5-2017

A Novel Hexapod Robot Design with Flight Capability

Mark C. Pitonyak
mcp3579@rit.edu

Follow this and additional works at: <http://scholarworks.rit.edu/theses>

Recommended Citation

Pitonyak, Mark C., "A Novel Hexapod Robot Design with Flight Capability" (2017). Thesis. Rochester Institute of Technology.
Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

A Novel Hexapod Robot Design with Flight Capability

by

Mark C. Pitonyak

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical and Microelectronic Engineering

Supervised by

Professor Dr. Ferat Sahin
Department of Electrical and Microelectronic Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
May 2017

Approved by:

Dr. Ferat Sahin, Professor
Thesis Advisor, Department of Electrical and Microelectronic Engineering

Dr. Sildomar Monteiro, Assistant Professor
Committee Member, Department of Electrical and Microelectronic Engineering

Dr. Sohail Dianat, Professor
Committee Member, Department of Electrical and Microelectronic Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

A Novel Hexapod Robot Design with Flight Capability

I, Mark C. Pitonyak, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Mark C. Pitonyak

Date

Dedication

This thesis is dedicated to my family, friends, and colleagues for their continued support and assistance.

Acknowledgments

I would like to thank Dr. Ferat Sahin for advising my thesis—suggesting the topic and providing helpful feedback throughout the project.

I am thankful to my parents for giving support and encouraging my engineering interests from a young age.

I am appreciative of my colleagues in the Multi Agent Bio-Robotics Laboratory. Thanks to Celal Savur and Shitij Kumar for their introduction to and help with ROS. Thanks to Matt Haywood for sharing knowledge and experience about 3D printing. To everyone else, I am grateful for the interesting conversations and useful input.

Abstract

A Novel Hexapod Robot Design with Flight Capability

Mark C. Pitonyak

Supervising Professor: Dr. Ferat Sahin

Multicopter aircrafts have become popular platforms for researching flight control and aerial image processing applications. These crafts could gain unique utility from the ability to fly to remote locations, land, then walk to perform close quarters inspection of the area. This thesis presents a novel hexapod-quadcopter design with multicopter flight hardware directly embedded in the hexapod legs. The robot was assembled from custom designed 3D printed structural components. Software was developed for the walking and flying locomotion methods as well as transitional procedures necessary to switch between these two modes. The robot functionality was tested and validated to work on the platform.

List of Contributions

- The design, 3D printing, and assembly of a hexapod-quadcopter robot with the multicopter flight hardware directly embedded in the hexapod legs.
- The software implementation developed to control the robot using a microcontroller to interface with the sensors and actuators and ROS on an embedded Linux computer to handle higher level algorithms.
- The development, testing, and validation of walking and flying locomotion and transitional procedures for switching between walking and flying modes.
- M. Pitonyak, F. Sahin, “A novel hexapod robot design with flight capability,” *System of Systems Engineering (SoSE), 2017 IEEE International Conference on*, 2017. Accepted for publication.
- M. Pitonyak, F. Sahin, “Locomotion and transitional procedures for a hexapod-quadcopter robot,” *Systems, Man and Cybernetics (SMC), 2017 IEEE International Conference on*, 2017. Submitted for publication.

Contents

Dedication	iii
Acknowledgments	iv
Abstract	v
List of Contributions	vi
1 Introduction	1
2 Background Literature	3
2.1 Combined Multipod-Multicopters	3
2.2 Hexapods	4
2.2.1 Kinematics	6
2.2.2 Walking Algorithms	7
2.3 Multicopters	8
2.3.1 Flight Mechanics	9
2.3.2 Flight Control	10
2.4 3D Printing	10
3 Preliminary Design	13
4 Detailed Design	17
4.1 Electrical System	17
4.2 Structural Components	29
4.2.1 Thrust Legs	29
4.2.2 Strut Legs	36
4.2.3 Body	37
4.2.4 Robot	44
4.3 Software	45
4.3.1 Microcontroller	46
4.3.2 ROS	47

5	Results	53
5.1	Locomotion	53
5.1.1	Walking	53
5.1.2	Flying	57
5.2	Transitional Procedures	58
5.2.1	Walking-to-Flying	58
5.2.2	Flying-to-Walking	59
6	Conclusions	64
7	Future Work	66
	Bibliography	67
A	PCB Schematics	69
B	3D Printing	75

List of Figures

2.1	Multipod-multicopters.	3
2.2	Hexapods.	5
2.3	Three DOF hexapod leg geometry.	6
2.4	Hexapod walking gaits.	7
2.5	Multicopters.	8
2.6	Quadcopter RPY characteristics.	9
2.7	Two-tiered RPY-based PID flight control.	10
2.8	MakerBot Replicator 2X.	11
2.9	3D printing support for overhanging features.	12
3.1	Dynamixel AX-12A servo.	13
3.2	DJI's E1200 Standard Tuned Propulsion System.	15
4.1	Electrical hardware block diagram.	18
4.2	ODROID-C1+ with USB peripherals.	19
4.3	Hardware interface hub PCB.	19
4.4	Robot logic-level electronic layer.	21
4.5	Pololu VL53L0X time-of-flight distance sensor.	22
4.6	IR sensor ring PCB.	23
4.7	Power distribution block diagram.	24
4.8	Zippy Compact 3S LiPo battery.	24
4.9	Power distribution PCB.	25
4.10	Voltage regulator output spike.	26
4.11	12 V regulator and distribution.	26
4.12	Robot power distribution layer.	28
4.13	Thrust leg tibia segment.	30
4.14	IR sensor ring with protective cap.	31
4.15	Upper tibia segment before mounting peripherals.	31
4.16	Upper tibia segment with peripherals mounted.	32
4.17	ESC embedded in the upper tibia segment.	33
4.18	Thrust leg femur segment.	34
4.19	Thrust leg coxa segment.	35

4.20	Strut leg.	36
4.21	Strut legs supporting the robot.	37
4.22	Central body frame.	38
4.23	Body frame with attached hardware.	39
4.24	Battery and power switch wire connections.	40
4.25	Bottom of robot body with electronics attached.	41
4.26	Bottom cover on robot.	41
4.27	Robot bottom with legs and bottom cover attached.	42
4.28	Top of robot body with electronics attached.	43
4.29	Robot top with connected wires and tibia pockets.	43
4.30	Full robot in walking stance.	44
4.31	Full robot in flying stance.	45
4.32	ROS node graph of the walking and hexapod processes.	51
4.33	ROS node graph of the flying processes.	52
5.1	Robot frames in Rviz.	55
5.2	Robot performing a walking test.	56
5.3	Power consumption of servos during walking test.	56
5.4	Flight test rig.	57
5.5	Robot hovering on the flight test rig.	58
5.6	Robot resting on the strut legs.	59
5.7	Propellers manually folded together.	60
5.8	Bimodal von Mises curve fit and mapping around the thrust motor.	61
5.9	Propellers resting on femur protrusions.	62
5.10	Propellers folded between femur segments.	63
A.1	PCB layer color assignments.	69
A.2	Hardware interface hub PCB layers.	70
A.3	IR sensor ring PCB layers.	70
A.4	Power distribution PCB layers.	71
A.5	Schematic for hardware interface hub PCB.	72
A.6	Schematic for IR sensor ring PCB.	73
A.7	Schematic for power distribution PCB.	74
B.1	Body frame modeled in Inventor.	76
B.2	Body frame prepared for 3D printing.	77
B.3	Makerbot Replicator 2X used to 3D print.	78
B.4	Body frame being 3D printed.	79

Chapter 1

Introduction

Multicopter aircrafts have become popular in both research outlets and the consumer hobby industry within the past decade. These platforms offer unique capabilities suited for search and rescue, disaster zone assistance, environmental conservation, and surveillance. The predominant use for multicopters in these fields is in obtaining aerial imagery, either manually or autonomously. Being aerial crafts, multicopters are generally not outfitted with a way to maneuver on ground. Certain circumstances would benefit from this ability, such as navigating away from areas of high smoke or dust density, exploring closed spaces not conducive to flying, and being able to move after a crash or tumbled landing. Significant utility could be gained from a multicopter's ability to fly to a remote location, such as on top of an oil pipeline, where it could land and perform close quarters inspection by walking.

Hexapod robots are capable of omnidirectional terrestrial mobility similar to a multicopter's aerial movement. With six legs, a hexapod can maintain static stability when three or more legs are grounded concurrently. This ability allows walking gaits to be selected for improved speed or stability. Legs can alternatively be controlled independent of a pre-programmed, periodic gait. Hexapods are well suited to navigation outdoors over uneven terrain where there may be obstacles.

Despite individual research prevalence, examples of combined hexapod and multicopter functionality in a single robot are sparse. The overarching goal of this thesis was to design and build a hexapod with integrated multicopter hardware for flight capability. The novel approach involved embedding the flight hardware directly into the hexapod legs. These legs were designed with three joints each—three degrees of freedom (DOF)—to provide full omnidirectional walking capability with minimum complexity. The structural pieces of the robot were designed in computer-aided design (CAD) software and 3D printed. 3D printing facilitated the rapid prototyping and testing of parts. After building the robot, basic locomotion algorithms for walking and flying were programmed, along with certain transitional procedures necessary to switch between these two modes of operation. The higher-level algorithms were implemented on an embedded Linux computer running Robot Operating System (ROS), while lower-level hardware interfacing was handled with a microcontroller.

The remainder of this paper is organized as follows. Chapter 2 covers related multipod-multicopter designs, as well as designs for each of these two robot types separately. Kinematic constraints are discussed along with locomotion implementations. Chapter 3 discusses the preliminary estimations and justifications for choosing general design elements. Chapter 4 elaborates on the design with specifics about the electronics system, structural components, and software architecture. The locomotion implementation and results are covered in Chapter 5, accompanied by transitional procedures to switch between both modes. Conclusions are organized in Chapter 6, followed by potential future applications for the robot in Chapter 7.

Chapter 2

Background Literature

2.1 Combined Multipod-Multicopters

A search for related multipod-multicopter designs returned no similar robots in published literature, but two robots were found on the internet at large. These robots were the hexapod-hexacopter HexaWalker from MadLab Industries [3] in Fig. 2.1a and a quadruped-quadcopter from an online blog by a radio controlled system enthusiast [2] in Fig. 2.1b. Both robots came from the late 2012, early 2013 era—coinciding with when multicopters began gaining traction in the consumer hobby industry due to reduced manufacturing costs. This timeframe combined with the robot designs suggests a top-down approach where both creators likely had the individual multipod and multicopter on hand and attached them for

(a)

(b)

Figure 2.1: MadLab Industries' HexaWalker hexapod-hexacopter [1] (a) and a quadruped-quadcopter [2] (b). Both robots employ separated walking and flying hardware.

the combined functionality. This method produced the desired capability, but was inherently not optimized for minimal size or weight, although some effort was taken in this direction. MadLab Industries replaced parts of the frame with lightweight carbon fiber [3] and [2] incorporated servo motors at the base of the quadcopter arms to fold them together.

A clear improvement to the bisected walking and flying hardware appeared to be a consolidated approach that embedded the flight hardware directly into the multipod legs. This configuration would have the legs double as the multicopter arms, effectively halving the robot size while minimizing the profile. All indications point to this thesis introducing the first multipod-multicopter design to implement this combination.

2.2 Hexapods

Thesis guidelines mandated that the core of the new robot design be a 3D printed hexapod frame with three DOF per leg. These attributes were emphasized while researching existing hexapod designs. A case study covering the history of hexapod designs is provided in [4]. Examples in that paper span back several decades, but more recent small-scale robots were considered for reference. Perhaps the best realized robot aligning to the aforementioned thesis criteria was the hexapod Dmitri [5] [6] developed by Matt Bunting at the University of Arizona. Dmitri was conceived as a class project, but matured into the Fig. 2.2a version after Intel funded the project to use as a platform for demonstrating the capability of company processors used to run the robot's kinematics and image processing. Although the functional objectives for Dmitri—biological behavior generation and image processing applications—differ from the challenges presented by multimodal locomotion integration,

Figure 2.2: Matt Bunting’s Dmitri hexapod robot developed for Intel (a) [5]. Structural pieces of the frame are 3D printed, allowing rapid design prototyping to quickly iterate and optimize mechanical components. The Trossen Robotics PhantomX AX Metal Hexapod Mark III (b) [7] sports Dynamixel servo motors with integrated sensors that provide useful diagnostic information.

the prototyping process can be taken as a template that verifies the viability of 3D printing for the hexapod frame.

Another hexapod of specific interest was the commercially available PhantomX AX Metal Hexapod Mark III from Trossen Robotics [7], shown in Fig. 2.2b. Although not 3D printed, this robot featured a simple, kit-based layout for generic utility. The legs incorporated Dynamixel AX-12A servo motors that have internal sensors to measure the servo speed, temperature, shaft position, voltage, and load torque. Dmitri utilized similar Dynamixel MX-28 servos with improved torque and speed performance compared to the AX-12As.

The hexapods in Fig. 2.2 have three DOF per leg. The leg joints are oriented in a biologically-inspired manner similar to insects and arachnids, where the servo at the base of the leg rotates forward and backward coplanar to the body and the two remaining servos extend the leg outward and inward relative to the body. Anatomical terminology is used when referencing the three leg segments, with *coxa* referring to the hip, *femur* indicating

the middle segment, and *tibia* describing the last segment that contacts the ground. These terms are used throughout the rest of this thesis.

2.2.1 Kinematics

The three DOF nature of each leg limits any tibia end point to only be reachable by at most two distinct joint sets. This makes it feasible to solve the inverse kinematics (IK) using an analytical approach [8] [9], as opposed to an iterative-style approach that would be necessary for more complex and higher-DOF architectures. The frame transformations between joints remain static except for the rotated angle, so geometric constraints can be applied to determine the necessary joint angles to reach an endpoint position from the leg base frame. Fig. 2.3 shows the relevant geometric considerations, with γ in the left image denoting the coxa angle and α and β in the right image denoting the femur and tibia angles, respectively. Analytical IK can be executed with limited computational power to calculate the joint set or determine that a leg endpoint is unreachable given the geometric constraints.

Figure 2.3: Depiction of a three DOF hexapod leg for the purpose of developing analytical inverse kinematics equations based on geometric constraints [8].

2.2.2 Walking Algorithms

Hexapod gaits are commonly implemented by programming the four-dimensional trajectories of the six leg endpoints and using inverse kinematics to actuate the robot into those positions. The specific trajectory planning can vary in implementation and complexity, but at the most basic level involves pre-programmed periodic sequences such as the tripod, wave, and ripple gaits in Fig. 2.4 [10]. The waveforms show rising pulses during sequence steps when a leg is lifting and extending forward, and low values if the leg is supporting the robot on the ground. Supporting legs will commonly follow a straight line to maintain the desired robot velocity and extending legs follow a spline curve to lift and swing forward [11]. Gait style can be selected to improve speed or stability, with tripod being the fastest gait but wave gait being the most stable since five legs are always grounded concurrently.

Figure 2.4: Visualization for three common periodic hexapod gaits: tripod, wave, and ripple gait [10]. High pulses in the waveforms indicate a leg that is advancing forward during a sequence step and low values indicate the associated leg is grounded, supporting the robot.

2.3 Multicopters

Many different styles of multicopters are available to buy commercially. These crafts commonly have four, six, or eight rotors—with propeller pairs spinning in opposite directions to cancel torque. Factors such as size, number of rotors, thrust motor ratings, and propeller length and pitch are selected to accommodate the desired thrust force and payload capacity. Fig. 2.5 shows two commercially available multicopters. Standard multicopter design practices are exhibited in each, where a central hub for the batteries and electronics connects to arms extending out to the rotors. The DJI Matrice 600 Pro [13] in Fig. 2.5b uses traditional struts to support the craft, whereas the DJI Inspire 1 [12] in Fig. 2.5a has supports below the rotors that lift after takeoff. Consumer grade crafts tend toward quadcopter configurations like the DJI Inspire 1, and are generally equipped with gimbals and lightweight cameras for recording aerial imagery.

Figure 2.5: DJI’s commercially available Inspire 1 quadcopter (a) [12], and hexacopter Matrice 600 Pro (b) [13]. The Matrice 600 Pro shows a standard multicopter layout, with a central hub for the batteries and electronics, arms extending out to the rotors, and struts for supporting the craft. The Inspire 1 has a similar layout but uses extensions on the arms to support the robot, which it lifts after takeoff.

2.3.1 Flight Mechanics

Aircraft rotations are commonly referred to in terms of roll, pitch, and yaw (RPY). RPY axes are pictured in Fig. 2.6a and correspond to left and right rotation for roll, front and back rotation for pitch, and rotation about the vertical axis for yaw. The fundamental mechanic of multicopter flight control is inducing opposing differential thrust changes to rotate about these axes. For a quadcopter configured as Fig. 2.6b is, rolling right would involve the left two rotors increasing thrust and the right two rotors decreasing thrust by an equivalent amount. Adjusting pitch works in a similar manner with the front and back two rotors, and yaw is adjusted by differential changes to diagonal rotors, since the spin directions alternate radially around the aircraft to cancel torque about the yaw axis. Differential changes are applied on top of a common thrust point set to control altitude.

Figure 2.6: Roll, pitch, and yaw (RPY) (a) are commonly used to describe the rotations of an aircraft [14]. Roll represents rotation left and right, pitch is rotation front and back, and yaw is rotation about the vertical axis. A quadcopter's rotors rotate in alternating directions (b) to cancel torque about the yaw axis [15]. RPY rotations are induced by opposing differential thrust amounts between rotors.

2.3.2 Flight Control

One of the simplest ways to manage multicopter flight control is implementing proportional-integral-derivative (PID) controllers for the RPY axes. PID controllers take two inputs: a target value and a state value. The controller will produce an output based on how the state value differs from the target value and how the proportional, integral, and derivative coefficients— K_p , K_i , and K_d respectively—are tuned. Flight control can be accomplished by a one-tier system where the controller takes RPY angular velocity inputs, or a two-tier system as in Fig. 2.7 where preceding RPY angle PID controllers feed into angular rate PID controllers. Generally an inertial measurement unit (IMU) provides state feedback for the robot angles and angular velocities. If an altitude measurement device is present, such as an altimeter or ultrasonic distance sensor, PID controllers can be used to control altitude.

Figure 2.7: The PID controller hierarchy for a two-tiered RPY-based flight control. The first controller takes either the roll, pitch, or yaw state value and the corresponding target angle as inputs to generate the target angular rate that feeds into a second controller. This target is compared against the gyroscope measurement to create the output that is mapped to differential motor adjustments.

2.4 3D Printing

3D printing has emerged in the 21st century as a low cost solution for rapidly prototyping objects of virtually any shape using a variety of materials. The 3D printer available for

Figure 2.8: The MakerBot Replicator 2X (a) [16] has dual extruders (b) to perform multi-material printing.

this project was a MakerBot Replicator 2X, shown in Fig. 2.8a. This printer uses additive manufacturing to create a part layer by layer in either acrylonitrile butadiene styrene (ABS) or polylactic acid (PLA) plastic. A spool of the chosen plastic is fed into the printing head where it is melted and pushed through either of the extruders in Fig. 2.8b in 2D layers that stack vertically, ultimately resulting in a 3D part. In order to build a part, it must first be modeled in CAD software, exported as a *.stl* file, and sliced into printable layers using the MakerBot Desktop software.

Several parameters must be considered before choosing how to print a part. ABS requires high extruder and build plate temperatures while printing and should be printed with the build volume sealed for thermal isolation. Maintaining a high internal temperature prevents the ABS from shrinking as it cools. Large ABS parts may warp and peel up from the print bed as the upper plastic layers cool faster and constrict more than the bottom layers closer to the heated build plate. By comparison, PLA can be printed at lower temperatures and does not significantly shrink after cooling. Parts printed in PLA can be more brittle

Figure 2.9: The part brought into MakerBot Desktop in (a) has overhanging geometry that requires support to print properly. The sliced output in (b) shows an added raft underneath the part and support material under overhanging features.

than ABS, which generally flexes when stressed. Whether ABS or PLA is used, painter's tape is often applied to the build plate to enhance the adhesion of the first print layer.

Other print parameters depend on characteristics of the part being printed. If the part has overhanging features, such as that in 2.9a, support material must be printed below this geometry. Fig. 2.9b shows the sliced output with the support present along with a raft underneath the part to improve adhesion to the build plate. These assistive elements are removed after printing. Parts can be rotated to reduce the amount of necessary support and minimize print time. Although parts can be printed at any orientation, keeping flat faces horizontal or vertical will produce the smoothest results. The expected torsion and shear forces on the part also inform the optimal print orientation. A part is most susceptible to shearing between vertical layers with low adhesion area and breaking at thin geometry.

Chapter 3

Preliminary Design

The hexapods in Fig. 2.2 incorporated Dynamixel servos for actuating the leg joints. These servos can provide diagnostic information including servo speed, temperature, shaft position, voltage, and load torque that would be useful for autonomously monitoring the robot state while walking and flying. Instead of pulse-width modulation control like traditional servos, these motors are controlled digitally using half-duplex serial, allowing them to be daisy chained and individually accessed. The AX-12A model in Fig. 3.1 and MX-28 model were used on the aforementioned hexapods. Comparing the two, the MX-28s were prohibitively expensive and weighed more than the AX-12A servos—72 g to 55 g—making them less optimal for a flying robot. The AX-12A servos cost \$224.50 for six, fitting nicely into

Figure 3.1: The Dynamixel AX-12A servo front (a) and back (b) [17]. The servos are daisy chained together by plugging cables into headers in the back.

the \$2500 robot budget. These servos can provide $15.3 \text{ kg} \cdot \text{cm}$ of stall torque, which is less than the MX-28 and other servos in the Dynamixel family but sufficient for the hexapod-quadcopter's proof-of-concept purposes. These were the primary justifications for selecting Dynamixel AX-12A servos to be used in the robot joints.

The hexapod foundation of the robot was selected to have three DOF per leg to provide full omnidirectional walking capability. More DOF in the legs would add unnecessary complexity and increase the robot weight—lowering the maximum flight time. The general robot shape was selected to be similar to those in Fig. 2.2 with the same leg joint orientations. The necessary body size and leg span were constrained by other characteristics. It was preferable to 3D print the body as one piece for structural integrity, limiting the part to fit in the $24.6 \text{ cm} \times 15.2 \text{ cm} \times 15.5 \text{ cm}$ build volume of the MakerBot Replicator 2X. The leg span needed to be long enough such that the multicopter propellers would not overlap the body and have ample space separating each other. Propeller length and pitch depended primarily on the necessary thrust force to lift the robot and whether the multicopter configuration would be a quadcopter or hexacopter.

The choice between quadcopter and hexacopter derived from the rotational torque-cancelling effect of clockwise (CW) and counterclockwise (CCW) propeller pairs. Choosing four or six thrust axes allowed for the simplest flight mechanics and eventual software control. Considering the hardware components of eighteen Dynamixel AX-12A servo motors for the legs, up to six thrust motors, likely a six-cell LiPo battery, and the 3D printed frame, the robot was estimated to eventually weigh 4800 g. This estimate would require

a thrust minimum of 1200 g/axis delivered from a quadcopter or 800 g/axis from a hexacopter. Full propulsion kits online containing propellers, thrust motors, and electronic speed controllers (ESCs) to drive the motors favored lower-length fixed propellers for smaller crafts and larger propellers with higher pitch—either fixed or folding—to lift heavier multicopters. Besides thrust capacity, the largest constraining factor of the flight hardware was propeller size and the feasibility of concealment along the hexapod leg. Comparing the thrust per axis estimates to the thrust capability of available propulsion kits led to a decision between fixed 13-inch propellers in hexacopter configuration or folding 17-inch propellers as a quadcopter. The flight hardware would have to be embedded in the leg no closer than half the propeller length from the body to prevent overlapping and losing thrust. The 17-inch propeller’s folded-state 8.5-inch profile would be easier to accommodate along a hexapod leg than the full 13-inches of the nominally shorter one. The concept developed for the robot involved mounting the thrust motor to the inside of the tibia segment so the

Figure 3.2: DJI’s E1200 Standard Tuned Propulsion System features 40 A ESCs (1), 3100 rpm/V thrust motors (2), and 17-inch long, 6-inch pitch carbon fiber composite folding propellers (3) [18]. These elements combine to support 3900 g/axis of thrust capacity.

propellers could be folded and constrained inside the femur segment. The femur and tibia segments would be sized for the robot to walk without hitting the propellers and fly without the propellers overlapping the body. Ultimately the DJI E1200 Standard Tuned Propulsion System [18] in Fig. 3.2 was selected for the 17-inch long, 6-inch pitch folding propellers and 3900 g/axis maximum thrust capacity rating. The general flying design emerged as a quadcopter “X” configuration using the front two and back two legs—*thrust legs*, where the middle two legs—*strut legs*—could serve as struts for takeoff and landing. To maintain reasonable robot proportions, the body would be sized to maximize the 3D printer’s lateral build area.

Chapter 4

Detailed Design

The following sections discuss specific details of the robot electrical, mechanical, and software systems. Section 4.1 includes block diagrams of the electrical hardware and power distribution, explanation of custom designed printed circuit boards (PCBs), and the arrangement of these electronics in the robot. Section 4.2 presents the 3D printed robot, with emphasis on the thrust legs, landing legs, and body. Section 4.3 describes the implementation of low level code on a microcontroller (MCU) to interface with the sensors and actuators and the higher level algorithms using Robot Operating System (ROS) on an embedded Linux computer.

4.1 Electrical System

The electrical system of the hexapod-quadcopter was designed to accommodate the needs for walking and flying. Fig. 4.1 shows a block diagram of the electronic hardware in which connecting lines represent information flow and state the communication protocol. The orange blocks are processing units, green blocks are actuators, and blue blocks include sensors and other peripherals. The larger gray groups reflect the three custom PCBs designed and fabricated for the robot. The PCBs were designed in KiCad and fabricated

Figure 4.1: A block diagram of the electrical system on the robot. Lines indicate the flow of information between two blocks and list the communication protocol. Orange blocks are the processing units, green are actuators, and blue are sensors and remaining hardware.

using the online service MacroFab. Other power distribution electronics including the battery and voltage regulators are shown later in Fig. 4.7. The electronics were developed with modularity in mind, incorporating breakout boards and screw terminals or cable connections when possible. This configuration allows individual hardware to easily be upgraded, changed, or expanded without affecting other components.

An ODRROID-C1+ was selected as the single board computer to run ROS and the higher level processing. This model uses a quad-core ARM Cortex-A5 architecture running at 1.5 GHz. Ubuntu 14.04 was installed as the operating system to provide the best support for ROS. Fig. 4.2 shows the ODRROID-C1+ connected to the WiFi adapter and Xbox wireless receiver. These devices provide sufficient wireless performance for close range testing, but for extended range flight an RF receiver could be added. The electronic system was designed so that only USB devices connect to the ODRROID-C1+, allowing the computer to easily be exchanged for another if more processing power is needed.

Figure 4.2: The ODROID-C1+ single board computer with connected WiFi adapter and Xbox wireless receiver.

The first gray block in Fig. 4.1 is the hardware interface board. This PCB was designed to connect all of the sensors and actuators to the Teensy 3.2 microcontroller. Fig. 4.3a shows the isolated board that predominantly features header rows for plugging cables and breakout boards into. A limited amount of circuitry is included on the board itself to manage communication flow. An NXP PCA9547 eight channel I²C multiplexer is included

(a)

(b)

Figure 4.3: The hardware interface hub PCB (a) with attached microcontroller and sensor breakout boards (b).

for selecting between the four infrared (IR) sensor rings and the Adafruit 10-DOF breakout board with an IMU and altimeter. The multiplexing is necessary to avoid I²C address confliction, since all four IR rings are identical and furthermore share certain addresses with integrated circuits (ICs) on the 10-DOF board. Other circuitry is provided to convert the microcontroller's full-duplex serial hardware lines to half-duplex serial for the AX-12A servos. This was accomplished by routing the MCU receive and transmit pins through a Philips 74LVC2G125 dual bus buffer driver to the unified servo data line. A Diodes Incorporated 74LVC1G14 single Schmitt-trigger inverter was used to enable either the receive or transmit connection with a single MCU output pin. Using a hardware inverter ensured that both lines could not be connected simultaneously. The servo cable plugged into a Molex 22-03-5035 SPOX header on the PCB.

Fig. 4.3b shows the hardware interface hub with the Teensy 3.2 MCU, Adafruit 10-DOF breakout board, and Adafruit GPS breakout board plugged in. These boards can easily be removed and replaced since they are not soldered directly to the board. The male header rows to the left take cable inputs from the I²C peripherals and provide pulse width modulation (PWM) signals to the thrust motor ESCs. The PCB layers are shown more clearly in Fig. A.2 and accompany the electronic schematic in Fig. A.5.

The single board computer and hardware interface board were fixed to the same layer on the robot. This setup was intended to keep logic-level components separated from the higher voltage connections. The 3D printed layer is shown unpopulated in Fig. 4.4a and with the boards attached in Fig. 4.4b.

A Pololu VL53L0X time-of-flight distance sensor was included to provide distance

Figure 4.4: The robot logic-level boards were fixed to the same layer. The bare frame is shown in (a) along with the attached boards in (b).

measurements from the robot to ground while flying. This information would be necessary for implementing an autonomous takeoff and landing routine. The sensor uses the time-of-flight of an infrared laser to measure distances up to 2 m at a 1 mm resolution. Fig. 4.5 shows the sensor embedded in the bottom cover of the robot.

Figure 4.5: The Pololu VL53L0X time-of-flight distance sensor fixed to the bottom cover of the robot. This sensor was included to provide the robot's distance to the ground during takeoff and landing.

The second gray block in Fig. 4.1 is the IR sensor ring. This PCB was designed to attach underneath a thrust motor and features a circular array of alternating IR detector and emitter components. By shining IR light around the motor and detecting the intensity it reflects back, a profile of the propeller directions can be determined. This information is necessary to verify the propellers were appropriately folded during the transition from flying to walking mode. Twelve emitter-detector pairs were used to provide an angular accuracy that ensures propellers can be detected even while in the sensor gaps. Fig. 4.6a shows the top of the PCB where the IR components are supported by a spacer 3D printed in the translucent material Taulman T-Glase. The spacer places the IR components close to the propellers while the translucency allows it to diffuse light from eight RGB LEDs on the board beneath it. The RGB LEDs modulate intensity and color to notify a user of errors or changes in robot state.

Fig. 4.6b shows the back of the IR sensor ring PCB. Three ICs are visible: two Texas

Figure 4.6: The IR sensor ring has a translucent spacer on the top (a) to support the IR components close to the thrust motor propellers and also diffuse light from RGB LEDs below it. The analog-to-digital converter and LED driver ICs are soldered to the back side of the PCB (b).

Instruments TLC59116 Constant-Current LED Sink Drivers that directly power the IR and RGB LEDs, and a Maxim Integrated MAX11611 analog-to-digital converter (ADC) for measuring the intensity of reflected IR light. The IR detectors are wired in voltage divider networks so that the measured voltage increases when more IR light is detected. These main ICs use the digital I²C protocol for communication, allowing control of the entire PCB with only four wires: power, ground, clock, and data. Since all four PCBs share identical I²C addresses, an I²C multiplexer is employed to ensure only one ring is connected at a time. The PCB layers are shown more clearly in Fig. A.3 and accompany the electronic schematic in Fig. A.6.

A block diagram of the power distribution in the robot is provided in Fig. 4.7. The lines connecting blocks represent the voltage level on the associated rail. Power on the robot is generated by two Zippy Compact 3S LiPo batteries with 6200 mAH capacity and

Figure 4.7: A block diagram of the power distribution in the robot with lines indicating the different voltage rails.

40 C discharge capability. This battery type is shown in Fig. 4.8 and the two are connected in series to produce 22.2 V in a 6S equivalent configuration. The battery voltage is nominally 22.2 V (3.7 V per cell), but when fully charged reaches 25.2 V (4.2 V per cell). The discharge maximum of 40 C (248 A) is sufficient for flying with the 40 A maximum ESCs.

The battery wires route into the power distribution PCB in Fig. 4.9—the third gray block

Figure 4.8: Two Zippy Compact 3S LiPo batteries are connected in series to power the robot.

Figure 4.9: The power distribution PCB facilitates distributing the four voltage rails to the peripherals and contains circuitry to monitor the power consumption of each rail. The top (a) and bottom (b) are shown.

in the Fig. 4.1 diagram. The leads screw into a Molex 38969 Series terminal block rated for 50 A. The direct battery voltage is routed through 15A terminal blocks on the edges of the board to the thrust motor ESCs. The battery also provides the input voltage for the 3.3 V, 5 V, and 12 V voltage regulators generating the voltage rails for the remaining electronics. The logic level Pololu 3.3 V and 5 V regulators are soldered directly onto the bottom of the PCB in Fig. 4.9a. Originally these regulators plugged into headers on the board to allow easy disassembly, but it was discovered that if a regulator was plugged in while the power was on it could induce a voltage spike of the input voltage on the logic level output. Fig. 4.10 shows the output of one regulator spiking to 22.2 V before settling to 3.3 V. This spike would be sufficient to burn out the connected circuitry, so the regulators were soldered in place and transient voltage suppression diodes were added to these two voltage rails.

An additional DROK adjustable voltage regulator in Fig. 4.11a was included separate from the power distribution board to provide 12 V for the servo motors. The voltage was

Figure 4.10: The Pololu 3.3 V and 5 V regulators can spike the input voltage on the regulated output if plugged in while the system is powered. This graph shows a brief 22.2 V spike before the output settles to 3.3 V.

distributed to the servos using two Trossen Robotics 6 Port AX/MX Power Hubs shown in Fig. 4.11b. The 3.3 V, 5 V, and 12 V voltage regulator models were selected for their respective 2.6 A, 2.5 A, and 8 A continuous current draw capabilities that would be sufficient for each of their connected circuitry. Each regulator sourced an input directly from the

(a)

(b)

Figure 4.11: A DROK adjustable voltage regulator (a) was used to generate the 12 V supply for the servo motors. The AX-12A servos plug into two Trossen Robotics 6 Port AX/MX Power Hubs (b) to receive power and connect data lines.

battery to prevent a cascading brownout scenario if the regulators were tiered. This configuration came at the expense of slightly lower efficiency from the larger input-to-output voltage gap.

Circuitry was included on the power distribution PCB to monitor the power consumption of all four voltage rails. Four TI INA219B Power Monitor ICs were used in conjunction with shunt resistors to achieve this goal. The chip contains an ADC to measure the voltage over the shunt resistor and can provide the bus voltage, current flow, and power consumption values. The shunt resistors can be seen on the back of the PCB in Fig. 4.9b. The INA219B chips contain an internal programmable gain amplifier, but the lowest unity gain level of ± 40 mV was used for the lowest power dissipation by the shunt resistors. Two $1\text{ m}\Omega$, 3 W shunt resistors were used in series for the main 22.2 V rail. These components were placed before the rail is distributed so the measured power consumption applies to the full robot. A shunt resistor of $3\text{ m}\Omega$, 2 W was used for the 12 V rail and $15\text{ m}\Omega$, 1 W resistors were used on the 3.3 V and 5 V rails. Large power planes were used on the PCB to accommodate the high current demands of the 22.2 V and 12 V rails. Thermal relief spokes to through-hole pads were removed to preserve high current integrity. The PCB layers are shown more clearly in Fig. A.4 and accompany the electronic schematic in Fig. A.7.

The power distribution board, 12 V regulator, and servo hub were fixed to the same layer on the robot. This setup was intended to keep high-voltage components separated from the logic-level processors. The 3D printed layer is shown unpopulated in Fig. 4.12a and with the boards attached in Fig. 4.12b.

(a)

(b)

Figure 4.12: The power distribution electronics were fixed to the same robot layer. The bare frame is shown in (a) along with the attached boards in (b).

4.2 Structural Components

The structural components of the robot were designed in the CAD software Autodesk Inventor Professional 2015. The entire robot was modeled, including components that were not 3D printed such as the servos and flight hardware. Pieces to be 3D printed were exported as *.stl* files and imported into the MakerBot Desktop software to prepare for printing on a MakerBot Replicator 2X. The print orientation for each part was intelligently decided to provide resilience against the incurred forces from walking and flying. All parts were printed using ABS for the best durability, except for the central body frame that was printed with PLA to avoid the part warping. Most parts were printed with only ten percent infill to reduce their weight without significantly altering the part strength. Appendix B provides supplementary pictures of the process to 3D print the largest part—the central body frame of the robot.

4.2.1 Thrust Legs

The principal design feature meshing the walking and flying systems is the thrust legs with embedded flight hardware. These legs share a similar structure to those of the hexapods in Fig. 2.2, but include certain specific adaptations. Fig. 4.13 shows a thrust leg in walking stance, where the thrust motor ① mounts to the upper tibia segment ④ on the opposing side from where the tibia contacts the ground. The thrust motor ESC ⑤ is embedded behind the motor in this hollow top segment. The tibia stem ⑦ is sized so that the robot can walk with the propellers ② folded inside the femur but unable to touch the ground. Walking traction is improved by a rubber cap ⑧ screwed to the tibia endpoint.

Figure 4.13: A 3D printed thrust leg with callouts to specific tibia components. Walking stance is pictured, where the propellers are folded inside of the bifurcated femur.

Since the incorporation of flight hardware into the leg is a key design attribute, figures are provided to elaborate on the assembly process of the associated components. First a protective cap was fixed to the back of the IR sensor ring. Fig. 4.14a shows the cap next to the back of the PCB. The cap was designed to protect the PCB circuitry while keeping the cable secured in place. The part was sized to snap-fit around the PCB edges, as shown in

Figure 4.14: A protective cap (a) was 3D printed to snap to the back of the IR sensor ring PCB (b).

Fig. 4.14b. Before being fixed to the upper tibia segment in Fig. 4.15, the IR sensor ring cable and thrust motor wires were routed through a gap in the top of the piece. Four M3, 16 mm screws were preemptively set in the mounting holes. The screws were first inserted

Figure 4.15: The upper tibia segment (a) has holes for mounting to the thrust motor and tibia servo. Before screwing the thrust motor in, the motor wires and IR sensor ring cable must be routed through a hole in the top of the piece (b).

Figure 4.16: The IR sensor ring is attached to the upper tibia segment first (a), along with a small spacer. The thrust motor screws in on top of these pieces (b).

through the holes in the IR sensor ring PCB, as in Fig. 4.16a. A small 3D printed spacer was added next to separate the thrust motor from the PCB face, avoiding the motor from shorting the electrical contacts and providing a gap for airflow through the vents on the motor underside. The thrust motor in Fig. 4.16b was screwed in on top of the spacer and its wires were pulled taut, residing in grooves in the IR sensor ring spacer. With the flight hardware mounted, the ESC in Fig. 4.17a was connected to the motor wires on the inside of the part. At this point, the thrust motor was actuated to verify the ESC was connected properly and ensure the motor spun in the intended direction. The ESC was finally snapped into the upper tibia segment in Fig. 4.17b and the IR sensor ring cable and ESC power and control wires were routed out of the part through a channel in the side.

Figure 4.17: The motor wires are plugged into the ESC (a) before it is snapped inside of the upper tibia segment (b). The IR sensor ring cable, and ESC power and control wires are routed out of the segment through a channel in the side.

The bifurcated femur design in Fig. 4.18 provides a central gap that the upper tibia segment can rotate through—a process necessary for transitioning between walking and flying mode. The ability to rotate in this manner required the femur segments to attach to the wheel of the tibia servo. The upper tibia segment width was kept slim to not obstruct the rotation and the femur gap was sized to allow the thrust motor and IR sensor ring to pass. At the base end of the femur, the two sides screw directly into the femur servo housing

①. A previous femur design had this end connect to the femur servo wheel instead, but having both ends connected to servo wheels allowed undesired radial twisting in the femur

Figure 4.18: The femur section of a thrust leg with callouts to specific femur components. The gap in the bifurcated femur allows the upper tibia portion to rotate inside it.

that would have impaired walking ability. Mounting the base to the servo housing greatly reduced this problem.

The asymmetric bulge of the femur provides adequate space for the wide propeller lobes to rotate through. Fig. 4.18 pictures CW-spinning flight hardware, so the right femur segment (3) spans wider than the left one (2). This is mirrored for CCW hardware, since the propeller lobes are on the opposite side. Regardless of the spin direction, a wire guard (4) protects the servo, ESC, and IR sensor ring wires (5) as they route along the right femur segment (3).

Fig. 4.19 depicts the thrust leg in a flying stance. The end of the tibia slots into a pocket (1) at the base of the leg and remains wedged inside with upward thrust force while flying. A ledge inside the pocket prevents tibia movement vertically and the pocket opening is restricted to prevent horizontal motion. Ultimately the leg remains rigid without needing to actuate the leg servos. The necessity to rest in this pocket is the primary reason for making

Figure 4.19: A thrust leg in flying stance with callouts to specific coxa and leg-base components. Flying stance involves the tibia slotting into a pocket at the base of the leg, where upward thrust force keeps the leg rigidly aligned without needing to actuate the leg servos.

the tibia straighter than those of the hexapods in Fig. 2.2. Increased tibia curvature would require a longer tibia length to reach the pocket and increase the fixed distance from the propellers to the tibia. This would constrain the tibia angular limits and result in a reduced range of walking motion.

The tibia pocket ① is fixed to a segment at base of the leg ② that connects to the body. This segment fits around a keyed protrusion on the body and secures with a single large screw. The coxa segment ⑥ connects to the wheel of both the coxa servo ④ and femur servo, and is sized for each to have a $\pm\pi/2$ rad range of rotation. A wire guard ⑤ along the coxa ⑥ keeps the wires from getting crimped in the joints. An additional wire guard ③ below the body connector streamlines the wires to fit through holes in the bottom cover of the robot.

4.2.2 Strut Legs

The two middle legs of the hexapod-quadcopter are absent of flight hardware and referred to as strut legs. One of the strut legs is shown in Fig. 4.20. Since no flight hardware is necessary, the upper tibia segment is removed along with the tibia pocket at the base of the leg. The same coxa components and tibia stem remain, but the femur is consolidated to a single segment for even further improved resilience to twisting. The femur wire guard profile was reduced since only the servo wires route along the leg.

The name *strut leg* derives from the supplementary function of these two legs supporting the robot in flying mode. Fig. 4.21 captures how the strut legs reorient to support the entire

Figure 4.20: A strut leg of the hexapod-quadcopter. The strut legs share the same design as the thrust legs with the exception of a unified femur segment and the absence of flight hardware.

Figure 4.21: Both strut legs reorient to support the entire robot, resting on the long tibia segments. The angle of the femur segment is limited mechanically against the tibia, removing the high-torque lifting demand from the tibia servo.

robot between the two of them. The tibia segment serves as the actual strut against the ground, and the femur segment is rotated to its physical limit against the tibia—removing the high-torque lifting demand from the tibia servo. The two coxa servos are rotated to opposing angles so the robot weight is balanced without imposing a large torque on the coxa or femur servos. The robot can stay supported even when the strut leg servos are deactuated.

4.2.3 Body

The body of the hexapod-quadcopter was assembled in several layers. The 3D printed piece in Fig. 4.22 served as the central body frame for attaching the legs and body components. This part was printed using PLA to ensure the edges would not warp up and change the angle in which the legs attach. As mentioned previously, Appendix B describes the process to 3D print this part. The frame ① is encircled by six leg connector protrusions ② that

Figure 4.22: The robot body frame with callouts to notable features.

insert into the leg base pieces. The protrusions are chamfered to slide into the leg cavity easily while still maintaining a tight fit. Since only a single screw secures each leg in place, they can easily be removed to make adjustments or perform maintenance. A seventh protrusion ⑥ was added on the front for connecting extra hardware such as a camera. Several channels ③ were included around the frame for routing wires from the legs and bottom layer to the top layer electronics. Two holes ④ were placed at the back for power switches to snap into. A large space was left in the center of the frame for the two LiPo batteries, with ledges ⑤ at the bottom for support.

Some components that connect to the frame are included in Fig. 4.23. A servo was connected to the front mount protrusion for the option to swivel added hardware or sensors. Two 10 A power switches were snapped into the square holes at the back of the frame. The two LiPo batteries were placed side by side symmetrically in the frame to evenly distribute their weight. The battery wires and balance charge cables were directed to the

Figure 4.23: The robot body frame with included middle components.

gap underneath the batteries. Fig. 4.24 shows the battery and switch wire connections on the frame underside. The main battery cables are connected in series for a 6S-equivalent configuration. The 3S balance charge cables are similarly consolidated to a 6S plug so the two batteries can be charged simultaneously. The positive battery terminal routes through the first switch, to the second, and out to be screwed into the power distribution board. An extra connector cable is added in between the two switches to provide unique functional capabilities depending how the two switches are configured. If both are switched on, the robot is powered by the internal batteries. If the first switch is turned off and the second is on, the extra connector can be used to power the robot from an external power source. Finally, if the first switch is turned on and the second one is off, the extra connector can be used to charge the batteries without unplugging them. Since the switches are only rated for 10 A, dual connections were made using both terminals. The power capability was deemed

Figure 4.24: The battery and switch wires connect on the underside of the body frame. The batteries are connected in series to generate 22.2 V.

sufficient for the robot, especially considering the switches would not be turned off while flying—the highest current state.

The power distribution electronics layer from Fig. 4.12b screws into the bottom of the body frame as shown in Fig. 4.25. The power and ground battery cables screw into the power distribution PCB to be distributed to the rest of the electronics system. The bottom cover in Fig. 4.26 was designed to protect the bottom electronics. The cover ② rests along the outside of the body frame and has grooves cut out at each leg for bringing wires into the robot. The laser range sensor ④ in Fig. 4.5 screws onto a cutout in the bottom of the cover. The entire cover is able to swivel open and snap closed on the robot. The front swivel ① and back clip ⑥ are small 3D printed pieces screwed into place, ensuring that if either breaks the entire cover will not need to be reprinted. Alternatively, the cover can be

Figure 4.25: The power distribution electronics layer in Fig. 4.12b screwed onto the bottom of the body frame.

Figure 4.26: A cover was 3D printed to protect the bottom electronics on the robot.

fixed into place by screwing through holes (5) in the back. A central hole (3) is included in the cover to allow the robot to be flight tested on a rig with a rope through its center.

With the bottom electronics mounted to the frame, the legs and bottom cover were

Figure 4.27: The robot bottom with legs and bottom cover attached. Wires are brought in from the legs and attached on the bottom layer or routed through body channels to reach the top electronics.

screwed into place. Fig. 4.27 shows the bottom of the robot with these attached peripherals. The leg servo wires and ESC power cables are directed to the bottom layer whereas the IR sensor ring and ESC data cables are routed through the body channels to the top of the robot. Other cables sent to the top include the I²C connections for the laser range sensor and power distribution board, the servo data cable, and the 5.0 V power for the ODROID-C1+.

The robot was turned back around and the logic-level electronics layer in Fig. 4.4b was screwed into the top of the body frame. This state is depicted in Fig. 4.28. The thrust leg tibia pockets were then screwed into place, secured by the same screws that connect the associated legs. Afterward, the remaining cables were connected into the hardware interface PCB and ODROID-C1+. Fig. 4.29 shows the robot with these parts in place.

Figure 4.28: The logic-level electronics layer in Fig. 4.4b screwed onto the top of the body frame.

Figure 4.29: The top of the robot with the thrust leg tibia pockets attached and wires connected.

4.2.4 Robot

The full hexapod-quadcopter robot is pictured in Fig. 4.30 standing in a walking stance. The folding propellers stay constrained between the bifurcated femur segments in this pose. The range of walking motion is relatively unimpeded with the propellers in this location, allowing the robot to maintain all normal hexapod functions despite the added hardware. Fig. 4.31 shows the robot in a quadcopter “X” configuration with the middle two strut legs elevating the body for takeoff and landing. The tibias of the thrust legs rest in pockets at

Figure 4.30: The full hexapod-quadcopter robot in walking stance. The propellers remain constrained inside the femur segments to prevent obstructing the walking motion.

Figure 4.31: The full hexapod-quadcopter robot in flying stance. While flying, the thrust legs remain secure in pockets at their leg base.

the base of each leg, where upward thrust force keeps the legs secured without needing to actuate the joint servos. The pockets are angled such that the thrust motors are symmetric distances laterally and longitudinally, despite the body being oblong. Weight in the body is evenly distributed to ensure roll and pitch responsiveness match closely.

4.3 Software

An embedded Linux computer and microcontroller were incorporated in the robot to handle the software processing. The Teensy 3.2 microcontroller ran code to read in from the robot sensors and control the actuators. The MCU was connected over serial to the

ODROID-C1+ computer running higher-level algorithms in ROS. The detailed software implementation for the MCU is explained in 4.3.1 and for ROS is described in 4.3.2.

4.3.1 Microcontroller

The Teensy 3.2 MCU was programmed using the Teensyduino add-on for the Arduino integrated development environment (IDE). Before programming the main code that interfaced with ROS, a C++ object-oriented backend was developed that loosely mirrored the hardware objects on the actual robot. The class representing the full robot provided general public functions for reading from sensors, writing to the servos and thrust motors, and configuring the system. Publicly available libraries were utilized for the AX-12A servo control, TLC59116 LED drivers, IMU, altimeter, laser range sensor, and INA219 power monitoring ICs. Other libraries were developed to interface with the PCA9547 I²C multiplexer, MAX11611 ADC, and IR sensor rings as a whole. These libraries provided classes for the associated hardware and general configuration and utility functions that dissociated the low-level I²C register communication from the user. The member objects in the robot class were kept private to disallow their direct editing. This provided a layer of protection against mismanagement of the hardware. For example, the IR LEDs could be left on without being visibly detected and draw over an amp of current in the background. Since the IR sensor ring object is private, however, the IR LEDs are not directly accessible and only turn on when polling the ring ADC.

The main MCU code used the developed C++ robot library and consisted of a setup function followed by a continuously repeating processing loop. The code was designed to interface with ROS using a library exported from the *rosserial_arduino* package. In the

setup function, the robot is first initialized to configure all the sensor chips. Afterward, the data publishers and services are advertised to ROS and other topics are subscribed to. The robot waits for the node handle to connect to ROS and then pings the I²C ICs and servos to verify they are connected. This step is taken after the ROS connection is initiated so that error messages can be viewed in the ROS console if hardware is not detected. The software then enters a repeating loop where Metro objects associated with each sensor track when they should be polled and published to ROS. The Metro objects store the polling interval and were edited to include the ability to disable polling altogether. A time stamp was included in the various custom sensor messages to record when each sensor was actually polled. A service was provided to allow nodes in ROS to dynamically change the polling rate of any sensor. ROS services act similarly to remote procedure calls, and can be used to request and receive data or trigger a function to run. Other services coded in the microcontroller allow for configuring the RGB LED modulation and color, reading the servo sensors, and configuring the servos to disable their torque output or limit their speed. When messages are published on a subscribed topic, callback functions are triggered to perform an operation—such as writing out servo and thrust motor values.

4.3.2 ROS

The high-level algorithms for the robot were implemented using ROS Indigo on the ODROID-C1+ embedded Linux computer running Ubuntu 14.04 operating system. The software architecture for controlling walking and flying used a mix of publicly available ROS packages and ones specifically coded for the tasks. For clarity, the node graphs for walking

and flying capability are separated into Fig. 4.32 and Fig. 4.33, respectively. In these figures, ovals represent independently running nodes and rectangles are the information topics transmitted between them.

For walking and hexapod processes in Fig. 4.32, robot control originates from the `/xbox` node. This node is an instance of the `joy_node` from the ROS *joy* package and reads values in from an Xbox controller. The game controller button and axes values are forwarded to the `/teleop` node on the `/cmd` topic. The `/teleop` node serves as a state machine for the robot, keeping track of state changes and determining how to interpret user input depending on the current state. Several of the nodes the `/teleop` node interacts with are action servers using the ROS *actionlib*. Action servers are commonly used for tasks that take an extended period of time to execute and may need to be preempted or canceled. The *actionlib* interface allows the user to start the action by sending a goal to the server, which can publish periodic feedback and the goal state after completion. In the robot program, a goal is sent to the `/stand` server with the desired standing body height and leg spread. The server updates the robot pose to stand up from a sitting position. The new robot body frame and leg vector endpoints are published on the `/pose` topic to the `/IK` node, which performs the inverse kinematics calculations to determine the leg joint angles. The joint set is published on `/joints` to the `/servos` node, which maps the values to the associated 10-bit servo position. The Teensy 3.2 receives the servo positions on the `/servos` topic via a 1 Mbaud serial connection from the `/teensy` node, a `serial_node.py` instance from the ROS *rosserial_python* package.

After standing, the robot switches to walking mode and linear and angular velocities

are sent to the `/walk` node on the `/vel` topic. The walking algorithm updates the robot pose to advance forward. Whenever joint values are updated from the `/IK` node, new frame transformations for the robot are published on the `/tf` topic. Incorporating the *tf* package for frame transformations allowed frames to be looked up by any node since the current robot pose is not stored in any one node. As the arrows indicate, the `/walk` node looks up the robot frames on `/tf` before updating them to walk forward. The robot uses the `/hex_to_quad` server to transition from walking to flying mode. The opposing transition is handled by the `/quad_to_hex` server.

For flying processes in Fig. 4.33, user input similarly generates from the `/xbox` node sending messages to the `/teleop` node. Instead of robot velocities, the `/teleop` node publishes the desired angles for roll and pitch and rates of change for yaw and altitude to the corresponding PID nodes. PID nodes are instances of the controller node from the ROS *pid* package. The `/teensy` node publishes the sensor measurements for the accelerometer and gyroscope on the `/imu/data_raw` topic, magnetometer on the `/imu/mag` topic, and altimeter on the `/altimeter` topic. The IMU measurements are filtered through the `/imu_filter` node that is a `complimentary_filter_node` from the ROS *imu_complimentary_filter* package. The filtered IMU values are published on `/imu/data` along with the RPY angles on `/imu/rpy/filtered`. Since these messages are formatted as `sensor_msgs/Imu` and `geometry_msgs/Vector3Stamped` respectively, they cannot be directly forwarded to the PID inputs. The `/parser` node splits the messages and publishes the individual values in `std_msgs/Float64` messages. The roll and pitch axes use an angle and rate PID, but the yaw and altitude control only require rate PIDs. PID effects are routed to the `/fly` node where they are mapped

to differential motor changes for RPY and a thrust setpoint for altitude. The thrust motor levels are sent on the /motors topic to be set by the microcontroller.

Figure 4.32: ROS node graph of the walking and hexapod processes.

Chapter 5

Results

The following sections discuss testing for the robot functionality. Section 5.1 includes results for walking and flying on a secure test rig. Section 5.2 explains the procedures developed to allow the robot to autonomously transition between walking and flying modes.

5.1 Locomotion

Walking and flying algorithms were developed and tested as a proof-of-concept that the hexapod-quadcopter is capable of each. In order to walk, analytical inverse kinematic equations were developed and use in conjunction with leg endpoint trajectory planning. Flight control was implemented as described in Section 4.3.2 and tested on a custom built rig.

5.1.1 Walking

The robot inverse kinematic equations were adapted from equations in [8] and reference angles in Fig. 2.3. The three leg angles are solved in (5.1), (5.2), and (5.3).

$$\gamma = \arctan\left(\frac{{}^cT_{Ey}}{{}^cT_{Ex}}\right) \quad (5.1)$$

$$\alpha = \arctan\left(-\frac{{}^F T_{Ex}}{{}^F T_{Ey}}\right) + \arccos\left(\frac{|{}^T T_E|^2 - |{}^F T_T|^2 - |{}^F T_E|^2}{-2|{}^F T_T||{}^F T_E|}\right) \quad (5.2)$$

$$\beta = \arccos\left(\frac{|{}^F T_E|^2 - |{}^T T_E|^2 - |{}^F T_T|^2}{-2|{}^T T_E||{}^F T_T|}\right) \quad (5.3)$$

The main difference between these equations and the original text is the linkage lengths have been replaced with frame transformation notation. This notation takes the form ${}^A T_B$, indicating a transformation from frame A to frame B . $|{}^A T_B|$ is used to represent the vector distance between A and B frame origins. In (5.1), (5.2), and (5.3) the variables C , F , T , and E refer to the coxa frame, femur frame, tibia frame, and tibia endpoint frame respectively. The original inverse kinematics only applied when the body remained parallel to the ground, but taking frame transformations into account has expanded the use to any body orientation. The inverse kinematics for the robot can be solved or determined nonexistent as long as the body frame and leg vector endpoints are provided.

The ROS visualization tool Rviz was used to view the robot frames published while moving. Fig. 5.1 shows the frames of the robot in Rviz while standing. Since the frames are updated before writing to the servos instead of by reading the servo positions, the robot frames can be updated without actually writing out to the robot. Walking development was tested in this manner by observing how the frames changed in Rviz before actuating the servos. This provided a level of protection against damaging the robot if algorithms were incorrectly implemented.

In order to walk the robot forward, the received linear and angular velocities were used to increment the /robot frame relative to the localized /world origin point. The /robot frame

served as a projection of the robot center onto the ground and was advanced by a transformation appropriate to the velocities and the time since last updated. This frame was purposefully separate from the robot /body frame to allow different body transformations while performing the walking routine. After incrementing the /robot frame forward, the supporting and extending legs were updated. The endpoints of supporting legs were locked to their positions relative to the /world frame. Extending legs lifted up and moved to new positions to support the robot. The new positions were determined by calculating where the /robot frame would be in the next gait cycle given the current velocities. The robot was tested with a wave gait to provide the most support and stability.

Fig. 5.2 shows the wave gait algorithm being tested by the robot. During this test the graph in Fig. 5.3 was recorded showing the power consumption of the servos. In the graph the 7 W flatline from 165-172 s reflects the quiescent power consumption of the servo

Figure 5.2: The robot performing a wave gait walking test.

electronics while not actuated. The robot stood up from 172-176 s, with the large peak at 175 s indicating when the body was lifted. The ranges 176-178 s and 202-205 s were when the robot was just standing and not moving. The wave gait movement occurred in the middle from 178-202 s. Large peaks likely correspond to when a new leg lifted to advance forward. The walking process used an average of about 40 W to move at 4 in/s.

Figure 5.3: A graph of the servo power consumption while testing a wave gait.

5.1.2 Flying

A wooden rig was built to test the flight control algorithm safely. Fig. 5.4a shows the large wooden arch with a nylon rope fixed down the middle. The rope routes through the center of the robot to prevent severe angles and drifting. The robot tested in Fig. 5.4b was the first design of the hexapod-quadcopter. Due to time constraints, the final robot was not able to be flight tested.

The flight control was set up as described in Section 4.3.2. The RPY inputs were generated from an Xbox controller but due to the constrained test rig the thrust point was set through a user interface. The robot yaw angle was kept constant to prevent tangling a connected communication cable. Testing verified that the robot could maintain a constant height and heading, level itself to hover, and react to roll and pitch angle changes. The PIDs were tuned to optimize the response of the robot. Testing proved the effectiveness of the flight control implementation but further work would be required to bring the robot to

Figure 5.4: A large wooden arch was built to test the flight control on the robot safely (a). A nylon rope routes through the center of the robot to prevent extreme angles and drifting (b).

Figure 5.5: The robot hovering on the flight test rig.

a free-flight ready state. Since the final version of the hexapod-quadcopter was not flight tested, a graph of the power consumption is not available for comparison to walking.

5.2 Transitional Procedures

System-specific transitional procedures were developed for the robot to safely and autonomously switch between walking and flying modes. The walking-to-flying transition in Section 5.2.1 discusses resting the robot on the strut legs and folding the thrust legs into a quadcopter configuration. The flying-to-walking transition in Section 5.2.2 is slightly more involved, with the need to manually fold the propellers together before standing up.

5.2.1 Walking-to-Flying

In order to change from walking to flying mode, the robot first rests on the two strut legs. This is accomplished with inverse kinematics by lifting the body higher than the normal standing height, rotating the strut legs sideways into position, and bringing the body back

down until the strut legs contact the ground. Fig. 5.6 shows the robot in this position. From that point, forward kinematics are used to directly bring the thrust leg tibias into their leg pockets. Preprogrammed angle sequences are sent to the leg servos for this task. The robot completes the process in the Fig. 4.31 flight mode.

Figure 5.6: The robot resting on the strut legs before rotating the thrust leg tibias into their pockets.

5.2.2 Flying-to-Walking

The flying to walking transition is more involved than the formerly mentioned. The primary concern, as seen in Fig. 4.31, is that after the robot finishes flying the propellers are not at known angles. This fact prevents the tibia from simply rotating back into walking mode. The solution starts with each thrust leg releasing from its pocket, then slowly rotating the

Figure 5.7: The thrust legs rotate the propellers slowly and dip down, colliding them with the pocket housing which folds them together from angular momentum.

propellers to induce enough rotational momentum that when the tibia dips down as in Fig. 5.7, the propellers hit the pocket housing and fold together with that momentum. Accurate folding is verified with the IR sensor ring.

The IR sensor rings are calibrated to remove channel scaling inconsistencies and offsets. The minimum and maximum measurements are taken by reading all twelve detector channels in the absence of IR light, and again with a reflective ring above the thrust motor and the IR LEDs turned on. These values are stored for each channel and future measurements are mapped from this range to 0-1 to remove scaling discrepancies. Afterward, any global DC offset is removed from the channels and they are rescaled. This static offset is generally caused by ambient IR light.

The filtered values are curve fit to a bimodal von Mises distribution in ROS using non-linear least-squares fitting functions from the GNU Scientific Library. The von Mises distribution, in (5.4), is an approximation of the wrapped normal distribution. The wrapped nature of the function allows the curve fit to work around the entire ring without causing problems at the angular limits.

$$f(x|\mu, \kappa) = \frac{e^{\kappa \cos(x-\mu)}}{2\pi I_0(\kappa)} \quad (5.4)$$

In (5.4), $I_0(\kappa)$ refers to the modified Bessel function of order 0. Since two propellers need to be detected, a bimodal von Mises distribution is used to curve fit and takes the form of (5.5).

Figure 5.8: Visualization of the bimodal von Mises curve fit to the normalized IR ring sensor data. The left plot shows the Cartesian results and the right plot maps the polar coordinates around a circle representing the thrust motor. The two gray lines in the circle are the calculated curve means, and the green arc represents a range where the propellers are considered appropriately folded.

$$f(x|\mu_0, \kappa_0, \mu_1, \kappa_1) = \frac{e^{\kappa_0 \cos(x-\mu_0)}}{2\pi I_0(\kappa_0)} + \frac{e^{\kappa_1 \cos(x-\mu_1)}}{2\pi I_0(\kappa_1)} \quad (5.5)$$

The variables of interest after curve fitting are μ_0 and μ_1 , corresponding to the means of the von Mises curves. These values indicate to an appropriate accuracy the angular positions of the propellers. Fig. 5.8 shows the bimodal von Mises curve fit for one particular data set. The left plot depicts the results in Cartesian coordinates, and the right plot shows the fit in polar coordinates around the thrust motor. The algorithm can be adapted to include the κ constants and the goodness of fit value to disregard poorly fitting data.

After verifying that the propellers were appropriately folded, the tibia quickly inverts,

Figure 5.9: After folding the propellers, the tibia inverts to rest them against protrusions on the femur sides. At this angle, the propellers will automatically fold together when the tibia rotates further into the femur.

moving into the position in Fig. 5.9. Here, the propellers may fall down to rest on protrusions from the femur segments. At this angle, the tibia is free to continue rotating through the femur, as in Fig. 5.10, and the propellers are fully folded between the two sides. The robot stands using the four thrust legs and rotates the middle legs back into walking position.

Figure 5.10: The propellers are drawn together between the two sides of the femur.

Chapter 6

Conclusions

The hexapod-quadcopter designed in this thesis uniquely incorporates multicopter flight hardware directly into the robot legs. This novel approach improved upon similar robots that attached independent, off-the-shelf multipods and multicopters. The 3D printed structural components were designed appropriately to withstand the stresses of walking and flying. No parts broke during testing of the robot functionality. The final robot layout provided a compact designed with integrated modularity for expanding and upgrading the system components. Additionally, significant effort was put toward making the robot design aesthetically pleasing, with a clean all black color scheme, filleted edges, and intelligent wire routing and protection in the legs.

The software implementation for the robot functionality successfully proved the design capability for multimodal locomotion and autonomous transitional management. The microcontroller program provided a simple interface to control the robot hardware from ROS. The ROS architecture was designed as a network of several independent distributed processes, allowing new nodes and procedures to easily be incorporated in the system. Walking tests for the robot validated the inverse kinematics implementation and performance of a wave gait. Flying was tested on a secure rig and showed the ability of the

robot to hover while maintaining a constant height and heading, as well as react to roll and pitch angle inputs using PID control. Further improvement would be necessary to bring the flight control to an autonomous or free-flight ready state. In hindsight, a commercial flight controller would be preferred over the custom implementation. The transitional procedures developed for the robot incorporate unique elements including using two legs as struts to support the robot and manually folding the propellers together. These procedures were proven to be valid options but require further development to improve reliability for autonomous operation.

Due to the heavy weight of the robot, extended walking will likely cause the servos to overheat. The maximum flight time, although not tested, is likely much less than other multicopters that do not include the added weight of servos and other components used in this robot. Testing could be performed to measure the trade-off between added weight in battery capacity and maximum flight time. The robot could potentially be optimized for the intended performance, with more battery capacity included for predominantly flying operation or lower battery capacity if the robot will mostly walk and only fly in short bursts.

The infrared sensor rings designed to detect the propeller angles work indoors but are rendered ineffective outside due to high levels of ambient infrared light. A new detection method would have to be incorporated for outdoor navigation. One solution could be embedding small cameras in the thrust legs and running image processing to identify the propeller locations. The infrared LEDs could remain under the thrust motors to illuminate the propellers in the dark for the cameras.

Chapter 7

Future Work

The hexapod-quadcopter platform is well suited to several areas of research. The robot could be used to explore the implementation and optimization of multimodal navigation. With added hardware, the robot could perform localization and mapping as part of the navigation process. The robot could fly in an area and gather an elevation map, then land and walk around using and updating the map. The power consumption sensors and servo information could be used to inform gait style or affect a rule-based free gait. Before implementing high level algorithms on the robot, the basic locomotion and transitional procedures would need to be developed further for autonomous reliability.

Bibliography

- [1] A. Chiuariu. (2012, Dec.) Hexapod hexacopter robot comes in a nightmarish design. [Online]. Available: <http://walyou.com/hexapod-hexacopter-robot/>
- [2] (2013) Birth of launch transformation quadrobot “dji naza gps & kondo multi legged robot”. [Online]. Available: <http://plaza.rakuten.co.jp/raizzien/diary/201304220000/>
- [3] (2012) Hexawalker. MadLab Industries. [Online]. Available: <https://madlabindustries.com/hexawalker>
- [4] F. Tedeschi and G. Carbone, “Design issues for hexapod walking robots,” *Robotics*, vol. 3, no. 2, pp. 181–206, 2014. [Online]. Available: <http://www.mdpi.com/2218-6581/3/2/181/htm>
- [5] N. Lanxon. (2014, Oct.) Giant 3d-printed robot spider sees world through an ipad. [Online]. Available: <http://www.wired.co.uk/article/giant-3d-printed-robot-spider>
- [6] M. Bunting and J. Sprinkle, “Rapid prototyping of dmitri, a hexapod robot,” in *Proc. of Robotics: Science and Systems Conf.*, 2014.
- [7] (2017) Phantomx ax metal hexapod mark iii kit. [Online]. Available: <http://www.trossenrobotics.com/phantomx-ax-hexapod.aspx>
- [8] O. Liang. (2013) Inverse kinematics tutorial introduction. [Online]. Available: <https://oscarliang.com/inverse-kinematics-and-trigonometry-basics/>
- [9] X. Duan, W. Chen, S. Yu, and J. Liu, “Tripod gaits planning and kinematics analysis of a hexapod robot,” in *2009 IEEE International Conference on Control and Automation*, Dec 2009, pp. 1850–1855.
- [10] M. Zak and J. Rozman, “Design, construction and control of hexapod walking robot,” in *2015 IEEE 13th International Scientific Conference on Informatics*, Nov 2015, pp. 302–307.
- [11] N. Koyachi, H. Adachi, M. Izumi, and T. Hirose, “Control of walk and manipulation by a hexapod with integrated limb mechanism: Melmantis-1,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 4, 2002, pp. 3553–3558 vol.4.

- [12] (2014) Inspire 1 v2.0. DJI. [Online]. Available: <http://store.dji.com/product/inspire-1-v2>
- [13] Matrice 600 pro. DJI. [Online]. Available: <http://store.dji.com/product/matrice-600-pro>
- [14] (2012, May) The quadcopter : control the orientation. [Online]. Available: <https://theboredengineers.com/tag/quadcopter/>
- [15] S. Swanson. (2017) The quadcopter class. [Online]. Available: <https://sites.google.com/a/eng.ucsd.edu/quadcopterclass/home>
- [16] Replicator 2x experimental 3d printer. MakerBot. [Online]. Available: <https://store.makerbot.com/printers/replicator2x/>
- [17] Dynamixel ax-12a robot actuator. Trossen Robotics. [Online]. Available: <http://www.trossenrobotics.com/dynamixel-ax-12-robot-actuator.aspx>
- [18] (2016) E1200 standard tuned propulsion system. [Online]. Available: <http://store.dji.com/product/e1200-standard>

Appendix A

PCB Schematics

Three printed circuit boards (PCBs) were designed for the hexapod-quadcopter robot. These boards were designed using KiCad and fabricated using the online service MacroFab. Fig. A.1 shows the PCB layer color assignments for the PCBs in Figs. A.4 - A.2. Figs. A.7 - A.5 provide the schematics for the preceding PCBs.

Bottom Paste	Bottom Paste
Bottom Silkscreen	Bottom Silkscreen
Bottom Soldermask	Bottom Soldermask
Bottom Copper	Bottom Copper
Top Copper	Top Copper
Top Soldermask	Top Soldermask
Top Silkscreen	Top Silkscreen
Top Paste	Top Paste
Board Outline	Board Outline
NC Drill	NC Drill

(a)
(b)

Figure A.1: The front (a) and back (b) PCB layer color assignments.

Figure A.2: The front (a) and transparent back (b) view of the hardware interface hub PCB layers.

Figure A.3: The front (a) and transparent back (b) view of the IR sensor ring PCB layers.

Figure A.4: The front (a) and transparent back (b) view of the power distribution PCB layers.

Figure A.5: Schematic for the hardware interface hub PCB, created in KiCad.

Figure A.7: Schematic for the power distribution PCB, created in KiCad.

Appendix B

3D Printing

This chapter contains supplementary pictures of the 3D printing process for the central body frame of the hexapod-quadcopter. Fig. B.1 shows the part modeled in Autodesk Inventor CAD software. The part was exported as an *.stl* file to be imported into the MakerBot Desktop software to prepare for printing. Fig. B.2a shows the part oriented to fit the build platform in software and the sliced output with raft and support material in Fig. B.2b. The exported *.x3g* file was printed on the MakerBot Replicator 2X in Fig. B.3. Since the part was printed using PLA plastic, the printer was left open with the print bed unheated. Fig. B.4a captures the print in progress, with support and gaps between infill clearly visible. The finished print is shown in B.4b.

Figure B.1: The body frame was modeled in Autodesk Inventor Professional 2015.

(a)

(b)

Figure B.2: The body frame model was imported to the MakerBot Desktop software (a) and configured to generate a sliced output to 3D print (b).

Figure B.3: The Makerbot Replicator 2X in lab used to 3D print.

(a)

(b)

Figure B.4: The body frame being 3D printed, in progress (a) and completed (b).