`DroneLibrary`: Multi-domain Drone Modeling in Modelica

Meaghan Podlaski[1] Luigi Vanfretti[2] Dietmar Winkler[3]

[1]GE Research, Niskayuna, NY, United States, meaghan.podlaski@ge.com
[2]Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA, vanfrl@rpi.edu
[3]Department of Electrical Engineering, Information Technology and Cybernetics, University of South-Eastern Norway, Porsgrunn, Norway, dietmar.winkler@usn.no

Abstract

In the development of complex, novel electrified aerial systems such as Unmanned Aerial Vehicles (UAVs) and electric vertical take-off and landing (eVTOL) systems, multi-domain modeling and simulation studies can provide indispensable insight on system design and performance. In this paper, a Modelica library that can be used to model multi-domain drones is introduced. This library models a drone in the electrical, mechanical, and control domains, with examples for applications such as battery-power analysis, virtual reality simulation and user interaction. *Keywords: drone, quadcopter, eVTOL, UAV*

1 Introduction

The `DroneLibrary` was created in response to the growing need for simulation-based studies in Unmanned Aerial Vehicle (UAV) drone and electric vertical take-off and landing (eVTOL) electric aircraft for Urban Air Mobility (UAM) (Doo et al. 2021). Multi-engineering domain, simulation-based studies are valuable in determining which design concepts and methods for eVTOL systems can best meet design requirements and specifications (Podlaski, Niemiec, et al. 2021; Podlaski, Vanfretti, et al. 2022). There are limited opportunities for creating and testing physical prototypes for such systems due to time and monetary constraints. These systems also have limitations on which physical device qualities are recorded during testing, making well-defined reliable models valuable at early design stages for eVTOL systems, such as those seen in conventional fixed-wing distributed electrical propulsion physical prototype systems (Borer et al. 2016)

To bridge this gap, the authors' prior efforts in this area (M. Podlaski 2020; M. Podlaski 2021) have resulted in a Modelica library that includes multi-domain models to represent each subsystem of a quadcopter, specifically focusing on the mechanical, electrical, and control domains. This library contains examples of the quadcopter model at multiple levels of detail under different operating conditions. The component models are developed in a manner to easily replace them for different simulation applications, creating replaceable models that are easy to maintain with broad application scope.

1.1 Related Works

This library builds off of the works in (M. Podlaski 2020), which show initial models and simulation results for the `DroneLibrary`. The aerodynamic and mechanical behavior of these models are derived from (Bresciani 2008) and (Luukkonen 2011).

System-level drone models created using Modelica have previously been discussed in (Kuric, Osmic, and Tahirovic 2017). However, these systems mainly focused on multirotor aerial vehicle (MAV) dynamics modeling while assuming ideal and constant power consumption. All dynamics in this MAV Modelica model are reduced to a single domain, linear model. In this library, non-idealities in the electrical system are introduced.

A drone PID controller developed using Modelica is discussed in (Ma, Li, and Nae 2016). This model assumes that the drone body is rigid and symmetrical with the force of each propeller proportional to the square of the angular speed of the propeller. The models in the `DroneLibrary` build off the ideal assumptions made in this work by adding flexibility to accommodate different airframe structures through the multibody mechanics.

In contrast with the aforementioned previous works that aims in modeling system dynamic aspects of specific drone or eVTOL systems, the work in (Coïc, Budinger, and Delbecq 2022) targets drone sizing and trajectory optimization aspects. While the library proposed could be used for these purposes too, this is out of the scope of the present paper.

Finally, it is necessary to note that the present work has its origin as a course project by Hao Chang in 2018 at Rensselaer Polytechnic Institute that was the basis of the start of the `DroneLibrary`. At that time there were no commercial or open source libraries offering the capabilities that have been now consolidated into the `DroneLibrary`. To the authors' knowledge, the `DroneLibrary` is the only open source license free library with the described scope of applications. In the case of commercial and proprietary libraries, with the release of Dymola 2022, Dassault Systèmes included the Aviation Systems Library (AVL), which offers a similar scope for multi-copter and extends it to fixed wing aircraft. Similarly, Claytex released a UAV Dynamics Library in 2020[1],

[1]There is limited publicly available information about these li-

that now has a similar scope and application areas as the AVL[2]. While these commercial libraries have a broader scope, the `DroneLibrary` provides an open-source option that reduces the barrier of entry for drone modeling and can be seen as a starting point before working with these specialized libraries if the need arises.

2 Library Overview

The package structure of the `DroneLibrary` is shown in Figure 1. Test cases of the quadcopter are located in the `Examples` package. The `Blocks` package has models used to create the controllers and input signal commands for the quadcopter. The `Electrical` and `Mechanical` contain models at varying levels of modeling fidelity representing the electrical and mechanical components used in the drone, such as motors, chassis frames, and power sources. The `Sensors` package contains sensor models used to track the drone's location and acceleration during flight to assist in the control. The `Visualization` package utilizes the DLR Visualization library to simulate the quadcopter in interactive virtual reality environments. This package contains multiple examples for interacting with the drone model via hardware-in-the-loop (HIL) using keyboard and joystick commands.

The library is dependent on the other following libraries:

- Modelica Standard Library, v4.0.0 (Modelica Association 2023)
- DLR Visualization Library Professional Edition, v1.6.0 (Hellerer, Bellmann, and Schlegel 2014), visualization examples only.
- Modelica Device Drivers Library, v2.1.1 (Thiele et al. 2017), visualization examples only.

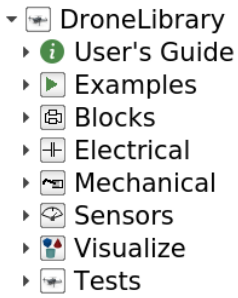

Figure 1. `DroneLibrary` package structure.

3 Examples

All of the components described herein are configured to create the simple quadcopter model shown in Figures 2 and 3. Figure 2 shows the system when an ideal power source is used, while Figure 3 shows the system with a battery power source. The quadcopter chassis in Figure 3 also has the `frame_a1` connector that can be linked to additional external payloads, such as a camera.

These quadcopter variants are tested in models configured in the `Examples` package. The examples include using different input signals to control the inputs `xcoord`, `ycoord`, and `zcoord`, which can be signals provided in the `DroneLibrary`, Modelica Standard Library, experimental data, and custom signal functions defined by the user. The inputs for the quadcopter model can also be left disconnected from any inputs and compiled as a Functional Mock-up Unit (FMU). By selecting this option, the model can be exported to other software tools for analysis and simulation.

4 Electrical Models

The models for the drone's motors and power sources are located in the `Electrical` package.

4.1 Source Models

The Modelica Standard Library includes a battery stack model that is included in the `Examples.DroneWithoutIdealPower` package, which is used when a battery model is desired.

The battery model is configured to interface with the rest of the drone's power system using two different power electronic typologies. One of these topologies is shown in Figure 4, where the battery model in the Modelica Standard Library is connected to a step-down DCDC converter to provide power to the drone's main controller unit (MCU). The other topology in the package includes an additional set of electrical pins to connect the battery to each of the drivetrains.

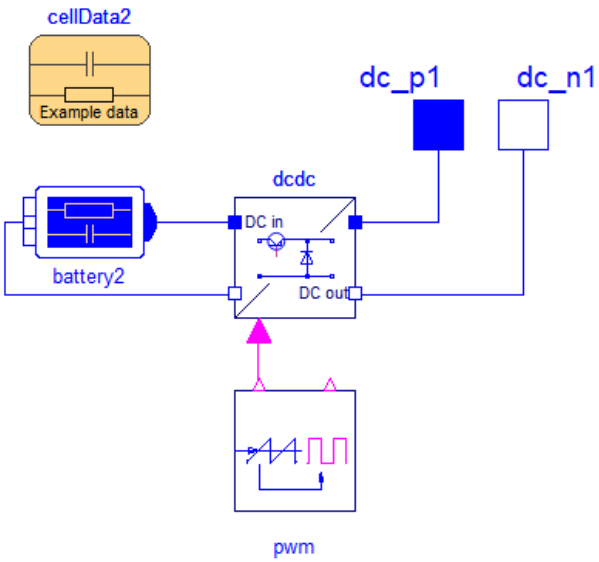

Figure 4. Battery with DCDC converter to interface with drone controller.

In the battery topology that connects the batteries to

braries, a description of the AVL is available online at: `https://tinyurl.com/DS-AVLib` and for the UAVDL at: `https://tinyurl.com/CT-UAVDLib`

[2]See: `https://tinyurl.com/CT-UAVDLibUpgrade`

DOI 10.3384/ecp204551

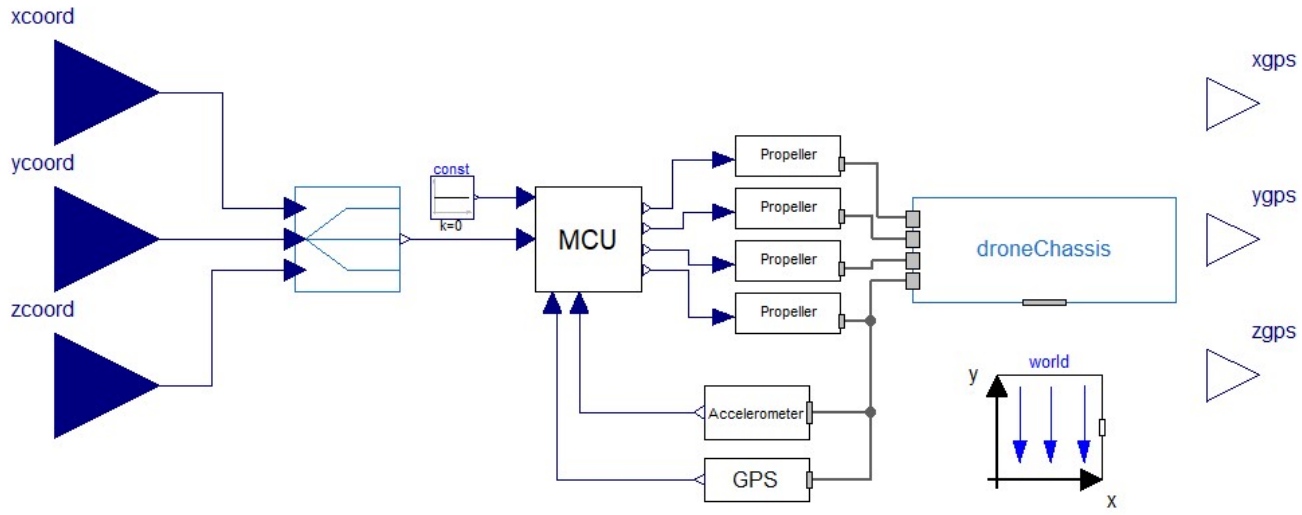

Figure 2. Quadcopter model used in example cases.

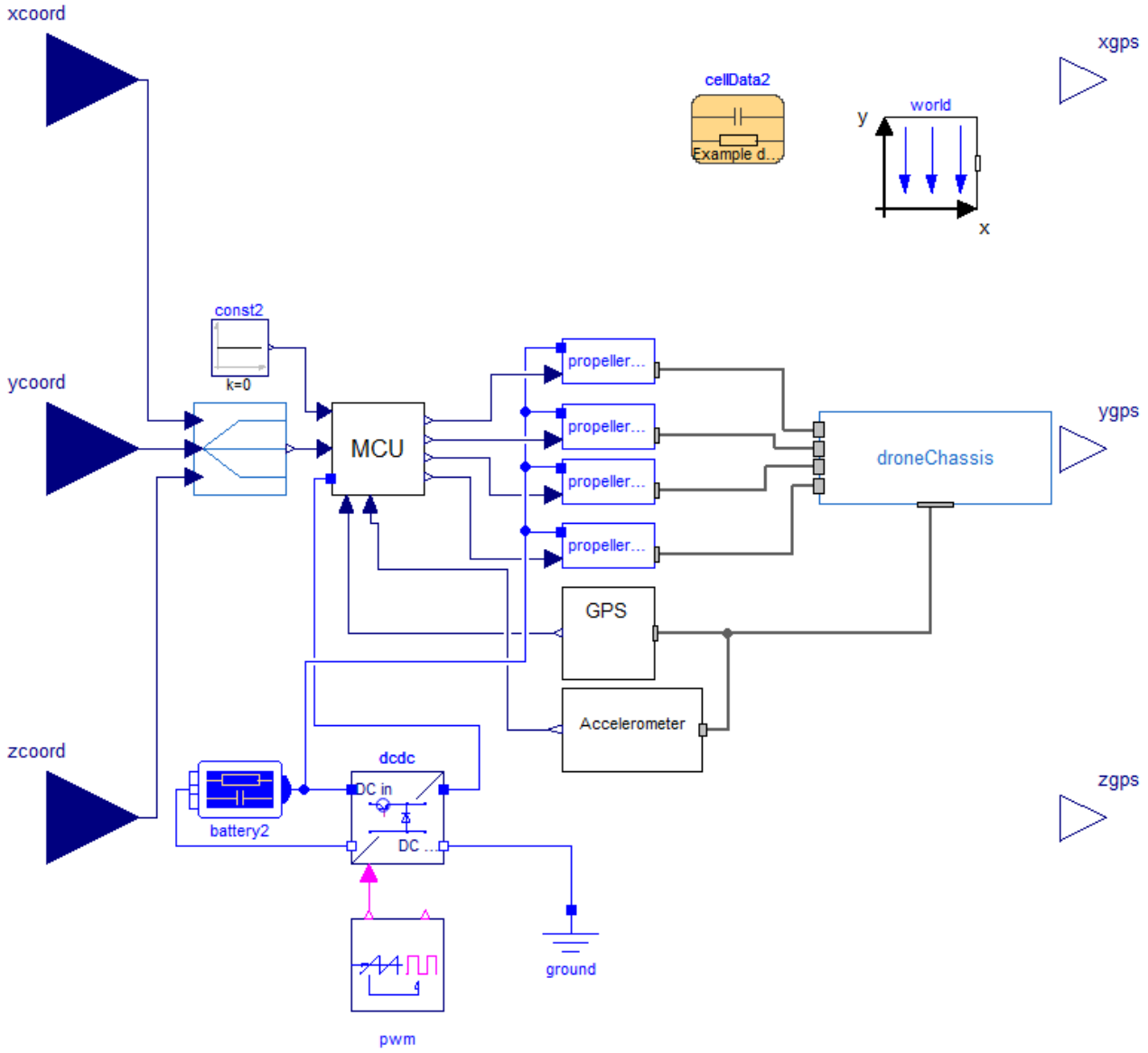

Figure 3. Complete drone model consisting of propellers, motor, controller, and chassis with battery power system. Inputs come from x, y, and z coordinate location commanded by the user.

each of the drivetrains, additional control to scale the voltage applied to the drivetrain is necessary. This is dependent on the SOC of the battery as well as the signal from the MCU. This component, `PowerControl`, is shown in Figure 5.

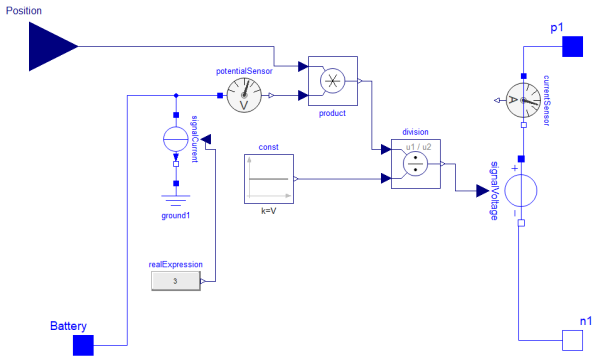

Figure 5. Control component to scale voltage applied to drivetrain based on battery SOC and MCU signal.

4.2 Machine Models

The library contains multiple machine models at varying degrees of modeling fidelity. The simplest representation of the motor is defined as `SimpleDCMotor`, which follows Equations (1)-(3) below. This model only considers torque τ, linear force f, motor speed ω, and current i to control the motor with no electrical, thermal, or mechanical losses considered in the model.

$$\tau = K_\tau \cdot i \tag{1}$$

$$J_p \frac{d\omega}{dt} = \tau - b \cdot \omega \tag{2}$$

$$f = K_f \cdot \omega \tag{3}$$

Most quadcopter systems use brushless DC motors in drivetrains due to their high efficiency and power-to-size ratio. To this end, the library includes an example using the permanent magnet DC machine model `DC_PermanentMagnet` from the Modelica Standard Library in the `Examples` package.

4.3 Control Modules

Each of the drivetrains in the quadcopter relies on a signal from a controller to enable flight. There are multiple representations of the control module in the library considering both the power consumption of the controller and different sampling methods of the controller's PID blocks. There are five representations of the control module:

- Discrete PID
- Discrete PID and electrical load
- Discrete PID using Synchronous Library
- Discrete PID using Synchronous Library and electrical load
- Continuous PID

An example of the control module is shown in Figure 6, which uses an XYZ-coordinate position command, XYZ-coordinate current position GPS measurement, 3-dimensional acceleration measurement, and a one-dimensional yaw command to control the quadcopter's position. These signals are fed into discrete PID blocks to determine the voltage command sent to each drivetrain, denoted by `y`, `y1`, `y2`, and `y3`. For the model variations that include an electrical load, the `pin`, `resistor`, and `ground` components are included in the model to represent a constant power draw.

5 Mechanical Models

The mechanical package includes the following subpackages to model the quadcopter in the mechanical domain:

- Blades
- Propeller
- Chassis
- Rotor
- Motor

The components in the mechanical package use a `replaceable` template structure, where the components use the same base model that is parameterized for different quadcopters and levels of modeling fidelity. Inside of each of these subpackages, there is a package called `Templates` where the template for the component is stored.

5.1 Blade Models

The blade models utilize the Modelica Standard Library's multibody functionalities to link with the rest of the system mechanically. The blade model uses a template defined in `DroneLibrary.Mechanical.Blades.Templates`, which includes the multibody connector `Input` that interfaces a 3-dimensional force and torque vector with the rest of the drivetrain. The blade model consists of two-point body masses representing the individual wings in the model, denoted as the `bodyShape` components in Figure 7. Since the drone models can be animated, the blade model utilizes the `fixedShape` component from the `MultiBody.Visualizers.FixedShape` package from the Modelica Standard Library. The blade model can be configured to be animated using a `.stl` file when the model is simulated, that is stored in the `A_Modelica/DroneLibrary/Resources/Images` folder of the library. The library also includes blade models parameterized for quadcopters such as the Mavic Air and Phantom. This entails defining the blade's mass m and relative distance (defined as an XYZ-coordinate) to the quadcopter body as a point body mass as shown in

DOI
10.3384/ecp204551

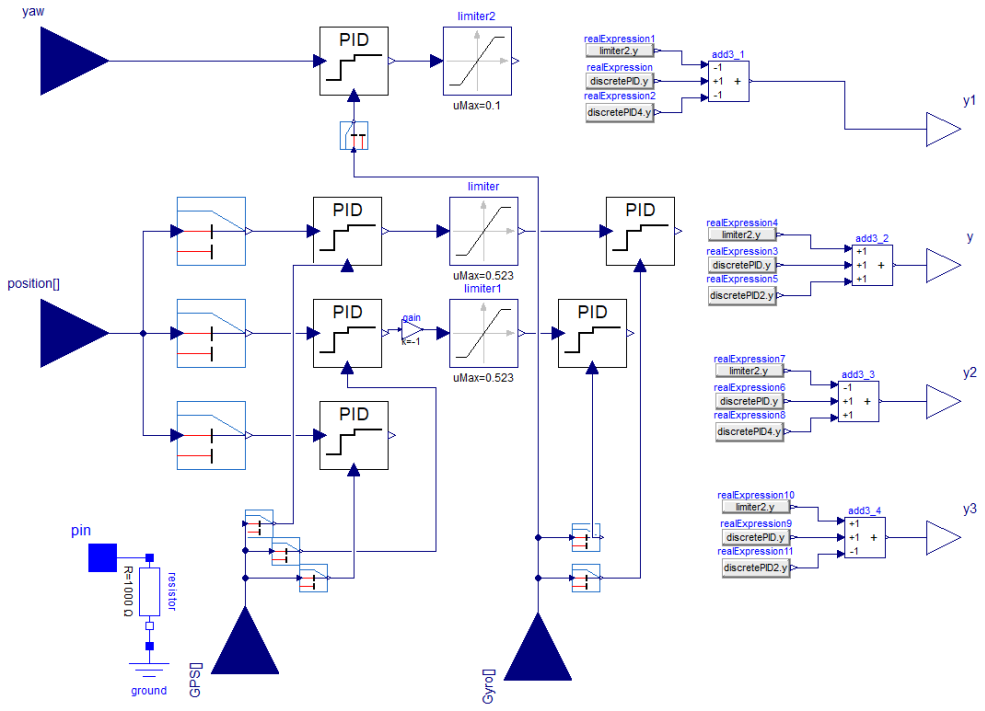

Figure 6. Main controller unit.

Figure 7.

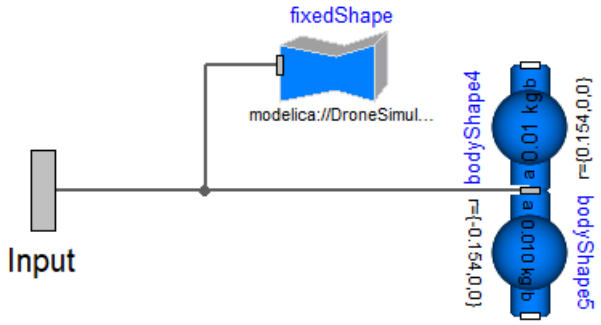

Figure 7. Blade model.

5.2 Propeller Models

The propeller template is shown in Figure 8. The propeller model combines the motor, rotor, and blade models using a replaceable model structure. Each propeller has a `position` input from the controller and has an `Airframe` mechanical connector that communicates a 3-dimensional force and torque value with the quadcopter's chassis.

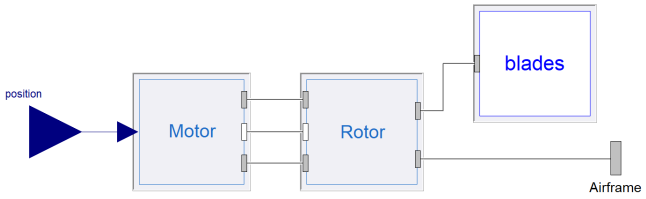

Figure 8. Propeller model.

The propeller model can also be configured to account for a system that has a non-ideal power source, e.g., a battery that changes voltage as it discharges. This is shown in Figure 9, where the electrical pin connector `p1` electrically links the voltage and current drawn by the motor to the quadcopter's power source.

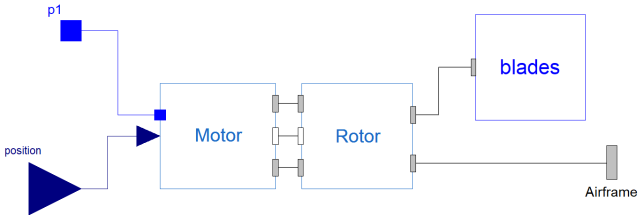

Figure 9. Propeller model with electrical power input.

5.3 Chassis Models

The quadcopter's body is modeled in the chassis subpackage. The template of the chassis model is shown in Figure 10, which connects mechanically to each of the propellers using the `frame_a` connectors. Each arm of the quadcopter's chassis is modeled as a point body mass `bodyShape`, and the centralized body of the quadcopter is represented by the `bodyCylinder` and `pointMass` components. The `frame_a4` interface connects the quadcopter chassis to various sensors that control the drone's positioning, pitch, and yaw. The chassis model is configured to be animated in Figure 11, which contains a `fixedShape` component that links a `.stl` file representing the chassis to be used in the animation of the quadcopter.

The chassis multibody masses follow the mathematical model in Equation (4). Each inertia tensor can be defined individually in this system, which reduces to Equation (5) as defined in the Modelica Standard Library (MSL)'s Multibody Library (Otter, Elmqvist, and Mattsson 2003).

$$\tau = I_{XYZ} * \alpha = \begin{bmatrix} I_{XX} & I_{XY} & I_{XZ} \\ I_{YX} & I_{YY} & I_{YZ} \\ I_{ZX} & I_{ZY} & I_{ZZ} \end{bmatrix} \begin{bmatrix} \dot{\omega}_X \\ \dot{\omega}_Y \\ \dot{\omega}_Z \end{bmatrix} \tag{4}$$

$$\tau = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix} \begin{bmatrix} \dot{\omega}_X \\ \dot{\omega}_Y \\ \dot{\omega}_Z \end{bmatrix} \tag{5}$$

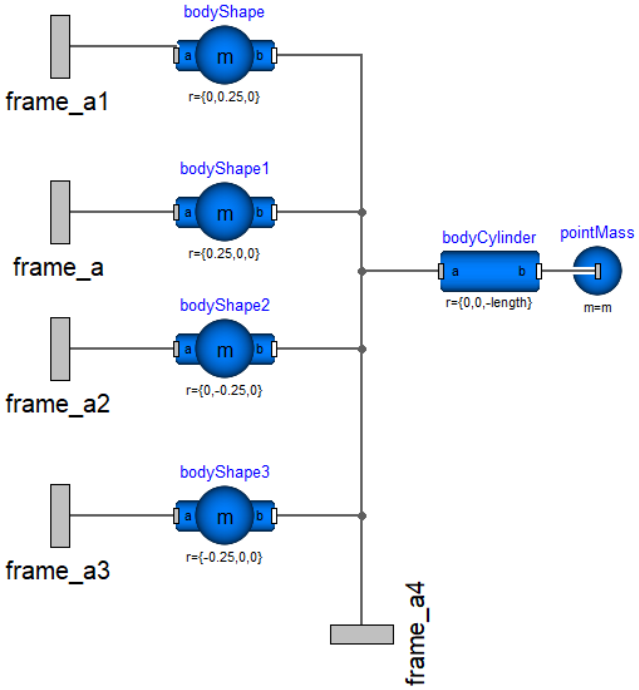

Figure 10. Chassis model template.

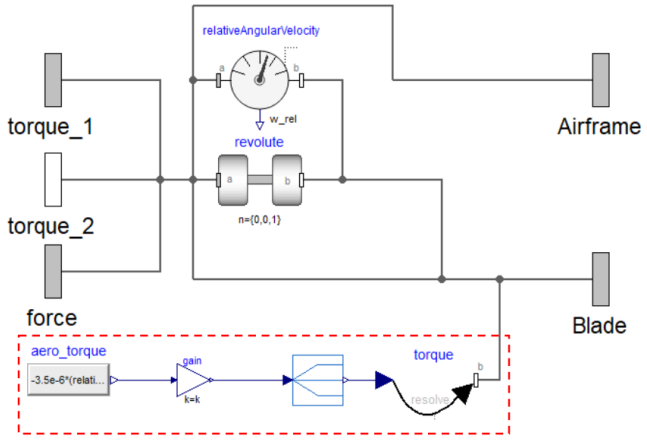

Figure 12. Rotor model. The red box in the lower left corner of the model represents the calculations needed to determine aerodynamic forces applied to the rotor.

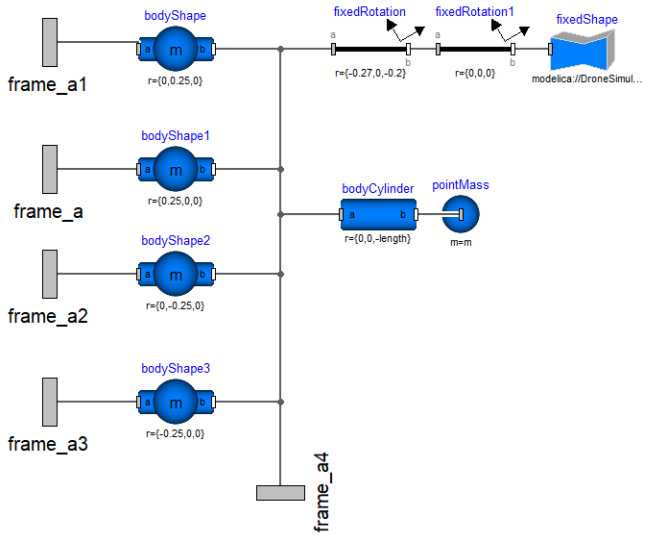

Figure 11. Chassis model with visualization functionalities.

5.4 Rotor Models

The rotor model is represented in Figure 12, which mechanically links to the motor, chassis, and blades. The multibody interfaces `torque_1` and `torque_2` link the torque from the machine to the revolute. The revolute's speed is determined by a scaled measurement of the relative angular velocity between `torque_1` and `torque_2`.

The aerodynamic forces are applied to the rotor using the ω^2 model. Equation (6) calculates the aerodynamic torque, `aero_torque`. More information about the derivation of the simplified model can be found in (Luukkonen 2011).

$$\tau_o = (3.5 \times 10^{-6})\omega^2 \tag{6}$$

5.5 Motor Models

The motor subpackage contains two templates, `DCMotor` and `DCMotor_Power`. These templates can be configured

DOI
10.3384/ecp204551

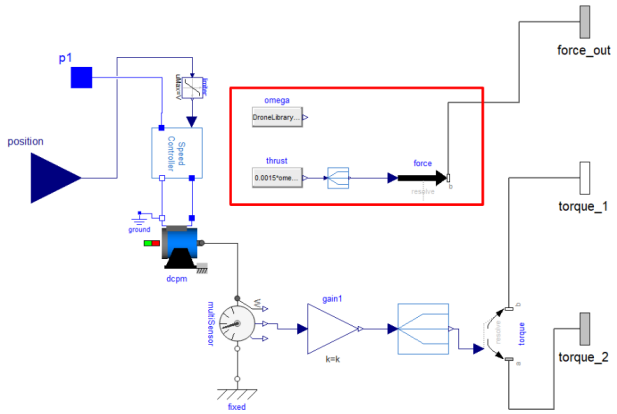

Figure 13. Motor model. The red box contains the components used to calculate the thrust.

with any of the models from the `Electrical.Motors` subpackage. The motor models receive a voltage command from the controller and link the rotor speed and torque mechanically with the rest of the system. In the case of the `DCMotor_Power` models such as the one shown in Figure 13, the electrical pin `p1` can be connected to an external power source such as a battery model. The voltage command is scaled according to the maximum voltage the power source provides using the `Electrical.Sources.PowerControl` component. For the motor models using the `DCMotor` template, the `Electrical.Sources.PowerControl` and `p1` components are removed to represent the electrical system at a lower level of modeling fidelity.

In the motor models, the aerodynamic forces applied to the quadcopter are calculated using the ω^2 model. Equation (7) determines the torque applied from the motor. The thrust provided by the motor is calculated in the component `thrust` in Figure 13. The thrusts are coupled to the motor component using the mechanical multibody connector `thrust`. More information about the derivation of the simplified model can be found in (Luukkonen 2011).

$$\tau_h = 0.0015\omega^2 \tag{7}$$

6 Sensor Models

The sensors package includes models used in the quadcopter to monitor its status and provide measurements for control. This package contains accelerometer models, gyroscope models, and GPS models to track the location of the quadcopter. Figure 14 shows the GPS sensor model, which uses the relative position sensor from the Modelica Standard Library to track the quadcopter's relative position to an origin point. It links mechanically to the quadcopter chassis through interface `frame_a` and communicates a three-dimensional position vector signal to the controller through real output `y[]`.

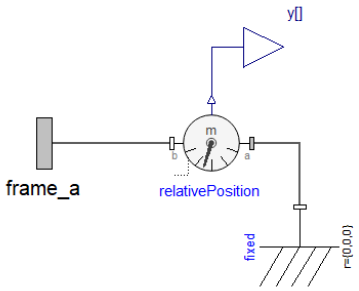

Figure 14. GPS sensor model.

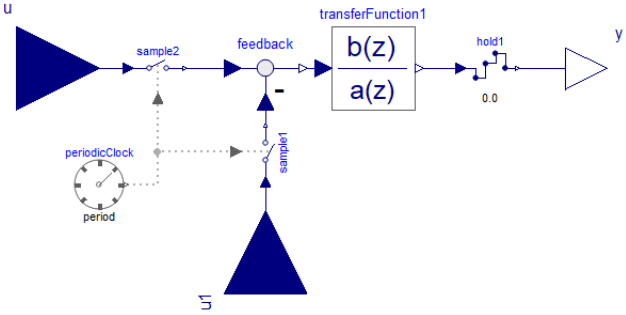

Figure 15. Discrete PID block using Modelica synchronous clocked components.

7 Other Packages

The library also contains various subpackages that provide ancillary functionalities for simulation and analysis.

7.1 Blocks

The `Blocks` subpackage contains models for sources, controls, and signal routing used in the quadcopter components. The `Sources` subpackage contains two models that can be used as XYZ-coordinate inputs to the quadcopter. The `circlePath` component gives a flight command to fly the quadcopter in a circular motion and the `linePath` command applies a ramp path along an axis to fly the quadcopter in a straight line.

In the `Control` subpackage, there are PID transfer function blocks that are used to model the main controller units using both discrete and continuous controls. The discrete PID transfer function component is shown in Figure 15, which uses the clocked components from the Modelica Standard Library to model the PID controls. These models are typically used in virtual reality applications. The purpose of including multiple representations of the PID control blocks is to support a broad range of simulation studies and applications where different integration and simulation methods are used.

7.2 Visualize

The `Visualize` package includes models that allow the user to use visualization functionalities provided both by the MSL and those that depend on the DLR Visualization Library (Hellerer, Bellmann, and Schlegel 2014). It provides component models for the user to

interact in real-time with the model using functionalities of the Modelica Devices Library, enabling keyboard and joystick input. Examples that use these functionalities can be found in `Examples.Visualize`. Before using the examples, see the information included in `Examples.Visualize.HowToRun`.

7.3 Tests

The `Tests` subpackage contains test model examples for individual subsystems in the quadcopter. These models verify the functionality of the discrete control elements, the command path models, and the quadcopter components incrementally. These models can also be useful for helping users through the debugging process when developing new models.

8 Examples

8.1 Basic Example

Several example model can be found within the package `Examples.DroneWithIdealPower`. The package contains a `TestSystem`, as shown enclosed in blue in Figure 16. The goal is to simulate the drone's take-off and hoovering by providing a ramp input to the z-axis reference. The figure also shows how the model can be quickly reconfigured to analyze diverse model variants by using the replaceable models, as illustrated in Figure 16 where the simplest representation of the drivetrain's machine is chosen.

The simplest representation, i.e., when selecting `Drone_IdealMachine` as shown in Figure 16, the drone is modeled with an ideal motor and voltage source power system. This model uses a constant voltage power source that can draw as much current as needed to satisfy any command applied to the motors. This means all electrical dynamics from the battery are also neglected in the model.

The drone is simulated under ideal conditions in Figure 17 with a ramp signal applied in the Z-direction from the ground to $5\,m$. The drone overshoots the $5\,m$ hovering height by 5.43 % in this test. These results can be compared by the user for other cases, e.g., by running the models with the discrete PID or the discrete PID using the synchronous components by selecting the desired model variant as shown in Figure 16.

8.2 Battery Powered Drone Example

In the `Examples.DroneWithoutIdealPower` package, the `TestSystem` model is configured similarly to that described for the previous example. However, here the drone variants/classes can be changed to study the drone's behavior when using a battery power source. This model uses a battery power source where the voltage and current are affected by the battery's state of charge. This means the electrical dynamics from the battery are also included in the model.

This model configuration also uses a DC permanent magnet machine, using the model in Figure 13. The battery is parameterized with a nominal voltage of $12.1V$ and the motor is parameterized with a nominal voltage of $12V$. This model variant considers the non-ideal behaviors that were neglected in the previous section. The system is given the same $5\,m$ ramp command in the Z-direction, with the response shown in Figure 18. Since the simulation is only for a 10-second period, the power consumption will have a negligible impact on the electrical dynamics. The overshoot and steady-state error in the Z-direction are 1.36 % and 0.13 % respectively. The lower error and overshoot compared to the simplified case is due to the higher damping modeled in the motors.

8.3 Example using DLR's `Visualization` Library and the `Modelica_DeviceDrivers` Library

The drone can be simulated using HIL and virtual reality software programs using the DLR Visualization Library. The drone is configured for HIL simulation using the DLR Visualization Library in Figure 19[3].

In the package, `Examples.Visualize` the model `ModuleTest_SimVis` is configured to use computer keyboard arrow inputs to direct the drone's flight path by using functionalities from the `Modelica_DeviceDrivers` library. The components from DLR's Visualization Library include `camera1` and `camera2` components that are connected to the `frame_a` interface on the chassis to follow the drone component around the environment during the simulation. The `world` component defines gravity and a reference frame to all moving components in the environment. This `world` component (from the MSL) is connected to `shape1` (also from from DLR's Visualization Library), which defines the physical terrain (i.e. the scene) that the drone is interacting with.

When the model is compiled and simulated, it appears in the environment in Figure 20. The user can then utilize the keyboard commands to maneuver the drone around the environment. The drone can also be configured to be controlled by a joystick or in other virtual reality environments, where more information can be found in (M. Podlaski 2021).

9 Conclusions

The `DroneLibrary` introduces a basis for open-source, multi-domain drone models at varying levels of detail and complexity. The models included in the `DroneLibrary` are more complex than those in previous literature with the capability to interface components between multiple engineering domains. Many of these systems previously only considered mechanical behaviors and aerodynamics, while the `DroneLibrary` considers electrical dynamics in the motors and power systems. The models in the `DroneLibrary` allow for animation of the system for a

[3]This requires a Professional Edition license of the Visualization Library, to obtain one, see:`https://visualization.ltx.de/`.

DOI 10.3384/ecp204551

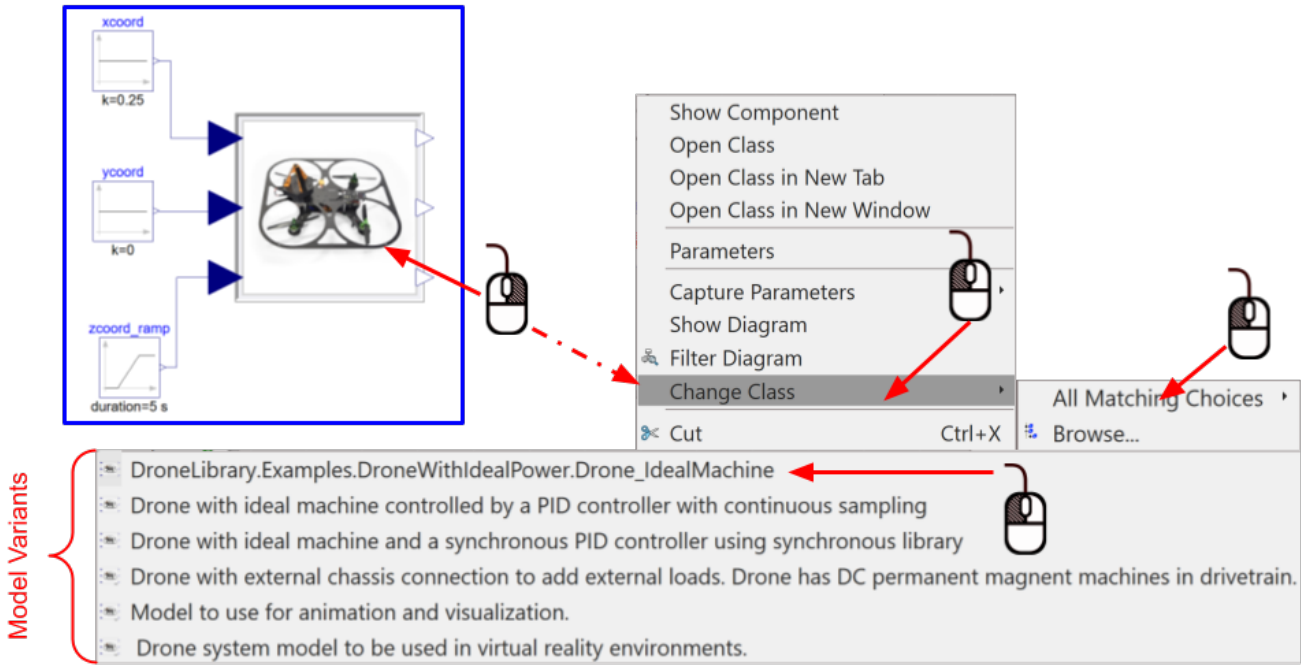

Figure 16. `Examples.DroneWithIdealPower.TestSystem` example model and how to modify it to use the different available variants.

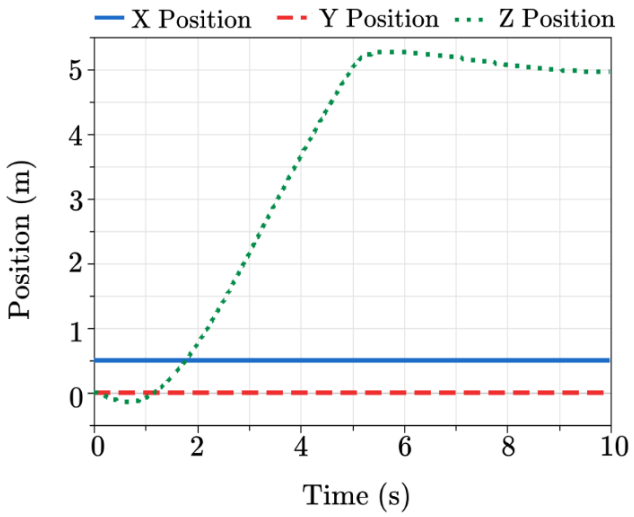

Figure 17. Drone with an ideal motor and ideal power system XYZ position under a ramp signal input.

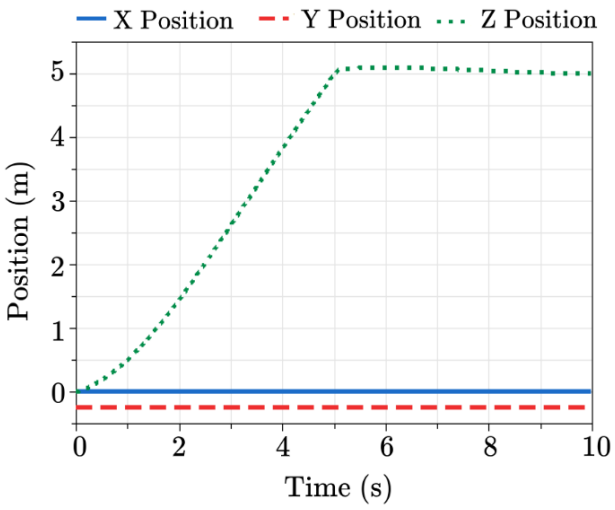

Figure 18. Drone with an DC permanent magnet motor and battery XYZ position under a ramp signal input.

given input, enhancing the insight, analysis, and communication between domain specialists.

The models in the library are designed in a manner that encourages future development, i.e., allowing to both increase the complexity and the flexibility of the component and system models. In addition, these models can be integrated with other software using the FMI Standard (Modelica Association 2014), especially with virtual reality software and game engines such as those in (M. Podlaski 2021).

The library has been developed in Dymola and the developers have successfully checked and simulated the models. In addition, the library was also tested with OpenModelica where currently the models pass checks but may not simulate.

The library has been released as open source software under the BSD 3-Clause License and is available online at: `https://github.com/ALSETLab/Modelica-Drone-3D-FMI`

Acknowledgements

This work was supported in part by the National Aeronautics and Space Administration under award number 80NSSC19M0125 as part of the Center for

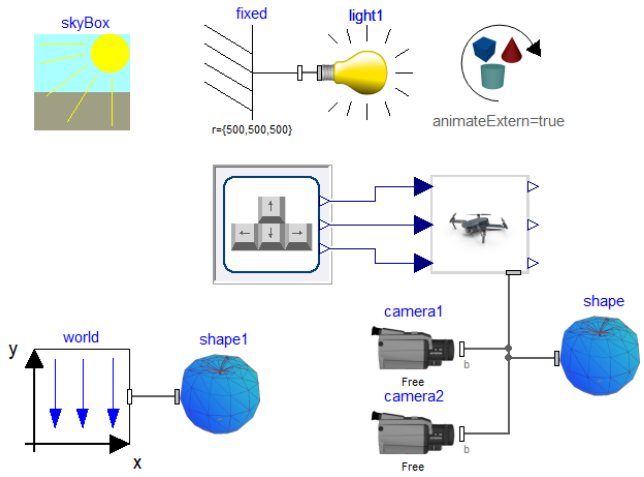

Figure 19. Drone model configured for HIL interaction using the DLR Visualization Library.

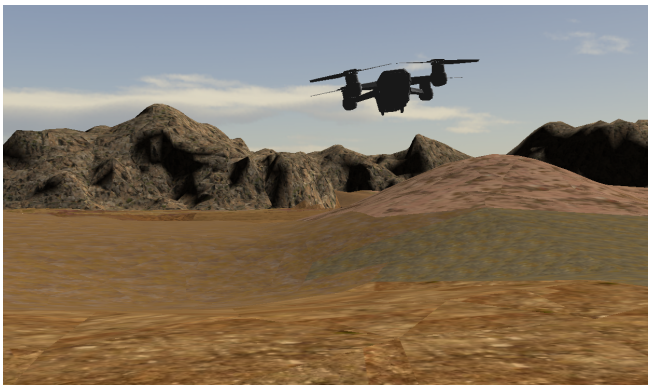

Figure 20. Drone simulated in desert terrain environment using DLR Visualization Library. A video can be found at: `https://youtu.be/1OfGjCZVH5o`.

High-Efficiency Electrical Technologies for Aircraft (CHEETA).

The work of M. Podlaski was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1744655 and the Chateaubriand Fellowship of the Office for Science & Technology of the Embassy of France in the United States.

The work of L. Vanfretti was supported in part by Dominion Energy Virginia.

The authors would like to sincerely thank the following former Rensselaer Polytechnic Institute students for their contributions to the library and this research: Hao Chang, Jake Montenieri, Jordan Grey, Eric Segall and James Lewin.

References

Borer, Nicholas K. et al. (2016). "Design and Performance of the NASA SCEPTOR Distributed Electric Propulsion Flight Demonstrator". In: *16th AIAA Aviation Technology, Integration, and Operations Conference*. DOI: 10.2514/6.2016-3920. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2016-3920. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2016-3920.

Bresciani, Tommaso (2008-10). "Modelling, Identification and Control of a Quadrotor Helicopter". MSc Thesis. Lund, Sweden: Department of Automatic Control, Lund University.

Coïc, Clément, Marc Budinger, and Scott Delbecq (2022). "Multirotor drone sizing and trajectory optimization within Modelon Impact". In: *Proc. American Modelica Conference 2022*. doi: 10.3384/ECP2118656, pp. 56–63.

Doo, Johnny T. et al. (2021). *NASA Electrical Vertical Takeoff and Landing (eVTOL) Aircraft Technology for Public Services - A white Paper. NASA Transformative Vertical Flight Working Group 4 (TVF4)*. White Paper NASA Document Number 20205000636. NASA. URL: https://www.sti.nasa.gov/.

Hellerer, Matthias, Tobias Bellmann, and Florian Schlegel (2014-03). "The DLR Visualization Library - Recent development and applications". In: *Proc. 10th Int. Modelica Conf.* doi: 10.3384/ecp14096899, pp. 899–911.

Kuric, Muhamed, Nedim Osmic, and Adnan Tahirovic (2017-07). "Multirotor Aerial Vehicle modeling in Modelica". In: *Proc. 12th Int. Modelica Conf.* doi: 10.3384/ecp17132373, pp. 373–380.

Luukkonen, Teppo (2011-08). "Modelling and Control of Quadcopter". Independent research project in applied mathematics. Espoo, Finland: School of Science, Aalto University.

M. Podlaski, et al (2020-10). "UAV Dynamic System Modeling and Visualization using Modelica and FMI". In: *VFS 76th Annu. Forum & Tech. Display*. doi: doi:10.4050/F-0076-2020-16289., pp. 1058–1072.

M. Podlaski, et al (2021). "Towards a VR-Based Early Design Interaction for Electric Vertical Take-Off & Landing (eVTOL) Cyber-Physical Models". In: *2021 NAFEMS World Congr.* NAFEMS.

Ma, X., Z. Li, and C. Nae (2016). "Multi-domain modeling and Simulation of a Quad-rotor aircraft based on Modelica". In: *Proc. Intl. Conf. Modeling, Simulation and Visualization Methods (MSV)*, pp. 120–124.

Modelica Association (2014-07). *Functional Mock-up Interface for Model Exchange and Co-Simulation Version 2.0*. Tech. rep. Linköping: Modelica Association. URL: https://fmi-standard.org.

Modelica Association (2023). *Modelica Standard Library v4.0.0 (2020-06-04)*. URL: https://github.com/modelica/ModelicaStandardLibrary/releases/tag/v4.0.0 (visited on 2023-06-08).

Otter, Martin, Hilding Elmqvist, and Sven Erik Mattsson (2003-11). "The New Modelica MultiBody Library". In: *Proc. 3rd Int. Modelica Conf.* Pp. 311–330.

Podlaski, Meaghan, Robert Niemiec, et al. (2021-05). "Multi-Domain Electric Drivetrain Modeling for UAM-Scale eVTOL Aircraft". In: *Proceedings of the 77th Annual Forum*. The Vertical Flight Society. Virtual.

Podlaski, Meaghan, Luigi Vanfretti, et al. (2022). "Extending a Multicopter Analysis Tool using Modelica and FMI for Integrated eVTOL Aerodynamic and Electrical Drivetrain Design". In: *Proc. American Modelica Conference 2022*. doi: 10.3384/ECP2118647, pp. 47–55.

Thiele, Bernhard et al. (2017-05). "Towards a Standard-Conform, Platform-Generic and Feature-Rich Modelica Device Drivers Library". In: *Proc. 12th International Modelica Conference*. doi: 10.3384/ecp17132713, pp. 713–723.