NASA/TM–20220011570

Rapid Flight Control Law Deployment and Testing Framework for Subscale VTOL Aircraft

Garrett D. Asper
Universities Space Research Association, Hampton, Virginia

Benjamin M. Simmons
Langley Research Center, Hampton, Virginia

September 2022

NASA/TM–20220011570

Rapid Flight Control Law Deployment and Testing Framework for Subscale VTOL Aircraft

Garrett D. Asper
Universities Space Research Association, Hampton, Virginia

Benjamin M. Simmons
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

September 2022

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

A set of procedures was developed to enable rapid flight control law deployment and testing on subscale vertical takeoff and landing (VTOL) aircraft. Low-cost, subscale flight vehicles have become well-suited testbeds for rapid flight dynamics and controls research progression; however, integration of custom flight control laws onto flight hardware has historically been an arduous task. The toolchain described in this report leverages Simulink® with the UAV Toolbox, a Pixhawk flight computer running PX4 firmware, and QGroundControl to efficiently design and flight test custom control algorithms. A subscale CL-84 VTOL aircraft was used as a testbed in this investigation to exercise the hardware integration process on a physical model. Implementation of custom attitude stabilization control laws and programmed test input excitations for aircraft system identification were demonstrated using the expeditious hardware integration process. The detailed procedures given in this report are expected to be used in future flight test efforts.

Contents

List of Figures

List of Tables

Acronyms

CAN	controller area network
ESC	electronic speed control
eVTOL	electric vertical takeoff and landing
MAV	micro air vehicle
PID	proportional integral derivative
PPM	pulse position modulation
PTI	programmed test inputs
PWM	pulse width modulation
QGC	QGroundControl
RC	remote control
RPM	revolutions per minute
SUPPQ	Simulink®, UAV Toolbox, PX4, Pixhawk, QGroundControl
UAM	Urban Air Mobility
UAV	Unmanned Aerial Vehicle
uORB	micro Object Request Broker
USB	universal serial bus
VTOL	vertical takeoff and landing

1 Introduction

Recent technology advancements in small, low-cost unmanned aerial vehicle (UAV) hardware have enabled the use of small UAVs for flight-test research in a variety of areas, including flight control law development and testing. However, integration of custom control laws onto UAV platforms has historically been time and resource intensive. The purpose of this report is to document a process used to overcome the gap between rapid design, implementation, and testing of experimental flight control algorithms. A model CL-84 vertical takeoff and landing (VTOL) UAV was used as a testbed to demonstrate integration of a custom controller onto hardware in a streamlined manner and validate a toolchain to rapidly flight test custom control laws.

An overview of the Simulink®, UAV Toolbox, PX4, Pixhawk, QGroundControl (SUPPQ) toolchain documented in this report is shown in Figure 1. The process is used to deploy a Simulink® model onto a Pixhawk flight computer using the MathWorks® UAV Toolbox [1] with a support package for PX4 autopilots [2]. This framework allows rapid integration of flight control algorithms onto the flight computer onboard the aircraft for flight testing. Flight test results can then be used to iteratively improve the flight control laws, as needed. For this study, the UAV Toolbox and QGroundControl (QGC) ground control software [3] interfaced with a Pixhawk Cube Black [4] running PX4 firmware [5], both while connected to a laptop computer in hardware-in-the-loop simulations and onboard an aircraft by embedding custom-generated firmware on the Pixhawk. Initial ground and flight demonstrations were used to validate the utility of the rapid firmware deployment process.

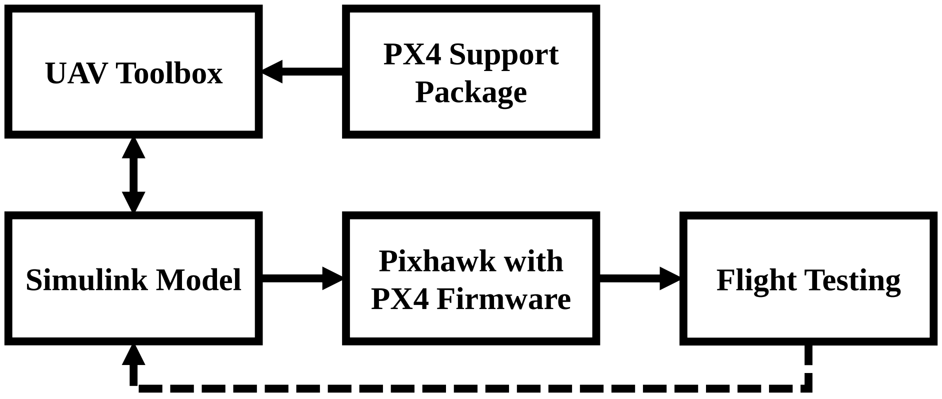
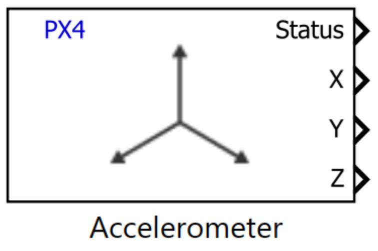


Figure 1: Overview of the SUPPQ toolchain for rapid flight control law integration and testing.

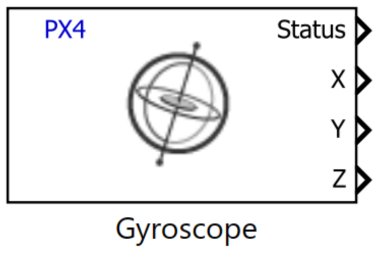
This report was developed as documentation of a 10-week summer internship project completed by the primary author in the Flight Dynamics Branch at NASA Langley Research Center under the mentorship of the second author and Jacob Cook. The report is organized as follows: Section 2 provides an overview of the flight control law deployment capabilities enabled by the UAV Toolbox. Section 3 describes initial hardware-in-the-loop simulations executed with the Pixhawk connected to a laptop computer. Integration and testing of custom control laws on a subscale CL-84 UAV are discussed in Section 4. Future work is discussed in Sections 5 and overall conclusions are summarized in Section 6. Appendices A-F provide documentation on the materials used, hardware setup, software setup, configuration parameters, troubleshooting steps, and flight test procedures. Please note that all work conducted for this study used MATLAB®/Simulink® R2021b and PX4 firmware version 1.10.2. Use of future software releases may require modifications to the procedures described in this report.

2 UAV Toolbox Capabilities for Flight Control Law Development

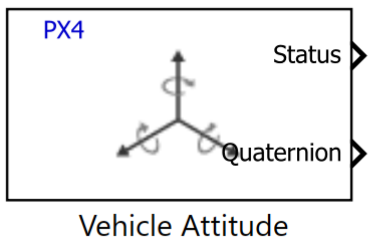
The MathWorks® UAV Toolbox [1] with the Support Package for PX4 Autopilots [2] provides the tools needed in the Simulink® environment to interface with PX4 firmware running on a Pixhawk flight computer. These tools allow users to easily implement and flight test custom algorithms on a UAV. Figure 2 shows Simulink® blocks provided in the Support Package for PX4 Autopilots that were investigated in this study. These blocks allow various sensor measurements, transmitter signals, and parameters to be read into a Simulink® model, and allow output PWM signals and uORB messages to be sent from a Simulink® model to the flight computer.



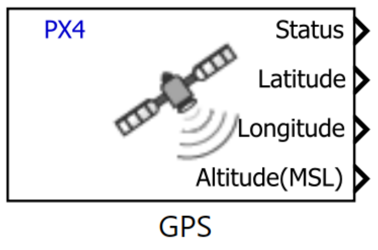
(a) Accelerometer block



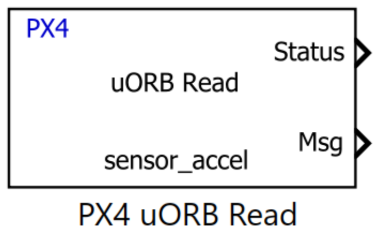
(b) Gyroscope block



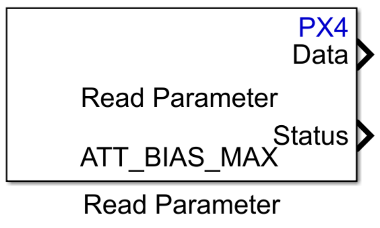
(c) Vehicle Attitude block



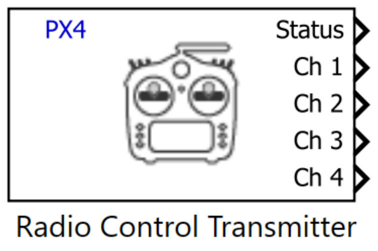
(d) GPS block



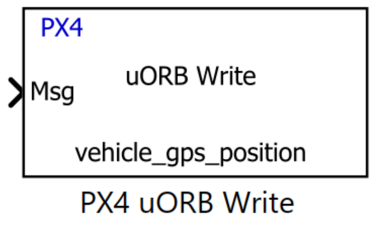
(e) uORB Read block



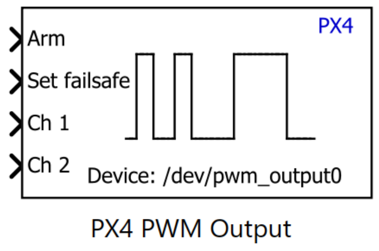
(f) Read Parameter block



(g) RC Transmitter block



(h) uORB Write block



(i) PWM Output block

Figure 2: PX4 support package Simulink® blocks investigated for this study.

The following subsections describe the different methods for interfacing with the Pixhawk, including (1) Connected I/O mode, (2) Monitor and Tune mode, and (3) Build, Deploy and Start mode. Appendices A-F outline procedures to set up the toolchain that enables Connected I/O simulations as well as embedding custom firmware with Build, Deploy and Start.

2.1 Connected I/O Mode

The most streamlined way to start interfacing with the Pixhawk was found to be using the "Connected I/O" connected mode provided under the Hardware tab in Simulink®. Throughout initial testing, setup, and UAV Toolbox capability investigation, this mode served as a useful tool for quickly investigating the effects of different blocks, settings, and signals on the Pixhawk hardware. Connected I/O generates the firmware and runs the code on a computer while sending and receiving signals using a hardwired connection to the Pixhawk. This mode is helpful for efficient firmware checkouts without needing to load custom firmware onto the Pixhawk. Additionally, scopes can be used to monitor internal and external signals. The behavior of control surfaces and motors can be observed on the bench by incorporating the hardware into the loop. Although the Connected I/O mode does not run in real-time, the consistency and utility of Connected I/O during bench testing proved useful for the initial testing of custom controllers before embedding the firmware onboard the Pixhawk. Connected I/O was used to perform all bench testing that will be described in Section 3.

2.2 Monitor and Tune Mode

"Monitor and Tune" is an alternative connected mode run by embedding the firmware onboard the Pixhawk, while maintaining a hardwired USB connection to tune parameters. This allows the ability to tune parameters and change gains in the Simulink® diagram while running firmware on the Pixhawk. Although this is a promising capability, for this study, Monitor and Tune mode presented consistent loading issues that would result in critical firmware failures. Firmware errors from Monitor and Tune required repeating the Hardware Setup process in Simulink® or completely reloading the PX4 firmware from QGC. Because of the consistent issues with Monitor and Tune, Connected I/O mode was used for all bench testing.

2.3 Build, Deploy and Start Mode

"Build, Deploy and Start" fully embeds the custom firmware onboard the Pixhawk and allows it to run independently from the computer. Build, Deploy and Start compiles the model developed in Simulink® and embeds the firmware to the board so that the firmware runs automatically when the board is powered on. This mode allows custom control laws developed in a Simulink® model to be rapidly integrated and flight tested. Build, Deploy and Start was used to perform the testing described in Section 4.

3 Preliminary Bench Testing Using Connected Simulations

The following subsections describe three bench tests that were conducted in Connected I/O mode to experiment with the functionality of the SUPPQ toolchain. The bench test setup is shown in Figure 3.

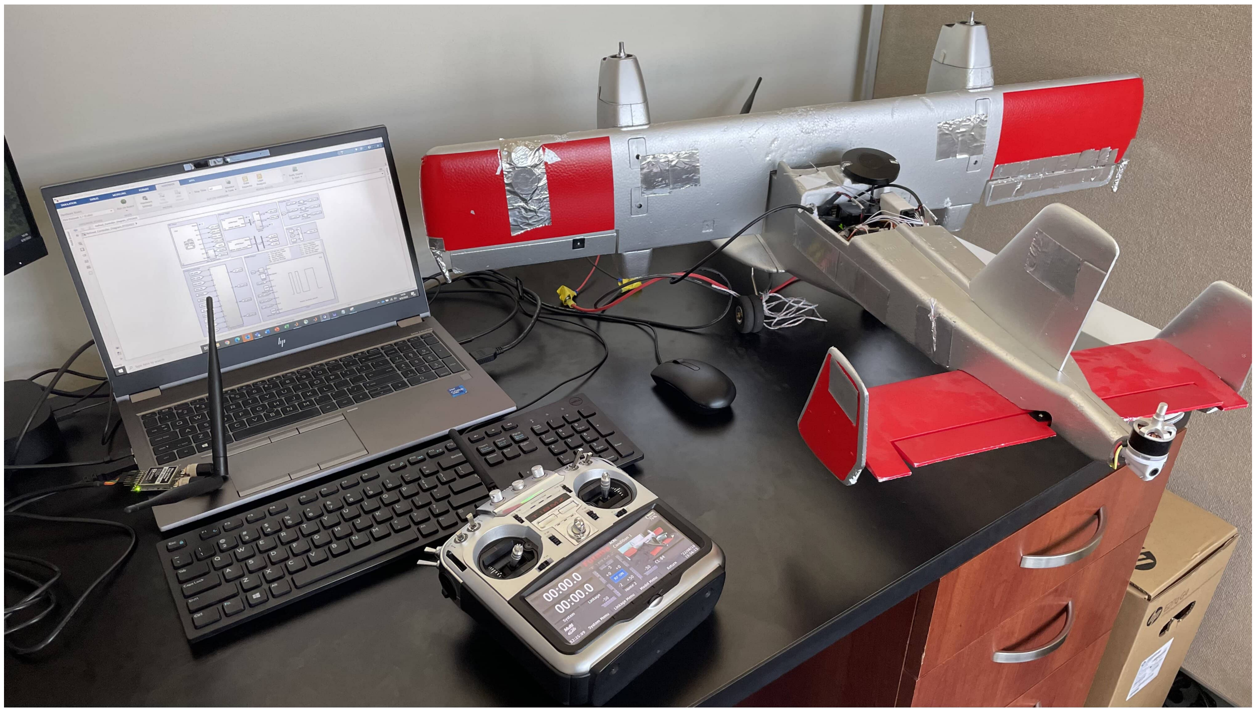


Figure 3: Bench test setup used to run connected simulations.

3.1 Bench Test 1: Initial Arming and Servo Response Test

An initial test was conducted to ensure that the Pixhawk was arming correctly and able to command servo-actuators and motors. To run connected simulations, the Pixhawk had to be armed using the PX4 PWM Output block in the UAV Toolbox (Figure 2i). The arming channel specified the state of the Pixhawk in accordance with the failsafe channel. The failsafe channel was set to 0, allowing the Pixhawk to start up disarmed. The PX4 prearm, arm, and disarm configurations are described further in the PX4 documentation [6].

To arm the Pixhawk, the physical safety switch was pressed for three seconds to prearm the system before beginning a Connected I/O bench test. Next, the `boolean` datatype value of 1 was passed into the arming channel of the PWM output block. Then, the Connected I/O mode would be started to arm the Pixhawk and begin sending test PWM values to the actuator channels. For this demonstration, servo and motor PWM commands were manually entered into a constant block to send signals to the PWM output channels using the `uint16` data type. An example Simulink® diagram used for this test is shown in Figure 4, where the constant block could be changed during the connected simulation to test how different servos and motors respond to varying signals.

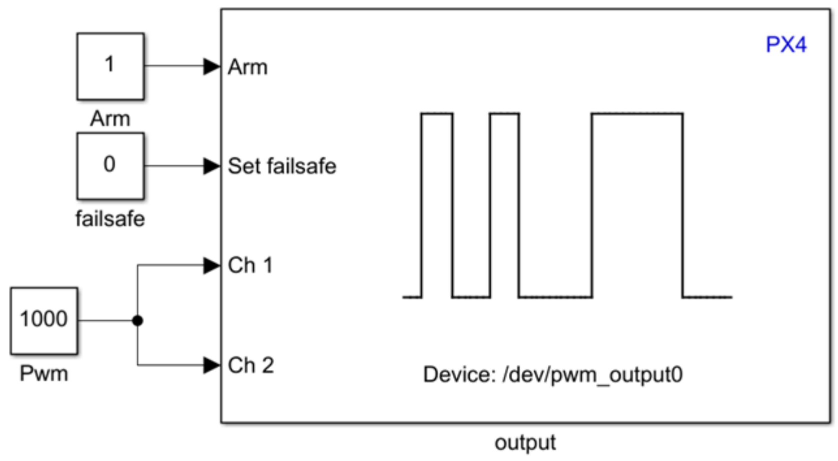


Figure 4: Simulink® model for the initial arming and servo response bench test.

3.2 Bench Test 2: Automated Servo Response from Sensed Vehicle Attitude

A second bench test was used to demonstrate servo responses controlled by changes in the Euler orientation angles obtained from the PX4 state estimation algorithm [7, 8]. The quaternion outputs from the Vehicle Attitude sensor block (Figure 2c) were converted to roll, pitch, and yaw Euler orientation angles in degrees. Gains were then applied to each control servo position centered at a PWM value of 1500 microseconds. The gains could be adjusted to provide more or less of a response based on the orientation of the aircraft. The Simulink® model used for this demonstration is shown in Figure 5. This model and test served as a proof of concept that sensor data could regulate actuator response using the UAV Toolbox and a PX4 autopilot.

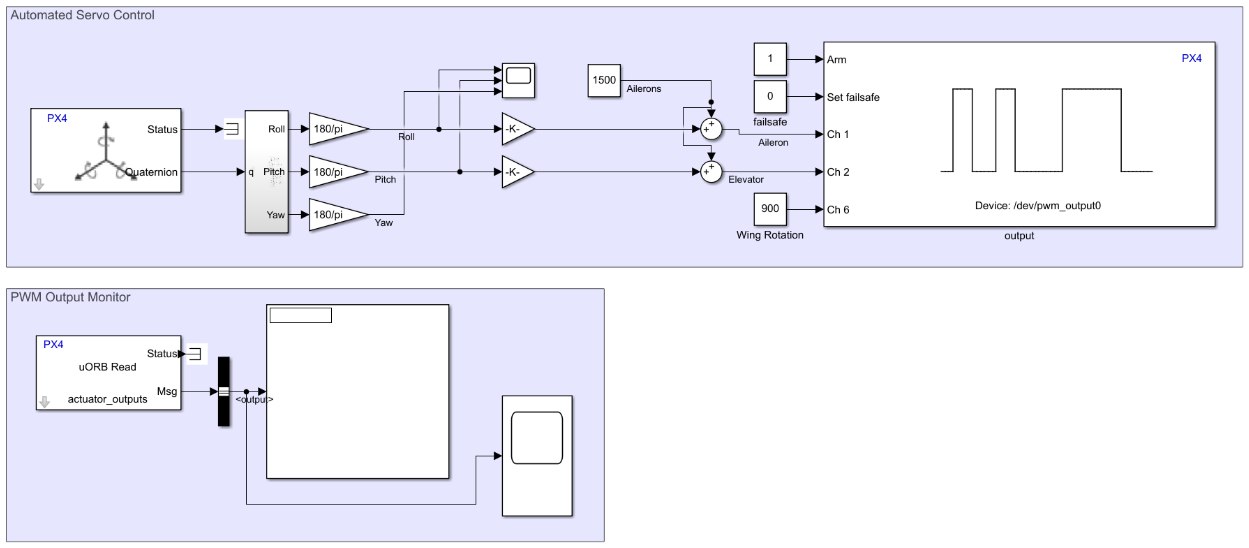


Figure 5: Simulink® model for the automated servo response bench test.

3.3 Bench Test 3: Automated Servo Response with RC Transmitter Input

A third bench test was used to demonstrate preliminary RC transmitter integration with an automatic flight controller. Figure 6 shows the Simulink® diagram for this test, which is a more complex version of the model shown in Figure 5. The "Automated Servo Control" group (the upper portion of the diagram) acted as a placeholder for a controller in this pedagogical demonstration. The RC transmitter commands were summed with the automatic controller commands to obtain the final PWM output signal sent to the Pixhawk.

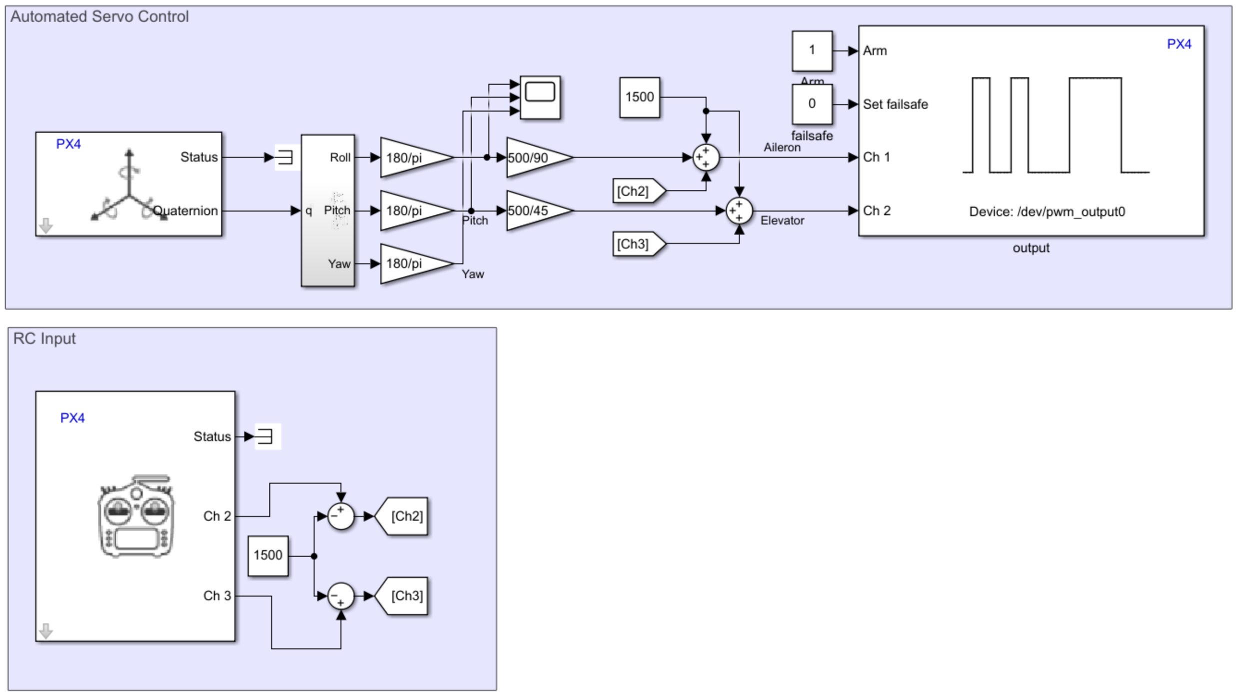


Figure 6: Automated servo response with radio control input Simulink® model.

4 CL-84 Hover Controller Simulink® Model Overview

A custom hover attitude stabilization controller was developed for a subscale CL-84 aircraft, shown in Figure 7, to investigate and demonstrate the SUPPQ toolchain described in this report. The CL-84 is a tilt-wing aircraft with two propellers mounted on the wing and one propeller mounted on the tail that can rotate about the body x-axis. The aircraft has an elevator control surface on the horizontal tail and a blown aileron on each wing. The aircraft model used for this study did not have a rudder, and the rear propeller rotation about the body x-axis was not used. In hover, differential left and right front propeller thrust commands provided roll control, differential front and rear propeller thrust commands provided pitch control, and actuation of the blown aileron surfaces was used for yaw control. The CL-84 hardware setup is discussed further in Appendix A.1.

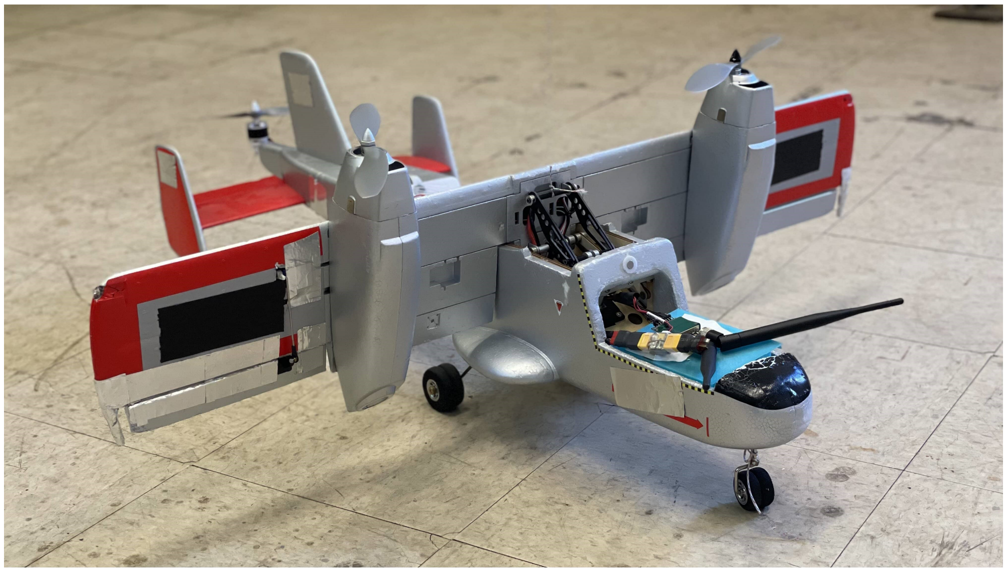


Figure 7: Subscale CL-84 aircraft in the hover configuration.

The custom CL-84 hover controller architecture is shown in the Simulink® model displayed in Figure 8. Although this study was specific to the CL-84 airframe, the overall framework can be readily adapted to develop a controller for different aircraft configurations and to test more advanced control laws. The following subsections describe each aspect of the model (differentiated by the shaded boxes in Figure 8), followed by a discussion of preliminary flight test results.

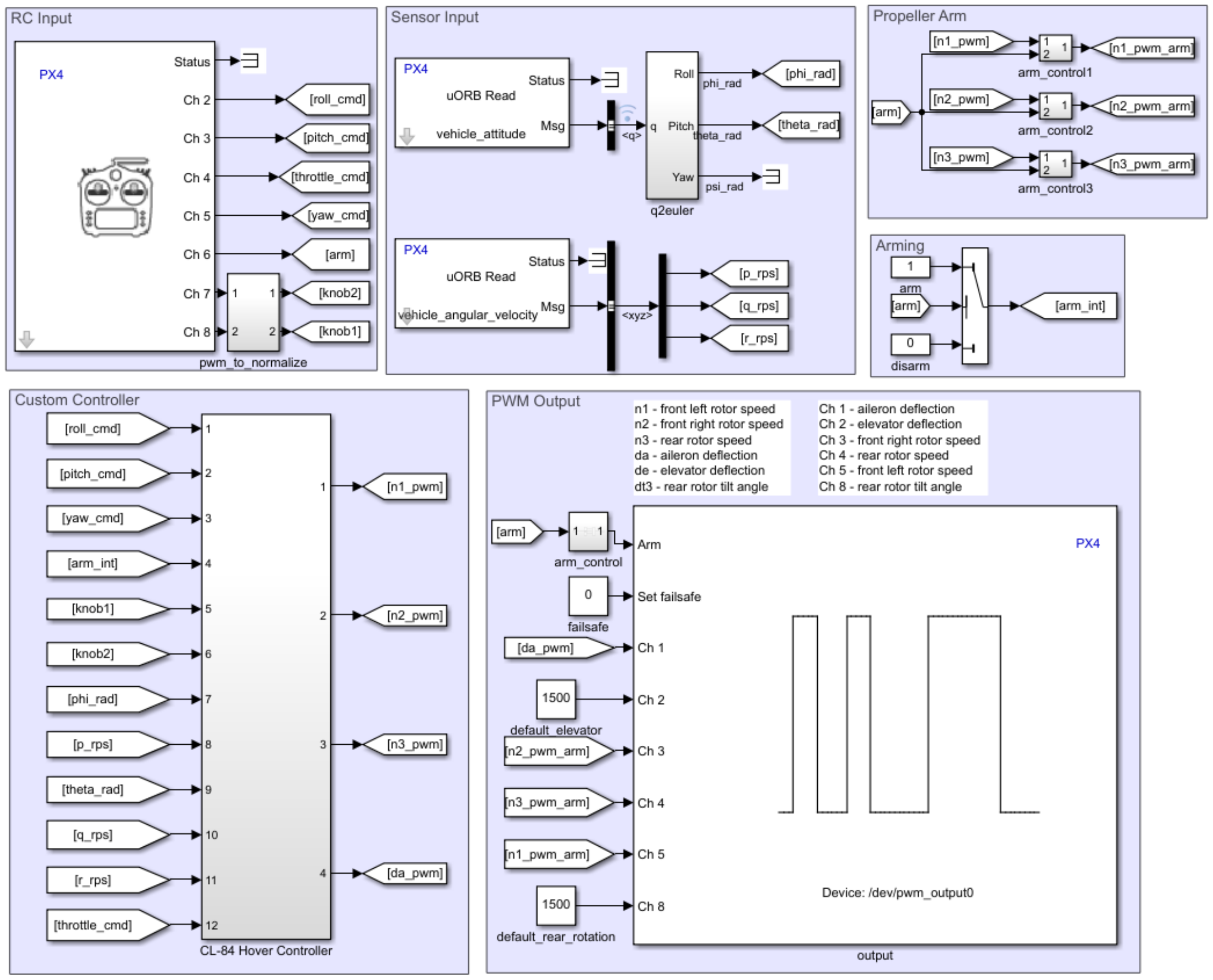


Figure 8: CL-84 hover controller Simulink® model.

4.1 RC Input

The RC Input group (shown in the upper left of Figure 8) reads the PWM signals from the transmitter via the connected receiver. Channels 2 through 5 were used to send roll, pitch, throttle, and yaw PWM commands, respectively, ranging from 1000 to 2000 microseconds. Channels 7 and 8 were configured as tuning inputs for controller gains and programmed test input injections. A picture of the RC transmitter used for testing in this work is shown in Figure 9, with labels showing the user inputs and corresponding channel numbers. Note that the PWM signals from the transmitter are a `uint16` datatype and should be converted to the `double` datatype before performing any calculations using the signals.

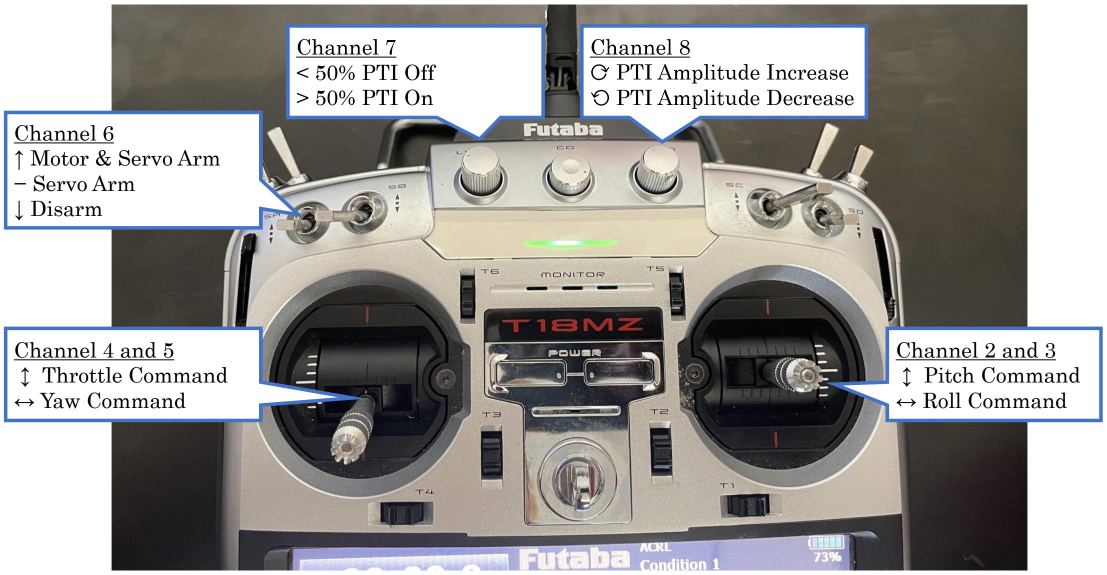


Figure 9: RC transmitter used for control law testing.

4.2 Sensor Input

The Sensor Input group (shown in the upper center of Figure 8) reads sensor measurements and state estimates from the PX4 state estimator, which are provided via the uORB messaging system. The uORB Read block in the UAV Toolbox (Figure 2e) allows the selection of a specified vehicle topic from the uORB messaging system [9]. The `vehicle_attitude` topic was used to read the quaternion estimates, which were subsequently converted to Euler roll, pitch, and yaw orientation angles (ϕ, θ, ψ) for use in the control logic. The `vehicle_angular_velocity` topic was used to read the angular velocity measurements (roll rate p, pitch rate q, and yaw rate r) from the onboard Pixhawk rate gyros for use in the control logic.

The Euler orientation angles are estimated signals fused using measured data from sensors integrated into the Pixhawk via the PX4 state estimator [7]. As described in Reference [8], the PX4 state estimation logic contains an extended Kalman filter (EKF) and output predictor algorithm that are said to have “between 100 and 200 milliseconds” and “less than 100 microseconds” of latency, respectively. The attitude states from the EKF and output predictor algorithm (contained in the `estimator_status` and `vehicle_attitude` uORB topics, respectively) were compared in bench testing to verify the attitude estimates to be used for feedback control. As can be seen in Figure 10, the roll, pitch and yaw angles derived from the `vehicle_attitude` uORB topic have less delay compared to the corresponding signals from the `estimator_status` uORB topic. Consequently, the `vehicle_attitude` topic was chosen for the controller because substantial state estimate latency will lead to undesirable flight characteristics. The same recommendation was made in Reference [10].

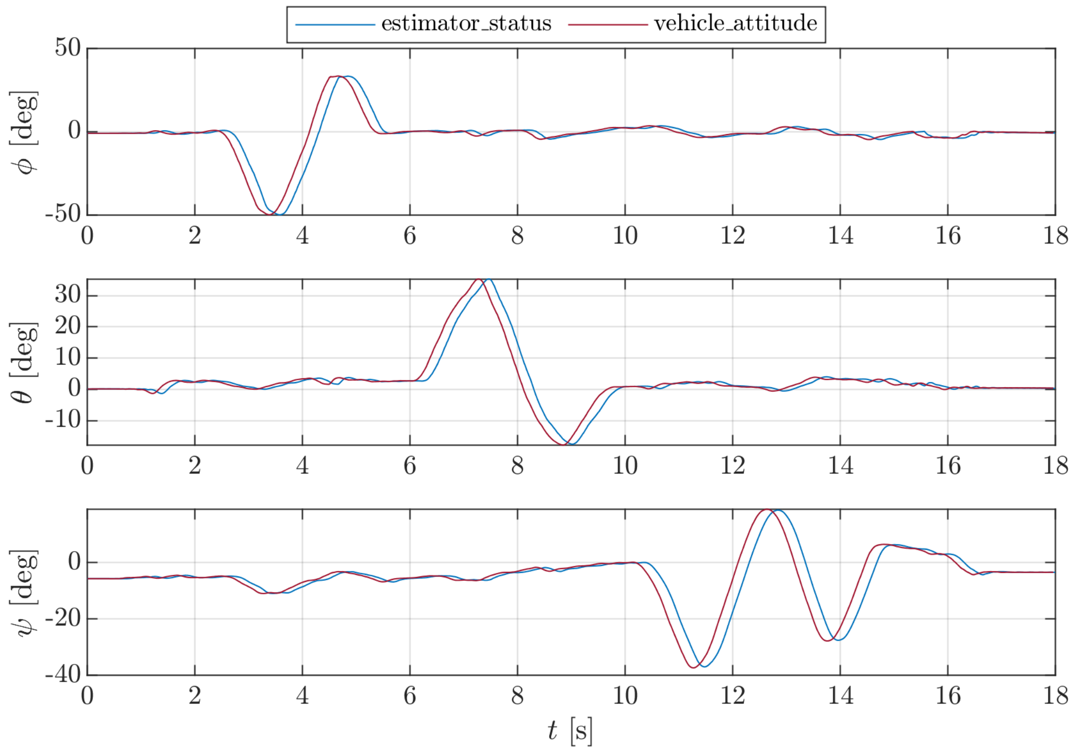


Figure 10: Comparison of orientation angle signals derived from the "estimator_status" (EKF) and "vehicle_attitude" (output predictor) uORB topics in bench testing.

4.3 Arming

The Propeller Arm and Arming groups (shown in the upper right of Figure 8) were controlled by a three-position switch on the transmitter. As summarized in Table 1, the PX4 PWM Output block was armed when the switch was set to the second or third position. The Propeller Arm section of the diagram limited motor arming to when the switch was set to the third position. The arm_control blocks in the Propeller Arm section of the diagram contained subsystems that required a PWM value above 1700 microseconds from the three-position switch on the transmitter to send signals to the motors. Otherwise, a disarmed value of 900 microseconds was sent to the motors.

Table 1: Three-position arming switch logic.

Position	Action
3 (↑)	motors and actuators armed
2 (−)	actuators armed
1 (↓)	disarmed

4.4 Custom Controller

The Custom Controller group (shown in the lower left of Figure 8) contained flight control logic for the CL-84 aircraft in hover. The CL-84 Hover Controller subsystem inputs include the transmitter signals, attitude angles, and angular rates, which were used to determine the outputs sent to the motors and control surfaces. The controller block was composed of proportional, integral, derivative (PID) attitude stabilization control logic; a mixer to convert roll, pitch, yaw, and throttle commands to PWM commands sent to the motors and servos; and a programmed test input (PTI) injection capability. The controller framework is depicted in a high-level diagram shown in Figure 11.

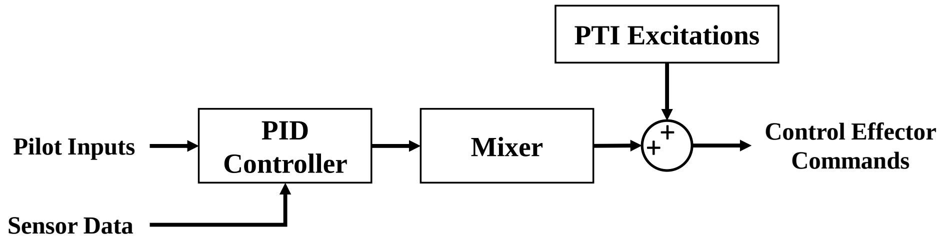


Figure 11: Control architecture overview including PTI excitations.

A decision structure was implemented in the Simulink® model to run different flight modes within the same custom firmware to enable PID gain tuning and PTI injection. The flight modes, listed in Table 2, included controller gain tuning and PTI input injection functionality, as well as normal flight. The parameter "CAM_CAP_FBACK" in QGC was used to change the flight mode by specifying an integer between 1 and 5 in the ground control station software. The CAM_CAP_FBACK parameter was chosen because, as configured for this study, its value did not affect the PX4 Autopilot in flight. The value of CAM_CAP_FBACK was read into the Simulink® model using the Read Parameter block (Figure 2f), which was then used to determine which flight mode to enable.

Table 2: Programmed CL-84 aircraft flight modes.

Value	Action
1	Roll Axis Tuning
2	Pitch Axis Tuning
3	Yaw Axis Tuning
4	PTI Injection
5	Normal Flight

The first three flight modes specified by CAM_CAP_FBACK were used to tune controller gains for the roll, pitch, and yaw response, respectively, using the knobs on the RC transmitter. Once an axis was tuned, another axis could be selected and tuned by changing the CAM_CAP_FBACK parameter value. The fourth flight mode enabled the ability to send PTI excitations to the motors and actuators for system identification. The system identification approach for the CL-84 aircraft would follow a similar strategy to the eVTOL aircraft system identification approach discussed in Reference [11]. The PTI excitations were orthogonal phase-optimized multisine inputs, described in further detail in References [12–15]. The designed multisine input spectra for each CL-84 control effector is shown in Figure 12 (shown as the power fraction P_k, where the sum of all P_k for each input is equal to 1). Figure 13 shows the first 20 seconds of the input excitation signals with an amplitude of one. This reflects how the signals are injected into the flight controller, where a gain is applied to scale each input signal to a sufficient amplitude to obtain an adequate signal-to-noise ratio. The transmitter was used both to enable or disable PTI injections and adjust the amplitude of the PTI injections. The fifth flight mode was for normal flight using fixed controller gain values without the ability to send PTI excitations.

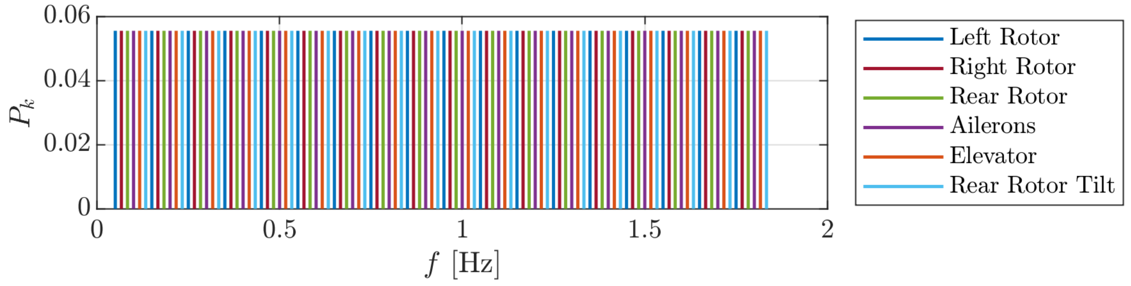


Figure 12: CL-84 aircraft multisine input spectra for each control effector.

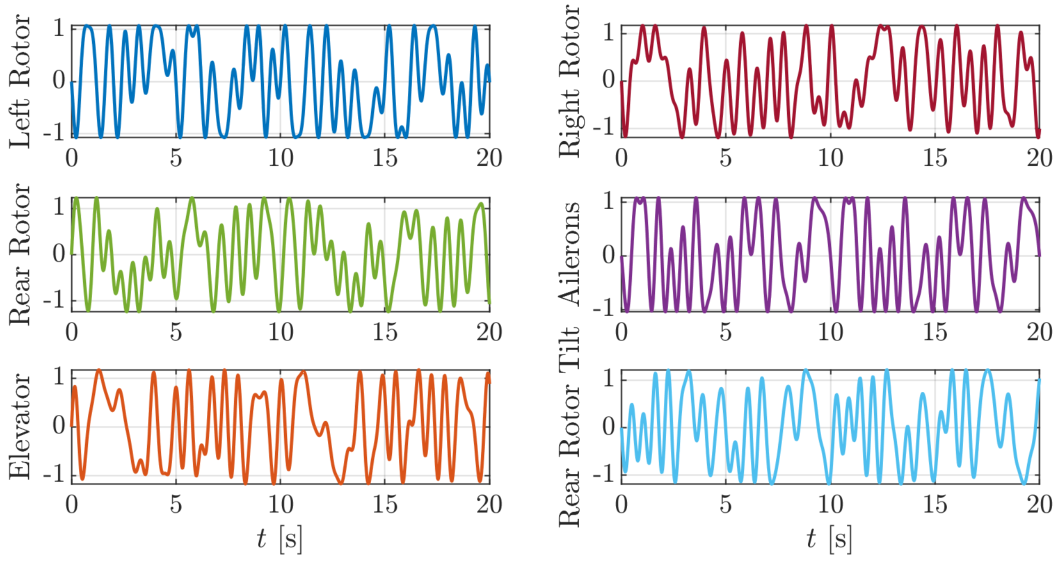


Figure 13: CL-84 aircraft normalized multisine inputs.

For the purpose of this investigation, the goals of deploying a custom flight controller via embedded firmware, tuning controller gains, and demonstrating a PTI injection capability were accomplished. However, ideas for future improvement for the controller tuning logic will be discussed in Section 5.2.

4.5 PWM Output

The "PWM Output" group (shown in the lower right of Figure 8) sent PWM signals in microseconds to the different channels on the Main output servo rail of the Pixhawk to command motor and control surface actuation. Channels were selected from the Main group, and signals were routed based on the hardware setup described in Table B1. The arming logic described in Section 4.3 was used for the `boolean` arming signal that allowed signals to be sent to the motors and control surfaces. Note that the PWM output signals must be converted to the `uint16` datatype before being sent to the PWM output block.

4.6 Preliminary Flight Testing

Preliminary flight testing was conducted in an indoor, netted multirotor testing facility to verify the controller gain tuning and PTI injection capabilities. The testing demonstrated the ability to tune the controller using the transmitter knobs and the ability to inject PTI excitations for system identification. This exercise also demonstrated the streamlined nature of the SUPPQ toolchain, which allowed for rapid modifications of the custom controller firmware based on the aircraft response observed during flight. The control logic could be modified and then re-deployed as new custom firmware running on the flight computer for testing in only a few minutes, which verified its utility for rapid custom controller deployment. Figure 14 shows the CL-84 UAV on a tether in the indoor, netted testing facility.

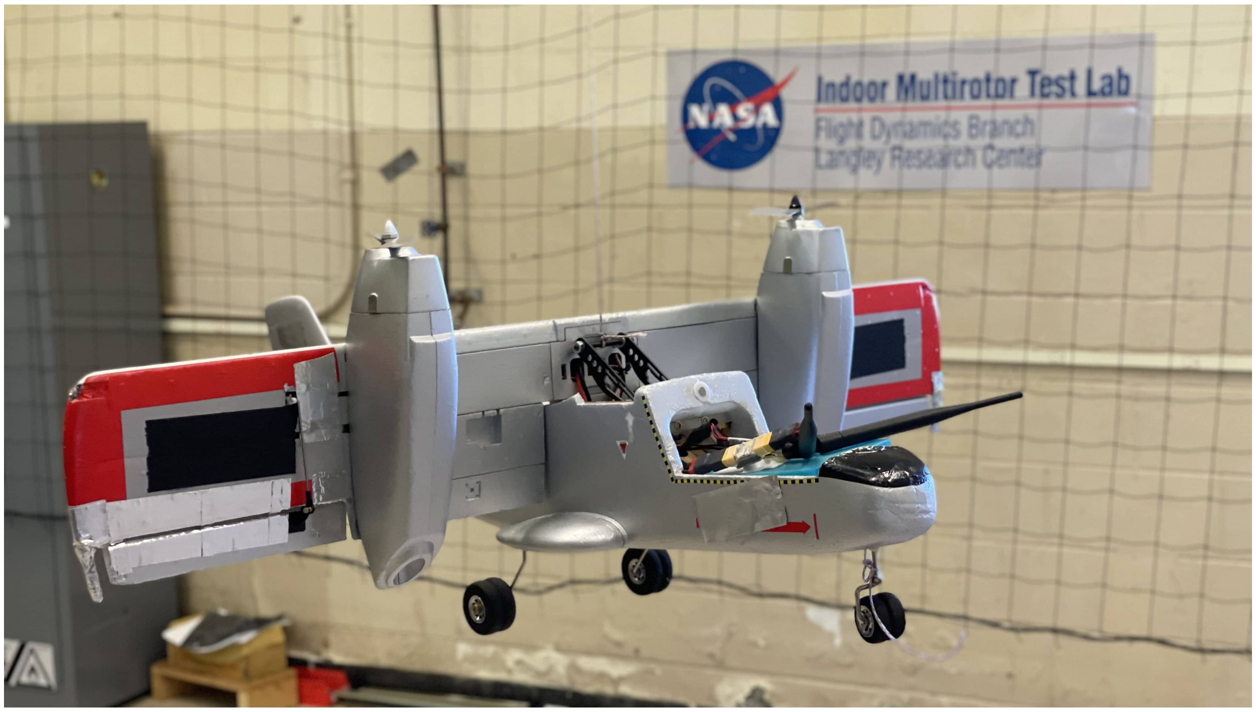


Figure 14: CL-84 UAV on a tether in the testing facility.

5 Future Investigation

This section discusses select topics for future study using the SUPPQ toolchain described in this report.

5.1 Pixhawk Hardware Improvements

The flight logs consistently showed relatively high CPU and RAM usage onboard the Pixhawk Cube Black used for this study (e.g., see the CPU and RAM display shown in Figure 15). The control logic used for this study was relatively simple and efficient, and was run using a modest 0.02-second fixed step size, which was set under the Solver tab of the Hardware Settings menu in Simulink®. It is suspected that the high RAM and CPU usage observed in this study is related to the Pixhawk Cube Black tested in this report. Newer Pixhawk hardware, such as the CubePilot Cube Orange [16], Holybro Pixhawk 4 [17], and Holybro Pixhawk 5X [18] provide increased RAM and CPU specifications which should better accommodate the computational resources of the PX4 firmware and flight controller in future studies. The specifications of different Pixhawk hardware available as of July 2022 are shown in Table 3. Note that the Cube Orange is not supported as of the R2022a version of the UAV Toolbox, but it is expected that the Cube Orange will be supported in future MATLAB® releases.

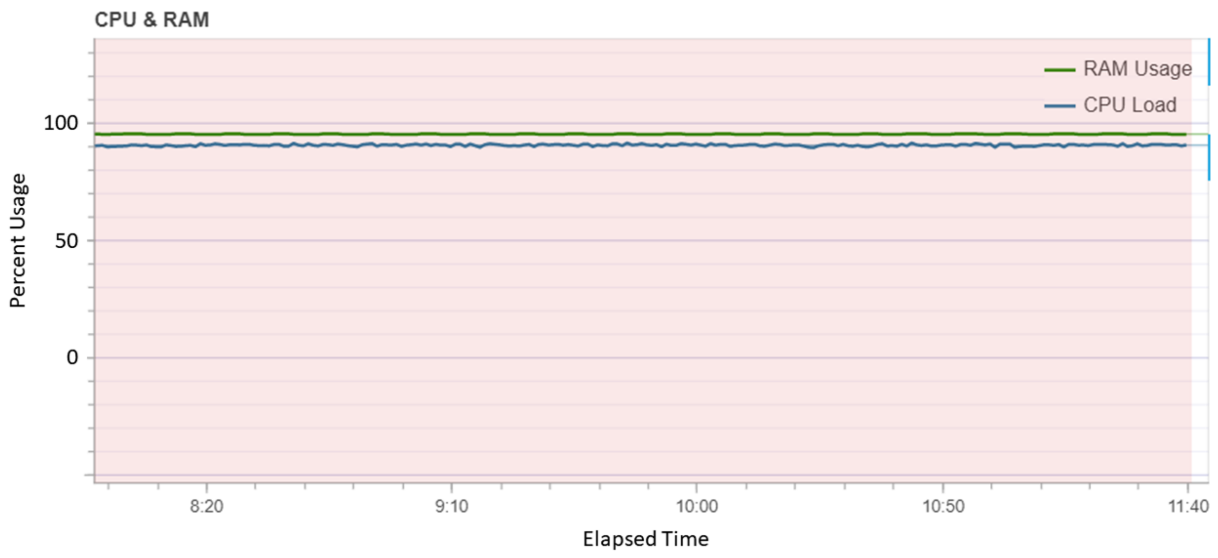


Figure 15: CPU and RAM usage on the Pixhawk Cube Black.

Table 3: Specifications for hardware running the PX4 autopilot firmware (as of July 2022).

Flight Controller Hardware	CPU	CPU Clock Speed	RAM
Pixhawk 1 [19]	STM32F427 ARM Cortex-M4	168 MHz	256 KB
Hex Cube Black* [4]	STM32F427 ARM Cortex-M4F	168 MHz	256 KB
Holybro Pixhawk 4 [17]	STM32F765 ARM Cortex-M7	216 MHz	512 KB
Holybro Pixhawk 5X [18]	STM32F765 ARM Cortex-M7	216 MHz	512 KB
CubePilot Cube Orange** [16]	STM32H743ZI ARM Cortex-M7	400 MHz	1 MB

*Tested in this report
**Not supported by the UAV Toolbox as of MATLAB® R2022a

5.2 Improved Parameter Tuning

The procedure for PID gain tuning could be improved by using additional parameter values sent using the telemetry connection managed by QGC and enhanced gain tuning logic within the Simulink® model. Parameters sent via the telemetry connection could trigger the model to save the current tuning values established by the transmitter knobs as the default gain values for that axis or update the gain values directly using parameters available within QGC. Improving the model with the ability to save the tuned values as new default values between flights would also allow for improved gain tuning procedures.

5.3 Logging Data within the Controller

The ability to log internal signals within a Simulink® model would be useful for control law development and certification testing. Current PX4 ULog data logs record specific predefined topics, as described in Appendix F.2. However, certain important internal signals, such as gains that are being tuned in a custom controller using the RC transmitter knob, are not logged directly. MathWorks® documentation for onboard MAT-file logging using the SD card shows a promising method for logging data within the custom controller [20].

6 Conclusions

This report explored the functionality of the MathWorks® UAV Toolbox and the associated PX4 support package to build and deploy custom firmware onto a Pixhawk flight computer onboard a UAV. A rapid custom control law integration toolchain was developed for a model CL-84 UAV, which was successfully demonstrated in bench testing and initial flight testing. The toolchain documented in this report bridges the gap between flight control law development and hardware integration onto increasingly capable low-cost, unmanned aircraft, providing the ability to rapidly flight test and refine custom flight controllers for various applications.

Acknowledgments

This research was funded by the NASA Aeronautics Research Mission Directorate (ARMD) Transformational Tools and Technologies (TTT) project. The authors gratefully acknowledge flight control system development support provided by Jacob Cook, as well as additional research support provided by George Altamirano, Mario Smith, Brian Duvall, Rachel Axten, Stephen Farrell, Ronald Busan, and Stephen Riddick.

References

1. "UAV Toolbox," https://www.mathworks.com/products/uav.html, Accessed 03 August 2022.

2. "UAV Toolbox Support Package for PX4 Autopilots," https://www.mathworks.com/help/supportpkg/px4/index.html, Accessed 03 August 2022.

3. "QGroundControl," http://qgroundcontrol.com/, Accessed 04 August 2022.

4. "Hex Cube Black Flight Controller," https://docs.px4.io/v1.12/en/flight_controller/pixhawk-2.html, Accessed 26 July 2022.

5. "PX4 Autopilot User Guide," https://docs.px4.io/master/en/, Accessed 04 June 2022.

6. "Prearm, Arm, Disarm Configuration," *PX4 User Guide*, https://docs.px4.io/v1.12/en/advanced_config/prearm_arm_disarm.html, Accessed 26 August 2022.

7. "Using the ECL EKF," *PX4 User Guide*, https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html, Accessed 06 August 2022.

8. Riseborough, P., "PX4 State Estimation Overview," https://www.youtube.com/watch?v=HkYRJJoyBwQ, Jun 2019. Accessed 06 August 2022.

9. "uORB Messaging," *PX4 User Guide*, https://docs.px4.io/main/en/middleware/uorb.html, Accessed 26 August 2022.

10. Gresham, J. L., Fahmi, J.-M. W., Simmons, B. M., Hopwood, J. W., Foster, W., and Woolsey, C. A., "Flight Test Approach for Modeling and Control Law Validation for Unmanned Aircraft," *AIAA SciTech 2022 Forum*, AIAA Paper 2022-2406, Jan. 2022. https://doi.org/10.2514/6.2022-2406.

11. Simmons, B. M., "System Identification for eVTOL Aircraft Using Simulated Flight Data," *AIAA SciTech 2022 Forum*, AIAA Paper 2022-2409, Jan. 2022. https://doi.org/10.2514/6.2022-2409.

12. Morelli, E. A., and Klein, V., *Aircraft System Identification: Theory and Practice*, 2nd ed., Sunflyte Enterprises, Williamsburg, VA, 2016.

13. Morelli, E. A., "Multiple Input Design for Real-Time Parameter Estimation in the Frequency Domain," *13th IFAC Conference on System Identification*, Aug. 2003. https://doi.org/10.1016/S1474-6670(17)34833-4.

14. Morelli, E. A., "Flight-Test Experiment Design for Characterizing Stability and Control of Hypersonic Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 3, 2009, pp. 949–959. https://doi.org/10.2514/1.37092.

15. Morelli, E. A., "Practical Aspects of Multiple-Input Design for Aircraft System Identification Flight Tests," *AIAA AVIATION 2021 Forum*, AIAA Paper 2021-2795, Aug. 2021. https://doi.org/10.2514/6.2021-2795.

16. "Cube Orange Flight Controller," https://docs.px4.io/v1.12/en/flight_controller/cubepilot_cube_orange.html, Accessed 26 July 2022.

17. "Pixhawk 4," http://docs.px4.io/main/en/flight_controller/pixhawk4.html, Accessed 28 July 2022.

18. "Holybro Pixhawk 5X," https://docs.px4.io/main/en/flight_controller/pixhawk5x.html, Accessed 28 July 2022.

19. "mRo Pixhawk Flight Controller (Pixhawk 1)," https://docs.px4.io/v1.11/en/flight_controller/mro_pixhawk.html, Accessed 28 July 2022.

20. "MAT-file Logging on SD Card for PX4 Autopilots," https://www.mathworks.com/help/supportpkg/px4/ref/px4-matfile-logging-example.html, Accessed 27 July 2022.

21. "Radio Control Systems," *PX4 User Guide*, https://docs.px4.io/main/en/getting_started/rc_transmitter_receiver.html, Accessed 26 August 2022.

22. "Setting Up Cygwin Toolchain and Downloading PX4 Source Code," https://www.mathworks.com/help/supportpkg/px4/ug/setup-cygwin-toolchain.html, Accessed 09 June 2022.

23. "ULog File Format," *PX4 User Guide*, https://docs.px4.io/main/en/dev_log/ulog_file_format.html, Accessed 27 August 2022.

24. "NSL@VT GitHub," https://github.com/NSL-VT, Accessed 05 August 2022.

Appendix A

Materials List

The hardware, software, and firmware used to conduct this study are listed below.

A.1 Hardware

- Pixhawk 2.1 (Cube Black)

> **Note:** As of July 2022 the Cube Black is the only autopilot in the Pixhawk Cube line of products supported by the UAV Toolbox in MATLAB®. Initial attempts to use a Cube Orange were not successful because it was not yet supported at the time of publication.

- Laptop computer running Windows 10
- Here3 GPS
- Safety Switch
- Servo Cables
- Futaba R6014 HS Receiver
- Futaba T18MZ Remote Control
- Micro SD Card
- PPM Encoder
- Power supply (for bench testing)
- Lithium polymer (LiPo) battery (for flight testing)
- Pixhawk 2.1 Power Module
- Model CL-84 UAV
- RFD900+ Telemetry Bundle

The hardware assembly used in this study can be adapted to user needs and depends on the components being used. Figure A1 shows the components placed in the front of the CL-84 UAV including the battery compartment, power distribution module, and telemetry. Figure A2 shows the components behind the wing of the CL-84 UAV including the Pixhawk flight computer and associated connections.

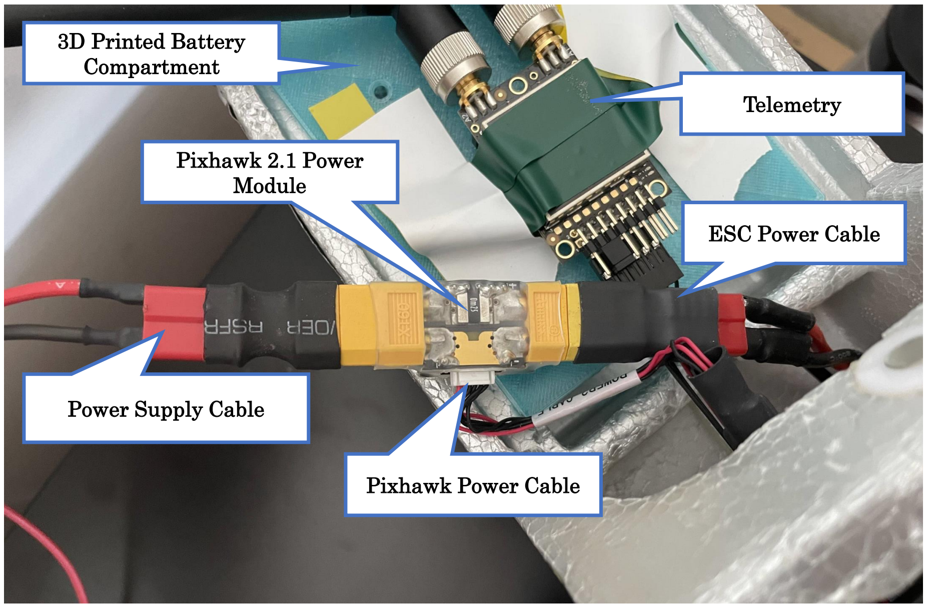


Figure A1: Labeled components in the front of the CL-84.

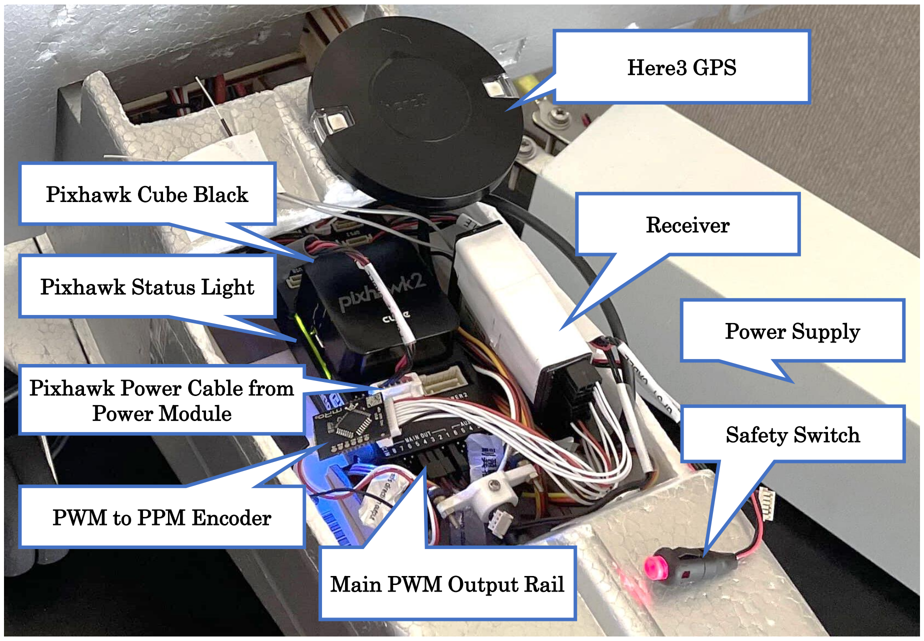


Figure A2: Labeled components in the middle of the CL-84.

A.2 Software

- QGroundControl (QGC)
- MATLAB®
- Simulink®
- UAV Toolbox
- UAV Toolbox Support Package for PX4 Autopilots

- MATLAB® Coder
- Simulink® Coder
- Embedded Coder
- MATLAB® Support for MinGW-w64 C/C++ Compiler

A.3 Firmware

- PX4 Firmware installed using the Cygwin Toolchain

Appendix B

Hardware Setup

Setting up hardware properly is critical to the success of connected simulations, bench testing, and embedding firmware. The following sections contain helpful information for successful and safe Pixhawk setup to allow testing custom Simulink® models.

B.1 Port Configuration

Table B1 shows the channels used for communication with the model CL-84 UAV.

Table B1: Pixhawk servo rail connections for the CL-84 UAV.

Port Label	Connection
RC IN	Receiver via PPM Encoder
Main 8	Rear Motor Rotation
Main 7	-
Main 6	-
Main 5	Left Motor
Main 4	Rear Motor
Main 3	Right Motor
Main 2	Elevator Servo
Main 1	Aileron Servos
Aux 6	-
Aux 5	-
Aux 4	-
Aux 3	-
Aux 2	-
Aux 1	Power to Receiver

Table B2 shows the hardware connected to the different serial ports available on the Pixhawk Cube Black.

Table B2: Pixhawk serial port connections for the CL-84 UAV.

Port Label	Connection
CAN1	Here 3 GPS
GPS	Safety Switch
TELEM1	Telemetry
Power1	Pixhawk 2.1 Power Module

B.2 Power Supply

A power supply was used for bench testing and controller validation. The Pixhawk 2.1 Power Module connects to the power supply and sends power to the ESCs and Pixhawk onboard the UAV. The power supply was set to run at 14.96 volts with a 5.03 amp limit, which was adequate for initial setup, bench testing, and Connected I/O model validation. Setting up an adequate power supply is crucial—without adequate power distribution from the power supply to the Pixhawk and ESCs, hardware-in-the-loop simulations will fail.

B.3 Rebooting the Pixhawk

Rebooting the Pixhawk is necessary during the firmware flashing with QGC and Simulink®. The best procedure for rebooting the Pixhawk during firmware updates is:

1. Unplug the Pixhawk USB connection from the computer.
2. Turn off the UAV power supply output for a few seconds.
3. Turn on the UAV power supply output.
4. Reconnect the Pixhawk to the computer via USB.

Because Micro USB cables vary greatly in terms of data throughput and power output capabilities, this procedure tends to work the best with connected hardware to ensure a proper reboot.

B.4 Propeller Safety

It is best practice to remove propellers from the aircraft during bench testing and firmware flashing in case of power supply or data connectivity errors. Note that running the motors without propellers for extended periods of time can damage the motors. Thus, it is recommended to only run the motors for short amounts of time without propellers.

B.5 Receiver Limitations

The receiver used during testing was capable of receiving 12 channels from the transmitter. However, the Futaba protocol receiver could only send PWM signals to the Pixhawk. Consequently, a pulse position modulation (PPM) encoder was used to convert these PWM signals to signals compatible with the RC-In port on the Pixhawk, but the PWM to PPM encoder was limited to 7 PWM channels. For future testing, it is strongly recommended to use a PPM encoder capable of encoding more than 7 PWM channels. Alternatively, using a transmitter and receiver pair that is capable of 8-channels or more via SBUS output connection from the receiver to the Pixhawk will allow for the integration of more transmitter channels (such as additional knobs and switches) into the controller. Most FrSky protocol receivers have the SBUS output integrated, enabling the use of all available transmitter channels. The PX4 documentation lists additional PX4 compatible transmitter/receiver pairs [21]. Further applications of additional RC channels include improved controller tuning and programmed test input injection capabilities.

B.6 Telemetry

The telemetry antenna can be connected to the Pixhawk during setup and bench testing. However, leaving the receiving antennas disconnected from the ground station helped to avoid port confusion when flashing firmware to the Pixhawk. The telemetry information is useful after embedding custom firmware using Build, Deploy and Start. The wireless telemetry connection to the vehicle enables calibrating various sensors, updating parameters, and viewing heading information in QGroundControl when the vehicle is powered on.

Appendix C

Software Setup

The following sections describe the setup procedures for the software used in the SUPPQ toolchain.

C.1 QGroundControl (QGC) Setup

The following procedures were used to setup QGroundControl.

1. Download QGroundControl (QGC) [3] and follow the standard installation steps.
2. When finished installing, connect the Pixhawk to the computer via USB.
3. Flash the desired version of the PX4 firmware to the Pixhawk. (The fixed-wing airframe was used for initial setup in this work.)
4. Proceed through the setup steps for the Pixhawk, including all necessary calibrations.
5. Disable the circuit breaker for airspeed calibration.
   - Set: `CBRK_AIRSPD_CHK` to 162128
6. Complete the radio calibration.
7. After completing the remaining sensor calibrations, change the following parameters for safety switch configuration:
   - `COM_PREARM_MODE` = 0 (Disabled)
   - `CBRK_IO_SAFETY` = 22027

   These parameter changes ensure that the safety switch is included in the loop when arming the Pixhawk from Simulink®.
8. Set all flight modes to "manual" to prevent interference from the pre-existing fixed-wing controller.

The following procedures were used to enable the Here 3 GPS for use with the Pixhawk Cube Black.

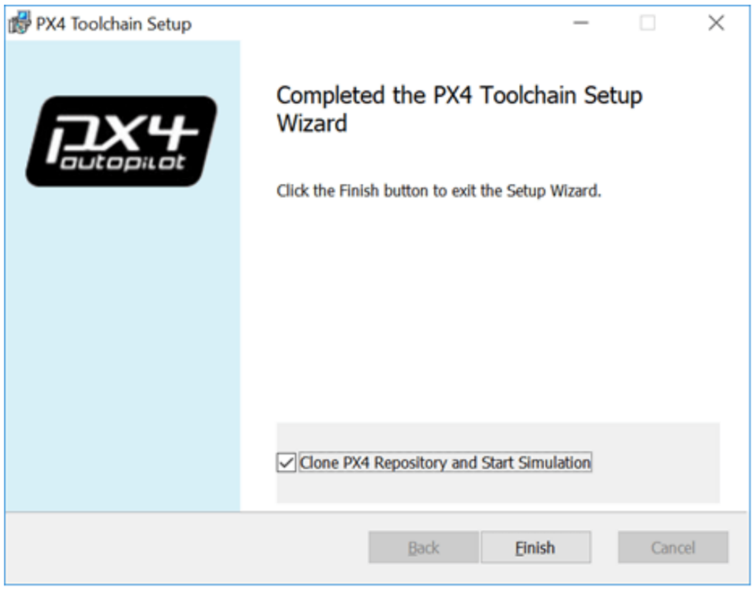
1. Plug the Here3 GPS into the CAN1 port on the Pixhawk.
2. Within QGC, set the parameter "UAVCAN_Enable" to "Sensor Automatic Config" to connect the Here3 GPS.
3. Ensure that the Pixhawk can be armed through QGC after all parameter changes are made.
4. After successfully arming the Pixhawk, disarm the Pixhawk, and close QGC.

Refer to Appendix E.1 for helpful troubleshooting procedures.

C.2 PX4 Firmware Repository Setup

The following procedures were used to setup the Cygwin toolchain that houses the PX4 firmware compatible with the SUPPQ toolchain.

1. Open the MathWorks® documentation for the Cygwin Toolchain (Reference [22]).
2. Download the .msi file: "PX4.Windows.Cygwin.Toolchain.0.8.msi"
3. Proceed through the setup steps in the PX4 Toolchain Setup Wizard.
4. At the end of the setup windows, ensure that "Clone PX4 Repository and Start Simulation" is checked, as shown in Figure C1.


PX4 Toolchain Setup

Completed the PX4 Toolchain Setup Wizard

Click the Finish button to exit the Setup Wizard.

☑ Clone PX4 Repository and Start Simulation

Figure C1: PX4 Toolchain Setup Wizard completion window.

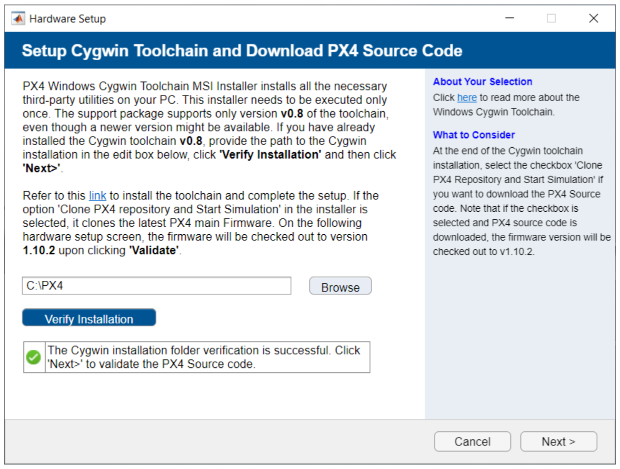
5. A bash shell should open and the PX4 firmware will begin cloning.
6. Upon successful PX4 firmware download, a simulation will begin and activity will cease in the bash shell.

C.3 PX4 Autopilot Hardware Setup from Simulink®

The following procedures were used to setup the PX4 Autopilot hardware to work with Simulink®.

Note: MATLAB®, Simulink®, Embedded Coder, MATLAB® Coder, Simulink® Coder, UAV Toolbox, UAV Toolbox Support Package for PX4 Autopilots, and MATLAB® Support for MinGW-w64 C/C++ Compiler should be installed before proceeding with the following procedures.

1. In MATLAB®, under the home tab, navigate to the "Add-Ons" drop down and select "Manage Add-Ons."
2. Select the gear icon in the Add-On Manager next to the UAV Toolbox Support Package for PX4 Autopilots. This will open the Hardware Setup wizard.
3. Verify the installation of Cygwin installation folder, as shown in Figure C2.
4. Verify that the PX4 Firmware is compatible with Simulink®, as shown in Figure C3.


Hardware Setup

Setup Cygwin Toolchain and Download PX4 Source Code

PX4 Windows Cygwin Toolchain MSI Installer installs all the necessary third-party utilities on your PC. This installer needs to be executed only once. The support package supports only version **v0.8** of the toolchain, even though a newer version might be available. If you have already installed the Cygwin toolchain **v0.8**, provide the path to the Cygwin installation in the edit box below, click **'Verify Installation'** and then click **'Next>'**.

Refer to this link to install the toolchain and complete the setup. If the option 'Clone PX4 repository and Start Simulation' in the installer is selected, it clones the latest PX4 main Firmware. On the following hardware setup screen, the firmware will be checked out to version **1.10.2** upon clicking **'Validate'**.

C:\PX4

✓ The Cygwin installation folder verification is successful. Click 'Next>' to validate the PX4 Source code.

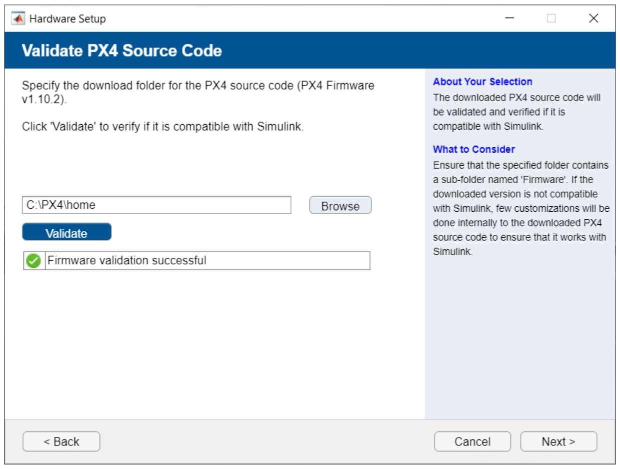
About Your Selection

Click here to read more about the Windows Cygwin Toolchain.

What to Consider

At the end of the Cygwin toolchain installation, select the checkbox 'Clone PX4 Repository and Start Simulation' if you want to download the PX4 Source code. Note that if the checkbox is selected and PX4 source code is downloaded, the firmware version will be checked out to v1.10.2.

Figure C2: Screenshot of the Cygwin toolchain setup window.


Hardware Setup

Validate PX4 Source Code

Specify the download folder for the PX4 source code (PX4 Firmware v1.10.2).

Click 'Validate' to verify if it is compatible with Simulink.

C:\PX4\home

✓ Firmware validation successful

About Your Selection

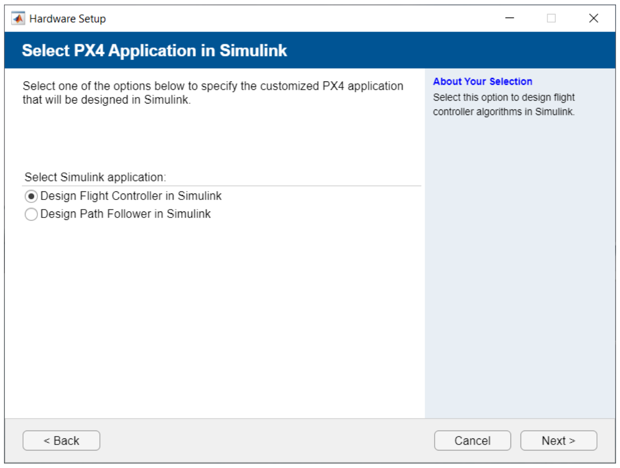
The downloaded PX4 source code will be validated and verified if it is compatible with Simulink.

What to Consider

Ensure that the specified folder contains a sub-folder named 'Firmware'. If the downloaded version is not compatible with Simulink, few customizations will be done internally to the downloaded PX4 source code to ensure that it works with Simulink.

Figure C3: Screenshot of the PX4 source code validation window.

5. Select the "Design Flight Controller in Simulink®" option, as shown in Figure C4.
6. Select the proper PX4 Autopilot and build target firmware (the options selected for this study are shown in Figure C5).


Hardware Setup

Select PX4 Application in Simulink

Select one of the options below to specify the customized PX4 application that will be designed in Simulink.

Select Simulink application:

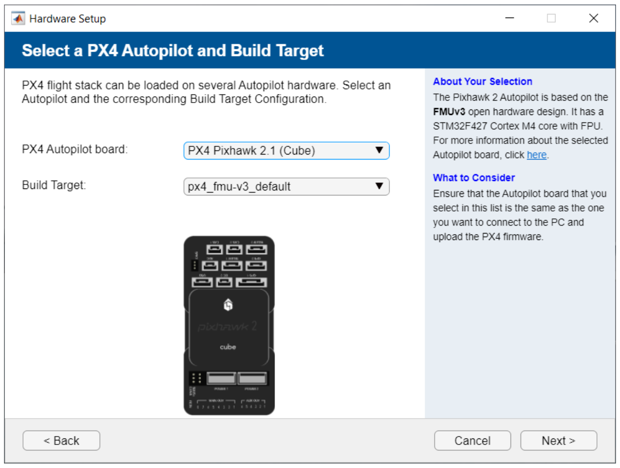
[✓ Design Flight Controller in Simulink]

Design Path Follower in Simulink

About Your Selection

Select this option to design flight controller algorithms in Simulink.

Figure C4: Screenshot of the PX4 application selection window.


Hardware Setup

Select a PX4 Autopilot and Build Target

PX4 flight stack can be loaded on several Autopilot hardware. Select an Autopilot and the corresponding Build Target Configuration.

PX4 Autopilot board: PX4 Pixhawk 2.1 (Cube)

Build Target: px4_fmu-v3_default

About Your Selection

The Pixhawk 2 Autopilot is based on the **FMUv3** open hardware design. It has a STM32F427 Cortex M4 core with FPU. For more information about the selected Autopilot board, click here.

What to Consider

Ensure that the Autopilot board that you select in this list is the same as the one you want to connect to the PC and upload the PX4 firmware.

Figure C5: Screenshot of the PX4 board and build target selection window.

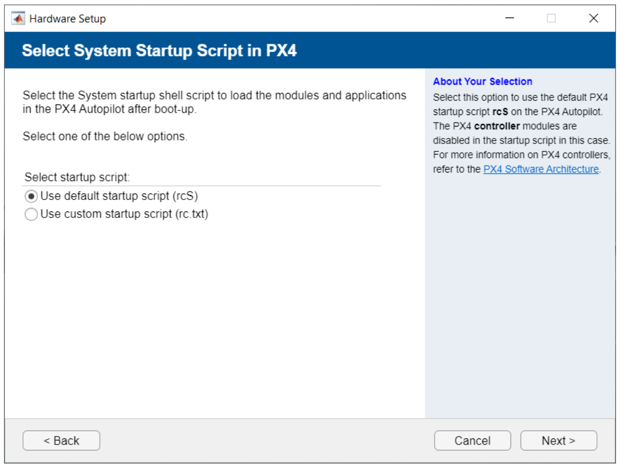
7. Choose whether to use the default startup script (rcS) or a custom startup script (rc.txt). For this study, the default startup script (rcS) was chosen, as shown in Figure C6.

Note: As of the version used in this procedure (R2021b), changes to the startup sequence desired by the user are able to be completed using the Configuration Parameters under the Model Settings in Simulink®. These changes will take affect with the specified firmware coders installed.

Note: A custom startup sequence can be placed on the SD card on the Pixhawk at any time. The SD card must be read by removing it from the Pixhawk and connecting it to the computer separately from the PX4 autopilot. Any "rc.txt" located in the root "etc" folder on the SD card will be run instead of the default startup script (rcS) when the Pixhawk is next powered on.

Note: No custom startup sequence was necessary in this study to deploy a custom controller using the MATLAB® 2021b version of UAV Toolbox.

8. Verify that QGroundControl is installed, as shown in Figure C7.



Hardware Setup

Select System Startup Script in PX4

Select the System startup shell script to load the modules and applications in the PX4 Autopilot after boot-up.

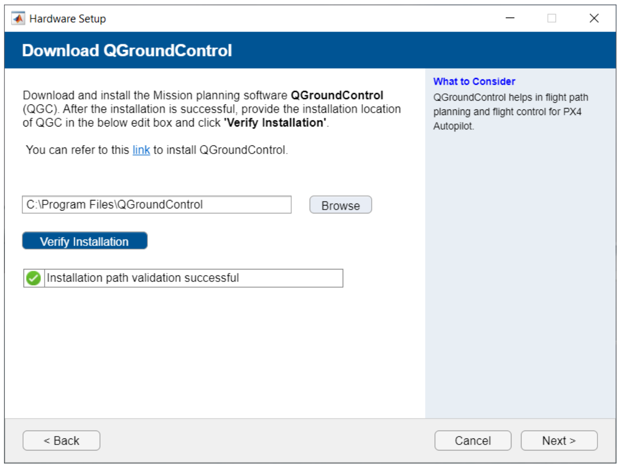
Select one of the below options.

Select startup script:
- [✓ Use default startup script (rcS)]
- Use custom startup script (rc.txt)

About Your Selection

Select this option to use the default PX4 startup script **rcS** on the PX4 Autopilot. The PX4 **controller** modules are disabled in the startup script in this case. For more information on PX4 controllers, refer to the PX4 Software Architecture.

Figure C6: Screenshot of the system startup script selection window.



Hardware Setup

Download QGroundControl

Download and install the Mission planning software **QGroundControl** (QGC). After the installation is successful, provide the installation location of QGC in the below edit box and click '**Verify Installation**'.

You can refer to this link to install QGroundControl.

C:\Program Files\QGroundControl

Installation path validation successful

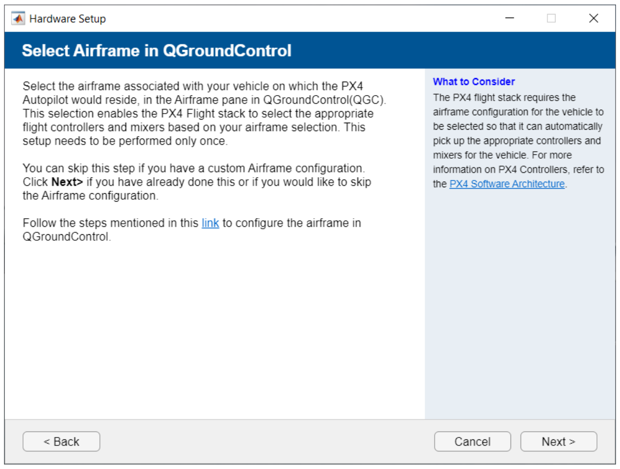
What to Consider

QGroundControl helps in flight path planning and flight control for PX4 Autopilot.

Figure C7: Screenshot of the QGroundControl verification window.

9. As shown in Figure C8, the next screen asks the user to select an airframe in QGroundControl. Selecting an airframe was completed during QGroundControl setup in Appendix C.1. Continue to the next step.
10. Start the firmware build. A green checkmark will appear when done, as shown in Figure C9.

Note: Progress and error codes can be viewed in the command window of MATLAB®.


Hardware Setup

Select Airframe in QGroundControl

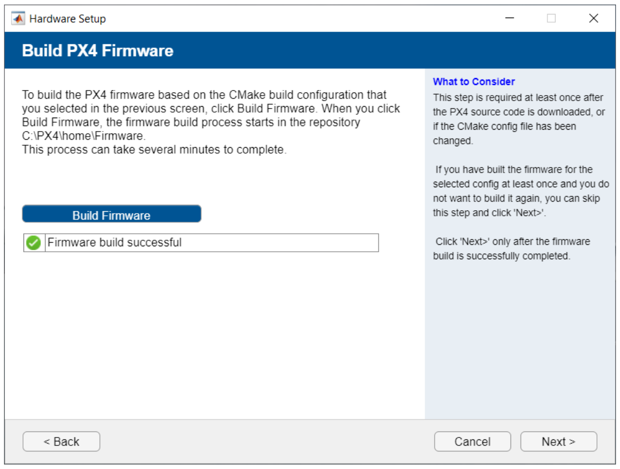
Select the airframe associated with your vehicle on which the PX4 Autopilot would reside, in the Airframe pane in QGroundControl(QGC). This selection enables the PX4 Flight stack to select the appropriate flight controllers and mixers based on your airframe selection. This setup needs to be performed only once.

You can skip this step if you have a custom Airframe configuration. Click **Next>** if you have already done this or if you would like to skip the Airframe configuration.

Follow the steps mentioned in this link to configure the airframe in QGroundControl.

What to Consider

The PX4 flight stack requires the airframe configuration for the vehicle to be selected so that it can automatically pick up the appropriate controllers and mixers for the vehicle. For more information on PX4 Controllers, refer to the PX4 Software Architecture.


Hardware Setup

Build PX4 Firmware

To build the PX4 firmware based on the CMake build configuration that you selected in the previous screen, click Build Firmware. When you click Build Firmware, the firmware build process starts in the repository C:\PX4\home\Firmware.
This process can take several minutes to complete.

Firmware build successful

What to Consider

This step is required at least once after the PX4 source code is downloaded, or if the CMake config file has been changed.

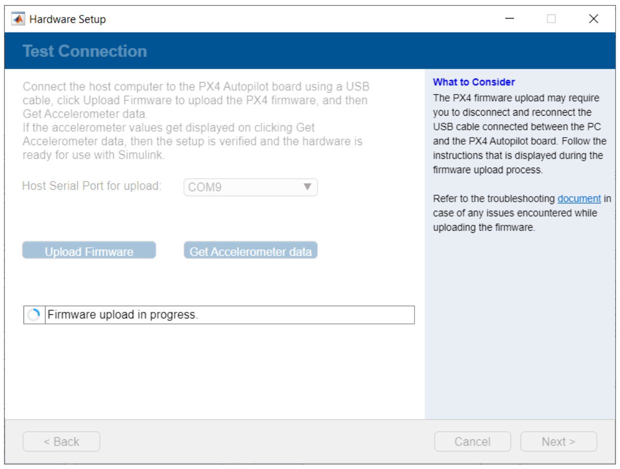
If you have built the firmware for the selected config at least once and you do not want to build it again, you can skip this step and click 'Next>'.

Click 'Next>' only after the firmware build is successfully completed.

Figure C8: Screenshot of the QGC airframe selection window.

Figure C9: Screenshot of the build PX4 firmware window.

11. On the Test Connection screen (Figure C10), choose the serial port where the bootloader is connected.
12. Upload firmware, as shown in Figure C11, and view the MATLAB® command window for progress.


Hardware Setup

Test Connection

Connect the host computer to the PX4 Autopilot board using a USB cable, click Upload Firmware to upload the PX4 firmware, and then Get Accelerometer data.
If the accelerometer values get displayed on clicking Get Accelerometer data, then the setup is verified and the hardware is ready for use with Simulink.

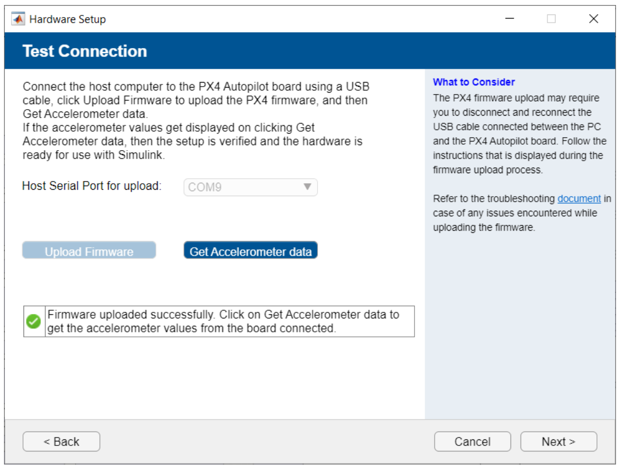
Host Serial Port for upload: COM9

Firmware upload in progress.

What to Consider

The PX4 firmware upload may require you to disconnect and reconnect the USB cable connected between the PC and the PX4 Autopilot board. Follow the instructions that is displayed during the firmware upload process.

Refer to the troubleshooting document in case of any issues encountered while uploading the firmware.


Hardware Setup

Test Connection

Connect the host computer to the PX4 Autopilot board using a USB cable, click Upload Firmware to upload the PX4 firmware, and then Get Accelerometer data.
If the accelerometer values get displayed on clicking Get Accelerometer data, then the setup is verified and the hardware is ready for use with Simulink.

Host Serial Port for upload: COM9

Firmware uploaded successfully. Click on Get Accelerometer data to get the accelerometer values from the board connected.

What to Consider

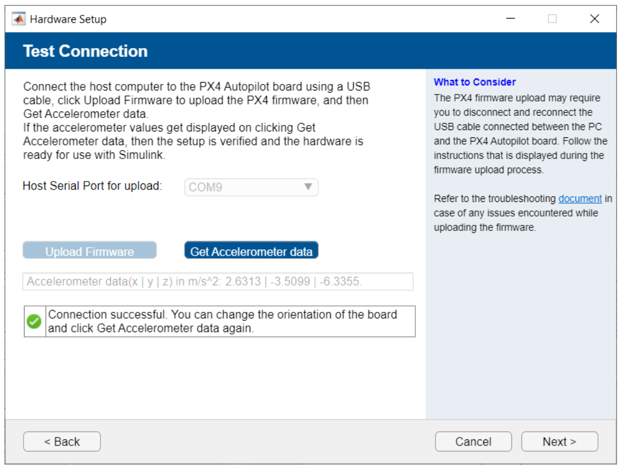
The PX4 firmware upload may require you to disconnect and reconnect the USB cable connected between the PC and the PX4 Autopilot board. Follow the instructions that is displayed during the firmware upload process.

Refer to the troubleshooting document in case of any issues encountered while uploading the firmware.

Figure C10: Screenshot of the firmware upload in progress window.

Figure C11: Screenshot of the firmware upload success window.

13. Click "Get Accelerometer data" to test the connection, as shown in Figure C12.
14. The final screen in Hardware Setup will show that the setup process is complete. The option to view UAV Toolbox examples is displayed upon completion (see Figure C13).


Hardware Setup

Test Connection

Connect the host computer to the PX4 Autopilot board using a USB cable, click Upload Firmware to upload the PX4 firmware, and then Get Accelerometer data.
If the accelerometer values get displayed on clicking Get Accelerometer data, then the setup is verified and the hardware is ready for use with Simulink.

Host Serial Port for upload: COM9

Accelerometer data(x | y | z) in m/s^2: 2.6313 | -3.5099 | -6.3355

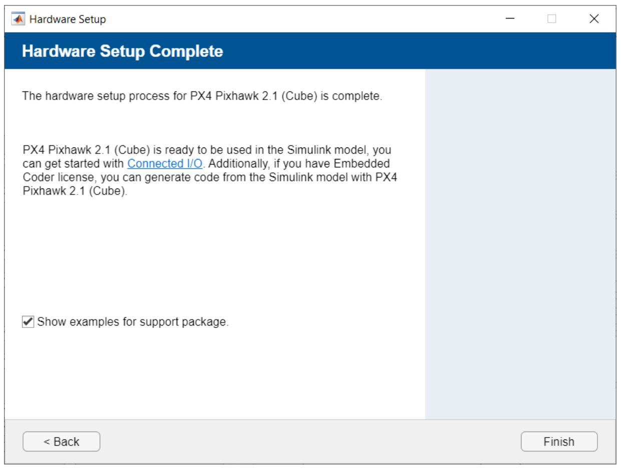
Connection successful. You can change the orientation of the board and click Get Accelerometer data again.

What to Consider

The PX4 firmware upload may require you to disconnect and reconnect the USB cable connected between the PC and the PX4 Autopilot board. Follow the instructions that is displayed during the firmware upload process.

Refer to the troubleshooting document in case of any issues encountered while uploading the firmware.

Figure C12: Screenshot of the test hardware connection window.


Hardware Setup

Hardware Setup Complete

The hardware setup process for PX4 Pixhawk 2.1 (Cube) is complete.

PX4 Pixhawk 2.1 (Cube) is ready to be used in the Simulink model, you can get started with Connected I/O. Additionally, if you have Embedded Coder license, you can generate code from the Simulink model with PX4 Pixhawk 2.1 (Cube).

[✓] Show examples for support package.

Figure C13: Screenshot of the hardware setup complete window.

This concludes the setup process for the Pixhawk. All software specified in Appendix A.2 has been installed and the SUPPQ toolchain for developing custom firmware for the PX4 is ready for use.

Note: The installation of UAV Toolbox comes with example models including quadcopter controllers and logger demonstrations. These models are available in the folder shown below.

File path:
`C:/ProgramData/MATLAB/SupportPackages/R2021b/toolbox/target/supportpackages/px4/px4examples`

Appendix D

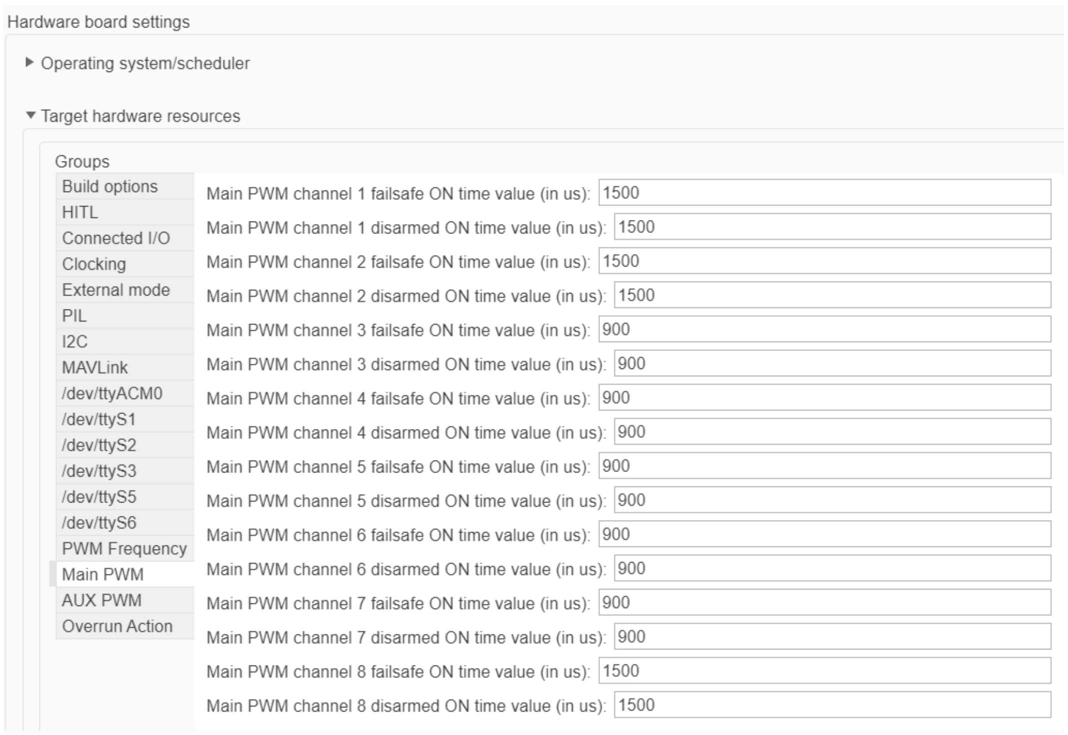
Configuration Parameters

The Configuration Parameters menu can be accessed through Model Settings or under Hardware Settings under the Hardware tab in Simulink®. The Hardware Implementation menu allows the user to select the PX4 autopilot being used and further modify custom startup configurations. The following Configuration Parameter changes were made to improve bench testing and firmware embedding functionality. The tabs described can be found under Target hardware resources.

Note: Changes to the Configuration Parameters require that the embedded coder is installed and will take effect when the next connected simulation or firmware deployment occurs.

D.1 Disarmed and Failsafe PWM Values

The Main PWM tab allows the user to set custom disarmed PWM values for each servo channel on the PX4 autopilot. The default and failsafe PWM signals for each channel is specific to the aircraft being used and the hardware configuration. When a connected simulation or custom firmware is run, the ESCs will arm automatically. The disarmed value specified in the Main PWM tab shown in Figure D1 will be sent to the control surfaces by default when disarmed. Table D1 shows the disarmed PWM values used for the model CL-84. Servos were set to 1500 microseconds when disarmed, and motors were set to 900 microseconds.



Main PWM channel 1 failsafe ON time value (in us):	1500
Main PWM channel 1 disarmed ON time value (in us):	1500
Main PWM channel 2 failsafe ON time value (in us):	1500
Main PWM channel 2 disarmed ON time value (in us):	1500
Main PWM channel 3 failsafe ON time value (in us):	900
Main PWM channel 3 disarmed ON time value (in us):	900
Main PWM channel 4 failsafe ON time value (in us):	900
Main PWM channel 4 disarmed ON time value (in us):	900
Main PWM channel 5 failsafe ON time value (in us):	900
Main PWM channel 5 disarmed ON time value (in us):	900
Main PWM channel 6 failsafe ON time value (in us):	900
Main PWM channel 6 disarmed ON time value (in us):	900
Main PWM channel 7 failsafe ON time value (in us):	900
Main PWM channel 7 disarmed ON time value (in us):	900
Main PWM channel 8 failsafe ON time value (in us):	1500
Main PWM channel 8 disarmed ON time value (in us):	1500

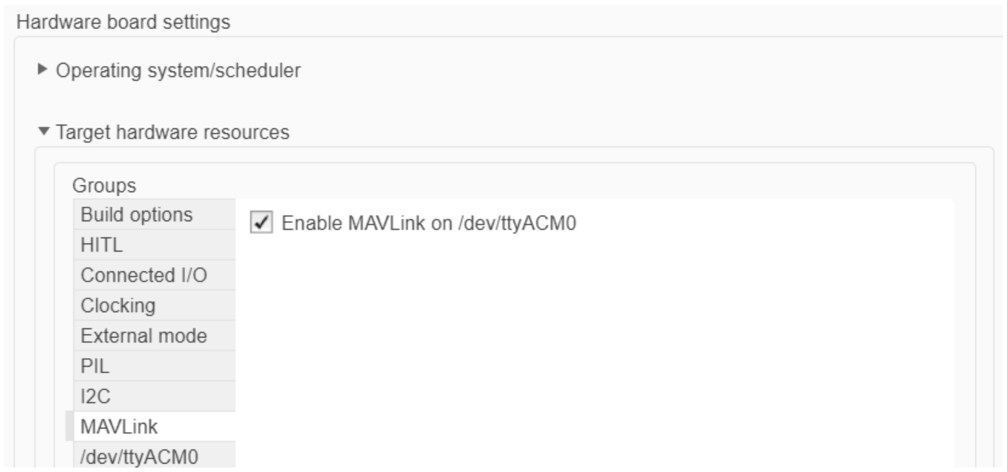
Figure D1: Disarmed and failsafe PWM values set in the Configuration Parameters menu.

Table D1: Pixhawk servo rail disarmed PWM values for model CL-84 UAV.

Port Label	Connection	Disarmed PWM
Main 1	Aileron Servos	1500
Main 2	Elevator Servo	1500
Main 3	Right Motor	900
Main 4	Rear Motor	900
Main 5	Left Motor	900
Main 6	-	900
Main 7	-	900
Main 8	Rear Motor Rotation	1500

D.2 Enabling MAVLink

MAVLink is a versatile communication protocol developed and utilized on PX4 autopilots. MAVLink, in the case of building custom controllers with Simulink®, allows for communication with the Pixhawk flight computer using QGroundControl over a telemetry connection. This communication allows for in-flight sensor monitoring, calibration, parameter changes, and the use of the MAVLink Console. The MAVLink Console allows for functionality such as starting the PX4 logger before flights. MAVLink is enabled in the Hardware Implementation sub-menu of the Configuration Parameters menu as shown in Figure D2. Connected I/O mode and Build, Deploy and Start mode are able to be run with MAVLink enabled.


Hardware board settings

▸ Operating system/scheduler

▾ Target hardware resources

Groups: Build options, HITL, Connected I/O, Clocking, External mode, PIL, I2C, MAVLink, /dev/ttyACM0

[✓] Enable MAVLink on /dev/ttyACM0

Figure D2: Enable MAVLink to allow QGroundControl connection.

Appendix E

Troubleshooting

This appendix describes various troubleshooting steps that were noted during this study. These procedures help to overcome errors that could occur when running Connected I/O mode bench test simulations or when using Build, Deploy and Start mode to embed custom firmware on the Pixhawk flight computer.

E.1 PX4 Firmware Flash from QGC

PX4 firmware flashes can fail for a number of reasons throughout testing. First, use the Firmware tab under the vehicle setup menu to flash ArduPilot firmware to the autopilot, then reboot and flash PX4 firmware. Try this procedure first before continuing on to further troubleshooting steps.

If issues persist, try to remove the SD card and attempt to flash PX4 firmware again. (The SD card could have corrupt startup files that are interfering with the normal processes of the autopilot. A corrupt rc.txt can prevent proper startup and firmware flashing abilities.) Remove the file and try flashing the firmware again. If this is successful, complete the setup outlined in Appendix C.1 to verify that the parameters are set and calibrations are complete.

E.2 Hardware Setup in Simulink®

If loading the firmware to the board is consistently failing from the Hardware Setup screens in Simulink®, try flashing ArduPilot to the board from QGC. Then flash PX4 firmware to the board and complete the setup in Appendix C.1 to verify that the parameters are set, and calibrations are complete. After setup in QGC is complete, attempt the Hardware Setup process again in Simulink®.

E.3 Connected I/O

If Connected I/O is failing consistently, attempt to reload the firmware from the Hardware Setup process in Simulink®.

E.4 Build, Deploy and Start

E.4.1 Deployment Frozen

If the Build, Deploy and Start process is frozen when attempting to reboot the Pixhawk board, try the following steps as needed to resolve the issue:

1. Disconnect and reconnect the USB cable.
2. Disconnect the Pixhawk, cycle power, and reconnect the Pixhawk.
3. If the previous steps fail, abort the process by using the ctrl + c command and refer to Section E.4.2 to troubleshoot a deployment error.

Note: It is standard for the Diagnostics Panel to show that the system is attempting to reboot the board for a couple of minutes and then successfully deploy the firmware with the success message shown in Figure E1. These steps are for when the process is frozen for an excessive amount of time.

E.4.2 Deployment Error

If a firmware build error with Build, Deploy and Start occurs, try the following steps as needed to resolve the issue:

1. Attempt the Build, Deploy and Start process again.
2. Run a Connected I/O Simulation. This will attempt to flash the Connected I/O firmware that is flashed to the board during the initial hardware setup in Simulink® and thus will take longer to run the connected mode than usual. Monitor the progress in the diagnostics panel. If the process freezes or displays an error message, refer to section C.3 to complete the Hardware Setup wizard again.
3. If the previous steps fail, return to QGroundControl and flash PX4 firmware. (Refer to Appendix C.1 to begin setup from this step.)

Note: Verify that the parameters specified in the QGroundControl setup (see Appendix C.1) are set correctly. Also, ensure the sensors are calibrated and the Pixhawk flight computer is able to arm from QGroundControl before continuing to setup in Simulink®.

E.4.3 Other Build Failures

A firmware build failure can be diagnosed by attempting to use the Build mode to create the firmware instead of using the Build, Deploy and Start option. Build failures can occur as a result of:

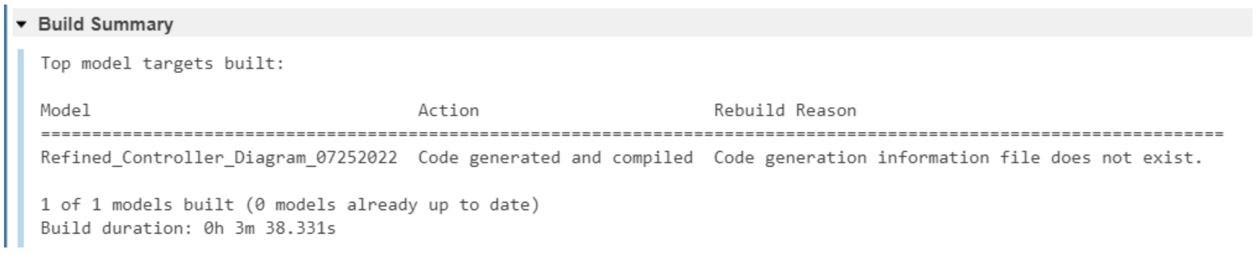
1. **Issue:** Data type inconsistency
   Solution: Verify that PWM signals are converted from `uint16` to `double` before any calculations are performed on the signal within the Simulink® diagram.
2. **Issue:** Corruption in the firmware build file
   Solution: Recreate the Simulink® diagram by starting a new model and pasting in one section of the controller at a time while building the firmware incrementally.

   Note: This method will indicate the controller section where the build is failing. If there are no apparent issues, the new Simulink® model and subsequent new corresponding firmware folder will allow for a successful build.

3. **Issue:** MATLAB® Embedded Coder was installed after the Simulink® model was created.
   Solution: To resolve this problem, recreate the Simulink® diagram by starting a new model and pasting in one section of the controller at a time while building the firmware incrementally.

E.4.4 Indication of a Successful Firmware Deployment

Successful firmware deployments result in success messages from the diagnostics panel in Simulink® shown in Figure E1. After deploying the firmware to the Pixhawk, a reboot of the board ensures that the firmware runs properly. Reference Appendix B.3 for instructions on rebooting the autopilot.



```
Build Summary

Top model targets built:

Model                                 Action                       Rebuild Reason
=====================================================================================================================
Refined_Controller_Diagram_07252022  Code generated and compiled  Code generation information file does not exist.

1 of 1 models built (0 models already up to date)
Build duration: 0h 3m 38.331s
```

Figure E1: Successful custom firmware deployment message in the diagnostic viewer.

Appendix F

Flight Test Procedures

F.1 QGroundControl

After custom firmware deployment, QGroundControl (QGC) is able to communicate with the Pixhawk using a telemetry connection, which is enabled in Configuration Parameters as discussed in Appendix D.2. The telemetry connection is established between the UAV and the antenna connected to the ground control station via USB. Vehicle information, including the gyroscope, heading, and various vehicle uORB topics can be read in real-time with QGC. Compass, accelerometer, and gyroscope calibrations are necessary for initial setup and will be completed with QGC. After initial setup, the level horizon calibration should be completed each time after deploying any updated firmware; although, more comprehensive sensor calibrations can improve sensor performance. Additionally, parameters can be changed before, during, or after flight using QGC and the telemetry connection.

F.2 Logging

PX4 offers a system to log various flight parameters and uORB messages as a ULog file [23]. This file can be imported into MATLAB® for post-flight analysis using functions available in the UAV Toolbox, as discussed in Reference [10].

F.2.1 Customization of Logging Topics and Sample Rates

To customize logging settings, create a .txt file and place it in the root folder of the SD card to specify the topics to be recorded and the sample interval of each topic in milliseconds.

The file path should be: `SDHC/etc/logging/logger_topics.txt`.

The topic and logging interval file used for this study (adapted from References [10, 24]) is printed below.

```
actuator_armed 500
actuator_controls_0 20
actuator_outputs 0 0
airdata 0
airspeed_validated 1000
battery_status 300
commander_state 500
cpuload 500
distance_sensor 0
ekf2_timestamps 500
ekf_gps_drift 200
estimator_attitude 0
estimator_local_position 0
estimator_innovation_test_ratios 200
estimator_innovation_variances 200
estimator_innovations 200
estimator_sensor_bias 1000
estimator_states 0
estimator_status 0
home_position 20000
input_rc 0
logger_status 1000
manual_control_setpoint 200
offboard_control_mode 100
px4io_status 1000
rate_ctrl_status 200
rpm 0
safety 1000
```

```
sensor_accel 20
sensor_baro 50
sensor_combined 0
sensor_gyro 20
sensor_mag 1000
sensor_preflight 200
system_power 500
telemetry_status 1000
vehicle_air_data 0
vehicle_angular_acceleration 1000
vehicle_angular_velocity 20
vehicle_attitude 20
vehicle_attitude_setpoint 100
vehicle_control_mode 500
vehicle_global_position 200
vehicle_gps_position 200
vehicle_imu 500
vehicle_imu_status 1000
vehicle_land_detected 1000
vehicle_local_position 20
vehicle_magnetometer 200
vehicle_rates_setpoint 100
vehicle_status 500
vehicle_status_flags 500
wind_estimate 1000
yaw_estimator_status 200
```

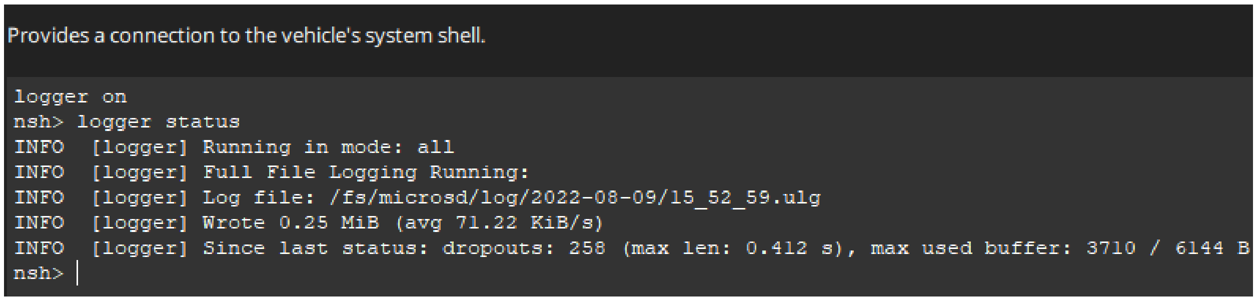
F.2.2 Logger Commands

After the topics and logging rates are configured and custom firmware is deployed, start logging through the MAVLink Console in QGC. Reference Table F1 for the MAVLink Console commands to start, stop, and verify the logger status. Logging typically starts when the Pixhawk is armed through the ground control station. However, because arming in a custom controller is configured differently, logging should be started manually through the MAVLink Console. Figures F1 and F2 show the MAVLink Console when starting and stopping a log.

Note: Logs on PX4 autopilots save automatically when powered off.

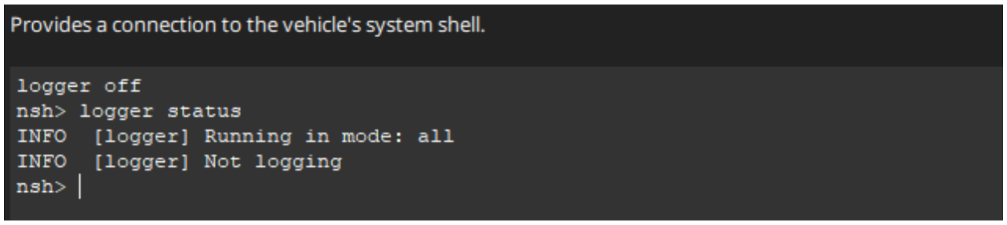
Table F1: Logger commands for the MAVLink console in QGroundControl.

Command	Action
`logger on`	Starts New Log
`logger status`	Checks if Pixhawk is Logging
`logger off`	Stops and Saves Log
`logger help`	Displays More `logger` Options


Provides a connection to the vehicle's system shell.

```
logger on
nsh> logger status
INFO  [logger] Running in mode: all
INFO  [logger] Full File Logging Running:
INFO  [logger] Log file: /fs/microsd/log/2022-08-09/15_52_59.ulg
INFO  [logger] Wrote 0.25 MiB (avg 71.22 KiB/s)
INFO  [logger] Since last status: dropouts: 258 (max len: 0.412 s), max used buffer: 3710 / 6144 B
nsh> |
```

Figure F1: Starting the Pixhawk logger with the MAVLink console in QGroundControl.



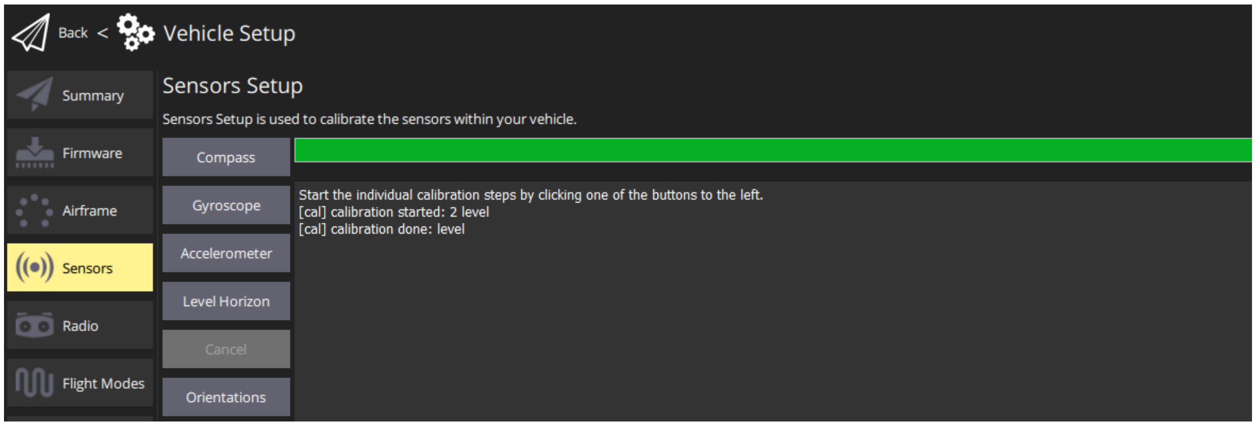
Provides a connection to the vehicle's system shell.

```
logger off
nsh> logger status
INFO  [logger] Running in mode: all
INFO  [logger] Not logging
nsh> |
```

Figure F2: Stopping the Pixhawk logger with the MAVLink console in QGroundControl.

F.3 Pre-Flight Procedures

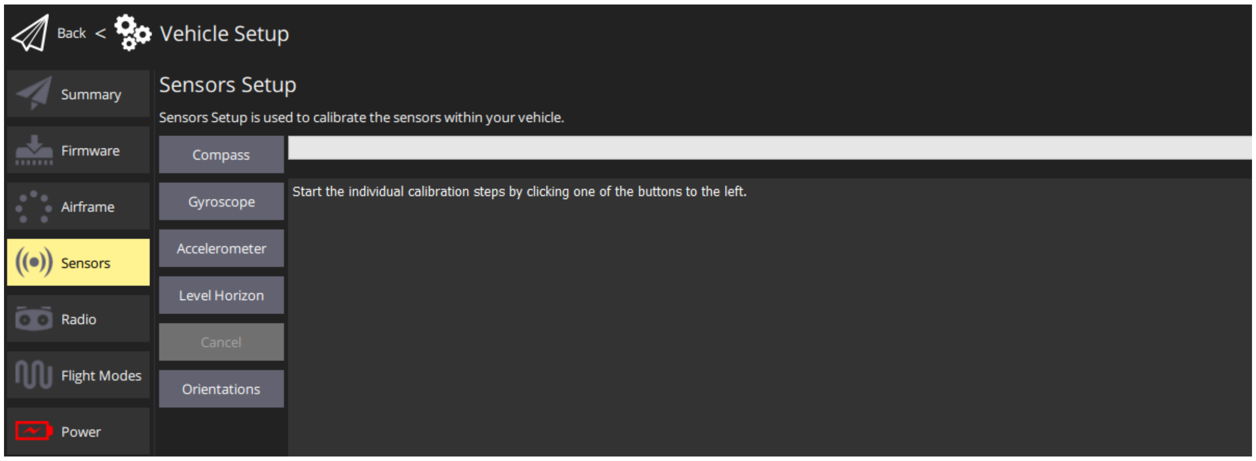
1. Power up the transmitter and ensure the throttle is at the zero setting. The arming switch should be set to disarmed.
2. Ensure the Pixhawk is secured and connected to the telemetry radio, receiver, Pixhawk 2.1 Power Module from the battery, GPS, and safety switch.
3. Make sure the SD card is inserted in the Pixhawk securely.
4. Connect the aircraft to the battery.
5. Open QGC and establish a telemetry connection to the aircraft.
6. Navigate to the MAVLink Console and activate the logger. Refer to Table F1.
7. Verify the logger status using the `logger status` command in the MAVLink Console. See Figure F1.
8. Complete the level horizon calibration, shown in Figure F3, and any other desired sensor calibrations, shown in Figure F4.



Sensors Setup is used to calibrate the sensors within your vehicle.

Start the individual calibration steps by clicking one of the buttons to the left.
[cal] calibration started: 2 level
[cal] calibration done: level

Figure F3: Level horizon calibration completed within QGroundControl.



Sensors Setup

Sensors Setup is used to calibrate the sensors within your vehicle.

Start the individual calibration steps by clicking one of the buttons to the left.

Figure F4: Other sensor calibrations completed within QGroundControl.

9. Press and hold the safety switch for three seconds and ensure the button light is blinking rapidly.
10. Arm the UAV from the transmitter and ensure the safety switch light is solid red and the Pixhawk status light is solid green.

F.4 Post-Flight Procedures

1. Ensure the UAV is safely on the ground and disarm it.
2. Disconnect the Pixhawk and ESC power source from the battery at the same time.

Note: Disconnecting the Pixhawk from power before the ESCs can lead to automatic spool up of the motors. Ensure that power is disconnected from the whole system at once.

3. Remove the SD card and save the flight logs on the computer. Flight logs can also be saved via telemetry, but this method may take several minutes for large flight logs.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-09-2022	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*

4. TITLE AND SUBTITLE Rapid Flight Control Law Deployment and Testing Framework for Subscale VTOL Aircraft	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Garrett D. Asper, Universities Space Research Association, Hampton, Virginia; Benjamin M. Simmons, Langley Research Center, Hampton, Virginia	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER 109492.02.07.07.05
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center, Hampton, Virginia 23681-2199	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration, Washington, DC 20546-0001	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) NASA/TM–20220011570

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A set of procedures was developed to enable rapid flight control law deployment and testing on subscale vertical takeoff and landing (VTOL) aircraft. Low-cost, subscale flight vehicles have become well-suited testbeds for rapid flight dynamics and controls research progression; however, integration of custom flight control laws onto flight hardware has historically been an arduous task. The toolchain described in this report leverages Simulink® with the UAV Toolbox, a Pixhawk flight computer running PX4 firmware, and QGroundControl to efficiently design and flight test custom control algorithms. A subscale CL-84 VTOL aircraft was used as a testbed in this investigation to exercise the hardware integration process on a physical model. Implementation of custom attitude stabilization control laws and programmed test input excitations for aircraft system identification were demonstrated using the expeditious hardware integration process. The detailed procedures given in this report are expected to be used in future flight test efforts.

15. SUBJECT TERMS
flight testing, flight controls, system identification, PX4, Pixhawk, UAV

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			HQ-STI-infodesk@mail.nasa.gov
U	U	U	UU	48	**19b. TELEPHONE NUMBER** *(Include area code)* (757) 864-9658

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18