Hardware-in-the-loop Testbed for Autopilot Development using Flight Simulation and Parallel Kinematics

Stephan Schulz, Hagen Hasberg and René Büscher
Faculty of Engineering and Computer Science, Hamburg University of Applied Sciences,
Berliner Tor 7, D-20099 Hamburg, Germany
{*stephan.schulz, hagen.hasberg, rene.buescher*}*@haw-hamburg.de*

Keywords: HIL Simulation, Flight Simulation, Unmanned Aerial Vehicles, Robotics, Inertial Sensors

Abstract: This paper presents a testbed for Hardware-in-the-loop stimulation of flight controllers and their sensors in a real-time simulation environment. The framework is especially designed for the development of flight control systems for small unmanned aerial vehicles. The testbed integrates the real-time stimulation of motion sensors by a Stewart platform using the attitude of the simulated aircraft from a real-time flight simulation. Compared to an isolated HIL simulation of a flight controller, this methodology allows simultaneous closed-loop testing of an embedded system with sensors and a flight controller. The robustness of the testbed with a self-developed flight control system for different sensors will be demonstrated in this work. The testbed of a flight simulator with a remotely controlled motion platform will be characterized regarding timing considerations. The applicability for autonomous virtual flight testing using waypoint navigation with a real-time flight controller will be demonstrated.

1 INTRODUCTION

Within the last decade, unmanned aerial systems (UAS) were introduced in a broad range of industrial and civil applications. Besides military applications, numerous civil use cases were developed. The UAS, as a mobile sensor platform, is mainly driven by remote sensing and monitoring capabilities of the versatile payloads. Multispectral detection and imaging combined with georeference data were used in interdisciplinary tasks in the fields of agriculture, archaeology, and photogrammetry. Especially for critical environmental tasks as pollution or fire monitoring, UAS has been adopted. Flight operations by drones transporting medical supplies in sparsely populated areas show a vital contribution to public health services. Future industrial applications require a stronger focus on the reliability and the safety of UAS (Kummer et al., 2014). The upcoming autonomous operations of small unmanned aerial vehicles (SUAV) in urban areas, such as delivery drones, is a demanding task that requires a stronger focus on safety-critical systems.

The model-based design approach enables an enhanced quality of testing by using hardware-in-the-loop (HIL) simulations as a development step (Sampaio et al., 2014), (Guo et al., 2020), (Cai et al., 2009). The reproducible stimulation of an isolated system component, i.e., an inertial sensor unit, interacting in the higher-level function chain, is fundamental for the operational system. In case of a failure, the integration of redundant sensors supports the reliability of systems operation. Additionally, the complexity of flight electronics is scaled up due to enhanced mission requirements. Ensuring the safety and security of more complex autonomous systems should be realized by extended verification and validation in virtual environments on a broader scale. Such virtual environments allow HIL tests of critical electronic components (Bittar et al., 2014), (Pizetta et al., 2016).

In general, HIL verification and validation of flight control algorithms and sensors with a focus on reliability and safety result in a significant reduction of development time and cost (Lepej et al., 2017), (Kang et al., 2019). Despite the substantial costs, HIL testing of flight electronics is an established and mandatory process in the development of safety-critical systems in larger aircraft (Pradipta et al., 2015), (Pradipta et al., 2013), (Yoo et al., 2010). The recent adaption of SUAV in safety-related tasks, like the operation in urban areas, requires HIL test infrastructures for SUAV embedded systems and sensors.

In actual HIL environments, mostly an isolated

flight control unit (FCU) is probed as a main component. HIL testing of the FCU for attitude control is realized for different scopes: In a static test case, HIL tests for a SUAV stimulate the FCU to optimize parameter tuning of attitude control algorithms (Paw and Balas, 2011), (Nguyen and Ha, 2018). Dynamic HIL testing of the FCU is realized by the stimulation of real-time sensor input and monitoring the output for the flight actuators (Khaligh et al., 2014). The integration of a self-developed FCU based on a FPGA in a HIL flight simulator environment was demonstrated but is not shown. In addition to that, HIL real-time attitude testing of an UAV combining the FCU with flight actuators shows the applicability for flight conversion from helicopter and airplane mode (Yoo et al., 2010). Obviously, HIL testing of FCU has been established in UAV development over the last decade.

In our work, we extend the HIL testing of an isolated FCU using a 6-DOF Stewart platform (Stewart, 1965) to stimulate the inertial sensor for attitude control of the SUAV in a flight simulation. The kinematic platform mimics the SUAV attitude from the flight simulator in real-time. The inertial sensor detects the orientation of the platform and generates a sensor input for the FCU. The FCU stimulates the control surfaces of the simulated SUAV, which controls its attitude. The demonstrated HIL testbed is validated with several sensors and simulated flights defined by waypoint navigation. The flight trajectory will be compared to an isolated HIL testing of the FCU. In the following sections, simulations incorporating the Stewart platform are called extended hardware-in-the-loop (HIL). In test cases without the platform, the term processor-in-the-loop (PIL) simulation is used.

2 SIMULATION ARCHITECTURE

The simulation architecture is composed of 4 main system components and is designed for real-time operation. In terms of timing, the pace of the simulation run is driven by the flight simulator. The other components are the Stewart-Gough kinematic platform, the inertial sensor (IMU), and the embedded flight and mission controller. The simulation architecture arranged as a control loop is shown in Fig. 1 with a color-coded (blue) device under test (DUT).

The flight simulator simulates the physical model of the fixed-wing SUAV, broadcasts the motion and the pose estimation, and receives the control values of the control surfaces from the embedded autopilot. The Stewart-Gough platform converts in real-time the virtual attitude into a real attitude. This attitude

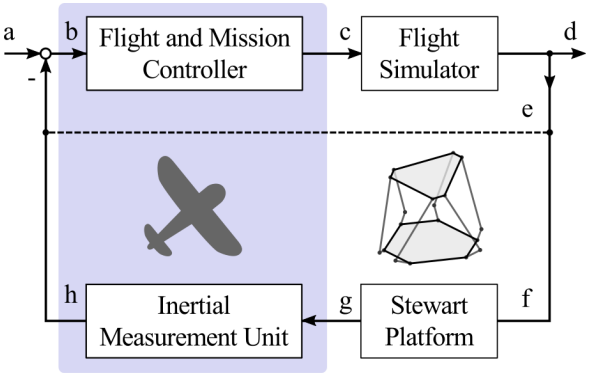

Figure 1: Simulation architecture control loop: (a) mission waypoints, (b) flight path based on current pose information (mission controller), (c) autopilot controls, (d) aircraft state, (e) aircraft state w/o roll and pitch, (f) roll and pitch stimuli, (g) kinematic motion stimulus, (h) measured roll and pitch by inertial sensor.

is measured by an inertial sensor, which is mounted close to the center on the kinematic platform. The embedded flight and mission controller reads the attitude from the inertial sensor in Fig. 1 (solid line). Other pose information like positions, accelerations, and velocities are received from the flight simulator in Fig. 1 (dashed line). The embedded system calculates the control values for mission navigation and sends the control values to the flight simulator. The setpoint of the control loop is represented by the external parameters, mission information, and waypoints. Therefore, the simulation is driven by the flight simulator, but the inputs are provided by the mission controller.

The simulation architecture is suited for rapid prototyping and testing of the flight system components under development. With this HIL testbed, the validation of these components can be demonstrated through measurements in the lab. Besides inertial sensors, the motion platform is capable of being used as a stabilizing platform for different DUT of various kinds, i.e., various payload configurations or gimbals. Since the simulation operates in real-time, testing can be performed with real flight dynamics. In contrast to flight tests with limited duration, the HIL test shows its substantial advantage of reproducibility and accuracy without constraints as limited flight time or positional limits. The simulation architecture can also be used to test or validate different algorithms for flight control. Through the nature of real sensors, it is capable of providing a testbed for the DUT with noise and jitter. The motion stimulation of the DUT is reproducible for different test cases, so real flight attitude and inertial flight states are simulated in the laboratory setup.

Even more complex popular flight controllers in the field of SUAV are designed as embedded

systems with monolithic architectures (Navio2™, Pixhawk™), combining the flight control hardware with various sensors for the estimation of position and orientation. The integrated sensors are tightly coupled to the flight controller logic providing a fixed spatial reference. Therefore, individual and isolated tests are prevented, and the system design forces them to perform measurements and experiments in conjunction only. The simulation architecture presented here is capable of testing these hardware components in isolation. This allows real-time system tests with real noise effects from the dynamics and the environment. Filter optimization for initial configuration is integrated into the development phase of the DUT with the advantage of high reproducibility because of the HIL setup. In Processor-in-the-loop (PIL) mode without including the motion platform, the possibilities for test cases that do not need real attitude mapping are equally vast. The simulation architecture allows the testing of avionics that are interconnected with a bus system, redundant components, and reconfiguring the authorities with that system in real-time or comparing different architectural approaches in general.

3 EMBEDDED FLIGHT AND MISSION CONTROLLER

The embedded system (Fig. 2), which is used in this setup, consists of a flight controller, a mission controller, and also a communication module which interacts with the simulation environment (see Section 5). The flight controller represents the autopilot and is responsible for attitude, speed, heading, and altitude control of the fixed-wing SUAV. In the control cascade, the mission controller prevails over the flight controller for navigating the fixed-wing SUAV through a series of waypoints. The setpoints calculated by the mission controller are processed by the flight controller loop. This represents a common divide and conquer scheme for UAV control strategies (Elkaim et al., 2015).

The embedded flight control system is an in-house development. With the help of the simulation infrastructure, the system performance is validated and verified. Furthermore, a self-developed flight control system ensures maximum flexibility for parameterization to reach a comprehensible and reproducible stimulation of the simulation environment. The flight controller of the embedded system is originated from a mathematical model which is based on a PID controller cascade. To implement the mathematical model Mathworks Simulink™ is used. With Simulink the functionality of the model using model-

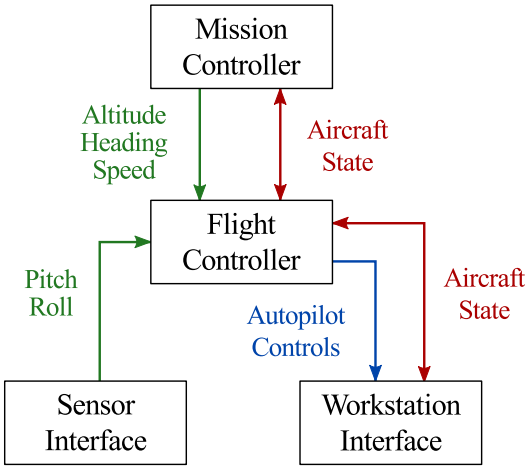

Figure 2: Embedded Flight and Mission Controller: The system components in the HIL setup and the directional signal flow with different data structures are color-coded.

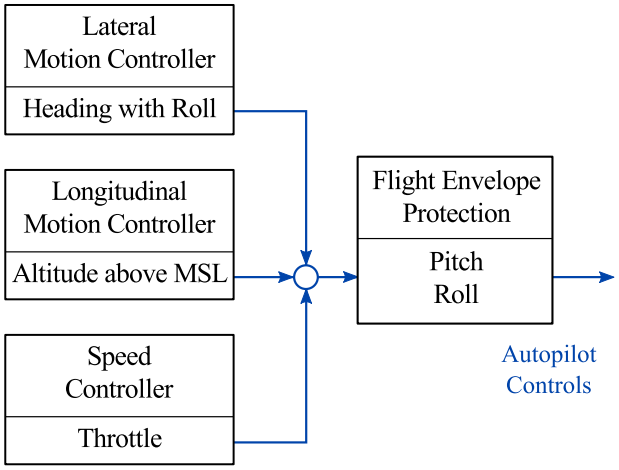

Figure 3: Flight Controller Design: The subsystem components supply the superior moduls for flight envelope protection, pitch, and roll with preprocessed data for the autopilot.

in-the-loop (MIL) simulation was proven. The model was translated into platform-independent and certified C code by Mathworks Embedded Coder™. The generated code was implemented successfully on different hardware platforms, like STM32 or AVR processors. The mission controller was programmed in C and is not a part of the Simulink model. Both modules, the flight and mission controller, were integrated into the embedded system and optimized in higher-level test cases.

3.1 Mission Controller Characteristics

The mission controller is configured with the mission data, which is provided as a list of waypoints. Furthermore, the controller is responsible for the navigation

between these waypoints. The mission data is an external input for the controller, which processes step by step as target points. Therefore the mission controller calculates the required directions based on the pose information to navigate to each waypoint. The mission controller decides whether a waypoint has been reached to complete this individual mission task. A waypoint is considered as reached when a projected radius limit calculated from the current to its specific target position is achieved. The input parameters are the pose information from the flight simulator, a list of waypoints, and the data from the attitude sensors, which is checked by the HIL simulation environment to override the inertial flight states from the flight simulator. The output parameters are heading, speed, and altitude, which are the input parameters for the flight controller.

3.2 Flight Controller Characteristics

The flight controller (Fig. 3) uses the setpoints provided by the mission controller and calculates generic control outputs for attitude control of the SUAV in each step. These outputs are scaled control values in the range $[-1, 1]$, which are mapped to the control surfaces of a specific aircraft by the flight simulator. The value range of the attitude can be limited by configuration, so that a certain flight envelope for roll and pitch cannot be exceeded. The flight controller calculates the generic control output on the basis of six concurrent controllers. These controllers operate concurrently and follow a single-responsibility paradigm. These responsibilities are lateral motion control, longitudinal motion control, speed control, pilot input, aircraft damping, and flight envelope protection.

Each module has different operation modes. For example, the module for controlling the longitudinal motion is allowed to alter pitch angle, climb rate, or altitude as setpoints to steer the SUAV. In this particular case we use a small subset of the available functions only. The lateral motion controller controls the heading of the SUAV and the longitudinal motion controller the altitude. The airspeed is determined by the speed controller. Figure 3 shows the modules with their respective setpoints used in our experiments. The mission controller calculates the required altitude, direction, and airspeed to reach the next waypoint. These values will be transferred from the mission controller to the flight controller, which calculates the required control values for the control surfaces and engines to carry out the mission.

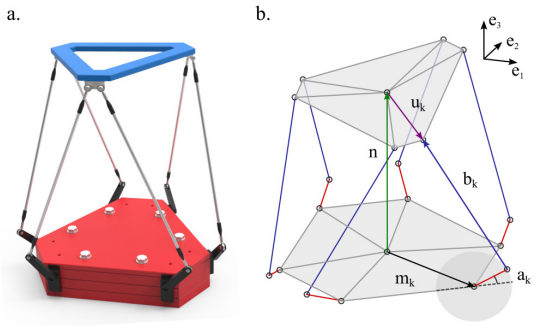

Figure 4: Stewart-Gough Platform

4 KINEMATIC PLATFORM DESIGN

The aircraft's attitude is calculated by the flight simulator and provides the real-time data for the manipulation of the motion platform. A parallel kinematic platform with 6 degree-of-freedom (6-DOF) in a hexapod configuration is used for independent translational and rotational motion in each direction. Because the parallel kinematic platform represents the flight attitude of the SUAV in real-time, the HIL simulation allows joint tests of the flight controller together with sensors for inertial navigation. An inertial measurement unit (IMU) is positioned on the motion platform for the measurement of the real-time flight attitude and is transferring the data to the flight controller.

A Stewart-Gough platform design (see figure 4) is used because of its rigidity, precision, and quick response in all directions, whether an independent or combined linear or angular movement is required (Furqan et al., 2017). The motion constraints are defined by the geometry of the platform. The control software of the 6-DOF platform allows individual mapping of the data provided by the flight simulator to arbitrary linear combinations of the platforms motional axes.

4.1 Operation

The algorithm for calculating the servo angles was developed in Matlab and translated with Matlab^{TM}Coder into platform independend C code. The control software is based on an algebraic derivation of the required angular positions of the servo motors with an exact solution. All specific mechanical variables are taken into account to obtain the corresponding motor positions for each required position of the platform head. The platform is based on a mechatronic design with rotary drives (Stewart,

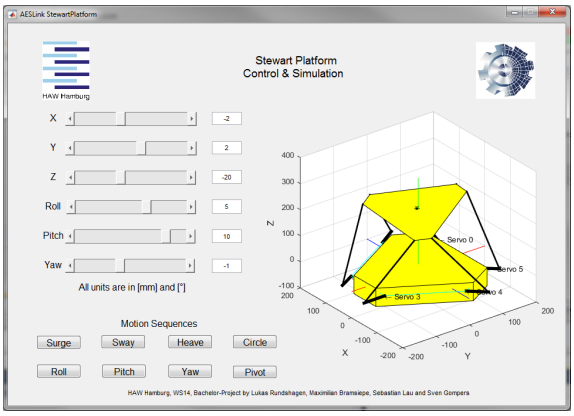

Figure 5: GUI for controlling and simulating the platforms motions.

1965) in order to achieve a small volume of the overall mechatronic system with the required positioning speed of the cost-effective drives and an optimal workspace coverage (Majid et al., 2000). A graphical user interface for manually controlling and simulating the platform is shown in figure 5.

The platform is controlled by a low cost embedded system (embedded Motor Control Unit, eMCU) which is located within the base of the platform. The motion is generated by COTS model-grade servos controlled using PWM. The connection between the host PC and the embedded Motor Control Unit is realized via UART. The eMCU accepts two different sets of parameters. First, the raw servo angles can be sent. This is useful, when the Matlab based control software shall be used on the host PC. The second option is to send the three rotational and translational control values. In this case, the servo values are calculated within the eMCU by the generated code.

The platform was assembled by hand. This results in rather inaccurate screw and strut placements. Hence, a static offset calibration without dynamic motion has to be performed. This was done by adjusting the neutral points of each servo in a levelled position. The calibration was done relative to the gravitational field of the earth measured by the inertial sensor. This can result in a systematic error, since the calibrating was not done in respect to the platforms base. In the case of an inclined base, a constant offset error with the magnitude of the inclination could decrease the long term performance significantly.

4.2 Delay Measurements and Rotational Dynamics

For the initial characterization of the kinematic platform's dynamical performance, its step response was measured. The process was controlled by an embedded system explicitly implemented for this task. Using a dedicated embedded system ensures that this measurement is done as close as possible to the platform, and no other timing delays like communication to a PC-based host system distort the measurements. This embedded system held the platform in a leveled position to ensure a steady-state. After a couple of seconds, a step function of 5° in the roll axis results in a moving platform with a time delay of about 80ms (Fig. 6). The final position is reached after roughly 280ms. The steepest slope determines the maximum rotational velocity of the platform of about $65°/s$. This measurement estimates the suitability of the platform for HIL flight simulation.

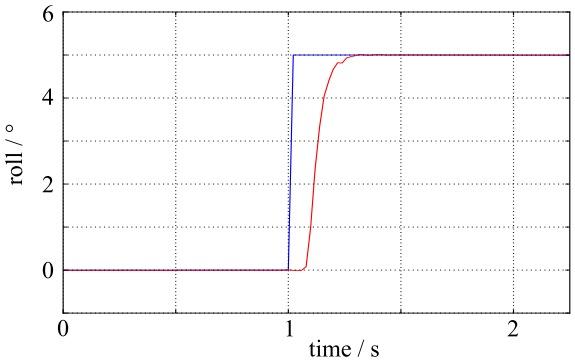

Figure 6: Time-dependent step response of the kinematic platform for a tilt of 5° in a specific direction: The command is shown in blue, the response in red. The small deviation in the shape of the response curve at the end of the movement is due to timing considerations.

Because of the parallel kinematics design, a specific platform tilt in a direction with different heading requires a new set of trajectories for the manipulators. In the measurement, a small deviation in the shape of the curve occurs, shortly before the maximum rotation is reached. This deviation from the ideal movement is caused by non-synchronized servo control. The control software computes the servo output values with an asynchronous point-to-point motion, due to the sequential data transmission and the constant rotational speed of the servos, there are minor deviations in the time each servo arrives at the target position. This behavior results in motions, where rotation in a single motional axis is shifted to another for a short time. Without synchronous compensation in the motion planner algorithm, this behavior is inevitable. Further consideration shows only a small impact.

5 HIL TESTBED SETUP

The simulation framework consists mainly of the flight simulator and a software bus. The flight simula-

tor calculates the motion and pose information of the simulated SUAV. The distribution of this information represents the central clock of the testbed. Therefore, the simulation rate is bound to the framerate of the flight simulator. The communication backbone of the simulation framework is the self-developed AESLink software bus (Airborne Embedded Systems Link). All messaging between the different modules of the simulation is realized using this communication bus.

5.1 AESLink Software Bus Characteristics

The AESLink software bus was specifically designed for distributed messaging between integrated or spatially separated modules in a distributed flight system (Hasberg, 2014). The software bus defines the messages and their formats. On the host side, it's based on an UDP-multicast network, enabling simulation runs on distributed workstations. The unified message format allows arbitrary embedded systems to participate in the simulation. This is done through several proxy applications, which convert the UDP messages to and from the physical interfaces supported by the embedded systems, like UART, CAN, or SPI. Since standard PCs usually don't support these interfaces, COTS USB converters can be used. AESLink specifies several message types with its main ones:

- Aircraft-State: Calculated by the flight simulator. Contains pose information like position, attitude, acceleration, velocity, etc.
- Autopilot-Controls: Calculated by the autopilot. Contains control values for the aircraft, i.e., roll, pitch, throttle, or yaw control.

The messages define their content, but they are not fixed to any specific application. The sources and sinks of the messages can have any arrangement in the overall setup.

The simulation framework consists of several additional applications and functions. These are for example a SIL-Autopilot that implements a flight controller in a desktop application, a logging application, a replay application, or a packet simulator. These applications support a rapid prototyping development process because of its debugging and reproducibility features. As stated above, the component instantiations are not constrained to particular systems. The functionalities can be placed in arbitrary locations within the network, host, or in an embedded system.

Nonetheless, there is a typical flow of information, where the flight simulator is the source of aircraft-state messages and a sink for autopilot-control messages. In the current state of development, X-Plane,

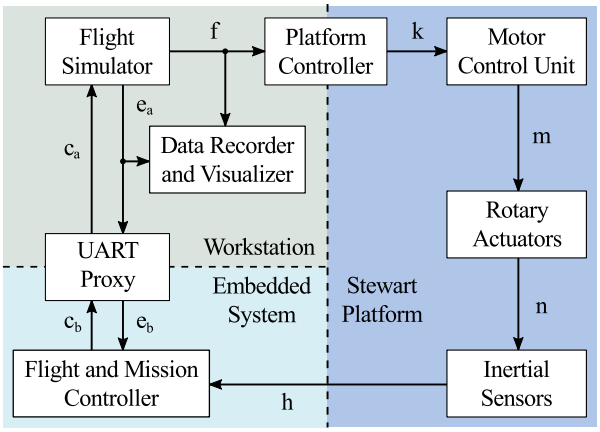

Figure 7: Testbed Architecture: The modules are assigned to the systems Stewart platform, embedded system and workstation. The platform controller and the UART-proxy are used for data distribution and translation. The interfaces for data transfer c_a, c_b, e_a, e_b, f, k, m, n, and h connect the modules of th real-time system.

AeroSimRC, and the packet simulator can be used as a flight simulator. An interface for Simulink based flight dynamic models is implemented. Autopilot-controls messages are typically generated and sent by components, which implement flight controller functionalities. These are a real-time version of the flight controller presented in section 3.2 running in Simulink (MIL), the SIL autopilot host application, or several embedded systems interfaced by the described proxy applications. This shows the flexibility of the framework since an autopilot can be instantiated in Simulink (MIL), as a workstation application (Software-in-the-loop, SIL) or in an embedded system (PIL, HIL), which are based on an FPGA or on microcontrollers like STM32- or AVR-based types. Other utility applications that were developed include real-time logging, plotting, replay, and a display applications. These tools greatly enhance the flexibility and capabilities of the software bus.

5.2 AeroSimRC Flight Simulator

In principle, everything that calculates the dynamic model of a SUAV can be denoted as a flight simulator, whether it is a workstation application, a real-time Simulink model, or an external full-scale flight simulator. For the AESLink software bus, X-Plane, AeroSimRC, and Simulink real-time models are supported.

In this work, the commercial PC flight simulator AeroSimRC was used. This simulator is specifically designed to simulate model aircraft. These compare better to SUAVs than full-sized aircraft. The low inertia and high agility of model aircraft suit the require-

ments for the experiments. For this reason, we decided to use AeroSimRC over X-Plane for calculating the dynamic model of the SUAV. The flight simulator was extended via a plugin. This plugin connects the simulator to the software bus and lets it participate in the AESLink simulation framework.

In every frame-cycle, an aircraft-state message containing the motion and pose information is broadcasted, and an autopilot-controls message containing the control values for the control surfaces is received. The simulator uses the received controls instead of the inputs from the keyboard or joystick for its calculation of the next frame-cycle. The plugin allows the framerate of the simulator, and therefore of the simulation overall, to be adjusted at runtime. The HIL experiments were realized at 60 frames per second.

5.3 Testbed Architecture

The system architecture of the testbed (Fig. 7) allows different types of simulation. Here, PIL and HIL simulations were realized. The infrastructure can be divided into the three areas Stewart platform for dynamic stimulation, Embedded System representing the flight system, and Workstation for the simulation regarding the interaction of the SUAV with the environment.

In a PIL simulation, the flight simulator is the central component: It simulates the behavior of the SUAV interacting with the environment, periodically broadcasts its state (e_a) and receives autopilot-controls. The flight simulator allows manual inputs with the keyboard, but this option was omitted during the simulation runs. The UART-Proxy application converts the UDP messages sent by the flight simulator into a serial stream of bytes and sends the data to the embedded flight and mission controller (e_b). The embedded system receives the data and runs its algorithms. The mission controller does the navigation and path planning. The flight controller calculates the required control values to follow the calculated path flight. The calculated control variables are encapsulated in an autopilot-controls messages and are sent to the UART-Proxy (c_b). From there on, the data is converted to an UDP message and sent to the flight simulator (c_a). The flight simulator receives the data and uses them for the next simulation step, which closes the control loop. For debugging and reproduction purposes, the simulation is being recorded and plotted in real-time.

The HIL simulation extends the pure virtual electronic loop. Here, the roll and pitch angles are no longer provided by the flight simulator directly. Instead, they are read from the IMU, which is mechanically mounted on the platform. Since the software bus works in a multicast fashion, multiple applications can receive the broadcasted messages. For efficient control of the kinematic platform with the data from the software bus, a desktop application was developed (Fig. 8). This platform controller takes the pose information (f) and maps them to the axis of the Stewart platform. The mapping is not static and can be adapted in any way, e.g., acceleration, velocities, and magnetic fields can be mapped to any rotational or translational axis. For our tests, the roll and pitch angles from the simulated SUAV where directly mapped to the roll and pitch axis of the platform.

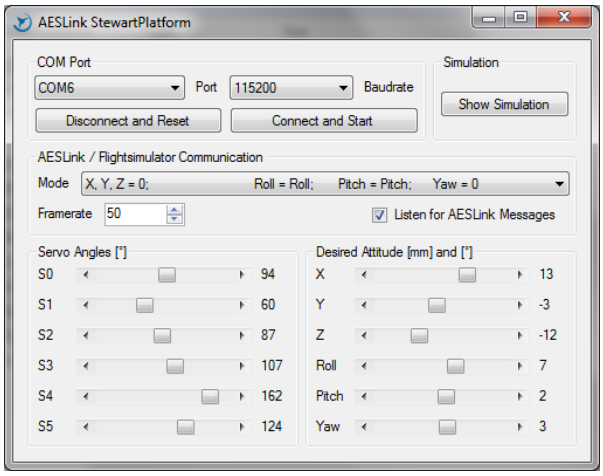

Figure 8: Platform control desktop application: The self-developed allows direct positioning of the servo angles for system checks, and direct attitute control for simulation purposes. Different modes were implemented for a configurable allocation of phase-space variables.

The axis values are sent via a serial connection (k) to the motor control unit (MCU), which is embedded in the base of the platform. The embedded MCU calculates the servo rotation angles, which are needed to positioning the platform and, e.g., to mimic the attitude of the simulated SUAV. The servos are fed with the values via standard pulse width modulation (m). The resulting motion of the platform takes effect on the inertial sensor (n). The values for roll and pitch (h) are read by the flight and mission controller and used for its calculation. The remaining motion and pose information is still received from the flight simulator (e_b), only the roll and pitch values are replaced. Similar to the PIL simulation, the calculated control values of the autopilot are sent back to the flight simulator closing the control loop.

5.4 Testbed Latencies

The latencies in a complete system cycle are essential parameters for the applicability of the HIL testbed

Table 1: Delay measurement of the autopilot response times: Dependent on the simulation mode, reaction times of the system were measured. The signal path shows the data flow (Fig. 7).

Mode	Signal path	Reaction time
SIL	$e_a c_a$	$<$1ms
PIL	$e_a e_b c_b c_a$	˜22ms
HIL/SP	$fkmnhc_b c_a$	˜121ms

using virtual flight testing. For the latency estimation, time delays for a closed cycle from a command sent to the reaction detected by the FCU were measured. Therefore, the time to respond and react to a step function in the pose information is analyzed. The analysis of the response times of the testbed in Table 1 shows increasing values for SIL, PIL, and HIL configurations as expected.

The SIL mode demonstrates that the host-only UDP-based software bus does not introduce any significant latency to the framework. In PIL mode at a baud rate of 250kBaud, the predominant portion of the 22ms latency is introduced by the transmission over the serial connection. If we assume the signals at k and e_b as well as e and c need the same amount of time (Fig. 7), the connections m, n and h introduce roughly 100ms of latency. The signals at m, n, and h span the calculation of the servo angles and the control of the servos, the generation of momentum, and the movement of the platform as well as the stimulation and sampling of the IMU. This time is roughly equivalent to the step response of the kinematic platform (Fig. 6). Both results show a response time in the order of 80ms to 100ms. The results show that the time behavior of the HIL simulation is not adversely affected by additional individual components.

6 SIMULATION RESULTS

The applicability of the HIL simulation using the presented testbed architecture is analyzed with simulated flight operation of a SUAV. The real-time capabilities and the accuracy of the extended HIL simulation using the Stewart platform are validated against the PIL simulation without any stimulation of the inertial sensor system based on the integrated FCU. The direct comparison of the flight path based on the extended HIL and PIL configuration for a predefined list of waypoints is a full-scale test case for the testbed. All noise and propagation delays are adding up in time to an upper bound representing a deviation caused by the motion stimulation.

Initially, internal and external calibration procedures for the kinematic platform itself and the inertial sensor mounted on the kinematic platform are required. At first, a static calibration of the servos compensates the mechanical deviations of the platform to minimize position and attitude error (section 4.1). A fixed position and orientation of the inertial sensor on the platform is not required due to the second calibration step. In the dynamic calibration, the orientation of the internal coordinate system of the sensor is detected and virtually aligned to the motion axes of the kinematic platform to minimize stimulation errors. A delay measurement of the platform was realized to get an estimation of the platform's dynamic response. The results are shown in sections 4.2 and 5.4 and indicate that the system meets the required timing constraints. To validate the capabilities of the testbed for the task of carrying out an SUAV mission, we compared the performance of the extended HIL simulation to the PIL mode. The results are presented in section 6.2.

6.1 Dynamic Calibration

Stimulating the sensor by the Stewart platform requires precise knowledge of the mechanical deviation of the rotational motion axis. A deviation in the orientation between coordinate systems of the platform and the IMU results in an increasing error over time. The measured state diverges from the proposed state of the simulated SUAV, and this error accumulates over time. To minimize the error, the roll and pitch axis systems have to be aligned by a rotation matrix, which is determined by a dynamic calibration algorithm. The calibration process maps roll, pitch, and yaw axes of the inertial sensor to the axes of the Stewart platform. Because of the motion characteristics of the parallel kinematic platform, any orientation of the coordinate system can be used as a zero position. The calibration process is generalized for this method, but individual in terms of the offset values for each inertial sensor. The sensor can be mounted in any orientation. The calibration process defines the mapping between the orientation of the inertial sensor and the axis of the platform. The process also compensates for possible misalignments in the mounting.

A detailed explanation of the calibration process follows. The initial orientation of the inertial sensor is arbitrary. As stated above, we assume that the platform is calibrated in regards to the servo control inputs. In other words, the axis of the platform are aligned to the axis system of the flight simulator. For the calibration of the mounted sensor, we define a cartesian coordinate system. One of the axes (here roll) gets stimulated with a sinusoidal motion. If the axis of the sensor and the platform are not collinear,

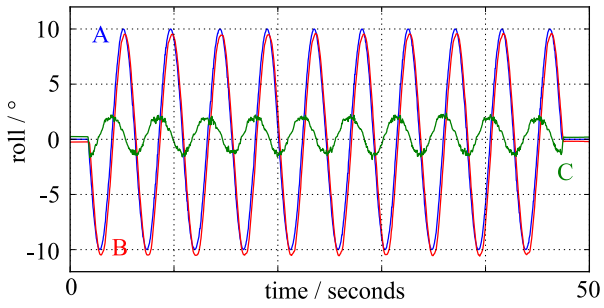

Figure 9: Example of external calibration signals: The platform is stimulated on the roll axes (A), and the roll angle (B) are measured with the pitch angle (C). The pitch angle is used as an error signal, which vanishes once the orientation of the platform and the inertial sensor is aligned.

other axis get stimulated as well. Figure 9 shows this behavior. Through misalignments of the sensor, not only the roll axes (red) is stimulated, the pitch axes (green) is affected as well. To compensate for this misaligned we can mathematically rotate the axis system of the sensor until the measured stimuli of the other axis are minimized. After the virtual rotation, only the desired axes gets stimulated. This ensures that the outputs of the inertial sensor is aligned to the axis system of the flight simulator. This is necessary for long time stability of the system, since otherwise the errors in the simulated and measured pose information would drift and produce a large difference between the PIL and the extended HIL simulation. This calibration procedure can be automated and completes when the error reaches it's minimum value. It results in a high precision, since the calibration is done over a long time period and the fit is calculated from a significant amount of measured data points. This calibration process correlates to an offset calibration for inertial measurement systems in aircraft systems (Magnussen et al., 2012).

Virtually rotating the sensor axis system results in creating collinearity with the platforms axis system. To find the rotation parameters, the roll axis was stimulated with a sinusoidal motion with a magnitude of 20 degrees around the leveled position. The system was stimulated with 10 periods in a total time of 25sec.

The calibration process was successfully realized with several sensors like a VectorNav VN-200, a Bosch BNO055 and a MicroStrain 3DM-GX2. Figure 9 shows the uncalibrated signals of the roll (red) and pitch (green) axis from the VectorNav VN-200. The pitch axis shows a severe misalignment, since it gets stimulated as well. The residual error in the rotation offsets were determined using the least square optimization. The pitch axis was rotated in discrete steps. The steps had a magnitude of 10 degree around zero and a step size of 0.01 degree, which is sufficient within the mechanical accuracy of the platform. In every iteration, the error was calculated using the squared norm. Plotting the error results in a parable. The polynomial nadir represents the angle that pitch has to be rotated around. This results in the smallest stimulation of the pitch axis, when actually only roll shall move.

The calibration process described above was done for the pitch and the yaw rotational axis. For all sensors we found that there is a delay induced by the sensors, probably through their internal filters and dynamic behaviors. The calibration process produced reproducible results for all sensors. For the VN-200 which we used in our experiments, the pitch offset yielded in -0.52 degrees with an error of $<0.17\%$, the yaw offset in -0.44 degrees with an error of $<0.08\%$. The measured accuracy of the motion platform is sufficient for the sensors used in the simulations.

6.2 Flight Path Measurements

The flight path measurements are the central element to present the qualification of the extended HIL simulation in comparison to the PIL mode. The flight simulator is used to generate the stimuli for the motion platform. It is controlled by the mission and flight controller to generate reproducible flight paths. In this setup, the system can also be used for the development and test of stabilized components, like gimbals. The flight path of the extended HIL simulation is compared to the flight path of the PIL simulation. The differences in the simulation results are discussed in this chapter. The flight measurements were performed with different inertial sensors. The presented results of the extended HIL flights were recorded with a VectorNAV VN-200. The whole flight data was obtained from the flight simulator while the attitude was sent to the motion platform. The platform stimulates the inertial sensor, whose values will be read and processed by the mission and flight controller.

The simulations were executed with the flight simulator AeroSimRC with an update rate of 60 Hz and a *Trainer 40* fixed-wing model aircraft. The flight path is predefined as a list of waypoints. These points are used by the mission controller to navigate the simulated SUAV. The calculated values from the mission controller are used by the flight controller to steer the SUAV. Since the entire function chain, excluding the flight simulator, is an in-house development, a high level of reproducibility is given. A waypoint is reached when a minimum distance, based on the lateral plane, is approached by the mission controller. Each waypoint consists of a longitude, latitude and al-

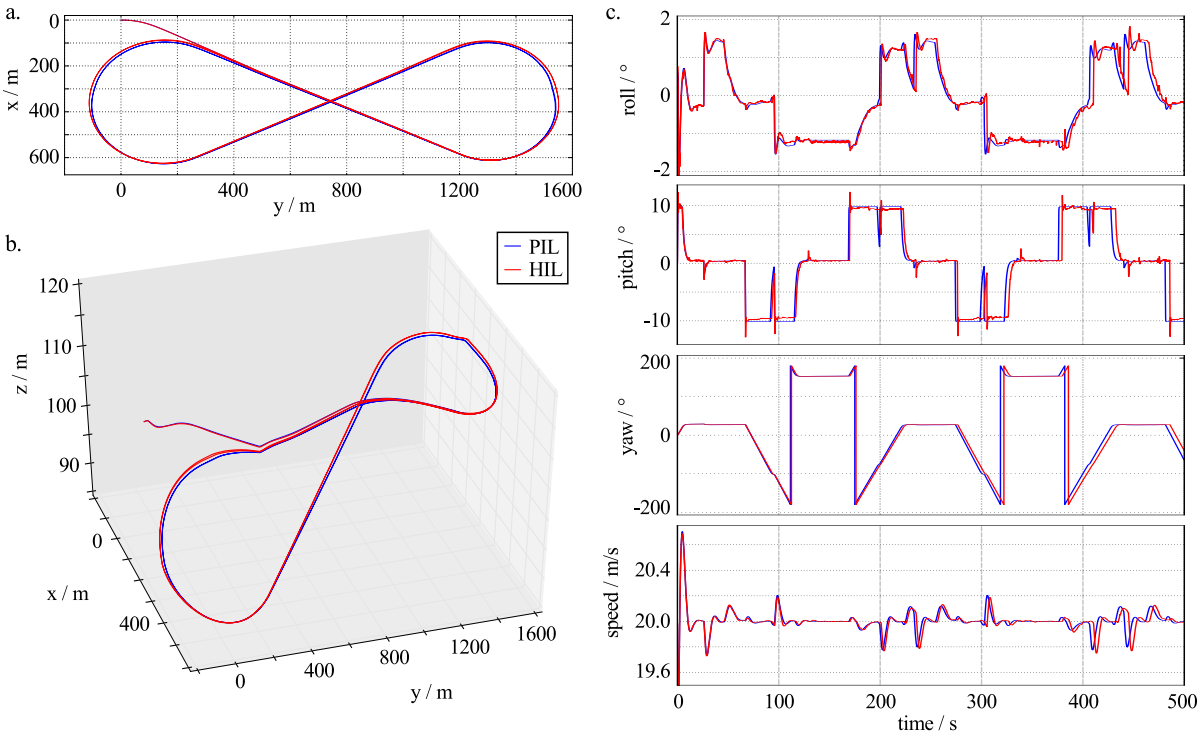

Figure 10: Flight Data: (a) 8-Figure in 2D Space, (b) 8-Figure in 3D Space, (c) Pose Information

titude. The airspeed of the UAV is controlled by the flight controller to keep the value constant.

The trajectory presented here is an inclined 8. It is defined through six waypoints with different altitudes. In 3D space, the eight was flattened in the x- and y-plane. The starting point is outside of the trajectory and the SUAV starts its flight immediately with the defined target speed. The motion and pose information of the flight is recorded during the full simulation. The simulation is limited to ten minutes flight time, in order to be able to identify accumulated differences over time. The stewart platform has a limited workspace, which defines the limits of the flight envelope. The roll and pitch angles were limited to 10 degrees. The climb rate to 0.33m/s. A waypoint is reached when the horizontal distance is smaller than 200m. The geometrical dimensions of the curve is about 600m in x-, 200m in y-axis and 20m in height. A constant speed of 20m/s was set as target speed.

The flight tests were executed in PIL and extended HIL mode. Based on the flown trajectory and the telemetry data, the total flight phase can be evaluated. Figure 10 shows the trajectory in 2D (a) and 3D (b) space. The results show that the 10-minute simulation flight shows a good match. Locally there are small differences between the PIL and extended HIL trajectory, but there is no global drift. During the 12 km long flight only minor deviations occur, smaller than

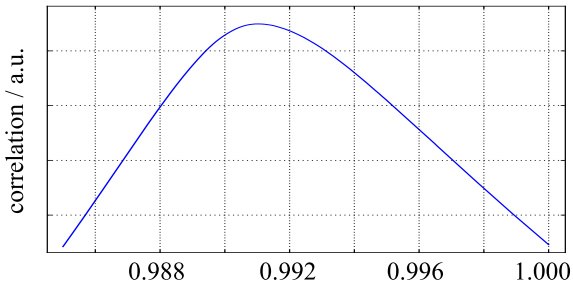

Figure 11: Correlation Coefficient between the two setups

12 metres. Roll, pitch, yaw and speed (see figure 10, (c)) from both modes are almost identical. Over the total flight time of 500s the comparison of PIL and extended HIL results to very good consistency in the measured data.

The similarity of the signal patterns of roll, pitch, yaw, and speed concludes that the control loop of both setups are almost unchanged. Further analysis of the signal shapes show a small linear drift over time of the extended HIL compared to the PIL mode. This probably corresponds to the small time delays in the control of the motion platform and its inertia (see table 1). When the SUAV e.g. rolls, the time until the change in attitude is propagated to the flight controller is higher in the extended HIL mode. This results in a slightly larger turn radius, which, although the speeds match, results to the HIL mode SUAV being slightly

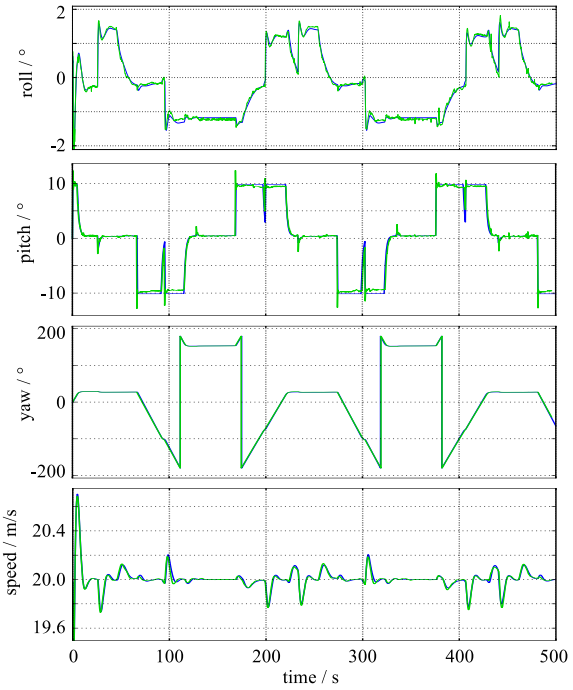

Figure 12: Time Corrected Pose Information

positioned behind the PIL mode SUAV at any given after the first roll maneuver. This error in position accumulates over time. This can be seen in figure 10 (c), where the paths match, but the extended HIL maneuvers are performed a few seconds later. At a flight time of 500s, the time delay is about 8s, or 1.6%. After correction of the linear drift (see figure 11), the signal shapes are in an excellent match (see figure 12). Locally, the deviation has a maximum of about 2.6%. The evaluation shows the delays induced by the software control and inertia of the motion platform. At the same time, the attitudes of the extended HIL and PIL simulations match very good. A second figure (square) was simulated as supplement. The evaluation and quality was identical.

7 CONCLUSION

This paper shows that an extended HIL framework with a stewart platform is suitable for scenarios, where real flight attitudes are needed. The analysis of the simulation results shows that the delays in the software control loop and the mechanical inertia don't induce a significant impairment. The delays of the stewart platform accumulate to roughly 100ms. This results in a linear temporal drift over time of only 1.6% compared to the PIL scenario, where the control loop is comprised of only electrical signals and no mechanical signal transfer exists. The stewart platform and in general the presented simulation framework therefore is ideally suited for the reproducible development, integration and testing of sensors and stabilization systems under the influence of real flight attitudes in the laboratory. In order to verify the results in a real use case and to have full control over the parameters, we have developed a flight and mission controller based on an FPGA. The system has first controlled a simulated SUAV on the stewart platform in the laboratory and later successfully flown a DORNIER DO-27 model aircraft.

The system can now already be used for different in-house projects, including comparing different inertial sensor systems, the development of camera gimbal control systems or as presented, developing and improve flight control algorithms. Future work may include replacing the simple point-to-point control algorithm of the platform by a continuous path control algorithm, to reduce the discontinuities of the platforms motions.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support of the German Federal Ministry of Economics and Energy (BMWi) for this project under the ZIM program.

REFERENCES

Bittar, A., Figuereido, H. V., Guimaraes, P. A., and Mendes, A. C. (2014). Guidance software-in-the-loop simulation using x-plane and simulink for UAVs. *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings*, pages 993–1002.

Cai, G., Chen, B. M., Lee, T. H., and Dong, M. (2009). Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters. *Mechatronics*, 19(7):1057–1066.

Elkaim, G. H., Pradipta Lie, F. A., and Gebre-Egziabher, D. (2015). Principles of Guidance, Navigation and Control of UAVs. *Handbook of Unmanned Aerial Vehicles*, pages 347–380.

Furqan, M., Suhaib, M., and Ahmad, N. (2017). Studies on Stewart platform manipulator: A review. *Journal of Mechanical Science and Technology*, 31(9):4459–4470.

Guo, A., Zhou, Z., Zhu, X., Zhao, X., and Ding, Y. (2020). Automatic control and model verifica-

tion for a small aileron-less hand-launched solar-powered unmanned aerial vehicle. *Electronics (Switzerland)*, 9(2):26.

Hasberg, H. (2014). *A test concept for flight controllers*. Bachelorthesis, Hamburg University of Applied Sciences.

Kang, D., Kim, S. H., and Jung, D. (2019). Real-time Validation of Formation Control for Fixed-wing UAVs using Multi Hardware-in-the-Loop Simulation. *IEEE International Conference on Control and Automation, ICCA*, 2019-July:590–595.

Khaligh, S. P., Martínez, A., Fahimi, F., and Koch, C. R. (2014). A HIL testbed for initial controller gain tuning of a small unmanned helicopter. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 73(1-4):289–308.

Kummer, N., Jee, C., Garbowski, J., and Nowak, E. (2014). Design and Development of the Hardware for a Vision-based UAV Autopilot. *Proceedings of The Canadian Society for Mechanical Engineering International Congress 2014*, pages 1–6.

Lepej, P., Santamaria-Navarro, A., and Sola, J. (2017). A flexible hardware-in-the-loop architecture for UAVs. *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 1751–1756.

Magnussen, Ø., Ottestad, M., and Hovland, G. (2012). Calibration Procedure for an Inertial Measurement Unit Using a 6-Degree-of-Freedom Hexapod. *International Conference on Unmanned Aircraft Systems (ICUAS)*.

Majid, M. Z., Huang, Z., and Yao, Y. L. (2000). Workspace analysis of a six-degrees of freedom, three-prismatic-prismatic-spheric-revolute parallel manipulator. *International Journal of Advanced Manufacturing Technology*, 16(6):441–449.

Nguyen, K. D. and Ha, C. (2018). Development of Hardware-in-the-Loop Simulation Based on Gazebo and Pixhawk for Unmanned Aerial Vehicles. *International Journal of Aeronautical and Space Sciences*, 19(1):238–249.

Paw, Y. C. and Balas, G. J. (2011). Development and application of an integrated framework for small UAV flight control development. *Mechatronics*, 21(5):789–802.

Pizetta, I. H. B., Brandão, A. S., and Sarcinelli-Filho, M. (2016). A Hardware-in-the-Loop Platform for Rotary-Wing Unmanned Aerial Vehicles. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 84(1-4):725–743.

Pradipta, J., Klunder, M., Weickgenannt, M., and Sawodny, O. (2013). Development of a pneumatically driven flight simulator Stewart platform using motion and force control. *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics: Mechatronics for Human Wellbeing, AIM 2013*, pages 158–163.

Pradipta, J., Knierim, K. L., and Sawodny, O. (2015). Force trajectory generation for the redundant actuator in a pneumatically actuated Stewart platform. *ICARA 2015 - Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications*, pages 525–530.

Sampaio, R. C. B., Cazarini, E., Hernandes, A. C., Becker, M., Magalhaes, D. V., and Siqueira, A. A. (2014). HiL evaluation of an on-chip-based optimal H-Infinity controller on the stability of a MAV in flight simulation. *IEEE Aerospace Conference Proceedings*.

Stewart, D. (1965). A Platform with Six Degrees of Freedom. *Proceedings of the Institution of Mechanical Engineers*, 180(1):371–386.

Yoo, C. S., Kang, Y. S., and Park, B. J. (2010). Hardware-in-the-loop simulation test for actuator control system of smart UAV. *ICCAS 2010 - International Conference on Control, Automation and Systems*, pages 1729–1732.