, do conforto e da segurança. **Blog dO Mundo em Movimento**, 2012. Disponivel em: <http://omundoemmovimento.blogosfera.uol.com.br/2012/12/21/a-tecnologia-a-servico-da-eficiencia-do-conforto-e-da-seguranca/>. Acesso em: 4 Novembro 2015.

(TEETOR, 1950) TEETOR, R. R. **Speed Control Device for Resisting Operation of the Accelerator**. 2519859, 22 August 1950.

(UBÍQUA, 2015) UBÍQUA. Computação ubíqua. **Site da Wikipedia**, 2015. Disponivel em: <https://pt.wikipedia.org/wiki/Computa%C3%A7%C3%A3o_ub%C3%ADqua>. Acesso em: 4 Novembro 2015.

(WAZE, 2015) WAZE. Waze Mobile. **Site do Waze Mobile**, 2015. Disponivel em: <https://www.waze.com/pt-BR>. Acesso em: 4 Novembro 2015.

APÊNDICE A – Código do Arduino

```
#include <SoftwareSerial.h>
#include <Timer.h>

#define R 2
#define PWM 3
#define L 4
#define MIN 40
#define SETTOL 10

float ontem = millis();

int i, j, posicaoCabo = 0;
int tolerancia = 0, toleranciaki = 20;
int targetAceleracao = 500, targetVelocidade = 50;
float soma = 0, FposicaoCabo = 0, lastError = 0;
float kc = 0.2, ki = 0.05, kd = 0;

Timer t;
SoftwareSerial mySerial(7, 8); // RX, TX

String check = "";
char c;
int flag = 0, sair = 0;
unsigned int throttle;

void setup() {
   Serial.begin(38400);
   pinMode(R, OUTPUT);
   pinMode(L, OUTPUT);
   pinMode(PWM, OUTPUT);
```

```
t.every(300, getThrottle);

mySerial.begin(38400);
delay(800);
mySerial.flush();
Serial.flush();

while (1) {
   if (mySerial.available())
   {
      delay(100);
      //transfere o buffer do mySerial para a string check
      while (mySerial.available() > 0)
      {
         c = mySerial.read();
         check += c;
         flag = 1;
      }
      if(flag==1) {
         Serial.print(check);
         Serial.write("\n");
         throttle_calc();
         check.remove(0);
         flag = 0;
      }
   }

   if (Serial.available()) {
      delay(100);
      while (Serial.available() > 0)  //transfere o buffer do mySerial para a string check
      {
```

```
            c = Serial.read();
            check += c;
            flag = 1;
          }

          if(flag==1) {
            if (check == "sair") {
              mySerial.write("0111\r\n");
              check.remove(0);
              mySerial.flush();
              break;
            }

            Serial.print(check);
            Serial.write("\n");
            mySerial.flush();
            check += "\r\n";
            mySerial.print(check);
            check.remove(0);
            flag = 0;
          }
        }
    }
}

void loop() {
//media dos valores obtidos pra diminuir o ruido
    FposicaoCabo = analogRead(A0);
    for (i = 0; i < 50; i++) {
        for (j = 0; j < 10; j++) {
            FposicaoCabo += analogRead(A0);
        }
```

```
        FposicaoCabo = FposicaoCabo/11;
    }
    posicaoCabo = (int)(FposicaoCabo);

    if (Serial.available() > 0) {
        delay(10);
        targetVelocidade = (Serial.read() - '0') * 100 + (Serial.read() - '0') * 10 +
(Serial.read() - '0');

        if (targetVelocidade < 20)
            targetVelocidade = 20;

        if (targetVelocidade > 80)
            targetVelocidade = 80;
    }

    int vel = (int)PID();

    t.update();

    if ((posicaoCabo > (targetAceleracao+toleranciaki)) || (posicaoCabo <
(targetAceleracao-toleranciaki))) {
        soma = 0;
    }

    if (soma > 10)
        soma = 10;
    if (soma < -10)
        soma = -10;

    if (posicaoCabo < targetAceleracao-tolerancia) {
        //somando o valor do vel com o MIN para ele rodar, caso seja menor q MIN
```

```
        vel=vel+MIN;
        Serial.print(" R ");
        Serial.print(vel);

        digitalWrite(L, HIGH);
        digitalWrite(R, LOW);
        analogWrite(PWM, vel);

    } else if (posicaoCabo > targetAceleracao+tolerancia) {
        //como o vel fica negativo, subtraimos o MIN pra aumentar o valor negativo
        vel = vel-MIN;
        Serial.print(" L ");
        Serial.print(vel);

        digitalWrite(R, HIGH);
        digitalWrite(L, LOW);
        //multiplicamos por -1 pois o pwm precisa ser positivo
        analogWrite(PWM, -1*vel);
        //tolerancia para manter estabilizado em uma certa faixa

    } else {
        digitalWrite(PWM, 0);
    }

    if (posicaoCabo == targetAceleracao)
        tolerancia = SETTOL;

    if ((posicaoCabo > targetAceleracao +tolerancia) || (posicaoCabo < targetAceleracao -
tolerancia))
        tolerancia = 0;

    Serial.println();
```

```
}

float PID() {
   float error, integral, proportional, derivative, dt;

   dt = (millis() - ontem)/1000;
   ontem  = millis();

   error = targetAceleracao-posicaoCabo;
   soma = soma + (error * dt);

   proportional = error * kc;
   integral = ki * soma;
   derivative = kd * ((error - lastError)/dt);

   lastError = error;

   return proportional + integral + derivative;
}

float ajuste() {
   if(throttle < targetVelocidade-1) {
      if (targetAceleracao > 20)
         targetAceleracao = targetAceleracao - 10;
   }
   else if(throttle > targetVelocidade+1) {
      if (targetAceleracao < 980)
         targetAceleracao = targetAceleracao + 10;
   }
}

//THROTTLE LOAD
```

```
void throttle_calc(){
   boolean valid;

   valid=false;

   if ((check[5]=='4') && (check[6]=='1') && (check[8]=='1') && (check[9]=='1')){
      valid=true;
   } else {
      valid=false;
   }

   if (valid){
      String loadHex(check[11]);
      String loadHex2(check[12]);

      String loadHexTotal=loadHex+loadHex2;
      int DecimalDecode=hexToDec(loadHexTotal);
      throttle=round((float(DecimalDecode)/255)*100); //Arredonda e devolve valor final
   }

}

unsigned int hexToDec(String hexString) {
   unsigned int decValue = 0;
   int nextInt;

   for (int i = 0; i < hexString.length(); i++) {
      nextInt = int(hexString.charAt(i));
      if (nextInt >= 48 && nextInt <= 57)
         nextInt = map(nextInt, 48, 57, 0, 9);
      if (nextInt >= 65 && nextInt <= 70)
         nextInt = map(nextInt, 65, 70, 10, 15);
```

```
        if (nextInt >= 97 && nextInt <= 102)
            nextInt = map(nextInt, 97, 102, 10, 15);

        nextInt = constrain(nextInt, 0, 15);
        decValue = (decValue * 16) + nextInt;
    }

    return decValue;
}



void getThrottle() {
    while (mySerial.available() > 0)
    {
        c = mySerial.read();
        check += c;
        flag = 1;
    }

    if(flag==1) {
        throttle_calc();
        ajuste();
        check.remove(0);
        mySerial.flush();
        mySerial.write("0111\r\n");
        flag = 0;
    }
}
```