UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Implementação de um sistema de controle automático de velocidade para veículos com inteligência

Guilherme Augusto Bileki

Implementação de um sistema de controle inteligente de velocidade para veículos com inteligência

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
Maio de 2016

“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 e maus 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 inteligente 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 àquela desejada pelo motorista. Para isso, o sistema deve contar com um servo-motor capaz de acionar o cabo do acelerador do veículo e também obter a velocidade atual e o estado do pedal do acelerador por meio de uma interface com a CPU do veículo. Quando o usuário mantém uma velocidade constante por 4 segundos, o sistema entende que deve ficar responsável por manter essa velocidade até que o pedal do acelerador seja novamente acionado pelo motorista.

O desenvolvimento do sistema é focado em utilizar ferramentas de código livre e dispositivos eletrônicos de baixo custo para manter a velocidade do veículo constante de forma pervasiva sem que o motorista tenha que acionar botões ou desviar sua atenção da atividade de conduzir o veículo.

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

LED - Light Emitting Diode

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 *Defense Advanced Research Projects Agency* (DARPA), pois estes são veículos modificados estruturalmente ou construídos com o intuito de serem autônomos.

Itens como o *cruise control* moderno, no qual além de escolher a velocidade desejada o motorista deve selecionar também a distância de segurança a ser observada por meio de algum dispositivo de interface entre o motorista e o sistema, tem sido implementado de fábrica nos veículos mais novos e luxuosos. E nos modelos mais econômicos, o *cruise control* mais simples que somente controla a velocidade (RODAS, 2015). Entretanto, nos modelos mais antigos, este item não vem incluso de fábrica, mas é possível ser adicionado por terceiros por um valor próximo dos R$ 3.000,00 (MITSUBISHI, 2015), 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, utilizando itens de baixo custo, sem a necessidade de dispositivos de interface entre o motorista e o sistema, e com o mínimo de modificações no veículo.

1.2. Objetivo

1.2.1. Objetivo Geral

O objetivo deste trabalho é desenvolver um sistema inteligente capaz de controlar a velocidade de veículos, utilizando ferramentas de código livre e dispositivos eletrônicos de baixo custo para manter a velocidade do veículo constante de forma pervasiva sem que o motorista tenha que acionar botões ou desviar sua atenção da atividade de conduzir o veículo. Esse sistema deve ser capaz de ler a velocidade do veículo por meio da interface de diagnóstico *On-Board Diagnostics* (OBD-II) 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 servo-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 servo-motor Tower Pro MG996 (HOBBYKING, 2015)e um Módulo de diagnóstico OBD-II Bluetooth ELM327 (ELM327, 2014).

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 *Engine Control Unit* (ECU). 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 3 Km/h) por 4 segundos. A partir daí o sistema emite um sinal luminoso e o pedal do acelerador deve ser liberado em até 2 segundos. 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, ultrapassando a aceleração atual controlada pelo sistema. 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 pervasiva ao usuário.

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 servo-motor 15 kg/cm;
- Usar um controlador *Proportional-Integral-Derivative Controller* (PID) 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 possível 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 do sistema, 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.

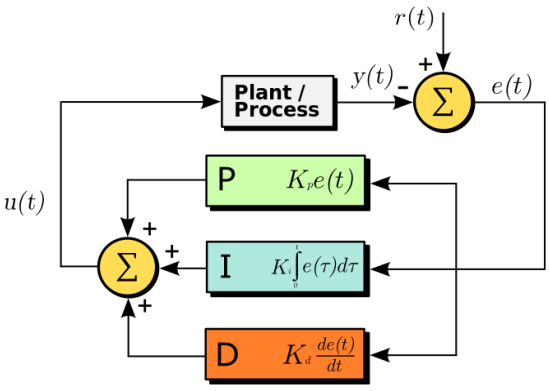

Figura 1: Diagrama de um PID

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, 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 diminuição de multas de trânsito (RODAS, 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 *Integrated Development Environment* (IDE) 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 (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 *Controller Area Network* (CAN). Dos 4 dígitos, os dois primeiros definem o modo. 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.

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 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 velocidade e sistema pervasivo. O diagrama geral pode ser visto na Figura 4.

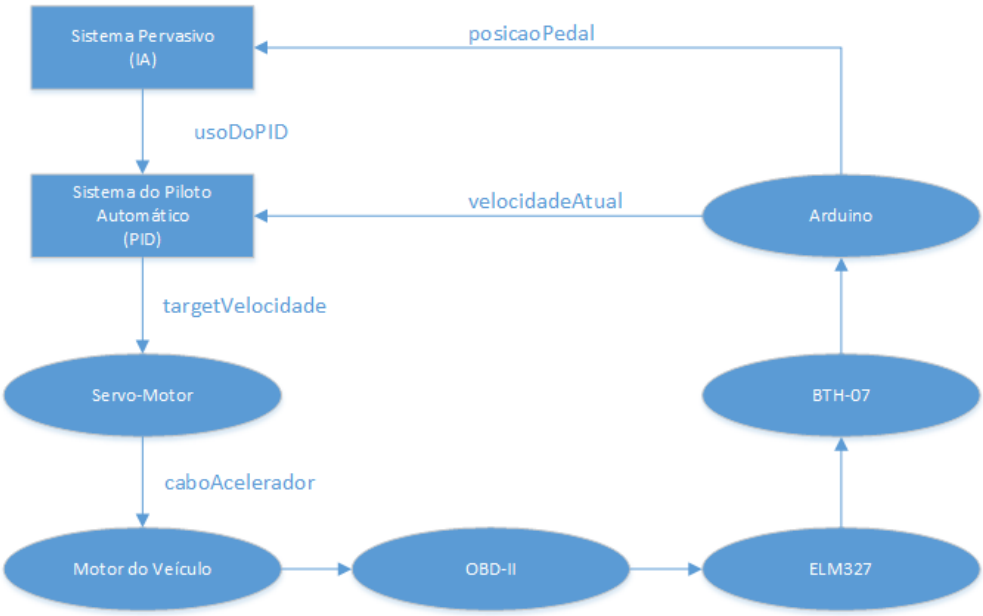

Figura 4: Diagrama do Projeto

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 duas subseções: Desenvolvimento do sistema do controlador de velocidade e Desenvolvimento do sistema Ubíquo.

3.1. Desenvolvimento do sistema pervasivo

"O conceito de computação pervasiva implica que o computador está embarcado no ambiente de forma invisível para o usuário. Nesta concepção, o computador tem a capacidade de obter informação do ambiente no qual ele está embarcado e utilizá-la para dinamicamente construir modelos computacionais, ou seja, controlar, configurar e ajustar a

aplicação para melhor atender as necessidades do dispositivo ou usuário" (ARAUJO, 2015). O desenvolvimento e o software do bloco "Sistema Pervasivo" do diagrama geral serão descritos nas subseções a seguir.

3.1.1. Lógica do sistema pervasivo

O sistema geral visa o mínimo de modificações no veículo e o sistema pervasivo não necessita de nenhuma interface entre o motorista e o sistema, com exceção do Light Emitting Diode (LED) de aviso que o sistema assumiu o controle da velocidade, usando apenas os componentes já disponíveis ao sistema e veículo, no caso o Arduino e o pedal do acelerador. Seu funcionamento é contínuo, ou seja, a cada iteração o módulo pervasivo avalia se a velocidade se manteve constante por pelo menos 4 segundos (utilizando uma faixa de tolerância de velocidade, na qual o motorista deve manter uma variação máxima de 3 km/h) e se antes de 2 segundos o motorista tira o pé do acelerador, o sistema do PID entra em ação e mantêm a velocidade constante. Isso irá ocorrer até que o motorista acione o pedal do acelerador, fazendo com que o valor da aceleração ultrapasse o valor atual do controlador. Nesse instante, o sistema retorna o controle da velocidade do veículo para o motorista. O sistema pode interromper o controle também se o pedal do freio for acionado (o que é monitorado por uma chave instalada na base do pedal) ou se o motorista acionar o botão de pânico.

3.1.2. Detalhamento do algoritmo pervasivo

```
Se usoDoPID está ativo Então

    Se velocidadeAtual está numa faixa constante Então
        IncrementaTempoDeIterações
    SeNão
        DefineFaixaDeVelocidade(velocidadeAtual)
        ZeraTempoDeIterações
    FimSe

    Se manteve faixa de velocidade constante Então
        LigaLED
        DefineTargetVelocidade(velocidadeAtual)
        DefineServoIgualPosiçãoDoCabo
        Se passou algumas iterações Então
            HabilitaUsoDoPID
        FimSe
    FimSe

SeNão

    Se aceleração ultrapassar target Então
        DesabilitaUsoDoPID
        DesligaLED
        ZeraTemposDeIterações
        DefineFaixaDeVelocidade(velocidadeAtual)
    FimSe

FimSe
```

Figura 5: Pseudocódigo do sistema pervasivo

A Figura 5 descreve em pseudocódigo a rotina da Inteligência Artificial (IA). Inicialmente, todas as *flags* de sinalização estão definidas como falsas, todos os contadores estão zerados e o LED desligado. A função de IA é chamada a cada iteração.

A primeira verificação refere-se à *flag* que indica se o PID está agindo no sistema. Caso ele não esteja agindo, é verificado se o motorista pisou no pedal a fim de obter o controle novamente, desabilitando todas as *flags* de controle e zerando os contadores. Caso ele esteja agindo, faz-se a segunda verificação.

A segunda verificação refere-se a faixa de velocidade. Caso a velocidade estiver dentro da faixa estipulada, incrementa-se o contador de iterações, caso contrário, zera-se o contador e redefine-se a faixa de velocidade.

A terceira verificação refere-se ao número de iterações necessárias para que o sistema notifique o motorista que deseja entrar em ação. Caso o contador de iterações chegue a esse valor, o sistema liga o LED, define o alvo de velocidade que deve ser

mantido e define a posição do servo como a atual posição do cabo, para evitar uma mudança brusca de velocidade.

A quarta verificação refere-se ao tempo em que o PID deve esperar para ser de fato acionado, pois devido ao acúmulo da soma do fator integrativo, os primeiros valores de ajuste são muito altos e interferem de forma negativa no uso deste.

3.2. Desenvolvimento do sistema do controlador de velocidade

O bloco “Sistema do Piloto Automático” do diagrama geral pode ser visto na Figura 6. Seu desenvolvimento e software foram concebidos no TCC1 (BILEKI, 2015) e a subseção 3.3.2 resume seu conteúdo, as subseções seguintes apresentam as modificações realizadas no decorrer deste trabalho.

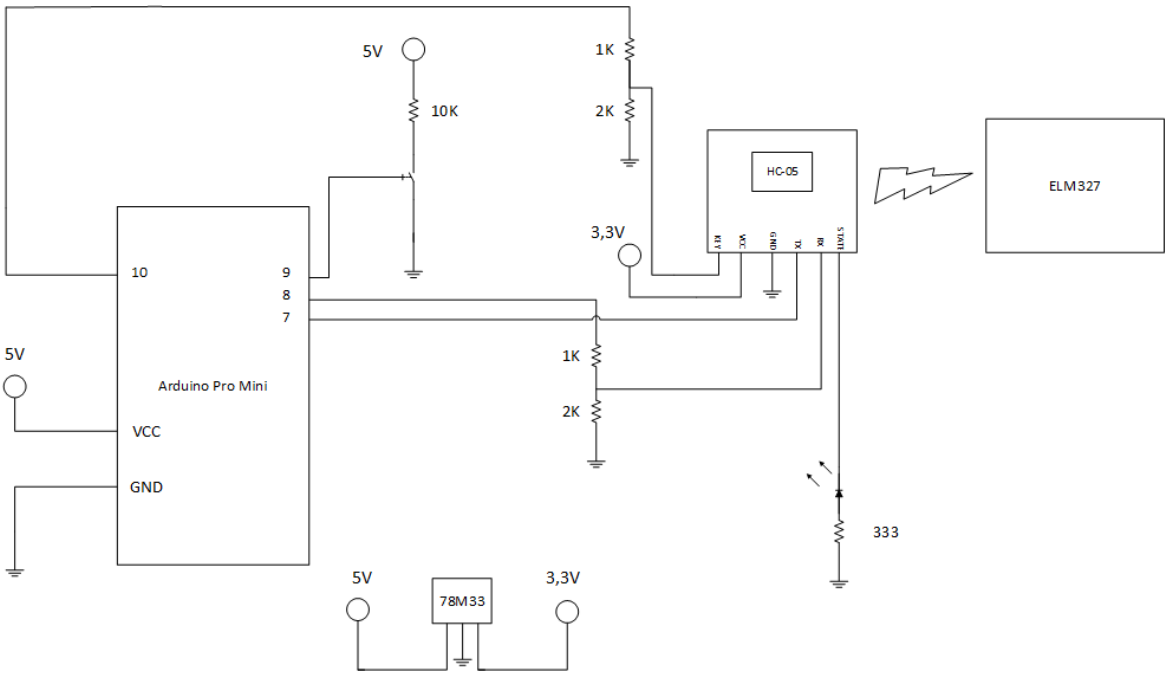

Figura 6: Circuito do sistema de piloto automático

3.2.1. Desenvolvimento do TCC1

O TCC1 realizou uma pesquisa de viabilidade com o propósito de comunicar o sistema à ECU sem modificações estruturais no veículo, o que levou ao uso do diagnóstico

OBD-II. Para os dois trabalhos o módulo ELM327 foi o escolhido por ser a opção mais barata entre os módulos de comunicação OBD-II. Apesar de sua comunicação ser problemática, devido aos problemas de comunicação Bluetooth, o software atual foi remodelado para resolver o problema de pareamento e leitura dos dados do veículo, utilizando funções de verificação de erro que aumentam o *delay* e definem o protocolo correto para o veículo.

3.2.2. Modificações para o TCC2

Do sistema proposto no TCC1, algumas modificações foram necessárias, a fim de resolver problemas encontrados no TCC1 e melhorar o sistema. As subseções a seguir tratam especificamente do que foi desenvolvido neste trabalho.

3.2.2.1 Materiais Utilizados

Como tratado nas limitações do TCC1, os materiais utilizados interferem no funcionamento do sistema, e os materiais propostos como melhoria foram adquiridos, entre eles o servo-motor e o cabo OBD-II/Serial. Entretanto, o cabo OBD-II/Serial foi adquirido apenas por ser de menor custo (5 dólares) em relação aos disponíveis no Brasil (150 reais), porém, foi constatado que não possuía um módulo de comunicação CAN-BUS, que é necessário para comunicar com o barramento de diagnóstico do veículo e por isso não foi utilizado.

3.2.2.1.1 Arduino

Para este projeto, foi mantido o Arduino do TCC1, um Arduino Pro Mini Atmega 328 5V/16MHz, de custo em torno de $4 (ARDUINO, 2015), adquirido pela internet ou em lojas de eletrônica do Brasil.

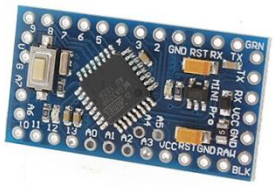

Figura 7: Arduino Pro Mini

3.2.2.1.2 BTH-07

Para este projeto foi mantido o módulo Bluetooth do TCC1, um BTH-07, de custo em torno de $10 (BTH-07, 2015), adquirido pela internet, mas de difícil acesso em lojas de eletrônica do Brasil, por ser um componente antigo.

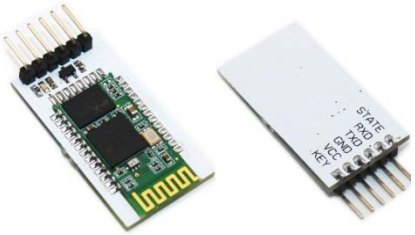

Figura 8: Módulo Bluetooth BTH-07

3.2.2.1.3 ELM327

Para este projeto foi mantido o módulo de diagnóstico Bluetooth OBD-II do TCC1, um ELM327, de custo em torno de R$40 (ELM327, 2014), adquirido pela internet ou em lojas de eletrônica do Brasil.

Figura 9: Módulo de Diagnóstico Bluetooth ELM327

3.2.2.1.4 Servo-motor

Para este projeto, ao invés do Motor DC 12V utilizado no TCC1, foi utilizado um servo-motor MG636, de custo em torno de R$30 (HOBBYKING, 2015), adquirido pela internet ou em lojas de eletrônica do Brasil.

Figura 10: Servo-motor Toward Pro MG996 15 kg/cm

3.2.2.1.5 Regulador de tensão

Para este projeto foi mantido o regulador de tensão do TCC1, um 78M33, de custo em torno de R$3 (78M33, 2003), adquirido pela internet ou em lojas de eletrônica do Brasil.

Figura 11: Regulador de tensão 78M33 de 5V para 3.3V

3.2.2.2 Software

Todas as etapas do desenvolvimento dos softwares para testes e dados coletados estão disponíveis no *Github* (https://github.com/simoesusp/CarPuter) e serão citadas as referências para cada etapa.

Para o desenvolvimento do software do TCC2, a base utilizada foi o software final desenvolvido no TCC1, que utilizava um potenciômetro para simular a leitura da posição do motor DC, um PID simples e uma entrada via interface serial para que se pudesse definir o *target* de velocidade para os testes.

Com os primeiros testes do PID do TCC1 ajustados para o servo, foi notado um comportamento inesperado na curva do Pulse Width Modulation (PWM) passado para o servo, pois com dois valores diferentes a mesma velocidade era mantida, como indicado no Gráfico 1, entre os intervalos 200-700 o servo mantém a mesma velocidade que no intervalo 1200-1500. Entretanto este comportamento se mantinha, independe dos ajustes das constantes do PID, o que indicava algum problema com o a função do software da biblioteca de controle do servo disponível para Arduino.

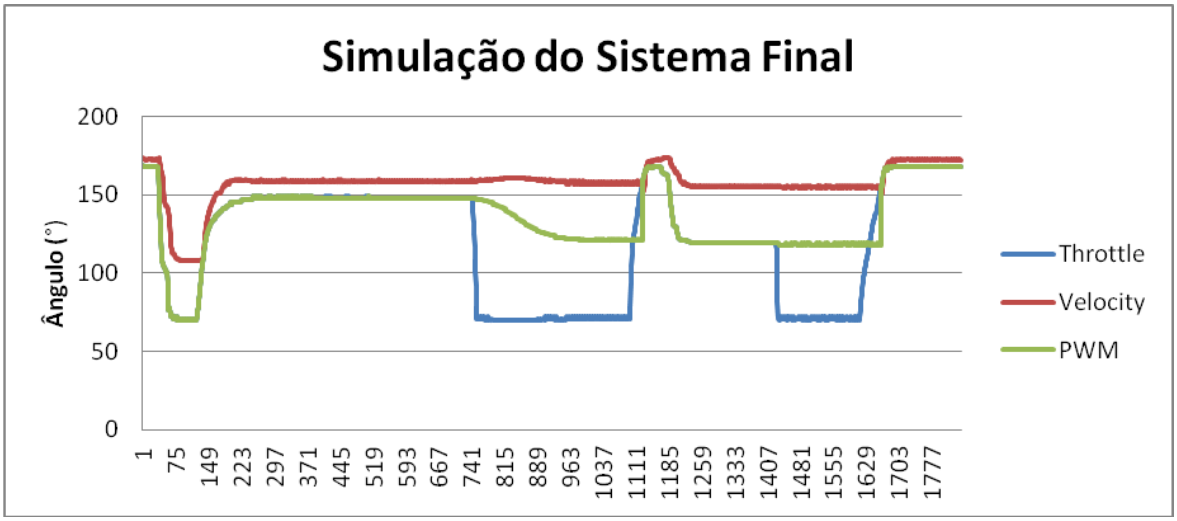

Gráfico 1: Simulação do Sistema Final

Para analisar o comportamento inesperado, um experimento simples com o servo foi executado: um programa responsável por rotacionar o servo de 0-180 continuamente com duas bibliotecas diferentes, a “Servo.h” e a “PWM.h”. Como mostra o Gráfico 2, os valores do servo na subida são diferentes na descida para as duas bibliotecas, entretanto, a biblioteca “PWM.h” tem um controle maior sobre os *timers* internos do Arduino, e, portanto, surtiu melhores resultados.

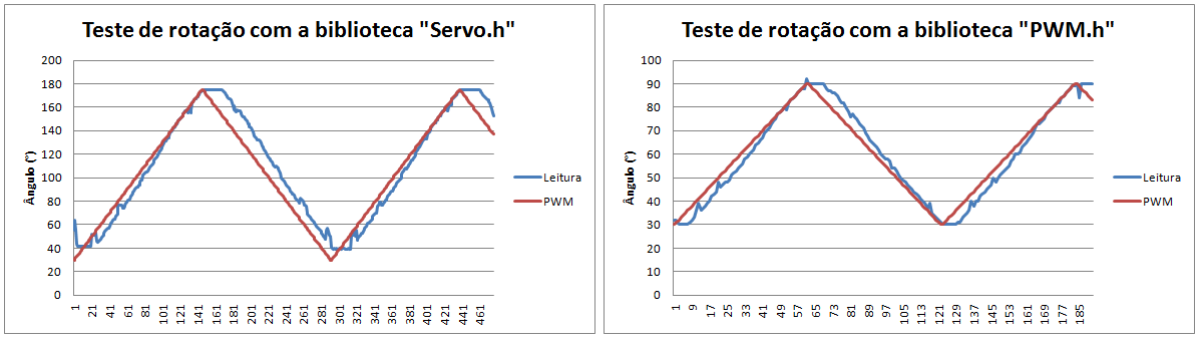

Gráfico 2: Comparação entre “Servo.h” e “PWM.h”

Portanto com o uso da biblioteca “PWM.h” os testes no sistema corresponderam ao objetivo, mantendo uma mesma velocidade com um mesmo valor do PWM, como mostrado no Gráfico 3.

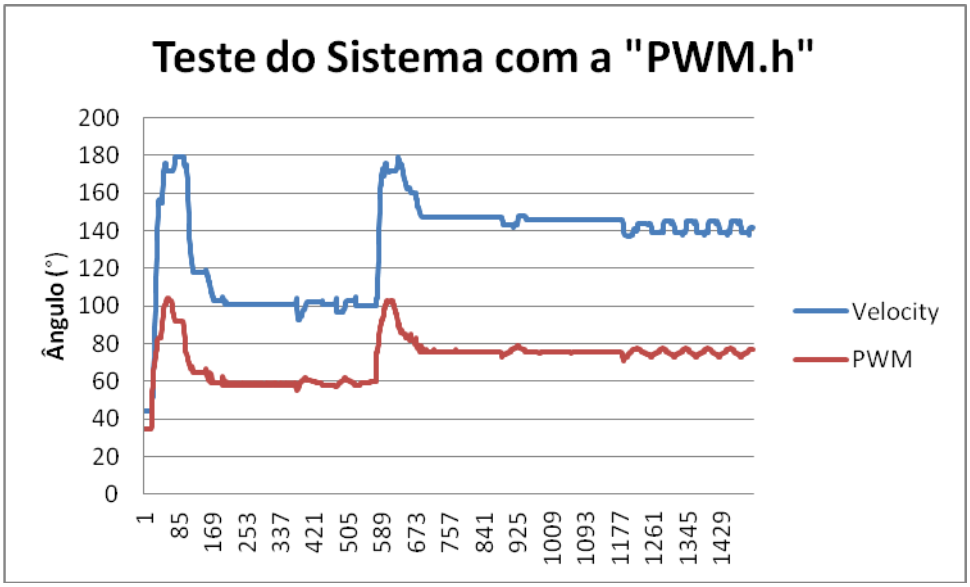

Gráfico 3: Teste do sistema com a biblioteca "PWM.h"

3.2.2.2.1 Modificações no PID

No desenvolvimento do PID, apenas o ajuste das constantes usadas no algoritmo (Kp, Ki e Kd) não foram suficientes para que o sistema conseguisse controlar a velocidade adequadamente, pois na simulação com potenciômetro o servo-motor não conseguia responder em menos de 300ms sem erros e ia além do *target* estipulado, como mostrado no Gráfico 4. Por conta desse problema, foi implementada uma nova componente do PID, a função Freio.

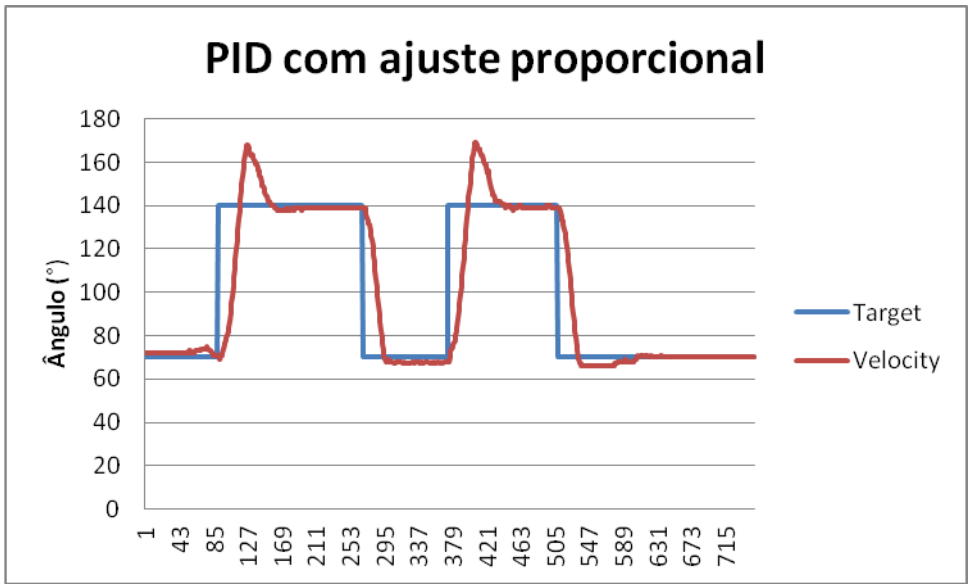

Gráfico 4: Curva de velocidade com PID simples

A primeira etapa (testePID), foi reajustar o PID para o servo-motor, alterando suas constantes e definindo limites de rotação do servo-motor, que varia de 0 a 180 graus. Com os dados coletados da leitura do potenciômetro (A0), devido à sensibilidade do servo-motor, apenas os ajustes nas constantes do PID não surtiram um resultado favorável, pois se os parâmetros de Kp e Ki forem incrementados para permitir uma rápida subida em direção ao *target* a curva de velocidade ultrapassa bastante seu *target*, necessitando de um freio. Portanto, foi inserida uma nova componente, chamada de função Freio, que será detalhada na seção seguinte. Esta função permite a utilização de maiores valores para esses parâmetros sem que a curva de velocidade ultrapasse significativamente o seu *target*.

Com a inserção do freio, quando a velocidade se aproxima do *target* o freio puxa com mais força a curva para respeitar o alvo e para de atuar assim que o *target* é alcançado, como mostrado no Gráfico 5. Com o novo PID (testePIDFreio), a curva da velocidade se manteve mais próxima da curva do *target*.

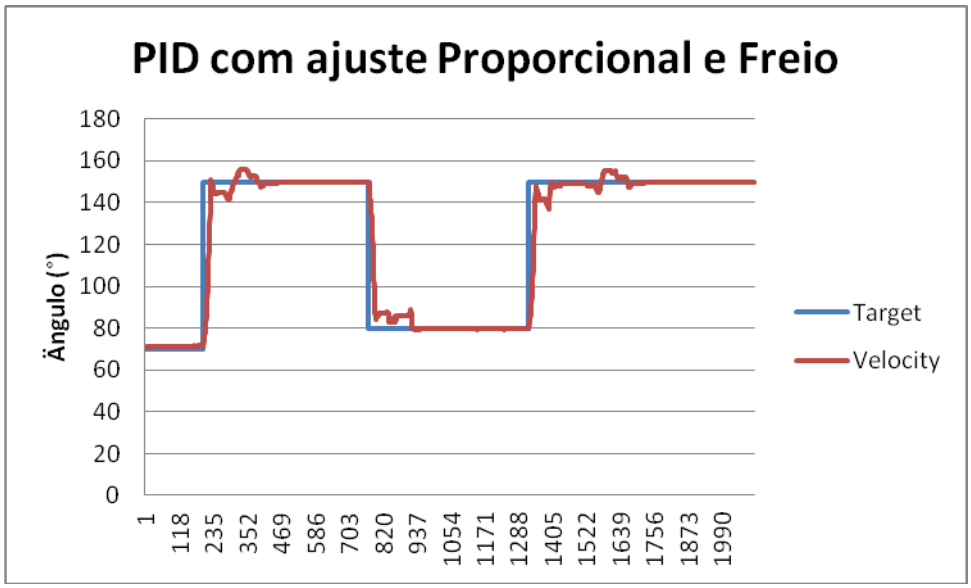

Gráfico 5: Curva de velocidade com controle do PID novo

Analisando os Gráficos 4 e 5, pode-se notar que o uso do freio no PID surtiu numa melhora no objetivo de chegar o mais rápido possível no alvo e manter a curva de velocidade igual ao alvo definido.

O freio é baseado na função matemática da Equação 1 que é proporcional a inclinação da curva e inversamente proporcional ao quadrado do erro.

Equação 1: Função Freio

$$F = \frac{\Delta}{erro^2}$$

Para definir a função, é necessário entender o comportamento da curva de velocidade controlada pelo PID. Quando um *target* é definido, o erro entre a velocidade atual e o *target* é máxima. Com a contínua correção do PID, a curva vai diminuindo o erro conforme chega perto do *target*, até o erro ser mínimo.

Para que a função atue contra a variação provocada pelos fatores proporcional e integrativo, o sinal do delta é alterado para o inverso e o erro é quadrático.

Quanto maior a inclinação da curva, mais ela deve ser freada, ou seja, se a curva está subindo muito rápido, ela precisa ser freada com mais força, para que não ultrapasse muito o *target*, por isso a função é proporcional ao delta e inversamente proporcional ao erro, pois assim os dois fatores contribuem com o movimento no início, mas atrapalham quando chega mais perto do *target*, atuando assim, como um freio. A função é desligada assim que o valor medido ultrapassa ou se iguala ao alvo.

3.2.2.3 Teste da IA

A partir da correção da biblioteca de PWM e com a inserção do freio, a função de IA pôde ser implementada (testeIA), e novos testes foram realizados, necessitando apenas de mais um potenciômetro (A1) para simular a posição do pedal do acelerador, excluindo assim a entrada via serial para definir o *target* da velocidade. Na simulação, a leitura de A1 é uma entrada do sistema, enquanto a leitura de A0 é uma entrada de controle do sistema (a velocidade atual do veículo, ou seja, o alvo escolhido para ser mantido). As Figuras 12, 13 e 14 mostram o sistema construído para simular a atuação do controlador no veículo.

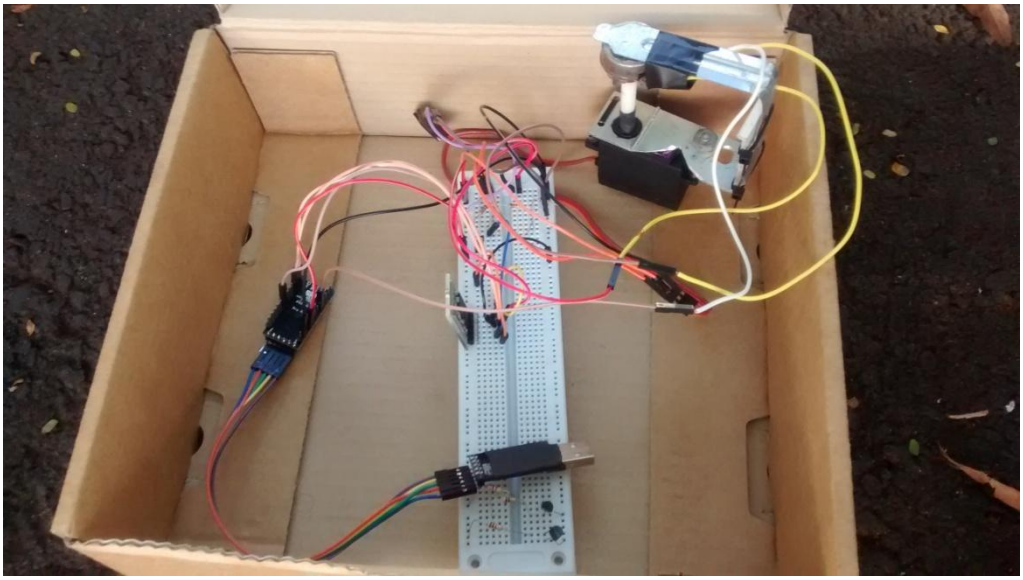

Figura 12: Circuito montado para a simulação do PID

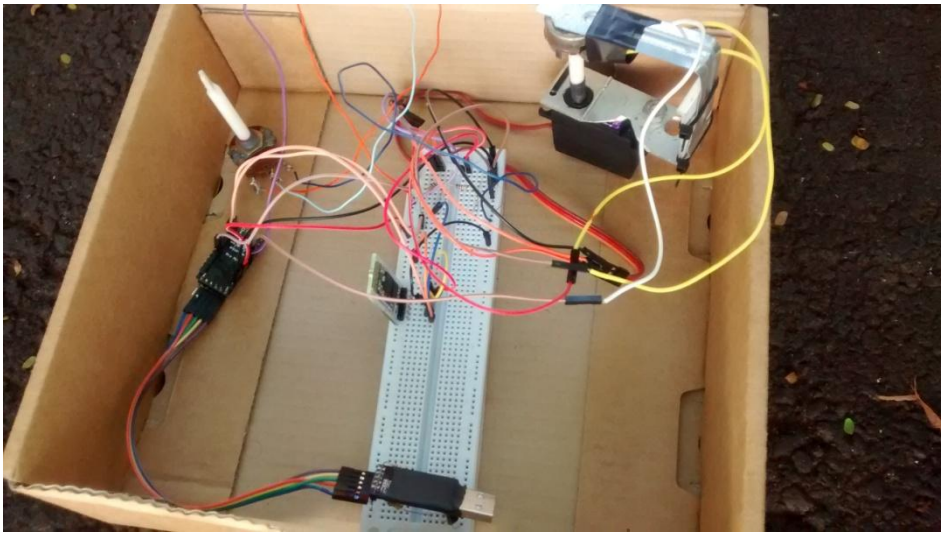

Figura 13: Circuito montado para a simulação do PID com IA

Figura 14: Circuito montado para o teste no veículo

Para este ambiente simulado, foram feitos diversos testes: o constante uso do PID para controlar o servo; o uso do PID após o motorista soltar o pedal do acelerador; e o uso do PID após a velocidade ser mantida por 4 segundos (testeIA1,2,3). Os Gráficos 6, 7 e 8 demonstram a evolução do sistema.

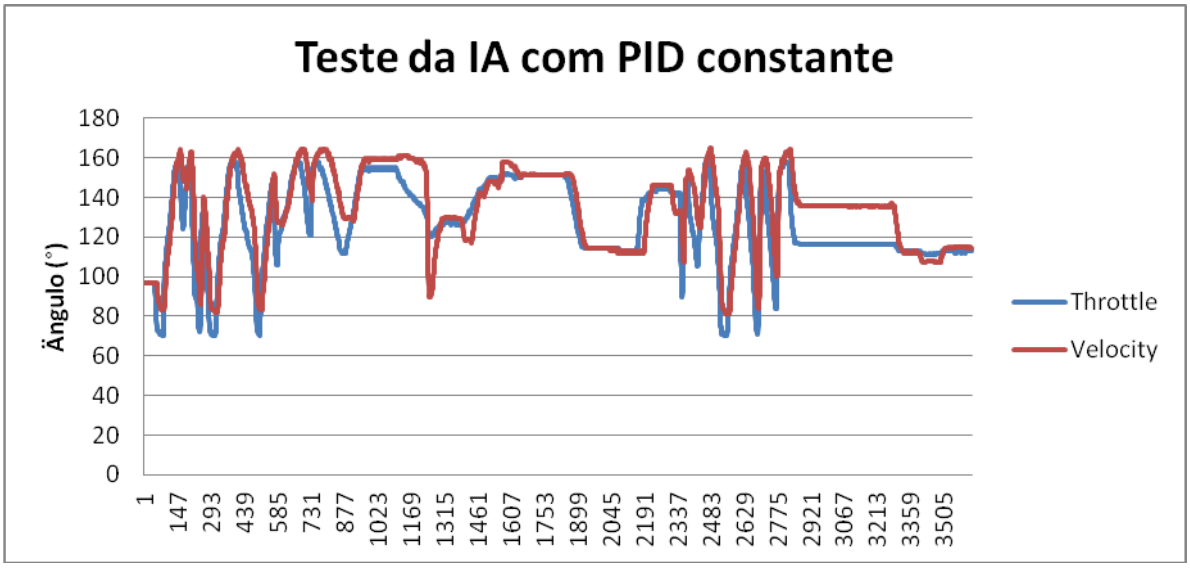

Gráfico 6: Curva de velocidade com IA e PID constante

O teste com o PID constantemente ativo do Gráfico 6 foi feito a fim de analisar seu comportamento em longo prazo devido aos acúmulos de erro. Não foi feito com a intenção de ser utilizado, apenas para fins de teste.

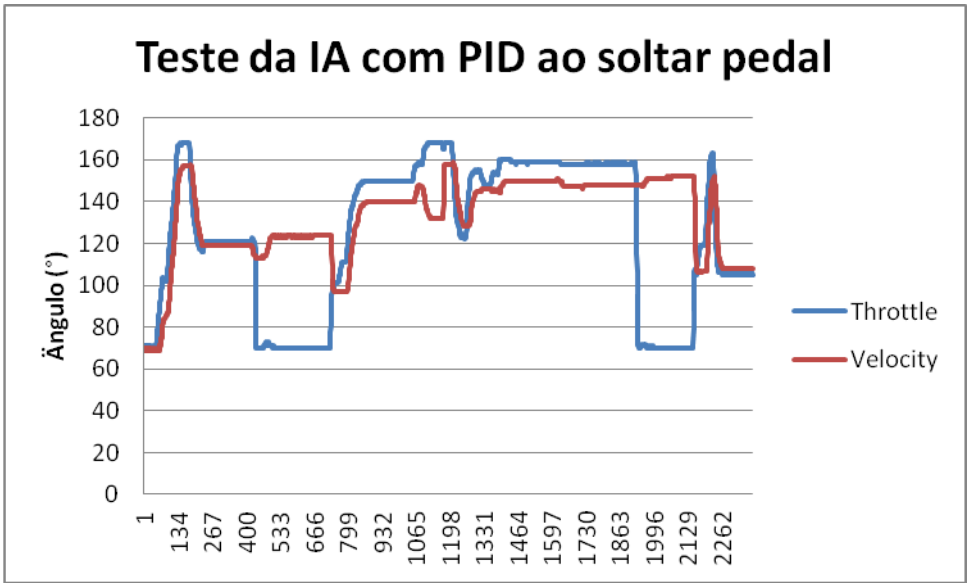

Gráfico 7: Curva de velocidade do PID atuando após o motorista soltar o pedal

O teste com o PID sendo ativado apenas quando o motorista soltasse o pedal do acelerador foi a primeira ideia de sistema, pois como pode ser visto no Gráfico 7, entre as iterações 400-700 e 1800-2000, assim que a curva do *throttle* despenca, simulando o motorista soltando o pedal, a curva da velocidade se mantém com o controle do PID. Caso o motorista não solte o pedal, o controle continua manual, como visto nos pontos 800 e 2100.

Um problema desse modelo para simular o ambiente real era que o servo precisava seguir constantemente o controle manual, pois a leitura do potenciômetro A0 depende da rotação do servo, enquanto que em um ambiente real, o servo deveria estar em repouso para não atrapalhar o controle manual. Isso acontece porque no caso real, quando o servo atua no acelerador, a leitura do valor da aceleração depende do valor que o servo estiver mantendo e não pode ser medido se o motorista soltar o acelerador. Por este motivo foi definido outro modelo, o de que o controle do PID fosse ativado assim que a velocidade fosse mantida por 4 segundos, só deixando de atuar quando o motorista ultrapassasse a velocidade mantida.

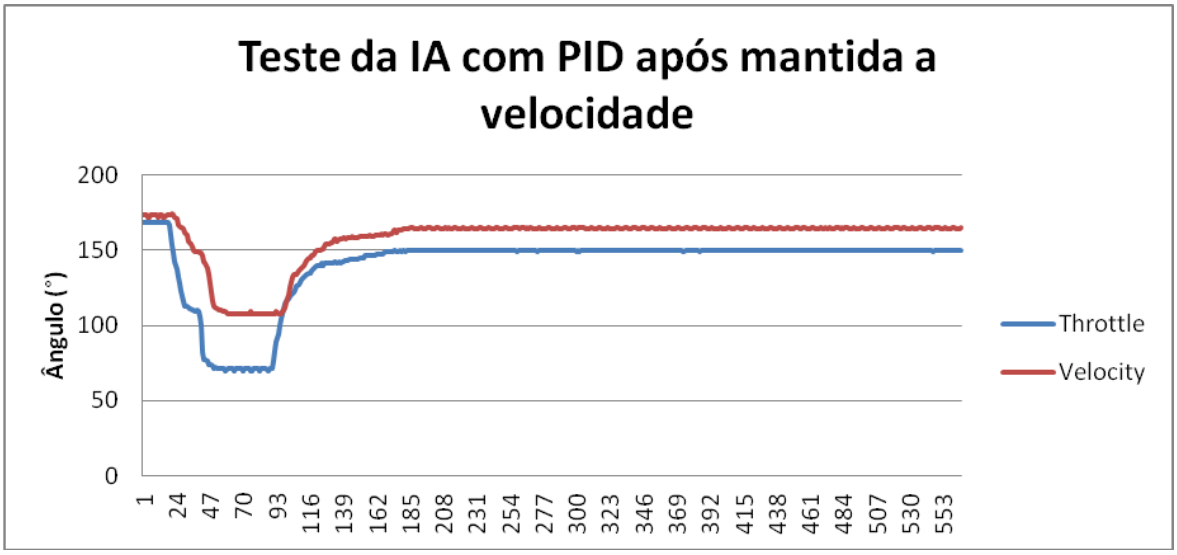

Gráfico 8: Curva de velocidade do PID atuando após a velocidade ser mantida

Como mostrado no Gráfico 8 a partir do ponto 185 a velocidade foi mantida por 4 segundos e o PID assumiu o controle.

Com os resultados obtidos no ambiente simulado, os testes no veículo foram possíveis de se realizar com mais segurança.

3.2.2.3 Teste no veículo

Para iniciar os testes no veículo, as leituras dos potenciômetros A0 (pedal do acelerador) e A1 (velocidade) foram substituídas respectivamente por leitura do *throttle* e RPM. Foi utilizada a leitura de RPM inicialmente por ser possível fazê-la com o veículo parado e por ela corresponder proporcionalmente à velocidade real uma vez que o veículo utilizado possui câmbio manual e que os testes foram realizados em quita marcha.

Devido aos valores de RPM estarem numa faixa diferente da faixa de valores do potenciômetro, foi necessário mapear os valores de 0-8000 de RPM para 0-180 dos ângulos do potenciômetro, pois os valores do PID ficam desproporcionais aos do *throttle*.

Logo no início dos testes houve problemas de perda de conexão com o Bluetooth. Após diversas observações em testes, foi constatado que o eletromagnetismo gerado pelo servo cortava o sinal do Bluetooth quando estes estavam próximos, o que foi solucionado

com o posicionamento do servo dentro do capô do veículo, conectando-o através de um longo cabo até o interior do veículo, onde fica o Arduino e o Bluetooth.

Os testes para ajustar o PID com o veículo parado não foram possíveis, pois estando desengatado, uma pequena alteração no servo causa grande impacto na RPM, ultrapassando a zona de segurança de 6000 RPM. Como o servo necessita de pelo menos 5 graus de correção para rotacionar, se o PID envia 1 grau de correção, o que seria suficiente para fazer a RPM chegar ao *target*, não surte efeito nenhum na RPM, então o PID aumenta a correção até chegar aos 5 graus e quando envia esta correção ao servo, a RPM sobe muito, tendo que enviar uma correção muito maior para o servo voltar ao *target*. Por isso foi necessário, a fim de ajustar o PID, fazer o teste do software com o veículo em movimento, o que não era o objetivo, pois por segurança, o sistema deveria estar totalmente ajustado e testado antes de ser testado em movimento.

Tomando as devidas precauções de fazer os testes em um ambiente pouco movimentado e com acostamento, foi testado o PID com o carro em movimento engatado na segunda marcha. Porém o problema do servo causar grande impacto na RPM se mantinha com a velocidade baixa. Necessitando assim fazer o teste com uma velocidade maior. Por precaução, foi implementada uma função de pânico que libera totalmente o controle do servo a fim de evitar acidentes e uma heurística limitando os limites de atuação do servo para que a sua oscilação diminua quanto mais perto ele estiver do *target*. Com as devidas precauções tomadas, pôde-se testar o sistema final na estrada, com apenas o fator proporcional do PID ligado e mantendo o veículo em quinta marcha, pois o servo necessita de um torque maior, diminuindo assim sua influência brusca na RPM.

3.4. Resultados Obtidos

Com a integração dos sistemas de controle de velocidade e o módulo pervasivo, para que fosse possível sua execução em um veículo real, foi necessário um ambiente simulado a fim de evitar acidentes. Este ambiente simulado gerou um refinamento no controlador PID, devido à sensibilidade das leituras dos potenciômetros utilizados. Entretanto, para o teste no veículo, este refinamento não foi necessário, pois os tempos de

resposta do sistema de aceleração do veículo em relação à tração do servo são bem lentos, portanto o uso de um PID apenas com fator proporcional e uma heurística de controle foram suficientes para calibrar o sistema. Com isso, o ciclo geral foi fechado e o teste do sistema completo no veículo foi realizado.

Os testes foram muito bem sucedidos, pois o sistema conseguiu manter a velocidade constante durante todo o trajeto, variando apenas 8 km/h nas subidas e descidas da estrada. O Gráfico 9 mostra os dados do trajeto todo de teste do software final no ambiente real. Os vídeos do *Github* do projeto (pasta Videos) mostram a execução de alguns testes e o vídeo "Teste Final" mostra a execução do teste em que os dados do Gráfico 9 foram obtidos.

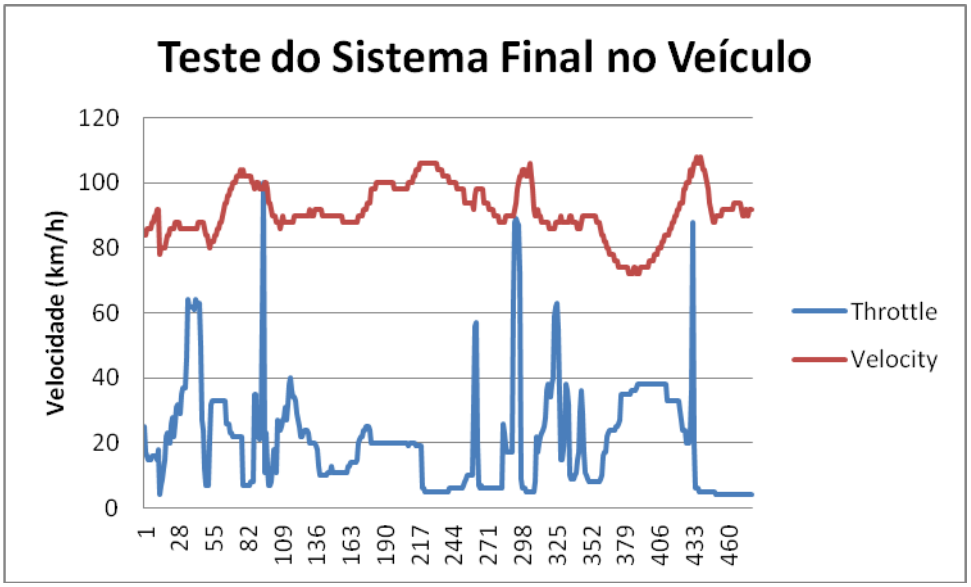

Gráfico 9: Curva de velocidade do veículo com o software final

3.5. Dificuldades, Limitações e Trabalhos Futuros

Das dificuldades do desenvolvimento do trabalho, pode-se considerar o difícil acesso no Brasil de componentes eletrônicos, o que fez com que parte do projeto tivesse que ser construído e testado de forma improvisada até a chegada dos componentes necessários. Além da falta de componentes, a documentação dos mesmos é precária e até mesmo incorreta, como no caso do manual dos módulos Bluetooth que continham

comandos que não existiam ou com muito pouca documentação. Isso acontece porque esses dispositivos são implementados na China e a documentação disponível muitas vezes é uma tradução precária do Chinês para o Inglês.

Das limitações para o desenvolvimento, pode-se considerar a falta de recursos para a aquisição de um componente, como o cabo serial/OBD-II que facilitaria o desenvolvimento do projeto. Sendo assim, o seu alto custo (R$ 150,00) limitou o desenvolvimento do trabalho.

Como trabalho futuro, pode-se aprimorar o sistema com o uso de um cabo de interface serial/OBD-II (padrão RS232 para CAN), assim as vantagens dessa implementação seriam: a diminuição do ruído causado tanto na leitura dos dados do veículo, como no circuito eletrônico; e principalmente a diminuição do tempo entre as iterações de 300 ms para 15 ms, fazendo com que o PID atue bem mais rápido. Isso deve diminuir muito a variação da velocidade a ser mantida. 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 muitas, 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

O ciclo do sistema de controle foi fechado com a integração dos sistemas de controle de velocidade e pervasivo, o que possibilitou a realização dos testes no veículo. Mesmo com algumas limitações e problemas no desenvolvimento, o sistema final proposto foi concebido e operou como esperado. O sistema foi capaz de manter o controle da velocidade do veículo dentro de uma variação de 3 km/h e trabalhou de forma pervasiva, permitindo que o usuário se concentrasse melhor na atividade de conduzir o veículo. 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. A utilização desses materiais para o desenvolvimento do sistema de piloto automático é muito mais viável economicamente do que a implantação do sistema oficial instalado pela concessionária.

Com os testes realizados de calibração das constantes do PID para um ambiente simulado, gerou-se a criação de uma nova componente, o Freio, que pode contribuir muito para os mais diversos sistemas que utilizam PID; dos testes em ambiente real, gerou-se conhecimento sobre a manipulação de componentes e sua integração com o sistema veicular.

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, bem como no padrão OBD-II para aquisição de dados da CPU do veículo.

Para o desenvolvimento completo do trabalho, não foram o bastante os conhecimentos inicialmente previstos, foi necessária muita pesquisa para resolução de problemas, dado que não há material de referência que tenha sido testado em ambiente real e de fato controlado um veículo sem o controle da ECU.

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.

Para este trabalho, diferente do TCC1, não houve colaboração externa, pois todos os problemas do TCC1 foram resolvidos devido à documentação disponível de pessoas que fizeram partes do sistema, entretanto os problemas surgidos quando implantado em ambiente real foram solucionados apenas com conhecimento teórico e experiência de trabalho do orientador.

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. O processo de desenvolvimento descrito abordou problemas com a manipulação dos componentes, tanto em ambiente simulado como no ambiente real, servindo de material de apoio e solução para quem trabalhe com tais componentes e tenha as mesmas dificuldades ou problemas. Além disso, as alterações propostas no PID com a inclusão do fator freio podem contribuir para outras aplicações deste tipo de estratégia de controle em diversos sistemas, não apenas para o controle de velocidade de veículos. 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 microcontrolador 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 Maio 2016.

(ARAUJO, 2015) ARAUJO, R. B. **Computação Ubíqua: Princípios, Tecnologias e Desafios**. São Carlos: [s.n.], 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.

(BILEKI, 2015) BILEKI, G. A. **Implementação de um sistema de controle inteligente de velocidade para veículos**. São Carlos: [s.n], 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.

(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.

(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.

(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.

(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.

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