UNIVERSIDADE FEDERAL DO PAMPA

HENRIQUE ROGGIA MACHADO

PROJETO E SIMULAÇÃO DE UM SISTEMA DE CONTROLE PARA VÁLVULAS DE MOTORES QUATRO TEMPOS

ALEGRETE

2018

HENRIQUE ROGGIA MACHADO

PROJETO E SIMULAÇÃO DE UM SISTEMA DE CONTROLE PARA VÁLVULAS DE MOTORES QUATRO TEMPOS.

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Mecânica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia Mecânica.

Orientador: Maurício Paz França

Coorientador: Gustavo Fuhr Santiago

Alegrete

2018

HENRIQUE ROGGIA MACHADO

PROJETO E SIMULAÇÃO DE UM SISTEMA DE CONTROLE PARA VÁLVULAS DE MOTORES QUATRO TEMPOS.

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Mecânica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia Mecânica.

Trabalho de Conclusão de Curso defendido e aprovado em:

Banca examinadora:

Prof. Dr. Mauricio Paz França
Orientador
UNIPAMPA

Prof. Dr. Gustavo Fuhr Santiago
UNIPAMPA

Prof. Me. Leandro Ferreira Friedrich
UNIPAMPA

“Até mais, e obrigado pelos peixes.”
Douglas Adams.

RESUMO

Motores de combustão interna são largamente usados, e para se ter competitividade no mercado é preciso investir em tecnologias que aumentem sua eficiência em relação aos concorrentes. Com o avanço tecnológico e dos métodos de fabricação, os custos de se utilizar microcontroladores no dia a dia estão cada vez menores, incluindo para aperfeiçoar produtos já estabelecidos no mercado, como os próprios motores a combustão interna. Este trabalho tem como objetivo desenvolver um sistema de controle para acionamento individual das válvulas de um motor quatro tempos, visando redefinir o ângulo de cruzamento e tempos de abertura e fechamento para mais próximo do ideal considerando diferentes regimes de funcionamento do motor. A metodologia aplicada neste trabalho foi o desenvolvimento de um firmware para Arduino, projetado como um sistema em que temos a rotação do motor como sinal de entrada, e os tempos de abertura e fechamento das válvulas como sinais de saída, e então estes sinais foram simulados em um osciloscópio e interpretados. Com estes dados em mãos, concluiu-se que o Arduino realmente é uma ótima ferramenta para prototipagem que demonstra resultados satisfatórios, mas o ideal seria a utilização de um controlador dedicado a esta função para obtermos maior precisão em um projeto real.

Palavras-Chave: Motor de quatro tempos, Comando de Válvulas Variável, Microcontrolador, Arduino

ABSTRACT

Internal combustion engines are widely used, and to be competitive in the market it is necessary to invest in technologies that increase their efficiency compared to competitors. With advances in technology and manufacturing methods, the costs of using microcontrollers on a day-to-day basis are steadily declining, including those to improve products already established in the market such as internal combustion engines This work aims to develop a control system for individual drive of the valves of a four stroke engine, aiming to redefine the overlap, the opening and closing times, to get closer to the ideal considering different engine operating regimes. The methodology applied in this work was the development of a *firmware* for Arduino, designed as a system in which we have the engine rotation as input signal, and the opening and closing times of the valves as output signals, and then these signals were simulated on an oscilloscope and interpreted. With this data in hand, it was concluded that Arduino really is a great tool for prototyping that demonstrates satisfactory results, but the ideal would be the use of a dedicated controller for this function to obtain greater precision in a real project.

Keywords: Four-stroke engine, Variable Valve Timing, Microcontroller, Arduino

LISTA DE FIGURAS

Figura 1 – OHV, ou válvulas no cabeçote. 18
Figura 2 - OHC, ou comando de válvulas no cabeçote 18
Figura 3 – DOHC, ou duplo comando de válvulas no cabeçote. 18
Figura 4 - Perfil genérico de um came 19
Figura 5 - Forma construtiva de um sensor e as formas de onda na saída depois e antes do condicionamento do sinal. 22
Figura 6 – Modelos Arduino UNO e Arduino MEGA. 25
Figura 7 - Há realimentação na malha fechada, que é subtraída da entrada gerando assim um erro. 27
Figura 8 - Não há realimentação na malha aberta 27
Figura 9 – Especificações dos modelos de cames TC Andrews da Harley Davidson. 30
Figura 10 – Perfil de acionamento dos cames em um ciclo de 720 graus. 31
Figura 11 – Leitura e aplicação dos valores nas variáveis. 32
Figura 12 – Estrutura de controle para reiniciar o ciclo de acionamento de 720 graus. 35
Figura 13 – Esquema para abertura das válvulas. 35
Figura 14 – Esquema para fechamento das válvulas. 36
Figura 15 – Modelo de microprocessador UNO R# da RoboDyn. 37
Figura 16 - Esquema elétrico utilizado na protoboard. 38
Figura 17 – Osciloscópio Tektronix DPO2024. 38
Figura 18 – Perfil de acionamento gerado em forma de onda quadrada em um ciclo de 720 graus. 39
Figura 19 – Leitura dos sinais de saída do sistema para 600 RPM, com auxílio de cursores. 40
Figura 20 - Leitura dos sinais de saída do sistema para 600 RPM, com auxílio de cursores. 41
Figura 21 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores. 41
Figura 22 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores. 42

Figura 23 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores.....42
Figura 24 – Especificações de LEDs.....43
Figura 25 - Leitura dos sinais de saída do sistema variando de 4000 RPM para 600 RPM. 43
Figura 26 - Leitura dos sinais de saída do sistema para 4000 RPM ampliado 20X. .44
Figura 27 - Exemplo da função Setup().....51
Figura 28 - Exemplo da função Loop(). 52
Figura 29 - Exemplo da estrutura de controle If. 52
Figura 30 - Componentes da estrutura de controle For.....53

LISTA DE TABELAS

Tabela 1 – Diferença de preços das placas Arduino de 2016 e 2018. 16
Tabela 2 – Conversão da dos tempos dos sinais para graus, lidos para 600 RPM. .44
Tabela 3 - Conversão da dos tempos dos sinais para graus, lidos para 4000 RPM. 45
Tabela 4 – Comparação das frações de tempo e ângulo dos valores teóricos e encontrados para 600 RPM.....46
Tabela 5 - Comparação das frações de tempo e ângulo dos valores teóricos e encontrados para 4000 RPM.....46

LISTA DE QUADROS

Quadro 1 – Operadores comparadores.....53

LISTA DE ABREVIATURAS E SIGLAS

APMI – Antes do Ponto Morto Inferior;
APMS – Antes do Ponto Morto Superior;
BDC – Botton Dead Center (Ponto Morto Inferior);
DOHC – Doble Over Head Camshaft (Duplo Comando no cabeçote);
DPMI – Depois do Ponto Morto Inferior;
DPMS – Depois do Ponto Morto Superior;
ECU – Eletronic Control Unit (Unidade de Controle Eletrônica);
IDE - Integrated Development Environment (Ambiente Integrado de Desenvolvimento);
OHC – Over Head Camshaft (Comando no cabeçote);
OHV – Over Head Valve (válvula no cabeçote);
PMI – Ponto Morto Inferior;
PMS – Ponto Morto Superior;
SOHC – Single Over Head Camshaft (Único Comando no cabeçote);
SV – Sidevalve (Válvula lateral)
TDC – Top Dead Center (Ponto Morto Superior);
VVC – Variable Valve Control (Comando de Válvulas Variável).

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Objetivos 15
1.1.1 Objetivo Geral 15
1.1.2 Objetivos específicos 15
1.1 Justificativa 15
2 REVISÃO DE LITERATURA 17
2.1 Comando de válvulas 17
2.1.2 Classificação das válvulas quanto à posição 17
2.2 Eixo de cames 19
2.2.1 Válvulas de admissão e escape 20
2.2.2 Ângulo de cruzamento 20
2.2.3 Comando de Válvulas Variável (VVT) 21
2.3 Sensoriamento do motor 22
2.3.1 Sensor de rotação e posição 22
2.3.2 Outros sensores 23
2.4 Sistema embarcado 24
2.4.1 Arduino 24
2.4.1.1 Microprocessador 25
2.4.1.2 Linguagem de programação 26
2.5 Sistema de controle em malha aberta x fechada 26
2.5.1 Malha fechada 26
2.5.2 Malha aberta 27
2.5.3 Aspectos do sistema real 27
2.5.4 Comparativo entre malha aberta e malha fechada 28
3 METODOLOGIA 29
3.1 Desenvolvimento de Firmware 29
3.1.1 Parâmetros 29
3.1.2 Modelo de Controle 33
3.1.3 Simplificações e ajustes 34
3.1.4 Abertura e Fechamento das Válvulas 34
3.2 Simulação dos sinais de saída 36

3.2.1 Equipamentos Utilizados 37
3.2.2 Microcontrolador 37
3.2.3 Esquema elétrico 37
3.2.4 Osciloscópio 38
3.2.5 Previsão da Simulação 39
4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS 40
5 CONSIDERAÇÕES FINAIS 47
5.1 Sugestões para trabalhos futuros 48
6 REFERÊNCIAS 49
APÊNDICE A - Biblioteca de Programação do Arduino 51
A.1 Estrutura 51
A.2 Variáveis 54
A.3 Funções 55

1 INTRODUÇÃO

Nos motores de combustão interna há entrada e saída de massa (mistura ar-combustível e gases provenientes da queima) na câmara de combustão através das válvulas de admissão e escape, que são controladas por um eixo de comando de válvulas. O eixo de comando de válvulas controla as válvulas através de cames, e o perfil destes cames é o que define o tempo de abertura e levantamento, e o ângulo entre o nariz dos cames da admissão e do escape define o ângulo de cruzamento das válvulas.

Atualmente a maioria dos motores se utiliza do sistema de comando de válvulas por eixo de cames, mas este apresenta sua maior eficiência em uma faixa muito estreita de rotação que vai depender de onde se deseja empregar cada motor, ou seja, se o seu melhor desempenho estará em altas ou baixas rotações. Para obter um melhor desempenho em altas e baixas rotações, diferentes formas de comando de válvulas variáveis têm sido desenvolvidas, o que permite mudar o tempo de abertura, o ângulo de cruzamento, e alguns sistemas mais modernos podem até mesmo mudar o levantamento das válvulas (Pulkrabek, 2003).

Com a evolução da tecnologia e dos métodos de fabricação, o preço dos microprocessadores e de outros componentes eletrônicos caiu de maneira que os microcontroladores pudessem se difundir, sendo assim empregados em qualquer máquina ou aparelho eletrônico sem grande acréscimo de custo no produto final.

O Arduino é um microcontroaldor que nasceu como uma ferramenta fácil e rápida para prototipagem, sendo acessível inclusive para pessoas que tiveram pouco contato com programação e eletrônica. Todas as placas Arduino e o software são completamente de código aberto, incentivando usuários a construí-las e modificá-las para suas necessidades (ARDUINO).

O objetivo deste trabalho é projetar um sistema de controle em uma placa de sistema embarcado Arduino para fazer a automação das válvulas, de modo que sejam acionadas individualmente no tempo desejado, sem mais depender da geometria do came para definir seu perfil de acionamento. O sinal enviado para o atuador responsável pelo controle de cada válvula pode ser dado em um tempo desejado, obtendo assim um perfil de acionamento mais próximo do ideal para diferentes faixas de funcionamento do motor.

A metodologia deste trabalho vai se dividir em duas etapas: a elaboração de um firmware para o Arduino, que deve ser feita visando a precisão dos sinais de saída, e eficiência do código, e a validação dos sinais de saída, que mostrará se o microcontrolador e o firmware elaborado na primeira etapa estão trabalhando conforme o esperado, gerando sinais que condizem com a realidade de funcionamento de eixos de comando voltados para diferentes faixas de funcionamento do motor.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um sistema de controle para acionamento individual das válvulas de um motor quatro tempos, visando redefinir o ângulo de cruzamento e tempos de abertura e fechamento para mais próximo do ideal considerando diferentes regimes de funcionamento do motor.

1.1.2 Objetivos específicos

- Programação de um firmware de controle em software atribuído a um sistema embarcado Arduino;
- Obter sinais de saída simulados pelo controlador condizentes com o real e comparar estes sinais de saída com o perfil de acionamento de diferentes comandos de válvulas com cames.

1.1 Justificativa

Algumas variáveis de projeto possíveis para um motor a combustão quatro tempos são o ângulo de cruzamento e os tempos de abertura e fechamento das válvulas de admissão e escape, e estes podem ser projetados para diferentes regimes de funcionamento. Entretanto, este projeto vai ter seu pico de eficiência em uma pequena faixa de rotação, visto que o comando de válvulas com came tem um perfil de acionamento fixo.

Com o tempo começou a ser aplicado o Comando de Válvulas Variável (CVV), que é capaz de variar o ângulo de cruzamento e os tempos de abertura das válvulas para que, ao variar o regime de funcionamento do motor, o pico de eficiência estaria variando junto. Algumas fabricantes de automóveis que estão na vanguarda já utilizam comando de válvulas em que a variação dos tempos é feita com controle eletrônico em modelos de luxo, mas a tecnologia utilizada não é divulgada por se tratar de segredo industrial. Com as citadas vantagens, a pesquisa voltada para desenvolver um comando eletrônico de válvulas variável passa a ser relevante, não apenas para o interesse da indústria, mas também para desenvolver e difundir a tecnologia.

Visto que nos dias de hoje os microprocessadores estão muito mais baratos, como mostra a Tabela 1 em que é comparado o preço atual com o levantado em 2016 (LOCKRIDGE; DZWONKOWSKI; NELSON; POWERS; 2016), em que o preço de uma placa Arduino MEGA 2560 chega a pouco mais de 16% mais barata em 2 anos, além de ser um modelo mais moderno, e isso faz com que se torna viável o uso de microcontroladores para fazer este controle, assim é possível obter um perfil de funcionamento das válvulas mais flexível, delimitado apenas pela programação do controlador. Este trabalho mostra também o quão importante é a junção das áreas da mecânica, eletrônica e automação, podendo ser aplicadas para resolver uma gama de problemas e aperfeiçoamentos de sistemas mecânicos de maneira eficiente e precisa.

Tabela 1 – Diferença de preços das placas Arduino de 2016 e 2018.

Modelo	**Fornecedor**	**Preço (US$)**	
		Ano 2016	**Ano 2018**
UNO	Arduino.cc	24,95	22,00
MEGA 2560	Arduino.cc	45,95	38,50

Fonte: Adaptado de Lockridge; Dzwonkowski; Nelson; Powers (2016) e adaptado de Arduino

2 REVISÃO DE LITERATURA

2.1 Comando de válvulas

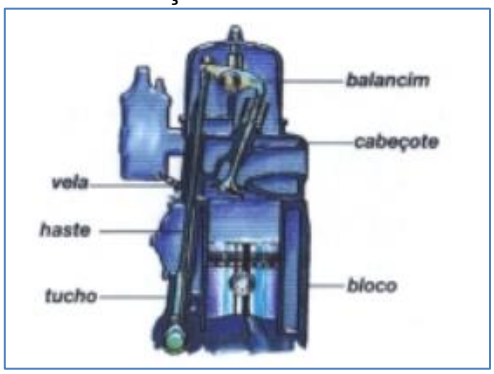
Atualmente o comando de válvulas dominante no mercado se utiliza do eixo de cames (ou eixo de comando de válvulas) que controla o tempo e altura de levantamento das válvulas mecanicamente. Este eixo é movido diretamente pelo virabrequim através de uma correia, com uma relação de dois para um (2:1), ou seja, para cada duas voltas do virabrequim o eixo de comando de válvulas gira uma volta. Esta relação é necessária tendo em vista que no motor quatro tempos há uma fase de compressão em quem as válvulas devem estar fechadas, e na volta seguinte do virabrequim seria a fase de exaustão e admissão, em que as válvulas serão acionadas.

Nos seguintes tópicos será melhor explorada a relação entre a geometria dos cames e os tempos de levantamento e abertura das válvulas, e os casos em que será interessante o atraso ou avanço da abertura das válvulas.

2.1.2 Classificação das válvulas quanto à posição

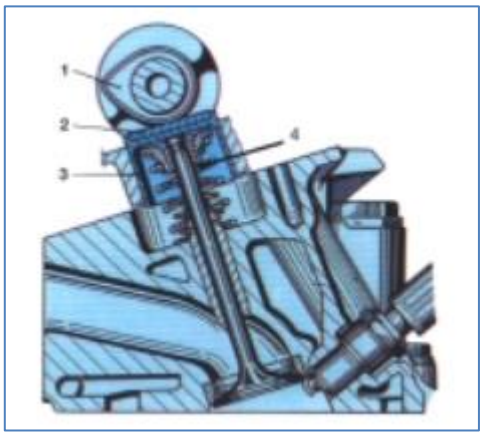
- **Válvulas no cabeçote: As válvulas estão no cabeçote e podem ser** acionadas por varetas e balancins com o comando de válvulas no bloco, esquema este chamado *Over Head Valve* (OHV), como demonstrado na Figura 1, ou diretamente pelo comando de válvulas no cabeçote, chamado Over *Head Camshaft* (OHC), demonstrada na Figura 2. Os modelos OHC podem ainda ser divididos em *Single Over Head Valve* (SOHC) onde há apenas um eixo de comando para as duas válvulas, e *Doble Over Head Valve* (DOHC), onde há um eixo de comando para a válvula de admissão e um para a válvula de escape, como visto na Figura 3. (SENAI, 2003).

Figura 1 – OHV, ou válvulas no cabeçote.



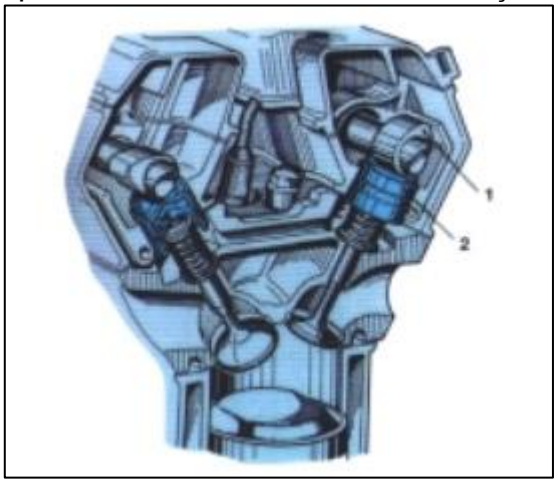
Fonte: SENAI (2003)

Figura 2 - OHC, ou comando de válvulas no cabeçote



Fonte: SENAI (2003)

Figura 3 – DOHC, ou duplo comando de válvulas no cabeçote.



Fonte: SENAI (2003)

- **Válvulas no bloco (*Flat Head* ou SV):** As válvulas estão no bloco do motor, ao lado do cilindro. Alguns modelos ainda possuem uma válvula de cada lado

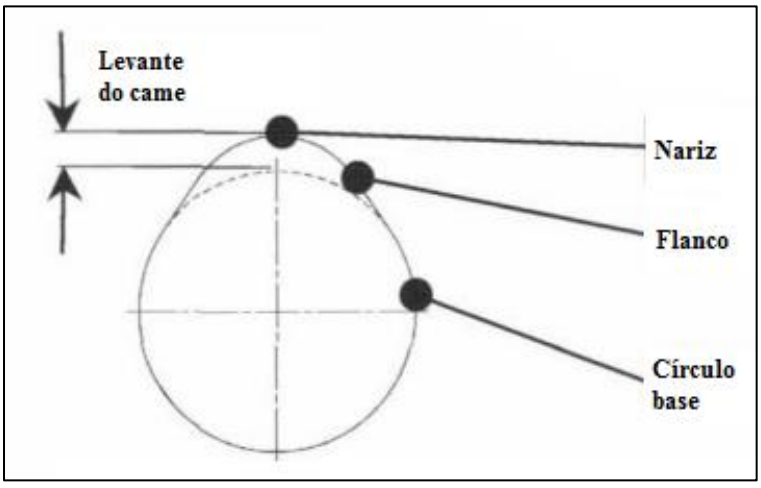
do cilindro, e são conhecidos como T Head. Esta configuração caiu em desuso, pois o comprimento da câmara de combustão é muito maior que nos OHV, perdendo muito calor e também facilitando a pré-ignição;

- **Mista:** Uma das válvulas no cabeçote (geralmente a admissão) e a outra no bloco. Esta configuração, conhecida como F Head, é muito menos comum (usada no Jeep Willys).

2.2 Eixo de cames

A função primária do eixo de cames é abrir e fechar as válvulas de admissão e escape em sincronia com a posição do pistão do motor e também com o virabrequim. As válvulas são abertas ao se transferir a força do came através do tucho, quando este entra em contato com o nariz do came, demonstrado na Figura 4, para os outros elementos de transferência de força até as válvulas, levantando-as contra a força das molas de válvula. Durante o fechamento, a força exercida pela mola empurra as válvulas contra o assento, e segue fechada enquanto o tucho estiver em contato com o círculo base do came.

Figura 4 - Perfil genérico de um came



Fonte: Adaptado de Basshuyssen (2002)

No motor quatro tempos, o eixo de cames é “tocado” pelo virabrequim e gira com metade da sua velocidade angular. O comportamento de cada válvula é determinado pela geometria dos cames, e as válvulas de admissão e escape podem

ser controladas pelo mesmo eixo de cames ou cada uma com seu eixo (BASSHUYSSEN, 2004).

2.2.1 Válvulas de admissão e escape

As válvulas são controladas pelo eixo de comando de válvulas e são responsáveis por liberar a passagem do combustível e gases para dentro ou fora da câmara de combustão, mais comumente localizadas no bloco do motor ou no cabeçote. Enquanto a válvula de admissão estiver aberta e o pistão estiver fazendo seu curso do ponto morto superior (PMS) ao ponto morto inferior (PMI), o aumento do volume faz com que a pressão caia e ajude a puxar a mistura. O fechamento da válvula de admissão seria ideal no momento em que a pressão no coletor de admissão é igual à pressão no cilindro do pistão.

Caso a pressão no coletor de admissão seja maior no momento de fechamento da válvula, ainda teria mistura entrando na câmara de combustão, e caso a pressão na câmara seja maior (após o inicio da compressão, quando o pistão passa o PMI e recomeça o curso em direção ao PMS), a mistura voltaria pela válvula de admissão.

Em altas rotações do motor o ar que passa pela válvula de admissão está em alta velocidade, fazendo com que haja uma queda de pressão, sendo assim é interessante que a válvula fique mais tempo aberta, pois a equalização da pressão ocorre mais tarde depois do PMI. Já em baixas rotações o diferencial de pressão causado pela válvula é menor, fazendo com que a equalização de pressão entre o coletor de admissão e o cilindro ocorra mais cedo após o PMI (PULKRABEK, 2004).

2.2.2 Ângulo de cruzamento

Na fase de admissão e escape, quando o pistão está no PMS há um momento em que as válvulas de admissão e escape estão abertas ao mesmo tempo. Este momento, chamado de ângulo de cruzamento, é o ângulo medido somando o ângulo de avanço da abertura da válvula de admissão com o ângulo de retardo do fechamento da válvula de escape.

Durante o ângulo de cruzamento pode acontecer de uma parte do gás proveniente da queima do combustível entre pela válvula de admissão e depois seja empurrado de volta para a câmara de combustão com a mistura ar-combustível, diminuindo assim a eficiência volumétrica. Isto ocorre principalmente em baixas rotações, pois a pressão no coletor de admissão ainda é baixa (PULKRABEK, 2004).

2.2.3 Comando de Válvulas Variável (VVT)

Os motores normalmente possuem um comando de válvulas com geometria fixa, configurado para um funcionamento em determinada rotação média, perdendo assim a eficiência para maiores ou menores rotações, sendo assim foi criado o comando de válvulas variável, que permite ajustar os tempos de abertura, levantamento e ângulo de cruzamento das válvulas, dependendo da atual rotação do motor.

Alguns motores que possuem comando variável de válvulas que controla apenas a fase dos eixos de comando em relação à árvore de manivelas, sem afetar o tempo de abertura e levantamento da válvula. Os comandos acionados eletricamente ou eletro-hidraulicamente permitem o acionamento das válvulas baseados em uma função que leva em conta a rotação do motor. Sistemas simples de controle permitem mudar entre alguns perfis dependendo da faixa de rotação, alguns mais sofisticados podem permitir um ajuste infinitamente variável, tendo um perfil para cada variação infinitesimal de rotação (BOSCH, 2005).

Em altas rotações o tempo real do ciclo é menor e é preciso mais mistura ar-combustível em menos tempo, então para otimizar o funcionamento é preciso abrir a válvula de admissão mais cedo, mantê-la aberta por mais tempo e se possível ter um maior levantamento. A válvula de escape também deve abrir mais cedo, ter maior levantamento e fechar mais tarde. Aumentar o ângulo de cruzamento é possível, pois a pressão no coletor de admissão é maior e o ciclo ocorre mais rápido.

Em baixas rotações o tempo real do ciclo é maior, e o volume de ar-combustível necessário tem mais tempo pra entrar na câmara de combustão, então as válvulas devem abrir mais tarde e fechar mais cedo. Em baixas rotações a pressão no coletor de admissão é menor, então o ângulo de cruzamento deve ser reduzido para evitar o retorno dos gases provenientes da queima para o coletor de

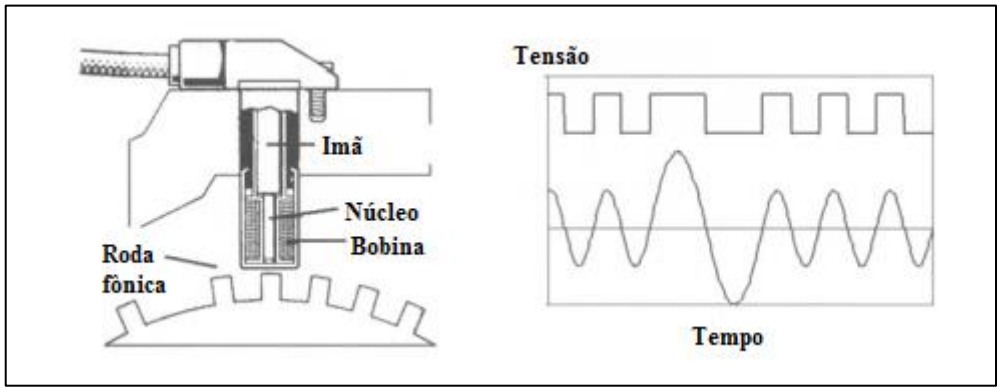
admissão. A válvula de admissão deve ter um menor levantamento para que a velocidade do fluxo continue alta o suficiente para manter uma boa mistura e dispersão (PULKRABEK, 2004).

2.3 Sensoriamento do motor

2.3.1 Sensor de rotação e posição

O sensoriamento da rotação do motor e posição do pistão é de extrema importância, principalmente na *Eletronic Control Unit* (ECU) para controlar o sistema de injeção e ignição, e para o comando de válvulas variável. O funcionamento do sensor de rotação se baseia em um sensor indutivo que “sente” a passagem de cada dente de um disco (roda fônica) acoplado ao eixo do motor. Baseado na frequência deste sinal pode se calcular a rotação do motor (BRUNETTI, 2013). Na Figura 5 é possível observar a forma construtiva do sensor e sua saída.

Figura 5 - Forma construtiva de um sensor e as formas de onda na saída depois e antes do condicionamento do sinal.



Fonte: Adaptado de Stone & Ball (2002)

Um campo magnético gerado pelo sensor vai passar pelo núcleo de ferro da bobina. A força deste campo magnético vai depender da “condutância magnética” do circuito. Quando um dente da roda fônica alinha com o núcleo da bobina (como mostrado na Figura 5), o campo magnético será mais forte, e quando tiver um espaço vazio (entre os dentes) alinhado com o núcleo da bobina, o campo magnético será fraco, e quando há um espaço maior, devido à falta de um dos

dentes, o sinal será mais fraco ainda. O sinal analógico recebido pode então ser convertido em um sinal digital, tendo assim pulsos de tensão constante, e quanto maior a frequência do sinal, maior a rotação do motor.

Na roda fônica existem 60 dentes, e dois destes são removidos para servir de referência para a posição, então temos como o passo da roda com um ângulo de 6 graus. O pulso de referência indica a posição do pistão, e os pulsos menores indicam a rotação do motor. Os pulsos podem alimentar um circuito “phase lock loop” que gera pulsos em uma frequência maior é múltipla da frequência base, gerando assim uma leitura mais fina do que a frequência gerada por cada dente (STONE; BALL, 2002).

2.3.2 Outros sensores

É possível citar outros sensores utilizados nos motores, descritos por Basshuyssen (2004), pois alguns deles podem ser úteis para o dimensionamento de um possível modelo de atuador, ou melhorias no controle das válvulas. Como estes outros sensores já são utilizados nos motores, seus sinais podem ser aproveitados.

- Sensor de Temperatura: Nos motores são empregados inúmeros modelos de sensores de temperatura, podendo medir de -40°C até 1000°C dependo de onde o sensor está sendo aplicado;
- Sensor de Detonação: Este sensor é responsável por alimentar a ECU com sinais de vibrações mecânicas, para que este interprete e descubra se está ocorrendo combustão anormal na câmara de combustão;
- Sensor de Exaustão de Gás: Controla a injeção de combustível, de modo que se consiga uma ótima taxa de conversão do conversor catalítico;
- Sensor de Massa de Ar: Este sensor diz o quanto de ar está entrando pela admissão, para determinar o estado de carga do motor;
- Sensor de Pressão do Coletor de Admissão: Os sinais destes sensores são comparados com os dados do Sensor de Massa de Ar

pela ECU para diminuir o número de parâmetros medidos, aumentando a precisão dos dados utilizados na injeção direta e no CVV;

- Há ainda outros sensores como Sensor Lambda, Sensores de Pressão e Sensor Nox, que não serão melhor descritos aqui.

2.4 Sistema embarcado

Um sistema embarcado é um dispositivo simples que não possui nada além do necessário para exercer uma tarefa específica. Possui especificações de *hardware* fixas, com um microprocessador poderoso o suficiente para exercer a tarefa proposta e este é completamente dedicado ao dispositivo que ele controla. Visto que o dispositivo é tão dedicado, é possível otimizá-lo ao máximo para sua função designada, tendo assim o máximo aproveitamento com mínimo consumo de energia, peso, tamanho e com o menor custo (LANGBRIDGE, 2014).

Langbridge (2014) levanta um questionamento para exemplificar ou delimitar um sistema embarcado, se um telefone celular o seria ou não. Alguns diriam que sim, pois o aparelho é customizado, e designado para apenas uma tarefa: realizar e receber chamadas. Outros diriam que não, pois o aparelho se tornou tão poderoso que está mais para um computador pessoal, com sistema operacional completo em que o usuário pode instalar softwares.

2.4.1 Arduino

O Arduino é uma plataforma eletrônica de código aberto baseada em um software e hardware de fácil uso. As placas do Arduino são capazes de ler entradas (luz em um sensor, o apertar de um botão, uma mensagem no *Twitter*) e transformar isto em uma saída (ativar um motor, ligar um LED, publicar algo online). O Arduino é programado ao receber um código no microcontrolador presente na placa, usando a linguagem de programação Arduino (baseado no *Wiring*, outra plataforma de prototipagem eletrônica de *hardware* livre) e o *Software* Arduino (IDE – *Integrated Development Environment*), baseado no *Processing* (outra linguagem de programação de código aberto).

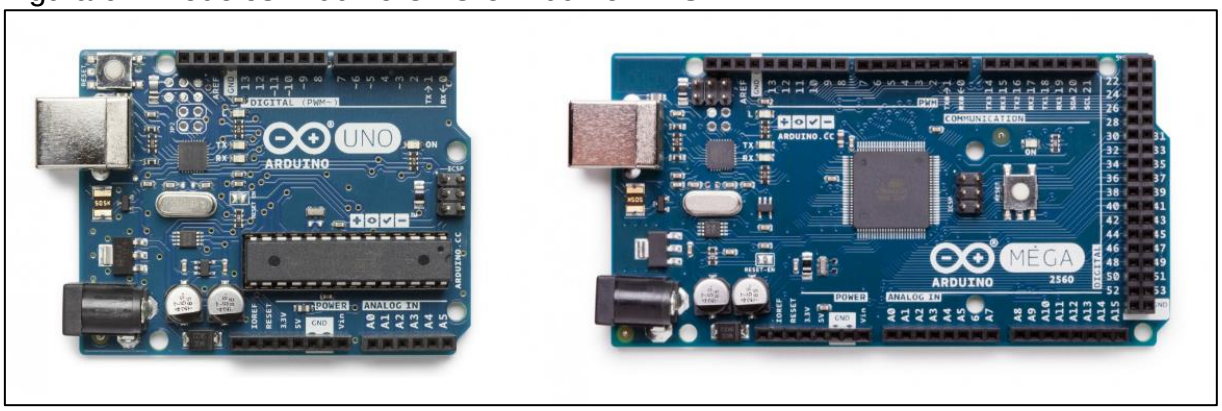
O Arduino nasceu no *Ivrea Interaction Design Institute* (instituto de Design de Interação Ivrea) como uma ferramenta fácil e rápida para prototipagem, visando alunos que não tiveram um grande contato com eletrônica e programação. Conforme a comunidade de usuários foi crescendo, novas placas foram sendo desenvolvidas para suprir diferentes novidades e desafios. Todas as placas Arduino são completamente de código aberto, incentivando usuários a construí-las e modificá-las para suas necessidades. O software também é de código aberto, e cresce através da contribuição dos usuários mundo afora (ARDUINO – INTRODUÇÃO, 2017).

Um código aberto (ou *Open Source*) é um modelo de desenvolvimento de um produto (como um software e seu código-fonte) com licenciamento livre, ou seja, qualquer um pode consultar, examinar ou modificar este produto, e a comunidade de usuários deve compartilhar informações técnicas para qualquer um, criando assim uma rede de desenvolvimento conjunta e sem fins lucrativos (visto que não é possível cobrar *royalty* ou taxas pela sua venda) (*SOFWARE* LIVRE, 2017).

2.4.1.1 Microprocessador

Os microcontroladores Arduino utilizam microprocessadores da linha ATmel AVR da ATmel Corporation (ATMEL, 2017), como o ATmega328P na placa Arduino UNO, e o ATmega2560 na placa Arduino MEGA 2560, apresentados na Figura 6, que são placas para projetos com menor e maior complexidade respectivamente.

Figura 6 – Modelos Arduino UNO e Arduino MEGA.



Fonte: ARDUINO Loja (2017).

2.4.1.2 Linguagem de programação

Os programas criados para o Arduino são feitos em linguagem C, que é uma linguagem de programação compilada, ou seja, o código fonte é executado diretamente pelo processador. No Arduino o código pode ser dividido em três partes principais: Estrutura, valores (variáveis e constantes), e funções. No Apêndice A serão apresentados alguns dos componentes que provavelmente serão utilizados no desenvolvimento do projeto proposto por este trabalho.

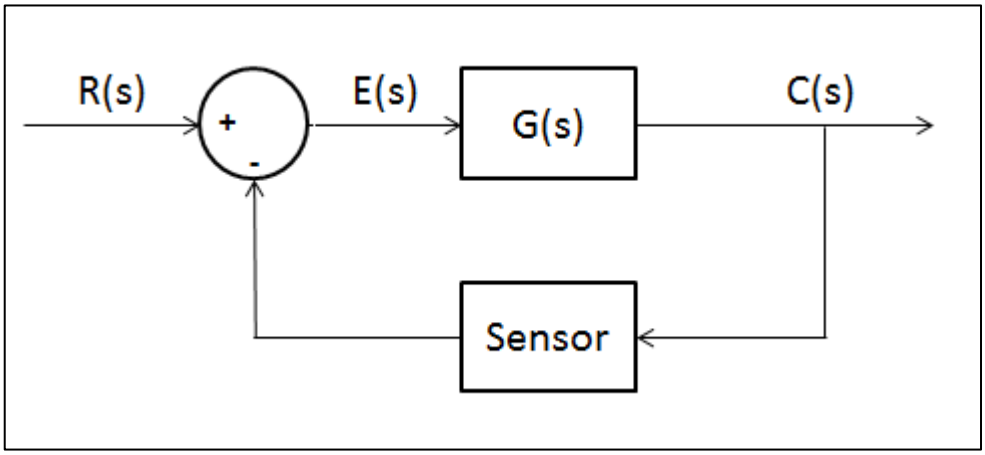
2.5 Sistema de controle em malha aberta x fechada

2.5.1 Malha fechada

O sistema de controle com malha fechada toma suas decisões baseado no sinal de erro que é a diferença entre a entrada e o sinal de realimentação, como visto na Figura 7, que pode ser a própria saída do sistema (a medição da resposta atual do estado do sistema), ou uma função desta e suas derivadas e/ou integrais, e está constantemente realimentando o controlador, minimizando assim o erro e fazendo com que a saída do sistema seja mais precisa e convirja mais rapidamente para o valor desejado.

Alguns problemas comuns do mundo real devem ser levados em conta na modelagem do sistema de controle, como perturbações, mudanças na dinâmica do processo e ruídos no sensor, e em cada projeto será definido o quando cada problema deve ser levado em consideração.

Figura 7 - Há realimentação na malha fechada, que é subtraída da entrada gerando assim um erro.



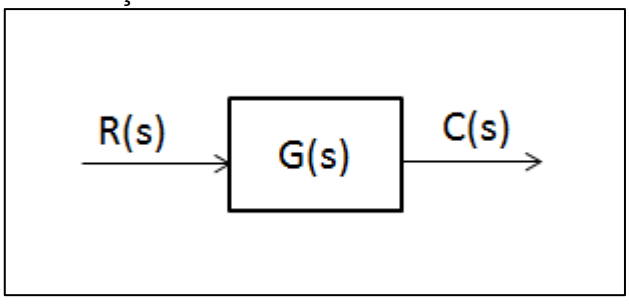
Fonte: Elaboração própria

2.5.2 Malha aberta

O sistema de controle com malha aberta é aquele em que o controle é feito sem se utilizar de nenhuma medição da resposta atual do sistema, ou seja, o sinal de entrada não é comparado com um sinal de realimentação proveniente da saída ou de sensores antes de entrar no controlador, como mostrado na Figura 8.

Sem sensoriamento e realimentação o sistema precisa estar bem calibrado para ser preciso, pois ele funciona de maneira “cega”, e para cada entrada diferente há um regime de operação. (OGATA, 2008)

Figura 8 - Não há realimentação na malha aberta



Fonte: Elaboração própria

2.5.3 Aspectos do sistema real

- **Perturbações:** São fatores que não estão sendo monitorados por um sensor ou levados em conta no sistema, mas interferem na resposta do

sistema provocando erros, como o vento ou a perda de calor para um ambiente não controlado;

- **Variações na dinâmica do processo:** Quando a dinâmica do processo muda de forma estrutural ou paramétrica, de modo que o controlador ficaria atuando sem receber a resposta de um sensor para então levar em conta estas mudanças. Mudança estrutural é algo drástico como a perda de um motor ou a asa de um avião, e mudança paramétrica é algo mais suave, como a variação da massa da aeronave ao queimar combustível ou ao abrir as superfícies de controle da asa;
- **Ruído do sensor:** O controle de malha fechada está constantemente medindo a resposta atual das variáveis controladas através de sensores, e estes podem apresentar ruídos ao emitir o sinal para o controlador. O ruído é incluído nas medições, portanto deve ser levado em conta ao se projetar o controlador.

2.5.4 Comparativo entre malha aberta e malha fechada

No controle em malha fechada, o uso de realimentação faz com que a resposta do sistema seja insensível a distúrbios externos e variações nos parâmetros do sistema, pois é possível corrigi-los enquanto o processo está em andamento, visto que o controlador os leva em conta.

3 METODOLOGIA

A metodologia deste trabalho vai se dividir em duas etapas, desenvolvimento de um firmware para o Arduino, e simulação dos sinais de saída.

3.1 Desenvolvimento de Firmware

3.1.1 Parâmetros

O controle de válvulas com eixo de cames pode ser interpretado como um sistema com entradas e saídas. A saída do eixo de cames pode ser interpretada como um sinal de aberto ou fechado, e as entradas são a rotação transmitida pelo motor e os ângulos de acionamento. Os ângulos de acionamento em um eixo de cames são fixos, mas neste trabalho serão variáveis, e dependem da rotação de entrada no sistema.

Na Revisão da Literatura foi visto que para maiores rotações é preferível que as válvulas abram mais cedo e fechem mais tarde, enquanto que em baixas rotações as válvulas devem abrir mais tarde e fechem mais cedo. Foram escolhidos ângulos de acionamento em uma tabela de especificações de eixos de cames da Harley Davidson dos modelos “TC Andrews” com ângulos de aberturas e cruzamento, isto foi feito para se ter valores reais dos ângulos com maiores e menores tempos de abertura. Foi escolhido o modelo TW64G para ter seus ângulos utilizados em altas rotações, e o modelo TW26 para baixas rotações, com seus valores demonstrados na Figura 9. Os tempos de abertura das válvulas da tabela utilizadas são as durações líquidas, e não será levada em conta a folga da válvula.

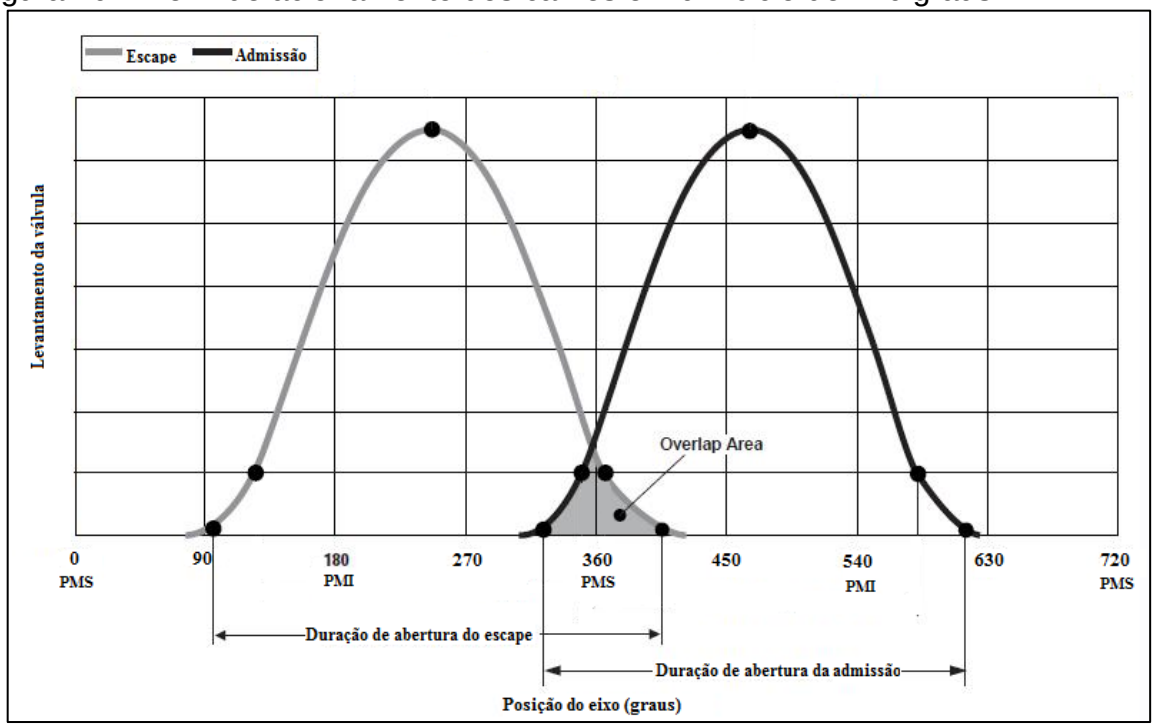
Figura 9 – Especificações dos modelos de cames TC Andrews da Harley Davidson.

Modelo	Bolt in ?	Abertura da admissão	Fechamento da admissão	Abertura do escape	Fechamento do escape	Duração da admissão	Duração do escape	Cruzamento
TW26	Y	09	37	43	07	226	230	16
TW26A	Y	11	35	41	09	226	230	20
TW26G	Y	11	35	41	09	226	230	20
TW31S	Y	10	46	52	08	236	240	18
TW37	Y	12	42	48	12	234	240	24
TW37B	Y	14	42	48	12	236	240	26
TW37G	Y	14	42	48	12	236	240	26
TW44	Y	21	41	49	17	242	246	38
TW44G	Y	21	41	49	17	242	246	38
TW50	-	20	48	54	18	248	252	38
TW50G	-	20	48	54	18	248	252	38
TW55	-	22	46	52	20	248	252	42
TW55G	-	22	46	52	20	248	252	42
TW59G	-	29	57	63	27	266	270	56
TW60	-	24	56	58	22	260	260	46
TW60G	-	24	56	58	22	260	260	46
TW64G	-	30	62	68	32	272	280	62

Fonte: Adaptado de CLASSIC CYRCLES (2018)

É preciso ainda observar como o funcionamento das válvulas se repete em relação ao eixo do motor, e assim é possível definir um período em que serão encaixados os parâmetros. O eixo de cames vai repetir seu funcionamento para cada duas voltas do eixo do motor, então dentro desses 720 graus deverão ser divididos os tempos de abertura das válvulas, gerando um gráfico similar ao da Figura 10.

Figura 10 – Perfil de acionamento dos cames em um ciclo de 720 graus.



Fonte: Adaptado de Grumpy's Performance

Essa divisão dos parâmetros pode ser feitas de duas maneiras:

a) Ângulo em que abre e ângulo em que fecha:

- TW26:
  - Abertura do escape (43 APMI): 180 – 43 = 137 graus;
  - Fechamento do escape (7 DPMS): 360 + 7 = 367 graus;
  - Abertura da admissão (9 APMS): 360 – 9 = 351 graus;
  - Fechamento da admissão (37 DPMI): 540 + 37 = 577 graus.
- TW64G:
  - Abertura do escape (68 APMI): 180 – 68 = 112 graus;
  - Fechamento do escape (32 DPMS): 360 + 32 = 392 graus;
  - Abertura da admissão (30 APMS): 360 – 30 = 330 graus;
  - Fechamento da admissão (62 DPMS): 540 + 62 = 602 graus.Ângulo em que abre e duração da abertura

A duração pode ser encontrada na tabela da Figura 9.

- TW26:

  Abertura do escape (43 APMI): 180 – 43 = 137 graus;

  - Duração do escape: 230 graus;
  - Abertura da admissão (9 APMS): 360 – 9 = 351 graus;
  - Duração da admissão: 226 graus.
- TW64G:
  - Abertura do escape (68 APMI): 180 – 68 = 112 graus;
  - Duração do escape: 280 graus;
  - Abertura da admissão (30 APMS): 360 – 30 = 330 graus;
  - Duração da admissão: 272 graus.

Para o desenvolvimento do firmware, foi decido por se utilizar os ângulos de abertura e de fechamento encontrados no item A (Ângulo em que abre e ângulo em que fecha), sendo assim, teremos a variação dos tempos conforme a Tabela 2.

Tabela 2 - Variação dos tempos para abertura e fechamento das válvulas para 600 RPM e 4000 RPM

Modelo	Rotação	Abertura do escape [°]	Fechamento do escape [°]	Abertura da admissão [°]	Fechamento da admissão [°]
TW26	600 RPM	137	367	351	577
TW64G	4000 RPM	112	392	330	602

Fonte: Elaboração Própria

A leitura e distribuição da rotação e a aplicação desses tempos nas variáveis são feitas como mostrado na Figura 11, utilizando a função “map”.

Figura 11 – Leitura e aplicação dos valores nas variáveis.

```
int Read = analogRead(A0);
RPM = map(Read, 0, 1023, 600, 4000);


timeOut1 = map(Read, 0, 1023, 137, 112);
timeIn1 = map(Read, 0, 1023, 351, 330);
timeOut2 = map(Read, 0, 1023, 367, 392);
timeIn2 = map(Read, 0, 1023, 577, 602);
```

Fonte: Elaboração Própria

Em que a variável “Read” vai ser a leitura do potenciômetro, variando de 0 a 1023, a variável “RPM” é a conversão do “Read” em um valor de 600 a 4000, “timeOut1” é o tempo de abertura do escape, “timeIn1” é o tempo de abertura da admissão, “timeOut2” é o tempo de fechamento do escape, e “timeIn2” é o tempo de fechamento da admissão.

3.1.2 Modelo de Controle

Para a programação no Arduino, era preciso uma equação que transformasse as entradas (ângulos e rotação) em um tempo em milissegundos, que seria o tempo de duração dos sinais de saída para aberto e fechado. Para se obter esta equação, foi feita uma análise dimensional a partir das unidades dos fatores que são mostrados a seguir, considerando o ângulo em graus e o tempo em milissegundos.

$$RPM = \frac{\text{rotações}}{\text{minuto}} = \frac{360\ \text{graus}}{60000\ \text{milisegundos}} \quad ...(1)$$

Com essas unidades podemos fazer um balanço da seguinte forma:

$$Tempo = \frac{\hat{A}ngulo}{RPM} = \frac{\cancel{graus}}{\left(\frac{360\ \cancel{graus}}{60000\ milisegundos}\right)} \quad ...(2)$$

$$Tempo = \frac{60000\ milisegundos}{360} = 166{,}666\ milisegundos \quad ...(3)$$

Sendo assim, temos a seguinte equação para ser usada no Firmware:

$$Tempo = 166{,}666 \times \frac{\hat{A}ngulo}{RPM} \quad ...(4)$$

A Eq. (4) pode ainda ser rearranjada de modo que a dizima periódica seja suprimida, no seguinte formato.

$$Tempo = \frac{\hat{A}ngulo}{0{,}006 \times RPM} \quad ...(5)$$

3.1.3 Simplificações e ajustes

Na Eq. (5) a variável RPM está no denominador, isto implica na equação se tornar indefinida no caso de se usar o valor zero, o que certamente fará com que o controlador apresente algum mau funcionamento. Para contornar esse problema foi utilizada uma rotação mínima de 600 RPM como entrada na equação, e a rotação máxima de 4000 RPM.

Para não precisar obter um meio de converter a leitura da roda fônica de um motor real em um sinal aceitável de rotação para o Arduino, e para não precisar ter um motor funcionando durante os testes, foi utilizado um potenciômetro para simular o sinal da rotação, utilizando a função *map* para converter o sinal deste em uma variação de rotação. Dessa forma, o zero do potenciômetro equivale à rotação mínima definida, e o 1023, que é o valor máximo do potenciômetro, equivalem à máxima rotação definida, enquanto que os valores intermediários são distribuídos de maneira equivalente.

Os atuadores não atuam instantaneamente, e cada atuador possui um tempo de atuação que depende de seu princípio de funcionamento, geometria, eficiência, etc. Um possível tempo de atuação do atuador não será levado em conta no firmware, apenas serão gerados os sinais que acionariam o atuador, em forma de onda quadrada. O funcionamento dos atuadores depende também da tensão e corrente utilizada, e em um carro estes valores variam, o que também faria necessário um fator de correção nos sinais de saída.

3.1.4 Abertura e Fechamento das Válvulas

Na Figura 12 é demonstrada a estrutura de controle if com um comparador para que a cada 720 graus as variáveis de controle são zeradas, de modo que se recomesse o ciclo de funcionamento das válvulas dentro dos 4 tempos do ciclo Otto.

Figura 12 – Estrutura de controle para reiniciar o ciclo de acionamento de 720 graus.

```
actualMillis = millis();

if(actualMillis - previousMillis > 720/(0.006*RPM)){
  previousMillis = millis();
  previousMillis1 = millis();
  previousMillis2 = millis();
  previousMillis3 = millis();
  previousMillis4 = millis();
  In1 = 0;
  In2 = 0;
  Out1 = 0;
  Out2 = 0;
}
```

Fonte: Elaboração Própria

Neste arranjo, a diferença entre a variável “actualMillis”, que é o tempo atual de duração de funcionamento do firmware, e a variável “previousMillis”, que é o valor de tempo em quem o último ciclo começou, for maior quem a conversão de 720 graus na rotação atual em milissegundos.

A primeira vez que este comparador vai ser verdade, com a velocidade inicial de 600 RPM na Eq. (5), é quando o firmware funcionou por mais de 200 milissegundos ((200 – 0) > 720/(0,006*100)). Na Figura 13 é demonstrado como é feita a abertura das válvulas.

Figura 13 – Esquema para abertura das válvulas.

```
if((actualMillis - previousMillis1 > (timeOut1/(0.006*RPM))) && Out1 == 0){
  sin7 = HIGH;
  digitalWrite(valveOut,sin7);
  Out1 = 1;
}

  if((actualMillis - previousMillis2 > (timeIn1/(0.006*RPM))) && In1 == 0){
  sin8 = HIGH;
  digitalWrite(valveIn,sin8);
  In1 = 1;
}
```

Fonte: Elaboração Própria

Onde o controle if compara a diferença do tempo para abertura da válvulas com o tempo que é a condição de acionamento das válvulas, convertido das variáveis definidas com o ângulo, vistas na Figura 12.

Quando a condição do controle if for verdadeira, o sinal de saída vai ser impresso como “alto” para o atuador da válvula e assim ela será acionada (a válvula

será aberta). A variável “Out1” e “In1” servem para que dentro do ciclo de 720 graus, apenas uma vez este controle if seja verdadeiro.

Para o fechamento das válvulas, como visto na Figura 13, é utilizada a mesma estrutura da abertura das válvulas, visto na Figura 14.

Figura 14 – Esquema para fechamento das válvulas.

```
if((actualMillis - previousMillis3 > (timeOut2/(0.006*RPM))) && Out2 == 0){
  sin7 = LOW;
  digitalWrite(valveOut,sin7);
  Out2 = 1;
}

if((actualMillis - previousMillis4 > (timeIn2/(0.006*RPM))) && In2 == 0){
  sin8 = LOW;
  digitalWrite(valveIn,sin8);
  In2 = 1;
}
```

Fonte: Elaboração Própria

É possível notar que para cada comparador if teremos uma variável “previousMillis” diferente, e todas serão reiniciadas no fim de cada ciclo de 720 graus, como foi demonstrado na Fig 12.

3.2 Simulação dos sinais de saída

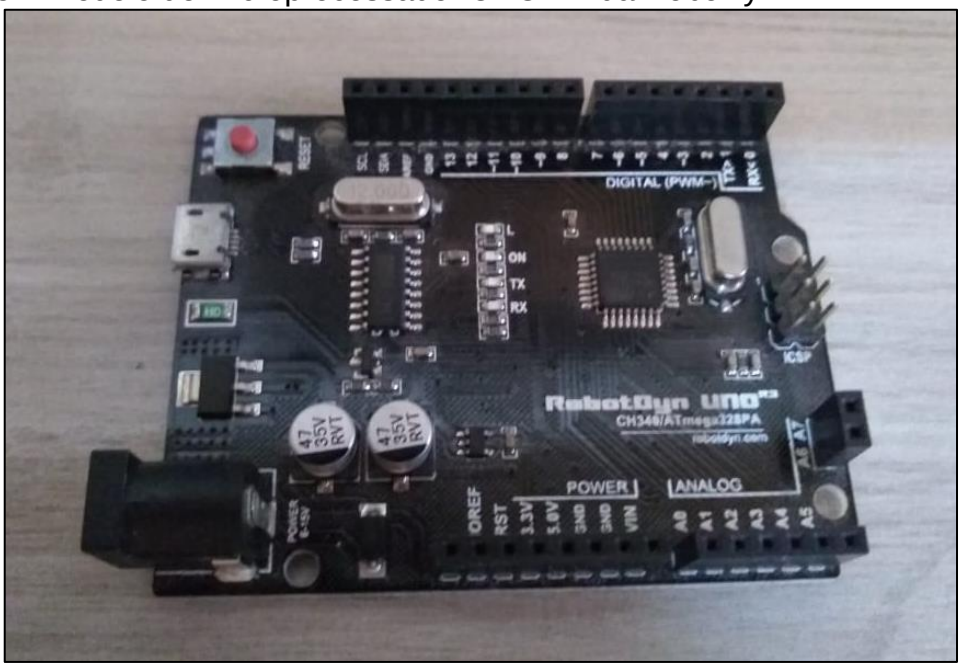
Os sinais de saída gerados na placa Arduino serão simulados e interpretados para ver se variam conforme o esperado quando há a mudança de sinal referente à rotação na entrada. Quanto ao que seria a duração de abertura (duração do sinal de onda quadrada, ou o que seria a duração do sinal “alto” na porta de saída do Arduino) e o tempo de abertura e fechamento (o momento que se dá o inicio e fim da onda quadrada, ou a mudança de “baixo” para “alto” e vice-versa na porta de saída do Arduino) serão comparados com a duração e os tempos de abertura e fechamento de diferentes modelos de eixos de comando para assim saber se são condizentes com a realidade.

3.2.1 Equipamentos Utilizados

3.2.2 Microcontrolador

O microcontrolador utilizado foi o UNO R3 da RoboDyn, um modelo genérico do Arduino UNO que utiliza o mesmo processador ATmega328P, como mostrado na Figura 15. Foram utilizadas as portas digitais 7 para escape, 8 para admissão, e a porta analógica A0 para a leitura do potenciômetro. O sinal para levantar a válvula será dado ao se “escrever” uma tensão de 5 Volts nas portas 7 e 8.

Figura 15 – Modelo de microprocessador UNO R# da RoboDyn.

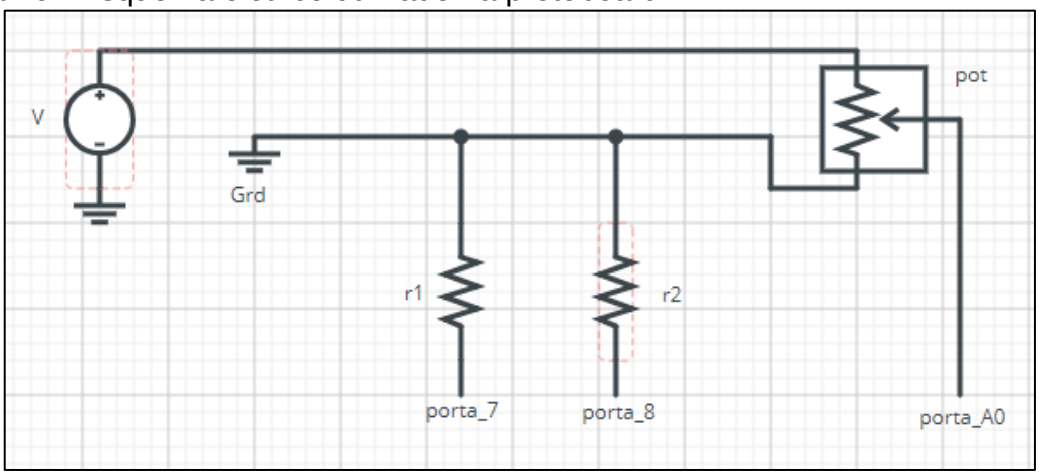


Fonte: Elaboração Própria

3.2.3 Esquema elétrico

O esquema elétrico utilizado está demonstrado na Figura 16, com resistores “r1” e “r2” de 150 Ohms, conectados nas portas 7 e 8 do Arduino, o potenciômetro “pot” conectado nas portas Grd e 5.0V para energizar, e na porta digital A0 para fazer a leitura. Todas as conexões foram feitas utilizando *jumpers,* comumente utilizados para projetos com o Arduino.

Figura 16 - Esquema elétrico utilizado na protoboard.



Fonte: Elaboração Própria

3.2.4 Osciloscópio

O osciloscópio utilizado foi o Tektronix DPO2024, como o observado na Figura 17, que se encontra no laboratório de eletrotécnica da Unipampa, campus Alegrete. Foram utilizados os canais 1 e 2, com o auxílio das ponteiras disponibilizadas pelos técnicos, para aferir a tensão sobre os resistores utilizados no esquema elétrico visto na Figura 17.

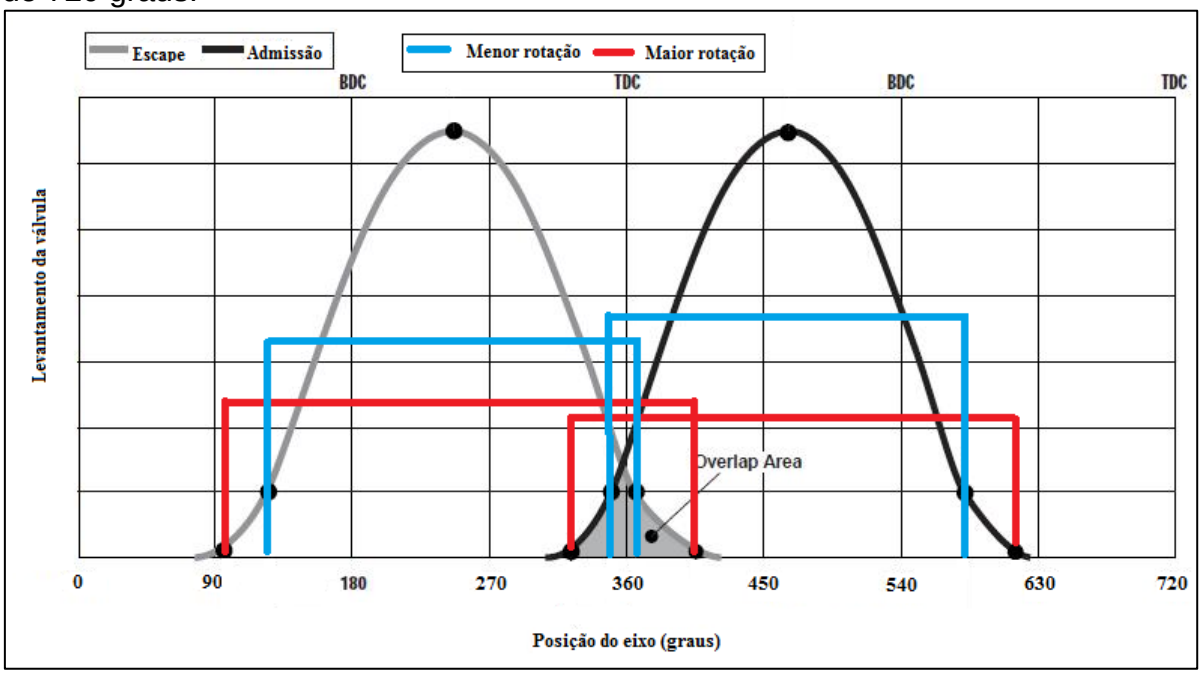
Figura 17 – Osciloscópio Tektronix DPO2024..

Fonte: Elaboração Própria

3.2.5 Previsão da Simulação

Ao se fazer a leitura dos sinais de saída do sistema com o auxílio do osciloscópio, a leitura prevista dos sinais seria algo similar ao demonstrado na Figura 18.

Figura 18 – Perfil de acionamento gerado em forma de onda quadrada em um ciclo de 720 graus.



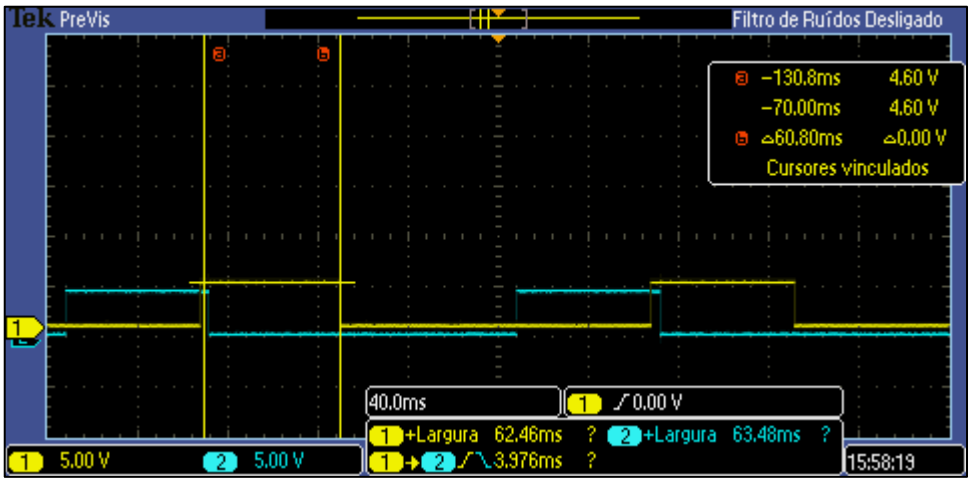
Fonte: Elaboração Própria

Neste gráfico o levantamento das válvulas não está em escala. As imagens adquiridas devem ser similares à curva “Menor rotação”, variando até a curva “Maior rotação” conforme o valor para a rotação na entrada do sistema varie do menor ao maior valor.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Na Figura 19 é demonstrada a leitura do sinal de saída com a entrada de 600 RPM.

Figura 19 – Leitura dos sinais de saída do sistema para 600 RPM, com auxílio de cursores.



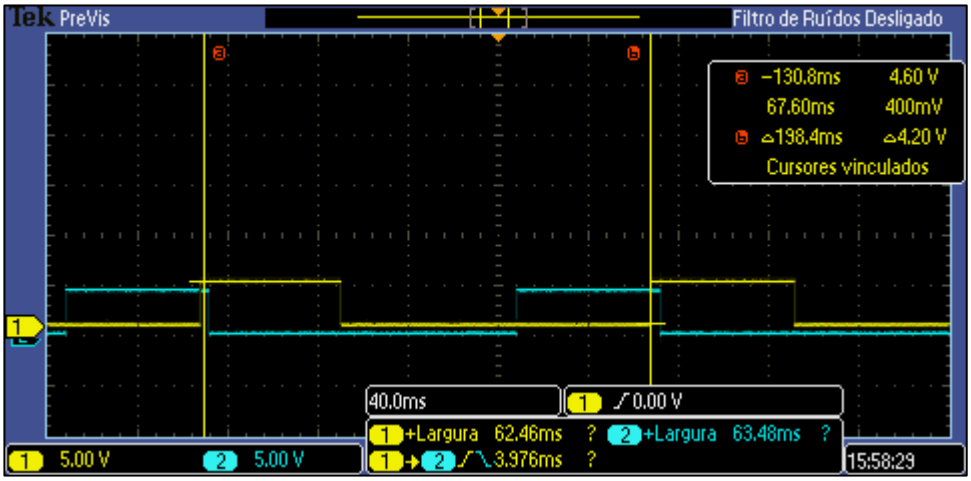
Fonte: Elaboração Própria

É possível ver que a duração do pulso do sinal de acionamento (marcador 1, amarelo) do escape é de 62,46 milissegundos, o pulso da admissão (marcador 2, verde) é de 63,48 milissegundos, e o ângulo de cruzamento (do degrau de subida do marcador 1 até o degrau de descida do marcador 2) é de 3,976 milissegundos.

Foram aplicados cursores para conferir se o valor dado pelo osciloscópio esta correto, e como pode ser visto no canto superior direito, do cursor “a” até o “b” está indicando 60,8 milissegundos, que é aproximadamente o mesmo valor da largura do sinal de admissão dado pelo osciloscópio.

Já na Figura 20 é demonstrado um cursor marcando a distância entre dois degraus de subida do pulso de admissão, podendo ser lido 198,4 milissegundos. Este valor seria o período do sinal, que equivale a um ciclo de 720 graus.

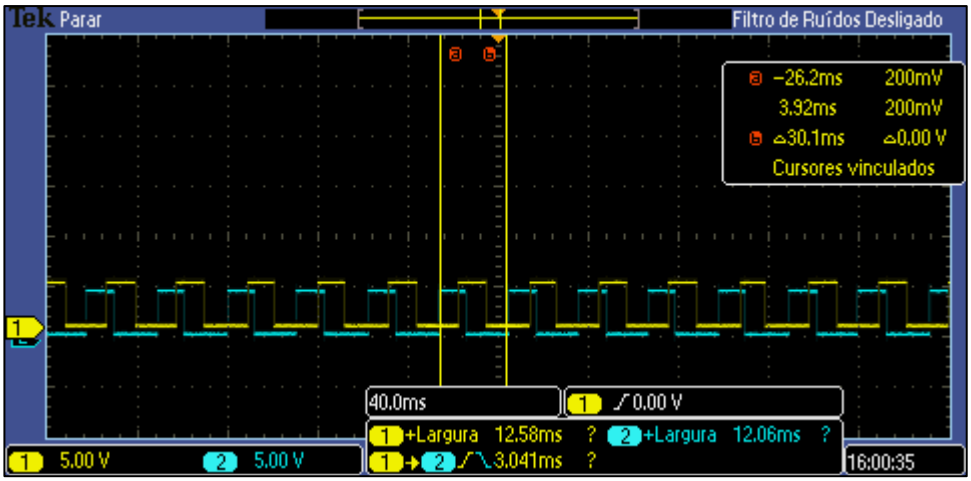
Figura 20 - Leitura dos sinais de saída do sistema para 600 RPM, com auxílio de cursores.



Fonte: Elaboração Própria

A Figura 21 é a leitura do sinal para 4000 RPM. Assim como na Figura 20, podemos observar a duração do sinal do escape (marcador 1, amarelo) com 12,58 milissegundos, a duração do sinal da admissão (marcador 2, verde) com 12,06 milissegundos, e a duração do cruzamento dos sinais com 3,041 milissegundos. O período do sinal está indicado como 30,1 milissegundos com o auxílio de cursores.

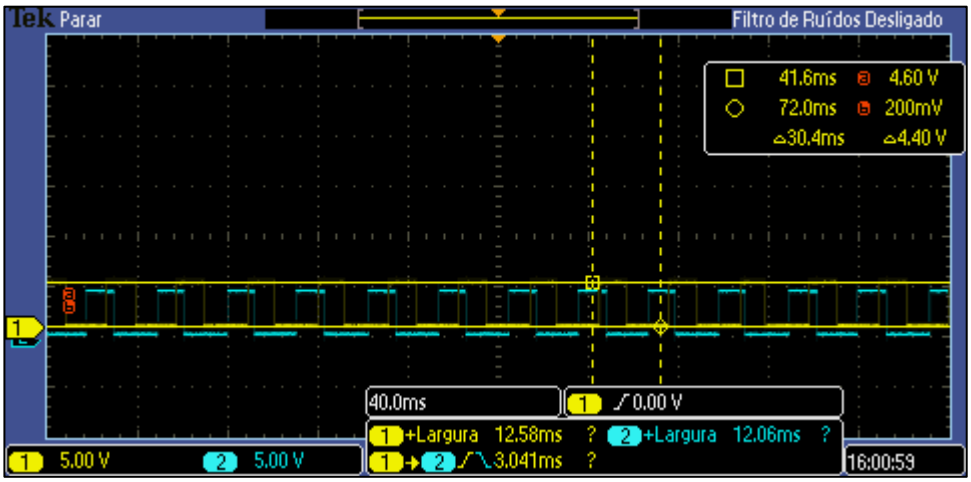
Figura 21 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores.



Fonte: Elaboração Própria

Na Figura 22 foram utilizados cursores para medir a tensão gerada em todas as leituras de sinais. A leitura foi feita sobre um resistor de 150 ohms, gerando uma tensão em torno de 4,4 volts.

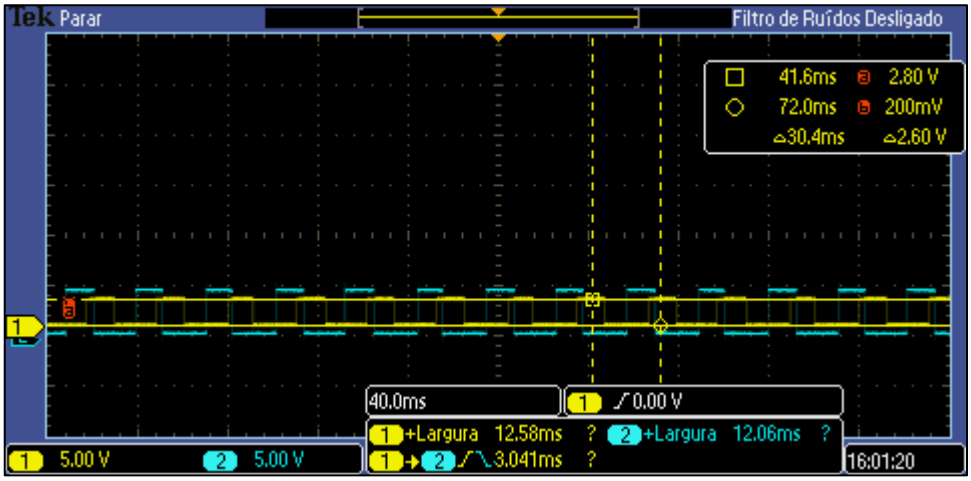
Figura 22 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores.



Fonte: Elaboração Própria

Na Figura 23 é demonstrada a leitura sobre o capacitor utilizando um led vermelho em série, gerando uma tensão de 2,6 volts.

Figura 23 - Leitura dos sinais de saída do sistema para 4000 RPM, com auxílio de cursores.

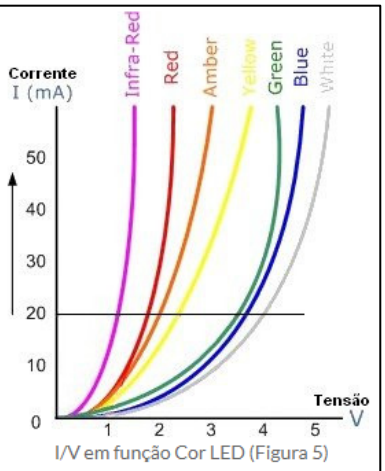


Fonte: Elaboração Própria

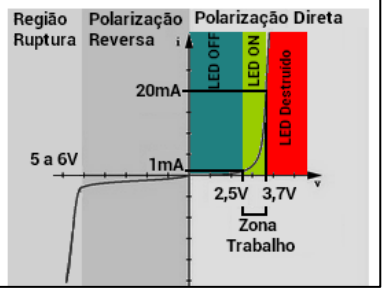
Isto acontece porque o LED vermelho gera uma queda de tensão de aproximadamente 1,8 volts, devido sua curva de funcionamento corrente X tensão, dados estes vistos na Figura 24.

Figura 24 – Especificações de LEDs.

Comp. Onda (nm)	Cor	Tensão Fwd (Vf@20ma)	Intensidade 5mm LEDs	Ang°	LED Material
940	Infravermelho	1.5	16mW @50mA	15°	GaAlAs/GaAs
880	Infravermelho	1.7	18mW @50mA	15°	GaAlAs/GaAs
850	Infravermelho	1.7	26mW @50mA	15°	GaAlAs/GaAs
660	Ultra Red	1.8	2000mcd @50mA	15°	GaAlAs/GaAs
635	High Efficiency Red	2.0	200mcd @20mA	15°	GaAsP/GaP
633	Super Red	2.2	3500mcd @20mA	15°	InGaAlP
620	Super Orange	2.2	4500mcd @20mA	15°	InGaAlP
612	Super Orange	2.2	6500mcd @20mA	15°	InGaAlP
605	Orange	2.1	160mcd @20mA	15°	GaAsP/GaP
595	Super Yellow	2.2	5500mcd @20mA	15°	InGaAlP



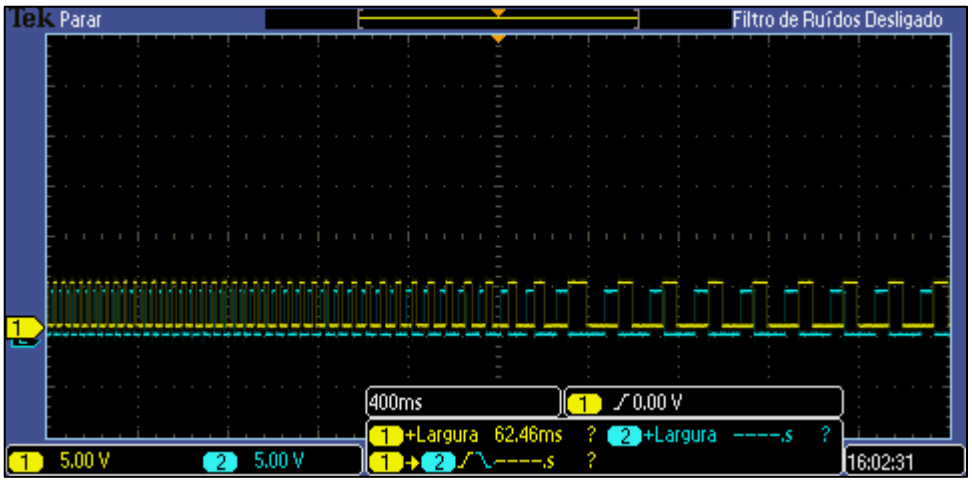
I/V em função Cor LED (Figura 5)



Fonte: Eletrônica PT

A Figura 25 mostra o comportamento da onda ao se variar a rotação (girando o potenciômetro) de 4000 até 600 RPM.

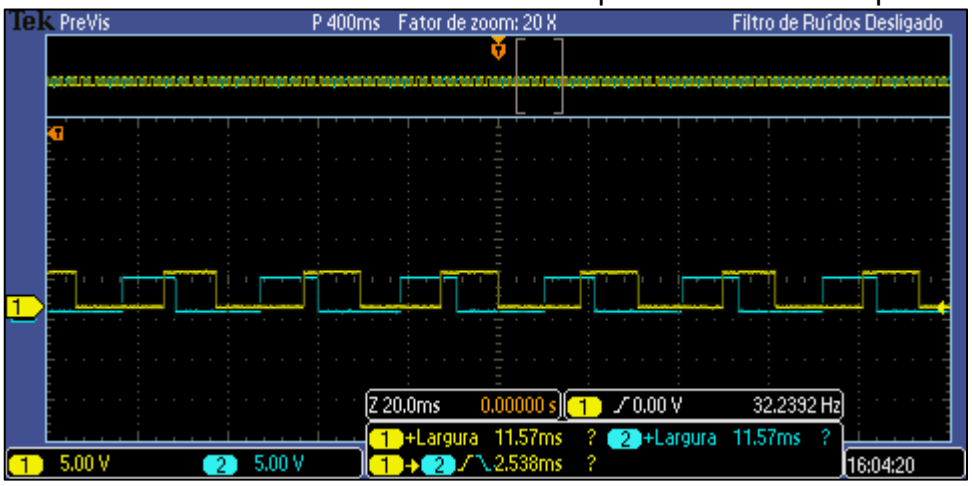
Figura 25 - Leitura dos sinais de saída do sistema variando de 4000 RPM para 600 RPM.



Fonte: Elaboração Própria

A Figura 26 Mostra outros valores lidos para 4000 RPM, com 11,57 milissegundos para a duração da admissão e do escape, e 2,538 milissegundos para a duração do cruzamento das válvulas.

Figura 26 - Leitura dos sinais de saída do sistema para 4000 RPM ampliado 20X.



Fonte: Elaboração Própria

Utilizando os valores da Figura 20 na Eq. (5), temos os valores de ângulo demonstrados na Tabela 2 para 600 RPM, com a conversão para graus utilizando a Eq. (5).

Tabela 2 – Conversão da dos tempos dos sinais para graus, lidos para 600 RPM.

	Leitura do sinal [ms]	**Conversão para graus**
Duração da admissão	62,46	224,85
Duração do escape	63,48	228,52
Ângulo de cruzamento	3,976	14,31
Período	198,4	714,26

Fonte: Elaboração Própria

Comparando estes valores com os vistos na Figura 9 para o modelo TW26, podemos observar que os ângulos simulads pelo controlador estão em torno de 2 graus menores, e apenas o valor do período demonstra em torno de 6 graus menor, mas o valor lido pelos cursores tende a variar mais dependendo da sintonia fina utilizada para posicioná-los.

Estes 2 graus na Eq. (5), com a rotação de 600 RPM, é de aproximadamente 0,555 milissegundos, o que é um tempo tão pequeno que pode indicar um erro de arredondamento dos valores, por parte do osciloscópio.

Utilizando os valores da Figura 21 na Eq. (5), temos os valores de ângulo demonstrados na Tabela 3 para valores de 4000 RPM.

Tabela 3 - Conversão da dos tempos dos sinais para graus, lidos para 4000 RPM.

	Leitura do sinal [ms]	**Conversão para graus**
Duração da admissão	12,58	301,92
Duração do escape	12,06	289,44
Ângulo de cruzamento	3,041	72,98
Período	30,1	720,24

Fonte: Elaboração Própria

Comparando estes valores com os vistos na Figura 9 para o modelo TW64G, podemos observar que os ângulos simulados pelo controlador nesta figura estão indicando aproximadamente 10 graus a mais na duração de escape e no ângulo de cruzamento, o que é aproximadamente de 0,416 milissegundos de erro, enquanto que o período está indicando um tempo praticamente perfeito. Nesta figura a duração da admissão está em torno de 30 graus de diferença, o que pela Eq. (5) indica aproximadamente 1,25 milissegundos a menos q deveria ser lido (em torno de 11,33 milissegundos).

Na Figura 26 Podemos ver um valor de 11,57 milissegundos de duração da admissão, que é em torno de 277,68 graus segundo a Eq. (5). Este valor esta aproximadamente 5 graus a mais do que o desejado, visto na Figura 9 para o modelo TW64G. Esta variação nos valores pode ser em decorrência de algum mau contato durante simulação, ou de algum arredondamento dos valores. A precisão necessária para aferir 1,25 milissegundos de diferença exigiria condições perfeitas de medição, com aparelhos mais avançados, cabeamento mais eficiente ou talvez um controle das condições ambientais durante a leitura dos sinais.

É demonstrado na Figura 25 o comprimento da onda quadrada aumentando para uma menor rotação, indicando que o que realmente acontece é que a duração da abertura das válvulas é menor para a maior rotação. Por exemplo, para 600 RPM a duração dos 720 graus é de 200 milissegundos, enquanto que a duração dos 720

graus para 4000 RPM é de aproximadamente 30 milissegundos, conforme visto que a rotação está no denominador da Eq. (5). Isto quer dizer que a variação da duração de abertura é na verdade uma variação da fração do ciclo de 720 graus. E o mesmo ocorre para o ângulo de cruzamento, em que quanto maior a rotação, maior é a fração dos 720 graus em que ocorre o cruzamento das válvulas.

Podemos conferir os valores simulados para 600 RPM, fazendo as frações dos valores encontrados na Tabela 2, como demonstrado na Tabela 4. A fração foi feita utilizando o tempo que se deseja conferir sua fração, dividido pelo período (ciclo de 720 graus).

Tabela 4 – Comparação das frações de tempo e ângulo dos valores teóricos e encontrados para 600 RPM.

	Valor teórico	**Valor simulado**
Fração da admissão	226/720=0,3138	62,46/198,4=0,3148
Fração do escape	230/710=0,3194	63,48/198,4=0,3199
Fração do cruzamento	16/720=0,0222	3,976/198,4=0,0200

Fonte: Elaboração Própria

Podemos fazer o mesmo para os valores encontrados para 4000 RPM na Tabela 3, como demonstrado na Tabela 5. Para a admissão será utilizado o valor de 11,57 milissegundos da Figura 26.

Tabela 5 - Comparação das frações de tempo e ângulo dos valores teóricos e encontrados para 4000 RPM.

	Valor teórico	**Valor simulado**
Fração da admissão	272/720=0,3777	11,57/30,1=0,3843
Fração do escape	280/720=0,3888	12,06/30,1=0,4006
Fração do cruzamento	62/720=0,0861	3,041/30,1=0,1010

Fonte: Elaboração Própria

É possível notar que as frações dos valores encontrados na leitura dos sinais são muito próximas dos valores teóricos, variando no máximo 0,2% (no ângulo de cruzamento) para 600 RPM e 1,49% (no ângulo de cruzamento) para 4000 RPM. Esta variação da fração de 0,0149 é equivalente a 10,728 graus, que é um valor baixo, enquanto que a variação dos tempos de abertura está menores ainda.

5 CONSIDERAÇÕES FINAIS

A elaboração de um *firmware* pode se tornar um desafio para alunos da Engenharia Mecânica, visto que programação não é algo recorrente durante o curso. Participar de cadeiras da grade curricular da Ciência da Computação ou Engenharia de *Software* pode ser uma grande contribuição para alunos interessados nesta área, além de estudos próprios sobre sistemas embarcados. O presente trabalho foi uma grande contribuição para o autor devido aos conhecimentos adquiridos acerca de programação e automação.

Os dados obtidos com o firmware elaborado para o controlador Arduino foram satisfatórios, principalmente para baixas rotações. Para altas rotações os dados ficaram satisfatórios do ponto de vista de projeto, mas faltou uma maior estabilidade para os sinais de saída, que pode ser resultado da simplicidade dos equipamentos utilizados. Equipamentos mais sofisticados podem colaborar para maior precisão dos sinais, visto que o firmware funcionou como desejado.

Apesar de haver alguma variação nos valores simulados no osciloscópio em relação aos valores desejados, os tempos de duração dos sinais de acionamento das válvulas encontrados foram satisfatórios. Ao comparar a fração de tempo do ciclo em que é dado um sinal para acionamento dos atuadores, o valor simulado ficou muito próximo do desejado, com uma pequena diferença de no máximo 1,3%, enquanto que na duração total do sinal, percebeu-se um erro em torno de 0,5 milissegundos. Independente da rotação o Arduino demonstra a mesma insuficiência, provavelmente um atraso no chaveamento da porta ao mudar o estado de zero volts para 5 volts, deixando o tempo de acionamento mais curto. Para 4000 RPM esse meio milissegundo equivale a 12 graus, que já é um valor suficiente pra se prestar atenção. A utilização de uma placa mais potente pode render resultados ainda mais precisos, aumentando ainda mais a eficiência do sistema.

Apesar de os dados obtidos com o *firmware* desenvolvido serem muito satisfatórios, ainda faltaria muito para ser aplicável. Várias simplificações foram feitas, que dependem de fatores construtivos externos a serem decididos, como os atuadores, fonte de alimentação, e até mesmo no caso da confecção de um sistema embarcado totalmente dedicado para essa função, o que aumentaria muito a precisão do sistema.

A variação na tensão sobre o resistor proveniente do LED conectado em série demonstra a importância da otimização do sistema, para que haja o mínimo possível de perdas. Uma possibilidade para adquirir a tensão desejada para os atuadores seria através da utilização de uma fonte de maior tensão, junto de um sistema de retificação com diodos, diminuindo assim a preocupação com a inconstância da fonte disponível, no caso de baterias.

5.1 Sugestões para trabalhos futuros

- Projetar um atuador para fazer a abertura e fechamento das válvulas, de modo que o este devolva um sinal de posição das válvulas para fins de segurança, e que aguente as altas temperaturas envolvidas na combustão;
- Aperfeiçoar o firmware apresentado neste trabalho, de modo a fazer com que seu funcionamento seja mais eficiente, e este leve em conta as peculiaridades do atuador, como tempo de atuação, folga, e uma parada de segurança quando houver algum problema com o posicionamento da válvula, além da variação da tensão na fonte, caso haja;
- Utilizar um controlador mais potente ou com processamento paralelo, para que os sinais sejam mais precisos, ou ainda um controlador totalmente dedicado à esta função.

6 REFERÊNCIAS

ARDUINO. **Biblioteca**. Disponível em: <https://www.arduino.cc/en/Reference/HomePage?from=Reference.Extended>. Acesso em 03 de julho de 2017.

ARDUINO. **Introdução**. Disponível em: <https://www.arduino.cc/en/guide/introduction#>. Acesso em 03 de julho de 2017.

ARDUINO. **Loja**. Disponível em: <https://store.arduino.cc/usa/>. Acesso em 03 de julho de 2017.

BOSCH **Manual de tecnologia automotiva,** 25ª Ed., São Paulo: Editora Edgar Blücher, 2005.

BRUNETTI, F.. **Motores de combustão interna** - **Volume 1**, São Paulo: Blucher, 2013.

CLASSIC CYRCLES. **Technical Resources.** <https://www.classiccycles.org/media//DIR_1653304/DIR_1679204/DIR_1832156/8edc6cecb60a7ac3ffff84c3ffffe415.pdf>. Disponível em 23 de setembro de 2018.

ELETRÔNICA PT. **LED, Diodo emissor de luz.** < https://www.electronica-pt.com/led> Disponível em 23 de setembro de 2018.

GRUMPY'S PERFORMANCE. <http://garage.grumpysperformance.com/index.php?threads/how-your-cam-lsa-effects-your-compression-torque-dcr.1070/>. Disponível em 23 de setembro de 2018.

LANGBRIDGE, J. A. **Professional embedded ARM development**, Indianapolis, IN: John Wiley & Sons, 2014.

LOCKRIDGE, G.; DZWONKOWSKI, B.; NELSON, R.; POWERS, S.; **Development of a Low-Cost Arduino-Based Sonde for Coastal Applications**; 2016.

OGATA, K.. **Engenharia de controle moderno**, 4ª edição, São Paulo: Pearson Education do Brasil, 2008.

STONE, R.; BALL, J. K. **Automotive engineering fundamentals**, Warrendale, PA: SAEInternational, 2004.

BASSHUYSEN, R. V.; SCHAFER, F., editores. **Internal combustion engine handbook – Basics, components, systems, and perspectives**, Warrendale, PA: SAEInternational, 2004.

CETINKUNT, S. **Mecatrônica**, Rio de Janeiro: LTC – Livros Técnicos e Cientificos, 2008.

SENAI **Mecânica de automóveis motores de combustão interna - Álcool e gasolina**, Santa Maria: 2003.

SOFTWARE LIVRE. O que é software livre. Disponível em: <http://www.softwarelivre.gov.br/tire-suas-duvidas/o-que-e-software-livre>. Acesso em 03 de julho de 2017.

PULKRABEK, W. W. **Engineering fundamentals of the internal combustion engine**, 2ª Ed., New Jersey: Pearson Prentice Hall, 2004.

APÊNDICE A - Biblioteca de Programação do Arduino

A.1 Estrutura

Setup(): É uma função chamada quando o programa começa. É usado para inicializar variáveis, modos de pinos, declarar bibliotecas, definir a velocidade de transmissão serial, etc. Esta função vai ocorrer apenas uma vez, a cada vez que o Arduino é ligado ou resetado.

No exemplo da Figura 27 é definido no Setup() que a velocidade de transmissão de dados em bits por segundo será de 9600 e o pino 3 (pois a variável buttonPin é 3) será uma saída).

Figura 27 - Exemplo da função Setup().

```
const int buttonPin = 3;

// setup initializes serial and the button pin
void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}
```

Fonte: Elaboração Própria

Loop(): A função Loop() cria um ciclo de repetição, fazendo com que todo o código escrito neste bloco seja lido e executado continuamente. Isto faz com que seja possível o Arduino estar constantemente lendo as entradas, processando e escrevendo nas saídas, tendo assim um sistema de controle.

Na Figura 28 é possível observar o exemplo do Loop() que repete o condicional If...Else e espera 1000 milissegundos continuamente.

Figura 28 - Exemplo da função Loop().

```
void loop()
{
  if (digitalRead(buttonPin) == HIGH)
    Serial.write('H');
  else
    Serial.write('L');

  delay(1000);
}
```

Fonte: Elaboração Própria

If: O If é uma estrutura de controle que é utilizada em conjunto com um comparador para testar quando uma certa condição foi satisfeita, como visto no exemplo na Figura 29 mostrando o uso do If, em que quando a variável “SomeVariable” for maior que 50, o que estiver dentro das chaves será executado.

Figura 29 - Exemplo da estrutura de controle If.

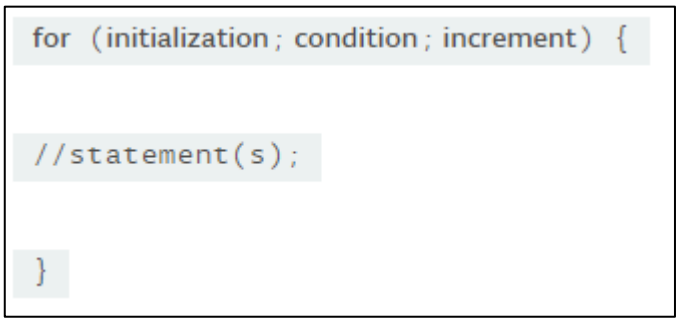
```
if (someVariable > 50)
{
  // do something here
}
```

Fonte: Elaboração Própria

For: O For é uma estrutura de controle que é utilizado para repetir um bloco de ações enquanto uma variável não chegar a um valor definido. Geralmente possui um contador que vai ser incrementado conforme o bloco é executado até que este alcance o valor que determina o fim do laço.

O exemplo da Figura 30 mostra que o For é composto por três partes, pela inicialização, que define a variável que será incrementada; a condição, que vai testar a variável definida na inicialização; e o incremento, que diz o quanto a variável definida vai ser incrementada.

Figura 30 - Componentes da estrutura de controle For.



Fonte: Elaboração própria

Operadores aritméticos: São caracteres responsáveis por fazer operações aritméticas, como adição (+), subtração (-), multiplicação (*), divisão (/). Temos ainda o sinal da igualdade (=) que é o operador de atribuição, responsável por dizer ao microcontrolador para atribuir a variável à esquerda do sinal com o valor ou expressão à direita dele.

Operadores Comparadores e Booleanos: Os comparadores são responsáveis por comparar dois valores, gerando assim um verdadeiro ou falso que geralmente é usado para decidir se o programa vai continuar dentro de um laço ou entrar em um condicional, por exemplo. Eles podem fazer diferentes comparações, como visto no Quadro 1. É importante não confundir o Operador Aritmético de Atribuição (=) com o Operador Comparador de Igualdade, pois o de atribuição não resulta um verdadeiro ou falso, sendo assim inútil onde se usaria o comparador.

Quadro 1 – Operadores comparadores.

Operador	Comparação
==	Igual
!=	Diferente
<	Menor
>	Maior
<=	Menor ou igual
>=	Maior ou igual

Fonte: Elaboração própria

Os Comparadores Booleanos são utilizados em conjunto com os Comparadores, fazendo a lógica booleana AND (&&), OR (||) e NOT (!). O

Comparador Booleano NOT nega o resultado de um comparador, atribuindo negativo a um resultado verdadeiro, e verdadeiro a um resultado falso, enquanto que os operadores AND e OR seguem suas respectivas tabelas verdade.

A.2 Variáveis

Constantes: Constantes são expressões pré-definidas linguagem do Arduino. São usadas para fazer com que o programa seja mais fácil de ler, e são divididas em alguns grupos:

1. Definindo nível lógico
    - **False:** É definido por 0;
    - **True:** É definido geralmente por 1, mas pode ser também qualquer inteiro diferente de zero, para a lógica booleana.
2. Definindo nível do pino
    - **HIGH:** Quando um pino é definido como entrada ele lê HIGH ao ter uma tensão maior que 3,0 V, e quando o pino é definido como saída e atribuido como HIGH ele vai estar com tensão de 5 V (ambos os casos em placas de 5V);
    - **LOW:** Quando um pino é definido como entrada ele lê LOW ao ter uma tensão menor que 1,5 V (em placas de 5V), e quando o pino é definido como saída e atribuído como LOW ele vai estar com tensão de 0 V.
3. Definindo modo dos pinos digitais
    - **INPUT:** Quando um pino está definido como IMPUT é dito que ele está em um estado de alta impedância. O pino faz uma demanda muito pequena de corrente do circuito em que está conectado, fazendo com que seja útil para leitura de um sensor.
    - **INPUT_PULLUP:** Quando o pino está definido como INPUT_PULLUP é como o IMPUT, mas estará ligado internamente à energia por um resistor pull-up, o que deixa seu estado como algo (HIGH), sendo assim útil para fazer a leitura de um baixo (LOW);
    - **OUTPUT:** Quando um pino está definido como OUTPUT é dito que ele está em um estado de baixa impedância, fazendo com que a placa possa suprir corrente para outros circuitos. O Arduino Mega pode

suprir até 40mA, o que é mais que suficiente pra acender lâmpadas LED.

Tipo de dado: O tipo de dado é o formato em que a informação é salva dentro das variáveis, e em muitas linguagens de programação é preciso declarar o tipo de dado no momento em que se cria a variável. As variáveis podem ter diferentes tipos de dados armazenados: números inteiros, frações, palavras, números binários, hexadecimal ou octal, etc.

A seguir serão descritos os principais tipos de dados armazenados:

- **Void:** Usado em declaração de funções;
- **Boolean:** Guarda um de dois valores: Verdadeiro ou Falso (1 ou 0), e ocupa apenas um byte de memória;
- **Char:** Pode guardar um caractere ou um vetor de caracteres (ou string);
- **Int:** Guarda um número inteiro de 2 bytes, variando de -32.768 até 32.767 (-2^15 até 2^15 – 1);
- **Long:** Guarda um número inteiro de 4 bytes, variando de -2.147.483.648 até 2.147.483.647;
- **Unsigned:** Guarda números sem contar os negativos, podendo variar de 0 até 65.545 (Unsigned int que guarda 2 byts), ou 0 até 4.294..967.295 (Unsigned long, que guarda 4 byts);
- **Float:** Número com ponto flutuante, em que caso não se coloque um ponto decimal este será tratado como um inteiro;
- **Arrays:** É um vetor que guarda um conjunto de variáveis acessadas por um índice.

A.3 Funções

1. Digital I/O (Input/Output)
    - **pinMode():** Configura se o pino será uma saída ou entrada (INPUT, OUTPUT ou INPUT_PULLUP);
    - **digitalRead():** Lê o valor que estiver no pino, sendo HIGH quando o pino estiver com mais de 3,0V ou LOW se estiver com menos de 1,5V;

- **digitalWrite():** Escreve LOW ou HIGH no pino. Se o pino estiver configurado como saída, será escrito 0V ou 5V, e se estiver configurado como entrada será desligado ou ligado seu resistor interno.

2. Analog I/O (Input/Output)
    - **analogRead():** Lê o valor que estiver no pino funcionando como um conversor A/D, transformando uma leitura entre 0V e 5V em um valor inteiro entre 0 e 1023, ou 0,049B por unidade. Leva 100 microsegundos para a leitura ser feita, sendo assim possível fazer 10.000 leituras por segundo;
    - **analogWrite():** Escreve um valor analógico (pulso PWM) no pino, na forma de uma onda quadrada continua, com uma frequência de 490Hz ou 980 Hz, dependendo do pino.
3. Tempo
    - **millis():** Retorna o número de milissegundos desde que o Arduino começou a rodar o atual programa.
    - **delay():** Interrompe o programa pelo tempo especificado no seu parâmetro (em milissegundos)

(ARDUINO Biblioteca, 2017)