Universidade Federal do Paraná

Rogério da Silva Martins

# Desenvolvimento de plataforma para navegação ponto-a-ponto de aeromodelo com telemetria e visualização em aplicação embarcada

Curitiba PR
2016

Rogério da Silva Martins

# Desenvolvimento de plataforma para navegação ponto-a-ponto de aeromodelo com telemetria e visualização em aplicação embarcada

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação.*

Orientador: Eduardo Todt.

Curitiba PR
2016

**Ficha Catalográfica**

*Esta folha deve ser substituída pela ficha catalográfica fornecida pela Biblioteca Central da UFPR, a pedido da secretaria do PPGInf/UFPR (vide arquivo* `ficha.tex`*).*

**Termo de aprovação**

*Esta folha deve ser substituída pela ata de defesa ou termo de aprovação devidamente assinado, que será fornecido pela secretaria do programa após a defesa ter sido concluída e aprovada (vide arquivo `aprovacao.tex`).*

*A meu pai Reni, meu modelo de caráter, que com muito apoio e carinho não mediu esforços para que eu chegasse até esta etapa de minha vida e minha avó Avani, que dedicou sua fé a mim e que muito quis presenciar esse momento, mas partiu antes de ele se concretizar.*

# Agradecimentos

A Deus por minha vida, família e amigos.

À minha família, pelo amor, incentivo e apoio incondicional.

Ao professor Eduardo Todt por toda orientação e ajuda que me foram dadas.

À Isadora Sebben, pelo apoio, carinho e participação ativa na construção deste trabalho com seu notório conhecimento da língua portuguesa.

Ao Afonso Bastos, que compartilhou sua experiência ajudando na construção do trabalho e sua esposa Sônia Bastos por me acolher carinhosamente nos dias de trabalho.

Aos meus amigos, de longa data e os que adquiri durante minha formação, por confiarem em mim e estarem ao meu lado, contribuindo e me apoiando neste trabalho e em todos os momentos da vida.

# Resumo

Este trabalho foi realizado com o objetivo de construir uma plataforma que permita a automação da navegação de aeromodelos e a sua telemetria. Para o desenvolvimento, foram pesquisados artigos recentes com propostas similares a deste trabalho com o intuito de prover uma base de conhecimento para a construção da plataforma. Foi desenvolvida uma pesquisa de componentes de hardware que captam informações necessárias para navegação e telemetria. Os componentes escolhidos foram integrados em uma única placa de circuito impresso projetada em software de modelagem para unificar o hardware de forma compacta e confiável. Um circuito de chaveamento automático/manual do acionamento dos comandos dos servo-motores que controlam a atitude do aeromodelo também foi projetado e construído como medida de segurança e usabilidade do sistema. Para a telemetria, um sistema de aquisição de dados de atitude e posição do aeromodelo foi desenvolvido e um computador de campo foi construído proporcionando a visualização em solo das informações de voo.

A plataforma funcionou de forma satisfatória, tendo em vista os resultados dos testes realizados para sua validação, e pode ser utilizada para automatizar aeromodelos incluindo telemetria. O chaveamento auto/manual funciona sem falhas e a comunicação da telemetria, baseada em Zigbee, atinge comunicação com taxa de perda de pacotes inferior a 13%, dentro da distância máxima de teste, 500 metros.

**Palavras-chave:** Navegação Autônoma, Telemetria, Sistemas Embarcados, UAV.

# Abstract

This work has as goal to construct a platform that allows to automate model aircraft navigation and telemetry. For development, recent articles with similar proposals were studied, aiming to provide a knowledge base to platform construction. A research in hardware components was conducted to collect information about navigation and telemetry. The chosen components were integrated in a single printed circuit board, projected with a circuit modelling software to build the hardware in a compact and reliable way. A switching circuit of automatic/manual actuation of commands to the servomotors, which control the behavior of model aircraft, was also projected and built as a security measure and also improving system usability. For telemetry, an acquisition system of behavior data and position of the model aircraft was developed. A robust field computer was also built to provide flight data visualization on ground.

The platform worked in a satisfactory way, taking into account the tests performed for its validation, therefore can be used for model aircraft automation and telemetry. The auto/manual switching works without failures and communication about telemetry, based on Zigbee, reaches communication with packet loss rate less than 13%, within maximum test distance of 500 meters.

**Keywords:** Autonomous Navigation, Telemetry, Embedded systems, UAV.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Acrônimos

| | |
|---|---|
| DINF | Departamento de Informática |
| UFPR | Universidade Federal do Paraná |
| IEEE | Institute of Electrical and Electronics Engineers |
| PPM | Pulse Position Modulation |
| UAV | Unmanned Aircraft Vehicle |
| VANT | Veículo Aéreo Não Tripulado |
| FPGA | Field Programmable Gate Array |
| PID | Proportional Integral Derivative |

# Lista de Símbolos

# Capítulo 1

# Introdução

Um piloto automático na aviação é um sistema que assume o controle de uma aeronave e a mantém em um curso. O primeiro piloto automático foi inventado por Lawrence Burst Sperry e sua patente data de 1929 [Sperry, 1929]. Desde a invenção até os dias de hoje, os sistemas de piloto automático vêm sendo aprimorados e os mais atuais são capazes até de pousar uma aeronave em condições de visibilidade zero, mas ainda assim não excluem a necessidade de um piloto. Por garantia de segurança, é altamente desejável que o piloto automático esteja sob controle do piloto o tempo todo. O piloto automático dá assistência ao piloto e o alivia da rotina das operações de voo, permitindo-o focar em outros problemas de voo [Young et al., 1944]. Uma função importante do piloto é a de assumir o controle em caso de falha ou mau funcionamento do sistema, sendo que essa troca de controle não pode ser lenta em relação às constantes de tempo do sistema.

Um UAV (Unmanned Aerial Vehicle) é um veículo aéreo não tripulado que pode ser controlado remotamente por um piloto ou voar por um trajeto pré programado através de um sistema de piloto automático. Os UAVs são muito utilizados no meio militar e nos últimos anos estão surgindo no meio civil, com a popularização dos helimodelos multimotores, que são helicópteros rádio controláveis, popularmente chamados de *drones*[1] e também aeromodelos, que são aviões rádio controláveis. Alguns destes *drones* possuem funcionalidades de piloto automático implementadas, como estabilização de altitude e "return to home", uma função de retorno do *drone* para um ponto cadastrado. As possibilidades de uso de um UAV são inúmeras e um sistema de piloto automático pode proporcionar ainda mais alternativas de uso, como por exemplo, monitoramento de áreas e fotografia aérea.

Em um piloto automático, há um software responsável pelo controle da aeronave e uma plataforma que o executa, atuando mecanicamente nos comandos do aeromodelo. O objetivo deste trabalho é a pesquisa e o desenvolvimento de uma plataforma que possa ser utilizada para a implementação de um sistema de navegação e também a telemetria de um aeromodelo. Para atingir esse objetivo, uma plataforma foi projetada para fornecer através de sensores as informações necessárias para que um sistema de navegação possa ser implementado. Os sensores utilizados na plataforma fornecem os dados para a telemetria e um software foi desenvolvido para a visualização remota desses dados.

A plataforma em questão não possibilita a implementação de um sistema que opere o aeromodelo em todos os seus estágios de voo. Decolagem e pouso, por exemplo, devem ser realizadas manualmente, pois a implementação dessas operações demandaria

[1] Drone (em português: zangão, zumbido) é um apelido informal, originado nos EUA, que caracteriza todo e qualquer objeto voador não tripulado.

equipamentos precisos e confiáveis, sendo menos acessíveis financeiramente e de uso mais complexo. Para contornar essa limitação a plataforma possibilita o chaveamento do controle manual para o sistema de navegação implementado podendo ser acionado quando o aeromodelo já estiver em voo. O chaveamento desenvolvido na plataforma também é um dispositivo de segurança para o aeromodelo por prover a retomada de controle manual quando o sistema implementado se mostrar inconsistente ou falho.

Este trabalho foi desenvolvido através da codificação de módulos de localização, posicionamento e comunicação, e a partir do desenvolvimento de um circuito de chaveamento do sistema, que também tem a função de dar segurança com a retomada de controle manual caso ocorra a perda de controle do UAV. Testes do funcionamento do hardware foram realizados em condições simuladas para validar a plataforma.

O trabalho está estruturado da seguinte forma:

- Capítulo 2 - Principais ideias dos artigos pesquisados e Conceitos básicos de um aeromodelo.
- Capítulo 3 - Apresentação dos componentes de hardware utilizados.
- Capítulo 4 - Detalhamento e explicação da implementação.
- Capítulo 5 - Testes da plataforma e telemetria.
- Capítulo 6 - Conclusões e considerações finais.

# Capítulo 2

# Trabalhos Relacionados e Conceitos Básicos de um Aeromodelo

Neste capítulo são apresentadas as principais abordagens utilizadas em artigos recentes, que possuem conteúdo no contexto proposto neste trabalho. Todos artigos mencionados são obtidos da base IEEE Xplore (www.ieee.org/xplore) com termos de busca como Autopilot, UAV e VANT. Os conceitos básicos de um aeromodelo também são apresentados neste capítulo.

## 2.1 Trabalhos Relacionados

Em *The Design of Flight Control System for Small UAV with Static Stability*[Yang and Geng, 2011] é descrito um sistema de controle de voo em um UAV com estabilidade estática[1]. Para este sistema foi utilizado um processador ARM para controle e navegação e um processador em FPGA para lidar com a interface dos periféricos, como giroscópio angular, acelerômetro, sensores de pressão e GPS. O sistema como um todo mantém o controle de navegação, controle de altitude, controle de direção e ainda manipula as situações especiais, como a perda de sinal de GPS, tentando manter o curso de voo. Seu controle de navegação trabalha verificando a rota que está sendo seguida para um ponto determinado e caso a rota não esteja correta, tenta retornar assim que possível. Depois de percorrer todos os pontos, se o controle em terra não mandar algum comando, o UAV pode continuar voando e seguindo pontos que formam um círculo.

Em alguns casos é necessário que a altitude do UAV seja mantida e para isso o controle de voo trabalha com os laços de controle de altitude e velocidade. Em ambos os laços o sistema verifica se o seu estado é o desejado e caso não seja, realiza a correção. A altitude do UAV pode variar com a velocidade e para mantê-la, o controle utilizado no comando do motor em geral é o PID[Ogata, 1982] (ver apêndice A.1) no comando do motor. No laço de altitude também é utilizado PID.

Nos casos de falha, a fim de aumentar a segurança de voo, se o sistema perder o sinal GPS, o UAV entra uma rota circular. Se o UAV perder a conexão com o controle em terra o controle de voo retorna o UAV. Além disso, quando a altitude do UAV for inferior a um determinado valor e a taxa de descida estiver alta, indicando que o UAV está caindo rapidamente, pode ser feita a abertura de um paraquedas.

[1] avião capaz de retornar ao seu estado original de equilíbrio após perturbações externas sem a ajuda de um sistema de controle

Já em *Tiriba - A New Approach of UAV based on Model Driven Development and Multiprocessors*[Branco et al., 2011], que apresenta um sistema o qual realiza todas as funções de um piloto de avião convencional, o sistema desenvolvido foi dividido em quatro subsistemas, que são Navegação, Controle, Inércia e Pressão e cada um deles possui um processador dedicado. Um smartphone é utilizado para enviar coordenadas para (GPS) que o UAV deve percorrer, diferente do sistema anterior[Yang and Geng, 2011] que já possui os pontos pré carregados e é a unidade de navegação que guia o UAV ao longo desses pontos, tirando uma fotografia a cada ponto alcançado. A unidade de controle é encarregada de manter a velocidade, ângulo de rolamento, ângulo de derrapagem, altitude e a velocidade, controlando o motor. Em resumo, ela tem a missão de iniciar o modo de voo (direto, estabilizado e autônomo) e executar os comandos recebidos da unidade de navegação. A unidade de inércia é responsável por estimar a postura da aeronave e a unidade de pressão fornece por meio de sensores a altitude, velocidade vertical e a velocidade aerodinâmica baseada na pressão do ar.

Para a segurança de uso, este sistema é pré-configurado de modo que não ultrapasse uma região delimitada, ou seja, os pontos enviados fora dessa região não serão aceitos. Um diferencial deste sistema é o fato de ele retornar à base após completar a sua missão e pousar autonomamente com paraquedas. Caso algum problema ocorra durante sua missão, o piloto automático faz o UAV retornar à base.

Outra implementação para UAV estudada é a de [Liu et al., 2012], que utiliza uma abordagem diferente para navegação. Em vez de utilizar GPS, este sistema gerencia sua localização com os sinais de Wi-Fi encontrados pelo percurso. Este sistema se mostrou eficaz por ter sido implementado em um drone, que é um veículo aéreo não tripulado com capacidade de ficar posicionado fixamente em um lugar e também de se locomover sem um limite mínimo de velocidade, ao contrário de um aeromodelo que necessita de uma velocidade mínima de sustentação para não ocorrer estol[2]. Há um sistema baseado na navegação Wi-Fi para aeromodelos também mencionado neste artigo, mas sua eficácia não é satisfatória devido às características de voo diferentes dos drones.

## 2.2 Conceitos Básicos de um Aeromodelo

Este capítulo descreve a estrutura e comportamento de um aeromodelo, que pode ser considerado uma aeronave em pequena escala remotamente controlada. As principais estruturas de um aeromodelo são apresentadas a seguir.

Um aeromodelo move-se em três dimensões. Os movimentos nessas dimensões são chamados de *roll, pitch e yaw*, como mostrado na Figura 2.1. Roll é a rotação sobre o eixo longitudinal, onde os *ailerons*, que são superficies moveis acopladas nas extremidades da asa, quando acionadas controlam a rotação sobre esse eixo. Pitch é a rotação sobre o eixo lateral, que é paralelo às asas do aeromodelo. A atuação nos profundores (*elevators*) controla a rotação de pitch. Por fim, *Yaw* é a rotação no eixo vertical, controlada pelo leme (*rudder*). O encontro destes três eixos se faz no centro de gravidade do aeromodelo[F. D. Anderson, 2001].

[2]A mínima velocidade em que a asa pode produzir sustentação suficiente para suportar a aeronave

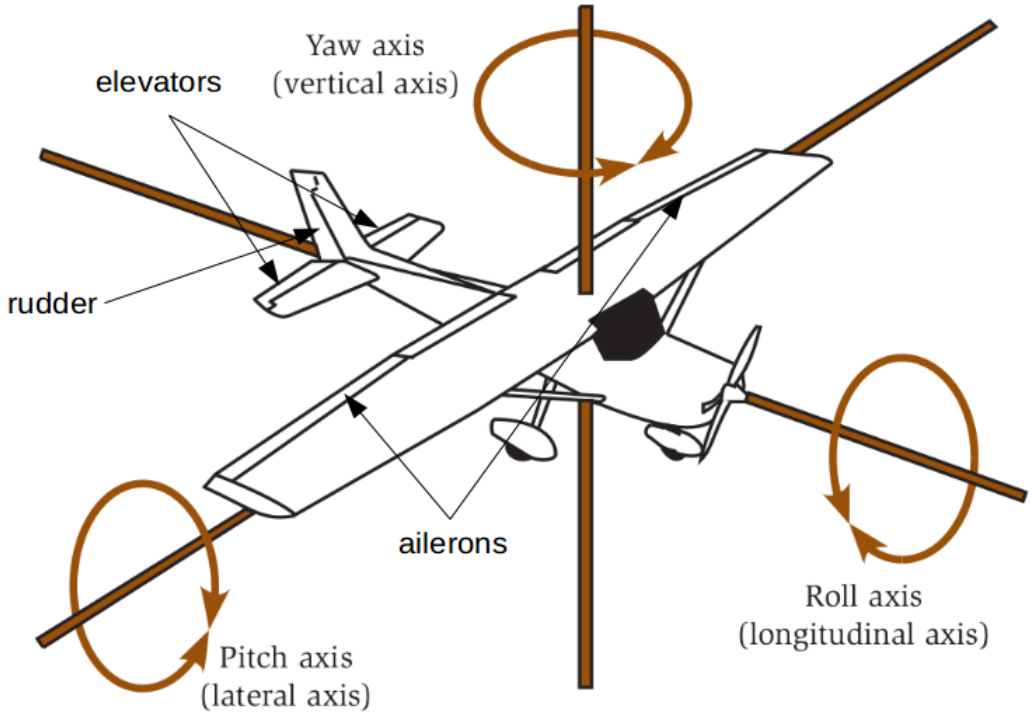


Figura 2.1: Eixos de atuação. Editado de [F. D. Anderson, 2001].

## 2.3 Considerações

Foram pesquisados artigos dentro do contexto e com o estudo formou-se uma base para o projeto e o desenvolvimento do que foi proposto neste trabalho.

No capítulo seguinte é detalhado o hardware desenvolvido para a construção da plataforma.

# Capítulo 3

# Hardware Utilizado

Neste capítulo é apresentado o projeto e o desenvolvimento do hardware necessário para a plataforma proposta, incluindo tanto os módulos prontos de prateleira como os circuitos projetados. A compatibilidade entre os componentes, a disponibilidade e os custos foram utilizados como critério de escolha para os componentes.

## 3.1 Arduino Mega 2560 e Nano

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre amplamente utilizada, baseada na família de microcontroladores de 8 bits da Atmega, com suporte de entrada e saída embutidos. Com ela é possível se comunicar com sensores e módulos através de seus pinos digitais e analógicos. Para trabalhar com esta plataforma é necessário carregar um programa na placa, que pode ser desenvolvido na IDE do Arduino. O programa é carregado via USB e sua linguagem é essencialmente C/C++["Arduino.cc", 2016] [Wikipédia, 2016a].

O Mega 2560, apresentado na Figura 3.1, é uma versão do Arduino baseada no microcontrolador ATmega2560 de 8-bits, frequência de 16 MHz, memória flash de 256 KB e 8 KB de SRAM. Este Arduino possui 54 pinos digitais para I/O, 16 pinos para entradas analógicas e 4 portas para comunicação serial ["Arduino.cc", 2016]. Já o Arduino Nano, apresentado na Figura 3.2, é uma versão de tamanho reduzido baseado no microcontrolador ATmega328 com frequência de 16 MHz, memória flash de 32 KB e 1 KB de SRAM. O Nano possui 14 pinos digitais e 8 analógicos.

O Arduino Mega foi escolhido para este projeto por prover portas e pinos de comunicação necessários para a integração dos componentes e também por ter capacidade de processamento suficiente para a execução de um sistema de navegação.

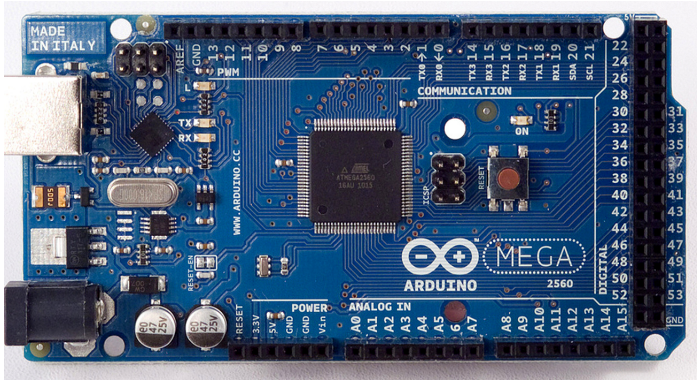


Figura 3.1: Arduino Mega 2560. [Arduino, 2016a]

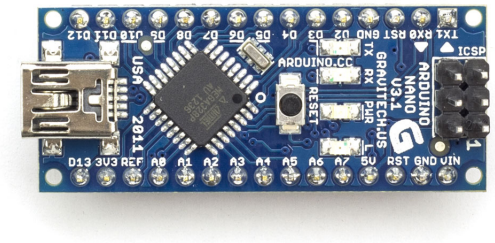


Figura 3.2: Arduino Nano. [Arduino, 2016b]

## 3.2 Raspberry Pi 2

O Raspberry Pi é uma plataforma livre baseada na família de processadores ARM, 32 e 64 bits, desenvolvida pela fundação inglesa Raspberry Pi, com dimensões pequenas e de baixo custo, com tamanho semelhante ao de um cartão de crédito e possui uma porta HDMI para conexão um monitor, uma porta ethernet para conectá-lo a redes de computadores e quatro portas USB para periféricos, como teclado e mouse[Wikipédia, 2016b].

O Raspberry Pi pode rodar algumas distribuições de SO, como Debian e Windows, basta carregar a imagem do SO escolhida no cartão SD que vai na placa. Esta placa, assim como o Arduino, possui pinos que podem interagir com sensores e módulos externos. Em termos de processamento o Raspberry Pi 2 tem uma grande vantagem, em relação ao Raspberry Pi, ARM quad-core de 900Mhz e 1GB de memoria RAM [Raspberry.org, 2016], enquanto o antecessor possuía um ARM single-core de 700Mhz e 512MB de memoria RAM, ambos processadores de 32 bits.

Esta plataforma foi utilizada por ter um baixo consumo de energia e por rodar distribuições compatíveis com o sistema de telemetria.

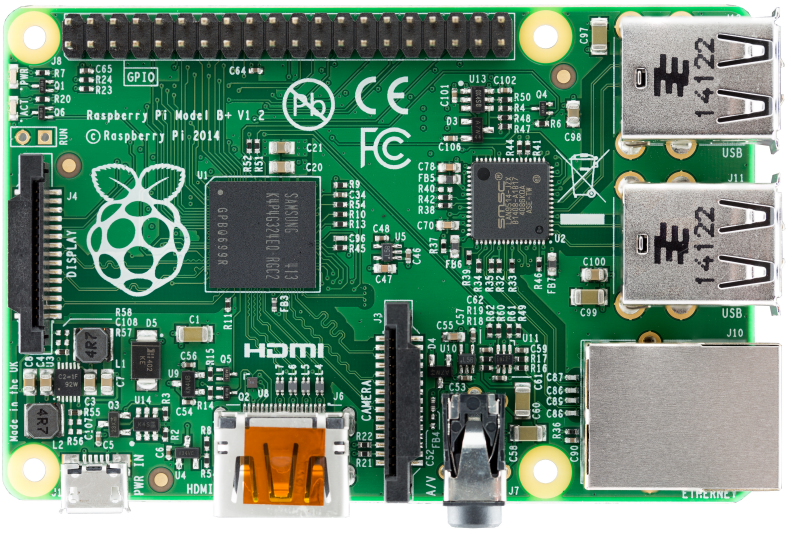


Figura 3.3: Raspberry Pi 2. [Raspberry, 2016]

## 3.3 XBee Pro S2

O Xbee Pro S2 é um módulo de rádio para comunicação sem fio entre dispositivos, com alcance de aproximadamente 1,6 quilômetros. Este módulo usa o protocolo IEEE 802.15.4 e é projetado para aplicações de alto rendimento que requerem baixa latência e tempo de comunicação previsível [XBee, 2016b].

Este módulo foi utilizado neste trabalho em conjunto com o Arduino e o Raspberry Pi, possibilitando a comunicação com o aeromodelo em voo para telemetria.


Figura 3.4: XBee, Módulo de Rádio. [XBee, 2016a]

## 3.4 EM-411

EM-411 é um módulo GPS (Global Position System) que pode ser integrado com o Arduino. Com ele é possível obter latitude, longitude, velocidade, curso e altitude[EM-411, 2016b].


Figura 3.5: EM-411, Módulo GPS. [EM-411, 2016a]

O GPS obtém a localização através da triangulação de satélites. São necessários pelo menos três satélites para apontar a posição e mais um para fornecer a altitude. Cada satélite se comunica com o módulo mapeando os tempos de resposta. Com este tempo é possível calcular a distância do módulo em relação a cada satélite e assim fazer a triangulação que aponta a sua posição[Garrett, 2011]. O módulo EM-411 possui precisão de cinco a dez metros[EM-411, 2016b].

## 3.5 GY-521 MPU6050

O MPU-6050 é um sensor de baixo consumo e baixo custo com alta performance, que combina acelerômetro e giroscópio na mesma placa. Com ele é possível obter os ângulos em que se encontram os três eixos do aeromodelo (Picth, Roll e Yaw) [MPU6050, 2016b].

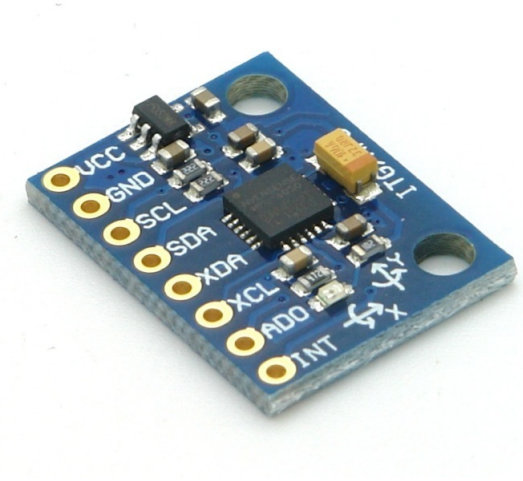


Figura 3.6: GY-521 MPU6050, Giroscópio e Acelerômetro. [MPU6050, 2016a]

## 3.6 HMC5883L

HMC5883L é um sensor magnético com interface digital que funciona como uma bússola [HMC5883L, 2016]. Este sensor fornece a direção em graus correspondente a qual o sensor está apontado.

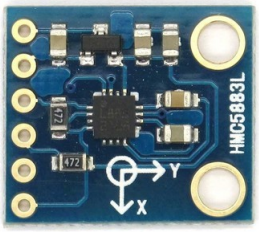


Figura 3.7: HMC5883L, Bussola digital. [HMC5883L, 2016]

## 3.7 Servo Motor

O Servo Motor é um dispositivo elétrico capaz de mover ou rotacionar um objeto com precisão. O movimento do Servo Motor pode variar de 0 ° a 180 ° ou 0 ° a 360 °, dependendo da especificação do dispositivo. Uma característica importante é o fato dos servos motores possuírem grande torque, tornando-os bastante utilizados em helicópteros e aeromodelos rádio controláveis, robôs, máquinas, entre outros.[Apoorve, 2016]


Figura 3.8: Servo Motor. [Apoorve, 2016]

## 3.8 Rádio Transmissor e Receptor

O Rádio Transmissor é o controle manual do aeromodelo. Para esse trabalho foi utilizado o rádio Turnigy 9x, apresentado na Figura 3.9. Este rádio possui um receptor que trabalha na frequência idêntica a do rádio para comunicação. Para controlar o aeromodelo manualmente os servos motores são conectados ao receptor.


Figura 3.9: Rádio Transmissor e Receptor. [robocore, 2016]

## 3.9 Circuito de chaveamento auto/manual

O circuito de chaveamento auto/manual foi desenvolvido neste trabalho. O propósito é fazer o chaveamento de comando do aeromodelo entre o rádio e o sistema autônomo, definindo se o Arduino ou o rádio aciona os servos motores em modo automático ou manual, respectivamente. Desta forma, a segurança do voo é reforçada, pois este circuito possibilita o retorno ao comando manual quando é identificada a instabilidade do sistema autônomo e também possibilita a programação de uma ação padrão em caso de perda de sinal de rádio, como por exemplo, retornar ao ponto de decolagem ou à uma posição dentro de um raio de operação estabelecido. O chaveamento é realizado por um canal do rádio controle. O funcionamento deste circuito é crítico para este projeto, pois o sistema é acionado em pleno voo.

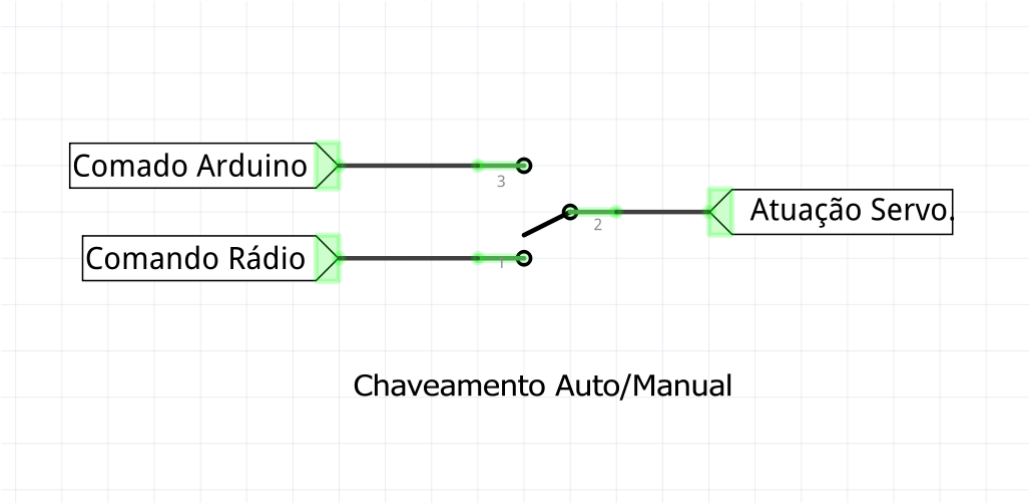


Figura 3.10: Chaveamento auto/manual. Este circuito permite escolher qual sinal de comando é transmitido a um servomotor, o proveniente do Arduino ou do receptor de rádio

Para a construção deste circuito foi necessário entender em detalhes o funcionamento da comunicação entre o rádio transmissor e seu receptor. O rádio controle emite um sinal codificado em PPM (*Pulse Position Modulation*), que é uma modulação da posição de pulso, implementada com a variação do intervalo entre pulsos, provocada pela variação do tempo do pulso ativo. Assim, a largura de cada pulso varia de acordo com o comando dado no rádio e é diretamente proporcional ao valor deste comando. Nos casos binários, para valor "zero"o pulso tem duração de 1ms e para o valor "um"duração de 2ms. Na Figura 3.11 é possível visualizar o pulso com valor mínimo e na Figura 3.12 com valor máximo. O rádio controle possui oito canais, sendo que quatro deles têm ganho ajustável entre "zero"e "um"os outros quatro são binários, assumindo os valores mínimo ("zero") e máximo ("um"). No circuito de chaveamento é utilizado um canal binário para a realização da troca de comando do sistema entre o Arduino e o rádio controle.

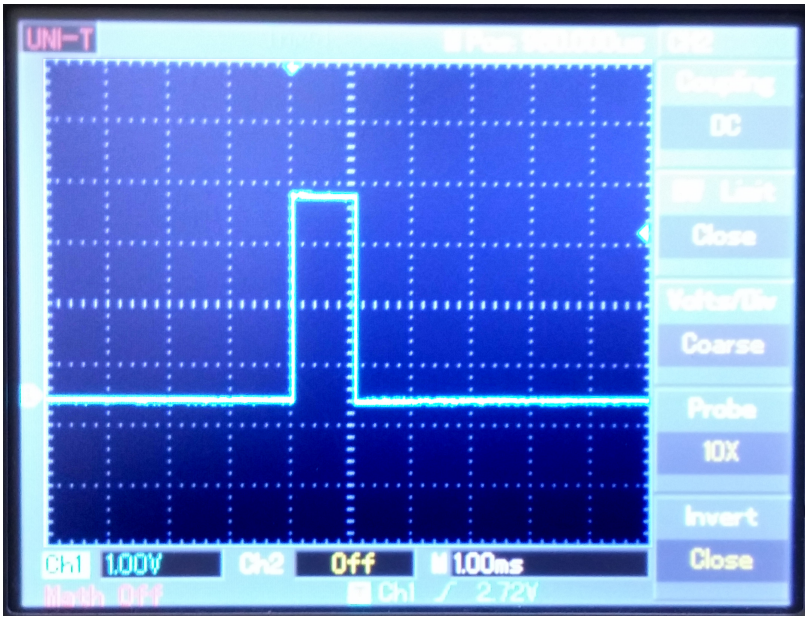


Figura 3.11: Sinal PPM com valor "zero", largura de pulso 1ms.

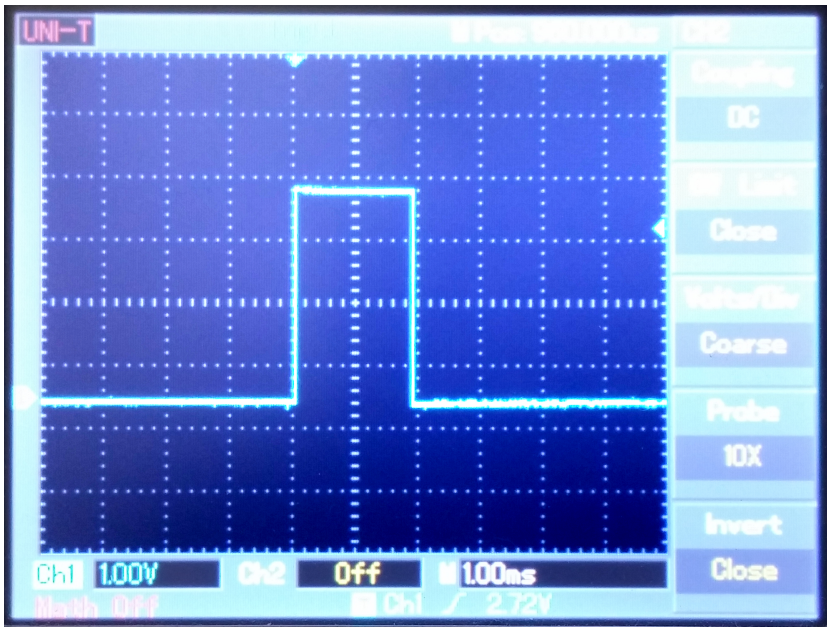


Figura 3.12: Sinal PPM com valor "um", largura de pulso 2ms.

Com o sinal de chaveamento auto/manual enviado pelo canal escolhido do rádio, duas opções para o desenvolvimento do circuito são analisadas. Uma delas, é receber este sinal no Arduino e chavear o sistema por software. A outra, ter um circuito independente para esta manobra. Por software no Arduino a interpretação do sinal do canal de chaveamento poderia ser feita de forma direta com a biblioteca *Servo.h* que já vem incorporada à IDE do Arduino. A biblioteca traduz o sinal recebido para um valor entre zero e cento e oitenta através da função *read()* e bastaria um desvio condicional para selecionar o controle desejado. Entretanto, a opção de fazer o circuito independente foi escolhida pelo fato de ser desejável atuação independente do estado do Arduino.

O circuito de chaveamento auto/manual deve reconhecer a duração do sinal recebido pelo rádio controle, distinguindo se está em uma posição, "auto", ou na outra "manual". Arbitrariamente para "auto"foi escolhido o valor máximo, pulso com largura de 2ms, e para "manual"o valor mínimo, 1ms.

O sinal recebido é dirigido a um integrador, construído com um amplificador operacional, LM358, conforme esquema na Figura 3.13. Este, ao integrar o sinal, transforma um sinal constante em uma rampa, com inclinação dada pelos valores de $R_{in}$ e $C$, conforme Equação 3.1.

$$V_{out} = -\frac{1}{R_{in}C}\int_0^t V_{in}dt = -\int_0^t V_{in}\frac{dt}{R_{in}C} \tag{3.1}$$

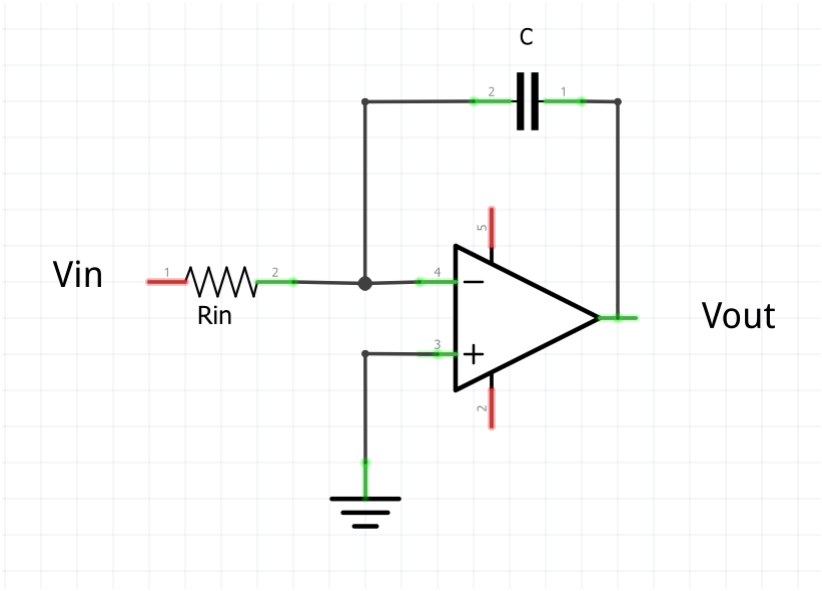


Figura 3.13: Esquema do integrador.

A rampa tem sinal da inclinação negativo, o que implica em que com o passar do tempo o valor da saída do integrador tem a tensão sendo reduzida, quando um valor constante positivo é apresentado na entrada. Assim, em 1ms o valor da saída do integrador varia de 5.44V a 3.12V, variando -2.32V em 1.08ms, conforme Figura 3.14. Quando o pulso tem 2ms de largura, a integração ocorre pelo dobro do tempo do caso anterior, a saída do integrador atinge 1.92V, variando 3.52V em 1.92ms, conforme Figura 3.15. O valor de saída do integrador na ausência de pulso tem o offset de 3.64V. Esses valores correspondem a uma variação de aproximadamente -2,1V/ms na saída do integrador na presença de pulso na entrada.

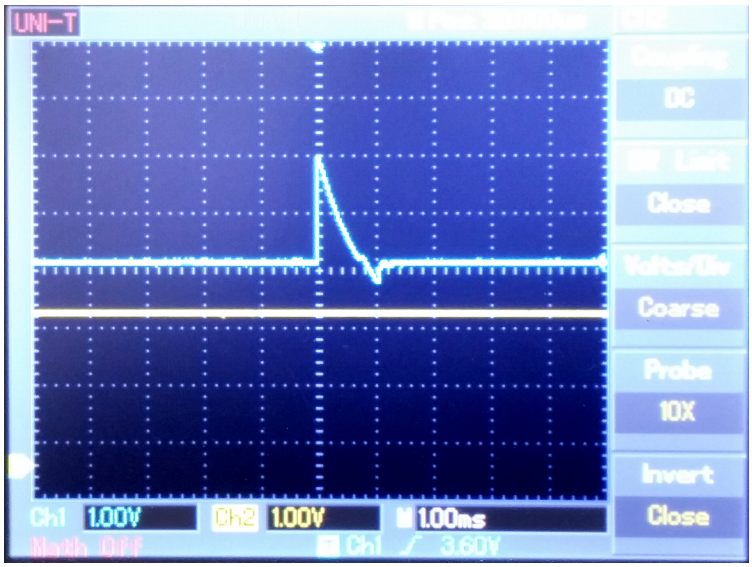


Figura 3.14: Sinal de controle recebido através do rádio com pulsos com duração de 1ms, saída do integrador varia de 5.44V a 3.12V, ficando superior ao limiar de comparação estabelecido. Saída do integrador corresponde à linha superior na tela do osciloscópio mostrada nesta figura e o limiar de comparação corresponde à reta constante, linha inferior.


Figura 3.15: Sinal de controle recebido através do rádio com pulsos com duração de 2ms, saída do integrador varia de 5.44V a 1.92V, atingindo valor inferior ao limiar de comparação estabelecido, 2.68V. Saída do integrador corresponde à linha superior na tela do osciloscópio mostrada nesta figura e o limiar de comparação corresponde à reta constante, linha inferior.

A saída do integrador é direcionada à uma entrada de um comparador, construído com outro amplificador operacional LM358 [Instruments", 2016], conforme esquema na Figura 3.16. A outra entrada do comparador recebe um valor de tensão constante, ajustado para um valor intermediário entre o mínimo valor da saída do integrador no caso de pulso de 1ms e no caso de 2ms, definindo um limiar de comparação entre os dois casos. Quando o valor do integrador é superior a este limiar, a saída do comparador se encontra saturada no valor superior da tensão de alimentação positiva do amplificador operacional, cerca de 5 Volts. Quando o valor do integrador é inferior ao limiar, a saída do comparador se encontra saturada próximo ao valor inferior da alimentação negativa do amplificador operacional, no caso, cerca de zero Volts. Com isto, a saída do comparador é constante no valor máximo de saída, próximo a 5 Volts, quando o pulso tem duração de 1ms, e assume valor mínimo nos momentos em que o valor de saída do integrador é menor que o limiar de comparação, o que ocorre nos casos de pulso com duração de 2ms, conforme ilustrado na Figura 3.17. O trimpot, que é um resistor variável, foi colocado para permitir ajuste preciso do limiar de comparação, compensando tolerâncias nos valores e parâmetros dos demais componentes.

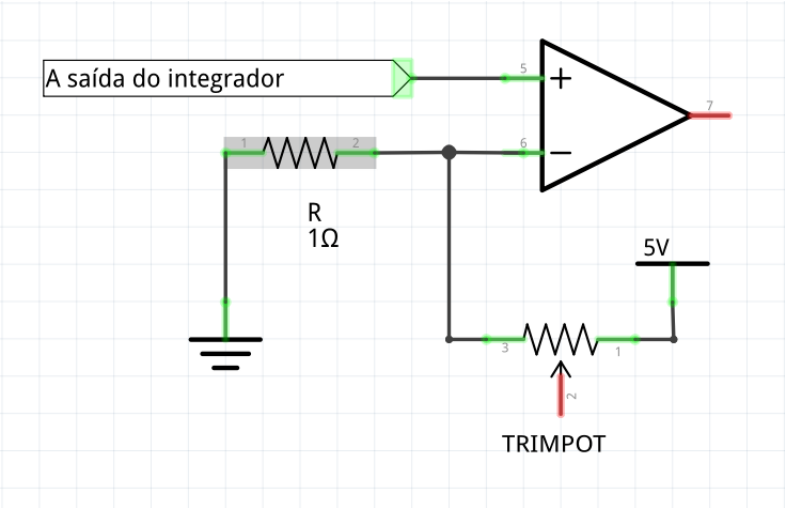


Figura 3.16: Esquema do comparador.

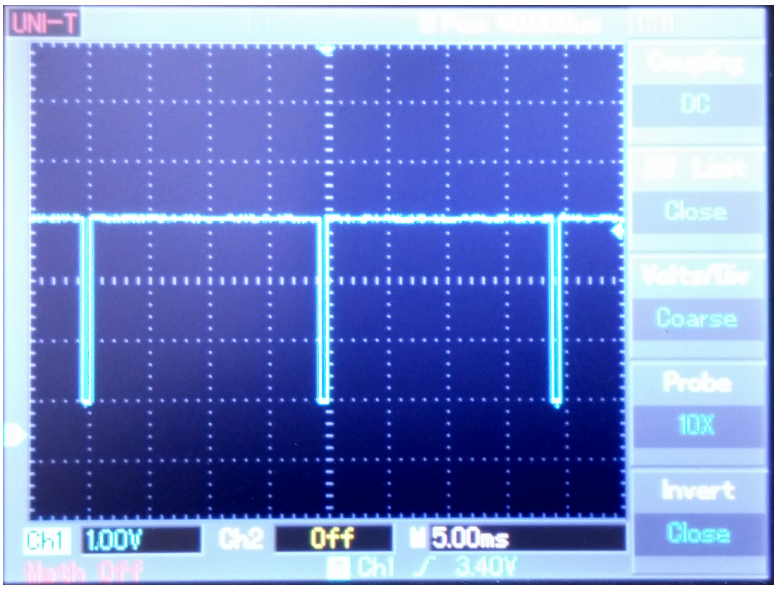


Figura 3.17: Sinal de saída do comparador, com valor próximo a 0 Volts quando o limiar de comparação é ultrapassado, indicando pulso de 2ms na etrada do integrador. Quando o pulso possui 1ms de largura, o limiar de comparação não é atingido, ficando o valor de saída do comparador constante próximo ao valor da alimentação positiva do amplificador operacional, no caso, 5 Volts.

Então, a presença de pulsos negativos na saída do comparador indica pulsos de comando recebidos pelo rádio na ordem de 2ms de largura. O que é desejado é um sinal que assuma valor lógico "zero"em uma condição, 1ms, e valor lógico "1"em outra, 2ms, podendo ser utilizado no controle da seleção de qual sinal de acionamento é encaminhado ao servomotor, o do Arduíno ou o do canal de rádio respectivo. Pulsos como os da Figura 3.17 não são adequados para isto, é necessário um sinal constante ("zero"ou "um") em cada situação.

A saída do comparador é direcionada ao acionamento de um temporizador monoestável retrigável, reponsável por prolongar os pulsos deste por 40 ms, de forma que

na ausência destes pulsos a saída do monoestável fica em nível lógico "zero"e na presença dos pulsos, que ocorrem a cada 20ms, a saída fica em nível lógico "um". Por ser retrigável, a contagem de 40 ms da saída do monoestável é reiniciada a cada pulso de entrada deste, resultando na saída constante em nível lógico "um"enquanto houver pulsos na entrada. O temporizador monoestável é implementado com um circuito integrado 74LS122 ["Motorola", 2016], com constante de tempo definida pelo resistor R8 e capacitor C2, conforme documentação do fabricante ["Motorola", 2016]. O circuito pode ser visto na Figura 3.18.

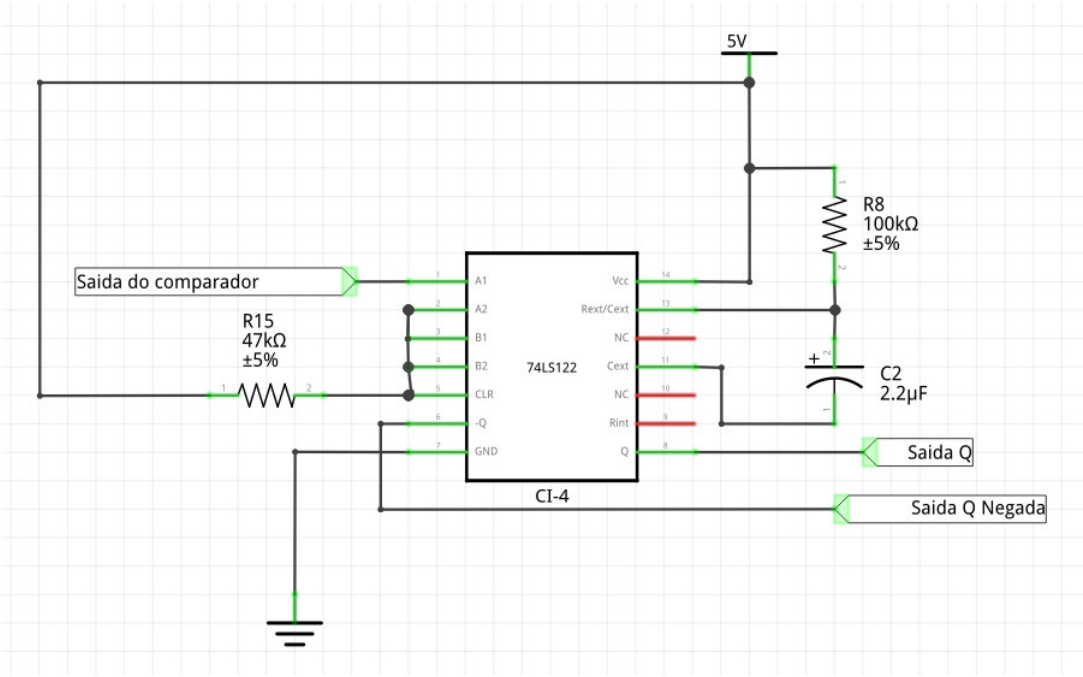


Figura 3.18: Temporizador monoestável retrigável.

Um multiplexador para a seleção entre o comando do Arduíno ou do rádio controle é implementado com portas NAND, com os sinais de saída do monoestável definindo qual destes comandos é enviado ao servomotor respectivo.

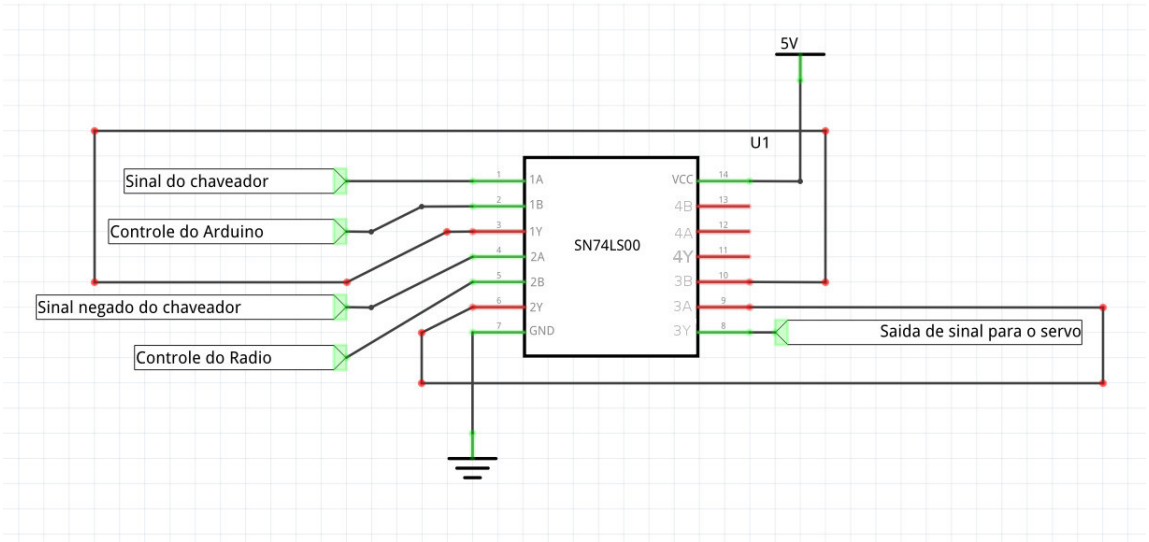


Figura 3.19: Modelo esquematico do multiplexador implementado com portas NAND

Finalmente, a Figura 3.20 mostra o esquema completo do circuito de chaveamento auto/manual desenvolvido.

As Figuras 3.21 e 3.22 mostram o projeto e implementação do circuito em protoboard. A Figura 3.23 mostra circuito em placa de circuito impresso desenvolvida pelo autor;

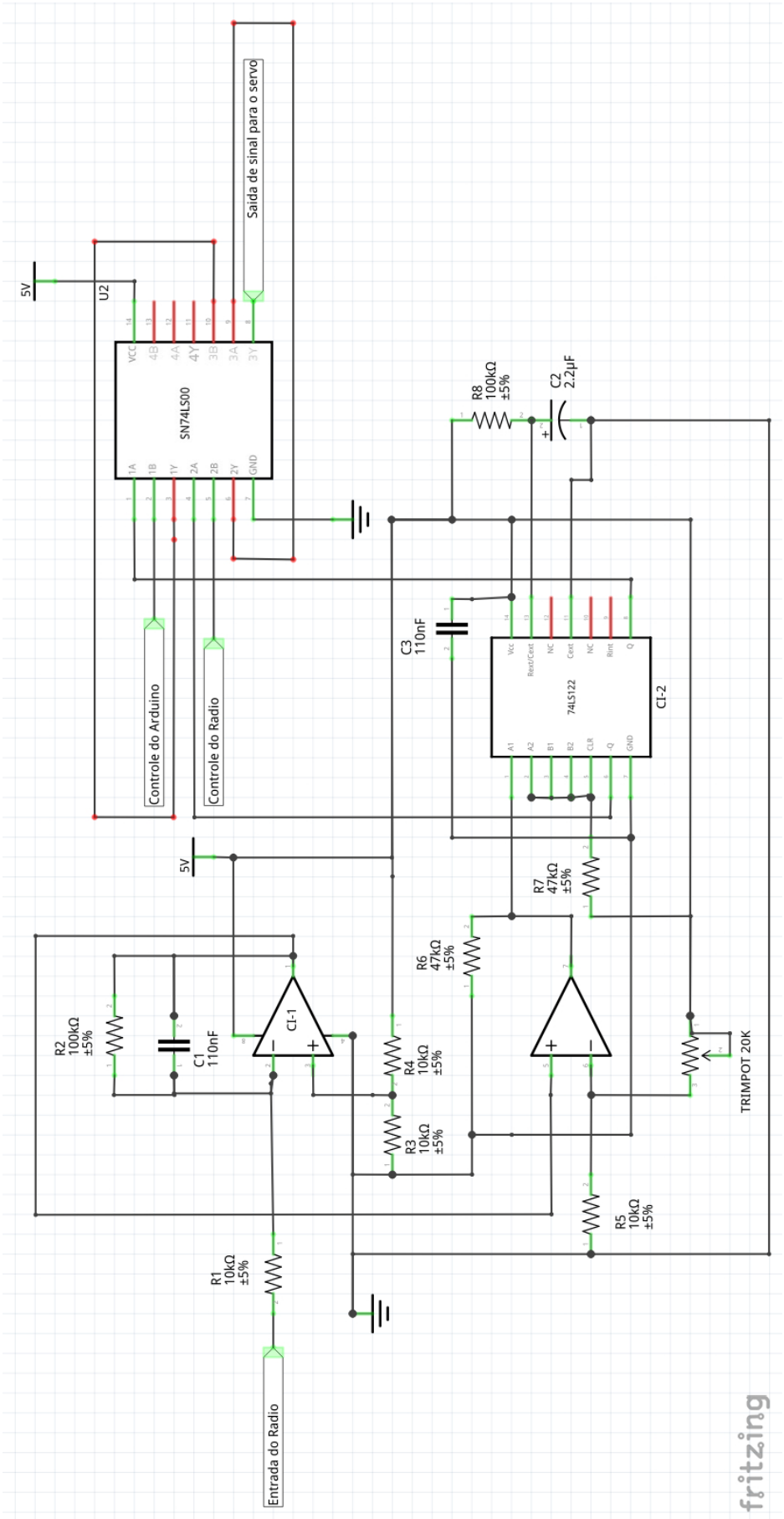


Figura 3.20: Esquematico chaveador

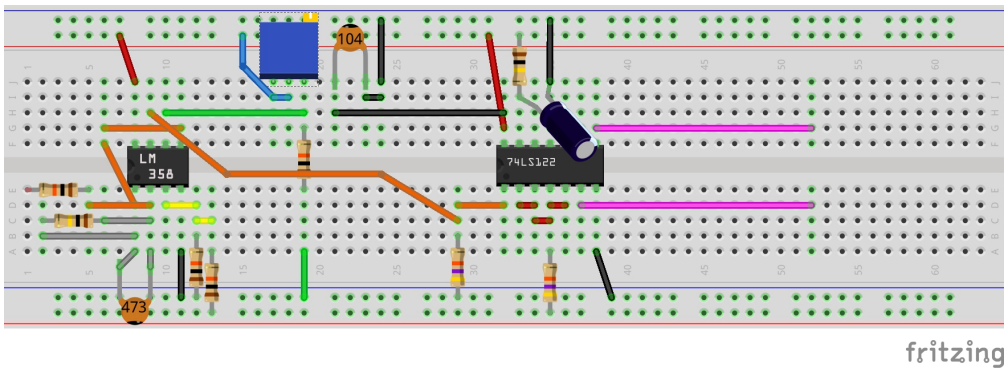


Figura 3.21: Projeto da implementação do circuito de chaveamento auto/manual em protoboard

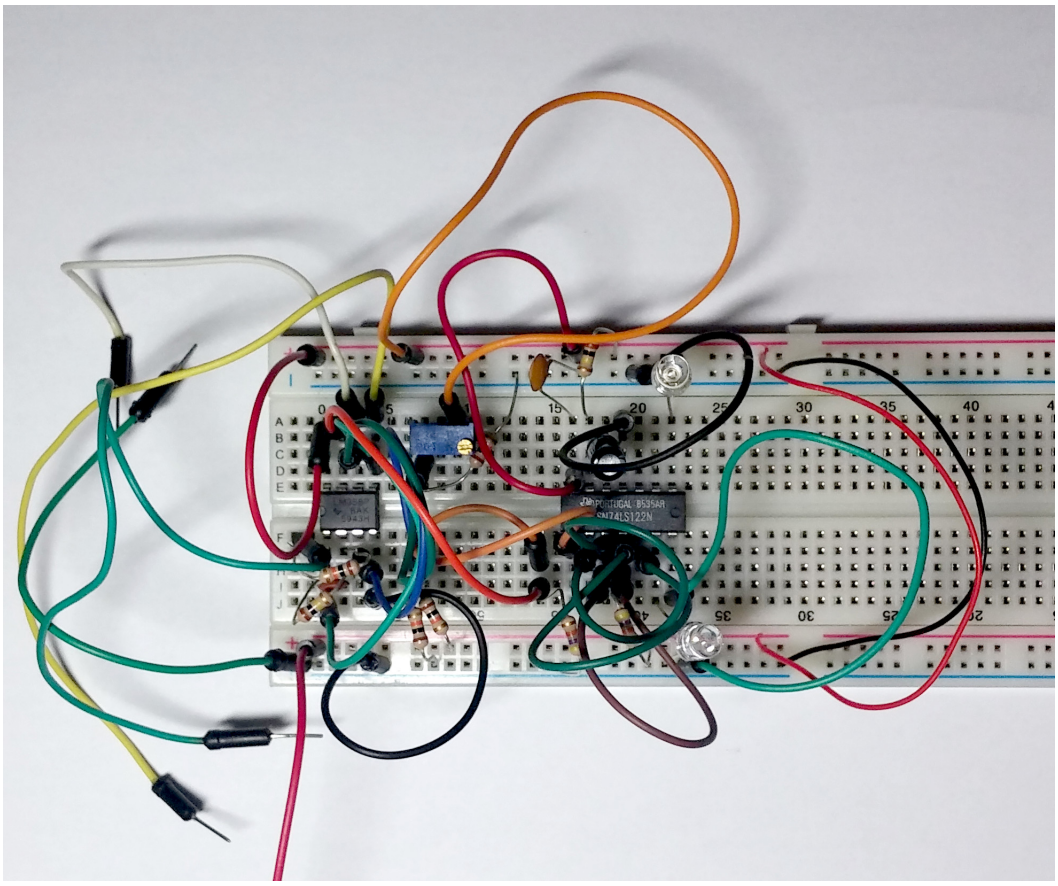

Figura 3.22: Implementação real do circuito de chaveamento auto/manual em protoboard

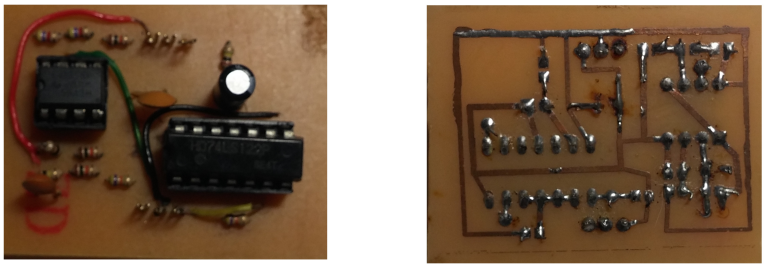

Figura 3.23

## 3.10 Aeromodelo

O aeromodelo utilizado como base para este trabalho é semelhante a um ultraleve, como pode ser visto na Figura 3.24. Este aeromodelo é de fabricação artesanal e possui 137 centímetros de envergadura, 90 centímetros de comprimento e 25 centímetros de altura. A massa do aeromodelo e seus componentes é de aproximadamente um quilograma.

Figura 3.24

## 3.11 Considerações

Foram pesquisados componentes, projetado e construído circuito que permite o desenvolvimento de uma plataforma dotada de sensores, telemetria e sistema de segurança baseado no chaveamento por comando de rádio entre os modos automático e manual de controle dos servos.

No capítulo seguinte é apresentado o desenvolvimento da plataforma baseada nestes recursos.

# Capítulo 4

# Implementação

Neste capítulo é apresentada a construção e a integração dos componentes de hardware para a construção da plataforma e também o desenvolvimento do sistema de telemetria. Um protótipo de sistema de navegação para esta plataforma também é proposto neste capítulo.

## 4.1 Integração do Hardware

Um dos desafios deste projeto foi o de unificar todos os componentes do hardware que controlam o aeromodelo, não só a parte física, como também sua integração por software para que funcionassem em conjunto. Dois fatores foram limitantes no que diz respeito ao hardware, o espaço que os componentes poderiam ocupar e o peso total desses componentes. A área em que o hardware ficaria acoplado no aeromodelo de referência tem dimensões muito próximas a dos componentes utilizados, o que obrigou alguns componentes a serem colocados em partes externas do aeromodelo. O peso adicional da plataforma influencia diretamente na velocidade de estol, exigindo maior aceleração do motor e mais uso da bateria. Para reduzir o peso adicional, a alimentação do sistema e a alimentação do motor foram compartilhadas, pois as baterias são os componentes mais pesados em questão de hardware. Esse compartilhamento reduz a autonomia de voo, mas em contrapartida, a velocidade de estol diminui e acaba equilibrando o consumo por exigir menos do motor e ainda melhorar a estabilidade de voo, uma vez que trabalha em velocidade mais baixa.

### 4.1.1 Bussola HMC5883L

A bússola digital HMC5883L foi integrada à plataforma através do barramento I2C[1] do Arduino. A biblioteca HMC5883L.h foi utilizada para comunicação com o sensor e obtenção de dados. Para maior precisão deste módulo, é necessário calibrá-lo com o ângulo de declinação magnética[2] do local em que ele vai ser utilizado.

[1] I2C é um protocolo de comunicação com modelagem mestre escravo usado para conectar um ou mais periféricos a um sistema embarcado

[2] Declinação magnética é a diferença em graus apontado pelo norte magnético terreno e o norte geográfico, determinado pelo eixo de rotação do planeta, que não tem obrigação de coincidir com a atração magnética.

### 4.1.2 Módulo GPS

O módulo GPS EM-411 funciona através da comunicação serial e foi conectado a uma das portas disponíveis do Arduino mega, integrando-se assim a plataforma. Uma vez conectado, o GPS comunica-se com o Arduino. A biblioteca *TinyGPS* foi utilizada para decodificar os dados recebidos e também para o cálculo de velocidade atual, distância entre pontos e outras informações a partir dos dados decodificados.

### 4.1.3 Xbee

O módulo de comunicação por rádio frequência Xbee, que é responsável pela telemetria do aeromodelo, foi conectado ao Arduino Mega via porta serial para ser integrado à plataforma. Para que dados sejam enviados pelo módulo, o Arduino é programado para escrever dados na porta em que o Xbee foi conectado, enquanto este transmite os dados para o receptor. Para ocorrer comunicação, os dois Xbees utilizados foram pré-configurados com o software XTCU[3] que foi utilizado para fazer essa configuração e cada um foi configurado para enviar ao endereço do outro. Além da configuração de endereço de envio, para que uma rede utilizando Xbee funcione, um deles deve ser configurado como *coordinator* para inicialização da rede e o outro como *router*, que funciona como um dispositivo final. Ambos devem estar com a mesma taxa de atualização, assim como o Arduino deve escrever na porta serial com a mesma taxa.

### 4.1.4 Integrando o MPU6050 ao sistema

O sensor MPU6050 é um sensor que também utiliza o barramento I2C para comunicação. A biblioteca *mpu6050.h* foi utilizada para conexão e comunicação com este sensor. O MPU6050 também necessita de uma calibração para funcionar perfeitamente e esta é obtida executando um código de calibração no Arduino. Como a interface I2C permite conectar vários sensores nos mesmos pinos de comunicação, este sensor foi conectado juntamente com a bússola e nos testes iniciais a comunicação com os sensores funcionou de forma satisfatória. Uma placa de fenolite foi confeccionada para integração dos sensores e dois resistores foram incluídos para funcionarem como um filtro de ruídos na comunicação.

Durante os testes com o MPU6050, foi observado que o Arduino parava de processar o *sketch* (programa criado e carregado com o ambiente de desenvolvimento do Arduino) carregado. Inicialmente, foi cogitado que poderia ser um defeito no Arduino, mas com alterações no código foi possível constatar que adicionando um delay[4] durante a execução do programa ou escrevendo em alguma das portas seriais do Arduino com uma taxa de atualização inferior 115200 ele interrompe seu processamento após algum tempo, que não foi fixo durante os testes, variando de segundos a alguns minutos. Este problema foi investigado e um reporte de erro[5] foi encontrado com as mesmas características também identificadas por outros desenvolvedores.

Como as escritas realizadas nas portas seriais não poderiam ser descartadas, uma solução para contornar o problema foi utilizar outro Arduino para receber os dados do sensor. Como o espaço e peso estavam bastante limitados, um Arduino nano, que tem

[3]Software desenvolvido pela Digi para que desenvolvedores possam configurar modos de comunicação por rádio frequência fabricados por ela.

[4]Tempo definido por parâmetro em que o processador não realiza operações

[5]https://github.com/jrowberg/i2cdevlib/issues/252#issuecomment-231742469

dimensões reduzidas, foi utilizado para ligar o sensor. O Arduino nano foi programado para fazer a leitura e enviar os dados como um pacote na sua porta serial, que foi conectada a uma outra porta serial no Arduino mega. O código de comunicação do Arduino nano com o sensor não precisou ser modificado, mas a comunicação entre os Arduinos foi implementada em sua totalidade.

A comunicação dos Arduinos pelo lado do nano foi feita com a criação de um pacote de dados com tamanho de 13 bytes, que possui um preâmbulo fixo contendo o caractere '$', seguido dos ângulos de roll, pitch e yaw fornecidos pelo sensor. O caractere '$' ocupa um byte deste pacote por ser do tipo char, que no Arduino possui 1 byte e cada um dos ângulos ocupa 4 bytes por serem do tipo float, que no Arduino contém 4 bytes. Este pacote é enviado continuamente pela porta serial a cada leitura fornecida pelo sensor.

Pelo lado do Arduino mega foi implementada uma função que quando chamada realiza a leitura da porta serial conectada ao Arduino nano. Essa leitura ocorre por meio de um laço que realiza sucessivas leituras até encontrar o preâmbulo para alinhar um pacote e adquirir os dados do sensor.

Com este método não houve mais a interrupção do processamento do Arduino, porém os dados do sensor estavam com certo atraso. Isto ocorreu porque o Arduino tem um buffer para suas portas seriais com tamanho de 64 bytes que entrega os dados seguindo uma fila FIFO[6], e os dados que o sensor fornece possuem tamanho de 13 bytes, o que permite ter até 4 pacotes no buffer, com apenas o último do buffer atual. Para obter o dado mais atual, o tamanho desse buffer foi reduzido para 20 bytes, o que garante a leitura dos dados mais recentes do sensor.

A figura 4.1 apresenta as conexões entre o Arduino mega, nano e o sensor MPU6050.

[6]First In First Out, o primeiro a entrar é o primeiro a sair

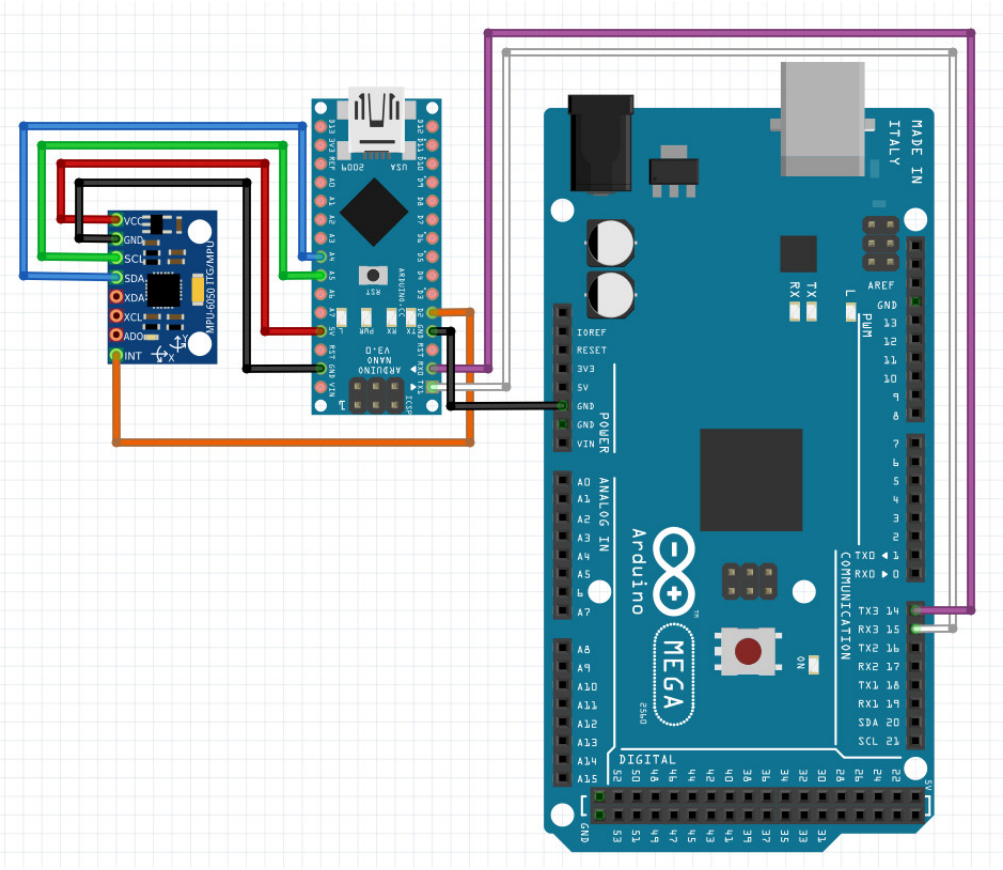


Figura 4.1: Integração entre MPU6050, Mega e Nano.

### 4.1.5 Servos

Os servos, como costumam ser denotados os servomotores, são responsáveis pelo controle da movimentação do aeromodelo em torno dos eixos de roll, pitch e yaw, conectados ao circuito de chaveamento. A biblioteca *servo.h* é utilizada para gerar os sinais PPM nos pinos do Arduino através da função *servo.write()*, que recebe como parâmetro um ângulo entre zero e cento e oitenta graus.

### 4.1.6 Shield para o arduino Mega

Com todos os sensores e módulos funcionais, incluí-los simultaneamente gerou um problema organizacional. A figura 4.2 exemplifica este problema.

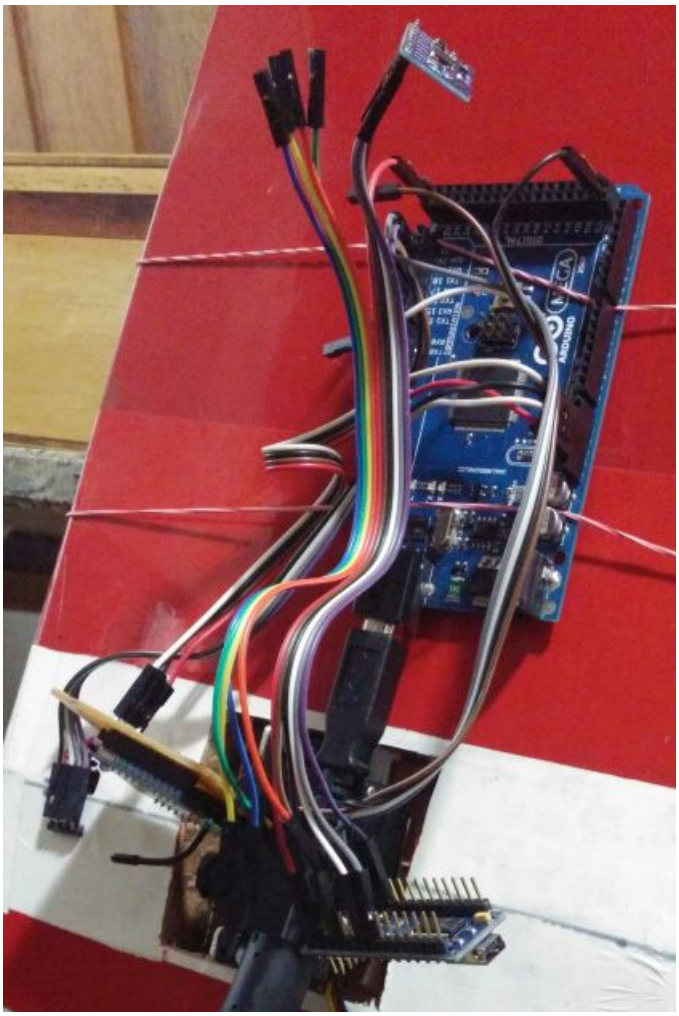

Figura 4.2: Projeto da placa do circuito de chaveamento

Para integrar todos os componentes de forma compacta e confiável, no sentido de reduzir o risco de mau contato em conexões, um shield [7] foi projetado e construído. O projeto deste shield foi feito utilizando o software de código aberto Fritzing[8]. Este software possibilita projetar hardware de forma relativamente simples e intuitiva. Suas bibliotecas incluem milhares de componentes eletrônicos e ainda permitem que se possa adicionar mais componentes caso eles não existam.

Além de integrar todos os sensores, este shield foi projetado para incluir o circuito chaveador ao hardware. A alimentação deste circuito foi integrada ao Arduino Mega. Todo o hardware que não foi acoplado diretamente sobre a placa tem um conector correspondente, para que se possa conectá-los com facilidade. A placa utilizada para construção, chamada de fenolite[9] tem apenas um lado com camada de cobre e por isso nem todas as trilhas que interligam os sensores são possíveis de serem projetadas sem que ocorra sobreposição. Por isso alguns componentes foram conectados com fios adicionais soldados na placa.

A figura 4.3 apresenta todas as ligações do hardware sem o usos do shield, com exceção das ligações do circuito de chaveamento que teve seus componentes omitidos nesta

[7]Nome dado a placas que encaixam nos pinos do Arduino ou de outra placa.

[8]http://fritzing.org

[9]Placa que possui uma ou duas camadas de cobre, que pode ser corroída para produzir trilhas e interligar componentes eletrônicos

imagem. O shield que foi projetado para reproduzir essas ligações na placa de fenolite e ser acoplado no Arduino Mega pode ser visto na figura 4.4.

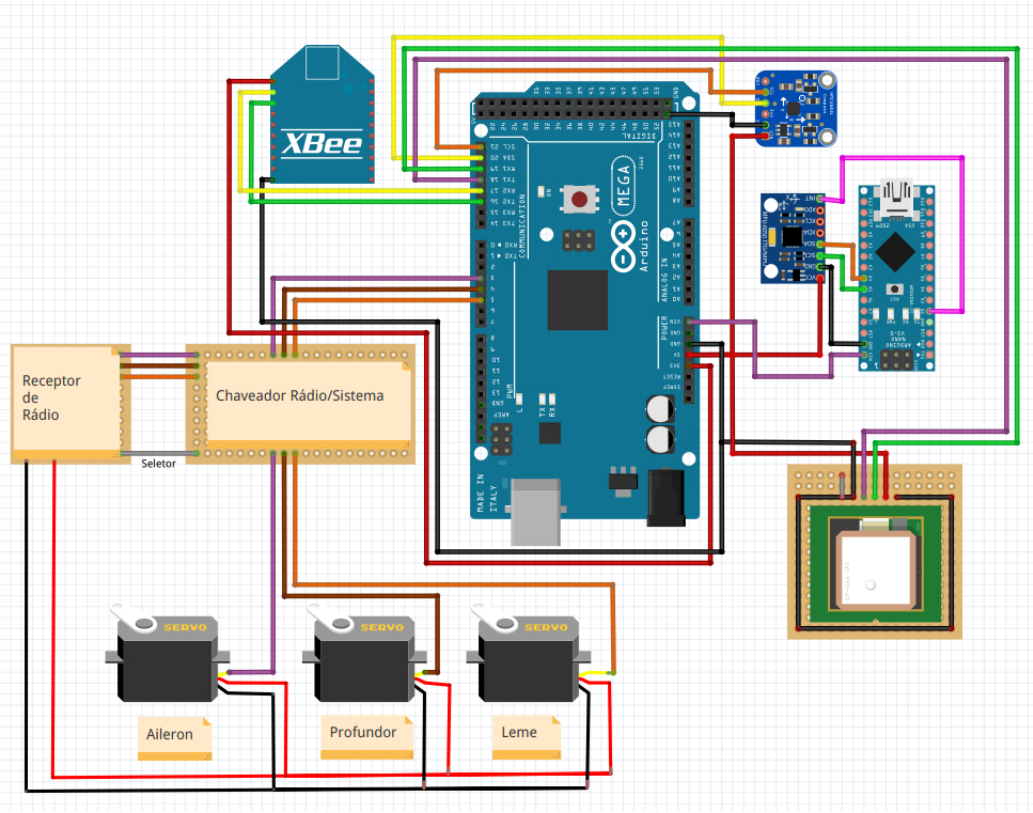


Figura 4.3: Conexões do Hardware sem shield.

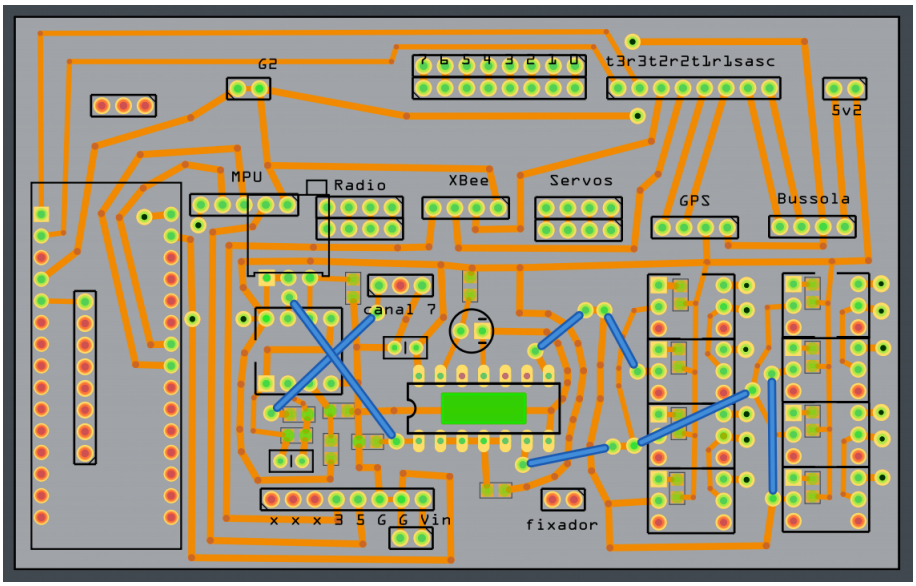


Figura 4.4: Projeto do shield.

Com o projeto pronto, o próximo passo foi transferir para a placa de fenolite as trilhas do circuito. Há várias formas de realizar este processo e a forma escolhida foi imprimir o arquivo gerado pelo Fritzng (figura 4.5 em papel fotográfico e transferir a tinta

deste papel para a placa através do aquecimento do papel em cima da placa com um ferro de passar roupa doméstico. A tinta do papel adere ao cobre da placa e assim ela pode ser imersa em percloreto de ferro, um componente químico que corrói o cobre não protegido pela tinta. Esse processo funciona apenas com impressoras a laser.

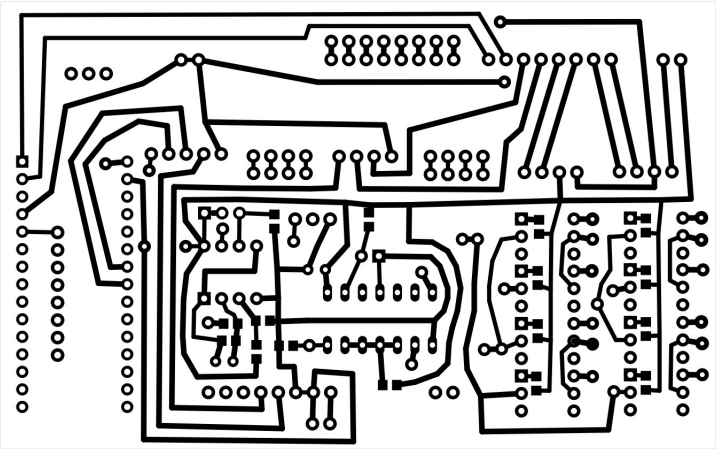

Figura 4.5: Molde para transferêcia.

O resultado da transferência do molde para a placa pode ser visto na figura 4.6. É importante observar as linhas que estão com retoques manuais. Isto ocorre porque no processo de transferência nem todas as trilhas aderem totalmente ao cobre da placa e isso implica em correções manuais para que o circuito interligue os componentes sem falha.

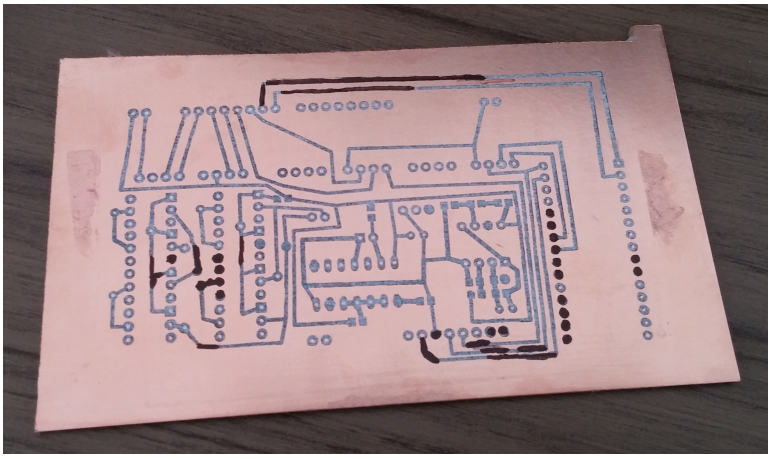

Figura 4.6: Molde transferido para a placa.

Após o processo de corrosão, a placa ficou apenas com as trilhas do molde. Na figura 4.6 a placa corroída e já perfurada para a colocação dos componentes eletrônicos pode ser vista.

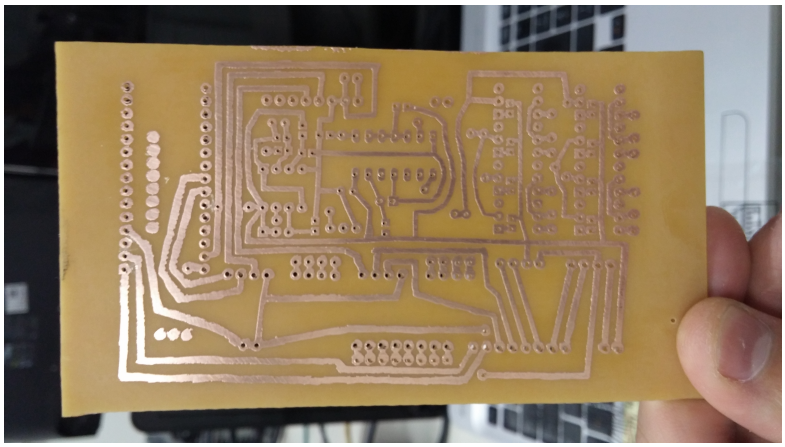

Figura 4.7: Placa corroída e perfurada.

Com todos os componentes soldados a placa foi concluída e as figuras 4.8, 4.9 e 4.10 apresentam o resultado de todo o processo.

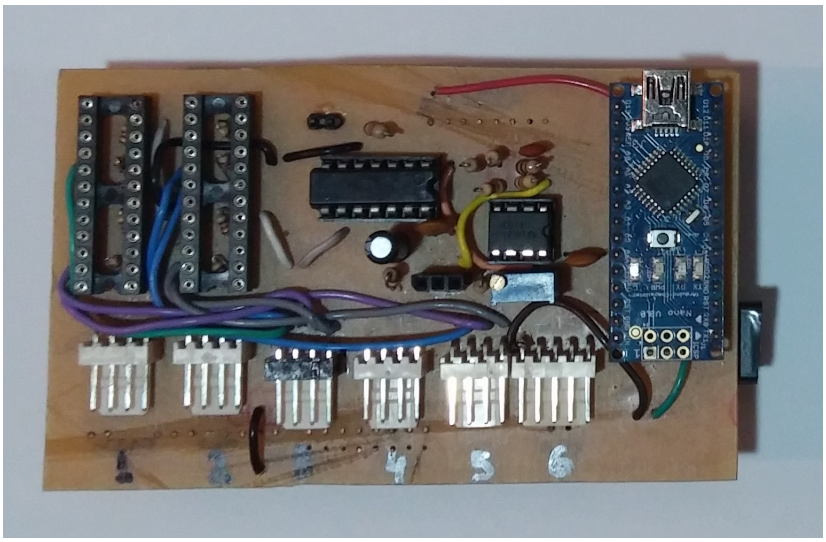

Figura 4.8: Placa com os componentes eletrônicos - visão superior

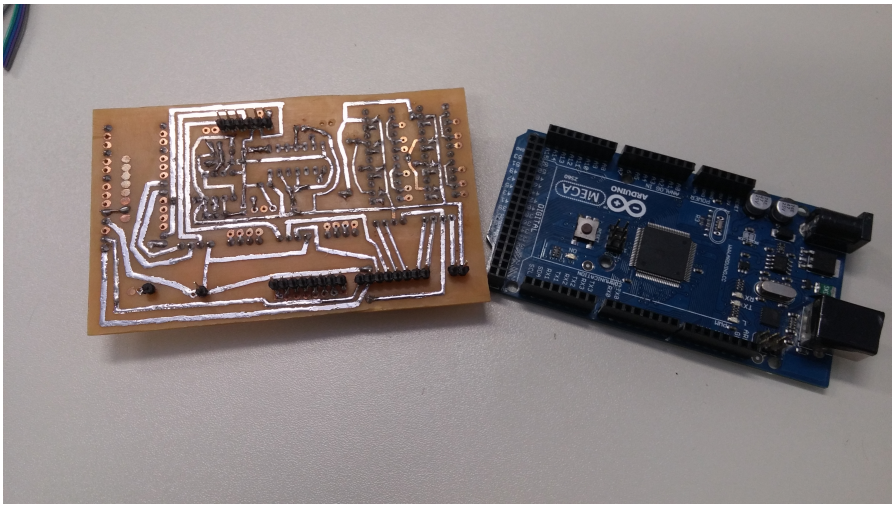

Figura 4.9: Placa com os componentes eletrônicos - visão inferior

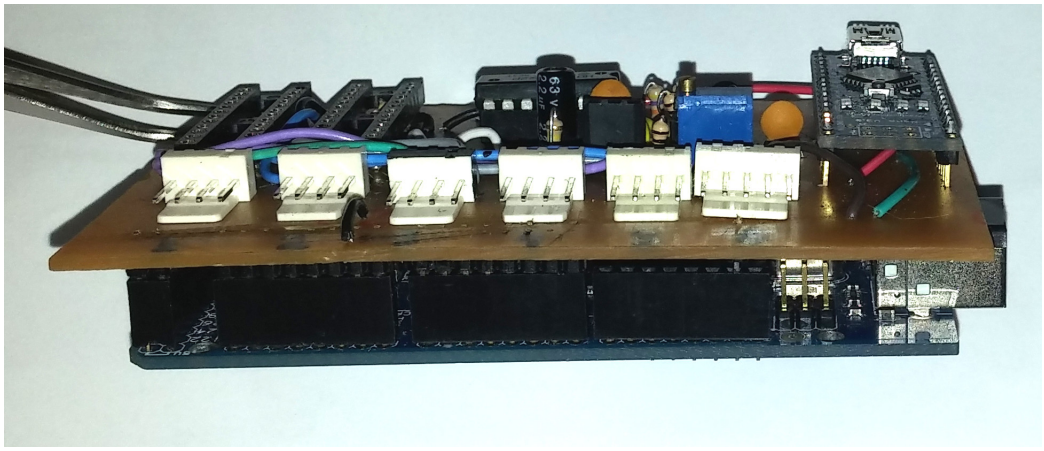

Figura 4.10: Shied conectado ao Arduino Mega.

O último passo de conclusão de hardware foi a integração dos componentes eletrônicos ao aeromodelo. As figuras 4.11, 4.12 e 4.13 mostram o hardware finalizado.

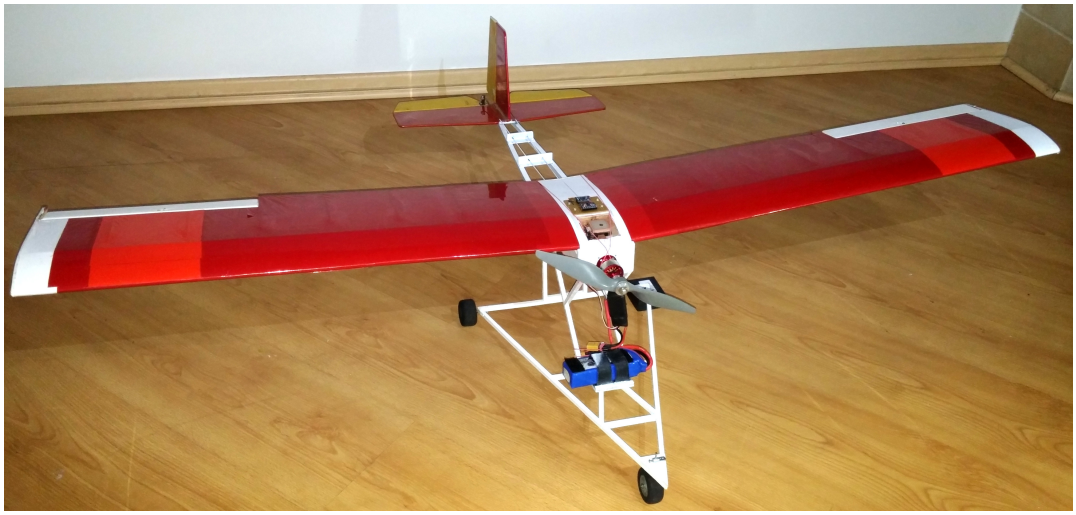

Figura 4.11: Visão frontal do aeromodelo em sua totalidade. Aqui pode-se notar que o centro da asa foi modificado para receber dos sensores.

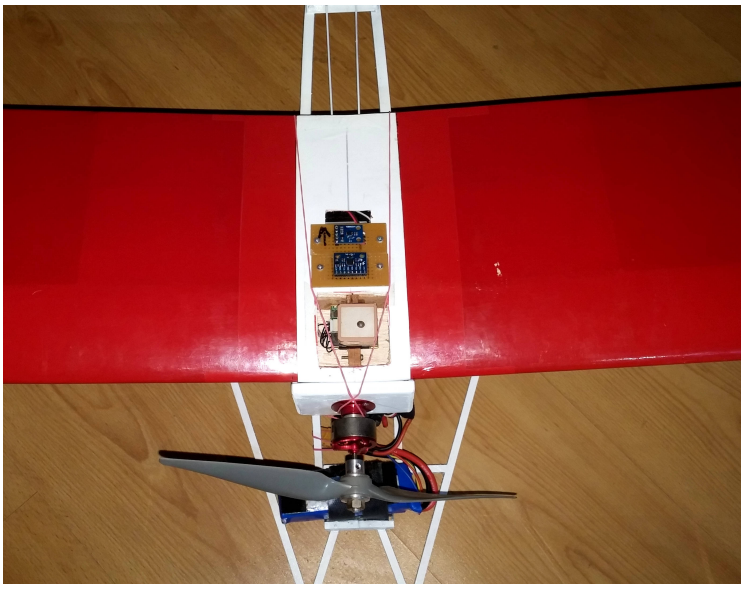

Figura 4.12: Imagem aproximada do local onde os sensores foram acoplados. Estes sensores foram posicionados o mais próximo possível do centro de gravidade do aeromodelo, para obtenção de dados precisos de posicionamento e localização.

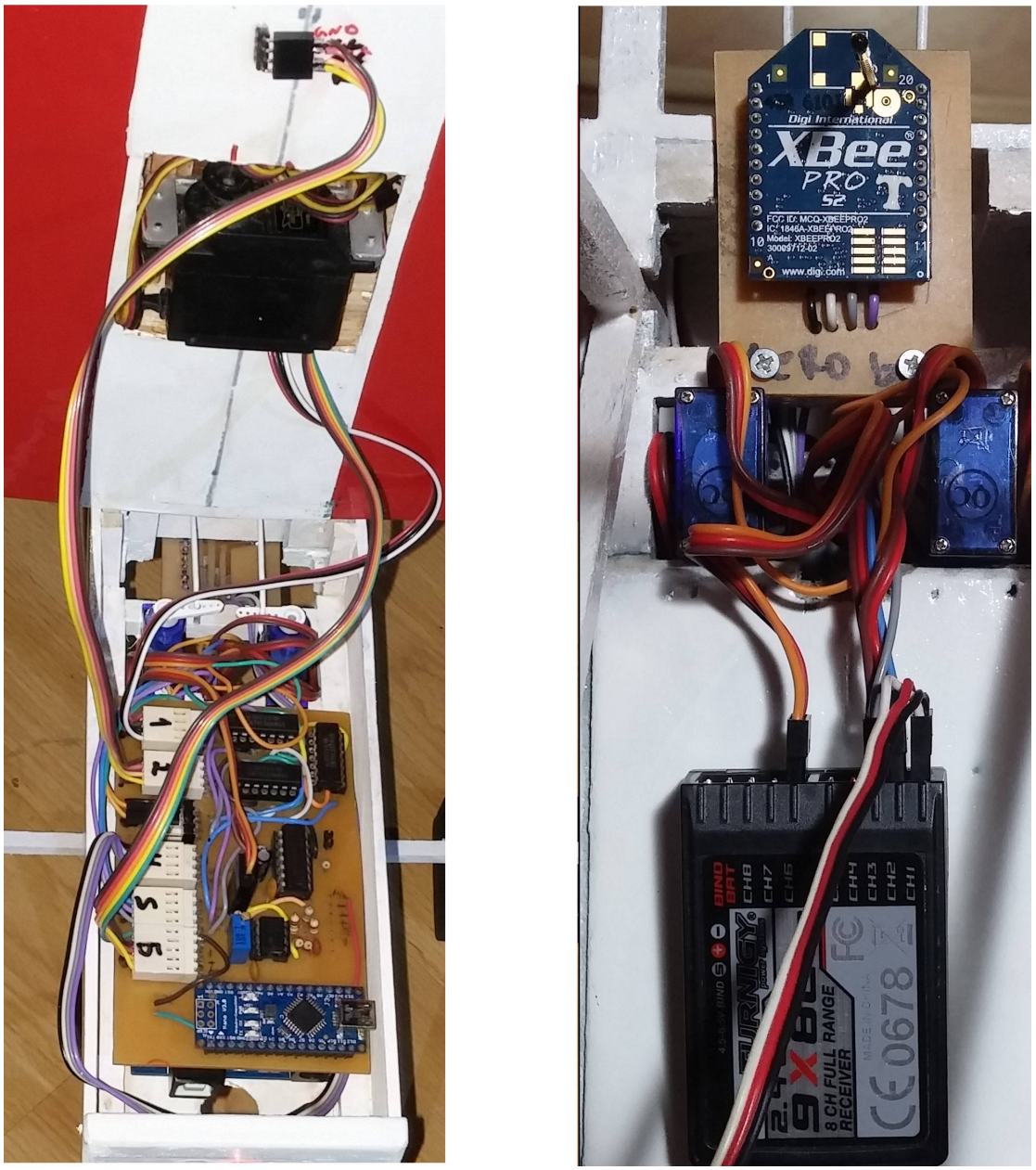


Figura 4.13: Visão da parte interna da carenagem do aeromodelo com a plataforma acoplada (imagem a esquerda), e da parte inferior externa da carenagem com o modulo de comunicação Xbee fixado e o receptor de rádio acoplado (imagem a direita).

## 4.2 Modelo de Sistema de Navegação Proposto para a Plataforma

Nesta seção é apresentada a proposta de um modelo de sistema de navegação simples para a plataforma desenvolvida, considerando os recursos disponíveis. Esse modelo segue a arquitetura da figura 4.14 que possui três controles responsáveis pelas ações do aeromodelo, que é o Controle de Roll, Pitch e Navegação. Neste modelo não foi proposto um controle para Yaw, a fim de simplificar a proposta. Mesmo sem esse controle é possível realizar manobras para que o aeromodelo convirja ao seu destino, porém é provável que

tenha uma velocidade de convergência ligeiramente menor. Os controles são alimentados pelos sensores e atuam nos servos para controlar o movimento do aeromodelo.

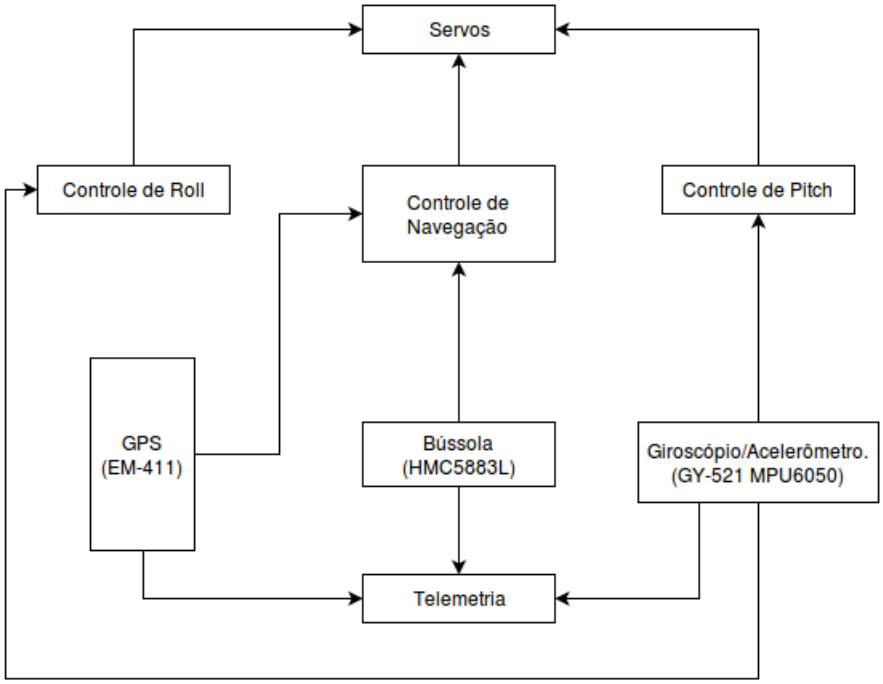


Figura 4.14: Arquitetura do Sistema de Navegação

O algoritmo do sistema é executado em um ciclo contínuo, cujo fluxo é mostrado na figura 4.15. Neste fluxograma é possível observar que quando o Controle de Roll ou o Controle de Pitch são acionados, o Controle de Navegação não é ativado na iteração corrente. Isto ocorre porque os controles de pitch e roll servem para manter o aeromodelo dentro de limites de operação e o controle de navegação acabaria interferindo no movimento de correção. Cada um dos três controles possui um controlador PID para atuar nas suas respectivas superfícies de comando. Nas subseções a seguir os controles são descritos com mais detalhes.

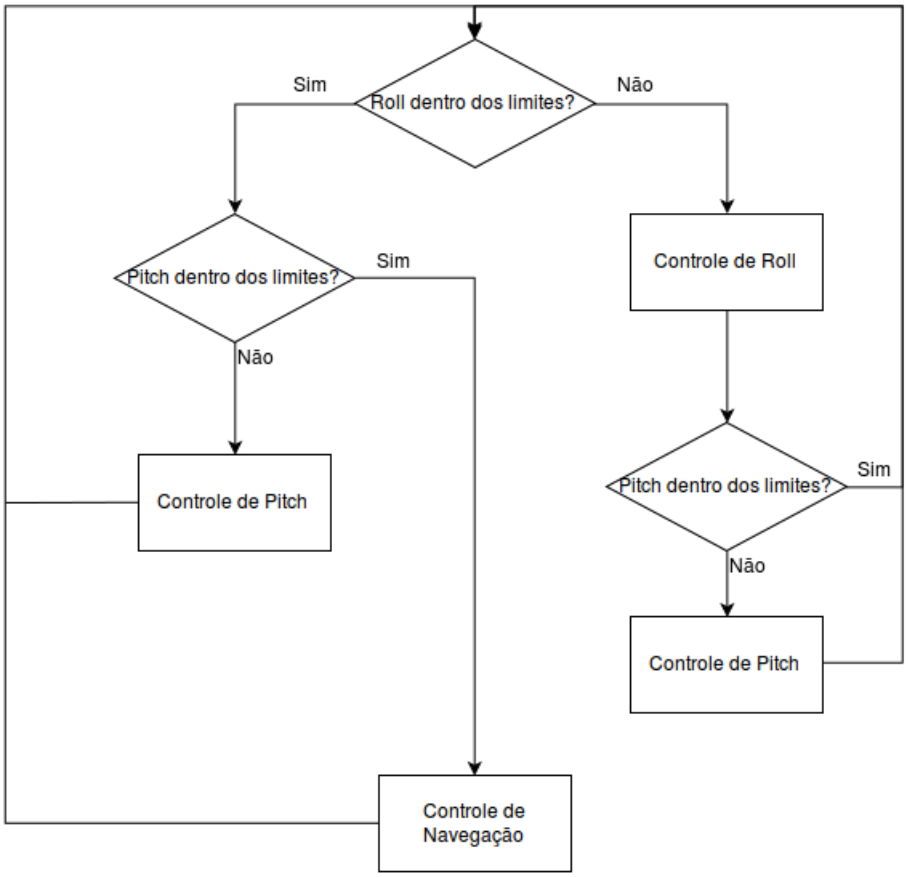


Figura 4.15: Fluxo do Sistema de Navegação.

### 4.2.1 Controle de Roll

O Controle de Roll tem a função de manter a inclinação sobre o eixo longitudinal dentro de dois limites estabelecidos, inferior e superior. Esse controle limita a atuação do Controle de Navegação em roll, para que o aeromodelo não realize curvas bruscas, o que poderia causar a instabilidade de voo e a necessidade de retomada de controle manual.

A cada iteração do ciclo o Controle de Roll realiza a leitura do sensor MPU6050 para obter a inclinação atual no eixo de roll. Esta inclinação fornecida pelo sensor pode variar de -90 ° a 90 °, como visto na figura 4.16, onde o lado da asa demarcado em vermelho é a referência de inclinação. Com o dado obtido do sensor verifica-se se o seu valor está dentro dos limites estabelecidos, caso não esteja, o algoritmo aciona o controlador PID para corrigi-lo. A entrada do controlador PID é o ângulo de roll atual e seu estado desejado é de 0 °. A saída do controlador de roll atua no servo do aileron. Aqui o leme também poderia ser utilizado, mas o aileron foi escolhido por ter uma resposta mais rápida na inclinação de roll, o que é desejável em uma situação de correção.

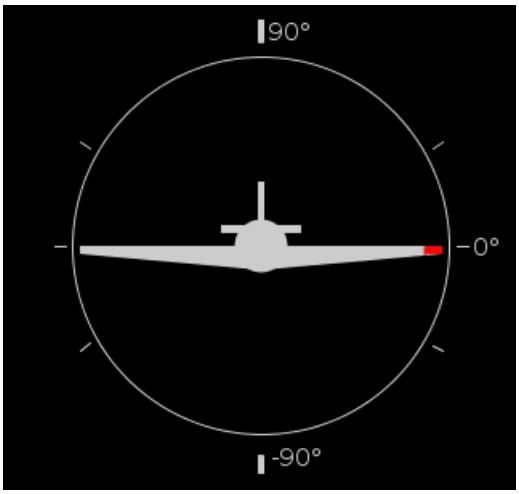


Figura 4.16: Comportamento de Roll.

### 4.2.2 Controle de Pitch

O controle de Pitch tem a função de manter a altitude do aeromodelo por meio da correção do ângulo de pitch. Este ângulo também é fornecido pelo sensor MPU6050 e atualizado em todas as iterações do ciclo. O valor deste ângulo também pode variar de -90 ° a 90 °, como é mostrado na figura 4.17. A demarcação em vermelho na frente do avião serve como referência de inclinação. Enquanto o ângulo de pitch estiver em zero, o aeromodelo irá manter sua altitude.

A cada iteração, o Controle de Pitch verifica se inclinação de pitch está dentro de um limite estabelecido, caso não esteja, o controlador PID é acionado para corrigir este ângulo. A entrada do controlador PID é o ângulo atual e o estado desejado é 0 °. A saída do controlador PID é direcionada ao servo que atua no profundor do aeromodelo, responsável movimentação do aeromodelo no eixo de pitch.

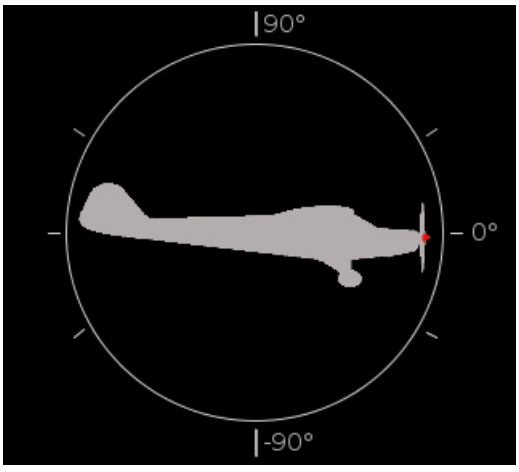


Figura 4.17: Comportamento de Pitch.

### 4.2.3 Controle de Navegação

O Controle de Navegação é responsável por direcionar o aeromodelo ao ponto programado. Para isso, ele trabalha com o curso atual, fornecido pelo sensor HMC5883L e

o curso desejado, fornecido pela função *course_to()* da biblioteca *TinyGPS*. Esta função calcula o curso para o ponto desejado utilizando a posição atual e a posição alvo. O Controle de Navegação calcula a diferença destes cursos e verifica se é necessário corrigir a trajetória do aeromodelo para a direita ou esquerda para alcançar o ponto desejado. Para corrigir a trajetória o controle aciona o controlador PID que atua no leme do aeromodelo, passando como parâmetro a direção para correção e os valores dos cursos. Tanto leme quanto aileron poderiam ter sido utilizados para realização de curvas com o aeromodelo, mas o leme foi escolhido por ter atuação mais lenta e suavizar a manobra.

## 4.3 Telemetria

A telemetria é um mecanismo de visualização em tempo real do estado em que o aeromodelo se encontra. Seu funcionamento é fundamental para que a execução de um sistema de navegação possa ser acompanhada. A figura 4.18 apresenta o modelo de contexto da comunicação entre aeromodelo e solo.

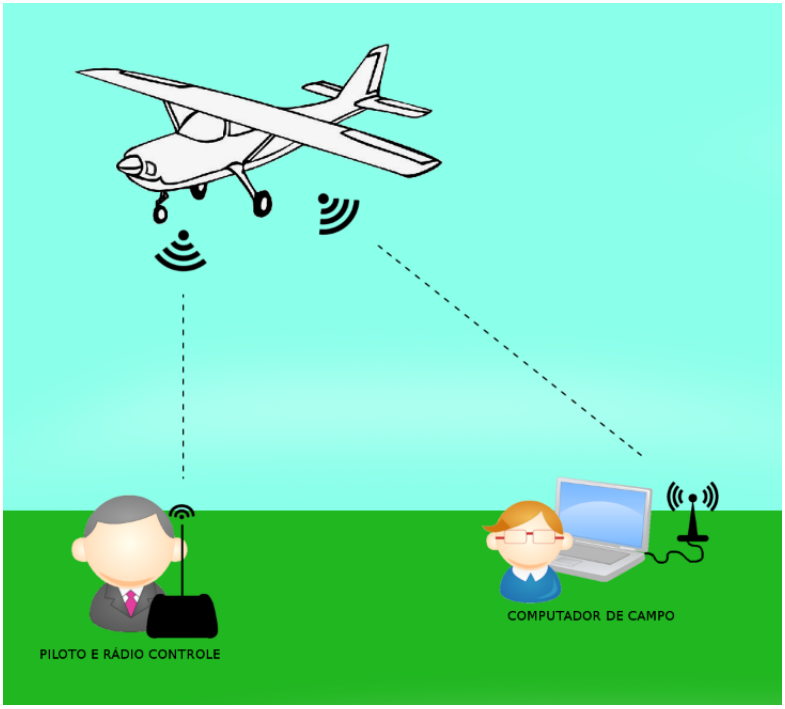


Figura 4.18: Comunicação aeromodelo e solo

Para a telemetria do aeromodelo foi desenvolvida uma aplicação em Processing[10], que é um software de código aberto que possui uma IDE de desenvolvimento para aplicações gráficas. Esta aplicação plota as informações de maneira semelhante a alguns instrumentos utilizados em aeronaves. Na figura 4.19 é apresentada a tela da aplicação desenvolvida com informações reais de execução.

[10]https://processing.org/


Figura 4.19: Tela da aplicação de telemetria

Nota-se na figura 4.19 que a tela da aplicação possui quatro instrumentos circulares e uma região retangular com informações textuais. O primeiro instrumento, localizado no canto superior esquerdo, mostra a rotação do aeromodelo sobre o eixo longitudinal (Indicador de Roll) e o instrumento logo abaixo indica rotação sobre o eixo lateral (Indicador de Pitch). Em ambos os instrumentos a inclinação é apresentada pela rotação dos elementos internos (visão frontal e perfil da aeronave) e seus valores são providos de medições do sensor MPU6050. O instrumento à direita do Indicador de Roll exibe o curso atual do aeromodelo em azul, que é fornecido pelo sensor HMC5883L e o curso para alvo em amarelo, que é o curso em que o aeromodelo deveria estar para atingir o ponto programado. O curso alvo só é atualizado quando o GPS estabelecer conexão com os satélites, pois ele é calculado de acordo com a posição atual através da função *course_to()* da biblioteca *TinyGPS*. O instrumento restante mostra a velocidade atual do aeromodelo, que é fornecida pelo GPS e se assemelha a um velocímetro convencional.

No retângulo ao lado dos instrumentos há informações da posição atual do aeromodelo, latitude, longitude e altitude, fornecidas pelo GPS. Logo abaixo há informações sobre o ponto destino (latitude e longitude) e a distância em que o aeromodelo se encontra deste ponto em metros. A distância é calculada ainda no arduino pela função *distance_between()* da biblioteca *TinyGPS*. O próximo conjunto de informações apresenta estatísticas de comunicação, das quais tem-se o número de pacotes recebidos, inválidos e perdidos. Os pacotes inválidos são identificados por valores fora da faixa fornecida pelos sensores. Por último, há a informação do estágio atual do sistema de navegação e status do GPS, que pode ser *Sem sinal* ou *Conectado*.

Para rodar o sistema de telemetria foi construído um computador de campo, apresentado nas figuras 4.20 e 4.21, que é composto de um Raspberry Pi 2, uma tela de sete polegadas, teclado, uma fonte de alimentação, bateria, Arduino Nano e um Xbee para a comunicação. A conexão do Xbee com o Raspberry é intermediada pelo Arduino Nano. A figura 4.22 apresentada as ligações.


Figura 4.20: Computador de campo (montagem e organização dos componentes).

Figura 4.21: Computador de campo finalizado

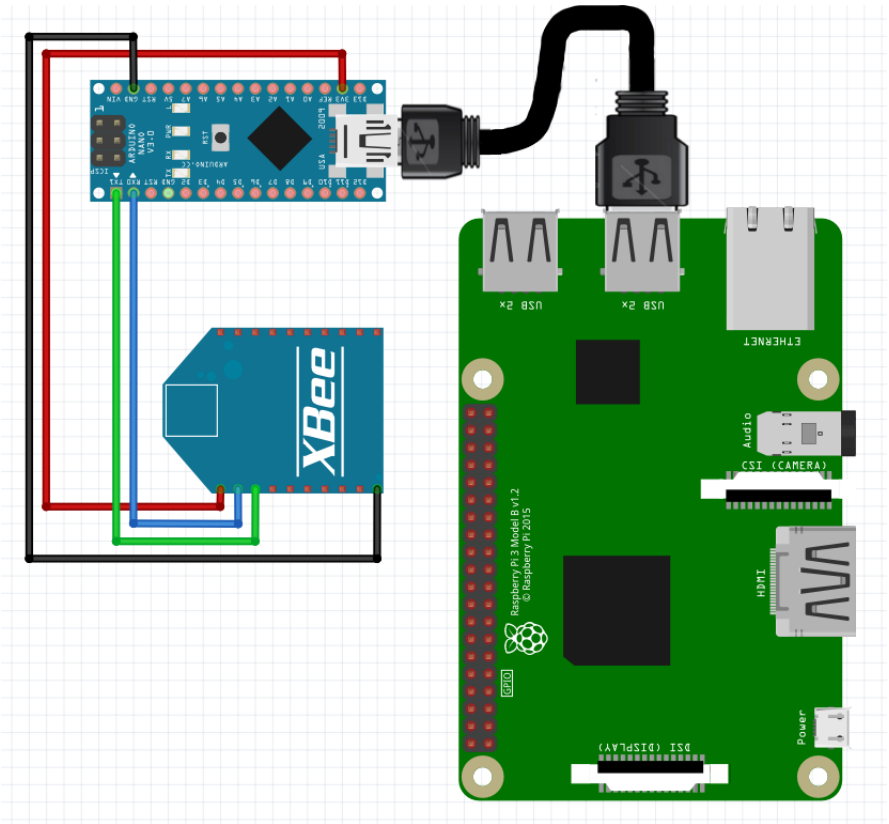


Figura 4.22: Conexão entre Xbee e Raspberry Pi 2

Para comunicação entre o aeromodelo e o computador de campo, a plataforma e o sistema de telemetria trabalham com um formato de pacote que foi desenvolvido explicitamente para essa comunicação considerando as limitações do hardware disponíveis. O pacote desenvolvido possui o tamanho de 8 bytes, onde o primeiro byte é destinado ao preâmbulo ($), quatro bytes para dados, um byte para identificar o tipo de dado e os dois bytes restantes para o contador de pacotes. O formato do pacote pode ser visto na figura 4.23.

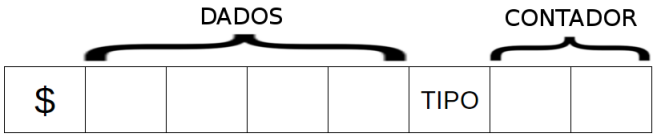


Figura 4.23: Estrutura do pacote de dados.

Com este formato de pacote é possível enviar todos os tipos de dados fornecidos pelos sensores da plataforma, pois eles são do tipo *inteiro* com representação de 16 bits ou *ponto flutuante* com representação de 32 bits, o que não ultrapassa o tamanho da região destinada a dados. O contador de pacotes é utilizado para estatísticas de comunicação e com ele é possível calcular os pacotes perdidos.

O Xbee recebe os pacotes enviados pelo aeromodelo e o Raspberry Pi 2 os lê através do Arduino Nano por uma porta serial. A aplicação Processing extrai os dados do pacote assim que ele é recebido e atualiza a tela da aplicação em tempo real.

## 4.4 Funcionamento do Sistema de Navegação Proposto

O sistema de navegação proposto para a plataforma desenvolvida neste trabalho possui um ciclo de execução definido, detalhado na figura 4.24.

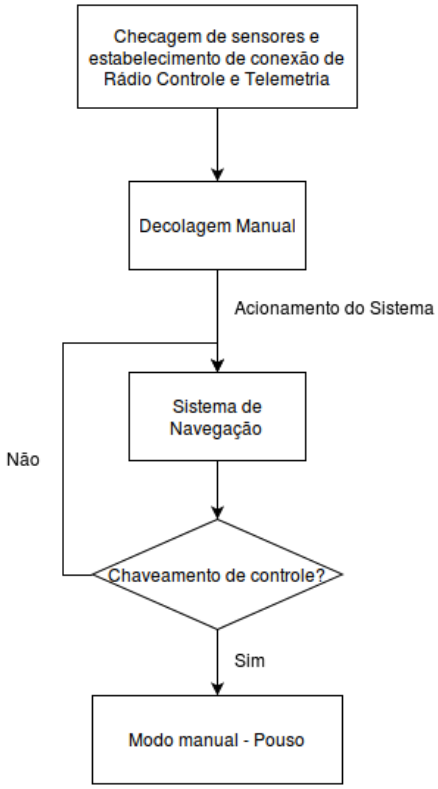


Figura 4.24: Ciclo de execução do sistema

No primeiro passo, o hardware implantado no aeromodelo é ligado. Então aguarda-se o estabelecimento da conexão entre o aeromodelo e o computador de campo utilizado para telemetria e também do rádio de controle manual. Depois de estabelecidas essas conexões, é possível verificar o estado dos sensores através da telemetria.

Concluído o primeiro passo, a decolagem do aeromodelo pode ser realizada manualmente. Uma vez que o aeromodelo está estabilizado, ou seja, em altitude e velocidade consideradas ideais pelo piloto em terra, o sistema de navegação pode ser acionado, para então a plataforma assumir o comando do aeromodelo.

A qualquer momento o controle pode ser retomado pelo piloto através do chaveamento no rádio controle, seja por emergência ou por conclusão da navegação. Após a retomada do controle, fica a critério do piloto em terra o momento do pouso.

## 4.5 Considerações

Neste capítulo foi integrado em uma única placa o hardware necessário para o funcionamento da plataforma. Também foi desenvolvido um sistema de telemetria e um computador de campo para sua execução. Um sistema de navegação para plataforma foi proposto dentro contexto dos recursos providos por ela.

No próximo capítulo são apresentados os teste executados para validação da plataforma e telemetria.

# Capítulo 5

# Testes

Neste capítulo são analisados testes produzidos para a validação da plataforma. Os testes foram realizados apenas em solo e em locais com condições similares, mas não adequadas, as quais se realizam voos com aeromodelos. A portaria 207[1] do DAC (Departamento de Aviação Civil) estabelece que a operação de aeromodelos deve ser suficientemente distante de áreas densamente povoadas e com o limite de 400 pés (aproximadamente 122 metros) acima da superfície terrestre.

## 5.1 Comunicação via Xbee para Telemetria

Para testar a comunicação entre o aeromodelo e o computador de base, foi planejada uma bateria de testes visando simular condições de comunicação que serão enfrentadas pela plataforma quando ela for utilizada em voo. Os testes foram realizados dentro do campus Centro Politécnico da Universidade Federal do Paraná, em locais previamente escolhidos para atingir as distâncias definidas para teste. A figura 5.1 mostra o local junto com a demarcação de pontos em que a comunicação foi testada.

[1]https://www.cobra.org.br/legislacao/10/portaria-dac-n-207


Figura 5.1: Pontos onde os testes foram realizados. O computador de campo ficou fixado no marcador com etiqueta "#0"e a plataforma foi posicionada em outras marcações para realizar a comunicação. Em cada ponto foi testado o envio de dez mil pacotes e esse processo foi repetido dez vezes a cada marcação para identificar ou não uma instabilidade na comunicação. Os pontos marcados estão a uma distância de 100 metros da base em "#1", 200 em "#2", 300 em "#3", 400 em "#4"e 500 metros no ponto "#5".

Com os dados obtidos da bateria de testes, foi construída a tabela 5.1 onde cada linha tem a média de dez iterações dos dados obtidos na distância em questão. O número de pacotes enviados refere-se aos que saíram da plataforma, enquanto o número de pacotes recebidos é referente aos pacotes recebidos na base. O número de pacotes perdidos corresponde à diferença entre os pacotes enviados e recebidos e o desvio padrão foi calculado com base no número de pacotes perdidos.

A partir da tabela 5.1 foi produzido o gráfico da figura 5.2 com as demarcações do desvio padrão que permite a análise visual do comportamento da comunicação entre plataforma e computador de campo.

Tabela 5.1: Estatísticas de Comunicação

| | Enviados | Recebidos | Perdidos | Desvio padrão | % de recebimento |
|---|---|---|---|---|---|
| 100 Metros | 10477,8 | 10048,6 | 429,2 | 134,15 | 95,90% |
| 200 Metros | 10452,8 | 9898,2 | 554,6 | 183,99 | 94,68% |
| 300 Metros | 10373,8 | 9089,2 | 1284,6 | 779,75 | 87,64% |
| 400 Metros | 10280,3 | 9630,8 | 649,5 | 491,35 | 93,68% |
| 500 Metros | 10199,2 | 9187 | 1012,2 | 804,79 | 90,06% |

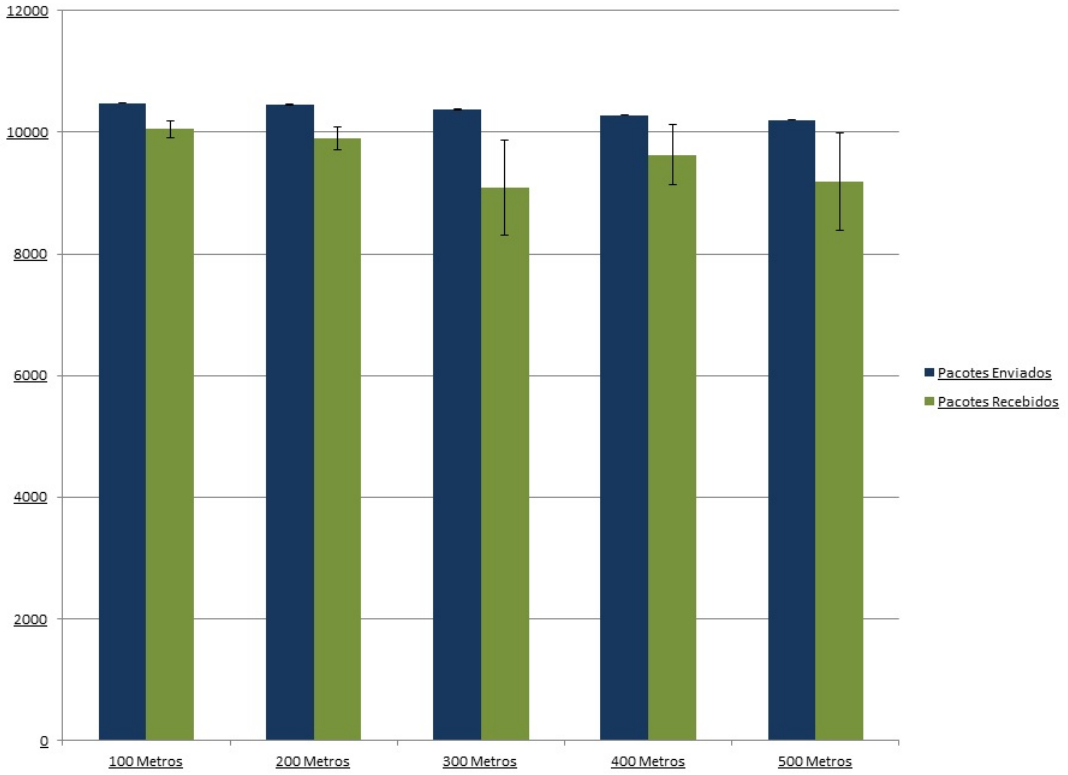


Figura 5.2: Grafico de Estatísticas de Comunicação

A perda de pacotes durante os testes se mostrou mais suscetível aos obstáculos presentes (árvores, estruturas metálicas e prédios) do que a distância em relação ao receptor. Onde o contato visual entre a plataforma e o computador de campo era possível, o percentual de recebimento de pacotes foi maior e o desvio padrão menor, demonstrando maior estabilidade na comunicação. Este é o caso dos testes em 100, 200 e 400 metros.

Mesmo com uma perda maior de pacotes em 300 e 500 metros, a telemetria não foi prejudicada e se mostrou contínua durante toda a execução, permitindo a visualização em tempo real do estado do aeromodelo.

## 5.2 Sensores HMC5883L e MPU6050

A partir dos dados coletados nos testes de comunicação, foram extraídos alguns intervalos com mil leituras dos sensores HMC5883L e MPU6050 para análise. Para isso, gráficos foram plotados com o valor obtido a cada leitura.

O gráfico 5.3 plota as leituras do HMC5883L que fornecem o curso atual e os gráficos 5.4 e 5.5 plotam as leituras do sensor MPU que entregam o ângulo de *roll* e *pitch*.

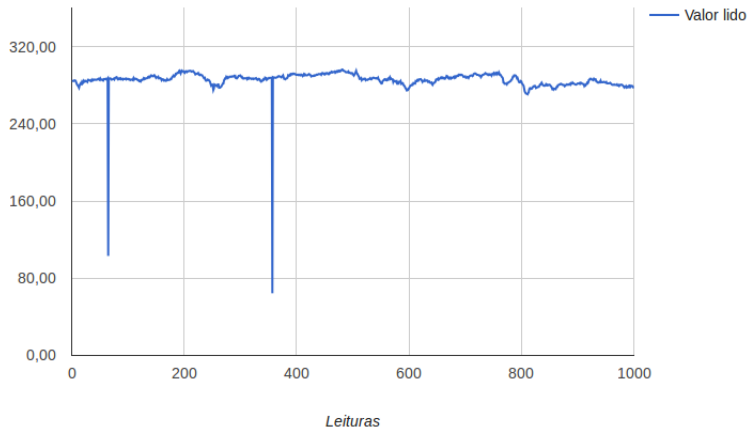


Figura 5.3: Leituras do sensor HMC5883L.

Com o gráfico 5.3 é possível observar que dois pontos que sobressaíram à continuidade das leituras do sensor. Essas duas linhas representam um ruído na saída do sensor. Esse tipo de ruído não é considerado um defeito de hardware e a maioria dos sensores são passíveis desse acontecimento. Com base nesse gráfico e na análise que foi feita em outra leituras, a proporção de ruído mostrou-se insignificante em comparação ao número de leituras corretas. Uma opção viável para atenuação de ruídos é a implementação de filtros por software, como o da média móvel, que realiza a média das últimas leituras obtidas, o que acaba dispersando o ruído.

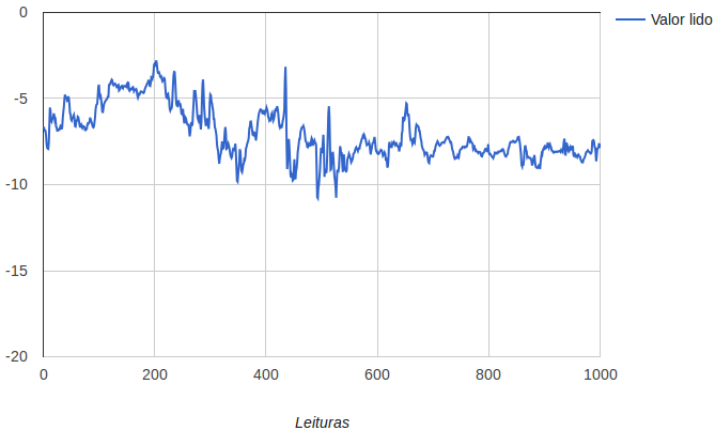


Figura 5.4: Leituras do angulo de *roll* do sensor MPU6050

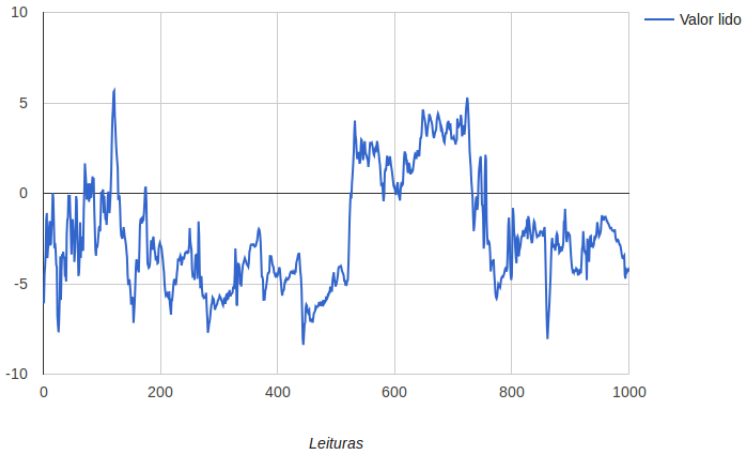


Figura 5.5: Leituras do angulo de *pitch* do sensor MPU6050

Nos gráficos apresentados com as leituras do sensor MPU6050 não foram observados ruídos nas leituras obtidas. A variação presente na linha projetada dá falsa impressão de alta variação nos valores lidos e isso ocorre porque a escala do gráfico é pequena. A variação presente apenas reflete movimentos reais do sensor.

## 5.3 Localização com GPS

Para testar o funcionamento do GPS, dois percursos foram realizados a pé com a plataforma. Ao longo deste percurso os pontos obtidos pelo GPS foram armazenados. Os pontos recebidos durante o percurso são projetados nas imagens de satélites nas figuras 5.6 e 5.7.


Figura 5.6: Percurso 1. Cada marcador em azul nesta imagem representa um ponto (latitude. longitude) obtido pelo sensor GPS


Figura 5.7: Percurso 2. Este trecho percorrido corresponde ao caminho que liga os pontos #4, #5, #3, #2 e #1 utilizados para os testes de comunicação, vistos na figura 5.1.

As leituras do módulo GPS ao longo dos percursos percorridos se mostraram bastante precisas e pode-se concluir que o módulo utilizado nesta plataforma é confiável para ser utilizado em voo.

## 5.4 Circuito de chaveamento auto/manual

O circuito de de chaveamento auto/manual foi testado em bancada, sendo verificado o correto funcionamento do chaveamento, o qual depende da recepção do sinal de controle do rádio. Na ausência de sinal o modo default é o automático, pois se não há recepção do sinal de controle de chaveamento provavelmente também não há recepção dos demais sinais de controle do rádio. Assim, este recurso da plataforma permite a operação do aeromodelo no modo mais seguro possível em caso de perda de comando do rádio.

# Capítulo 6

# Conclusão

## 6.1 Análise Geral do Trabalho

Ao longo do desenvolvimento deste trabalho, vários componentes de hardware foram integrados e um circuito foi projetado com o propósito de possibilitar a automação e telemetria de um aeromodelo. O funcionamento em sincronia de todos os componentes, indispensáveis para a automação, foi o grande desafio deste trabalho. Com o uso do *Fritzing*, software de código aberto destinado a projeto de hardware, uma placa de circuito impresso foi projetada e construída para unificar todo o hardware, proporcionando maior usabilidade da plataforma, pois peso e tamanho costumam ser fatores limitantes de voo.

Através dos testes realizados em solo, a plataforma construída mostrou-se apta a ser utilizada para um sistema de navegação ponto a ponto. A telemetria que foi desenvolvida em *Processing*, outro software de código aberto, destinado ao desenvolvimento de aplicações gráficas, permitiu o monitoramento e validação da plataforma durante os testes. O sistema de telemetria funcionou de forma satisfatória provendo a visualização do estado do aeromodelo em tempo real, complementando o uso da plataforma. O circuito de chaveamento, peça fundamental para o uso em voo, mostrou-se preciso e confiável, garantindo a segurança por proporcionar a retomada instantânea do controle manual quando necessário.

Com isso, conclui-se que os objetivos deste trabalho foram alcançados com o desenvolvimento desta plataforma, utilizando os componentes propostos encontrados no comércio. A base necessária para o desenvolvimento de um sistema de navegação foi provida e validada para uso através dos testes.

As principais dificuldades enfrentadas neste trabalho, foram relacionadas a integração e construção dos componentes de hardware, visto que enquanto módulos que foram programados e testados isoladamente funcionaram sem problemas, quando unificados apresentaram falhas ou até mesmo não funcionaram. A construção do circuito de chaveamento foi um dos grande desafios.

## 6.2 Trabalhos Futuros

Para trabalhos futuros tem-se a ideia de desenvolver um sistema de navegação com base no modelo proposto neste trabalho, com a plataforma construída e seus recursos disponibilizados. Com a implementação de um sistema de navegação, o leque de possibilidades de uso da plataforma aumenta e uma ideia cogitada é a de usar o aeromodelo para captação de imagens aéreas.

Além do desenvolvimento de sistemas para a plataforma, a revisão e otimização dos componentes eletrônicos também são possibilidades pra trabalhos futuros, buscando um melhoramento contínuo da plataforma.

# Referências Bibliográficas


[Apoorve, 2016] Apoorve (2016). Servo motor. http://circuitdigest.com/article/servo-motor-basics. Acessado em 16/10/2016.

[Arduino, 2016a] Arduino (2016a). Arduino mega 2560. https://hifiduino.wordpress.com/2012/04/13/ iteadstudio-tft-display-for-arduino/. Acessado em 01/09/2016.

[Arduino, 2016b] Arduino (2016b). Arduino nano. "https://www.arduino.cc/en/Main/ArduinoBoardNano". "Acessado em 19/11/2016".

["Arduino.cc", 2016] "Arduino.cc"("2016"). "arduino.cc". https://www.arduino.cc/en/Main/ArduinoBoardMega2560. "[Acessado em 01/09/2016]".

[Branco et al., 2011] Branco, K. R. L. J. C., Pelizzoni, J. M., Neris, L. O., Trindade, O., Osório, F. S., and Wolf, D. F. (2011). Tiriba - a new approach of uav based on model driven development and multiprocessors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4.

[EM-411, 2016a] EM-411 (2016a). Em-411. http://www.hobbytronics.co.uk/gps-em411. Acessado em 01/09/2016.

[EM-411, 2016b] EM-411 (2016b). Em-411. http://www.dpcav.com/data_sheets/ EM411Product_Guide1_2.pdf . [Acessado em 01/09/2016].

[F. D. Anderson, 2001] F. D. Anderson, S. E. (2001). In *Understanding Fligth. New York: McGraw-Hill*.

[Garrett, 2011] Garrett, F. (2011). Gps. http://www.techtudo.com.br/artigos/noticia/2011/12/como-funciona-o-gps.html. Acessado em 16/10/2016.

[HMC5883L, 2016] HMC5883L (2016). Hmc5883l. https://www.itead.cc/hmc5883l-triple-axis-compass-magnetometer-sensor-module.html. Acessado em 01/09/2016.

[Instruments", 2016] Instruments", T. ("2016"). "lm358". www.ti.com/lit/ds/symlink/lm158-n.pdf.

[Liu et al., 2012] Liu, Z., Chen, Y., Liu, B., Cao, C., and Fu, X. (2012). Hawk: An unmanned mini helicopter-based aerial wireless kit for localization. In *INFOCOM, 2012 Proceedings IEEE*, pages 2219–2227.

["Motorola", 2016] "Motorola"("2016"). "74ls122". www.uni-kl.de/elektronik-lager/417682.

[MPU6050, 2016a] MPU6050 (2016a). Gy-521 mpu6050. http://www.hotmcu.com/gy521-mpu6050-3axis-acceleration-gyroscope-6dof-module-p-83.html. Acessado em 01/09/2016.

[MPU6050, 2016b] MPU6050 (2016b). Gy-521 mpu6050. https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/. Acessado em 01/09/2016.

[Ogata, 1982] Ogata, K. (1982). *Engenharia de controle moderno*. Pearson Prentice Hall.

[PID, 2016] PID (2016). Pid. http://cs.brown.edu/ tld/courses/cs148/02/images/pid.gif . [Acessado em 01/12/2016].

[Raspberry, 2016] Raspberry (2016). Raspberry pi 2. http://www.pcworld.com/article/3038201/hardware/the-evolution-of-the-raspberry-pi-from-concept-to-raspberry-pi-3.html#slide3. Acessado em 01/09/2016.

[Raspberry.org, 2016] Raspberry.org (2016). Raspberry.org. [Acessado em 01/09/2016].

[robocore, 2016] robocore (2016). Rádio transmissor e receptor. https://www.robocore.net/loja/produtos/radio-turnigy-9x-2_4ghz-9-canais-com-receptor.html. Acessado em 16/10/2016.

[Sperry, 1929] Sperry, L. B. (1929). Automatic pilot for airplanes.

[Wikipédia, 2016a] Wikipédia (2016a). Wikipédia, arduino. https://pt.wikipedia.org/wiki/Arduino. [Acessado em 01/09/2016].

[Wikipédia, 2016b] Wikipédia (2016b). Wikipédia, raspberry. https://pt.wikipedia.org/wiki/Raspberry_Pi. [Acessado em 01/09/2016].

[XBee, 2016a] XBee (2016a). Xbee. http://www.filipeflop.com/pd-148028-modulo-xbee-pro-60mw-wire-antenna-serie-1.html. Acessado em 01/09/2016.

[XBee, 2016b] XBee (2016b). Xbee. http://www.digi.com/products/xbee-rf-solutions/modules/xbee-802-15-4. [Acessado em 01/09/2016].

[Yang and Geng, 2011] Yang, H. and Geng, Q. (2011). The design of flight control system for small uav with static stability. In *Mechanic Automation and Control Engineering (MACE), 2011 Second International Conference on*, pages 799–803.

[Young et al., 1944] Young, C. M., Lynch, E. E., and Boynton, E. R. (1944). Electrical control in automatic pilots. *Transactions of the American Institute of Electrical Engineers*, 63(12):939–943.

# Apêndice A

## A.1 Controlador Proporcional Integral Derivativo (PID)

O controlador PID é um algoritmo de controle de processos composto por três coeficientes: proporcional, integral e derivativo. A ideia básica de um controlador PID é manter uma variável de um processo sob controle, o que pode ser compreendido como manter a variável em um estado desejado. Para fazer isso, o controlador lê um sensor que fornece o estado atual da variável e calcula a saída somando o cálculo proporcional, integral e derivativo. Cada coeficiente tem um ganho relacionado, ou seja, o proporcional tem o ganho Kp, o integral tem o ganho Ki e derivativo Kd e são estes ganhos que são calibrados para obter o resultado ideal. A resposta do controlador alimenta um atuador que visa corrigir a variável do processo para o estado desejado. A figura A.1 apresenta o modelo de controle PID.

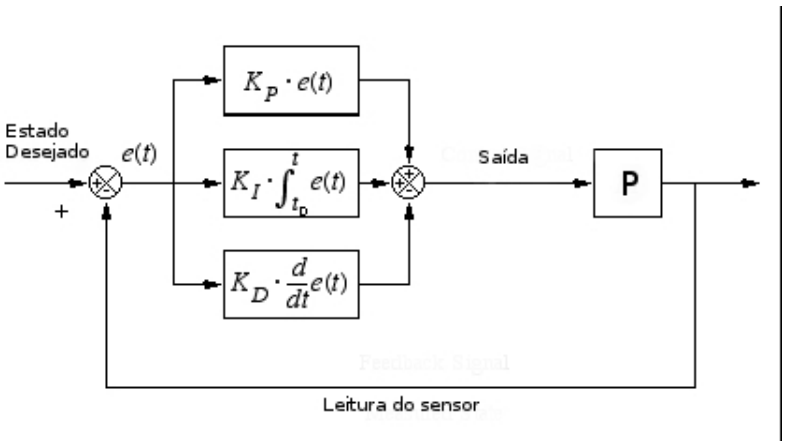


Figura A.1: Controlador PID, editado de [PID, 2016].

### A.1.1 Cálculo Proporcional

O cálculo proporcional trabalha com a diferença do valor obtido pelo sensor e o valor em que a variável deveria estar, portanto, o erro. O ganho proporcional Kp determina a resposta de saída para esse erro de forma direta. Por exemplo, se o ganho proporcional Kp for 5 e o erro for 10, a resposta proporcional será 50.

### A.1.2 Cálculo Integral

O cálculo integral soma o erro ao longo do tempo, assim, mesmo com erro a integral irá aumentar lentamente até o processo atingir erro zero. O propósito da integral é corrigir um erro estacionário, caso ele exista, pois o ganho proporcional não conseguiu produzir uma resposta que eliminasse esse erro.

### A.1.3 Cálculo Derivativo

O cálculo derivativo atua de forma a diminuir a resposta caso a leitura da variável esteja aumentando rapidamente. Aumentar o ganho Kd fará com que o sistema de controle reaja com uma resposta mais forte com a variação do erro.