TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

DYNAMIC POWER CONTROL OF AN INERTIALLY STABILIZED CAMERA GIMBAL

INERTSIAALSELT STABILISEERITUD KAAMERAPIIRAJA DÜNAAMILINE VÕIMSUSE JUHTIMINE

MASTER THESIS

Student: Michael Casey Roberts

Student Code: 184675MAHM

Supervisor: Professor Mart Tamre

Tallinn 2020

AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

"....." ..... 20…..

Author: .....
/signature /

Thesis is in accordance with terms and requirements

"....." ..... 20…..

Supervisor: ….....
/signature/

Accepted for defense

"....."..... 20…..

Chairman of theses defense commission: .....
/name and signature/

Non-exclusive License for Publication and Reproduction of Graduation Thesis[1]

I, *Michael Casey Roberts* (name of the author) (date of birth: 24/02/1984) hereby

1. grant Tallinn University of Technology (TalTech) a non-exclusive license for my thesis

Dynamic Power Control of an Inertially Stabilized Camera Gimbal,

(*title of the graduation thesis*)

supervised by

Professor Mart Tamre,

(*Supervisor's name*)

1.1 reproduced for the purposes of preservation and electronic publication, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright;

1.2 published via the web of TalTech, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright.

1.3 I am aware that the author also retains the rights specified in clause 1 of this license.

2. I confirm that granting the non-exclusive license does not infringe third persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

---

[1] *Non-exclusive License for Publication and Reproduction of Graduation Thesis is not valid during the validity period of restriction on access, except the university`s right to reproduce the thesis only for preservation purposes.*

_____ (*signature*)

_____ (*date*)

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: Michael Casey Roberts 184675MAHM

Study program, main specialty: MAHM02/18 - Mechatronics

Supervisor(s): Professor, Mart Tamre, 620 3202

Consultants:

Thesis topic:

Design and prototype a power saving gimbal control algorithm

Projekteerige ja prototüüp energiasäästlik hübriidjuhtimisalgoritm

Thesis main objectives:

1. Produce a functional gimbal prototype for testing
2. Model motor temperature and dynamic power control
3. Test prototype with dynamic power control and record results

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Design and build prototype	01/03/20
2.	Develop models for motor temperature and power control	01/03/20
3.	Perform tests and record results	01/05/20

Language: English

Deadline for submission of thesis: 25/05/2020

Student: Michael Casey Roberts _____

/signature/date/

Supervisor: Professor Mart Tamre _____

/signature/date/

Head of study program: Professor Mart Tamre _____

/signature/date/

CONTENTS

PREFACE

This thesis explores the possible gains in efficiency provided by a control algorithm which dynamically adjusts the power being delivered to gimbal motors using position error as an input. All work was performed for the department of mechatronics at the Tallinn University of Technology in Tallinn, Estonia under the guidance of Professor Mart Tamre.

I would like to express my gratitude to Professor Tamre for allowing me the opportunity to attend university in Estonia, as well as Dhanushka Liyanage for his invaluable suggestions over the last two years.

I would also like to thank my family for their support and understanding during this process, as well as George Timbianakis for helping motivate me when things didn't go as planned along the way.

Keywords: Dynamic Power Control, Stabilized Gimbal, Master Thesis

Selles lõputöös uuritakse juhtimisalgoritmi abil saadavat efektiivsuse võimalikku suurenemist, mis dünaamiliselt reguleerib jõuaalmootoritele tarnitavat võimsust, kasutades sisendina positsiooniviga. Kõik tööd tehti Tallinnas Tallinna Tehnikaülikooli mehhatroonika osakonnas professor Mart Tamre juhendamisel.

Tahaksin tänada professor Tamret selle eest, et ta andis mulle võimaluse Eestis ülikoolis käia, samuti Dhanushka Liyanage'ile viimase kahe aasta hindamatute ettepanekute eest.

Samuti soovin tänada oma perekonda toetuse ja mõistmise eest selle protsessi ajal ning George Timbianakist, kes aitasid mind motiveerida, kui asjad ei läinud plaanipäraselt.

Märksõnad: Dünaamiline Võimsuse Juhtimine, Stabiliseeritud Gimbal, Magistritöö

List of abbreviations and symbols

API Application Program Interface

DC Direct Current

DOF Degrees of Freedom

GSD Ground Sampling Distance

IMU Inertial Measurement Unit

INS Inertial Navigation System

LOS Line of Sight

PID Proportional-Integral-Derivative (Controller)

PWM Pulse Width Modulated

RMS Root Mean Square

RX Receive

TX Transmit

UAV Unmanned Aerial Vehicle

VTOL Vertical Takeoff and Landing

1 INTRODUCTION

Aerial imaging with Unmanned Aerial Vehicles (UAV) is possible with many camera configurations. Hyperspectral pushbroom cameras are a powerful tool for collecting data across a wide electromagnetic spectrum. As these cameras have become more lightweight and UAV payloads have increased, combining the two technologies has yielded a low-cost and accessible imaging systems. Such systems are often used for aerial mapping but can also assist in infrastructure inspection [1][2] and environmental monitoring [3].

UAV imaging systems provide an alternative to larger, more expensive, imaging systems, but at the tradeoff of more disturbances which affect image quality. Therefore, the imaging system must be stabilized against disturbances and vibrations of the UAV [4][5]. A 2 or 3-axis gimbal utilizes an IMU to estimate inertial changes and a control scheme to compensate for these changes before they can affect the image quality. However, a more stable system typically requires more power, resulting in faster battery drain and reduced flight time.

UAVs are broadly classified as either fixed-wing or rotating-wing (helicopter and multirotor) craft. Fixed-wing UAVs generally offer longer flight endurance, higher maximum speed, and more stable data collection than multirotor UAVs. Multirotor UAVs, however, are capable of Vertical Takeoff and Landing (VTOL) and can operate in more confined locations [3]. However, hybrid UAVs having the capability of both VTOL and fixed wing flight have recently been developed.

Pushbroom cameras typically offer a good combination of spatial and spectral resolution. They are typically more stable than whiskbroom sensors and offer a greater spectral resolution. Pushbroom cameras are also typically smaller than framing and windowing devices and are therefore a better choice for lightweight UAV applications. In pushbroom cameras, a sensor row tangent to the flight path of the drone records a line of spectral information per exposure [3]. The sensor must be "swept" across the target, and software must be used to "stitch" the sequence of spectral information together to produce a complete image. Image distortion will occur due to disturbances to the flight path of the UAV, therefore the camera must be stabilized to minimize these effects. An IMU driven gimbal is the preferred solution.

Disturbances to the camera system can be due to wind, motor torque, and vibration. This system functions independently from the UAV host and will therefore require no modification to the UAV other than the mounting system. While the gimbal is designed around the camera system, it is composed of commonly available hardware and software wherever possible. Common components include Direct Current (DC) brushless motors, motion controller, Inertial Measurement Units (IMU), and microcontroller.

This paper presents a novel solution to prototype a gimbal which will affix a pushbroom style hyperspectral camera to a hybrid fixed-wing UAV capable of VTOL, as well as design an algorithm capable of dynamically adjusting power to each gimbal motor in real time in order to maximize flight time.

This paper is organized as follows. Chapter 2 begins with a review of related literature. Chapter 3 presents the problem and the approach taken to solve it. Chapter 4 explains the methodology used when performing experiments. Chapter 5 presents the experimental results, and Chapter 6 presents the conclusions made.

2 LITERATURE OVERVIEW / ANALYSIS

2.1 Existing Gimbal Design – 2DOF

UAVs can be used for infrastructure inspection in locations inaccessible to humans. One prototyped solution for bridge and tunnel inspection is described here [1]. In this example, a camera is stabilized by a 2-DOF mechanism which corrects for roll and pitch. Yaw, in this instance, is locked to the body of the UAV and not stabilized to reduce weight (thereby reducing moment of inertia.) An IMU is fixed to the camera mounting plate and provides information about the camera attitude in real time.

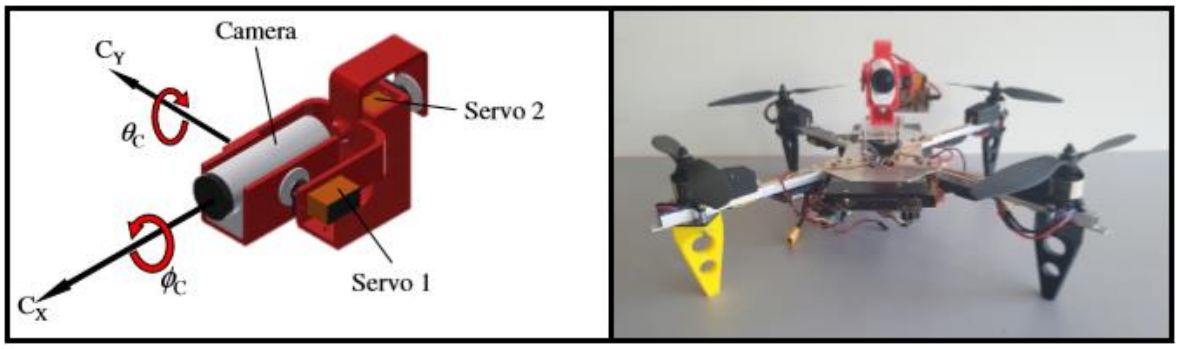

Figure 2.1 2-DOF Gimbal Example [1]

This prototype demonstrates that a camera can effectively be used to inspect infrastructure when mounted to a UAV and properly stabilized. However, this gimbal design is only useful for framed cameras producing video or still images. While this setup could, in theory, support a pushbroom camera and scan vertically, any slight rotation errors caused by unbalanced motor torque will produce orthographic errors in the image. Adding yaw control to produce a 3-DOF system would remedy this, but at the disadvantage of adding weight and increasing mechanical complexity and power consumption.

2.2 Existing Gimbal Design – 3DOF

Another prototype designed for infrastructure inspection is described here [2]. In this case, the camera is stabilized by a 3-DOF mechanism which corrects for roll, pitch, and yaw. This prototype utilizes the same roll/pitch mechanism described in section 2.1, and the IMU is affixed to the camera mounting plate in the same location.

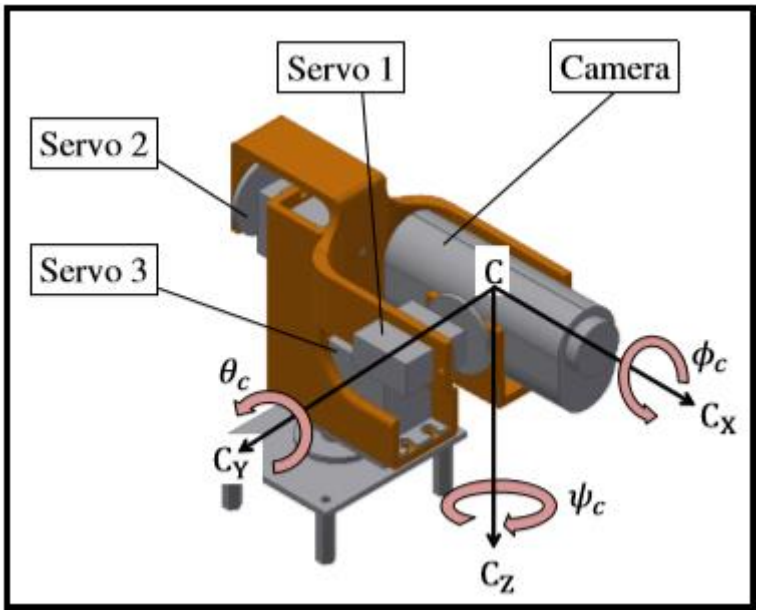

Figure 2.2 3-DOF Gimbal Example [70]

The authors conclude that stable operation is possible with this prototype, however there was an issue with controller performance. Because the moments of inertia are different between the X and Y axes of the UAV, and since the PID controller does not have separate parameters for each axis, the control is tuned for one axis and under-damped in the other, resulting in an oscillated response. While this performance is acceptable for a framed camera, again, it will not produce acceptable results for a pushbroom style camera.

2.3 Gimbal Control Algorithms

An inertially stabilized gimbal generally consists of DC brushless motors, an IMU, and a motion controller. Many different control schemes have been proposed and modeled, and their parameters have been determined using many different methods. This section will briefly cover the methods that have been proposed by other researchers.

It has been shown that a 2-DOF gimbal with PID controller can be tuned using Particle Swarm Optimization [4]. In another study, a modeled PID controller yielded better results when parameters were calculated with Particle Swarm Optimization as compared to Genetic Algorithm [5]. A modeled Fractional Order PI Control with parameters calculated using a genetic algorithm was shown to be an improvement over a standard PI controller optimized using the same method [6].

A hybrid fuzzy-PID controller was found to reduce overshoot by adjusting gains [7], because fuzzy performs well with nonlinear disturbances [8]. It has been demonstrated that the fuzzy-PID controller is, in theory, simpler to implement than other commonly used controllers with similar robustness but has higher computational load [9]. This approach is effective with 2-axis gimbal targeting systems [10], and it has also been demonstrated that a fuzzy-PID and Neural Network based controller can be effective with a 3-axis gimbal [11].

A simulated model using Pole-Placement with State Feedback control performed better than a lead-lag compensator controller and PID controller [12].

Modeled 2-axis gimbal using Linear Quadratic Gaussian control with Loop Transfer Recovery was found to be more stable than a PI-Lead controller [13][14].

A 2-DOF model using an adaptive control with modified error approach is compared to an auxiliary error structure. While the modified approach is easier to implement and has less calculation, the auxiliary error structure is shown to have superior performance and time response [15].

A comparison of Lead-Lag, PID, and Fuzzy controllers was made and showed that all showed good performance under the presence of nonlinearities. Each control method has its own strengths and advantages, and in the end the engineer must choose based upon the problem. It is noted that deep learning and robust control methods provide better solutions to highly non-linear control problems, and model predictive control provides a solution for black box and grey box models [16].

A model demonstrates that a sliding mode control is as effective as a PI controller, and in some situations better [17]. Another model comparing a fixed-order H∞ controller to a full-order H∞ controller shows that the full-order controller has better disturbance rejection and reference tracking performance than a PI controller [18].

If gimbal targeting is to be performed, a system designed to point the camera at a fixed GPS coordinate based on the relative position of the UAV was successfully prototyped and tested in [19] and [20].

Finally, the Kalman filter [21] and complimentary Kalman filter [22] have been shown to be effective in reducing sensor noise and increase system stability.

2.4 Camera Function and Orthorectification

Pushbroom style cameras are favored for aerial scanning applications due to their high resolution. They also have higher frame rates than whiskbroom style cameras, but both are susceptible to difficulties in post-processing due to their sensitivity to disturbances [3]. It is important to note that pushbroom cameras are directional and must be “swept” across the target to produce a series of orthophotos.

The process of combining the sensor information into complete images is quite complex. Generally, the scanner is assumed to be in a nadir-facing orientation to the target. The pushbroom scanner "sweeps" in the same direction as the flight path of the UAV. Because of this, a 2-DOF gimbal is often used for vertical scanning such as this. A nadir-facing scan could also occur across a vertical surface with a horizontal scan, and this application would instead require a 3-DOF camera if mounted beneath the UAV.

When making images on a moving platform such as a UAV, motion blur can make images unusable. The following calculations should be considered when designing the UAV imaging system [23].

First, the GSD should be calculated as shown in equation (2.1). GSD represents the smallest dimension that the camera needs to resolve and should be at least twice as small as the smallest feature to be measured. The UAV should have a set working distance from the target area. The following formula is used to determine the ratio between the lens' focal length and camera's pixel size.

$$GSD = \frac{Dp_x}{f} \tag{2.1}$$

where *GSD* – Ground Sampling Distance, m
D – distance to target, m
p_x – pixel dimension, m
f – focal length of lens, m

Blur occurs when the same target information is registered by multiple pixels during the exposure time due to the movement of the camera with respect to the target. The blur value should not exceed 0.5 pixels, and can be calculated using equation (2.2).

$$b = \frac{v*t}{GSD} \tag{2.2}$$

where b – blur, # of pixels
v – aircraft velocity, m/s
t – camera exposure time, s

The maximum velocity (or relative ground speed) without image blur can be calculated once the GSD and camera exposure time are known.

$$Vmax = \frac{GSD}{2t} \tag{2.3}$$

2.5 Motor Selection and Control

It is beneficial for the motors selected for a UAV platform to be of reduced size, weight, power, and cost [24]. Motors are often servo, brushed, or brushless type. As a general rule, efficiency is improved as motor size is decreased. However, motor specifications for many commercially available motors are not usually published, and these parameters must be measured directly to predict the performance of a DC motor.

2.6 Conclusion of Research

Typically, 2-axis gimbals are used for vertical linescanning photography at high elevation. 3-axis gimbals are generally used with framed imaging cameras. Linescan cameras are sensitive to disturbances, often resulting in blurred images. Linescan blur is dependent upon camera pixel size, focal length, distance to target, exposure time, and UAV velocity. The system response should be adequate to prevent image blur.

While there are many control regimes that can be used, the simple PID is most common and often used in motion controllers. However, one power setting is used and the system is tuned based on the system response. While this results in steady holding torque, it also consumes power whether the holding torque is needed or not.

Based upon this research, the thesis topic is to develop a general purpose 3-axis gimbal capable of stabilizing a hyperspectral linescan camera, with an algorithm to reduce power consumption when possible. This system consists the gimbal hardware, motor control software, and algorithm to dynamically adjust motor power. Prototyping will produce a functional unit, but it focuses on the electromechanical components and the power control algorithm. To simplify the system for the inital prototype, advanced filters, such as the Kalman, will not be used within this work.

3 PROBLEM STATEMENT AND APPROACH

3.1 Problem Statement

The purpose of this thesis was to develop a novel system for conserving gimbal power and maximizing UAV flight time. Specifically, a control algorithm was developed to adjust power delivery to each gimbal motor in real time based on positional error, and it was tested on a functional prototype. The major problems solved consisted of designing and building a gimbal prototype for testing, protecting the gimbal motors from overheating, communicating with the motion controller via serial connection, and developing a power-saving algorithm to update the power level and PID parameters of each motor in real time. This work specifically aimed to provide a working solution to mount a Ximea hyperspectral linescan camera onto a Wingcopter drone, thereby providing a platform for future thesis work within the department of mechatronics.

At the time of writing, there was no solution for mounting a Ximea camera to the Wingcopter drone. Therefore, a functional protype was designed and built to serve as a test platform. Commercially available brushless DC motors and motor controller were used, and all structural components were designed with SolidWorks modeling software and 3D printed with a Creality Ender 3D printer. Every reasonable effort was made to provide a lightweight and balanced assembly, but the optimization of the physical design of the gimbal is beyond the scope of this work.

Once a functional prototype was assembled, a method to protect the motors from overheating was needed. The motion controller manual states that temperatures of 80°C or greater will damage the magnets in the DC motors [25], therefore a thermal model of system was developed to estimate each motor's temperature in real time. The motor parameters were determined experimentally.

Once the temperature of the motors was modeled, a method of controlling the power to each motor was devised. The motion controller documentation includes a serial API which provides the means to request data from and update parameters in the motion controller. A commonly available microcontroller was utilized to communicate with the motion controller via serial connection.

Once a communication link was established between the microcontroller and the motion controller, an algorithm was created to modulate power to each motor in real time based on data from the motion controller and the temperature of each motor. Since motor response is directly related to power, a control scheme to adjust the PID parameters in real time was developed as well. While the values used during testing are functional, optimizing the PID parameters for this system is beyond the scope of this work.

Assumptions

For the purpose of this thesis, it is assumed that the gimbal will be supplied power by a dedicated battery source. It is a good idea in general to isolate the gimbal's power supply from that of the drone to mitigate the possibility of a gimbal failure causing a catastrophic electrical failure and/or crash. Further, while it is reasonable for both the imaging system and gimbal to share the same power source, such a system is not considered in this work. While at the test bench, the prototype was supplied with 13 V. During testing, motor temperatures were measured in still air. It is assumed that even while the drone is in flight, the gimbal is enclosed in a protective shell which also serves to isolate the electronics within from any high velocity air currents. Therefore, assuming a static environment is reasonable.

Tuning PID controllers is a complex process and optimizing the parameters for this gimbal prototype could be a project of its own. Therefore, it is assumed that the parameters produced by the auto-tuning macro provided with the motion controller are adequate for the purpose of this thesis.

3.2 Gimbal Design and Function

While the design of the gimbal prototype is beyond the scope of this thesis, the selection of components and their wiring are discussed here as they have a direct effect on power consumption. All components are numbered in order as they are described, and each is labeled using these numbers in figures (3.6) and (3.7).

Component 1: One BGM5208-200-12 brushless DC motor was selected to drive the yaw axis of the gimbal. This motor was located at the top of the gimbal and is shown in figure (3.1).

Figure 3.1 BGM5208-200-12 Motor

Component 2: Two BGM2606-90T brushless DC motors were selected to drive the pitch and roll axes. These were located beside (pitch) and behind (roll) the camera within the gimbal and are shown in figure (3.2).

Figure 3.2 BGM2606-90T Motors

Component 3: One SimpleBGC 32bit motion controller was selected to drive the three brushless DC motors. This motion controller operates as a PID controller and allows remote communication via serial connection send real-time system information and update parameters. The motion controller used is shown in figure (3.3).

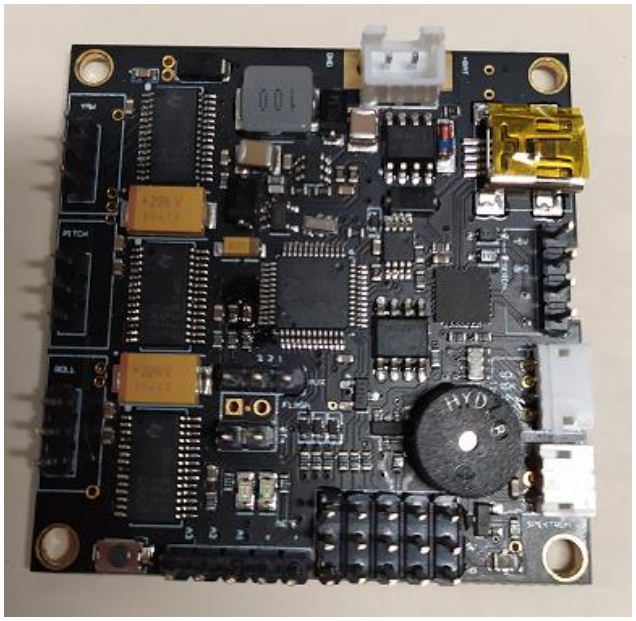

Figure 3.3 SimpleBGC 32bit Motion Controller

Component 4: Two Inertial Measurement Units were used in this gimbal. Both were included with the SimpleBGC motion controller and are integral to the function of the system. One IMU was mounted directly behind the camera, and the other (optional) unit was mounted to the frame to improve system accuracy. The IMUs each contain an accelerometer, gyroscope, and temperature sensor. This data is used by the motion controller to determine the camera's orientation. Both IMUs are shown in figure 3.4.

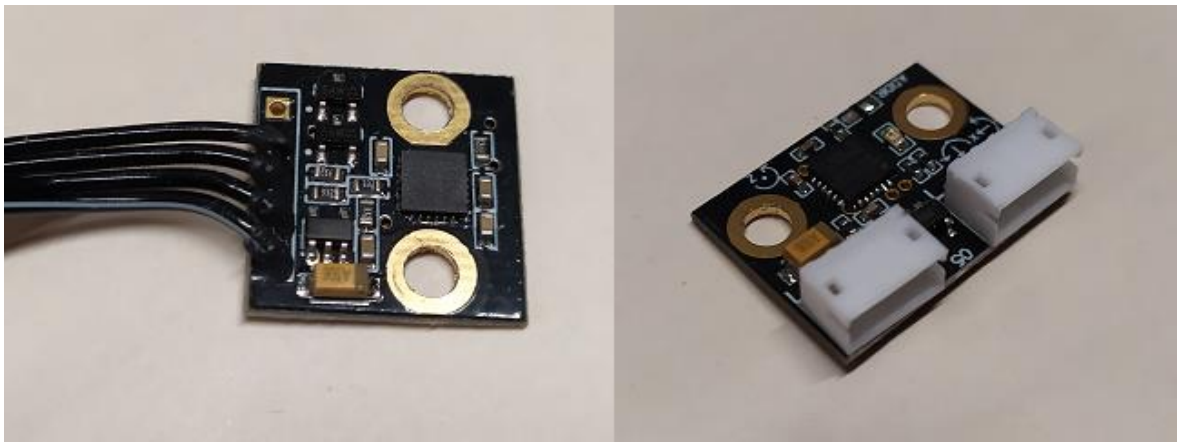

Figure 3.4 IMU Sensors

Component 5: One Arduino Nano microcontroller was used to run the power control algorithm. It was connected to the motion controller via serial connection, described in detail in the methodology section. Figure (3.5) shows the microcontroller that was used.

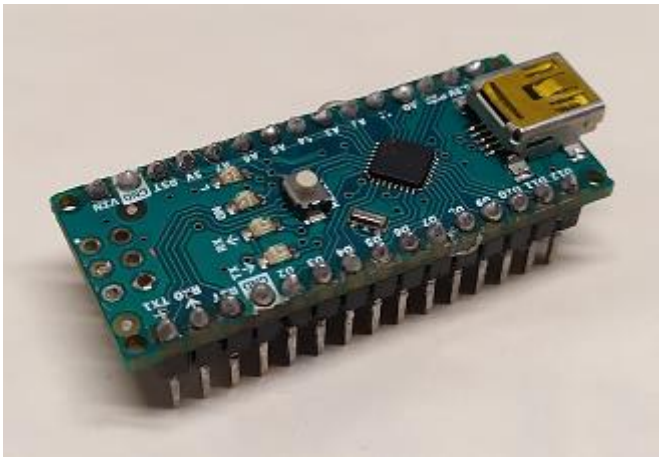

Figure 3.5 Arduino Nano Microcontroller

Figure (3.6) shows the complete gimbal prototype used for testing, excluding the aerodynamic housing as it was unnecessary for testing. All of the gimbal compoments were designed around the Ximea hyperspectral linescan camera, shown here. All structural components were designed in SolidWorks and printed in PLA on a Creality Ender 3 printer. The system is suspended from braided steel wire, which is intended to dampen high frequency vibrations. All wires were secured with cable ties to minimize their influence on the system while in operation.

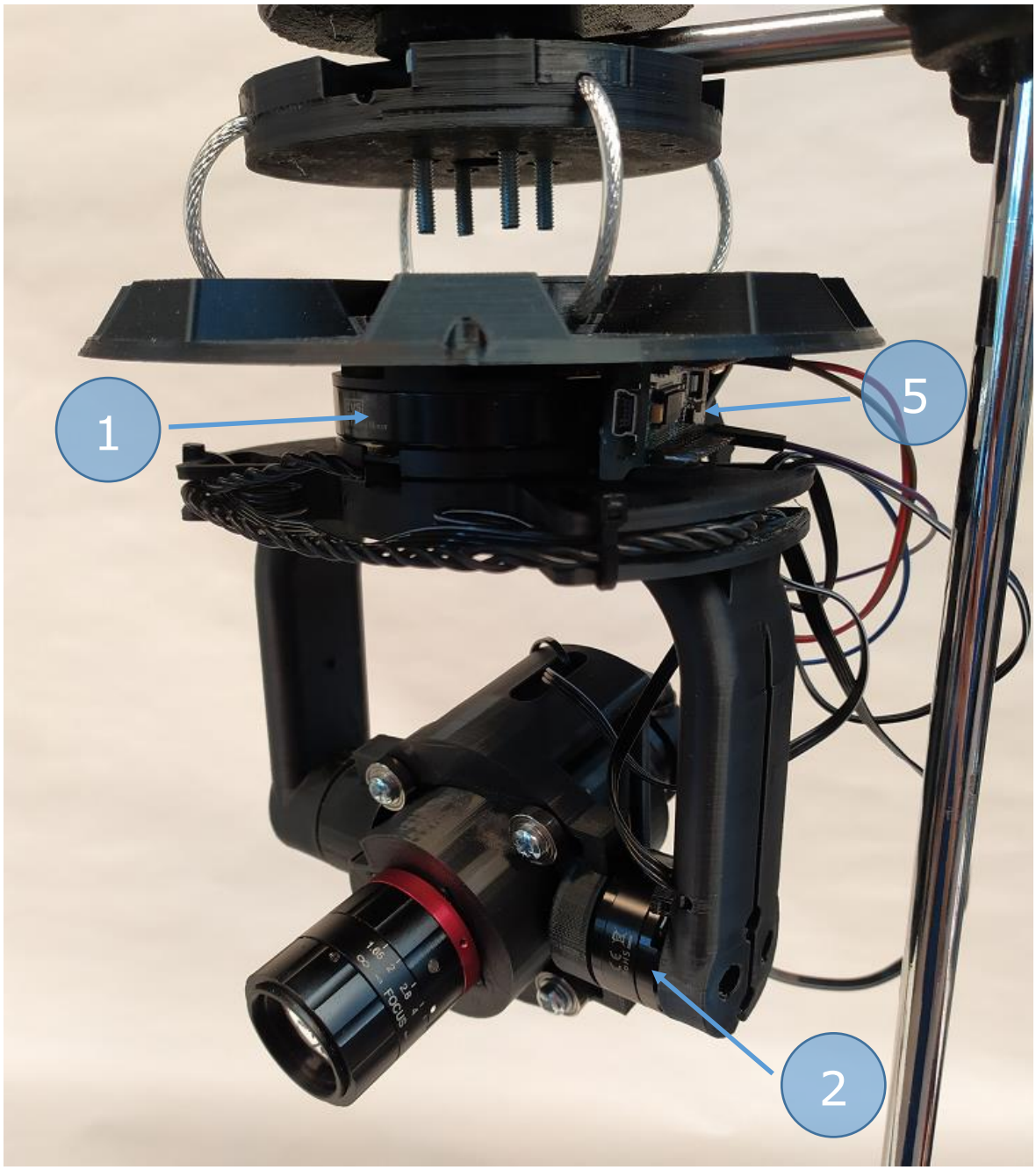

Figure 3.6 Gimbal Components, Front View

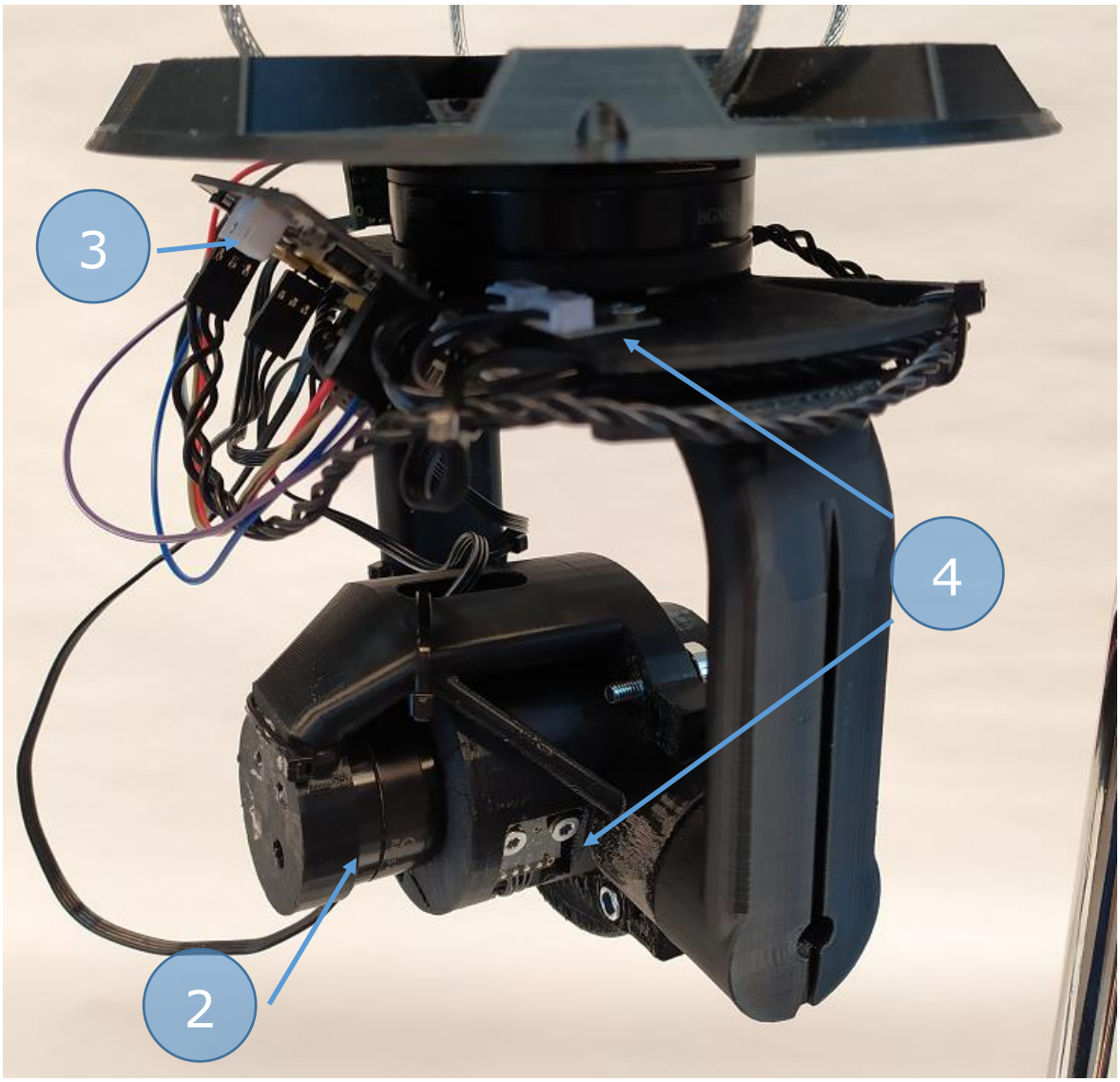

Figure 3.7 Gimbal Components, Side View

3.3 Mathematical Model

Modeling the Temperature Change of a Motor

This section explains the mathematical model used to estimate the temperature of a motor in real time as its power is increased or decreased with demand. Based on the first law of thermodynamics, all energy entering or leaving the system must be conserved as either heat or work, represented by equation (3.1) [26][27][28].

$$\Delta U = Q - W \tag{3.1}$$

where *U* - internal energy, J
Q - net heat transferred to the system, J
W - net work done by the system, J

Energy enters the motor in the form of electrical energy from the power supply and must be converted into one of two forms, either heat or mechanical work. Mechanical work is defined in equation (3.2) [26][27][28].

$$W = F * s \tag{3.2}$$

where *F* – force, N
s – displacement, m

In a gimbal system, the motors are generally making very small movements or holding the system in a fixed position. Therefore, the displacement is often zero or very near zero. When displacement is zero, mechanical work can be assumed to be zero. Equation (3.1) can then be simplified to show that the change in internal energy of a motor must be equal to the heat absorbed by the motor less the heat lost by the motor, as shown in equation (3.3) [26][27][28].

$$\Delta U = Q_{in} - Q_{out} \tag{3.3}$$

where Q_{in} – heat moving into the system, J
Q_{out} – heat moving out of the system, J

The relationship between heat and temperature is given by equation (3.4) [26][27][28].

$$\frac{Q}{mC} = \Delta T \tag{3.4}$$

where *m* – mass, kg
C – heat capacity, J/kg-°C
ΔT – change in motor temperature, °C

Therefore, dividing all terms by mass and heat capacity yields the relationship that the change in temperature of the motor is the difference between the temperature changes caused by heat entering and leaving the system, shown in equation (3.5).

$$\Delta T_{motor} = \Delta T_{Q_{in}} - \Delta T_{Q_{out}} \tag{3.5}$$

where ΔT_{motor} – temperature of the motor at time t, °C
ΔT_{Qin} – temperature change due to heat entering system, °C
ΔT_{Qout} – temperature change due to heat leaving system, °C

The change of temperature over time is shown by equation (3.6).

$$\dot{\Delta T_{motor}} = \dot{\Delta T_{Q_{in}}} - \dot{\Delta T_{Q_{out}}} \tag{3.6}$$

Calculating Temperature Gain

The only source of energy entering the system is in the form of electrical power, so it can be assumed that the heat gained by the system is a function of power. A coefficient which combines the heat capacity and inefficiency of the motor can be created to describe the change in temperature due to this heat exchange, as shown in equation (3.7). This coefficient of heating must be determined experimentally.

$$\frac{dT_{motor}}{dt} = rP \tag{3.7}$$

where T_{motor} – temperature of the motor at time t, °C
t – time, s
r – motor coefficient of heating, °C/J
P – power supplied to motor, J/s

Calculating Temperature Loss

Heat is assumed to be lost in the form of conduction, as described by Newton's law of cooling, shown in equation (3.8)[26]. A coefficient of cooling is used to describe the rate at which this heat exchange occurs. This coefficient must be determined experimentally.

$$\frac{dT_{motor}}{dt} = -k(T_{motor} - T_{air}) \tag{3.8}$$

where T_{air} – temperature of the air, °C
k – motor coefficient of cooling, s^{-1}

Temperature Model

By substituting equations (3.7) and (3.8) into equation (3.6), the rate of temperature change of the motor can be modeled as shown in equation (3.9).

$$\frac{dT_{motor}}{dt} = rP - k(T_{motor} - T_{air}) \tag{3.9}$$

Calculating Motor Voltage

The stabilization controller sends a three-phase *Pulse Width Modulated* (PWM) signal of specific duty cycle to each motor based on a power parameter defined for each motor. The power parameter is an 8-bit unsigned integer value between 0 and 255, and the relationship between power parameter and duty cycle is a simple ratio, as shown in equation (3.10) [29].

$$D = \frac{P_{param}}{255} \tag{3.10}$$

where D = Duty Cycle, %
P_{param} = Power Parameter, unitless

The amplitude of this signal is equal to the voltage supplied to the stabilization controller, but the *Root Mean Square* (RMS) value of this signal is what determines the current consumed by the motor and, therefore, the torque produced. The RMS voltage value can be calculated with equation (3.11) [29].

$$V_{RMS} = V_{supply} * \sqrt{D} \tag{3.11}$$

where V_{RMS} = RMS Voltage, V
V_{supply} = Supply Voltage, V

Calculating Motor Current

Motor current can be calculated using Ohm's law, as shown in equation (3.12).

$$I = \frac{V_{RMS}}{Z} \tag{3.12}$$

where I – current, A
Z – impedance, Ω

Impedance is defined in equation (3.13).

$$Z = R + jX \tag{3.13}$$

where R – resistance, Ω
jX – reactance, Ω

Impedance is required to describe the system due to the nature of a switched DC signal producing a waveform. Impedance is the combination of resistance, a function of the wire material and length, and reactance, a function of the inductance of the circuit. However, it was determined experimentally that impedance is not constant and changes with duty cycle and motor temperature. This relationship is not easily modeled mathematically and was instead modeled experimentally, as described in section (4.2).

4 METHODOLOGY

4.1 Gimbal Design

While the gimbal design is largely out of scope, it directly affects the performance of the system and will therefore be discussed here briefly. The objective while designing each printed component was to maintain a high moment of inertia while minimizing material (and therefore weight) simultaneously ensuring the final design was suitable for rapid manufacture on a 3D printer. To minimize weight, a wall thickness of 0.8 mm an infill of 15 % were used while printing all parts. Bearings were used to reduce drag on the motors and prevent parts from being cantilevered and solely supported by the brushless DC motors. Inlets designed for storing excess wire were integrated into the frame and wires were secured to prevent them from impacting the performance of the system.

4.2 Mathematical Model

The primary objective of this system is to dynamically adjust the power sent to each motor in real time. To accomplish this, a mathematical model was created to predict the behavior of the system. The model provides a way to estimate motor temperature and predict a safe maximum power that can be delivered without overheating each motor. Once this is determined, the power demand is calculated based upon the magnitude of positional error for each motor. Because many variables are functions of themselves (for instance, current is a function of motor temperature, but also determines the rate at which the motor heats), an iterative process is used to approximate the state of each variable at a discreet point in time, then use those variables to predict their initial states at the next discreet point in time. The motor variables calculated are current, rate of cooling due to conduction, rate of heating due to power, position error, maximum allowable power, and actual power to deliver.

Calculating BGM9606-90T Current

To calculate the current consumed by a BGM2606-90T motor, an experiment was performed to measure how much current a motor is suppled at various power settings and motor temperatures. The initial motor temperature was recorded, then the motor was supplied a constant RMS voltage from the motion controller by assigning a power parameter. The temperature of the motor and supplied current were measured at regular intervals until the motor was near either thermal equilibrium or 70 °C. The motor was then allowed to cool without disturbance. The test was repeated at a different power

parameter. Four different power parameters were tested in total. A Teklab TLP603 DC power supply was used to provide power to the motion controller. A Flir E50 thermal imaging camera was used to measure motor temperature. The system was initialized with all motors at zero power to measure the current supplied to the motion controller; this measured current was subtracted from all motor measurements. Only one motor was energized at a time. Table (4.1) summarizes the results from this test.

Table 4.1 BGM2606-90T Current as a Function of RMS Voltage and Motor Temperature

			Experimental Values		Calculated Values			
		t min	T_{motor} °C	I mA	$\Delta T_{motor}/\Delta t$ °C/min	*Cooling* °C/min	r °C/J	r^{-1} J/°C
Experiment 1	*Power Parameter*: 50 V_{RMS}: 5.76 V T_{air}: 25 °C	0	25.9	56	n/a	n/a	n/a	n/a
		1	27.5	55	1.6	0.275	5.91856	0.16896
		2	28.8	55	1.3	0.418	5.42298	0.1844
		3	29.8	55	1	0.528	4.82323	0.20733
		4	30.8	54	1	0.638	5.2662	0.18989
		5	31.7	54	0.9	0.737	5.26299	0.19001
Experiment 2	*Power Parameter*: 100 V_{RMS}: 8.14 V T_{air}: 26 °C	0	31.7	204	n/a	n/a	n/a	n/a
		1	36.6	197	4.9	1.166	3.78279	0.26436
		2	40.8	194	4.2	1.628	3.69057	0.27096
		3	44.1	192	3.3	1.991	3.38542	0.29538
		4	47.1	190	3.0	2.321	3.44045	0.29066
		5	49.3	189	2.2	2.563	3.09595	0.323
Experiment 3	*Power Parameter*: 150 V_{RMS}: 9.97 V T_{air}: 26 °C	0	35.4	410	n/a	n/a	n/a	n/a
		1	46.3	399	10.9	2.233	3.30138	0.3029
		2	54.8	389	8.5	3.168	3.00851	0.33239
		3	61.3	383	6.5	3.883	2.71912	0.36777
		4	66.6	378	5.3	4.466	2.59137	0.3859
		5	71.3	373	4.7	4.983	2.60379	0.38406
Experiment 4	*Power Parameter*: 200 V_{RMS}: 11.5 V T_{air}: 26 °C	0	31.6	678	n/a	n/a	n/a	n/a
		1	50.1	650	18.5	2.651	2.82957	0.35341
		2	63.9	629	13.8	4.169	2.48414	0.40255
		3	74.5	604	10.6	5.335	2.29413	0.4359
		4	-	-	-	-	-	-
		5	-	-	-	-	-	-
Experiment 5	*Power Parameter*: 250 V_{RMS}: 12.9 V T_{air}: 26 °C	0	31.9	975	n/a	n/a	n/a	n/a
		0.5	43.6	954	23.4	1.936	2.05873	0.48574
		1	56.4	925	25.6	3.344	2.42564	0.41226
		1.5	67.2	906	21.6	4.532	2.23591	0.44724
		-	-	-	-	-	-	-
		-	-	-	-	-	-	-

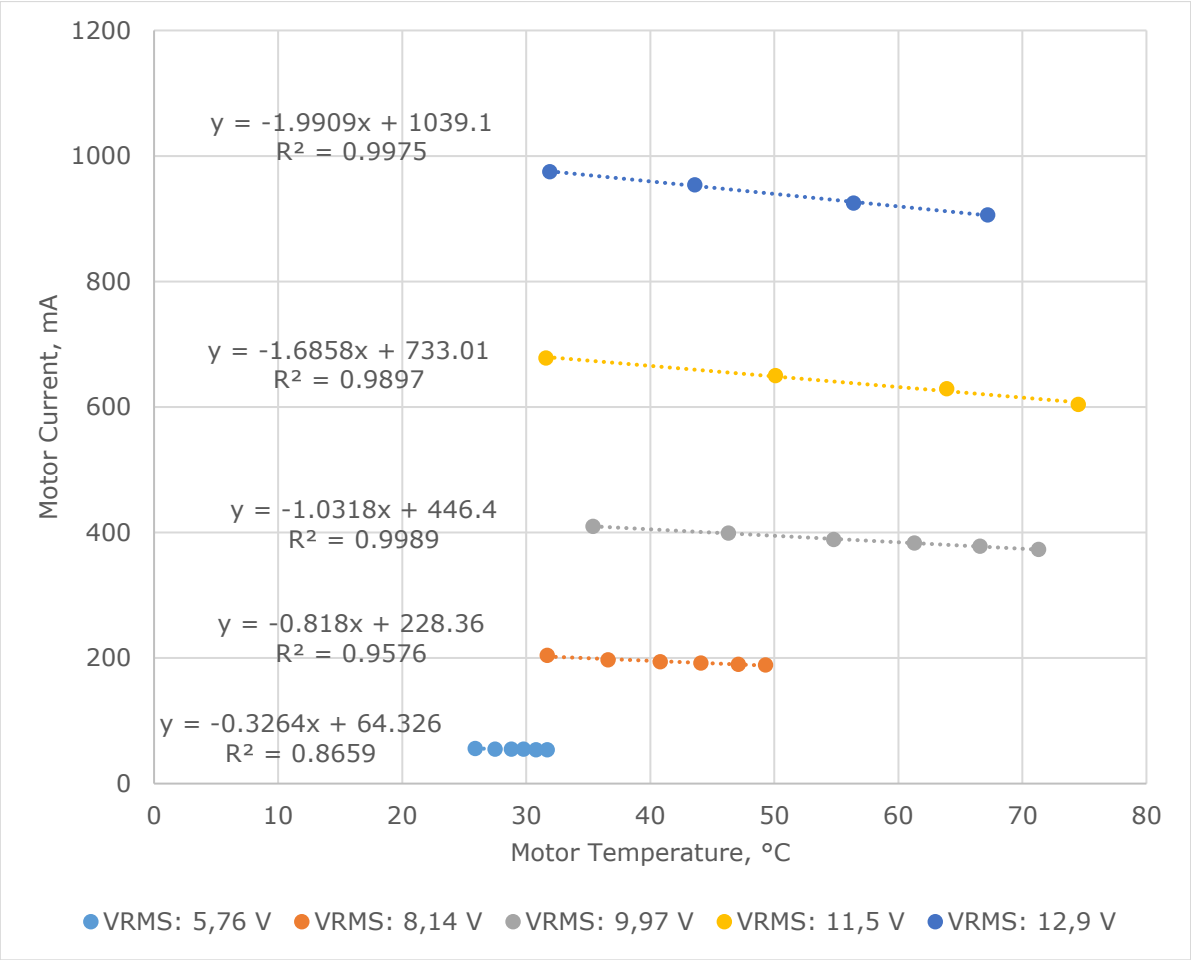

Figure 4.1 BGM2606-90T Current as a function of Temperature by RMS Voltage

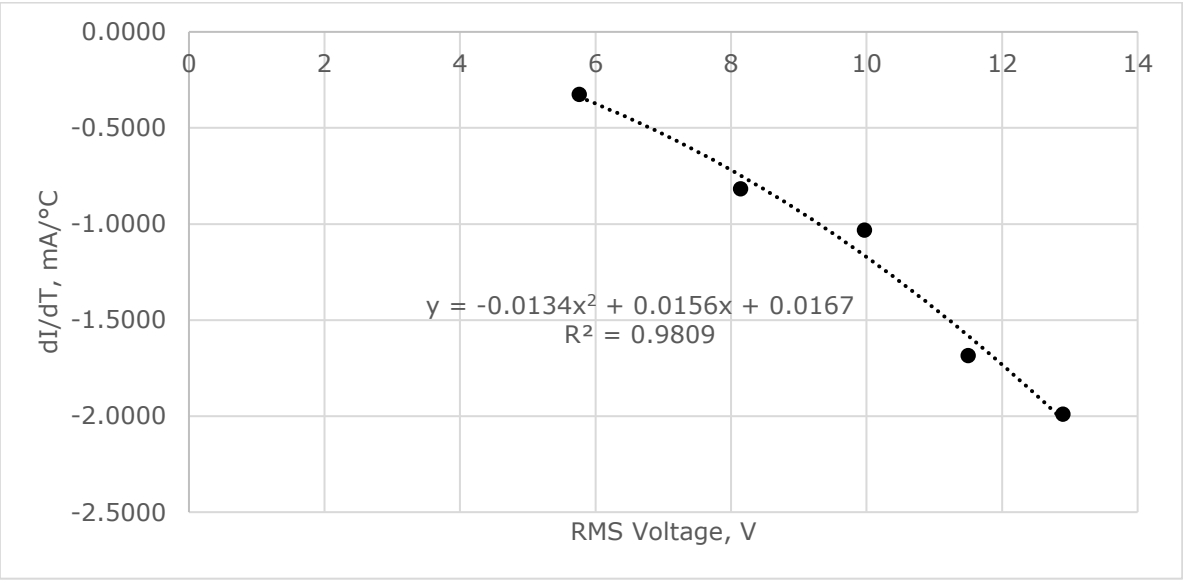

Figure 4.2 BGM2606-90T Slope Producing Equation for Current Calculation

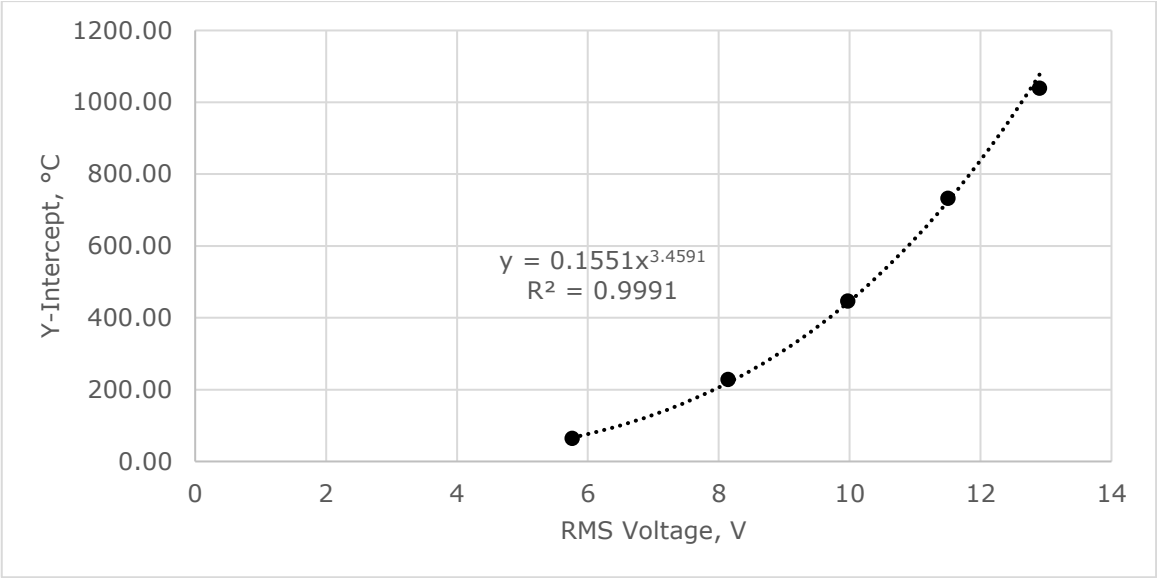

Figure 4.3 BGM2606-90T Y-Intercept Producing Equation for Current Calculation

Figure (4.1) shows the experimental data as a function of motor temperature. Figures (4.2) and (4.3) provide curves that are used to determine the slope and y-intercept of the linear current function. Given an RMS voltage, the function for current can now be produced in slope-intercept form as shown in equation (4.1).

$$I = mT_{motor} + b \tag{4.1}$$

where m = slope as a function of VRMS, mA·°C^{-1}
b = y-intercept as a function of VRMS, °C

The slope of the current equation can now be calculated with equation (4.2).

$$m = 0.0134{V_{RMS}}^2 + 0.0156V_{RMS} + 0.0167 \tag{4.2}$$

The y-intercept of the current equation can now be calculated with equation (4.3).

$$b = 0.1551{V_{RMS}}^{3.4591} \tag{4.3}$$

Calculating BGM5208-200-12 Current

The same test was then repeated for the BGM5208-200-12 motor, and the experimental results are summarized in table (4.2).

Table 4.2 BGM5208-200-12 Current as a Function of RMS Voltage and Motor Temperature

			Experimental Values		Calculated Values			
		t min	T_{motor} °C	I mA	$\Delta T_{motor}/\Delta t$ °C/min	*Cooling* °C/min	r °C/J	r^{-1} J/°C
Experiment 1	*Power Parameter*: 50 V_{RMS}: 5.76 V T_{air}: 25 °C	0	25.3	34	n/a	n/a	n/a	n/a
		1	25.5	33	0.2	0.025	1.18371	0.84480
		2	25.7	33	0.2	0.035	1.23632	0.80885
		3	25.9	33	0.2	0.045	1.28893	0.77584
		4	26.1	33	0.2	0.055	1.34154	0.74541
		5	26.3	33	0.2	0.065	1.39415	0.71728
Experiment 2	*Power Parameter*: 100 V_{RMS}: 8.14 V T_{air}: 26 °C	0	26.2	127	n/a	n/a	n/a	n/a
		1	26.6	125	0.4	0.03	0.4226	2.36628
		2	27.3	124	0.7	0.065	0.75791	1.31942
		3	27.8	124	0.5	0.09	0.58453	1.71078
		4	28.4	124	0.6	0.12	0.71332	1.40189
		5	28.8	124	0.4	0.14	0.53499	1.86919
Experiment 3	*Power Parameter*: 150 V_{RMS}: 9.97 V T_{air}: 26 °C	0	28.5	273	n/a	n/a	n/a	n/a
		1	29.2	268	0.7	0.16	0.32186	3.10693
		2	30.3	266	1.1	0.215	0.49585	2.01675
		3	31.8	265	1.5	0.29	0.6775	1.47601
		4	32.6	264	0.8	0.33	0.42932	2.32927
		5	33.8	263	1.2	0.39	0.60638	1.64913
Experiment 4	*Power Parameter*: 200 V_{RMS}: 11.5 V T_{air}: 26 °C	0	31.8	460	n/a	n/a	n/a	n/a
		1	33.4	448	1.6	0.37	0.38238	2.61523
		2	35.5	444	2.1	0.475	0.50431	1.98291
		3	37.1	440	1.6	0.555	0.42589	2.34803
		4	38.8	438	1.7	0.64	0.46456	2.15256
		5	40.1	435	1.3	0.705	0.4008	2.49501
Experiment 5	*Power Parameter*: 250 V_{RMS}: 12.9 V T_{air}: 26 °C	0	32.3	689	n/a	n/a	n/a	n/a
		0.5	34.5	663	2.2	0.425	0.30692	3.25817
		1	37.6	656	3.1	0.58	0.43486	2.29957
		1.5	40.2	649	2.6	0.71	0.39536	2.52934
		-	42.8	644	2.6	0.84	0.41408	2.41500
		-	45.1	639	2.3	0.955	0.39488	2.53244

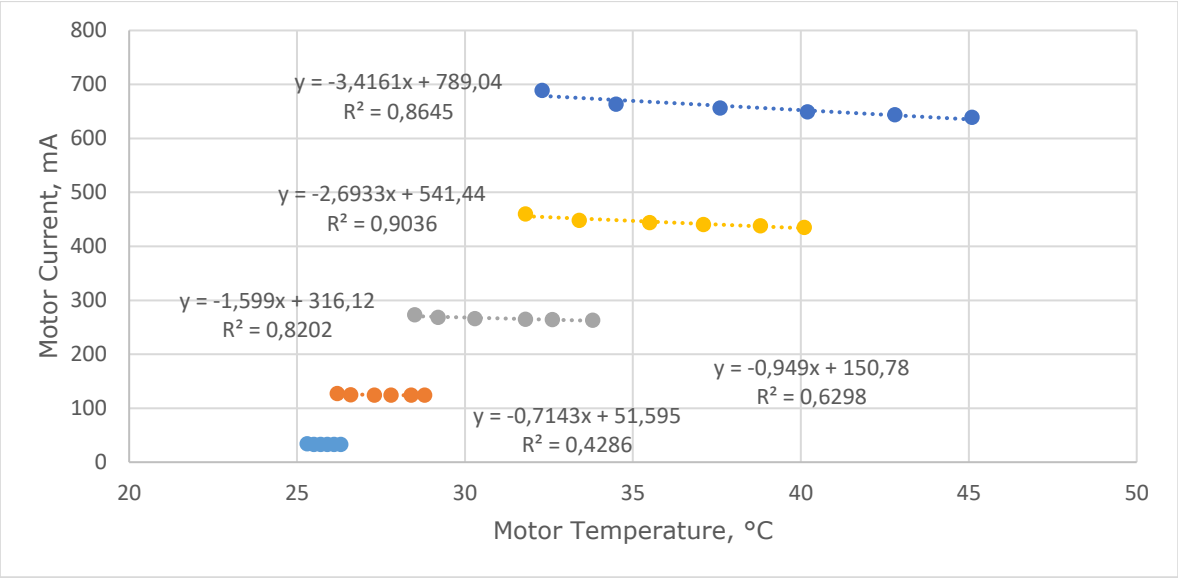

Figure 4.4 BGM5208-200-12 Current as a function of Temperature by RMS Voltage

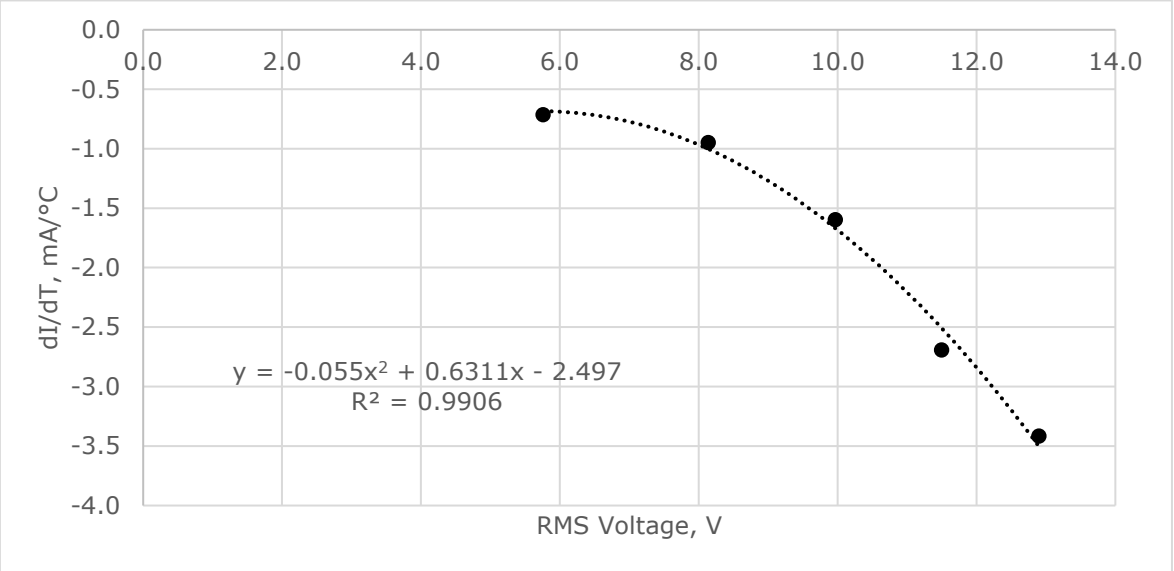

Figure 4.5 BGM5208-200-12 Slope Producing Equation for Current Calculation

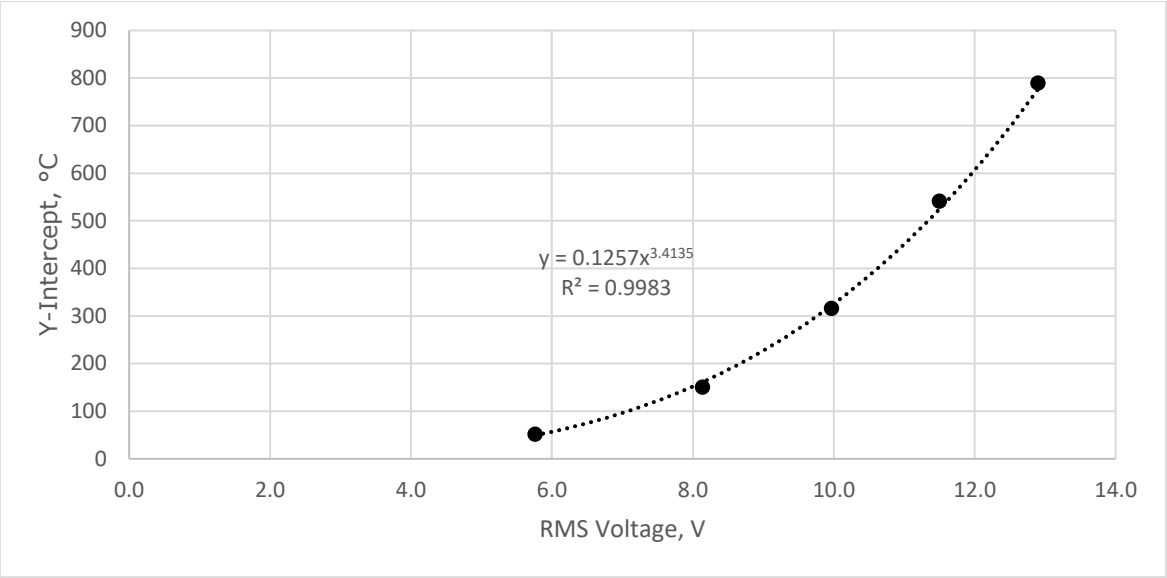

Figure 4.6 BGM5208-200-12 Y-Intercept Producing Equation for Current Calculation

Figure (4.4) shows the experimental data as a function of motor temperature. Figures (4.5) and (4.6) provide curves that are used to determine the slope and y-intercept of the linear current function. Given an RMS voltage, the function for current can now be produced in slope-intercept form as shown in equation (4.1).
The slope of the current equation can now be calculated with equation (4.4).

$$m = 0.055{V_{RMS}}^2 + 0.6311V_{RMS} - 2.497 \quad (4.4)$$

The y-intercept of the current equation can now be calculated with equation (4.5).

$$b = 0.1257{V_{RMS}}^{3.4135} \quad (4.5)$$

Determining Motor Coefficient of Cooling

In order to calculate the coefficient of cooling for a motor, the condition in which a heated motor is deenergized and allowed to cool is analyzed. Since the power entering the motor is zero in this situation, equation (3.8) can be used.
Integrating yields the relationship of motor temperature as a function of time, as shown below in equation (4.6).

$$\int dT_{motor} = \int -k(T_{motor} - T_{air})dt$$

$$\int \frac{1}{T_{motor} - T_{air}} dT_{motor} = \int -k\, dt$$

$$\ln(T_{motor} - T_{air}) = -k * t + C$$

$$T_{motor} - T_{air} = C * e^{-kt}$$

$$T_{motor} = C * e^{-kt} + T_{air} \quad (4.6)$$

where C – constant of integration

BGM2606-90T Coefficient of Cooling

An experiment was then performed to determine the constant of integration. The motor was heated to approximately 60°C by setting a power parameter of 200 in the motion controller. The power was then shut off and the motor temperature was measured at regular intervals as the motor cooled. The air temperature, as measured by the motion controller, was recorded at each interval as well. The results are shown in table (4.3) below.

Table 4.3 BGM2606-90T Motor Temperature while Cooling

t min	T_{motor} °C	T_{air} °C
0	61.8	25
1	56.3	25
2	51.9	25
3	49.5	25
4	47.3	25
5	45.4	25
6	43.7	25
7	42.1	25
8	40.8	25
9	39.7	25
10	38.8	25

Next, the initial condition is substituted into equation (4.6) to solve for *C*:

$$(61.8\,^{\circ}\mathrm{C}) = C * e^{-k(0\ min)} + (25\,^{\circ}\mathrm{C})$$

$$36.8\,^{\circ}\mathrm{C} = C * 1$$

$$C = 36.8\,^{\circ}\mathrm{C}$$

Now *C* can be substituted into equation (4.6) to produce equation (4.7):

$$T_{motor} = 36.8 * e^{-kt} + T_{air}$$

$$e^{-kt} = \frac{T_{motor} - T_{air}}{36.8}$$

$$\ln(e^{-kt}) = \ln\left(\frac{T_{motor} - T_{air}}{36.8}\right)$$

$$-kt = \ln\left(\frac{T_{motor} - T_{air}}{36.8}\right)$$

$$k = \frac{-1}{t} * \ln\left(\frac{T_{motor} - T_{air}}{36.8}\right) \qquad (4.7)$$

The value of *k* can now be calculated for each of the data points in table (4.3), as shown in table (4.4).

Table 4.4 BGM2606-90T Calculated Cooling Coefficients

t min	k s^{-1}
0	n/a
1	0.0936
2	0.0447
3	0.0290
4	0.0211
5	0.0164
6	0.0133
7	0.0110
8	0.0094
9	0.0081
10	0.0071

While, in theory, the value of *k* is a constant in Newton's law of cooling, the calculations in table (4.4) demonstrate that the model is imperfect in this application. However, it can still be useful to approximate the rate at which heat leaves the system. Substituting the value of C into equation (4.6) produces equation (4.8).

$$T_{motor} = 36.8 * e^{-kt} + T_{air} \qquad (4.8)$$

Table (4.5) shows the predicted temperature values for three different assumed values of k, calculated using equation (4.8). Figure (4.7) shows how the curves produced by these values compare to the experimental data.

Table 4.5 BGM2606-90T Predicted Cooling Values with Various Coefficients

t min	T_{air} ˚C	T_{motor} ˚C k=0.08	T_{motor} ˚C k=0.12	T_{motor} ˚C k=0.16
0	25	61.8	61.8	61.8
1	25	59.0	57.6	56.4
2	25	56.4	53.9	51.7
3	25	53.9	50.7	47.8
4	25	51.7	47.8	44.4
5	25	49.7	45.2	41.5
6	25	47.8	42.9	39.1
7	25	46.0	40.9	37.0
8	25	44.4	39.1	35.2
9	25	42.9	37.5	33.7
10	25	41.5	36.1	32.4

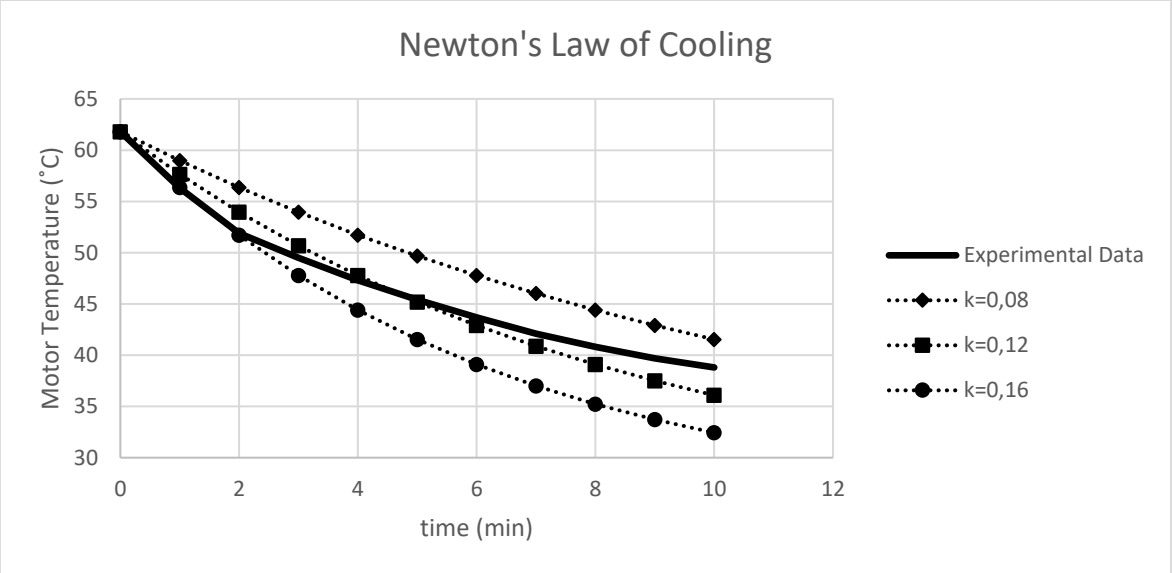

Figure 4.7 BGM2606-90T Temperature Change Estimates by Coefficient of Cooling

The value of k=0.11 was selected for use in this model and is assumed to be a constant in all calculations. Equation (3.8) can now be updated to include this coefficient, as shown in equation (4.9).

$$\frac{dT_{motor}}{dt} = -0.11(T_{motor} - T_{air}) \tag{4.9}$$

BGM5208-200-12 Coefficient of Cooling

The experiment was then performed a second time with the BGM5208-200-12 motor. It was heated to approximately 65˚C by setting a power parameter of 255 in the motion controller. The power was then shut off and the motor temperature was measured at regular intervals as the motor cooled. The air temperature, as measured by the motion controller, was recorded at each interval as well. The results are shown in table (4.6) below.

Table 4.6 BGM5208-200-12 Motor Temperature while Cooling

t min	T_{motor} ˚C	T_{air} ˚C
0	65.0	26
1	63.2	26
2	60.9	26
3	58.9	26
4	57.2	26
5	55.8	26
6	54.2	26
7	53.0	26
8	51.9	26
9	50.7	26
10	49.5	26

Next, the initial condition is substituted into equation (4.6) to solve for *C*:

$$(65.0\,℃) = C * e^{-k(0\,min)} + (26\,℃)$$

$$39.0\,℃ = C * 1$$

$$C = 39.0\,℃$$

Now *C* can be substituted into equation (4.6) to produce equation (4.10):

$$T_{motor} = 39.0 * e^{-kt} + T_{air}$$

$$e^{-kt} = \frac{T_{motor} - T_{air}}{39.0}$$

$$\ln(e^{-kt}) = \ln\left(\frac{T_{motor} - T_{air}}{39.0}\right)$$

$$-kt = \ln\left(\frac{T_{motor} - T_{air}}{39.0}\right)$$

$$k = \frac{-1}{t} * \ln\left(\frac{T_{motor} - T_{air}}{39.0}\right) \quad (4.10)$$

The value of *k* can now be calculated for each of the data points in table (4.6). The results are shown below in table (4.7).

Table 4.7 BGM5208-200-12 Calculated Cooling Coefficients

t min	*k* s^{-1}
0	n/a
1	0.0473
2	0.0555
3	0.0567
4	0.0558
5	0.0538
6	0.0540
7	0.0525
8	0.0512
9	0.0508
10	0.0507

The value of k is again demonstrated to not be constant over time. Table (4.8) shows the predicted temperature values for three different assumed values of k, calculated using equation (4.10). Figure (4.8) shows how the curves produced by these values compare to the experimental data.

Table 4.8 BGM5208-200-12 Predicted Cooling Values with Various Coefficients

t min	T_{air} ˚C	T_{motor} ˚C, k=0.06	T_{motor} ˚C, k=0.05	T_{motor} ˚C, k=0.04
0	26	65.0	65.0	65.0
1	26	62.7	63.1	63.5
2	26	60.6	61.3	62.0
3	26	58.6	59.6	60.6
4	26	56.7	57.9	59.2
5	26	54.9	56.4	57.9
6	26	53.2	54.9	56.7
7	26	51.6	53.5	55.5
8	26	50.1	52.1	54.3
9	26	48.7	50.9	53.2
10	26	47.4	49.7	52.1

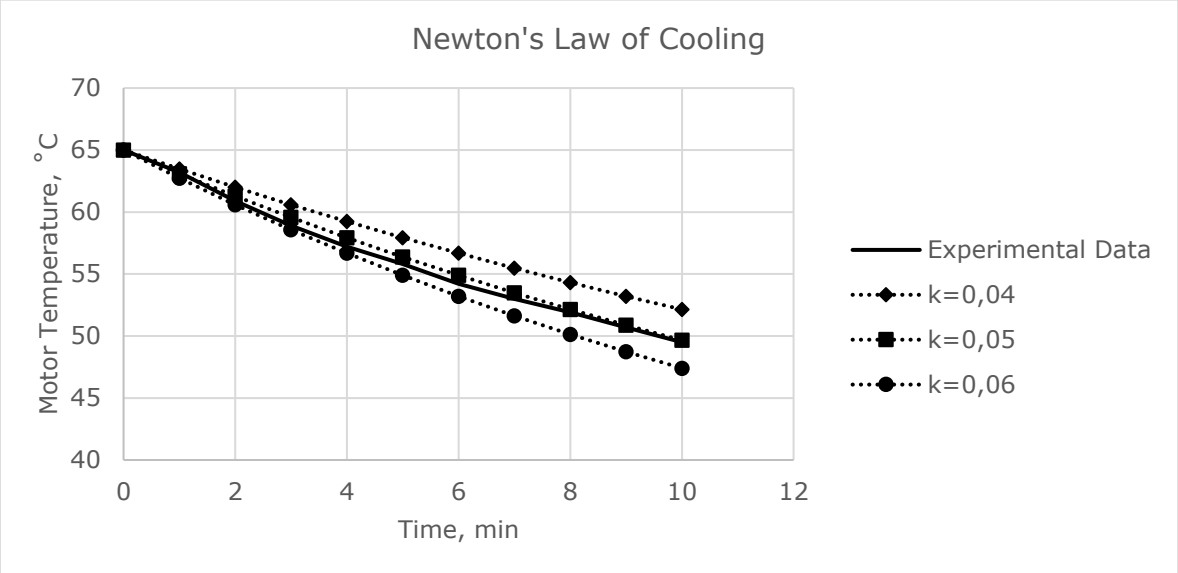

Figure 4.8 BGM5208-200-12 Temperature Change Estimates by Coefficient of Cooling

The value of k=0.05 was selected for use in this model and is assumed to be a constant in all calculations. Equation (3.8) can now be updated to include this coefficient, as shown in equation (4.11).

$$\frac{dT_{motor}}{dt} = -0.05(T_{motor} - T_{air}) \tag{4.11}$$

Determining Motor Coefficient of Heating

The data collected in the motor current experiment, were used to determine the motor coefficient of heating. The calculated values in the table were produced as described below and used for the calculations.

The motor temperature rate of change was calculated between each time interval with equation (4.12).

$$\frac{dT_{motor}}{dt} = \frac{T_2 - T_1}{t_2 - t_1} \tag{4.12}$$

The rate of cooling was calculated using equations (4.9) and (4.11) depending on the motor being tested. Power was calculated by using equation (4.13).

$$P = V_{RMS} I \tag{4.13}$$

The rate of heating for each data point was then calculated by substituting the rate of cooling and power into equation (3.9), producing equation (4.14).

$$\frac{dT_{motor}}{dt} = r * P - [rate\ of\ cooling]$$

$$\frac{dT_{motor}}{dt} + [rate\ of\ cooling] = r * P$$

$$r = \frac{\frac{dT_{motor}}{dt} + [rate\ of\ cooling]}{P} \tag{4.14}$$

BGM9606-90T Coefficient of Heating

The motor coefficients of heating were then graphed for the various power settings as a function of motor temperature to produce figure (4.9). The motor coefficient is relatively linear, but a function of both motor temperature and RMS voltage. Therefore, each data set was described with a linear best fit function, and the slopes and y-intercepts, shown in table (4.9), were graphed as functions of RMS voltage, figures (4.10) and (4.11), respectively. It should be noted that because the calculated coefficients of heating decrease as motor temperature increases, the inverse of *r* is plotted in figure (4.9).

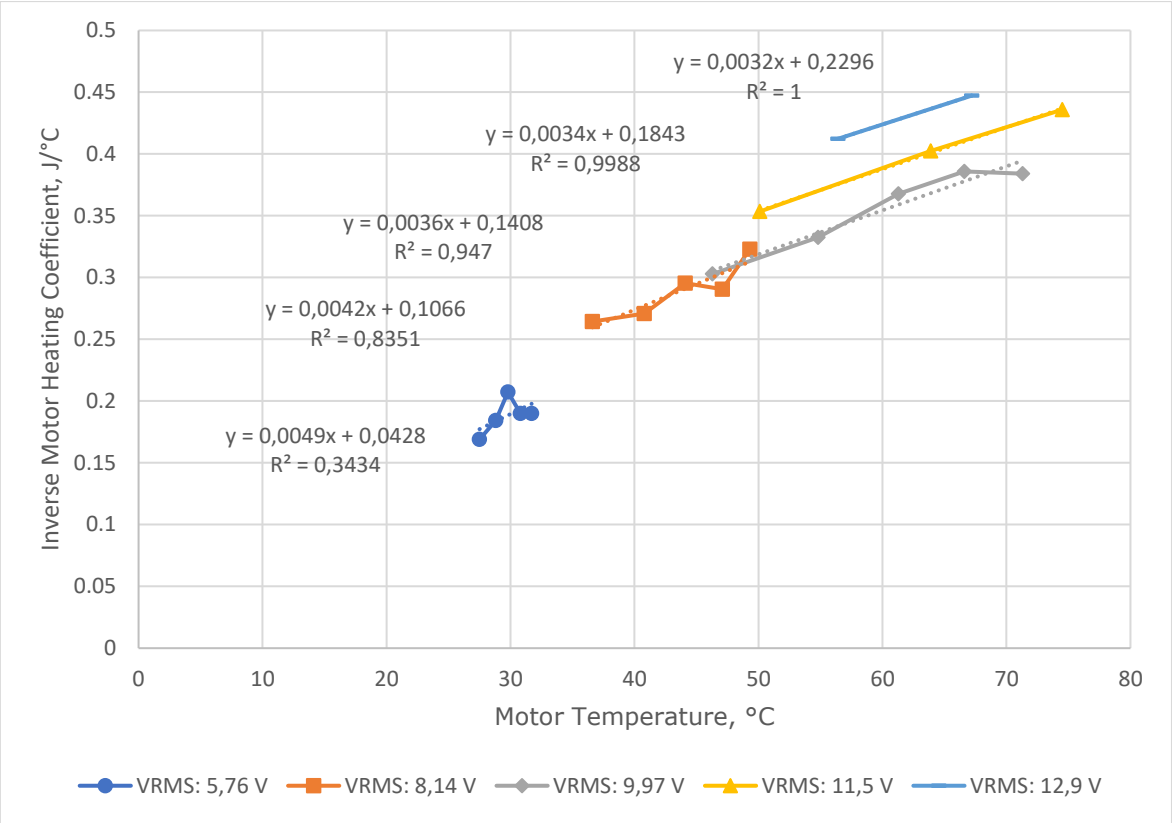

Figure 4.9 BGM9606-90T Heating Coefficient as function of Motor Temperature

Table 4.9 BGM9606-90T Heating Coefficient Slope and Y-Intercept Values by RMS Voltage

V_{RMS}	*slope*	*y-int*
5.76	0.0049	0.0428
8.14	0.0042	0.1066
9.97	0.0036	0.1408
11.5	0.0034	0.1843
12.9	0.0032	0.2296

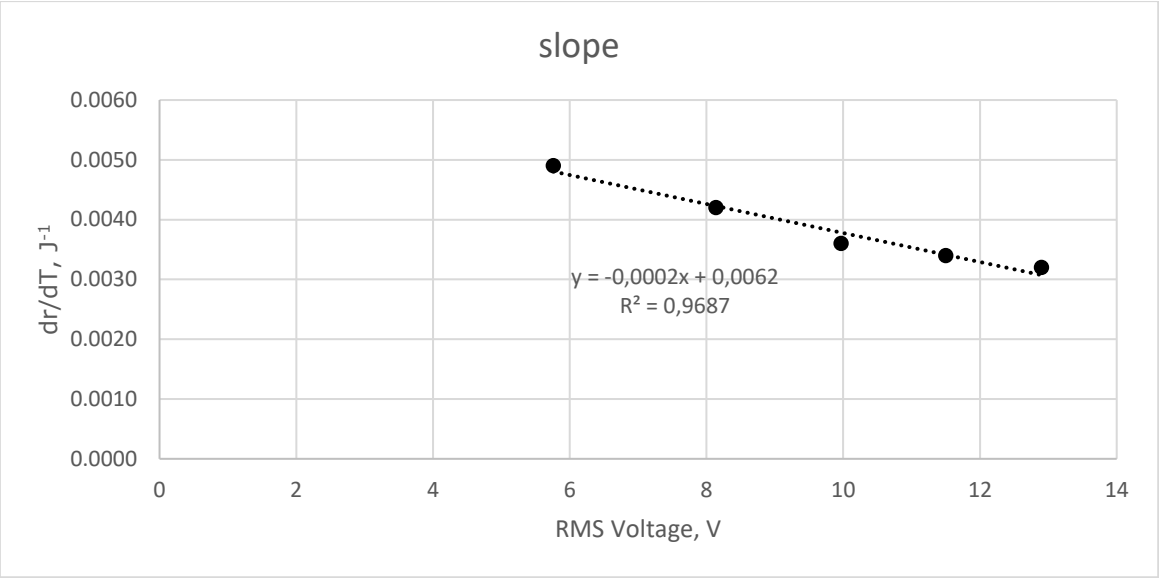

Figure 4.10 BGM9606-90T Slope Producing Equation for Heating Coefficient

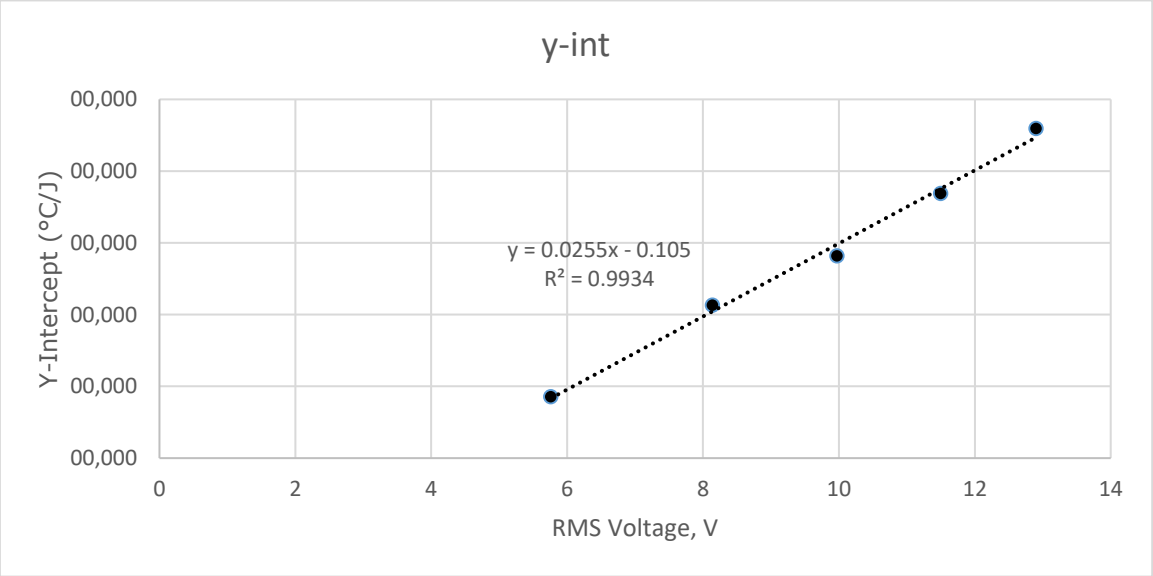

Figure 4.11 BGM9606-90T Y-Intercept Producing Equation for Heating Coefficient

Given an RMS voltage, the function for current can now be produced in slope-intercept form as shown in equation (4.15).

$$I = mT_{motor} + b \tag{4.15}$$

The slope of the motor heating coefficient equation can now be calculated with equation (4.16).

$$m = -0.0002V_{RMS} + 0.0062 \tag{4.16}$$

The y-intercept of the motor heating coefficient equation can now be calculated with equation (4.17).

$$b = 0.0255V_{RMS} - 0.105 \tag{4.17}$$

BGM5208-200-12 Coefficient of Heating

The motor coefficients of heating were then graphed for the various power settings as a function of motor temperature to produce figure (4.12). The motor coefficients are not linear, probably due to the low temperature changes in this experiment combined with the relative inaccuracy of the measurements. For this reason, the averages of each test, shown in table (4.10), were graphed. The linear best fit shown in figure (4.13) was used to describe the heating coefficient.

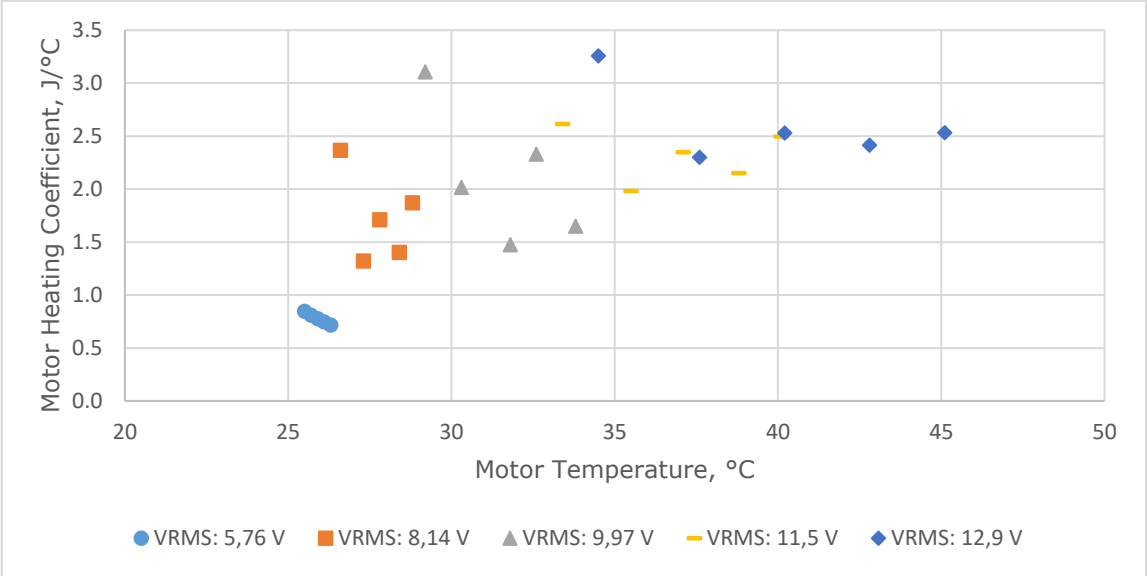

Figure 4.12 BGM5208-200-12 Heating Coefficient as function of Motor Temperature

Table 4.10 BGM5208-200-12 Calculated Heating Coefficient Values by RMS Voltage

V_{RMS} V	r_{avg} °C/J	r_{avg}^{-1} J/°C
5.76	1.2889	0.7758
8.14	0.6027	1.6593
9.97	0.5062	1.9756
11.5	0.4356	2.2958
12.9	0.3892	2.5692

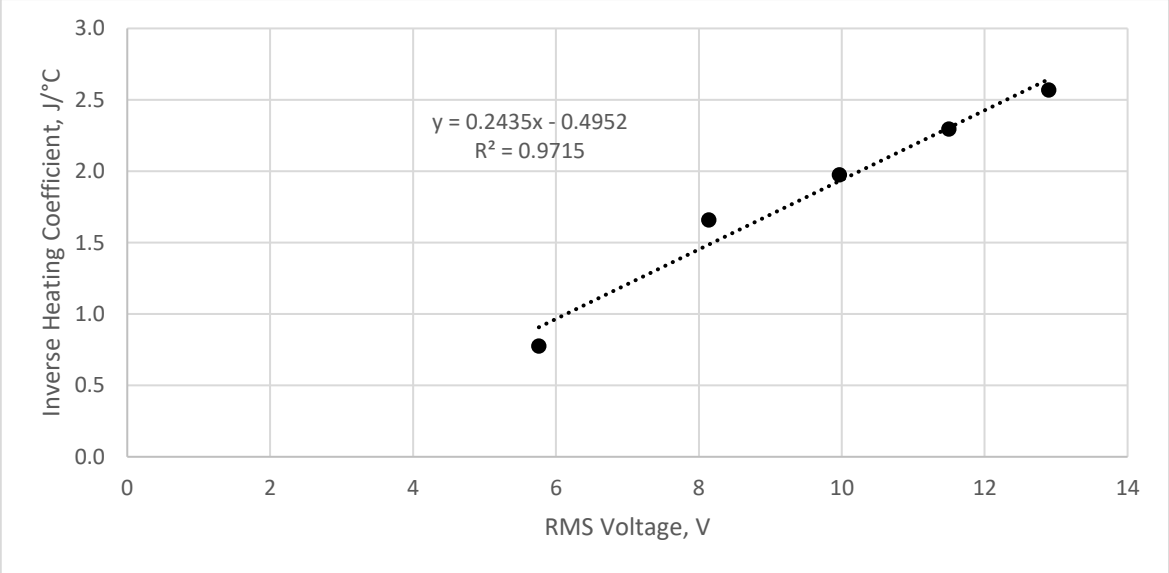

Figure 4.13 BGM5208-200-12 Slope Producing Equation for Heating Coefficient

Given an RMS voltage, the function for current can now be calculated with equation (4.18).

$$m = 0.2435V_{RMS} - 0.4952 \tag{4.18}$$

Calculating System Position Error

This prototype utilizes a pair of IMU sensors. The primary sensor is located on the camera mount and measures the orientation of the camera. The secondary sensor is mounted to the frame beneath the yaw axis motor and serves to improve the precision of stabilization [25]. The motion controller interprets the signals from both IMUs and calculates, among other things, the orientation of the primary sensor in space.

The motion controller is assigned a set of target angles (roll, pitch, and yaw) as parameters. Each program cycle, it attempts to move the motors as needed to match the orientation of the primary IMU with these target angles. Any difference between the two is considered position error. Therefore, error for each axis can be calculated as shown in equation (4.19).

$$error = A_{target} - A_{measured} \tag{4.19}$$

where *error* – position error, °
A$_{target}$ – target angle, °
A$_{measured}$ – measured angle, °

Calculating Air Temperature

Both the primary and secondary IMU have temperature sensors onboard. The readings from both sensors are averaged together to calculate air temperature, as shown in equation (4.20).

$$T_{air} = \frac{T_P + T_S}{2} \tag{4.20}$$

where T_p = temperature measured by primary IMU, °C
T_s = temperature measured by secondary IMU, °C

Calculating Motor Temperature

When the program executes, motor temperature is initialized as being equal to air temperature. The program then calculates the rate of change of motor temperature and calculates how long the loop has been running. The amount of temperature change is then calculated as shown in equation (4.21).

$$T_{motor2} = T_{motor1} + \frac{dT_{motor}}{dt}(t_2 - t_1) \tag{4.21}$$

where T_{motor2} – temperature of motor at time 2, °C
T_{motor1} – temperature of motor at time 1, °C
t_2 – time 2, s
t_1 – time 1, s

Calculating Power to Deliver

The power control algorithm separates the power parameter into ten discreet levels and steps between these levels based upon system error. This stepped adjustment mode simply divides the error by the maximum error value and multiplies by 10. This value is then rounded down to the nearest integer value and multiplied by 25 (approximately ten percent of the maximum value of 255). In this way, the system response is scaled

to one of ten steps directly proportional to the amount of position error. Large disturbances to the system are corrected by large system responses. If error is very large, the power parameter is capped at 255 (the maximum allowable). This process is described by equation (4.22).

$$P_{param} = 25\left(integer\left(10\frac{error}{error_{max}}\right)\right) \tag{4.22}$$

where $error_{max}$ – error value to apply max voltage, °, Euler Angle

Calculating Power Limit

In the event the system is operating in a dynamic environment which requires high power supply to the motors for extended periods of time, the power must be limited to protect the motors even if performance is reduced. Firstly, a maximum allowable motor temperature is defined in the algorithm; for this prototype, a value of 60 °C was selected. (The gimbal components were printed in PLA, which softens above this temperature. Otherwise, 75 °C would have been used to protect the motors.) A safety temperature is calculated as shown in equation (4.23).

$$T_{safety} = T_{max} - \left(t_{safety}\frac{dT_{motor}}{dt}\right) \tag{4.23}$$

where T_{safety} – safety temperature,
T_{max} – maximum allowable motor temperature,
t_{safety} – safety time, s

The safety temperature is effectively the temperature at which, at the motor's current rate of heating, the motor must not exceed if it is to remain operational for the designated safety time; for this prototype, a value of 30 seconds was used.

If the motor temperature exceeds the maximum allowable temperature, the power parameter is set to 1 (a value of 0 is not valid). If the motor temperature does not exceed the allowable but does exceed the safety temperature, the maximum allowable power parameter is reduced by 1 each cycle until either the parameter becomes equal to 1 or the safety temperature rises above the motor temperature. If the motor temperature is below the safety temperature, the maximum allowable power parameter is increased by 1 each cycle until the maximum value of 255 is reached or the motor temperature exceeds the safety temperature. Each cycle, if the calculated power parameter is greater than the maximum allowable, it is reduced to the maximum allowable instead. Figure (4.14) shows how the safety temperature prevents a motor from overheating.

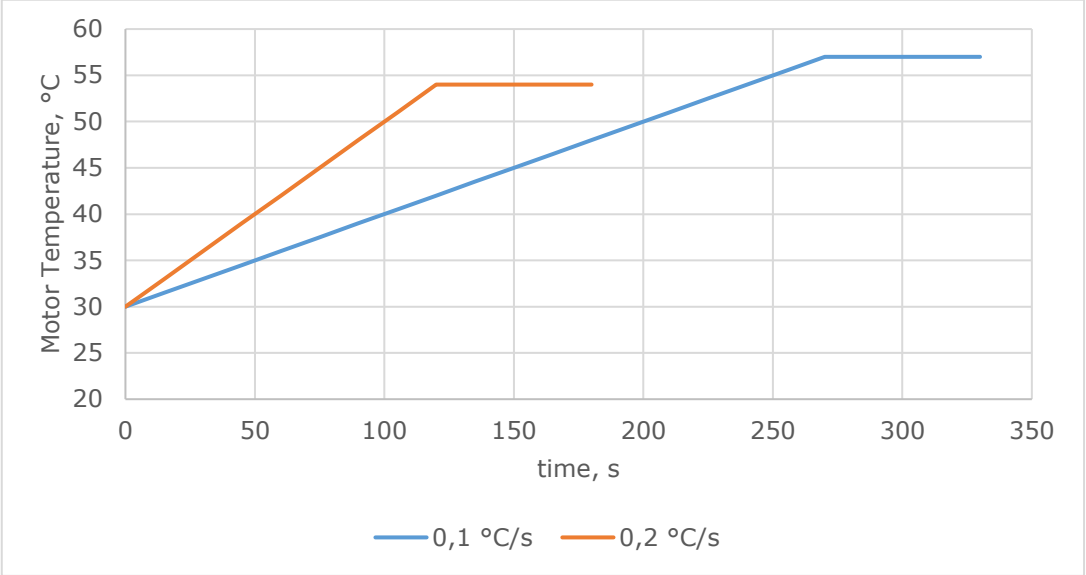

Figure 4.14 Theoretical Safety Temperature as a Function of Time

Calculating Power Usage

The dynamic power control algorithm response is tested by observing the IMU measurements while the gimbal is at rest, in motion, and subject to large disturbances. For each motor, the positional error, current, and power parameter is recorded every program cycle (approximately 30 ms). The power for each motor is calculated using equation (4.13). Total energy consumed is calculated using the Riemann sum technique, as shown in equation (4.24) [30].

$$\sum_{t=0}^{n-1} P(t)\Delta t \tag{4.24}$$

where *P(t)* – calculated power

Total energy is then divided by the total time of the observations to yield an average power consumption for the experiment, as shown in equation (4.25).

$$E = Pt$$

$$P = \frac{E}{t} \tag{4.25}$$

where *E* – energy, J

Percent difference is calculated according to equation (4.26).

$$Difference = 100\left(\frac{P_1 - P_2}{P_1}\right) \tag{4.26}$$

where *Difference* – percent difference between power consumption, %

Calculating Proportional Control Parameter

The proportional control parameter, *P* in *PID*, must be recalculated every time the power parameter is changed. Because motor torque increases with power, the proportional parameter must be reduced to keep the system response the same. Because this is an inverse relationship, the value is higher for low power settings and vice versa. The P parameters are determined for the lowest and highest values used by the system by using the auto-tune function included with the motion controller. The necessary P value is then interpolated from these values for each motor as shown in equation (4.27).

$$P_{PID} = P_{paramLow} - \left(\frac{P_{param}}{255}\right)\left(P_{paramLow} - P_{paramHigh}\right) \tag{4.27}$$

where P_{PID} – proportional parameter of PID control
$P_{paramLow}$ – the proportional parameter for the lowest power parameter
$P_{paramHigh}$ – the proportional parameter for the hightest power parameter

4.3 System Control

System Layout

Figure (4.15) shows the layout of the electromechanical components and how they interact with one another. The SimgleBGC motion controller is the heart of the system and drives the motors. As the motors move, the IMUs also move based on their location within the system. The IMUs send data back to the motion controller which it interprets as motor position and compares with the target position to determine error. This positional error is read by the Arduino Nano microprocessor, which runs the power control algorithm. The algorithm determines the appropriate power level to operate each motor at based on positional error, then calculates a new proportional parameter. The calculated power levels and proportional parameters are then sent to the motion controller and updated. The cycle then repeats for the next program cycle.

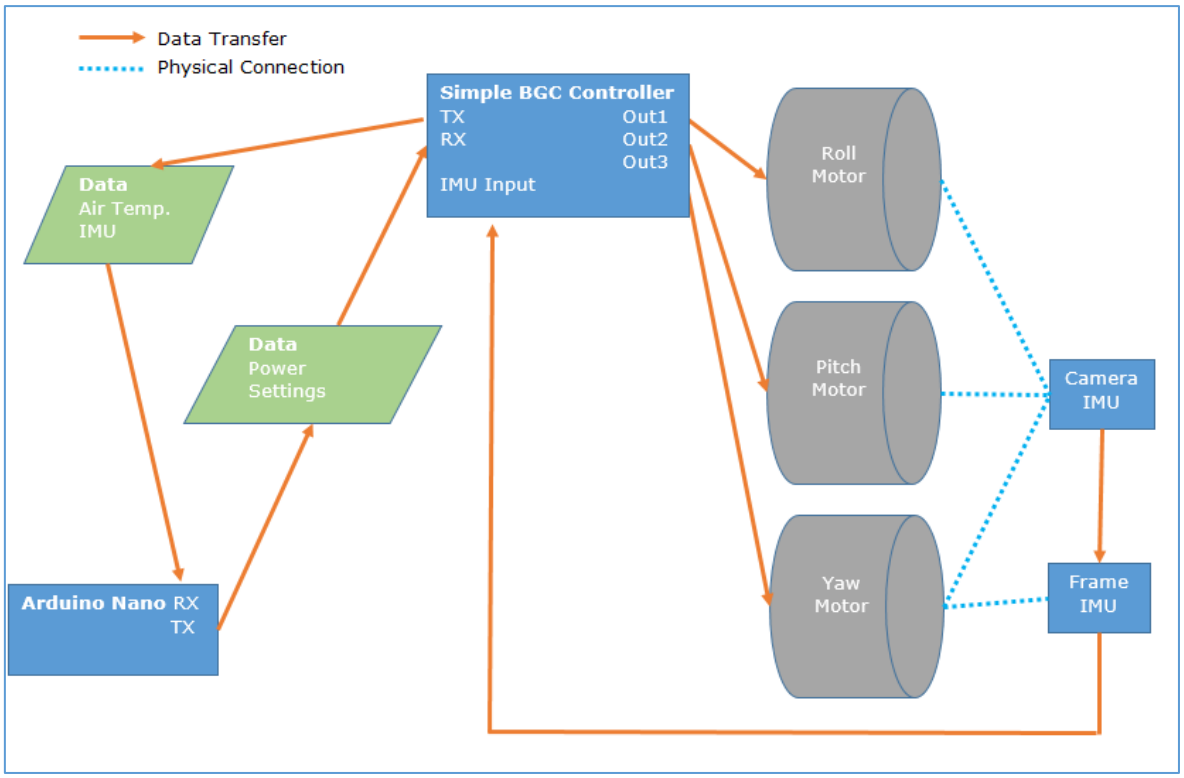

Figure 4.15 Electromechanical System Diagram

Serial API and Communication

The SimpleBGC motion controller includes a serial API which allows communication with an external device. Commands can be both sent and received via serial connection, which facilitates the retrieval of IMU and system state data from the motion controller in real time, as well as updating its parameters. An Arduino Nano microcontroller was connected to the motion controller on this prototype.

The motion controller parses incoming commands every 8 ms, which are sent by the microcontroller. The SBGC_CMD_REALTIME_DATA_4 was used to retrieve air temperature, IMU Angles, and Target Angles. The SBGC_cmd_set_adj_vars_var_t command was used to write new power parameters and PID parameters to the roll, pitch, and yaw axes.

4.4 Program Flow and Calculations

The system control algorithm is executed on an Arduino Nano. The TX and RX pins on the Arduino are connected directly to the RX and TX pins on the SimpleBGC motion controller, respectively. These two connections are all that are required for serial connection between the two controllers. The algorithm works by first sending a real-

time data request (from the Arduino) to the serial buffer on the SimpleBGC controller. The controller processes this command and sends the requested data to the Arduino serial buffer. The algorithm then makes all necessary calculations to determine power and PID parameters for each axis based on the mathematical model. The Arduino then sends the commands to update these parameters to the SimpleBGC serial buffer. The program then loops back the beginning and runs again continuously as long as the Arduino is powered. The flowchart in figure (4.16) shows the general order in which routines are executed.

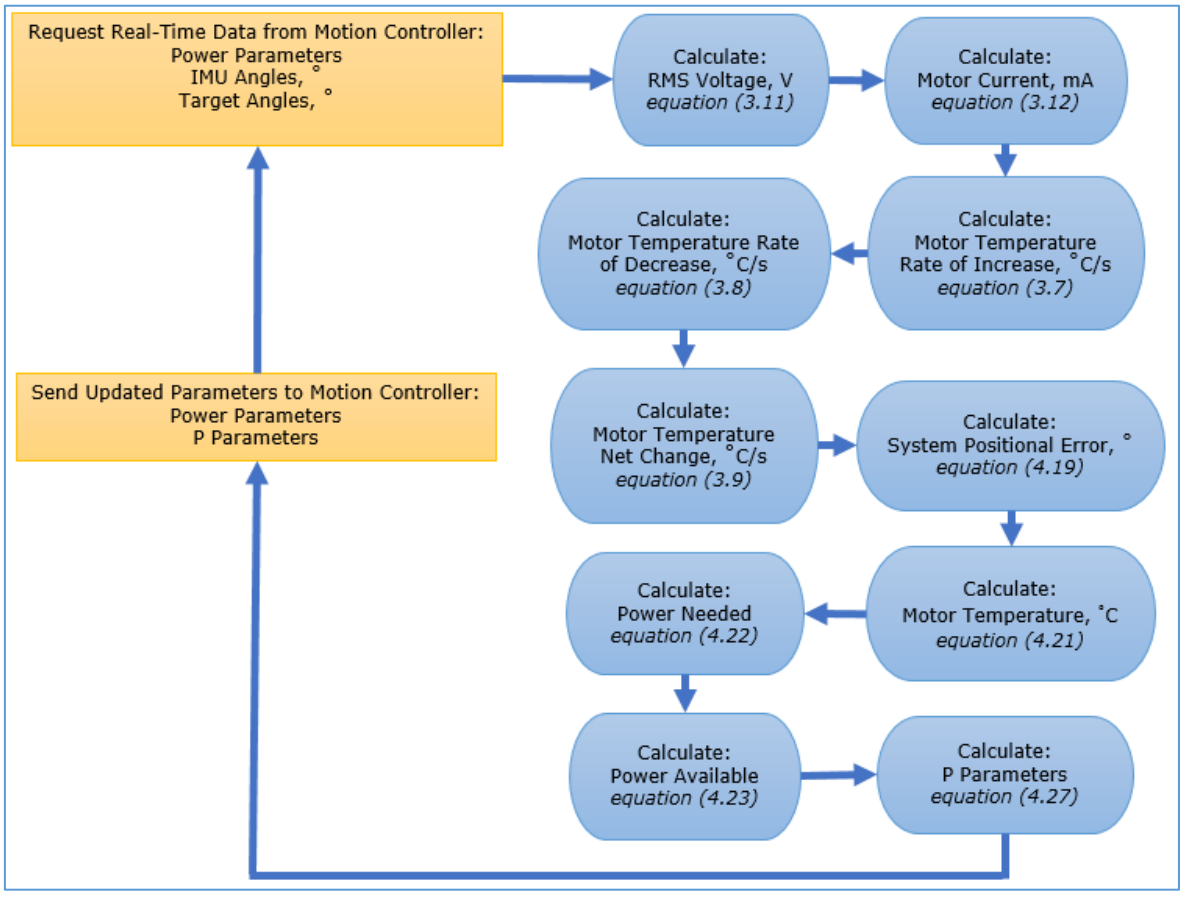

Figure 4.16 Program Process Flowchart

5 RESULTS AND ANALYSIS

5.1 Temperature Modeling Test

Testing the Rate of Cooling Model

The results from the rate of cooling test are shown below in table (5.1). These results are graphed in figures (5.1) and (5.2) to show the relative accuracy of the cooling model for the BGM5208-200-12 and BGM2606-90T motors, respectively.

Table 5.1 Results of Motor Cooling Test

	BGM5208-200-12		BGM2606-90T	
t min	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C
0	47.0	47.0	54.0	54.0
1	45.8	45.9	47.1	51.0
2	44.7	44.9	43.2	48.5
3	43.4	44.1	40.9	46.2
4	42.1	43.2	39.2	44.2
5	41.1	42.3	38.0	42.3

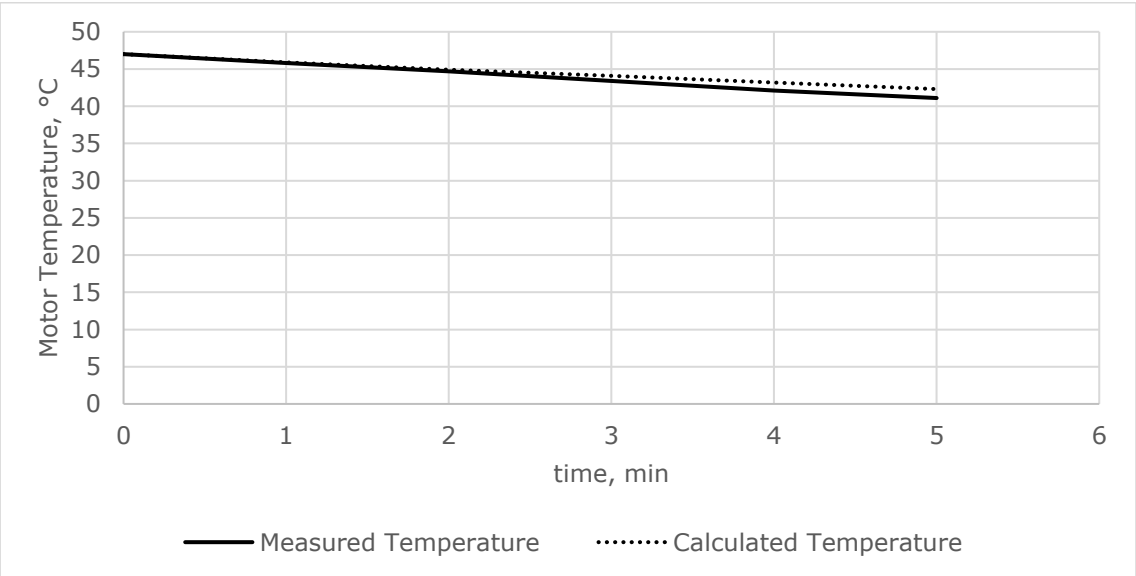

Figure 5.1 BGM2606-90T Measured Versus Calculated Temperature While Cooling

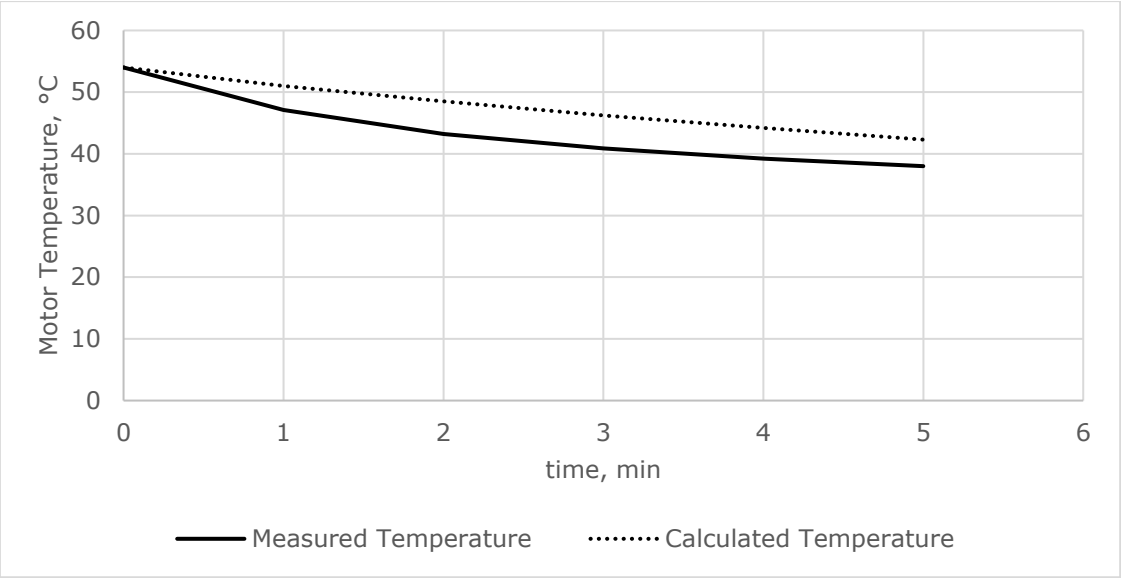

Figure 5.2 BGM5208-200-12 Measured Versus Calculated Temperature While Cooling

Testing the Rate of Heating Model

The results of the rate of heating test for the BGM5208-200-12 motor are shown below in table (5.2). These results are graphed in figures (5.3), (5.4), and (5.5) to show the relative accuracy of the heating model for each of the power settings tested. The calculated values are quite different from the measured values, likely due to the small temperature range during testing.

Table 5.2 BGM5208-200-12 Heating Test Results

	Power Parameter: 150 V_{RMS} = 9.97 V		*Power Parameter*: 200 V_{RMS} = 11.5 V		*Power Parameter*: 250 V_{RMS} = 12.9 V	
t min	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C
0	25.6	25.6	33.0	32.9	30.2	30.2
1	26.1	27.9	33.2	34.8	31.6	32.6
2	26.5	29.7	33.6	37.0	32.4	36.6
3	26.9	31.1	34.2	39.0	33.2	40.1
4	27.3	32.4	34.9	40.9	34.0	43.2
5	27.6	33.6	35.2	42.5	34.5	46.1

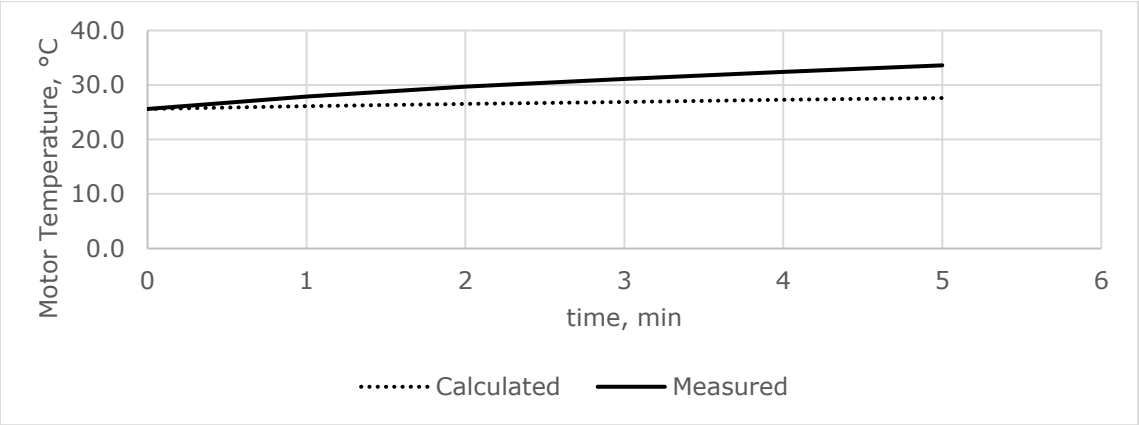

Figure 5.3 BGM5208-200-12 Motor Heating Model Test Results at Power Parameter 150

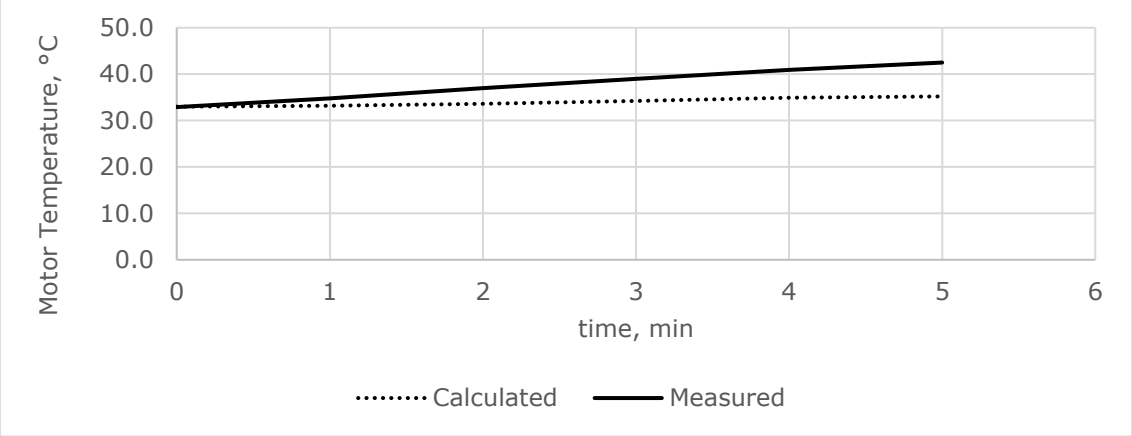

Figure 5.4 BGM5208-200-12 Motor Heating Model Test Results at Power Parameter 200

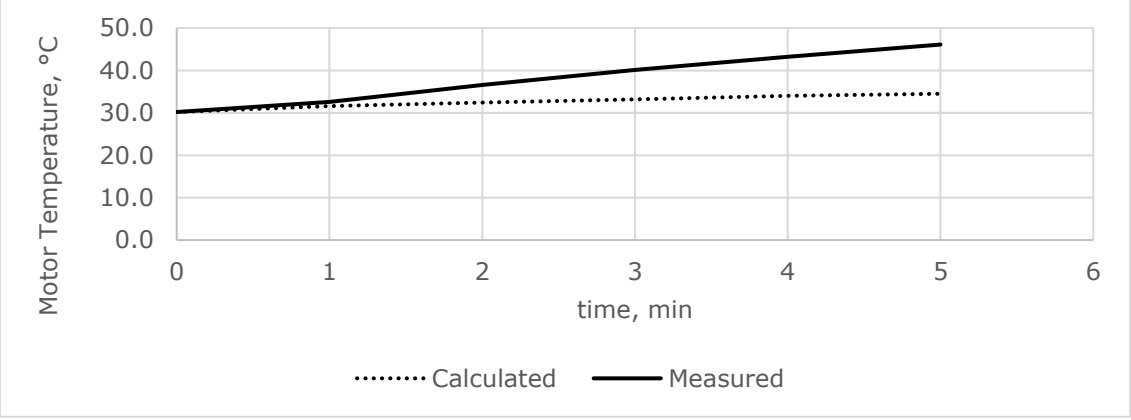

Figure 5.5 BGM5208-200-12 Motor Heating Model Test Results at Power Parameter 250

The results of the rate of heating test for the BGM2606-90T motor are shown below in table (5.3). These results are graphed in figures (5.6), (5.7), and (5.8) to show the

relative accuracy of the heating model for each of the power settings tested. The calculated temperatures are very close to those measured, and conservative in their difference.

Table 5.3 BGM2606-90T Heating Test Results

t min	*Power Parameter*: 150 V_{RMS} = 9.97 V		*Power Parameter*: 200 V_{RMS} = 11.5 V		*Power Parameter*: 250 V_{RMS} = 12.9 V	
	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C
0	26.8	26.8	30.5	30.5	33.6	33.6
1	40.9	38.7	49.6	46.7	58.6	53.5
2	49.9	48.1	63.0	58.0	79.1	68.2
3	56.7	55.7	75.2	69.6	-	-
4	62.4	61.3	-	-	-	-
5	-	-	-	-	-	-

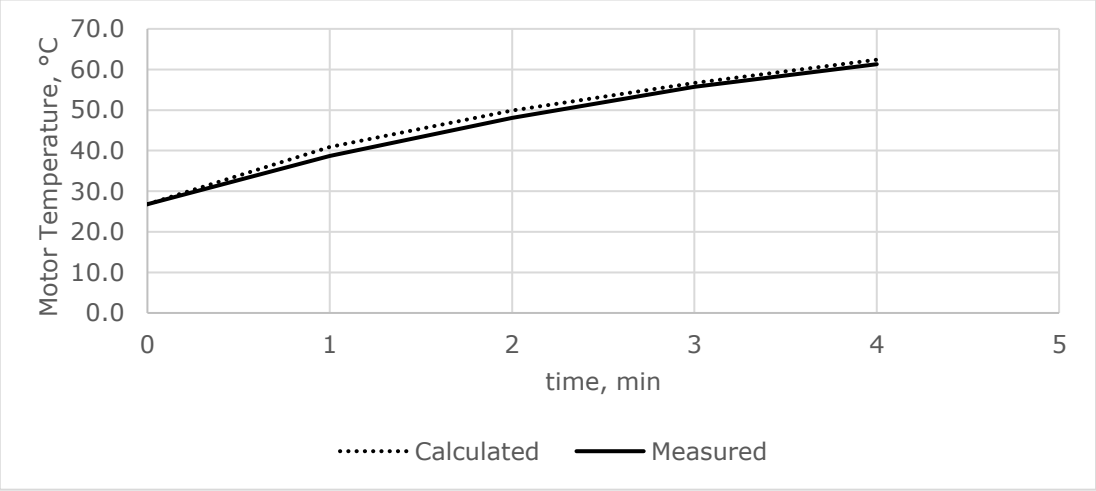

Figure 5.6 BGM2606-90T Motor Heating Model Test Results at Power Parameter 150

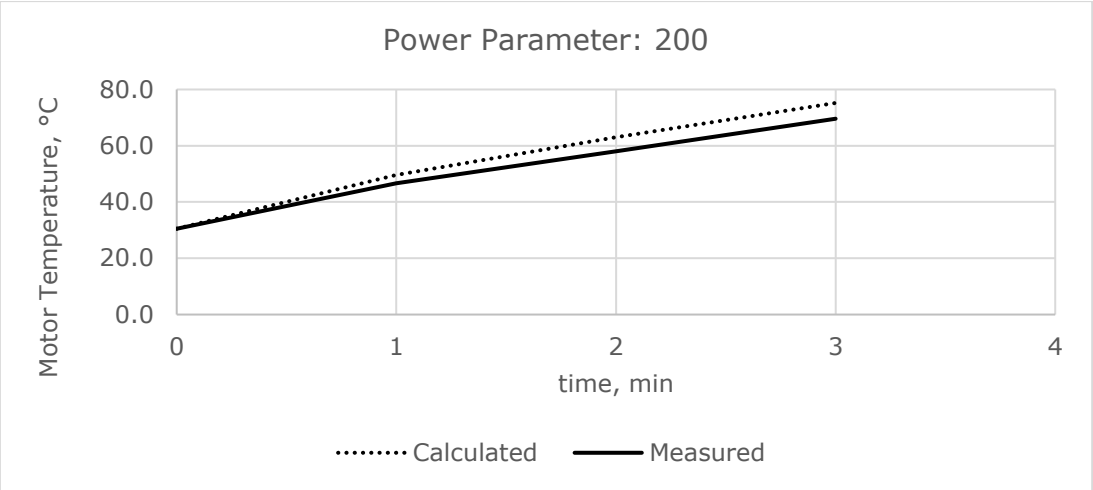

Figure 5.7 BGM2606-90T Motor Heating Model Test Results at Power Parameter 200

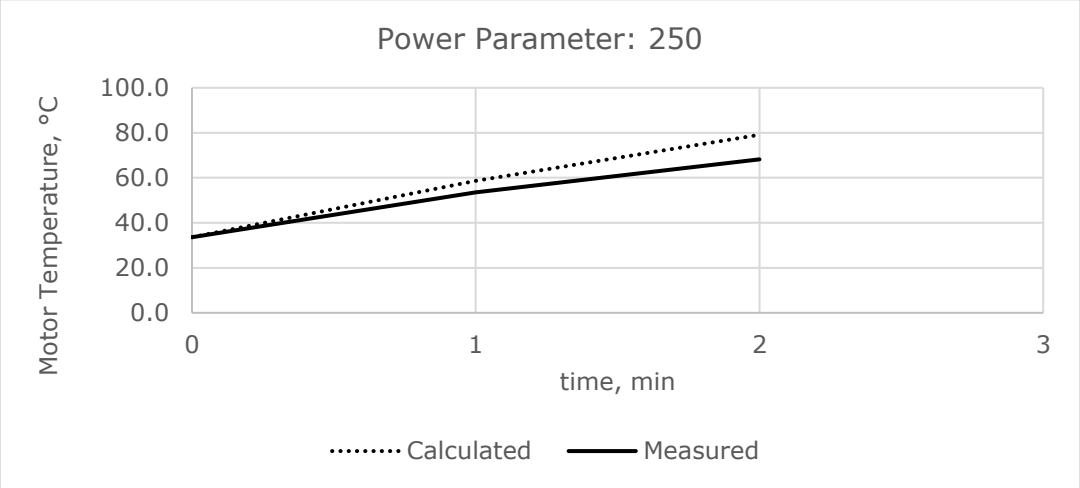

Figure 5.8 BGM2606-90T Motor Heating Model Test Results at Power Parameter 250

Overtemperature Protection Results

The results of the overtemperature protection test are shown in table (5.4), and the data are plotted in figures (5.9), (5.10), and (5.11) for power parameters of 150, 200, and 250, respectively. The calculated temperatures are conservative, but effectively prevent the motor from overheating.

Table 5.4 Overtemperature Protection Test Results

t min	*Power Parameter*: 150 V_{RMS} = 9.97 V		*Power Parameter*: 200 V_{RMS} = 11.5 V		*Power Parameter*: 250 V_{RMS} = 12.9 V	
	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C	$T_{calculated}$ °C	$T_{measured}$ °C
0	35.0	34.6	34.8	34.4	32.9	33.0
1	44.5	43.9	53.6	49.3	52.9	48.9
2	53.5	51.1	59.2	53.7	59.0	54.0
3	59.0	56.3	59.9	54.1	59.9	54.2
4	59.8	57.2	60.0	54.5	60.0	54.5
5	59.9	57.4	60.0	55.0	60.0	55.0

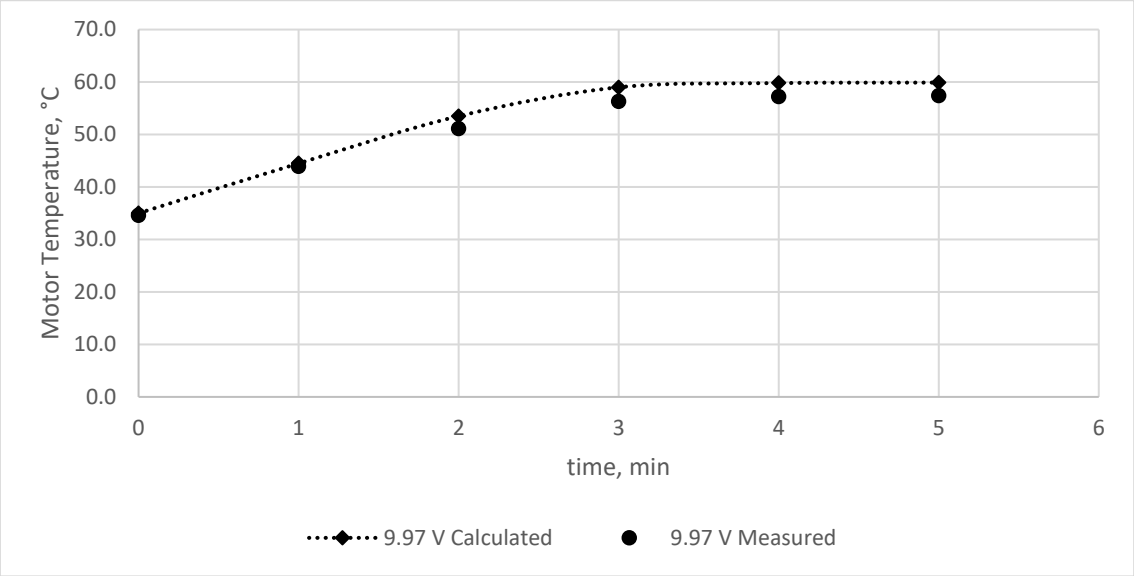

Figure 5.9 Heating and Overtemperature Test Results at V_{RMS} 9.97

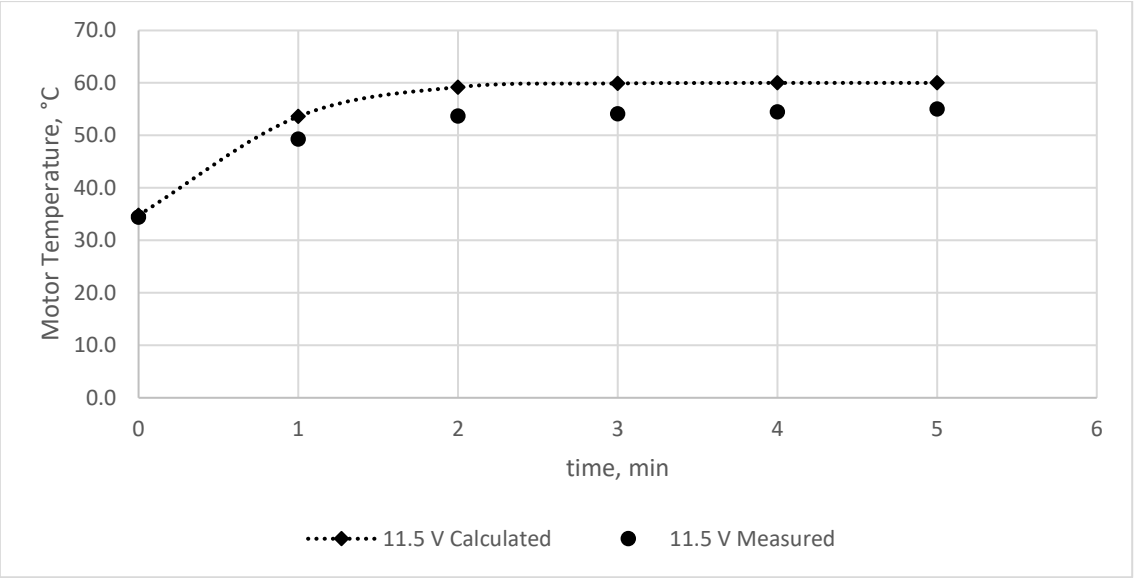

Figure 5.10 Heating and Overtemperature Test Results at V_{RMS} 11.5

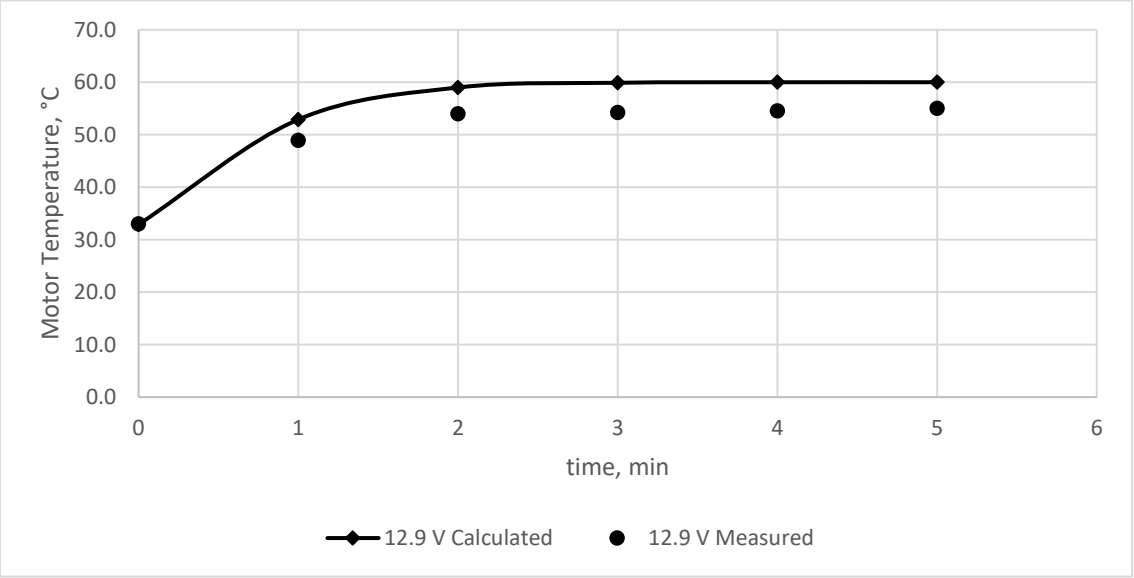

Figure 5.11 Heating and Overtemperature Test Results at V_{RMS} 12.9

5.2 Error Correction Test

The following results demonstrate the performance of the system under two conditions. The first is rapid movement of the gimbal to simulate fast disturbances to the drone platform in flight, for instance gusting wind. The dashed lines show the position error measured by the IMU, while the solid lines indicate the power level supplied by the motion controller in terms of power parameter. Error values were normalized to match the corresponding power parameter, and large error values were intentionally cropped

to maintain the shape of the curve. Figures (5.12) and (5.13) show the performance of the roll and pitch axes, respectively.

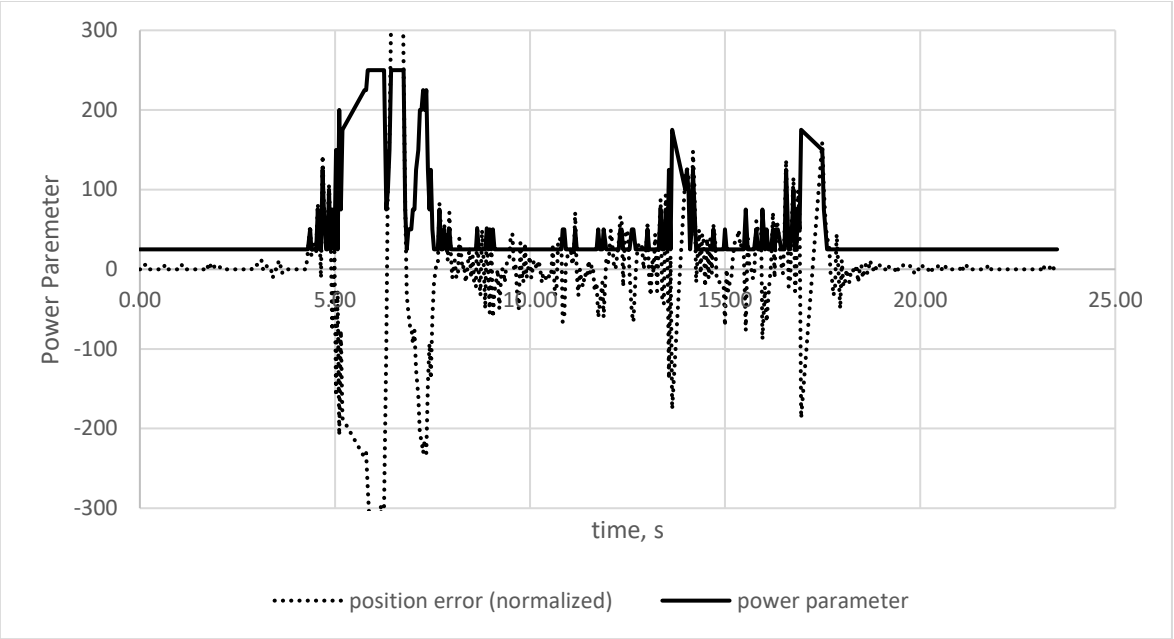

Figure 5.12 Roll Axis System Response under Moving Condition

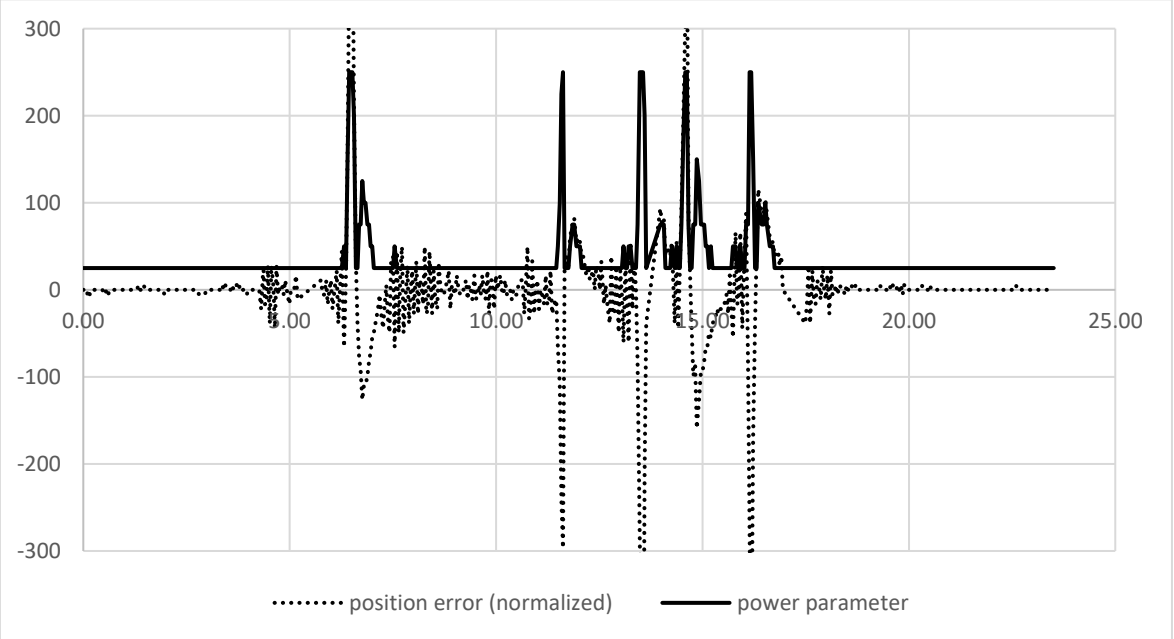

Figure 5.13 Pitch Axis System Response Under Moving Condition

The second condition simulates a series of sudden and large disturbance to the gimbal itself, such as a wind load or impact rapidly shifting the position of the camera. Again,

the dashed lines show the position error measured by the IMU, while the solid lines indicate the power level supplied by the motion controller in terms of power parameter. Error values were normalized to match the corresponding power parameter, and large error values were intentionally cropped to maintain the shape of the curve. Figures (5.14) and (5.15) show the performance of the roll and pitch axes, respectively.

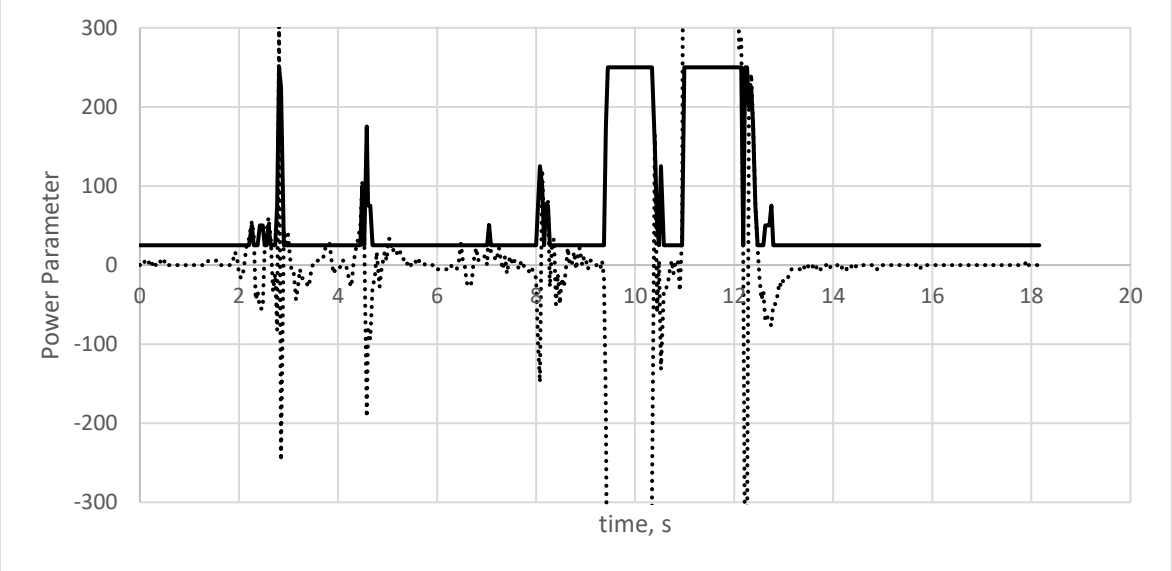

Figure 5.14 Roll Axis System Response Under Large Disturbances

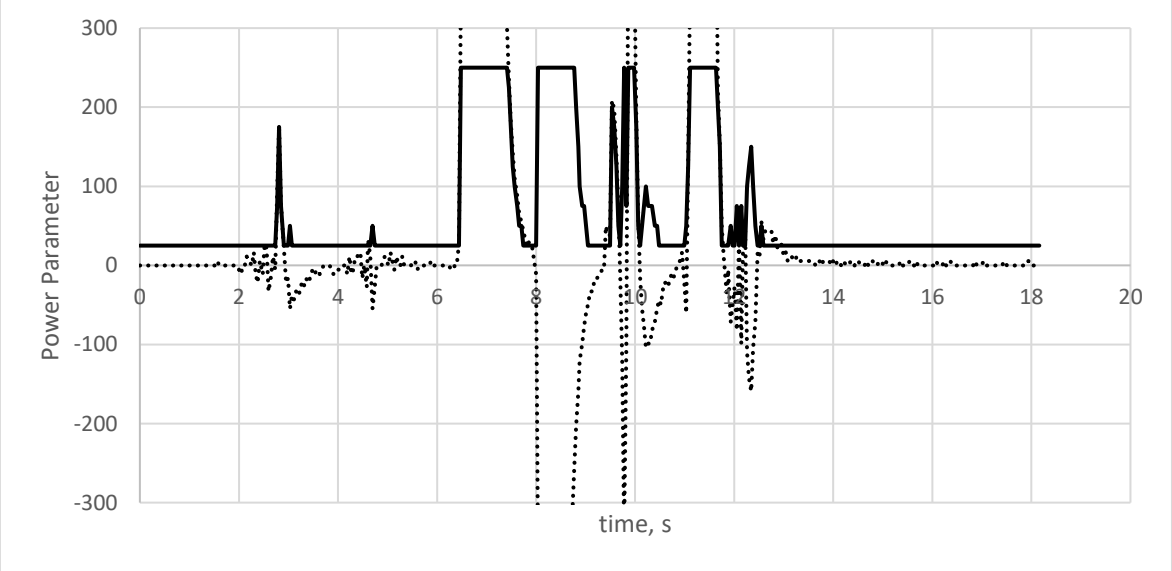

Figure 5.15 Pitch Axis System Response Under Large Disturbances

The yaw response is discussed separately because it had very poor performance. For an unknown reason, the motion controller was returning IMU error values 18 times larger than those for the roll and pitch axes. Because the algorithm was tuned for much smaller values, the system response was far too large. Figure (5.16) shows the yaw error and response. The test started with the system in the moving condition, then a series of large disturbances, then finishing with the gimble at rest.

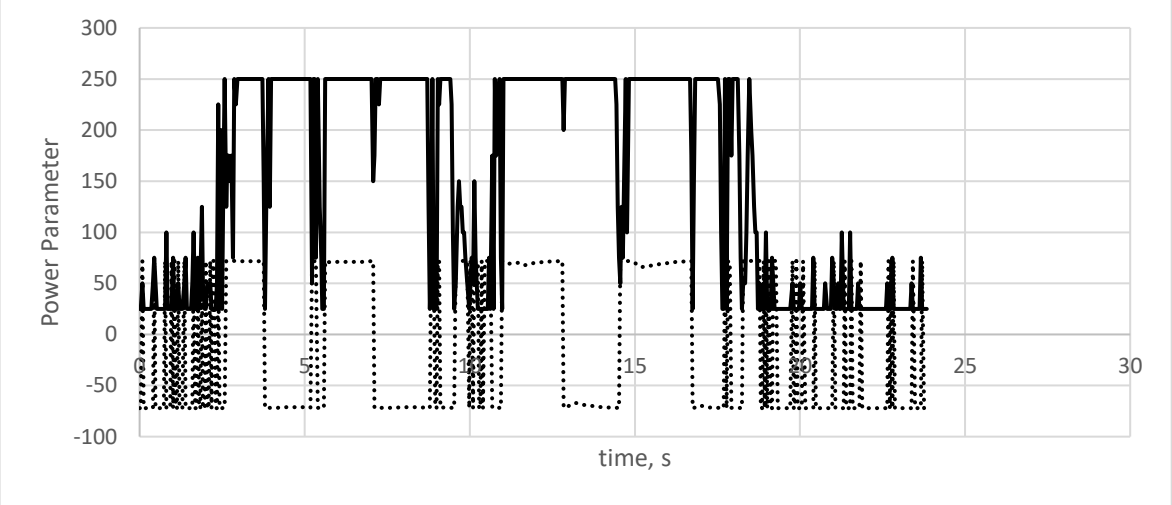

Figure 5.16 Yaw Axis System Response Under Small and Large Disturbances

5.3 Power Consumption Comparison

The power savings of this approach can be estimated by making a few assumptions about how the system is used. Per the motion controller manual, a power setting is selected based on how hot each motor gets. During testing, it was observed that the roll and pitch motors were only able to operate continuously at an RMS voltage of 7.7 V and remain below 60 °C, so this is the voltage assumed for standard operation. Based on equation (4.1), it can be assumed that the motors will draw 141 mA (as measured from the power supply at 13 V). Therefore, each motor will draw 1.8 W continuously. Because of the poor performance of the yaw motor with this technique, it will be ignored for these calculations.

The nature of disturbances will have no influence on the power consumed with the standard setup as it is constant. However, large disturbances will cause a substantial difference in the dynamic algorithm. Therefore, it is assumed that the system will operate under the same conditions as the disturbance tests. The data from those tests is integrated over time to produce energy, then divided by the time to produce Watts. According to these estimates, the roll axis will consume 1.4 W on average, whereas the pitch axis will consume 0.7 W. This represents a power savings of 22.2 % and 61.1 % for the roll and pitch axes, respectively.

6 CONCLUSIONS

Overall, the prototype functioned well and met all of the objectives of this work. The temperature modeling was adequate, the motor protection excellent, the system response performed well in many ways and poorly in a few, and the potential power savings of the algorithm are substantial. Overall, the design of the gimbal proved to be effective.

6.1 Temperature Modeling

The temperature model works well to predict the rate at which a motor will heat while powered as well as cool while unpowered. The algorithm is more accurate for the roll and pitch axes in large part because the data spanned a much larger temperature range than the yaw motor. The results tend to be a bit conservative, but this is not a problem given the objective of preventing motor overheating. It is important to note that air currents will cool a motor faster than predicted since all measurements were limited to the laboratory environment, however this is likely to make the system even more conservative and not manifest as a problem. A fully functional model will have aerodynamic covers which limit air currents across the motor.

6.2 Motor Protection

The motor protection routine works very well to prevent an overheat situation. The system successfully limits maximum power available to a motor as it heats, and provides a larger buffer as a motor heats more quickly. This greatly decreases the chances that the maximum defined temperature will ever be reached. As the rate of heating decreases, the buffer decreases, allowing more power to be consumed again. Once this algorithm was implemented, a motor never exceeded the maximum defined temperature during testing.

6.3 System Response

Acceptable positional error is a function of distance to target and ground speed, according to the blur equation. It must be decided by the operator how much error is acceptable and tune the system accordingly. Both the maximum positional error value and PID parameters must be tuned to provide a system response which is slow enough to save power but fast enough to prevent any image blur. This thesis utilized the auto-tuner function included with the motion controller, which proved to be functional, but

there are occasional oscillations which would affect image quality, especially when the gimbal is positioned at extreme angles relative to its initial calibration. Further PID analysis is required for a fully functional system.

6.4 Power Savings

For this particular design and setup, power savings were shown to be significant for the roll and pitch axes, at 22.2 % and 61.1% power savings respectively. The yaw axis performed relatively poorly with the same algorithm, but further advancement of a yaw specific algorithm will likely yield good results. The motor selected for the yaw axis was capable of utilizing higher RMS voltage than those used with the roll and pitch axes, which could provide improved performance as well. Regardless, balance proved to be critical to the quality of stabilization provided by a motor, especially at lower power settings.

7 SUMMARY

This work demonstrates a successful design capable of mounting a hyperspectral linescan camera to a fixed wing drone, paired with a power conserving algorithm that dynamically scales power to each motor as needed. The system created will provide a platform upon which other thesis work can be completed within the mechatronics department.

The control algorithm successfully adjusts power delivery to each gimbal motor in real time based on positional error. It also prevents the motors from overheating if the demand for power remains too high for too long. While system performance will suffer, the system protects itself to prevent permanent damage. The PID parameters successfully scale with the changing power level, preventing under and over-dampened response.

While the work in this thesis demonstrates the potential efficiency gains that dynamically adjusting power can yield in an inertial gimbal system, there are many improvements that can be made to the prototype developed here.

First of all, a well balanced system is critical to reducing power consumption. The center of mass must be located along the axis of each motor to prevent gravity from inducing a rotation that the motor must counter, thereby increasing the necessary torque to hold the axis in the desired position. Dangling cables increase motor load both by being pulled and compressed as the axis moves and limit the rotation of the system. Slip rings were not available, but would offer a significant improvement in efficiency, especially if the camera requires an external cable.

The frame was designed to be easily prototyped and 3D printed. While every reasonable attempt was made to reduce its weight and keep the system balanced, the final design is by no means optimized. Further study and refinement of the mechanical system could offer improvements.

The PID parameters were generated with the assistance of the auto-tuner routine included with the SimpleBGC controller. A proper study of the system response as a function of RMS voltage and motor temperature will likely yield better results. The prototype is susceptible to oscillations at certain orientations, especially at extreme angles.

The dynamics properties of the power supply under load were ignored in this study. A bench top power supply was used, and batteries will likely behave differently under load. Adjusting power and PID parameters based on battery drain may be required for a fully functional model.

While the Flir thermal scope was very effective, its precision of measurement is limited to one tenth of one degree. This produces poor curves when the rate of temperature

change is slow. Also, the methods used to record data were not automated and thus could not be performed quickly. Using a smaller measurement interval will yield more accurate graphs and therefore better fit curves. Faster measurements will also allow temperatures to be recorded at higher RMS voltage settings which overheated the small motors quickly in this study.

The material used in this protytpe is PLA, and therefore softens at 60 °C. The maximum temperature can be increased to be closer to the limitations of the motors with a more heat resistant material. Also, the screws regularly loosened as the PLA softened, requiring maintenance often.

A dynamic "holding torque" variable can be implemented to adaptively determine how much holding torque each axis needs based on the flight data.

Using the version of the motion controller that measures current can allow the system to self-initialize the motor temperatures based on power parameter and current draw. The system currently assumes air temperature if not overwritten by the user.

Kokkuvõte

See töö demonstreerib õnnestunud disainilahendust, kus on paigaldatud hüperspektraal linescan kaamera fikseeritud tiivaga droonile ja rakendatud toitesäästlik algoritm, mis dünaamiliselt reguleerib iga mootori võimsust vastavalt vajadusele. Loodud süsteem pakub platvormi, mille abil saab mehhatroonika laboris teha uusil lõputöid.

Juhtimisalgoritm reguleerib positsioonivea põhjal edukalt reaalajas iga kardaanmootori energiatarbimist. Samuti hoiab see ära mootorite ülekuumenemise, kui energiatarve püsib liiga kaua liiga kõrge. Ehkki süsteemi jõudlus kannatab, kaitseb süsteem end püsivate kahjustuste vältimiseks. PID-parameetrid muutuvad koose muutuva võimsustasemega, hoides ära ala- ja üle summutatud reageerimise.

Kuigi käesoleva lõputöö käigus näidatakse potentsiaalset efektiivsuse kasvu, mida dünaamiliselt reguleeriva jõu abil saab inertsiaalses pöördesüsteemis kasutada, on siin välja töötatud prototüübi jaoks võimalik edaspidi palju täiendusi.

Esiteks on hästi tasakaalustatud süsteem kriitiline energiatarbimise vähendamiseks. Massi keskpunkt peab asuma piki iga mootori telge, et gravitatsioon ei põhjustaks pöörlemist, millele mootor reageerib, suurendades sel viisil vajalikku pöördemomenti telje hoidmiseks soovitud asendis. Rippuvad kaablid suurendavad mootori koormust nii painutamisel kui ka sirgestumisel telje liikumisel ja piiravad süsteemi pöörlemist. Libisevad rõngaskontaktid polnud saadaval, kuid need parandaksid tõhusust märkimisväärselt, eriti kui kaamera vajab välist kaablit.

Raam oli mõeldud hõlpsaks prototüüpide valmistamiseks ja 3D-printimiseks. Ehkki selle efektiivsust üritati vähendada ja süsteemi tasakaalus hoida, pole lõplik ülesehitus mingil

juhul optimeeritud. Mehaanilise süsteemi täiendav uurimine ja täiustamine võiks pakkuda täiendusi.

PID-parameetrid genereeriti SimpleBGC kontrolleriga kaasasoleva automaatse häälestaja rutiini abil. Süsteemi reaktsiooni nõuetekohane uurimine RMS-i pinge ja mootori temperatuuri funktsioonina annab tõenäoliselt paremaid tulemusi. Prototüüp on tundlik võnkumistele teatud orientatsioonide korral, eriti äärmiste pöördenurkade korral.

Selles uuringus jäeti kõrvale koormuse all oleva toiteallika dünaamika omadused. Kasutati fikseeritud pinge toiteallikat ja akud käituvad koormuse korral tõenäoliselt erinevalt. Täielikult töötava mudeli jaoks võib olla vajalik aku tühjenemisel põhinevate võimsuse ja PID-parameetrite reguleerimine.

Kuigi Flir kaamera temperaturri mõõtmise ulatus oli väga lai, on selle mõõtmise täpsus piiratud ühe kümnendikuga ühest kraadist. Kui temperatuuri muutuse kiirus on aeglane, saadakse ebaadekvaatsed kõverad. Samuti ei olnud andmete registreerimiseks kasutatavad meetodid automatiseeritud ja seetõttu ei olnud neid võimalik kiiresti teostada. Väiksema mõõtmisintervalli kasutamisel saadakse täpsemad graafikud ja seetõttu sobivad kõverad paremini. Kiiremad mõõtmised võimaldavad temperatuure registreerida ka kõrgema RMS-pinge korral, mis selles uuringus väikemootorid üle kuumendas.

Selles prototüübis on kasutatud konstruktiivsete elementide jaoks PLA-d ja see pehmeneb temperatuuril 60 °C. Maksimaalset temperatuuri saab tõsta, et olla lähemal kuumuskindlama materjali puhul mootorite piirangutele. Samuti kruvisid kruvid regulaarselt lahti, kui PLA pehmenes, vajades sageli hooldust.

Dünaamilist hoidemomendi muutujat saab rakendada lennuandmete põhjal adaptiivselt määramaks, kui palju hoidemomenti iga telg vajab.

Voolu mõõtva liikumiskontrolleri versiooni kasutamine võimaldab süsteemil mootori temperatuuri ise juhtida vastavalt parameetrile ja vooluhulgale. Süsteem eeldab praegu kindlat õhutemperatuuri, kui kasutaja pole seda üle kirjutanud.

LIST OF REFERENCES

[1] K. Watanabe, N. Moritoki and I. Nagai, "Attitude control of a camera mounted-type tethered quadrotor for infrastructure inspection," IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, Beijing, 2017, pp. 6252-6257.

[2] K. Watanabe, K. Kinoshita, I. Nagai and M. K. Habib, "Development of a camera-mounted tethered Quadrotor for inspecting infrastructures," *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, 2016, pp. 6128-6133.

[3] Stuart, M.B.; McGonigle, A.J.S.; Willmott, J.R. Hyperspectral Imaging in Environmental Monitoring: A Review of Recent Developments and Technological Advances in Compact Field Deployable Systems. *Sensors* 2019, *19*, 3071.

[4] R. J. Rajesh and C. M. Ananda, "PSO tuned PID controller for controlling camera position in UAV using 2-axis gimbal," 2015 International Conference on Power and Advanced Control Engineering (ICPACE), Bangalore, 2015, pp. 128-133.

[5] R. J. Rajesh and P. Kavitha, "Camera gimbal stabilization using conventional PID controller and evolutionary algorithms," 2015 International Conference on Computer, Communication and Control (IC4), Indore, 2015, pp. 1-6.

[6] R. Caponetto and M. G. Xibilia, "Fractional order PI control of a gimbal platform," 2017 European Conference on Circuit Theory and Design (ECCTD), Catania, 2017, pp. 1-4.

[7] V. Obiora and I. E. Achumba, "Adaptive control of Aerial vehicle gimbal using fuzzy-PID compensator," 2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON), Owerri, 2017, pp. 451-456.

[8] S. Paik, M. P. Nandakumar and S. Ashok, "Model development and adaptive control implementation of a 3-axis platform stabilization system," *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, Chennai, 2015, pp. 526-531.

[9] M. Serra and J. Quelas, "Robust fuzzy gain schedulling PID implementation for gimbal stabilization system," 2017 XVII Workshop on Information Processing and Control (RPIC), Mar del Plata, 2017, pp. 1-6.

[10] M. Abdo, A. R. Vali, A. R. Toloei and M. R. Arvan, "Modeling control and simulation of two axes gimbal seeker using fuzzy PID controller," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, 2014, pp. 1342-1347.

[11] G. Singh, M. P. Nandakumar and Ashok S, "Adaptive Fuzzy-PID and Neural network based object tracking using a 3-Axis platform," *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, Coimbatore, 2016, pp. 1012-1017.

[12] K. Bansal and L. Dewan, "Comparison of different controllers for line-of-sight stabilization of a gimbal system," 2014 Students Conference on Engineering and Systems, Allahabad, 2014, pp. 1-5.

[13] K. Seong, H. Kang, B. Yeo and H. Lee, "The Stabilization Loop Design for a Two-Axis Gimbal System Using LQG/LTR Controller," 2006 SICE-ICASE International Joint Conference, Busan, 2006, pp. 755-759.

[14] Ki-Jun Seong, Ho-Gyun Kang, Bo-Yeon Yeou and Ho-Pyeong Lee, "The digital anti-windup LQG/LTR controller design for a two axis gimbal system," 2007 International Conference on Control, Automation and Systems, Seoul, 2007, pp. 1717-1720.

[15] M. Khayatian and P. K. Aghaee, "Adaptive control of a two axis gimbal system using auxiliary error structure," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), Tehran, 2014, pp. 1366-1370.

[16] S. Aggarwal, A. Joshi and S. Kamal, "Line of sight stabilization of two-dimensional gimbal platform," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 2553-2558.

[17] C. Espinosa, K. Mayen, M. Lizarraga, S. S. H. Romero and R. Lozano, "Sliding mode line-of-sight stabilization of a two-axes gimbal system," 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Cancun, 2015, pp. 431-438.

[18] D. Nguyen and V. Nguyen, "Robust Control of Two-Axis Gimbal System," 2019 International Symposium on Electrical and Electronics Engineering (ISEE), Ho Chi Minh, Vietnam, 2019, pp. 177-182.

[19] C. E. Lin and S. Yang, "Camera gimbal tracking from UAV flight control," 2014 CACS International Automatic Control Conference (CACS 2014), Kaohsiung, 2014, pp. 319-322.

[20] Ranganathan, Jaganathan, and Semke, William H. "Three-Axis Gimbal Surveillance Algorithms for Use in Small UAS." *Proceedings of the ASME 2008 International Mechanical Engineering Congress and Exposition*. *Volume 1: Advances in Aerospace Technology*. Boston, Massachusetts, USA. October 31–November 6, 2008. pp. 31-40. ASME.

[21] F. Wang, D. Tian and Y. Wang, "High accuracy inertial stabilization via Kalman filter based disturbance observer," *2016 IEEE International Conference on Mechatronics and Automation*, Harbin, 2016, pp. 794-802.

[22] X. Yang, Z. Wang, X. Xi, G. Zhang, S. Feng and Y. Zhang, "Dynamic leveling method under independent work mode for airborne remote sensing stabilized platform," *2017 36th Chinese Control Conference (CCC)*, Dalian, 2017, pp. 5020-5024.

[23] Lumenera Corporation, "The Most Important Camera Parameters for Aerial Imaging," Lumenera White Paper Series, https://www.lumenera.com/media/wysiwyg/documents/casestudies/Aerial_Imaging_WP_Most_Important_Camera_Parameters_for_Aerial_Imaging.pdf.

[24] Harrington, Aaron and Christopher M. Kroninger. "Characterization of Small DC Brushed and Brushless Motors.", 2013.

[25] Basecam Electronics, "SimpleBGC 32bit 3-Axis Software User Manual," 4 Apr 2019.

[26] Michael C. McGoodwin and Dayong Gao, "Engineering Thermodynamics: Summary of topics from University of Washington course," 2016.

[27] R. K. Rajput, "Engineering Thermodynamics," Third Edition, SI Units Version, Patiala, Punjab, 2007.

[28] Tarik Al-Shemmeri, "Engineering Thermodynamics," Ventus Publishing ApS, ISBN 978-87-7681-670-4, 2010.

[29] "Introduction to Electronics Course Notes," Electrical Waveforms, accessed March 2020,https://courses.engr.illinois.edu/ece110/sp2020/content/courseNotes/files/?waveforms.

[30] "Calculating the area under a curve using Riemann sums," Math Insight, accessed March 2020,https://mathinsight.org/calculating_area_under_curve_riemann_sums

APPENDIX

Appendix 1 Microprocessor Power Control Algorithm

```
1. //be sure to increase serial buffer size
2. //in the ProgramFiles/Arduino/ directory, find HardwareSerial.h
3. //change SERIAL_RX_BUFFER_SIZE to 256
4. //change SERIAL_TX_BUFFER_SIZE to 256
5.
6. //the realtime data request code is based heavily on the examples provided at
   https://www.basecamelectronics.com/
7.
8. #include <inttypes.h>
9. #include <SBGC.h>
10. #include <SBGC_Arduino.h>
11.
12. #define SERIAL_SPEED 115200
13. #define serial Serial
14. #define REALTIME_DATA_REQUEST_INTERNAL_MS 10 //interval between realtime data req
    uests
15.
16. #define conversionAcc 512
17. #define conversionGyr 0.06103701895
18.
19. inline void process_cmd_realtime_data();
20. inline void process_in_queue();
21.
22. void request_adj_vars_val();
23.
24. static SBGC_cmd_realtime_data_t rt_data;
25. static uint16_t cur_time_ms,rt_req_last_time_ms,last_cmd_time_ms;
26.
27. const float SupplyVoltage = 13.0; //IMPORTANT!!! SET THIS TO MATCH THE POWER SOURC
    E!
28.
```

```
29. //MOTOR CONSTANTS                    {   roll,  pitch,  yaw}
30. const float kMotor[3]         = { 0.110,  0.110, 0.05};
31.
32. const float iMotorSlope_term1[3] = {-0.0134,-0.0134, -0.055};
33. const float iMotorSlope_term2[3] = { 0.0156, 0.0156, 0.6311};
34. const float iMotorSlope_term3[3] = { 0.0167, 0.0167, -2.497};
35. const float iMotorYint_base[3]  = {  0.1551, 0.1551, 0.1257};
36. const float iMotorYint_powr[3]  = {  3.4591, 3.4591, 3.4135};
37.
38. const float rMotorSlope_slope[3] = {-0.0002,-0.0002,     0};
39. const float rMotorSlope_yint[3]  = { 0.0062, 0.0062, 0.2435};
40. const float rMotorYint_slope[3]  = { 0.0255, 0.0255,     0};
41. const float rMotorYint_yint[3]   = { -0.105, -0.105, -0.4952};
42.
43. //P parameters
44. const float pParamLo[3]       = {    24,   180,  40};
45. const float pParamHi[3]        = {     4,   105,   2};
46.
47.
48. //POWER CONSTANTS
49. const float maxTemp = 60.0;   //max temperature, not to exceed, degrees C
50. const float errFullval = 1.0; //degrees of error resulting in a system response of 10
    0% available power
51.
52. //GLOBAL VARS
53. boolean varsInit;
54. float Vsupply;
55. float Tair;
56. float Tmotor[3];
57. uint8_t pwrParam[3];
58. int16_t imuAng[3]; //actual angle for each motor
59. int16_t tgtAng[3]; //target angle for each motor
60. float errAng[3]; //error between target and actual angles, calculated
61. uint16_t timePrev; //used to save the previous time for integration
```

```
62. float corP=10; //proportional correction value for angle error, pwrParm=corP*errAn
    g
63. float tMotorRate[3]; //current rate of motor temperature change, degrees C / secon
    d
64. uint8_t maxPwr[3]; //stores the current maximum power level allowed by the safety
    routine
65.
66. uint16_t prevOutput; //used to store the last output time, ms
67. boolean printInfo;
68. uint16_t loopCount; //used to limit serial print commands
69. uint8_t segDisp=1; //used to rotate though segments to display each loop
70.
71.
72. void setup(){
73.   serial.begin(SERIAL_SPEED);
74.   SBGC_Demo_setup(&serial);
75.   varsInit=false;
76.
77.   loopCount=0;
78.   //prevOutput=0;
79.   printInfo=false;
80. }
81.
82. void loop(){
83.   loopCount++;
84.
85.   cur_time_ms=millis();
86.   process_in_queue();
87.   //request realtime data
88.   if((cur_time_ms - rt_req_last_time_ms) > REALTIME_DATA_REQUEST_INTERNAL_MS){
89.     SerialCommand cmd;
90.     cmd.init(SBGC_CMD_REALTIME_DATA_4);
91.     sbgc_parser.send_cmd(cmd,0);
92.     rt_req_last_time_ms = cur_time_ms;
```

```
93.   }
94. }
95.
96. // Process incoming commands. Call it as frequently as possible, to prevent overr
    un of serial input buffer.
97. void process_in_queue() {
98.  while(sbgc_parser.read_cmd()) {
99.    SerialCommand &cmd = sbgc_parser.in_cmd;
100.          last_cmd_time_ms = cur_time_ms;
101.
102.          uint8_t error = 0;
103.          switch(cmd.id){
104.          // Receive realtime data
105.          case SBGC_CMD_REALTIME_DATA_3:
106.          case SBGC_CMD_REALTIME_DATA_4:
107.            error = SBGC_cmd_realtime_data_unpack(rt_data, cmd);
108.           if(!error){
109.             //check struct BGC_cmd_realtime_data_t within SBGC_cmd_helpers.h for
    member names (starts on line 133)
110.
111.             //store supply (battery) voltage
112.            if(float(rt_data.battery_voltage)/100>3){ //the onboard voltage measureme
    nt is very inaccurate, but can be used to tell if the system has power or not for
     testing
113.               Vsupply=SupplyVoltage;
114.            }else{Vsupply=0.0;}
115.
116.           //store ambient temperature
117.            Tair=float(rt_data.imu_temp_celcius+rt_data.frame_imu_temp_celcius)/
    2;
118.
119.            if (!varsInit){
120.            //Serial.print("Vsupply: ");Serial.println(Vsupply);
121.             for(uint8_t x=0;x<3;x++){
122.             maxPwr[x]=255;
```

```
123.                 Tmotor[x]=Tair; //initialize motor temp, assume same as Tair
124.                pwrParam[x]=150; //initialize initial power parameter
125.                }
126.
127.                timePrev=0; //init for integration
128.               varsInit=true;
129.               }
130.
131.               //store parameters for each motor
132.              for(uint8_t x=0;x<3;x++){
133.                imuAng[x]   = rt_data.imu_angle[x];
134.               tgtAng[x]   = rt_data.target_angle[x];
135.                errAng[x]   = (float(tgtAng[x])-
    float(imuAng[x]))/45.5111; //16384/360
136.              }
137.
138.              Serial.println("");
139.               Serial.print("t:");Serial.print(millis()/1000.0);Serial.print(",");
140.
141.               uint16_t timeCur=micros();
142.              float RMS;
143.               float Imotor;
144.              float TmotorInc;
145.               float TmotorDec;
146.              for(int x=0;x<3;x++){ //for each motor, x
147.                RMS=calc_RMS(pwrParam[x]);
148.               Imotor=calc_I(x,RMS);
149.                TmotorInc=calc_rateTempInc(x,RMS,Imotor);                    //per
    minute
150.               TmotorDec=calc_rateTempDec(x);                    //per minute
151.                tMotorRate[x]=(TmotorInc-
    TmotorDec)/60;                    //per second
152.               Tmotor[x]=Tmotor[x]+tMotorRate[x]*float(timeCur-
    timePrev)/1000000; //C + C/s * micros * (s/1000micros) = C
```

```
153.
154.              Serial.print("#:");Serial.print(x);Serial.print(",");
155.                Serial.print("e:");Serial.print(errAng[x]*100);Serial.print(",");
156.              Serial.print("I:");Serial.print(Imotor);Serial.print(",");
157.                Serial.print("Tm:");Serial.print(Tmotor[x]);Serial.print(",");
158.              Serial.print("TI:");Serial.print(TmotorInc);Serial.print(",");
159.                Serial.print("TD:");Serial.print(TmotorDec);Serial.print(",");
160.             }
161.
162.             timePrev=timeCur;
163.
164.             if(Vsupply>0){adjust_power();}
165.
166.             printInfo=false;
167.             } else {
168.             sbgc_parser.onParseError(error);
169.             }
170.            break;
171.           }
172.          }
173.         }
174.
175.         void adjust_power( ){
176.          //uint8_t numVars=3;
177.          //this command is defined as a struct in SBGC_cmd_helpers.h
178.          SBGC_cmd_set_adj_vars_var_t a[6]; //init struct
179.          //possible ids are defined on p57, vals 0-255
180.          a[0].id = ADJ_VAR_POWER_ROLL; //see SBGC_adj_vars.h for list
181.          a[1].id = ADJ_VAR_POWER_PITCH;
182.          a[2].id = ADJ_VAR_POWER_YAW;
183.          a[3].id = ADJ_VAR_P_ROLL;
184.          a[4].id = ADJ_VAR_P_PITCH;
185.          a[5].id = ADJ_VAR_P_YAW;
```

```
186.
187.        for(int x=0;x<3;x++){
188.         float e=abs(errAng[x]);
189.         while(e>180){e=abs(e-360);}
190.
191.         //stepped response mode
192.        uint16_t responseRange=int(e*10.0/errFullval);
193.         if(responseRange<1){responseRange=1;}
194.        else if(responseRange>10){responseRange=10;}
195.         pwrParam[x]=25*responseRange;
196.        Serial.print("#:");Serial.print(x);Serial.print(",");
197.         Serial.print("rrange:");Serial.print(responseRange);Serial.print(",");
198.
199.         //limit power output based on estimated motor temperature
200.        float Tsafety=maxTemp-(30*tMotorRate[x]);
201.
202.        if(Tmotor[x]>maxTemp){//over max temp!
203.          maxPwr[x]=1;
204.        }else if(Tmotor[x]>Tsafety){//over safety temp
205.          if(maxPwr[x]>1){maxPwr[x]=maxPwr[x]-1;}
206.        }else{//temp is fine
207.          if(maxPwr[x]<255){maxPwr[x]=maxPwr[x]+1;}
208.        }
209.
210.        if(pwrParam[x]>maxPwr[x]){pwrParam[x]=maxPwr[x];}
211.         Serial.print("pP:");Serial.print(pwrParam[x]);Serial.print(",");
212.        Serial.print("pM:");Serial.print(maxPwr[x]);Serial.print(",");
213.
214.        if(x<2){a[x].val=90;}
215.
216.        //update power parameters
217.         a[x].val = pwrParam[x]; //value of 0-255
218.        a[x+3].val = calc_pParam(pwrParam[x],x); //value of 0-255
```

```
219.          }
220.          SBGC_cmd_set_adj_vars_send(a,6,sbgc_parser);
221.        }
222.
223.        float calc_pParam(uint8_t powParam,uint8_t motorNo){
224.          return uint8_t(pParamLo[motorNo]-
    (float(powParam)/255*(pParamLo[motorNo]-pParamHi[motorNo])));
225.        }
226.
227.        float calc_RMS(uint8_t pwrPar){
228.          float d=float(pwrPar)/255.0; //duty cycle
229.          return Vsupply*pow(d,0.5);
230.        }
231.
232.        float calc_I(uint8_t motorNo, float RMS){
233.          if(RMS>1){
234.            float motorTemp=Tmotor[motorNo];
235.            float slope=iMotorSlope_term1[motorNo]*pow(RMS,2)+iMotorSlope_term2[
    motorNo]*RMS+iMotorSlope_term3[motorNo];
236.            float yint=iMotorYint_base[motorNo]*pow(RMS,iMotorYint_powr[motorNo]);
237.            float current=motorTemp*slope+yint;
238.            if(current>0){
239.              return current;
240.            }else{
241.              return 0;
242.            }
243.          }else{ //RMS is zero, so return zero current
244.            return 0;
245.          }
246.        }
247.
248.        float calc_rateTempInc(uint8_t motorNo, float RMS, float I){
249.          if(RMS>1){
```

```
250.          float rSlope=rMotorSlope_slope[motorNo]*RMS+rMotorSlope_yint[motorNo];

251.          float rYint=rMotorYint_slope[motorNo]*RMS+rMotorYint_yint[motorNo];
252.          float rCoeff=1.0/(rSlope*Tmotor[motorNo]+rYint);
253.          if(rCoeff>0){return rCoeff*(I/1000.0*RMS);}else{return 0;} //r*P
254.        }else{
255.          return 0;
256.        }
257.      }
258.
259.      float calc_rateTempDec(uint8_t motorNo){
260.        float kCoeff=kMotor[motorNo];
261.        float motorTemp=Tmotor[motorNo];
262.        return kCoeff*(motorTemp-Tair); //k(Tmotor - Tair)
263.      }
```