2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

# Cross-Layer Retrofitting of UAVs Against Cyber-Physical Attacks

Fan Fei[+*], Zhan Tu[+*], Ruikun Yu[+], Taegyu Kim[++], Xiangyu Zhang[+++], Dongyan Xu[+++], and Xinyan Deng[+]

***Abstract*— As a rapidly growing cyber-physical platform, unmanned aerial vehicles are facing more security threats as their capabilities and applications continue to expand. Adversaries with detailed knowledge about the vehicle could orchestrate sophisticated attacks that are not easily detected or handled by the vehicle's control system. In this work, we purpose a generic security framework, termed BlueBox, capable of detecting and handling a variety of cyber-physical attacks. To demonstrate an application of BlueBox in practice, we retrofitted an off-the-shelf quadcopter. A series of attacks were then launched by embedding malicious code in the control software and by altering the vehicle's hardware with the specific targeting of sensors, controller, motors, vehicle dynamics, and operating system. Experimental results verified that BlueBox was capable of both detecting a variety of cyber-physical attacks, while also providing the means in which to recover from such attacks.**

## I. INTRODUCTION

In recent years, we have witnessed an increased popularity of unmanned aerial vehicles (UAVs) used in a variety of applications including aerial photography, search and rescue, precision agriculture, surveying and inspection [1]. With the ever expanding usage of UAVs, the level of sophistication of their electronic control units (ECUs) have been required to keep pace; with command and control algorithms running on real-time operating systems instead of directly on micro-controllers. This trend raises concern about the security of such systems. As an entity that senses and interacts with the real world, a cyber-physical system (CPS) could be exploited by an adversary and result in harmful impacts to the physical world. One such example is the cyber attack against a physical infrastructure [2]. Currently, there exists no universal operating system or security solution for either professional or hobby UAVs.

To address potential security problems with UAVs', a variety of control algorithms have been proposed, such as state-estimation, robust control, fault detection and diagnosis (FDD) and fault tolerant control (FTC) [3]–[6]. These methods work well with a faulty sensor or actuator, assuming no active real-time manipulation of the vehicle, in a dynamic and insidious way. Other methods have used parameter and and controller estimation to detect changes in the vehicles dynamics or control algorithm [7]. Such detection is reliable only if the input and state measurement are not compromised. Complementary to these works, cyber attacks have also been extensively studied as a computer security problem.

[+]School of Mechanical Engineering, Purdue University
[++]School of Electrical and Computer Engineering, Purdue University
[+++]Dept. of Computer Science, Purdue University
[*]These two authors contributed equally

This work was supported in part by ONR under Grant N00014-17-1-2045.(Email: xdeng@purdue.edu).

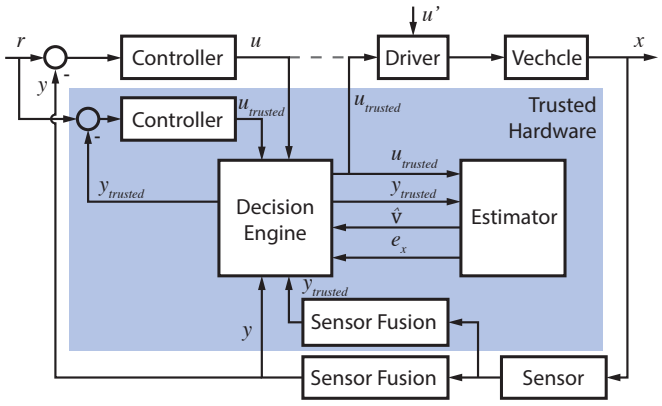


Fig. 1. BlueBox diagram. Components within the blue shade will be implemented on the external hardware. BlueBox reads the command reference, controller output, raw sensor data and sensor fusion results. BlueBox is not accessed by the original system and has the sole control over the actuation of the vehicle.

Examples include works that have aimed at deriving an attack detection model and improvements to controller robustness [8], [9]. In the CPS domain, system abnormal response is not only from physical fault or uncontrollable disturbance, but rather from malicious cyber attack. Cyber domain analysis of the threat models and vulnerabilities of UAVs are proposed in great detail [10]–[12]. For example, successful cyber-domain attacks were demonstrated [13]–[15], in which the vehicle take-over is achieved using sensor spoofing and estimator tricking strategies.

Recently, more sophisticated approaches have been proposed for robotic vehicle attack detection and counteraction with stronger threat model. Attacker are assumed to have full access to sensing and actuation of the system [16]. However, if the attacker has sufficient knowledge about the vehicle and its defense strategy, the control software is still subject to tampering. Attack detection requires comparison of true and faulty signals, thus necessitating redundancies. Many model and model-free based approaches can be used to detect system fault as summarized in [17]. Failure in the system can be relatively easy to detect provided that not all system components are compromised. With a redundant controller and an intelligent decision unit, even if the main controller is faulty the system has been shown to survive [18]. To achieve a high security level, redundant software and hardware are needed. If the vehicle is assumed to be vulnerable to attacks, then the attack scope can encompass the entire software and hardware components. However, if an extra ECU hardware is retrofitted to the vehicle with no access channel from the outside, it is shielded from the attacker. In this manner,

the vehicle still operate as intended with its original control software, while the control signal can also be used to enable comparison against the retrofitted ECU for attack detection and response.

In this work, we propose a generic security framework called BlueBox, which can be retrofitted to a range of multi-rotor UAVs. Instead of changing the original control system, an external piece of hardware was used to monitor the vehicle with minimum modification to the original system. From vehicle source code or through binary reverse engineering [19], [20], the vehicle's control logic can be extracted. Independently implementing the control and sensing logic on the external hardware enables high-accuracy error detection. A smooth variable structure filter (SVSF) was used to estimate system states and identify system parameters, which is proven to be robust to model uncertainty and noise. Combining the software and hardware redundancy, the framework was able to detect a variety of attacks on the sensor, controller, vehicle dynamics, actuator, and controller operating system. The vehicle was able to fully recover to normal operation once the attacks were detected. We implemented this framework on a 3DR IRIS+ quadcopter running ArduPilot 3.3 on NuttX operating system. Five different types of attacks were tested on the system and the attacks were identified and nullified.

## II. BlueBox Design

### A. Definition of Security

Before describing our approach in detail, it is important to establish the criteria of security for a UAV system. The system is said to be secure when it satisfies three properties, confidentiality, integrity and availability [10].

*1) Confidentiality (or Secrecy):* refers to the restricted information availability to only authorized personnel/agent. For UAVs, this type of security is usually ensured by the ground station software, regardless of the means of communication (satellite, cellular, radio, etc.). Breach of confidentiality will result in the information being exposed to a third party, and expose the system to other types of threats. Possible attacks including compromising the communication link and the human operator.

*2) Integrity:* refers to the information trustworthiness in the system. For UAVs, various methods like trojan code or exploit subroutines of the software, jamming/capturing/editing signals can compromise the data integrity, with or without compromising the confidentiality first. With various detection mechanisms already available, this type of attack is hard to launch without sufficient knowledge of the target system, but will cause the most significant damage to the vehicle if successful. Loss of integrity can lead to the vehicle being affected or even controlled by the attacker while the operator receives false or deceptive information.

*3) Availability:* refers to the availability of the information or the control of the system. If the integrity is compromised, the availability will be affected as the attacker could influence the system partially or in the most extreme case, take over the system.

### B. Threat Model and Attacks

As an aerial unmanned vehicle, a quadcopter utilizes multiple sensors to help to maintain its posture and ability to maneuver. Through the onboard sensor fusion algorithm, the raw data from inertial measurement unit (IMU) is converted into the quadcopters roll, pitch, and yaw angles. With data from barometer and GPS, the longitude, latitude, and altitude values of the quadcopter can also be measured. With this information, the flight control system can then control four different rotors to produce necessary thrust forces to let quadcopter either maneuver or maintain position according to users command.

Based on the above description, a quadcopter can be attacked from various aspects. We focus on exploiting the different aspects of the control system's physical dynamics rather than just the cyber perspective. These attacks can be implemented by trojaning the control software. Four attack models are proposed here:

*1) Sensor Attack:* The raw sensor reading in the control software or the result from sensor fusion algorithm is being hijacked by the attacker. This results in the quadcopter lose the ability to acquire authentic information about its posture, altitude, and location. This attack will mislead the control algorithm to accept wrong control input. In this attack model, it is assumed that the sensor hardware is not compromised.

*2) Controller Attack:* In this attack model, the flight control software is compromised. The controller output could be either modified or replaced by the attacker. For even stealthier attacks, for example, the attacker could modify the conditional parameters to force the control software branching into a different execution path that leads to undesirable behavior. These attacks could lead to loss of stability or control.

*3) Actuation Attack:* Proper actuation is crucial to the vehicle since it provides the necessary control authority for maneuvering. Flight control software is usually designed based on detailed knowledge about the actuation dynamics. The attacker could sabotage the physical component of the vehicle through human channels, for example installing propeller blades with an incorrect angle of attack. The actuation could also be compromised by integrity attacks such as modifying driver software for more stealthy attacks or directly hijacking the controller's output to take over the vehicle.

*4) Vehicle Dynamics Attack:* Similar to actuation system, the vehicle dynamics properties such as mass and moment of inertia are considered in the vehicle's design process. Illegal payloads or unbalanced mass distribution could result in shortened flight time or undesirable maneuverability.

*5) Operating System Attack:* This type of attack can be categorized as denial-of-service (DoS) attack. By exploiting the vulnerabilities in the control software, the attacker could effectively alter the behavior of the operating system of the

flight controller such as slow down the control loop, kill a process or crash the system.

### C. BlueBox Approach

*1) overview:* To defend against the above mentioned security threats, we propose the cross-layer security framework BlueBox. It contains an external computing unit, with both hardware and software, that is retrofitted onto the original system and can access information within the original system but not the other way around. Once any abnormal status has been detected, it will immediately isolate the most suspicious component and make a decision such as takeover the control, return to start-point or land directly. The BlueBox does not have the full attack detection capability on its own. However, by running the redundant sensor fusion and control algorithm on the redundant hardware, BlueBox can detect changes in the original system by comparing instantaneous outputs in real-time, while itself is shielded from attackers. The entire diagnosis process is designed with both cyber and physical domain approaches in mind, hence the name cross-layer.

*2) Retrofitting:* The retrofitting of a vehicle with BlueBox requires very minimum software and hardware modifications. For software retrofitting, we enable the original system to send necessary data to BlueBox. For hardware retrofitting, we rewire the actuation system through BlueBox hardware and add wirings to allow BlueBox to directly access the sensor. The software component of the BlueBox consists of sensor fusion algorithm, controller, parameter identification, states estimation and a decision-making module. The architecture diagram is shown if Fig.1.

*3) Detection Mechanism:* BlueBox's detection mechanism is described as follows. At vehicle takeoff, the estimator will update the vehicle's physical parameters. After a short period of time (<30 seconds in our testing), the vehicle can proceed its mission and the BlueBox starts fault/attack detection. As the foundation of a feedback control system, safeguard of feedback is critical. Based on binary code reverse engineering and Platform Independent Executable Trace technology [20], we can infer the control and sensing algorithms from the original system and *independently* implement them for BlueBox. Through a direct wiring from the sensor, BlueBox extracts sensors raw data for reference states calculation. It is used to be compared with the feedback from original system to determine if the sensor fusion result or code in the original system has been modified. The decision engine will integrate the error between the two measurements within a fixed time window and identifies sensor attacks using thresholding. BlueBox sensor fusion results are also fed to its internal controller and SVSF based estimator for further security diagnosis and attacks detection. To increase the level of security, extra redundant sensors can be added and sensor hardware failure can be detected [6].

In the case of a quadcopter, 4 PWMs (Pulse Width Modulation) as the control commands from original system are captured by BlueBox peripheral interrupt. After the sensor check, if an evident discrepancy of motor commands between BlueBox and captured value has been detected by the decision engine, the threat diagnosis would consider the original controller's integrity compromised and moreover, takeover the vehicle control. In cyber aspect, some higher level attacks, such as software-layer attacks, timer modification or interrupt vector re-mapping, can also be detected since all of them cause control commands discrepancy.

Based on the above two checks, the cyber level safety can be ensured and system integrity can be preserved. Other potential attacks could be delivered on the physical system. In particular, for actuation system safety, BlueBox observes motors thrust based on the vehicle dynamics. If the motor/rotor is broken or the driver circuit is compromised by integrity attacks, fault tolerant control can be employed to ensure the system's availability. In contrast to actuation system attack, physical properties attack is more stealthy since it usually won't directly affect flight mission. Instead, it changes the system sensitivity or increases the power consumption. With fine-tuned system parameter at takeoff, the estimator could detect changes in system parameters. Therefore any illegal payload attachment during flight time could be identified immediately. Depending on batteries capacity, either return to home position or landing could guarantee the safety of the vehicle.

## III. System Modeling

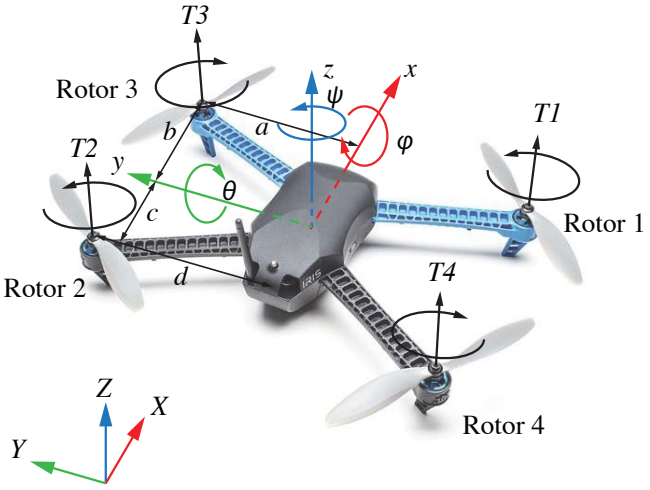


Fig. 2. Quadcopter coordinate system definition. In our implementation, a 3DR IRIS+ is used. $a$, $b$, $c$ and $d$ denotes the distances from the motor to the center of mass.

The Smooth Variable Structure Filter (SVSF) is a method for parameter and state estimation that was first proposed in 2007 [21] and proven to be robust to model uncertainties. The advantage of SVSF is that it could obtain state estimation as well as parameter identification simultaneously [22]. SVSF has the ability to identify the source of model uncertainty and improve estimation performance based on this knowledge. Thus, it has been considered as a better alternative to RLS. [23]. Similar to many existing detection methods, we use estimation error (residual) to detected changes in the system.

### A. Vehicle Dynamics

The coordinates definition is shown in Fig. 2. With the rigid body assumption, the vehicle's dynamics model can be described using standard Newton-Euler equation. The dynamics can be described as

$$\begin{aligned} &\dot{p} = v, \\ &m\ddot{p} = Rf^b + mg\mathbf{e}_3, \\ &\dot{R} = R\hat{\omega}^b, \\ &\mathcal{I}\dot{\omega}^b + \omega^b \times \mathcal{I}\omega^b = \tau^b. \end{aligned} \tag{1}$$

where $p = [x, y, z]^T$ and $v$ are the vectors describing the vehicle's position and velocity in the earth frame, $m$ is the mass of the vehicle, $g$ is the gravity acceleration, $R \in \mathbb{R}^{3\times3}$ is the rotation matrix, $f^b = [0, 0, F_z]^T$ is the total force vector in the body frame, $\mathbf{e}_3$ is the unit vector $[0, 0, 1]^T$, $\omega^b = [p, q, r]$ is the angular velocity in the body frame, $\hat{\cdot}$ denotes the skew-symmetric matrix mapping from $\hat{x}y$ to $x \times y$, $\mathcal{I} \in \mathbb{R}^{3\times3}$ is the full inertia matrix of the vehicle and $\tau^b = [T_x, T_y, T_z]^T$ is the total torque vector in the body frame.

The motor dynamics can be approximated by a first order system

$$\dot{W} = -\alpha W + K_m u \tag{2}$$

where $W$ is the squared angular velocity, $\alpha$ is the pole location, $K_m$ is the lumped gain, and $u$ is the input voltage. We can model the thrust and drag force of each rotor as

$$T_i = K_{T_i} W_i, \tag{3}$$

$$Q_i = K_{Q_i} W_i. \tag{4}$$

where $i$ is the motor index, $K_{T_i}$ and $K_{Q_i}$ are the lift and drag coefficient of each individual rotor.

The entire system model can be written into the following discrete form

$$\mathbf{x}_{k+1} = \boldsymbol{\mathcal{F}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \tag{5}$$

$$\mathbf{y}_{k+1} = \boldsymbol{\mathcal{H}}\mathbf{x}_k + \mathbf{v}_k. \tag{6}$$

where $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r, W_1, W_2, W_3, W_4]^T$ is the state variables vector, $\mathbf{u} = [u_1, u_2, u_3, u_4]^T$ is the vector of input voltage to the motors, $\mathbf{w}$ is the disturbance, $\mathbf{y}$ is the output, $\mathbf{v}$ is the measurement noise, $k$ represents the time step, $kdt$ and $dt$ is the sampling time. Since the velocity of the motor are not measured, the output matrix $\boldsymbol{\mathcal{H}}$ takes the form $\boldsymbol{\mathcal{H}} = [\boldsymbol{\mathcal{H}}_u \ \mathbf{0}] = [\mathbf{I}_{12} \ \mathbf{0}]$, where $\mathbf{I}_{12}$ is a $12 \times 12$ identity matrix and $\mathbf{0}$ is a $12 \times 4$ zero matrix.

### B. SVSF State Estimator

The SVSF state estimator uses a discontinuous corrective action to drive down the error, which is similar to sliding mode observer. The estimation consists of two steps. A prediction model $\hat{\boldsymbol{\mathcal{F}}}$ is used first to generate the a priori estimation $\hat{\mathbf{x}}_{k|k-1}$, then a corrective term $K_k$ was added to generate the a posteriori estimation $\hat{\mathbf{x}}_{k|k}$ and subsequently the output $\hat{\mathbf{y}}_{k|k}$. With a zero width smoothing layer in the corrective term $\mathbf{K}_k$, the error signal is referred as chattering. The a posteriori chattering will decay over time, however, the a priori chattering is caused by the magnitude of the model uncertainties and can be used to point out the source and magnitude of modeling error [24].

Since the system has fewer measurements than its states, a reduced order estimator can be constructed [24]. We can partition the system states as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_u \\ \mathbf{x}_l \end{bmatrix} \tag{7}$$

where $\mathbf{x}_u = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, p, q, r]^T$ are the measurable states of the system and $\mathbf{x}_l = [W_1, W_2, W_3, W_4]^T$ are the unmeasurable rotor velocity.

Since $\boldsymbol{\mathcal{H}}_u$ is identity matrix, the system can be partitioned and written with measurements and unmeasurable states as:

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{x}_{l_k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mathcal{F}}_{u_k}(\mathbf{y}_{k-1}, \mathbf{x}_{l_{k-1}}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}, \mathbf{v}_{k-1}) \\ \boldsymbol{\mathcal{F}}_{l_k}(\mathbf{y}_{k-1}, \mathbf{x}_{l_{k-1}}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}, \mathbf{v}_{k-1}) \end{bmatrix} \tag{8}$$

The unmeasurable states can be recovered by inversing the $\boldsymbol{\mathcal{F}}_{u_k}$ and $\boldsymbol{\mathcal{F}}_{l_k}$. A temporary estimation can be constructed as:

$$\boldsymbol{\sigma}_{k-1} = \begin{bmatrix} \boldsymbol{\sigma}_{u_{k-1}} \\ \hat{\boldsymbol{\sigma}}_{l_{k-1}} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_{k-1} \\ \hat{\boldsymbol{\mathcal{F}}}_{u_k}^{-1}(\mathbf{y}_k, \mathbf{y}_{k-1}, \mathbf{u}_{k-1}) \end{bmatrix} \tag{9}$$

The a priori state estimation can be calculated as:

$$\hat{\mathbf{x}}_{k|k-1} = \begin{bmatrix} \hat{\mathbf{x}}_{u_{k|k-1}} \\ \hat{\mathbf{x}}_{l_{k|k-1}} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\mathcal{F}}}_{u_k}(\hat{\boldsymbol{\sigma}}_{k-1}, \mathbf{u}_{k-1}) \\ \hat{\boldsymbol{\mathcal{F}}}_{l_k}(\hat{\boldsymbol{\sigma}}_{k-1}, \mathbf{u}_{k-1}) \end{bmatrix} \tag{10}$$

The a posteriori estimation is given as:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tag{11}$$

with corrective action

$$\mathbf{K}_k = (|\mathbf{e}_{\mathbf{x}_{k|k-1}}| + \boldsymbol{\gamma}|\mathbf{e}_{\mathbf{x}_{k-1|k-1}}|) \circ \text{sgn}(\mathbf{e}_{x_{k|k-1}}) \tag{12}$$

where the a priori and a posteriori estimation error are

$$\mathbf{e}_{\mathbf{x}_{k|k-1}} = \boldsymbol{\sigma}_{k-1} - \hat{\mathbf{x}}_{k|k-1}, \tag{13}$$

$$\mathbf{e}_{\mathbf{x}_{k|k}} = \boldsymbol{\sigma}_{k-1} - \hat{\mathbf{x}}_{k|k}. \tag{14}$$

Notice the unmeasurable states error contains model uncertainty from $\hat{\boldsymbol{\mathcal{F}}}_{u_k}^{-1}$ due to the absence of explicit measurements, however this uncertainty will decay as model parameters update.

For a quadcopter, $\hat{\boldsymbol{\mathcal{F}}}_{u_k}^{-1}$ is given as:

$$\hat{\boldsymbol{\mathcal{F}}}_{u_k}^{-1} = \mathbf{P}^{-1}\hat{\mathbf{C}}(\mathbf{x}_k, \mathbf{x}_{k-1}) \tag{15}$$

where $\mathbf{P}$ is the force mapping matrix for the motor with frame dimensions $a, b, c, d$ described in figure 2.

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -a & d & a & -d \\ -b & c & -b & c \\ -1 & -1 & 1 & 1 \end{bmatrix} \tag{16}$$

and $\hat{\mathbf{C}}$ is given as:

$$\hat{\mathbf{C}}(\mathbf{x}_k, \mathbf{x}_{k-1}) = \begin{bmatrix} \frac{\dot{z}_k - \dot{z}_{k-1} + dtg}{(\frac{\hat{K_T}}{m})(C\theta_{k-1}C\phi_{k-1})dt} \\ \frac{p_k - p_{k-1} - dt(\hat{I_1})q_{k-1}r_{k-1}}{(\frac{\hat{K_T}}{I_x})dt} \\ \frac{q_k - q_{k-1} - dt(\hat{I_2})p_{k-1}r_{k-1}}{(\frac{\hat{K_T}}{I_y})dt} \\ \frac{r_k - r_{k-1} - dt(\hat{I_3})p_{k-1}q_{k-1}}{(\frac{\hat{K_Q}}{I_z})dt} \end{bmatrix} \tag{17}$$

where $\hat{(\cdot)}$ denotes the combined estimated parameter, $I_x, I_y, I_z, m$ are the moment of inertias and mass of the vehicle, $I_1 = \frac{I_y - I_z}{I_x}$, $I_2 = \frac{I_x - I_z}{I_y}$, $I_3 = \frac{I_x - I_y}{I_z}$ and $dt$ is the sampling time. The estimated control forces are

$$\hat{\mathbf{F}} = \mathbf{P}\hat{\boldsymbol{\sigma}}_{l_{k-1}} \tag{18}$$

The a priori chattering is given as

$$\mathbf{Ch}_{k|k-1} = dt \begin{bmatrix} 0 \\ 0 \\ 0 \\ \Delta_{(\frac{K_T}{m})}(C\psi_{k-1}S\theta_{k-1}C\phi_{k-1} + S\psi_{k-1}S\phi_{k-1})\hat{F}_1 \\ \Delta_{(\frac{K_T}{m})}(S\psi_{k-1}S\theta_{k-1}C\phi_{k-1} - C\psi_{k-1}S\phi_{k-1})\hat{F}_1 \\ \Delta_{(\frac{K_T}{m})}(C\psi_{k-1}C\phi_{k-1})\hat{F}_1 \\ 0 \\ 0 \\ 0 \\ \Delta_{(I_1)}q_{k-1}r_{k-1} + \Delta_{(\frac{K_T}{I_x})}\hat{F}_2 \\ \Delta_{(I_2)}p_{k-1}r_{k-1} + \Delta_{(\frac{K_T}{I_y})}\hat{F}_3) \\ \Delta_{(I_3)}p_{k-1}q_{k-1} + \Delta_{(\frac{K_Q}{I_z})}\hat{F}_4 \\ -\Delta_{(\alpha_1)}\hat{W}_{1_{k-1}} + \Delta_{(K_{m_1})}u_1 \\ -\Delta_{(\alpha_2)}\hat{W}_{2_{k-1}} + \Delta_{(K_{m_2})}u_2 \\ -\Delta_{(\alpha_3)}\hat{W}_{3_{k-1}} + \Delta_{(K_{m_3})}u_3 \\ -\Delta_{(\alpha_4)}\hat{W}_{4_{k-1}} + \Delta_{(K_{m_4})}u_4 \end{bmatrix} \tag{19}$$

Using the chattering of $\dot{z}$ and the law of large numbers, $\Delta_{(\frac{K_T}{m})}$ can be calculated as

$$\Delta_{(\frac{K_T}{m})} = \Sigma_n^{n+l}\left(\frac{Ch_{\dot{z}_{k|k-1}}}{dtC\psi_{k-1}C\phi_{k-1}\hat{F}_1}\right) \tag{20}$$

where $l$ is some large number of samples.

For $\Delta_{(I_1)}$ and $\Delta_{(\frac{K_T}{I_x})}$ in $Ch_{p_{k|k-1}}$, since there are two variables, two segments of sample are needed. Let

$$\bar{\mathbf{CH}}_1 = \begin{bmatrix} \Sigma_n^{n+l} Ch_{p_{k|k-1}} \\ \Sigma_{n+l+1}^{n+2l} Ch_{p_{k|k-1}} \end{bmatrix} \tag{21}$$

and

$$\mathbf{PR}_1 = \begin{bmatrix} \Sigma_n^{n+l} q_{k-1}r_{k-1} & \Sigma_n^{n+l}\hat{F}_2 \\ \Sigma_{n+l+1}^{n+2l} q_{k-1}r_{k-1} & \Sigma_{n+l+1}^{n+2l}\hat{F}_2 \end{bmatrix} \tag{22}$$

The model uncertainty can be calculated as

$$\begin{bmatrix} \Delta_{(I_1)} \\ \Delta_{(\frac{K_T}{I_x})} \end{bmatrix} = \mathbf{PR}_1^{-1}\bar{\mathbf{CH}}_1 \tag{23}$$

The remaining model uncertainties for $I_2$, $\frac{K_T}{I_y}$, $I_3$, $\frac{K_Q}{I_z}$, $\alpha_i$ and $K_{m_i}$ can be calculated using similar approach. The parameters can be updated using the calculated uncertainties as $\hat{(\cdot)} := \hat{(\cdot)} + \Delta_{(\cdot)}$. After a period of parameter fine tuning, the model parameter will converge to their respective true value. Individual physical parameters $I_x$, $I_y$, $I_z$, $m$, $K_T$, $K_Q$, can be solved from the combined parameters if necessary.

### C. FDD and FTC

As stated earlier, the chattering error generated by SVSF can be used to evaluate model uncertainty and detection system fault. After the initial parameter fine-tuning, parameters will stop updating and stay as constants. By estimating the model uncertainty, we can select a bound $\hat{\beta}$ to smooth out the estimation chattering. If any failure or attacks occur, the magnitude of the a priori chattering will increase and the fault can be detected [24]. If an illegal payload was attached to the vehicle during flight, $\dot{z}$ chattering would have a significant increase and the attack can be detected. Similar strategies can be used for detecting changes in $I_x$, $I_y$ and $I_z$.

For detecting motor failure, if a motor was damaged or overtook by an adversary, the thrust estimation will deviate from its expected value. The expected motor thrust can be calculated from its voltage command from the controller, then compare with the thrust estimation from SVSF. Therefore the residual of the motor thrust can be used for detection.

The detection algorithm for vehicle dynamics and motor thrust will integrate the error/residual between the estimation and the command/measurement. If the error integral exceeds a threshold, the detection flag will be raised.

To compensate for a failed motor, FTC technique was used to stabilize the vehicle. If motor $i$ is detected to be faulty, similar to [4], the output to that motor will remain unchanged. With the corresponding thrust estimation $\hat{T}_i$ and its estimated input $\hat{v}_i$, we can redistribute the control torque to stabilize the roll and pitch angle. The control effort for z, roll, pitch yaw are given as:

$$\begin{bmatrix} u_z \\ u_{roll} \\ u_{pitch} \\ u_{yaw} \end{bmatrix} = \mathbf{Pu} \tag{24}$$

If motor 1 fails, with it thrust input estimation $\hat{v}_1$, the control output can be remapped as:

$$\mathbf{u}_1^+ = \begin{bmatrix} \hat{v}_1/u_z & 0 & 0 & 0 \\ \hat{v}_1/u_z & .25 & .25 & 0 \\ (u_z - 2\hat{v}_1)/2u_z & .25 & -.5 & 0 \\ (u_z - 2\hat{v}_1)/2u_z & -.5 & .25 & 0 \end{bmatrix} \begin{bmatrix} u_z \\ u_{roll} \\ u_{pitch} \\ u_{yaw} \end{bmatrix} \tag{25}$$

Other motor failure can be compensated similarly.

The BlueBox software was implemented on an STM32F3-DISCOVERY board (http://www.st.com) and was retrofitted to a 3DR IRIS+ quadcopter (https://3dr.com). The original firmware was modified to let the Pixhawk flight controller send all the measurements via a serial port. The controller output was captured from the PWM output. Only 8 physical

wires modification was done. Specifically, an I2C bus tie-line connection and 4 PWM channels rewiring must be done for BlueBox sensor feedback receiving and control command checking respectively.

## IV. Results and Evaluation

Before deploying the full functionality of the BlueBox, the accuracy of the parameter estimation was verified. The vehicle mass estimation is presented here. An initial guess of 1282 grams based on the manufacturer's specification was chosen. The actual weight of the vehicle was measured at 1317.2 grams. The on-line parameter estimation during a take-off is shown in Fig. 3. Before take off, error was introduced but quickly reduced after lift-off. The estimation error before 10-second mark was due to the errors from other parameter estimations such as motor thrust coefficient. The estimation converged within 20 seconds and showed good accuracy.

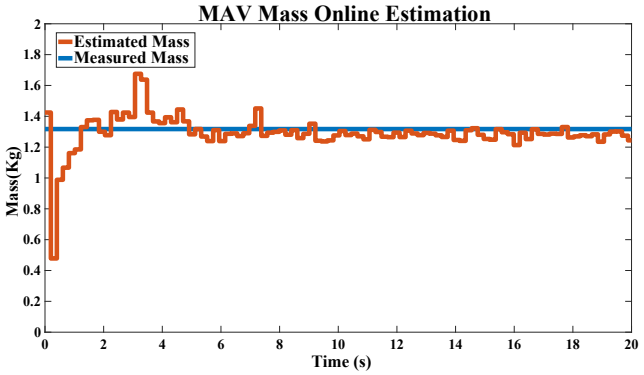


Fig. 3. The vehicle mass estimation during a take off. The estimation show good accuracy and fast convergence despite of initial error introduced before take off.

Five attacks were designed to test different aspects of BlueBox's detection and recovery capability. For each experiment, one of the five planned attack methods is applied to the quadcopter. The test ground is chosen to be an outdoor empty space for safety concerns. Relevant experimental results are presented in this section for discussion and evaluation.

### A. Sensor Attack

For the experiment purpose, the quadcopter used has been flashed with a retrofitted version of the flight control software. The sensor fusion algorithm of this system has been trojaned so that the calculated attitude of the vehicle can be manipulated by the attacker. The sensor data hacking is triggered by a remote command through a common communication protocol MAVLink. During the flight test, the quadcopter will initially stay on its hovering position. Once the attack is triggered, BlueBox will detect the sensor measurement discrepancy and subsequently takes over the control of the vehicle.

During the course of experimentation, a variety of sensor manipulations were tested. For the purpose of demonstration, 3 degrees direct addition to the angle measurement is presented. Other attacks such as position, velocity, and acceleration on different degrees of freedoms or a combination of them were also tested and easily detected.

In the experiment, a comparison test of BlueBox on/off was demonstrated in Fig. 4.

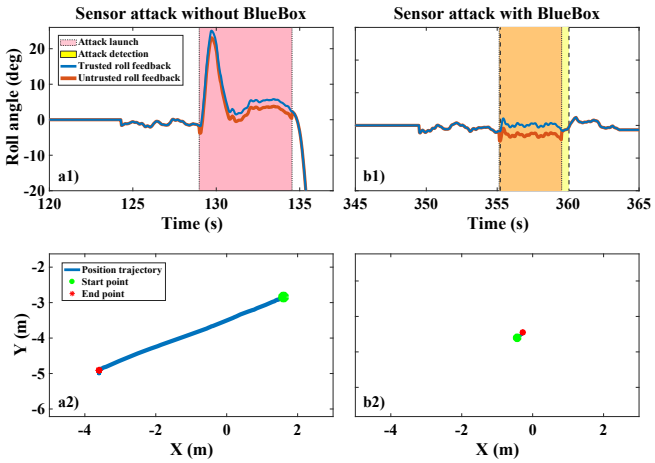


Fig. 4. Figure a) describes the performance of the quadcopter under sensor attack with BlueBox deactivated. When sensor attack is triggered, a 3-degree offset is added to the roll angle from the sensor fusion result. Without BlueBox protection, the vehicle drifted in the negative $x$ direction. Figure b) shows when BlueBox is active, once the sensor attack was launched, the protection was triggered within 0.5 seconds. The yellow region indicates the detection flag of sensor attack. The vehicle maintained its position as shown in the $x - y$ plot.

### B. Control Attack

The controller attack launched in the experiment is similar to the sensor attack. The onboard flight control system is a trojaned version so that the control gains can be changed responding to an external remote command. The attack was triggered during a loitering flight.

Aside from manipulation of control gains, many other attacks that fall into the control attack category were also tested. Such attacks including changing controller software branching conditions and changing the integration limits for integral windup prevention in the PID controller. Attacks like branching parameter modification will not be detected before the branching condition is met, such as landing speed change will not occur during normal flight. The results of gain manipulation are presented in this section for its stronger illustration of BlueBox's capability.

### C. Motor Attack

The actuation system, specifically the ESC, is trojaned so that one of the four motors will run at an abnormal RPM once an external trigger signal is given. The compromised motor provides thrust that is unknown to the control software, therefore, the control effort cannot be realized by either the original controller or the BlueBox. However, with trusted state measurement, the unknown thrust can be estimated and the control effort can be re-mapped to the vehicle accordingly.

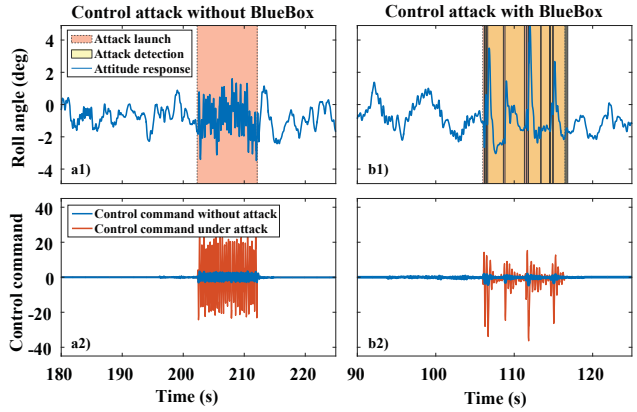


Fig. 5. Figure a) describes the performance of the quadcopter under control attack with BlueBox deactivated. Figure b) describes the performance of the quadcopter under control attack with BlueBox activated. In both cases, the control gain was only multiplied by 2 for the safety of the test vehicle. When the attack is detected, the BlueBox redundant controller will take over the vehicle. In Figure a) (BlueBox is inactive), the attack lasted for about 10 seconds. The vehicle will oscillate correspondingly and the control output difference is shown. In the protected case (BlueBox is active), the detection shows some false negatives due to the small difference between the control output when the vehicle is close to stable.

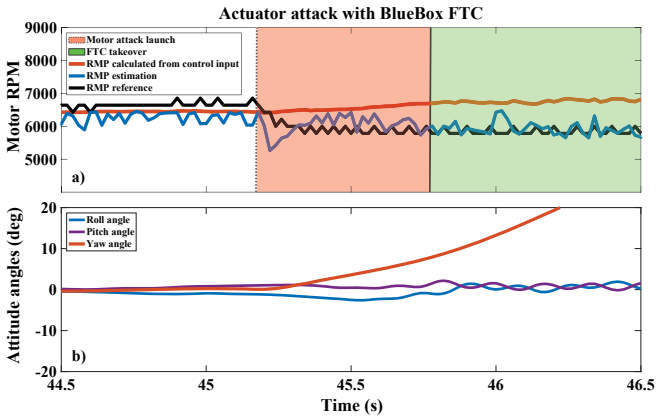


Fig. 6. These two figures describe the quadcopter's performance under motor attack with BlueBox activated. Once the motor attack is triggered, the PWM signal of one motor is controlled by the trojaned ESC. The attack is detected by BlueBox within about 0.75 seconds after the attack is launched. The FTC is activated subsequently. With a compromised motor, the quadcopter lost its yaw control as described in Figure b), while roll and pitch angle are maintained.

To validate estimation results, we installed a hall effect sensor which measures the instantaneous RPM of the attacked motor. When the motor attack has been detected, the decision-making module will enable FTC function to maintain the vehicle stability in roll and pitch directions. Consequently, the quadcopter could land safely.

FTC can handle cases with not only one faulty motor, but also up to three faulty motors according to the work done by Mueller et al. [25]. Therefore, BlueBox's FTC function actually possesses greater potential than the attack case we presented here.

In this experiment, the instantaneous motor speed and corresponding estimation result, attack launching and detection flag status, and yaw angles were recorded and plotted.

### D. Vehicle Dynamics Attack

For the experiment of vehicle dynamics attack, the inertia along the z-axis of the quadcopter was altered. This is achieved by attaching an extra weight to the quadcopter through a string. The experiment can be considered to have two stages. When the quadcopter is flying at a lower altitude with the string relaxed, the vehicle dynamics are not changed since the extra payload is not applied on the quadcopter yet. When the quadcopter flies to a higher altitude, the weight is pulled by the vehicle, the vehicle dynamics is then considered to be changed.

As described in the system modeling section, a change in the quadcopter's total mass will cause a noticeable change in the chattering signal given by the SVSF estimator. The feasibility of this concept has been well illustrated by the experiment results of this section.

The actual and estimated z-axis velocities were recorded and plotted.

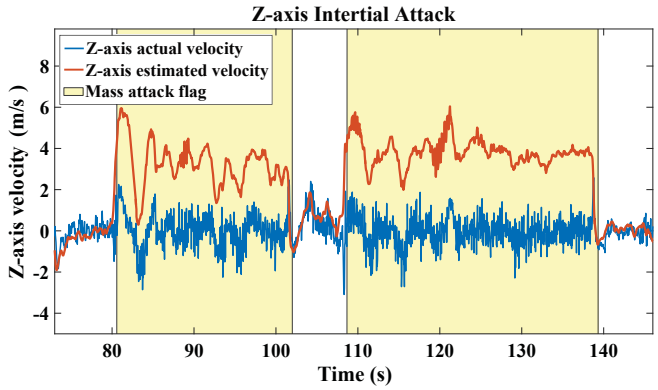


Fig. 7. This figure describes the detection of vehicle dynamics attack. Once the extra payload is attached, the estimation of velocity along z-axis will be different from the actual velocity. The attack detection flag is then raised and the control is taken-over by BlueBox. In realistic settings, BlueBox could either drive the quadcopter back to start-point or initiate landing directly. In this test case, no specific action was taken.

### E. Software Attack

We experimented three different software attack techniques: the control process shutdown, the system interrupt table remap, and the system timer slowdown. These attacks will alter the program execution differently.

In our experiments, we exploit real software vulnerability in the control program [26] to trigger our attacks. The first two attacks lead to crashing control processes by sending a processes termination command or changing hardware I/O addresses into invalid addresses. On the other hand, the last attack makes the control process scheduled infrequently. Therefore, the control program cannot run or read sensor values in time. If BlueBox receives no command or sensing information in time, it will consider the system compromised.

In this experiment, the PWM signal and the attack detection flag are recorded and plotted for each attack.

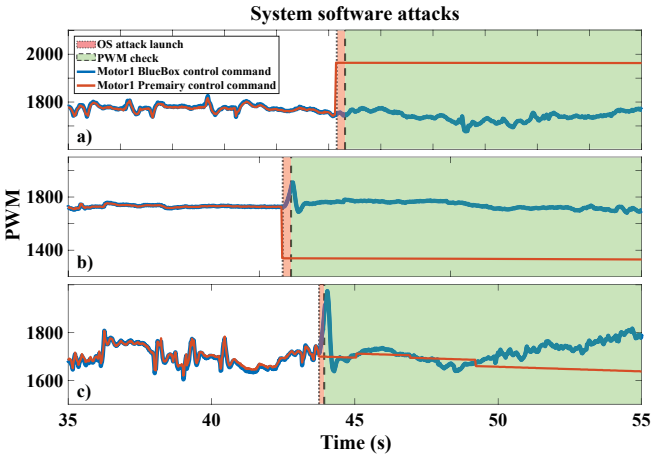


Fig. 8. Figure (a)-(c) describe three different software attacks. In case (a), the ArduPilot process is terminated when the attack is triggered. (a) shows that the attack is immediately detected by BlueBox since the PWM signals are no longer updated by the original control system. In case (b), the system's interrupt table is remapped by the attacker. This will crash the flight control system software. Similarly, the attack is easy to detect as it can be seen from (b). In case (c), the slowed timer will produce a great discrepancy in both sensor reading and controller output. In all cases, the original control system is taken-over by BlueBox once the attack is detected, and thus crashing is prevented.

## V. Conclusion

In this work, a generic retrofitting framework called BlueBox for enhancing UAV cyber-physical security is presented. Instead of altering the software on the target vehicle, BlueBox runs on a separate piece of hardware. Leveraging software redundancy and diversity, it is able to detect the discrepancy in sensing and control outputs on the target vehicle. The external hardware implementation provides the target vehicle with extra security features without increasing its attack surface. Combining the estimator and vehicle dynamics, stealthy attacks exploiting the physical properties can also be detected and responded to. The framework was deployed and tested on a 3DR IRIS+ quadcopter, five different types of attacks targeting the vehicle's sensing, control, dynamics and operating system software were implemented. BlueBox's ability to successfully detect and defend against these attacks was demonstrated on real flight tests. BlueBox can be conveniently applied to a variety of UAVs and be ported to other autonomous vehicles (e.g., an unmanned ground vehicle), as shown in our ongoing work.

## References


[1] I. H. Beloev, "A review on current and emerging application possibilities for unmanned aerial vehicles," *Acta Technologica Agriculturae*, vol. 19, no. 3, pp. 70–76, 2016.

[2] M. B. Kelley, "The stuxnet attack on iran's nuclear plant was 'far more dangerous' than previously thought," Nov 2013. [Online]. Available: http://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11/

[3] H. Aguilar-Sierra, G. Flores, S. Salazar, and R. Lozano, "Fault estimation for a quad-rotor mav using a polynomial observer," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1-4, pp. 455–468, 2014.

[4] F. Sharifi, M. Mirzaei, B. W. Gordon, and Y. Zhang, "Fault tolerant control of a quadrotor uav using sliding mode control," in *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*. IEEE, 2010, pp. 239–244.

[5] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5266–5271.

[6] G. Heredia, A. Ollero, M. Bejar, and R. Mahtani, "Sensor and actuator fault detection in small autonomous helicopters," *Mechatronics*, vol. 18, no. 2, pp. 90–99, 2008.

[7] Z. Birnbaum, A. Dolgikh, V. Skormin, E. OBrien, D. Muller, and C. Stracquodaine, "Unmanned aerial vehicle security using recursive parameter estimation," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 107–120, 2016.

[8] Q. Zhu and T. Başar, "Robust and resilient control design for cyber-physical systems with an application to power systems," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 4066–4071.

[9] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[10] A. Y. Javaid, W. Sun, V. K. Devabhaktuni, and M. Alam, "Cyber security threat analysis and modeling of an unmanned aerial vehicle system," in *Homeland Security (HST), 2012 IEEE Conference on Technologies for*. IEEE, 2012, pp. 585–590.

[11] N. M. Rodday, R. d. O. Schmidt, and A. Pras, "Exploring security vulnerabilities of unmanned aerial vehicles," in *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 2016, pp. 993–994.

[12] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, "Cyber attack vulnerabilities analysis for unmanned aerial vehicles." in *Infotech@ Aerospace*, 2012, pp. 1–30.

[13] D. P. Shepard, J. A. Bhatti, T. E. Humphreys, and A. A. Fansler, "Evaluation of smart grid and civilian uav vulnerability to gps spoofing attacks," in *Proceedings of the ION GNSS Meeting*, vol. 3, 2012, pp. 3591–3605.

[14] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via gps spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[15] C. Kwon, W. Liu, and I. Hwang, "Security analysis for cyber-physical systems against stealthy deception attacks," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 3344–3349.

[16] P. Guo, H. Kim, N. Virani, J. Xu, M. Zhu, and P. Liu, "Exploiting physical dynamics to detect actuator and sensor attacks in mobile robots," *arXiv preprint arXiv:1708.01834*, 2017.

[17] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

[18] R. F. Stengel, "Intelligent failure-tolerant control," in *Intelligent Control, 1990. Proceedings., 5th IEEE International Symposium on*. IEEE, 1990, pp. 548–557.

[19] D. Kim, Y. Kwon, W. N. Sumner, X. Zhang, and D. Xu, "Dual execution for on the fly fine grained execution comparison," in *ACM SIGPLAN Notices*, vol. 50, no. 4. ACM, 2015, pp. 325–338.

[20] Y. Kwon, X. Zhang, and D. Xu, "Pietrace: Platform independent executable trace," in *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*. IEEE, 2013, pp. 48–58.

[21] S. Habibi, "The smooth variable structure filter," *Proceedings of the IEEE*, vol. 95, no. 5, pp. 1026–1059, 2007.

[22] S. Habibi and R. Burton, "Parameter identification for a high-performance hydrostatic actuation system using the variable structure filter concept," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 2, pp. 229–235, 2007.

[23] M. A. Al-Shabi, K. S. Hatamleh, and A. A. Asad, "Uav dynamics model parameters estimation techniques: A comparison study," in *Applied Electrical Engineering and Computing Technologies (AEECT), 2013 IEEE Jordan Conference on*. IEEE, 2013, pp. 1–6.

[24] M. A. Al-Shabi, "The general toeplitz/observability smooth variable structure filter," Ph.D. dissertation, 2011.

[25] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 45–52.

[26] "Stack overflow bug in ardupilot," Oct 2016. [Online]. Available: https://github.com/PX4/Firmware/issues/5643