UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO
PROGRAMA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

RENAN APARECIDO PELOGIA

PROJETO DE CORREÇÃO DE UMA PLATAFORMA COM DOIS GRAUS DE LIBERDADE

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2019

RENAN APARECIDO PELOGIA

PROJETO DE CORREÇÃO DE UMA PLATAFORMA COM DOIS GRAUS DE LIBERDADE

Trabalho de conclusão de curso apresentado ao Programa de Graduação em Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de “Bacharel em Engenharia de Controle e Automação”.

Orientador: Prof. Dr. Cristiano Marcos Agulhari

CORNÉLIO PROCÓPIO
2019

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia de Controle e Automação

FOLHA DE APROVAÇÃO

Renan Aparecido Pelogia

Projeto de correção de uma plataforma com dois graus de liberdade

Trabalho de conclusão de curso apresentado às 13:00hs do dia 27/06/2019 como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação no programa de Graduação em Engenharia de Controle e Automação da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____

Prof(a). Dr(a). Cristiano Marcos Agulhari - Presidente (Orientador)

_____

Prof(a). Dr(a). André Sanches Fonseca Sobrinho - (Membro)

_____

Prof(a). Dr(a). Luiz Francisco Sanches Buzachero - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

AGRADECIMENTOS

Agradeço primeiramente a minha família pelo incentivo e apoio incondicional ao longo dos anos de universidade.

Agradeço também ao professor Cristiano pela atenção e suporte dado durante toda a criação deste projeto, me ajudando inúmeras vezes a encontrar a luz no fim do túnel.

A todos os meus amigos que de alguma forma participaram da minha vida acadêmica compartilhando momentos de alegria e de angústia, trocando conhecimentos e habilidades.

Agradeço, por fim, a UTFPR por me proporcionar um conhecimento nunca antes imaginado, contribuindo para que me torne um ótimo profissional e, principalmente, um cidadão consciente.

”Se a educação sozinha não transforma a sociedade, sem ela, tampouco, a sociedade muda.” - Paulo Freire

RESUMO

PELOGIA, Renan. **Projeto de correção de uma plataforma com dois graus de liberdade**. 2019. 72 f. Trabalho de conclusão de curso – Bacharelado em Engenharia de Controle e Automação, Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2019.

Estabilidade é um conceito presente em muitos equipamentos cotidianos e é essencial para a segurança, conforto e bons resultados nessas aplicações. Este trabalho contempla o projeto de um sistema que consiste de duas placas, uma inferior que sofrerá uma série de perturbações, e uma superior inicialmente orientada de forma paralela à inferior, ambas conectadas por uma barra. O sistema também conta com um controlador, cujo objetivo é manter a placa superior na posição desejada independentemente da movimentação imposta à placa inferior. Este projeto pode ser aplicado em sistemas onde ocorrem muitas perturbações indesejadas, como veículos que trafegam por caminhos irregulares, cadeiras de rodas capazes de corrigir a posição de seu usuário em rampas acentuadas e mudanças bruscas de terreno em geral. Para a correção da placa são utilizadas técnicas de controle clássico aplicadas a tecnologias atuais como os microcontroladores e servomotores.

Palavras-chave: Controle. Correção. Microcontrolador. Servomotor.

ABSTRACT

PELOGIA, Renan. **Correction Project for a Two-Degree-of-Freedom Platform.** 2019. 72 f. Monograph – Control and Automation Engineering Graduate Program, Federal Technological University of Paraná. Cornélio Procópio, 2019.

Stability is a concept present in many equipments and it's essential to safety, comfort and good results in these applications. This work contemplates the design of a system consisting of two plates, one inferior that is affected by a series of perturbations, and one superior initially parallel to the inferior, both connected by a bar. The system also have a controller whose purpose is to keep the top plate in the desired position regardless of the imposed movement on the lower plate. This project can be applied in systems where there are many undesirable disturbances, such as vehicles traveling on irregular roads, wheelchairs capable of correcting the position of their seat when on steep ramps and abrupt shifts of terrain in general. The board control uses classic control techniques applied to current technologies such as microcontrollers and servo motors.

Keywords: Control. Correction. Microcontroller. Servomotor.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

A/D	Analógico/Digital
ACK	*Acknowledge*
CCP	*Capture/Compare/PWM*
GND	Ground
I2C	*Inter Integrated Circuit*
NACK	*Not Acknolegment*
PI	Proporcional Integral
PID	Proporcional Integral Derivarivo
PWM	*Pulse Width Modulation*
SPI	*Serial Peripheral Interface*
USART	*Universal Serial Asynchronous Receiver Transmitter*
VCC	Voltage Common Collector

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

O projeto de sistemas de controle está presente em milhares de equipamentos e ferramentas atuais, tornando a vida das pessoas muito mais facilitada e segura. Processos industriais de risco, como manuseio de serras e produtos químicos (ou tóxicos), podem ser administrados à distância por um controlador automatizado, aumentando consideravelmente a precisão do corte e reduzindo o número de acidentes, por exemplo. Um projeto geral de um sistema desse tipo se divide em duas partes: automação e controle.

A automação é o ato de tornar determinada ação automática, ou seja, usar de técnicas para que um dado sistema possa realizar determinada tarefa sem a ajuda humana. Segundo RIBEIRO (1999, apud VILELA; VIDAL, 2003), a automação está intimamente ligada à instrumentação, sendo esta a responsável por medir, transmitir, comparar e atuar em um sistema de forma a obter resultados sem a intervenção humana. Tais instrumentos são muito importantes para o sensoriamento e avanço da automação, além da evolução dos componentes eletrônicos , que anteriormente eram caros e não muito práticos. É graças à automação que hoje podemos telefonar para conhecidos sem a intervenção de uma telefonista, como era antigamente, por exemplo.

Controle, como o próprio nome diz, é a ação de controlar um sistema de modo a fazer com que ele se comporte de uma maneira definida. Sistemas de controle rústicos eram realizados manualmente e dependiam da ação humana para funcionar, como por exemplo a introdução de lenha em uma fogueira quando o fogo está fraco. Com o advento da automação, o controle passou a funcionar sem a intervenção humana e em constante ciclo de correção, como é o caso do controle por boia de um tanque simples (MATIAS, 2002). Muitas técnicas de controle foram desenvolvidas e continuam em desenvolvimento até hoje, como o controlador PID e suas variantes que continuam sendo aplicadas até os dias atuais (OGATA, 2010).

Uma plataforma auto-ajustável que é capaz de se manter horizontal é um projeto físico que compreende tanto a automação quanto o controle, e pode ser utilizado em diversas aplicações industriais e cotidianas. Manter a estabilidade de um veículo, por exemplo, contribuiria para um transporte mais seguro e confortável tanto para mercadorias quanto para pessoas. Uma aplicação em uma cadeira de rodas melhoraria a qualidade de vida de diversas pessoas, tendo em vista a dificuldade de locomoção em terrenos acidentados e inclinados. Sendo assim, é válido pensar na contribuição futura que um projeto como este pode exercer sobre projetos de pesquisa e aplicação.

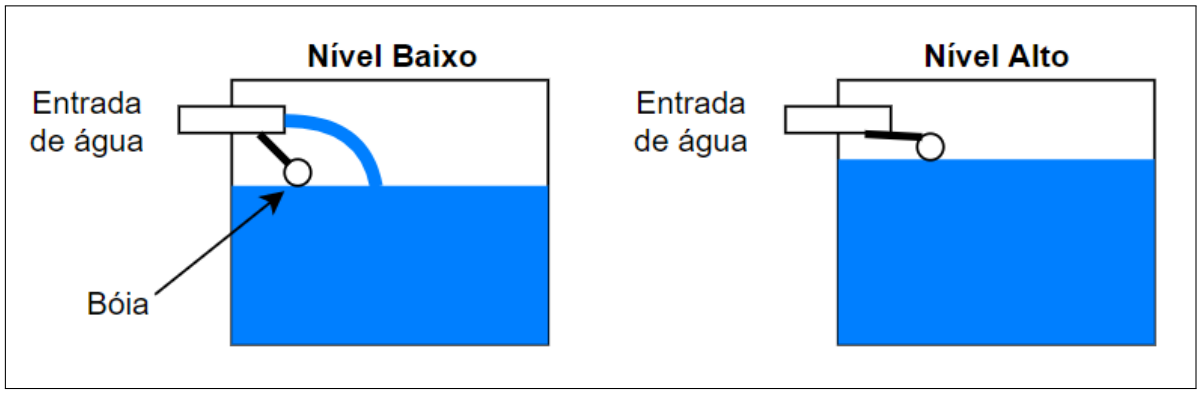
Este trabalho tem como objetivo o projeto de um protótipo com placas paralelas ligadas por uma haste na qual haja livre movimentação com dois graus de liberdade. Especificamente construir um protótipo físico funcional de auto-correção via microcontroladores PIC, detectar mudanças de inclinação de forma rápida e precisa, testar a viabilidade de um controle PID no sistema e identificar os melhores parâmetros para o funcionamento do controle.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 TEORIA DE CONTROLE

Muitos equipamentos atuais funcionam de forma automatizada para garantir que determinado evento ocorra da maneira desejada. Isso se deve às técnicas de controle empregadas na atuação e coleta de dados destes equipamentos. Um exemplo clássico de controle é o tanque simples, como o ilustrado na Figura 1. O objetivo do controle é regular a válvula de entrada que deve abrir ao passo em que o tanque perde volume de água, e fechar quando um certo limite for atingido.

Figura 1 – Tanque simples com boia



Fonte: (MATIAS, 2002, Adaptado)

A presença da boia no tanque representa um controle clássico ainda muito utilizado hoje em dia. Como pode ser visto na imagem, toda a vez em que o nível de água no tanque baixa, a boia desce abrindo a passagem para que mais água entre e preencha o reservatório. Por outro lado, se o reservatório chega a determinado volume, a boia sobe e fecha a válvula de entrada (MATIAS, 2002).

Com o passar dos anos, a tecnologia aplicada ao controle evoluiu consideravelmente, dando origem assim ao hoje conhecido Controlador Proporcional Integral Derivarivo (PID), composto pela soma de três algoritmos clássicos de correção: Proporcional, Integral e Derivativo. Apesar de relativamente antigos, os controladores PID são muito utilizados ainda hoje pela sua robustez e facilidade de ajuste, além de uma padronização de sua versão eletrônica a partir de 1950 (CASTRUCCI; BITTAR; SALES, 2011).

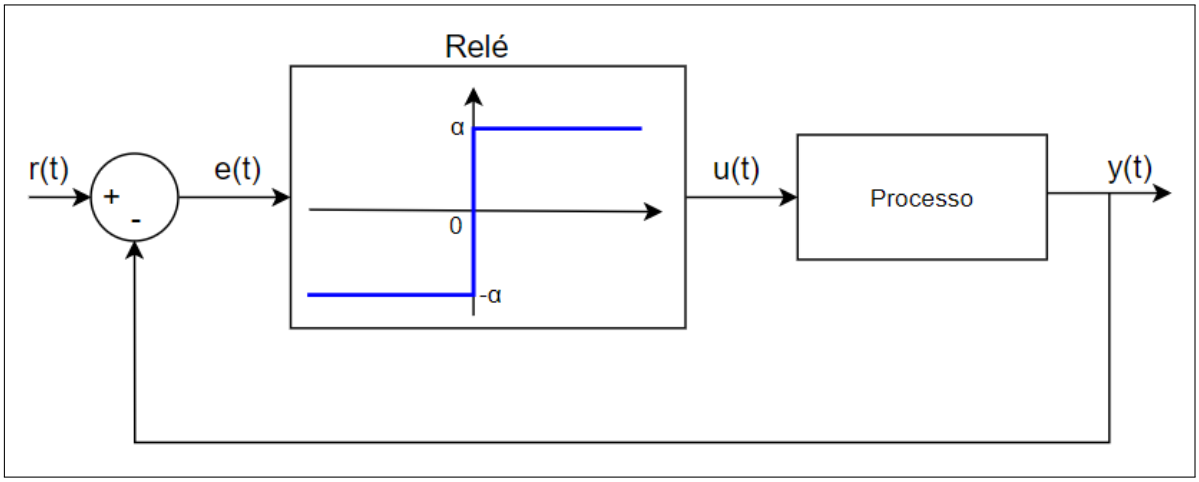
Existem ainda outras técnicas de controle de processos como o controle *On-Off* e algoritmos mais modernos como os controladores *Fuzzy*, sendo este último utilizado para simular a tomada de decisão humana frente a várias condições diversas, algo que não se enquadra no escopo da presente proposta.

2.1.1 Controle ON-OFF

O controle *On-Off* é o controle de mais simples implementação. Seu funcionamento é binário, apresentando apenas duas posições de funcionamento: aberto/fechado, ligado/desligado, ativo/inativo (Figura 2). Apesar de ser um dos controles mais fáceis de se trabalhar, é também o

menos robusto e preciso de todos os controles pois baseia-se na correção quando a variável desejada não se encontra na faixa de valores desejada, o que causa oscilações indesejadas na saída. Tomando como exemplo o tanque de nível, a correção ocorre quando o nível está abaixo ou acima do valor desejado. Dessa forma, o nível do tanque nunca permanecerá constante, mas sim variando ao redor do valor de referência (MATIAS, 2002).

Figura 2 – Funcionamento de um controle *on-off*



Fonte: (BAZANELLA; SILVA, 2000, Adaptado)

Matematicamente, pode-se considerar $u(t)$ como o sinal de entrada do sistema, $y(t)$ o sinal de saída e $e(t)$ o valor do erro em regime estacionário ao redor da referência definida. A equação de funcionamento do chaveamento é dada então por

$$u(t) = \begin{cases} \alpha & se \quad e(t) > 0 \\ -\alpha & se \quad e(t) < 0 \end{cases} \tag{1}$$

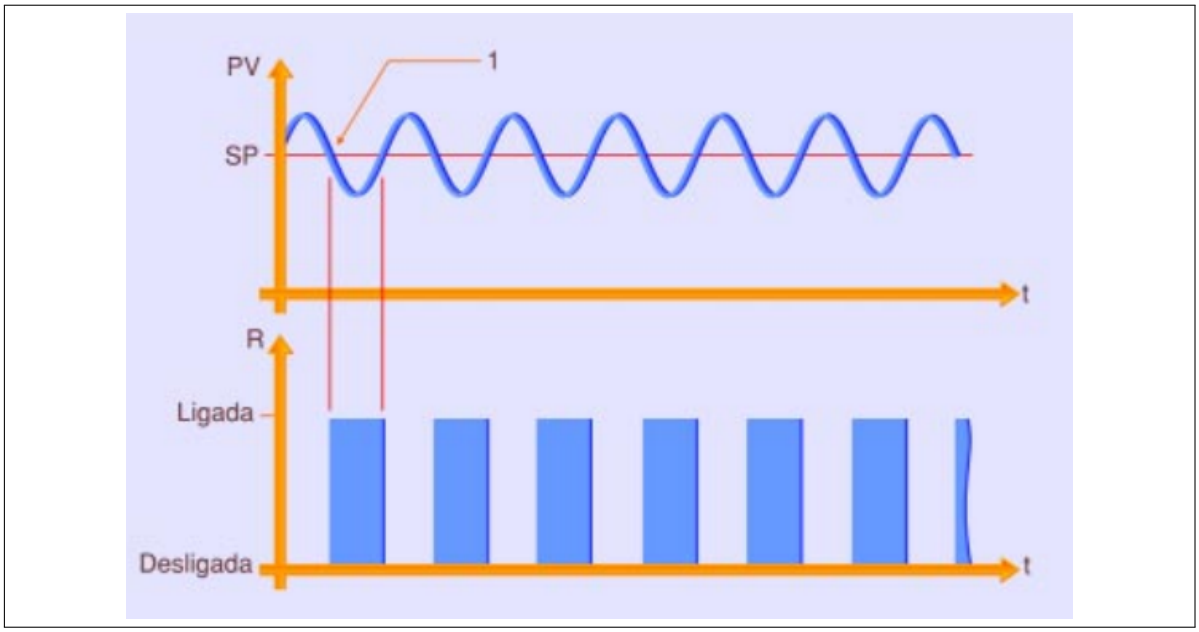
O chaveamento pode ser representado como um relé de duas posições que é chaveado em α e $-\alpha$ de acordo com o valor de erro $e(t)$ obtido, como visto na Figura 3. Como esse chaveamento só possui duas posições extremas e irá depender da saída para se autocorrigir, pode-se então dizer que o controle *on-off* sempre apresentará uma oscilação em torno da referência de acordo com o intervalo $[\alpha, -\alpha]$. Devido a esta oscilação percebe-se que o controle do tipo *on-off* não é preciso e, portanto, só torna-se viável sua aplicação em sistemas que não necessitem de precisão (BAZANELLA; SILVA, 2000). Alguns equipamentos atuais utilizam este tipo de controle por não necessitarem de respostas rápidas. É o caso dos ferros de passar, ar-condicionados e estufas, por exemplo (MATIAS, 2002).

2.1.2 Controle Proporcional

O controle proporcional se caracteriza por oferecer uma relação linear entre os valores do atuador e sua variável de controle. Diferentemente do controle *on-off*, este apresenta um comportamento mais suave de transição em seu funcionamento, agindo proporcionalmente ao valor controlado (MATIAS, 2002).

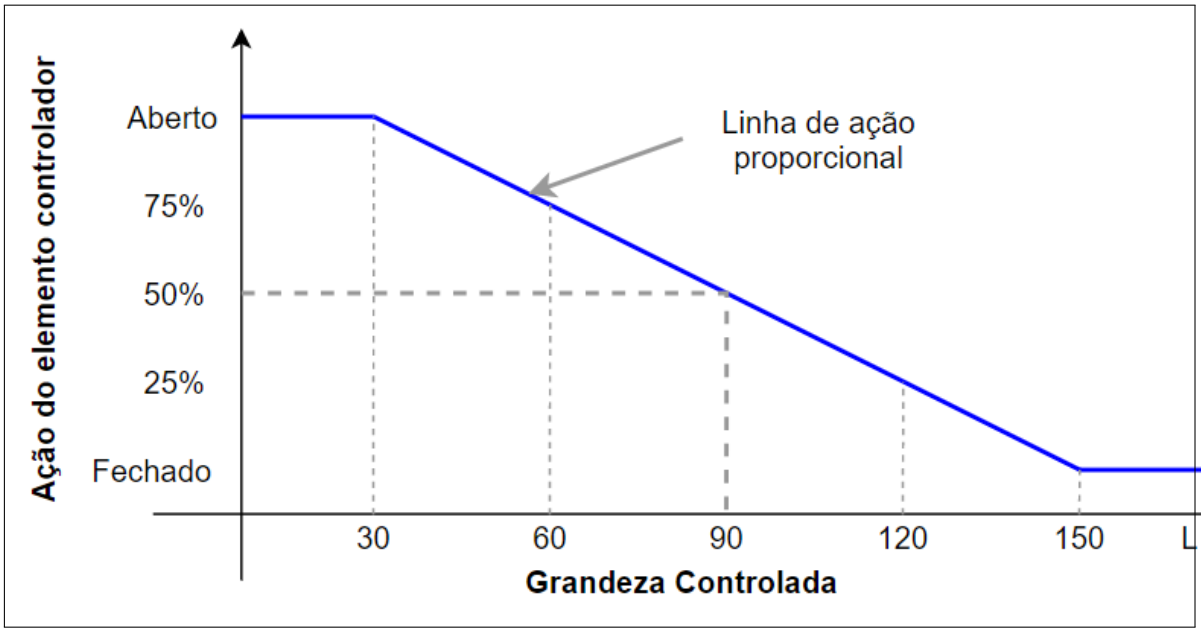
Na Figura 4 é representada a ação de controle sobre uma válvula de vazão controlada ao passo em que o volume de água no interior de um tanque se modifica. Quando a quantidade

Figura 3 – Comportamento da saída do sistema



Fonte: (MATIAS, 2002)

Figura 4 – Rampa proporcional



Fonte: (MATIAS, 2002, Adaptado)

de água atinge o valor de 30 litros, a válvula começa a fechar até atingir a posição de meio-aberta (50%) a 90 litros. Da mesma forma a variação continua até a marca de 150 litros, onde a válvula estará totalmente fechada (MATIAS, 2002).

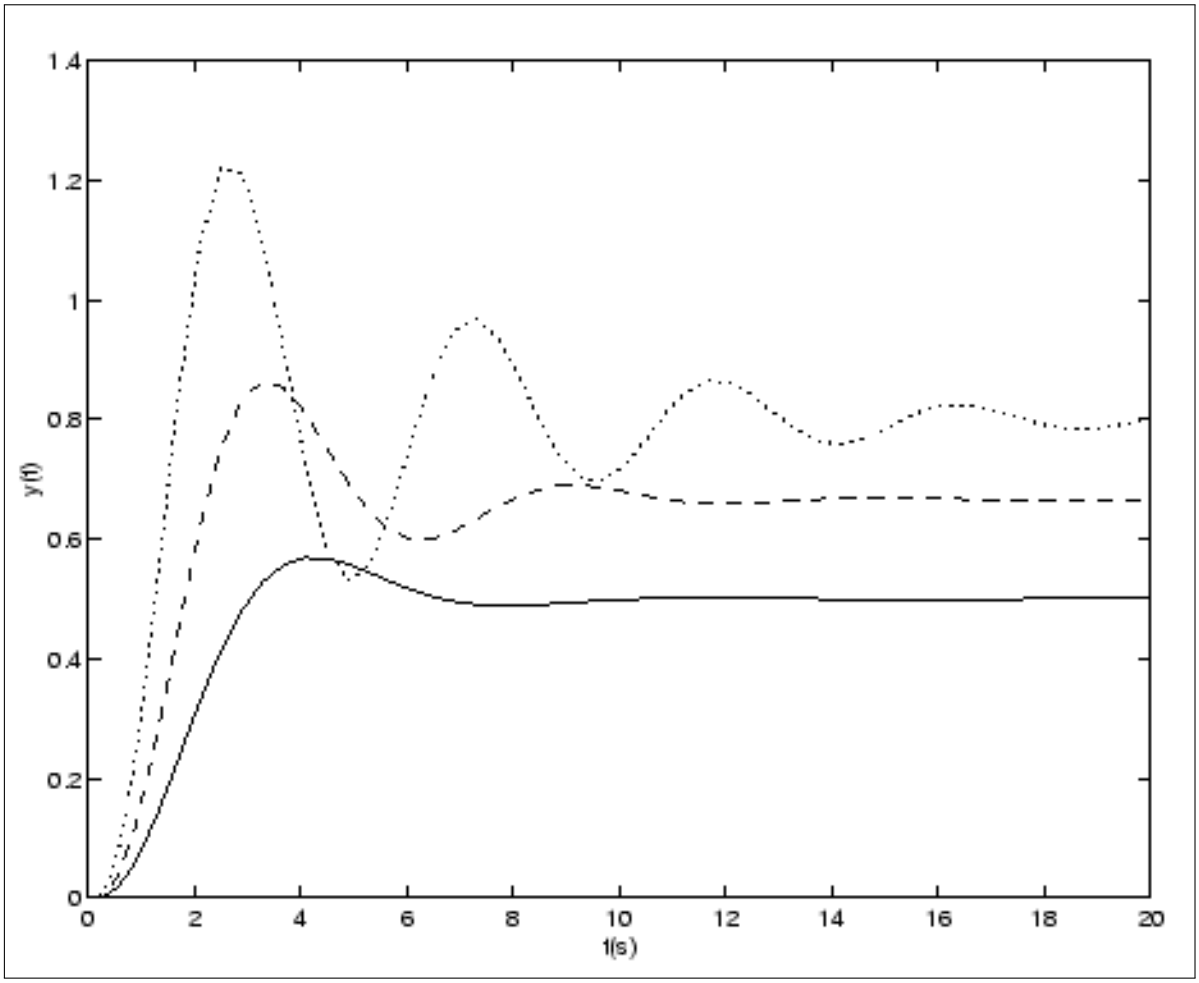
Matematicamente a ação proporcional pode ser descrita como mostra a Equação (2).

$$u(t) = K_p e(t) \tag{2}$$

Essa equação indica a proporcionalidade existente entre o valor de entrada $u(t)$ e o erro

de saída $e(t)$. Para valores de erro acima do valor de referência ($e(t) > 0$), o controle aplicado deverá ser positivo para estabilizar a resposta. O mesmo ocorre para valores de erro negativos ($e(t) < 0$), onde o controle deverá ser negativo. Alguns gráficos de resposta do controlador proporcional podem ser observados na Figura 5.

Figura 5 – Ação proporcional para diferentes valores de K_p. $K_p = 1$ (contínuo), $K_p = 2$ (tracejado), $K_p = 4$ (pontilhado). Valor de referência adotado = 1



Fonte: (BAZANELLA; SILVA, 2000)

Note que, quanto maior o valor de K_p, mais precisa é a resposta final em regime permanente, porém em contrapartida mais oscilatório fica o comportamento transitório em malha fechada, devendo-se tomar cuidado para não levar o sistema para a instabilidade (BAZANELLA; SILVA, 2000).

2.1.3 Controle Integral

Como visto na seção anterior, o controle proporcional estabiliza a resposta do sistema, porém resulta em um *off-set* (erro em regime permanente) em relação ao valor de referência. Esse valor é geralmente corrigido aplicando-se um somador à saída do sistema (manualmente), ou então adicionando a ação integral ao controlador (MATIAS, 2002).

A ação de controle integral é dada pela Equação (3), sendo T_i o tempo integral ou *reset-time*.

$$u(t) = \frac{1}{T_i} \int e(t)dt \tag{3}$$

A ação de controle integral tem a função de somar os valores de erro obtidos até então e garantir que em dado momento t, quando o erro $e(t)$ será nulo, o sinal $u(t)$ será mantido proporcional ao valor da integral. Dessa forma, a ação integral garante (em malha fechada) que o sistema siga por uma referência determinada com erro nulo, garantindo que $y(t) = r(t)$.

A função de transferência integral adiciona um polo na origem do sistema, garantindo erro nulo para determinadas referências, porém tal adição tende a piorar a estabilidade do sistema, chegando em alguns casos a torná-lo instável. Dessa forma, a ação integral deve sempre ser implementada acompanhada de outra, como a proporcional (BAZANELLA; SILVA, 2000).

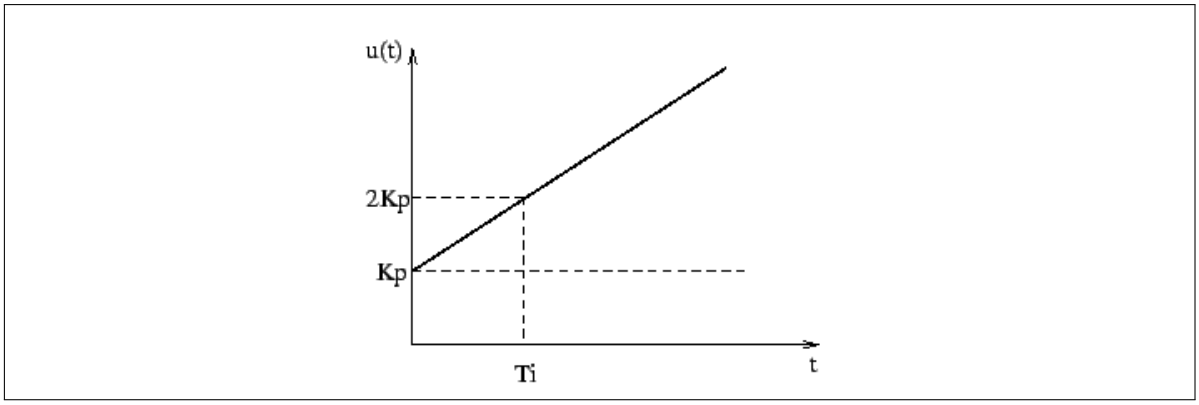
2.1.4 Controle Proporcional-Integral

A ação integral de controle garante que não ocorra erro em regime permanente pela adição de um polo na origem do sistema, porém essa mesma ação implica na instabilidade do sistema se implementada isoladamente. Para corrigir esse problema, a ação integral pode ser utilizada em conjunto com a proporcional, constituindo assim o controlador Proporcional Integral (PI), dado pela Equação (4) (BAZANELLA; SILVA, 2000).

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) \right) \tag{4}$$

A influência da ação integral sobre a proporcional é ilustrada na Figura 6. Na figura, T_i é o tempo integral que corresponde ao tempo em que dobra a ação de controle.

Figura 6 – Efeito da ação integral



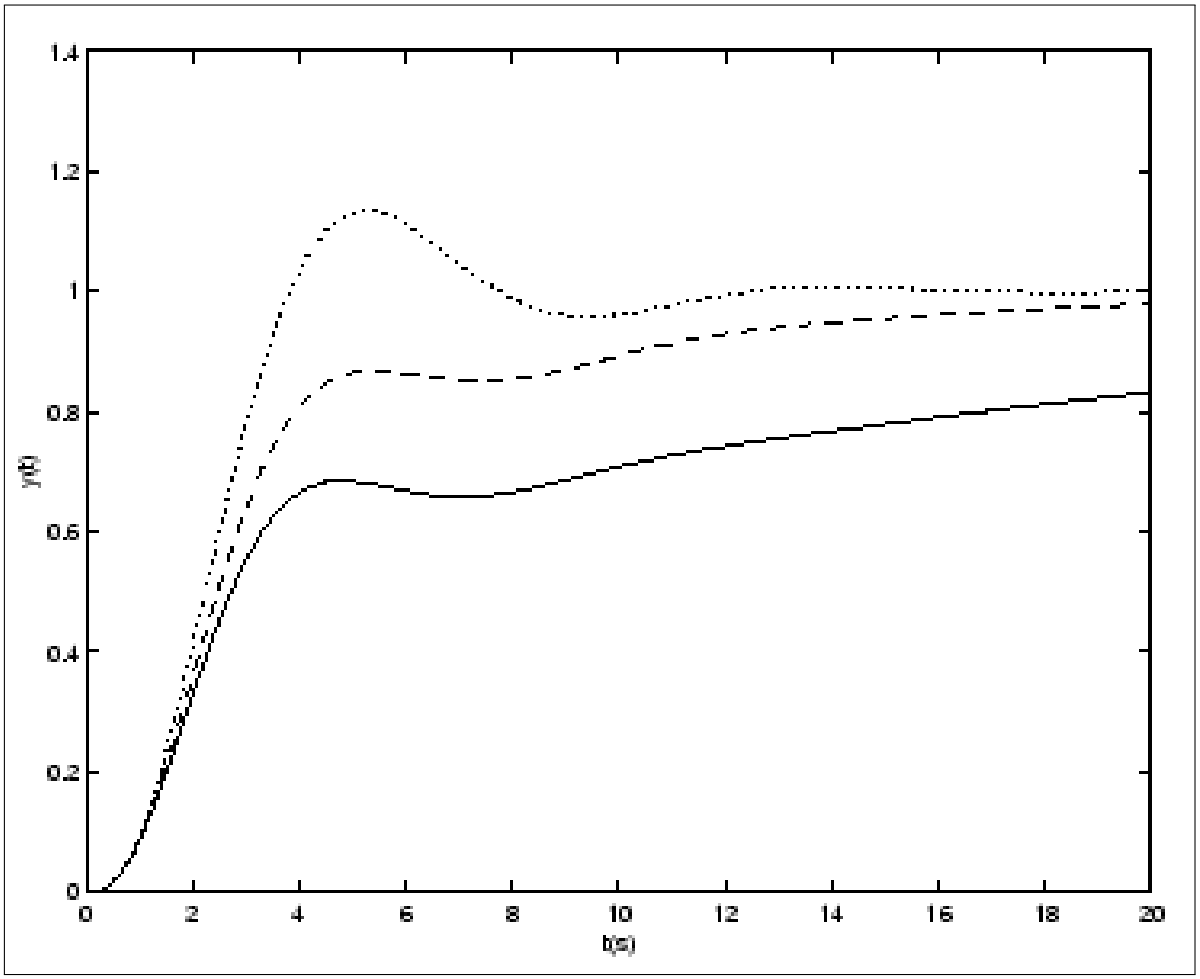
Fonte: (BAZANELLA; SILVA, 2000)

A função de transferência do controlador PI é dada pela Equação (5), sendo $1/T_i$ responsável por equilibrar o efeito instabilizante do polo na origem, tendo um zero em $-1/T_i$.

$$G_{pi}(s) = \frac{u(s)}{r(s)} = \frac{K(s + 1/Ti)}{s} \tag{5}$$

Quanto mais alto o valor de T_i, mais a ação proporcional se sobressai sobre a ação integral, contudo para valores muito altos o sistema apresentará erro de *off-set* na resposta. Da mesma forma, se T_i for adotado muito pequeno, o sistema apresentará oscilações e tenderá à instabilidade (BAZANELLA; SILVA, 2000). A Figura 7 mostra as respostas para T_i igual a 2, 4 e 10, considerando o valor de referência como sendo 1.

Figura 7 – Comportamento da resposta para diferentes valores de T_i. $T_i = 2$ (pontilhado), $T_i = 4$ (tracejado), $T_i = 10$ (contínuo)



Fonte: (BAZANELLA; SILVA, 2000)

2.1.5 Controle Derivativo

O controle derivativo se caracteriza por corrigir a resposta do sistema proporcionalmente à velocidade com que o desvio aumenta. Da mesma forma que a ação integrativa, a derivativa não é capaz de atuar de forma isolada e necessita da componente proporcional para seu bom funcionamento. Essa combinação de ações resulta em uma correção prévia de um desvio antes mesmo do mesmo ocorrer (MATIAS, 2002).

A equação da ação derivativa é dada conforme a Equação (6).

$$u(t) = T_d \frac{de(t)}{dt} \tag{6}$$

A implementação derivativa se dá pela função de transferência da ação anterior, como mostra a Equação (7), com a introdução de um polo em alta frequência para limitar o ganho em alta frequência, como mostrado na Equação (8). Sem a ação deste pólo, o sistema se torna mais sensível a ruídos.

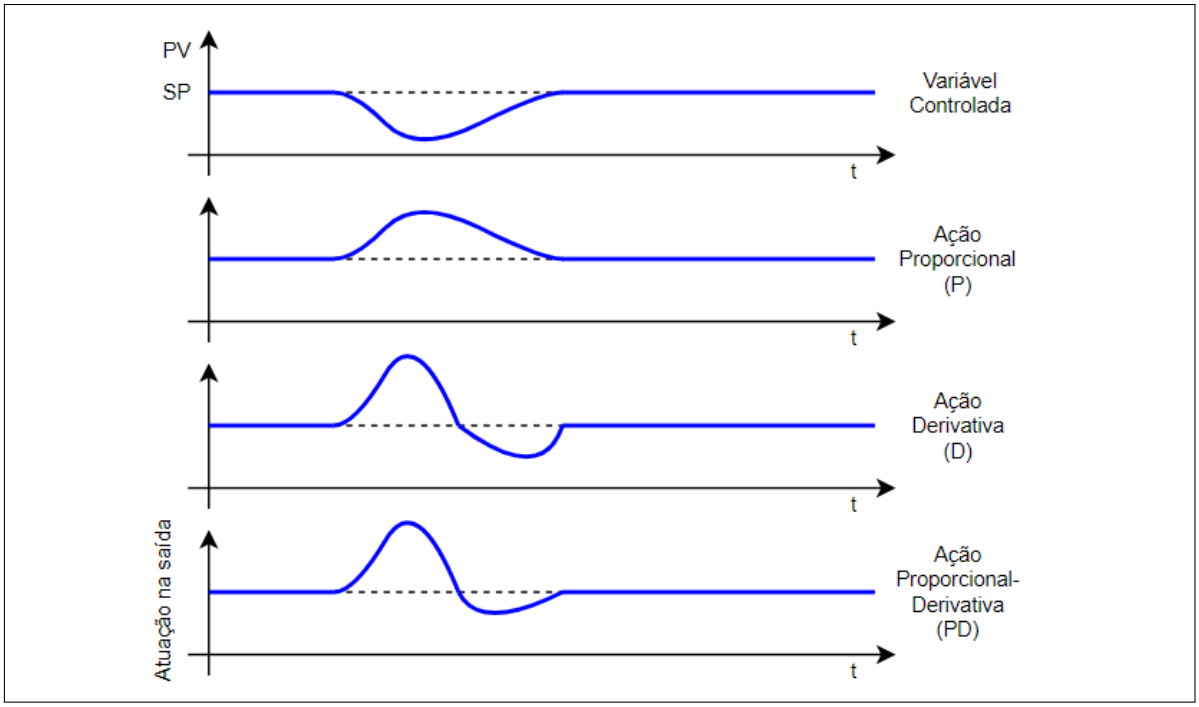
$$G_c(s) = \frac{u(s)}{e(s)} = T_d s \tag{7}$$

$$G_c(s) = \frac{u(s)}{e(s)} = T_d \frac{sp}{s+p} \tag{8}$$

A ação derivativa atua de forma intensa no início do controle aumentando a velocidade com que o sistema atinge a sua referência, porém deixa de atuar gradativamente para que as outras componentes de controle (proporcional ou integral) atuem na estabilização da resposta de saída (MATIAS, 2002).

Na Figura 8 o funcionamento da ação derivativa é representada junto a uma atuação proporcional.

Figura 8 – Atuação do controle derivativo



Fonte: (MATIAS, 2002, Adaptado)

2.1.6 Controle PID

Atualmente mais da metade dos controladores de processos industriais utilizam o controle PID ou variações do mesmo. Sua utilidade está na capacidade de ser aplicado em diversos sistemas, principalmente naqueles onde não é conhecido o modelo matemático do problema. O controle PID é uma ótima opção para atuação em controle de processos, e embora muitas vezes não proporcione um controle ótimo, garante sua utilidade com um controle satisfatório (OGATA, 2010).

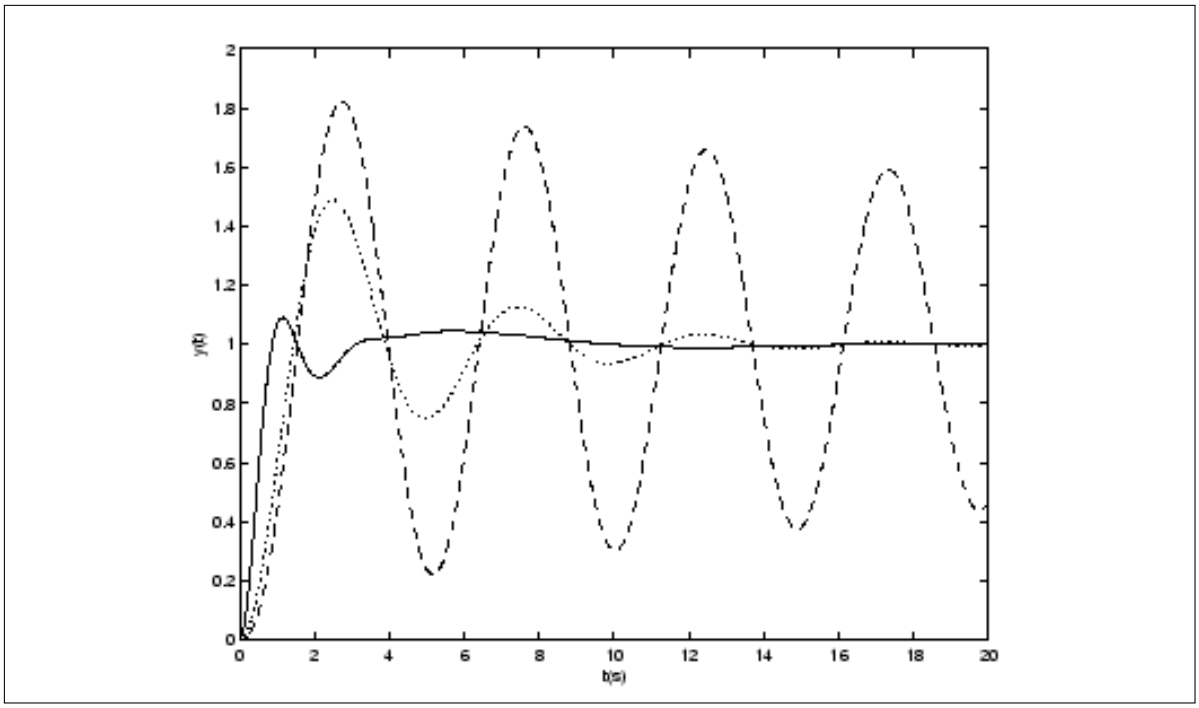
O controlador PID combina as ações de controle previamente vistas para proporcionar precisão e rapidez no tempo de resposta do sistema. A componente proporcional-integral (PI) leva o sistema a um erro nulo em regime permanente, e a adição do fator derivativo (D) contribui para que o sistema não instabilize, além de aumentar a velocidade por prever o valor de desvio na saída (BAZANELLA; SILVA, 2000).

A função de transferência do controlador PID é dada na Equação (9).

$$G_{PID}(s) = \frac{u(s)}{r(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d \frac{sp}{s+p}\right) = \frac{K_p(s^2 + \frac{1+T_d T_i}{T_i} s + \frac{p+T_i p}{T_i})}{s(s+p)} \tag{9}$$

Fixando-se valores para $K = 1$ e $T_i = 2$ e admitindo valores distintos para T_d, observa-se a resposta de um controlador PID na Figura 9.

Figura 9 – Saída PID para diferentes valores de T_d. $K_p = 4$, $T_i = 1,5$, $T_d = 0,1$ (tracejado), $T_d = 0,4$ (pontilhado), $T_d = 2$ (contínuo)



Fonte: (BAZANELLA; SILVA, 2000)

A efetividade do controlador PID está diretamente ligada com a qualidade da determinação dos parâmetros de controle K_p, T_i e T_d, sendo necessário métodos para obtenção dos mesmos.

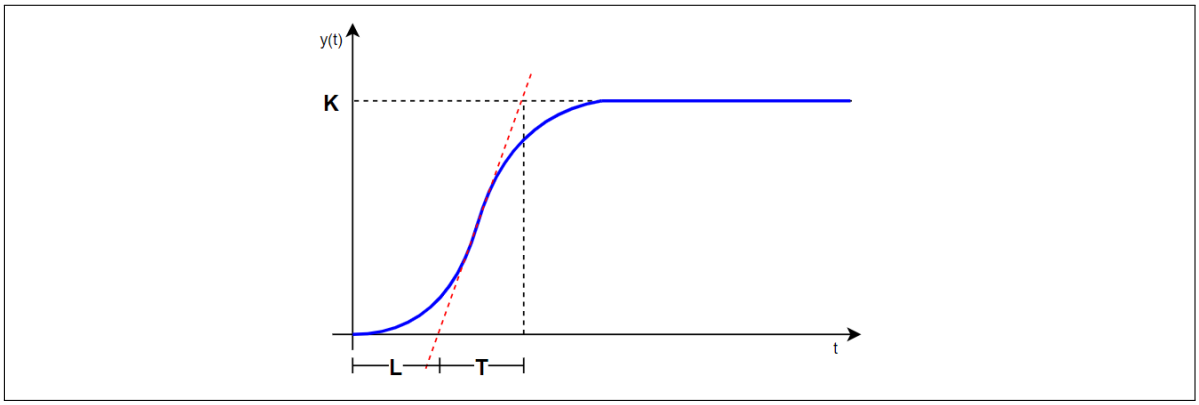
2.1.6.1 Sintonia de PID por Ziegler-Nichols

O processo de seleção dos parâmetros de um controlador para que seja atingido um dado desempenho é chamado de sintonia do controlador. Ziegler e Nichols propuseram métodos para obtenção das constantes K_p, T_i e T_d através da resposta transitória da planta. Há duas formas de sintonizar o controlador dependendo de como a planta pode ser tratada (OGATA, 2010).

Primeiro método:

No primeiro método é aplicado um degrau unitário à planta e observa-se sua resposta. Se a saída for em formato de S como na Figura 10, significa que a planta não possui integradores ou polos complexos conjugados dominantes. Ainda a partir dessa resposta, traçando-se uma tangente ao ponto de inflexão da curva, obtemos valores para o atraso L e a constante de tempo T.

Figura 10 – Curva de resposta em forma de S



Fonte: (OGATA, 2010, Adaptado)

A função de transferência do controlador pode ser aproximada por um sistema de primeira ordem com um atraso de transporte e em seguida, para escolher os valores de K_p, T_i e T_d é necessário realizar os cálculos segundo a Tabela 1 (OGATA, 2010).

Tabela 1 – Regra de sintonia de Ziegler-Nichols para o método de resposta ao degrau

Tipo de Controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0,9\frac{T}{L}$	$\frac{L}{0,3}$	0
PID	$1,2\frac{T}{L}$	$2L$	$0,5L$

Fonte: (OGATA, 2010)

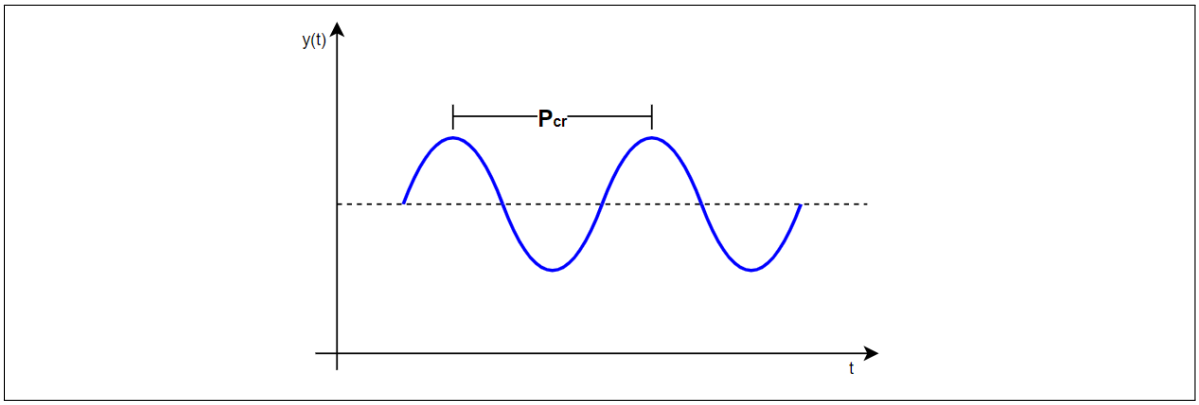
Dessa forma a função de transferência final do controlador torna-se a Equação (10).

$$G_c(s) = \frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} = 0,6T\frac{(s+\frac{1}{L})^2}{s} \tag{10}$$

Segundo método:

No segundo método, adota-se $T_i = \infty$ e $T_d = 0$ e aumenta-se K_d gradativamente até um momento que o sistema fique oscilatório. Quando isso ocorrer, o valor atual de K_d será o ganho crítico K_{cr} e pelo gráfico da resposta obtêm-se o período crítico P_{cr}. Da mesma maneira que no primeiro método, utiiza-se uma tabela para o cálculo das constantes (OGATA, 2010).

Figura 11 – Oscilação crítica para o segundo método de Ziegler-Nichols



Fonte: (OGATA, 2010, Adaptado)

Tabela 2 – Regra de sintonia de Ziegler-Nichols para o método com base em K_{cr} e P_{cr}.

Tipo de Controlador	K_p	T_i	T_d
P	$0,5K_{cr}$	∞	0
PI	$0,45K_{cr}$	$\frac{1}{12}P_{cr}$	0
PID	$0,6K_{cr}$	$0,5P_{cr}$	$0,125P_{cr}$

Fonte: (OGATA, 2010)

Após as substituições dos respectivos cálculos na equação do PID, temos a função de transferência da Equação (11).

$$G_c(s) = 0,075K_{cr}P_{cr}\frac{\left(s+\frac{4}{P_{cr}}\right)^2}{s} \tag{11}$$

2.2 DETERMINAÇÃO DE MODELO MATEMÁTICO

2.2.1 Princípios de Modelagem

Ao conhecer o modelo matemático da planta a ser controlada, a determinação dos parâmetros do controlador é facilitada. Para sistemas mecânicos adota-se a lei fundamental de que o somatório de todas as forças aplicadas ao modelo resultam em zero (SANTOS, 2008). Para a determinação da função de transferência, os seguintes passos são adotados:

1. Escolher um sentido positivo para o movimento;
2. Esboçar um diagrama de corpo livre com todas as forças que atuam no projeto;
3. Determinar a equação diferencial a partir da soma das forças igualadas a zero;
4. Aplicar a transformada de Laplace sob condições nulas e encontrar a função de transferência[1].

[1] Supondo para os cálculos um sistema linear

A Tabela 3 apresenta algumas relações de força para os três principais tipos de componentes.

Tabela 3 – Componentes mecânicos e relações equivalentes

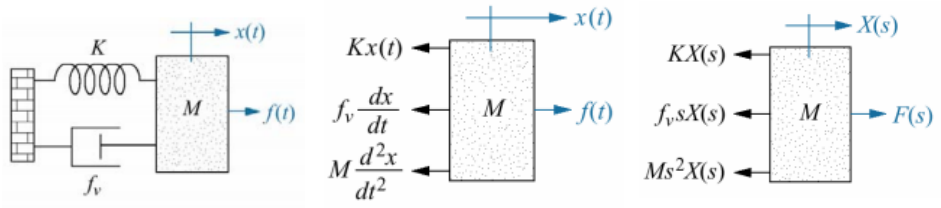
Componente	Força-velocidade	Força-deslocamento	Impedância $Z_m(s) = \frac{F(s)}{X(s)}$
Mola	$f(t) = K\int_0^t v(\tau)d\tau$	$f(t) = Kx(t)$	K
Amortecedor viscoso	$f(t) = f_v v(t)$	$f(t) = f_v \frac{dx(t)}{dt}$	$f_v s$
Massa	$f_(t) = M\frac{dv(t)}{dt}$	$f_(t) = M\frac{d^2x(t)}{dt^2}$	Ms^2

Fonte: (SANTOS, 2008)

Considerando um problema de modelagem mecânica em translação que apresenta mola, amortecimento viscoso e massa (os três casos da tabela) como o da Figura 12 (a), podemos aplicar os passos anteriores para encontrar a função de transferência do modelo. A Figura 12 (b) mostra um diagrama de corpo livre com as forças atuantes no sistema, com as respectivas relações apresentadas pela tabela. Ao aplicar a transformada de Laplace em cada uma delas, tem-se o resultado da Figura 12 (c).

Figura 12 – Sistema com massa, mola e amortecimento viscoso

(a) Problema físico **(b) Diagrama de corpo livre** **(c) Forças em Laplace**



Fonte: (SANTOS, 2008)

Para encontrar a função de transferência então são somadas e igualadas a zero as forças atuantes no sistema. Assim, a Equação (12) apresenta o somatório de forças, e a Equação (13) a transformada de Laplace da equação anterior.

$$M\frac{d^2x(t)}{dt^2} + f_v\frac{dx(t)}{dt} + Kx(t) = f(t) \tag{12}$$

$$Ms^2X(s) + f_v sX(s) + KX(s) = F(S) \tag{13}$$

$$(Ms^2 + f_v s + K)X(s) = F(s) \tag{14}$$

Manipulando a equação anterior, encontra-se a função de transferência final do pro-

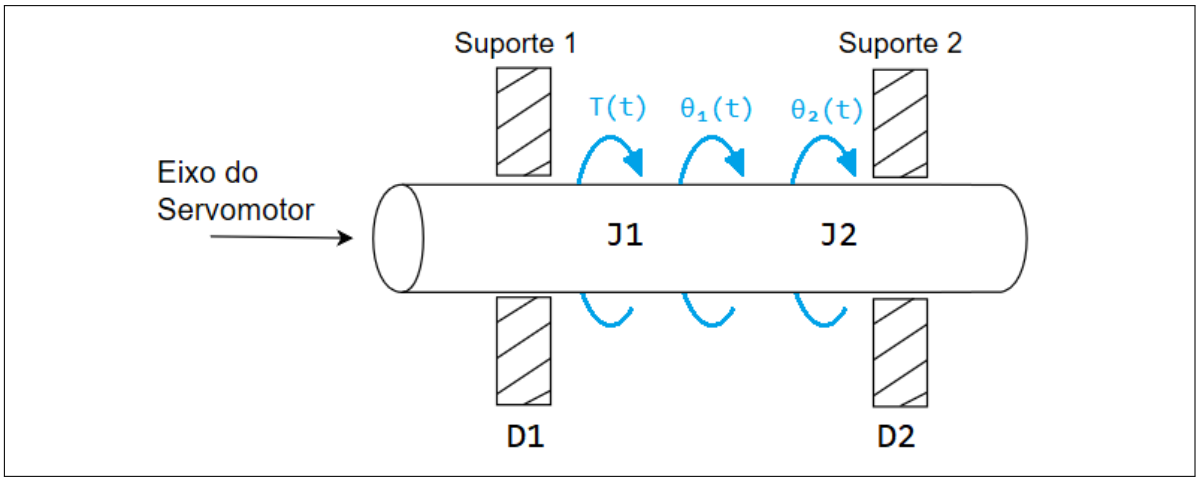
blema, indicada na Equação 15.

$$G(s) = \frac{X(s)}{F(s)} = \frac{1}{Ms^2 + f_v s + K} \tag{15}$$

2.2.2 Modelagem do sistema proposto

Para o protótipo, podemos considerar a modelagem de um eixo em rotação conforme a Figura 13, onde há a aplicação de um torque em uma das extremidades e torções resultantes ao longo do eixo.

Figura 13 – Esquema visual do eixo de rotação



Fonte: (SANTOS, 2008, adaptado)

Cada elemento mecânico presente na movimentação é descrito na Figura 14 com suas devidas fórmulas equivalentes e impedâncias em Laplace, que serão utilizadas para compor os cálculos. Para dar início a modelagem, temos que considerar ambas as torções ao longo do eixo e realizar o somatório de forças sobre as mesmas.

A Figura 15 ilustra as forças atuantes em cada uma das torções, enquanto a Figura 16 separa de forma mais gráfica todas os elementos distribuídos ao longo do eixo em questão.

Ao somar e igualar a zero todas as componentes para cada uma das torções, o sistema de equações para o modelo resulta nas equações abaixo:

$$\left(J_1 s^2 + D_1 s + K\right)\theta_1(s) - K\theta_2(s) = T(s) \tag{16}$$

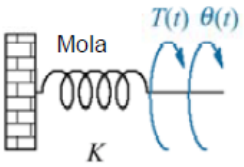
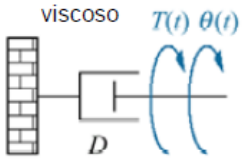
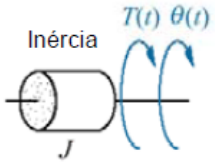
$$-K\theta_1(s) + \left(J_2 s^2 + D_2 s + K\right)\theta_2(s) = 0 \tag{17}$$

Isolando θ_2 nas duas equações temos

$$\theta_2(s) = \frac{(J_1 s^2 + D_1 s + K)\theta_1(s) - T(s)}{K} \tag{18}$$

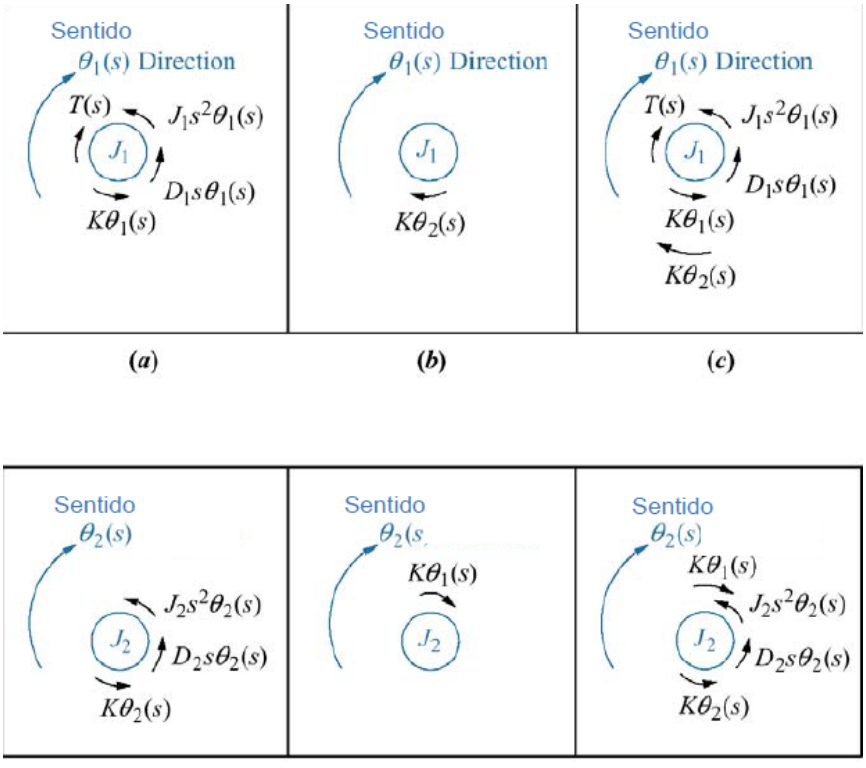
$$\theta_2(s) = \frac{K\theta_1(s)}{(J_2 s^2 + D_2 s + K)} \tag{19}$$

Figura 14 – Componentes mecânicos e equações equivalentes

Componente	Torque - velocidade angular	Torque - deslocamento angular	Impedância $Z_m(s) = T(s) / \theta(s)$
Mola 	$T(t) = K\int_0^t \omega(\tau)\,d\tau$	$T(t) = K\theta(t)$	K
Amortecedor viscoso 	$T(t) = D\omega(t)$	$T(t) = D\frac{d\theta(t)}{dt}$	Ds
Inércia 	$T(t) = J\frac{d\omega(t)}{dt}$	$T(t) = J\frac{d^2\theta(t)}{dt^2}$	Js^2

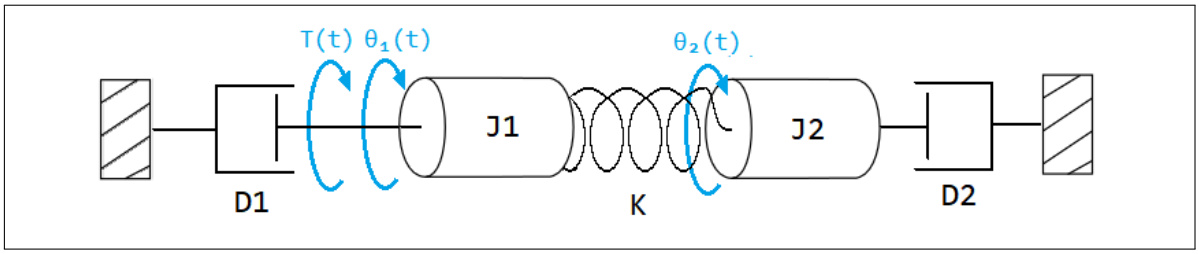
Fonte: (SANTOS, 2008)

Figura 15 – Forças atuantes para cada torção



Fonte: (SANTOS, 2008)

Figura 16 – Diagrama de corpo livre



Fonte: (SANTOS, 2008, adaptado)

Igualando ambas as equações em termos de θ_2

$$\frac{(J_1 s^2 + D_1 s + K)\theta_1(s) - T(s)}{K} = \frac{K\theta_1(s)}{(J_2 s^2 + D_2 s + K)} \tag{20}$$

$$(J_1 s^2 + D_1 s + K)\theta_1(s) = K\left(\frac{K\theta_1(s)}{J_2 s^2 + D_2 s + K}\right) + T(s) \tag{21}$$

$$(J_1 s^2 + D_1 s + K)\theta_1(s) - \frac{K^2\theta_1(s)}{(J_2 s^2 + D_2 s + K)} = T(s) \tag{22}$$

$$\theta_1(s)\left[(J_1 s^2 + D_1 s + K) - \frac{K^2}{(J_2 s^2 + D_2 s + K)}\right] = T(s) \tag{23}$$

$$\theta_1(s)\left[\frac{(J_1 s^2 + D_1 s + K)(J_2 s^2 + D_2 s + K) - K^2}{(J_2 s^2 + D_2 s + K)}\right] = T(s) \tag{24}$$

$$\frac{\theta_1(s)}{T(s)} = \frac{(J_2 s^2 + D_2 s + K)}{(J_1 s^2 + D_1 s + K)(J_2 s^2 + D_2 s + K) - K^2} \tag{25}$$

A Equação (25) caracteriza a função de transferência para o sistema. Realizando a multiplicação do denominador, obtem-se

$$\frac{\theta_1(s)}{T(s)} = \frac{(J_2 s^2 + D_2 s + K)}{J_1 J_2 s^4 + (J_1 D_2 + J_2 D_1)s^3 + (J_1 K + D_1 D_2 + J_2 K)s^2 + (D_1 K + D_2 K)s} \tag{26}$$

Pelo eixo do protótipo ser curto o suficiente para que ocorra algum efeito significativo em sua outra extremidade, pode-se considerar a torção θ_2 como nula. Neste caso, podemos substituir J_2 e D_2 por 0 na Equação (26). Dessa forma, a função de transferência simplificada do sistema é a descrita na Equação (27).

$$\frac{\theta_1(s)}{T(s)} = \frac{1}{J_1 s^2 + D_1 s} \tag{27}$$

2.3 PRINCÍPIO DO ACELERÔMETRO

Muito utilizados nas indústrias em aplicações diversas, os acelerômetros são responsáveis por medir inclinações ou vibrações em partes mecânicas, graças à sua construção. Seu

funcionamento se dá através da leitura de cargas elétricas produzidas através da atuação de forças sobre o material piezoelétrico do sensor (OMEGA, 2018).

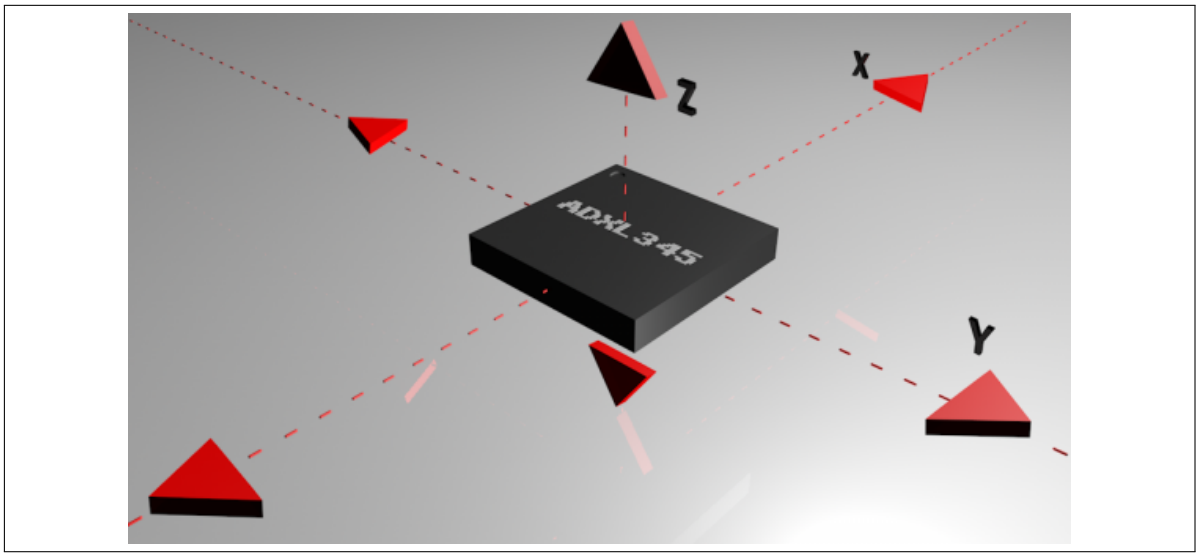
Usualmente são considerados dois tipos principais de acelerômetros: de Alta Impedância e Baixa Impedância. Para o de Alta Impedância, os cristais piezoelétricos geram uma carga elétrica que é transmitida diretamente para os equipamentos de medição, sendo necessárias condições especiais para seu manuseio. Também por serem resistente a altas temperaturas, são geralmente usados em laboratórios, onde há infraestrutura adequada para sua manutenção. Já os modelos de Baixa Impedância transformam a carga elétrica gerada em baixa graças a presença de um microcircuito interno e transistores FET. Por sua simplicidade e versatilidade, este tipo de acelerômetro é comumente utilizado em aplicações industriais (OMEGA, 2018).

A forma de comunicação para aquisição de dados da aceleração é determinada pelo modelo do sensor em questão. Há três formas distintas de leituras:

- Analógico: Fornecem dados através da variação de tensão entre sua alimentação e terra, podendo ser lida através de um conversor Analógico/Digital (A/D).
- Digital: Comunicam-se através dos protocolos I2C ou SPI, produzindo menos ruído que os modelos analógicos.
- PWM: Fornecem dados através de um sinal PWM, com valores de *duty-cycle* proporcionais aos valores de aceleração (SPARKFUN).

Ainda como outro fator para se distinguir sensores acelerômetros, podemos considerar a quantidade de eixos presentes em sua constução: Uniaxial, Biaxial ou Triaxial (Figura 17). A quantidade de eixos determina a quantidade de leituras de aceleração em torno de cada um dos eixos (VIEIRA; STEVAN, 2013). Juntos, é possível determinar a ângulação relativa à componente g e medir inclinações (NAYLAMPMECHATRONICS).

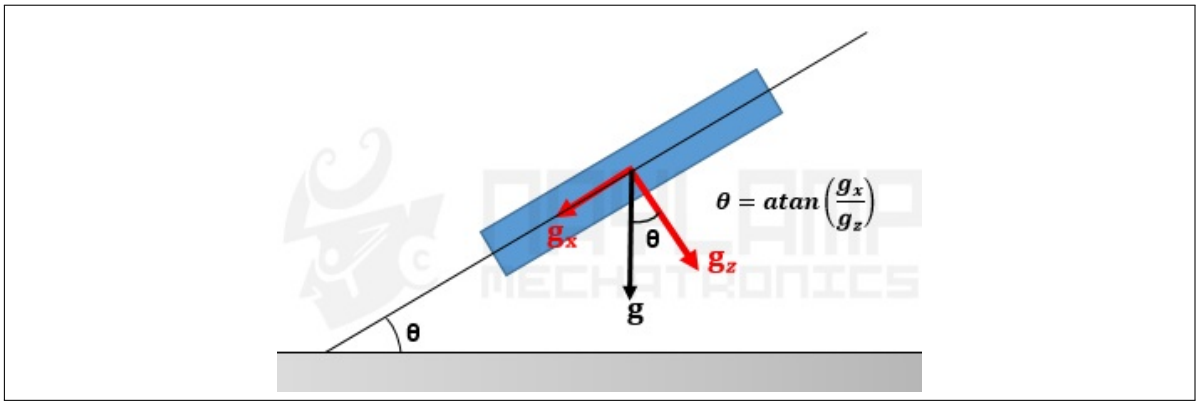
Figura 17 – Modelo de um acelerômetro de 3 eixos



Fonte: (SPARKFUN)

Conforme mostra a Figura 18, através dos três eixos do módulo é possível determinar um ângulo resultante através da Equação (28).

Figura 18 – Inclinação do sensor MPU6050 e vetores



Fonte: (NAYLAMPMECHATRONICS)

$$\theta_y = \tan^{-1}\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \tag{28}$$

2.4 PRINCÍPIO DO SERVOMOTOR

Servomotores são motores que podem se posicionar em determinada posição e mantê-la mesmo ao sofrer uma força que tenda a modificá-la (LEDUC, 2017). Em geral, todo o servomotor é composto de um potenciômetro, circuito de controle, motor DC e engrenagens, que juntos atuam no controle da posição do eixo do mesmo, como mostra a Figura 19.

Figura 19 – Composição interna de um servomotor

Fonte: (LEDUC, 2017)

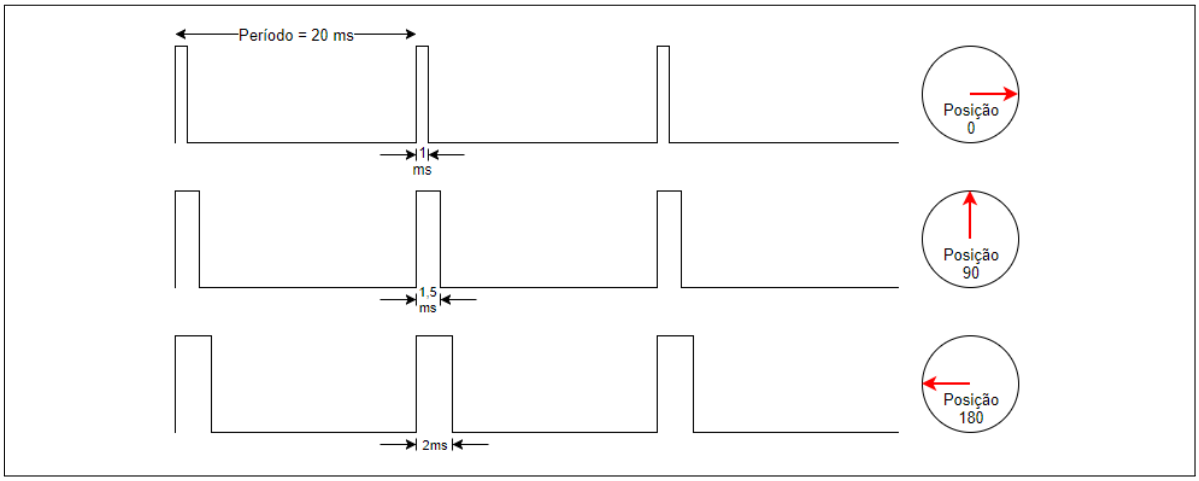
A sequência de engrenagens presente nos servomotores permite que eles possam movimentar cargas bem mais pesadas do que eles próprios. Por essa razão, são muito utilizados em projetos de aeromodelismo para controle do leme e outras partes atuantes na movimentação das asas, por exemplo (PERILLO, 2011).

Os servomotores possuem três fios para conexão: Voltage Common Collector (VCC), Ground (GND) e controle. Usualmente o fio vermelho e preto correspondem à alimentação e terra respectivamente, restando o fio amarelo (ou de cor distinta dos anteriores) como o receptor do sinal de controle de posição. Em geral os servomotores possuem sua movimentação limitada em 180 graus, e podem manter sua posição estática em qualquer valor ângular entre 0 e 180 (PERILLO, 2011).

2.4.1 Controle do servomotor

Para controlar a posição de um servomotor, é necessário aplicar um sinal *Pulse Width Modulation* (PWM) no fio de controle do mesmo. A onda PWM deve possuir frequência de 50Hz (Período de 20ms) e valores de *duty-cycle* (tempo de onda ativa) de acordo com o especificado no *datasheet* do modelo a ser controlado (MICROPIK). Geralmente os valores de *duty-cycle* são adotados entre 1 ms e 2 ms, como mostrado na Figura 20.

Figura 20 – Tempo de *duty-cycle* para diferentes posições do servo



Fonte: (PERILLO, 2011, Adaptado)

Como a figura mostra, para a posição zero do servo é necessário um valor de *duty-cycle* de 1 ms, enquanto para a posição final de 180 graus o valor é de 2 ms. A partir desse intervalo podem-se relacionar quaisquer outros valores de *duty-cycle* para representar os ângulos entre 0 e 180.

2.5 MICROCONTROLADORES PIC

Microcontroladores são componentes que possuem periféricos, memória e processador unidos em um único chip, sendo capazes de realizar cálculos e controlar outros componentes externos através de entradas e saídas conectadas a ele. Muitos eletrodomésticos possuem microcontroladores em seu circuito, estando presentes em *displays* LCD, relés, sensores de temperatura, dentre outros (FUTURE ELECTRONICS).

Os microcontroladores podem ser programados em linguagem C ou *Assembly*, através da mudança de valores binários nas portas de entrada/saída. Para o valor 1 em uma porta de saída, o nível lógico da mesma será alterado para alto, fornecendo uma tensão de 5 V na dada porta. Caso o valor seja 0, o nível lógico será definido como baixo e não haverá tensão de saída.

O registrador PORT, falando especificamente do microcontrolador PIC16F877A, responsável por agrupar as portas de um microcontrolador em um número binário, pode ser reescrito individualmente para cada porta do microcontrolador. Para tanto, a sintaxe do código muda para PORTAbits.RAx, sendo o valor de x o número de bits no qual trabalha o microcontrolador (para oito bits, os números variam de zero a sete). Outro registrador importante para a programação do microcontrolador é o TRIS. Esse comando possui a mesma sintaxe do PORT, porém sua finalidade não é definir níveis lógicos mas sim determinar quais portas serão entradas ou saídas. Para valores binários 1 a porta é uma entrada e caso 0, saída.

Cada modelo de microcontrolador possui características distintas em relação a preços, número de portas e módulos internos, podendo esse último ser o elemento principal para a escolha do melhor microcontrolador para a aplicação desejada. Dentre os mais comuns podemos citar os *Timers* e Interrupções.

Timers são estruturas de contagem de tempo que se repetem continuamente enquanto

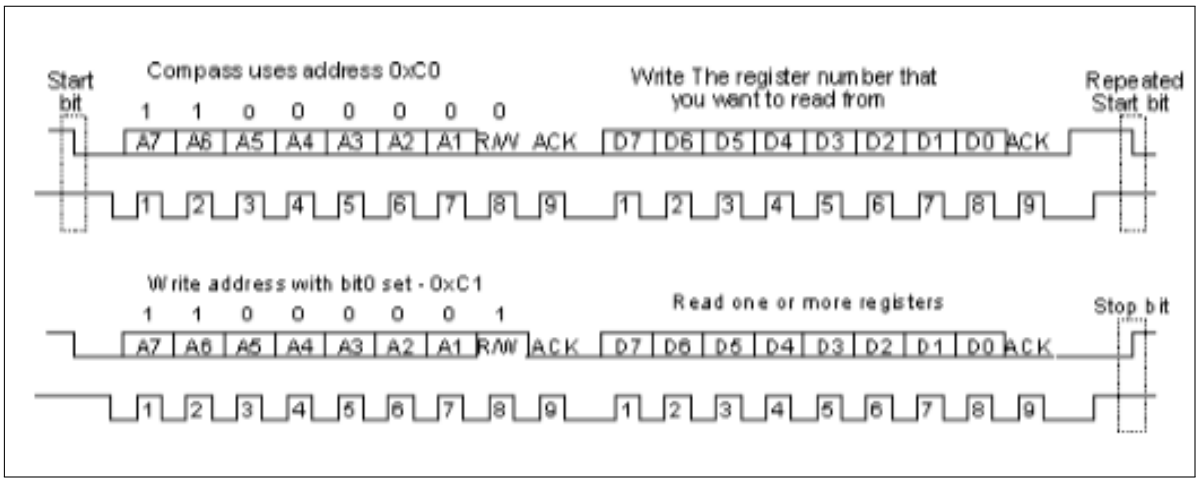
o microcontrolador estiver executando sua programação. São utilizados para medir intervalos de tempo e, se combinados com interrupções, podem gerar ondas ou realizar determinada ação. As interrupções por sua vez são mecanismos que respondem a determinados eventos e são executadas instantaneamente no momento em que ocorrem, pausando a rotina padrão até que sua ação seja tratada. A união desses dois módulos pode criar soluções robustas de tratamento de problemas, agindo como intermediadores de erros e contribuindo para que o sistema realize sua ação com sucesso (MATIC, 2000).

Os microcontroladores são ótimas opções para controlar equipamentos, possuindo versões com recursos diferenciados e preços baixos.

2.6 COMUNICAÇÃO I2C

Criado na década de 90 pela Phillips, o protocolo I2C é um barramento de comunicação serial muito utilizado para realizar a troca de dados entre componentes eletrônicos distintos. Composto por dois canais de comunicação, SDA e SCL, a transferência de dados se dá através do envio e recepção de números binários, transmitidos através da alteração do nível lógico do canal SDA (dados), de forma síncrona com os pulsos emitidos pelo canal SCL (clock) (CAMARA, 2013). A Figura 21 mostra um exemplo de transmissão.

Figura 21 – Representação de uma comunicação I2C

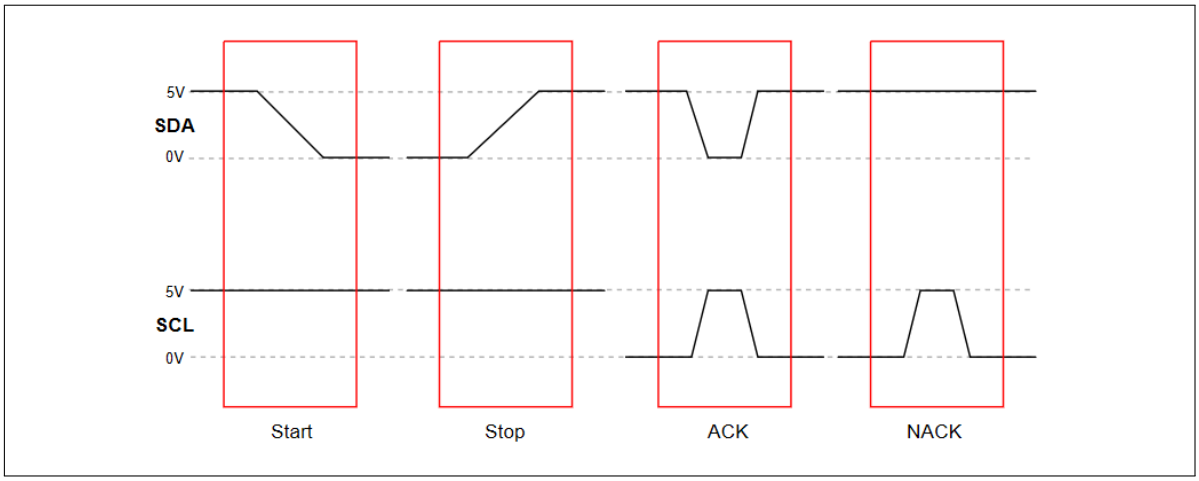


Fonte: (CAMARA, 2013)

Na Figura, é mostrada a sequência de elementos de uma transmissão de dados comum. Inicia-se pelo sinal de *Start* gerado pelo mestre, seguido pela instrução de escrita do endereço do componente escravo (endereço + bit de leitura/escrita). O componente escravo retornará um bit de *Acknowledge* (ACK) para confirmar o recebimento/envio da instrução com sucesso. Na sequência, é enviado o endereço do registrador que se deseja obter os dados + bit de escrita (0), *Restart* da transmissão, endereço do registrador + bit de leitura (1), e por fim é retornado o valor do registrador solicitado. Ao término das leituras, é gerado um sinal de *Stop*, finalizando a transmissão. Em todas as estapas onde são transferidos bytes, sempre haverá um nono bit de ACK em seguida.

A Figura 22 mostra os principais sinais presentes na comunicação I2C:

Figura 22 – Principais sinais presentes em uma transmissão I2C

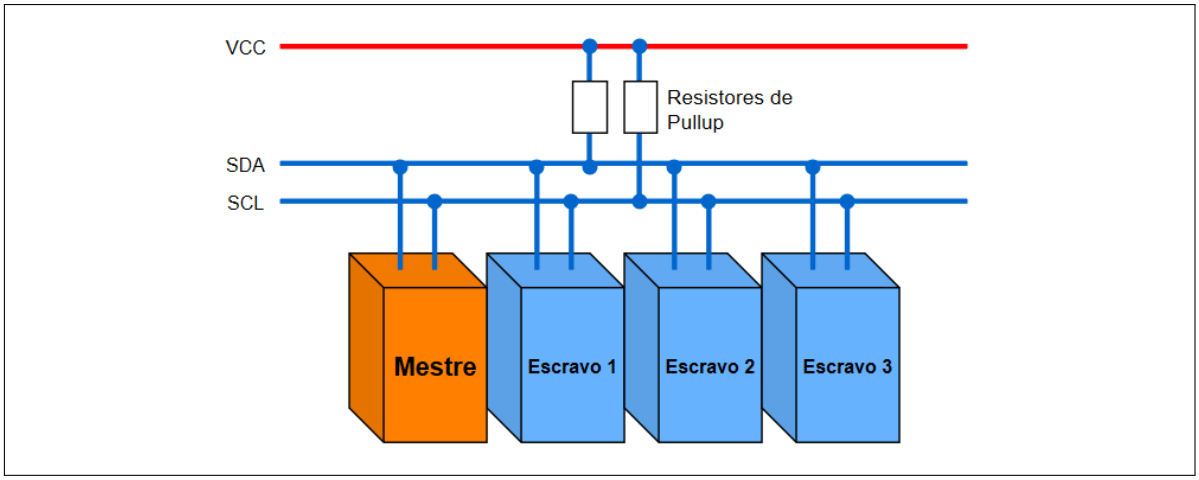


Fonte: Autoria Própria

Em estado de espera, tanto o canal SDA quanto SCL permanecem em estado lógico alto. Para o início da transmissão, o mestre baixa o nível lógico da linha SDA enquanto SCL permanece alta, caracterizando um sinal de *Start*.

A comunicação I2C também permite que vários mestres comandem vários escravos. Na Figura 23 está representada a comunicação entre um mestre e três escravos. Para comutar entre um e outro, basta seguir com a rotina de comunicação utilizando o endereço específico de cada um deles.

Figura 23 – Conexão I2C entre mestre-escravo para 3 escravos



Fonte: Autoria Própria

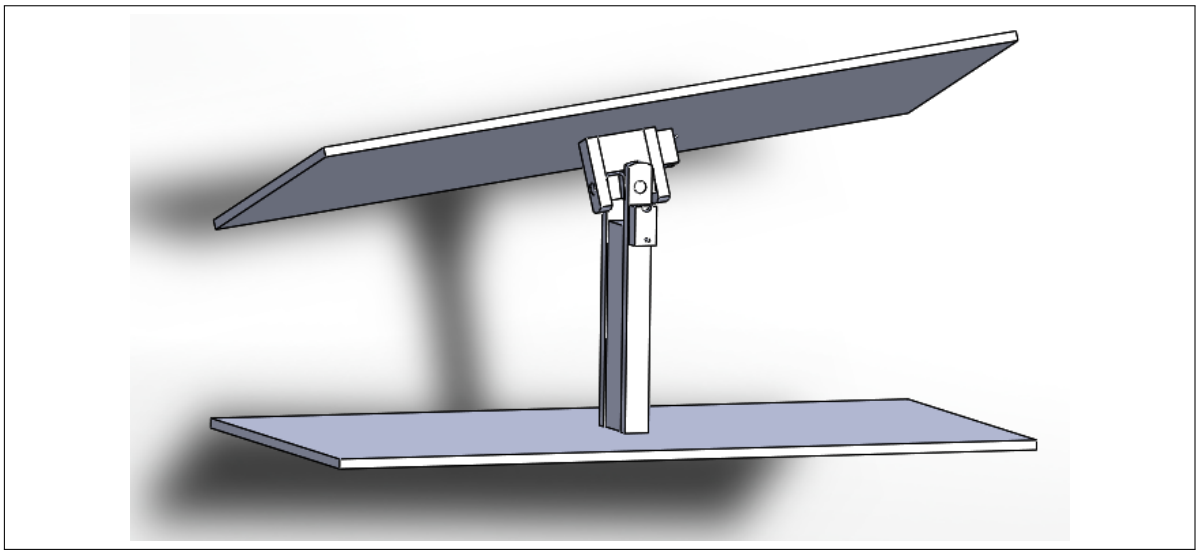
3 MATERIAIS E MÉTODOS

Em geral, o sistema consiste de uma plataforma auto-ajustável que permanecerá estável mesmo após sofrer perturbações. Para que o protótipo seja funcional, são necessários materiais precisos para a análise dos erros e movimentação de correção. Nesta seção cada um dos componentes utilizados serão apresentados, bem como a montagem e funcionamento do sistema como um todo.

3.1 PROTÓTIPO FÍSICO

O projeto proposto foi construído com o auxílio da ferramenta SolidWorks® e consiste de duas placas inicialmente paralelas cuja placa superior deverá permanecer na posição desejada ao passo em que a inferior sofre alterações de posição. O protótipo final foi construído em madeira resistente e leve para que não houvesse estresse sobre os motores.

Figura 24 – Estrutura final do protótipo proposto

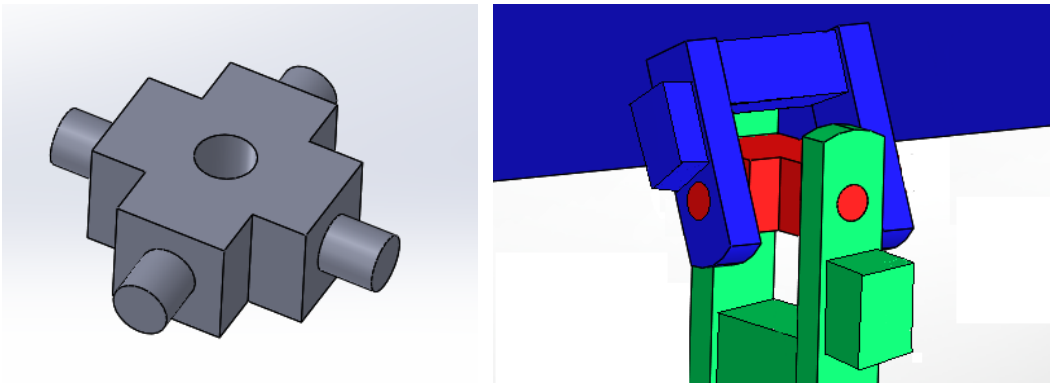


Fonte: Autoria Própria

Na Figura 24, a placa inferior será submetida a variações de posição que serão lidas por dois sensores acoplados a ela. A placa superior deve manter-se em equilíbrio enquanto a placa inferior estiver sofrendo perturbações, corrigindo sua posição a todo momento através da movimentação de servomotores. A peça central contribui para a movimentação e união das placas para que o movimento de uma não influencie o movimento da outra. Na Figura 25 (a) é mostrada a peça de união das placas bem como em (b) sua localização na montagem (no centro). A sequência de etapas para realizar a correção da posição da placa superior é ilustrada na Figura 26. Essas etapas estarão em constante repetição a cada nova perturbação lida pelo sistema.

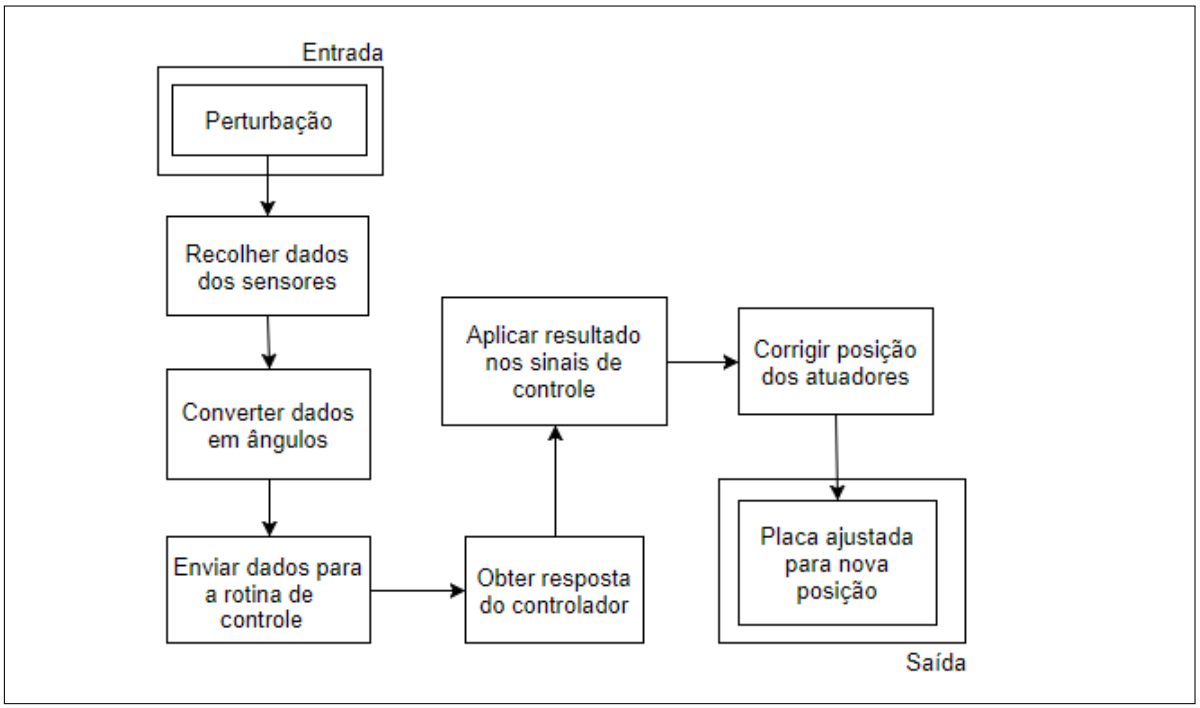
Figura 25 – Peça de união das placas

(a) Peça de junção **(b) Localização no protótipo**



Fonte: Autoria Própria

Figura 26 – Diagrama Funcionamento da correção



Fonte: Autoria Própria

3.2 ITENS E ESPECIFICAÇÕES

3.2.1 Servomotor TowerPro SG90 9g

A movimentação da placa superior é realizada através de dois servomotores acoplados em direções perpendiculares entre si. Cada um dos dois atua em um movimento da placa, sendo que com a ajuda de ambos é possível movimentar livremente em torno do eixo central. Os atuadores que realizam a movimentação da placa suspensa são micro servomecanismos TowerPro SG90 9g, devido a serem modelos de servomotores conhecidos e estarem a disposição

para o projeto. O modelo está ilustrado na Figura 27.

Figura 27 – Microservo TowerPro SG90 9g

Fonte: (MICROPIK)

De acordo com o *datasheet* do componente, a Tabela 4 mostra suas características de funcionamento e operação.

Tabela 4 – Especificações do servomotor SG90

Peso	9 g
Dimensão	22.2 x 11.8 x 31 mm
Torque de parada	1.8 kgf.cm
Velocidade de Operação	0.1 s/60 graus
Tensão de Operação	4.8 V (5V)
Largura de banda morta	$10\mu s$
Faixa de Temperatura	$0\,^{\circ}\mathrm{C}$ - $55\,^{\circ}\mathrm{C}$

Fonte: (MICROPIK)

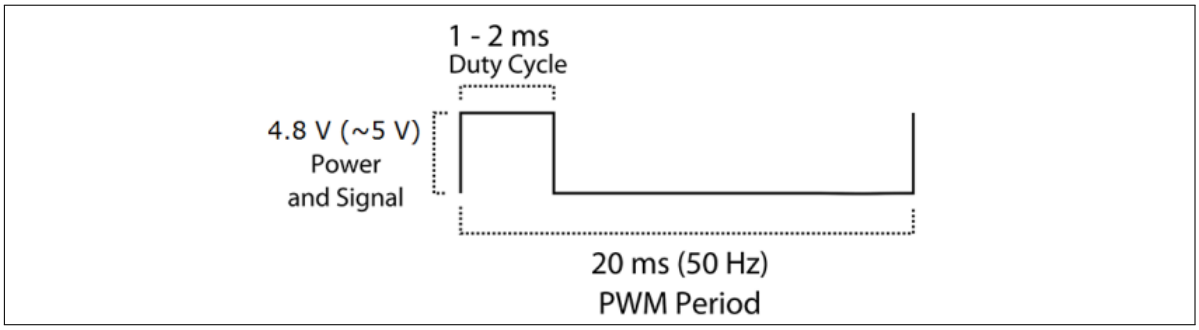
Como explicado na Seção 2, um servomotor altera sua angulação através de uma onda PWM por valores de *duty-cycle*. Em geral, a faixa de valores para o *duty-cycle* de servomotores comuns é apresentada na Figura 28 e varia de 1 para 2 ms, sendo respectivamente o primeiro equivalente a 0 graus e o segundo a 180. Qualquer valor entre 1 e 2 ms tem um equivalente em graus entre 0 e 180 (MICROPIK).

Dessa forma, sabendo o valor de *duty-cycle* correspondente a cada ângulo e relacionando com o valor obtido do acelerômetro, é possível saber o valor de correção necessário para estabilizar a placa superior. Na Figura 29 estão indicadas as posições dos servomotores no protótipo final.

3.2.2 Módulo MPU6050

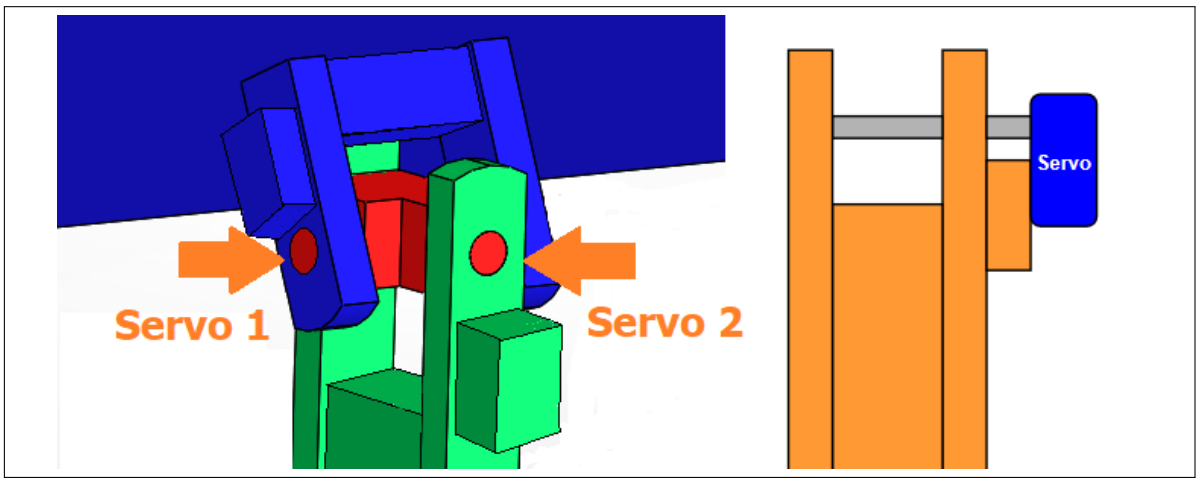
O módulo acelerômetro-giroscópio MPU6050 é o responsável pela medição dos ângulos da placa inferior em relação ao solo, selecionado principalmente por ser um modelo conhecido, de fácil acesso e comunicação. Considerando que qualquer inclinação da placa pode ser indicada como a soma das inclinações nos pontos A e B da Figura 30, serão colocados sensores nestes

Figura 28 – Onda PWM de controle do micro servo TowerPro SG90



Fonte: (MICROPIK)

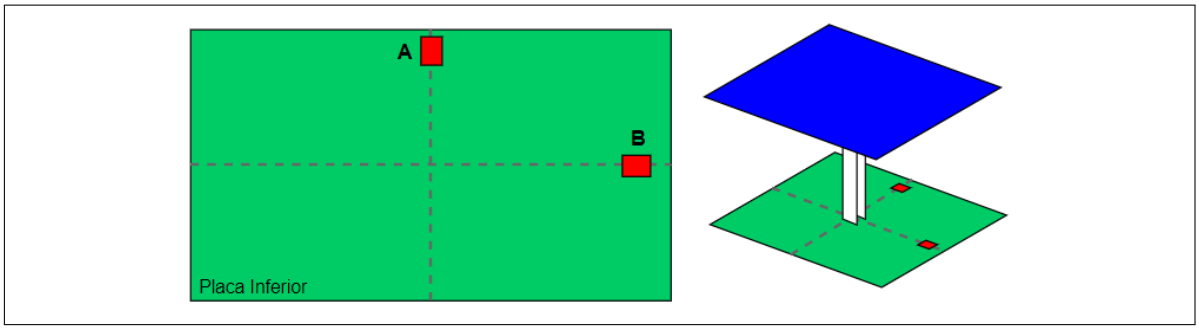
Figura 29 – Localização dos servomotores



Fonte: Própria

devidos pontos para que não ocorra interferência de um para com o outro e seja abrangida toda possível inclinação.

Figura 30 – Localização dos módulos MPU na placa inferior

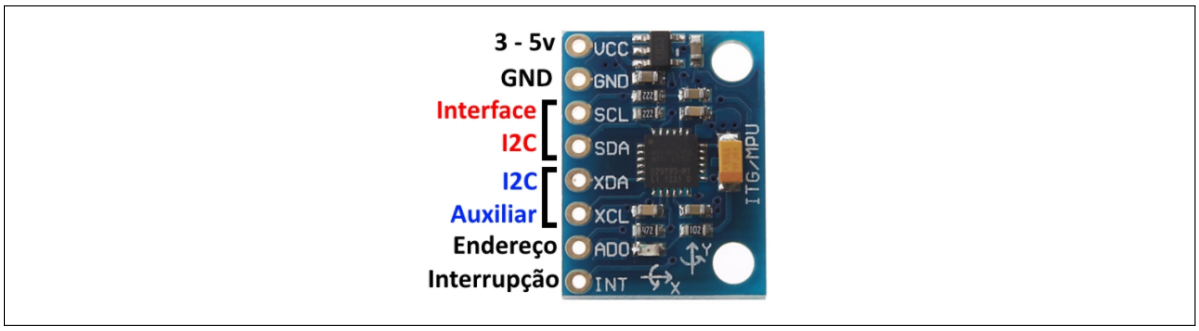


Fonte: Autoria Própria

Este módulo é composto por um giroscópio de 3 eixos, um acelerômetro de 3 eixos e um processador de movimento digital (*Digital Motion Processor - DMP*) além de possuir dois canais de comunicação I2C para conexão com o microcontrolador (INVENSENSE, 2013). A Tabela 5 indica as principais características do módulo. Por si só, o MPU6050 necessita de uma

alimentação de 3V, além de resistores de *pullup* em ambas as linhas SCL e SDA. Felizmente, este módulo é facilmente encontrado junto à placa GY-521, presente na Figura 31, que possui um regulador de tensão e os resistores em sua construção, permitindo que possa ser realizada a conexão direta entre VCC e 5V além de não necessitar de resistores nos canais de comunicação.

Figura 31 – Módulo MPU6050 na placa GY-521



Fonte: (FILIPEFLOP)

Tabela 5 – Especificações do módulo MPU6050

VDD	2.375 V - 3.43 V
VLÓGICA	1.71 V para VDD
Interface Serial Suportada	I2C
Pino 8	*VLOGIC*
Pino 9	*AD0*
Pino 23	*SCL*
Pino 24	*SDA*

Fonte: (INVENSENSE, 2013)

De acordo com a orientação do módulo, é retornado um valor digital correspondente a sua posição nos três eixos do acelerômetro. Quando está totalmente na horizontal, o valor ângular do módulo MPU6050 registrará o valor 0, significando que a placa inferior naquele ponto não está inclinada. Ao haver alterações na inclinação, este valor irá mudar para mais ou menos dependendo da direção da inclinação. Por padrão, a conversão dos valores brutos em ângulos variam de +90 a -90 graus.

Através dos dois módulos é possível obter valores equivalentes para os ângulos de inclinação sem que um módulo influencie diretamente o outro. Assim, cada correção de valores nos servos podem ser tratadas individualmente.

3.2.3 PIC16F877A

O microcontrolador utilizado é o PIC16F877A da MicroChip, mostrado na Figura 32, que é um microcontrolador que contém periféricos que serão úteis para o sistema. Este microcontrolador possui 40 pinos, sendo 33 deles configuráveis como entrada ou saída, 15 interrupções, três timers sendo dois de 8 bits e um de 16 bits, comunicação *Serial Peripheral Interface* (SPI), *Inter*

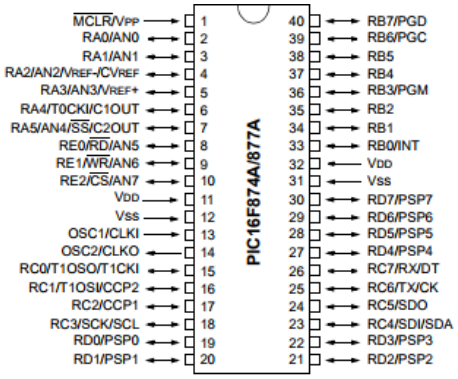
Integrated Circuit (I2C) e *Universal Serial Asynchronous Receiver Transmitter* (USART) além de possuir conversores analógicos e dois módulos *Capture/Compare/PWM* (CCP) (SOUZA, 2007).

A movimentação dos servomotores é realizada através da geração de ondas PWM nas portas de controle dos mesmos, enquanto a comunicação com os módulos MPU6050 é feita via comunicação serial I2C. As interrupções então têm como utilidade a aquisição de dados e correção da posição via algoritmo de controle.

Figura 32 – Microcontrolador PIC16F877A

(a) Componente

(b) Pinagem



Fonte: (MICROCHIP, 2003)

Para implementação de um controle PID temos que levar em conta que a equação PID em sua forma natural exige muito processamento do microcontrolador para a realização dos cálculos integrais e derivativos. Por esse motivo, é necessário simplificar esses cálculos por meio de aproximações (MACIEL, 2012).

A aproximação da derivada pelo método de Euler diz que a primeira derivada de uma função no tempo pode ser aproximada por uma diferença no tempo, mostrada na Equação (29).

$$\frac{de(t)}{dt} \cong \frac{e(kT) - e(kT - T)}{T} \tag{29}$$

Sendo T o período de amostragem, $e(kT)$ o valor do erro no instante de amostragem atual e $e(kT - T)$ o valor do erro no instante anterior ao atual.

A integral pode ser aproximada pelo Método de Tustin através da regra trapezoidal

$$\int_0^{kT} e(t)dt = \int_0^{kT-T} e(t)dt + \int_{kT-T}^{kT} e(t)dt \tag{30}$$

Dessa forma, aproxima-se o primeiro termo no intervalo T e define-se $u(kT)$ como a soma das áreas dos trapézios até o instante kT, resultando na Equação (31).

$$u(kT) = u(kT - T) + \frac{e(kT - T) + e(kT)}{2}T \tag{31}$$

A equação final aproximada torna-se então

$$u(t) = K_p e(t) + I(kT - T) + K_i \frac{e(kT - T) + e(kT)}{2} T + K_d \frac{e(kT) - e(kT - T)}{T} \quad (32)$$

Com $I(kT - T)$ sendo o valor Integral anterior, K_p, K_i e K_d as constantes do PID e T o período de amostragem. Uma rotina PID digital que pode ser aplicada é mostrada no Algorítmo 1 a seguir (BAZANELLA; SILVA, 2000, Adaptado).

Algoritmo 1 – Rotina PID para um microcontrolador

```
1: while (1) do
2:     P = Kp * erro
3:     I = Iant + Ki * ((erroant + erro)/2) * T
4:     D = Kd * ((erro - erroant)/T)
5:     controle = P + I + D
6:     erroant = erro
7:     Iant = I
8: end while
```

3.3 METODOLOGIA

A metodologia aplicada ao desenvolvimento do protótipo consiste em uma série de etapas de pesquisa, testes e montagens. Cada elemento do sistema foi testado individualmente e em conjunto com os outros componentes para compor o sistema final.

3.3.1 Servomotores

Cada servomotor foi testado com o código PWM gerado pelo PIC e verificada sua variação e resposta a valores diferentes de *duty-cycle*. Por serem atuadores que usualmente consomem muita corrente, também foi importante testar a eficiência da fonte em fornecer energia para o sistema com um servo ou dois.

3.3.2 Módulo Acelerômetro

O módulo leitor de inclinação foi testado com o código de controle I2C, verificando a coleta de informações através de leds conectados ao PORTD do microcontrolador. Os registradores de dados do giroscópio e acelerômetro devem variar, enquanto os de temperatura permanecer estáticos. Constatado o teste efetivo, foi realizada a conversão das componentes em ângulo e refeita a análise com os resultados, verificando se a inclinação física do sensor condizia com a leitura digital da angulação.

3.3.3 Microcontroladores

O microcontrolador e sua programação foram testados em *protoboard* com a ajuda de contadores para verificar as taxas de atualização das rotinas e presença de variações indesejadas nas funções. Com o auxílio de um contador e um cronômetro foi também verificado o comportamento do código acompanhando o *blink* de um led medido em segundos. Qualquer desvio foi investigado e corrigido.

3.3.4 Subsistemas

Cada subsistema é composto de um PIC, sensor e atuador, logo, após as etapas anteriores foram testados todos estes 3 componentes em conjunto. Inicialmente os códigos individuais de cada um foram unidos em um só, e as ondas PWM e I2C verificadas com a ajuda de simuladores, como o Proteus. Após esta etapa, os componentes foram conectados fisicamente ao circuito e testada a movimentação do servo de acordo com a inclinação do sensor. Posteriormente o subsistema foi duplicado para o controle do outro grau de liberdade e os testes seguiram para verificação do sistema final.

3.3.5 Sistema final

Nesta etapa foi realizada a verificação do funcionamento de todo o sistema de controle, aplicado também ao protótipo físico desenvolvido. Foram realizadas verificações da funcionalidade de cada um dos graus de liberdade bem como sua integração com a correção de inclinação em conjunto. Feito este último teste, o protótipo ficou pronto para funcionar.

4 DESENVOLVIMENTO E RESULTADOS

4.1 CONTROLE DOS SERVOMOTORES

4.1.1 Análise Preliminar

Para realizar o controle dos servos, o PIC16F877A dispõe de um periférico específico capaz de gerar sinais PWM, o módulo CCPx. Através deste módulo, é possível gerar até dois sinais PWM independentes sem precisar de interrupções ou *delays*, que por vezes acabam impedindo a execução do código.

Como visto nas seções anteriores, os servomotores são controlados através de ondas PWM de frequência 50Hz (ou período de 20ms), cujo valor de *duty-cycle* varia entre 1ms e 2ms. Da mesma forma, o módulo CCPx em modo PWM utiliza o *Timer* 2 para contagem de tempo, de maneira que tanto o período quanto o valor de *duty-cycle* são calculados, respectivamente, seguindo as equações abaixo.

$$P = [(PR2) + 1] * 4 * T_{OSC} * PS \tag{33}$$

$$dc_{PWM} = (CCPRxL : CCPxCON < 5 : 4 >) * T_{OSC} * PS \tag{34}$$

Sendo:

- P o período do sinal PWM;
- $PR2$ um valor entre 0 e 255 para o estouro do *timer*;
- T_{OSC} o tempo de oscilação dado por $1/Freq_{cristal}$;
- PS o valor de *prescaler* do *Timer* 2 (1, 4 ou 16);
- CCPRxL:CCPxCON<5:4> um valor de 10 bits para configurar o *duty cycle* do sinal, sendo os 8 bits mais significativos alocados em CCPRxL e os dois restantes nos bits 5 e 4 do registrador CCPxCON.

Com as fórmulas acima, considerando valores distintos de cristais osciladores e todos os *prescalers* disponíveis, mantendo o valor de PR2 como máximo inicialmente (255), temos a Tabela 6 com os resultados de cada análise.

Observa-se que não há como gerar, exclusivamente pelo módulo CCPx, um sinal PWM de 50Hz com os cristais de 20MHz e 4Mhz, visto que suas menores frequências são de 1220,703Hz e 244,141Hz, respectivamente. Isso se dá devido ao *Timer* 2 possuir apenas 8 bits (PR2 deveria ser 1250 para o cristal de 4MHz funcionar). Porém ao diminuir a frequência do cristal para 800KHz, com uma pequena alteração no valor de PR2, pode-se chegar à frequência exata necessária para o controle.

$$P = [(255) + 1] * 4 * \frac{1}{800000} * 16 = 20,48ms \tag{35}$$

$$P = [(249) + 1] * 4 * \frac{1}{800000} * 16 = 20ms \tag{36}$$

Tabela 6 – Frequências de Onda PWM para PIC16F877A

Frequências de Onda PWM para PIC16F877A			
Cristal 20MHz			
Prescaler	1	4	16
Frequência Obtida (Hz)	19531,250	4882,813	1220,703
Período em (ms)	0,051	0,205	0,819
Cristal 4MHz			
Prescaler	1	4	16
Frequência Obtida (Hz)	3906,250	976,563	244,141
Período em (ms)	0,256	1,024	4,096
Cristal 800KHz			
Prescale	1	4	16
Frequência Obtida (Hz)	781,250	195,313	48,828
Período em (ms)	1,280	5,120	20,480
Cristal 400KHz			
Prescaler	1	4	16
Frequência Obtida (Hz)	390,625	97,656	24,414
Período em (ms)	2,560	10,240	40,960

Fonte: Autoria Própria

Apesar de garantir o período correto para o controle, esse método de geração PWM não consegue fornecer uma precisão de qualidade na angulação do servomotor. Seguindo a Equação (34) para o *duty-cycle*, encontramos os valores de (CCPRxL:CCPxCON<5:4>) que correspondem a 100% (0,02s ou 20ms), 10% (2ms) e 5% (1ms) para análise da precisão.

$$dc_{PWM} = 0,02s = (CCPRxL : CCPxCON < 5 : 4 >) * \frac{1}{800000} * 16 \tag{37}$$

$$(CCPRxL : CCPxCON < 5 : 4 >)_{100\%} = \frac{0,02}{\frac{1}{800000} * 16} = 1000 \tag{38}$$

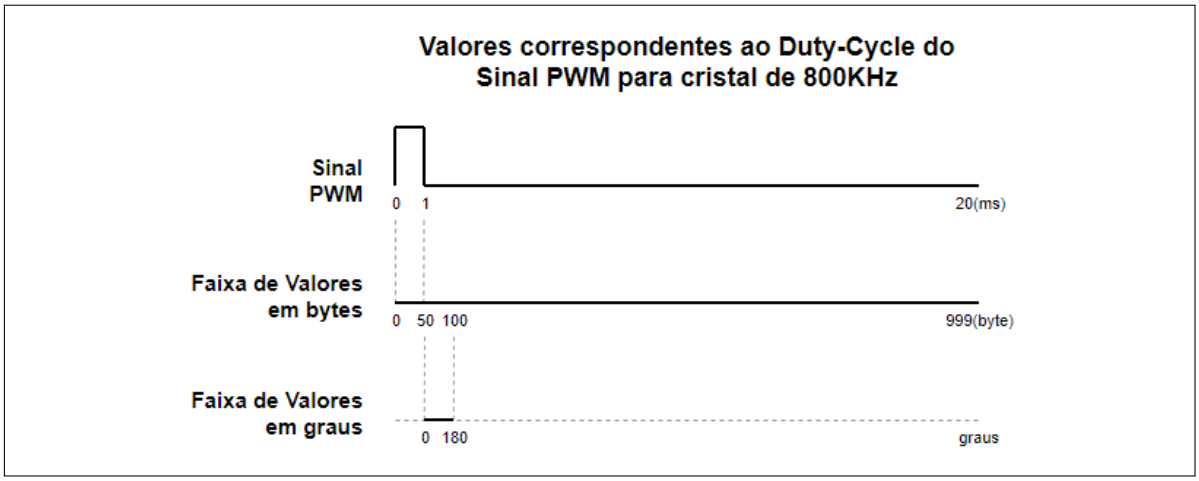
$$(CCPRxL : CCPxCON < 5 : 4 >)_{10\%} = \frac{0,002}{\frac{1}{800000} * 16} = 100 \tag{39}$$

$$(CCPRxL : CCPxCON < 5 : 4 >)_{5\%} = \frac{0,001}{\frac{1}{800000} * 16} = 50 \tag{40}$$

Para melhor visualização, os dados foram comparados e estão presentes na Figura 33.

Como visto na imagem, tem-se uma grande variação no movimento do servo para cada incremento realizado via software. No caso do cristal de 800KHz o controle, que se dá entre um DC de 1ms e 2ms, corresponde a valores de CCPRxL:CCPxCON<5:4> de 50 a 100. Dividindo-se então 180 graus por 50 bytes, têm-se 3,6 graus/byte. Isso significa que ao incrementar apenas uma unidade nos registradores de *duty-cycle*, o servo se moverá 3,6 graus. Com o cristal de 400KHz essa imprecisão aumenta para 7,2 graus/byte, o que acaba impossibilitando o uso do

Figura 33 – Valores de controle do Servomotor e correspondentes



Fonte: Autoria Própria

mesmo. Outro problema encontrado é a disponibilidade destes cristais de baixa velocidade no mercado. Enquanto o de 400KHz (cerâmico) foi encontrado apenas em um site de vendas internacional, o de 800KHz não foi encontrado.

Com tudo isso observa-se que o módulo CCPx cumpre bem seu papel de geração PWM, contudo, para sinais específicos para servomotores, não há uma resolução adequada que compreenda os 180 graus de movimentação. Dessa forma, para melhorar a qualidade da movimentação do servomotor, é necessário buscar alternativas que garantam a melhor precisão possível.

4.1.2 Special Event Trigger

A geração PWM via módulo CCPx pode não conseguir gerar uma onda para controle de um servo, mas ainda há uma maneira utilizando-se do modo *Capture/Compare* do mesmo. O PIC16F877A, em particular, possui uma função extra chamada *"Special Event Trigger"* (Figura 34) que, em modo *Compare*, gera um sinal de hardware pelo CCP1 e reseta os valores dos registradores do *Timer* 1. Dessa forma, pode-se utilizar um cristal de 4MHz juntamente com o *Timer* 1, que possui 16 bits, para controlar o período e *duty-cycle* de um sinal PWM.

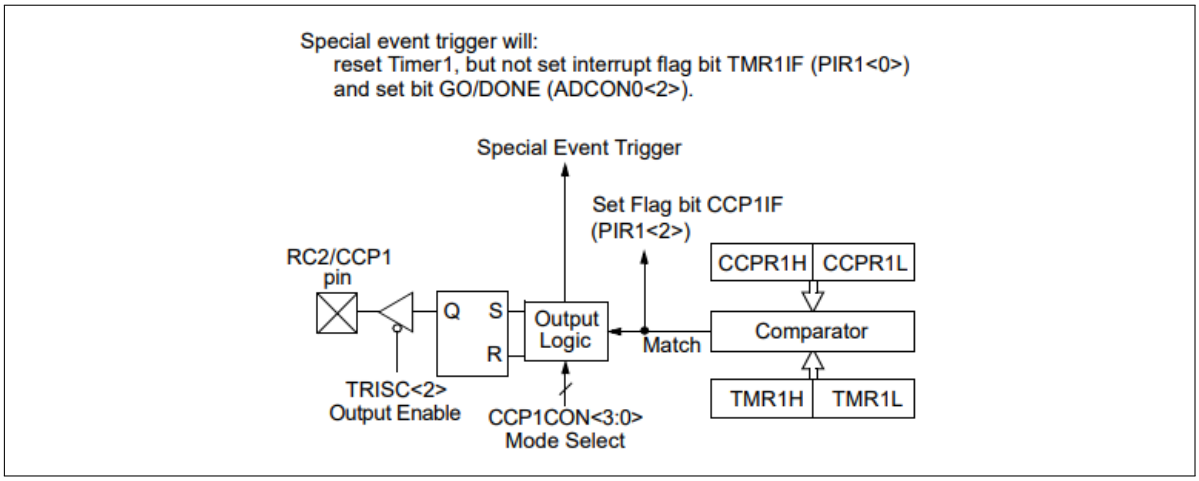
O modo *Compare* então permite a seleção de valores fixos para os registradores CCPR1H e CCPR1L que serão comparados com a contagem de tempo do TIMER1. Logo que os valores forem iguais será gerada uma interrupção e a contagem de tempo reiniciará. Para calcular o valor correto de estouro do *timer* em um período de 20ms é utilizada a Equação (41).

$$T = \frac{4}{F_{osc}} * Prescaler * (65536 - TMR1H : TMR1L) \tag{41}$$

$$0,02 = \frac{4}{4000000} * 8 * (65536 - TMR1H : TMR1L) \tag{42}$$

$$0,02 = 0,000001 * 8 * (65536 - TMR1H : TMR1L) \tag{43}$$

Figura 34 – Estrutura de funcionamento do Special Event Trigger



Fonte: (MICROCHIP, 2003)

$$\frac{0,02}{0,000008} = 65536 - TMR1H : TMR1L \tag{44}$$

$$TMR1H : TMR1L = 65536 - \frac{0,02}{0,000008} = 63036 \tag{45}$$

Pela Equação (45) temos que o valor do *Timer* 1 para a construção do período de 20ms é de 63036, o que também nos informa que 20ms equivale a um valor total de 2500.

No protótipo, o valor de CCPR1H e CCPR1L é ajustável de acordo com o valor de *duty-cycle* definido, porém a interrupção trata o período de onda como sempre sendo 20ms, realizando as comutações de nível lógico e contando o tempo remanescente para completar todo o período da onda.

Exemplificando, considerando que o *Special Event Trigger* fucionará como um gerador de onda PWM, é necessário especificar um valor de *duty-cycle* para o tempo de nível lógico alto do sinal. Como foi calculado que o valor via software para uma onda de período 20ms equivale a 2500, e supondo que desejamos um valor de *duty-cycle* de 1ms (equivalente a 125), a interrupção deve fazer com que o valor do *duty-cycle* préviamente definido seja descontado do valor do período total, 2500, e que entre um valor e outro o nível lógico da porta do microcontrolador seja invertido. Considerando isso, a interrupção será chamada duas vezes ao longo de 20ms. O Algoritmo 2 ilustra a rotina de geração PWM inserido na interrupção.

Como é possível observar, são realizados testes para saber em qual situação se encontra o valor de *duty-cycle* escolhido e, caso se enquadre em algum dos casos, o nível lógico da porta OUT (RC2 ou CCP1) será invertido. Uma variável CCPR armazenará a diferença entre o valor atual e valor total do período, e adicionará aos registradores CCPR1H e CCPR1L a nova contagem necessária para serem completados os 20ms. Dessa forma, a onda PWM pode ser gerada e mantida estável, o que é primordial para um bom controle de um servomotor.

Algoritmo 2 – Rotina para geração de sinal PWM via Special Event Trigger

```
if ((c_periodo > 0) && (c_periodo < t_periodo)) { // Se o periodo e >
  ↪ 0% e < 100%:

if (OUT == 1) {                                   // Caso a saida seja
  ↪ 1 -> estava em alta.
OUT = 0;                                          // Saida agora e 0 p/
  ↪ gerar o tempo de baixa.
CCPR = t_periodo - c_periodo;                     // periodo total -
  ↪ periodo atual => baixa = total - alta.
}
else {                                            // Se saida era 0 ->
  ↪ estava em baixa.
OUT = 1;                                          // Muda saida para 1 para
  ↪ contabilizar o tempo em alta
CCPR = c_periodo;                                 // CCPR armazena periodo
  ↪ de alta p/ contagem.
}
} else {                                              //Se o periodo
  ↪ atual nao esta entre 0ms e 20ms entao:
if (c_periodo == t_periodo) {//
OUT = 1;
}                                                 // se duty = 100%,
  ↪ entao OUT e 1 todo tempo.
if (c_periodo == 0) {
OUT = 0;
}                                                 // se duty = 0%,
  ↪ entao OUT e 0 todo tempo.
}
```

4.2 MPU6050 E ACELERÔMETRO

4.2.1 Estrutura de funcionamento

O módulo MPU6050 utilizado para leitura da inclinação do protótipo é composto por um giroscópio e acelerômetro de 3 eixos cada. Para leitura dos ângulos foi utilizado o acelerômetro, com seus eixos convertidos em relação ao eixo y para leitura.

A fórmula pra converter os valores brutos dos eixos do acelerômetro se dá pela Equação (46), bastando apenas adicionar a biblioteca math.h para validar as operações matemáticas da mesma.

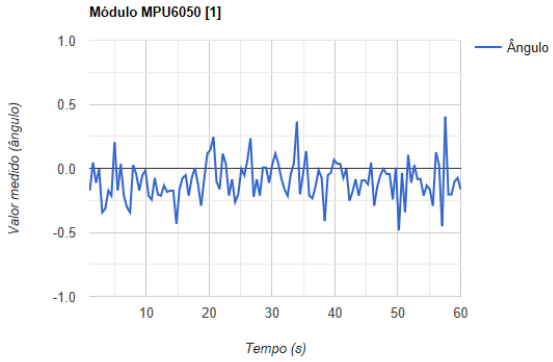
$$\text{angulo} = \text{atan}(\text{AY}/\text{sqrt}(\text{pow}(\text{AX}, 2) + \text{pow}(\text{AZ}, 2))) * (180.0/3.14); \tag{46}$$

Inicialmente, para teste da leitura do sensor, foi utilizado um Arduino principalmente pela vantagem de visualização dos resultados em tempo real, graças ao monitor serial nativo do mesmo. Utilizando os mesmos valores obtidos pelo sensor juntamente com a fórmula acima, foi possível plotar os gráficos mostrados na Figura 35 para análise da sensibilidade do sensor.

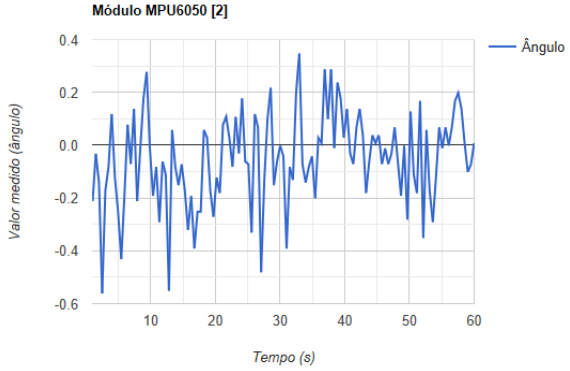
Em ambos foram realizados testes ao longo de 60 segundos, com 2 amostras de ângulos por segundo, considerando a posição horizontal dos sensores.

Figura 35 – Gráfico de valores calculados para os módulos MPU6050

(a) Sensor MPU6050 1



(b) Sensor MPU6050 2



Fonte: Autoria Própria

No PIC foram realizados testes utilizando leds nas portas RD1, RD3, RD5 e RD7, testando inicialmente as componentes *high* e *low* dos eixos do acelerômetro. Feito isso, a fórmula de conversão do ângulo foi implementada, onde foi constatado um tempo de cálculo de aproximadamente 82ms, e cada led ficou responsável por uma faixa angular. O led RD1 acenderia caso o ângulo estivesse abaixo de 0 graus, RD2 na faixa de 25 a 35 e assim sucessivamente conforme mostra o Algoritmo 3.

Os resultados desses testes são mostrados na Figura 36. Adicional ao teste dos ângulos, também foi testada a movimentação dos servomotores conforme a leitura do módulo MPU. Foi através desse teste que foi constatado que os servos do protótipo funcionam com uma faixa de controle entre 0,6ms e 2,4ms.

4.2.2 Comunicação I2C

A comunicação entre o sensor de inclinação e o microcontrolador se dá através do protocolo I2C (*Inter-Integrated Circuit*). O módulo MPU6050 possui instruções para leitura e escrita conforme a necessidade de obtenção de dados. Como descrito na Seção 2.6, é necessário realizar uma rotina de configuração e escrita antes de qualquer leitura. Pelas informações presentes no *datasheet* do módulo, a escrita de um único registrador pode ser realizada seguidas as etapas da Figura 37.

O mestre, neste caso o microcontrolador, enviará os sinais de início da transmissão (S), endereço do componente escravo junto com o bit para escrita (AD+W), endereço do registrador (RA) e, por fim, o valor a ser escrito no registrador (DATA). Simulando o código no Proteus para uma comunicação com um componente genérico, o resultado do osciloscópio é mostrado na Figura 38.

Na imagem, é escrito no registrador `PWR_MGMT_1` o valor binário 00000000, desativando o modo sleep do MPU6050 para iniciar o processo de leitura. Da mesma forma que a escrita, o processo de leitura é realizado com o envio do endereço do sensor escravo, porém

Algoritmo 3 – Algoritmo para testes da angulação utilizando Leds

```
if (angulo<0){
PORTDbits.RD1 = 1;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 0;
PORTDbits.RD7 = 0;
}
if ((angulo>=25)&&(angulo<35)){
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 1;
PORTDbits.RD5 = 0;
PORTDbits.RD7 = 0;
}
if ((angulo>=55)&&(angulo<65)){
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 1;
PORTDbits.RD7 = 0;
}
if ((angulo>=80)&&(angulo<=90)){
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 0;
PORTDbits.RD7 = 1;
}
```

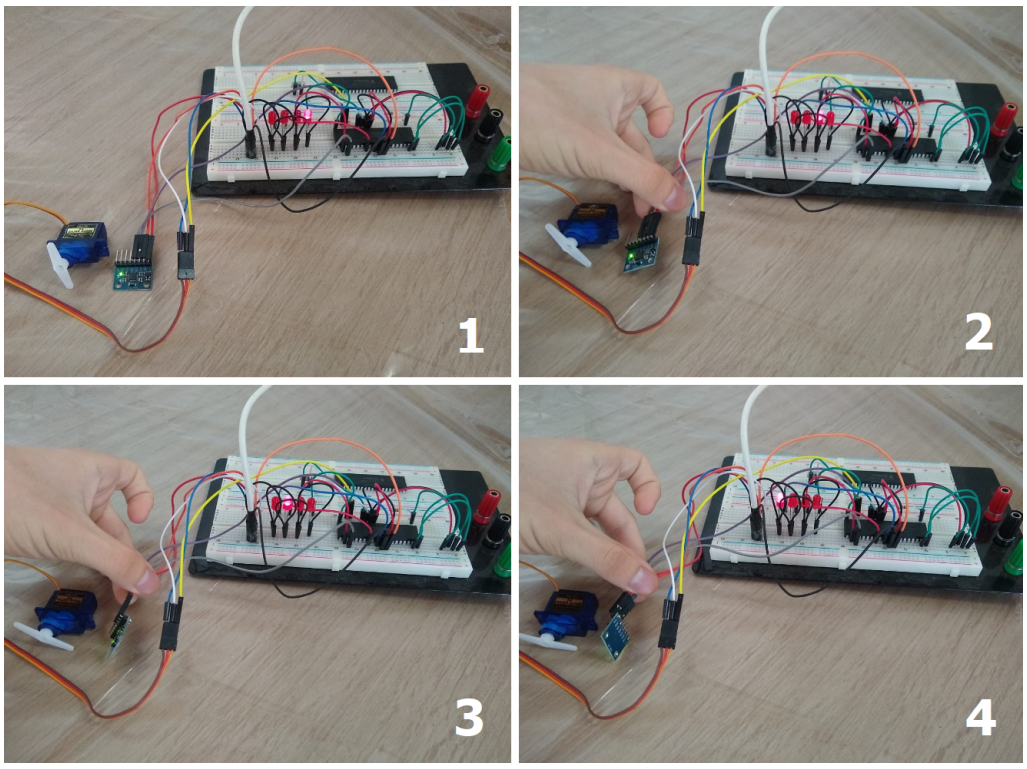
desta vez será requisitada uma sequência de registradores e não apenas um, como na escrita anterior. A sequência de etapas é descrita pelo *Datasheet* do módulo como mostra a Figura 39.

A simulação da aplicação do código para a leitura dos registradores é mostrada nas Figuras 40 e 41.

Para cálculo do ângulo, é necessário obter os valores de *roll*, *pitch* e *yaw* do acelerômetro, que são armazenados em seis registradores de oito bits no módulo. Para leitura então é enviado o endereço do primeiro registrador que inicia as leituras, seguido pelo bit de leitura (1). Em seguida os valores são armazenados em variáveis para serem trabalhados mais tarde.

Para cada *byte* transmitido via comunicação I2C, é realizado um teste de ACK/NACK. Caso haja algum problema durante a transmissão de dados, o escravo retornará um sinal de *Not Acknolegment* (NACK). Como observado nas simulações, há a presença de NACKs após o envio dos endereços. Isso se dá devido ao ambiente de simulação do Proteus não ter o componente em sua biblioteca. Dessa forma, foi utilizado apenas o *I2C Debbugger* para verificação dos sinais. Por consequência disso, o microcontrolador solicita o endereço e não obtém resposta, resultando em um NACK.

Figura 36 – Teste de funcionamento do módulo MPU6050 aplicado ao controle de um servo



Fonte: Autoria Própria

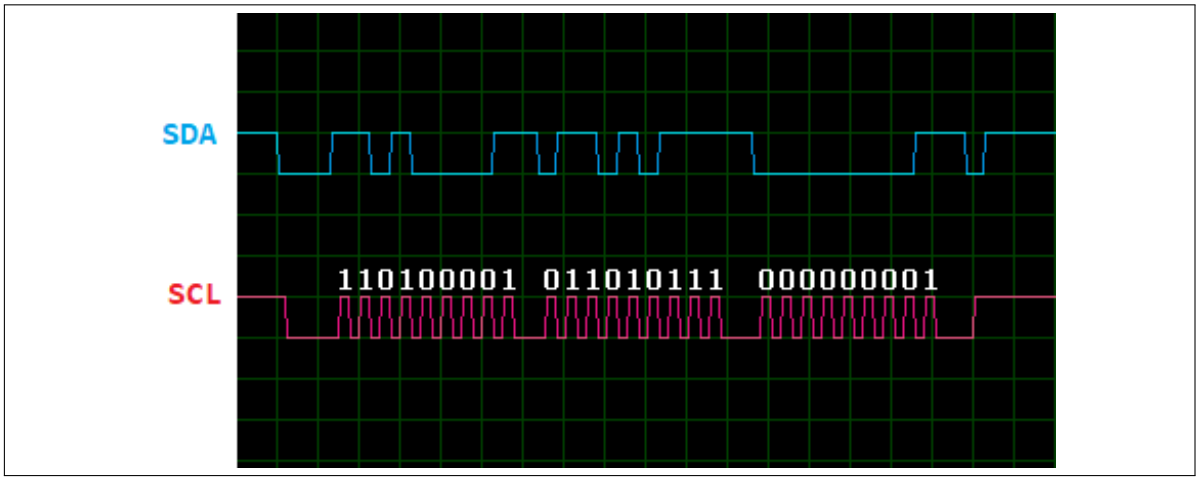
Figura 37 – Rotina I2C de escrita para o MPU6050

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Fonte: (INVENSENSE, 2013)

Figura 38 – Simulação do código para comunicação I2C com o sensor



Fonte: Autoria Própria

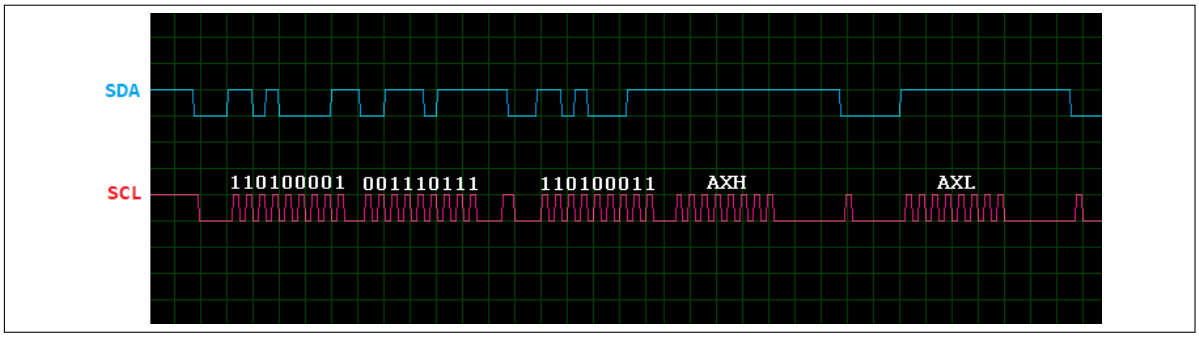
Figura 39 – Rotina I2C de leitura sequencial para o MPU6050

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

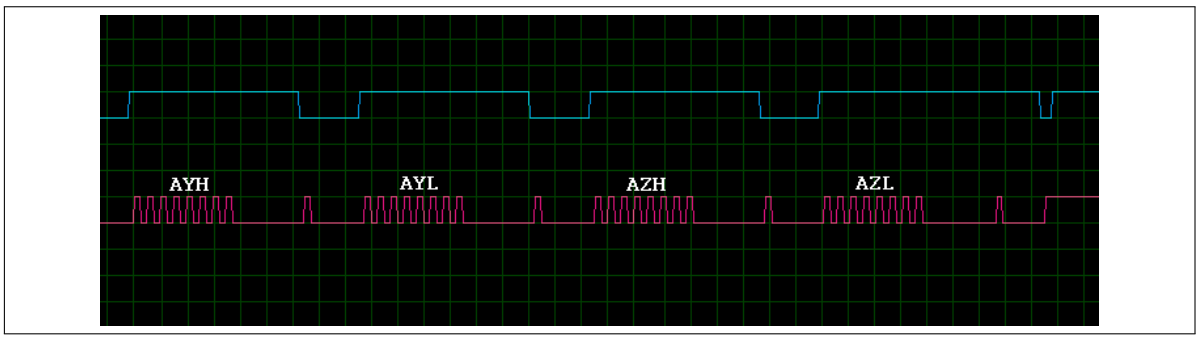
Fonte: (INVENSENSE, 2013)

Figura 40 – Simulação do código de leitura de valores do acelerômetro via I2C



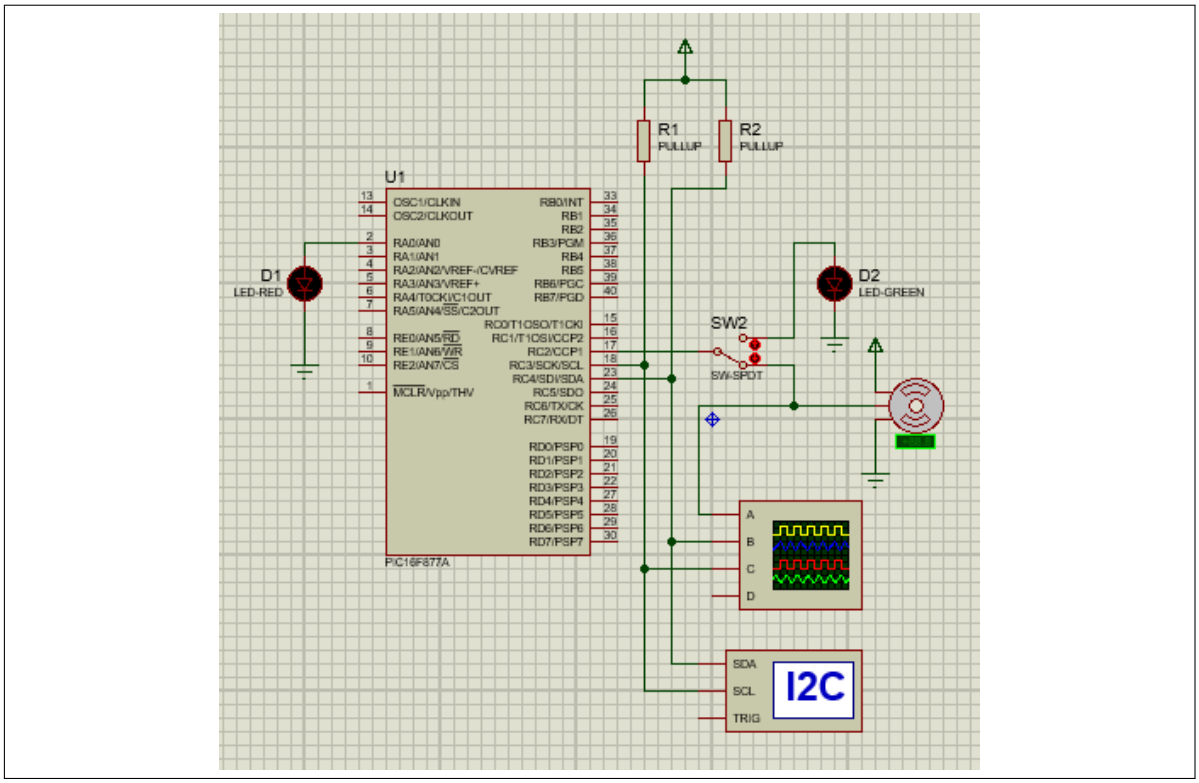
Fonte: Autoria Própria

Figura 41 – Simulação do código de leitura de valores do acelerômetro via I2C



Fonte: Autoria Própria

Figura 42 – Circuito montado para simulação de comunicação I2C no Proteus



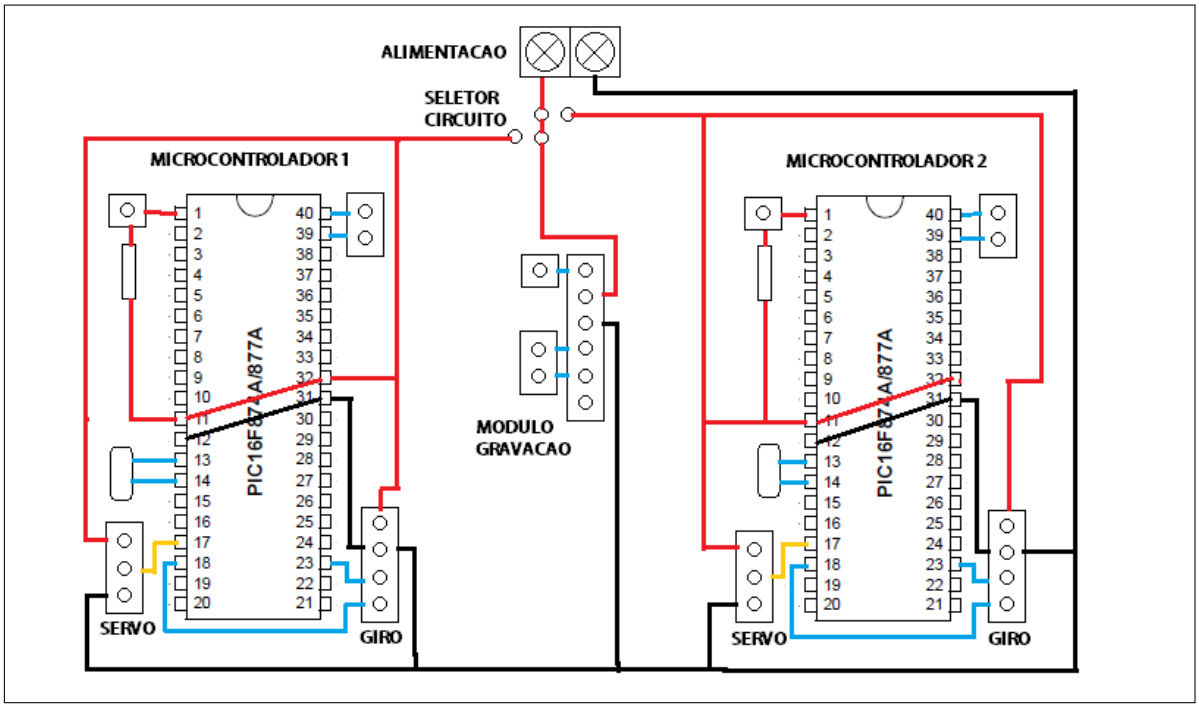
Fonte: Autoria Própria

4.3 ESQUEMA ELÉTRICO

4.3.1 Circuito elétrico

O circuito de montagem para o protótipo consiste, principalmente, de dois microcontroladores, dois servomotores e dois módulos MPU6050 com acelerômetro para medição de angulação. Cada conjunto de módulo, servomotor e microcontrolador é responsável por gerenciar o controle em um dos dois graus de liberdade, atuando de forma independente entre si, porém contribuindo para a correção juntos. A Figura 43 representa a montagem de forma estruturada do circuito, indicando a localização aproximada de cada conexão. Devido a necessitar de localizações específicas e distantes, os servomotores e módulos MPU são conectados também por *jumpers*.

Figura 43 – Circuito de montagem



Fonte: Autoria Própria

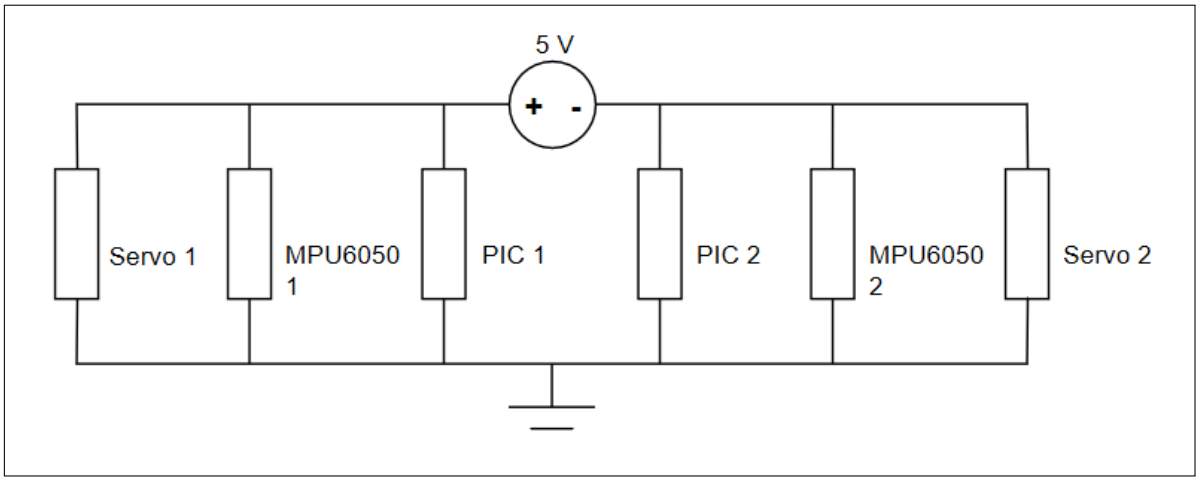
Para o funcionamento do protótipo foram realizados estudos de consumo de energia e desenvolvido o esquema elétrico conforme mostra a Figura 44. O circuito funciona com tensão de alimentação de 5V e corrente total drenada máxima de 3A.

Foram realizados testes e verificação de consumo individualmente para cada atuador, sendo os dados mostrados na Tabela 7.

4.3.2 Diagrama de momentos

Os servos TowerPro SG90 utilizados possuem torque de 1,2Kgf.cm, o que significa que cada servo é capaz de rotacionar cargas de até 1,2Kg a uma distância de 1cm de seu eixo. Neste protótipo a distância da placa superior até o eixo de ambos os servos equivale a 6cm. Dessa

Figura 44 – Circuito Elétrico



Fonte: Autoria Própria

Tabela 7 – Correntes medidas durante o funcionamento do sistema

Correntes drenadas pelos elementos do circuito	
Servomotor 1	800mA (pico)
Servomotor 2	1200mA (pico)
MPU6050 1	5,24mA
MPU6050 2	5,17mA

Fonte: Autoria Própria

forma, encontra-se o valor máximo, em Kg, que o sistema é capaz de suportar pela equação abaixo.

$$\text{Momento} = \text{Força} * \text{deslocamento} \tag{47}$$

$$1,2Kgf.cm = x * 6cm \tag{48}$$

$$x = \frac{1,2Kgf.cm}{6cm} = 0,200Kg \tag{49}$$

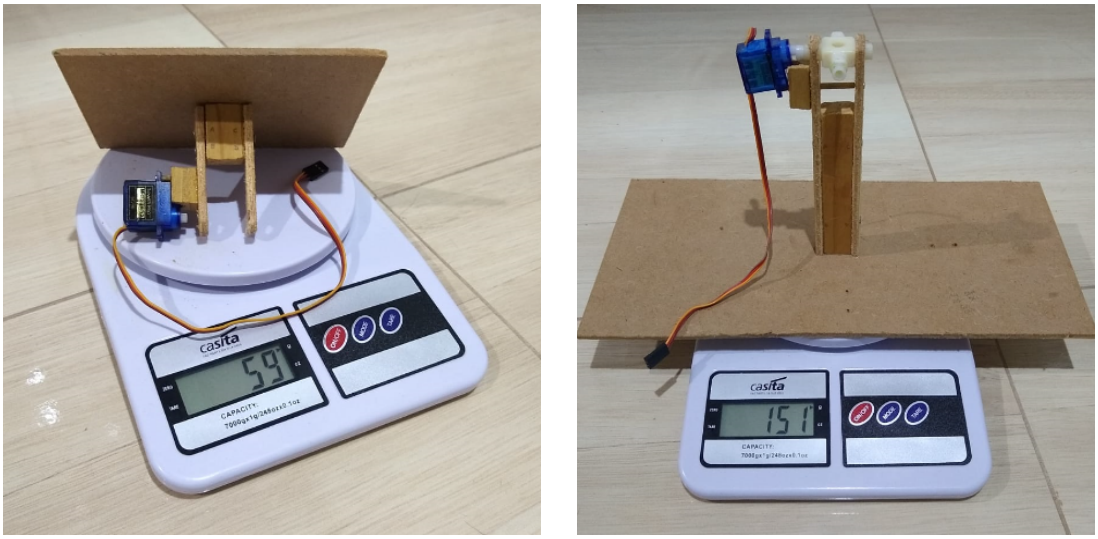
Para garantir um bom funcionamento mecânico do protótipo e ainda permitir que cargas adicionais possam ser adicionadas à placa superior, a estrutura foi projetada para ser mais leve, pesando 59g.

4.4 CONTROLE

4.4.1 Estratégia de Controle

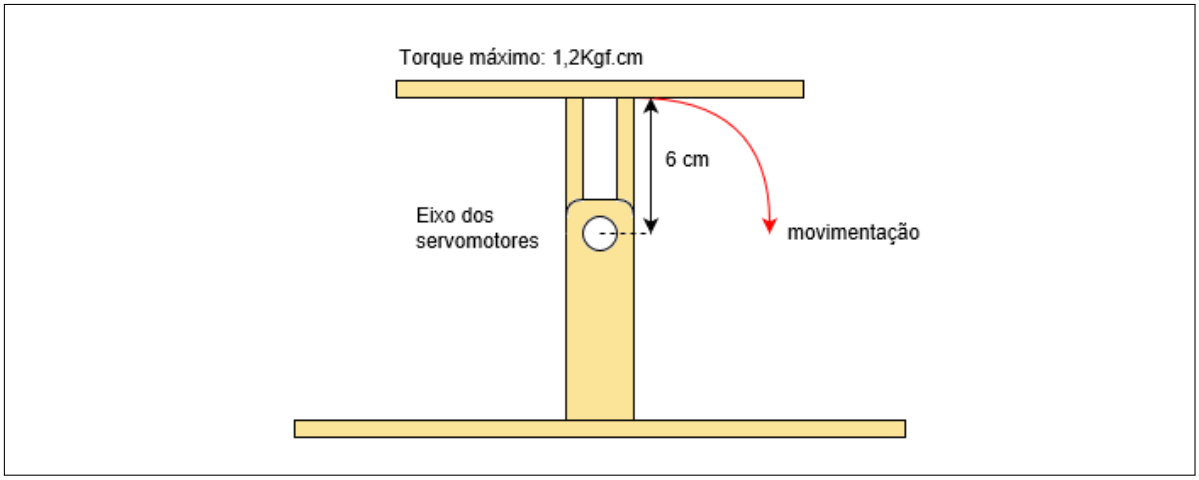
O controle do protótipo foi inicialmente idealizado para ser implementado via controlador PID, porém há particularidades nos equipamentos que impedem a implementação desse tipo de controle. O maior responsável por isso são os servomotores devido à forma como são

Figura 45 – Pesagem do protótipo



Fonte: Autoria Própria

Figura 46 – Vista lateral da movimentação do protótipo



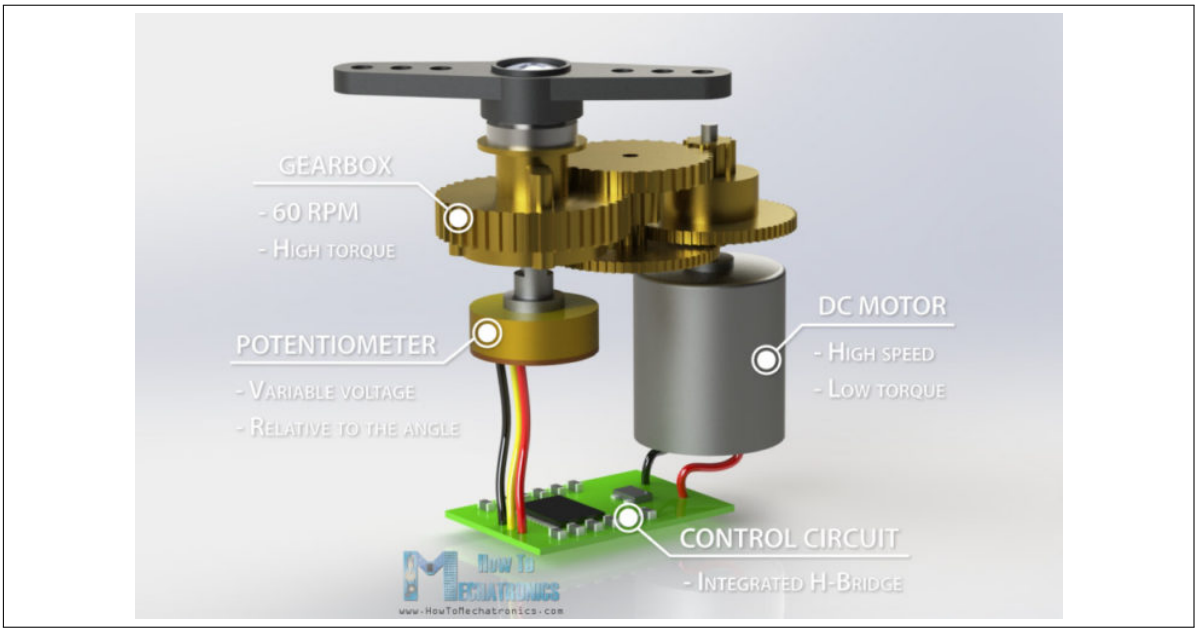
Fonte: Autoria própria

construídos.

Os servomotores utilizados são do tipo DC, compostos por um motor de corrente contínua, potenciômetro, circuito de controle e engrenagens para compor o mecanismo de angulação característico dos servomotores. Servomotores são dispositivos capazes de se movimentar até uma posição e permanecer na posição mesmo ao sofrer intervenções externas. Isso se deve ao circuito de controle interno, que recebe o valor da angulação e compara com o potenciômetro interno, corrigindo a posição do atuador e permanecendo na mesma enquanto o sinal permanecer o mesmo. Basicamente, o próprio servomotor possui um circuito de correção interno que realiza o ajuste de posição e o mantém (HANNIFIN).

Ao testar a implementação de um controlador PID sobre o projeto, houveram problemas com a implementação do integrador. Como o próprio servomotor gerencia o erro internamente para se corrigir (possuindo polo na origem), não há oscilações no controle de posição em torno

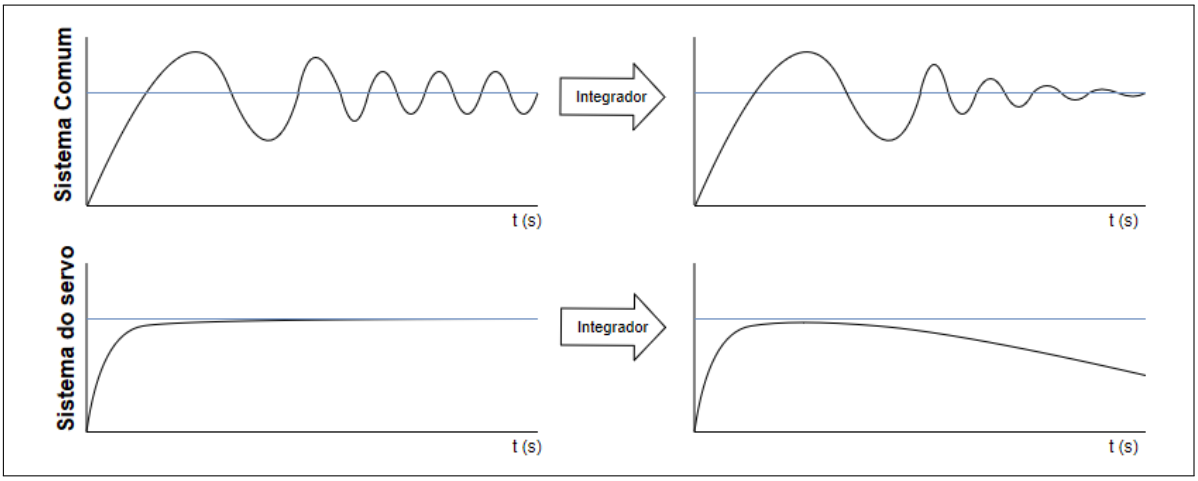
Figura 47 – Composição de um servomotor



Fonte: (NEDELKOVSKI, 2018)

da referência, logo, qualquer aplicação de um controle com a componente integral vai gerar acúmulo de erro em regime permanente, retirando o protótipo da estabilidade e desregulando as posições do servo junto ao sensor. A Figura 48 representa a aplicação do Integrador em uma curva de controle característico do servomotor em comparação com um sistema comum.

Figura 48 – Aplicação da componente integradora em sistemas um servomecanismo



Fonte: Autoria própria

Seguindo dessa maneira, a melhor implementação de controle possível é a de um controle proporcional. Assim, o algoritmo de controle foi realizado de acordo com o Algoritmo 4.

No algoritmo, o valor de *setpoint* é fixado em 0, que é a posição horizontal da placa inferior, e o erro é calculado pela variação lida entre o *setpoint* e a angulação atual medida pelo módulo MPU (variável angulo). O controlador proporcional é dado pela linha seguinte, com constante Kp = 1,25. Este valor é o valor necessário para fazer com que o erro se adeque ao

Algoritmo 4 – Algoritmo de controle proporcional

```
erro = setpoint - angulo;
P = 1.25 * erro;
c_periodo = 187.5-(P);
```

longo da faixa de valores para geração do PWM conforme a Equação 50. Os servos utilizados funcionam com valores de *duty-cycle* entre 0,6ms a 2,4ms, equivalentes aos valores de 75 e 300, respectivamente.

$$180 * x = 300 - 75 = 225 \tag{50}$$

$$x = 1.25 \tag{51}$$

Por fim, a última linha corrige a posição da placa superior através do envio da posição correta ao atuador servomotor. O valor de 187.5 se refere a posição zero (totalmente horizontal, ou 90 graus) e (P) é o controlador aplicado para corrigir a posição. Assim, quando há um desvio da inclinação, o módulo envia o valor a ser corrigido para o microcontrolador que trata o erro e corrige a posição do atuador.

Como grande parte dos sistemas mecânicos, o protótipo não consegue atingir o máximo de movimentação fisicamente, sendo necessários limitantes para evitar que um grau de liberdade acabe interferindo no outro e forçando o motor para uma posição impossível. Para contornar o problema, foram colocadas restrições via software que impedem que ocorra quebra de partes do projeto.

Algoritmo 5 – Algoritmo de controle proporcional com restrições

```
erro = setpoint - angulo;
P = 1.25 * erro;
if(angulo >= 30)
{
c_periodo = 150;
}
if(angulo <= -30)
{
c_periodo = 225;
}
else
{
c_periodo = 187.5-(P);
}
```

No Algoritmo 5 estão descritas três condições simples para tratamento do erro lido pelo sistema. Como fisicamente, de forma empírica, foi verificado um limite máximo de 60 graus de movimentação, o algoritmo foi projetado para que, caso houver inclinações maiores do que 30 graus, o protótipo permanecer fixo em 30 graus.

Outro ponto relevante é a taxa de atualização da rotina de controle. Aproveitando-se do período regrado da interrupção é possível criar um contador para atualização da posição dos servomotores. O código para geração da onda PWM realiza duas interrupções em um período de 20ms (para gerar o tempo de alta e baixa da onda). Dessa forma, sabe-se que a cada 2 contagens do contador tem-se 20ms passados. Assim, para 1 segundo o valor do contador deve ser de 100. Para critérios de ajuste do melhor tempo de atualização foi adotado o valor de 100 (1 segundo) e decrescido 10 unidades até chegar próximo do valor ideal. Assim, foi encontrado o valor de 300ms, que fornece uma taxa de atualização aceitável e confiável.

Algoritmo 6 – Estrutura de controle com utilização de contador

```
if(count>=30){                          //100 = 1 seg, 30 = 300ms
erro = setpoint - angulo;
P = 1.25 * erro;
if(angulo >= 30)
{
c_periodo = 150;
}
if(angulo <= -30)
{
c_periodo = 225;
}
else
{
c_periodo = 187.5-(P);
}
count=0;
}
```

Com esse valor é possível ler a variação de ângulos pelo sensor e corrigir a posição sequencialmente, sem ocorrer problemas de conflito de informações.

4.4.2 Diagrama de blocos

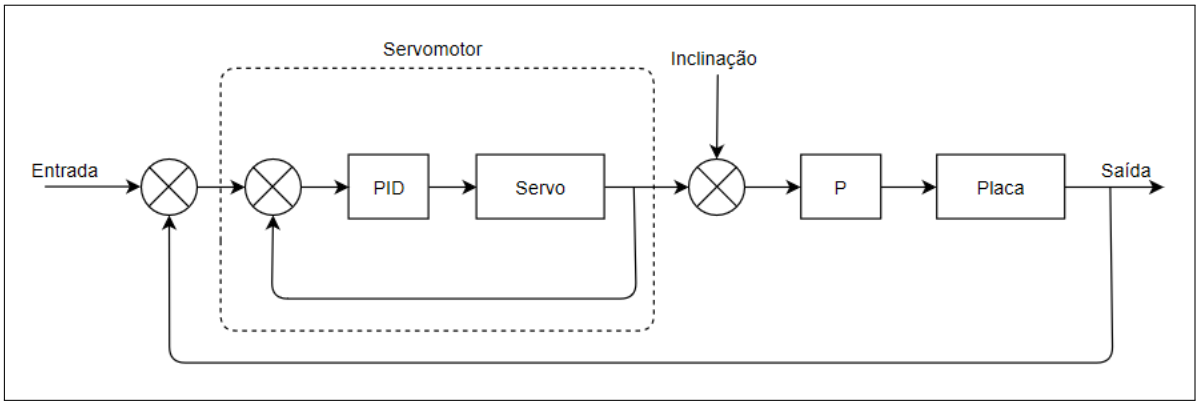
O diagrama de blocos do protótipo é indicado na Figura 49. A entrada se refere ao valor de *setpoint*, a inclinação o valor lido pelo módulo MPU6050 e a saída o valor tratado para controlar a posição do atuador servomotor. Em destaque dentro do retângulo tracejado está representado o controle interno do servomotor, responsável por manter a posição constantemente.

Como observado na Figura 49, o próprio atuador se encarrega de manter a posição de entrada, enquanto o controle externo recebe a inclinação, aplica o controle proporcional junto a placa e retorna o erro para o início do sistema onde o novo valor de posição será enviado para o servomotor.

4.4.3 Resultados finais

O protótipo final, mostrado na Figura 50, mostrou bons resultados de correção de posição mesmo com pequenas imperfeições. Por não ter uma construção totalmente perfeita, há

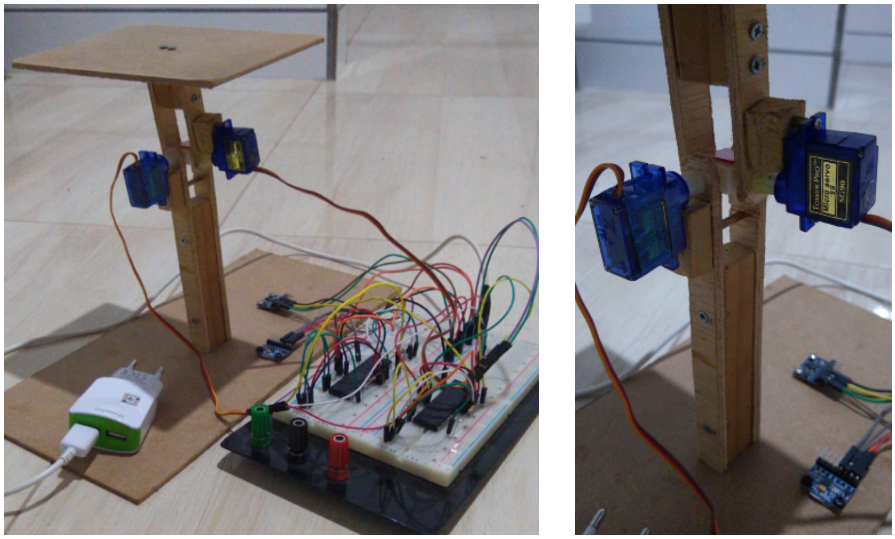
Figura 49 – Diagrama de blocos do sistema



Fonte: Autoria própria

variações na horizontalidade da placa superior. A velocidade de correção, 300ms, é razoável para uma correção rápida, considerando uma variação angular na placa inferior sem muita mudança em curtos espaços de tempo.

Figura 50 – Protótipo finalizado

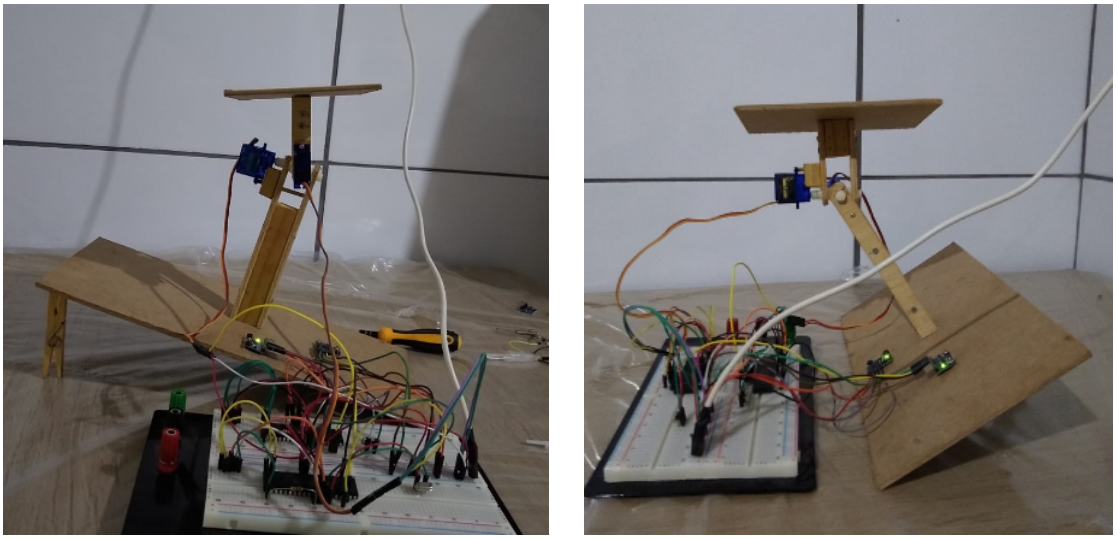


Fonte: Autoria Própria

Quanto às inclinações, a Figura 51 apresenta a comunicação entre cada um dos servomotores e módulos acelerômetros, onde é possível verificar que a correção é efetiva e precisa, considerando a estrutura do projeto. A fixação dos componentes também pode ser considerada estável para o peso da estrutura, porém pode ser necessário desenvolver novas tecnicas para fixação em um cenário onde há mais massa envolvida.

No geral, o Controle Proporcional cumpre bem o prometido e garante uma boa movimentação do protótipo.

Figura 51 – Resultados de correção para cada uma das posições do protótipo



Fonte: Autoria própria

5 CONCLUSÃO

Controlar um sistema pode ser uma tarefa árdua, mas com uma implementação adequada e um bom ajuste, o controle poderá permitir que o projeto realize de forma satisfatória qualquer coisa que lhe for imposta. O objetivo principal deste trabalho é o desenvolvimento de uma plataforma de estabilidade com dois graus de liberdade, utilizando um controle clássico para a correção de cada uma das posições. Conforme análise e testes realizados, para os atuadores presentes no sistema não há uma forma simples de aplicação de um controle PID que gere vantagens neste caso, restando apenas implementar um Controle Proporcional ajustado à faixa de controle de um servomotor. Porém, apesar de não ser tão completo quanto o PID, cumpre muito bem o propósito ao qual foi implementado, corrigindo proporcionalmente o erro de acordo com as leituras realizadas pelo módulo acelerômetro. Para o protótipo desenvolvido, o peso e forma de acoplamento dos elementos, bem como fonte de alimentação e componentes foram posicionados especificamente para ele, portanto, caso haja uma implementação do sistema em uma maior escala, talvez sejam necessários ajustes nos motores e principalmente na estrutura física, permitindo que seja mais resistente a cargas extras. No geral, o protótipo cumpriu o objetivo proposto, provando ser possível um controle de posição com servomotores. Para futuros trabalhos ou derivados, pode ser considerada a utilização de um microcontrolador com mais poder computacional, capaz de controlar todo o sistema com apenas uma unidade. Ainda como sugestão, os atuadores podem ser substituidos por motores DC com redução, que são capazes de garantir certo torque e velocidade, permitindo ainda uma implementação funcional de controle PID para a velocidade dos mesmos. Para aprimorar a estrutura, podem ser idealizadas hastes extras ou pistões nas extremidades das placas, sendo assim capazes de distribuir o peso da carga e fortalecer o protótipo como um todo.

REFERÊNCIAS

BAZANELLA, Alexandre; SILVA, João da. **Ajuste de Controladores PID**. [S.l.], 2000. Disponível em: <http://www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/apostila.html>. Citado 6 vezes nas páginas 16, 18, 19, 20, 22 e 41.

CAMARA, Romulo. **Protocolo I2C**. 2013. Disponível em: <http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf>. Citado na página 33.

CASTRUCCI, P.B. De Lauro; BITTAR, A.; SALES, R.M. **Controle Automatico**. [S.l.]: LTC, 2011. Citado na página 15.

FILIPEFLOP. **MPU6050 Pinagem**. Disponível em: <https://www.filipeflop.com/wp-content/uploads/2014/09/GY-521-MPU-6050-Pinos1.jpg>. Citado na página 39.

FUTURE ELECTRONICS. **What is a Microcontroller?** [S.l.]. Disponível em: <http://www.futureelectronics.com/en/Microcontrollers/microcontrollers.aspx>. Citado na página 32.

HANNIFIN, Parker. **Fundamentals of Servo Motion Control**. Disponível em: <http://www.compumotor.com/whitepages/ServoFundamentals.pdf>. Citado na página 55.

INVENSENSE. **MPU6050 Data Sheet**. [S.l.], 2013. Disponível em: <https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf>. Citado 4 vezes nas páginas 38, 39, 50 e 51.

LEDUC, John. Servo motors and control with arduino platforms. **Digi-Key Electronics**, 2017. Disponível em: <https://www.digikey.com/en/articles/techzone/2017/mar/servo-motors-and-control-with-arduino-platforms#whatisaservomotor>. Citado na página 31.

MACIEL, Marcelo. Controle pid com aproximação digital para utilização no pic. **MicroControlado**, 2012. Disponível em: <http://microcontrolado.com/controle-pid-no-pic/>. Citado na página 40.

MATIAS, Juliano. Teoria de controle pid. **Mecatrônica Atual**, n. 3, 2002. Citado 7 vezes nas páginas 14, 15, 16, 17, 18, 20 e 21.

MATIC, Nebojsa. **PIC microcontrollers, for beginners too**. Microchip, 2000. Disponível em: <http://groups.csail.mit.edu/lbr/stack/pic/pic-microcontrollers.pdf>. Citado na página 33.

MICROCHIP. **PIC16f87XA Data Sheet**. [S.l.], 2003. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf>. Citado 2 vezes nas páginas 40 e 46.

MICROPIK. **SG90 9g MicroServo**. [S.l.]. Disponível em: <http://www.micropik.com/PDF/SG90Servo.pdf>. Citado 3 vezes nas páginas 31, 37 e 38.

NAYLAMPMECHATRONICS. **Tutorial MPU6050, Acelerómetro y Giroscopio**. Disponível em: <https://naylampmechatronics.com/blog/45_Tutorial-MPU6050-Acelerómetro-y-Giroscopio.html>. Citado 2 vezes nas páginas 29 e 30.

NEDELKOVSKI, Dejan. How servomotors work and how to control servos using arduino. 2018. Disponível em: <https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/>. Citado na página 56.

OGATA, Katsuhiko. **Engenharia de Controle Moderno**. 5. ed. [S.l.]: Pearson Prentice Hall, 2010. Citado 5 vezes nas páginas 14, 21, 22, 23 e 24.

OMEGA. Accelerometers: Introduction to acceleration measurement. ago. 2018. Disponível em: <https://www.omega.com/en-us/resources/accelerometers>. Citado na página 29.

PERILLO, Matheus. **Servos analógicos e digitais: funcionamento, usos e diferenças entre eles**. [S.l.], 2011. Disponível em: <http://www.ventonortebh.com.br/pdfs/servos.pdf>. Citado 2 vezes nas páginas 31 e 32.

SANTOS, Josemar Dos. **Modelagem Matemática**. [S.l.], 2008. Disponível em: <http://www3.fsa.br/mecanica/arquivos/MEC442%20-%20Modelagem%20Josemar.pdf>. Citado 5 vezes nas páginas 24, 25, 26, 27 e 28.

SOUZA, David José De. **Conectando o PIC16F877A: Recursos Avançados**. 4. ed. [S.l.]: Erica, 2007. Citado na página 40.

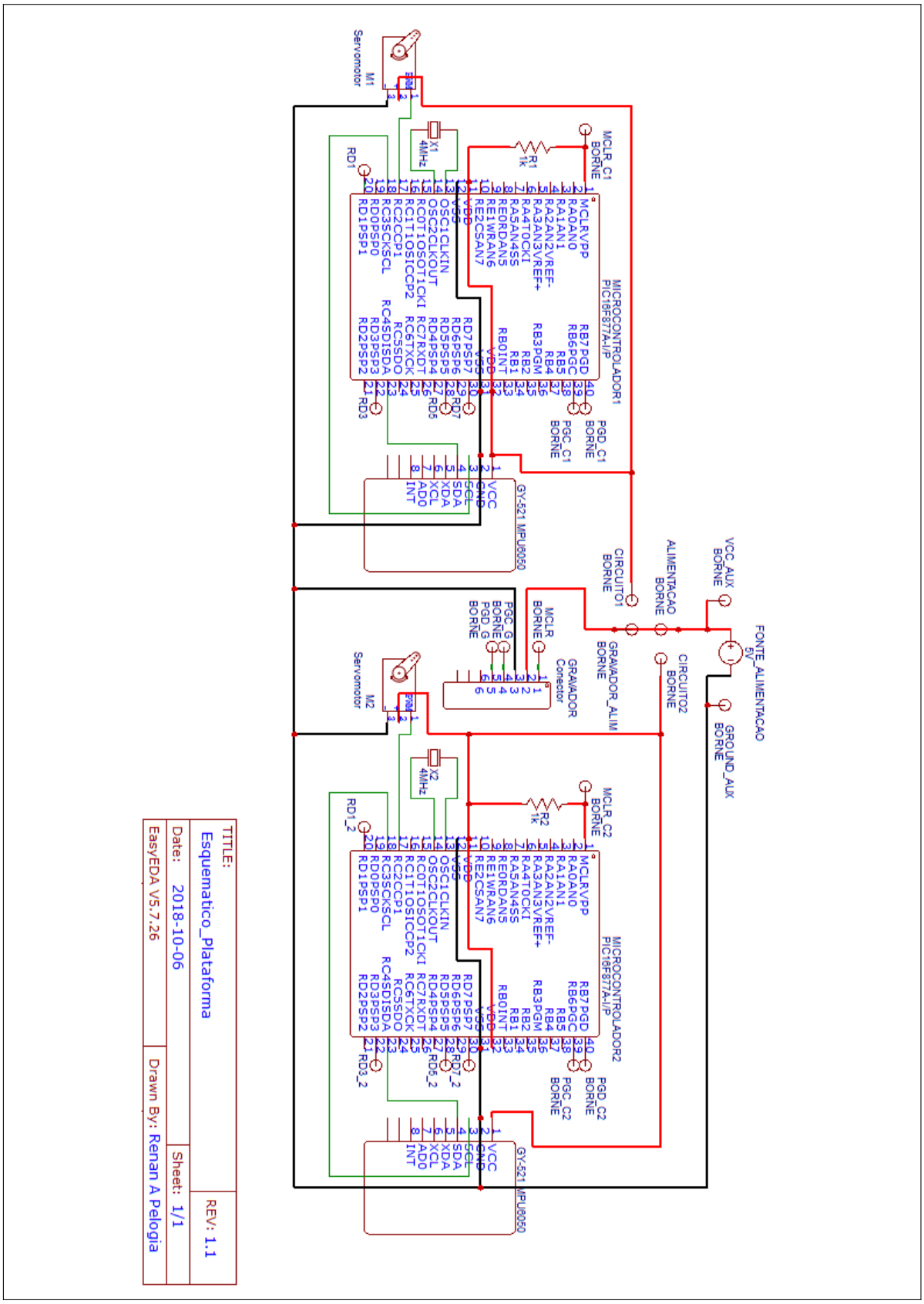
SPARKFUN. Accelerometer basics. Disponível em: <https://learn.sparkfun.com/tutorials/accelerometer-basics/all>. Citado na página 29.

VIEIRA, Mario Elias Marinho; STEVAN, Sergio Luiz. Análise geral sobre o acelerômetro. 2013. Disponível em: <https://www.researchgate.net/publication/277323483_Analise_geral_sobre_o_acelerometro>. Citado na página 29.

VILELA, Paulo Sergio Da Camara; VIDAL, Francisco Jose Targino. Automação industrial. **DCA - Universidade Federal do Rio Grande do Norte**, 2003. Disponível em: <https://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_19.pdf>. Citado na página 14.

Apêndices

APÊNDICE A – IMPLEMENTAÇÃO EM CIRCUITO



APÊNDICE B – CÓDIGO EM C

Algoritmo 1 – Código do protótipo em C

```
//====================================================================
//  CÓDIGO GERAL PARA CONTROLE   |   I2C Adaptado de electrosome.com/
//  DE SERVO E MODULO ACCL. COM  |   i2c-pic-microcontroller-mplab-xc8/
//  PIC16F877A                   |
//  XC8                          |   01/05/2019
//====================================================================
//--------------------------------------------------------------------
//  DEFINIÇÕES
//--------------------------------------------------------------------
#define _XTAL_FREQ 4000000          // Frequência do Cristal em 4MHz
#define TMR1PRESCALE 8              // Timer1 prescaler 8
#define OUT RC2                     // RC2 definido como OUT
//--------------------------------------------------------------------
//  BIBLIOTECAS
//--------------------------------------------------------------------
#include <xc.h>
#include <math.h>
//--------------------------------------------------------------------
//  CONFIGURAÇÃO
//--------------------------------------------------------------------
#pragma config FOSC = HS    // Oscillator Selection bits (HS
  ↪ oscillator)
#pragma config WDTE = OFF   // Watchdog Timer Enable bit (WDT enabled)
#pragma config PWRTE = OFF  // Power-up Timer Enable bit (PWRT
  ↪ disabled)
#pragma config BOREN = ON   // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = OFF    // Low-Voltage (Single-Supply) In-Circuit
  ↪ Serial Programming Enable bit (RB3 is digital I/O, HV on MCLR
  ↪ must be used for programming)
#pragma config CPD = OFF    // Data EEPROM Memory Code Protection bit
  ↪ (Data EEPROM code protection off)
#pragma config WRT = OFF    // Flash Program Memory Write Enable bits
  ↪ (Write protection off; all program memory may be written to by
  ↪ EECON control)
#pragma config CP = OFF     // Flash
//--------------------------------------------------------------------
//  DECLARAÇÃO DE VARIÁVEIS
//--------------------------------------------------------------------
```

```
unsigned long CCPR = 0; // armazena valor para ser alocado em CCPxH e
  ↪ CCPxL
unsigned long c_periodo = 187.5; //Posição central do servomotor ou
  ↪ 1.5ms
const unsigned long t_periodo = 2500; // 20ms p/ 50hz frequencia.
float angulo = 0; //variável para captura do angulo
int count =0; //contador
unsigned char AXH, AXL, AYH, AYL, AZH, AZL; //Variáveis para receber
  ↪ valores dos registradores do módulo MPU
//unsigned short AX, AY, AZ;
double AX, AY, AZ; //Composição das componentes H e L de cada eixo
float setpoint = 0, erro = 0, P = 0; //pos. central, erro e controle
//-------------------------------------------------------------------
//  INTERRUPÇÃO DE GERAÇÃO PWM
//-------------------------------------------------------------------
void interrupt tmr1isr() {
if (CCP1IF == 1) {                                    // Se a flag
  ↪ de interrupção do modo compare - CCP ta on
if ((c_periodo > 0) && (c_periodo < t_periodo)) { // Se o período é >
  ↪ 0% e < 100%:

if (OUT == 1) {                                  // Caso a saída seja
  ↪ 1 -> estava em alta.
OUT = 0;                                        // Saída agora é 0 p/
  ↪ gerar o tempo de baixa.
CCPR = t_periodo - c_periodo;                   // periodo total -
  ↪ periodo atual => baixa = total - alta.
}
else {                                           // Se saída era 0 ->
  ↪ estava em baixa.
OUT = 1;                                        // Muda saída para 1 para
  ↪ contabilizar o tempo em alta
CCPR = c_periodo;                               // CCPR armazena periodo
  ↪ de alta p/ contagem.
}
} else {                                              //Se o período
  ↪ atual não está entre 0ms e 20ms então:
if (c_periodo == t_periodo) {//
OUT = 1;
}                                                // se duty = 100%,
  ↪ então OUT é 1 todo tempo.
if (c_periodo == 0) {
OUT = 0;
```

```
}                                                    // se duty = 0%,
  ↪ então OUT é 0 todo tempo.
}
// Atualizar valores referência para as contagens de estouro:
CCPR1H = CCPR >> 8; // seleciona os 8 bits mais significativos de
  ↪ CCPR e armazena no registrador CCPR1H
CCPR1L = CCPR; // seleciona os 8 bits menos significativos de CCPR e
  ↪ armazena no registrador CCPR1L
CCP1IF = 0; // reseta a flag de interrupção do CCP1
}
count = count+1;
if(count>=30){                          //100 = 1 seg, 30 = 300ms

erro = setpoint - angulo;          //0, que é o setpoint - a leitura do
  ↪ mpu
P = 1.25 * erro;                  // que é 187,5 (centro) - 75 (min) =
  ↪ 112,5. Divide por 90 dá 1,25
if(angulo >= 30)
{
c_periodo = 150;
}
if(angulo <= -30)
{
c_periodo = 225;
}
else
{
c_periodo = 187.5-(P);
}
count=0;
}

}
//---------------------------------------------------------------------
//  CONFIGURAÇÃO DE INICIO I2C
//---------------------------------------------------------------------
void I2C_Master_Init(const unsigned long c)
{
SSPCON = 0b00101000;              //SSP Module como Mestre
SSPCON2 = 0;
SSPADD = (_XTAL_FREQ/(4*c))-1; //Velocidade de Clock
SSPSTAT = 0;
TRISCbits.TRISC3 = 1;                    //Entrada
TRISCbits.TRISC4 = 1;                    //Entrada
```

```
}
//---------------------------------------------------------------------
// ESPERA I2C
//---------------------------------------------------------------------
void I2C_Master_Wait()
{
while ((SSPSTAT & 0x04) || (SSPCON2 & 0x1F)); //Transmissão em
  ↪ progresso
}
//---------------------------------------------------------------------
// INÍCIO I2C
//---------------------------------------------------------------------
void I2C_Master_Start()
{
I2C_Master_Wait();
SEN = 1;                //Condição de início (S)
}
//---------------------------------------------------------------------
// REINÍCIO I2C
//---------------------------------------------------------------------
void I2C_Master_RepeatedStart()
{
I2C_Master_Wait();
RSEN = 1;               //Condição de Re-start
}
//---------------------------------------------------------------------
// PARADA I2C
//---------------------------------------------------------------------
void I2C_Master_Stop()
{
I2C_Master_Wait();
PEN = 1;                //Stop
}
//---------------------------------------------------------------------
// ESCRITA I2C
//---------------------------------------------------------------------
void I2C_Master_Write(unsigned d)
{
I2C_Master_Wait();
SSPBUF = d;             //Escrita SSPBUF
}
//---------------------------------------------------------------------
// LEITURA I2C
//---------------------------------------------------------------------
```

```
unsigned short I2C_Master_Read(unsigned short a)
{
unsigned short temp;
I2C_Master_Wait();
RCEN = 1;
I2C_Master_Wait();
temp = SSPBUF;       //Leitura de dados de SSPBUF
I2C_Master_Wait();
ACKDT = (a)?0:1;     //Acknowledge bit
ACKEN = 1;           //Acknowledge sequence
return temp;
}
//-------------------------------------------------------------------------
//  ROTINA PRINCIPAL MAIN
//-------------------------------------------------------------------------
void main()
{
TRISCbits.TRISC2 = 0;            //Movimentação do servo
TRISD = 0;                       //Portas D como saída
//PORTDbits.RD1 = 0;
//PORTDbits.RD3 = 0;
//PORTDbits.RD5 = 0;
//PORTDbits.RD7 = 0;
PORTCbits.RC2 = 0;               //Porta C2 com nv. lógico baixo

//TIMER 1 E CCP1 – GERAÇÃO DE ONDA
  ↪ 20MS-----------------------------------
T1CON = 0b00110000; // timer1 com prescaler 8 e desligado
TMR1H = 0; // timer1 iniciando de 0
TMR1L = 0;

CCP1CON = 0x0b; // módulo CCP em compare e special event trigger
  ↪ configurado
CCPR = 0; // inicio em 0.
CCP1IF = 0; // limpa flag de interrupção CCP1
CCP1IE = 1; // interrupção CCP1 habilitada
INTCON = 0xC0; // habilitando interrupções global e periférica
T1CON = 0b00110001; // inicia contagem do timer1

I2C_Master_Init(100000);                //I2C a 100KHz clock
while(1)
{
//-------------------------------------------------------------------------
//  ROTINA DE CONFIGURAÇÃO
```

```
//---------------------------------------------------------------------
I2C_Master_Start();                 //Start condition
I2C_Master_Write(0b11010000);       //7 bit address (0x68) + Write
  ↪ (Endereço do Slave + W)
I2C_Master_Write(0x6B);             //Registrador de Power_up_MNG
I2C_Master_Write(0);                //escreve no registrador 0x6B o valor
  ↪ 00000000
I2C_Master_Stop();                  //Stop condition
//__delay_ms(100);
//---------------------------------------------------------------------
//  ROTINA DE LEITURA DE DADOS
//---------------------------------------------------------------------
I2C_Master_Start();                 //Start condition
I2C_Master_Write(0b11010000);       //7 bit address + Write (Endereço do
  ↪ Slave + W)
I2C_Master_Write(0x3B);             //Registrador que inicia as leituras
  ↪ (parece que esse já é o acc)

I2C_Master_RepeatedStart();         //restart como mostra o datasheet
I2C_Master_Write(0b11010001);       //7 bit address + Read (Endereço do
  ↪ Slave + R)

AXH = I2C_Master_Read(1);           //Read + Acknowledge na verdade aqui
  ↪ ja é o Acc
AXL = I2C_Master_Read(1);           //Read + Acknowledge
AYH = I2C_Master_Read(1);           //Read + Acknowledge
AYL = I2C_Master_Read(1);           //Read + Acknowledge
AZH = I2C_Master_Read(1);           //Read + Acknowledge
AZL = I2C_Master_Read(0);           //Read + Acknowledge
//OBS: O ultimo byte lido deve retornar o teste de ACK/NACK como
  ↪ NACK, por padrão do MPU6050.

I2C_Master_Stop();                  //Stop condition

AX = (AXH << 8) | (AXL & 0xff);
AY = (AYH << 8) | (AYL & 0xff);
AZ = (AZH << 8) | (AZL & 0xff);
angulo = atan(AY/sqrt(pow(AX,2) + pow(AZ,2)))*(180.0/3.14);


if (angulo<0){
PORTDbits.RD1 = 1;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 0;
```

```
PORTDbits.RD7 = 0;
}
if ((angulo>=25)&&(angulo<35)){        //30
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 1;
PORTDbits.RD5 = 0;
PORTDbits.RD7 = 0;
}
if ((angulo>=55)&&(angulo<65)){        //60
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 1;
PORTDbits.RD7 = 0;
}
if ((angulo>=80)&&(angulo<=90)){
PORTDbits.RD1 = 0;
PORTDbits.RD3 = 0;
PORTDbits.RD5 = 0;
PORTDbits.RD7 = 1;
}
__delay_ms(20);
//-------------------------------------------------------------------------------
}
}
```