COLLISION RESILIENT FOLDABLE MICRO AERIAL ROBOT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

By
Levent Dilaveroğlu
September 2019

COLLISION RESILIENT FOLDABLE MICRO AERIAL ROBOT
By Levent Dilaveroğlu
September 2019

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Onur Özcan(Advisor)

Yıldıray Yıldız

Özgur Ünver

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

COLLISION RESILIENT FOLDABLE MICRO AERIAL ROBOT

Levent Dilaveroğlu
M.S. in Mechanical Engineering
Advisor: Onur Özcan
September 2019

Collision management strategies are integral part of micro air vehicles for the reliability of their operation. Collision avoidance strategies require enhanced environmental and situational awareness for generating evasive maneuver trajectories. Simpler and more adaptable option is to prepare for collisions and design the physical frame around predicted collision patterns. In this work, a mechanically compliant frame design collaborating origami-inspired foldable robotics methods with protective shock absorbing or guiding elements has been proposed for a collision resilient quad-rotor UAV. General workings and mathematical model of quadrotor has been explained to inform the reader further about the quadrotor mechanics. 2D design of the foldable structure and the manufacturing process, including electronic hardware elements and software has been discussed. Control scheme, communication and operation is explained in detail to be an informative guideline for the future air vehicle projects of the Bilkent Miniature Robotics Lab.

Keywords: Robotics, UAV, foldable robotics, micro aerial vehicles.

ÖZET

ÇARPIŞMA DİRENÇLİ KATLANABİLİR MİKRO HAVA ARACI

Levent Dilaveroğlu
Makine Mühendisliği, Yüksek Lisans
Tez Danışmanı: Onur Özcan
Eylül 2019

Mikro hava ara çları hassas elektronik ve mekanik sistemleri dolayısıyla çarpışmalar karşısında uçuş kabiliyetlerinden ödün vermektedirler. Çarpışma önleyici sistemlerin hava araçlarına entegrasyonunda algılama kapsitesinin arttırılması için sensör modifikasyonları yapılması gerekmektedir. Bu ç alışmada amaçlanan, çarpışmaların görev üzerindeki etkisini mekanik koruma ile indirgemek ve gövdenin esnekliğini arttırarak gövdenin hasar görmesini engellemektir. Bu hedefle origami esinlenmeli katlama esasına dayanan bir gövdeye sahip dört pervaneli bir insansız hava aracı tasarlanmış ve üretilmiştir. Çalışma kapsamında hava aracının uçuş dinamikleri anlatılmıştır. İHA üzerinde kullanılmak üzere uçuş kontrol kartı ve yazılımı anlatılmıştır. Elde edilen İHA, Bilkent Minyatür Robotik Labaratuvarı kapsamında ilerideki çalışmalarda kullanılacaktır.

Anahtar sözcükler: Robotik, IHA, katlanabilir robotik, mikro hava araçları.

Acknowledgement

I would like to express my gratitude for everyone who helped me complete the presented work. First and foremost, I thank my academic advisor, Professor Özcan for giving me the opportunity to research a topic I desired and supporting me all the way through the pitfalls and hardships. His patience, sincerity and counsel has been what enabled me to pursue an academic career and kept me going. It has been a joy and honor to work under his supervision. I hope our collaboration and friendship will continue in the future.

I am grateful to my labmates, thanks to whom I had a chance to learn and see from different perspectives. Furthermore, they made working in the lab a fun and enjoyable experience. I thank Cem Karakadioglu, Mohammad Askari, Furkan Güç, Furkan Ayhan, Cem Aygül, Didem Fatma Demir, Nima Mahkam, Tamer Taşkıran and Mert Kalın for great memories. I wish them all the best and success. I also would like to thank my undergraduate student Osman Afandiyev who helped me greatly in the late stages of the research. I wish him the best of luck in his bright career.

My fellow graduate students from the other labs, Mehmet Kelleci, Hande Aydoğmuş, Selçuk Oğuz Erbil, Atakan Arı, Çağatay Karakan, Levent Aslanbaş, Müge Özcan, Mert Yüksel, Ozan Temiz, Dilara Uslu and Cem Kurt also deserve huge credits both in my personal life and academic career. They encouraged me to seek new perspectives to become an interdisciplinary researcher. We shared remarkable friendships which I hope will be lifelong and sincere as it has been.

I also would like to thank my dear friend Doğanay Altıparmak for his intellectual and logical counsel to almost all my problems in life and sharing the worst and best days with humor and sympathy. We have grown together since the day we met before our academic lives and we hopefully will continue to learn and share our lessons in life.

I especially thank Eren Özgün for being available for every bad day of the last

couple of years. He endured every bad joke I made and have been the guiding force who kept me on my track both in my personal and academic life. He shared my passion for music and for that I'm grateful for encouraging me to fulfill my music goals.

Kayhan and Ecem, I wish you prosperity and happiness for the rest of your lives together. I unfortunately was not there to share your joy during your wedding, but know that I will always be here for both of you as your lifelong friend.

I would like to acknowledge the Scientific and Technological Research Council of Turkey, TÜBITAK, for financially supporting me during my studies under Grant No. 116E177.

My mother, father and sister are the ones who deserves the most credit for everything I have ever done and will ever do. Their example, love and respect in every step of my life has been what made me today and I'm grateful for that. This work and everything I will accomplish has been dedicated to them...

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the birth of perspective art in the late 14th century architect by Filippo Brunelleschi, realistic depictions of conceptual perspectives came into being and through amazing creativity of artists, we glanced into what it could be to see the world from above, like a bird. Four centuries after the Hogenberg and Braun's paintings of European cities from above, we gazed upon the same landscapes from above and beyond through the windows of airplanes, space stations, from the surface of the moon and finally the camera of Voyager (figure 1.2). In merely 110 years after the invention of flight, flying machines have become so abundant that a child can attain one as a toy. This have been made possible by the dramatic achievements made in the semiconductor technology that paved the way for the miniature unmanned flying vehicles.

A stable flying platform has many advantages in capability. Flying vehicles, compared to ground vehicles of same weight, is able to cover vast distances faster and independent of the terrain conditions and perform pre-planned maneuvers accurately. Prospectively, it can be argued that micro UAV's will supplement or replace many existing platforms that are carrying out transportation, surveillance

Figure 1.1: Comparison between detailed map of Venice by Hogenberg and Braun from 1574 (left), and aerial photography of Venice in 2016 by Dimitar Karanikolov (right), used with Creative Commons permission.

and physical manipulation.

Miniature air vehicles, such as remotely controlled helicopters and airplanes have been around for more than three decades. Mostly used by hobby communities, their capabilities were limited due to lack of smart components and sensors. After the release of general-purpose microprocessors, field of robotics gained much needed capability of complex on-board computation which is required for command, control and communication critical for holonomic flight. Fused with CMOS and MEMS sensors, situational awareness of the UAV can be extended further, enabling adaptability and automation.

Consequently, robotic research community have gained a considerable interest in such platforms which offer fruitful outcomes in interdisciplinary fields. Success of the kinematic abilities of micro UAV's enabled numerous applications. Micro UAV's are utilized by NASA on the next mission to Mars with Mars Helicopter Scout project, which is a contra-rotating reconnaissance helicopter able to fly in low density atmosphere of Mars [5]. Another use of the concept, popularized in the recent years, camera equipped professional drones which are deployed extensively in the visual arts [6]. Most advanced examples of this concept can be found in the military applications, where small size and capability is a requirement. These robots are used by militaries around the world to provide its operators with critical tactical reconnaissance information. Dubbed the Black

Figure 1.2: Examples of drones and their practical utilities. NASA 2020 Mars Scout (top left) is a contra-rotating helicopter robot, DJI Mavic Zoom & Mavic 2 (top right) aerial photography drones, Prox Dynamics AS Black Hornet Nano MAV used for military reconnaissance purposes (bottom left), Amazon Prime Air delivery drone (bottom right).

Hornet, it is a small micro helicopter weighting 18 grams, equipped with daylight and thermal cameras packed into an extremely small size [7]. Though larger than their counterparts, drones that are capable of carrying heavy payloads are emerging as a new automated delivery system to meet the immense demands of cargo industry. Amazon, is currently in the process of proving the concept of automated delivery using aerial drones such as Prime Air prototype.

These benefits of micro air vehicles continues to promote researchers to improve micro air vehicles to be more durable and adaptable in harsh conditions, able to reach longer distances faster, understand and carry out complex open ended missions with less inputs from its operator with the ultimate goal of reliably autonomous future.

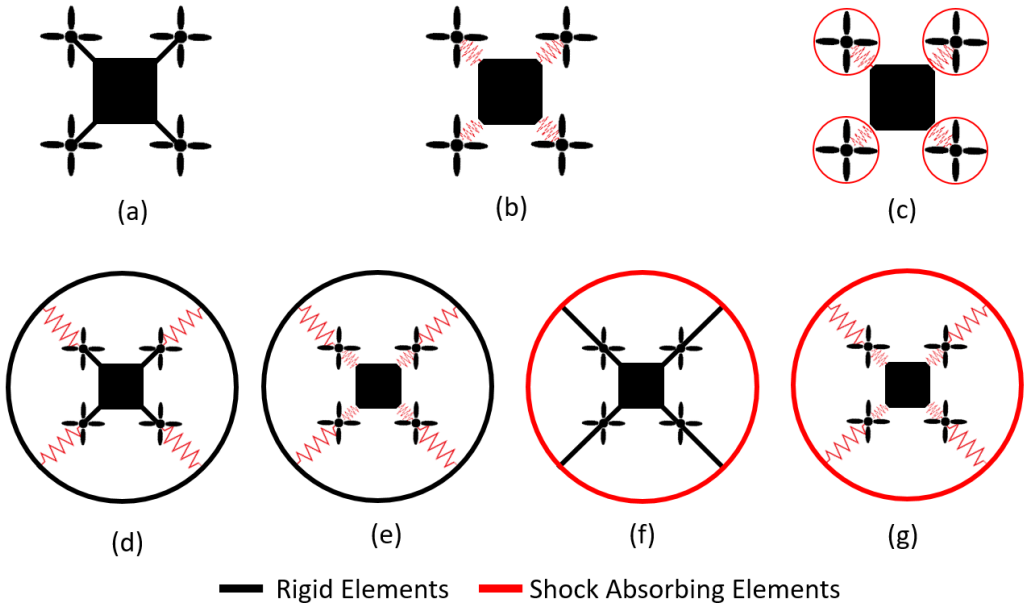

Figure 1.3: Collision protection strategies. (Inspired from [1])

1.2 Research Aims & Objectives

In this work, advancements in the miniaturized electronics and foldable robotic design techniques will be capitalized to design a collision resilient micro air vehicle in the form of a quad-rotor. Said design is envisioned to be a capable and mechanically flexible flying platform built using cheap, available materials. In the future work, design will be utilized to study multi-robot cooperation algorithms.

This work proposes a collision resilient quadrotor body design. Powered air vehicles are prone to collisions due to immense requirement of lift generation. Any disruption in the supply of lift such as power outages, malfunctions in the rotors, sensors or controller errors are critical in the absence of redundancy. Miniature air vehicles are inherently more resilient to mechanical stresses due to laws of scaling. However, rotors and electronics are more susceptible to damages and ruptures in case of an impact.

Aim of this project is to minimize the effect of collisions on the flight and operation. There are number of ways to achieve this task (figure 1.3). Avoiding

the collision is the most common strategy requiring little physical changes, however, it necessitates major modifications on sensory systems and control system. The other option is to allow for collisions and implement physical protection to absorb impact stresses. This protective structures protect rotor blades from being disrupted by impact, which increase the chance of the drone recovering its flight. This work focuses on achieving that goal using foldable robotic techniques using soft elastic materials such as PET films which can be tailored into required stiffness by carefully designed folds.

1.3 Literature Review

Quadrotors have been a hot topic in robotics research in the last decade. Research in general focused on the autonomous technologies with the increasing capability of localization technologies integrated on drones such as cameras or GPS. However, new approaches to the design of core components have also gained interest in the interdisciplinary research communities. Interesting modes of flight have been discovered with modified morphology of drones.

Commercially abundant branch of the micro UAV's, multi-rotors -or more commonly known as drones- can be traced back to the early days of flight. In 1907 Bréguet brothers along with Professor Charles Richet devised the first quadrotors design, namely, Gyroplane No. 1 [8]. Due to unavailability of light-weight, high-power engines the concept left unattended for decades.

With the commercial success of general-purpose micro-controllers, research community gained interest in quadrotors in 2000's. Highlights of the early works this era generally focused on establishing the dynamic model basis with system identification techniques and setups [9][10][11][12] . Model based design techniques (PID, LQR etc.) has been used for attitude stabilization in [13][14] for piloted control augmentation. Non-linear control techniques have been formalized and implemented in [15][16]. Early suggestions of autonomous operation of

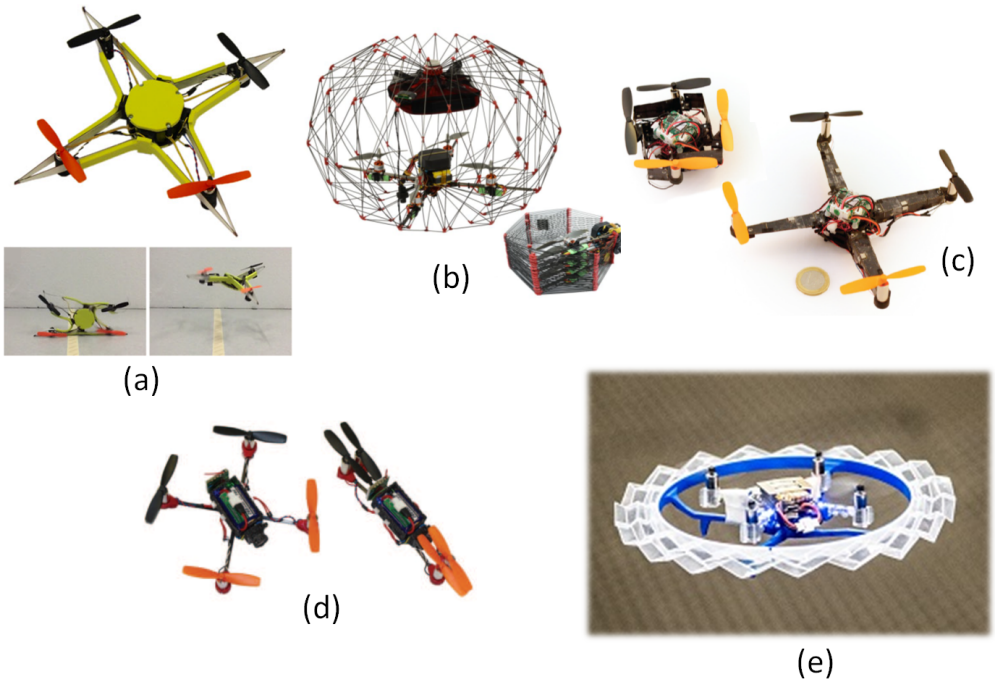

Figure 1.4: (a) Collision resilient drone [1], (b)Origami inspired protective cage equipped drone [2], (c) Pocket-size self deployable drone [3], (d) Pocket-size collapsible drone [4] , (e) Rotorigami: origami cushion for collision protection.

quadrotors have been discussed by leading researchers in [17]. Early success of attitude stabilization combined with emergence of advanced ARM micro-controllers and off -board high rate position feedback technologies, trajectory control applications have emerged [18]. Consequently, literature has drastically expanded as quadrotors became popular among interdisciplinary researchers working on controls [19][20][21][22], autonomous operation [23][24], machine learning[25], machine vision [26], decision making[27], cooperative robotics [28][29].

In parallel, foldable robotics have flourished in the literature thanks to its versatile utility, enabling new techniques for robot manufacturing. Foldable robotics is a new branch in the field of robotics. It dictates a new perspective on design philosophy suitable for miniature robotic designs. Miniature robotics require delicacy in the manufacturing process as design include complex movable mechanisms and compliant materials. Foldable robotics take large inspiration from traditional Japanese craft origami, which is a compound word literally meaning

"folding paper". "Kirigami", a variation of origami, involves cutting of the paper which allows for more complex designs and mechanisms. This technique allows for use of cheap and accessible slabbed polymer materials which contributes in the flexibility against impacts. Foldable robotics takes advantage of this design flexibility, relatively low barrier to realize designs in time efficient manner compared to additive manufacturing, molding, or machining techniques.

Foldable robotics literature offers interesting alternatives for the hard robotics. In [30] axioms and examples of origami inspired robotics has been reviewed. Self-folding structures are one of the most popular research topic among the robotics community. Field of micro-nano robotics employs foldable robotics approach to produce complex shapes and mechanisms [31][32]. Self-folding robots use variety of inventive actuation techniques using shape memory alloys, pneumatic channels, piezo-electric actuators to accomplish self-folding without external help[33][34][35].

Collision resilience and foldable quadrotors have a presence in the literature providing different perspectives on design and operation. [4] shows a deployable quad-rotor which is able to reduce its size by collapsible arms. A The researchers have designed a similar drone in [3] using a different configuration of arms though which is able to deploy its arms by utilizing reaction torques.In [1], researchers used a plastic structural material with high elasticity to absorb and redirect the collision energy into deformation. Research in [2], shows a different approach to collision management by covering the quadrotor body with a spherical cage. An origami inspired protective bumper has been designed by researchers in Imperial Collage London in [36] which uses very similar materials used in this project.

1.4 Structure of the Thesis

The work is compiled in the manner explained in this section.

Chapter 2 focuses on the principles of quadrotors in general. Later, mathematical model of the quadrotors will be explained. This chapter includes a non-linear model which accounts for the motor forces, gyroscopic effects and coupling to provide guidelines for a simulation environment.

In chapter 3, design of the foldable drone is discussed with the related manufacturing processes in conjunction. Electronic suite and sensory capabilities have been given. Reasoning behind the design has been explained in detail. PCB designs for the purpose made flight controller board as well as the commercial flight controller used in the design will be discussed. Design iterations culminated towards an optimized design has been explained in detail to give a larger picture of the research.

Chapter 4 will introduce the control schemes employed to stabilize the attitude system with its implementation on the flight controller.Communication and embedded hardware software interface in the flight code will be explained with the general workflow of the flight code.

Chapter 5 presents the experiments conducted on the assembled drone. Flight experiments including collision tests will be shown and analyzed. Results of tests show footage from slow motion videos of drop tests and collision tests. Later sections concludes thesis with analysis of the work output, lessons learned, and possible improvements to increase resilience in high angle of attack, high speed collisions. Successive work will be explained in the light of the experience gained during this project and literature.

Chapter 2

Quadrotors

2.1 Principles of Quadrotors

With the advancements in silicon technology which allows fast computation with responsive electric motors enabled the multi-rotor technology. Multi-rotors in general consist of three or more vertically mounted trust generating rotor propeller assembly. It is comparatively easier to control the motion in multi-rotors than helicopters which require complex actuator mechanisms for axial control.

Multi-rotors utilize the relative rotation speeds of fixed rotors to achieve torques in 3 axes. Since all rotors are pointing in the vertical direction, trust vector is fixed along the z-axis of the frame. By controlling the rotation of the frame, trust vector can be directed along reference frame axes. This means that a multi-rotor can achieve 6 degrees-of-freedom non-holonomic motion by using four control inputs, namely, thrust, roll, pitch and yaw.

Ideally, quadrotors can hover steadily in open-loop configuration when certain conditions are satisfied. In ideal configuration, each rotor-propeller assembly is positioned in the same distance from the center of mass and parallel along the same axis. Rotors must generate the equal reaction forces and thrust. Cumulative

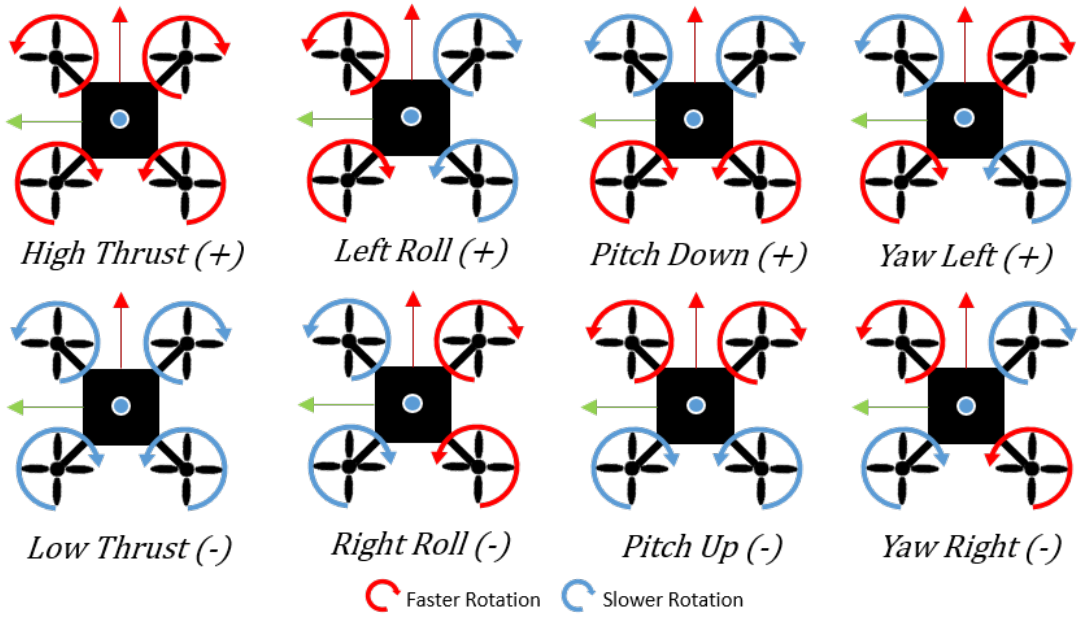

Figure 2.1: Roll, pitch, yaw operator inputs translated into rotor inputs for X-Quad configuration.

thrust should be equal and opposite to the weight of the quadrotor. Additionally air density across the body is homogeneous and undisturbed. These conditions, as noticed, are unrealistic, therefore, must be monitored and reacted upon during operation. Even though, hovering stability is possible under these conditions, steering is not possible because of the fact that system inputs -rotor speeds- are mixed to achieve plant angle outputs, yaw, pitch, and roll (figure 2.1). Therefore, quad-rotors are deemed dynamically unstable.

Nature of quadrotors necessitates a feedback control system. Controller makes thousands of adjustments at any given second to compensate for internal (vibration) and external disturbances (wind). Addition of translating rotor inputs into more intuitive thrust, roll, pitch, and yaw makes it easier for human pilots. This indirect control technique can be extended to one more level of abstraction with trajectory control.

2.2 Quadrotor Dynamic Modelling

Dynamical modeling of a micro aerial vehicle is a crucial step of the development stage. It is important to understand the behavior of the vehicle on air where sensory information would not supplement the full picture. Full dynamic model of quadrotors including the rotor and propeller dynamics with flapping mechanisms and aerodynamics of the flight has been explained thoroughly in [14], [37]. Generally neglected hub forces, ground effects and flapping dynamics have been discussed in [38].

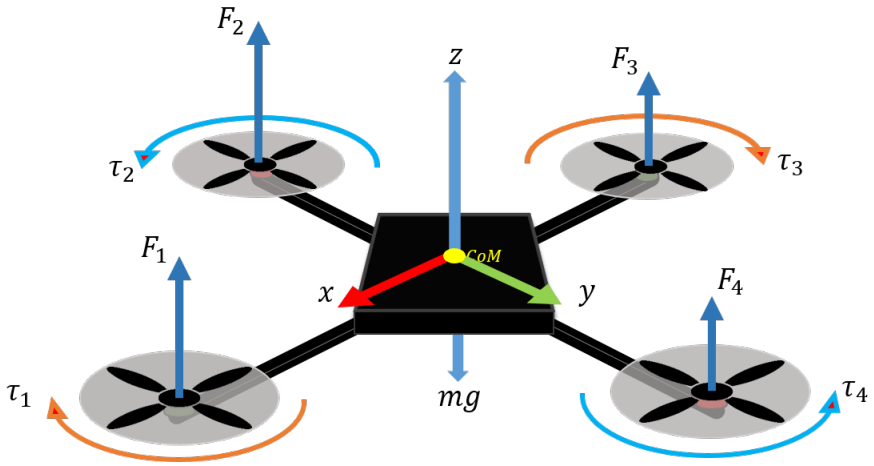

Figure 2.2: Free Body Diagram of a quadrotor. $F_{1,2,3,4}$ are forces generated by propeller assemblies, $\tau_{1,2,3,4}$ are reaction torques generated by rotating propellers.

2.2.1 Quadrotor Dynamic Model

To simplify the dynamic model quadrotor frame is assumed to be rigid, drag forces are negligible for small speeds, body center of gravity is located in the geometric center of the body coinciding with the center of thrust and at the origin of body reference frame B consisting of $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$. Earth reference frame E is defined by $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$. Body frame is related to the rotation matrix $\boldsymbol{R}$, where R is a member of 3D rotation group in R^3.

$$R = \begin{bmatrix} C_\phi C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi - S_\phi S_\psi \\ C_\theta C_\psi & S_\phi S_\theta S_\psi - C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ S_\phi & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix}$$

Using the Newton-Euler equations, which formulates body motion under influence of forces and moments, motion can be expressed with respect to coordinate frame B:

$$\begin{bmatrix} mI_{3x3} & 0 \\ 0 & I_com \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} w \times mV \\ w \times Iw \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \tag{2.1}$$

m is the body mass,I_com is the inertia matrix w.r.t. B, I_{3x3} is 3 by 3 identity matrix , V is the axial velocity vector and w is angular velocity vector. Therefore, equations of motion can be written in as,

$$\begin{aligned} \dot{r} &= v \\ m\dot{v} &= RF_b \\ \dot{R} &= R\hat{w} \\ J\dot{w} &= -w \times Jw + \tau_a \end{aligned} \tag{2.2}$$

Using the free body diagram shown in fig.2.2, equations can be re-written in detail as:

$$\begin{aligned} \dot{r} &= v \\ m\dot{v} &= -ge_z + Re_z(\frac{b}{m}(\Omega_1 + \Omega_2 + \Omega_3 + \Omega_4)\hat{k}) \\ \dot{R} &= R\hat{w} \\ I\dot{w} &= -w \times Iw - \sum_{n=1}^{4} J_r(w \times e_z)\Omega_i + \tau_a \end{aligned} \tag{2.3}$$

Where r is the position vector, g is gravitational acceleration, e_z is the basis vector in B. $\hat{w}$ is the skew symmetric rotation matrix, b is propeller thrust factor, Ω_i is rotor rotation speed and J_r denotes the propeller inertia.

Roll, pitch and yaw control inputs can be arranged in terms of torques to be inserted into the quadrotor dynamics can be shown as:

$$
\begin{aligned}
\tau_{roll} &= lb(\Omega_4^2 - \Omega_2^2) \\
\tau_{pitch} &= lb(\Omega_3^2 - \Omega_1^2) \\
\tau_{yaw} &= d(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2)
\end{aligned} \tag{2.4}
$$

Consequently, we can define system inputs without moment arm length l which is a body parameter:

$$
\begin{aligned}
U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
U_2 &= b(\Omega_4^2 - \Omega_2^2) \\
U_3 &= b(\Omega_3^2 - \Omega_1^2) \\
U_4 &= d(\Omega_1 - \Omega_2 + \Omega_3 - \Omega_4)
\end{aligned} \tag{2.5}
$$

Yaw constant d is obtained by angular momentum equation which is parameterized with rotor speeds $\sum \Omega_i$,rotor inertia J_r, body inertia I_z and body yaw rate. Additionally, gyroscopic disturbance caused by rotation of rotors that effects pitch and roll can be noted as:

$$
\Omega_{gd} = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4
$$

Equations (2.3) and (2.5) are combined to obtain equations of motion in 6 degrees of freedom as:

$$
\begin{aligned}
\ddot{\phi} &= \dot{\theta}\dot{\psi}(\frac{I_y - I_z}{I_x}) - \frac{J_r}{I_x}\dot{\theta}\Omega_g d + \frac{l}{I_x}U_2 \\
\ddot{\theta} &= \dot{\phi}\dot{\psi}(\frac{I_z - I_x}{I_y}) - \frac{J_r}{I_y}\dot{\phi}\Omega_g d + \frac{l}{I_y}U_3 \\
\ddot{\psi} &= \dot{\phi}\dot{\theta}(\frac{I_x - I_y}{I_z}) + \frac{l}{I_z}U_4 \\
\ddot{x} &= (cos\phi\ sin\theta\ cos\psi + sin\phi\ sin\psi)\frac{1}{m}U_1 \\
\ddot{y} &= (cos\phi\ sin\theta\ sin\psi + sin\phi\ cos\psi)\frac{1}{m}U_1 \\
\ddot{z} &= -g + (cos\phi\ cos\theta)\frac{1}{m}U_1
\end{aligned} \tag{2.6}
$$

It is apparent that non-dynamics introduce coupling effect which can be neglected around hovering.Thus it is possible to introduce linearization around hover position, $\phi = 0, \theta = 0$ to remove gyroscopic terms to obtain a linear attitude model to design a controller scheme for attitude stabilization.

$$\begin{aligned} \ddot{\phi} &= \frac{l}{I_x} U_2 \\ \ddot{\theta} &= \frac{l}{I_y} U_3 \\ \ddot{\psi} &= \frac{l}{I_z} U_4 \end{aligned} \tag{2.7}$$

2.2.2 Rotor Dynamics

Rotor dynamics are the bridge between controller scheme and the quadrotor dynamics. Rotors are electro-mechanical actuation systems which are driven by digital micro-controllers in quadrotors. Therefore, a system block is required to associate a voltage input with mechanical rotation output.

Rotor dynamics are well established in the literature [39][40][41]. [15]. The governing equations are:

$$L\frac{di}{dt} = u - Ri - k_e w_m, \qquad J\frac{dw_m}{dt} = \tau_m - \tau_d \tag{2.8}$$

Since modeled motors are small, second order dynamics can be approximated by,

$$J\dot{w}_m = \frac{k_m^2}{R} w_m - \tau_d + \frac{k_m}{R} u \tag{2.9}$$

Where,

Symbology	
L	motor inductance
u	motor input/PWM
τ	motor time-constant
R	motor internal resistance
k_e	back-emf constant
k_m	torque constant
w_m	motor angular speed
τ_m	motor torque
τ_d	motor load
J_t	total inertia

Chapter 3

Foldable Micro Quad Design

Foldable robotics offer a number of advantages for micro air vehicles. Suitability of abundant, lightweight materials allows for lighter structural designs compared to their counterparts by trading design complexity. It also offer fast, low cost batch prototyping and fabrication. Most important aspect, however, is that foldable robotics offer customized mechanic properties of structures that can be tuned for the purpose. As shown in Fig. 2.2, quadrotor frame experiences high forces compared to its size. Body design must provide the quadrotor with reasonable stiffness to hold the rotors firmly, while able to provide shock absorbing feature with spring-like elasticity.

3.1 Material Selection

The definition of requirements eliminates the use of a range of materials such as paper which undergoes plastic deformation easily under stress. This means low stiffness materials require more folds, consequently more surface area and weight to be folded. Same level of elasticity and durability can be obtained with the use of polymer materials. PET, acetate film is cheap and high availability material. This material was also used in the other projects of the Bilkent Miniature Robotics

Lab [42], [43]. Therefore, using the material with the know-how and infrastructure of the Lab has proven beneficial.

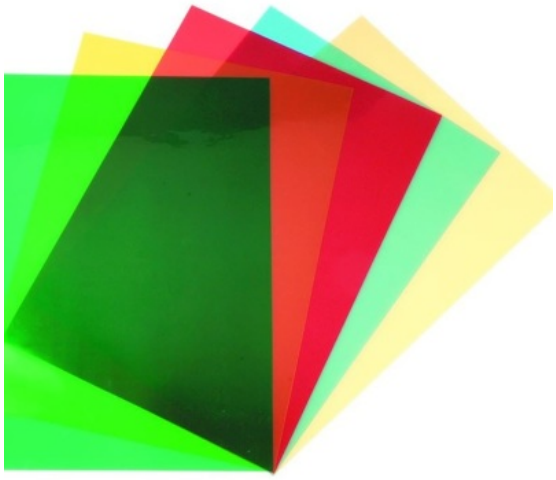

Figure 3.1: PET film sheets used in the body of quadrotor.

PET material can be found in slabbed A4 dimensions with variety of thicknesses of 100, 250, and 500 μm. Experimentation with different thicknesses have shown that low thickness films offer more elasticity and better fatigue protection under cyclic stresses, however, folded structures made with the material are unable to keep the intended shape. On the contrary, thicker films offer greater strength under higher loads, however, it becomes difficult to fold and they are prone to sudden breakages and fatigue. $250\mu m$ thickness PET film has been chosen as the primary material for this project.

3.2 Body Design

3.2.1 Design Guidelines

Quadrotor body provides protective housing for the electronics and solid mounting for the rotors. This is achieved through folds by directing the material stiffness in various directions and using stiffness of folded regions as tie-beams to bear

loads. Locking mechanisms and connections provide docking and stress relief measures.

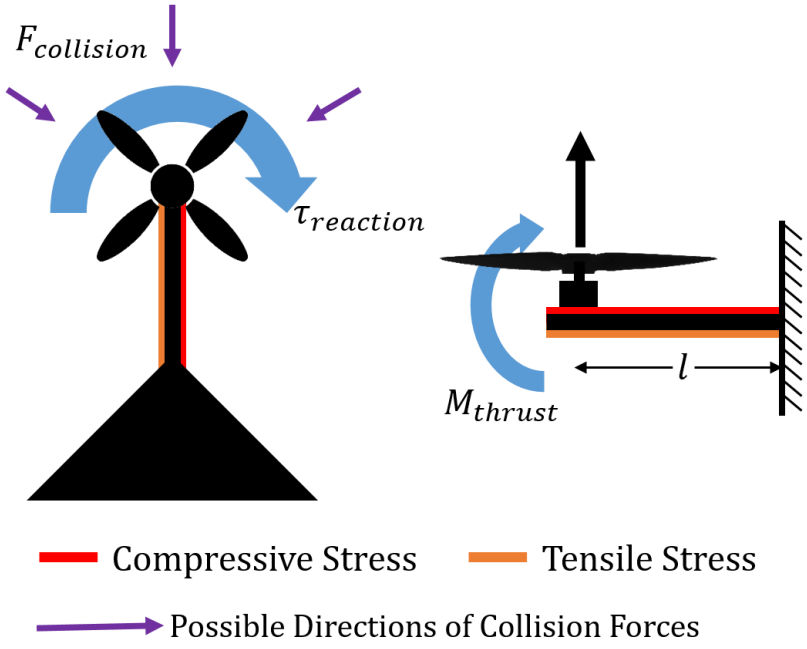

Figure 3.2: Forces acting on the arms.

Quadrotor body has been shaped in the X-Quad configuration where set of 2 rotors are placed on a flexure arm which is fixed to another arm holding the remaining 2 two rotors. Rotors are placed at the tip of arms to moment generating ability. Arms length selection affects the operation of the quadrotor drastically. Longer arms increase the generated moment and moment of inertia. Increased length must be accompanied by mechanically stronger arm to accommodate for the increased moments which may result in quadrotor folding on itself. Shorter arms generally require more thrust to steer the quadrotor since the moment arm is reduced, but, it is lighter and easier to design for required stiffness specifications.

When air is pushed by the propellers, fraction of air has to go over the arms. This reduces the thrust as the air momentum is lost on the quadrotor body. To minimize trust loss, arms must be constructed to avoid blow-back. Therefore, arm width has been reduced as possible.

3.2.2 Early Design

The first foldable body was constructed as a proof of concept, using the same X configuration, with brushed motors and the earliest manufactured PCB mentioned in Ch. 3.2.2. Structurally, early version lacked strong load bearing fold structures, rather, tab-slab locking mechanisms provided the adequate amount of stiffness to prevent arms from bending around the enclosure. The comparatively large electronics are enclosed inside two piece box enclosure.

Figure 3.3: First prototype of the foldable micro quadrotor equipped with Arduino Nano, MPU6050 IMU, and coreless DC motors.

3.2.3 Arms and Rotor Fixtures

Arms are constructed to restrict relative motion with the body core along thrust direction and about the rotation axis of the rotor (Fig 3.2). Arms are designed to be flexures which are engineered to be compliant in z-axis rotation about the body fixture during impacts to prevent irreversible damages. A more rigid construction would equate the ultimate strength of the material to the ultimate strength of the construction. Compliant foldable mechanisms can sustain larger amounts of stresses especially. As the deformation propagates, load is distributed more homogeneously.

In order to restrict motion along the thrust vector, a T-fold has been implemented. The T-fold has a side benefit of having a component along the plane of the reaction force which can restrict this motion with minimal modification of adding a slanted surface to connect both ends of the half T shape. Two triangular shaped elongated cells are folded from the same piece of PET film. These cells, when combined form a larger triangle which increases the overall strength and enables easier folding and manufacturing.

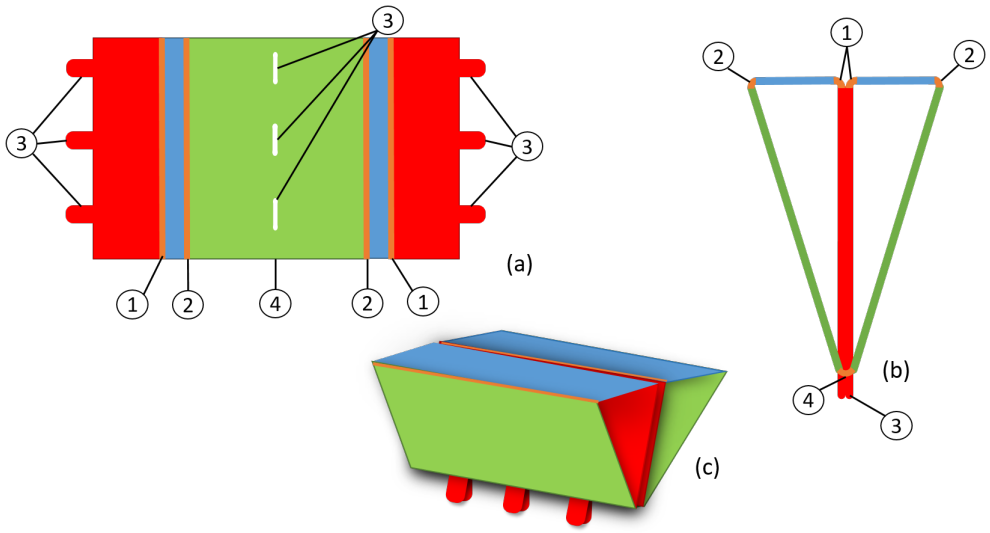

Figure 3.4: Folding of the arms. (a) Unfolded paper sheet, orange seam lines are perforated with laser for accurate folding. Folding hierarchy is noted with numbers. (b) Cutaway of the folded load bearing arms, triangular cells can be seen. (c) Overall shape of the folded arms.

Figure 3.4 shows the folding pattern. Surface shown in red is bearing the load of generated thrust by forming the tail of the T-fold. Blue surfaces restrict red surface from buckling under compression stress. To keep the blue surfaces from collapsing under collision, green surfaces provide another point of fixture. Locking extensions (fig. 3.4, No. (4)) are inserted into the slots engraved on green surface. This blocks the fold from unwinding and provides the attachment point to the main body enclosure. For each quad-rotor, two diagonal arm flexures are required, each supporting 2 rotors. Arms are joined together to reach X configuration using half-lap joint in the middle point.

Rotors are required to hold their intended configuration for smooth operation.

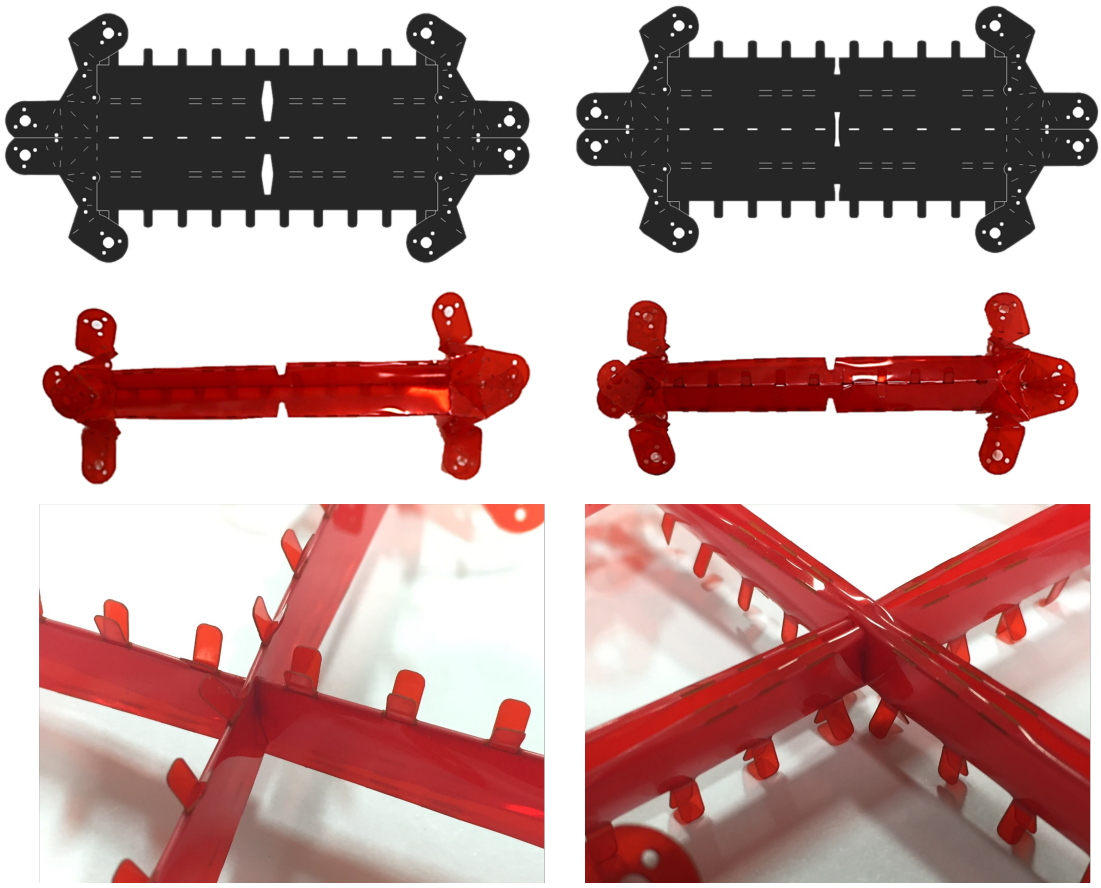

Figure 3.5: Two diagonal arms joined together with half-lap joint.

Misaligned rotors do not cause major disruptions thanks to the feedback controller, however, it increases power consumption due to loss of thrust. Eachine TC0803 rotors are held in their place firmly using three 1.2mm screws which connects the rotor base to arm tips. Fixtures can be considered as the most critical part of the design. It experiences most of the stresses caused by thrust, reaction torques, vibrations, and impact stresses. Moreover, mounting points experience dissipated heat by the rotors. In order to prevent local warping of the PET film temperature of the rotors must be monitored during the operation. Additional layer of DuPont Kapton can be applied in the contact surface between the motor and fixture.

In order to provide easy access to screw holes and add extra durability, a 3D three pointed star shape has been placed at the en of the tips. Three mounting

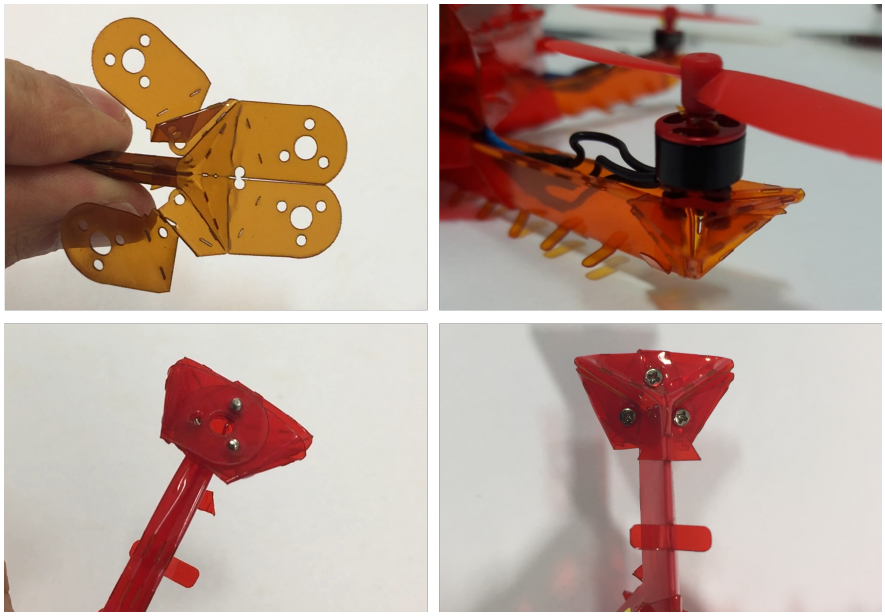

Figure 3.6: Half folded fixture pattern cut with laser engraver (top left). Fully folded and fixed with screws seen from top (top right), bottom (bottom left). Fixture mounted with rotor attached (bottom right).

points share the load and direct the stress along the side surfaces of the arm.

3.2.4 Body Enclosure

To support the X-Quad arm assembly and to provide housing for the electronics and batteries, the main body housing plays an important role. Main body housing is attached to the assembled arms through the locking extensions of arms. The locking requires pre-stressing the body to align the locking extensions with the slots and wraps around the arms. Locking can be completed by splitting the locking extensions apart. Pre-stressing drastically increases overall body integrity by adding another layer of triangular fold over the arms.

Upper body enclosure is folded around the body to provide a steady fixture for flight controller and batteries. Enclosure is a open top rectangular box on the bottom and a half pyramid on the top. The angling fold between these two shapes increases the rigidity of side faces of the enclosure.

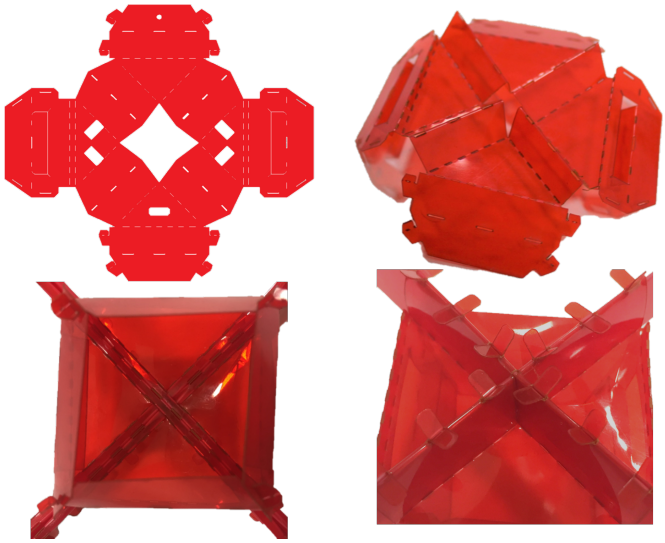

Figure 3.7: Unfolded main body (top left), Main body enclosure half-folded (top right), Main body closure fixed to the arms using the locking extensions seen from top (bottom left) and from bottom (bottom right).

Batteries are larger than the enclosure dimensions. Thus, rather than making the enclosure larger, batteries are held by restricting openings on two opposing sides of the enclosure. Tips of batteries remain outside by 1 cm. Precisely because of that bottom part of each side warps due to diminished strength of the surfaces in the face of pre-stressed body . This has been solved by an extra fold under the battery holders openings.

Crazybee F4 motor and USB sockets are given an opening in the enclosure floor. Consequently, motors can be easily plugged without time consuming disassembly.

3.2.5 Protective Bumpers

Collision protection ability has been established using bumpers made out of PET film sheets. Outermost protection is required around the rotors to prevent propeller damage as the first point of contact with obstacle. Protective bumpers absorb the collision energy by elastically deforming, and releases the energy by pushing the quadrotor body away from the obstacle.

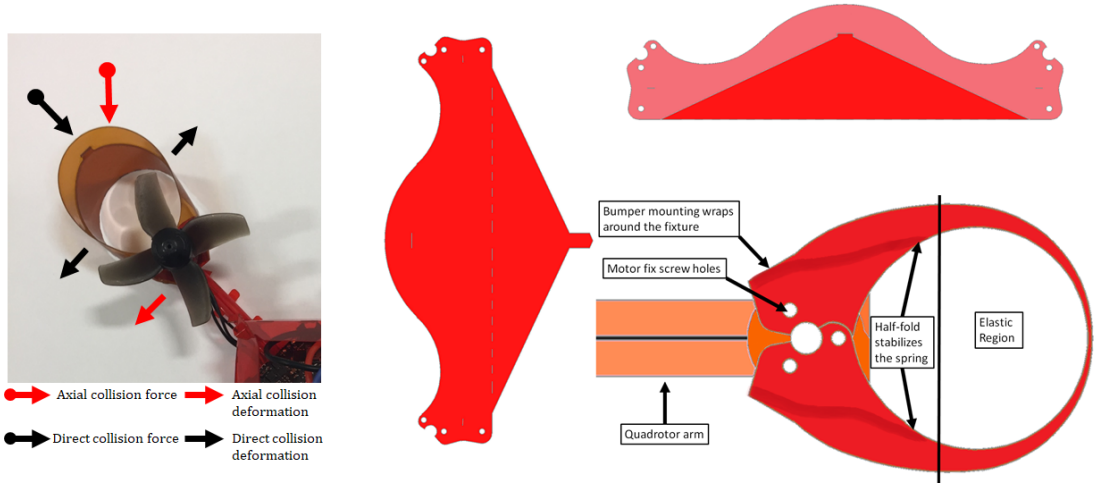

Figure 3.8: Collision scenarios and related force reactions(left), Unfolded bumper part (middle), folded and fixed bumper part (right top), bumper and rotor fixture assembly seen from top (right bottom)

PET material is formed into spring loop mounted around the motor fixture using the motor screws. An extra layer of pet film is folded on the bumper part to increase spring coefficient. Mounting point plays an important role in forming the spring. Bumper mount wraps around the motor fixture from the sides. Half-fold encapsulates the rotor fixture with U-shaped wrappings around the sides. This stabilizes low stiffness PET film to form a directional spring around the rotors. This springy section of the bumper absorbs the most of the collision energy.

In two collision scenarios, different parts of the bumpers are subjected to stress. In direct collisions shown in fig. 3.8, most of the collision energy has been stored in the elastic deformation of the elastic region. The stored energy is turned into momentum opposing the collision surface. Bulge over the larger fold acts as a corrective measure. Bulge creates a small amount of moment to compensate for large angle of attack impacts and corrects for a departure angle. In the second scenario, energy absorption is largely taken over by elastic arms. Bumpers act as a guide for a planned collision to utilize elasticity of arms.

A larger protective origami structure has been designed in [36]. These types of protective bumpers require a stiffer platform to keep intended shape which is

s

Figure 3.9: Implementation of the bumper, mounting point and half-fold is visible.

harder to accomplish using only PET films.

3.3 Electronics & PCB Design

To equip quadrotor with the flight ability, regulating feed-back control system to provide rotor inputs is essential. On that end, on board computation and sensing capability have to be provided. This need can be satisfied with the commercial flight controller boards such as CC3D, Naze-32 or Crazybee F4 boards. Said boards are capable of fast sensing/computation capability with options to outfit communication suites and extra sensors. However, they are comparatively harder to program into the requirements of a researcher and require cumbersome modifications to the flight code.

3.3.1 Purpose Made Flight Controller

In the scope of the project a flight controller board has been designed and manufactured for the use of researchers with little embedded programming experience. Board is outfitted with Atmel ATMEGA328P-AU chip which can be integrated with Arduino IDE interface to skip over cumbersome Embedded C problems . Arduino provides ease of use with medium level abstraction with extended libraries for otherwise cumbersome embedded electronics integration. ATMEGA chip is an 8-bit reduced instruction set computer based micro-controller with 32kB ıf ISP RAM. Chip, being 8-bit, is not as capable as STM-32 is a 32-bit chip meaning it can process 32-bits of data every clock cycle.

Flight controller board is equipped with 6-axis Invensense MPU6050 MEMS accelerometer and gyroscope chip. Chip has a built in motion processing unit. This on-board processor utilizes a sensor fusion algorithm combining accelerometer data with the angle rates obtained from the gyroscope to give accurate attitude estimations. This consequently means additional filtering is not required on the main microprocessor. To allow for autonomous operation in the future projects, Bosch BMP280 atmospheric pressure sensor has been implemented. The sensor can detect pressure changes of ± 1 Pascal which lies in the range of ± 8cm. Both sensors are transmitting data to the main CPU via Inter-Integrated CircuitI^2C protocol which allows for 400 kHz data transmission speed.

To communicate with the drone, an RF suite is required to wireless data transmission and reception. This is achieved by nRF24L01 2.4 GHz RF Transceiver. Module allows point-to-point half-duplex communication within the range of 100 meters in ideal conditions. The module is connected to the microprocessor via Serial Peripheral Interface(SPI) connection.

The board design is most suitable for use with brushed DC micromotors, although it requires minimal modification to drive brushless DC motors via Electronic Speed Controller(ESC) interface. In order to drive brushed DC motors, four IRLML2505 high-power MOSFET transistors are used for fast switching.

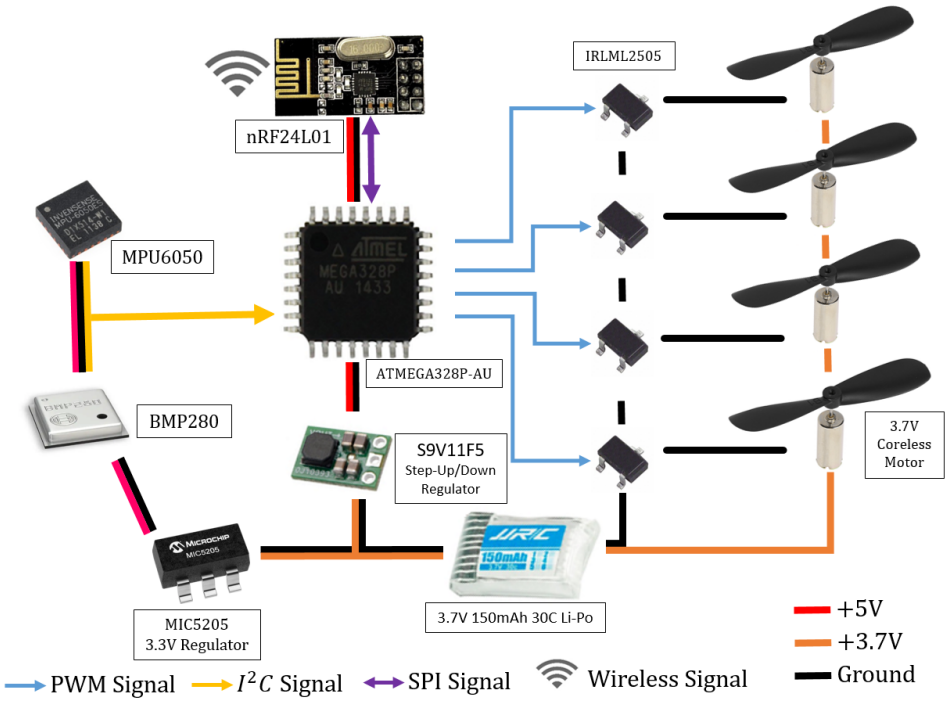

Figure 3.10: Active electronic elements and circuitry including power and signal layout. Passive components are not included.

High performance coreless brushed DC motors are easily sourced from the internet, however, their manufacturer, model and specifications are inaccessible in most cases. These motors are generally rated about 12000KV at 3.7V. Small propellers found on the market are also not standardized and untraceble. This poses some problems in the modeling and control since it directly affects the thrust characteristics.

In-house board manufacturing started with the early trial of using the breakout boards connected via a Printed Circuit Board (PCB). This prototype PCB was produced from DuPont Pyralux which is a copper plated flexible Kapton. Use of Kapton considerably reduces the weight compared to FR4 type copper plated materials or point to point electrical wiring. Flexibility of the material also allows for folds to reduce housing dimensions.

To produce a PCB, one must remove the unwanted copper around the desired pattern. Conventional milling of the copper is long and unsuitable process for

Pyrolux. Most practical and low-cost approach is chemical etching using H_2O_2 and HCl etchant which removes cooper surface to form $CuCl_2$ and H_2O_2. This amplifies the etching process since $CuCl_2$ also reacts with Cu to form 2 CuCl molecules. One more step of etching can be achieved by dissolving more oxygen to react with $CuCl$ and HCl to produce more $CuCl_2$ This etchant can be used as long as oxygen and HCl levels are maintained in the solution. This is a fast etchant, therefore it may cause over-etching problems if left unattended.

Wiring pattern should be covered by a protective layer to deny etchant from contacting with the surface. This is done by coating the surface of the Pyrolux with Nitrocellulose based nail polish. This is less expensive and easier to use compared to commonly used photo-resist types which requires additional photo-lithography and curing steps before etching.

Masking step is followed by lithography to remove the excessive nail polish outside pads and traces. Nail polish is ablated using Universal Systems Universal Laser systems VLS 3.30 laser engraver. The drawback of this method is that it requires careful work of cleaning the burnt residuals left by the laser. This may disrupt the functioning of the etchant by masking unwanted areas, creating copper shorts. Experiments have shown that spin-coating the copper surface 3 times in 5000 rpm for 10 seconds is the optimal for highest resolution of 200 μm feature size.

This version of the board was adopted with the early foldable body design. Although the board had a lighter PCB, use of breakout boards were accounted for 5 grams of additional weight. Therefore, a new design has been devised to use the manufacturing PCB to hold surface mount components. This resulted in a smaller surface area packed with the required electronics. Overall weight of the board has been removed to 3 grams with the soldered components.

Pyrolux board was electronically successful. Mechanically the board failed into the tests with sudden breakages in the vicinity of copper borders. Although stopgap repairs were possible, in long term use of Pyrolux have proven to be unfeasible. To fix this problem, a 0.6mm FR4 has been supplied and manufactured

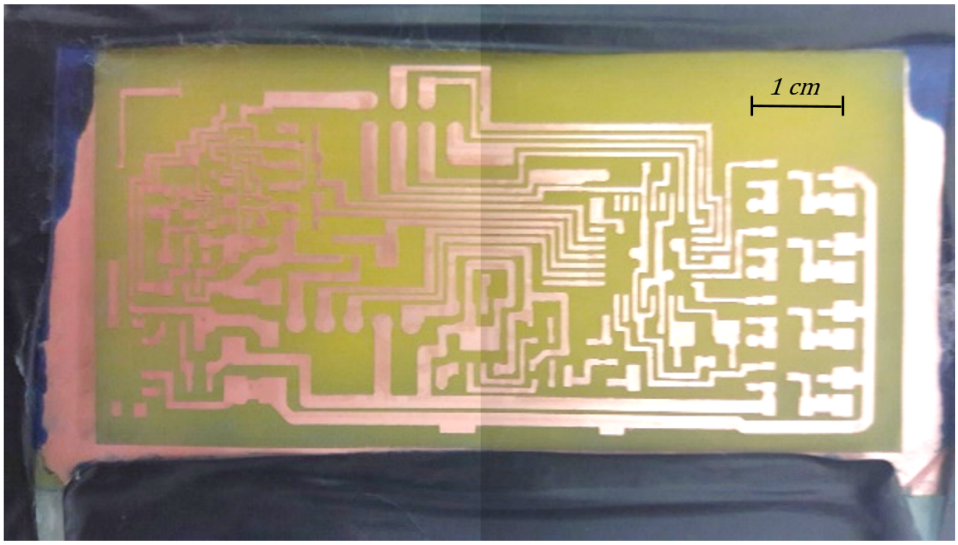

Figure 3.11: Manufactured PCB using Laser Lithography and H_2O_2 and HCl etchant. Small feature sizes can be noticed around QNF type chips.

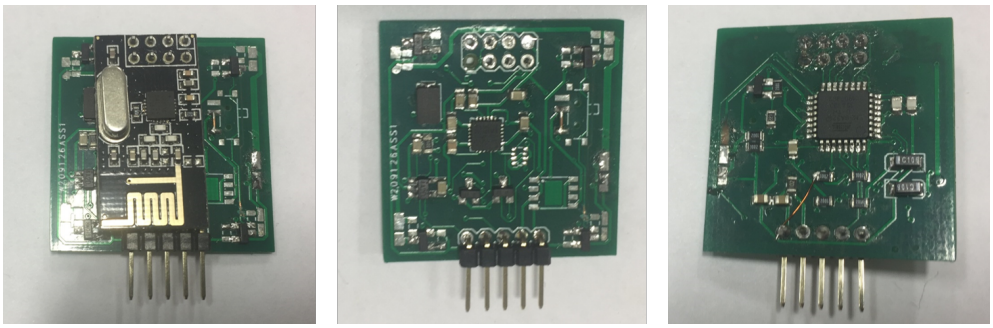

Figure 3.12: Manufactured PCB using Laser Lithography and H_2O_2 and HCl etchant. Small feature sizes can be noticed around QNF chips.

using the layout devised for the small board. Nevertheless, the accuracy of used masking and etching methods were not refined enough for QNF type mountings to be repeatable and reliable.

Next version of the flight controller board was outsourced for production. Commercial PCB makers offer low prices with reliable accurate boards. FR4 with 0.6 mm thickness was used for this board. This board has proven to be capable of meeting most of the requirements. However, combination of ATMEGA328 and MPU6050 via I^2C connection resulted in extremely low gyro rate (285 Hz) which

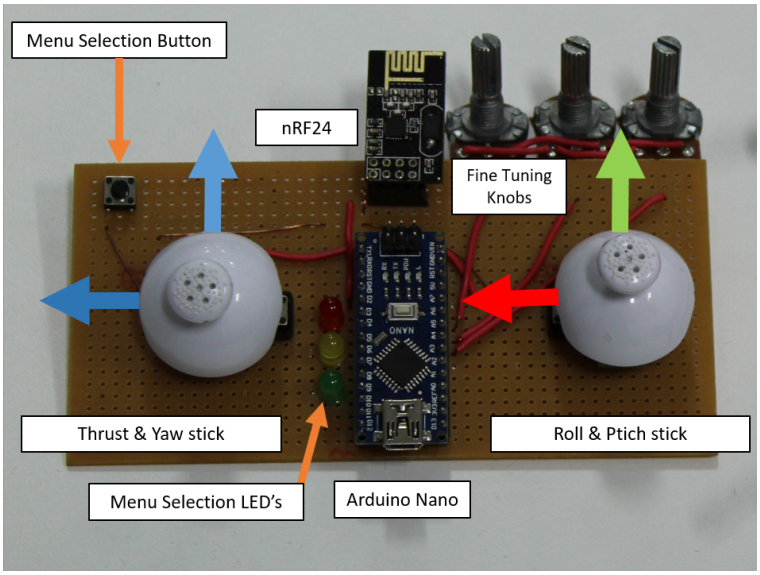

Figure 3.13: Remote Controller hardware.

is unsatisfactory for drone flight. This will be explained further in Chapter 4.

3.3.1.1 RC Controller

To work alongside flight controller, a remote controller has been built to transmit user data and receive flight information from the flight controller. Remote controller is equipped with an Arduino Nano to handle data and connect to PC, 4 channel joystick for manual thrust, yaw, pitch and roll inputs. and a complimentary nRF24 to communicate with the board. Additional controls to fine tune controller and a push button is provided to customize flight settings. Operation of the RC controller is explained in section 4.2.2.

3.3.2 Commercial Controllers

Unsatisfactory gyro refresh rate obtained from the purpose-made flight controller have incapacitated the flight ability of the quadrotor. In order to conduct experiments, an alternative solution have been investigated. Conclusively, Crazybee F4 Pro, a commercial flight control board with extended capabilities have been

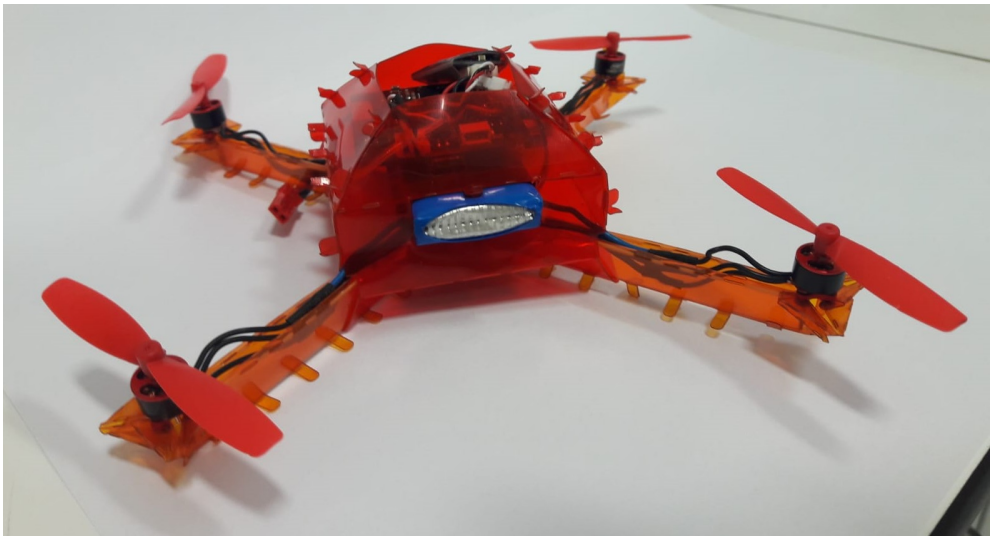

Figure 3.14: Larger design equipped with CC3D flight controller and 2S ESC.

adopted for the project.

First commercial controller used in the project is Open-Pilot based CC3D. Its main processing unit is an STM32F104 32-bit micro-controller integrated with MPU6000 inertial measurement unit with 1kHz refresh rate. Board must be supplied with 2S 4-in-1 Electronic Speed Controller to drive motors and small RF receiver to receive steer commands. The bulky construction of the ESC and the flight controller resulted in a 50 grams quad-rotor. To accommodate the larger electronics, quad-rotor body got larger. The main benefit of the CC3D is its availability in the country, however, compatible ESC's are hard to obtain.

Crazybee F4 is another flight controller that employs 32-bit STM32F4 type micro-controller unit integrated with MPU6000 inertial measurement unit which provides 8kHz refresh rate. Along with the flight control unit, FlySky compatible 2.4GHz RF receiver and 4-in-1 Electronic Speed Controller unit for 2S brushless DC motor control has been packed inside 4.3g package. Board offers exceptional advantages compared to its counterparts making it suitable for micro quad-rotors.

The board has been supplied with suitable 4 Eachine TC0803 brushless DC motors with 15000KV rating. Combined with the batteries the resulting quad-rotor including the body weighed 41.3 grams, 6 grams being the body. It has been

noted that peak thrust of each motor is in average 25 grams, Thrust-to-Weight ratio of the final quad-rotor is around 2.5. However, the resulting electronic package is unavailable in Turkey and has been supplied from abroad.

Flight software has been provided by open source BetaFlight which enables users to customize controller settings easily. However, being an open-source platform, it has a complicated backhand software which has been constructed by numerous coders. Comprehensive workarounds require meticulous work to analyze the software and time. However, for the purposes of this project extensive modifications are unnecessary, therefore, the base code has been left unchanged to conduct tests.

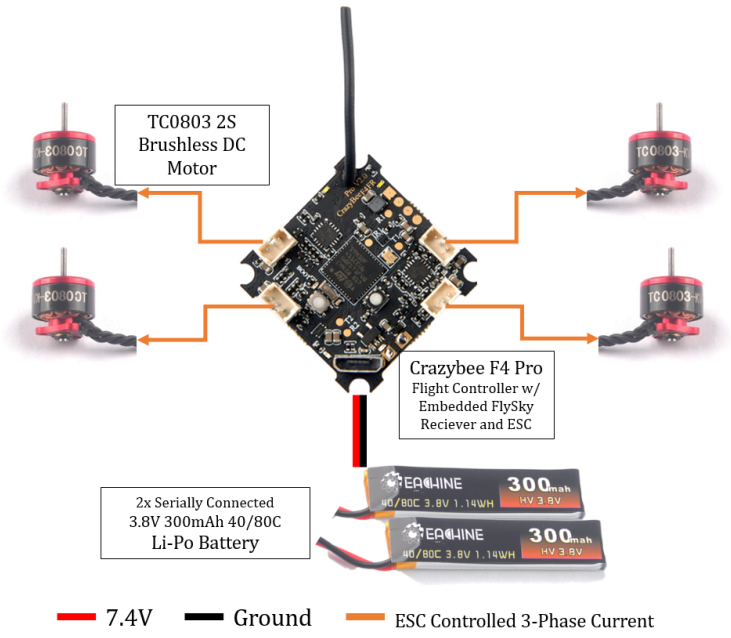

Figure 3.15: Setup of the finalized electronic suite equipped with Crazybee F4 Pro.

3.4 Dynamic Collision Structural Modeling

In order to formalize the behavior of the deformation of the foldable body under flight stresses and collisions, we can employ analytical techniques. Compliant

characteristics of the foldable body and the outlier bumpers can be studied analytically by modeling the compliance as a spring phenomenon with energy losses due to friction modeled as dampers. By calculating the forces acting upon each component we can obtain a useful dynamic collision model albeit simplified. This precursor study will be expanded to fully cover elastic collisions of compliant airframes. Simplifications arose from the assumptions made in the model. It is assumed for this study, collision takes places in the hover position where pitch and roll angles are zero with the body having an initial velocity vector given in the moment of contact. Additionally collision surface is perpendicular to the quadrotor initial velocity vector, which is called a direct collision. Yaw angle is zero and the direction vector is along the velocity vector. We assume tho bumper ends are identical and contacting the obstacle at the same instance.Therefore, in the preliminary study symmetrical conditions are assumed.

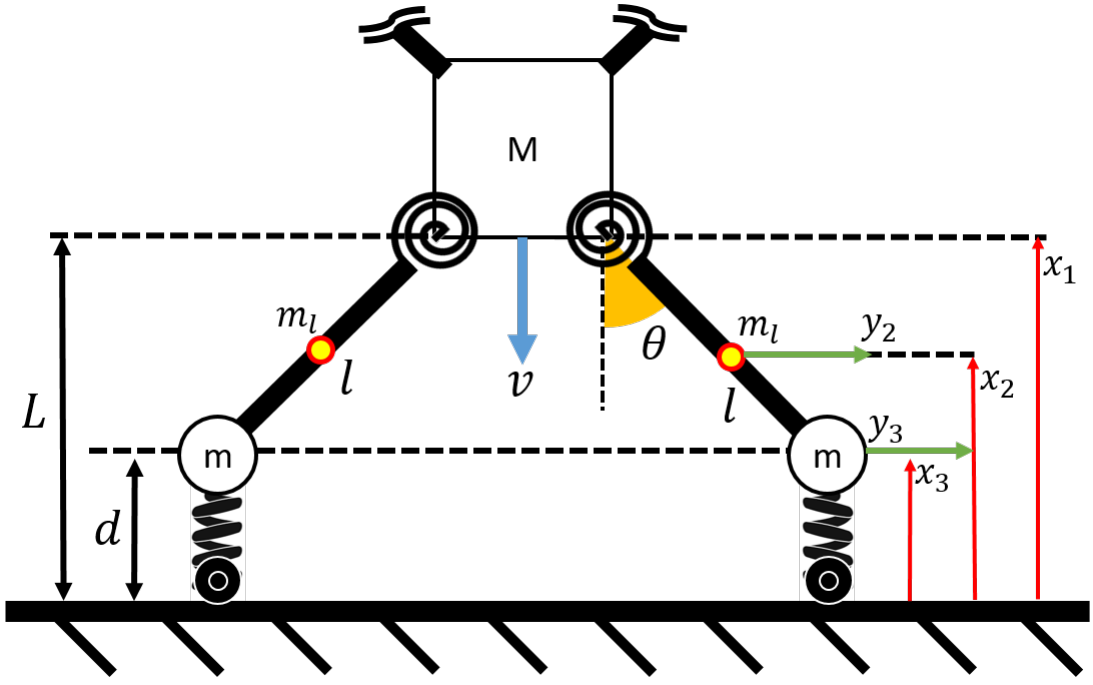

Figure 3.16: Structural model of the quadrotor in prescribed collision conditions.

The compliance of frame is considered analogous to spring. It retains the deformation energy with small amount of losses due to the characteristics of the plastic. It is possible to implement dampers but for the first study it is not considered. Collapsible arms can be modeled as a torsional springs complemented

with beam equation to obtain the spring constant. Bumpers as well as other structural elements have been assumed to have negligible mass and inertia. It is possible to dissect each masses individually to obtain equations of motion and internal forces.

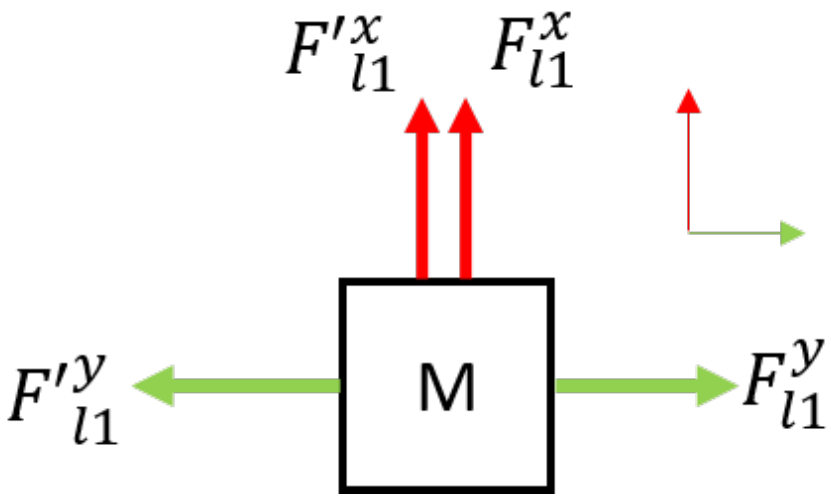

Figure 3.17: Free body diagram of the forces acting on the mass hinge point.

F_{l1}s are the link forces. The forces F^y shown on the figure 3.17 are equal due to symmetry. Therefore we can write through force balance in the x direction as;

$$\begin{aligned} M\ddot{x}_1 &= F^x_{l1} + F'^x_{l1} \\ M\ddot{x}_1 &= 2F^x_{l1} \end{aligned} \tag{3.1}$$

Next, equations of motion for the link can be written with the help of the figure 3.18 as;

$$m_l\ddot{x}_2 = -F^x_{l1} + F^x_{l2} \tag{3.2}$$

$$m_l\ddot{y}_2 = -F^y_{l1} - F^y_{l2} \tag{3.3}$$

$$I_l\ddot{\theta} = \tau_{ts} + F^y_{l2}lcos\theta - F^x_{l2}lsin\theta \tag{3.4}$$

Figure 3.19 shows the forces acting upon the motor mount point. F_b is the spring force generated by the bumpers around the motors. In reality bumper has

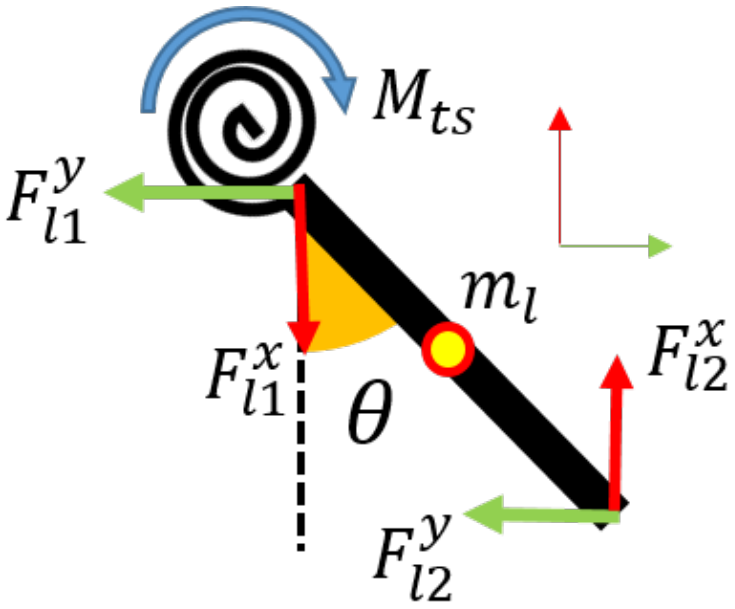

Figure 3.18: Free body diagram of the forces acting on the mass hinge point.

a complex spring characteristics which is directional and non-linear depending on the impact angle. In this model it has been assumed linear. Friction is an important factor in the distribution of the impact energy. As the friction increases, impact energy will not be expanded on the torsional spring. This will, consequently, increase the reliance on bumpers. In the real application thanks to the PET film sheets, friction between common wall surfaces and the bumper is very small and therefore is neglected for this study.

So we can write the equation of motion for the motor mount as:

$$m_m \ddot{x_3} = F_b - F_{l2}^x \tag{3.5}$$

$$m_m \ddot{y_3} = F_{l2}^y \tag{3.6}$$

where:

$$\begin{aligned} F_b &= k_b x \\ \tau_{ts} &= k_{ts} \theta \end{aligned} \tag{3.7}$$

where k_{ts} can be approximated from using Euler–Bernoulli beam equation for the bending of the arms by assuming the interface between the motor and the main body is cantilevered while the moment is calculated at the fixed point. By assuming small angle approach, we can assume:

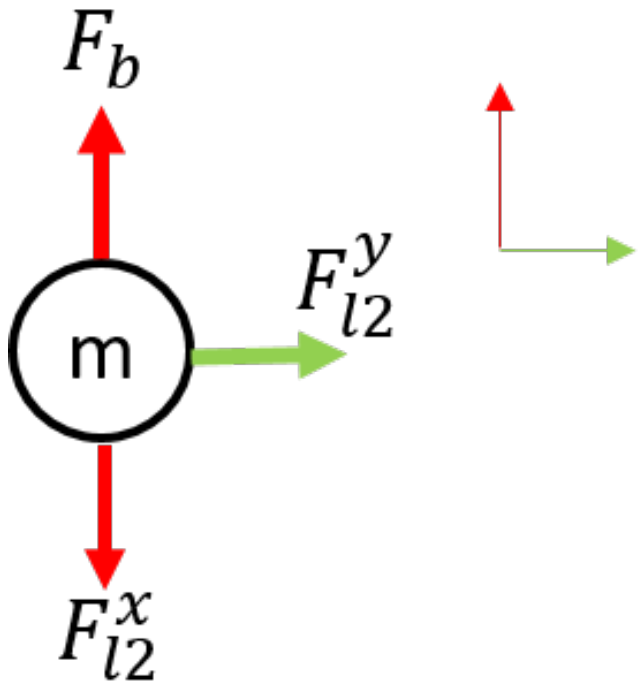

Figure 3.19: Free body diagram of the forces acting on the mass hinge point.

$$M = \frac{2EI}{L^2}\theta \tag{3.8}$$

Where E is the Young's modulus, I is the second moment of inertia of the beam, and L is the length of the beam. By analogy to rotational spring system, we can deduce:

$$\begin{aligned} \tau &= k_t s\theta \\ k_t s &= \frac{2EI}{L^2} \end{aligned} \tag{3.9}$$

We can write through geometrical relations, kinematic equations which reduces the equations into 2 degrees of freedom, x_3 and θ.

$$x_1 = x_3 + lcos\theta \tag{3.10}$$

$$x_2 = x_3 + \frac{l}{2}cos\theta \tag{3.11}$$

$$y_2 = \frac{l}{2} sin\theta \tag{3.12}$$

$$y_3 = l sin\theta \tag{3.13}$$

Then, we can take the second derivatives of the kinematic relations;

$$\begin{aligned} \ddot{x_1} &= \ddot{x_3} - l(cos\theta\dot{\theta}^2 + sin\theta\ddot{\theta}) \\ \ddot{x_2} &= \ddot{x_3} - \frac{l}{2}(cos\theta\dot{\theta}^2 + sin\theta\ddot{\theta}) \\ \ddot{y_2} &= \frac{l}{2}(-sin\theta\dot{\theta}^2 + cos\theta\ddot{\theta}) \\ \ddot{y_3} &= l(-sin\theta\dot{\theta}^2 + cos\theta\ddot{\theta}) \end{aligned} \tag{3.14}$$

Using these equations we can find the link forces by inserting them into the equations of motion:

$$\alpha = cos\theta\dot{\theta}^2 + sin\theta\ddot{\theta} \quad \beta = -sin\theta\dot{\theta}^2 + cos\theta\ddot{\theta} \tag{3.15}$$

$$\begin{aligned} F_{l1}^x &= \frac{M}{2}(\ddot{x_3} - \alpha l) \\ F_{l2}^x &= m_l(\ddot{x_3} - \alpha\frac{l}{2}) + F_{l1}^x \\ F_{l2}^y &= m_l l\beta \\ F_{l1}^y &= -F_{l2}^y - m_l\beta\frac{l}{2} \end{aligned} \tag{3.16}$$

Using the link force equations, we obtain two ordinary differential equations of x_3 and θ as:

$$m_m\ddot{x_3} = k_b(x_3^0 - x_3) - m_l(\ddot{x_3} + \beta\frac{l}{2}) - \frac{M}{2}(\ddot{x_3} + \alpha l) \tag{3.17}$$

$$I\ddot{\theta} = k_t s(\theta^0 - \theta) - m_l l sin\theta(\ddot{x_3} + \frac{l}{2}\beta) + m_m\alpha l cos\theta \tag{3.18}$$

We can impose conditions just before contact as;

$$\begin{aligned} x_3^0 &= 0 \quad \dot{x_3^0} = v \\ \theta_0 &= \frac{\pi}{4} \quad \dot{\theta_0} = 0 \end{aligned} \tag{3.19}$$

We can use a non-linear ODE solver to solve for x_3 and *theta*. Inner forces and component accelerations can be calculated to observe the material behavior

during impact. The resulting dynamic model will be then utilized to obtain the impact force by calculating the deformation time and the peak acceleration.

This part of the model has been left as a future work solely focused on the impact force calculation. The future work will include more accurate spring model for the bumpers as they have been modeled linearly in this work. The bumpers can be modeled as a proving ring where the elastic region has more degrees of freedom compared to leaf spring model.

Chapter 4

Control and Software Implementation

In this chapter, attitude stabilization and overall control layout will be explained. Additionally command structure and operation of the code on embedded controller firmware will be discussed.

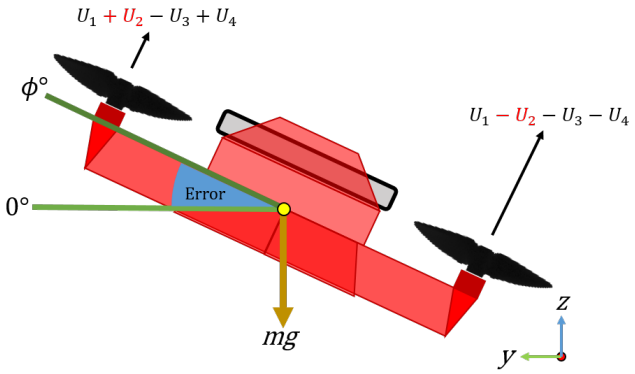

Figure 4.1: Roll Attitude control example.

4.1 Controller Scheme

As discussed before, quadrotors require controller scheme to compensate for dynamic inconsistencies and to follow commands of the human operator or the autonomous algorithms. This requires co-operation of efficient algorithm and essential electronic hardware. In general, controllers are implemented using microprocessing units. Therefore, signal processing and control techniques must be employed together. In this chapter the control scheme will be shown however will not be explained in full detail. The reader is encouraged to check [44] [13][45][46].

Proportional, integral , derivative (PID) type controllers are widely employed in the literature and industry. It offers comparatively easier implementation and intuitive approach to control system design.

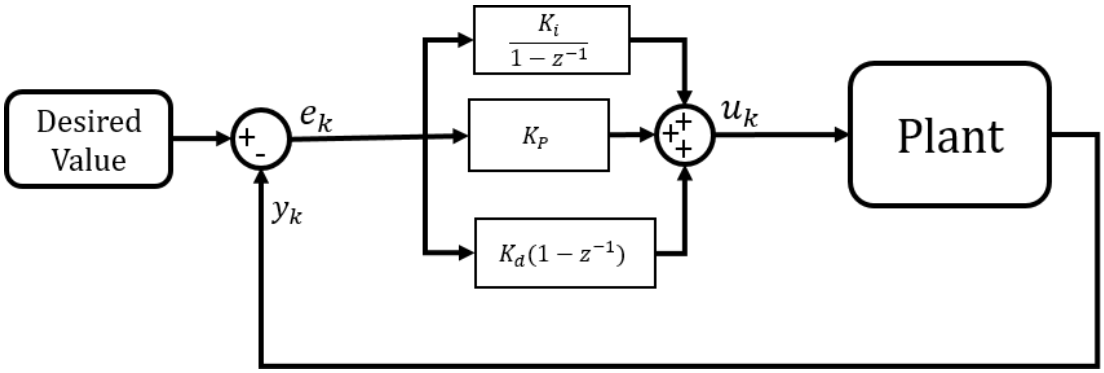

Figure 4.2: Basic discrete PID controller scheme z-domain.

Presence of the digital microprocessor necessitates the use of digital controllers. Thus,the PID controller must be discretized to be implemented on microprocessors. The designed quad-rotor employs P-PID loop for attitude stabilization. P-PID loop is a cascade controller type. Inner loop controls the angular velocity -angle rate- while the outer loop controls the attitude at the given axis. For each controlled rotation axis separate control loops are used. This controller type offers robust angle adjustment. However, this results in un intended sways in x and y axes if adjustments are not provided by the user. To eliminate this problem, a position feedback loop is required. This application is not implemented on this project.

A good approach to smoother operation can be obtained by rate only control. A PID loop controls the angular velocity with inputs from the user. Rate controller gives the user better control over the motion, however, it adds more workload on the operator as small adjustments are required to cancel every given input, normally made by the microprocessor on the P-PID loop. Practically, Rate loop requires more training to master, thus it is easier to use P-PID loop for more casual flight.

Yaw control differs from roll and pitch control loops. Yaw angle holding is problematic due to the gyro drift of the yaw axis. Effect of gyro drift is most apparent in the yaw axis because, DMP filter accounts for the gravity which is parallel to the yaw axis in hover mode, therefore, it is harder to estimate. In some controllers magnetometer is implemented to provide more reliable reference for yaw angle. In the absence of accurate angle feedback, it is usually preferable to control the angle rate. In any case however, user input, is treated as a rate controller input to adjust yaw rate. Ergonomically, it is more intuitive to control yaw rate of the quad than to control the angle unlike roll and pitch.

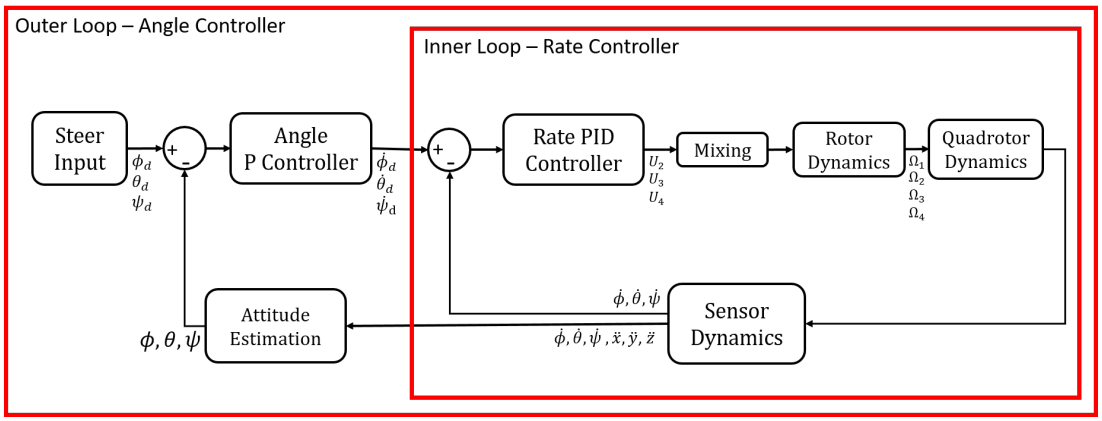

Figure 4.3: P-PID control loop used for attitude stabilization. Inner loop forms the basis for Rate PID controller.

4.2 Software Implementation

4.2.1 Flight Controller

The final version of foldable quad-rotor has been equipped with Betaflight 4.0 firmware for STM32 F4 type Crazybee F4 Pro flight controller. BetaFlight employs Rate PID with options to use P-PID loop. Alternatively, a firmware has been constructed to be used in Arduino compatible chips, mainly for the chip introduced in 3.3.1. Full code has been provided in Appendix A. Arduino compared to standard C language has easier abstraction suitable embedded applications for non-expert coders. It comes with variety of libraries for external protocols, hardware and sensors.

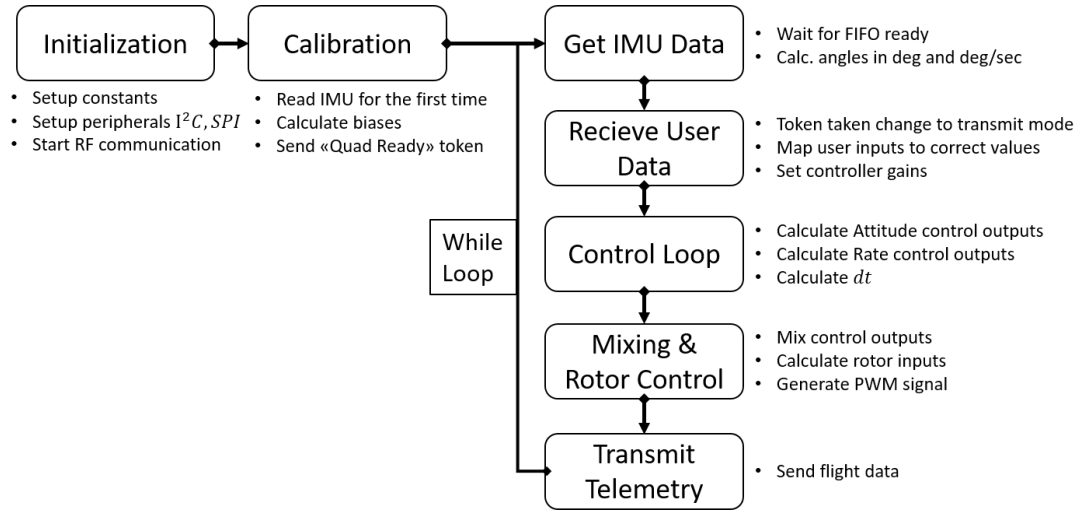

Figure 4.4: Code flow of the Quad-rotor

The code consists of five main subroutines during the operation given in fig. 4.4. Namely, gyro data acquisition, calibration, nRF24 data reception and acknowledgment telemetry data transmission, control loop and rotor mixing and control.

MPU6050 IMU includes an integrated digital motion processing (DMP) unit which can be programmed through main CPU to calculate attitude estimations. DMP programming library has been provided by Jeff Rowberg in [47]. Code

executes subroutines to program DMP unit which provides accurate attitude estimations, yaw,pitch and roll. Shortcoming of the MPU6050 and the software is the low gyro refresh rate of 285 kHz. DMP generates 42 byte data which includes quaternion information, body angles, angle rates and accelerometer data in 3 axis. The data is sent to First In First Out (FIFO) buffer volatile memory where the gyro data is transmitted through 400 kHz I^2C communication speed which result in 0.004 second loop time.

Calibration is important for angle stabilized control loops. Small deviations in the IMU positioning cause the attitude to be stabilized to a bias angle. Therefore before each flight a calibration scheme is conducted to calculate biases to zero the angle on each rotation axis. This is done by averaging the gyro data for a specified time on a level surface. Calculated biases are included in the error calculation. Similarly, gyro chip is affected by ambient conditions and therefore need to be calibrated. This done in every flight before the zeroing calculation.

User input is transmitted through 2.4GHz wireless RF transmitter/receiver module nRF24. nRF24 is connected to the main CPU via SPI communication protocol. SPI library is included Arduino firmware to boot peripheral systems. SPI allows for low power fast data transmission of 32 bytes of data. The module is communicating with another nRF24 attached to an remote controller joystick which supplies the user input. It is possible to connect the remote controller to a PC in order to receive telemetry data via UART serial connection. The communication between two nRF24 modules is a half-duplex protocol, meaning it cannot transmit and receive at the same time, thus, a token algorithm is employed to synchronize transmission direction.

RF module has two modes, transmit and receive. Initially, quadrotor side of communication is given a token to transmit. The transmission is held back until the initialization and calibration of the board is completed. The token is then transferred to the remote controller. The quadrotor is put to receive mode and RC is put to transmit mode after transmission. When the token is received for the first time by RC, it is put on hold if the thrust lever is not in the zero thrust position to prevent sudden motor action. When the safety conditions are met,

RC side transmits user inputs. These inputs are thrust, yaw ,pitch, roll and controller gains. Receiving live controller gains from the remote controller allows for in flight tuning. Quadrotor immediately sends the flight parameters to the RC side when the token is received inside the user data.

Control loop is a P-PID loop attitude stabilization scheme explained in 4.1. The loop takes 8-bit user angle and thrust values and maps them into corrected range. Since Atmel chip is limited to 8-bit PWM resolution, maximum rotor output value of %100 duty cycle is 255. To achieve thrust to weight ratio of 2, maximum thrust input is set to %60 duty cycle which is 150. The angle inputs for pitch and roll are limited to $\pm 10^{\circ}$ bank angle in which it is safer to operate and minimize non-linear effects. Yaw input is limited to $\pm 20^{\circ}/sec$ rotation rate. Using the processed user inputs and gyro data from DMP, error value is generated and corrected using the calibrated bias values. The error signal is then processed through attitude controller gain for pitch and roll, and fed into the rate PID controller. To prevent I-term overflow in the presence of the saturation , Back Calculation anti-windup scheme is implemented.

Outputs of the controller are U_1, U_2, U_3, U_4. Controller outputs are translated into rotor PWM commands by mixing equation:

$$\begin{bmatrix} PWM_1 \\ PWM_2 \\ PWM_3 \\ PWM_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & -1 & 0 & -1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{4.1}$$

The mixing matrix is dependent on the configuration of the body and the gyro axes. The matrix given here is the correct control mapping for the configuration in the figure 2.2. Resulting motor PWM values are then saturated to prevent data overflow in 8-bit to be in between 0 and 255. The PWM signal is sent to the transistors to drive the rotors.

Last step of the loop is transmission of telemetry data. It consists of vital information on flight and software status to give insight to the researcher. Data includes Angles and their rates, rotor PWM values, error and control signals.

4.2.2 Remote Controller

The RC controller complements the Atmel based flight controller by transmitting and receiving data. In addition to that, it allows for computer based operation via serial communication. This ability can be utilized with MatLAB and Simulink interface to process and replicate flight data as well as sending steer commands to quadrotor through the nRF24 module. It has a generic 4 channel joystick setup to allow for manual control.

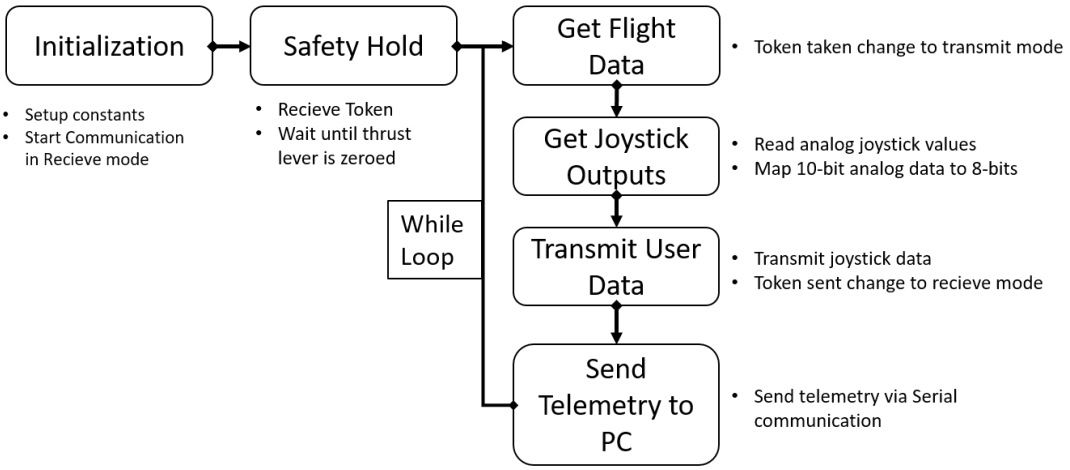

Figure 4.5: Code flow of the Remote Controller

Chapter 5

Results & Conclusion

In this chapter, the culmination of the work conducted will be presented in the light of details in other chapters with comments which may prove to be useful in the future work.

5.1 Collision tests

Collision tests are conducted to verify the survivability of the quad-rotor under collision stresses. First test is a free fall test on the bumpers from height of 1.5 meters. It can be seen that bumpers are guiding the arms to deform in the pre-planned configuration to avoid expansion of collision energy on critical parts. Stored energy then recalls the arms to their original configuration all the while pushing against the ground. The test shows the flexible behavior of the quad-rotor during impacts without any failure event.

Figure 5.1 shows the flexibility of arms under the collision stresses. Maximum strain of the arms has been achieved at $t = 0.56s$ which is geometrically constrained. The remaining energy is expanded on the body enclosure. Propellers make a contact with the collision surface due to flexure of This can be further improved by increasing the spring constant of the bumpers so that the collision

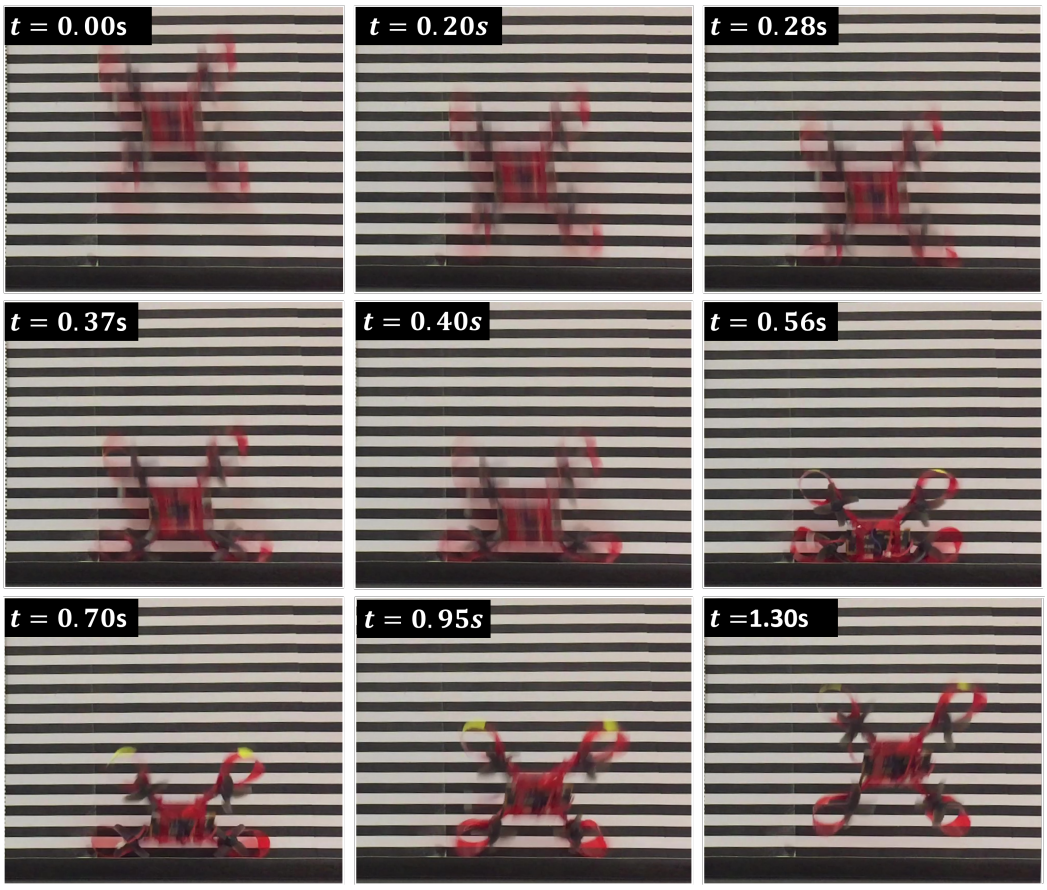

Figure 5.1: Drop test from 1.5 meters, width of the stripes is 1cm.

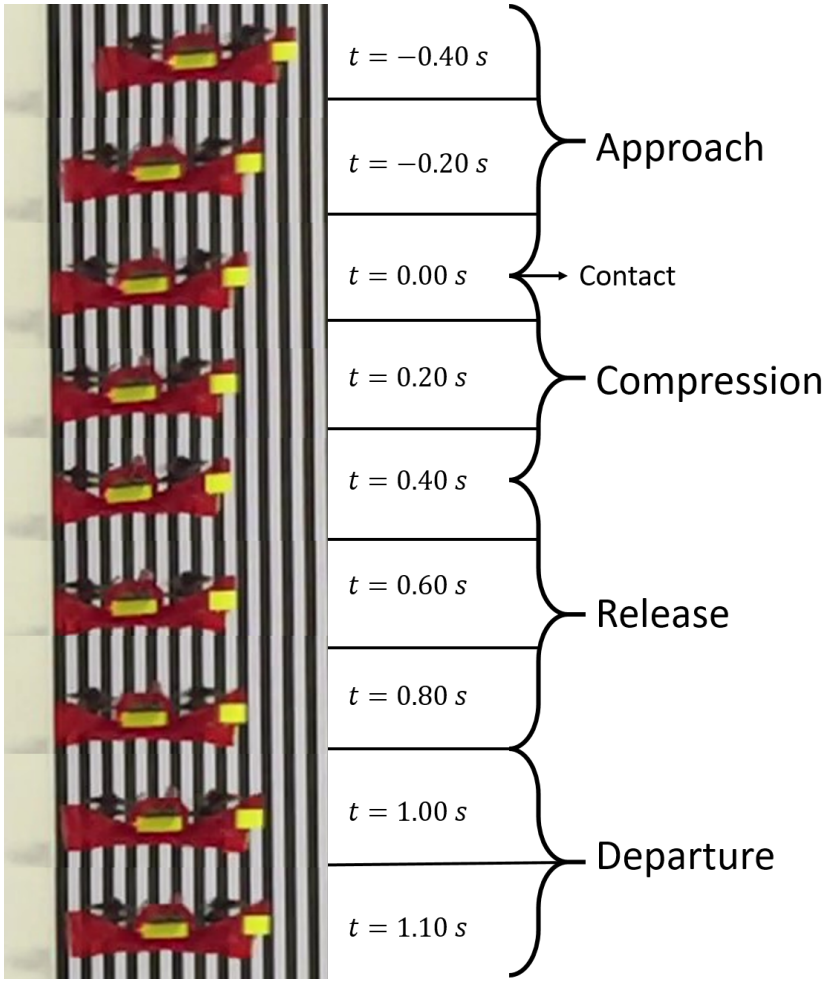

Figure 5.2: Powered collision with -5° pitch angle , width of the stripes is 1cm.

strain is restricted to the expandable bumpers.

Figure 5.2 is an in-flight test with the objective of examining collisions during flight. It is expected of the quadrotor continue operation after the collision and depart from the object within manageable attitude limits. Test suggest under low speed ($1m/s$) conditions with -5° pitch angle which is within the limits of normal attitude conditions of near-hover operations like observation and inspection[48]. Bumpers exhibit adequate compression to block propeller collision and to push the frame away from the obstacle. The design will be modified in the future work to exhibit the same behavior under medium speed and high attitude angles.

Figure 5.3: 3D printed frame quadrotor with PMFC, coreless motors and final version of the PCB.

5.2 Flight Controller Tests

Purpose made flight controller has been demonstrated using a 3D printed body frame. Hardware subsystems, ATMEGA328P-AU, rotor control, communication suit and sensors have been tested and observed to be functioning properly.

However, resulting IMU refresh rate of $285Hz$ has not been have proven to be unsatisfactory for classical control strategies. This is mainly caused by the low frequency feedback obtained from the MPU6050 with the I^2C communication. Another contributing factor is the 8-bit PWM resolution which undermines the accuracy provided by the controller.

Some form of stability has been salvaged without the yaw control, however, the slow reaction of the controller results in a violent pitch and/or bank angle, increasing the speed, and making positional stability unfeasible. The MPU6000 series with SPI communication combined with a faster CPU such as STM32 may improve the controller efficiency of the quad-rotor. Another possibility is to use delay resistant controllers with attitude observers to calculate an estimation in between the real gyro readings to increase the control loop speed.

5.3 Quadrotor Performance

Matured design is a 41.3 grams brushless motor foldable micro quad-rotor with dimensions of $10x10x4cm$ without the bumpers. The quad-rotor is able to lift up to 45 grams of additional weight and function which corresponds to 2.1+ Thrust/Weight ratio. With the current unloaded configuration and sustained stable hovering, flight time is measured to be around 4.5 minutes. Flight time can be extended further with larger capacity batteries with high power-to-weight ratio.

Figure 5.4: Fol-Quad (foldable quadrotor) during flight

Foldable body provides collision resilience as an alternative to traditional quadrotor design with additional benefits of easy assembly, high availability and low cost. Origami inspired design introduce modularity and adaptability, thanks to easy prototyping. Assembly of the quadrotor requires 10 minutes to be engraved from A4 PET film, 30 minutes to fold and 15 minutes to integrate with electronics and it is made ready to fly in total of 45 minutes.

Final design employs the Crazybee F4 Pro flight controller with Flysky receiver setup. The controller used on the final design is a rate controller loop explained in section 4.1. The foldable body provides durability and stability in flight. Combined with the bumpers collision energy is absorbed to protects rotors and electronics in low speed collisions with low bank attitude. Higher collision and flip-over protection can be obtained with a cage-like design which is left for

Figure 5.5: Fol-Quad air-lifting anothe foldable robot Vasler (35 grams).

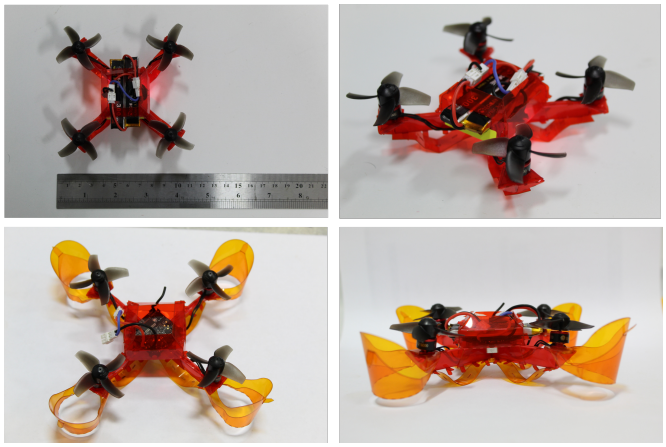

Figure 5.6: Final verison of the foldable quadrotor seen from different angles.

the future work.

In sustained high speed collisions, bumpers tend to deform or snap due to accumulation of fatigue cracks. This can be solved by using superior materials like Kapton which has impressive fatigue characteristics, but, is expensive and lower strength than the PET film. Alternative solution is forming a composite material of PET film covered by low-cost polypropylene bands to increase elasticity. ,

Although, electronics are functionally suitable, practically, low build quality and faulty components presents unreliable sustained operation. Brushless motors

require close monitoring by the user for any sign of temperature increase or magnetic polarity decay. this may result in total loss of the rotor if left unattended.

5.4 Conclusion & Future Work

This thesis presents the practical approach conducted to design of a quad-rotor. The study focused on understanding the underlying implementation requirements and address problems currently seen on available drone and implementation of a quadrotor to build design experience. A compliant foldable quadrotor has been designed and produced with origami inspired design techniques that offer low-cost, high availability and molecularity. Weight reduction thanks to low density materials benefit to the performance of the quadrotor compared to counterparts with molded plastic frame. An attempt has been made to produce a flight controller board and a complementary firmware which exposed shortcomings of the 8-bit micro-controllers in micro quadrotor flight. Quirks of the controller implementation have been discussed. More testing is required to formalize the collision dynamics, the resulting quadrotor is capable of handling the requirements needed in the future work. It presents an adaptable modular platform to increase its capability by implementing additional sensors and functions. Output of this work will be used in the future as a platform for a multi-robot formation research to inspect infrastructure. The quad-rotor will be equipped with additional sensors such as a camera and GPS module to navigate and to keep the formation flight. On that end, improvements will be made to increase electronic reliability of the quad-rotor. Collision resilience will be improved for higher velocity and higher pitch collisions. An elastic model will be further extended and simulated to explain the soft elastic behavior of the quad-rotor during collisions and flight. Additionally, system identification to obtain the parameters of the model will be employed to create an accurate dynamic model explained in this work to predict quadrotor behavior and simulate formation flight.

Bibliography

[1] S. Mintchev, S. de Rivaz, and D. Floreano, "Insect-inspired mechanical resilience for multicopters," *IEEE Robotics and automation letters*, vol. 2, no. 3, pp. 1248–1255, 2017.

[2] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano, "A collision-resilient flying robot," *Journal of Field Robotics*, vol. 31, no. 4, pp. 496–509, 2014.

[3] S. Mintchev, L. Daler, G. L'Eplattenier, L. Saint-Raymond, and D. Floreano, "Foldable and self-deployable pocket sized quadrotor," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2190–2195, IEEE, 2015.

[4] S. Mintchev and D. Floreano, "A pocket sized foldable quadcopter for situational awareness and reconnaissance," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 396–401, Ieee, 2016.

[5] L. Young, E. Aiken, V. Gulick, R. Mancinelli, and G. Briggs, "Rotorcraft as mars scouts," in *Proceedings, IEEE Aerospace Conference*, vol. 1, pp. 1–378, IEEE, 2002.

[6] C. Smith, *The photographer's guide to drones*. Rocky Nook, Inc., 2016.

[7] K. Fregene, "Unmanned aerial vehicles and control: Lockheed martin advanced technology laboratories," *IEEE Control Systems Magazine*, vol. 32, no. 5, pp. 32–34, 2012.

[8] J. G. Leishman, "The breguet-richet quad-rotor helicopter of 1907," *Vertiflite*, vol. 47, no. 3, pp. 58–60, 2002.

[9] P. Pounds, R. Mahony, P. Hynes, and J. M. Roberts, "Design of a four-rotor aerial robot," in *Proceedings of the 2002 Australasian Conference on Robotics and Automation (ACRA 2002)*, pp. 145–150, Australian Robotics & Automation Association, 2002.

[10] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quadrotor," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5, pp. 4393–4398, IEEE, 2004.

[11] P. Y. Oh, "Flying insect inspired vision for micro-air-vehicle navigation," 2004.

[12] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 153–158, Ieee, 2007.

[13] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2451–2456, IEEE, 2004.

[14] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA guidance, navigation and control conference and exhibit*, p. 6461, 2007.

[15] S. Bouabdallah and R. Siegwart, "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 2247–2252, IEEE, 2005.

[16] R. Xu and U. Ozguner, "Sliding mode control of a quadrotor helicopter," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 4957–4962, IEEE, 2006.

[17] S. Bouabdallah, P. Murrieri, and R. Siegwart, "Towards autonomous indoor micro vtol," *Autonomous robots*, vol. 18, no. 2, pp. 171–183, 2005.

[18] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, p. 7410, 2008.

[19] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, IEEE, 2011.

[20] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *2011 IEEE International Conference on Robotics and Automation*, pp. 763–770, IEEE, 2011.

[21] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *2010 IEEE international conference on robotics and automation*, pp. 1642–1648, IEEE, 2010.

[22] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *2011 IEEE/RSJ international conference on Intelligent Robots and Systems*, pp. 5113–5120, IEEE, 2011.

[23] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2011.

[24] G. Ducard and R. D'Andrea, "Autonomous quadrotor flight using a vision system and accommodating frames misalignment," in *2009 IEEE International Symposium on Industrial Embedded Systems*, pp. 261–264, IEEE, 2009.

[25] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[26] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4557–4564, IEEE, 2012.

[27] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC proceedings Volumes*, vol. 44, no. 1, pp. 1485–1491, 2011.

[28] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.

[29] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," *Autonomous Robots*, vol. 35, no. 4, pp. 287–300, 2013.

[30] E. A. Peraza-Hernandez, D. J. Hartl, R. J. Malak Jr, and D. C. Lagoudas, "Origami-inspired active structures: a synthesis and review," *Smart Materials and Structures*, vol. 23, no. 9, p. 094001, 2014.

[31] A. T. Baisch and R. J. Wood, "Design and fabrication of the harvard ambulatory micro-robot," in *Robotics Research*, pp. 715–730, Springer, 2011.

[32] A. T. Baisch, O. Ozcan, B. Goldberg, D. Ithier, and R. J. Wood, "High speed locomotion for a quadrupedal microrobot," *The International Journal of Robotics Research*, vol. 33, no. 8, pp. 1063–1082, 2014.

[33] S. Miyashita, S. Guitron, M. Ludersdorfer, C. R. Sung, and D. Rus, "An untethered miniature origami robot that self-folds, walks, swims, and degrades," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1490–1496, IEEE, 2015.

[34] S. M. Felton, M. T. Tolley, C. D. Onal, D. Rus, and R. J. Wood, "Robot self-assembly by folding: A printed inchworm robot," in *2013 IEEE International Conference on Robotics and Automation*, pp. 277–282, IEEE, 2013.

[35] X. Sun, S. M. Felton, R. Niiyama, R. J. Wood, and S. Kim, "Self-folding and self-actuating robots: A pneumatic approach," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3160–3165, IEEE, 2015.

[36] P. Sareh, P. Chermprayong, M. Emmanuelli, H. Nadeem, and M. Kovac, "Rotorigami: A rotary origami protective system for robotic rotorcraft," *Science Robotics*, vol. 3, no. 22, p. eaah5228, 2018.

[37] M. Gäfvert, *Modelling of the eth helicopter laboratory process*. Department of Automatic Control, Lund Institute of Technology (LTH), 2016.

[38] S. Bouabdallah and R. Siegwart, "Design and control of a miniature quadrotor," in *Advances in unmanned aerial vehicles*, pp. 171–210, Springer, 2007.

[39] P. Pillay and R. Krishnan, "Modeling, simulation, and analysis of permanent-magnet motor drives. ii. the brushless dc motor drive," *IEEE transactions on Industry applications*, vol. 25, no. 2, pp. 274–279, 1989.

[40] P. Yedamale, "Brushless dc (bldc) motor fundamentals," *Microchip Technology Inc*, vol. 20, pp. 3–15, 2003.

[41] R. Condit, "Brushed dc motor fundamentals," *Microchip Technology Inc, http://ww1. microchip. com/downloads/en/AppNotes/00905a. pdf*, 2004.

[42] C. Karakadıoğlu, M. Askari, and O. Özcan, "Design and operation of miniaq: An untethered foldable miniature quadruped with individually actuated legs," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 247–252, IEEE, 2017.

[43] M. Askari, *Design, control, modeling, and gait analysis in miniature foldable robotics*. PhD thesis, Bilkent University, 2018.

[44] C. H. Houpis and G. B. Lamont, *Digital control systems*. McGraw-Hill Higher Education, 1991.

[45] G. Bo, L. Xin, Z. Hui, and W. Ling, "Quadrotor helicopter attitude control using cascade pid," in *2016 Chinese Control and Decision Conference (CCDC)*, pp. 5158–5163, IEEE, 2016.

[46] H. S. Khan and M. B. Kadri, "Position control of quadrotor by embedded pid control with hardware in loop simulation," in *17th IEEE International Multi Topic Conference 2014*, pp. 395–400, IEEE, 2014.

[47] J. Rowberg, "I2cdevlib. mpu-6050 6-axis accelerometer/gyroscope," 2013.

[48] I. Sa and P. Corke, "Vertical infrastructure inspection using a quadcopter and shared autonomy control," in *Field and service robotics*, pp. 219–232, Springer, 2014.

Appendix A

Flight Code for the Purpose Made Flight Controller

```
/////////MINIQUAD v0.7 BDC MOD//////////

#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
#include <SPI.h>
#include <RF24.h>
#include <avr/wdt.h>
#include <digcomp.h>
#include "FlySkyIBus.h"

MPU6050 mpu;

//                    front
//               1            2
//                *    +X    *
//                   *    *
//                      *         +Y  From Top
//                   *    *
//                *           *
//               4            3

//FLIGHT MODES
//int mode=1;

//MOTOR PINS
const int Rotor_1=3;
const int Rotor_2=5;
const int Rotor_3=6;
const int Rotor_4=9;

//MOTOR PWMs
uint8_t  Rotor1_pwm  =0;
uint8_t  Rotor3_pwm  =0;
uint8_t  Rotor2_pwm  =0;
uint8_t  Rotor4_pwm  =0;

#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
#define LED_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)

//CHOOSE A CONTROLLER ////Choose just ONE!!!
#define nRF_controller  //For nRF Equipped Transmitter Setup
//#define FS_controller //For FlySky Transmitter Setup

#define manualTune
//#define constTune

//PRINTING VALUES
#define looptime; //  for the looptime value
#define pwm;    //  for motor pwm values
#define cont;   //  for control signal values
#define angles; //  for yaw pitch roll values
#define err;    //  for error signals
#define cont_rate;
//#define datas;

//NRF24 DATA TO SEND
//We define each byte of data input,
//up to 32 channels(currently 4)

struct MyData {
byte throttle;
byte yaw;
byte pitch;
byte roll;
byte newCalibrate; //for Calibration command
byte Kp_y=0;
byte Kp_p=0;
byte Kp_r=0;

byte Kp_y_rate=0;
byte Kp_p_rate=0;
byte Kp_r_rate=0;

byte Ki_y_rate=0;
byte Ki_p_rate=0;
byte Ki_r_rate=0;

byte Kd_y_rate=0;
byte Kd_p_rate=0;
byte Kd_r_rate=0;
};
MyData data;

struct telData {
#ifdef angles
byte angles_signs=0b00000000;
byte throttle;
byte yaw;
byte pitch;
byte roll;
#endif
#ifdef looptime
byte loopt;
#endif
#ifdef err
byte err_signs=0b00000000;
byte errRoll;
byte errPitch;
byte errYaw_rate;
byte errRoll_rate;
byte errPitch_rate;
#endif
#ifdef cont
byte cont_signs=0b00000000;;
byte U1;
byte U2;
byte U3;
byte U4;
#endif
#ifdef cont_rate
byte cont_rate_signs=0b00000000;
byte U2_rate;
byte U3_rate;
byte U4_rate;
#endif
#ifdef pwm
byte Rotor1pwm;
byte Rotor2pwm;
byte Rotor3pwm;
byte Rotor4pwm;
#endif

#ifdef datas
byte data_signs=0b00000000;
byte data[4];
#endif

};
telData telemetry_send;
//RADIO PIPE
const uint64_t pipeIn = 0xE8E8F0F0E1LL;
//Remember that this code is the same as in the transmitter
```

```
RF24 radio(7, 10); //RADIO PINS

unsigned long lastRecvTime = 0;


//Filter variables

float a_lp[2]={1.00000000, -0.72848950};
float b_lp[2]={0.13575525, 0.13575525};

float gyro_lp_in[2];
float gyro_lp_out[2];

dig_comp gyro_lpfilt(b_lp,a_lp,gyro_lp_in,gyro_lp_out,2,2);
//Shared Control Variables

float Kp_ypr[3];

float Kp_gyro[3];
float Ki_gyro[3];
float Kd_gyro[3];
float gain_scaler=10.00;

float target[3];
float error_gyro[3]={0,0,0};
float error_ypr[3]={0,0,0};
float lasterror_gyro[3]={0,0,0};
float lasterror_ypr[3]={0,0,0};
float yaw_prev=0;

float iTerm_gyro[3]={0,0,0};
float dTerm_gyro[3]={0,0,0};
//float dTerm_gyro_prev[3]={0,0,0};
int16_t U1;

float U_ypr[3]={0,0,0};
float U_Gyro[3]={0,0,0};

float U_Gyro_prev[3]={0,0,0};
float Us_Gyro_prev[3]={0,0,0};

uint8_t mix_t=50;
uint8_t mix_y=50;
uint8_t mix_p=50;
uint8_t mix_r=50;

float dt=0;
float lastTime=0;
//unsigned long currentTime=0;
//Calibration Targets
bool calibrate_flag=1;
int calibrate_counter=-1;
float derivative_scaling=1;

float cYaw=0;
float cRoll=0;
float cPitch=0;
float cYaw_rate =0;
float cRoll_rate=0;
float cPitch_rate=0;
float cal_ypr[3]={0,0,0};
float cal_gyro[3]={0,0,0};
float cal_target[3]={0,0,0};

// MPU control/status vars
bool dmpReady = false;
// set true if DMP init was successful
uint8_t mpuIntStatus;
// holds actual interrupt status byte from MPU
uint8_t devStatus;
 // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;
// expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;
// count of all bytes currently in FIFO
uint8_t fifoBuffer[64];
// FIFO storage buffer

// orientation/motion vars
Quaternion q;
// [w, x, y, z]         quaternion container
VectorInt16 aa;
 // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;
// [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;
// [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;
// [x, y, z]            gravity vector
float euler[3];
// [psi, theta, phi]    Euler angle container
float ypr[3];
// [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
int16_t gyro[3];

float Gyro[3];               // [x, y, z]            Gyro Vector
float Ypr[3];
float lastGyro[3]={0,0,0};

volatile bool mpuInterrupt = false;
// indicates whether MPU interrupt pin has gone high

int cntr=0;
// ================================================================
// ===                      INITIAL SETUP                       ===
// ================================================================

void setup() {
initialize();
//Peripherals, rotor output pins are initialized
init_mpu();
//Gyro connections are checked and initiated
init_nrf();
// SPI Bus activated
}
// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================

void loop() {
main_loop();
}
// ================================================================
// ===                  CODE INITIALIZATION                   ===
// ================================================================
void initialize(){

Serial.begin(115200);
//   Serial.println("MiniQuad v. 0.7-Brushed DC");
//   Serial.println(F("Initializing Comms and I2C devices..."));
// join I2C bus (I2Cdev library doesn't do this automatically)

Wire.begin();
Wire.setClock(400000);
```

```
pinMode(Rotor_1,OUTPUT);
pinMode(Rotor_2,OUTPUT);
pinMode(Rotor_3,OUTPUT);
pinMode(Rotor_4,OUTPUT);
//    pinMode(Set_button, OUTPUT);
//    pinMode(Power_button, OUTPUT);

wdt_enable(WDTO_8S);
}
// ================================================
// ===                    MPU INITIALIZATION                    ===
// ================================================
void init_mpu(){
//Serial.println(mpu.testConnection() ?
 F("MPU6050 connection successful") :...
   F("MPU6050 connection failed"));
//Serial.println("Initializing MPU6050");
mpu.initialize();
pinMode(INTERRUPT_PIN, INPUT);

devStatus = mpu.dmpInitialize();

//Gyro Offsets (scaled)
mpu.setXGyroOffset(46);
mpu.setYGyroOffset(6);
mpu.setZGyroOffset(36);
mpu.setZAccelOffset(2332);
// 1688 factory default for my test chip


if (devStatus == 0) {
// make sure it worked (returns 0 if so)
// turn on the DMP, now that it's ready
mpu.CalibrateAccel(6);
mpu.CalibrateGyro(6);
mpu.PrintActiveOffsets();
mpu.setDMPEnabled(true);

// enable Arduino interrupt detection
attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
mpuIntStatus = mpu.getIntStatus();

// set our DMP Ready flag so the main loop() function knows it's okay to use it
dmpReady = true;

// get expected DMP packet size for later comparison
packetSize = mpu.dmpGetFIFOPacketSize();
} else {
// ERROR!
// 1 = initial memory load failed
// 2 = DMP configuration updates failed
// (if it's going to break, usually the code will be 1)
}
//Serial.println("MPU6050 Online");
//pinMode(LED_PIN, OUTPUT);    // Configure LED for output
//Serial.println("Calibrating the IMU");
}
void init_nrf(){
//Radio
resetData();
radio.begin();
radio.setAutoAck(true);
radio.enableAckPayload();
radio.setDataRate(RF24_1MBPS);
radio.openReadingPipe(1,pipeIn);

//we start the radio comunication
radio.startListening();

IBus.begin(Serial);

}
void main_loop(){  // Mainly consists of DMP data acquisition Jeff Rowbergs code
if (!dmpReady) return;

// wait for MPU interrupt or extra packet(s) available
while (!mpuInterrupt && fifoCount < packetSize) {
if (mpuInterrupt && fifoCount < packetSize) {
// try to get out of the infinite loop
fifoCount = mpu.getFIFOCount();
}
// other program behavior stuff here
// .
// .
// .
// if you are really paranoid you can frequently test in between other
// stuff to see if mpuInterrupt is true, and if so, "break;" from the
// while() loop to immediately process the MPU data
// .
// .
// .
}

// reset interrupt flag and get INT_STATUS byte
mpuInterrupt = false;
mpuIntStatus = mpu.getIntStatus();

// get current FIFO count
fifoCount = mpu.getFIFOCount();
if(fifoCount < packetSize){
//Lets go back and wait for another interrupt.
// We shouldn't be here, we got an interrupt from another event
// This is blocking so don't do it
//while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
}
// check for overflow (this should never happen unless our code is too inefficient)
else if ((mpuIntStatus & _BV(MPU6050_INTERRUPT_FIFO_OFLOW_BIT)) || fifoCount >= 1024) {
// reset so we can continue cleanly
mpu.resetFIFO();
//  fifoCount = mpu.getFIFOCount();  // will be zero after reset no need to ask
//Serial.println(F("FIFO overflow!"));

// otherwise, check for DMP data ready interrupt (this should happen frequently)
} else if (mpuIntStatus & _BV(MPU6050_INTERRUPT_DMP_INT_BIT)) {

// read a packet from FIFO
while(fifoCount >= packetSize){ // Lets catch up to NOW,
//someone is using the dreaded delay()!
mpu.getFIFOBytes(fifoBuffer, packetSize);
// track FIFO count here in case there is > 1 packet available
// (this lets us immediately read
// more without waiting for an interrupt)
fifoCount -= packetSize;
}

// Get sensor data
mpu.dmpGetQuaternion(&q, fifoBuffer);
mpu.dmpGetGyro(gyro, fifoBuffer);
mpu.dmpGetGravity(&gravity, &q);
mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
////////// Rest of the Code Starts Here//////
getGyro(); //

#ifdef FS_controller // For uses with Flysky remote controller
IBus.loop();
#endif
```

```
if(calibrate_flag==1){//Calibrate at the start
telemetry();  //Send telemetry
calibrateTargets();//Calibration of tagets and angle bias
}
else{
if(data.newCalibrate==1 && U1<=2){
// If calibration command is sent by the ground control
calibrate_flag=1;
//Serial.println(F("Calibration Begins"));
resetCalibration();
}
else if(data.newCalibrate==2 && U1<=2){
resetCalibration();
}

in_outLoop();
//Main Control Loop
controlMixing();
//Control Mixing for X quad Configuration
rotor_control();
//Rotor pins are set to PWM values
dataPrint();
//Prepare telemetry data
telemetry();
//Send telemetry data
//dt=time_now-time_last;
wdt_reset();
// Watchdog timer if the code gets stuck
}

}
}
void thrust_hold(){
bool isQuadReady=0;
//Serial.println("Move the Throttle stick to 0");
while(!isQuadReady){
if ( IBus.readChannel(3)>500) {
radio.read( &data,sizeof(MyData) );
IBus.loop();
if( IBus.readChannel(2)==0) isQuadReady=1;
radio.writeAckPayload( 1, &isQuadReady, sizeof(bool) );
}
wdt_reset();
}
//Serial.println("Quad Ready!");
delay(1000);
}

void calibrateTargets(){
wdt_reset();
int deadzone=1300;
int sample=200;
calibrate_counter++;
if(calibrate_counter>deadzone){
//    Serial.println(++cntr);
cYaw_rate  += Gyro[0];
cYaw+= Ypr[0];
cPitch += Ypr[1];
cRoll += Ypr[2];
cPitch_rate += Gyro[1];
cRoll_rate += Gyro[2];
cal_target[0]+=(float)data.yaw * (10) / (255) -5;
cal_target[1]+=(float)data.pitch*20/255-10;
cal_target[2]+=(float)data.roll*20/255-10;
}
if(calibrate_counter==deadzone+sample){
calibrate_flag=0;
calibrate_counter=0;
cal_ypr[0]=cYaw/sample;
cal_ypr[1]=cPitch/sample;
cal_ypr[2]=cRoll/sample;
cal_gyro[0]=cYaw_rate/sample;
cal_gyro[1]=cPitch_rate/sample;
cal_gyro[2]=cRoll_rate/sample;
cal_target[0]=cal_target[0]/sample;
cal_target[1]=cal_target[1]/sample;
cal_target[2]=cal_target[2]/sample;

Serial.println(F("Calibration Done"));
Serial.print("Pitch\t");
Serial.print("Roll\t")
;Serial.print("Yaw_Gyro\t");
Serial.print("Pitch_Gyro\t");
Serial.print("Roll_Gyro\t");
Serial.print("Yaw Target\t");
Serial.print("Pitch Target\t");
Serial.println("Roll Target");
Serial.print(Ypr[1]);Serial.print("\t");Serial.print(Ypr[2]);
Serial.print("\t");
Serial.print(Gyro[0]);Serial.print("\t");Serial.print(Gyro[1]);
Serial.print("\t");
Serial.print(Gyro[2]);Serial.print("\t");
Serial.print(-data.yaw * (10) / (255) -5);S
erial.print("\t");Serial.print((float)data.pitch*20/255-10);
Serial.print("\t");
Serial.println((float)data.roll*20/255-10);
Serial.print(cal_ypr[1]);Serial.print("\t");
Serial.print(cal_ypr[2]);Serial.print("\t");
Serial.print(cal_gyro[0]);Serial.print("\t");
Serial.print(cal_gyro[1]);Serial.print("\t");
Serial.print(cal_gyro[2]);Serial.print("\t");
Serial.print(cal_target[0]);Serial.print("\t");
Serial.print(cal_target[1]);Serial.print("\t");
Serial.println(cal_target[2]);
delay(1000);
//thrust_hold();
}
}

void resetCalibration(){
cYaw_rate  =0;
cYaw=0;
cPitch  =0;
cRoll =0;
cPitch_rate =0;
cRoll_rate =0;
cal_target[0] =0;
cal_target[1] =0;
cal_target[2] =0;
}

void in_outLoop(){

#ifdef nRF_controller
in_outLoop_nRF();
#endif
#ifdef FS_controller
in_outLoop_FS();
#endif
}

void in_outLoop_nRF(){
getGains();
U1=map(data.throttle,0,255,0,150);
target[0]=(float)-(((data.yaw) * (10) / (255)...
```

```
 -5-cal_target[0]+0.18)/5)*10;
target[1]=(float)data.pitch*20/255-10-cal_target[1];
target[2]=(float)data.roll*20/255-10-cal_target[2];
dt=(millis()-lastTime)/1000;
for(int i=0;i<3;i++){
if(i==0) U_ypr[i]=target[i];
else{
error_ypr[i]=target[i]-Ypr[i]+cal_ypr[i];
U_ypr[i]=Kp_ypr[i]*error_ypr[i];
}
if(i==0) error_gyro[i]=U_ypr[i]-Gyro[i];
else error_gyro[i]=U_ypr[i]-Gyro[i]+cal_gyro[i];

iTerm_gyro[i]+=Ki_gyro[i]*error_gyro[i]*dt...
+(Us_Gyro_prev[i]-U_Gyro_prev[i]);
dTerm_gyro[i]=(-Gyro[i]+lastGyro[i])/dt;
U_Gyro[i]=Kp_gyro[i]*error_gyro[i]...
+iTerm_gyro[i]+Kd_gyro[i]*dTerm_gyro[i];
lasterror_gyro[i]=error_gyro[i];

U_Gyro_prev[i]=U_Gyro[i];
U_Gyro[i]=constrain(U_Gyro[i],-100,100);
Us_Gyro_prev[i]=U_Gyro[i];

lastGyro[i]=Gyro[i];
}
lastTime=millis();
}
#ifdef FS_controller
void in_outLoop_FS(){
U1=IBus.readChannel(2)/6.666-150;
target[0]=IBus.readChannel(3)/100-15;
target[1]=(IBus.readChannel(1)/50-30)*(-1);
target[2]=IBus.readChannel(0)/50+30;
dt=(millis()-lastTime)/1000;
for(int i=0;i<3;i++){
if(i==0) U_ypr[i]=target[i];
else{
error_ypr[i]=target[i]-Ypr[i]+cal_ypr[i];
U_ypr[i]=Kp_ypr[i]*error_ypr[i];
}

error_gyro[i]=U_ypr[i]-Gyro[i]+cal_gyro[i];
iTerm_gyro[i]+=Ki_gyro[i]*error_gyro[i]...
*dt+(Us_Gyro_prev[i]-U_Gyro_prev[i]);
dTerm_gyro[i]=Kd_gyro[i]*(error_gyro[i]...
-lasterror_gyro[i])/dt;
U_Gyro[i]=Kp_gyro[i]*error_gyro[i]...
+iTerm_gyro[i]+dTerm_gyro[i];

lasterror_gyro[i]=error_gyro[i];

U_Gyro_prev[i]=U_Gyro[i];
U_Gyro[i]=constrain(U_Gyro[i],-100,100);
Us_Gyro_prev[i]=U_Gyro[i];
}
lastTime=millis();
}
#endif

void getGains(){
Kp_ypr[0]=data.Kp_y/gain_scaler;
Kp_ypr[1]=data.Kp_p/gain_scaler;
Kp_ypr[2]=data.Kp_r/gain_scaler;

Kp_gyro[0]=data.Kp_y_rate/gain_scaler;
Kp_gyro[1]=data.Kp_p_rate/gain_scaler;
Kp_gyro[2]=data.Kp_r_rate/gain_scaler;

Ki_gyro[0]=data.Ki_y_rate/gain_scaler;
Ki_gyro[1]=data.Ki_p_rate/gain_scaler;
Ki_gyro[2]=data.Ki_r_rate/gain_scaler;

Kd_gyro[0]=data.Kd_y_rate/gain_scaler;
Kd_gyro[1]=data.Kd_p_rate/gain_scaler;
Kd_gyro[2]=data.Kd_r_rate/gain_scaler;
}

// ================================================
// ===              MPU  INTERRUPT DETECTION ROUTINE
// ================================================
void getGyro(){
for(int i=0;i<3;i++){
Gyro[i]=gyro[2-i]/131.0;
Ypr[i]=ypr[i] * 180/M_PI;
//Serial.print(Gyro[i]);Serial.print("\t");
}
//  Gyro[2]=lpfilt.calc_out(-Gyro[2]);
// Gyro[1]=lpfilt.calc_out(-Gyro[1]);
Gyro[0]=gyro_lpfilt.calc_out(-Gyro[0])*10;
}

void dmpDataReady() {
mpuInterrupt = true;
}

\section{Rotor Control}
void rotor_control(){

//
if(U1<=4 ||  IBus.readChannel(4)==2000){

analogWrite(Rotor_1,0);

analogWrite(Rotor_2,0);

analogWrite(Rotor_3,0);

analogWrite(Rotor_4,0);

iTerm_gyro[0]=0;
iTerm_gyro[1]=0;
iTerm_gyro[2]=0;

}
else{
analogWrite(Rotor_1,Rotor1_pwm);

analogWrite(Rotor_2,Rotor2_pwm);

analogWrite(Rotor_3,Rotor3_pwm);

analogWrite(Rotor_4,Rotor4_pwm);

}

return;
}

void controlMixing(){

Rotor1_pwm  =constrain(U1+U_Gyro[1]+U_Gyro[2]-U_Gyro[0],10,255);
//TRY REVERSING ROTORS Rotation 1 and 2, 3 ,4
Rotor2_pwm  =constrain(U1+U_Gyro[1]-U_Gyro[2]+U_Gyro[0],10,255);
//if yaw spin occurs
Rotor3_pwm  =constrain(U1-U_Gyro[1]-U_Gyro[2]-U_Gyro[0],10,255);
```

```
Rotor4_pwm  =constrain(U1-U_Gyro[1]+U_Gyro[2]+U_Gyro[0],10,255);

}

\section{Telemetry Data Generation}
void dataPrint(){

#ifdef angles
for(int i=0;i<3;i++) ...
bitWrite(telemetry_send.angles_signs,i,ypr[i]*180/M_PI>=0);
telemetry_send.throttle=U1;
telemetry_send.yaw=abs(ypr[0] * 180/M_PI);
telemetry_send.pitch=abs(ypr[1] * 180/M_PI);
telemetry_send.roll=abs(ypr[2] * 180/M_PI);

#endif

#ifdef looptime

telemetry_send.loopt=(dt*1000);

#endif

#ifdef err
for(int i=0;i<3;i++){
bitWrite(telemetry_send.err_signs,i,error_ypr[i]>=0);
bitWrite(telemetry_send.err_signs,(i+3),error_gyro[i]>=0);
}
telemetry_send.errRoll =abs(error_ypr[2]);
telemetry_send.errPitch=abs(error_ypr[1]);
telemetry_send.errYaw_rate=abs(error_gyro[0]);
telemetry_send.errRoll_rate=abs(error_gyro[1]);
telemetry_send.errPitch_rate=abs(error_gyro[2]);
#endif

#ifdef cont
for(int i=0;i<3;i++)...
 bitWrite(telemetry_send.cont_signs,i,U_ypr[i]>=0);
telemetry_send.U1=U1;
telemetry_send.U2=abs(U_ypr[2]);
telemetry_send.U3=abs(U_ypr[1]);
telemetry_send.U4=abs(U_ypr[0]);
#endif

#ifdef cont_rate
for(int i=0;i<3;i++)...
 bitWrite(telemetry_send.cont_rate_signs,i,U_Gyro[i]>=0);

telemetry_send.U2_rate=abs(U_Gyro[2]);
telemetry_send.U3_rate=abs(U_Gyro[1]);
telemetry_send.U4_rate=abs(U_Gyro[0]);
#endif

#ifdef pwm
telemetry_send.Rotor1pwm=Rotor1_pwm;
telemetry_send.Rotor2pwm=Rotor2_pwm;
telemetry_send.Rotor3pwm=Rotor3_pwm;
telemetry_send.Rotor4pwm=Rotor4_pwm;
#endif

#ifdef datas

telemetry_send.data[0]=
telemetry_send.data[1]=
telemetry_send.data[2]=
telemetry_send.data[3]=
for(int i=0;i<4;i++) {
bitWrite(telemetry_send.data_signs,i,telemetry_send.data[i]>=0);
telemetry_send.data[i]=abs(telemetry_send.data[i])
}
#endif

// telemetry_send.rcValues0=IBus.readChannel(0);
// telemetry_send.rcValues1=IBus.readChannel(1);
// telemetry_send.rcValues2=IBus.readChannel(2);
// telemetry_send.rcValues3=IBus.readChannel(3);
// telemetry_send.rcValues4=IBus.readChannel(4);
// telemetry_send.rcValues5=IBus.readChannel(5);

#ifdef serialStandard
#ifdef angles
Serial.print(F("T:"));
Serial.print(telemetry_send.throttle);
Serial.print("\t");
Serial.print(F("Y:"));
Serial.print(telemetry_send.yaw);
Serial.print("\t");
Serial.print(F("P:"));
Serial.print(telemetry_send.pitch);
Serial.print("\t");
Serial.print(F("R:"));
Serial.print(telemetry_send.roll);
Serial.print("\t");
#endif
#ifdef looptime
Serial.print(F("dt:"));
dt= telemetry_send.loopt;
Serial.print(dt);Serial.print("\t");
#endif
#ifdef err
Serial.print(F("ER:"));
Serial.print(telemetry_send.errRoll);
Serial.print("\t");Serial.print(F("EP:"));
Serial.print(telemetry_send.errPitch);
Serial.print("\t");
#endif
#ifdef err_rate
Serial.print(F("EGY:"));
Serial.print(telemetry_send.errYaw_rate);
Serial.print("\t");
Serial.print(F("EGP:"));
Serial.print(telemetry_send.errPitch_rate);
Serial.print("\t");
Serial.print(F("EGR:"));
Serial.print(telemetry_send.errRoll_rate);
Serial.print("\t");
#endif
#ifdef cont
Serial.print(F("U1:"));
Serial.print(telemetry_send.U1);Serial.print("\t");
Serial.print(F("U2:"));
Serial.print(telemetry_send.U2);Serial.print("\t");
Serial.print(F("U3:"));
Serial.print(telemetry_send.U3);Serial.print("\t");
Serial.print(F("U4:"));
Serial.print(telemetry_send.U4);Serial.print("\t");
#endif
#ifdef cont_rate
Serial.print(F("U2_r:"));
Serial.print(telemetry_send.U2_rate);Serial.print("  \t");
Serial.print(F("U3_r:"));
Serial.print(telemetry_send.U3_rate);Serial.print("  \t");
Serial.print(F("U4_r:"));
Serial.print(telemetry_send.U4_rate);Serial.print("  \t");
#endif
#ifdef pwm
```

```
Serial.print(F("1:"));
Serial.print(telemetry_send.Rotor1pwm);Serial.print("\t");
Serial.print(F("2:"));
Serial.print(telemetry_send.Rotor2pwm);Serial.print("\t");
Serial.print(F("3:"));
Serial.print(telemetry_send.Rotor3pwm);Serial.print("\t");
Serial.print(F("4:"));
Serial.print(telemetry_send.Rotor4pwm);Serial.print("\t");
#endif
#ifdef datas
Serial.print(F("data1:"));
Serial.print(telemetry_send.data1);Serial.print("\t");
Serial.print(F("data2:"));
Serial.print(telemetry_send.data2);Serial.print("\t");
Serial.print(F("data3:"));
Serial.print(telemetry_send.data3);Serial.print("\t");
Serial.print(F("data4:"));
Serial.print(telemetry_send.data4);Serial.print("\t");
#endif
#endif

#ifdef serialdata
#ifdef angles
Serial.print(F("Y:\t"));
Serial.print(ypr[0] * 180/M_PI);
Serial.print("\t");
Serial.print(F("Y:\t"));
Serial.print(ypr[1] * 180/M_PI);
Serial.print("\t");
Serial.print(F("R:\t"));
Serial.println(ypr[2] * 180/M_PI);
#endif

#ifdef pwm
Serial.print(F("1:  "));
Serial.print(Rotor1_pwm);
Serial.print("\t");
Serial.print(F("2:  "));
Serial.print(Rotor2_pwm);
Serial.print("\t");
Serial.print(F("3:  "));
Serial.print(Rotor3_pwm);
Serial.print("\t");
Serial.print(F("4:  "));
Serial.print(Rotor4_pwm);
#endif

#ifdef err
Serial.print(F("E_p: " ));
Serial.print(error_ypr[1]);
Serial.print("\t");
Serial.print(error_ypr[2]);
#endif

#ifdef cont
Serial.print(U1 );
Serial.print("\t");
Serial.print(U_ypr[2]);
Serial.print("\t");
Serial.print(U_ypr[1]);
Serial.print("\t");
Serial.print(U_ypr[0]);
#endif

#ifdef err_rate
Serial.print(F("yaw_err: " ));
Serial.print(error_gyro[2]);
Serial.print("\t");
Serial.print(F("pitch_err: " ));
Serial.print(error_gyro[1]);
Serial.print("\t");
Serial.print(F("roll_err: " ));
Serial.print(error_gyro[0]);
#endif

#ifdef cont_rate
Serial.print("\t");
Serial.print(U_Gyro[2]);
Serial.print("\t");
Serial.print(U_Gyro[1]);
Serial.print("\t");
Serial.print(U_Gyro[0]);
#endif
Serial.print("\n");
#endif
}

\section{nRF24 Subroutines}
void resetData(){
//We define the initial value of each data input
//3 potentiometers will be in the middle position so 127 should be neutral
data.throttle = 0;
data.yaw = 127;
data.pitch = 127;
data.roll = 127;
//data.AUX1 = 0;
//data.AUX2 = 0;
}

void recvData(){

while ( radio.available() ) {
radio.read(&data, sizeof(MyData));
lastRecvTime = millis(); //here we receive the data
}
}

void telemetry(){
if ( radio.available() ) {
radio.writeAckPayload( 1, &telemetry_send, sizeof(telData) );
radio.read( &data,sizeof(MyData) );
}
}
```