Development of a Low—Cost Experimental Quadcopter Testbed Using an Arduino Controller for Video Surveillance

A project present to
The Faculty of the Department of Aerospace Engineering
San Jose State University

in partial fulfillment of the requirements for the degree
Master of Science in Aerospace Engineering

By

Ankyd Ji

May 2015

approved by

Dr. Kamran Turkoglu
Faculty Advisor

The Designated Project Committee Approves the Project Titled

DEVELOPMENT OF A LOW-COST EXPERIMENTAL QUADCOPTER TESTBED USING AN ARDUINO CONTROLLER FOR VIDEO SURVEILLANCE

By

Ankyd Ji

APPROVED FOR THE DEPARTMENT OF AEROSPACE ENGINEERING

SAN JOSE STATE UNIVERSITY

MAY 2015

Dr. Kamran Turkoglu, Project Advisor
Department of Aerospace Engineering

18th May 2015
Date

Development of a Low-Cost Experimental Quadcopter Testbed using an Arduino controller for Video Surveillance

Ankyd Ji and Kamran Turkoglu [y]
San Jose State University, San Jose, CA 95112, USA

This paper outlines the process of assembling an autonomous quadcopter platform and designing control laws to stabilize it using an Arduino Mega. Quadcopter dynamics are explored through the equations of motion. Then a quadcopter is designed and assembled using o -the-shelf, low-cost products to carry a camera payload which is utilized for video surveillance missions. The unstable, non-linear quadcopter dynamics are stabilized using a generic PID controller. System identi cation of the quadcopter is accomplished through the use of sweep data and CIF ER[r] to obtain the dynamic model.

Nomenclature

b thrust factor of the propeller
d drag factor of the propeller
l distance from motor axis to the center I_{XX} moment of inertia about the x-axis
$I_{Y\ Y}$ moment of inertia about the y-axis
I_{ZZ} moment of inertia about the z-axis
$J_{T\,P}$ moment of inertia about the propeller axis
p roll rate
q pitch rate r yaw rate
U_1 vertical thrust factor
U_2 rolling torque factor
U_3 pitching torque factor
U_4 yawing torque factor
u velocity in the x-axis direction
v velocity in the y-axis direction w velocity in the z-axis direction total propellers' speed
1 front right propeller speed
2 rear right propeller speed
3 rear left propeller speed
4 front left propeller speed
roll angle
yaw angle
pitch angle

Graduate Student, Aerospace Engineering, ankyd.ji@sjsu.edu
[y] Assistant Professor, Department of Aerospace Engineering, kamran.turkoglu@sjsu.edu

I. Introduction

Quadcopters are small rotary craft that can be used in various environments, where they are able to maintain hover capabilities like a conventional helicopter, but are mechanically simpler and can achieve higher maneuverability. They use 4 xed pitch propellers to control lift and a combination of propeller torques to control roll, pitch, and yaw. Early designs had poor performance due to very high pilot workload. Current day control techniques and small sensors have increased the popularity of the quadcopter as an autonomous Unmanned Aerial Vehicle (UAV) platform.

The quadcopter is initially an unstable and underactuated plant with highly coupled and non-linear dynamics. These features make it an attractive experimental set-up and a system for controller design methodologies. There are 6 degrees of freedom to be controlled by 4 motor inputs and modeling the dynamics of the quadcopter is essential to understand its performance. For better understanding of plant behaviour, linearization of the nonlinear quadcopter model through analytical equations[1, 2] or system identi cation[3, 4] becomes essential for controller design and analysis.

In existing literature, there are many valuable works conducted on quadcopter analysis, and numerous practical applications of quadcopters ranging from disaster zone surveilence to photography, and so on. In this paper, we aim to provide a genuine approach in build and design of a prototype quadcopter purely based on low-cost, o -the-shelf products and Arduino controllers. Another unique aspect of this study is we provide a modi ed architecture of Arduino Mega software code, and through several modi cations, we make it possible to implement many more advanced control methodologies on Arduino Mega controllers (such as adaptive, robust, optimal, sliding mode and many more). Even though existing PID based controllers on Arduino Mega boards work just ne, this removes the restriction of the code, and makes it possible to use this prototype test-bed as a research platform for the demonstration of any desired control algorithm.

As it is well known from literature, due to unstable nature of quad-copter dynamics, stabilization of the quadcopter requires moving or adding stable poles in the s-plane.[5] Popular control techniques include Proportional Integral Derivative (PID) controllers[5, 6] or Linear Quadratic Regulator (LQR) controllers.[7, 8] Once a stable plant is obtained, a linearized quadcopter model is used to design desired control (gains) which are then applied to the actual plant dynamics. Generally speaking, the nominal position for a quadcopter is in hover mode, where the deviation from hover is calculated using Inertial Measurement Units (IMU) and fed back into the system for control.

Modern quadcopters are heavily dependent on sensor measurements but have become more popular than ever due to the improvements in IMU and Global Positioning Systems (GPS) and their great potential for control applications. Due to heavy dependency on GPS and IMU measurements, complementary and Kalman ltering techniques are heavily used to adjust the GPS and IMU measurements to provide consistent values for the controller.[11] In some cases, depending on the assigned mission requirements, it is also possible to complement the sensor measurements with vision based tracking using live video during ight.[17, 18] However, robust controllers are required to stabilize the quadcopter from disturbances in an outdoor environment[12] or in fast moving references.[13]

In the light of these cases, this paper discusses the integration, development, build and analysis of a quadcopter platform that is aimed to operate autonomously on pre-programmed missions and/or survey disaster zones with an onboard camera. The unstable, non-linear quadcopter dynamics are stabilized using a conventional PID controller. Included is the extraction of the linear model using analytical equations and system identi cation for comparison.

II. Modeling

A. Quadcopter Dynamics

A quadcopter model consists of a cross beam structure with 4 motors on each end and collection of sets of electronic equipment. The 4 motor torques are the only inputs for a 6 degrees of freedom (DOF) system which de ne the quadcopter as an underactuated system. Without a controller to compensate for underactuation, there are 2 states that cannot be directly commanded. This, eventually, will cause a drift to undesired values over time. The motors are stationary and do not have any mechanical linkages to change the blade pitch. In that sense, the quadcopter utilizes a combination of the four motor torques to control all the states. The testbed is designed using the X-formation shown in Figure 1. One pair of propellers ($_1$ and $_3$) rotate clockwise (CW) while the other pair of propellers ($_2$ and $_4$) rotate counter-clockwise (CCW).

To command throttle, all four propellers must rotate at the same speed which provides a vertical force in the z-axis. If each propeller provides a quarter of the weight in thrust, the quadcopter will hover. Rolling motion is generated by either increasing or decreasing the torque in pair of motors on the left side ($_3$ and $_4$) while applying an opposite increase or decrease to the right pair of motors ($_1$ and $_2$). This produces a torque in the x-axis which creates the rolling motion. Pitching motion is generated by increasing/decreasing the front motors ($_1$ and $_4$) while applying the opposite action (increase/decrease) to the rear motors ($_2$ and $_3$). This combination produces a torque in the y-axis which creates a pitching motion. Yawing motion is generated by increasing/decreasing the CW motor pair while applying the opposite action (increase/decrease) to the CCW motor pair. This combination produces a torque in the z-axis which creates a yawing motion.

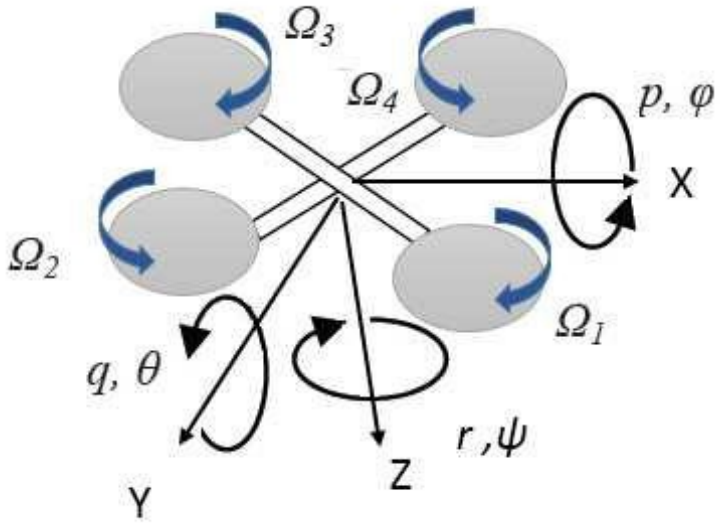

Figure 1. Quadcopter model schematic with coordinate system.

B. Equations of Motion

The quadcopters' non-linear, coupled equations of motion (EoMs) have been analyzed extensively in literature and are summarized below for convenience.[1, 2] These EoMs are derived by applying Newton's 2nd Law to the quadcopter body. Some of the basic assumptions include that i) the quadcopter is a rigid body and ii) it is symmetrical along the x and y axes.

$$
\begin{aligned}
&\underline{u} = (vr\ wq) + gs\ w = \\
&(\underline{w}p\ ur)\ gc\ s \\
&\bar{w} = (uq\ \ vp)\ \ gc\ s\ +\ \frac{U_1}{m} \\
&\underline{p} = \frac{I_{YY}\ I_{ZZ}}{I_{ZZ}}qr\ \ \frac{J_{TP}}{I_{XX}}q\ +\ \frac{U_2}{I_{XX}} \\
&\underline{q} = \frac{I_{ZZ}\ I_{XX}}{I_{YY}}pr\ \ \frac{J_{TP}}{I_{YY}}p\ +\ \frac{U_3}{I_{YY}} \\
&\underline{p} = \frac{I_{XX}\ I_{YY}}{I_{ZZ}}pq\ \ \frac{J_{TP}}{I_{ZZ}}
\end{aligned}
\qquad (1)
$$

The outputs of the EoMs are translational velocities u, v, w; rotational velocities p, q, r; positions x, y, z; and the attitude angles , , and . These outputs are calculated by integrating the EoMs, given in Eq.(1). The inputs of the EoMs are the propeller speed inputs where U_1; U_2; U_3; U_4 are associated with throttle, roll, pitch and yaw respectively. Here, is the sum of the propellers rotational speed. These inputs are functions of the propellers rotational speed $_{1\ 4}$, where lift and drag factors of the propeller blade (b and d respectively) and length l. Here, lift and drag factors of the propeller blade (b and d respectively) are

calculated from the Blade Element Theory[1] . From the quadcopter dynamics discussed in Section A, the inputs can be expressed as shown in Eq.(2).

$$
\begin{aligned}
U_1 &= b(\ ^2_1 + \ ^2_2 + \ ^2_3 + \ ^2_4) \\
U_2 &= lb(\ \ ^2_1 \quad ^2_2 + \ ^2_3 + \ ^2_4) \\
U_3 &= lb(\ _1 \quad _2 \quad _3 + \ _4) \\
U_4 &= d(\ ^2_1 + \ ^2_2 ^2_3 + \ ^2_4) \\
&= b(\ _{12} + \ _3 \quad _4)
\end{aligned}
\tag{2}
$$

C. Linearization of Non-Linear EoMs

A linearized model of the EoMs are desired to analyze quadcopter behavior at an operating point where all states are e ectively zero. For the quadcopter, the operating point is selected as the hover condition where the motors provide enough thrust to counteract the force of gravity. Matlab is used to linearize the model dynamics,

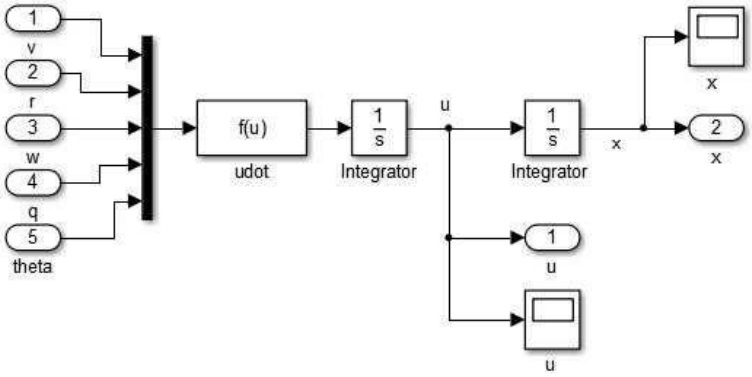

Figure 2. Simulink model of u equation, where the remaining equations follows the same .

where Figure 2 demonstrates one of the 6 non-linear EoMs that are modeled in Simulink. The function block contains the u equation. This block is integrated twice to obtain the velocity u and position x. The control system toolbox embedded linmod function is utilized to extract the linear state space model (SSM). The same procedure is repeated for the remaining equations.

III. Quadcopter Platform Integration and Development

The quadcopter testbed is built and assembled from scratch, while the programming is coded using an Arduino Mega control board. A standard thrust to weight ratio of 2 and above is kept for maneuverability of the quadcopter. This directly results in a desire for small, lightweight components. Utilization of low cost components is one important gure of merit but the reliability of each part is also taken into account extensively. The total cost of the quadcopter is around $388 (without a GoPro camera) and a breakdown of the o -the-shelf components can be seen in Table 1. The testbed (in its current condition) can be seen in Figure 3.

A. Hardware and Components

To keep the thrust to weight ratio above 2, each motor and propeller combination must provide at least one half of the total weight in thrust. In order to accomplish this, NTM28-30 800kv BLDC motors were selected to provide thrust using 12x6 carbon ber propellers. The low kv rating provides high torque at lower rotational speeds to push the large props. Initial testing of these motors, using a custom thrust stand, show that each motor and propeller combination provide 900 grams of thrust at around 150 watts. Further thrust testing at maximum output is necessary since these motors are rated up to 300 watts. Afro electronic

Table 1. Quadcopter components and cost.

Item List	Price ($)
Frame	54
Arduino	38
IMU	10
GPS	38
ESC	13
Motor(4 motors)	15
Propeller(2 pairs)	9
Xbee	20
Power Distribution Board	4
6 Channel Receiver	49
Landing Gear	24
Battery	21

Figure 3. Assembled quadcopter testbed.

speed controllers (ESCs) are used to control the motor and propeller rotational speed. The ESCs are ashed with SimonK rmware to obtain higher update rates to change the rotational speed if necessary.

The onboard IMU is a Geeetech 10DOF board. This IMU includes a 3 axis ADXL345 accelerometer with 4 mg/LSB resolution, a 3 axis L3G4200D gyro with 2000 degrees/s range, a 3 axis HMC5883L magnetometer with an accuracy of 1-2 degrees, and a BMP085 barometer which can achieve 0.03hPa accuracy. The EM-406a Global Positioning System (GPS) is included to complement the IMU measurements with positional data.

A carbon ber frame is incorporated, which provides higher strength and about 130 grams of weight savings over other commercially available quadcopter frames. Some of the quadcopter testbed parameters are listed in Table 2 for convenience.

Table 2. Calculated quadcopter platform parameters.

Parameter	Value	Units
mass	1800	grams
I_{XX}	$7.06*10^{3}$	kgm^2
I_{YY}	$7.06*10^{3}$	kgm^2
I_{ZZ}	$7.865*10^{3}$	kgm^2
J_{TP}	$14.2*10^{4}$	kgm^2
b	$4.5*10^{4}$	meters
d	$1.8*10^{5}$	meters
l	8.25	meters

B. Software Development and Novel Arduino Code Modi cation

There are many di erent quadcopter controllers that use Arduino based software (Ardupilot, MultiWii, etc..), but one major trait in those controllers is that they are di cult to modify and lack modularity. This becomes problematic when researchers want to use such test-beds for the application of more advanced (and sophisticated) control methodologies, such as adaptive control, mu-synthesis, and so on. The reason why many of these systems (like Ardupilot, MultiWii, etc..) are di cult to modify is that they are very limited in terms of their own, specialized libraries and speci cally named functions/values. Currently, there is also not enough documentation provided to explain each software, due to its open-source nature. A speci c example can be found in Ardupilot whose stabilization algorithm uses a PID library which is calculated using values that are found in a di erent library making it a very round about process.

In this research, for the sake of modularity and applicability, the Arduino code was written from scratch. Emphasis was put on simplicity by providing important values like roll and pitch angles in the main script and by using better notation scheme that relates a value to what it actually means. This made it much more simple to modify and to apply di erent stability and control algorithms as well as being applied to hands on experience for classroom/teaching usage.

Most arduino codes are based on using libraries to provide utility functions that can be used in the main sketch. A library is essentially used to provide the code with a complete procedure without clumping up the main sketch in our novel code modi cation. The fundamental calculations of our novel modi ed code are written in libraries and rely on di erent libraries to provide quadcopter state information. This makes it very simple to apply a formula in the main script just by naming the function and providing the data that is used in the calculation. In our novel code modi cation, IMU data is read through a custom library while the GPS makes use of the open source TinyGPS library. GPS and IMU data are all read through the library and provide measurements for the main Arduino code to calculate PID values for stability. Since the inputs for calculations are all from libraries, it is very simple to apply (if necessary) other sensor inputs to provide the same information, which makes the Arduino code modular. Currently the arduino code uses a modi ed version of the open source PID library by Beauregard,[14] but can be easily swapped out for other controller schemes (such as LQR, Adaptive control, Robust control ... etc.) by creating new libraries with the corresponding formulas/calculations and including them in the main script.

Some of the known limitations on our modi ed arduino system include sampling time, telemetry, and accurate measurement data. Sampling time is relatively di cult to keep consistent because it is a function

of how many calculations must be completed. If complicated controllers or extra modules are added to the system, more computing power is used, which will lower the sampling time to a point where the quadcopter's stability is endangered. The current code runs at 100[Hz] which leaves enough safety factor for calculations, and therefore the stability. Previous hardware testing had showed unstable behavior below 20[Hz].

Since the telemetry data relies on wireless XBee communication, there are many times when packets will be lost in ight testing because of signal inteference in the atmosphere that cannot be controlled. The loss of packets means not all the data will be continuous and could cause problems if the data is lost during an experiment. One option to avoid such problem was applied by using an SD card to record the data on-board from the Arduino. This resulted in a large drop in sampling time to around 20[Hz] because of bu ering issues between the microSD card and Arduino. This is not desirable and as explained earlier, 20[Hz] will cause the quadcopter to become unstable. Accuracy in the measurement data refers to the GPS, which relies on multiple satellite links. This becomes an issue if the satellite links are lost during ight testing which results in no translational data for analysis. The loss of GPS signal becomes a very severe issue because trajectory pathing cannot be tracked, which could jeopardize an autonomous mission.

With all these in mind, with the novel arduino code modi cation, in our research we provide the exibility and modularity of not only improved schematics, but also an experimental research/teaching platform which is not restricted only to PID based conventional controllers, and can be extended to more advanced control methodologies such as adaptive, robust, real-time, non-linear and/or optimal control.

C. Controller Implementation

In this study, the Arduino Mega 2560 is used as the microcontroller due to its inexpensive nature and relatively powerful characteristics. It is simple to program and its open source nature, with extensive documentation, provides tremendous bene ts. The Arduino Mega operates the ATmega2560 chip microcontroller with an internal bootloader preinstalled. The Arduino Mega contains 54 digital input/output pins that can be used to control the ESCs and receiver, an I2C bus, and three pairs of TX/RX pins to transmit serial data. The Spektrum AR6255 6 channel remote control (RC) receiver is used in combination with a DX-6 RC controller. The receiver is connected through the digital input and output lines to receive motor commands from the DX-6 for initial testing and debugging.

The Mega provides six 8-bit pulse width modulation (PWM) ports. Six of these PWM ports are used as RC receiver input signals from the RC controller. The output signals are sent from the ESC which will then control the motor speed. The arduino command attachinterrupt will be used to trigger interrupts on the targeted pins to read the PWM values. The PWM values are then mapped to ESC values after calibrating the ESCs using a standard servo controller.

The IMU is connected to the Arduino Mega through the I2C ports. Each module on the IMU has a unique I2C address to separate the measurement data. GPS, telemetry, and video footage will provide data to be transmitted over the serial TX/RX pins. The Arduino will process and calculate the measurement data and transmit all of it over the wireless XBee that is used as telemetry. A camera is implemented underneath which can be used to stream video wirelessly to a laptop or phone. A summary of the arduino pins used for each component is shown in the Table 3.

Table 3. Arduino pin con guration.

Component	Type of Pin	Pin value(s)
Electronic speed controllers	Digital I/O	3, 5, 6, 7
Spektrum 6 channel receiver	Analog I/O	A8-A13
Inertia measurement unit	I2C	SCL, SDA
EM406a GPS	TX/RX	16, 17
XBee telemetry	TX/RX	18, 19
Camera	TX/RX	14, 15

IV. State Space Model and Stability Analysis

A state space matrix representation of the quadcopter can be found using equations discussed in Section II-B. This model will be used to nd initial PID values to stabilize the quadcopter platform. Since the model is a rough estimate these PID values, which only are starting estimates, are further tuned to accomodate pilot's preference and mission's requirements.

A. State Space model

The linearized state space model (SSM) is derived using the Simulink block model discussed in Section II-C, where the input values for the quadcopter were provided in Table 2. The nal state space matrices are shown in Eq.'s (3)-(4).

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.81 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.81 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.63 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.63 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.63 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.7143 & 0 & 0 & 0 \\ 0 & 12.3457 & 0 & 0 \\ 0 & 0 & 12.3457 & 0 \\ 0 & 0 & 0 & 0.7143 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{4}$$

B. Open-loop Analysis

Analyzing the eigenvalues of these matrices from Section IV-A indicates an initially unstable system, as expected. Thus, any small disturbance will cause the quadcopter to become unstable without a controller to stabilize it. However, the SSM is fully controllable due to the di erent combination of propeller speed inputs. Analysis of the state space model shows that disturbances in roll and pitch angles result in very large displacements. Roll and pitch angles increase over 20 degrees in 10 seconds because the roll and pitch rates do not stabilize after an impulse. The yaw angle and rate stay small in comparison to pitch and roll angles. The translational modes of the SSM show some very odd behavior because they exclude aerodynamic forces. The result of the SSM analysis show that the overall system is unstable and translation modes cannot be controlled directly, which brings the need for closed-loop system analysis.

C. Closed-loop Analysis

An initial Proportional Integral Derivative (PID) inner-loop controller is designed to stabilize the system where the PID gain value characteristics inherits the conventional form as shown in Eq.(5)[5, 6] .

$$PID(s) = K_p + \frac{K_i}{s} + T_d s \tag{5}$$

For stability, a cascaded control architecture will be implemented which uses multiple inner loops and multiple input signals. The innermost PID will use rate values and the outer loop will stabilize any angular disturbances. The PID values chosen to stabilize the open loop dynamics are shown in Table 4.

Table 4. Initial PID values to stabilize roll and pitch.

	Roll/Pitch Angle Gains	Roll/Pitch Rate Gains	Yaw Rate Gains
K_p	3.604	0.2209	0.1141
K_i	0	0	0.6340
T_d	0	0.014	0

The goal is to use these values as a starting point to achieve a stabilized closed-loop system. The PID gains are further tuned to the pilot's preference to perform sweeping maneuvers for system identi cation. Although the quadcopters EoMs are coupled, PID controllers for the other states (such as translation) are not necessary because the disturbances are relatively small compared to the roll and pitch angles. The closed loop response of the quadcopter system with the PID values from Table 4 are shown in Figure 4.

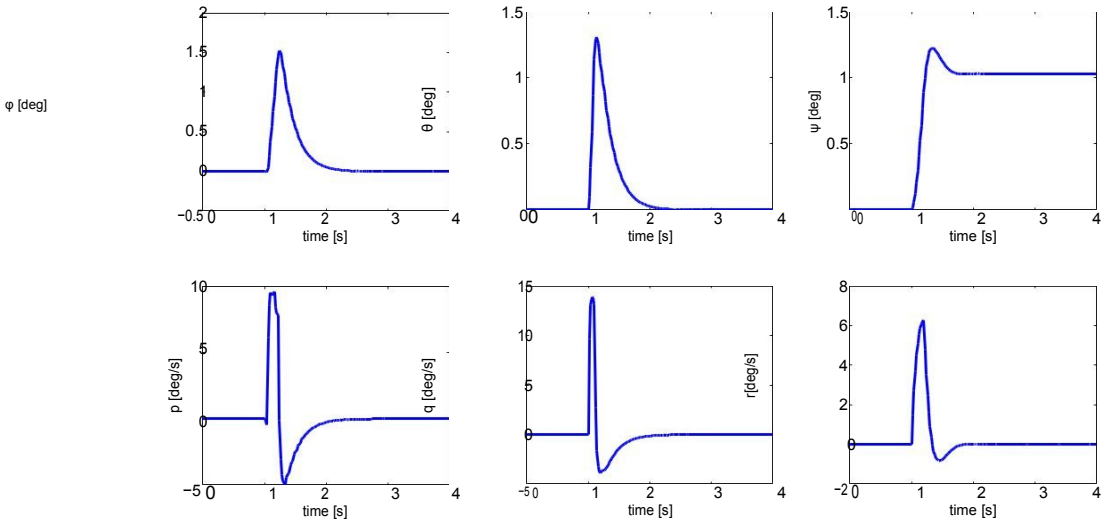

Figure 4. Impulse response of closed loop PID quadcopter dynamics.

It is clear from Figure 4 that roll, pitch, and yaw responses are stable and return back to the trim value at zero within a second. Since there is no closed loop control on position, translational motion(x,y,z) of the quadcopter will have small steady state errors, which are relatively small and therefore negligable.

V. System Identi cation

From literature, it is well known that system identi cation can be used to develop a linearized SSM using experimental ight data[3, 4, 15] . This involves mapping a known input signal to the ight data response in the frequency domain to obtain the transfer function of the corresponding state. Once this is achieved for all 6 states of the system, the model structure can be built from the estimated transfer functions for further

theoretical/simulational analysis. The goal is to compare and verify the system identi cation model with actual quadcopter experimental data.

A. Data Collection

The identi cation process starts with retrieving frequency domain sweep data via a chirp signal. A chirp signal is essentially a sinusodial function which starts at a low frequency and slowly increases to higher frequencies to cover (and excite) all the di erent modes of the system. The sweep, in our case, is implemented manually through a pilot input and is applied to roll, pitch, and yaw states with the output data for angles and rates being logged for analysis in CIF ER^r.[16] A sample sweep in roll from the pilot input is shown in Figure 5. The magnitude and frequency of the sweep command vary since a pilot input is not perfect, but it still provides valuable data for analysis.

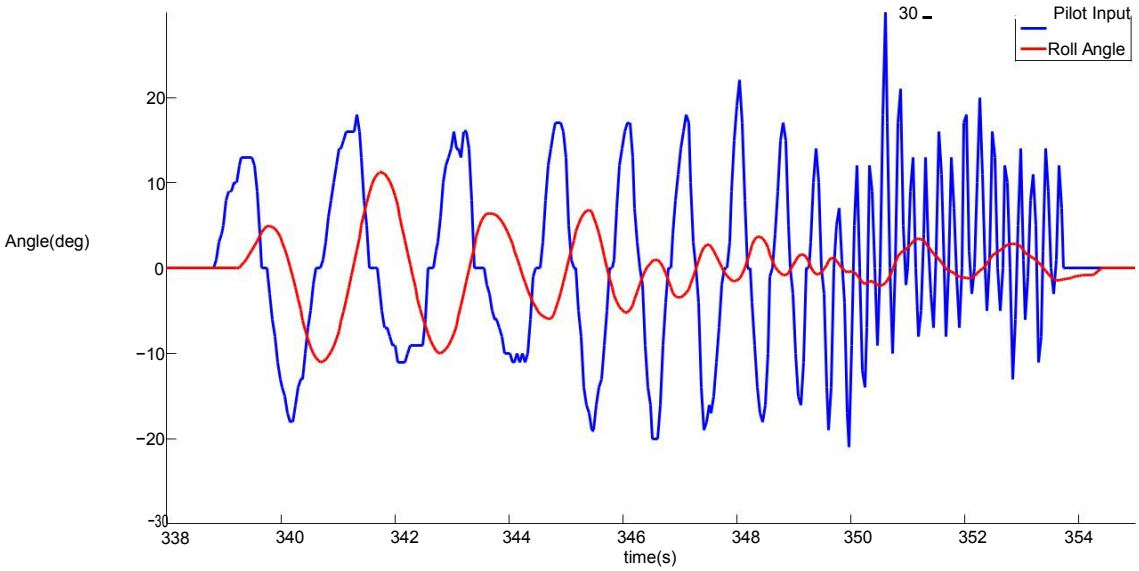

Figure 5. Roll Coherence in CIF ER^r of a set of sweep data.

B. SISO Identi cation Analysis

For system identi cation analysis, the student version of CIF ER^r is utilized.[16] It is designed to take test data, extract transfer functions, and state space models by analyzing the data in the frequency domain. In this study, only angles were able to be analyzed because the translational measurements were not precise enough for analysis. This is a topic of an ongoing research, and results will be reported in future studies. A complete SSM could not be extracted due to a lack of consistent measurements. However, the SISO transfer functions that were extracted have a fairly accurate t. The resulting identi ed SISO transfer functions for roll and pitch are shown in Eq.'s (6) - (7).

$$TF_{roll} = \frac{0{:}89s + 1{:}40}{s^2 + 1{:}08s + 5{:}02} e^{0{:}1937s} \tag{6}$$

$$TF_{pitch} = \frac{1{:}26s + 1{:}24}{s^2 + 1{:}9s + 9{:}18} e^{0{:}1726s} \tag{7}$$

It can be seen from Figure 6 that the coherence at lower frequencies are fairy adequate but at higher frequencies, the sweep data is not consistent past 10[rad=s]. Some of the sweep data would show negative coherence at the higher frequencies which can be interpreted to mean that the high frequency sweep data was not satisfactory. Due to that reason, yaw transfer functions could not be extracted because of roll coupling into the yaw sweep from motor saturation.

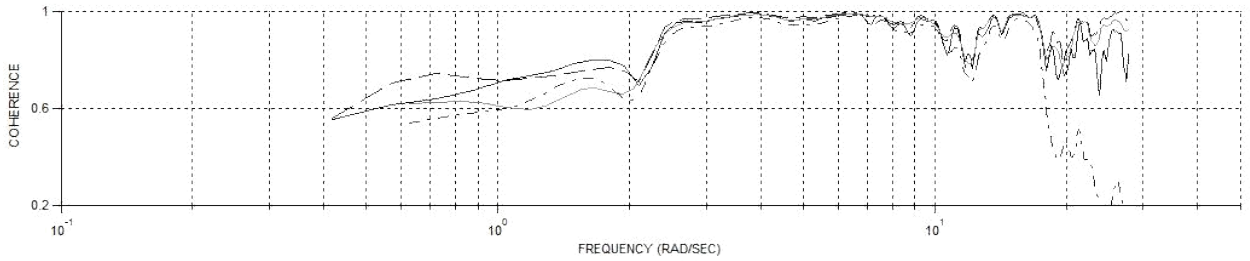

Figure 6. Roll Coherence in CIF ER^r for the set of sweep data.

C. Veri cation

Following to identi cation results, derived dynamics are compared with the actual ight data to verify the accuracy of the model dynamics. The pilot inputs are formed as a doublet signal and the result is compared with a doublet of the same magnitude using the identi ed transfer functions. The result of a pitch doublet is shown in Figure 7 using the transfer function from Eq.(7). It can be seen that the magnitude and the time delay of the transfer function matches up with the actual system very well which validates the transfer functions obtained from CIF ER^r.[16]

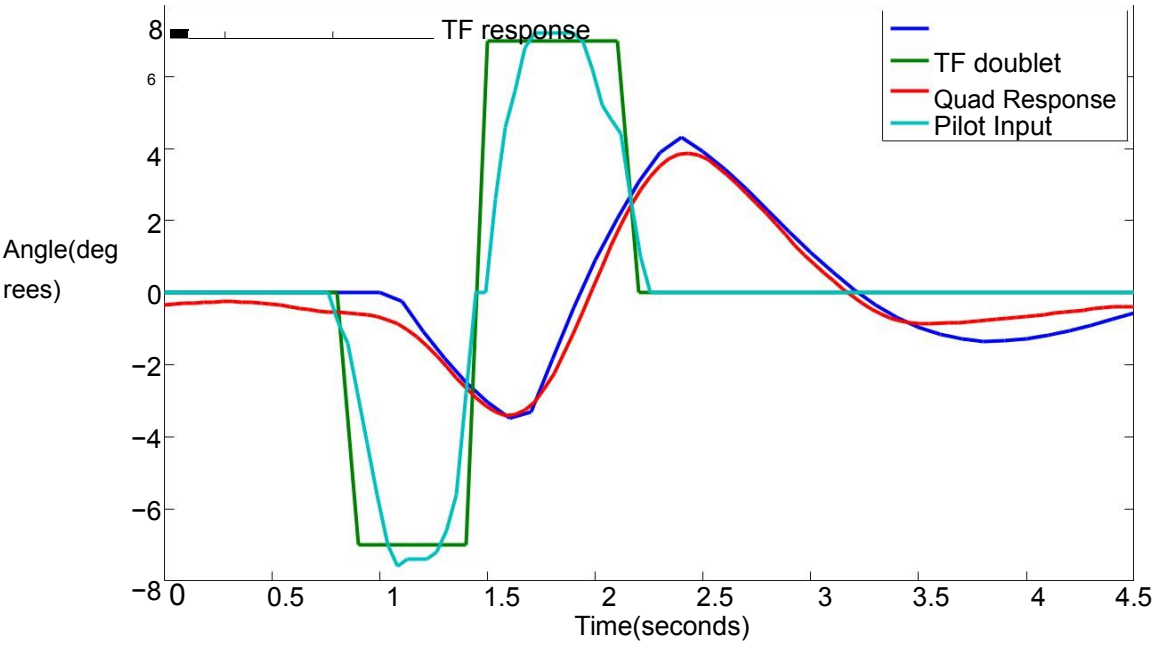

Figure 7. Pitch doublet response of quadcopter compared with CIF ER^r transfer function.

VI. Conclusion

In this study, we provided a process of assembling an autonomous, low-cost, o -the-shelf product based quadcopter platform from scratch. We designed control laws to stabilize it using an Arduino Mega. The EoMs for a quadcopter have been studied and a linear state space model was extracted for analysis. The quadcopter testbed was assembled and programmed using the Arduino Mega to be own using a DX-6 controller. One novelty we presented in this paper is that we provide a modi ed, novel version of Arduino code which is not restricted to PID controllers only, and can be extended to more advanced control methodologies due to its

modular and highly exible structure. For measurements, the IMU and GPS have been calibrated with a complementary lter. A closed loop PID is designed in Simulink using the state space model to stabilize non-linear plant dynamics. The closed loop controller has also been coded into the Arduino Mega. With this design, ight data could be sent back to a laptop through the Xbee receiver. Transfer functions for roll and pitch were identi ed and veri ed using doublet data from real ight.

In future studies, because the sweep data was lacking coherence in the higher frequencies, an automated chirp signal is aimed to be used instead of manual pilot inputs, for system identi cation.

References

[1]Bresciani, T., "Modeling, Identi cation and Control of a Quadrotor Helicopter," MS Thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2008.

[2]Balas, C. "Modelling and Linear Control of a Quadrotor," MS Thesis, Cran eld University, Bedford, UK, 2007.

[3]Lee, G., Jeong, D. Y., Khoi, N. D., and Kang, T., "Attitude Control System for a Quadrotor Flying Robot," 8th Interna-tional Conference on Ubiquitous Robots and Ambient Intelligence, URAI, Incheon, Korea, 2011, pp. 74-78.

[4]Miller, D. S., "Open Loop System Identi cation of a Micro Quadrotor Helicopter from Closed Loop Data," MS Thesis, Aerospace Engineering Dept., Maryland Univ., College Park, MD, 2011.

[5]Sa, I., and Corke, P., "System Identi cation, Estimation and Control for a Cost E ective Open-Source Quadcopter," Internal Conference on Robotics and Automation, IEEE, Saint Paul, Minnesota, 2012, pp. 2202-2209.

[6]DiCesare, A., Gustafson, K., and Lindenfeler, P., "Design Optimization of a Quad-Rotor Capable of Autonomous Flight" BS Report, Aerospace and Mechanical Dept., Worchester Polytechnic Institute, Worchester, MA.

[7]Andreas, R., "Dynamics Identi cation & Validation, and Position Control for a Quadrotor," MS Thesis, Autonomous Systems Lab, Swiss Federal Institute of Technology, Zurich, Switzerland, 2010.

[8]Valeria, E., Caldera, R., Lara, S., and Guichard, J., "LQR Control for a Quadrotor using Unit Quarternions: Modeling and Simulation," IEEE, Puebla, Mexico, 2013.

[9]Magnussen, O., and Sjonhaug, K. E., "Modeling, Design and Experimental Study for a Quadcopter System Construction," MS Thesis, Engineering Dept., Agder Univ., Kristiansand, Norway, 2011.

[10]Domingues, J. M. B., "Quadrotor prototype," MS Thesis, Mechanical Engineering Dept., Technical University of Lisbon, Lisbon, Portugal, 2009.

[11]Abas, N., Legowo, A., and Akmeliawati, R.,"Parameter Identi cation of an Autonomous Quadrotor," 4th International Conference on Mechatronics, IEEE, Kuala Lumpur, Malaysia, 2011.

[12]Siebert, S., and Teizer, J., "Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system," sl Automation in Construction, IEEE, Vol. 41, 2014.

[13]Sorensen, A. F., "Autonomous Control of a Miniature Quadrotor Following Fast Trajectories," Control Engineering Department, Aalborg University, Aalbord, Denmark, 2010.

[14]Beauregard, B., Arduino (open source) PID Library, available at: http://brettbeauregard.com/blog/tag/pid/.

[15]Tischler, M. B. and Remplr, R. K. , Aircraft and Rotorcraft System Identi cation, AIAA Education Series, 2012.

[16]CIFER, student version, available at http://uarc.ucsc.edu/ ight-control/cifer/.

[17]Saakes, D. Choudhary, V., Sakamoto D., Inami, M., and Igarashi, T., "A Teleoperating Interface for Ground Vehicles using Autonomous Flying Cameras," 23rd International Conference on Arti cial Reality and Telexistence, IEEE, 2013.

[18]Hurd, M. B., "Control of a Quadcopter Aerial Robot Using Optic Flow Sensing," MS Thesis, Mechanical Engineering Department, University of Reno, Reno, CA, 2013.

[19]Stanculeanu, I., and Borangiu, T., "Quadrotor Black-Box System Identi caion," World Academy of Science, Engineering and Technology, International Science Index, Vol. 5, 2011.

[20]Martinez, V. M., "Modeling of the Flight Dynamics of a Quadrotor Helicopter," MS Thesis, Aerospace Sciences Dept., Cran eld Univ., Bedford, United Kingdom, 2007.

[21]Parker, M., Robbiano, C., and Bottor , G., "Quadcopter," BS, Electrical and Computer Engineering Dept., Colorado State Univ., Fort Collins, CO, 2011.

[22]Schreier, M., and Darmstadt, T., "Modeling and Adaptive Control of a Quadrotor," /sl International Conference on Mechatronics and Automation, IEEE, Chengdu, China, 2012, pp. 383-390

[23]Schreurs, R. J. A., Weiland, S., Zhang, H. T. Q., Zhu, Y., and Xu, C., "Open Loop System Identi cation of a Quadrotor Helicopter System," 10th IEEE International Conference on Control and Automation, IEEE, Hangzhou, China, 2013, pp. 1702-1707.