Low-cost Image-assisted Inertial Navigation System for a Micro Air Vehicle

Adam Polak[1]
Arizona State University, Tempe, Arizona, 85281

The increasing civilian demand for autonomous aerial vehicle platforms in both hobby and professional markets has resulted in an abundance of inexpensive inertial navigation systems and hardware. Many of these systems lack full autonomy, relying on the pilot's guidance with the assistance of inertial sensors for guidance. Autonomous systems depend heavily on the use of a global positioning satellite receiver which can be inhibited by satellite signal strength, low update rates and poor positioning accuracy. For precise navigation of a micro air vehicle in locations where GPS signals are unobtainable, such as indoors or throughout a dense urban environment, additional sensors must complement the inertial sensors to provide improved navigation state estimations without the use of a GPS. By creating a system that allows for the rapid development of experimental guidance, navigation and control algorithms on versatile, low-cost development platforms, improved navigation systems may be tested with relative ease and at reduced cost. Incorporating a downward-facing camera with this system may also be utilized to further improve vehicle autonomy in denied-GPS environments.

Nomenclature

A	=	area
a	=	acceleration
BEC	=	battery elimination circuit
C_d	=	coefficient of drag
CAD	=	computer aided design
$CPPM$	=	combined pulse-position modulation
DOF	=	degrees of freedom
DCM	=	direction cosine matrix
$e_{m\%}$	=	motor efficiency
ESC	=	electronic speed controller
F	=	force
FDM	=	fused deposition modeling
g	=	gravitational acceleration
GPS	=	global positioning system
$GUIDE$	=	graphical user interface design environment
H	=	sensor dynamics
I	=	current
$\bar{I}$	=	identity matrix
IMU	=	inertial measurement unit
J	=	inertia
K	=	controller
k_d	=	derivative gain
k_e	=	electrical torque constant
k_f	=	force constant
k_i	=	integral gain
k_p	=	proportional gain
k_t	=	torque constant
L	=	inductance
LED	=	light-emitting diode
M	=	mass
m	=	moment
n	=	noise
$MEMS$	=	micro-electro-mechanical systems
NED	=	North-East-Down Coordinate System
ρ	=	density
pwm	=	pulse width modulation
PID	=	proportional-integral-derivative
R	=	resistance
r	=	radius
$RTOS$	=	real-time operating system
T	=	time
T_e	=	electrical constant
T_m	=	mechanical constant
$UART$	=	universal asynchronous receiver transmitter
USB	=	universal serial bus
V	=	volts
v	=	velocity
ω	=	angular velocity

I. Introduction

Unmanned aerial robotics systems have become widely used by the military as well as private corporations, research institutions and hobbyists. As with many technologies, the components are now smaller and less expensive to manufacture. These systems often utilize a suite of inertial navigation sensors to provide feedback to the embedded controller, effectively stabilizing the naturally unstable aircraft, and in some instances offering nearly complete autonomy. However, many autonomous systems rely on the use of a global positioning satellite receiver as a low cost means to effectively estimate the position of the vehicle when complemented by an inertial navigation system. The problem with GPS is that it is often dysfunctional indoors or within dense urban environments with large structures reaching towards the sky. In addition, the positioning accuracy of GPS is often quite poor for low-cost systems, commonly accurate to two

[1] Aerospace Engineering, apolak@asu.edu

meters under ideal circumstances. To combat this, computer vision systems offer a self-contained solution that allows for estimation of the vehicle's navigation states in denied-GPS environments with increased precision and accuracy.

Incorporating these vision systems onto an air vehicle can be quite challenging, from both a hardware and software perspective. Multiple image sensors may be used for improved three-dimensional positioning accuracy, but this comes at the expense of added weight and increased complexity. Image sensors may also need to be vibration isolated or mechanically stabilized. Significant computational power is required to process the information provided by these systems.

Many low-cost microcontrollers utilized for robotics have relied on variations of the widely used 8-bit Atmel-based microcontroller known as the Arduino. The Arduino has a reputation of being the go-to electronic prototyping tool for amateur robotics projects; however, the increasing complexities of aerial robotic systems demand a more powerful microcontroller with integrated sensors and control peripherals. The PX4 autopilot system supported by the PIXHAWK Project of the Computer Vision and Geometry Lab of ETH Zurich is offered as an all-in-one flight control solution for aerial roboticists. This flight controller integrates a powerful 32-bit ARM processor with an array of sensors, including a nine degree of freedom inertial measurement unit as well as interconnects for servos, electronic speed controllers and communication with GPS modules and computer vision systems.

Despite providing an excellent suite of low-cost sensors with an open-source repository of software, testing unique guidance, navigation and control algorithms with the PX4 requires the knowledge to apply various programming languages to the embedded systems. Fortunately, industry standard tools such as Simulink allow embedded code generation from block models and other high-level programming languages. By developing the code and processes necessary for generating embedded software for the PX4, guidance, navigation and control algorithms may easily be written in Simulink and run on the PX4 autopilot system without the need to directly edit the code of the embedded software. This makes the PX4 autopilot system an ideal flight control solution for experimental micro air vehicles.

Using the PX4 autopilot system and PX4FLOW vision system as the central components of this vehicle, an airframe capable of vertical takeoff and landing is required to support them. A CAD model for a miniature 250mm diameter quadcopter frame is designed and the inertial properties of each component of the vehicle are modeled using Simulink. The characteristic equations are theoretically calculated and experimentally verified wherever feasible. The complete model is then used to develop a stable feedback control system capable of maintaining the attitude of the vehicle relative to a set of corresponding reference commands issued by the pilot from a radio controller.

Once the vehicle is verified to be flightworthy using PID control, work on the development of experimental control algorithms can begin. The PX4FLOW is implemented into the embedded code generation framework and used to obtain optical flow velocity and ultrasonic vertical position updates. These sensors are characterized in the vehicle model, and the measurements used to improve control over the local position of the aircraft in an effort to develop an improved image-assisted inertial navigation system.

The framework developed to test these experimental control systems has proven to be a vital tool in prototyping a variety of control algorithms on a low-cost micro air vehicle. The time savings and simplicity of the process when compared with alternative methods, allows engineers of varying levels of experience to utilize the PX4 autopilot system for many low-cost robotics demonstrations.

II. Vehicle Design

A. System Design

For these demonstrations, an air vehicle has to be designed that is capable of testing a variety of control methods using a low-budget autopilot system. The PX4 autopilot system with PX4FLOW vision system is selected as the best option for accomplishing this task. Considering that this vehicle undergoes many flight tests involving a high probability of failure, the airframe of the vehicle has to be designed rugged. Fortunately, the PX4FMU and PX4FLOW modules are very small with a combined mass of approximately 20 grams and a footprint hardly greater than that of a playing card (Figure 1).

Since the aircraft is often tested indoors or within a laboratory environment, it has to be capable of precise control at low velocities. The ability to hover and perform vertical takeoffs and landings is an essential requirement for the vehicle. A miniature four-rotor helicopter, also known as a quadcopter, is chosen as the primary basis for the design of the vehicle.

In an effort to reduce the potential for crash damage, the airframe is designed to be manufactured as a single piece. The sensitive components and autopilot system are ideally mounted near the center of gravity and away from any strong electromagnetic

fields generated by the motors or electronic speed controllers. With that in mind, the center of the airframe should remain open to mount electronic equipment. Unlike a conventional 'x' or '+' style quadcopter airframe, this requires the arms to be spread out and restructured about the center.

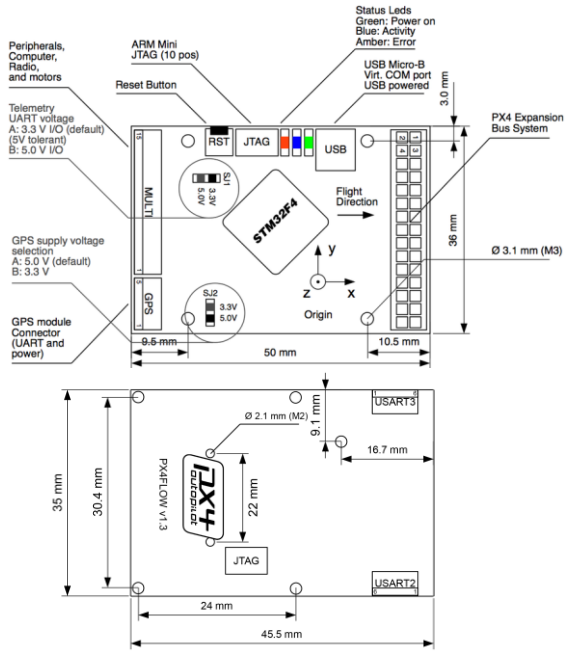

Figure 1. PX4FMU (top) and PX4FLOW (bottom).

The interior components are boxed in to protect them from crash damage and the arms angled out from the edges of the box-frame as opposed to being arranged perpendicular to the corners. This improves the resilience of the airframe during an impact, much as the spokes of a wheel distribute the load it is supporting. The interior of the frame is sized to fit the PX4FMU and PX4FLOW modules side by side with the camera of the PX4FLOW facing downward through the frame and the battery positioned beneath the PX4FMU. A diameter of 250mm from motor to motor is ideal for supporting the electronics and for manufacturing the frame as a single piece.

With a 250mm layout, off-the-shelf 5 ½" diameter propellers are selected to provide the maximum thrust with adequate clearance for the remaining components. For such a propeller, a micro brushless out-runner motor is needed to directly drive the propeller. This motor has to be lightweight, possess high-quality bearings and operate at the nominal voltage of a two-cell lithium polymer battery. The T-Motor model MT1306 is chosen to best suit this function (Figure 2). The advertised ratings show this motor is capable of producing 150 to 200 grams of equivalent thrust using a 7.4 volt battery and appropriately-sized propellers.

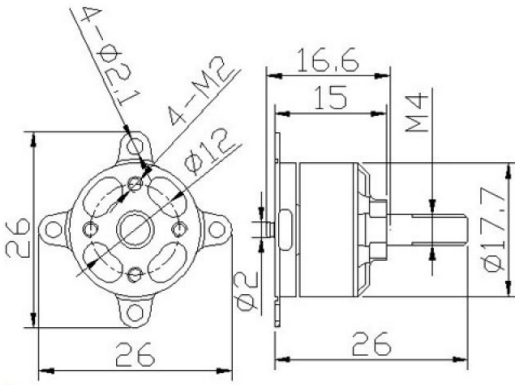

Figure 2. T-Motor Model MT1306 3100KV.

To power the aircraft, a lightweight 7.4 volt battery is chosen. Assuming a nominal load of two amps per motor, the necessary capacity of the battery is estimated to be about 1300mAh. Since the mass of these components is likely to be less than the advertised nominally-rated thrust at two amps, a 1000mAh lithium polymer battery is selected due to its high energy density and reduced size and weight.

$$\boldsymbol{capacity\ (mAh) = I_{nominal} T} \quad (1)$$

A light-weight set of 6A 7.4 volt compatible electronic speed controllers are chosen to drive the brushless motors and complete the power system. With this conceptual design, a mass budget can be created to determine and validate the flight-worthiness of the vehicle. In order to ensure adequate thrust for a variety of maneuvers, a 2:1 thrust to weight ratio is followed and incorporated into the mass budget. A conservative estimate of the equivalent thrust can then be determined from the rated specifications of the selected power system. This estimate amounts to approximately 600 grams of equivalent thrust. With a 2:1 thrust to weight ratio, the mass budget is determined to be approximately 300 grams.

Table 1. Mass Budget.

Component	Mass	Quantity	Margin	Subtotal
PX4FMU	10g	1	1g	11g
PX4FLOW	10g	1	1g	11g
MT1306 Motor	12g	4	2g	56g
APC 5545 Prop	2g	4	1g	12g
1000mAh 7.4V	60g	1	5g	65g
6A ESC	6g	4	2g	32g
Radio Receiver	12g	1	2g	14g
			Mass:	201g
			Budget:	300g
			Margin:	99g

A preliminary mass estimate of 201 grams indicates that the aircraft should be able to accommodate an airframe mass of up to 100 grams while maintaining an adequate thrust to weight ratio. With an estimated takeoff weight of 250 grams distributed between the four motors, a nominal load of less than two amps per motor is reasonable to assume and further supports that flight times of up to ten minutes may be achievable.

With a complete selection of components that conform to the design requirements, the base 250mm airframe is ready to be modified to accommodate each part. Motor mounting patterns are incorporated into each arm of the vehicle's airframe along with mounting patterns for the flight controller, vision system and electronic speed controllers.

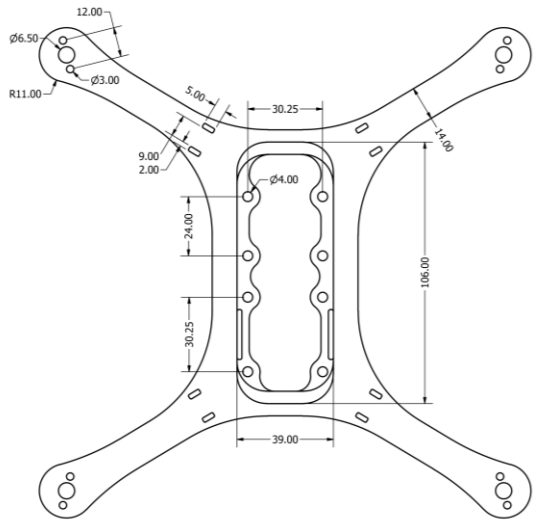

Figure 3. 250mm Airframe.

B. Power System

The power system for the aircraft consists of three primary components. These are the lithium polymer battery power source, the electronic speed controllers and the motors. A wiring harness is assembled from multiple lengths of 18AWG stranded silicon wire to supply power to each of these components. Each of the electronic speed controllers shares a common ground and power source with the lithium polymer battery. Each electronic speed controller also has a battery elimination circuit, or BEC. The BEC is a five volt linear regulator used to supply power to the flight controller, radio equipment and any auxiliary components demanding a five volt power supply. The three wires from the motors connect to the output of the electronic speed controller. The order of the wires should be reversed for every other motor to ensure alternating directions of rotation.

To improve performance of the system and reduce latency from the electronic speed controllers, an improved open-source firmware is installed to the electronic speed controllers. This allows the output pulse width modulated signal of the motors to be increased, the controller optimized and the input pulse width modulated reference command from the flight controller to be increased beyond the standard 50Hz cycle up to 400Hz. Reprogramming the electronic speed controllers requires a Silicon Labs programming tool. The BLHeli-Setup tool may be used to program and reconfigure the settings for each speed controller.

C. Flight Controller

The PX4 autopilot system is selected primarily because of its low cost, numerous features and open-source software. When compared with a conventional AVR-based Arduino microcontroller, the benefits of the ARM-based PX4 are quite apparent. The 168MHz 32-bit ARM Cortex-M4F onboard the PX4 is capable of performing 32 bit floating point calculations at 252 million instructions per second with 192KB of SRAM and 1024KB of flash memory; compared to the 20MHz 8-bit ATmega328 at about 20 million instructions per second with 2KB of SRAM and 32KB of flash memory. This allows the PX4 to handle complex arithmetic operations at minimal computational load, essential for estimation filters and advanced control systems.

The PX4 integrates this powerful mobile processor with an MPU-6000 six degrees of freedom (6DoF) inertial measurement unit, HMC5883L 3DoF magnetometer and MS5611 precision barometric pressure sensor. The InvenSense MPU-6000 is a micro-electro-mechanical system (MEMS) inertial measurement unit with a sample rate of up to 1kHz containing a 3-axis gyroscope and 3-axis

accelerometer. The Honeywell HMC5883L is a 3-axis digital compass, or magnetometer, with one degree accuracy and a sample rate of up to 160Hz. The MS5611 is a high-resolution barometric pressure sensor capable of 10 centimeter resolution accuracy as an altimeter. Despite the abundance of sensors, the PX4FMU has a very small form factor.

Figure 4. PX4 Flight Management Unit (FMU).

The integrated expansion bus of the PX4FMU may directly control four pulse width modulation enabled devices and communicate with peripherals using a Futaba serial bus (S.Bus), combined pulse-position modulation (CPPM), collective universal asynchronous receiver/transmitter serial connection (UART) or the universal serial bus (USB) connection. The expansion bus may be connected to an external PX4IO shield with additional input and output connections. Connecting directly to the PX4FMU expansion port is adequate for a quadcopter, requiring only four output channels. Commands are issued to the PX4FMU with the use of the CPPM connection from a 2.4GHz radio receiver. The radio receiver of choice is the FrSky D8R-XP telemetry and CPPM capable receiver. Eight independent radio channels may be simultaneously transmitted over this radio connection. A minimum of four channels are required to pilot the vehicle.

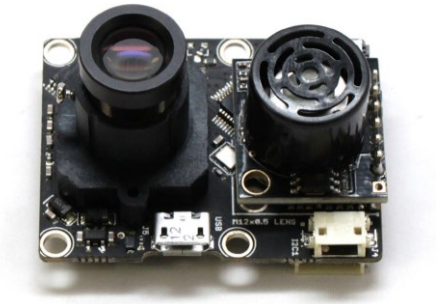

Figure 5. PX4FLOW Optical Flow Sensor.

The UART serial port of the PX4FMU is connected to the PX4FLOW module to obtain optical flow velocity estimates, secondary gyroscope and ultrasonic range measurements. This device allows for the advanced implementation of computer vision systems.

III. Vehicle Construction

A. Rapid Prototyping

A fused deposition modeling (FDM) system is used to construct the complete airframe for the vehicle. This method of 3D printing allows a semi-hollow infill pattern to be specified for the interior fabrication of each part. It is also more than capable of building the simple geometric shape of the frame. A rigid yet light-weight airframe is desired, so a honeycomb-patterned infill of approximately 30% fill density is set for fabrication of the final model. The resulting weight of the airframe is about 70 grams mass compared to a solid infill model which would amount to approximately 96 grams mass. Polylactide (PLA) thermoplastic filament is chosen over acrylonitrile butadiene styrene (ABS) to ensure that the airframe will not warp or deform while printing; a common problem encountered while printing large models with ABS plastic. The melting point of PLA plastic is significantly lower than ABS, but suitable for the range of temperatures these parts are expected to perform under.

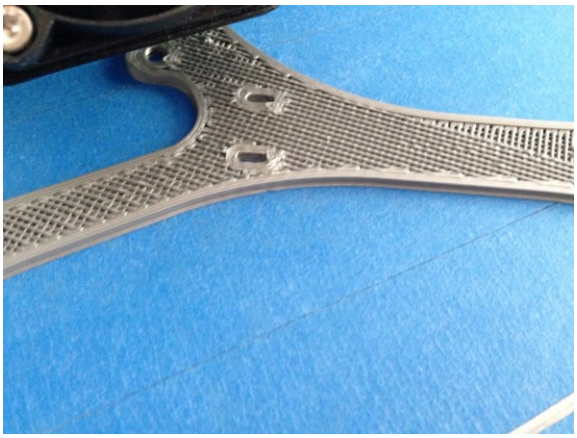

Figure 6. Semi-hollow 3D Printed Airframe Infill.

B. Wiring and Assembly

Final assembly of the vehicle requires mounting each component to the 3D printed airframe using standard metric hardware. Two of the three cables for the front left and back right motors are swapped to reverse the direction of the motor rotation, essential for yaw control. A standard UART serial cable connects the PX4FLOW with the PX4FMU. PWM outputs one through four are connected from the PX4FMU to the electronic speed controllers. An eight-channel FrSky D8R-XP 2.4GHz radio receiver

is mounted above the PX4FMU and configured to output all control data over CPPM to the PX4FMU radio input. All devices are wired to share a common ground and the PX4FMU is powered by a 5 volt battery elimination circuit from one of the four electronic speed controllers.

Figure 7. Complete Vehicle Assembly.

IV. Modeling and Simulation

A. Force and Moment Computations

For modeling and simulation purposes, the forces and moments acting on the rigid body must first be characterized. The model inputs are considered to be the eight pwm signals used to control the velocity of each rotor. The phase-lag contribution of the electronic speed controllers has been neglected due to their high operational frequency. For small inputs, the effect of the electronic speed controllers on the motors is assumed to be equivalent to a proportional change in voltage supplied to the brushless motor. The power supply voltage is scaled by the normalized pwm signal and an efficiency factor, $e_{m\%}$ (Equation 2). This voltage is the input signal to the brushless motor transfer function. The transfer function for rotor angular velocity is computed and reduced to a first order approximation because of the small size of the MT1306 motors (Equation 6). The electrical motor constant K_e is measured in rad/s per volt, R is the winding resistance, L is the winding inductance and J is the rotor inertia.

$$V_m = V_s\left(\frac{e_{m\%}}{100}\right)\left(\frac{pwm-1000}{1000}\right) \tag{2}$$

$$T_m = \frac{3RJ}{K_e K_t} \tag{3}$$

$$T_e = \frac{L}{3R} \cong 0 \tag{4}$$

$$K_t = \frac{K_e}{0.0605} \tag{5}$$

$$\frac{\omega_r(s)}{V_m(s)} = \frac{\frac{1}{K_e}}{T_m T_e S^2 + T_m S + 1} \tag{6}$$

The angular velocity is used to determine the thrust of each rotor. A set of models are used to approximate the forces and moments as a function of angular velocity (Equations 7 & 8). Force and moment matrices are used to equate the moment arm and orientations of each motor with the effective magnitude and direction of the applied forces. The force of gravity in the body frame is computed by rotating the gravity vector with a direction cosine matrix and multiplying each element of the resulting vector by the aircraft mass (Equation 9). Translational drag is computed along each axis using approximates of the cross-sectional areas. Rotational drag is approximated from the mean tangential velocity, moment arm and radial cross-sectional areas. The total forces and moments on the aircraft are summed within the body-frame.

$$F_r = k_f \omega_r \tag{7}$$

$$m_r = k_t \omega_r \tag{8}$$

$$F_g = M * DCM_{be}\begin{bmatrix} 0 \\ 0 \\ 9.81 \end{bmatrix} \tag{9}$$

$$F_d = -\frac{\rho}{2} v^2 C_d A \tag{10}$$

$$m_d = -\frac{\rho}{4} {\omega_b}^2 r^3 C_d A_r \tag{11}$$

$$\sum F_b = \sum F_r + F_g + F_d \tag{12}$$

$$\sum m_b = \sum m_r + m_d \tag{13}$$

B. Rigid Body Kinematics

The rigid body kinematics are computed using the Simulink 6DoF quaternion fixed-mass equations of motion aerospace block from MathWorks. Body-frame forces and moments are input to the system and solved using a quaternion-based implementation of the 6DoF Euler Newtonian equations of motion. This solution considers a flat-earth reference frame, neglecting the Coriolis Effect of the Earth. The 6DoF block outputs Earth-frame velocity, position, angular position, the body-to-earth direction cosine matrix, body-frame acceleration, velocity, angular velocity and angular acceleration. Gyroscopic effects of the

rotors are considered to be negligible at this scale and are not implemented into the model.

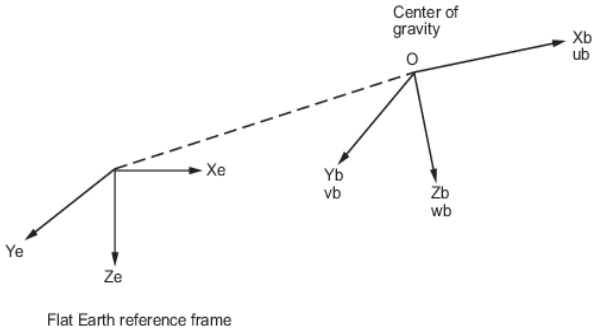

Figure 8. Flat Earth Reference Frame.

C. Inertial Measurement Unit (IMU) Model

The sensor noise and discrete characteristics of the strap-down 6DoF MPU-6000 IMU are characterized using the Simulink three-axis inertial measurement unit block. Noise power, update rate and operational limits are specified to match with the documented and verified characteristics of the Invensense MPU-6000. The use of this tool simplifies and consolidates the strapdown IMU model into a single configurable block.

D. Auxiliary Sensor Models

Additional sensor models are implemented for more complex verification and testing of estimation filters and control algorithms. These additional auxiliary sensors include the barometric pressure, temperature, ultrasonic range and optical flow velocity. The noise power of each signal is acquired from documented noise characteristics of each instrument and verified with high-rate data logs of each sensor. The standard deviation of the noise of each sensor signal is compared with logged data to ensure matching characteristics.

E. MultiSim Graphical User Interface

To provide a visual aid for the design and simulation of the system, a graphical user interface is developed to complement the system model. This user interface, MultiSim, is designed to run within MATLAB using the Graphical User Interface Design Environment (GUIDE). MultiSim is developed to be a flexible platform for modeling and simulation of vehicles operated using a conventional six-channel radio system. This helps to visually debug potential design flaws and modeling errors.

The vehicle is represented by a CAD model imported from a stereo lithography (.stl) file. The model mesh is rotated by the angular position of the vehicle and displayed to the user interface. Alternate camera angles may be selected for an improved perspective of the vehicle's performance. The vehicle-fixed camera may be used for visualization of attitude stability while the Earth-fixed camera provides visual feedback of the vehicle's velocity and position.

Gyroscope and accelerometer signals are plotted to aid in debugging and assist in visualizing the frequency of angular rotation of the vehicle over time. All other navigation data is printed to the window to provide real-time feedback of the vehicle's position and velocity.

MultiSim is developed to run in real-time alongside the Simulink model. Accomplishing this requires a fair amount of computational power to maintain real-time simulation. To improve performance, each feature of the graphical simulation may be independently enabled or disabled by the user during simulation.

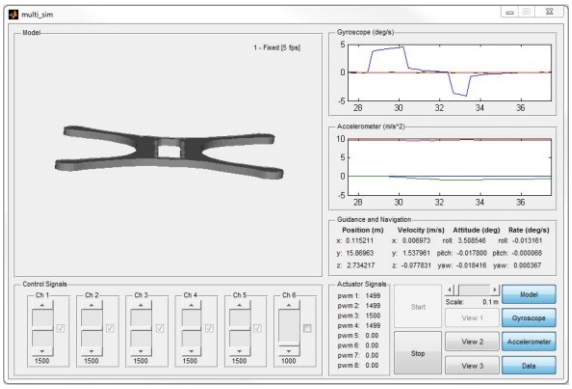

Figure 9. MultiSim Graphical User Interface.

V. Attitude Control System

For attitude control of the vehicle, an inner-loop angular velocity (rate) controller, with an outer-loop angular position (attitude) controller must be designed. The model for the plant is simplified to represent linear dynamics under low-speed equilibrium conditions. The plant P_0 is represented by the transfer function from the normalized roll command to the angular roll acceleration where α is the maximum torque, J_{xx} is the inertia and β is the motor constant T_m (Equation 14). P_1 and P_2 are both integrators to model angular roll velocity and angular roll position. Sensor dynamics H_0 and H_1 are treated as a unity gain.

$$P_0 = \frac{\alpha}{J_{xx}(\beta s+1)} \tag{14}$$

$$P_1 = P_2 = \frac{1}{s} \tag{15}$$

To insure system stability with zero steady state error, good disturbance rejection, high frequency noise attenuation and rapid system response, a proportional-integral-derivative (PID) rate controller with high-frequency roll-off filter is designed (Equation 16). To prevent overshoot to step commands, a pre-filter W_0 is incorporated within the system.

$$K_0 = k_{p0}\left[\frac{k_{d0}s^2+s+k_{i0}}{s}\right]\left[\frac{n_0}{s+n_0}\right] \quad (16)$$

$$W_0 = \left[\frac{w_0}{s+w_0}\right] \quad (17)$$

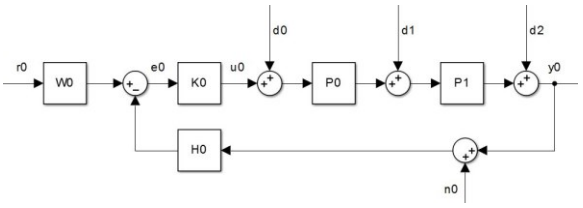

Figure 10. Rate Control System.

For control of the roll attitude, a similar proportional-integral (PI) attitude controller must be designed as an outer loop to the rate control system S_0.

$$S_0 = \frac{P_0K_0}{1+P_0K_0H_0} \quad (18)$$

$$K_1 = k_{p1}\left[\frac{s+k_{i1}}{s}\right] \quad (19)$$

$$W_1 = \left[\frac{w_1}{s+w_1}\right] \quad (20)$$

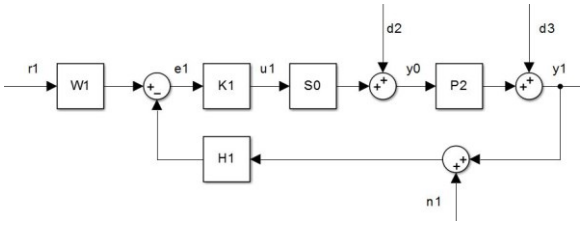

Figure 11. Attitude Control System.

This simple controller design is implemented into the complete system model as the basis for further testing and verification of the attitude control system. The control gains are selected with the assistance of the Simulink control systems toolbox. The system response is tuned to achieve a rise time of less than 50ms with an overshoot of less than 10%.

VI. Embedded Software

A. System Architecture

Generating embedded software for the PX4 relies on the use of the existing system architecture created by the Pixhawk development team. The PX4 software utilizes NuttX, a 32-bit real-time operating system (RTOS), for execution of all processes. The modified PX4 source code is reconfigured to incorporate only essential drivers and standard processes into the RTOS. For the purpose of using Simulink-generated embedded software, all existing controllers are removed from the PX4 makefile and replaced by the new process 'simulink_app'. The Simulink application operates as a high-priority task with a cycle time of four milliseconds. It communicates with all essential drivers and system processes while assuming direct operation of input-output control.

B. Inter-process Communication

The Simulink-generated embedded software runs on the PX4 as an independent high-priority application thread. Input and output signals are handled via a publish-subscribe pattern. Device drivers publish high-rate data to a micro Object Request Broker (uORB) and applications may subscribe to these data publications for real-time data acquisition. The Simulink-generated embedded software subscribes to a set of common uORB topics and routinely polls these topics for new data during each cycle. Output signals of the Simulink-generated embedded software may be published to another topic or directly handled by the use of the input-output control. This inter-process communication allows any standard PX4 features and device drivers to be easily incorporated with the Simulink-generated embedded software.

C. Boot Procedure

The standard operating procedure at boot is written to a startup script 'rc.txt' and stored to the memory of the MicroSD flash card '\etc\rc.txt'. This startup script initializes all processes required for nominal operation of the system. The startup script is preconfigured for use with the Pixhawk flight controller and may be edited to operate with the PX4FMU. All required drivers, communication protocols, estimation algorithms and controllers must execute at boot from the startup script. The default Simulink-generated embedded software application, simulink_app, executes all Simulink generated code

and automatically handles rate transitions between cycles, not to exceed the primary update rate of 250Hz.

VII. Optical Navigation

A. State Estimation

One of the greatest obstacles to implementing an optical navigation system is the system state estimation of the vehicle position. This process is responsible for providing continuous and accurate state estimates of the local vehicle position by use of stochastic estimation techniques. Using a variety of sensor inputs including optical flow, acceleration, pressure altitude and ultrasonic range, a Kalman filter is designed to provide improved estimates of the local position of the vehicle. The Kalman filter is used under the assumptions that the sensor noise distribution is normal with a zero mean and the internal model is accurately representative of the system. This Kalman filter uses an internal model to predict the estimated state output of a system input. The state prediction is updated by sensor measurements and the Kalman gain is adjusted to improve future state predictions.

The design of this Kalman filter incorporates ten total states including three position states (Equation 21), three velocity states (Equation 22), three acceleration states and an additional pressure altitude state, all in the local NED earth-frame coordinate system. These ten state equations are represented by the state transition matrix A and state input matrix B of the state prediction (Equation 24).

$$\bar{p}_x = p_x + v_x dt + \frac{a_x}{2} dt^2 \tag{21}$$

$$\bar{v}_x = v_x + a_x dt \tag{22}$$

$$x = [\vec{p}_{x,y,z} \quad \vec{v}_{x,y,z} \quad \vec{a}_{x,y,z} \quad alt]^T \tag{23}$$

$$\bar{x} = Ax + Bu \tag{24}$$

The Kalman gain is computed as a function of the projected state error covariance P, observation matrix H and measurement error covariance R; where the projected state error covariance is a function of the state transition matrix A and process error covariance Q (Equation 26). The measurement error covariance is established from the sensor noise covariance.

$$P = APA^T + Q \tag{25}$$

$$K = \frac{PH^T}{HPH^T + R} \tag{26}$$

The state prediction is then updated by the observed sensor measurements Z, and the error covariance is again updated. The resulting state estimates may be used as feedback signals for the navigation system.

$$Z = \begin{bmatrix} altitude \\ flow_x \\ flow_y \\ flow_x \frac{d}{dt} \\ flow_y \frac{d}{dt} \\ sonar \end{bmatrix} \tag{27}$$

$$\bar{x} = x + K(Z - Hx) \tag{28}$$

$$P = (\bar{I} - KH)P \tag{29}$$

B. Navigation System

Similar to the aforementioned attitude control system, the position controller is the outermost loop of the position control system. Position reference commands propagate attitude and rate commands through the controllers to each motor. Reference commands for the position controller are acquired from either user-input radio commands or the guidance and navigation system.

The guidance and navigation system stores predefined waypoints to memory and routinely references the current vehicle position. Each set of waypoints are issued as a step command to the position controller. When the navigation system has verified the vehicle position to be within a specified radius of the desired waypoint, the next waypoint will be issued by the guidance system to the position controller.

VIII. Conclusion

Despite the complexity of the system as a whole, the design and construction of a quadcopter for these tests has been a true testament to the advantages of using a multicopter over conventional model aircraft. Construction of the vehicle requires very few components, and the airframe is easily replaceable. Furthermore, the ultra-compact light-weight vehicle is resilient to crashes. It also allows testing to easily be performed in a variety of environments, eliminating the need to set up a large controlled environment for testing.

An attitude control system was successfully implemented and verified through several flight tests. Limitations on real-time serial communication and data logging from the flight controller inhibit precise verification of the control system. Dynamic behavior very closely resembles that of the computer model. The MATLAB control systems toolbox proved very useful in linearizing and tuning the controller parameters. Some discrepancies in the model resulted with invalid regions of stability, but this was to be expected and adjustments to the controller parameters were made to ensure system stability.

Through the process of developing an optical navigation system, it became apparent that a suitable estimation algorithm was most essential to a functional optical navigation system. Although progress was made in developing a Kalman filter to estimate the vehicle position, accurately modeling the system dynamics and noise characteristics proved quite complex, though essential for developing a working Kalman filter. The resulting estimation algorithm performed suitably for the computer simulation, but failed to produce usable data in most recent flight trials. Waypoint navigation prompted the need for a quaternion-based navigation system as Euler angles provided invalid solutions under some circumstances.

In an effort to accelerate the process from prototyping and simulating models on a computer directly to flight-testing and verification of the control algorithms, the integration of the PX4 autopilot system with MathWorks' embedded coder has proven to be an invaluable tool. This process eliminates the further need to write hundreds of lines of embedded code and greatly reduces the likelihood of encountering routine software bugs and glitches. The process of generating the embedded software is simple enough to be used without any knowledge of embedded programming languages. Since the software has been developed around the PX4 project, maintaining the code and incorporating new features can be easily and reliably accomplished by including the Simulink wrapper code with the most recent source code and altering the Makefile to include the Simulink application.

Further progress must be made with the optical navigation system in order for it to reach a usable state. Fortunately, the development of this project has laid the groundwork for experimenting with a variety of complex modern guidance, navigation and control techniques. Not only has it proved useful in developing systems for multicopters, but it is versatile enough to be implemented on ground vehicles, fixed-wing aircraft and even rockets.

Acknowledgments

Mike Hannan for his expert mentoring, project inspiration and support. Dr. Armando Rodriguez and Dr. Srikanth Saripalli for their expert project guidance and support. Steve Kuznicki, Ryan Gordon and Jamie Winter of MathWorks for software support, licensing and product support development.

References

[1]Bresciani, Tommaso. "Modelling, Identification and Control of a Quadrotor Helicopter." (2008). Department of Automatic Control. Lund University. Web. <http://www.control.lth.se/>.

[2]Brogaad, Rune Yding. "Control of Multi Rotor Helicopter." (2012). DTU Electrical Engineering. Technical University of Denmark. Web.

[3]Fux, Samuel. "Development of a Planar Low Cost Inertial Measurement Unit for UAVs and MAVs." (2008). Autonomous Systems Lab. ETH Zurich. Web.

[4]García Carrillo, Dzul López, Lozano, and Pégard. Quad Rotorcraft Control. Springer. Print.

[5]Gede, Gilbert. "Introduction to Kalman Filtering An Engineer's Perspective." Http://biosport.ucdavis.edu/. University of California, Davis, 1 Jan. 2011. Web.

[6]Kong, Xiaoying. "Inertial Navigation System Algorithms for Low Cost IMU." (2000). Department of Mechanical Engineering. The University of Sydney. Web. <http://db.acfr.usyd.edu.au/>.

[7]Lorenz, Meier. "Pixhawk." PX4 Autopilot Platform. 1 Jan. 2009. Web. <https://pixhawk.ethz.ch/>.

[8]Magnussen, Øyvind, and Kjell Eivind Skjønhaug. "Modeling, Design and Experimental Study for a Quadcopter System Construction." (2011). Department of Engineering. University of Agder. Web. <http://brage.bibsys.no/>.

[9]Oguntoyinbo, Oludayo. "PID Control of Brushless DC Motor and Robot Trajectory Planning Simulation with MATLAB/SIMULINK." (2009). Theseus. Universities of Applied Sciences. Web. <http://publications.theseus.fi/>.