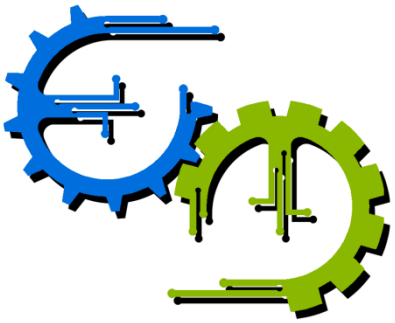
TRABALHO DE GRADUAÇÃO

SISTEMA DE CONTROLE DE VANT COM EMPREGO DE PLATAFORMA INERCIAL

Por,
Cássio Rodrigues Pantoja 11/0147413
Paulo Vitor de Araújo Garcia 11/0151291

Brasília, Julho de 2017



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SISTEMA DE CONTROLE DE VANT COM EMPREGO DE PLATAFORMA INERCIAL

POR,

Cássio Rodrigues Pantoja 11/0147413
Paulo Vitor de Araújo Garcia 11/0151291

Relatório submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. D. Sc. Ricardo Zelenovsky, PUC-RJ, UnB/ ENE (Orientador)

Prof. Eduardo Peixoto Fernandes da Silva, UnB/ ENE

Prof. Edson Mintsu Hung, UnB/ ENE

Brasília, Julho de 2017

FICHA CATALOGRÁFICA

PANTOJA, CÁSSIO RODRIGUES.
GARCIA, PAULO VITOR DE ARAÚJO.
Sistema de Controle de VANT com Emprego de Plataforma Inercial.

Distrito Federal, 2017.

xii, 81p., 210 x 297 mm (Mecatrônica/FT/UnB, Engenheiro, Controle e Automação, 2017).
Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.

1. Drone
2. IMU
3. Potenciômetro
4. Controle

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

PANTOJA, C. R.; GARCIA, P. V. A. (2017). Sistema de Controle de VANT com Emprego de Plataforma Inercial. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT. TG-nº 01/2017, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 81p.

CESSÃO DE DIREITOS

AUTORES: Cássio Rodrigues Pantoja e Paulo Vitor de Araújo Garcia.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Sistema de Controle de VANT com Emprego de Plataforma Inercial.

GRAU: Engenheiro de Controle e Automação ANO: 2017

Cássio Rodrigues Pantoja
Laboratório de Robótica Aérea
Prédio SG-11
Faculdade de Tecnologia
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 – Brasília – DF – Brasil

Paulo Vitor de Araújo Garcia
Laboratório de Robótica Aérea
Prédio SG-11
Faculdade de Tecnologia
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 – Brasília – DF - Brasil

AGRADECIMENTOS

" Se eu vi mais longe, foi por estar
de pé sobre ombros de gigantes."
- Isaac Newton

Sinto-me privilegiado de estar onde estou, de ter a companhia das pessoas que convivo, e trilhar o caminho que trilhei. Entrego um profundo sentimento de gratidão à minha família: Felipe, Helvio e Maray, que sempre se preocuparam comigo, me proveram condições e motivações para ir mais longe, que sempre me apoiaram em momentos difíceis, e sempre acreditaram na minha capacidade e perseverança em alcançar meus objetivos. Agradecimentos aos meus amigos de Porto Velho: Ana, Athos, Bruno, Fábio, Luis, Rafael, Tobias, Yara e Yuri, que mesmo de longe me acompanharam nesse caminho e me proporcionaram bons momentos. Agradecimentos especiais a Cleonice Rossi Gomes, Francisco Carlos Adriano e família por me acolherem e me tratarem com muito carinho e cuidado durante esses longos anos de faculdade. Agradecimentos aos meus amigos de Brasília: Alex, Davi, Gibson, Izabella, Letícia, Luiz, Mickael e especialmente ao Paulo, que veio a ser meu parceiro neste trabalho de graduação, pois juntos vivenciamos esses anos de faculdade e compartilhamos muitos momentos, tanto os alegres como os difíceis, juntos. Sinto, também, gratidão à equipe de professores da UnB, destacando os professores do ENM, ENE e CIC, que compõem o curso de engenharia mecatrônica que finalizo. Agradecimento especial ao professor Ricardo Zelenovsky, que esteve sempre prestativo a nos ouvir e nos orientar na confecção deste projeto. Obrigado a todos os gigantes em minha vida, que me permitiram enxergar ainda mais longe.

Cássio Rodrigues Pantoja

Agradeço primeiramente a Deus, por me fortalecer e me dar sabedoria para trilhar meus sonhos com perseverança e disciplina. À minha família, em especial meus pais, por terem batalhado para que eu pudesse chegar onde cheguei, estando sempre ao meu lado e construindo meu caráter como pessoa íntegra e responsável. Sem vocês esses vários anos de engenharia não seriam possíveis. Agradeço em especial ao professor Ricardo Zelenovsky e ao meu companheiro de projeto Cássio Rodrigues Pantoja por me darem a oportunidade de fazer esse trabalho, sempre com motivação e alegria. Gratifico também todos os profissionais de educação que me ensinaram ou me fizeram aprender tudo que sei, me tornando um engenheiro competente e curioso para o mercado de trabalho. Agradeço aos meus amigos mais próximos, que me fizeram ter paciência e calma para vencer os obstáculos da vida acadêmica e pessoal. Á todas as pessoas que, de certa maneira, passaram pela minha vida, mesmo que de forma rápida, mas que construíram minha identidade como ser socialmente correto. Reconheço especial valor ao programa Ciência Sem Fronteiras por me mostrar desafios que me fizeram crescer individualmente. Por fim, agradeço à equipe de competição da engenharia mecatrônica, a DROID, por me tornar um engenheiro mais prático desde o começo do curso e aos diversos profissionais da área técnica que me ajudaram a complementar minhas habilidades e competências.

Paulo Vitor de Araújo Garcia

RESUMO

Por este trabalho se objetiva desenvolver um *joystick* sensível a rotações nas três dimensões, e que esse movimento seja traduzido em informações que atue em um VANT. Dessa maneira se busca tornar a pilotagem do VANT mais intuitiva para um piloto aprendiz, permitir que experientes o opere com somente uma das mãos, além de maior diversão para aqueles que pilotam por lazer. Desenvolveu-se, então, dois sistemas que, em conjunto, substituem o *joystick* de um VANT (Syma X5SW) e permite controla-lo por movimentos de rotação em um sistema de controle. A ideia consiste em utilizar uma Unidade de Medição Inercial (IMU) composta por pelo menos um acelerômetro e um giroscópio, e utilizar seus dados para se enviar ao VANT em questão. No projeto do controle foram utilizados uma IMU (MPU-6050), uma unidade de processamento programável (Arduino Pro Mini) e um módulo de comunicação (Bluetooth) para se enviar sinais a uma base de atuação. Essa base contém outro módulo de comunicação (Bluetooth) para receber a comunicação enviada pelo controle, uma outra unidade de processamento programável (Arduino UNO), a placa do *joystick* original do fabricante com seus potenciômetros removidos, e quatro potenciômetros digitais conectados à placa para substituir os removidos. A atuação então recebe os dados do controle e configura os potenciômetros digitais para fornecer a tensão necessária de acordo com o sinal recebido. Alguns problemas surgiram, em decorrência de atrasos necessários para se ajustar os potenciômetros em seus devidos níveis, mas de modo geral, o grupo obteve sucesso em controlar o VANT dessa maneira, em fazê-lo levantar voo e pilotá-lo.

ABSTRACT

The objective in this work is to build a three dimentional motion-sensitive rotational joystick that can be used to control a drone. By that, it is expected an easier piloting for drone flighting beginners, allow experienced flighters to control it single handed, and even extra fun for those who flight for recreational purposes. To achieve that, a system capable of replacing a drone's (Syma X5SW) joystick was developed. The idea is to use an Inertial Measurement Unit (IMU) with at least one accelerometer and a gyroscope to obtain rotational data and use them to control the drone. For so, it was necessary an IMU (MPU-6050), an programmable processing unit (Arduino Pro Mini) and a communication module (Bluetooth) in the controller, for it to send data for a base of operations. This base has another communcation module (Bluetooth) to recieve communication data, another programmable processing unit (Arduino UNO), the drone's joystick board with its analog potentiometers removed, and four digital potentiometers to replace those removed. The base of operations then recieves data from the controller and sets the digital potentiometers to grant necessary voltage accordingly. Because of some necessary delays to set those digital potentiometers on their correct levels, the flight wasn't as easy as expected, but, altogether, the group succeded in flying the drone.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

F_{BW}	Frequência de banda passante	[Hz]

Símbolos gregos

τ	Constante de tempo	[s]

Subscritos

in	Entrada
out	Saída

Siglas

AC	Alternating current
AD	Analógico-digital
CI	Circuito integrado
DAC	Digital-analog converter
DC	Direct current
FFT	Fast Fourier transfer
FPV	First person view
HD	Hard drive
IDE	Integrated development environment
IEC	Internation electrotechnical comission
IMU	Inertial measurement unit
JDH	*Joystick* direita horizontal
JDV	*Joystick* direita vertical
JEH	*Joystick* esquerda horizontal
JEV	*Joystick* esquerda vertical
MEMS	Micro-electro-mechanical system
PCB	Printed circuit board
PWM	Pulse width modulation
RF	Rádio frequência
RPA	Remotely Piloted Aircraft
RPM	Rotações por minuto
SPI	Serial Peripheral Interface
TTL	Transistor-transistor logic
UART	Universal asynchronous receiver/transmitter
VANT	Veículo aéreo não tripulado
VTOL	Vertical take-off and landing

Capítulo 1

INTRODUÇÃO

" Você não consegue ligar os pontos olhando para frente; você só consegue ligá-los olhando para trás. Então você tem que confiar que os pontos se ligarão algum dia no futuro. Você tem que confiar em algo – seu instinto, destino, vida, carma, o que for. Esta abordagem nunca me desapontou, e fez toda diferença na minha vida. "

- Steve Jobs

1.1 Contextualização

A história da aviação remonta a tempos pré-históricos. O desejo de voar está presente na humanidade, provavelmente, desde o dia em que o homem pré-histórico passou a observar o voo dos pássaros e de outros animais alados. Ao longo da história há vários registros de tentativas malsucedidas de voos. Alguns até tentaram voar imitando pássaros: usar um par de asas (que não passavam de um esqueleto de madeira e penas, imitando as asas dos pássaros), colocando-os nos braços e balançando-os.

Figura 1 - "*Icarus* e *Daedalus*" por Charles Paul Landon *[1]*

Muitas pessoas acreditavam que voar fosse impossível e que era um poder além da capacidade humana. Mesmo assim o desejo existia e várias civilizações contavam histórias de pessoas dotadas de poderes divinos que podiam voar, como o mito de Ícaro e Dédalo (Figura 1); ou pessoas que foram carregadas ao ar por animais voadores. A história moderna da aviação é complexa. Desenhistas de aeronaves esforçaram-se para melhorar continuamente suas capacidades e características tais como alcance, velocidade, capacidade de carga, facilidade de manobra, dirigibilidade, segurança, autonomia e custos operacionais, entre outros. Aeronaves passaram a ser feitas de materiais cada vez menos densos e mais resistentes. Anteriormente feitas de madeira, atualmente a grande maioria das aeronaves usa materiais compostos - como alumínio e fibras de carbono. Recentemente computadores têm contribuído muito no desenvolvimento de novas aeronaves e componentes. [1]

Os primeiros passos da humanidade na área da aviação vieram com o surgimento dos primeiros planadores. Pipa, o planador mais rudimentar, foi invenção dos chineses em 300 A.C. Depois, tiveram-se algumas invenções nunca implementadas, e estudos de máquinas capazes de voar da época renascentista do século XIV, figurando o italiano Leonardo da Vinci (Figura 2), como protagonista. Até que se conseguiu chegar ao primeiro voo bem-sucedido, realizado por um balão de ar quente (A Passarola), de Bartolomeu de Gusmão no ano de 1709 em Lisboa, Portugal.

Figura 2 - Projetos de máquina voadora feita por Leonardo da Vinci *[2]*

O primeiro voo realizado com sucesso, de acordo com a mídia europeia e norte-americana, foi protagonizado pelo brasileiro Santos Dumont, em Paris na França no ano de 1906 (Figura 3). O 14-Bis, famoso avião que marcou este momento histórico, foi projetado e construído por ele, sendo o primeiro avião na história da aviação a ter seu voo homologado por uma instituição pública aeronáutica, o Aeroclube da França. O destaque a este voo se deve principalmente por não ter dependido de nenhum tipo de exterioridade, como ventos e propulsões não-próprias, como catapultas. Esse é o principal argumento utilizado para afirmar que o brasileiro foi de fato o pioneiro na criação do avião, em detrimento dos testes feitos pelos irmãos Wright. Os irmãos, em contrapartida, juntamente com os estudos de Bryan (1911) e Lanchester (1908) forneceram uma base de conhecimento em controle e estabilidade considerável para as próximas gerações. [3]

Figura 3 - Ilustração do voo do Santos-Dumont 14-bis em 12 de novembro de 1906, que lhe rendeu o Prêmio do Aeroclube da França *[1]*

A tecnologia dos aviões avançou surpreendentemente durante a Primeira Guerra Mundial: criaram-se motores mais poderosos e melhorou-se a aerodinâmica, além das adaptações de fins bélicos.

Os anos que se seguiram, principalmente no período entre guerras, foram então marcados por rápidos avanços neste ramo e linhas aéreas começaram a operar. Assim, gerou-se um crescente impacto sócio econômico mundial. Durante a era de ouro da aviação, década de 1930 em especial, melhoraram-se consideravelmente os equipamentos de controle e a tecnologia de rádio comunicação, permitindo seu uso na aviação gerando técnicas de navegação aérea mais estáveis, como o próprio piloto automático. Nas décadas seguintes, com a corrida armamentista, marca da Guerra Fria, foram criadas tecnologias militares como o voo automático de mísseis de longa distância e aeronaves de espionagem. Neste ponto, a história da aviação e da robótica já se misturava: tais tecnologias foram as que mais contribuíram para o desenvolvimento da automatização do transporte aéreo comercial contemporâneo, influenciando diretamente o surgimento de aeronaves robóticas. [2]

Verifica-se então que desde o começo da década de 1990, a aviação comercial passou a desenvolver tecnologias que tornaram o avião cada vez mais automatizado, assim reduzindo gradativamente a importância do piloto na operação da aeronave, visando a diminuição dos acidentes aéreos causados por falha humana. Os fabricantes de aviões comerciais continuaram a pesquisar maneiras de melhorar as aeronaves, tornando-as cada vez mais seguras, eficientes e silenciosas. Ao mesmo tempo, pilotos, controladores de espaço aéreo e mecânicos passaram a ser cada vez mais bem-treinados, e os aviões são cada vez mais vistoriados, para evitar acidentes causados por falha humana ou mecânica. [1]

Atualmente, aplicações aéreas de risco, como reconhecimento ou resgates em campos de combate ou áreas impróprias para o homem ou de difícil acesso, decolagem e aterrissagem em terrenos de dimensões reduzidas e/ou irregulares, detecção de incêndios, reconhecimento de falhas em estruturas, aplicações de monitoramento na agropecuária ou reflorestamento e até mesmo como ferramentas de entrega de produtos, como a Amazon já testou, geraram a necessidade de um novo tipo de aeronaves. Os Veículos Aéreos Não-Tripulados (VANTs) vêm sendo cada vez mais utilizados para esses tipos de aplicações (Figura 4). Nesse contexto encontram-se os chamados *Vertical Take-off and Landing* (VTOL) que são aeronaves com capacidade de decolar e pousar em terrenos com dimensões reduzidas e de difícil acesso.

Os VTOLs podem ter características de voo lento e *hover* como helicópteros, e também características para voos para longa distâncias onde é necessária uma velocidade alta, para isso geralmente adiciona-se uma asa fixa em sua estrutura e a transição de modos de voo *hover* para cruzeiro é dada com a angulação dos rotores. [4]

Figura 4 – *Drone* Phantom 2 Vision+ *[5]*

Em paralelo ao desenvolvimento da aviação tem-se o desenvolvimento da indústria de telecomunicações. Aqui dando mais destaque à história dos *smartphones*. Em 2007, a Apple lançou o seu primeiro smartphone, o iPhone que combinava recursos dos iPods aos celulares, além de novas funcionalidades, foi o primeiro a vir com acelerômetro.

A Figura 5, abaixo, ilustra uma IMU (*Inertial Measurement Unit*) ou Plataforma Inercial, que consiste em um acelerômetro juntamente com giroscópio, presente em *smartphones*. Caracterizada por ser um sensor proprioceptivo, pela capacidade em reconhecer a localização espacial de um corpo, sua posição e orientação, a IMU, atualmente, é amplamente utilizada na indústria de petróleo (monitoramento das cargas e vibrações por ondas e marés que os dutos de extração são expostos), na indústria automotiva (nos sistemas de *airbag*, no controle de estabilidade e na análise da vibração interna do veículo), na robótica (como sensor de percepção espacial) e na indústria de entretenimento de jogos e realidade virtual (como em videogames, o Wii da Nintendo, por exemplo, ou em celulares mesmo, como os óculos de realidade virtual, tudo para aumentar a imersão do usuário no mundo digital).

Figura 5 - IMU e Smartphones

O desafio principal aqui então será iniciar a união dessas duas grandes frentes: aviação e propriocepção. Será implementado um controle para um VANT usando uma IMU, simulando a de um smartphone. Pretende-se desvendar novas formas de integração e, eventualmente, descobrir novas facilidades para que a passagem para a IMU de um celular seja feita da forma mais eficiente e otimizada.

1.2 Motivação

O surgimento e a constante evolução tecnológica nos VANTs fizeram com que seu mercado crescesse exponencialmente nos últimos anos. Dentre várias aplicações em que eles são usados hoje, pode-se destacar algumas delas: [6]

- Emergência e Recuperação de Desastres:
  - Monitoramento de áreas críticas e com materiais perigosos;
  - Coordenação de situações de emergência (apoio a equipes em campo).
- Planejamento Urbano:
  - Gerenciamento de construção;
  - *Design* ecológico;
  - Mapeamento (2D e 3D).
- Segurança:
  - Vigilância pública e privada;
  - Operação tática em segurança pública.
- Agricultura:
  - Monitoramento biológico e químico;
  - Detecção de Fogo e Inundação;
  - Inventário;
  - Controle de pragas;
  - Agricultura de precisão.
- Negócios e Comércio:
  - Serviços para indústria de mineração, óleo e gás;
  - Inspeções;
  - Serviços de entrega.
- Mídia, Comunicação e Publicidade:
  - Produção artística;
  - Entretenimento;
  - Jornalismo investigativo.

Figura 6 - VANT em campo monitorando linhas de transmissão *[6]*

A figura 6, acima, ilustra o propósito de um VANT em campo. A figura 7, na página seguinte, mostra diversas áreas e a participação relativa de VANTs nelas.

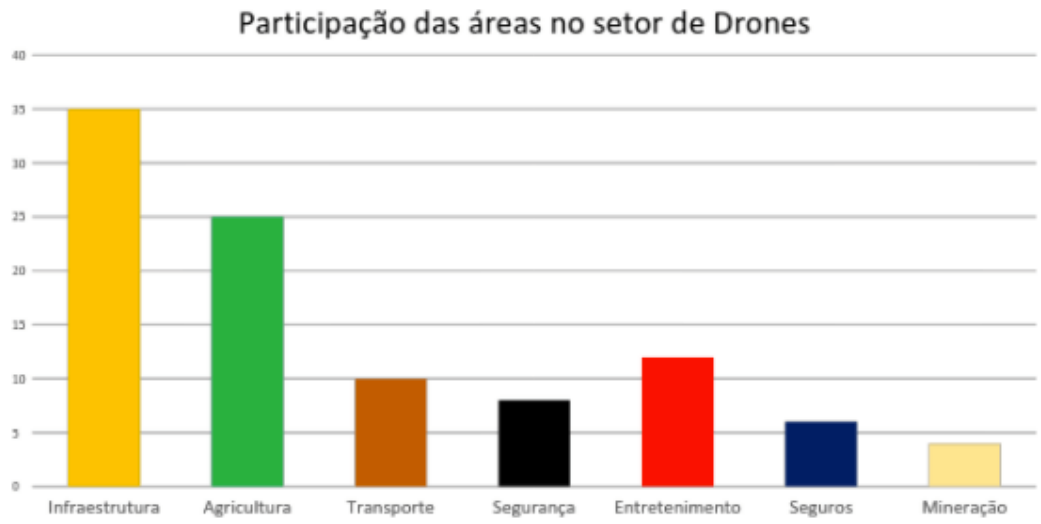


Figura 7 - Divisão do uso de drones por setor. Fonte: PWC *[6]*

A ideia principal nesse projeto é construir um dispositivo portátil que, com seu movimento de rotação nos 3 eixos, seja capaz de controlar a movimentação dos respectivos três eixos de um VANT. Ou seja, uma rotação no sistema de controle induz uma rotação de mesmo sentido no VANT, conforme ilustrado pela Figura 8. Sua altura será controlada por botões de subida e descida.

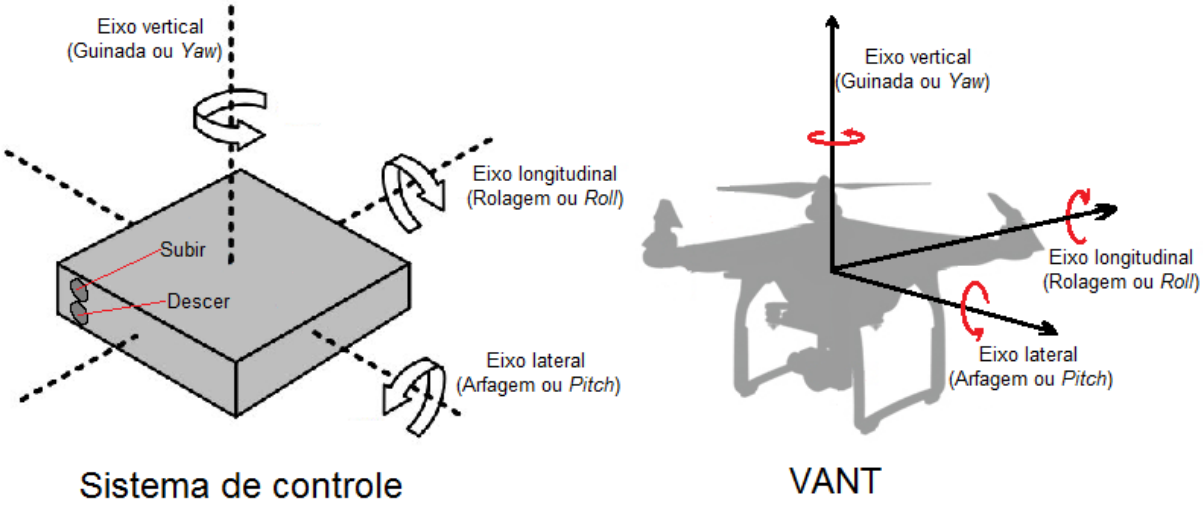


Figura 8 – Representação dos eixos de rotação de um corpo e os mesmos eixos em um VANT

A grande e imediata facilidade verificada com essa abordagem é aproximar o mundo da aviação com o cotidiano mais próximo do usuário. A manipulação por inclinação é bem mais intuitiva do que os controles normais de uma aeronave não tripulada, tecnologia mais recente e menos conhecida. Além desse aspecto prático, pretende-se fazer uma estrutura em que a pilotagem possa ser feita com uma única mão, deixando a outra livre para outra utilização. Tal característica mostra-se impossível quando se trata de um controle remoto comum de um VANT, pois comumente são necessários dois *joysticks* (uma para cada mão) para controlar todos os seus 6 graus de liberdade. Como exemplo de facilidade obtida pode-se citar qualquer aplicação de campo em que se necessite realizar uma série de tarefas de alto risco e precisão ao mesmo tempo. O *drone* auxiliando operadores de manutenção de linhas de transmissão, como suporte em operações táticas e de supervisão e alguma tarefa simples de inventário que seja necessário anotações constantes são alguns exemplos em que o projeto irá ajudar.

O projeto foi pensado em ser realizado da forma mais rentável e de baixo custo possível. A ideia aqui, aliada com o fato de tornar o processo de manuseio mais fácil, é que essa implementação de controle seja usada em treinamentos de pilotagem, ajudando como forma de simulação. Pretende-se reduzir o custo e o tempo de treinamento desses cursos. Atualmente tanto para algumas aplicações mais críticas, como as atividades de policiais e de militares, quanto para uso em entretenimento e em publicidade, como cursos de fotografia e filmagem com *drones*, essa redução de tempo e investimento será muito bem-vinda.

Por último, a implementação do projeto irá aumentar a interação e imersão do usuário e poderá ser aplicada de forma lúdica ou até mesmo competitiva nas corridas de *drone*, que vêm ganhando espaço cada vez mais que seu custo de comercialização cai. A motivação principal aqui é reduzir o custo de treino para um piloto competitivo que, nos dias de hoje, para começar a pilotar em alto nível, precisa de um *drone* altamente equipado e de um óculos de realidade virtual apropriado para ele.

1.3 Objetivos do projeto

O objetivo final é conseguir substituir completamente o controle original de um VANT por um sistema *standalone* com um acelerômetro. Deseja-se também melhorar a curva de aceleração dos motores para que seja possível fazer um *soft-start* e melhorar a estabilidade e dirigibilidade do dispositivo.

O acelerômetro enviará as ordens de atitude que o usuário deseja que o VANT execute. Essa informação é processada e transmitida via *Bluetooth* para outra unidade de processamento que atua corretamente 4 potenciômetros digitais, que representam todo o controle, uma vez que vão substituir os *joysticks* originais do comando do *drone*. O valor de tensão variável resultante dos potenciômetros passa por um circuito controlador proporcional, construído por outros potenciômetros digitais, que farão a compensação para mudança de curva de aceleração, e, por fim, o sinal é enviado via rádio frequência para o VANT realizar a tarefa requisitada.

1.4 Cronograma

O fator tempo para a realização deste projeto é o crucial para que o mesmo seja bem-sucedido. Para tanto, deve-se definir bem as tarefas a serem realizadas, assim como os deadlines associados a elas.

A fase 1 durou um semestre, o segundo semestre de 2016, sendo parte integral do Trabalho de Graduação 1.

Nela compreende-se a parte de estudos bibliográficos e testes por prototipagem rápida dos sensores e atuadores principais para a realização do objetivo. Aqui destacam-se o estudo das rotinas de interrupção do Arduino, da IMU MPU 6050 e do potenciômetro digital X9511.

Também na fase 1, foi requerida uma maior caracterização dos potenciômetros originais do VANT, submetendo-o a testes de bancada para análise de como a tensão dos potenciômetros altera a velocidade angular das hélices. Além de substituir e testar o controle do VANT somente pelos potenciômetros digitais X9511 com o Arduino UNO.

A fase 2 deste trabalho é o período que engloba o primeiro semestre de 2017 e que finaliza o Trabalho de Graduação 2.

Nessa parte, o MPU 6050 terminou de ser corretamente implementado, estudou-se a configuração e o protocolo de comunicação dos módulos Bluetooth, fez-se a estrutura de

acrílico para o ‘volante’, estudou-se profundamente o outro potenciômetro digital, AD 5206, e foi feita a integração de todos os sistemas bem como a melhor otimização de hardware e software para a utilização do usuário.

1.5 Apresentação e estrutura do manuscrito

Daqui em diante... refazer imagem do fluxo do sinal.

Os capítulos deste trabalho estão organizados da seguinte maneira:

- ***Capítulo 1*** *– Introdução: Contextualização e motivação, apresentação dos objetivos do projeto e cronograma;*
- ***Capítulo 2*** *– VANT: entendendo o funcionamento de um VANT, especificações e tomadas de decisão para desenvolvimento do projeto;*
- ***Capítulo 3*** *– Hardware de atuação: Breve descrição do trabalho passado, modificações da placa original e desenvolvimento de circuito adicional;*
- ***Capítulo 4*** *– Software de atuação – Arduino UNO: Definição de funcionalidades e desenvolvimento;*
- ***Capítulo 5*** *– Hardware de controle: Estrutura mecânica e descrição do circuito elétrico;*
- ***Capítulo 6*** *– Software de controle – Arduino Pro Mini: Definição de funcionalidades e desenvolvimento;*
- ***Capítulo 7*** *– Ensaios e testes: Apresentação dos resultados obtidos dos ensaios e testes realizados durante e após o desenvolvimento;*
- ***Capítulo 8*** *– Conclusão: Resumo do projeto, conclusões e sugestões para trabalhos futuros.*

Capítulo 2

Veículos Aéreos Não-Tripulados (VANT)

“ Uma vez que você tenha experimentado voar, você andará pela terra com seus olhos voltados para céu, pois lá você esteve e para lá você desejará voltar. ”

– Leonardo da Vinci

O termo VANT, criteriosamente falando, é utilizado para fins não-recreativos, ou seja, é o termo utilizado em pesquisas, experimentos ou fins comerciais. O termo *drone* é mais informal e engloba qualquer objeto aéreo e não tripulado, sendo mais utilizado em um contexto de lazer e esporte, assim como o termo aeromodelo. De forma ainda mais específica, existem os RPAs, “*Remotely Piloted Aircraft*” (aeronave remotamente pilotada), que como o próprio nome sugere, são os VANTs que possuem um piloto remoto. Neste trabalho, os termos VANT e *drone* serão utilizados como sinônimos, visto que o modelo utilizado no projeto é um modelo de lazer e foi utilizado como objeto de pesquisa.

2.1 Syma X5SW

O projeto baseia-se na modificação do sistema de controle de um *drone*. Sendo assim, a fim de se evitar demasiado tempo para uma atividade não ligada ao tema do projeto, se fez necessária a aquisição de um modelo de mercado. O modelo escolhido, para se iniciar o trabalho foi o *drone* Syma X5SW (Figura 9), que se mostrou o equipamento com melhor custo-benefício para atender aos objetivos estabelecidos.

Figura 9 - Syma X5SW (manual do fabricante)

O Syma X5SW é um dos mais novos e avançados modelos entre os quadricópteros da série Syma X5. Está entre os mais dispendiosos dentre os de sua categoria, apesar da variação

de preço ser pequena. Considerado um excelente *drone* para iniciantes em aeromodelismo, foi o primeiro modelo a apresentar vídeo FPV (*First Person View* ou Visão em Primeira Pessoa) por conexão Wi-Fi. O Syma X5SW destaca-se principalmente por seu preço de mercado: aproximadamente U$60, contra preços exorbitantes de modelos semiprofissionais que chegam a U$1400, como o DJI Phantom 4.

Além do mencionado acima, o modelo conta com uma grande popularidade virtual, o que contribui na busca por informações e até mesmo por acessórios e peças extras caso necessário. Concluiu-se então, que as qualidades deste modelo de *drone* tornavam suas deficiências relativamente insignificantes e poderiam ser contornadas durante o projeto. Citando-as rapidamente: duração da bateria (3,7V 500mAh) extremamente baixa resultando em um tempo de voo de 6 a 8 minutos e do atraso na transmissão de vídeo (FPV) de cerca de 3 segundos, que torna bem complicado o controle feito exclusivamente por meio dela.

2.1.1 Funcionamento de um VANT

O VANT aqui considerado é um *drone* quadrotor, também chamado de helicóptero quadrotor. É uma aeronave que decola e é impulsionada por quatro motores. Esse tipo de veículo é caracterizado por ter asas rotativas. Para um melhor desempenho e algoritmos de controle mais simples, os rotores e hélices devem ser colocadas equidistantes. Quanto ao material, a fibra de carbono, recentemente, tornou-se popular devido à sua leveza e rigidez estrutural. Entretanto, o *drone* usado aqui é feito de plástico.

Com seu pequeno tamanho e capacidade de manobra ágil, esses quadricópteros podem ser pilotados tanto dentro de casa, quanto ao ar livre. Existem várias vantagens nos quadricópteros, comparando com os helicópteros: em primeiro lugar, um quadricóptero não exige ligações mecânicas para variar o ângulo das pás do rotor enquanto giram, isto simplifica a criação e manutenção do veículo; em segundo, o uso de quatro rotores permite que cada rotor individual tenha um diâmetro menor ou igual ao rotor do helicóptero, o que lhes permite ter menos energia cinética durante o voo. Isto reduz o dano causado quando algum de seus rotores tem sua rotação bruscamente interrompida, comprovando sua relativa robustez.

Na aviação é muito comum se falar em *Pitch* (Arfagem), *Yaw* (Guinada) e *Roll* (Rolagem). A arfagem é o movimento de rotação em torno do eixo lateral do avião. A guinada é a rotação no eixo vertical e, por fim, a rolagem, no eixo longitudinal, como mostra a Figura 10.

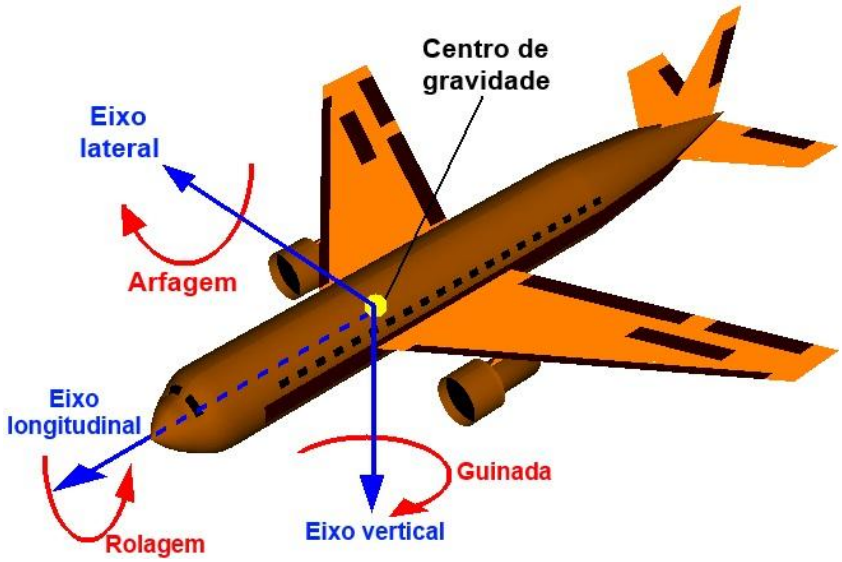


Figura 10 - Ilustração de movimentação de um avião *[7]*

No caso de um *drone*, o controle de movimento pode ser realizado variando-se a velocidade relativa de cada rotor para alterar o empuxo e o torque produzido por cada um. Para melhor entendimento, decidiu-se abstrair o *drone* da seguinte forma:

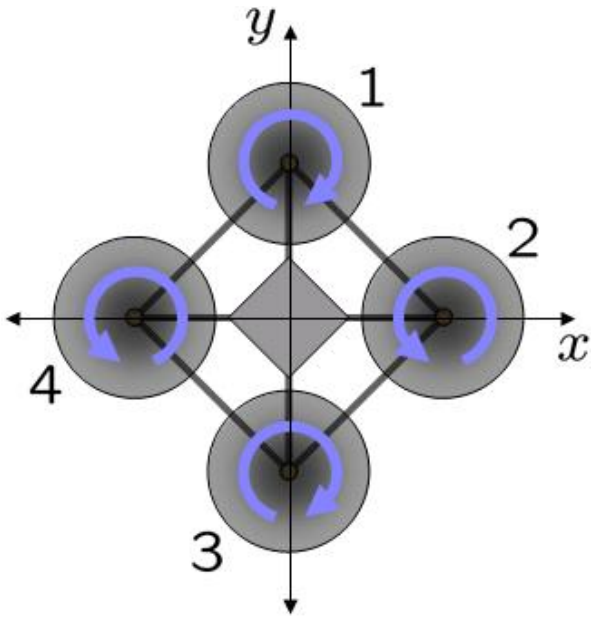


Figura 11 - Esquemático de torques de reação em cada motor. Os motores 1 e 3 giram em uma direção, enquanto os motores 2 e 4 giram no sentido oposto, produzindo torques opostos para o controle *[7]*

Cada motor produz um empuxo e o torque sobre seu centro de rotação. Se todos os motores giram na mesma velocidade angular, com os motores um e três girando no sentido horário e os motores dois e quatro, anti-horário, haverá torque líquido aerodinâmico. Consequentemente, a aceleração angular sobre o eixo de rotação é exatamente zero, o que implica que a guinada é nula é há uma ascensão reta sem rotação. Isso ocorre exatamente no caso representado pela Figura 11. [7]

Um quadricóptero ajusta sua guinada (gira ao redor de seu próprio eixo) aplicando mais pressão nos rotores que giram na direção de rotação desejada. Veja a Figura 12.

Figura 12 - Guinada no sentido horário sendo executada *[7]*

Um quadricóptero ajusta sua rolagem (vai para a direita ou esquerda) ou arfagem (para frente ou para trás), aplicando mais pressão para um rotor e menos impulso ao seu rotor diametralmente oposto. Veja a Figura 13 para mais detalhes.

Figura 13 - Exemplo de rolagem ou arfagem sendo executada *[7]*

Entendendo o funcionamento geral de um VANT, torna-se possível focar nos aspectos específicos do Syma X5SW.

2.1.2 Câmera do Syma X5SW

O Syma X5SW possui uma câmera de 2MP. A transmissão FPV é feita utilizando o aplicativo chamado SYMA FPV disponível na *Google Play* e na *Apple Store*. O *drone* gera uma rede Wi-Fi própria que deverá ser achada e conectada no telefone onde o aplicativo foi instalado. Após a conexão na rede, abre-se o aplicativo e aperta-se o botão START. Se tudo estiver correto, é mostrada a imagem da câmera com o pequeno atraso de transmissão já mencionado.

O aplicativo (Figura 14) oferece para o usuário um *pallet* na parte superior da tela com opções de gravar vídeo, salvar o *frame* corrente, verificar o que foi salvo e gerenciar os arquivos (excluindo-os ou transferindo-os para a memória do celular) e, por fim, um ícone que mostra se a rede Wi-Fi está ou não conectada.

Figura 14 - Tela inicial do aplicativo SYMA FPV (manual do fabricante)

2.1.3 Baterias do Syma X5SW

O Syma X5SW possui um compartimento para sua bateria. Ele fica na sua parte de trás (sempre considerar que a frente do *drone* é para onde a câmera está apontando, isso é de suma importância quando ele é controlado) e pode ser facilmente aberto pressionando uma pequena peça de plástico. O compartimento é relativamente pequeno tanto em comprimento quanto em largura e o encaixe da bateria é relativamente difícil de manipular, como pode ser visto na Figura 15.

Figura 15 - Compartimento da bateria

Ao todo têm-se 7 baterias disponíveis para alimentar o *drone* (Figura 16 e Figura 17):

- 1 de 500 mAh (original do *drone*);
- 4 de 600 mAh (compradas separadamente);
- 2 de 1100 mAh (compradas separadamente).

As quatro baterias de 600 mAh foram pedidas para aumentar a autonomia de voo. Já as duas de 1100mAh, também compradas a parte, foram dedicadas somente para os testes de bancada para levantamento do gráfico de rotação por tensão das hélices e também para os testes de modificação dessas curvas. Essas foram escolhidas por serem compatíveis com o *drone*, mas, por não caberem no compartimento de bateria, tornou-se inviável o voo com essa área aberta.

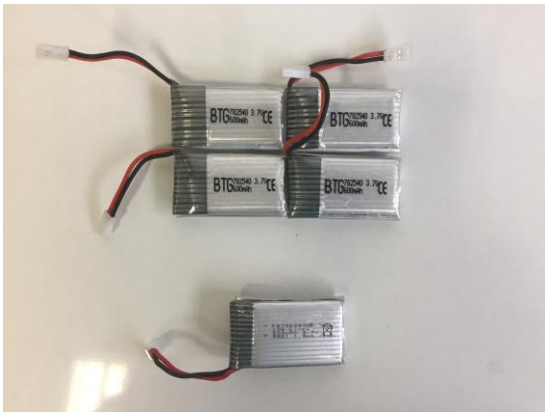


Figura 16 - 4 baterias de 600 mAh (parte superior) e a bateria original de 500 mAh (parte inferior)

Figura 17 - 2 baterias de 1100 mAh

Foram adquiridos posteriormente também um carregador USB de 4 portas, além do carregador unitário que acompanha na caixa original do *drone*, vide Figura 18.

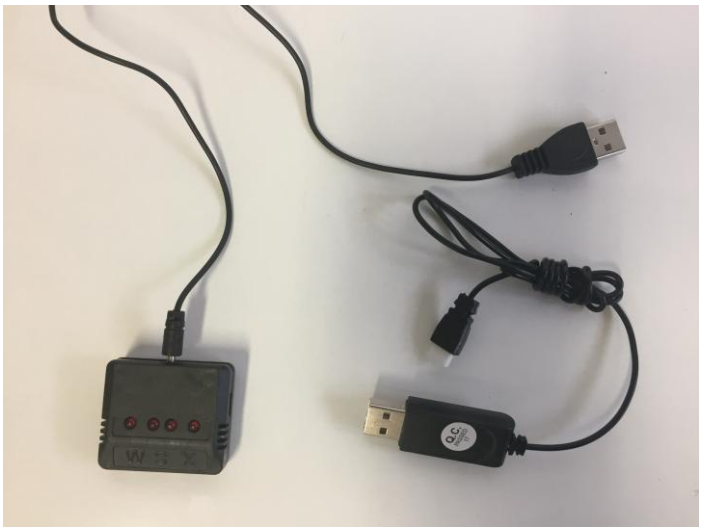


Figura 18 - Carregador de 4 baterias e carregador unitário

2.1.4 Aspectos adicionais do Syma X5SW

O *drone* possui 4 linhas de LEDs abaixo de cada um de seus braços, 2 verdes e 2 vermelhos. Essa sinalização luminosa é muito importante para voos noturnos e para verificação de que a sincronização com o controle remoto foi feita corretamente. A rotina de sincronização no controle remoto será abordada a seguir. No momento é importante saber que enquanto não tem nenhum controle emparelhado, os LEDs ficam piscando, já quando há a sincronia eles ficam acesos constantemente.

Além dos LEDs, é possível verificar, nas 4 hélices, uma proteção de plástico branco. Com o decorrer do trabalho, muitos testes foram necessários e houve muitas colisões do *drone* com diversos objetos, danificando alguma dessas proteções. Foi verificado que a ausência de alguma dessas peças alterava o centro de massa do equipamento e alterava a estabilidade dele, dificultando o andamento do projeto. Decidiu-se, portanto, retirar todas as proteções. Isso não significou grande prejuízos de segurança para o VANT pois ele mostrou-se bem robusto, apesar de leve e de aparência frágil.

2.1.5 Controle remoto

O controle original é este mostrado na Figura 19 :

Figura 19 - Controle do *drone* Syma

Os comandos convencionais do controle de direção e altitude são executados pelos dois *joysticks*, esquerdo e direito. O *joystick* da esquerda controla a potência dos motores, utilizado para aumentar ou diminuir a altitude do quadricóptero, além de executar a sua rotação em torno do seu próprio eixo. Enquanto isso, o da direita controla a movimentação no que se pode chamar de plano x-y: para frente e para trás, para a direita e para a esquerda.

O controle também conta com alguns outros botões que realizam diversas outras funções, tais como: *loop* de 360 graus, seleção de velocidades, troca de modo de voo, entre outros. Um esquemático dessas funcionalidades e seus respectivos botões pode ser verificado na Figura 20, bem como a antena, o botão liga/desliga, a tela LCD, a luz indicadora e os *joysticks*.

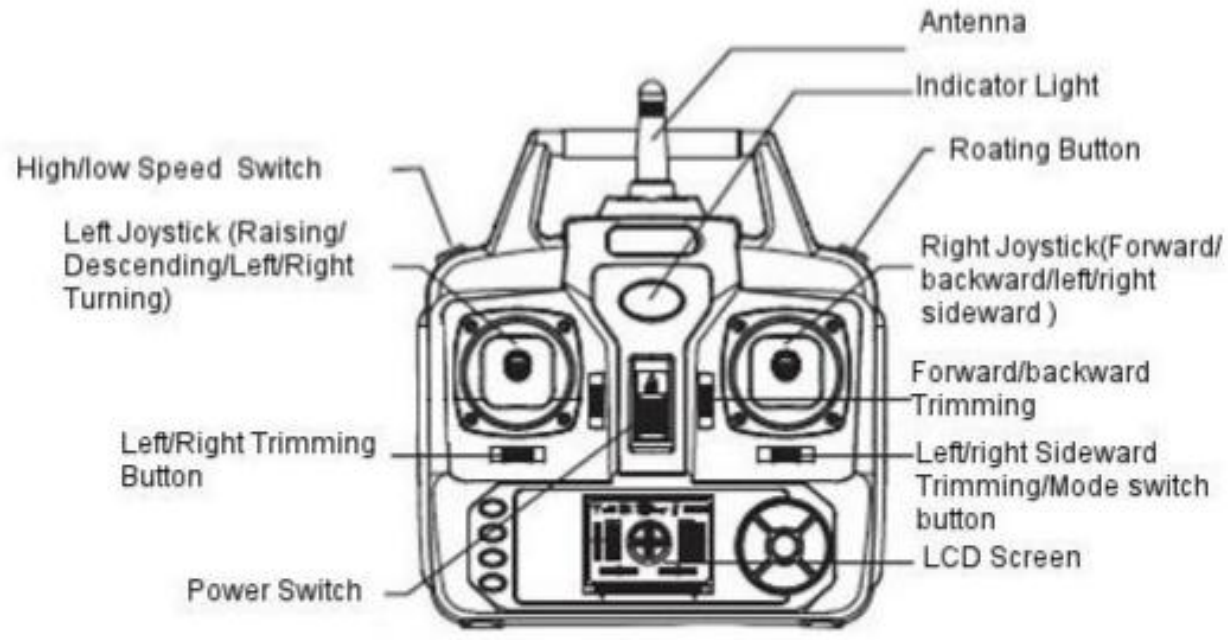


Figura 20 - Esquema indicador de funções de controle (manual do fabricante)

Das principais funções deste controle, destacamos os 4 potenciômetros, 2 à esquerda e 2 à direita. Três deles são retráteis, ou seja, sempre voltam para sua posição média. O único não retrátil é o potenciômetro vertical da esquerda, que controla a altitude do *drone* e que também é responsável pela rotina de sincronização dele.

Também é possível notar na Figura 20, os 8 botões chamados de *Trimming*. "Timmar" um avião, resumidamente na aviação, significa informá-lo sobre as condições externas, das quais ele não sabe a priori, como uma corrente de ar lateral, por exemplo. O avião, sabendo de tais alterações, atua no comando de seus motores para corrigi-las, mantendo-o estável. É possível testar essa funcionalidade apertando, por exemplo, o *left/right Sideward Trimming* muitas vezes para a direita e ligando-se o *drone* para voar só para cima. Vai ser possível ver que ele fica "puxando" para a direita. Se houvesse uma força à sua direita de mesmo módulo simulando a tal corrente de ar por exemplo, o *drone* voaria como desejado, ascendendo reto para cima. Uma forma rápida de fazer isso é apertar o botão *Trimming* e segurá-lo até ouvir um bip. Será possível verificar no LCD, nas barras laterais que ele está devidamente "trimmado". Portanto, uma boa prática de trabalho que foi adotada para os testes foi sempre ajustar os *Trimmings* do *drone* de acordo com o nível de bateria (lembrando que a tensão na bateria altera significativamente o comportamento do VANT) ou se acontecer de se notar que algum motor não está em equilíbrio com os demais, sempre com o objetivo de manter o voo ascendente em linha reta. Esse último exemplo ocorreu com esse projeto, pois foi preciso trocar um dos motores (o da hélice 1 – vide *drone* com a numeração) durante o desenvolvimento do trabalho.

Para finalizar as aplicações mais relevantes do controle, o *Roating Button* é o botão que permite com que o *drone* entre numa rotina de realização de um *loop* de 360 graus em torno do seu próprio eixo. Como quando ele fica de cabeça para baixo, perde-se sustentação e ele cai um pouco, então é aconselhado que essa manobra seja feita em um lugar aberto e em uma altura maior que o usuário. Recomenda-se que mantenha o *joystick* vertical da esquerda sempre para cima para que ele ganhe sustentação novamente e continue voando após realizar a volta no ar.

Dentre todas essas funções, apenas o botão liga/desliga, os controles de movimentação e a tela LCD serão, de fato, utilizadas no presente trabalho. Isso porque são elas que realizam as atividades fundamentais de todo sistema de controle.

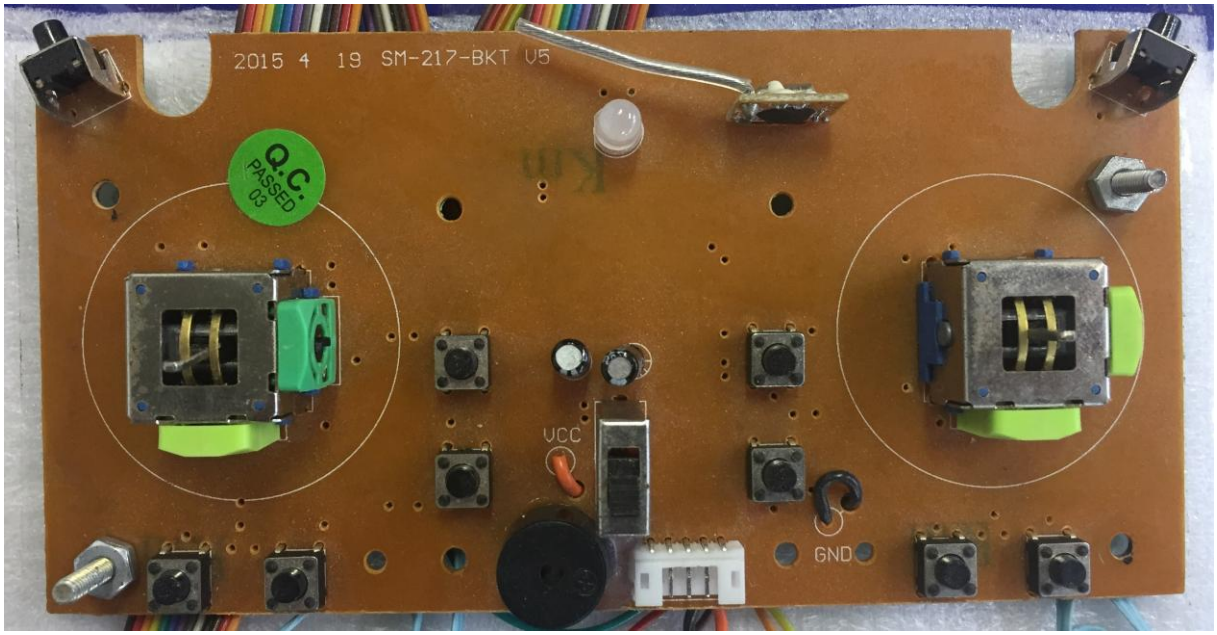


Figura 21 - Placa de controle original. Vista superior

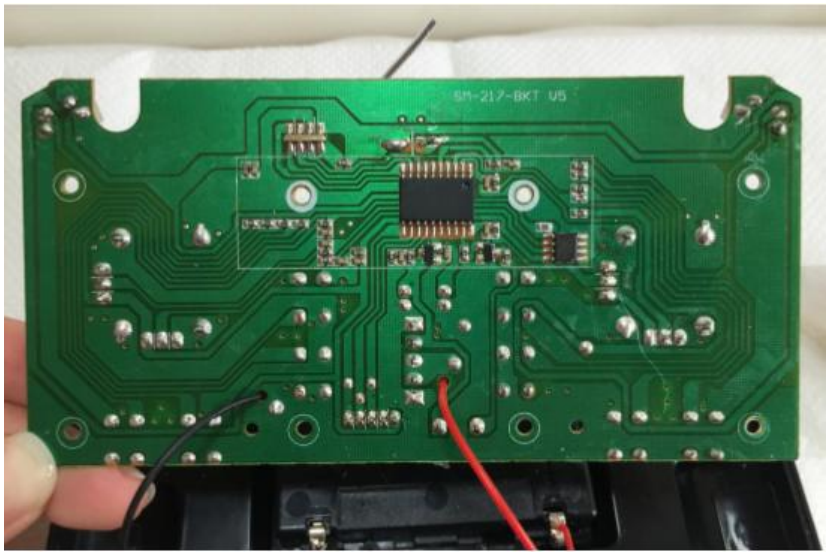


Figura 22 - Placa de controle original. Vista inferior

Figura 23 - *Drone* por dentro

Olhando as placas, do controle remoto e interna do *drone* (Figura 21, Figura 22 e Figura 23), de forma mais atenta, verifica-se que o *design* é bem organizado e, predominantemente, de lado único, o que é esperado, já que torna a produção mais barata. A placa do controle, foco relevante de estudo nesse trabalho, conta principalmente com um micro controlador Winbond 8 bits da série W79E804 e um chip de memória EEPROM.

2.1.6 Sincronização com o controle remoto

A sincronização é feita da seguinte forma: liga-se o *drone* primeiro em um *switch* vermelho, que fica na parte inferior dele, ao lado da câmera. Depois, liga-se o controle acionando o *Power Switch* para cima.

O *Indicator Light*, que é um LED de cor azul, acende e fica piscando numa dada frequência, assim como os LEDs presentes nos braços do *drone*. Desliza-se então o *joystick* vertical da esquerda totalmente para seu ponto superior, ouve-se então um bip de um *buzzer* interno, o LED azul passa a piscar numa frequência menor. O mesmo ocorre com os LEDs do

drone. Por fim, move-se o mesmo *joystick* todo para seu ponto inferior e ouve-se outro bip. A luz azul permanece sempre acesa, bem como as 4 linhas de LED presentes nos braços do *drone*. O visor LCD é aceso e passa a mostrar o comportamento do *drone*.

2.1.7 Especificações Gerais

Para resumir todas as informações acima, seguem as especificações técnicas do *drone*, Syma X5SW: [8]

- Dimensões: 310x310x105mm;
- Tamanho do motor: 8x20mm;
- Peso: 120g (com câmera e bateria);
- Tempo de voo:
    - De 6 a 8 minutos;
    - Aproximadamente 4 minutos com a câmera ligada.
- Bateria: 3,7V 500mAh LiPo;
- Tempo de carga: 40-45 minutos;
- Distância de controle: aproximadamente 50m;
- Transmissão em 2.4GHz;
- Bateria do transmissor: 4x1,5V AA;
- Câmera de 2MP;
- Transmissão WiFi em tempo real – FPV.

2.2 Tomadas de decisão para o projeto

O objetivo deste trabalho é o desenvolvimento de um novo sistema de controle de um VANT. A ideia inicial de automação seria possibilitar ao usuário pilotar o *drone* utilizando um novo controle que possui um acelerômetro interno, de tal forma que a manipulação seja feita por meio de inclinações nos três eixos nesse novo controle.

Esse projeto é continuação de outro Trabalho de Graduação finalizado em junho de 2016. Nessa pesquisa tinha-se o objetivo de fazer uma aplicação mobile para Android em que os manetes do controle original eram simulados. Os dados do telefone eram enviados via *Bluetooth* para um Arduino Uno que tratava a mensagem e acionava adequadamente um conversor AD simples montado. Essa conversão analógico-digital substituía a função dos potenciômetros analógicos da placa original do controle remoto. A placa original foi, então, modificada para se reaproveitar a transmissão de Rádio Frequência (RF) com o *drone*. A Figura 24 ilustra o caminho do sinal até chegar no VANT.

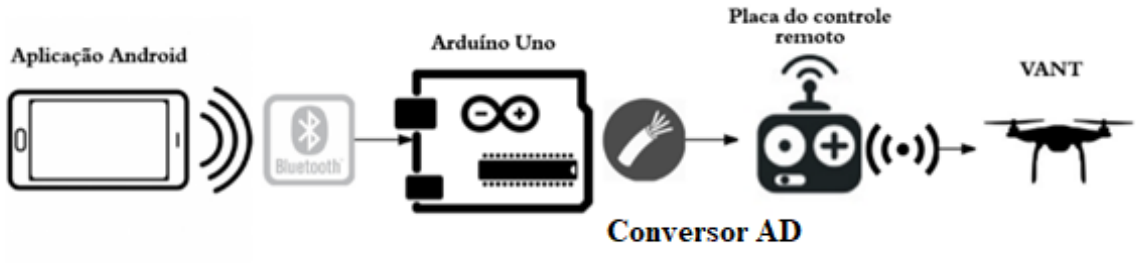


Figura 24 - Esquemático de funcionamento do sistema do Trabalho de Graduação passado *[2]*

Esse projeto cumpriu parcialmente com o prometido, pois conseguiu-se criar os *joysticks* virtuais no celular, mas a dirigibilidade estava muito a quem do esperado.

O controle com o dedo na tela sensível mostrou-se muito impreciso e a imperícia de um usuário comum fazia com que o drone não levantasse voo corretamente. Outro gargalo importante foi o conversor AD. Feito com pulsos PWM do próprio Arduino e filtrado por um filtro RC simples, ele é muito simples e nada robusto para a aplicação.

A motivação então para esse projeto foi abandonar, por hora, o uso do celular e fazer uma estrutura nova como um novo controle. A característica adicional de inovação é que esse novo controle vem com uma plataforma inercial embarcada, possibilitando a pilotagem do avião através dos movimentos de angulação da mão. Pretende-se também melhorar o conversor AD, interface que gerencia os sinais de controle: antes vindos do celular e agora, do novo controle com acelerômetro. Essa melhoria foi feita trocando os potenciômetros analógicos originais por potenciômetros digitais.

Não se aproveitou muito do trabalho passado, somente a ideia de modificar a placa original do controle remoto para ter fácil acesso a comunicação RF.

Esse projeto vai ter duas grandes frentes: O sistema de controle e o sistema de atuação. O sistema de controle será uma caixa fechada com uma Unidade de Medida Inercial (IMU). Será embarcado também nele um Arduino Mini Pro, que adquire e trata os sinais de movimento vindos da Plataforma Inercial. Adiante, essa informação é mandada, via *Bluetooth*, para o sistema de atuação. O sistema de atuação, bem parecido com a ideia do trabalho passado, terá um Arduino Uno, cuja função é a de receber as mensagens do sistema de controle, comparar o estado atual dos potenciômetros com o estado desejado recebido e atuar nos potenciômetros digitais que fornecem tensão para a placa original do VANT. A Figura 25 ilustra o caminho do sinal e a divisão lógica de cada sistema: controle e atuação.

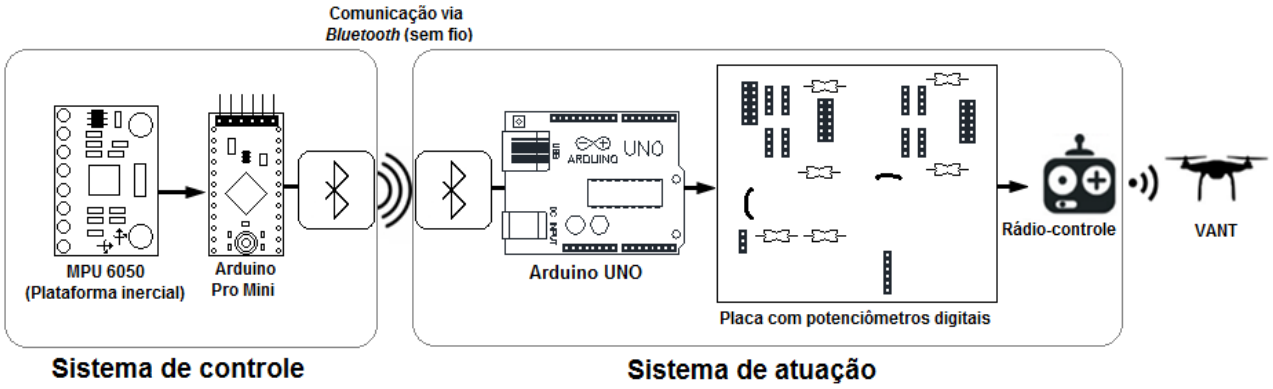


Figura 25 - Esquemático de funcionamento do sistema do presente trabalho

Essa metodologia de abordagem ainda carrega alguns problemas de dirigibilidade enfrentados pelo Trabalho de Graduação passado, empecilhos estes majoritariamente causados por características intrínsecas do *drone* Syma X5SW. Verificou-se que o foco desse conflito é o comportamento de rotação dos motores de acordo com a tensão enviada pelos potenciômetros. Ao longo desse projeto, pretende-se suavizar a influência negativa desse fator para se ter uma dirigibilidade mais fluida.

Capítulo 3

Hardware de Atuação

“ Eu não temo os computadores. Eu temo a ausência deles. ”

– Isaac Asimov

Tendo em vista que este projeto é a continuação de um projeto passado com objetivos similares, porém com meios diferentes, faz-se necessária uma breve explicação da parte elétrica utilizada nele. Até porque esse esquemático serviu de base para as novas decisões de projeto.

3.1 Placa antiga [2]

O foco do projeto anterior foi utilizar o Arduino UNO, construir alguns filtros, fazer algumas adaptações na placa para que se pudesse simular as funcionalidades básicas do controle e que se pudesse pilotar o *drone*. Em outras palavras, tinha-se o domínio remoto de seus *joysticks*. Essa ideia de possuir domínio remoto do *joystick* original do VANT foi mantida do trabalho anterior e também aplicada no presente trabalho.

3.1.1 Arduino UNO

Arduino, exemplificado na Figura 26, é uma plataforma de prototipagem de código aberto em hardware e software. Geralmente projetada com micro controladores Atmel AVR de 8, 16 ou 32 bits com suporte de entrada e saída embutido e linguagem padrão de programação semelhante a C/C++ essencialmente. Tem a finalidade de criar ferramentas acessíveis e de baixo custo, para que novos usuários tenham facilidade de se adaptar a ela e fazer uso de forma prática.

Figura 26 - Plataforma Arduino UNO *[9]*

O Arduino Uno é um dos modelos utilizados neste projeto. Este micro controlador é uma placa baseada no ATMega328P, e conta com 14 pinos de entrada/saída digital, dos quais 6 podem ser usadas como saídas de PWM (*pulse width modulation*), 6 entradas analógicas, um cristal de quartzo de 16 MHz, conexão USB, conector de energia e um botão de reset. A placa Uno é a primeira de uma série de placas Arduino USB e o modelo de referência da plataforma Arduino. [10]

O ATMega328P possui 3 temporizadores, sendo os temporizadores “0” e “2” de 8 bits e o temporizador “1” de 16 bits. Todos podem ser configurados para a geração de um sinal de

PWM em até 2 pinos da plataforma Arduino. Este processador também provê comunicação serial UART nos pinos digitais 0 (RX) e 1 (TX). Além disso, fazendo uso da biblioteca *SoftwareSerial*, é possível realizar comunicação serial em qualquer um dos pinos digitais do Uno. Os temporizadores são muito importantes para diversas aplicações pois regulam as interrupções de sistema, possibilitando com que várias tarefas sejam executadas em tempo real por um só processador. As interrupções serão tratadas mais a fundo no Capítulo 5. Informações mais detalhadas sobre pinos e registradores específicos podem ser achadas no Anexo 1.

O ambiente de desenvolvimento integrado (IDE) padrão para compilação do código é a Arduino Software e é disponibilizada no *site* da companhia. Por ela, também é feito o *download* do código para a plataforma.

Muitas destas configurações já vêm no padrão de fábrica e, portanto, são automaticamente ajustadas. Elas seriam, por exemplo, a definição do *clock* para 16MHz. A Arduino Software também é uma ferramenta imprescindível para a realização de testes: um monitor serial que, quando habilitado, permite que simples mensagens de texto sejam enviadas da placa e para ela, sendo muito útil para testes principalmente.

Verificar o Anexo 1 para o diagrama completo de pinos do Arduino UNO, amplamente utilizado nos programas utilizados na placa.

3.1.2 Potenciômetro Analógico

Potenciômetros são resistores cujo valor de resistência pode ser variado, girando-se um eixo que movimenta um contato móvel. Possuindo 3 terminais, a resistência entre suas 2 extremidades é fixa em seu valor nominal, enquanto que este valor entre uma extremidade e sua derivação central muda de acordo com a posição do segundo. A Figura 27 retrata o funcionamento acima descrito, onde A e B representam as extremidades fixas, e W seu cursor móvel. [11]

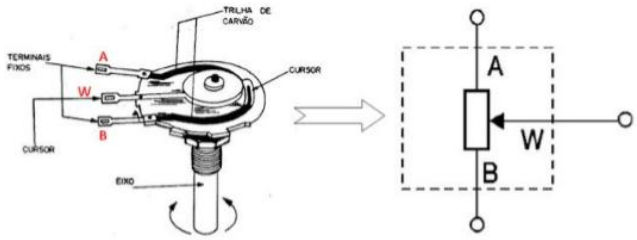


Figura 27 - Esquemático de um potenciômetro *[11]*

Por meio de testes com multímetro na placa original do controle remoto, verificou-se que os 4 potenciômetros analógicos, vistos na Figura 28, têm resistências nominais de 5KΩ e possuem os 2 pinos mais externos ligados um ao terra da placa e outro à uma tensão positiva que se verificou ser 3,3V.

Figura 28 - Potenciômetros analógicos originais

Para facilitar a descrição ao longo do trabalho, nomeou-se os 4 potenciômetros originais da placa como na Tabela 1.

Tabela 1 - Descrição das siglas para cada potenciômetro da placa original

Potenciômetros Originais	
JEV	*Joystick* da Esquerda Vertical
JEH	*Joystick* da Esquerda Horizontal
JDV	*Joystick* da Direita Vertical
JDH	*Joystick* da Direita Horizontal

Todos os potenciômetros retráteis (todos, menos o JEV) têm sua posição considerada como atuação nula em seu meio, ou seja, aproximadamente 1,65V. Os potenciômetros horizontais crescem seu valor de tensão da esquerda para a direita, já os verticais, de cima para baixo, ao contrário do que se pensa normalmente.

Essa peculiaridade que vale a ressalva é sobre os potenciômetros verticais, tanto da direita quanto da esquerda. Apesar de eles comportarem-se de forma linear assim como os demais, ambos, naturalmente na placa original, têm suas tensões trocadas. O quer dizer que, em seu ponto mais inferior, a tensão é máxima de 3,3V e, ao contrário, no ponto mais superior, a tensão é bem próxima de zero. Essa informação é de suma importância para entendimento do resto do desenvolver do projeto.

Visto que o Arduino não possui recursos para operar diretamente como um DAC (conversor digital-analógico), a ideia no trabalho passado foi de utilizá-lo em modo "*Fast PWM*" aliado a um filtro RC para gerar diretamente as tensões desejadas, de modo que fosse possível fornecer de forma analógica a tensão que seria gerada pelo movimento dos *joysticks*. Dessa maneira, desconectou-se os potenciômetros do restante do circuito da placa.

3.1.3 Sinal PWM na placa antiga

Na Figura 29 a seguir, as linhas verdes representam o período da onda PWM, cuja duração é de aproximadamente 2 ms, e *duty cycle* (ciclo de carga) é a mencionada proporção de tempo em que a onda está em seu valor máximo. *[12]*

Para calcular a tensão média em cada valor de ciclo de carga, basta multiplicar este pela tensão máxima aplicada, de 5 V para o caso do Arduino, ou seja:

$$V_{média} = V_{máxima} * Duty\ Cycle$$

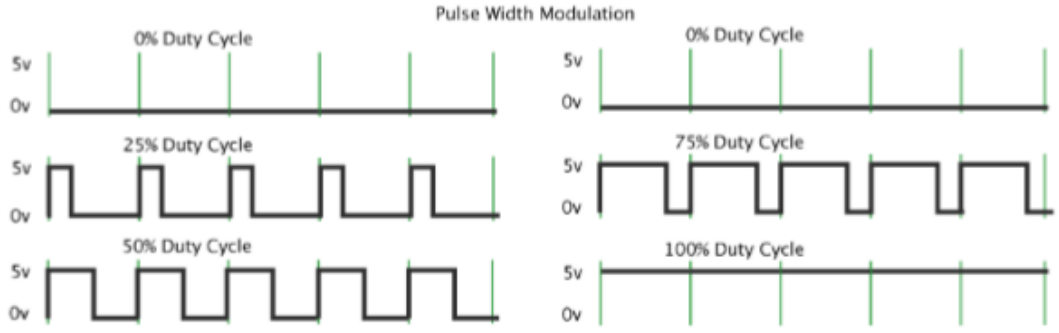


Figura 29 - Comportamento da onda PWM de acordo com a variação do duty cycle *[2]*

Os sinais vistos acima podem ser decompostos, cada um, em duas componentes: uma DC e uma outra onda quadrada de mesmo *duty cycle*, mas com uma amplitude média valendo zero. A ideia por trás da conversão DA (digital-analógica) visada no circuito é passar a onda gerada em um filtro passa-baixas para remover a maior parte dos componentes de alta frequência.

3.1.4 Filtro RC da placa antiga

A utilização de um filtro simples de primeira ordem já era suficiente para as necessidades do projeto. Outra vantagem do uso desse é a eliminação de possíveis harmônicos de alta frequência presentes na onda PWM.

Idealmente, um filtro passa-baixas possuiria um ganho constante e igual a 1 durante toda a banda passante F_{bw} e imediatamente cairia para zero assim que a atingisse seu limite. Tais filtros de alta precisão são extremamente caros e difíceis de se construir, então, por motivos financeiros práticos, montou-se um cujo comportamento é demonstrado e comparado com o ideal na Figura 30.

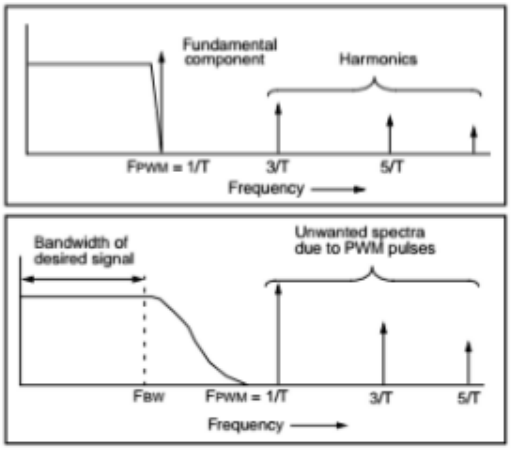


Figura 30 - Resposta em um filtro passa-baixas ideal *[2]*

Aqui o filtro passa-baixas consiste em um resistor de 10kΩ e um capacitor de 100nF conectados em série. Assim, a constante de tempo desse circuito é $\tau = 1$ ms e a resposta no tempo da tensão sobre o capacitor pode ser vista na Figura 31.

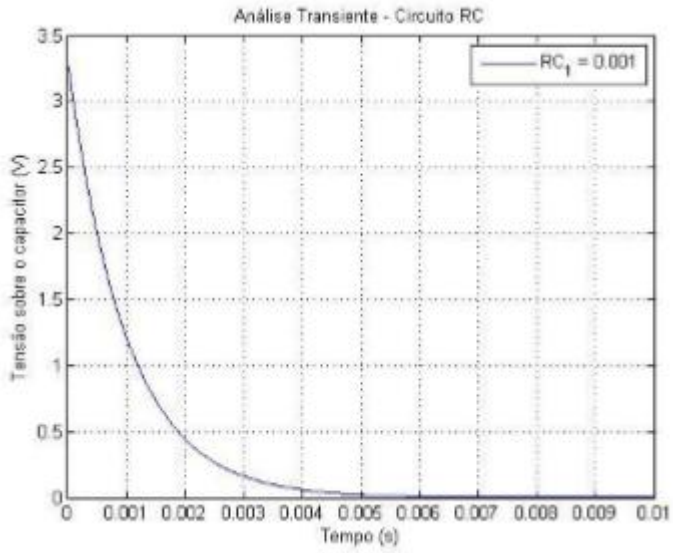


Figura 31 - Análise transiente do circuito RC [2]

Para se certificar de que a frequência da onda gerada F_{pwm} seja necessariamente maior do que a F_{bw}, basta configurar o *clock* do Arduino para operar em uma dada frequência grande o suficiente.

A frequência de banda passante é dada por:

$$F_{bw} = \frac{1}{RC} = \frac{1}{10 * 10^3 * 100 * 10^{-9}} = 100\ Hz$$

Na descrição das configurações em que o Arduino trabalha em nosso projeto, veremos adiante que a condição das frequências é respeitada com $F_{pwm} = 7{,}81kHz$. A Figura 32 mostra a FFT da onda PWM configurada nessa frequência, mostrando que, de fato, a constante de tempo do filtro é capaz de retirar quase toda a parte AC gerada pelo PWM.

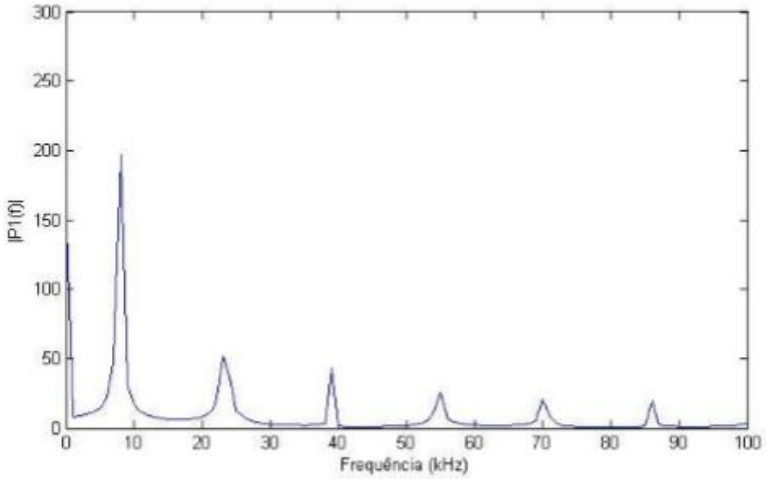


Figura 32 - Resposta em frequência do PWM [2]

3.1.5 Alterações na placa antiga

Removeu-se um trecho da trilha do circuito que liga o potenciômetro ao restante do controle na intenção de instalar um *switch* nos controles analógicos esquerdo e direito, para que se pudesse alternar entre controlar o sistema via Arduino e via *joystick* do controle original. Deste modo, a comparação entre saída esperada e saída real do circuito tornou-se extremamente mais simples e rápida. O que foi feito para instalação dos *switches* pode ser conferido na Figura 33.

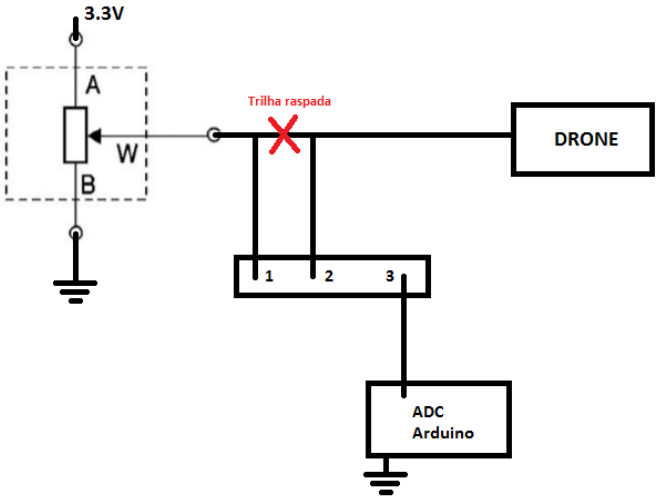


Figura 33 - Esquemático de funcionamento dos switches

Os *switches* podem ser verificados pelos conjuntos de três fios azuis claro, como ilustra a Figura 34. Eles possibilitam trocar facilmente entre o controle feito pelo conversor AD do Arduino, gerado pelo filtro RC ou continuar usando o sinal do potenciômetro analógico original da placa.

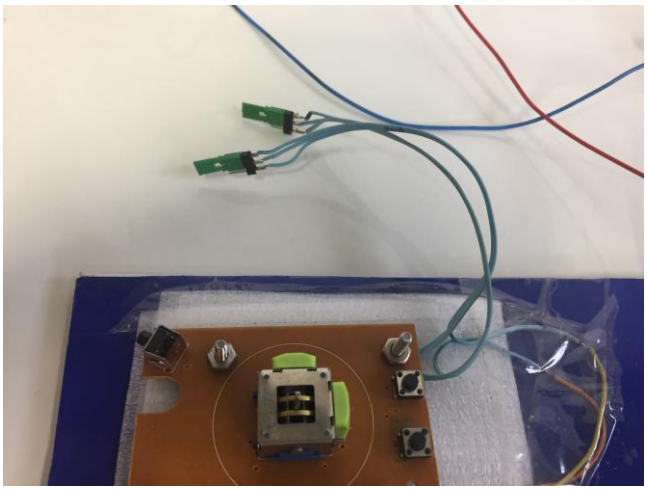


Figura 34 - Switches na placa antiga

A Figura 35 mostra a camada inferior da placa antiga modificada:

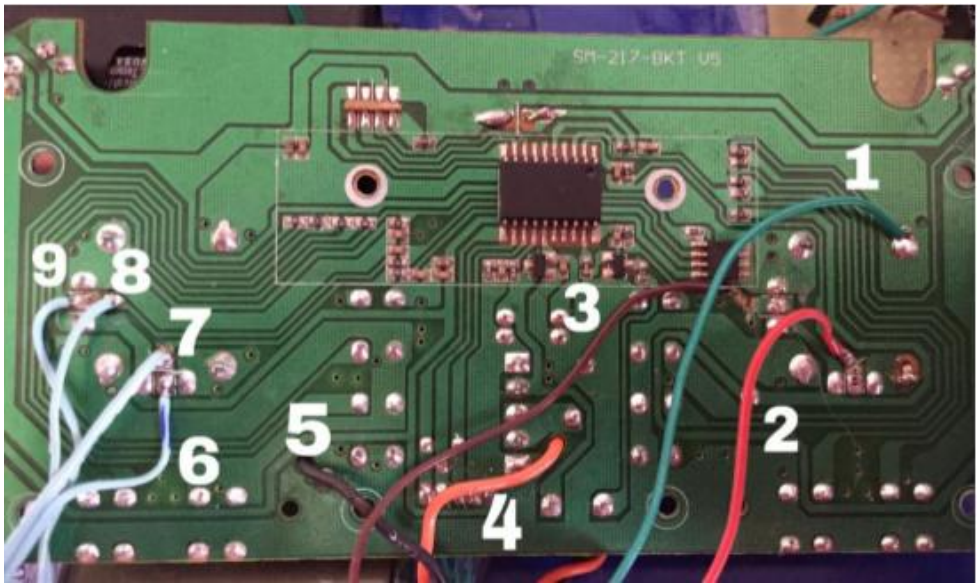


Figura 35 - Placa original com adaptações

Os fios numerados representam os seguintes sinais:

1. Terra da placa;
2. Sinal do *Joystick* da Esquerda Horizontal (JEH);
3. Sinal do *Joystick* da Esquerda Vertical (JEV);
4. VCC da placa – 6V – 4 pilhas AA de 1,5V cada;
5. Terra da placa, vindo da alimentação das pilhas;
6. Pino 1 da Figura 33 para o *Joystick* da Direita Horizontal (JDH);
7. Pino 2 da Figura 33 para o *Joystick* da Direita Horizontal (JDH);
8. Pino 1 da Figura 33 para o *Joystick* da Direita Vertical (JDV);
9. Pino 2 da Figura 33 para o *Joystick* da Direita Vertical (JDV).

3.1.6 Circuito adicional da placa antiga

A montagem do circuito adicional da placa antiga deu-se em uma placa padrão, onde todos os componentes, fios e filtros foram soldados para assegurar a resiliência do sistema. Para mais detalhes, conferir a montagem na Figura 37.

Esta placa conta com quatro circuitos que controlam sua respectiva direção de entrada única V_{in} e saída V_{out}: cada um com seu próprio filtro passa-baixas, Figura 36, mas possuindo uma referência comum a todos;

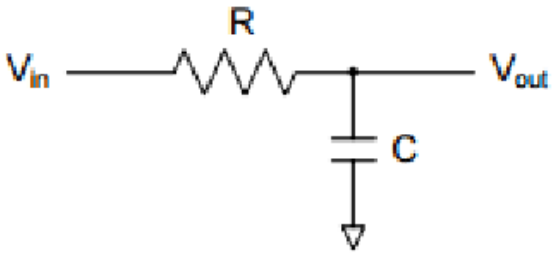


Figura 36 - Esquemático de um filtro passa-baixas de primeira ordem *[2]*

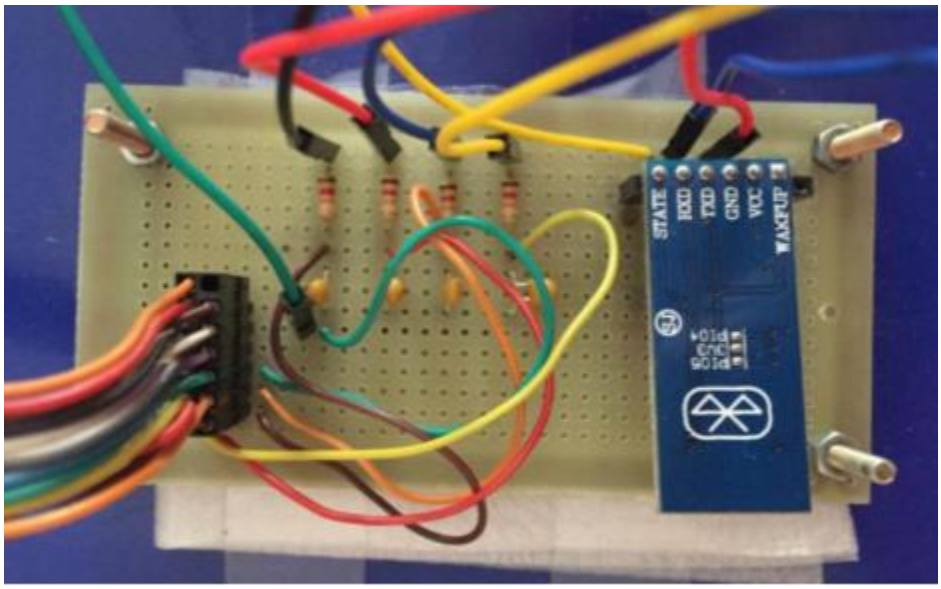


Figura 37 - Circuito adicional: filtros e módulo Bluetooth HC-05

O final da montagem do trabalho passado pode ser conferido no Anexo 2. Até mesmo para efeito de comparação com a placa nova.

3.2 Placa Nova

Como o novo objetivo do projeto é melhorar a forma de controle feita anteriormente, decidiu-se usar uma placa original semelhante ao do projeto passado e retirar por completo os quatro potenciômetros originais. No lugar deles, foram colocados 4 potenciômetros digitais, cujo controle será responsabilidade do programa no Arduino. Tanto o controle como o funcionamento mais exato desses dispositivos serão tratados mais à frente.

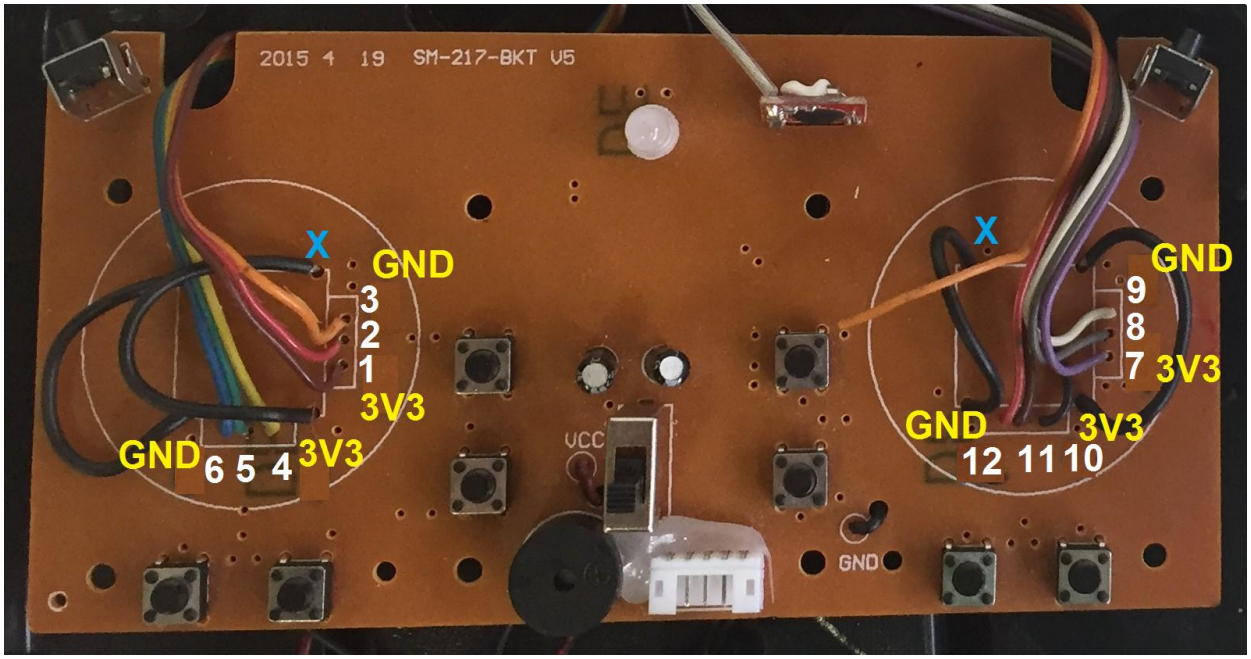


Figura 38 - Nova placa modificada

Na Figura 38, é possível notar 12 fios saindo da placa. Eles são os sinais de controle para os 4 potenciômetros digitais - 3 fios para cada (Analogamente, na Figura 27, aos terminais A, W e B).

- 1 ao 3 – JEV;
- 4 ao 6 – JEH;
- 7 ao 9 – JDV;
- 10 ao 12 – JDH.

Verificou-se que os pontos 1, 4, 7 e 10 têm tensão alta de 3,3V e os pontos 3, 6, 9 e 12 têm tensão baixa, em outras palavras, são o terra da placa. Os demais pontos são os pontos de controle de cada potenciômetro.

As cores de cada fio foram atribuídas a cada número obedecendo à norma internacional de código de cores, IEC 60062 [13]. Ela é usada para indicar valores ou classificações de componentes eletrônicos como resistores, capacitores, indutores e outros componentes.

Por fim, os dois X, em azul claro na figura, servem para lembrar a importância dos fios pretos, excluindo-se o do ponto 10, para o funcionamento da placa após a remoção dos componentes originais. Eles fazem as vezes das carcaças metálicas dos potenciômetros antigos. Essa estrutura era aterrada no GND da placa, fechando vários circuitos internos. Sem a adição desses fios pretos, a placa nem sequer liga.

A nova placa pode ser conferida no Anexo 3.

3.2.1 Embasamento Teórico do novo projeto

Na seção 3.1, foi descrito em detalhes o circuito eletrônico usando um sinal PWM para simular os sinais dos potenciômetros. Essa foi a ideia do Trabalho de Graduação anterior, inserida aqui para introduzir o contexto do problema. De agora em diante, concentra-se em detalhar bem a teoria por trás do hardware envolvido no novo projeto, destacando-se principalmente o potenciômetro digital x9511 e a Unidade de Medida Inercial, IMU MPU 6050.

3.2.1.1 Potenciômetro Digital

Como já fora explicado na seção 3.1.2, o potenciômetro é um componente que permite a variação da resistência em uma determinada parte de um circuito. Potenciômetros analógicos basicamente possuem uma trilha resistiva, em que um cursor condutivo caminha e fecha o contato em diferentes pontos da trilha.

Por ser analógico, o cursor possui um posicionamento contínuo ao longo da trilha, fazendo a conexão entre um terminal extremo qualquer e o terminal do cursor ter a possibilidade de atingir qualquer valor intermediário da resistência total da trilha.

O potenciômetro digital x9511 utilizado neste projeto, em essência se assemelha ao potenciômetro analógico, porém, o posicionamento do seu cursor, que é um contador digital, é comandado por um sinal digital e possui passos discretos ao longo de uma fileira de resistores. Veja um esquemático que detalha seu funcionamento:

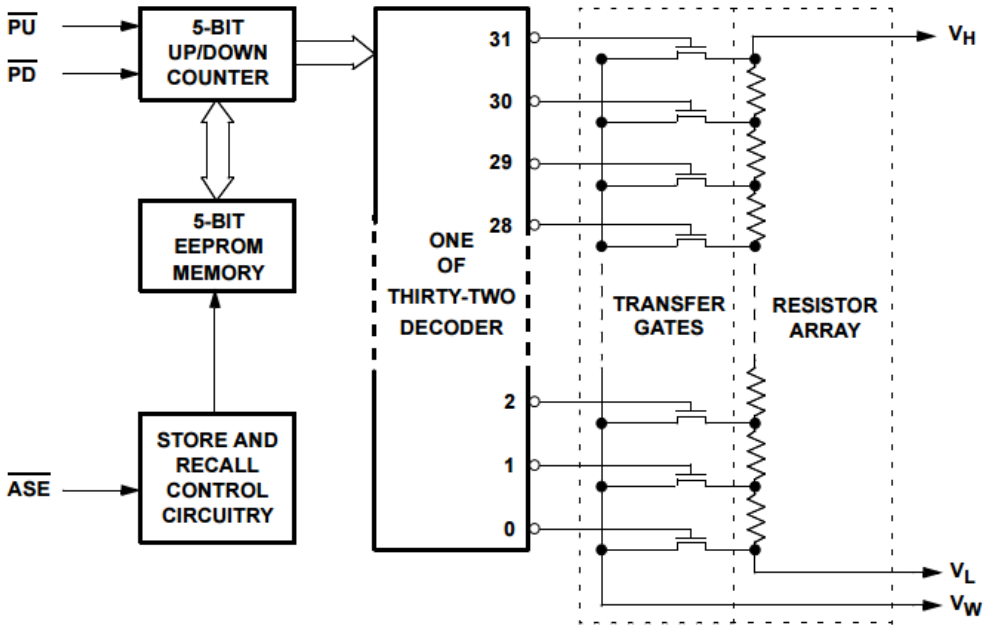


Figura 39 – Estrutura interna do potenciômetro digital x9511 (manual do fabricante)

A Figura 39 mostra o funcionamento do potenciômetro, que consiste em um contador digital de 5 bits (32 passos), um decodificador, que dada a entrada de 5 bits seleciona chave correta para se ativar, e uma fileira de resistores que fecha o circuito na posição que possui a quantidade específica de resistores.

Além disso, a Figura 39 também mostra suas entradas digitais: PU (*pull-up*), que incrementa o contador em uma unidade. PD (*pull-down*), que decrementa o contador em uma unidade. ASE (*autostore enable*), que armazena o valor atual do contador em memória, utilizado para não perder a posição atual caso o potenciômetro seja desligado. Os terminais V_L

e V_H que são equivalentes aos terminais da trilha de um potenciômetro analógico, e o terminal V_W equivalente ao terminal cursor.

Os sinais PU, PD e ASE, são logicamente invertidos, ou seja, internamente eles são conectados a um sinal digital HIGH (5V) e para sua ativação deve-se aplicar um sinal LOW (0V). Dessa forma, pode-se enviar sinais para suas entradas PU e PD que seleciona a tensão desejada em sua saída V_W. Essa tensão varia em passos discretos intermediários da tensão aplicada nos terminais V_H e V_L.

Para uma tensão de 3,3V aplicada aos terminais V_H e V_L, que é a tensão utilizada neste projeto, permite a tensão de saída em V_W variar de 0 a 3,3V em passos de 3,3/32 = 0,103125V.

3.2.1.2 Bluetooth

Os módulos *Bluetooth* foram implementados para realizar a comunicação entre os dois Arduinos. De baixo custo, os módulos *Bluetooth* dividem-se em dois: HC-05 e HC-06. A comunicação é feita por um perfil chamado SPP (*Serial Port Profile*), que é como uma porta serial sem fio.

Figura 40 - Módulo Bluetooth HC-05 *[14]*

O módulo HC-05, mostrado na Figura 40, é o módulo Master (pode também assumir o papel de *slave*). O início da conexão sempre parte deste módulo. Já o HC-06 é o módulo unicamente *Slave*, que se conecta quando o *Master* desejar. Visualmente é fácil de identificá-los: o HC-05 tem 6 pinos e o HC-06 tem 4. O modelo HC-05 possui algumas variações no mercado, podendo apresentar o controle do pino KEY ou WAKEUP através de um botão na placa ou um pino externo. A função desse pino é a mesma nos dois módulos: inicializar o módulo para a entrada de comandos AT.

Os comandos AT servem, dentre várias funções para: nomear o módulo, mudar sua senha de pareamento, mudar a velocidade de transmissão, definir o módulo como *master* ou *slave*, verificar o *status* corrente do módulo, entre outros. Para mais informações, consultar o *datasheet* do módulo.

No maior caso, no HC-05, são 6 pinos disponíveis para uso: STATE (não utilizado), WAKEUP (3V3 para ativação), VCC e GND (5V e 0V, respectivamente), TXD (saída do transmissor), RXD (entrada do receptor). Esses dois últimos, RXD e TXD, podem entrar em

qualquer porta digital do Arduino desde que uma comunicação serial virtual seja estabelecida neles. Nesse projeto usou-se a biblioteca *SoftwareSerial.h* para esse gerenciamento.

Foi projetado um divisor de tensão de duas resistências de valores iguais para a porta digital do Arduino responsável pelo pino de entrada RXD do módulo, para que ele opere com uma tensão abaixo de 3V3, como recomendado pelo fabricante. A mesma tensão de 3,3V é usada no pino de configuração WAKEUP. A mesma solução pode ser adotada ou até mesmo um divisor de tensão de três resistências idênticas, como foi usado em uma das placas desse projeto.

Algumas especificações dos módulos:

- Protocolo Bluetooth: v2.0+EDR;
- Firmware: Linvor 1.8;
- Frequência: 2,4GHz Banda ISM;
- Modulação: GFSK;
- Segurança: Autenticação e Encriptação;
- Banda de Onda: 2,4GHz-2,8GHz, Banda ISM;
- Corrente: Pareado 35mA; Conectado 8mA;
- Alcance: 10m;
- Baud Rate: 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1382400;
- Dimensões: 26,9 x 13 x 2,2mm;
- Peso: 9,6g.

3.2.2 Projeto Hardware Atuação

Neste novo hardware de atuação, o Arduino Uno tem a função de interpretar a palavra recebida pelo módulo Bluetooth que contém informações sobre qual passo cada potenciômetro digital deve estar (vide descrição do protocolo de comunicação no Capítulo 6). Os sinais de UP ou DOWN são enviados para os 4 CIs x9511 de acordo com a necessidade e, além disso, os seus terminais de saída V_L, V_H e V_W são os que serão enviados para a placa original, de acordo com a Figura 38 para que seja possível reutilizar a comunicação via rádio com o *drone*. O sinal do pino V_W também retorna para uma entrada analógica do Arduino, como feedback, para controle e monitoramento, em tempo real, do nível de tensão que se está sendo enviando aos motores.

Sendo assim, foi necessário o projeto de uma placa de circuito impresso, em inglês PCB (Printed Circuit Board), que fosse capaz de receber os dados via *Bluetooth*, mandá-los para o Arduino e que tivesse os 4 potenciômetros digitais com seus feedback e saídas para a placa original.

3.2.2.1 Esquemático Hardware Atuação

Destaca-se, isoladamente, as ligações de cada potenciômetro digital, uma vez que eles são parte importante do projeto. A Figura 41 ilustra como ele foi montado no projeto da placa.

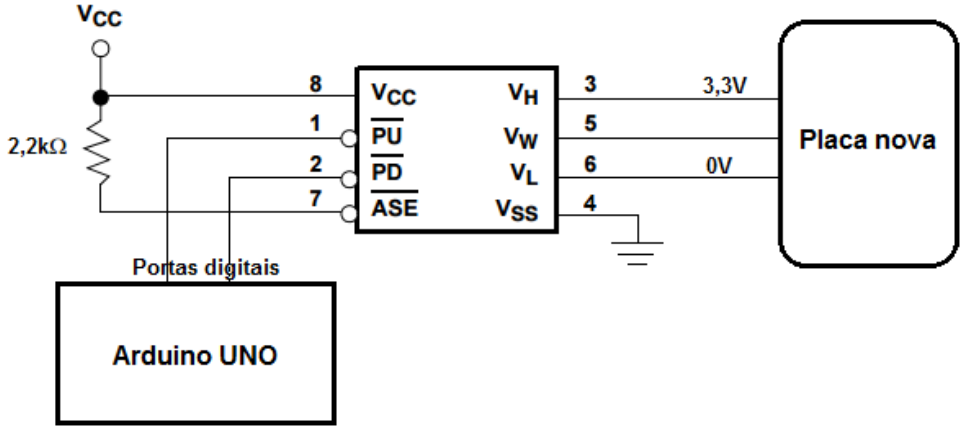


Figura 41 – Conexões de do potenciômetro x9511wp com a placa e o Arduino (manual do fabricante)

O Pino 7, ASE barrado, representa uma funcionalidade importante dentro do projeto. O pino de *Autostore*, quando muda de nível HIGH para LOW, armazena a posição atual do potenciômetro na EEPROM interna do CI, para, então, quando ele for religado, esse valor ser resgatado novamente para a saída V_W.

Essa aplicação foi necessária para fazer a calibração correta da placa com o *drone*, pois três dos potenciômetros originais do controle são, por construção, retráteis (sempre voltam para sua posição central). Considerando isso, a calibração só é necessária para o *joystick* denominado JEV, o único não retrátil, pois não se sabe em que posição, a priori, ele se encontra no momento de se ligar a placa. Por *default*, a placa original entende que, quando ligada, todos seus potenciômetros estão centralizados e procede-se subindo e descendo totalmente o JEV para calibração e sincronização com o *drone*. Em testes, foi verificado que, se algum dos potenciômetros retráteis não estiver na posição central no momento de ligar a placa, o sistema entende equivocadamente esse nível de tensão como a tensão central do *joystick* correspondente.

Com a substituição dos potenciômetros originais pelos x9511, este último então, sempre que a placa é reiniciada, precisa mandar um sinal intermediário entre 0V e 3V3. Isso foi feito somente uma vez na fase de projeto: colocou-se todos os x9511 com tal tensão desejada e tocou-se brevemente os pinos 7 de cada um deles com um GND, para que eles gravassem esse valor e, assim simulassem, ao ligar, a posição central dos *joysticks* retráteis.

O esquemático de todo o circuito é mostrado na Figura 42.

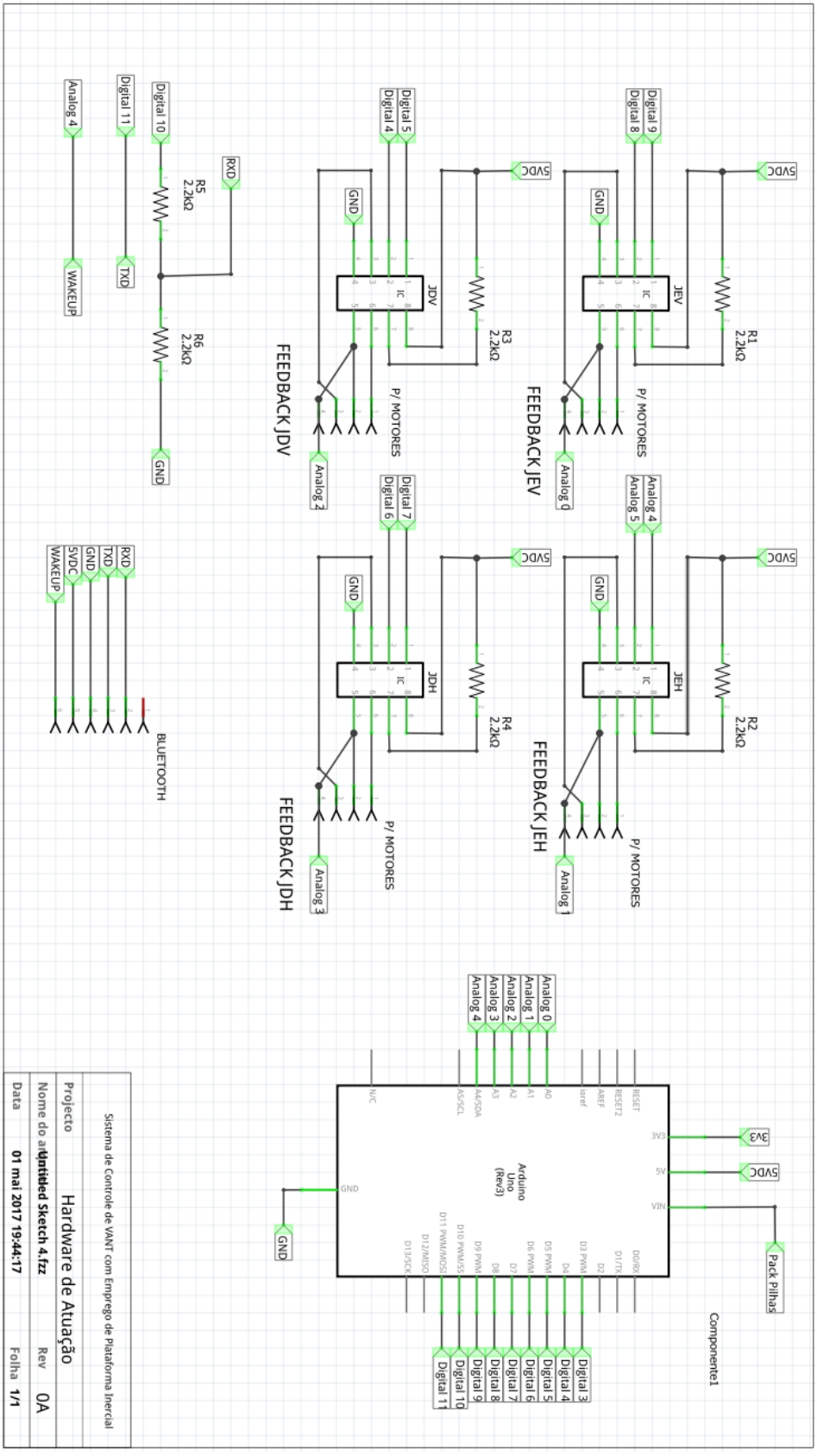


Figura 42 - Esquemático - Hardware de Atuação

Ao canto da Figura 42, pode-se se ler no campo “Rev” 0A. Isso diz respeito somente à versão do projeto. Esta é a primeira revisão do projeto, por isso denominada de 0A. Se alguma pequena alteração for feita nele como ajuste fino no circuito, como um remendo, recomenda-se mudar a letra, para uma versão 0B. Se alguma alteração que for feita mude ou acrescente funcionalidades ao projeto muda-se o número e volta-se para a letra A, na sequência então 1A. Sendo assim, esse termo não se refere a corrente elétrica, a sigla A na figura não representa a unidade de medida de corrente elétrica (ampere).

O protótipo foi previamente testado em *protoboard* e validado para montagem definitiva em uma PCB. Para efeito de comparação, essa montagem parcial pode ser conferida no Anexo 4, juntamente como sistema montado para o teste de rotação das hélices.

3.2.2.2 Placa de circuito impresso – Atuação

A fabricação demandou um pouco de cuidado por ser um processo sensível e feito integralmente de forma manual.

Inicialmente foi feito o roteamento das trilhas elétricas com base no esquemático. Utilizou-se aqui o software *Fritzing*. Essa etapa é crítica e de suma importância para se ter uma PCB que funcione com robustez. Aqui preocupou-se em compactar ao máximo o circuito e aumentar a espessura das trilhas (elas ficaram com uma espessura de 1,22 mm), para facilitar a passagem do circuito para a placa de cobre. Além disso cuidou-se para que os caminhos não tivessem cantos vivos de 90°, sendo todos chanfrados em 45°, e que as trilhas não ficassem dando voltas, pois seriam geradas interferências nos sinais que trafegam por elas pelo surgimento indesejado de indutâncias. O resultado pode ser conferido na Figura 43.

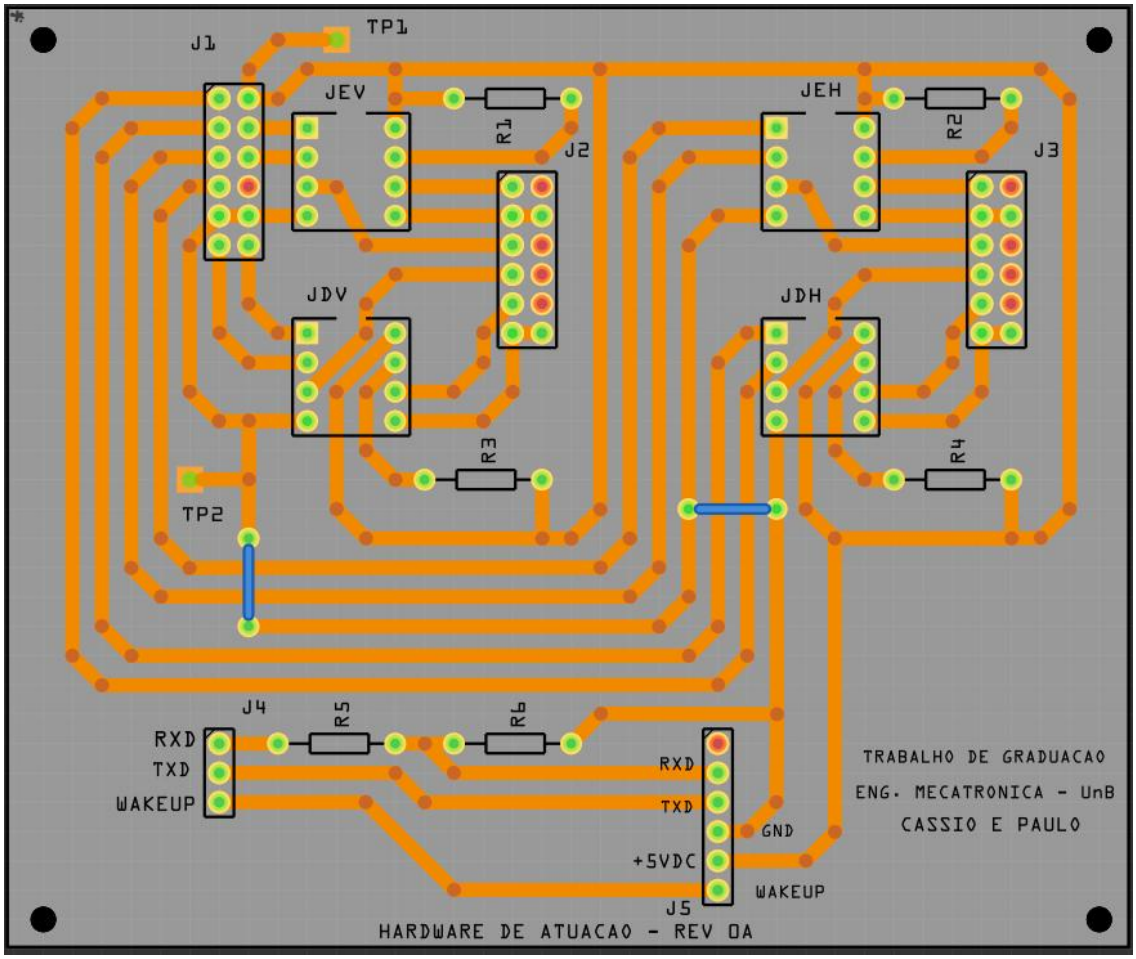


Figura 43 – Roteamento - Hardware de Atuação

Depois do mapeamento das trilhas via software, existem várias formas de fabricação de uma Placa de Circuito Impresso. A primeira decisão que se deve tomar é o material da placa que contém o cobre. Tinham-se duas opções disponíveis: a placa de fenolite e a placa de fibra de vidro. A escolha da placa de fenolite, apesar de mostrar que a aderência das trilhas nela é, de certa forma, complicada, foi tomada pelo baixo custo e pela facilidade na furação das ilhas, em detrimento das placas de fibra de vidro.

Escolhido o material, parte-se para a fabricação em si. Alguma das formas mais robustas são aquelas mais automatizadas. Dentre algumas, pode-se citar o uso de uma fresadora específica para impressão de circuitos eletrônicos diretamente na placa de cobre (esse processo necessita ainda passar pela corrosão do cobre restante). Ou até mesmo a fabricação usando luz ultravioleta que, em suma, permite gravar as trilhas na placa, deixando somente elas com o material condutivo (não existe processo de corrosão aqui). Infelizmente não foi possível utilizar nenhuma dessas tecnologias por não estarem disponíveis de forma rápida.

O processo de fabricação escolhido foi o de transferência térmica. Todos os materiais utilizados podem ser vistos na Figura 44. Esse processo necessita três etapas: a impressão do circuito em um papel, a transferência da tinta das trilhas para a placa de fenolite e a corrosão do material de cobre não protegido pela tinta depositada.

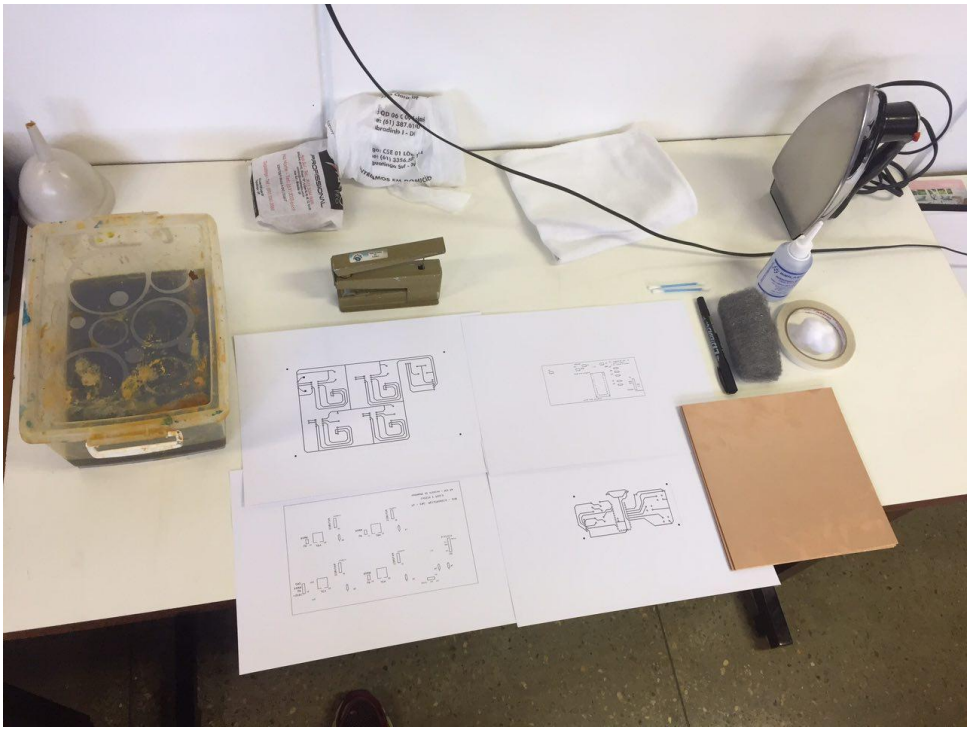


Figura 44 - Materiais para fabricação da PCB

Na primeira fase, a impressão deve ser a laser caso contrário não será possível transferir a tinta do papel para a placa de cobre, uma vez que uma impressão a jato penetra na textura do papel, dificultando seu desprendimento para outro meio. Quanto ao papel, existem mais de uma opção possível. A primeira seria a folha de acetato, porém deve ser folha de acetato específica para impressoras a laser, caso contrário, é possível que ocorra o derretimento dela no momento da impressão. A segunda opção é o papel que se cola o adesivo do papel contact, por ser um papel muito maleável a impressão é delicada. A terceira opção, que foi a escolhida para a confecção das placas neste projeto, foi o papel de fotografia. Ele possui um lado com textura lisa e reflexiva que se desprende facilmente durante a transferência para a placa de fenolite.

A segunda fase, que abrange a transferência do papel para a placa de fenolite, é a parte mais delicada de todo o processo. Como a opção escolhida foi a de transferência térmica, utilizou-se, então, uma chapa quente para esquentar o papel e fazer a tinta e um pouco do papel passarem para o fenolite. O que não foi possível passar ou ficou falhado, foi corrigido com um marcador permanente preto com ponta grossa. Depois de algumas tentativas, o circuito foi devidamente marcado na placa de fenolite. Deve se ter o cuidado com o espelhamento do circuito para que ele fique na orientação desejada no final.

Por último, na terceira fase, a placa com o circuito segue para a corrosão numa solução de água com percloreto de ferro. A função do percloreto é corroer todo o cobre onde não existe tinta. Esse processo pode demorar de 30 a 40 minutos dependendo de quantas corrosões a solução já realizou, já que ela envelhece com o uso e não com o tempo. A Figura 45 mostra essa placa nesse exato momento.

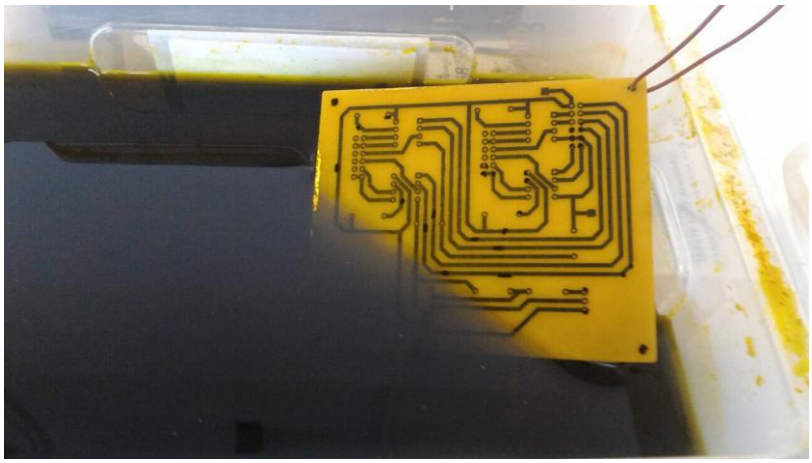


Figura 45 - Circuito de atuação no processo de corrosão

Após a corrosão, lixa-se a placa com palha de aço para tirar a superfície do papel de fotografia remanescente e aplica-se álcool isopropílico para limpeza final. Fura-se a placa com um grampeador adequado e segue para a soldagem dos componentes em seus devidos locais. A face superior da placa final pode ser conferida na Figura 46.

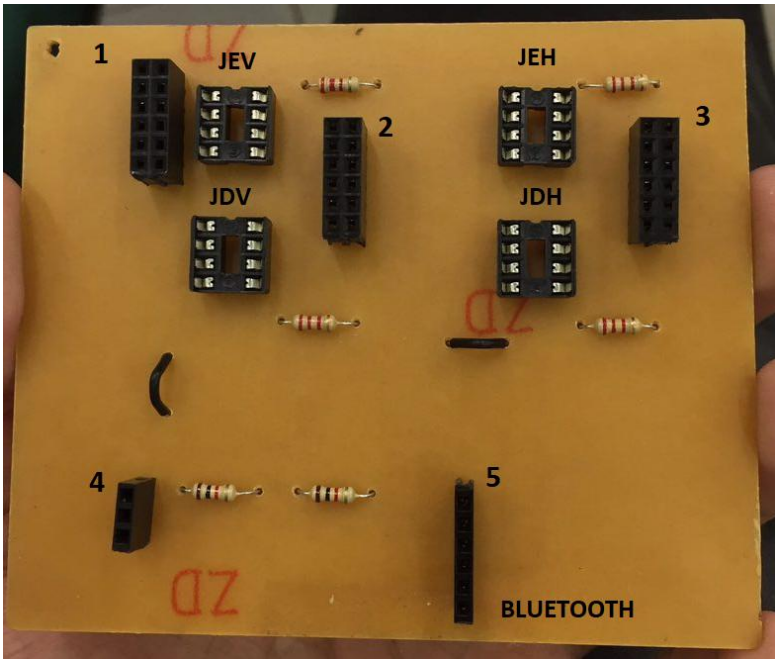


Figura 46 - Face superior - Hardware de atuação

Preocupou-se, durante o projeto, em fazer o devido mapeamento dos pinos de cada *header*, numerados de 1 a 5 na Figura 46, para que seja feita a instalação correta dos fios para o Arduino e para a placa original do *drone*. O pino 1 dos *headers* duplos é o primeiro a esquerda na parte de cima, os demais seguem como disposto nas tabelas.

Tabela 2 - Mapeamento Header 1

HEADER 1			
DESCRIÇÃO	PINO	PINO	DESCRIÇÃO
DOWN_JDH	1	2	ARDUINO_5VDC
UP_JDH	3	4	UP_JEV
DOWN_JEH	5	6	DOWN_JEV
UP_JEH	7	8	NDA
ARDUINO_GND	9	10	NDA
DOWN_JDV	11	12	UP_JDV

Tabela 3 - Mapeamento Header 2

HEADER 2			
DESCRIÇÃO	PINO	PINO	DESCRIÇÃO
LOW_JEV	1	2	NDA
WIPER_JEV	3	4	FEEDBACK_JEV
HIGH_JEV	5	6	NDA
HIGH_JDV	7	8	NDA
LOW_JDV	9	10	NDA
WIPER_JDV	11	12	FEEDBACK_JDV

Tabela 4 - Mapeamento Header 3

HEADER 3			
DESCRIÇÃO	PINO	PINO	DESCRIÇÃO
LOW_JEH	1	2	NDA
WIPER_JEH	3	4	FEEDBACK_JEH
HIGH_JEH	5	6	NDA
HIGH_JDH	7	8	NDA
LOW_JDH	9	10	NDA
WIPER_JDH	11	12	FEEDBACK_JDH

Tabela 5 - Mapeamento Header 4

HEADER 4	
PINO	DESCRIÇÃO
1	RXD
2	TXD
3	WAKEUP

Tabela 6 - Mapeamento Header 5

HEADER 5 - BLUETOOTH	
PINO	DESCRIÇÃO
1	NDA
2	RXD
3	TXD
4	GND
5	5VDC
6	WAKEUP

O projeto de atuação todo montado pode ser conferido na Figura 47. Nele o conjunto de pilhas, gerando 6V, alimenta tanto a placa original quanto o Arduino e a nova PCB quando o *switch* da nova placa é ligado.

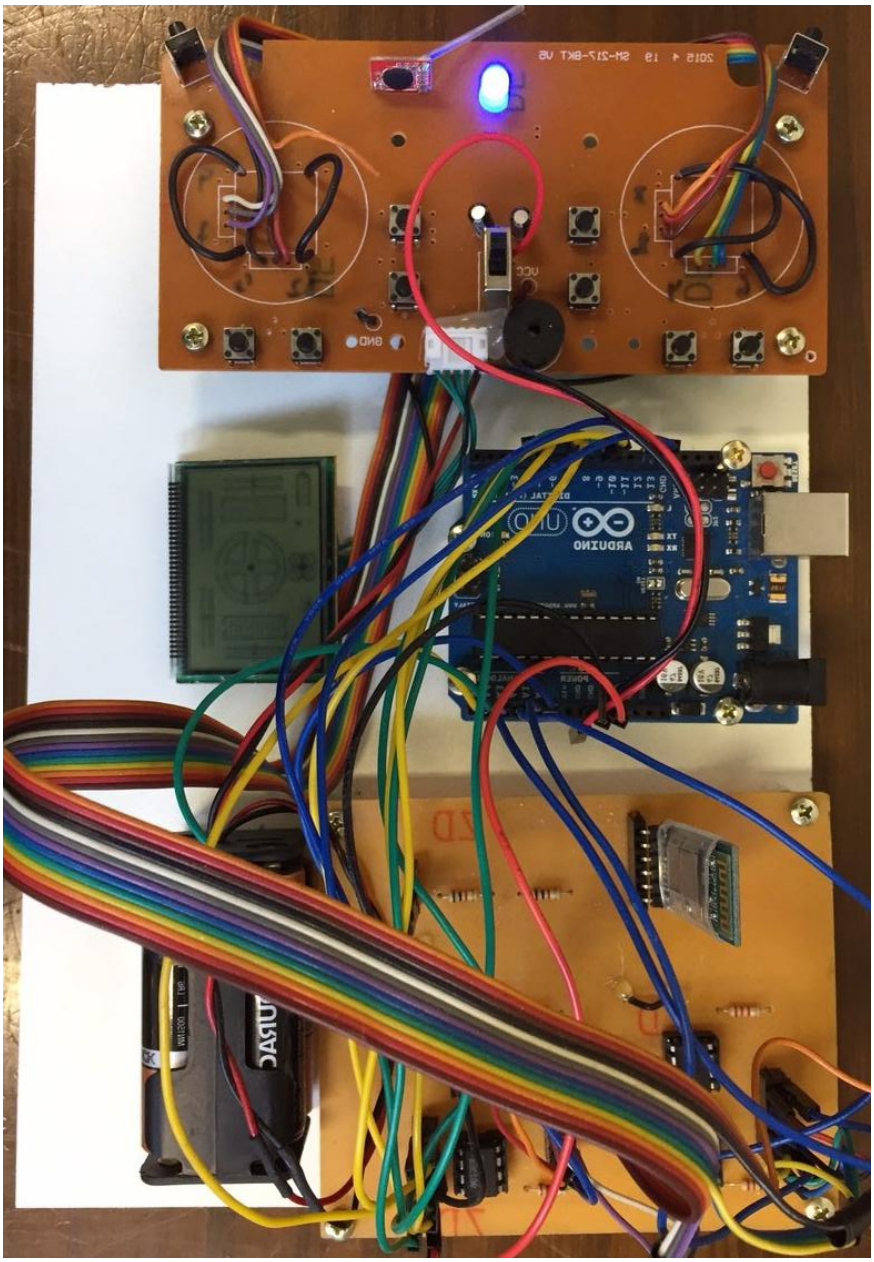


Figura 47 - Sistema de atuação completo.

Capítulo 4

Software de Atuação

“ Em todo o espaço há energia, é só uma questão de tempo até que os homens tenham êxito em associar seus mecanismos ao aproveitamento desta energia. ”

– Nikola Tesla

4.1 Embasamento teórico

O Arduino dispõe de muitas funcionalidades que garantem precisão quando se trabalha com intervalos de tempo e contagens. Dentre elas, as que convém serem detalhadas neste trabalho são os temporizadores e as interrupções.

4.1.1 Temporizadores

Os temporizadores são estruturas que permitem o programador configurar, baseado na frequência de relógio interno do próprio dispositivo, um intervalo de tempo bem definido e confiável para a execução de procedimentos periódicos ou um contador para se medir o tempo real decorrido em determinada aplicação. Sabe-se que a frequência do relógio do Arduino é de precisos 16 MHz, com isso, o programador pode escolher com um pré-escalonador a frequência que deseja utilizar para seu temporizador, como ilustra a Figura 48.

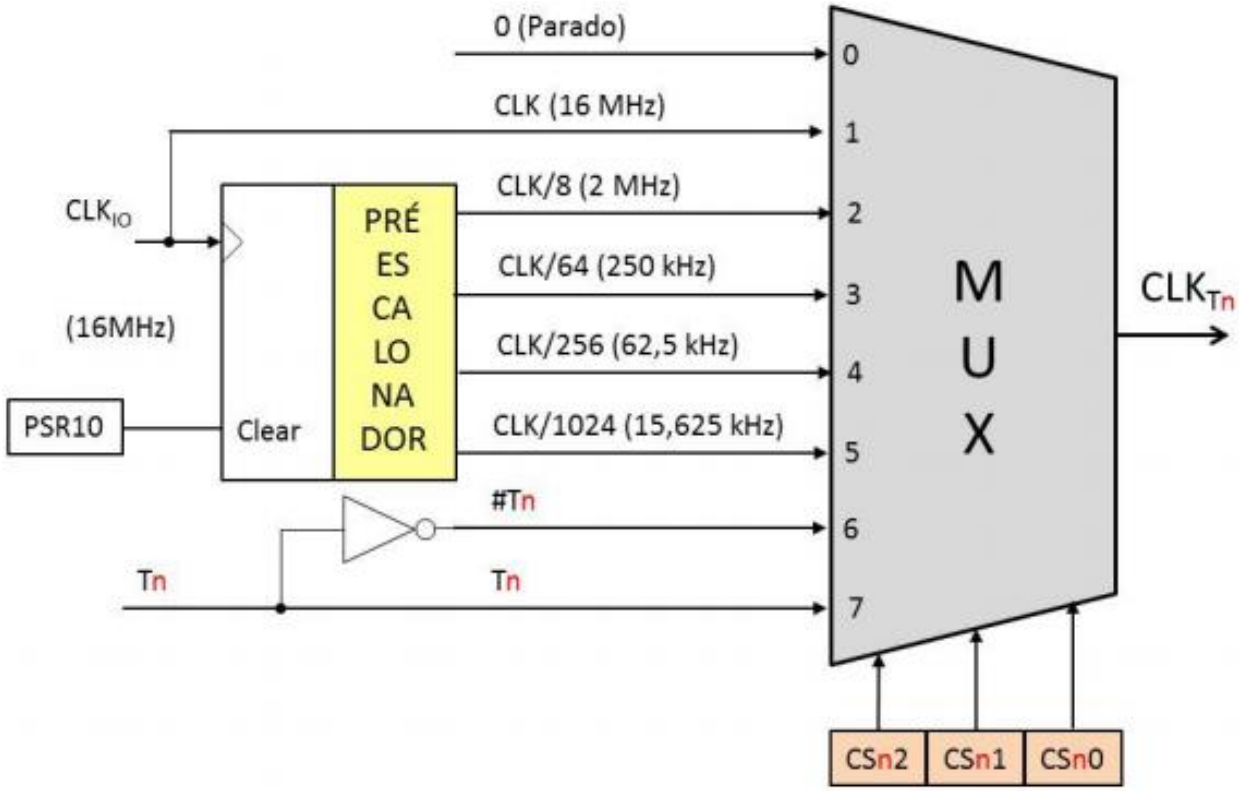


Figura 48 – Seleção da saída do pré-escalonador

Com isso, as frequências possíveis são subdivisões de 16 MHz. Essas são:

- 16 MHz
- 16 MHz / 8 = 2 MHz
- 16 MHz / 64 = 250 kHz
- 16 MHz / 256 = 62,5 kHz
- 16 MHz / 1024 = 15,625 kHz

Os temporizadores dispõem de comparadores. Esses comparadores permitem ao programador definir um valor de referência, que pode ser utilizado como sinal de que determinado instante de tempo foi atingido e utilizá-lo para executar funções. Ondas moduladas pela largura de pulso (PWM) possuem instantes de tempo bem definidos, onde normalmente utiliza-se desses sinais, em forma de interrupções, para se controlar uma saída em forma de PWM.

Diferentemente do projeto anterior, esse projeto não utiliza ondas PWM e circuitos RC. Mas o temporizador ainda é necessário para se executar uma rotina com um intervalo fixo.

Supondo que se objetiva uma interrupção gerada a cada segundo, o programador pode optar pela frequência de 62,5 kHz (16 MHz / 256) e definir o valor de um dos comparadores em 62.500. Por definição, 62,5 kHz geram 62.500 contagens em um segundo, dessa forma, a cada segundo uma interrupção aconteceria e poderia ser utilizada para execução de rotinas.

4.1.2 Interrupções

Interrupção em um processador é definida como um sinal de um dispositivo que tipicamente resulta em uma troca de contextos, isto é, o processador interrompe sua execução atual, executa o procedimento requerido, e retorna à sua execução anterior.

Como explicado no item anterior, temporizadores podem gerar interrupções, mas também, interrupções podem ser disparadas por sinais em pinos específicos do Arduino e iniciar um procedimento programado.

Para se utilizar de tais interrupções no Arduino UNO, o programador dispõe somente de duas entradas de interrupção, INT0 e INT1, que correspondem às portas digitais 2 e 3 (vide Anexo 1). Deve-se então em código, decidir qual será a rotina que será executada com a interrupção e qual o critério para a ocorrência de tal interrupção. As possibilidades para tal critério são:

LOW →	Dispara a interrupção quando o pino está em LOW.
HIGH →	Dispara a interrupção quando o pino está em HIGH.
CHANGE →	Dispara a interrupção quando o pino é modificado
RISING →	Dispara a interrupção quando o pino muda de LOW para HIGH.
FALLING →	Dispara a interrupção quando o pino muda de HIGH para LOW.

Para este trabalho, a utilização de interrupções geradas por sinal externo foi utilizada no ensaio de rotação (Seção 7.1) e devido à dinâmica do circuito montado para a aplicação, a condição RISING foi a escolhida para ativação da interrupção.

4.1.3 Software Serial: Bluetooth

Como explicado na sessão 3.2.1.2, os módulos Bluetooth utilizam-se da comunicação Serial em seus pinos TX e RX. Para conseguirmos trabalhar com sua comunicação, utilizou-se da biblioteca *SoftwareSerial.h*.

Essa biblioteca emula uma comunicação serial em pinos digitais sem utilizar o hardware serial da CPU (pinos 0 e 1). Como durante o desenvolvimento do projeto utilizamos a saída Serial do Arduino para comunicação com o computador, optamos por não utilizar as portas 0 e 1 do Arduino afim de evitar conflitos.

Após a definição em código do Bluetooth e suas portas virtuais de TX e RX. Utiliza-se normalmente o módulo Bluetooth como se utiliza uma porta Serial, com todas as suas funcionalidades. A única diferença é que em vez de se escrever na tela de comando de um computador, as mensagens são enviadas diretamente para o outro módulo Bluetooth previamente pareado.

Dessa maneira, os módulos Bluetooth foram utilizados para se fazer a comunicação entre os módulos de controle (móvel, na mão do usuário) e de atuação (base fixa próxima ao usuário).

4.2 Funcionamento e elaboração

O software do sistema de atuação tem como função ler os dados recebidos pelo Bluetooth do sistema de controle, interpretá-los, e através dos potenciômetros digitais, atuar na placa que envia sinal de rádio para o VANT.

O sistema de atuação tem a capacidade de escolher qualquer passo (entre 0 e 31) para qualquer um dos quatro potenciômetros utilizados. O software deve ter em memória em qual passo cada um dos potenciômetros se encontra, receber pelo Bluetooth qual é o novo passo desejado, e gerar um pulso incremental ou decremental dependendo se deseja subir ou diminuir o passo de determinado potenciômetro.

4.2.1 Apresentação

O sistema de atuação inicia seu funcionamento com rotinas de calibração e habilitação de interrupções. Seu laço principal tem seu foco na leitura de novos dados enviados pelo Bluetooth do sistema de controle. O sistema dispõe também de duas rotinas com períodos definidos, gerados por interrupções dos comparadores dos temporizadores.

O contador do temporizador, é redefinido quando ocorre igualdade no seu valor com o valor do comparador A. Sendo assim, utiliza-se um comparador B intermediário para gerar uma interrupção sem interferir na contagem.

A primeira interrupção (Interrupção B), atua em todos os potenciômetros que devem ser alterados, definindo LOW no pino em que se deseja incrementar ou decrementar. A segunda interrupção (Interrupção A), verifica quais pinos foram alterados na interrupção anterior e devolve seu estado para HIGH.

A Figura 49 mostra o fluxograma que descreve o funcionamento do software de atuação.

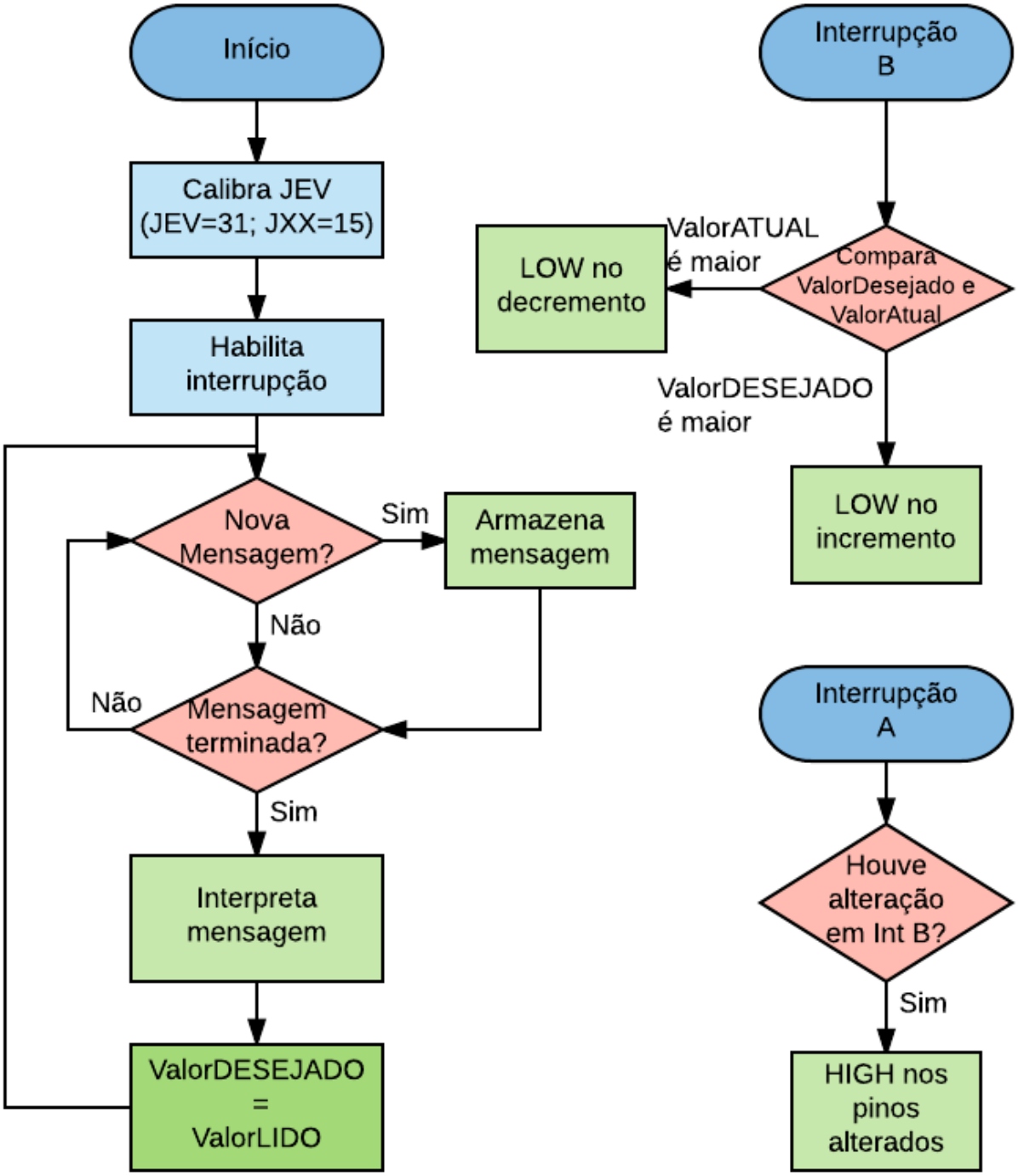


Figura 49 – Fluxograma do software de atuação

4.2.2 Detalhes

Até então, explicou-se o funcionamento do software de modo funcional, porém superficial. Nesta sessão busca-se aprofundar em detalhes de algumas partes onde encontramos dificuldades durante sua elaboração.

4.2.2.1 Calibração

A calibração consiste em posicionar os potenciômetros digitais em uma posição conhecida e armazenar em memória a posição atual. Para tal, faz-se 32 iterações de um ponto qualquer de partida para até 0. Ao fim das 32 iterações garante-se que o potenciômetro está na menor posição possível. Atualiza-se então a variável em memória que armazena as posições atuais de cada potenciômetro para 0 também.

A partir dessa igualdade, qualquer pulso aplicado nas entradas incremental ou decremental de qualquer potenciômetro deve ser também ser registrada em memória, de modo que o sistema sempre tenha o controle de qual posição atual cada potenciômetro se encontra.

Em seguida o código comanda 31 pulsos na entrada incremental do JEV para que atinja o seu nível lógico menor (Como já explicado anteriormente: os potenciômetros verticais são invertidos. Ao aplicar 0V tem-se sua intensidade aumentada e 3,3V, sua menor). Essa rotina simula a calibração e sincronia do rádio com o VANT.

Esse procedimento é realizado somente para o JEV. Durante testes realizados em bancada com o VANT e seu rádio, observou-se que ao ligar da placa é necessário que todos os potenciômetros retráteis estejam em seu valor médio, ou seja, JEH, JDV e JDH.

Para que os potenciômetros estejam posicionados em seu valor médio no ligar do sistema, utilizou-se da função AUTOSTORE do potenciômetro digital. Dessa forma sempre que ligado, o potenciômetro inicia da posição armazenada em sua memória ROM, armazenamos, então, seu valor médio.

4.2.2.2 Interrupções

O sistema dispõe de duas interrupções que ocorrem com períodos bem definidos por comparadores de um temporizador. A primeira interrupção é responsável por aplicar sinal LOW nos pinos incrementais ou decrementais dos potenciômetros que necessitam ser alterados. A segunda interrupção devolve para HIGH os pinos que foram alterados para LOW na interrupção anterior.

Pela leitura do *datasheet* do potenciômetro digital (x9511), descobriu-se que se deve manter o sinal do pulso em LOW por um tempo mínimo de 40ms para que ocorra a alteração de um passo. Pulsos com largura menor que 40ms são desprezados pelo circuito do potenciômetro. Com isso, deve-se manter sempre um intervalo mínimo de 40ms entre as interrupções B e A.

Para se medir o tempo com precisão, utilizou-se o pré-escalonador em CLK/64, ou seja: 16 MHz / 64 = 250 kHz. Sabe-se, por definição, que nessa frequência o contador incrementa 250.000 a cada segundo. Ou seja, em 40 ms conta-se 10000 *ticks* no contador do temporizador. Como o valor 0 está incluso no intervalo de contagem, definiu-se os comparadores B e A, que geram interrupções, em 9.999 (10.000 - 1) e 19.999 (20.000 - 1). Veja:

$$250.000\ Hz\ \times\ 0{,}04\ s = 10.000\ ticks$$

A fim de melhor se otimizar o tempo, o sistema dispõe de variáveis voláteis que servem de *flags* para sinalizar qual potenciômetro, e se incremental ou decremental, sofreu alteração. Dessa forma, a segunda interrupção (Interrupção A) encarrega de redefinir para HIGH somente os pinos que foram de fato definidos em LOW na interrupção anterior.

4.2.2.3 Leitura e interpretação das mensagens recebidas

Como o *loop* do Arduino do sistema de atuação ocorre em uma frequência muito mais elevada que o envio de mensagens pelo Bluetooth do controle, uma mensagem grande, sem nenhum tratamento no envio, é precocemente entendida como terminada. Como envio não acompanha o ritmo da leitura, é entendido, pela atuação, ao buscar nova leitura e não encontrar novos dados, que a mensagem foi finalizada. Por isso, as mensagens estavam sendo quebradas em vários pedaços.

Para que se tornasse possível o envio contínuo e confiável de mensagens de um Bluetooth para o outro, fez-se necessário protocolar a mensagem enviada, com um sinal que define o início e um sinal que define o fim. Além disso, utilizou-se um sinal que define a divisão entre as partes das mensagens.

O formato da mensagem protocolada é:

#**.**.**.**;

Onde ** são números que descrevem o passo desejado enviado pelo controle. O primeiro campo da mensagem refere-se ao JEV, o segundo ao JDV, o terceiro ao JDH e o último ao JEH. A mensagem inicia-se com **#** e finaliza com **;** .

Um exemplo pode ser descrito por:

#29.15.15.20;

Isso implica que o controle está enviando força 29 para o JEV (apenas 2 passos de intensidade em relação ao seu mínimo, pois 31 é o VANT parado), JDV e JDH estão em seu médio, e o JEH está levemente posicionado à direita do seu ponto médio, ou seja, recebendo sinal para rotacionar em seu próprio eixo em sentido horário.

Dessa maneira, mesmo que a mensagem seja interrompida durante seu envio, o Bluetooth do sistema de atuação entende que o envio ainda não foi finalizado pois ainda não recebeu o caractere (**;**) de finalização.

Uma vez que a mensagem seja recebida e armazenada em um vetor em memória, sabe-se a correspondência de cada campo com cada potenciômetro e atualiza-se, então, novos valores desejados para cada um dos potenciômetros de atuação.

Capítulo 5

Hardware de Controle

“ Qualquer produto que precise de um manual para funcionar está mal feito. ”

– Elon Musk

5.1 Arduino Pro Mini

O Arduino Pro Mini tem a maior parte dos confortos e facilidades que fazem do Arduino Uno uma alternativa bastante amigável para os iniciantes no universo das placas de desenvolvimento com microcontroladores, mas remove uma parte importante do que gera o custo e o consumo de energia1 do Uno: a porta USB própria. Custa cerca de 20 reais no Brasil e é recomendado para instalações *standalone* com restrições de espaço com a mesma forma intuitiva de se programar.

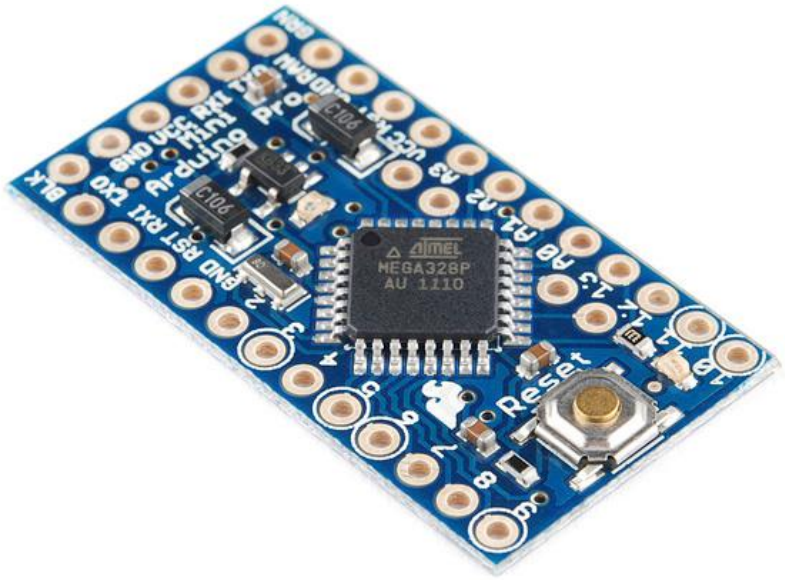


Figura 50 - Arduino Pro Mini *[15]*

O Pro Mini, como visto na Figura 50, tem os mesmos pinos analógicos, digitais e PWM do Uno, incluindo até o LED interno do pino 13. Do ponto de vista da programação usual, ela é baseada na IDE comum do Arduino. Ele difere muito pouco do Arduino Uno: tem os mesmos 16MHz de *clock*, o mesmo número de pinos analógicos, digitais e PWM, além de ter também o mesmo LED interno associado ao pino 13, suporte nativo a I2C, SPI e TTL, *bootloader* nativo com *auto-reset* no momento da programação, etc. O microcontrolador, o ATMega328p, tem as mesmas configurações das memórias *Flash*, SRAM e EEPROM da versão ATMega328: 16KB, 2KB e 512B, respectivamente.

A descrição mais detalhada de cada pino desse Arduino pode ser vista no Anexo 2.

A ausência da porta USB fez necessário o uso de um módulo extra, o FTDI *chip*. FTDI é o nome da empresa escocesa que criou o dispositivo, mas devido sua grande popularidade e difusão é comum chamá-lo de FTDI somente.

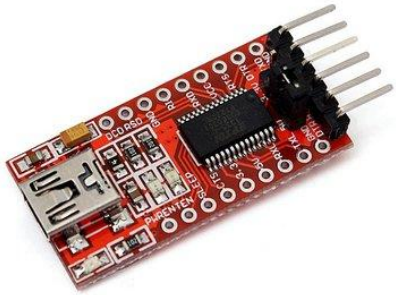


Figura 51 – FTDI chip *[16]*

Esta placa FTDI, vista na Figura 51, é baseada no chip FT232RL que funciona como um conversor USB para serial TTL ou RS232 e vice e versa, permitindo a interface de dispositivos que utilizam esses protocolos.

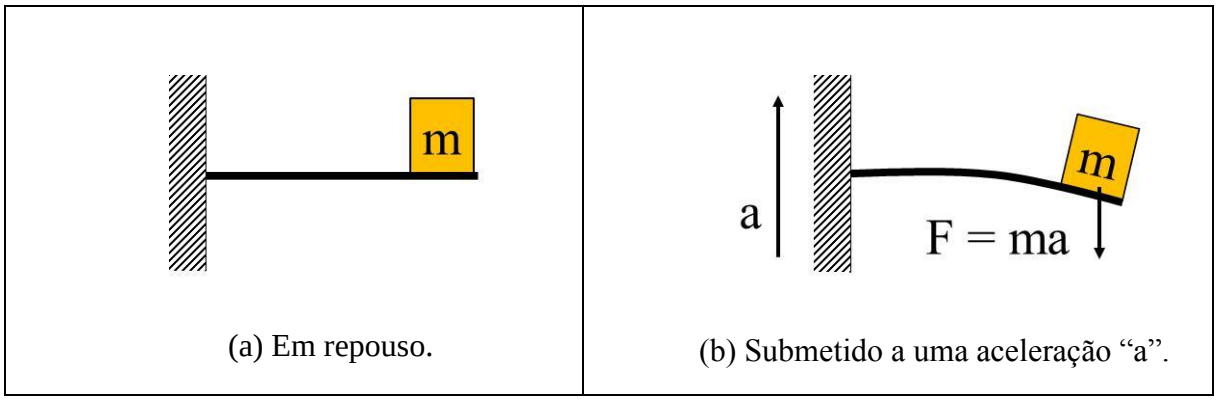
Sua "pinagem" foi especialmente distribuída da mesma forma que um cabo FTDI, auxiliando por exemplo na gravação de Arduinos que operam em 3,3 e 5V.

5.2 Unidade de Medida Inercial [17]

A unidade de medida inercial é um dispositivo que combina acelerômetros e giroscópios para medir e enviar dados sobre as forças específicas, velocidades e acelerações angulares de um corpo. Nessa seção será detalhado o funcionamento de cada uma das partes.

5.2.1 Física de um acelerômetro e de um giroscópio

O acelerômetro serve para medir a aceleração a que o dispositivo está submetido. A forma mais simples de se entender o funcionamento de tal dispositivo é analisar uma barra flexível com uma massa presa em sua extremidade, como mostrado na Figura 52. Quando em repouso, a haste deve estar estendida, porém, quando submetida a uma aceleração, essa haste deve fletir em função da inércia da massa colocada em sua extremidade. O uso de 3 sensores desse tipo, um em cada direção nos permite criar um acelerômetro de 3 eixos. Como estamos na superfície terrestre, o dispositivo estará sempre sujeito à aceleração da gravidade (g = 9,81 m/s^2), ou seja, um dos eixos deverá indicar 1 g (a não ser que se posicione o acelerômetro de forma inclinada).

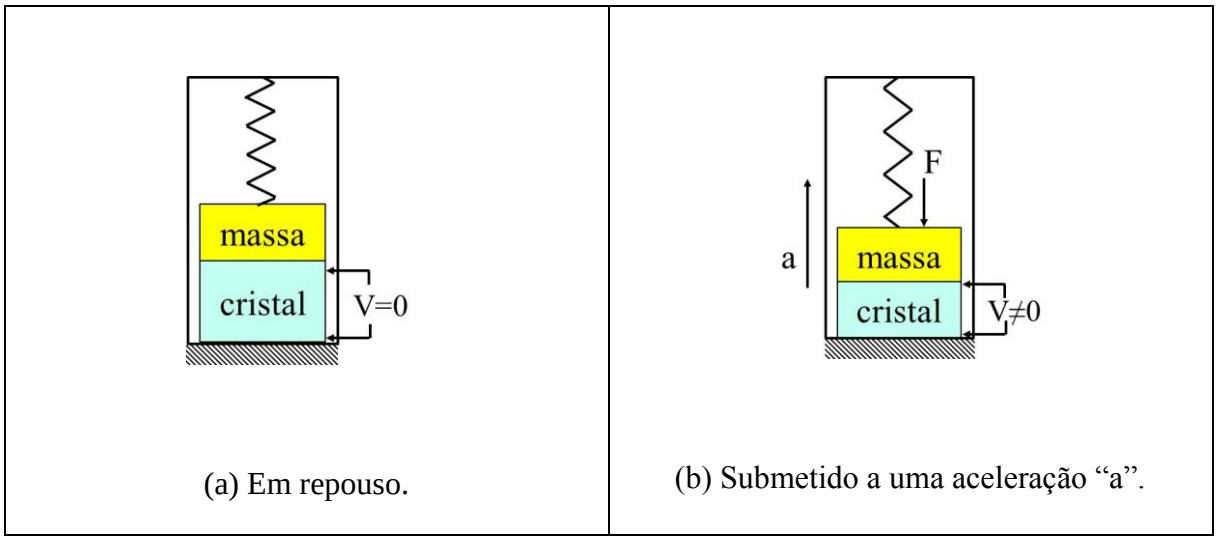


(a) Em repouso.

(b) Submetido a uma aceleração "a".

Figura 52 - Exemplo ilustrativo de um acelerômetro usando uma haste flexível com uma extremidade fixa e uma massa "m" na outra extremidade.

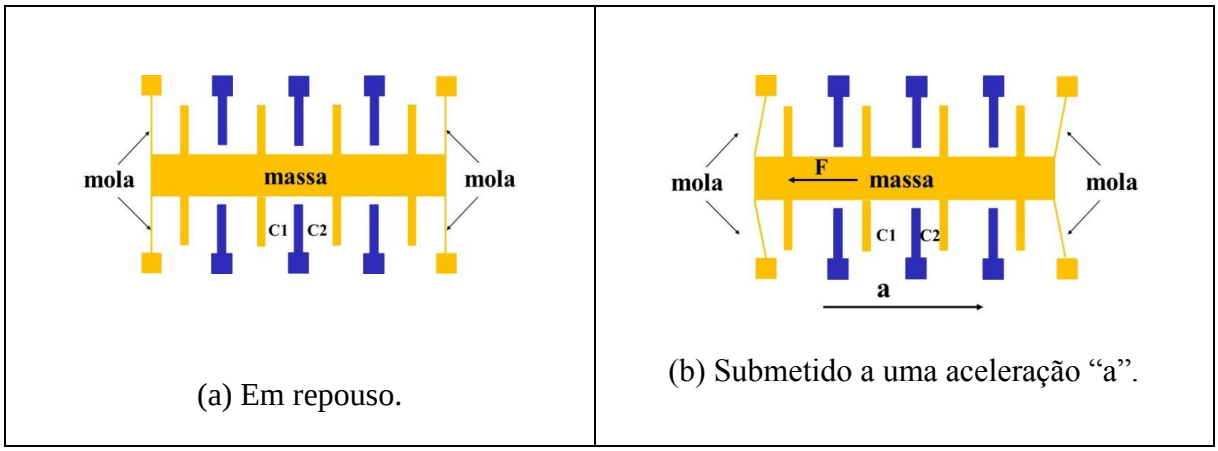
Outra forma de se medir a aceleração é com o emprego de cristal piezoeléctrico. Tal cristal apresenta deformação quando submetido a uma tensão e, por outro lado, apresenta uma tensão quando sofre uma deformação. A Figura 53 apresenta um acelerômetro que explora esta propriedade. Quando em repouso, uma mola mantém o cristal na posição e com tensão de saída igual a zero. Já quando submetido a uma aceleração, devido a inércia da massa, surge uma força que vai deformar o cristal e gerar uma tensão diferente de zero. Quanto maior a força, maior a tensão.



(a) Em repouso. (b) Submetido a uma aceleração “a”.

Figura 53 - Exemplo ilustrativo de um acelerômetro usando um cristal piezoeléctrico e uma massa que o deforma quando submetida a uma aceleração.

O MPU-6050 usa um princípio um pouco diferente, construído com a tecnologia denominada Sistema Micro Eletromecânico, MEMS, (*Micro-Electro-Mechanical System*). Esta tecnologia permite construir sistemas mecânicos miniaturizados. No caso do acelerômetro, o tamanho é da ordem de 0,1 mm. A Figura 54 apresenta uma representação simplificada de um acelerômetro. Note a massa e a presença de duas “molas”. As placas da porção fixa e da porção móvel formam diversos pares de capacitores. Esses capacitores são denominados de C1 e C2 (a figura apresenta apenas um par de C1 e C2). Quando em repouso essas capacitâncias devem ser idênticas. Porém, quando a peça é submetida a uma aceleração, da esquerda para a direita no caso do exemplo da figura, a inércia faz a parte móvel se movimentar (relativamente) na direção oposta e com isso a capacitância de C2 fica maior que a de C1. Pela relação entre os esses dois capacitores, é possível estimar a aceleração a que a peça está submetida.



(a) Em repouso. (b) Submetido a uma aceleração “a”.

Figura 54 - Exemplo ilustrativo de um acelerômetro MEMS, usando uma peça fixa e uma peça móvel cuja posição relativa, quando submetida a uma aceleração, provoca variação dos capacitores C1 e C2.

A explicação do giroscópio é um pouco mais complexa, pois envolve o conceito da Aceleração de Coriolis. Iniciamos imaginando uma massa que se desloca numa determinada direção, como mostrado na Figura 55. Se a essa massa aplicarmos um giro ou uma rotação, surge uma força perpendicular à direção do movimento, resultado da Aceleração de Coriolis. Se girarmos no sentido oposto, a Força de Coriolis é invertida. Quanto maior for a velocidade angular deste giro, maior será o módulo da força.

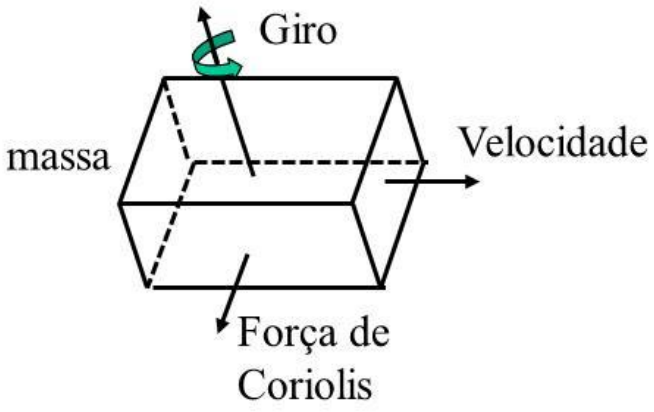


Figura 55 - Ilustração da Força de Coriolis agindo sobre uma massa que foi submetida a um giro enquanto estava em movimento.

O giroscópio que vamos descrever é na verdade um acelerômetro construído para medir a Aceleração de Coriolis. Na Figura 56 vemos o acelerômetro MEMS, descrito na Figura 54, mas agora submetido a um movimento oscilatório na horizontal. Esta oscilação garante que a massa esteja sempre em movimento. Quando for submetido a um giro, o acelerômetro irá indicar o valor da Aceleração de Coriolis, que permite estimar a velocidade angular aplicada à peça e o ângulo do dispositivo por meio de integrações. Naturalmente esse processo propaga erros ao longo do tempo, então constantemente acelerômetro e giroscópio se complementam na medição.

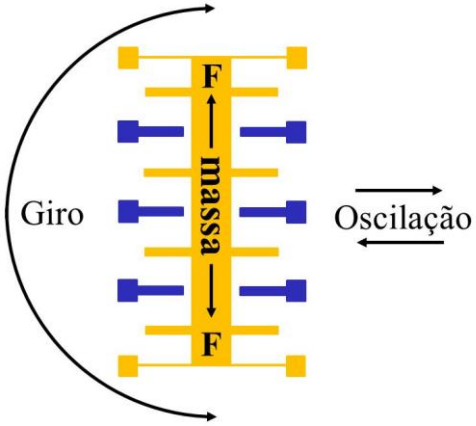


Figura 56 - Ilustração de um giroscópio que estima a velocidade angular a partir da Aceleração de Coriolis medida por um acelerômetro MEMS que é mantido em permanente oscilação.

De uma forma um pouco diferente, a Aceleração de Coriolis pode ser experimentada com um HD USB. Após ser conectado, o disco que está dentro do HD começa a girar. Se pegarmos esse HD com a mão e o movimentarmos com um giro, vamos sentir um comportamento um pouco inesperado, como se uma força estivesse atuando em uma de suas extremidades.

5.2.2 MPU 6050

O MPU 6050 é um dispositivo integrado que combina um acelerômetro de 3 eixos e um giroscópio também de 3 eixos. Sua principal finalidade é a detecção e acompanhamento de movimentos. Citam-se o emprego na detecção da posição de celulares e *tablets*, nos vídeos games, na detecção de movimentos humanos e até na eliminação da trepidação em câmeras fotográficas e filmadoras.

O circuito é um chip fabricado pela Invense e disponibilizado num encapsulamento para montagem em superfície. Devido às dificuldades para se trabalhar com tal encapsulamento, é comum encontrarmos pequenas placas que já o trazem soldado, o que facilita muito seu emprego. O acelerômetro trabalha nas escalas +/- 2 g, +/- 4 g, +/- 6 g e +/- 16 g. O giroscópio trabalha nas escalas de +/- 250 gr/s, +/- 500 gr/s, +/- 1000 gr/s e +/- 2000 gr/s (gr/s = graus/segundo). Os valores entregues são inteiros de 16 bits com sinal. Isto significa que varrem a faixa de -32.768 até 32.767 (65.536 passos). Por exemplo, supondo a escala de +/- 2g, a resolução é dada por 4 g / 65.536 (ou então, 2 g / 32.767).

A placa mais comum no Brasil é a GY-521, que será objeto deste estudo. A Figura 57 traz duas sugestões para sua conexão com o Arduino. Essa placa possui um regulador interno de 3,3 V, assim, pode ser alimentada com 5 V. Seus pinos são:

- VCC = 5V;
- GND = Terra;
- SCL e SDA = porta TWI (I2C) do Arduino;
- AD0 = endereço alternativo;
- INT = pino de interrupção;
- XDA e XCL = pinos não são usados neste estudo.

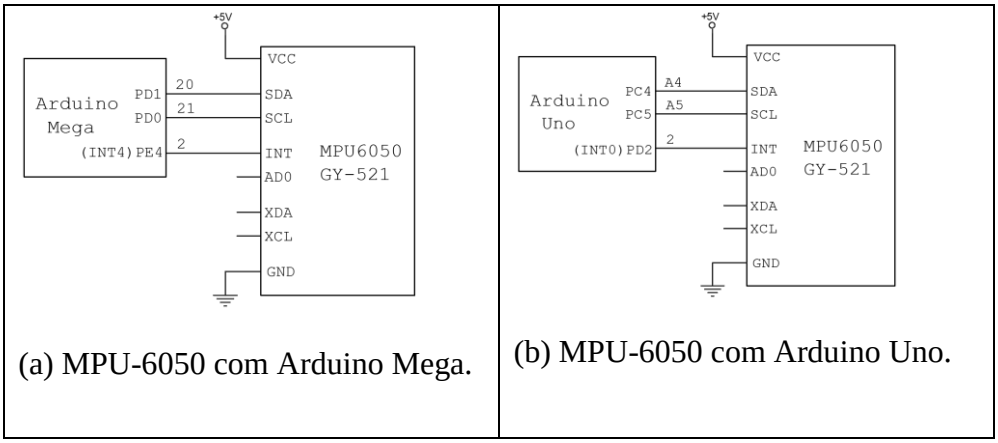
O dispositivo GY-521 será aqui chamado de MPU-6050 ou apenas MPU. O pino AD0 permite que se usem dois desses dispositivos num único barramento I^2C. Quando solto, este pino AD0 vai para zero graças a um resistor de *pull-down* interno. Com um resistor de pull-up externo é possível colocá-lo em nível alto e assim configurar a placa para operar no endereço alternativo. A comunicação (SDA e SCL) é feita com o barramento I^2C.

A placa em questão, cuja foto é apresentada na Tabela 7. Existem diversas recomendações para que se use 5 V para beneficiar a precisão. A Tabela 7 lista a "pinagem" desta placa. As linhas SCL e SDA já têm resistores de pull-up (4,7 kΩ) e a linha AD0 tem um resistor de *pull-down* (4,7 kΩ). Se a linha de endereço AD0 for deixada solta, o endereço da placa é 0x68. Porém, se ela for conectada a VCC, o endereço passa a ser 0x69. Isto permite que se use dois chips MPU-6050 em um mesmo sistema. As linhas XCL e XDA formam o barramento I^2C auxiliar para conectar um outro dispositivo, como um magnetômetro. Esse barramento auxiliar não será abordado neste estudo.

Tabela 7 - Pinagem da placa GY-521 (MPU-6050)

Pino	Descrição	Observações	Foto
VCC	3,3 até 5,0 Volts	-	
GND	Terra	-	
SCL	Bus I²C (TWI)	*pull-up* de 4,7 kΩ	
SDA	Bus I²C (TWI)	*pull-up* de 4,7 kΩ	
XDA	Bus I²C (TWI) aux.	-	
XCL	Bus I²C (TWI) aux.	-	
AD0	Endereço	*pull-down* de 4,7 kΩ	
INT	Interrupção	-	

A conexão com uma placa Arduino é muito simples. É recomendado que se remova a alimentação antes de se fazer as conexões. A Figura 57 traz duas sugestões: uma para o Arduino Mega e outra para o Uno. O VCC e o Terra devem ser ligados aos respectivos pinos do Arduino. O barramento I²C (TWI) deve ser ligado aos pinos SDA e SCL. O pino INT, que é o de interrupção (se for usada) deve ser ligado a alguma interrupção do Arduino. Para efeito de ilustração, no caso do Uno foi escolhida a INT0 (PD2) e no Mega a INT4 (PE4), porque a INT0 do Mega é compartilhada com a linha SCL. As linhas do barramento I²C auxiliar devem ser deixadas soltas. Para terminar, o pino de endereço AD0 deve estar solto para se usar o endereço 0x68 ou conectado a VCC, para se usar o endereço 0x69.



(a) MPU-6050 com Arduino Mega.

(b) MPU-6050 com Arduino Uno.

Figura 57 - Conexão do MPU-6050 com o Arduino Mega ou Uno

Antes de se usar o MPU-6050 é necessário fazer o auto teste para ver se o dispositivo está dentro das especificações e depois fazer a calibração.

5.3 Projeto Hardware Controle

O objetivo dessa etapa do projeto foi construir uma estrutura independente que fizesse todas as funções de um controle remoto normal e que acrescentasse a aplicação do acelerômetro. O Arduino Pro Mini, nesse contexto, tem a função de gerenciar o *status* de todos os botões eventualmente presentes e tratar o posicionamento do MPU, fornecendo, para a comunicação Bluetooth, uma palavra já com quais passos cada potenciômetro digital do hardware de atuação deve estar.

Decidiu-se deixar o controle total sob a responsabilidade do MPU, logo *Pitch*, *Roll* e *Yaw* são controlados pelo movimento do usuário como se ele segurasse um volante. Fez-se necessária a presença de 5 botões adicionais:

- Botão UP – JEV;
- Botão DOWN – JEV;
- Botão Sentido horário – JEH;
- Botão Sentido Anti-Horário – JEH;
- Botão Emergência.

Os dois primeiros são para controle de altitude do *drone*. Eles funcionam de forma incremental.

Os que competem ao JEH servem somente para ajuste fino da orientação do ângulo de giro em torno do próprio eixo do *drone*. Eles funcionam diferente dos anteriores: o VANT irá rotacionar no sentido desejado enquanto o botão é pressionado. Ao final do giro, aquela orientação será considerada como o ângulo zero em torno do eixo Z (esse tratamento é feito via software, pois o botão é um *push button* comum como os demais). Essa aplicação foi pensada tendo em vista duas necessidades verificadas em testes.

Primeiramente, foi pensado que esse controle de *Yaw* fosse usado majoritariamente para manter o *drone* direcionado na mesma orientação do seu operador, porém há certa imprecisão intrínseca do projeto, por estar ainda atuando em malha aberta. Assim sendo, pode ser que a orientação do VANT não seja exatamente a do usuário, este pode, então, ajustá-la pelos botões.

A outra necessidade abrange outro uso do *Yaw*, aquele que não precisa manter a orientação entre *drone* e operador. Se for preciso fazer uma manobra de rotação no eixo Z e o usuário não desejar reorientar seu corpo junto, ele iria girar a estrutura de controle e teria que operar seu volante dessa forma, a orientação do controle oblíquo à pessoa. Isso demonstrou ser muito inconveniente e não intuitivo. Logo se isso for necessário, recomenda-se usar os botões de ajuste e continuar pilotando normalmente com o volante paralelo ao corpo.

Por fim, o botão de emergência surgiu para garantir a integridade física do *drone*. Apesar de se mostrar muito robusto às colisões, foi verificado que se uma das hélices é travada enquanto gira, a corrente no motor DC que a controla aumenta, levando à sua queima. Como o controle de tensão enviado aos motores é feito pelos botões incrementais UP e DOWN JEV, naturalmente é necessário um tempo até zerá-los completamente, no máximo 32 pulsos, por construção dos potenciômetros digitais. O botão de emergência então vai forçar com que os 3 potenciômetros retráteis voltem para sua posição intermediária e o JEV envie potência nula. Uma boa prática que foi adotada para garantir a segurança do usuário e do próprio *drone* é que, sempre que pousar o *drone*, pressionar o botão de emergência. Uma vez apertado, todo o sistema de controle deve ser reiniciado, ligando e desligando o *switch*.

5.3.1 Estrutura Mecânica

A estrutura externa foi pensada para suprir as especificações requisitadas do projeto. O material escolhido foi um acrílico de 3mm de espessura por ser suficientemente resistente a impacto e leve. O fato dele ser totalmente transparente foi um diferencial para que fosse possível mostrar seu interior com os circuitos eletrônicos, facilitando a montagem interna e os eventuais *debugs* visuais.

As dimensões externas foram projetadas de tal forma que o sistema tivesse, pelo menos, um mínimo formato ergonômico de um pequeno volante. Preocupou-se em fazer uma tampa na parte de trás, para manuseio do interior, além de um suporte bem centralizado para receber o MPU.

Por fim, na estrutura de acrílico, foram calculadas as posições dos rasgos para os botões, os espaçadores de fixação, o *switch* e a entrada USB. A tampa é fechada com dois velcros.

O modelo CAD, com as vistas e dimensões, pode ser encontrado nos Anexos 5.1 a 5.3.

A modelagem 3D da estrutura feita no software *SolidWorks* pode ser conferida na Figura 58.

Figura 58 – Modelamento 3D da estrutura de acrílico projetada.

5.3.2 Esquemático Hardware Controle

O esquemático mostra-se mais simples que o anterior pois foi preciso somente alocar os botões, o Arduino Pro Mini, o Bluetooth e o suporte para entrada do MPU. Para efeito de facilitar a montagem, os botões foram ligados como INPUT_PULLUP, em que o Arduino, internamente mantem o nível lógico da porta em alto e quando o botão, que é ligado ao GND, é pressionado, o sinal assume nível baixo.

O esquemático do Hardware de Atuação completo pode ser visto na Figura 59.

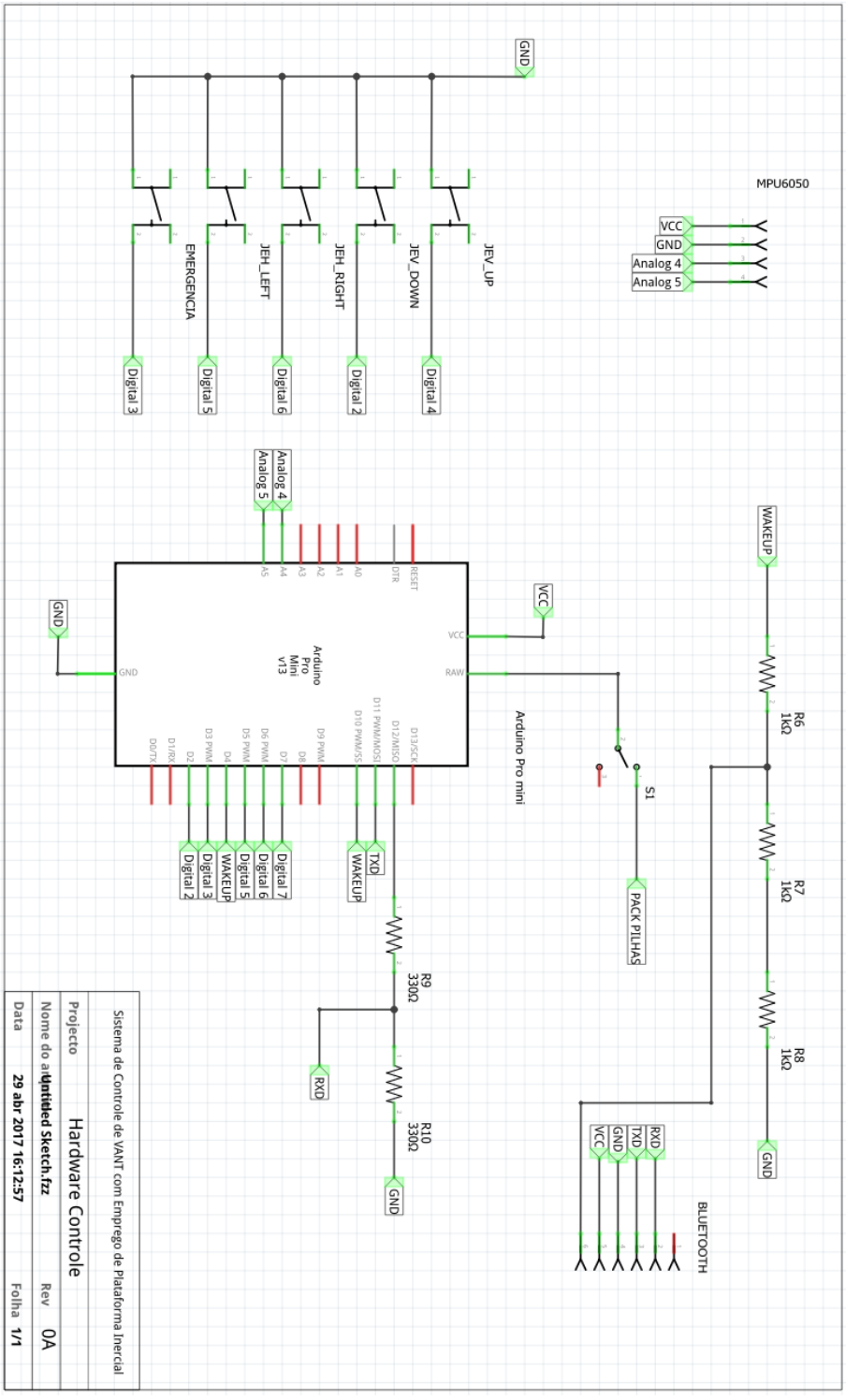


Figura 59 - Esquemático Hardware Controle

5.3.3 Placa de circuito impresso – Controle

Novamente o esquemático foi fortemente testado como protótipo em uma *protoboard* e quando os resultados foram satisfatórios, partiu-se para a confecção da PCB.

Aqui houve uma preocupação adicional que foi a restrição de espaço determinada pelas dimensões gerais da estrutura de acrílico, além do posicionamento dos botões, principalmente o UP-JEV, o DOWN-JEV e o de emergência, que vêm na fase da frente da caixa.

A preocupação com a forma de roteamento foi a mesma que o projeto de atuação. Aqui, os três botões que ficam na face frontal da estrutura de acrílico serão montados na mesma face onde as trilhas estão presentes. Os demais componentes e *headers* serão soldados na parte inferior, isso pode ser notado por apresentarem um contorno mais apagado na Figura 60.

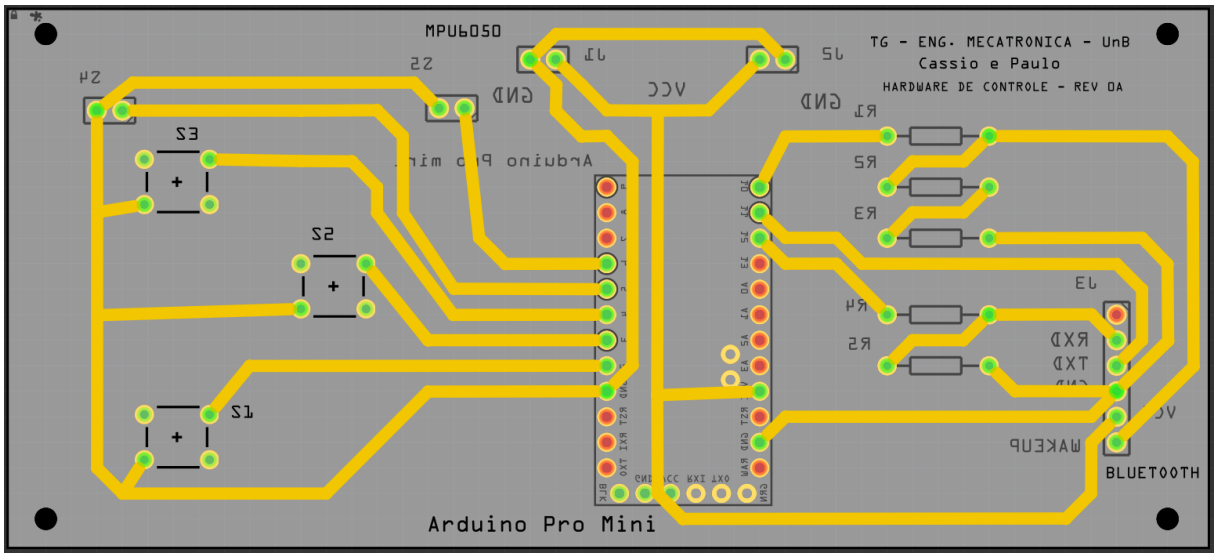


Figura 60 - Roteamento - Hardware de Controle

Os cuidados e o método de fabricação foram os mesmos explicados na Seção 3.2.2.2. E o resultado anterior à soldagem pode ser conferido na Figura 61.

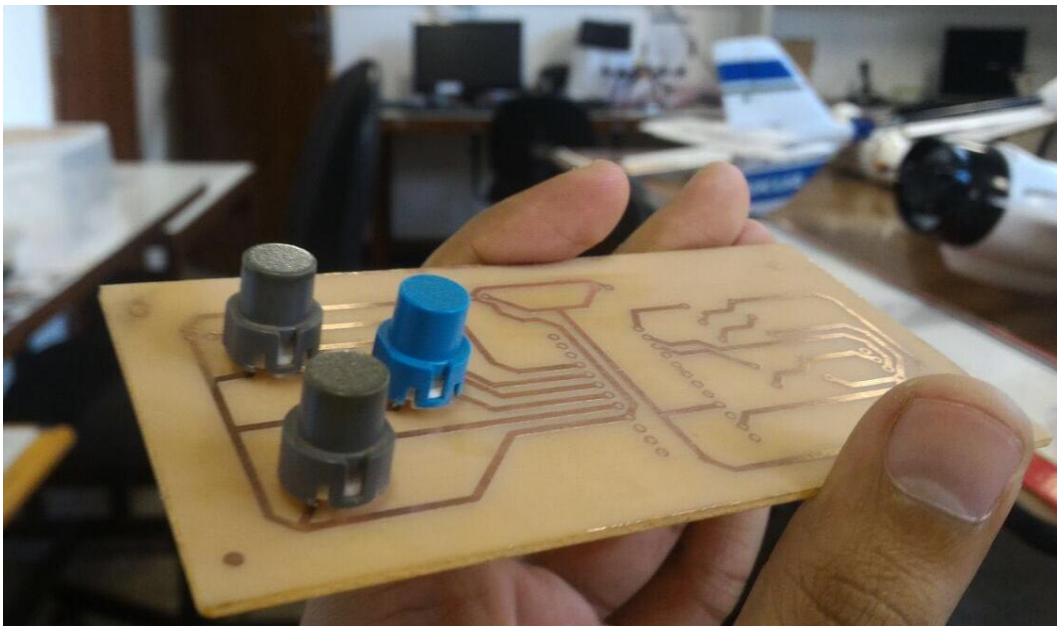


Figura 61 - Placa de controle com botões frontais após a corrosão

5.3.4 Montagem Final

A montagem final, além dos componentes já explicados, inclui o soquete de 4 pilhas e a placa FTDI que grava o Arduino Pro Mini, ambos presos com fita dupla face. Além de um interruptor que liga e desliga todo o circuito. A montagem final pode ser conferida na Figura 62 e Figura 63.

Aqui faz-se necessária uma breve explicação da ordem recomendada para ligar e desligar todo o sistema. Para ligar, liga-se o hardware de controle, depois o *drone* e, por fim, o sistema de atuação, nessa ordem. Para desligar, desliga-se o *drone*, o hardware de atuação e, então, o de controle. Isso evita que sinais erráticos sejam mandados para o VANT e que ele execute alguma operação indesejada.

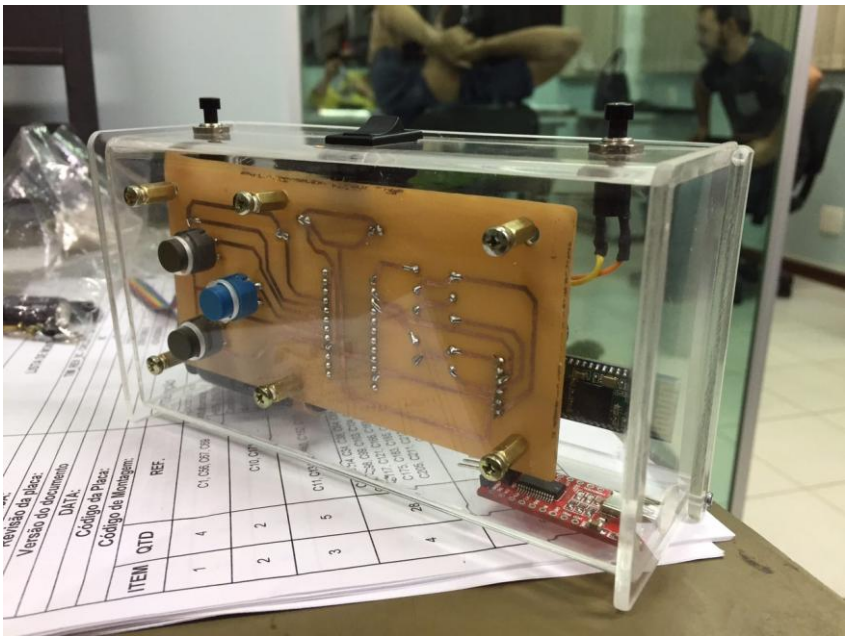


Figura 62 - Vista frontal - Montagem Final

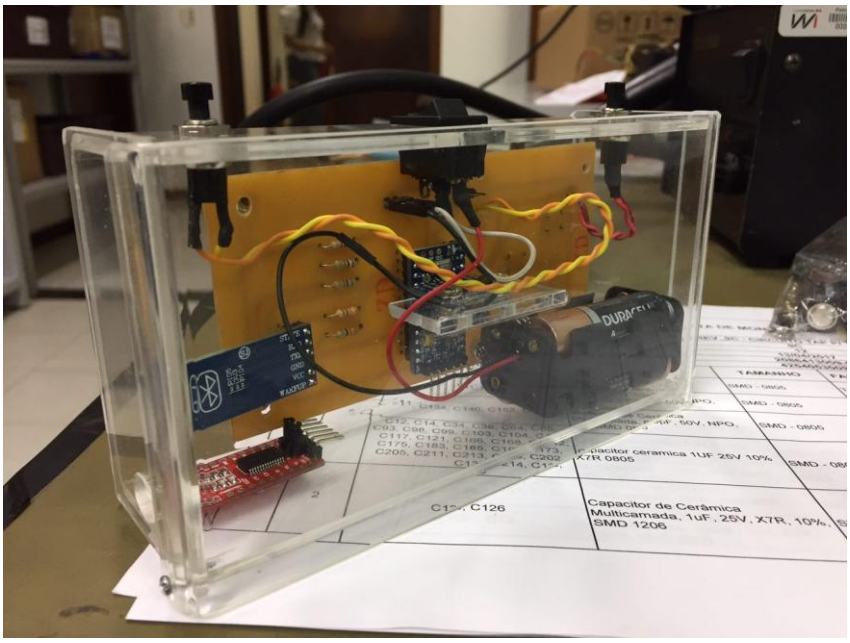


Figura 63 - Vista traseira - Montagem Final

Capítulo 6

Software de Controle

“ The hacker way é uma abordagem para a construção que envolve a melhoria contínua e a iteração. Hackers acreditam que algo sempre pode ser melhor, e que nada nunca está completo. ”

– Mark Zuckerberg

6.1 Embasamento teórico

Como já apresentado nos objetivos do projeto, espera-se que o controle perceba movimentações físicas realizadas nele, isto é, rotações em torno de si mesmo em três eixos. Espera-se também, que o controle seja capaz de traduzir essa movimentação em informações que serão enviadas para a atuação.

Para tal, como explicado na sessão anterior (Hardware), utilizamos um Arduino Pro Mini, uma IMU com acelerômetro e giroscópio. A comunicação da IMU com o Arduino é feita por um protocolo de comunicação denominado I²C, com suporte de uso com o Arduino através da biblioteca *Wire.h* inclusa na IDE oficial fornecida no *site* do Arduino [18].

Uma das vantagens da comunicação I²C é o número reduzido de fios. O mesmo necessita somente dois fios conectados entre os dispositivos que se comunicam. O primeiro denominado SCL, e o segundo SDA.

O pino SCL é o pino que define o tempo de relógio (*clock*) da comunicação. O pino SDA é o que transmite as informações (*data*). Por conta de um *clock* de comunicação comum entre os dois dispositivos, definido pelo pino SCL, a comunicação I²C possui uma boa velocidade em sua comunicação [19]. Além, o protocolo I²C dispõe de mecanismos de verificação de dados enviados e recebidos, ou seja, o protocolo também garante uma boa confiabilidade na troca de informações.

A biblioteca *Wire.h* gerenciando a comunicação I²C é capaz de organizar a comunicação de diversos dispositivos conectados ao mesmo Arduino. Cada dispositivo é identificado com um endereço de 7 bits (0 a 127 em decimal). Obedecendo as regras do protocolo, a biblioteca inicia a comunicação endereçando um dispositivo e pode então enviar ou receber dados.

O Arduino dispõe de um *buffer* interno que armazena as últimas mensagens recebidas. Como a comunicação pode ocorrer de maneira paralela à execução do código, ao se receber dados em seus pinos I²C, o dado é armazenado em *buffer*. Este dado é então utilizado somente quando requisitado em código.

6.2 Funcionamento

Diferentemente do software da atuação, o software do controle não utiliza interrupções. Espera-se que o software gerencie toda a comunicação com a IMU, envio e recebimento de dados, tratamento desses dados, além de gerenciar o envio de dados pelo Bluetooth.

6.2.1 Apresentação

O código inicia realizando procedimentos de calibração do giroscópio da IMU e definindo um temporizador para controlar o tempo de repetição do laço principal. Ao iniciar o laço principal, o Arduino solicita, recebe, e armazena dados de resposta da IMU, corrige os dados com os valores da calibração, e passa por todo um tratamento de dados até se ter uma informação confiável de ângulo armazenado em variáveis.

Em seguida o software também verifica se algum dos botões foi pressionado para se computar informações do JEV, a ser posteriormente enviada, e incrementa um contador de repetição.

O código então verifica quanto tempo se passou desde o início do laço principal, e espera esse tempo completar 4 milissegundos para se começar um novo laço. Dessa forma garante-se uma frequência de repetição do laço em 1/4 ms = 250 Hz.

Além disso, quando o contador atinge a marca de 50 contagens, o software prepara a mensagem a ser enviada por Bluetooth, obedecendo o protocolo escolhido (explicado na sessão 4.2.2.3) e a envia para o sistema de atuação.

Para melhor entendimento do funcionamento do código, veja o fluxograma da Figura 64.

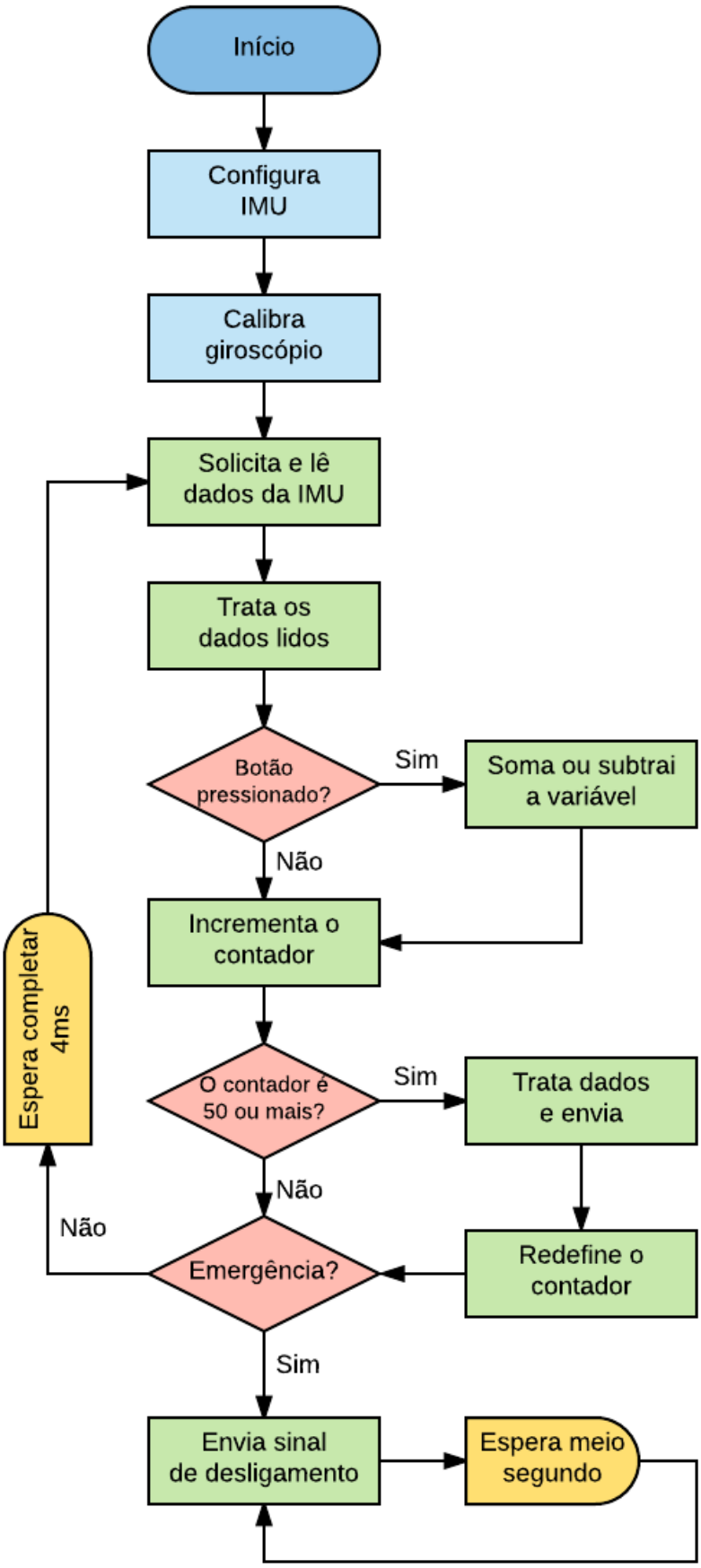


Figura 64 - Fluxograma do funcionamento do software de controle

6.2.2 Detalhes

O fluxograma ilustra bem o funcionamento geral do código, o escopo completo do software do sistema de controle. Porém, ele não aprofunda nos processos e não deixa claro a complexidade das partes. Essa sessão visa explicar em detalhes o funcionamento de algumas partes vitais do funcionamento do software.

6.2.2.1 Configuração da IMU

Para se comunicar com a IMU e definir suas configurações, utilizando da biblioteca *Wire.h*, inicia-se a comunicação com uma função "*beginTransmission"*, no endereço I²C da IMU (0x68; definição de fabricação). Em seguida com funções "*write"*, a IMU espera na ordem, um registrador, e qual valor deve ser escrito nesse registrador. Após, uma função "*endTransmission"* deve ser utilizada para encerrar a comunicação.

Dessa maneira, ativa-se a IMU, iniciando seu funcionamento e define o modo de operação do acelerômetro e do giroscópio presentes na IMU.

6.2.2.2 Calibração

Sem nenhum tipo de tratamento de dados, ruídos e imprecisões no giroscópio da IMU inferem movimentos mesmo quando em completo repouso. Para se corrigir esse problema, o software utiliza uma estratégia de calibração.

Nessa calibração, também utilizando funções "*write*" da biblioteca, o software solicita 2000 leituras do giroscópio, e calcula a média aritmética entre elas. Dessa forma se tem uma média das imperfeições presente e tal média é utilizada para a correção de futuras leituras.

Uma nova calibração é feita e novos valores são calculados toda vez que o sistema é ligado e o *software* se inicia. A posição em que a IMU se encontra durante a calibração não interfere no processo, porém é de suma importância que esteja em completo repouso enquanto o procedimento é feito.

6.2.2.3 Solicitação e leitura de dados da IMU

Semelhante à configuração da IMU, deve-se utilizar de funções de se iniciar a comunicação, enviar dados em registradores referentes ao que se é solicitado, e finalizar a comunicação.

Para se fazer a leitura, tem alguns detalhes a mais: É utilizada a função "*requestFrom"* solicitando 14 bytes da IMU, e a função *"available*" para se esperar a disponibilidade dos 14 bytes para leitura.

Dos 14 bytes lidos, 6 são bytes do acelerômetro (dois para cada eixo), 6 bytes do giroscópio, e 2 bytes do sensor de temperatura, que não é utilizado, por isso desprezado.

Todas a comunicação é feita no máximo com um byte de cada vez, mas as informações são compostas por dois bytes (*long int*). Para se fazer a leitura, deve-se juntar as partes *low* e *high* dos bytes e se ter a leitura como concluída.

6.2.2.4 Tratamento dos dados lidos

Os dados lidos são valores sem significado direto, mas proporcionais a grandezas físicas. A leitura do giroscópio responde com um valor de 65,5 a cada 1º/s que ele rotaciona (no fundo de escala escolhido, conforme *datasheet* do fabricante).

Dessa forma, temos uma resposta que depende de dois fatores. O valor resultante do giroscópio depende de quanto ele se desloca, e de quanto tempo esse deslocamento levou. Então, para que se torne possível adquirir uma leitura confiável, define-se um intervalo de tempo fixo (4ms) e sabe-se que a leitura sempre vai estar relacionada unicamente ao deslocamento angular do giroscópio.

Com isso, para se ter o ângulo em graus deslocado por um eixo do giroscópio nesse intervalo de 4ms, calcula-se:

$$\theta(°) = \frac{leitura \times 0{,}004}{65{,}5} = leitura \times 0{,}0000611$$

Ou seja, para se transformar o valor lido em graus, nesse fundo de escala e nessa frequência de leitura (1/4 ms = 250 Hz), basta multiplicar por 0,0000611.

Porém, em testes realizados com essa correção calculada, eram obtidos apenas 1/10 do valor esperado (exceto para o eixo Z (*yaw*) que se obtia o valor correto). Após consultas no manual do fabricante, foi decidida a modificação do valor. De 0,0000611 seria modificado para 0,000611, de forma que os valores dos eixos X e Y (*pitch* e *roll*) se tornassem corretos, e o valor do eixo Z (*yaw*) fosse multiplicado por fator de 10. Os ajustes então se tornam:

$$pitch_1 = X \times 0{,}000611$$

$$roll_1 = Y \times 0{,}000611$$

$$yaw_1 = Z \times 0{,}000611$$

Além disso, para a correção de rotações sucessivas, ou seja, rotacionar primeiro em X e depois em Z, ou primeiro em Y e depois em Z, é necessário adicionar, proporcional ao seno da rotação em Z, o eixo Y em X ou subtrair o eixo X em Y. Ou seja:

$$pitch_2 = pitch_1 + roll_1 \times sen(yaw/10)$$

$$roll_2 = roll_1 - pitch_1 \times sen(yaw/10)$$

Não só os dados do giroscópio, mas os do acelerômetro também são lidos sem significado direto. Para o modo de operação escolhido, o acelerômetro responde com o valor de 4.096 quando sofre aceleração de 1g (9,81 m/s²).

Para se calcular os valores de *roll* e *pitch* fornecidos pelo acelerômetro, utiliza-se a seguinte estratégia:

- Calcula-se o vetor total de aceleração baseado nas acelerações individuais de cada eixo. (Raiz da soma dos quadrados de cada eixo)

- Calcula-se a inclinação total de cada eixo baseando a resposta de cada eixo em relação ao vetor total.

A Figura 65 ilustra tal estratégia:

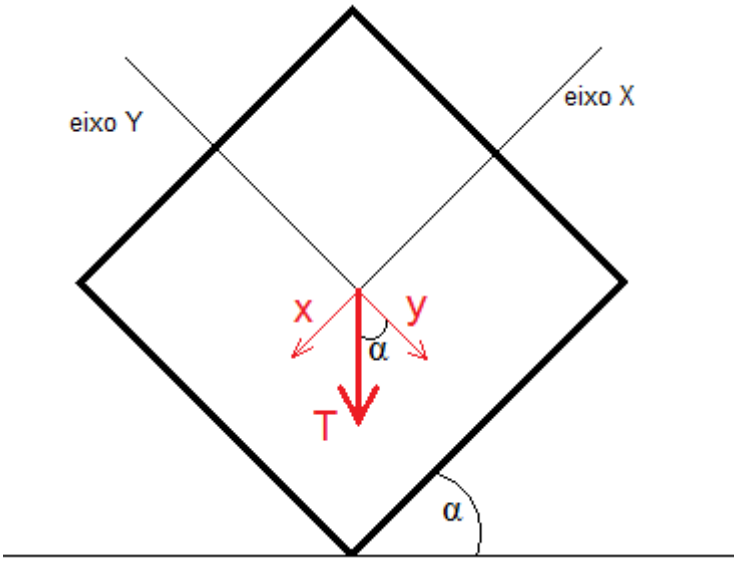


Figura 65 – Representação de eixos e ângulos do acelerômetro

Sendo T o vetor total de aceleração, pode-se dizer que:

$$x = T \times \text{sen}(\alpha)$$

Logo:

$$\alpha = arcsen\left(\frac{x}{T}\right)$$

Como não utilizamos valores absolutos fornecidos pelo fabricante (4.096) como o valor de referência para se calcular a inclinação, mas sim valores relativos a uma nova referência calculada (vetor total de aceleração), garantimos que imprecisões no equipamento, utilização do equipamento em outras partes do mundo (diferentes acelerações gravitacionais) e translações no equipamento durante movimentações de angulares não debilitem o funcionamento do sistema. Com isso se tem:

$$apitch = arcsen\left(\frac{aY}{T}\right)$$

$$aroll = arcsen\left(\frac{aX}{T}\right)$$

Onde:

apitch é o ângulo de *pitch* calculado pelo acelerômetro.

aroll é o ângulo de *roll* calculado pelo acelerômetro.

aY é a resposta do acelerômetro para o eixo Y.

aX é a resposta do acelerômetro para o eixo X.

No cálculo do valor final de *pitch* e *roll*, após alguns testes em bancada, se escolheu 90% de contribuição pelo giroscópio e 10% de contribuição pelo acelerômetro. O resultado final de pitch e roll então se torna:

$$pitch_{final} = 0{,}9\, pitch_2 + 0{,}1\, apitch$$

$$roll_{final} = 0{,}9\, roll_2 + 0{,}1\, aroll$$

Para o yaw, como não se precisa de quantos graus ele rotacionou em relação à sua posição inicial (posição que se encontrava quando o sistema foi ligado), e como esse é um método iterativo, onde um intervalo de 4ms separa os instantes de leitura, multiplica-se o valor de *yaw* por um "fator de zeramento", ou seja, um valor que faça com que a cada iteração o valor de yaw decaia e se aproxime de 0. O valor escolhido na versão final do projeto foi 0,995. Assim:

$$yaw_{final} = 0{,}995\, yaw_1$$

6.2.2.5 Tratamento dos dados para envio

A sessão anterior mostra como os dados são pré-tratados. Essa sessão agora visa mostrar o tratamento final, convertendo o valor de ângulo em passos para os potenciômetros digitais e o envio da mensagem.

A função de tratar e enviar as mensagens para envio é chamada somente a cada 50 contagens no laço principal. Como já dito na sessão 4.2.2.3, a mensagem enviada é da forma "#**.**.**.**;", com seus quatro campos de informações (representados por **) sendo: JEV, JDV, JDH e JEH, respectivamente. Sua primeira atividade é, então, um algoritmo de seleção para interpretar o valor do JEV que deve ser enviado.

Internamente no software de controle, tem um valor de força que varia em 7 níveis (de 0 a 6). Dependendo do valor de força presente no controle, ele associa a um valor de passo para o potenciômetro digital JEV da atuação. A associação de valor de força para passo em JEV é a mostrada pela Tabela 8.

Tabela 8 – Correspondência em Valor de força para JEV.

Valor de força no controle (0 a 6)	Passo no potenciômetro digital JEV (0 a 31)
0	31
1	29
2	26
3	22
4	17
5	12
6	0

Em seguida, o software transforma os valores de *roll* e *pitch* obtidos anteriormente para valores em passos para JDV e JDH. Para melhor acoplamento das peças no interior da caixa de acrílico projetada, a IMU teve sua orientação modificada, fazendo com que o *roll* da IMU correspondesse ao *pitch* do VANT, e vice-versa.

A primeira verificação, é definir os limites de operação. Primeiro é verificado o *pitch* da IMU e se sua inclinação é superior a 60° ou inferior a -60°. Caso seja, é enviado à atuação 31 ou 0 respectivamente. Caso não seja, calcula-se o valor a ser enviado da seguinte maneira:

$$\text{JDV} = \frac{pitch_{final} + 60}{4}$$

O mesmo é realizado para o JDH, porém com algumas correções de eixo. JDH então é:

$$\text{JDH} = \frac{-roll_{final} + 60}{4}$$

Dessa maneira, os potenciômetros JDV e JDH operam de maneira quase linear, possuindo somente uma descontinuidade quando o numerador se torna 120, ou seja, quando $pitch_{final}$ é +60 ou o $roll_{final}$ é -60. O funcionamento de ambos pode ser visto na Figura 66.

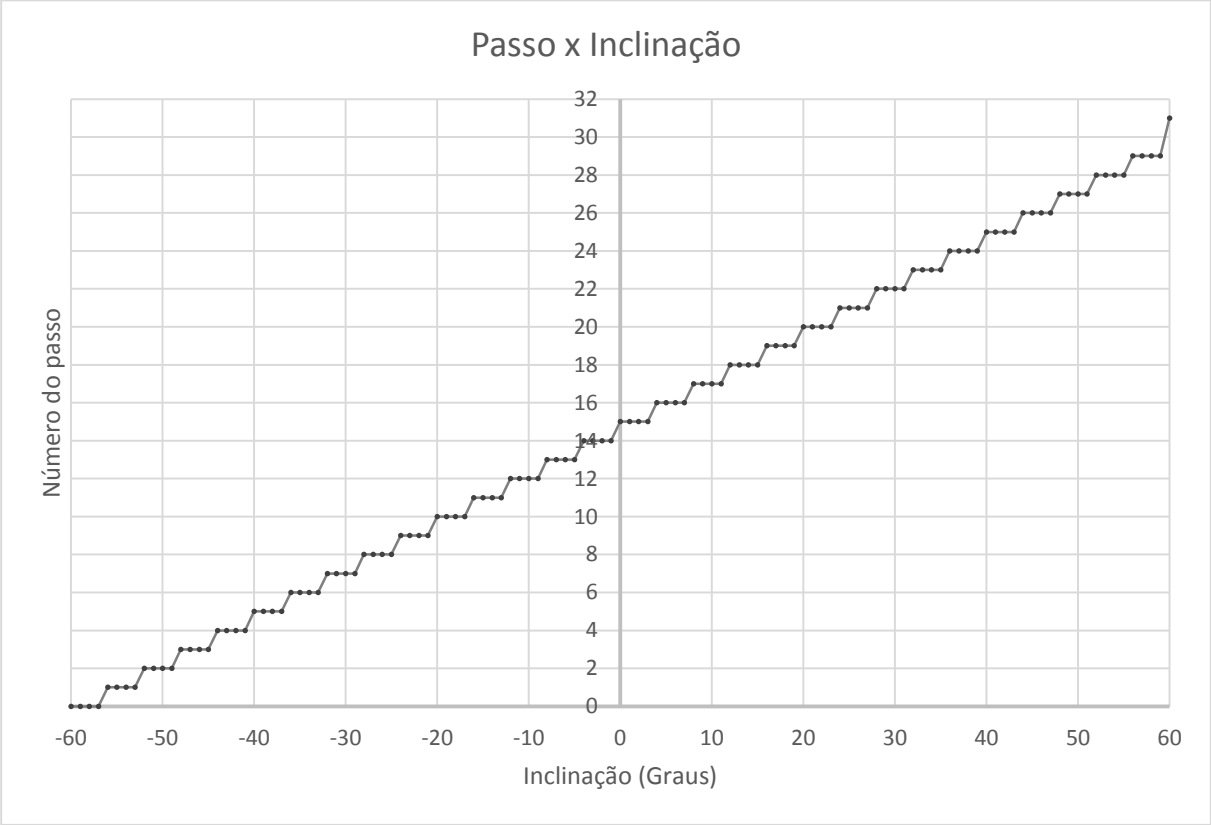


Figura 66 – Tratamento quase linear de graus para número de passos

Por fim, o *yaw* possui um tratamento único. Pois além de responder com o ângulo multiplicado a um fator de 10, ele possui um decaimento ao longo do tempo.

Por meio de testes realizados durante o desenvolvimento, uma virada brusca de 90° para uma direção provocava uma resposta de 900 (90 x 10) e ao longo do tempo, a cada iteração (4ms) é multiplicada por 0,995, ou seja, a saída decai ao longo do tempo. Além, para se trazer esse valor em passos entre 0 e 31, esse valor bruto de 900 é refinado por um divisor de 40. O comportamento do decaimento pode ser conferido na Figura 67.

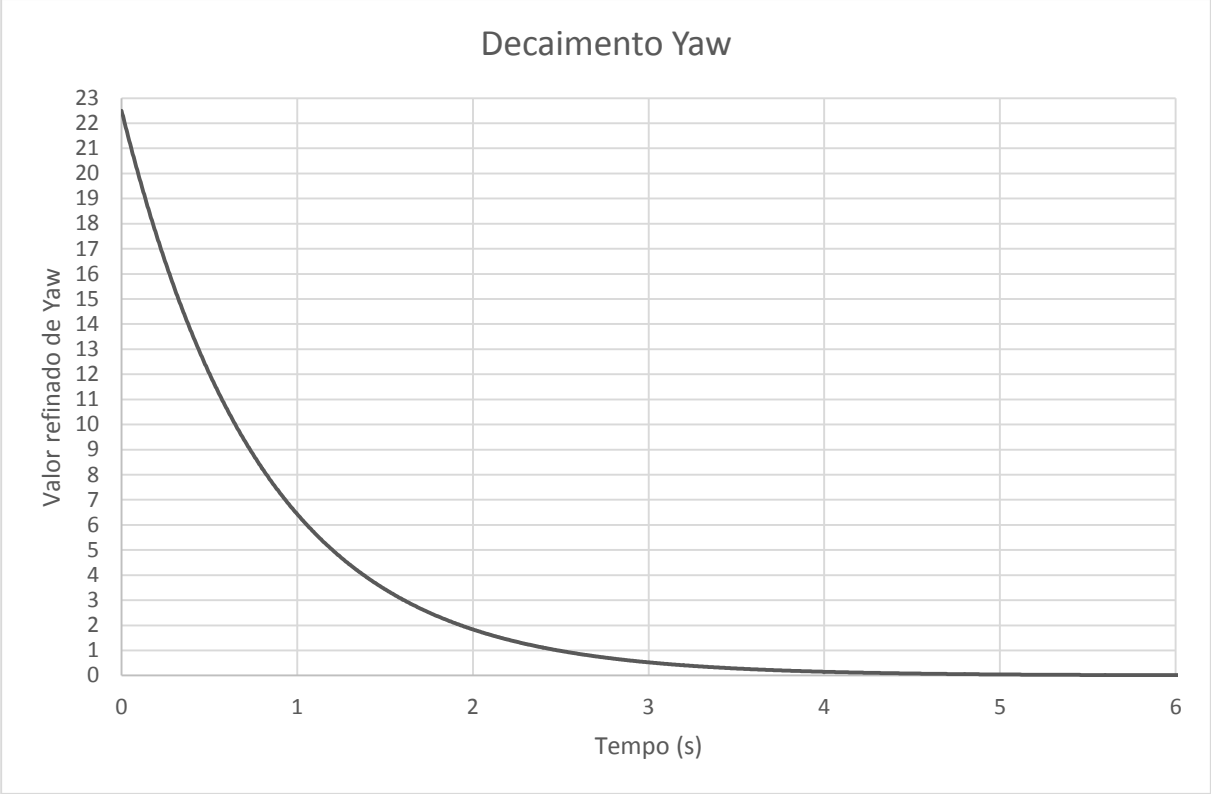


Figura 67 – Decaimento refinado de Yaw ao longo do tempo

Para se calcular, então, o valor enviado ao potenciômetro JDH, adiciona-se o valor refinado de *yaw* ao valor central do potenciômetro, ou seja, 15 (quinze). O *yaw* também recebe outro tratamento em respeito à sua saturação, ou seja, o valor de JDH a ser enviado não pode ultrapassar 31 nem ser menor que 0.

Como as posições de passos são valores sempre inteiros, basta que o valor refinado de *yaw* seja menor que 1 (não necessariamente precisa igualar-se a 0), para que seja considerado 0 em uma soma inteira. A figura 68 acima, assume que a movimentação em torno do eixo Z do giroscópio é de perfeitos 90° e o movimento é realizado em um intervalo menor que 4ms, ou seja, um movimento muito brusco. O gráfico acima mostra que, mesmo nessa situação absurda, o valor refinado de *yaw* torna-se menor que 1 em torno de 2,5s após o movimento.

Dessa forma, o potenciômetro JEH, sempre limitado entre 0 e 31, é:

$$\text{JEH} = 15 - \frac{yaw_{final}}{40}$$

Por fim, essa função também se encarrega de verificar o pressionamento dos botões de ajuste fino da rotação em torno do próprio eixo do VANT. Ela simplesmente verifica se o botão está pressionado, então adiciona ou subtrai 5 passos, de acordo com qual botão foi pressionado. Quando o botão esquerdo é pressionado, soma-se 5 passos, quando o direito, subtrai-se 5.

A mensagem pode enfim ser construída no formato #JEV.JDV.JDH.JEH; e enviada para a atuação utilizando um simples *println* com o Bluetooth.

6.3 Funcionalidades removidas

Durante o desenvolvimento do sistema de controle, pensou-se em uma suavização na curva de conversão de graus para passos apresentada na sessão anterior (Figura 67). Por métodos empíricos matemáticos chegou à seguinte fórmula a se aplicar para JDV e JDH:

$$Passo = \left(\frac{Inclinação}{60}\right)^3 15{,}5 + 15{,}5$$

Dessa maneira se teria uma curva de terceira ordem que tanto para valores menores ou maiores que o central (passo 15, inclinação 0), teria um crescimento suave em baixas inclinações e crescimento acentuado em altas inclinações.

O comportamento desse tratamento em comparação ao tratamento quase linear utilizado pode ser visto na Figura 68.

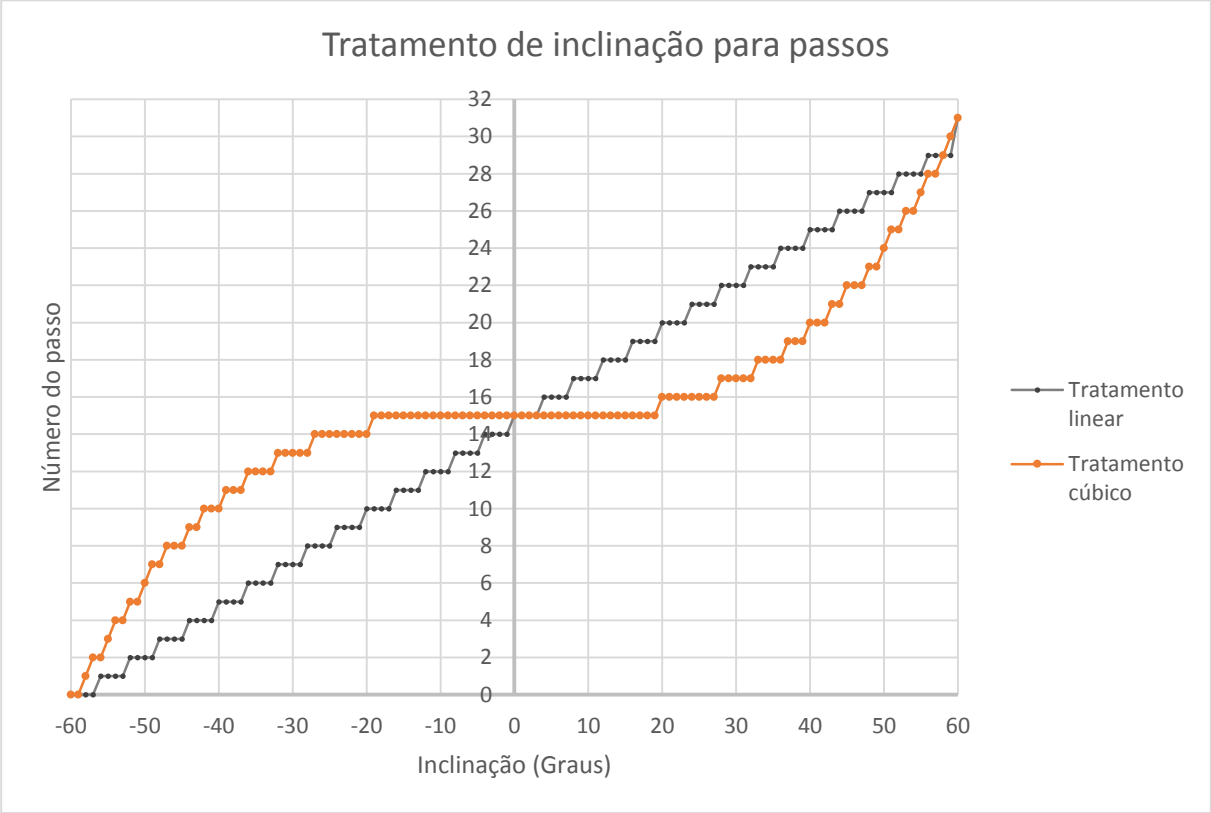


Figura 68 – Comparação de tratamentos linear e cúbico

Mas ao se ter o projeto quase em seu estágio final, realizando os últimos testes de voo em campo, percebeu-se que o tratamento cúbico dificultava a pilotagem.

Observando o gráfico acima, percebe-se que para inclinações menores que 20° o controle trata para o passo central de atuação. Criando, assim, uma banda morta entre -20° e +20°. Para o piloto, por conta dessa banda morta, a sensação é de que o VANT não responde à medida que você inclina, porém, após determinado ponto o VANT subitamente atua para determinada direção com muita intensidade.

Como essa funcionalidade proporcionou um funcionamento exatamente oposto ao desejado, optamos por removê-la da versão final do projeto e manter o tratamento linear.

Capítulo 7

Ensaios e Testes

> “ Tente uma, duas, três vezes e se possível tente a quarta, a quinta e quantas vezes for necessário. Só não desista nas primeiras tentativas, a persistência é amiga da conquista. Se você quer chegar onde a maioria não chega, faça o que a maioria não faz. ”
>
> – Bill Gates

7.1 Teste de Rotação

Para um bom controle, primeiramente é de suma importância caracterizar bem nossos atuadores. No caso do *drone* em questão, quem comanda as tensões individuais para cada motor é a placa interna do *drone* – não confundir que cada potenciômetro controla um motor (vide caracterização da placa original do controle). Sendo assim, nesse teste, através de um potenciômetro digital de 32 passos, aplica-se uma tensão na placa do rádio controle original do *drone*.

O objetivo desse teste é caracterizar a velocidade de rotação das hélices do drone de acordo com o aumento da tensão fornecida à placa do rádio controle.

Com esse objetivo, nesse teste, feito no início do trabalho, projetou-se um sistema para medir o comportamento da velocidade angular de uma das hélices em relação à tensão gerada por um dos potenciômetros. O potenciômetro de teste foi o JEV, por fazer o movimento ascendente do *drone*, aplicando tensões iguais e crescentes aos 4 motores. Todo o sistema de medição montado pode ser conferido na Figura 69.

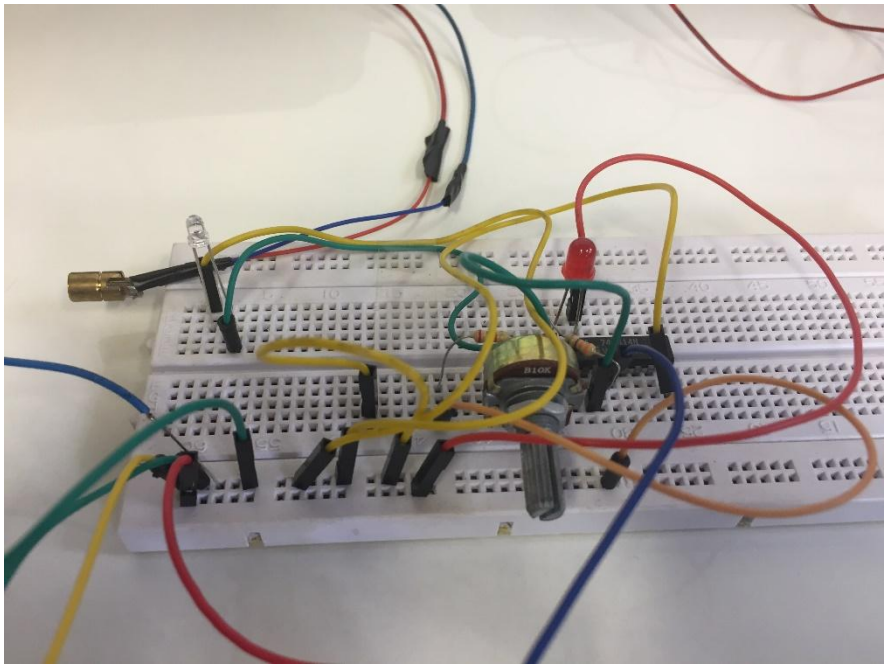


Figura 69 – Circuito eletrônico do sistema de medição

Com o intuito de compreender melhor o funcionamento de tal circuito, vamos estabelecer alguns conceitos básicos que foram usados.

7.1.1 Encoder absoluto e incremental

O *encoder* é um dispositivo usado na automação em geral que transforma o movimento rotatório em um trem de impulsos elétricos [20] servindo para determinar o deslocamento de movimentos circulares ou lineares e até mesmo a velocidade, se o tempo for levado em conta nos cálculos. Existem dois tipos de *encoders*: *Encoder* Absoluto e *Encoder* Incremental.

O *encoder* incremental, presente na Figura 70, gera um pulso para cada unidade de deslocamento. O *encoder* absoluto gera um código binário para cada unidade de deslocamento. Os dois sistemas usam a detecção fotoelétrica onde o trem de pulsos é gerado pela passagem da luz através de um disco codificado firmemente encaixado ao eixo de um motor ou em dispositivos mecânicos que transformem o deslocamento linear em deslocamento circular. Este sistema pode ser usado para detectar a posição de distâncias superior a 0,01mm.

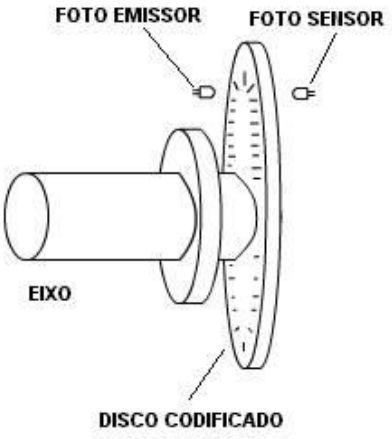


Figura 70 – Diagrama de um encoder incremental [21]

7.1.2 Foto transistor infravermelho

O TIL78 é um fototransistor geralmente montado como na Figura 71. Ele possui dois terminais, correspondendo ao coletor e emissor do transistor. A base é ativada pela luz; quando uma quantidade suficiente de luz é captada, o transistor conduz, permitindo a passagem de corrente do coletor para o emissor. Sem a luz, o transistor não conduz e coletor e emissor ficam isolados.

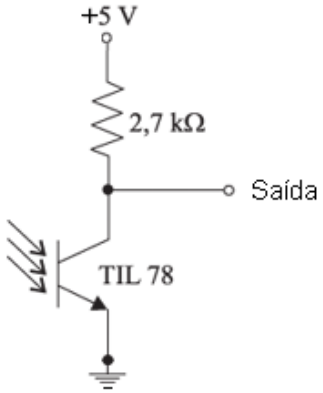


Figura 71 – Circuito exemplo com o TIL 78 (manual do fabricante)

Sem luz, o fototransistor não conduz e a saída está em 5V. Com luz, o fototransistor conduz. Com o laser usado, conectado diretamente na saída 3V3 do Arduino, apontado

corretamente, o fototransistor estará saturado e a tensão na saída será da ordem de 0,2V. Cuida-se para o alinhamento entre laser e o fototransistor e, para tanto, deve-se especial atenção para as dimensões e descrição dos pinos desse componente, mostradas na Figura 72.

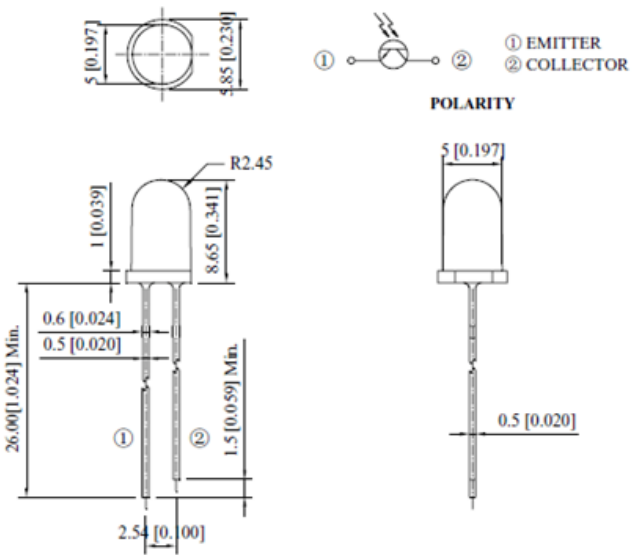


Figura 72 – Detalhes do Fototransistor TIL 78 - 5 mm (manual do fabricante)

Fisicamente o TIL78 é idêntico a um LED, sendo encontrado com encapsulamento transparente ou azulado. O terminal com o chanfro é o coletor, que deverá estar positivo em relação ao outro terminal (emissor). Note que tipicamente o terra é ligado ao lado "não chanfrado" do LED e ao lado "chanfrado", o resistor ligado ao VCC.

Para conferir o funcionamento, usamos novamente o multímetro na escala de 20k com o positivo ligado ao coletor e o negativo ao emissor. A resistência medida varia conforme a intensidade da luz.

Uma fonte de luz emissora que tentou-se usar foi a própria luz do ambiente ou uma luz difusa como o flash de uma câmera e pretendia-se registar a falta de luminosidade com sombra. Tendo em vista que, com alta rotação, as hélices não projetaram sombra sobre o fototransistor, optou-se por usar um laser, um feixe colimado de luz, pontual e bem mais forte que as opções anteriores.

Para termos controle da sensibilidade à luz do fototransistor, inseriu-se um potenciômetro analógico em série com o resistor já existente no pino coletor dele. Quanto maior a resistência, mais sensível à variação de luz fica o circuito. Ajustando adequadamente esse potenciômetro, conseguiu-se um resultado muito bom de detecção de interrupção das hélices.

7.1.3 Buffer Schmitt Trigger

Em muitas aplicações práticas o sinal de entrada de um circuito digital, não é um sinal perfeito, mas carregado de ruídos, o que prejudica consideravelmente funcionamento dos circuitos que dele dependem.

O *Schmitt trigger* é um circuito lógico usado especialmente para transformar uma forma de onda irregular ou senoidal em uma forma de onda quadrada, livre de imperfeições. O circuito integrado que utilizamos foi o 7414 da família TTL, ele é um sêxtuplo inversor limitador *Schmitt*.

O disparador *Schmitt* opera com duas tensões de referência, sendo uma inferior de 0,9V e uma superior de 1,7V, tipicamente, quando o sinal de entrada atinge 1,7V (referência superior) a saída comuta de nível lógico alto para nível lógico baixo, permanecendo neste estado até que o sinal de entrada alcance 0,9V (referência inferior).

O sinal de entrada e saída, em um Schmitt trigger típico, podem ser notados na Figura 73.

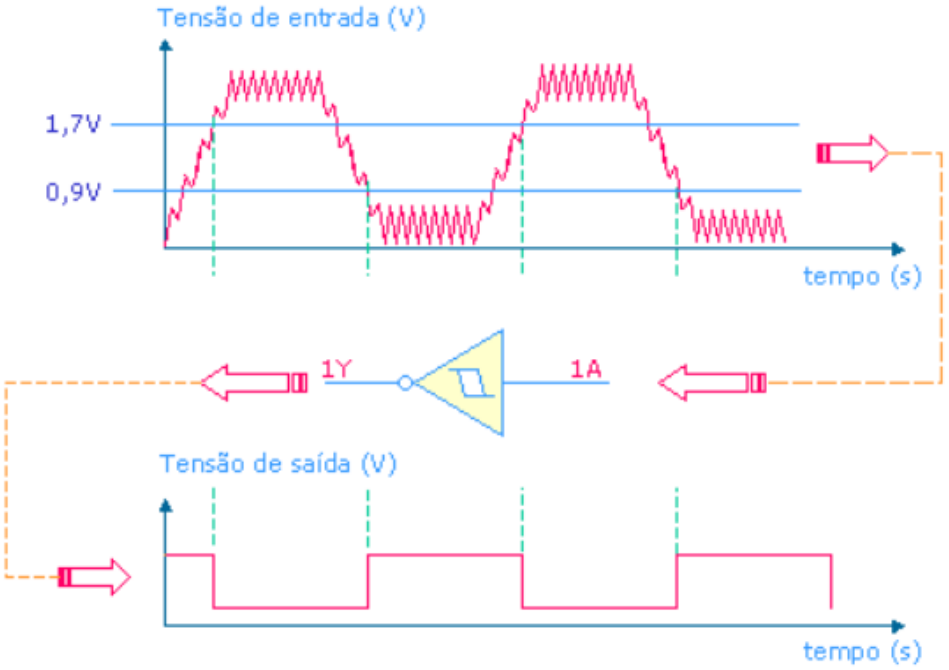


Figura 73 – Esquemático de funcionamento de um disparador Schmitt Trigger [22]

O sinal ruidoso representado é conhecido como *bounce*, muito comum em circuitos de comutações como chaves mecânicas. A função dos disparadores *Schmitt trigger* é garantir um sinal de saída limpo.

7.1.4 Funcionamento – Teste de rotação

A ideia na montagem para aplicação para o *drone*, em suma, foi fazer uma espécie de um grande *encoder* incremental para determinação da velocidade de rotação da hélice, sempre fazendo um paralelo com a tensão instantânea aplicada naquele momento pelo JEV.

A metodologia foi unir os conceitos de trem de pulsos por interrupções fotoelétricas vista comumente nos *encoders* [20] para, então, fazer a medida da velocidade angular da hélice. Aqui emitiu-se um feixe de laser de cima para baixo, passando pela região em que a hélice gira e chegando na bancada onde tem-se um receptor de luz, que é um fototransistor TIP 78 (Vide Anexo 4).

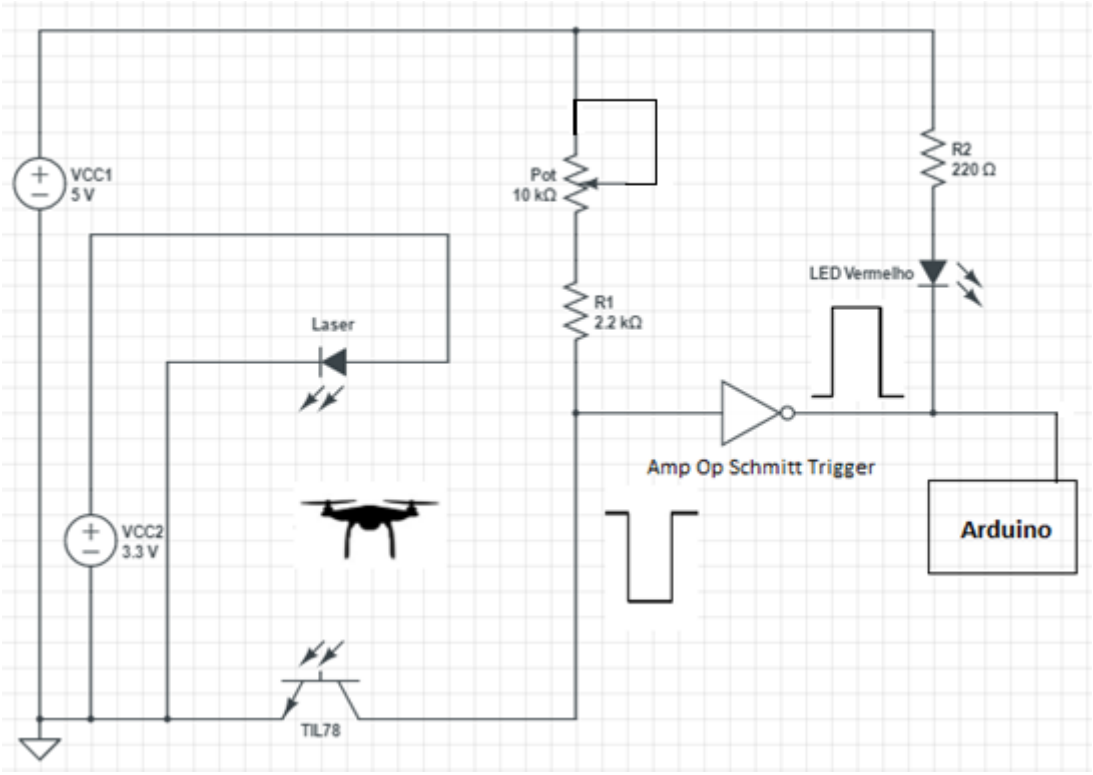


Figura 74 – Diagrama do circuito elétrico montado

Para se obter a onda quadrada desejada, monitorou-se o pino do coletor do transistor, como no diagrama da Figura 74. Como, por construção, a interrupção é verificada na borda de descida da onda, adicionou-se um Amplificador Operacional *Schmitt Trigger* para inverter a onda e corrigir eventuais histerese do processo.

Decidiu-se discretizar o problema no que diz respeito à variação da tensão enviada para os motores, em outras palavras, utilizar entradas subdivididas em passos discretos. Como já usamos os potenciômetros digitais para realizar tal controle e, com ele, conseguimos 32 passos, escolheu-se, portanto, medir essas 32 velocidades. Por interrupção no Arduino, fez-se um contador ser incrementado a cada interrupção por 2 segundos e para cada modo de velocidade. Naturalmente uma volta é contada após dois pulsos completos, devido as duas abas da hélice. Logo,

$$\frac{Count}{2} \quad ---------- \quad 2s$$

$$x \quad ---------- \quad 60s$$

É mostrado então que, para cada velocidade definida, tem-se 15*Count RPM (rotações por minuto).

Empiricamente, verificou-se que o nível da bateria influenciava muito nas curvas obtidas. Sendo assim, decidiu-se fazer o ensaio para duas baterias diferentes: a de 600 mAh e a de 1100 mAh. A intenção foi fazer vários testes até o *drone* dar sinal de bateria fraca e não conseguir mais girar as hélices, mesmo com o comando sendo enviado.

7.1.5 Resultados - Testes de rotação

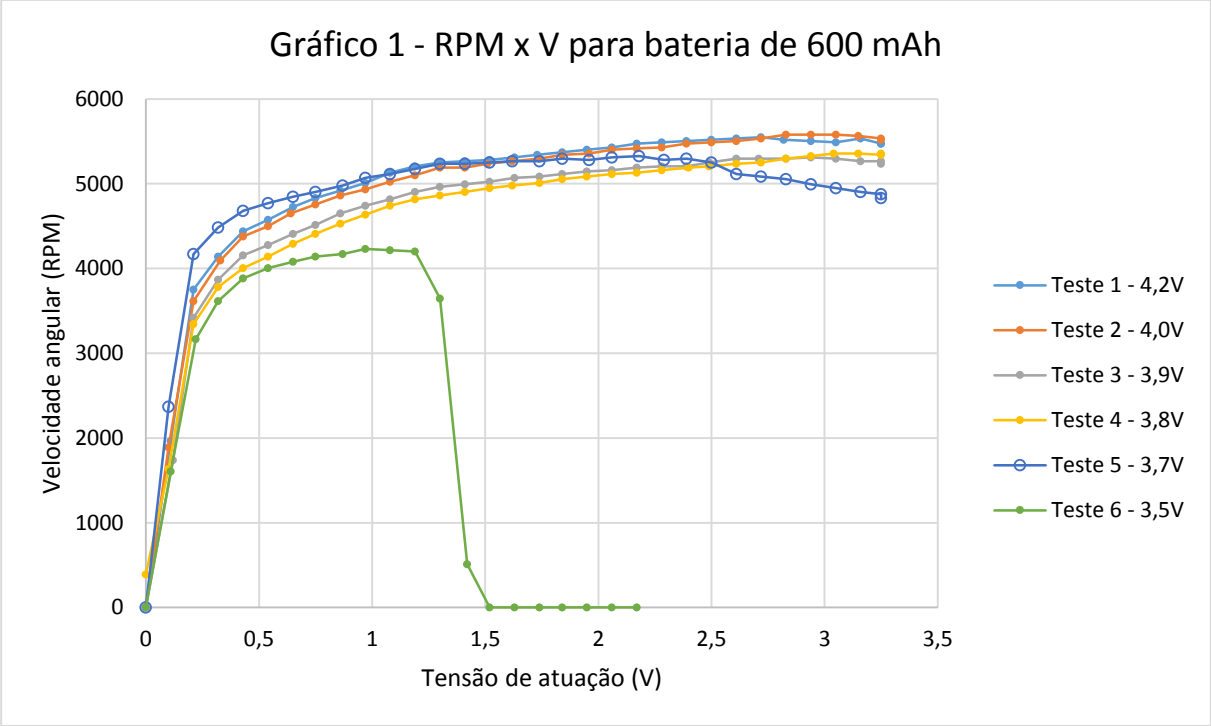


Figura 75 - RPM x V - 600 mAh

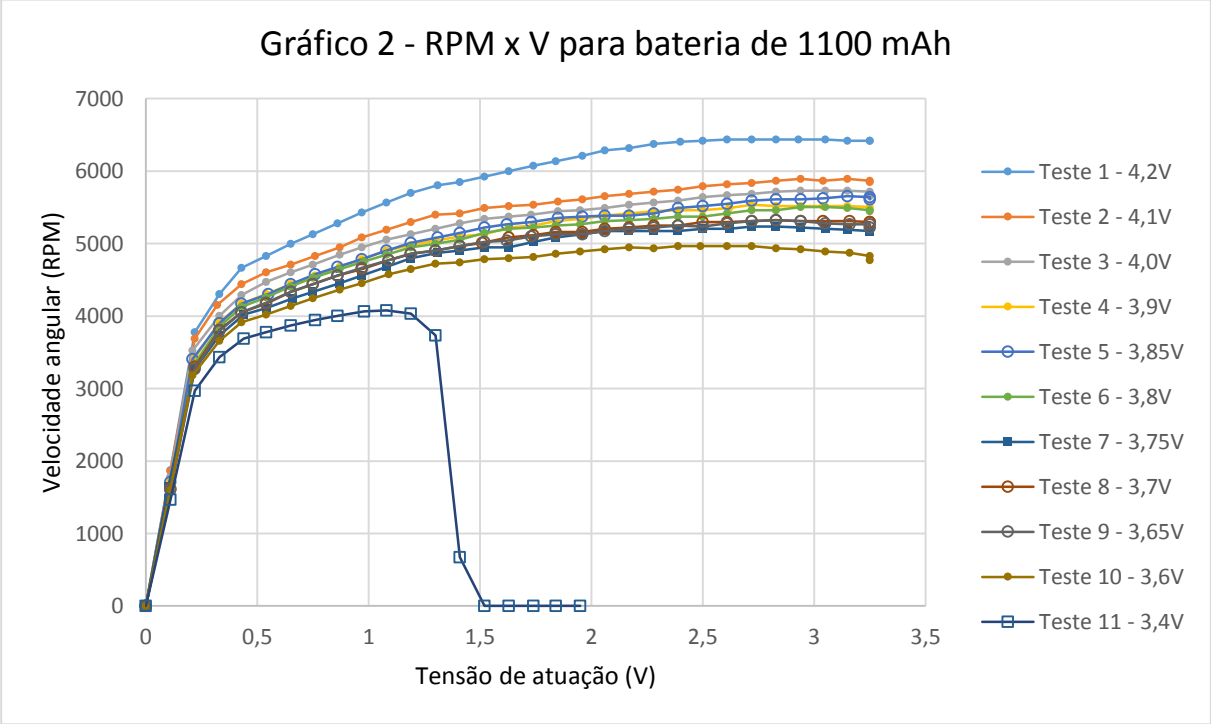


Figura 76 - RPM x V - 1100 mAh

Os gráficos 1 e 2 (Figura 75 e Figura 76) mostram as velocidades de rotação das hélices do VANT a medida que se aumentava a tensão fornecida à placa do rádio controle, para distintos testes com dois tipos diferentes de baterias. Para cada bateria realizava-se um teste e armazenava os dados, entre cada teste, media-se, com um multímetro, a tensão nos terminais da bateria para acompanhar seu descarregamento ao longo do uso.

À medida que a bateria foi sendo utilizada, menor foi se tornado a velocidade de rotação para uma mesma tensão de atuação. Como já fora explicado anteriormente, pelas conexões internas da placa o potenciômetro do *joystick* esquerdo na orientação vertical (JEV) tem sua atuação invertida, ou seja, 3,3V corresponde ao ponto morto e 0V sua velocidade máxima. Então, para melhor visualização, denominou-se 0V o ponto morto e 3,3V o ponto de velocidade máxima. Além disso, observou-se que de 3,3V efetivos até 3,25V não havia reação dos motores, eles se mantinham parados. Para tal, corrigiu-se os valores obtidos chamando de 3,25V a nova origem e a diferença entre a tensão efetiva e 3,25 seria a então denominada tensão de atuação.

Ou seja:

$$V_{atuação} = 3{,}25 - V_{efetiva}$$

Somado a tudo isso, com esse teste, notou-se que a bateria de 1100 mAh além de permitir uma autonomia significativamente maior que a de 600 mAh (pois permitiu a realização de 10 testes completos em contrapartida à outra que permitiu 5 testes completos). Com a bateria de 1100 mAh tem-se uma velocidade de rotação maior quando em comparação com a bateria de 600 mAh para um mesmo valor tensão de atuação.

Por fim, observou-se que, para as duas baterias, quando sua autonomia está próxima ao fim, a velocidade de rotação não passa muito além de 4000 RPM e a tensão da bateria encontra-se próxima de 3,35V.

7.2 Suavização da curva de rotação

Por conta do ensaio de rotação realizado, explicado na seção anterior, descobrimos que a rotação das hélices do VANT tem um crescimento acentuado em pouca tensão de atuação, e esse crescimento atenua à medida que a tensão de atuação aumenta (Figura 75 e Figura 76).

Uma das ambições do projeto, tornou-se, então, suavizar o crescimento dessa curva de forma a proporcionar mais suavidade ao piloto durante o voo. Para tal, contava-se com a utilização de um potenciômetro de 256 passos, além de se tornar necessário uma pequena alteração no circuito e no código.

Optamos por não substituir o potenciômetro de 32 passos por um potenciômetro de 256 passos a fim de se manter o trabalho anteriormente realizado. A alteração no circuito planejada, utilizando o potenciômetro digital de 32 passos e adicionando um potenciômetro novo de 256 passos é exemplificada pela Figura 77.

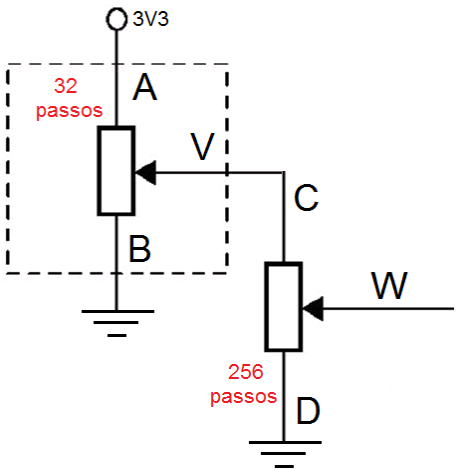


Figura 77 – Potenciômetros em utilizados em conjunto

Dessa forma, o segundo potenciômetro (com 256 passos) tem a capacidade de subdividir a tensão no terminal V do primeiro potenciômetro (com 32 passos) em qualquer valor entre tensão nula e a tensão no terminal V. Operando de forma semelhante a um ganho de 0 a 1 em 256 passos discretos na tensão presente no terminal médio do primeiro potenciômetro.

Para se encontrar qual ganho seria necessário para cada um dos 32 passos, tornou-se necessário encontrar uma função aproximada da rotação pela tensão de atuação. Utilizou-se então o resultado do teste de bancada para rotação em relação à tensão aplicada com a bateria de 600 mAh (Figura 75). Com tal resultado calculou-se uma linha média dos testes válidos. A regressão foi feita em cima dessa linha média, como mostra a Figura 78.

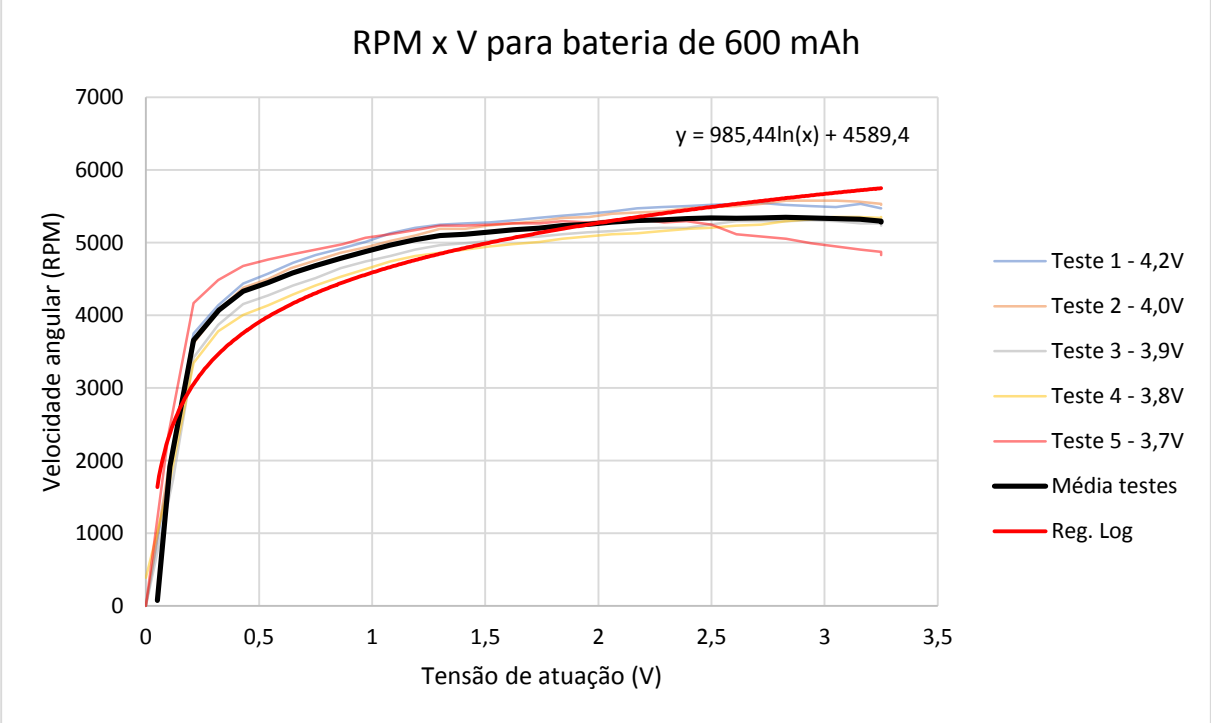


Figura 78 – Curvas de teste com média e regressão

A curva traçada em cor preta mostra a média dos testes 1 a 5, e a curva em cor vermelha mostra a regressão. A regressão utilizada foi regressão logarítmica, que foi a melhor encontrada.

A função de regressão, mostrada no gráfico acima, utiliza y para o eixo vertical e x para o horizontal. Traduzindo as variáveis se tem:

$$RPM = 985{,}44 \ln(V) + 4589{,}4$$

Com essa equação encontra-se sua inversa:

$$V = e^{\frac{RPM - 4589{,}4}{985{,}44}}$$

Com sua inversa, torna-se possível encontrar qual tensão confere determinada rotação desejada. Fazendo a razão entre as tensões desejadas para cada passo e tensões no terminal de saída do potenciômetro de 32 passos, encontra-se um valor de ganho entre 0 e 1 que deve ser aplicado pelo potenciômetro de 256 passos. Esse ganho pode facilmente ser convertido para um valor entre 0 e 255, como visto na Tabela 9.

Tabela 9 - Cálculo do passo necessário no potenciômetro de 256 passos para cada passo do potenciômetro de 32 passos

Passo (pot 32)	Tensão original	Tensão desejada	Ganho	Passo (pot 256)	Nova rotação
0	0	0	*	0	0
1	0,107	0,011	0,107	28	186
2	0,214	0,014	0,065	17	372
3	0,321	0,017	0,052	14	558
4	0,428	0,020	0,047	13	745
5	0,535	0,024	0,046	12	931
6	0,642	0,029	0,046	12	1117
7	0,749	0,036	0,048	13	1303
8	0,856	0,043	0,050	13	1489
9	0,963	0,052	0,054	14	1675
10	1,07	0,063	0,059	15	1862
11	1,177	0,076	0,064	17	2048
12	1,284	0,092	0,071	19	2234
13	1,391	0,111	0,080	21	2420
14	1,498	0,134	0,089	23	2606
15	1,605	0,161	0,101	26	2792
16	1,712	0,195	0,114	30	2979
17	1,819	0,236	0,130	34	3165
18	1,926	0,285	0,148	38	3351
19	2,033	0,344	0,169	44	3537
20	2,14	0,415	0,194	50	3723
21	2,247	0,502	0,223	57	3909
22	2,354	0,606	0,257	66	4096
23	2,461	0,732	0,297	76	4282
24	2,568	0,884	0,344	88	4468
25	2,675	1,068	0,399	102	4654
26	2,782	1,290	0,464	119	4840
27	2,889	1,558	0,539	138	5026
28	2,996	1,882	0,628	161	5213
29	3,103	2,273	0,733	187	5399
30	3,21	2,746	0,855	219	5585
31	3,317	3,317	1,000	255	5771

Com isso, se tem uma associação direta entre o passo do primeiro potenciômetro (32 passos) e qual deve ser o passo no segundo potenciômetro (256 passos). Para cada uma das posições possíveis no primeiro, se tem uma posição necessária no segundo. (Primeira e quinta coluna da tabela 7).

A última coluna mostra a nova rotação, ou seja, a rotação esperada para cada um dos 32 passos do primeiro potenciômetro com a correção realizada pelo segundo potenciômetro. De modo a facilitar sua visualização, esses dados foram organizados em um gráfico, presente na Figura 79.

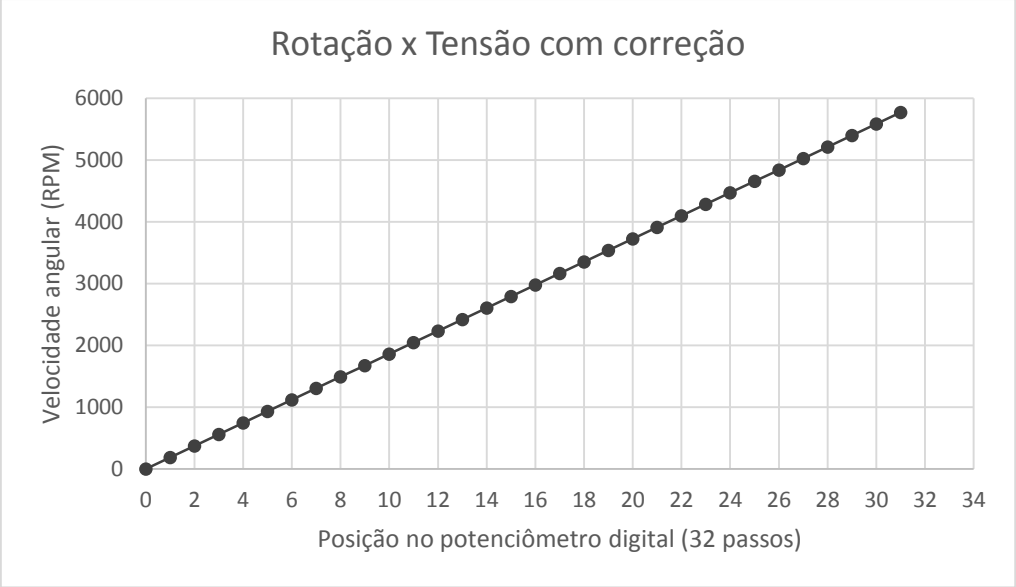


Figura 79 – Curva de rotação linearizada por um novo potenciômetro

Em código, poucas alterações foram necessárias. A única alteração necessária foi a criação de uma nova rotina. Essa rotina deveria ser chamada em toda alteração no potenciômetro de 32 passos e realizaria um grande teste para corresponder o passo necessário no potenciômetro de 256 passos.

Por fim, essa capacidade extra ao sistema não foi adicionada. Ao se montar o circuito, e iniciar os testes, observou-se que a rotação das hélices do VANT em tensões próximas a tensão de atuação nula é muito instável. Inconsistências em relação ao procedimento de calibração, que define os valores mínimos e máximos de tensão que serão considerados para aquele uso, a queda de tensão na bateria do VANT que interfere nas velocidades de rotação, e também, por imprecisão no modelo encontrado pela regressão logarítmica, causou imperfeições durante o acionamento dos motores.

Ao tentar iniciar o acionamento dos motores, partindo do repouso, os primeiros passos, por possuírem uma tensão muito baixa, nem sequer iniciavam a rotação das hélices. O mesmo ocorria para vários passos em diante, quando repentinamente um único passo iniciava uma rotação já muito forte (próximo ao 10º passo).

Em cálculos teóricos, esse sistema seria ótimo para o manuseio do VANT, mas como na prática causou exatamente o efeito oposto que buscávamos, ou seja, em vez de um crescimento suave observava-se um crescimento abrupto próximo ao 10º passo, optamos por não incluir essa funcionalidade nas versões finais do trabalho.

7.3 Testes finais

Após todas as montagens realizadas ao longo do projeto, decidiu-se testar o funcionamento de todos os sistemas trabalhando em conjunto e realizar voos com o VANT. Apesar do voo ter sido realizado com sucesso, a pilotagem ainda não está perfeita e pode ser melhorada. A Figura 80 mostra o *drone* em pleno voo. No voo testado teve-se a utilização integral dos sistemas (de controle e de atuação) desenvolvidos neste trabalho.

Figura 80 – Voo do VANT controlado com movimento

Como já citado anteriormente (Seção 4.2.2.2), a configuração dos potenciômetros digitais exige um pulso de largura mínima de 40ms. Além disso, esses potenciômetros tem seu posicionamento de forma incremental, ou seja, para se deslocar do 1º ao 10º passo, são necessários 9 intervalos de 40ms. Por conta disso, não se obteve um bom tempo de resposta de atuação do VANT, o que dificulta a pilotagem e não cumpre com o objetivo de fazer o voo intuitivo para iniciantes.

Apesar da pilotagem não ser intuitiva como objetivada, ainda é possível pilotar o *drone* após uma certa prática do piloto, o *drone* move-se corretamente obedecendo os movimentos de rotação realizados pelo piloto no sistema de controle.

O voo do teste foi gravado e pode ser assistido pelo acesso do seguinte endereço *web*: https://youtu.be/6tNt1V9qX0s. Endereços para mais vídeos que mostram em destaque o controle e a resposta do drone podem ser encontrados no apêndice (Apêndices 4 a 11).

Capítulo 8

Conclusão

“ O único modo de escapar da corrupção causada pelo sucesso é continuar trabalhando. ”

– Albert Einstein

8.1 Considerações Gerais

Nesse projeto foi possível estudar com detalhes as peculiaridades de um Veículo Aéreo Não-tribulado, desde a construção do contexto histórico favorável para seu surgimento, passando pela compreensão da sua dinâmica de voo e, por fim, à pesquisa minuciosa de todos os aspectos de um *drone* versão comercial.

Com base nos ensinamentos primários sobre o Syma X5SW do trabalho de graduação anterior, juntamente com a forma de abordagem para alteração de seu funcionamento, foi possível fazer uma série de modificações positivas para o sistema.

Primeiramente, abandonou-se a aplicação *mobile* que simulava os *joysticks* e controlava o *drone* de forma imprecisa com os dedos na tela, para um sistema mais robusto e estável, que também não dependesse de um sinal PWM do Arduino e um filtro passa baixas simples para construir um DAC. Além disso, teve-se fortemente a motivação inicial de caracterizar melhor o comportamento elétrico dos motores do *drone* respondendo aos níveis de tensão fornecidos pelos potenciômetros, com o objetivo de se tentar fazer um controle mais refinado da atuação do VANT.

Para substituir o sistema DAC com um PWM mais filtro RC, teve-se a ideia de usar potenciômetros digitais. Na época do projeto, o potenciômetro digital com a resistência nominal adequada para o projeto que foi possível encontrar foi o incremental de 32 passos, o x9511wp. Apesar de suas limitações, ele mostrou-se bem mais fácil de entender, de implementar e de fazer manutenção, bem como solucionou o problema de alteração da placa original de forma mais precisa que anteriormente.

De posse dessa funcionalidade, logo de início foram feitos os ensaios de rotação, para mapear como a rotação das hélices variava com o nível de tensão fornecido. Além de ver que o nível de bateria impactava significativamente os resultados, viu-se que a curva rotação x tensão apresentou um comportamento logaritmo, com derivada bem próxima do valor unitário para as rotações iniciais e seguindo para uma assíntota horizontal ao final no regime de rotação máxima do motor. Como esse teste foi feito de forma sistemática com os potenciômetros de 32 passos, ou seja, variando a tensão com intervalos de aproximadamente 0,1 V, acreditou-se que o comportamento fosse, de fato, contínuo.

Posteriormente, em um estudo mais cuidadoso e minucioso com foco somente nessa parte inicial da curva, verificou-se que a hipótese inicial de continuidade da função logarítmica estava equivocada. Esse teste, que foi feito com uma fonte DC precisa, mostrou que, não só o *drone* tinha uma zona morta de tensão em que ele não levanta voo, como existe uma rotação mínima que já se mostrou elevada para o *soft-start*, além das descontinuidades intrínsecas notadas, em que um certo intervalo pequeno de tensão resultava em uma dada rotação e o intervalo seguinte, num valor de RPM mais elevado, impossibilitando que se atuasse nesse intervalo. Infelizmente

essa constatação eliminou a solução que se tinha de usar o potenciômetro digital de 256 passos, que chegou na segunda etapa do projeto, para se atingir valores de tensão menores para rotações baixas, através de um circuito. Para solucionar esse problema foi pensado em fazer um controlador proporcional com um amplificador operacional de ganho menor que 1, para reduzir a tensão na região que se queria, mas como o gargalo não está em como gerar a tensão, essa alternativa também não iria obter sucesso.

Mais adiante, já na segunda etapa do projeto, foram feitos o estudo preliminar e a implementação do sistema com o acelerômetro MPU6050. A alternativa de usar a plataforma de medida inercial veio de aumentar a interatividade com o controlador e como uma forma de explorar as limitações dessa ferramenta para controle de um *drone* comercial. O módulo MPU, o módulo Bluetooth e o Arduino Pro Mini foram todos testados e validados para a aplicação previamente em ensaios em *protoboard* para então se fazer a PCB para que fosse possível alocar tudo dentro de uma estrutura de acrílico que fosse confortável de se controlar um *drone* com os movimentos de inclinação e rotação. Juntamente nessa etapa, o hardware de atuação também perdeu seu caráter de protótipo e foi integrado numa PCB própria.

Os dois sistemas entraram em comunicação e se fez testes de desempenho da resposta do *drone* voando com o novo controle. O foco principal dessa etapa foi verificar a eficiência do método matemático criado para mapear os ângulos gerados pela MPU em número do passo do potenciômetro correspondente. Mesmo com a forma original da placa sendo uma função linear, tentou-se incrementá-la como uma função polinomial em que se atenuava as respostas do *drone* para ângulos em torno de 0, com o objetivo de melhorar a dirigibilidade. Em testes de campo, a forma linear inicial mostrou-se melhor, pois o VANT reagia de forma mais rápida e mais confortável para o usuário, talvez pelos mesmos problemas de descontinuidades nas rotações verificados logo no início do projeto.

Por fim, de forma geral, o projeto atingiu os objetivos previamente propostos da forma mais otimizada possível. O controle remoto original foi integralmente substituído pelo sistema com um acelerômetro, da mesma forma como caracterizado nos objetivos. A única parte que não foi possível ser implementada foi o *soft-start*, entretanto esse desafio trouxe muitos ensinamentos e ideias para pesquisas que vão se seguir.

8.2 Aprendizados

Ao longo do projeto, algumas constatações importantes foram tiradas, evitando que tempo fosse desperdiçado e que novas alternativas fossem criadas. Essas informações servem para trabalhos futuros também, pois poupam muito tempo gasto de forma desnecessária.

O *drone* Syma X5SW tem suas restrições típicas de um *drone* de baixo custo. Seu controle de estabilidade funciona bem somente para pitch, roll e yaw, mas não existe nenhum tratamento para controle de altitude, como alguns *drone* mais caros têm. A ausência dessa funcionalidade faz o comportamento de ascensão vertical muito instável, exemplo disso são as constantes faltas de sustentação no ar que demonstra o *drone*, figurando uma grande dificuldade para quem o está operando.

Um dos fatores que pode explicar o problema da sustentação é a alta sensibilidade do *drone* com o nível de bateria. A bateria representa o maior gargalo de desenvolvimento com esse *drone*. A autonomia é baixa e, como verificado nos gráficos de rotação por tensão, qualquer descarga mínima altera como o avião reage. O compartimento da bateria também não comporta uma de potência maior.

O problema da descontinuidade dos motores, que inviabilizou o *soft-start* (vide Seção 7.2), tem possível origem em três locais, não se sabe ainda qual. Ou no protocolo de comunicação via rádio e/ou na pequena placa interna ao *drone*, que trata os dados recebidos, e/ou nos próprios motores. Por limitação de tempo, nenhum deles foi estudado a fundo.

Outro aprendizado importante que vale relembrar aqui é a forma de calibração do VANT. A ideia de que ele não precisa calibrar aqueles *joysticks* que são retráteis, somente o JEV, deve ser sempre levada em conta. Se tais potenciômetros não estiverem nas suas posições centrais no momento de se ligar a placa original, após a sincronização com o *drone*, ele vai ficar pendendo para algum dos lados.

A opção *Trimming* original do *drone* também deve ser sempre usada toda vez que for usá-lo, uma vez que não se tem como garantir que a bateria está 100% carregada e nem que os motores e seus sistemas de engrenagem e eixo estão perfeitamente iguais. Além do uso do *trimmer* para eliminar agentes externos ao voo, como correntes de vento, por exemplo.

Um aprendizado grande foi a fabricação das placas de circuito impresso. Um esforço significativo de tempo foi gasto nessa etapa, para que fosse possível fazer várias tentativas até chegar num resultado suficientemente bom. Dentre as diversas variáveis que influenciaram aqui, destacam-se a forma de roteamento das trilhas, as alternativas de passagem das trilhas para a placa de fenolite e os métodos de eliminação de cobre não usado.

No que tange ao ambiente de software, o MPU requisitou um tempo para o entendimento da forma como trabalha e como trata seus resultados. Principalmente na conversão de velocidade angular para ângulo e na forma como tenta isolar cada inclinação das demais, eliminando as influências entre elas.

No mesmo contexto, o conhecimento aprofundado das funções de interrupção foi crucial para o funcionamento do projeto como um todo. Aqui a limitação de tempo de resposta alterava muito a navegação com o *drone*, fazendo com que uma alternativa que usasse *polling*, por exemplo, provocasse a falha completa da aplicação.

8.3 Trabalhos Futuros

Tendo em vista todas as considerações gerais desse projeto com relação à pesquisa passada, bem como os diversos aprendizados deste, pode-se garantir grande potencial de desenvolvimento nessa área do conhecimento. O mercado de *drone* tanto profissional quanto para entretenimento, vem crescendo cada vez mais e, dessa forma, qualquer forma de explorar essa tecnologia é bem-vinda.

Ao longo do projeto alguns pontos não foram aprofundados por limitação de tempo ou de material. Destacam-se aqui algumas sugestões para os eventuais próximos trabalhos que podem seguir deste. As primeiras sugestões dizem respeito à melhora das limitações do Syma X5SW e as demais servem para qualquer tipo de *drone*.

Em primeira instância, sugere-se que se pense em uma alternativa para melhorar o desempenho da bateria do *drone*, fazendo com que ele não fique tão sensível às variações desta. Sobre a ideia de alteração da curva dos motores, aqui há espaço para um estudo mais detalhado da placa interna do *drone* com o objetivo de entender melhor seus atuadores, os motores.

Segundamente, a ideia, que foi a que inspirou o início desse trabalho, é a de expandir tudo que foi feito aqui para uma aplicação *mobile*, utilizando-se da plataforma inercial de um *smartphone*. Foi visto que isso é totalmente possível e que a experiência de direção dessa forma

facilita muito o aprendizado de um novato nessa área da aviação, além de assim ser possível difundir a tecnologia para um maior número de pessoas.

Como terceiro ponto encoraja-se fortemente que um controle mais refinado fosse feito no sistema. Nesse trabalho, tudo é feito em malha aberta e, para corrigir as imperfeições, adicionou-se alguns botões de ajuste fino. A ideia aqui seria integrar alguns sensores no próprio *drone* para que eles fizessem o feedback da sua resposta. Com outra MPU, por exemplo, acoplada ao *drone* seria possível comparar, em tempo real, a posição desejada com a posição real, tentando fazer o erro tender a zero. Isso pode ser feito de forma simples comparando esses valores via software e deixando-os bem próximos quanto possível. Todavia, para maior robustez do sistema, uma abordagem de um controle digital tradicional poderia ser aplicada. Seria preciso fazer a modelagem matemática da função de transferência da planta, o *drone*, no domínio de Laplace, para então partir para o projeto de um controlador. As técnicas seriam as mais conhecidas dependendo do objetivo, a saber: Lugar geométricos das raízes (LGR), controlador em avanço e/ou atraso, espaço de estados ou *deadbeat*. Sabendo que essas técnicas demandam um esforço matemático elevado e que dependem muito da exatidão com que o modelo eletromecânico do *drone* foi feito, recomenda-se um estudo maior sobre controle adaptativo, que promete ajudar bastante nesse contexto.

Como quarto ponto, uma prática, porém muito eficiente melhoria seria a substituição dos potenciômetros digitais de 32 passos para outros potenciômetros que possuam respostas mais rápidas, ou seja, não possuem a limitação de 40 ms para se deslocar um passo, que em vez de ser incremental tivesse a capacidade de livremente se deslocar para qualquer posição. Como durante o desenvolvimento deste trabalho já se fez a aquisição de um novo potenciômetro digital com 256 passos, os equipamentos necessários para tal substituição já está disponível e após essa substituição certamente o sistema teria um tempo de resposta significativamente menor e isso afetaria diretamente na pilotagem do VANT.

Por fim, como quinto fator de possível melhoria, destacam-se outros sensores que seriam totalmente possíveis de serem integrados ao sistema para seu aperfeiçoamento. Sensores de distância, como ultrassons e lasers - LADAR (*Laser Detection and Ranging*) - serviriam para melhorar o controle de altitude, que o Syma X5SW infelizmente não possui, apontando-os para baixo, ou até mesmo para fazer desvio de obstáculos e evitar colisões, sendo colocados ao redor do *drone*. Esses sensores juntamente com uma câmera melhor ajudariam também a fazer um estudo melhor sobre SLAM (*Simultaneous Localization And Mapping*) com *drones* em ambientes inexplorados. Outra alternativa usando visão computacional seria o controle pelo reconhecimento de objetos cotidianos, fazendo com que o VANT possa segui-los, como a mão do usuário ou alguma *tag* em seu corpo. Por fim, um sensor embarcado que seria bem útil é um GPS, em que se poderia traçar a trajetória do avião somente com coordenadas geográficas.

Portanto, é visto que ainda existe muito o que se possa melhorar nessa área do conhecimento em âmbito acadêmico. A única forma de saber se as ideias serão boas também na prática é tentando implementadas, mesmo porque o fato de tentar algo aguça a criação de novas ideias e alternativas para próximos trabalhos, de tal forma que a inspiração que esse trabalho traz nunca se acabe.

Referências Bibliográficas

[1] "Wikipédia - História da Aviação," 02 Novembro 2016. [Online]. Available: https://pt.wikipedia.org/wiki/Hist%C3%B3ria_da_avia%C3%A7%C3%A3o.

[2] B. C. da Silva e P. O. Libardi , "Sistema de Controle de um VANT," Universidade de Brasília, Brasília, 2015.

[3] M. V. COOK, Flight Dynamics Principles, vol. Elsevier Aerospace Engineering Series, Burlington: Elsevier/Butterworth-Heinemann, 2007.

[4] M. Â. A. da Costa e D. F. Vale, "Implementação do hardware e software de estabilização de um robô aéreo com dois rotores articulados.," Universidade de Brasília, Brasília, 2016.

[5] DJI, [Online]. Available: https://www.dji.com/phantom. [Acesso em 17 Setembro 2016].

[6] D. Drone. [Online]. Available: http://doctordrone.com.br/drones-para-que-confira-40-usos-da-tecnologia/. [Acesso em 22 Abril 2017].

[7] "Wikipédia - Quadrotor," Wikipédia - Quadrotor, [Online]. Available: https://pt.wikipedia.org/wiki/Quadrotor. [Acesso em 11 Novembro 2016].

[8] D. F. QUADCOPTER, "Syma X5SW FPV Real-Time WiFi Quadcopter First Look," 02 Novembro 2016. [Online]. Available: http://www.quadcopterflyers.com/2015/04/syma-x5sw-fpvreal-time-wifi-quadcopter.html.

[9] Comphaus, "Comphaus - Tecnologia da nova geração," Comphaus, [Online]. Available: http://comphaus.com.br/home/?wpsc-product=arduino-uno-r3. [Acesso em 14 Outubro 2016].

[10] ARDUINO, "Arduino UNO: Overview," 02 Novembro 2016. [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno.

[11] "Como funciona um Potenciômetro," Como funciona um Potenciômetro, 27 Maio 2013. [Online]. Available: http://eletronicaemcasa.blogspot.com.br/2013/05/como-funciona-umpotenciometro.html. [Acesso em 06 Junho 2016].

[12] Arduino, "Arduino UNO: PWN," [Online]. Available: https://www.arduino.cc/en/Main/Tutorial/PWN. [Acesso em 06 Junho 2016].

[13] "Eletronic Color Code," [Online]. Available: https://en.wikipedia.org/wiki/Electronic_color_code. [Acesso em 04 Janeiro 2017].

[14] Filipeflop, “Filipeflop,” Filipeflop, [Online]. Available: http://www.filipeflop.com/pd-b4742-modulo-bluetooth-rs232-hc-05.html. [Acesso em 21 Junho 2017].

[15] BR-Arduino, “BR-Arduino,” BR-Arduino, [Online]. Available: http://br-arduino.org/2015/02/arduino-pro-mini-conhecendo-o-modelo-economico.html. [Acesso em 10 Julho 2017].

[16] Embarcados, “Embarcados,” [Online]. Available: https://www.embarcados.com.br/ftdi-ft232r/. [Acesso em 10 Julho 2017].

[17] R. Zelenovsky e A. Mendonça, Apêndice H - MPU6050, Brasília: UnB, 2016.

[18] Arduino, “Wire Library,” Arduino, [Online]. Available: https://www.arduino.cc/en/Reference/Wire. [Acesso em 4 Julho 2017].

[19] SFUptownMaker, “I2C Protocol,” Sparkfun, [Online]. Available: https://learn.sparkfun.com/tutorials/i2c. [Acesso em 4 Julho 2017].

[20] EPC, “What IS an encoder?,” Encoder Products Company, 25 Fevereiro 2016. [Online]. Available: http://encoder.com/blog/company-news/what-is-an-encoder/. [Acesso em 4 Julho 2017].

[21] “EBAH,” [Online]. Available: http://www.ebah.com.br/content/ABAAAesEcAJ/encoder-absoluto-incremental. [Acesso em 24 Setembro 2016].

[22] “Quora,” [Online]. Available: https://www.quora.com/How-does-a-Schmitt-trigger-circuit-work. [Acesso em 07 Novembro 2016].

[23] ANDROID, 02 Novembro 2016. [Online]. Available: https://source.android.com.

[24] F. CORDEIRO, “Android Aprendiz: Um guia para iniciantes,” 02 Novembro 2016. [Online]. Available: http://www.brodowski.sp.gov.br/novo/wpcontent/uploads/2016/04/Android_Aprendiz_Novo.pdf.

[25] A. Studio, “Wikipedia Enciclopédia Virtual,” 02 Novembro 2016. [Online]. Available: https://pt.wikipedia.org/wiki/Android_Studio.

[26] J. D. Irwin e R. M. Nelms, Análise Básia de Circuitos para Engenharia, Rio de Janeiro: LTC, 2010.

[27] H. S. B. Products, “User Instructional Manual,” [Online]. Available: http://www.rcscomponents.kiev.ua/datasheets/hc_hc-05-user-instructionsbluetooth.pdf.. [Acesso em 2016 Junho 20].

ANEXOS

Anexo 1 – Diagrama de pinos do Arduino Uno.

Anexo 2 – Diagrama de pinos do Arduino Pro Mini.

Anexo 3 – Circuito eletrônico da placa antiga.

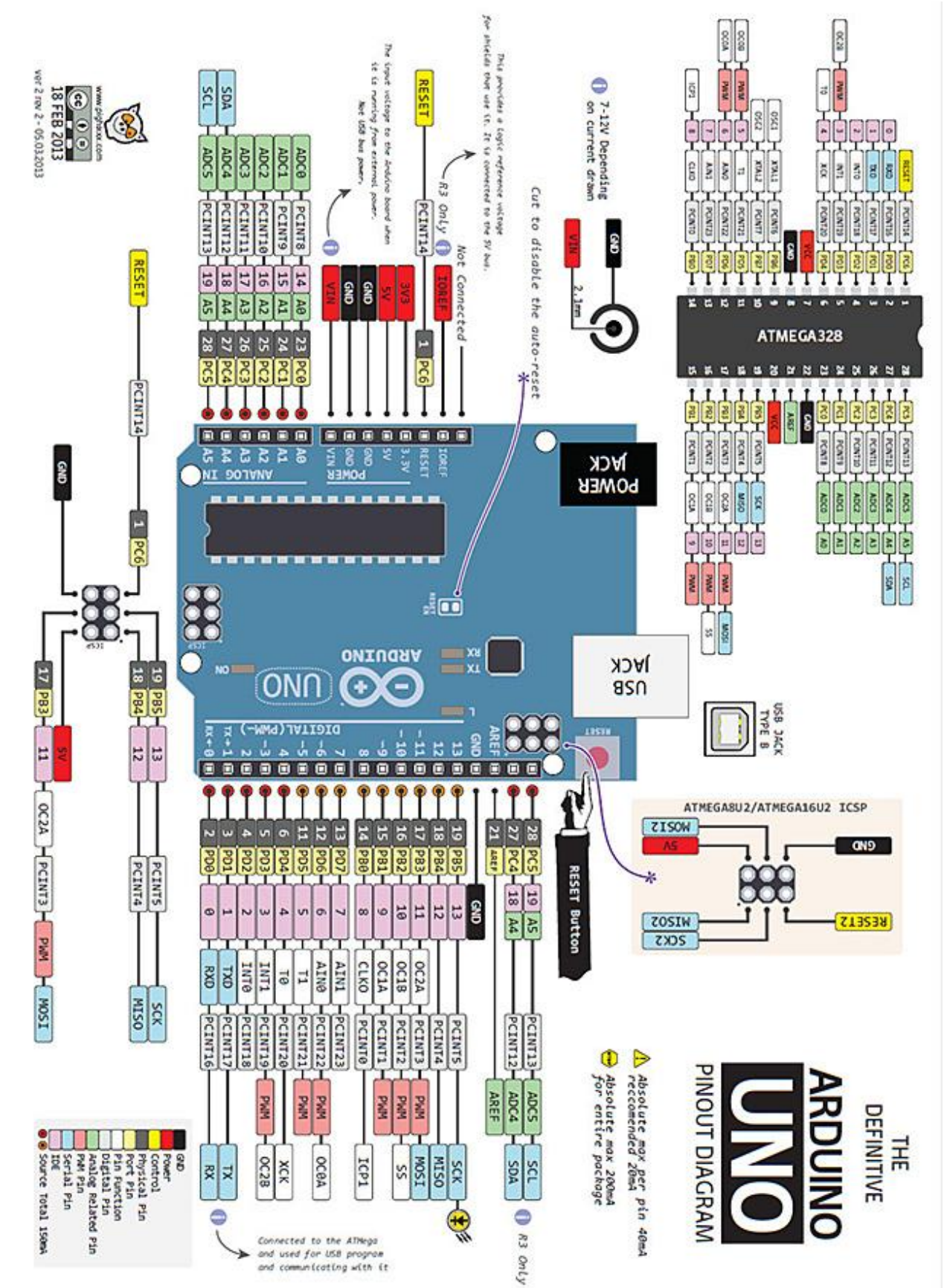
Anexo 4 – Montagem para ensaios de rotação.

Anexo 5.1 – Desenho técnico da estrutura de acrílico – Vista frontal.

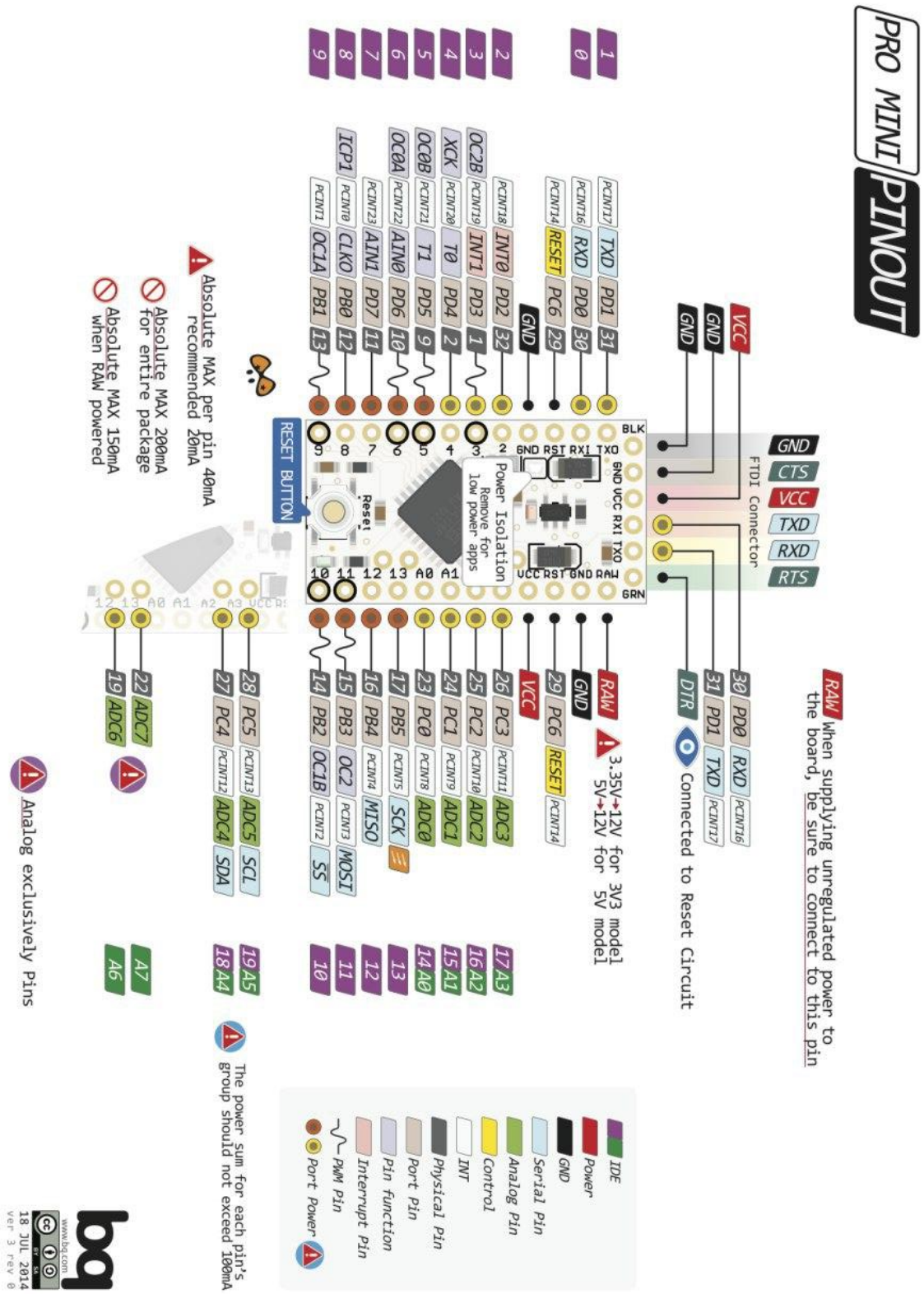
Anexo 5.2 – Desenho técnico da estrutura de acrílico – Vista superior.

Anexo 5.3 – Desenho técnico da estrutura de acrílico – Vista lateral.

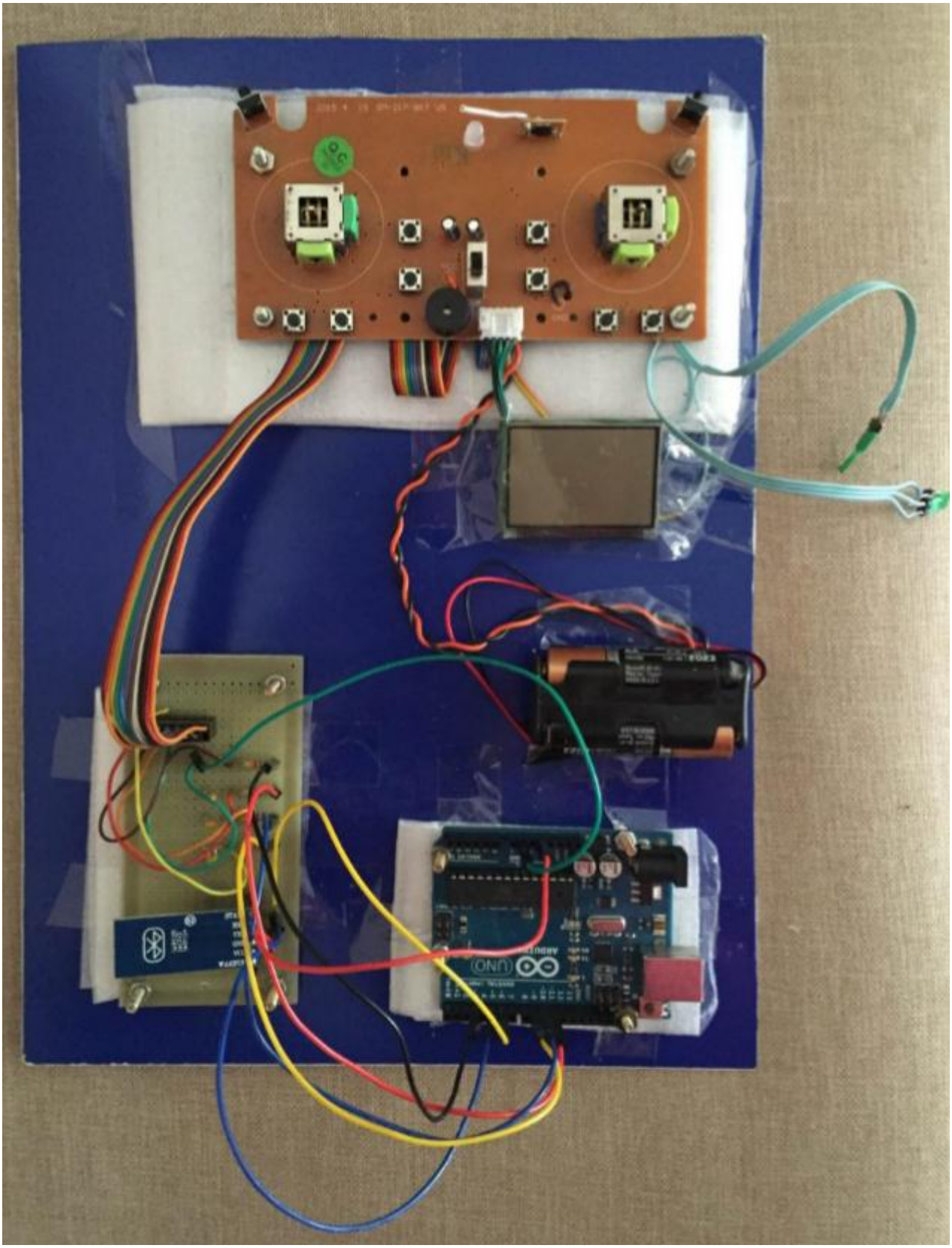
ANEXO 1: Diagrama de pinos do Arduino Uno.



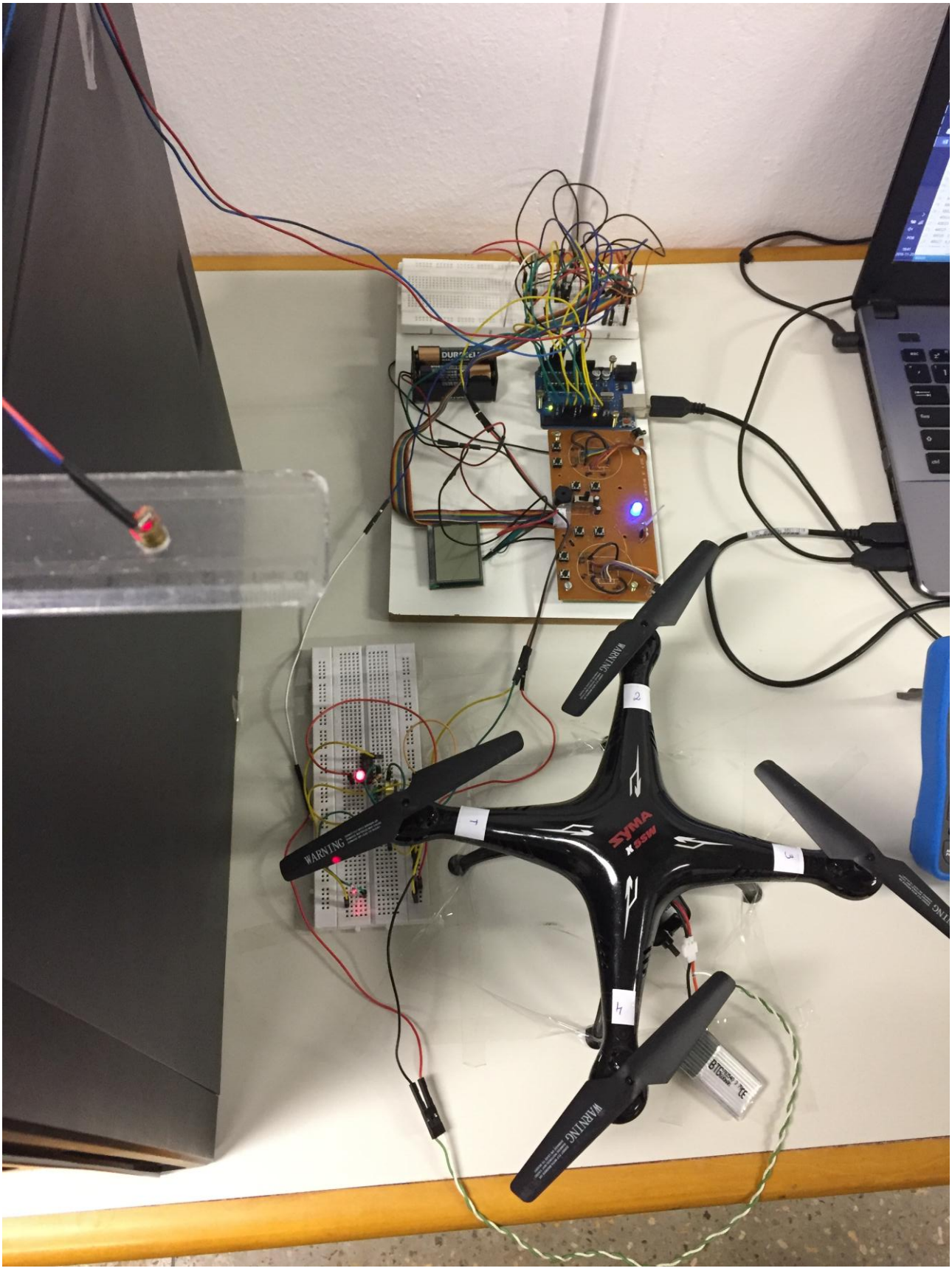
ANEXO 2: Diagrama de pinos do Arduino Pro Mini.



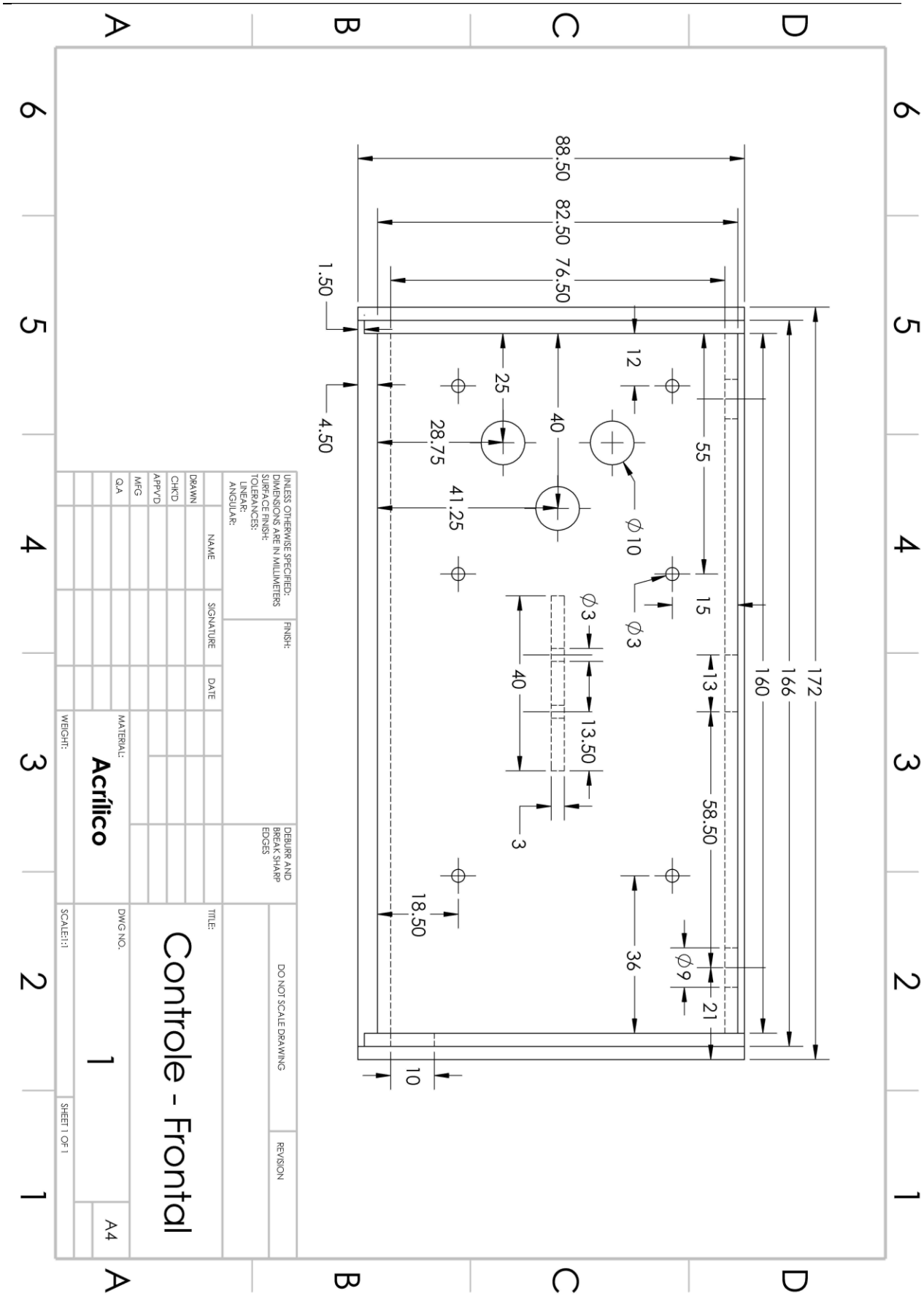
ANEXO 3: Circuito eletrônico da placa antiga.



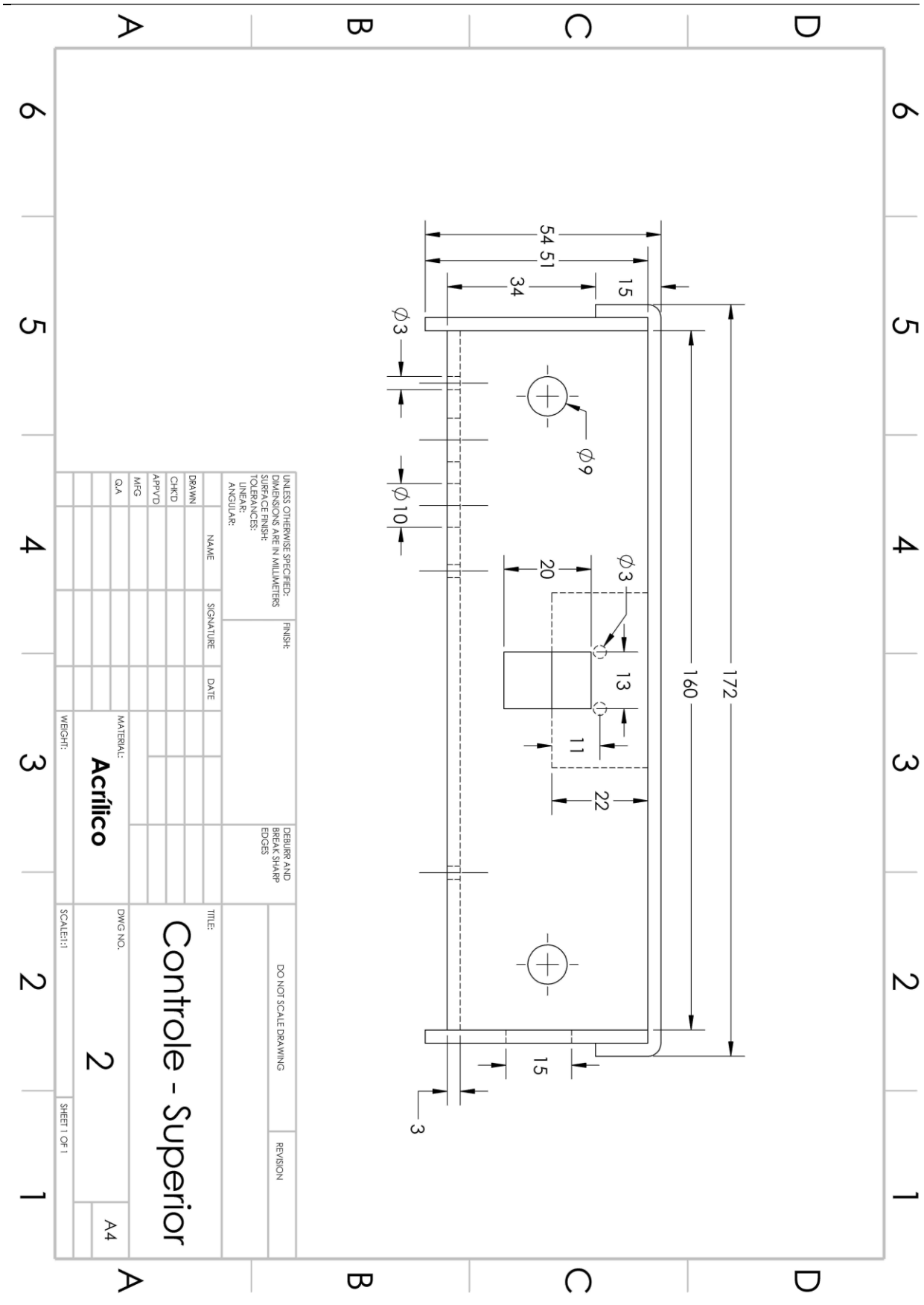
ANEXO 4: Montagem para ensaios de rotação.



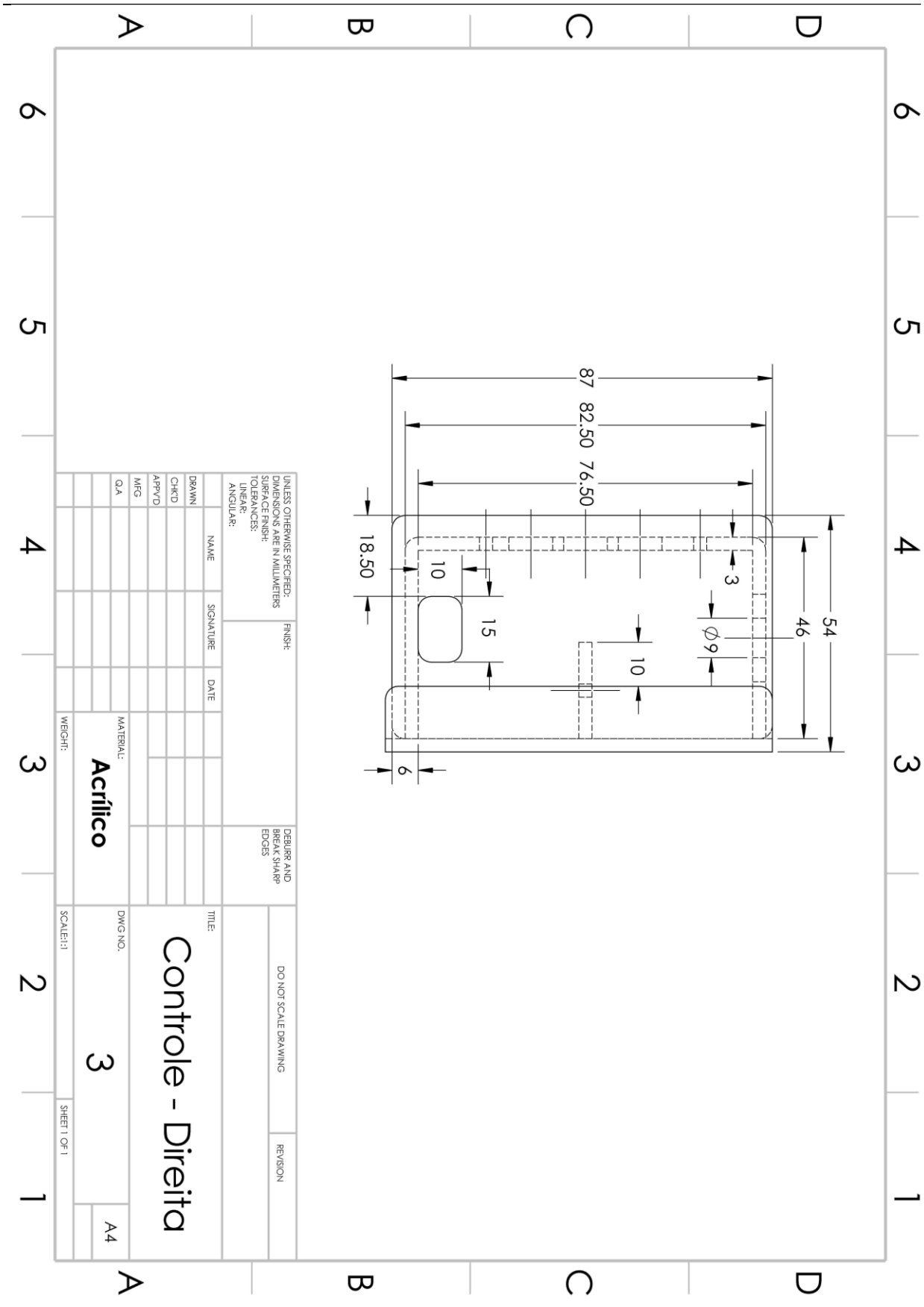
ANEXO 5.1: Desenho técnico da estrutura de acrílico – Vista frontal.



ANEXO 5.2: Desenho técnico da estrutura de acrílico – Vista superior.



ANEXO 5.3: Desenho técnico da estrutura de acrílico – Vista lateral.



APÊNDICES

Apêndice 1 – Código utilizado para o teste de rotação das hélices.

Apêndice 2 – Código que gerencia todo o sistema de atuação.

Apêndice 3 – Código que gerencia todo o sistema de controle.

Apêndice 4 – VANT respondendo à inclinação para a direita do controle.

Apêndice 5 – VANT respondendo à inclinação para a esquerda do controle.

Apêndice 6 – VANT respondendo à inclinação para frente do controle.

Apêndice 7 – VANT respondendo à inclinação para trás do controle.

Apêndice 8 – VANT respondendo à rotação para a direita do controle.

Apêndice 9 – VANT respondendo à rotação para a esquerda do controle.

Apêndice 10 – VANT respondendo ao botão direito do controle.

Apêndice 11 – VANT respondendo ao botão esquerdo do controle.

APÊNDICE 1

```
#define JEVd 8
#define JEVu 9
#define JEHd 10
#define JEHu 11
#define JDVd 4
#define JDVu 5
#define JDHd 6
#define JDHu 7
#define DELAY 40

int contador = 0;
int valorJEV;
int valorJEH;
int valorJDV;
int valorJDH;

float voltsJEV;
float voltsJEH;
float voltsJDV;
float voltsJDH;

char msg[100];

void setup() {
  int i = 0;
  Serial.begin(9600);

  for (i = 4; i < 12; i++) {
    pinMode(i, OUTPUT);
    digitalWrite(i, HIGH);
  }

  attachInterrupt(
digitalPinToInterrupt(2),int_laser,
RISING);

  Serial.println("Ligar DRONE");
  delay(2000);

  Serial.println("Sobe todos");
  //A CADA ITERAÇÃO APLICA UM PULSO NO
UP DE CADA POTENCIOMETRO
  //ao fim das 32 iterações todos os
potenciometros estão no máximo
  for (i = 0; i < 32; i++) {
    digitalWrite(JEVu, LOW);
    digitalWrite(JEHu, LOW);
    digitalWrite(JDVu, LOW);
    digitalWrite(JDHu, LOW);
    delay(DELAY);
    digitalWrite(JEVu, HIGH);
    digitalWrite(JEHu, HIGH);
    digitalWrite(JDVu, HIGH);
    digitalWrite(JDHu, HIGH);
    delay(DELAY);
    printpot();
  }

  Serial.println("Desce todos");
  //A CADA ITERAÇÃO APLICA UM PULSO NO
DOWN DE CADA POTENCIOMETRO
  //ao fim das 32 iterações JEV está no
seu mínimo (máximo logico)
  //JEH, JDV e JDH estão calibrados em
seus pontos intermediários
  for (i = 0; i < 32; i++) {
    digitalWrite(JEVd, LOW);
    if (i < 15) {
      digitalWrite(JEHd, LOW);
      digitalWrite(JDVd, LOW);
      digitalWrite(JDHd, LOW);
    }
    delay(DELAY);
    digitalWrite(JEVd, HIGH);
    if (i < 15) {
      digitalWrite(JEHd, HIGH);
      digitalWrite(JDVd, HIGH);
      digitalWrite(JDHd, HIGH);
    }
    delay(DELAY);
    printpot();
  }

  //JEV para o máximo (mínimo lógico)
  Serial.println("Calibra JEV");
  for (i = 0; i < 32; i++) {
    digitalWrite(JEVu, LOW);
    delay(DELAY);
    digitalWrite(JEVu, HIGH);
    delay(DELAY);
    printpot();
  }
}

void loop() {
int i = 0;

  //JEV para o mínimo (máximo lógico)
  for (i = 0; i < 32; i++) {
    digitalWrite(JEVd, LOW);
    delay(DELAY);
    digitalWrite(JEVd, HIGH);
    delay(DELAY);
    contador = 0;
    delay(2000);
    printjevrpm();
  }

  //Finaliza a execução
  for (i = 0; i < 32; i++) {
    digitalWrite(JEVu, LOW);
    delay(DELAY);
    digitalWrite(JEVu, HIGH);
    delay(DELAY);
  }
  while (1);
}
```

```
// Imprime em Volts o valor de todos os potenciômetros
void printpot() {
  valorJEV = analogRead(0);
  valorJEH = analogRead(1);
  valorJDV = analogRead(2);
  valorJDH = analogRead(3);

  voltsJEV = (5/1024.)*valorJEV;
  voltsJEH = (5/1024.)*valorJEH;
  voltsJDV = (5/1024.)*valorJDV;
  voltsJDH = (5/1024.)*valorJDH;

  Serial.print("JEV: ");
  Serial.print(voltsJEV);
  Serial.print(" V");
  Serial.print("  -  ");

  Serial.print("JEH: ");
  Serial.print(voltsJEH);
  Serial.print(" V");
  Serial.print("  -  ");

  Serial.print("JDV: ");
  Serial.print(voltsJDV);
  Serial.print(" V");
  Serial.print("  -  ");

  Serial.print("JDH: ");
  Serial.print(voltsJDH);
  Serial.println(" V");
}

//Imprime a tensão em volts e a rotação em RPM
void printjevrpm() {
  valorJEV = analogRead(0);
  voltsJEV = (5/1024.)*valorJEV;

  Serial.print(voltsJEV);
  Serial.print(" ; ");
  sprintf(msg, "%d", contador * 15);
  Serial.println(msg);
}

void int_laser () {
  contador++;
}
```

APÊNDICE 2

```
////////////////////////////////////////////////////////////
/// Universidade de Brasilia - Faculdade de Tecnologia     ///
/// Programa: Software de atuacao - Trabalho de Graduacao  ///
/// Alunos: Cássio Pantoja e Paulo Garcia                  ///
/// Orientador: Ricardo Zelenovsky                         ///
////////////////////////////////////////////////////////////
/// Programa que gerencia todo o sistema de atuacao        ///
////////////////////////////////////////////////////////////

#define JEVd 8
#define JEVu 9
#define JEHd A4
#define JEHu A5
#define JDVd 4
#define JDVu 5
#define JDHd 6
#define JDHu 7
#define DELAY 40

#define BluetoothRX 13
#define BluetoothTX 12

#include <SoftwareSerial.h>

SoftwareSerial Bluetooth(BluetoothTX, BluetoothRX);

volatile int fjevu, fjevd, fjehu, fjehd, fjdvu, fjdvd, fjdhu, fjdhd;

bool started = false;
bool ended = false;

char aux[5];
char buf[25];
char auxbuf;
char msg[37];

long looptimer;
long pottimer;

int count = 0;
int contador = 0;

int valorJEV;
int valorJEH;
int valorJDV;
int valorJDH;

int JEVatual = 0;
int JEHatual = 0;
int JDVatual = 0;
int JDHatual = 0;

int JEVlido;
int JEHlido;
int JDVlido;
int JDHlido;

int JEVdesejado;
int JEHdesejado;
```

```
int JDVdesejado;
int JDHdesejado;

int flagJEV = 0;
int flagJEH = 0;
int flagJDV = 0;
int flagJDH = 0;

int i = 0;
int j = 0;

float voltsJEV;
float voltsJEH;
float voltsJDV;
float voltsJDH;

void setup() {

  Serial.begin(9600);
  Bluetooth.begin(38400);

  pinMode(JEVu, OUTPUT);
  pinMode(JEVd, OUTPUT);
  pinMode(JEHu, OUTPUT);
  pinMode(JEHd, OUTPUT);
  pinMode(JDVu, OUTPUT);
  pinMode(JDVd, OUTPUT);
  pinMode(JDHu, OUTPUT);
  pinMode(JDHd, OUTPUT);
  digitalWrite(JEVu, HIGH);
  digitalWrite(JEVd, HIGH);
  digitalWrite(JEHu, HIGH);
  digitalWrite(JEHd, HIGH);
  digitalWrite(JDVu, HIGH);
  digitalWrite(JDVd, HIGH);
  digitalWrite(JDHu, HIGH);
  digitalWrite(JDHd, HIGH);

  Serial.println("Ligar DRONE");

  delay (2000);

  Serial.println("Desce todos");
  //A CADA ITERAÇÃO APLICA UM PULSO NO DOWN DE CADA POTENCIOMETRO
  //ao fim das 32 iterações todos os potenciometros estão no mínimo.
  for (i = 0; i < 32; i++) {
    desceJEV();
    printpot();
  }

  JEVatual = 0;

  fjevu = 0;
  fjevd = 0;
  fjehu = 0;
  fjehd = 0;
  fjdvu = 0;
  fjdvd = 0;
  fjdhu = 0;
  fjdhd = 0;
```

```
  Serial.println("Calibra e sincroniza");

  //ao fim das 31 iterações JEV está no seu mínimo (máximo logico)
  //JEH, JDV e JDH estão em seus pontos intermediários
  for (i = 0; i < 31; i++) {
    sobeJEV();
    JEVatual++;
    printpot();
  }

  JEHatual = 15;
  JDVatual = 15;
  JDHatual = 15;

  JEVlido = JEVatual;
  JEHlido = JEHatual;
  JDVlido = JDVatual;
  JDHlido = JDHatual;

  cli();            //Int: Desabilitação geral
  TCCR1A = 0;       //WGM11=WGM10=0
  // Modo CTC e CLK/64
  TCCR1B = (1 << WGM12) | (1 << CS11) | (1 << CS10);
  TIMSK1 = (1 << OCIE1B) | (1 << OCIE1A); //hab. interrupção por comparação

  OCR1A = 249;  OCR1A = 10249;   // 1ms  e 41ms
  //OCR1A = 2499; OCR1A = 12499;   // 10ms   50ms
  //OCR1A = 4999; OCR1A = 14999;   // 20ms e 60ms
  //OCR1A = 7499; OCR1A = 17499;   // 30ms e 70ms
  //OCR1B = 9999; OCR1A = 19999;     // 40ms e 80ms

  sei();            //Int: Habilitação geral

}

void loop() {
  //rotina de montagem das mensagens recebidas
  if (Bluetooth.available()) {
    auxbuf = (char)Bluetooth.read();
    if (auxbuf == '#') {
      i = 0;
      buf[i] = auxbuf;
      started = true;
      ended = false;
      i++;
    }
    else if (auxbuf == ';') {
      buf[i] = auxbuf;
      ended = true;
      buf[i + 1] = '\0';
      //return;
    } else {
      buf[i] = auxbuf;
      i++;
    }
  }
```

```
  // Verifica se as mensagens sao montadas corretamente no array ao fim da mensagem
  // (mostra em tela)
  if (ended) {
    for (j = 0; j < i; j++) {
      Serial.print(buf[j]);
    }
    Serial.print('\n');

    sprintf(msg, "@%02d.%02d.%02d.%02d", JEVatual, JDVatual, JDHatual, JEHatual);
    Serial.println(msg);

    printpot();
  }

  //se a mensagem foi terminada, armazena-se as partes de acordo
  if (started && ended) {

    aux[0] = buf[1];
    aux[1] = buf[2];
    aux[2] = '\0';
    JEVlido = atoi(aux);

    aux[0] = buf[4];
    aux[1] = buf[5];
    aux[2] = '\0';
    JDVlido = atoi(aux);

    aux[0] = buf[7];
    aux[1] = buf[8];
    aux[2] = '\0';
    JDHlido = atoi(aux);

    aux[0] = buf[10];
    aux[1] = buf[11];
    aux[2] = '\0';
    JEHlido = atoi(aux);
  }
  JEVdesejado = JEVlido;
  JEHdesejado = JEHlido;
  JDVdesejado = JDVlido;
  JDHdesejado = JDHlido;

}

// Imprime em Volts o valor de todos os potenciômetros
void printpot() {

  valorJEV = analogRead(0);
  valorJEH = analogRead(1);
  valorJDV = analogRead(2);
  valorJDH = analogRead(3);

  voltsJEV = (5 / 1024.) * valorJEV;
  voltsJEH = (5 / 1024.) * valorJEH;
  voltsJDV = (5 / 1024.) * valorJDV;
  voltsJDH = (5 / 1024.) * valorJDH;

  Serial.print("JEV: ");
  Serial.print(voltsJEV);
  Serial.print(" V");
```

```
  Serial.print("  -  ");
  Serial.print("JEH: ");
  Serial.print(voltsJEH);
  Serial.print(" V");
  Serial.print("  -  ");

  Serial.print("JDV: ");
  Serial.print(voltsJDV);
  Serial.print(" V");
  Serial.print("  -  ");

  Serial.print("JDH: ");
  Serial.print(voltsJDH);
  Serial.println(" V");

}

void sobeJEV() {
  digitalWrite(JEVu, LOW);
  delay(DELAY);
  digitalWrite(JEVu, HIGH);
}

void desceJEV() {
  digitalWrite(JEVd, LOW);
  delay(DELAY);
  digitalWrite(JEVd, HIGH);
}

void sobeJEH() {
  digitalWrite(JEHu, LOW);
  delay(DELAY);
  digitalWrite(JEHu, HIGH);
}
void desceJEH() {
  digitalWrite(JEHd, LOW);
  delay(DELAY);
  digitalWrite(JEHd, HIGH);
}

void sobeJDV() {
  digitalWrite(JDVu, LOW);
  delay(DELAY);
  digitalWrite(JDVu, HIGH);
}
void desceJDV() {
  digitalWrite(JDVd, LOW);
  delay(DELAY);
  digitalWrite(JDVd, HIGH);
}

void sobeJDH() {
  digitalWrite(JDHu, LOW);
  delay(DELAY);
  digitalWrite(JDHu, HIGH);
}
void desceJDH() {
  digitalWrite(JDHd, LOW);
  delay(DELAY);
  digitalWrite(JDHd, HIGH);
}
```

```
//interrupcao B
ISR(TIMER1_COMPB_vect) {

  if (JEVdesejado > JEVatual) {
    digitalWrite(JEVu, LOW);
    fjevu = 1;
  } else if (JEVdesejado < JEVatual) {
    digitalWrite(JEVd, LOW);
    fjevd = 1;
  }

  if (JEHdesejado > JEHatual) {
    digitalWrite(JEHu, LOW);
    fjehu = 1;
  } else if (JEHdesejado < JEHatual) {
    digitalWrite(JEHd, LOW);
    fjehd = 1;
  }

  if (JDVdesejado > JDVatual) {
    digitalWrite(JDVu, LOW);
    fjdvu = 1;
  } else if (JDVdesejado < JDVatual) {
    digitalWrite(JDVd, LOW);
    fjdvd = 1;
  }

  if (JDHdesejado > JDHatual) {
    digitalWrite(JDHu, LOW);
    fjdhu = 1;
  } else if (JDHdesejado < JDHatual) {
    digitalWrite(JDHd, LOW);
    fjdhd = 1;
  }

}

//interrupcao A
ISR(TIMER1_COMPA_vect) {

  if ( fjevu) {
    digitalWrite(JEVu, HIGH);
    fjevu = 0;
    JEVatual++;
  } else if ( fjevd) {
    digitalWrite(JEVd, HIGH);
    fjevd = 0;
    JEVatual--;
  }

  if ( fjehu) {
    digitalWrite(JEHu, HIGH);
    fjehu = 0;
    JEHatual++;
  } else if ( fjehd) {
    digitalWrite(JEHd, HIGH);
    fjehd = 0;
    JEHatual--;
  }
```

```
  if ( fjdvu) {
    digitalWrite(JDVu, HIGH);
    fjdvu = 0;
    JDVatual++;
  } else if ( fjdvd) {
    digitalWrite(JDVd, HIGH);
    fjdvd = 0;
    JDVatual--;
  }

  if ( fjdhu) {
    digitalWrite(JDHu, HIGH);
    fjdhu = 0;
    JDHatual++;
  } else if ( fjdhd) {
    digitalWrite(JDHd, HIGH);
    fjdhd = 0;
    JDHatual--;
  }

}
```

APÊNDICE 3

```
////////////////////////////////////////////////////////////
/// Universidade de Brasilia - Faculdade de Tecnologia     ///
/// Programa: Software de controle - Trabalho de Graduacao ///
/// Alunos: Cássio Pantoja e Paulo Garcia                  ///
/// Orientador: Ricardo Zelenovsky                         ///
/// Baseado no programa de FILIPEFLOP e Joop Brokking      ///
////////////////////////////////////////////////////////////
/// Programa que gerencia todo o sistema de controle       ///
////////////////////////////////////////////////////////////

#define pinDesce           2
#define pinSobe            4
#define pinEsquerda        5
#define pinDireita         6
#define pinEmergencia      3
#define pinConfigBluetooth 10
#define pinLed             13
#define BluetoothRX        12
#define BluetoothTX        11

#define CONFIGBLUETOOTH LOW

#define DIVISORYAW 40          // quanto menor, mais intenso (valor entre 1 e 100)
#define FATOR_DE_ZERAMENTO_YAW 0.995    // quanto menor, menos dura o giro (valor
entre 0 e 1)

#include <Wire.h>
#include <SoftwareSerial.h>

SoftwareSerial Bluetooth(BluetoothTX, BluetoothRX);

//declaracao de variaveis
long ax, ay, az, tmp, gx, gy, gz;
long gxcal, gycal, gzcal;
long looptimer;
long avetor;
float roll, pitch, yaw;
float aroll, apitch, ayaw;
int jev = 6;
int i;
int startflag;
int count = 0;
int jevupantigo = 0;
int jevup = 0;
int jevdownantigo = 0;
int jevdown = 0;
int jehright = 0;
int jehleft = 0;
int emergencia = 0;
int emergaux;
char msg[13];

void setup() {

  Serial.begin(9600);
  Bluetooth.begin(38400);
  Wire.begin();
```

```
  pinMode(pinDesce, INPUT_PULLUP);
  pinMode(pinSobe, INPUT_PULLUP);
  pinMode(pinEsquerda, INPUT_PULLUP);
  pinMode(pinDireita, INPUT_PULLUP);
  pinMode(pinEmergencia, INPUT_PULLUP);
  pinMode(pinConfigBluetooth, OUTPUT);
  pinMode(pinLed, OUTPUT);

  digitalWrite(pinConfigBluetooth, CONFIGBLUETOOTH);

  imusetup();

  //calibra giroscópio
  //Bluetooth.print("Calibrando giroscopio");
  Serial.print("Calibrando giroscopio");
  for (i = 0; i < 2000 ; i++) {                //Calibração: 2.000 leituras
    if (i % 125 == 0) Serial.print(".");
    //A cada 166 iterações (aproximadamente 0,5s) o LED acedente (LOW -> HIGH)
    if (i % 166 == 0) digitalWrite(pinLed, HIGH);
    //E em seu contra-tempo, o led apaga (HIGH -> LOW)
    if (i % 166 == 83) digitalWrite(pinLed, LOW);
    if (i == 1999)  Serial.println(".");

    imuread();

    gxcal += gx;
    gycal += gy;      //Soma-se todas as 2.000 leituras.
    gzcal += gz;

    delay(3);
  }

  gxcal /= 2000;
  gycal /= 2000;      //Por fim divide por 2.000 (média aritmética)
  gzcal /= 2000;

  roll = 0;
  pitch = 0;
  yaw = 0;

  digitalWrite(pinLed, HIGH); //Mantém o LED aceso ao fim do Setup.

  looptimer = micros(); //Inicia o cronômetro
}

void loop()
{
  imuread();    //Rotina que solicita e lê os dados da IMU.

  gx -= gxcal;
  gy -= gycal;  //Corrige-se os valores lidos com os valores da calibração
  gz -= gzcal;

  //0.0000611 = 1 / (250Hz / 65.5)
  pitch += gx * 0.000611;
  roll  += gy * 0.000611;
  yaw   += gz * 0.000611;

  //0.000001066 = 0.0000611 * (3.142(PI) / 180degr).
  //sin(); do arduino é em radianos.
  pitch += roll * sin(gz * 0.00001066);
```

```
//corrige-se o pitch e o roll quando se ocorrem rotações sucessivas.
roll  -= pitch * sin(gz * 0.00001066);


//Calculo pelos acelerometros
avetor = sqrt((ax*ax) + (ay*ay) + (az*az)); //Calcula o vetor total de aceleração
//57.296 = 1 / (3.142 / 180) pois asin(); também é em radianos
apitch = asin((float)ay / avetor) * 57.296;
aroll = asin((float)ax / avetor) * -57.296;

//correcao de calibracao do acelerometro
apitch -= -0.21;
aroll  -= -2.34


if (startflag) {
  //leitura combinada pelo giroscopio e acelerometro
  pitch = pitch * 0.9 + apitch * 0.1;
  roll = roll   * 0.9 + aroll  * 0.1;
  yaw = yaw * FATOR_DE_ZERAMENTO_YAW;
}
else {
  //isso faz com que a IMU possa ser ligada em qualquer posicao inicial
  pitch = apitch;
  roll = aroll;
  startflag = true;
}

//confere os botoes
jevdown  = digitalRead(pinDesce);
jevup    = digitalRead(pinSobe);
jehleft  = digitalRead(pinEsquerda);
jehright = digitalRead(pinDireita);

//comparacoes para se evitar que uma pressionada resulte em multiplas alteracoes
if (jevdownantigo == HIGH && jevdown == LOW && jev < 6) {
  jev++;
}
if (jevupantigo == HIGH && jevup == LOW && jev > 0) {
  jev--;
}
jevdownantigo = jevdown;
jevupantigo   = jevup;

//soma o contador de iteracoes
count++;
if (count > 49) {
  imuprint();   //e envia apos 50 loops
  count = 0;
}


//rotina que verifica o botão de emergencia e envia mensagem de desligamento
if (!digitalRead(pinEmergencia))
  while (1) {
    sprintf(msg, "#31.15.15.15;");
    Serial.print(msg);
    Serial.println(".emergencia");
    Bluetooth.print(msg);
    delay (500);
  }
```

```
  //espera o tempo do laço atingir o marco de 4ms
  while (micros() - looptimer < 4000);
  looptimer = micros(); //reinicia-se o cronometro
}

void imusetup() {
  //Acorda a IMU
  Wire.beginTransmission(0x68);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission();

  //Configura o acelerometro
  Wire.beginTransmission(0x68);
  Wire.write(0x1C);
  Wire.write(0x10);
  Wire.endTransmission();

  //Configura o giroscopio
  Wire.beginTransmission(0x68);
  Wire.write(0x1B);
  Wire.write(0x08);
  Wire.endTransmission();
}

void imuread() {
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission();
  Wire.requestFrom(0x68, 14);   //solicita-se 14 bytes
  while (Wire.available() < 14);//espera a disponibilidade dos 14 bytes
  ax  = Wire.read() << 8 | Wire.read();
  ay  = Wire.read() << 8 | Wire.read();
  az  = Wire.read() << 8 | Wire.read();
  tmp = Wire.read() << 8 | Wire.read();    //le-se todos, unindo as partes HIGH e
LOW
  gx  = Wire.read() << 8 | Wire.read();
  gy  = Wire.read() << 8 | Wire.read();
  gz  = Wire.read() << 8 | Wire.read();
}

void imuprint() {

  int aux;
  int auxjev;
  int auxpitch;
  int auxroll;
  int auxyaw;
  int botaoyaw = 0;

  //tratamento de JEV
  switch (jev) {
    case 6:
      auxjev = 31;
      break;
    case 5:
      auxjev = 29;
      break;
    case 4:
      auxjev = 26;
```

```
      break;
    case 3:
      auxjev = 22;
      break;
    case 2:
      auxjev = 17;
      break;
    case 1:
      auxjev = 12;
      break;
    case 0:
      auxjev = 0;
      break;
  }

  //tratamento de pitch
  if (pitch > 60) {
    auxpitch = 31;
  } else if (pitch < -60) {
    auxpitch = 0;
  } else {
    auxpitch = ( ( ( (int)pitch ) + 60 ) / 4 );
  }

  //tratamento de roll
  if (roll > 60) {
    auxroll = 0;
  } else if (roll < -60) {
    auxroll = 31;
  } else {
    auxroll = ( ( ( -(int)roll ) + 60 ) / 4 );
  }

  //tratamento de yaw
  auxyaw = 15 - ((int)yaw / DIVISORYAW);
  if (jehright == LOW) botaoyaw += 5;
  if (jehleft  == LOW) botaoyaw -= 5;
  auxyaw += botaoyaw;
  if (auxyaw > 31) {
    auxyaw = 31;
  } else if (auxyaw < 0) {
    auxyaw = 0;
  }

  //correcao por conta do reposicionamento do acelerometro
  //(troca roll com pitch)
  aux = auxroll;
  auxroll = auxpitch;
  auxpitch = aux;

  //envia mensagem
  sprintf(msg, "#%02d.%02d.%02d.%02d;", auxjev, auxpitch, auxroll, auxyaw);
  Serial.println(msg);
  Bluetooth.print(msg);
}
```

APÊNDICE 4

Inclinação para a direita:

Endereço:

https://www.youtube.com/watch?v=-hjdWAiqKWU

APÊNDICE 5

Inclinação para a esquerda:

Endereço:

https://www.youtube.com/watch?v=TsgH9kuPQhU

APÊNDICE 6

Inclinação para frente:

Endereço:

https://www.youtube.com/watch?v=0BUiYT2zZC0

APÊNDICE 7

Inclinação para trás:

Endereço:

https://www.youtube.com/watch?v=0KXFLNq9RlM

APÊNDICE 8

Rotação para a direita:

Endereço:

https://www.youtube.com/watch?v=1AQ5J4hmzg0

APÊNDICE 9

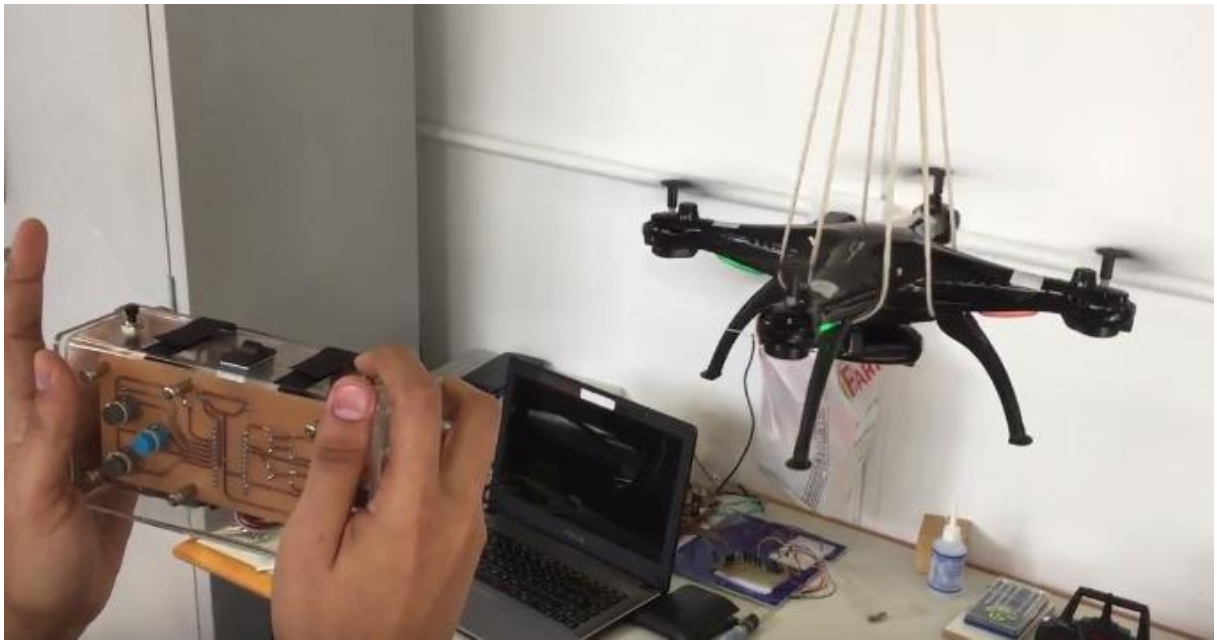
Rotação para a esquerda:

Endereço:

https://www.youtube.com/watch?v=47tkbrg1Ok0

APÊNDICE 10

Botão direito:

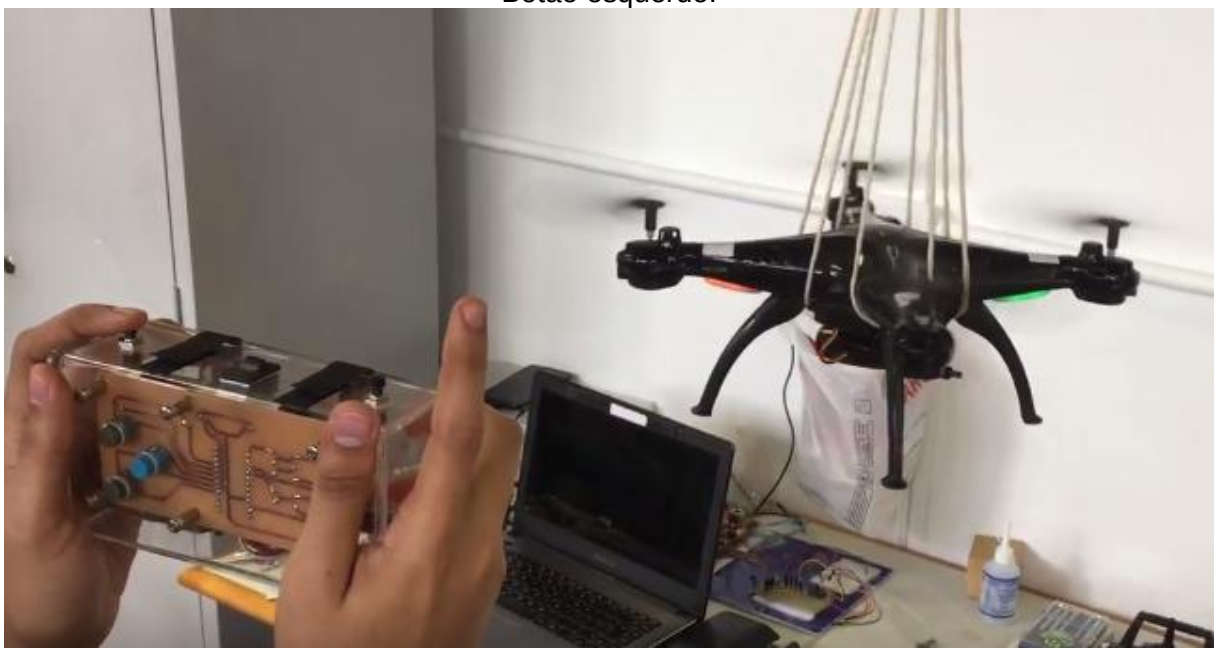


Endereço:

https://www.youtube.com/watch?v=ELrjkb8pGzM

APÊNDICE 11

Botão esquerdo:



Endereço:

https://www.youtube.com/watch?v=xzuxxLpMuEw