CONCEPTUAL DESIGN OF HYBRID BATTERY MODULES FOR OPTIMISING ELECTRIC AIRCRAFT PERFORMANCE

Gabriel-André Dominic Damian

Bachelor Thesis

Chair: Prof.dr.ir. Braham Ferreira

Supervisor: M. Sc. Alexander Matthee, PhD Student

Submitted in fulfilment of the requirements for the degree of
Electrical Engineering Bachelor

Power Electronics & Electromagnetic Compatibility (PE)
Electrical Engineering, Mathematics and Computer Science Department (EEMCS)
University of Twente
Academic Year 2019/2020

Keywords

A3T, Aerobotic Tech Team Twente, Alice, Battery, EA, Electric Aicraft, Ehang184, Energy Density, EV, Electric Vehicle, Eviation, Fixed Wing, Full Electric, Hybrid Battery, Lithium polymer, Lithium-ion, Load Switch Circuit, Multicopter, Power Density, Specific Energy, Specific Power, UAM, VAU, VTOL,

Abstract

The presented research investigates the usage of a hybrid battery solution for full-electric aircraft as mean of battery weight optimization. The concept is to utilize a high-energy dense battery module for the cruise phase and a high-power dense battery module for the take-off and climb phase, rather than to use a singular battery for the whole flight. It was previously expected that such approach would overall increase the specific energy of the battery without reducing the power performance for take-off, hence reducing the weight of the battery.

The result, however, pointed out that a hybrid solution will in most cases lead to a minor weight increase of the battery, but will also significantly increase the energy density. It was found that a hybrid solution inherently cost additional weight for an-extra power-dense battery module for the take-off and climb phase.

Overall, a hybrid solution is not recommended for long range fixed-wing configuration. But for multicopter and VTOL, a hybrid solution is recommended as it may increase the operational range by 63% and 84% with only 7% and 16% battery weight increase respectively.

Finally, the research has found that the optimal circuit topology for switching between two battery sources is best achieved using N-Chanel power MOSFETs in parallel and in dual configuration. With this approach, a scalable circuit can be developed to handle high-power requirements with minimal energy losses.

Table of Contents

Keywords....i
Abstract....ii
Table of Contents....iii
List of Figures....v
List of Tables....vi
List of Nomenclature....vii
Statement of Original Authorship....viii
Acknowledgements....ix
Chapter 1: Introduction....1
1.1 Background....1
1.2 Context....2
1.3 Purposes....2
1.4 Significance, Scope and Definitions....2
1.5 Thesis Outline....4
Chapter 2: Technology Landscape Review....5
2.1 Electric Vehicles....5
2.2 Energy Storage and batteries....8
2.3 Load Switching Circuit....11
2.4 Summary and Implications....12
Chapter 3: Design and Simulation....13
3.1 Methodology and research design....13
3.2 Software and Instruments....16
3.3 Procedure and Timeline....17
3.4 Analysis....18
3.5 The application....18
Chapter 4: Results & Validation....25
4.1 Alice....25
4.2 Ehang 184....26
4.3 VAU....28
Chapter 5: Parametric Analysis....29
5.1 The results....29
5.2 Aicraft Performance review....31
5.3 Load Switching Circuit....32
Chapter 6: Conclusions and Future Work....33

Bibliography **35**
Appendices **39**
Appendix A 39
Appendix B 40
Appendix C 43
Appendix D 44
Appendix E 45

List of Figures

Figure 2-1: Hybrid battery powertrain [30] [31] 8
Figure 2-2: Ragone chart for various energy storage types [37] 10
Figure2-3: Switch on the left, Relay on the right (symbols taken from Sparkfun) 11
Figure 2-4: From left to right: Thyristors, BJT, JFET, MOSFET, IGBT symbol (symbols taken from Sparkfun and EasyEDA) 11
Figure 2-5: Dual MOSFETs configuration to prevent reverse current [40] 12
Figure 3-1: Alice aircraft from Eviation [41] 14
Figure 3-2: EHANG 184 [26] 14
Figure 3-3: VAU, designed by members from A3T (courtesy of A3T) 15
Figure 3-4: Energy & Power Profile on the left and Battery Profile on the right 19
Figure 3-5: Simplified Flight Profile of a Fixed Wing 23
Figure: 3-6: Simplified Flight Profile of a Multicopter on the left and VTOL on the right 24
Figure 4-1: Range performance relative to battery weight 25
Figure 4-2: Results of the parametric analysis for the fix wing configuration 25
Figure 4-3: Range performance relative to battery weight for the multicopter 26
Figure 4-4: Results of the parametric analysis for the multicopter configuration 26
Figure 4-5: Battery weight reduction achieved at different cruise speed for Ehang184 27
Figure 4-6: Battery weight reduction achieved at different range speed for Ehang184 27
Figure 4-7: Results of the parametric analysis for the VTOL configuration 28
Figure 4-8: Range performance relative to battery weight for the VTOL configuration 28

List of Tables

[The List of Tables can be created automatically and updated with the F9 key – refer to *Thesis PAM*.]

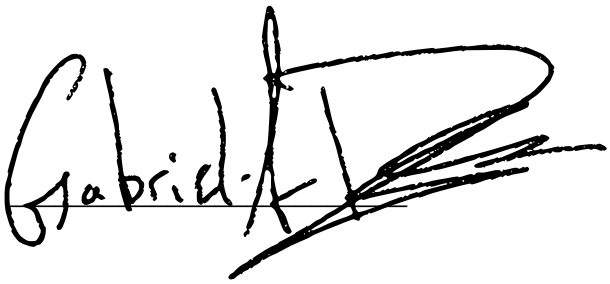
Table 2-1: Recent EA designs, [22] [23] [24] [25] [26] [27] [28] [29] 7
Table 3-1: Variables studied 16
Table 4-1: Weight and capacity comparison 25
Table 4-2: Weight and capacity comparison 28

List of Nomenclature

ASSB:	All Solid-State Battery
CDCL:	Climb Drag Coefficient
CDCR:	Cruise Drag Coefficient
CDDC:	Descend Drag Coefficient
CDTO:	Take-Off Drag Coefficient
EA:	Electric Aircraft
EV:	Electric Vehicle
ICE:	Internal Combustion Engine
TOW:	Take-Off Weight
UAM:	Urban Air Mobility
VTOL:	Vertical Take-Off and Landing

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature: 

Date: 04/02/20

Acknowledgements

Over the course of 12 weeks, a considerable amount of time and effort went into the realisation of this research. However, it would not have been possible without the kind (and patient support) of few individuals.

First, I would like to thank Akarsh for sharing his impressive pool of knowledge and understanding in the field of aeronautic. Secondly, for Alex to guide me through this journey and providing me with precious feedback. Thirdly, for Braham for accepting my first research proposal and supporting the pursuit of my passion.

And of course, a special thank you for my mother who supported me throughout my engineering studies, from mechanical to electrical and from Canada to the Netherlands.

Chapter 1: Introduction

To facilitate the development of full-electric aircraft (EA) for a greener future, the following research investigates how an EA's battery system can be further optimised using current technology. The focus of the research is to evaluate the conceptual design of a hybrid battery system solution consisting of two modules. The first module with high-energy density, optimised for flight endurance, and the second module with high-power density optimised for power requirements during take-off and climb. If proven successful, such a solution may result in overall lower aircraft battery weight, and thus could increase the EA's performance. Furthermore, the study will investigate scalable switching circuit approaches that could be used in the realization of such hybrid battery modules to ensure minimal energy loss.

The first section of this chapter covers the basis that led to this research (section 1.1), its context (section 1.2) and its purposes (section 1.3). The significance and scope are then outlined (section 1.4) and the main terms are defined. Finally, a general overview of the following chapters is presented (section 1.5).

1.1 BACKGROUND

"Humanity is now standing at a crossroads. We must now decide which *path* we want to take" (Greta Thunberg, 2019). As of 2020, it is now well agreed by the scientific community [1] and general public that human is responsible for causing global warming and that climate action is necessary. This realisation is now fuelling the *path* to the electrification of the future and the energy revolution essential to curb carbon dioxide emissions.

Almost all carbon dioxide emissions originate from the combustion of fossil fuel [2], therefore it is only logical that reducing the use or finding an alternative to fossil fuel is an effective way to lower emissions. One way is to electrify means of transportation, which is responsible for about 20% of global emissions [3], for which 2% originates from the aviation sector [4]. Although the impact of aviation is relatively small for now, this number is expected to considerably grow as aviation intensifies and other emission sources such as energy production make rapid progress to decarbonise [5]. Moreover, one-third of the operating costs of airlines is spent on fuel [6]. With the

continuous rise in fuel prices, the financial incentive is pressuring the industry to focus on fuel efficiency and find alternatives such as full electric flying, as it could cut the direct operating cost since electricity is cheaper than jet fuel.

1.2 CONTEXT

While battery technology for land-based vehicles is now adequate to compete against their ICE counterpart, there is still a long way to go before electric aircraft can do the same. This difficulty mainly arises from the weight attributed to power the vehicles. In comparison, lithium-ion cells approximately hold 2% of the energy density of jet fuel. Even after considering the loss in efficiency due to the thermodynamic cycle, jet fuel yield about 14 times more energy than a lithium-ion battery at equivalent weight [7]. Moreover, unlike cars, weight is a primary concern for performances [8], and thus the electrification of aircraft is not as straight forward as it would be for a car.

Before electric aircraft become viably and financially competitive to conventional aircraft, further development is required to optimise the performance of batteries. For this reason, the presented research will evaluate performance optimisation of batteries based on combining two separate battery modules, each optimised for different phases of flight.

1.3 PURPOSES

The central objective of this thesis is to evaluate if a hybrid battery solution can reduce the overall weight of an aircraft's battery, and if so, to determine to which extent. The secondary objective is to offer a possible electric circuit design to realize such a concept.

Through the thesis, readers will acquire the basic knowledge and foundation to further research on fully electric hybrid battery solutions.

1.4 SIGNIFICANCE, SCOPE AND DEFINITIONS

Hybrid battery solutions for electric aircraft has been mentioned multiple times in previous studies but has not been directly studied nor as the practical implications been evaluated. This research aims to cover this gap in the literature and further expand the foundation for the development of future full electric aircraft.

The scope of the research limits itself to the case studies of three aircraft configurations and fundamental electronics. For all models, the analysis will be based on simplified aerodynamics models and conducted through a dedicated MATLAB application. The proposed switching circuit configuration will also be kept simplified by using the minimum of components required to function.

Since definitions of terms may vary from research to research, it is best to clarify the meaning of the main recurring terms used in this study:

- **Fixed-wing** refers to aircraft using wings and forward propulsion as their only mean of sustaining flight. They are mostly known as airplanes. This type of configuration generally offers the best range and flight endurance.
- **Multicopter** refers to aircraft using upward propulsions as their only mean of sustaining flight. This type of configuration generally offers the least range and endurance but can manoeuvre in all directions and take off vertically.
- **VTOL**, or Vertical Take-Off and Landing, refers to aircraft using upward propulsion as mean to take off, and wings with forward propulsion to sustaining flight at cruise. They form a hybrid between fixed-wings and multicopters, resulting in both high manoeuvrability, long-range and endurance.

Moreover, the two main factors used in the study to qualify a battery's performance are the following:

- **Energy density** (or specific energy) refers to the energy-to-weight ratio, typically in Watt-hour (Wh) per kilogram (kg). This can be compared to the EA flight range relative to its battery weight.
- **Power density** (or specific power) refers to the power-to-weight ratio, typically in Watt (W) per kilogram (kg). This describe the available maximum power that the EA can draw per battery unit weight.

For abbreviations, refers to the list of abbreviations.

1.5 THESIS OUTLINE

The presented research follows a standard thesis outline, with a total of six chapters.

- **Chapter 1** introduces the motives behind the thesis and its content.
- **Chapter 2** reviews the current technology landscape related to the study.
- **Chapter 3** outlines the design and methodology of the research.
- **Chapter 4** provides the results of the study without interpretations.
- **Chapter 5** discusses the parametric impact of variables issued from the results.
- **Chapter 6** concludes the research and points to further possible studies.

Further information, additional references, and results can be found in the appendixes.

Chapter 2: Technology Landscape Review

There is over a century of research and development that has led to the current state of the art aeronautic technology required to propel modern airplanes in the sky [9]. Since electric aircraft relies on this already well establish bases, the scope of this research will not accommodate for complex aerodynamic studies. Rather, the focus of the research is brought on the emerging electric advancement that has led to the resurgence of electric vehicles.

The first section (2.1) will evaluate the broad nature of electric vehicles, from the ground to the sky. The second section (2.2) will investigate the facets of energy storage, with a closer look to batteries and its limitation. The third section (2.3) will introduce different types of load switching circuit techniques. Finally, the last section (2.4) will summarise the technology landscape review and offer its further implications for the remaining of the study.

2.1 ELECTRIC VEHICLES

Electric vehicles (EV) are vehicles that use electric motors as a mean of propulsion. The electricity required for the motors can come from various sources, such as batteries, electric generator, solar panels and hydrogen fuel cells among others.

Their debuts dates to the mid-19th century, when electricity used to provide superior ease of operation to power vehicles compared to gasoline engines of the time. In comparison, around one third of all vehicles on the road in the U.S. were electric in the early 20th century. Electric vehicles then saw a decline in 1920 as soon as gasoline became cheaper and readily accessible with the venue of gasoline stations [9].

2.1.1 Land Vehicle

Currently, electric vehicles are gaining in popularity as the focus on renewable energy has pushed new advancement in battery technology and economics are more favourable. However, it wasn't always the case.

Due to the traditionally high cost of batteries, full-electric cars were relatively expensive and were out of reach for most of the population. Though, the concept of electric cars became more known in the late 1990s with the first mass-produced hybrid

vehicle, the Toyota Prius [10]. Back then, hybrid vehicles worked primarily on gas engines and only used autogenerating to recharge a small battery pack. The pack was then used for quick acceleration of the car to considerably reduce the fuel consumption in urban stop-start conditions. Regenerative breaking also appeared with the mass production of the Prius. It was only in 2008 that the regenerative braking technology was used in other hybrids as Ford licenced the technology from Toyota [11]. Finally, it's only later that hybrid used a larger battery pack that could be recharged at home [9].

Since batteries do not have an unlimited serviceable life, various researches have come up with solutions to prolong the EV's battery life. For example, it is recommended to not charge the car to full capacity, nor to fully discharge it [12] and to slowly charge and discharge the battery to prolong its life [13] .

However, these solutions may not always be adequate for EA. Not using the full capacity of the aircraft's battery is non-optimal and slow charging cost time. Consequently, electric car improvements in term of performance cannot always be translated for EA.

2.1.2 Aerial Vehicle

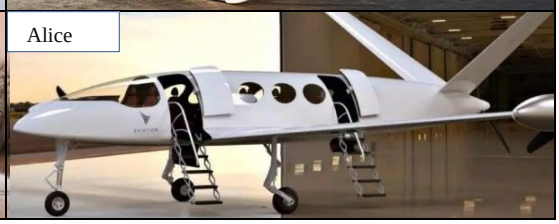
Surprisingly, electric aircrafts date back as early as the 1880s, when the Tissandier brothers first flew an electrically powered airship [14]. Though, it is only in the 1970s that EA returned when Robert Boucher was commissioned by the Defence Advanced Research Project Agency (DARPA) to develop multiple experimental aircraft [15]. This led to the development of a solar-power aircraft named Sunrise II [16]. EA later gained in popularity after the famous flights of Helios in the early 2000s [15]. Helios was a flexible flying wing power by solar panels and hydrogen fuel cells [17].

These advancements soon pushed for the creation of a series of experimental aircraft with various energy storage. For example, Boeing announced their concept SUGAR Volt, a hybrid hydrogen-powered aircraft initially brainstormed in 2006 [18]. Airbus followed with a different approach announcing their aim to reduce by half their carbon emission by 2050, and to develop a fully electric regional aircraft by 2030 [19]. To start the process, they have built and successfully tested the E-Fan 1. However, the first successful commercial electric flight was achieved by a much smaller company,

named Harbour Air, who tested their EA in December 2019 [20]. Thus, proving that batteries may be the best solution for storing energy. Since Helios, other solar-powered aircraft has been developed and tested, but with limited practical success. Impulse 2 has completed an impressive round of the world tour fully powered by the sun [21]. But due to its size, limited payload and speed, solar-powered EA will require further development to achieve mass adoption.

Beside these experimental aircraft, a new wave of EA is taking shape with the development of Urban Air Mobility (UAM). The concept is simple, make cities transport airborne, with a network of air taxi to reduce congestion and pollution, and to follow the conversion from fuel to electric. Many companies are in the race to make UAM a reality with some concepts in advanced development stages. These concepts are mostly VTOL or multicopter configurations for urban operations, and fixed-wing for regional flights. Although these designs all have their unique shapes, they all share a key feature. They are all powered with batteries.

Table 2-1: Recent EA designs, [22] [23] [24] [25] [26] [27] [28] [29]

Boeing	Hyundai
Lilium	Vahana
Ehang 184	Volocopter
Harbour Air	Alice 

Numerous studies [30] [31] [32] have already been conducted to research optimal means of powering EA. The common key elements of these studies are the energy and power requirement of EA, often pointing out that current battery technology is not yet adequate to meet these requirements [30]. The main issue arises from the short take-off phase, which requires significant power in comparison to cruise. Therefore, a single battery configuration aircraft design makes a compromise between battery optimisation to account for the take-off and cruise. This has led to the proposed hybrid battery system to both reach energy and power requirements with two separate batteries instead of under optimised a singular battery.

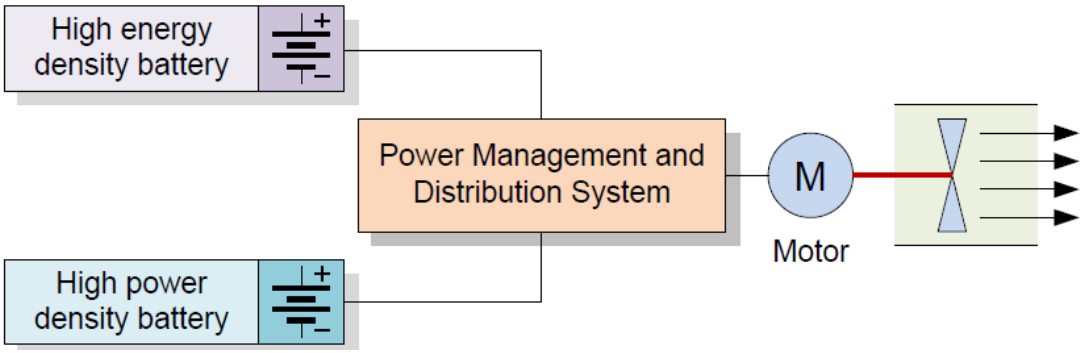


Figure 2-1: Hybrid battery powertrain [30] [31]

Although these studies [30] [31] [32] look intensively into the conceptual design of EA, ranging from full-electric, hybrid-electric and hybrid, they do not offer specific insight into battery's weight reduction and means of realizing a hybrid-electric system. To cover this gap, the presented research will look specifically into hybrid-electric system's performance in terms of battery weight reduction and propose a circuit design.

2.2 ENERGY STORAGE AND BATTERIES

The birth of battery technology can be traced back to as early as 1800 when Volta invented the voltaic pile [33]. The battery was then made of a pile of copper and zinc discs, separated by a layer of cloth soaked in brine. Since then, batteries have been drastically improved, using a variety of materials and chemicals to satisfy the general industries demand. However, current batteries are still below their theoretical potential [34], and other energy storage devices like fuel-cells or supercapacitors have their

limitation. Further advancements have yet to be before the aviation's industry demand is satisfied.

2.2.1 Technology Review

Energy storage comes in all forms and shapes, storing energy either in solid, liquid or gas solutions. When it comes to comparing various energy storage, the two main factor remains the same. Specific energy, which is the energy to weight ratio, and the specific power, which is the power to weight ratio. In simple aeronautical term, the specific energy links to the flight endurance of the vehicle, and the specific power links to the acceleration of the vehicle.

Some of the most used batteries across industries are lithium-ion (Li-Ion) and lithium-ion polymer (Li-Po) [35]. Li-Ion uses a liquid electrolyte solution, while Li-Po uses solid electrolyte. Li-Ions are widely used as batteries for electric cars, power tools and powerbanks since they are cheaper to manufacture, offer a higher energy density and longer lifetime. On the other hand, Li-Pos are widely used to power UAVs and cell phones for examples as they have a much higher specific power and can be more easily shaped.

Another common mean of energy storage is done with supercapacitors. A very high Farad rates capacitor allows short-term high-power charge and discharge, making them ideal for short-term energy storage or burst power delivery. For example, they are commonly used in automobiles and buses for regenerative braking [11]. However, even the latest supercapacitor advancements barely reach 10Wh/kg [36] compared to 200Wh/kg of a regular Li-Ion battery, thus greatly limiting their usability for energy-intensive applications.

On the other end, hydrogen fuel-cells are known to have high specific energy, with up to 10 times greater than Li-Ion (Figure 2-2: Ragone chart for various energy storage types . Hydrogen gas being the lightest element of all makes it attractive as a mean to store energy for weight-restricted applications such as EA. But unfortunately, storing hydrogen has been proved to be challenging, and the power performance of fuel-cells are much lower than Li-Ion batteries.

Other possibilities include methods of collecting power in flight, such as using solar panels, tethered to ground station or even by beaming power with a laser, but these solutions are impractical. Rather, the solution to power full-electric aircraft relies

on finding a compromise between specific energy and specific power or to use multiple storages.

2.2.2 Upcoming advancement

One of the most anticipated upcoming battery is the lithium-air (Li-air), which has a theoretical energy storage of 11kW/kg [34], compared to 200W/kg for a Li-ion. Such battery could outperform gasoline in specific energy, but the technology is far from being ready.

On a shorter-term, another awaited development in battery technology is the all-solid-state battery (ASSB). They are projected to hold as much as twice the capacity of a regular Li-ion cell, but they are expected to be more costly and have limited power performance due to the higher internal resistance. Nevertheless, Toyota has announced its production and demonstration at the 2020 Olympics in Tokyo, proving that the technology seems to be soon available [37]. If solid-state keeps its promise of doubling energy density of batteries, EV could greatly beneficiate from it.

2.2.3 Limitation

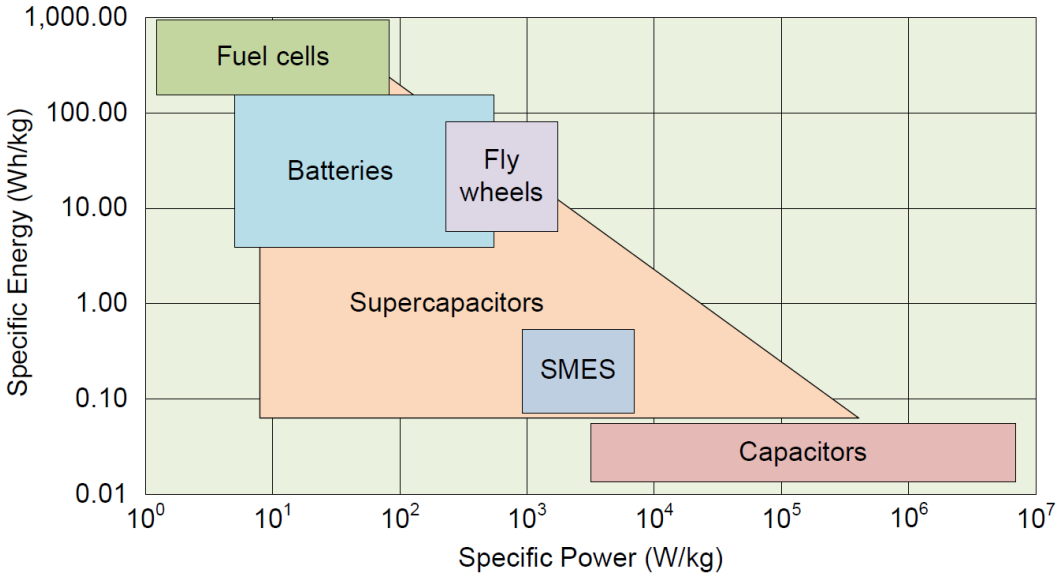


Figure 2-2: Ragone chart for various energy storage types [37]

As mentioned earlier, the selection of a battery system for any EV inevitably includes a trade-off between high specific energy and high specific power, which can be visualized with the Ragone chart above.

2.3 LOAD SWITCHING CIRCUIT

If two batteries with different voltages are placed in parallel, the battery with the lower voltage will receive current from the higher voltage, which can lead to the battery malfunctioning, overheating or even exploding. Therefore, switching between two power sources, or batteries, requires a circuit to prevent the batteries of affecting one an another.

2.3.1 Mechanical

The most straightforward way of realising the circuit would be with a manual, mechanical switch where an operator could select between two different battery sources, but this would be impracticable. One step further could lead to the use of relays, where a mechanical switch is activated digitally. However, relays do not scale well as they are relatively heavy, costly and can carry a limited amount of current. This would not be an ideal solution for large scale EA application.

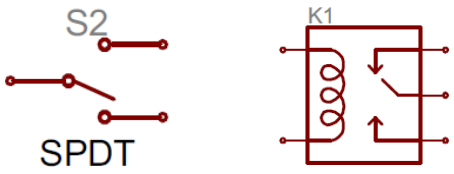


Figure2-3: Switch on the left, Relay on the right (symbols taken from Sparkfun)

2.3.2 Digital

Looking at a digital approach, there are two choices currently used in a wide range of applications to serve as a load switch. First are the thyristors, but they are not easily scalable, and they are costly. The second choice, which is the most used, are transistors, such as BJTs, JFET, MOSFETs and IGBTs. When it comes to power applications, however, MOSFET and IGBTS are the usual solutions as they are relatively cheap, scalable and can share large current [39]. Though, N-Channel MOSFET traditionally offer the lowest resistance of all option, therefore meaning it is the most efficient solution for high power application.

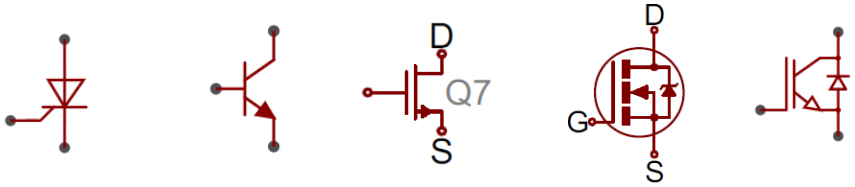


Figure 2-4: From left to right: Thyristors, BJT, JFET, MOSFET, IGBT symbol (symbols taken from Sparkfun and EasyEDA)

2.3.3 Solving the body diode problem

A MOSFET comes with an internal body diode, which in load switching applying causes an issue. Because of the reverse body diode, current can flow back to the power source, which is as previously discussed, undesirable. The simple solution to prevent the body diode problem is to use a dual MOSFET configuration. With this approach, the counter facing diodes prevent any undesired current flow, but to the cost of doubling the power loss, component count and cost.

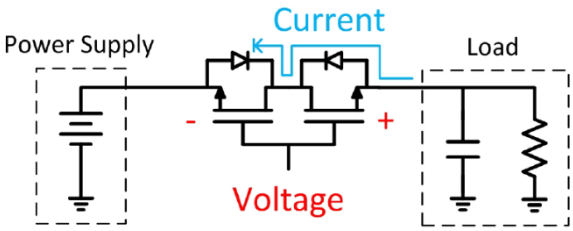


Figure 2-5: Dual MOSFETs configuration to prevent reverse current [40]

2.4 SUMMARY AND IMPLICATIONS

On December 10, 2019, the first fully electric commercial airplane completed a successful test flight, marking the beginning of a new electric aviation era [20]. Taken in consideration with other models such as Alice, EHANG 184 and Hyundai's Smart Mobility Solution, the UAM movement seems to be right around the corner. However, battery technology is not yet ready to see electric air taxis take over the sky and let alone large-scale commercial airplanes. Further improvements in energy storage must be achieved before the electric aircraft revolution can become a practical reality.

Besides waiting for battery technology to improve, previous studies [30] [31] [32] has already pointed out to means of improving current performance by utilizing and hybrid battery systems. Such a proposed system would be composed of a small, power-dense module for take-off phase, and a larger energy-dense module for the cruise phase. However, these studies have not investigated this solution in depth. For this reason, the proposed thesis will conduct a more direct approach to validate this solution based on measurable results.

The final parametric analysis will examine how various variables influence the answer to the research question and determine to which extent such solution can be beneficial, or not. Through the methodology of the conceptual design of a hybrid battery module, this thesis can contribute to the development of future electric aircraft.

Chapter 3: Design and Simulation

To best answer the research question, a step by step approach is undertaken to start from a 3D model of an aircraft and to end up with a conceptualized battery module. This approach can then be automated in a loop to gather data points for later analysis to corroborate to the research question.

This step by step approach will primarily consist of two tools. First, OpenVSP is used to evaluate the aerodynamic aspects of the aircraft, and secondly, a MATLAB application will be created to use the previously found aspects and turn it into a conceptualized battery module.

This chapter first discusses the methodology and design of the research. Secondly, the tools used are described and justified. Then, the procedure and timeline of the research are presented. Section 3.4 discusses the parametric analysis approach, and section 3.5 discusses the MATLLAB application in more detail as well as its simplifications and limitations.

3.1 METHODOLOGY AND RESEARCH DESIGN

3.1.1 Methodology

There exist a broad range of different aircraft configurations and user cases, thus making the conceptual design of a single powertrain impractical. Instead, this research will conduct three case studies of three main categories and sizes of aircraft to best cover the broad range of powertrain design requirements. Each case study will investigate if a hybrid battery system is tangible for the specific configuration.

The first evaluated configuration will be a fixed-wing aircraft. They conventionally are the most efficient for long-range and at carrying heavy payloads. The case study for this configuration will be on the design of Alice from Eviation, a promising regional passenger airplane.

Figure 3-1: Alice aircraft from Eviation [41]

The second evaluated configuration will be a multirotor aircraft. Their modern design, derived from consumer UAVs (drones), gives them the agility required to navigate in dense urban areas. This configuration has been often used to represent the air taxi trend. The case study for this configuration will be on the design of EHANG184 from Ehang, a single-seater autonomous air taxi.

Figure 3-2: EHANG 184 [26]

The third and last evaluated configuration will be a VTOL aircraft. They combine the efficiency of fixed-wing with the agility of multicopter. The case study for this configuration will be on the design of a humanitarian rescue UAV from A3T, the student drone team from the University of Twente.

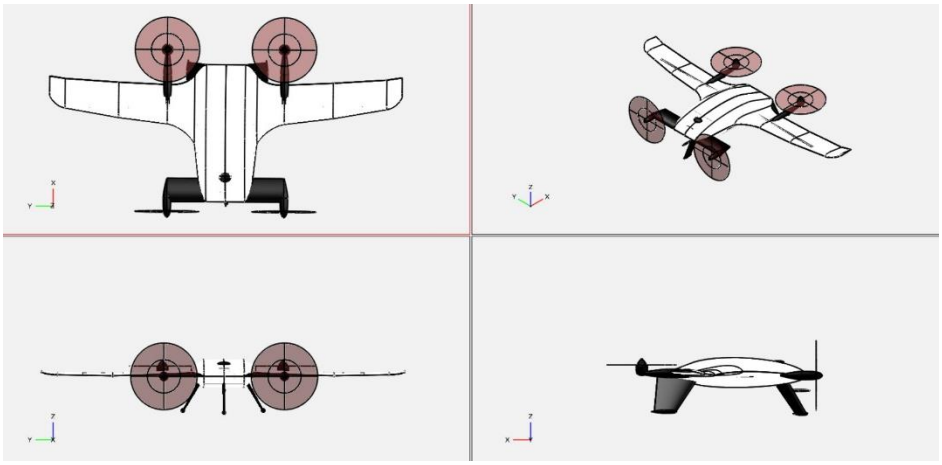


Figure 3-3: VAU, designed by members from A3T (courtesy of A3T)

The step by step approach is relatively straightforward. The goal is to find the drag coefficient of the aircraft relative to different phases of flight, input essential parameters in the MATLAB application and it outputs results. These steps are:

1. Model the aircraft in OpenVSP.
2. Use the parasite drag tool to find C_{D0} at for each flight phase.
3. Use VSPAERO to find the lift coefficient at each segment.
4. Calculate the average to find the total lift coefficient.
5. Calculate the Oswald factor.
6. Calculate the drag coefficient.
7. Repeat steps 1 to 7 for each flight phase.
8. Input the drag coefficient and the required parameters in the application.

With this approach, the application will be able to determine how much power and energy is required to complete the mission profile of the aircraft and conceptualise a battery module. Using the application, the process can be automated to obtain various data point from variables and use it to complete the analysis.

3.1.2 Research Design

There are countless variables that come into play when studying aircraft performances. However, this study will focus solely on the principal factors influencing power and energy consumption to avoid going out of scope. The first three variables are relative to the design phase of the aircraft, and the five remaining are relative to the mission profile. The list of variables used in this study is presented below:

Table 3-1: Variables studied

Variable	Classification	Expected impact
Drag Coefficients	Dependent on aircraft's design	Important impact as it is directly linked to the force drag formula
Surface area	Dependent on aircraft's design	Important impact as it is directly linked to the force drag formula
Motor Efficiency	Dependent on the motor's design, cruise speed & altitude	Important impact as it is converting power to force
Speed	Independent	Most important impact since it is exponential to the force drag
Range	Independent	Major impact as it relates to energy requirements
Altitude	Independent	Minor impact as air density slightly varies with altitude
Acceleration	Independent	Major impact as it relates to power requirements
TOW	Independent	Major impact as it relates both to energy and power requirements

3.2 SOFTWARE AND INSTRUMENTS

The study will principally make use of two software. OpenVSP (Appendix A) is used to cover the aeronautics, and MATLAB is used for the simulations. Besides, two other software will be employed, Excel and Tinker CAD (Appendix B).

First, the study is founded on OpenVSP, a parametric aircraft geometric tool designed by NASA engineers [41]. This tool allows the virtual creation testing of any aircraft configuration. Once the model is built, the first two variables mentioned in section 3.1.2 can be derived using OpenVSP.

The core of the research however will be completed using MATLAB, as it offers rapid and reliable simulations. An application interface will be developed to facilitate the user interface and thus making this tool readily available for further studies by other parties.

Beside this two software, Excel can be used as a tool to sum outputs results from OpenVSP to parameters for the application. Concerning the circuit, rudimentary simulations are run on TinkerCAD for validation.

3.3 PROCEDURE AND TIMELINE

The overall research was structured on a standard project management workflow, conducted in 5 phases. The Gant chart can be found in the Appendix C.

- **Phase 1: Research Conception and Initiation** marks the beginning of the study. This phase is dedicated to research and review the literature, form the hypothesis and initialize the overall planning.
- **Phase 2: Research Definition and Planning** serves to define the thesis statement, to outline the research scope and goals and to decide on the experimental methodology. The last step of this phase is to finalize the planning of the remaining three phases.
- **Phase 3: Modelling and Testing** form the core of the study. Throughout this phase, the time is first allocated to realize the MATLAB application, to create the 3D models, to run the simulation and to realize a load switch circuit prototype. This phase ends by testing and collecting data from the application and the circuit.
- **Phase 4: Data Analysis** investigates the results obtained from the previous phase. The analysis processes the data in view to uncover meaningful information to confirm or reject the thesis statement. For this thesis, the analysis should be able to determine if a weight reduction can be obtained, and by how much.
- **Phase 5: Research closure** finalizes the study by summarizing in a report the integrity of the research, the key findings and points out to future further study opportunity. The final presentation marks the completion of the thesis.

3.4 ANALYSIS

The selected method for analysing the data is to complete a parametric analysis for each study cases. A parametric analysis allows to isolate and evaluate the impact of each variable, thus providing an understanding toward its direct influence on the resulting battery weight. In more details, each variable from Table 3-1: Variables studied will be varied from -25% to +25% at 5% intervals and plotted in a radar chart to present the resulting battery weight variation impact.

Since the objective of the research is to ultimately reduce the weight of an EA's battery, the first step is to understand the causes behind the battery's weight and requirements. Using a parametric analysis paired with study cases should, theoretically, pinpoint the causes delimiting the battery's performance.

Since each configuration has different performance and requirements, it is expected that a hybrid battery system will be more useful in some cases than others. For example, since Alice has a large battery to achieve its long-range, the battery should be able to cope with the power requirement for take-off, thus not benefiting from a hybrid solution. For Ehang, it is expected to have a minimal weight reduction since the EA uses direct power to sustain flight instead of wings. Therefore, the power requirements will be high throughout the flight. However, the hybrid battery system is expected to have the most impact for VTOL configuration, since take-off requires a high-power burst and cruise requires minimal power.

Model	Configuration	Payload	Typical Range	Expectation
Alice	Fixed wing	9 passengers and 2 crew members	1000km	No benefit
Ehang 184	Multicopter	1 passenger or 100kg	40km	10% to 20% weight reduction
VAU	VTOL	2.5kg	30km	20% to 30% weight reduction

3.5 THE APPLICATION

Instead of manually calculating each study case, a MATLAB application is written to serve as a tool to conduct the simulations (Appendix E). Through the application and its simulations, it will be possible to evaluate how much energy and

power is required for different aircraft configurations and mission profiles. Then, the application will be able to propose a hybrid battery module configuration from the calculated energy and power requirements. The application also possesses a sweep function that evaluates the impact of each selected variables separately through a parametric analysis. This sweep function can then highlight the main factors responsible for validating the research question.

3.5.1 Front-end

The front-end of the application is a simple window to input all variables required to fulfil the simulations and visualize the results. The application is split over two tabs, the first being the "Energy & Power Profile" and second for the "Battery Profile". Each tab holds multiple subsections to classify the data. The subsection in blue are for input, and the subsection in grey are for outputting the results.

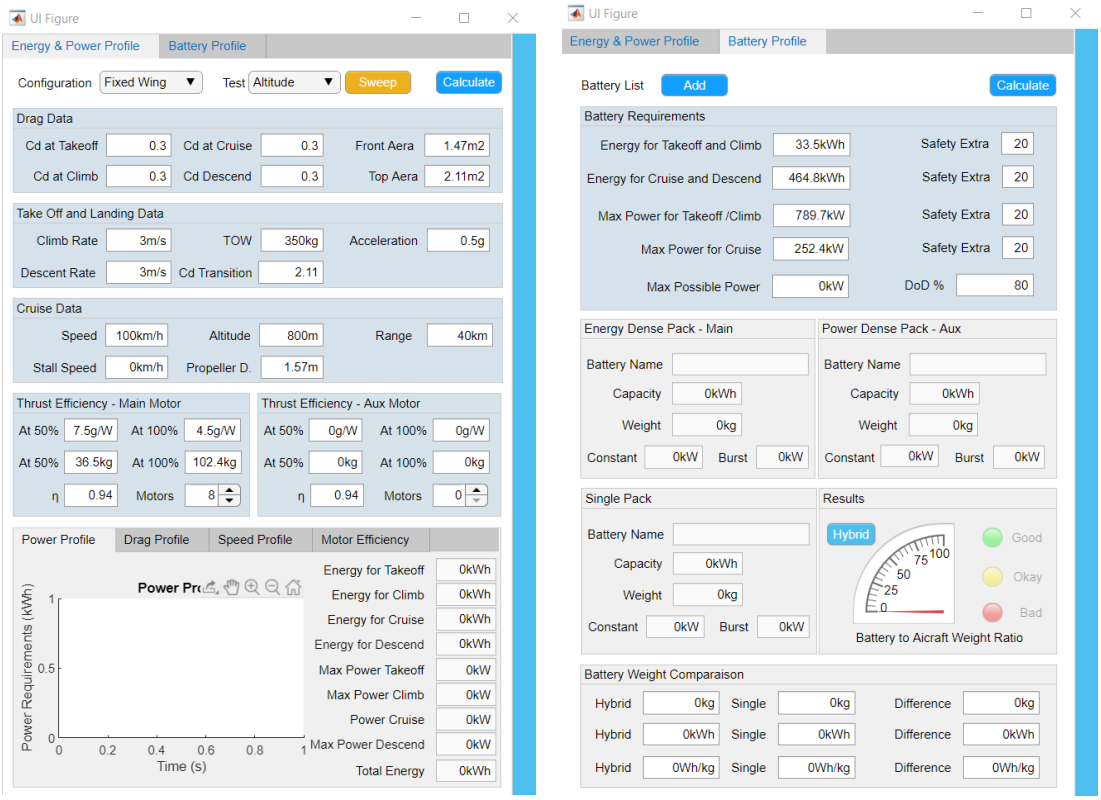


Figure 3-4: Energy & Power Profile on the left and Battery Profile on the right

3.5.2 Back-end

Once all parameters are entered and the "calculate" button is clicked, the program will execute a series of calculations to deduct the energy and power requirements of the aircraft.

3.5.2.1 General structure

The program simulates a hypothetical flight by calculating data points at a one second intervals through each flight phase. The data points are recorded in data structure under each of their respective variable. This data structure consists of the following variables:

- **Time**, to mark the timestamp in second
- **Altitude**, in meters
- **Range**, to mark the current distance travelled in meters
- **Speed**, in meters per seconds
- **ForceDrag**, to specify the force induced by drag in Newtons
- **ForceTotal**, to specify the thrust required by the propulsion in Newtons
- **Power**, to specify the power required by the motors in Watts
- **Efficiency** of the motors in g/W
- **Phase,** to indicate which is the current phase.

The first phase is the take-off. During that phase, the program loops and calculate each variable until the final take-off speed is achieved. For fixed-wing, the final take-off speed is 1.1 the stall speed [42], and for both multicopter and VTOL, it is until climb rate speed is achieved. Since the time step is every second, the speed is calculated by adding the acceleration at each step. For the VTOL, a transition loop is added to accelerate the aircraft up to 1.1 the stall speed.

The second phase is the climb. During that phase, the program loops and calculate each variable until the vehicle reaches cruise altitude and speed. The altitude is calculated by adding the climb rate at every step. Furthermore, a constant acceleration is calculated for the vehicle to go from 1.1 stall speed until cruise speed for the fixed-wing and VTOL, and to go from 0km/h (horizontal speed) to cruise speed for the multicopter.

The third phase is the cruise. Before starting the calculation loop, the program first checks the distance the vehicle will travel in it descends. The cruise distance is then equal to the input range, minus the distance covered during climb and descend.

Once the cruise distance is known, the program loops and calculate each variable until the vehicle achieve the cruise distance.

The final phase of the flight is the descend and landing. The descend follows the same steps as the climb phase but using the descend rate and final speed of 1.3 stall speed [42] for the fixed-wing and VTOL. The landing phase is omitted in this study as the vehicle usually uses parasite drag or relatively minimal energy to stop the vehicle, thus it is not expected to influence the results.

During each loop, the program utilizes three functions where the variables from the current status of the aircraft are inputted to calculate the following status.

3.5.2.2 Functions

The first function is the barometric calculator. This function takes as an input the altitude and outputs the air density. The barometric calculator uses three different standard formulas [43] depending if the aircraft is either in the troposphere, lower stratosphere or upper stratosphere.

When in the troposphere, between 0 to 11km altitude, the formulas for the temperature (T) and pressure (P) are:

$$T = 15.04 - 0.00649 \cdot h$$

$$P = 101.29 \cdot \left[\frac{T + 273.1}{288.08}\right]^{5.256}$$

When in the lower stratosphere, between 11km and 25km, the formulas for the temperature and pressure are:

$$T = -56.46$$

$$P = 22.65 \cdot e^{(1.73 - 0.000157 \cdot h)}$$

When in the upper stratosphere, above 25km, the formulas for the temperature and pressure are:

$$T = -131.21 + 0.00299 \cdot h$$

$$P = 2.488 \cdot \left[\frac{T + 273.1}{216.5}\right]^{-11.388}$$

The air density can then be calculated using the temperature and pressure:

$$\rho = \frac{P}{0.2889 \cdot (T - 273.1)}$$

The second function is the drag calculator. At every step, the air drag is calculated using as input the drag coefficient, the altitude, the speed and the surface area of the vehicle respective to its current status. The air pressure is calculated using the previous function and provides the air density for the drag formula. This function then returns the force caused by drag [44]:

$$F_D = \frac{1}{2}\rho v^2 C_D A \qquad \text{Equation 3-1}$$

The last function is the power calculator, which calculate the power required to propeller the vehicle. The function takes as input arguments the total force required and motor parameters. The total force is calculated by adding the force caused by drag with the force required for the acceleration or deceleration of the vehicle.

Then, the program uses the motor parameters of the efficiency versus thrust at both 50% and 100% power to establish a linear function. Using this function, the power can be estimated for any given thrust requirement. However, the program implements two corrections factors before translating to the power requirement. First, the thrust of the motors is corrected to account for the lower air density in altitude. This is done by multiplying the thrust parameter with the ratio of the air density at ground level to the air density at the current altitude. Secondly, the force required is corrected for the loss of force due to the vehicle's velocity. As seen below, the given force of a motor and its propeller can be model with this formula, with ρ being the air density, A the area covered by the propeller, $v_e$the exit air velocity and $v_0$the intake air velocity [45].

$$F = \frac{1}{2}\rho A(v_e^2 - v_0^2) \qquad \text{Equation 3-2}$$

When testing is done or when the vehicle is static, the formula can be simplified as below:

$$F_s = \frac{1}{2}\rho A v_e^2$$

Thus, the actual force provided by the propeller when the aircraft is at v_0 is then:

$$F_D = F_s - \frac{1}{2}\rho A V_0^2$$

The final power can then be calculated provided with the correction of the air density at altitude and the extra force required for each motor due to the speed of the vehicle.

3.5.3 Battery Weight Calculation

Once the program as calculated all the data points, the power requirement of each point is integrated over time to provide the energy consumption for all phases of flight. As for the power, the highest power requirement of each phase is set as the phase's power requirement.

The program then loops through the battery list to calculate the optimal main battery weight to cover cruise and descend, as well as to calculate the optimal auxiliary battery to cover take-off and climb requirements. For each battery in the list, the program checks the estimated weight of the battery module based on the energy density and power density of the listed battery, then select the highest resulted weight and store this value in an array. After cycling through the list, the program outputs the lowest weight result of the array as the optimal battery configuration. The process is subsequently repeated but for a singular battery solution to serve as reference weight.

3.5.4 Simplifications and Limitations

As previously mentioned, it was required to make multiple simplifications due to the complexity of aerodynamics and the time constraint. Theses simplifications largely originate from the flight profile for each study case and the efficiency of the motors.

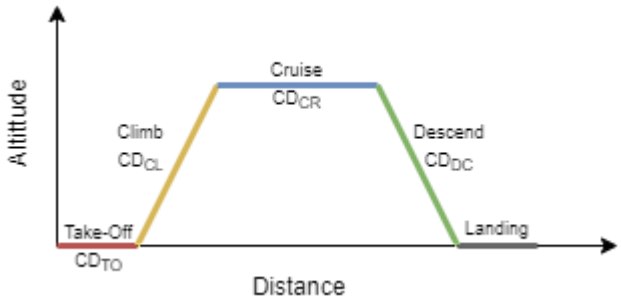


Figure 3-5: Simplified Flight Profile of a Fixed Wing

For this simplified study, the flight profile of the fixed-wing model starts with the take-off phase, where the aircraft accelerates until it reaches 1.1 stall speed. The vehicle then climbs and accelerates until it reaches cruise speed and altitude. After the cruise, the aircraft descends until reaches ground and 1.3 stall speed. However, real-

life profiles are more complex and include for example more detail flight phases such as departure, approach and taxi, but this is beyond the scope of the project.

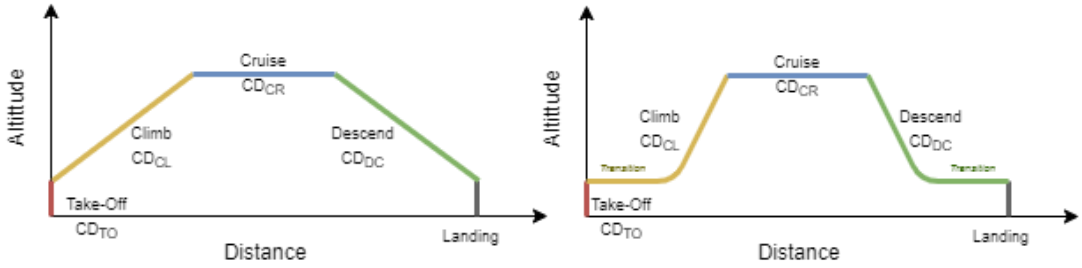


Figure: 3-6: Simplified Flight Profile of a Multicopter on the left and VTOL on the right

The multicopter and VTOL have slightly different flight profile. For the multicopter and VTOL, the take-off and landing are vertical, for which the change in altitude is considered. For the VTOL, a transition stage is added to change between multicopter mode to fixed wing.

The other main simplification is the for the motor efficiency within the power calculator function. The thrust reduction due to the lower air density and dynamic state of the aircraft is complex and would require advance fluid computation and analysis to be done accurately. But for the scope of the project, a direct ratio to adjust for the air density and a force deduction corrected with a factor of 0.25 is estimated to be sufficiently accurate. However, these two simplifications do reduce the accuracy of the simulations and would need to be further improved for later studies.

Besides these two main simplifications, other factors such as the weather conditions and wind current are not considered. As for all the parameters, approximations were taken when the exact number could not be found or calculated. Moreover, the list of battery used for the simulation is limited to Kokam Lithium Ion cell [46] as their inventory adequately covers battery ranging from ultra-high energy density to ultra-high power density.

Chapter 4: Results & Validation

This chapter presents the results from the parametric analysis and the calculated battery weight for each study cases. In every simulation, the parametrically analysed variable is varied from -25% to +25% with increments of 5% to graph its impact on the resulting battery weight between a hybrid and singular solution. The reference values for these simulations are presented in the Appendix D.

4.1 ALICE

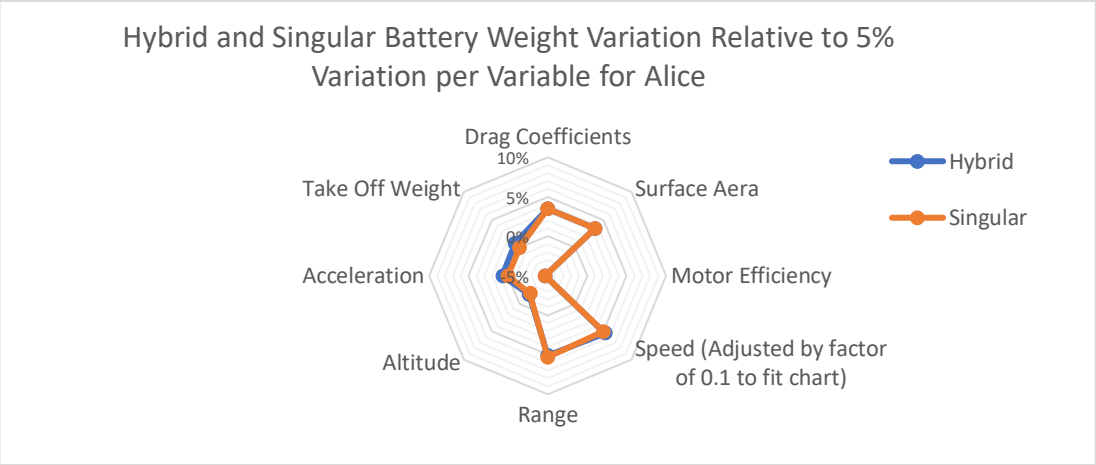


Figure 4-2: Results of the parametric analysis for the fix wing configuration

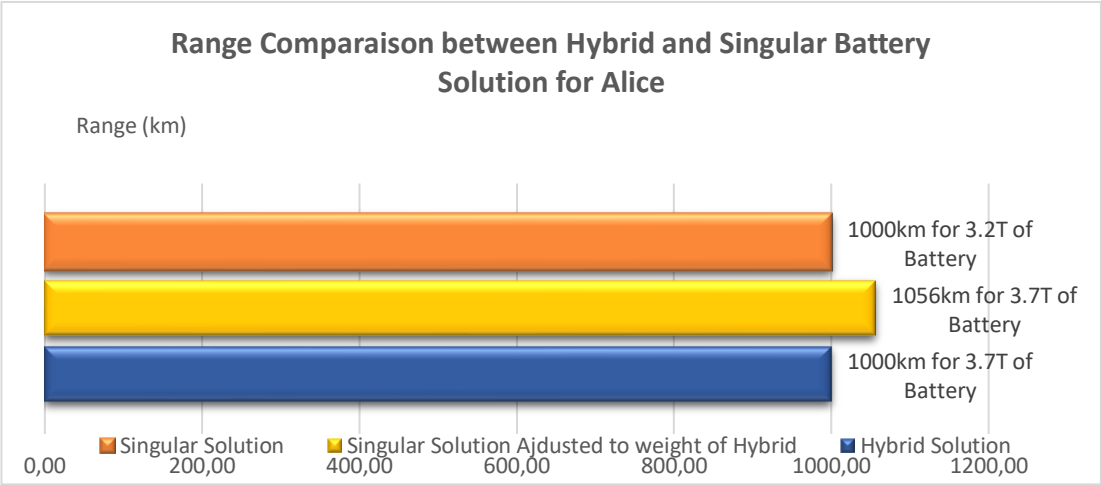


Figure 4-1: Range performance relative to battery weight

Table 4-1: Weight and capacity comparison

	Weight (kg)	Capacity (kWh)
Singular	147.96	26.51
Singular - Weight Adjusted	158.78	28.42
Hybrid	158.78	35.59

4.2 EHANG 184

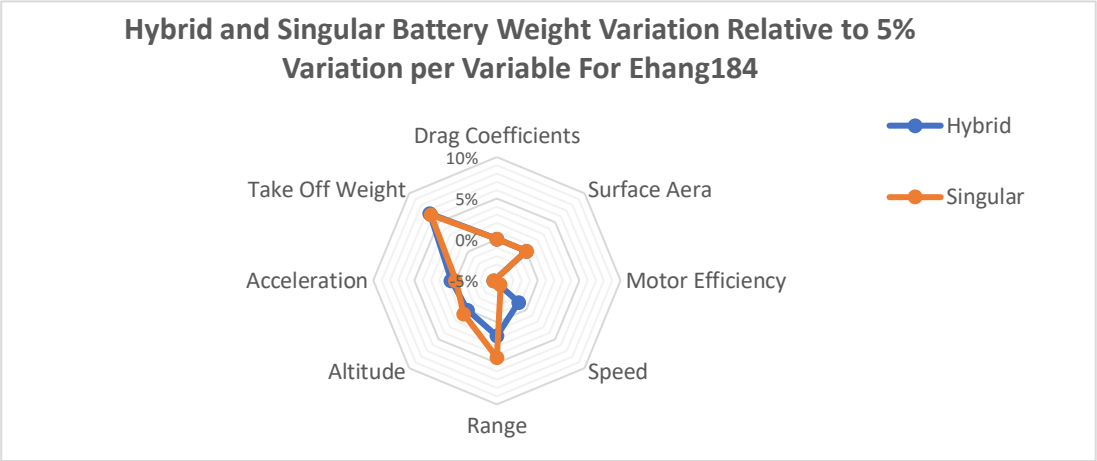


Figure 4-4: Results of the parametric analysis for the multicopter configuration

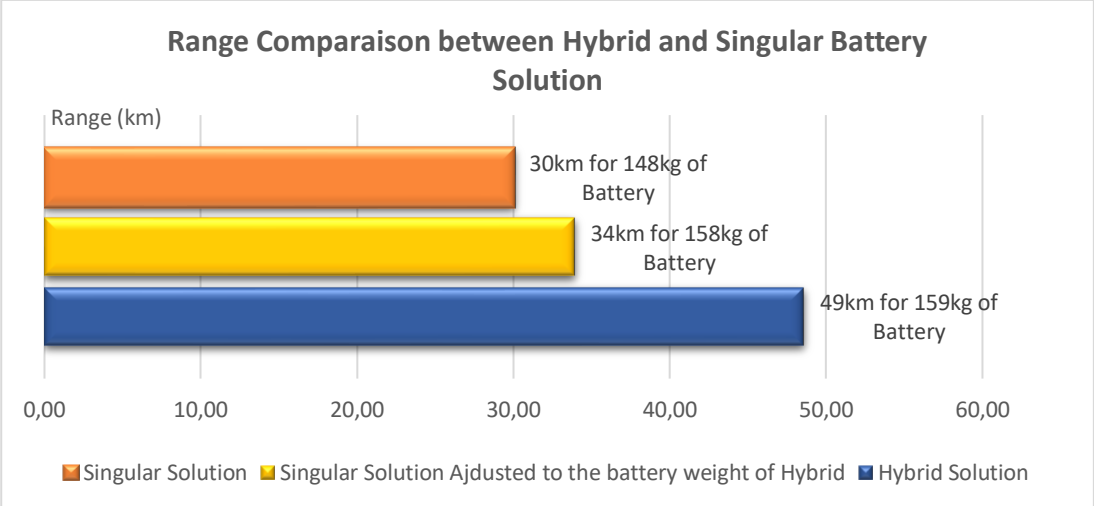
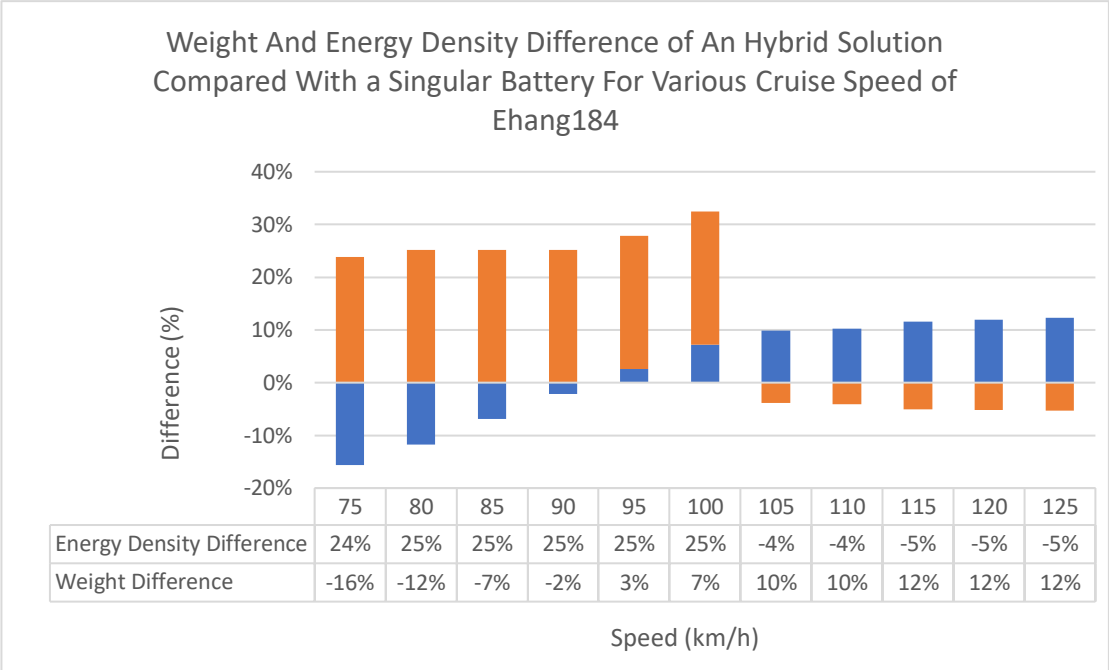


Figure 4-3: Range performance relative to battery weight for the multicopter

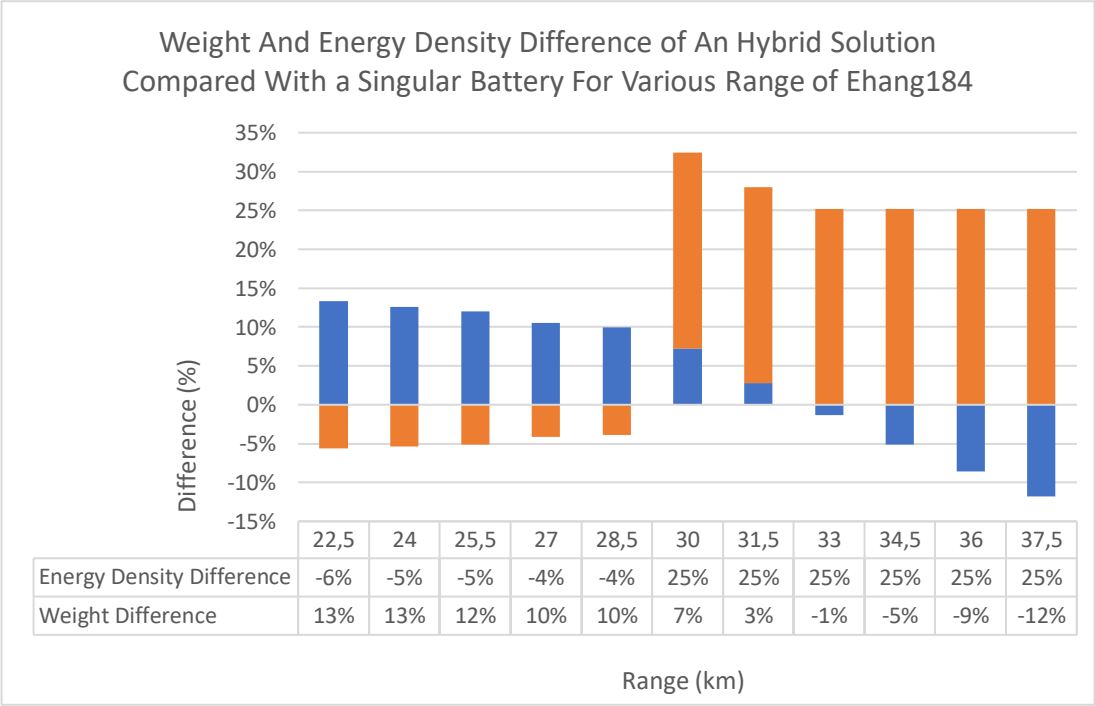
Table 4-2: Weight and capacity comparison

	Weight	Capacity
Singular	147.96	26.51
Singular - Weight Adjusted	158.78	28.42
Hybrid	158.78	35.59



	75	80	85	90	95	100	105	110	115	120	125
Energy Density Difference	24%	25%	25%	25%	25%	25%	-4%	-4%	-5%	-5%	-5%
Weight Difference	-16%	-12%	-7%	-2%	3%	7%	10%	10%	12%	12%	12%

Figure 4-5: Battery weight reduction achieved at different cruise speed for Ehang184



	22,5	24	25,5	27	28,5	30	31,5	33	34,5	36	37,5
Energy Density Difference	-6%	-5%	-5%	-4%	-4%	25%	25%	25%	25%	25%	25%
Weight Difference	13%	13%	12%	10%	10%	7%	3%	-1%	-5%	-9%	-12%

Figure 4-6: Battery weight reduction achieved at different range speed for Ehang184

4.3 VAU

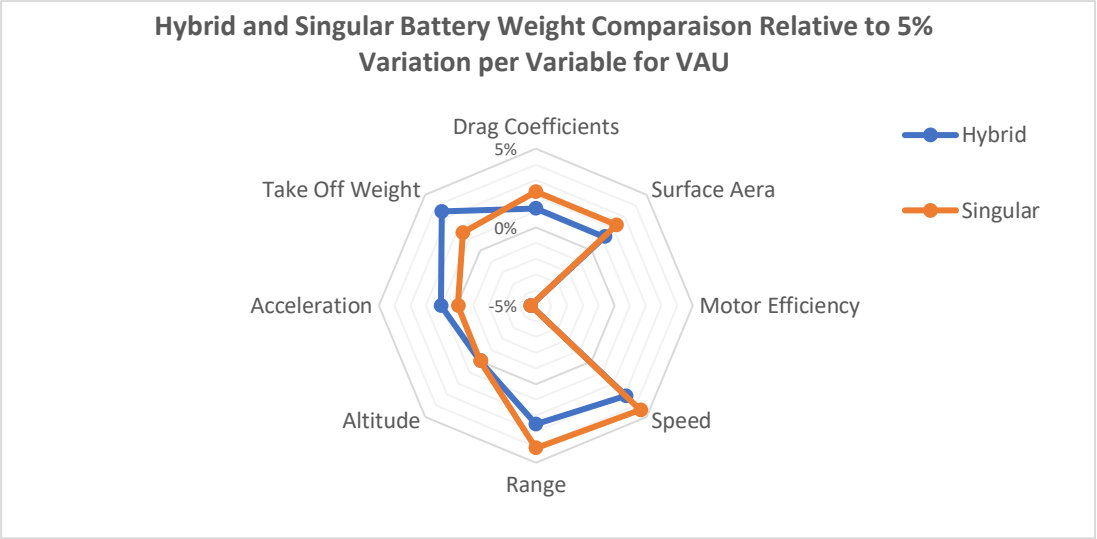


Figure 4-7: Results of the parametric analysis for the VTOL configuration

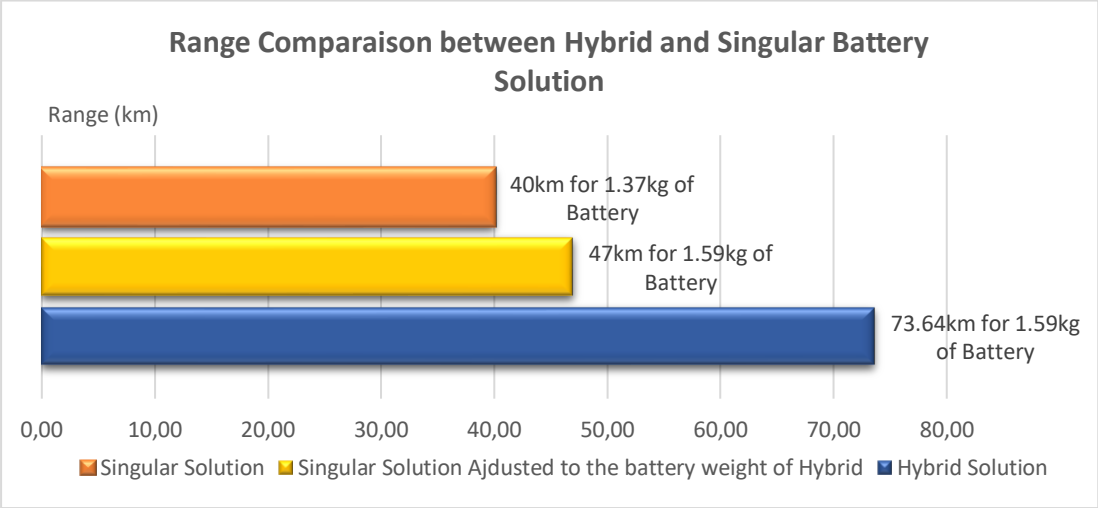


Figure 4-8: Range performance relative to battery weight for the VTOL configuration

Table 4-2: Weight and capacity comparison

	Weight (kg)	Capacity (Wh)
Singular	1.37	204.06
Singular – Weight Adjusted	1.59	226.35
Hybrid	1.59	312.14

Chapter 5: Parametric Analysis

A significant amount of data was collected through the parametric analysis of the 8 variables tested for each study case. Since these results originate from simulations based on multiple simplifications and assumptions, the analysis will not focus on the numerical significance of the results. Instead, the analysis will focus on the relations between these results and develop a conclusion from thereon.

This chapter is divided into three sections. The first section will review the results and their underlying findings, and the second section will evaluate the resulting aircraft performance. The last section will review a proposed load switch circuit approach.

5.1 THE RESULTS

5.1.1 Alice

As it was expected, a hybrid battery solution for the long-range fixed-wing proved not to be beneficial and even increased the overall weight of the battery. This can be explained due to the already large battery required to keep the aircraft flying for at least 2 hours. Since the battery has a relatively large capacity, it can cope with the power-intensive phase at the take-off. Dividing the battery into a hybrid solution would only reduce the overall energy density, thus making the battery heavier.

From the Table 3-1: Variables studied, variations of each variable gave similar battery weight variation for both the hybrid and singular solution, except for the take-off weight and acceleration for which the hybrid is more susceptible. This difference can be attributed to the hybrid solution requiring a singular battery to provide the acceleration power for the take-off phase, thus optimising this battery for power density instead of energy density. Moreover, the drag coefficient, surface area and range variations did closely match with the variation of the battery weight, thus outlining that these variables have a direct impact with the battery weight.

Surprisingly, however, variation of the speed did significantly influence the weight of the battery. At 75% of the reference cruise speed, the battery weight for both the hybrid and singular solution was cut in half. On the other hand, the battery weight went up by 5 folds when simulated with 125% the reference cruise speed. This may be

explained by the equations of drag ($F_D = \frac{1}{2}\rho v^2 C_D A$ Equation 3-1) and propeller force ($F = \frac{1}{2}\rho A(v_e^2 - v_0^2)$ Equation 3-2). At low speed, the drag force of an aircraft remains minimal and the force provided by the propeller is the greatest. However, these change at a higher speed. Since velocity is squared, for example, doubling the velocity of the aircraft will cause four times more drag and thus requires four times more thrust. This also adds to the propeller being negatively affected by the aircraft velocity, as the propeller performance decreases as the aircraft goes faster.

It seems as the performance of Alice could potentially be improved by reducing the cruise speed to achieve the same distance with less energy. But it is most likely that the velocity correction for the motor efficiency in the simulation is not accurate and that Alice's design is already optimised. Besides this possible inaccuracy, the resulting battery weight of 3.2 Tones closely matches the actual Alice's battery of 3.7 Tones. Therefore, the simulation seems to be sufficiently accurate for this study case.

5.1.2 Ehang184

Overall, the results also pointed out that a hybrid solution for Ehang184 is not always beneficial. In cruise speed of 75 to 90km/h and in the operational range of 33km to 48km, a hybrid solution did provide a battery weight reduction relative to a singular solution. Unfortunately, this is because the singular battery solution was heavier compared to the reference value for lower speed and longer range, and so does not give a significant reduction compared to the reference weight. This can likely be attributed to the design already being optimised for this speed and range.

Unexpectedly, however, a hybrid solution resulted in a 25% energy density increase for only 7% weight increase, thus substantially improving the aircraft's performance for a minimal weight cost. This result points out that a hybrid solution inadvertently cost additional weight compared to a singular battery, but with the benefits of a higher energy density.

As anticipated, the parametric analysis of the multicopter gave distinct results compared to the fixed-wing. Since multicopter does not travel as fast as fixed-wing, variables directly linked to the drag force, as the drag coefficients and surface area, have minimal impact on the battery weight. Another interesting result, however, is that a hybrid solution seems to be more stable to speed and range variation. But for both

solutions and increase in take-off weight can be directly linked to an increase in the aircraft's battery weight, thus outlining that TOW is the most impactful variable for this configuration.

Simulation wise, the estimated weight of 148kg paired with a person of 75kg would leave about 100kg to realise the construction of the vehicle. Although there is no mention on the vehicle's battery weight online, it is safe to assume that the simulation likely matches the real value. Furthermore, the hybrid solution would give an additional 20km range on top of the original 30km range for only 10kg battery weight extra. This would be achievable aircraft optimisation.

5.1.3 VAU

Once again, the parametric analysis profile is unique to the aircraft's configuration. For VAU, a hybrid solution seems to be generally more stable to multiple variable variations compared to a singular solution. For exception, a hybrid solution is more susceptible to variation of take-off weight and take-off acceleration. But for both variables, a variation of 5% still results in less than 5% battery weight variation.

Surprisingly again, the hybrid solution did improve the aircraft's performance but not as expected. For an 11% weight increase, the hybrid solution gives a 38% energy density increase. Thus, further supporting the result that a hybrid solution can improve an aircraft's range but to the cost of a slight weight increase.

Finally, the simulations appeared to be accurate as of the estimated 1.4kg of battery matches to the 1.2kg of the real design. Furthermore, the results conclude that an increase of 200gr in battery could improve the operational range from 40km to 73km, which would be an acceptable weight increase.

5.2 AICRAFT PERFORMANCE REVIEW

Overall, the simulations provided different results than it was previously expected. Due to the nature of the hybrid solution dividing the battery in two separate batteries, the smaller power-dense pack requires a minimal additional weight for both multicopter and VTOL to meet the power requirements of take-off. However, this pays off by resulting in a higher energy-density for the energy dense pack, thus resulting in an overall energy density increase.

For the multicopter and VTOL configurations, a hybrid solution could considerably increase the operational range but to the cost of a heavier battery, thus slightly limiting the original payload capacity. For long-range fixed-wing, a hybrid solution is likely not beneficial in either weight reduction or range increase.

5.3 LOAD SWITCHING CIRCUIT

Since N-Channel power MOSFETs have the lowest resistance (section 2.3), is easily scalable as they function well in parallel [47], and can be used for high-power switch applications, they are the component of choice to realise the load switching circuit. Further development of the circuit was realized, but it is out of the scope of this research. More information can be found in the Appendix B.

Chapter 6: Conclusions and Future Work

Through the presented research, an in-depth and practical approach was taken to analyses the benefit of utilizing a hybrid battery solution for optimising EA performance.

As the research question entails: can a hybrid battery system made of a high specific energy battery pack and a high specific power battery pack optimise the performance of an electric aircraft by reducing the overall weight of the vehicle's battery?

To which the answer is, not exactly. Hybrid battery solution proved to result in a heavier battery system for EA with fixed-wing, multicopter and VTOL configuration. However, a hybrid battery solution resulted in a significant energy density increase for a relatively small battery weight increase. Thus, a hybrid solution could potentially increase the operational range of multicopter and VTOL by 63% and 84% with only 7% and 16% battery weight increase respectively.

Concerning the load switch circuit technique, it is recommended to use N-Channel MOSFET in parallel and in a dual configuration as presented in Figure 2-5: Dual MOSFETs configuration to prevent reverse current With this approach, the load switch circuit is expected to be scalable for EA and minimize energy losses.

Further study is nevertheless required to confirm these results, as the simulations used multiple assumptions and simplifications, as well as a limited reference battery list. A more in-depth look into the weight penalty of a hybrid solution may also be interesting to study to find potentially solutions. Perhaps, a multiple energy system might even provide a further performance increase.

To conclude, a hybrid battery solution seems to be a promising approach to further improve EA performance by increasing their operational range. Although this approach may not always be worth the weight penalty, it might prove itself indispensable in the future if ASSB cannot meet the demanding power requirement of EA. For now, however, the next step of this research is to apply the findings to VAU and hopefully, validate its results.

Bibliography

[1] NASA, “Do scientists agree on climate change?,” 15 January 2020. [Online]. Available: https://climate.nasa.gov/faq/17/do-scientists-agree-on-climate-change/.

[2] C. Nunez, “Carbon dioxide levels are at a record high. Here's what you need to know.,” National Geographic, 13 May 2019. [Online]. Available: https://www.nationalgeographic.com/environment/global-warming/greenhouse-gases/.

[3] H. R. a. M. Roser, “CO2 and Greenhouse Gas Emissions,” Our World in Data, December 2019. [Online]. Available: https://ourworldindata.org/co2-and-other-greenhouse-gas-emissions#citation.

[4] Climate Action, “Reducing emissions from aviation,” European Commision, [Online]. Available: https://ec.europa.eu/clima/policies/transport/aviation_en.

[5] G. Topham, “Airlines' CO2 emissions rising up tp 70% faster than predicted,” The Guardian, 19 September 2019. [Online]. Available: https://www.theguardian.com/business/2019/sep/19/airlines-co2-emissions-rising-up-to-70-faster-than-predicted.

[6] ATAG, “Facts & Figures,” Air Transport Action Group, January 2020. [Online]. Available: https://www.atag.org/facts-figures.html.

[7] E. Adams, “The Age of Electric Aviation Is Just 30 Years Away,” WIRED, 5 August 2017. [Online]. Available: https://www.wired.com/2017/05/electric-airplanes-2/.

[8] L. Gartenberg, “Battery Centric Serial Hybrid Aircraft Perforamance and Design Space,” Embry-Riddle Aeronautical University, Daytona Beach, Florida, 2017.

[9] M. H. Sadraey, Aircraft Design, A systems Engineering Approach, A John Wiley & Sons, Ltd., Publication, 2013.

[10] R. Matulka, “The Histroy of the Electric Car,” U.S. Department of Energy, 15 September 2014. [Online]. Available: https://www.energy.gov/articles/history-electric-car.

[11] B. Berman, “When Old Things Turn Into New Again,” The New York Times, 24 October 2007. [Online]. Available: https://www.nytimes.com/2007/10/24/automobiles/autospecial/24history.html.

[12] W. W. C. I. a. G. Cooke, Global Energy Innovation: Why America Must Lead, Praeger, 2011.

[13] E. Wikner, “Lithium ion Battery Aging: Battery Lifetime Testing and Physics-based Modeling for Electric Vehicle Applications,” Chalmers University of Technology, Göteborg, Sweden, 2017.

[14] B. H. B. N. P. Gang Ning, “Capacity fade study of lithium-ion batteries cycled at high discharge rates,” University of South Carolina, 2002.

[15] Library of Congress, “Tissandier Collection,” United State Legislative Information.

[16] R. Klein, “The Current State of Electric Aircraft,” The Leardo Time, [Online]. Available: http://leonardotimes.com/2016/08/11/the-current-state-of-electric-aircraft/.

[17] Y. Gibbs, “NASA Armstrong Fact Sheet: Solar-Power Research,” NASA, 28 February 2014. [Online]. Available: https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-054-DFRC.html.

[18] Y. Gibbs, “NASA Armstrong Fact Sheet: Helios Prototype,” NASA, 7 August 2017. [Online]. Available: https://www.nasa.gov/centers/armstrong/news/FactSheets/FS-068-DFRC.html.

[19] M. Bradley, “How sweet the future of aviation,” Boeing, August 2017. [Online]. Available: https://www.boeing.com/features/innovation-quarterly/aug2017/feature-technical-sugar.page.

[20] AIRBUS, “Electric flight: Bringing zero-emission technology to aviation,” AIRBUS, [Online]. Available: https://www.airbus.com/innovation/future-technology/electric-flight.html.

[21] A. L. Samantha Kent, “Harbour Air and magniX Announce Successful Flight of World’s First Commercial Electric Airplane,” Habrour Air Seaplanes, 10 December 2019. [Online]. Available: https://www.harbourair.com/harbour-air-and-magnix-announce-successful-flight-of-worlds-first-commercial-electric-airplane/.

[22] Solar Impulse, “Historic flight,” Solar Impulse, [Online]. Available: https://aroundtheworld.solarimpulse.com.

[23] Boeing, [Online]. Available: https://www.boeing.com.

[24] Hyundai, [Online]. Available: https://www.hyundai.com/worldwide/en/brand/ces-2020.

[25] Lilium Jet, “Enabling a world where anyone can fly,” Lilium, [Online]. Available: https://lilium.com.

[26] AIRBUS, [Online]. Available: https://www.airbus.com/innovation/urban-air-mobility/vehicle-demonstrators/vahana.html.

[27] EHang, [Online]. Available: https://www.ehang.com/ehang184/.

[28] Volocopter, “We Bring Urban Air Mobility To Life,” Volocopter, [Online]. Available: https://www.volocopter.com/en/.

[29] Harbour Air Seaplanes, [Online]. Available: https://www.harbourair.com.

[30] Eviation, “Alice, All Electric,” [Online]. Available: https://www.eviation.co.

[31] Y. V. A. A. Robert Cameron Bolam, “Review of Electrically Powered Propulsion for Aircraft,” Glyndwr University & Moscow Power Engineering Institute.

[32] A. S. L. L. A. I. A. S. H. Kuhn, “PROGRESS AND PERSPECTIVES OF ELECTRIC AIR,” International congress of the aeronautical sciences.

[33] J. M. Vegh, “SIZING METHODOLOGIES FOR AIRCRAFT,” Stanford University, 2018.

[34] MIT Libraries, “The Volataic Pile,” Massachusetts Institute of technology, https://libraries.mit.edu/collections/vail-collection/topics/electricity/the-voltaic-pile/.

[35] B. L. e. Lyonel-Feininger, “FUNDAMENTAL PREREQUISITES FOR ELECTRIC FLYING,” Deutscher Luft- und Raumfahrtkongress, 2012.

[36] Battery University, “What's the best battery,” Battery University, 21 March 2017. [Online]. Available: https://batteryuniversity.com/learn/archive/whats_the_best_battery.

[37] R. T. S.-I. O. Y. S. L. a. J.-H. J. Janardhanan. R. Rani, “An Ultra-High-Energy Density Supercapacitor; Fabrication Based on Thiol-functionalized Graphene Oxide Scrolls,” Nanomaterials, 2019.

[38] A. Rufer, “The Dream of Efficient Energy Storage,” EPFL. Ecole Polytechnique Fédérale de Lausanne.

[39] A. J. Hawkins, “Toyota will offer robot taxi rides during the 2020 Summer Olympics in Tokyo,” The Verge, 24 October 2019. [Online]. Available: https://www.theverge.com/2019/10/24/20930227/toyota-self-driving-car-taxi-tokyo-test-2020-olympics.

[40] C. B. a. C. Bull, “IGBT or MOSFET: Choose Wisely,” International Rectifier.

[41] Z. S. a. A. Kaknevicius, “Reverse Current Protection in Load Switches,” Texas Instruments, 2016.

[42] OpenVSP, “Open Source VEhicle Sketch Pad”.

[43] Glen Research Center, “Earth Atmosphere Model,” Nasa, [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/airplane/atmosmet.html.

[44] Glenn research Center, “The Drag Equation,” NASA, [Online]. Available: https://www.grc.nasa.gov/www/k-12/airplane/drageq.html.

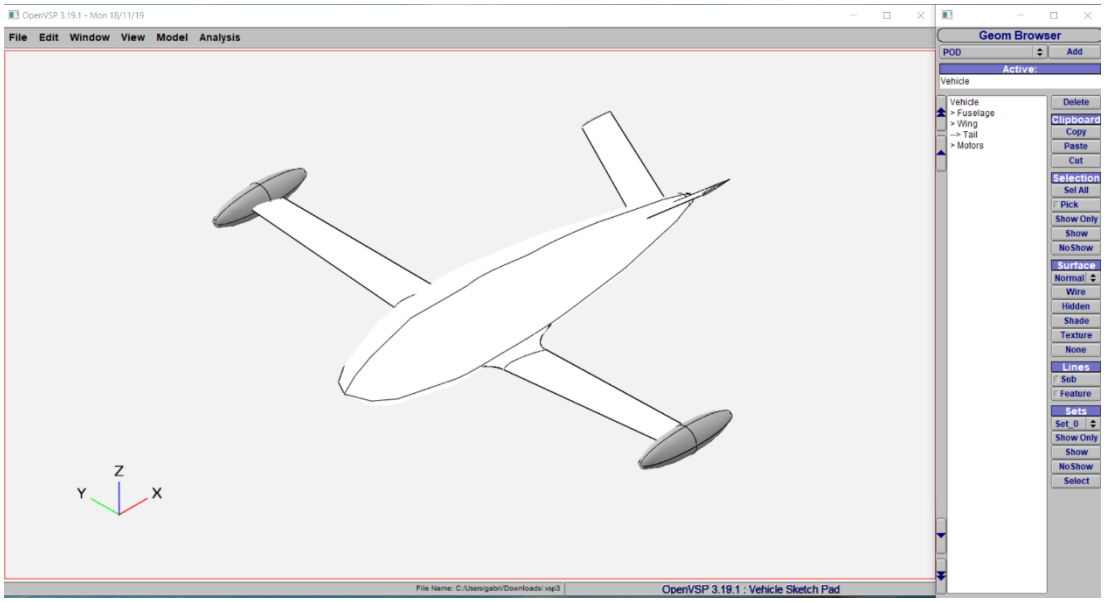
[45] Glenn Research Center, “Propeller Thrust,” NASA, [Online]. Available: https://www.grc.nasa.gov/WWW/K-12/airplane/propth.html.

[46] Kokam, [Online]. Available: www.kokam.com.

[47] NXP Semiconductors, “Using power MOSFETs in parallel,” NXP, 2015.

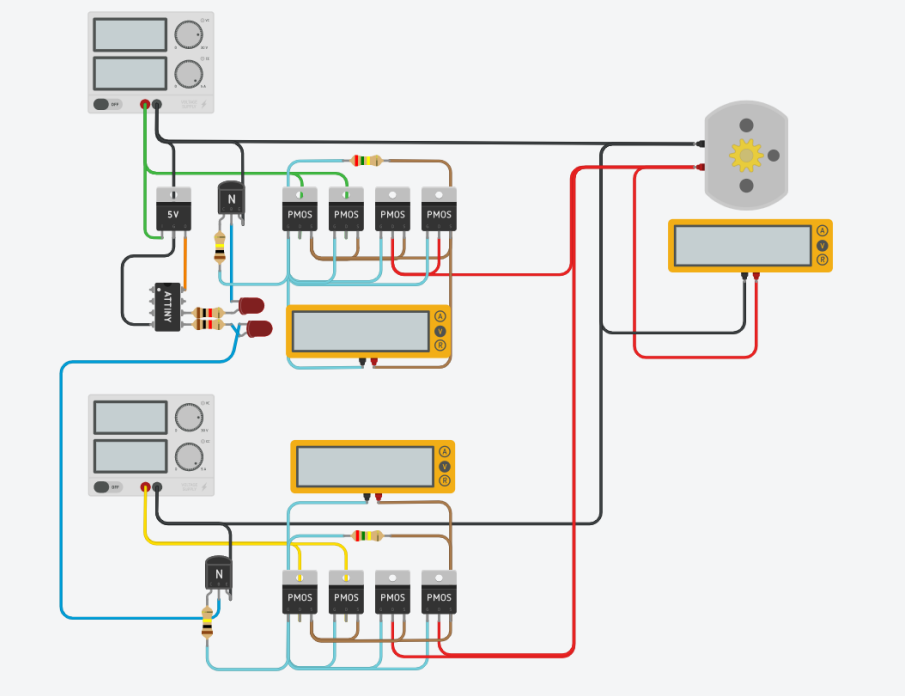
Appendices

Appendix A

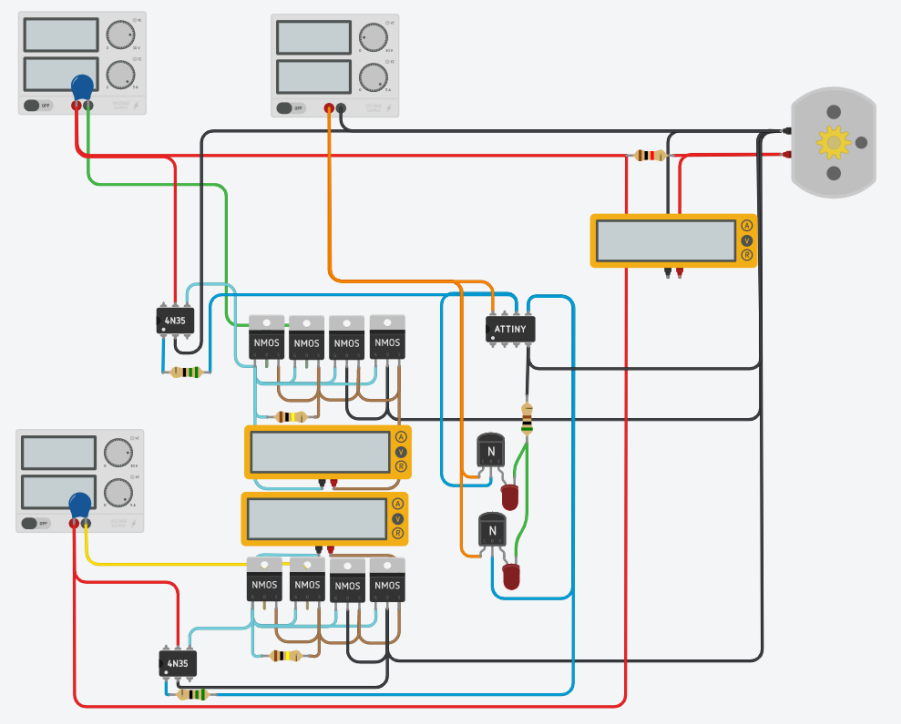


Appendix A 1: Model of Alice in OpenVSP

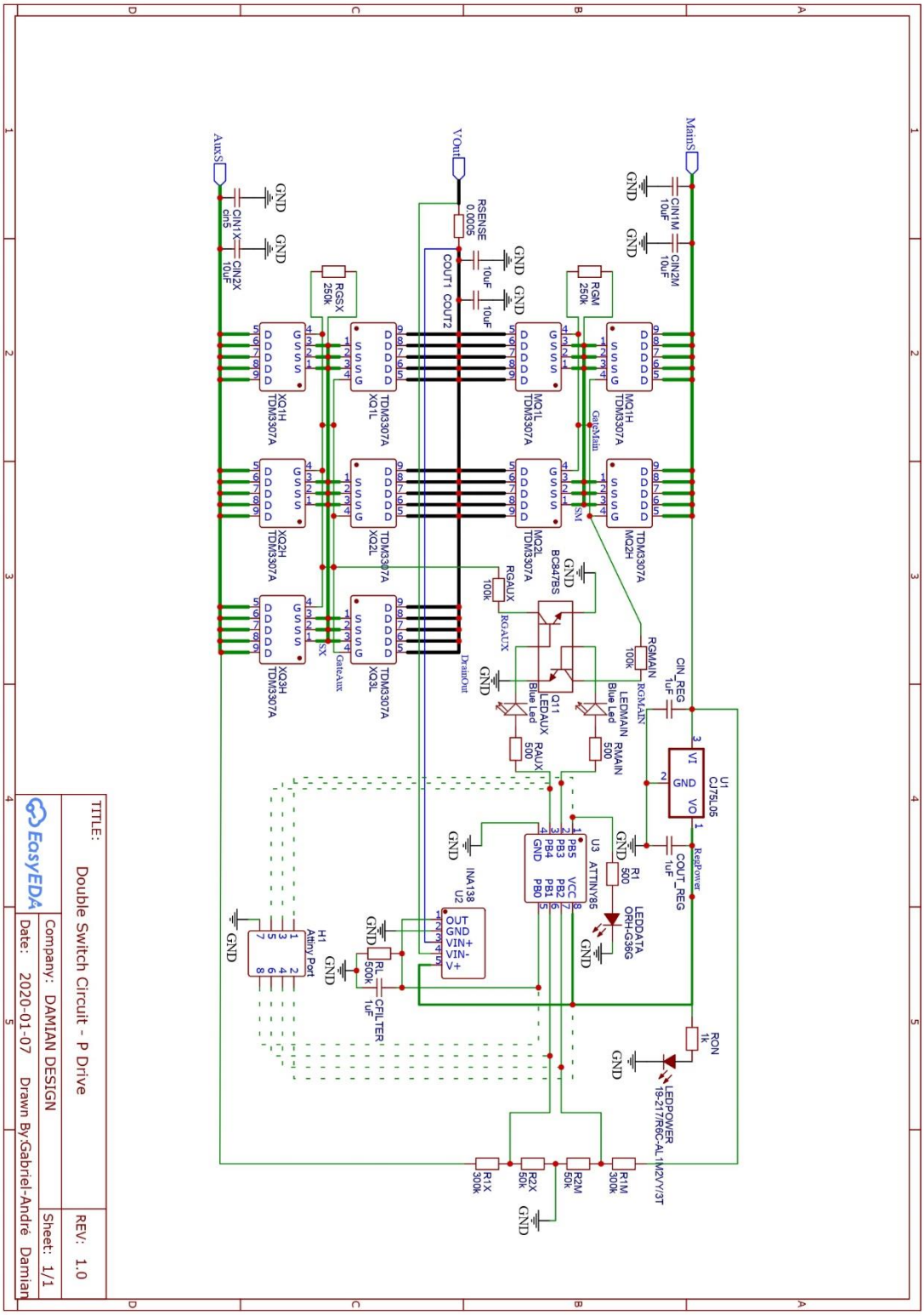
Appendix B



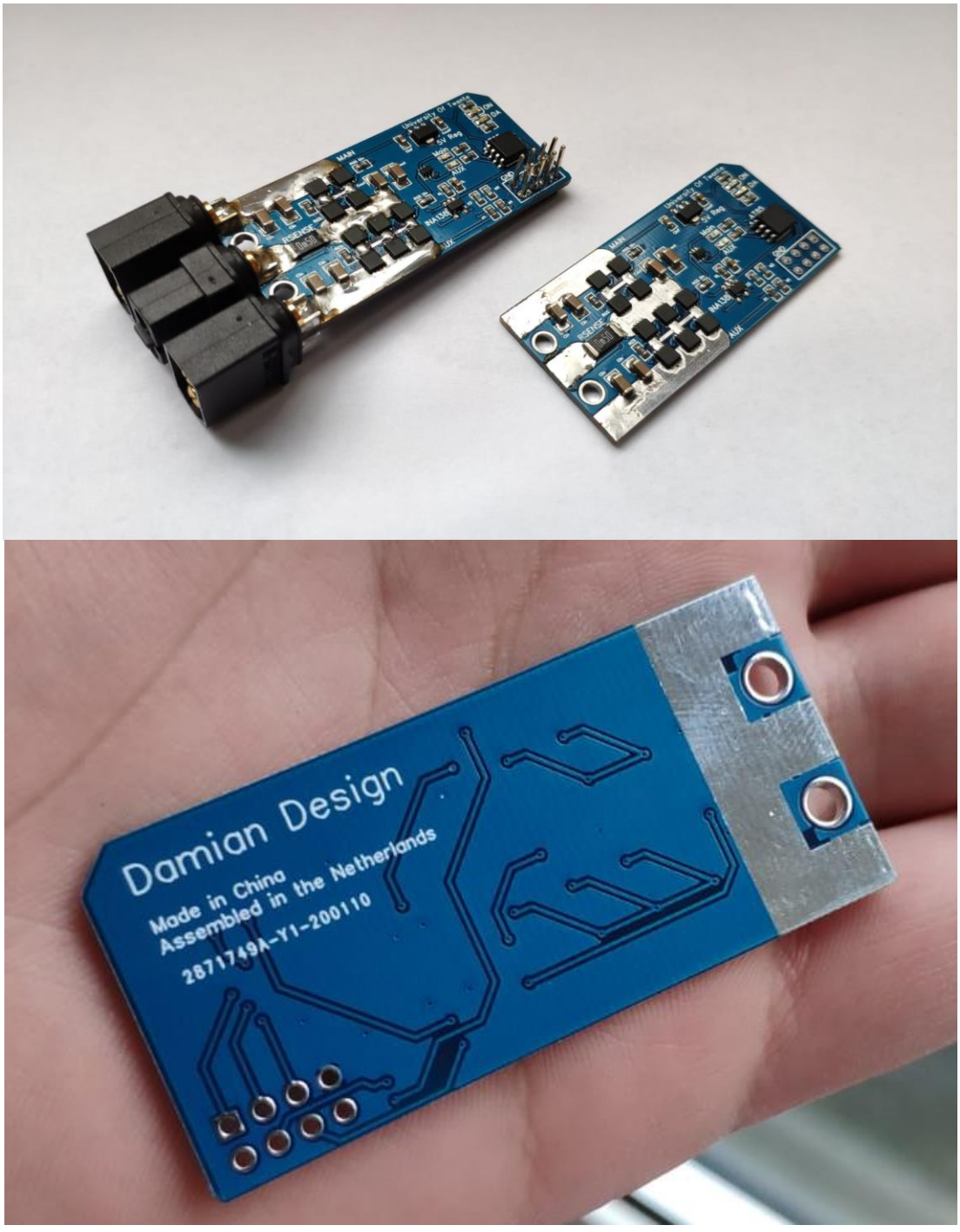
Appendix B 2: P-Channel Load Switch Circuit Tested in TinkerCAD



Appendix B 1: N-Channel Load Switch Circuit Tested in TinkerCAD

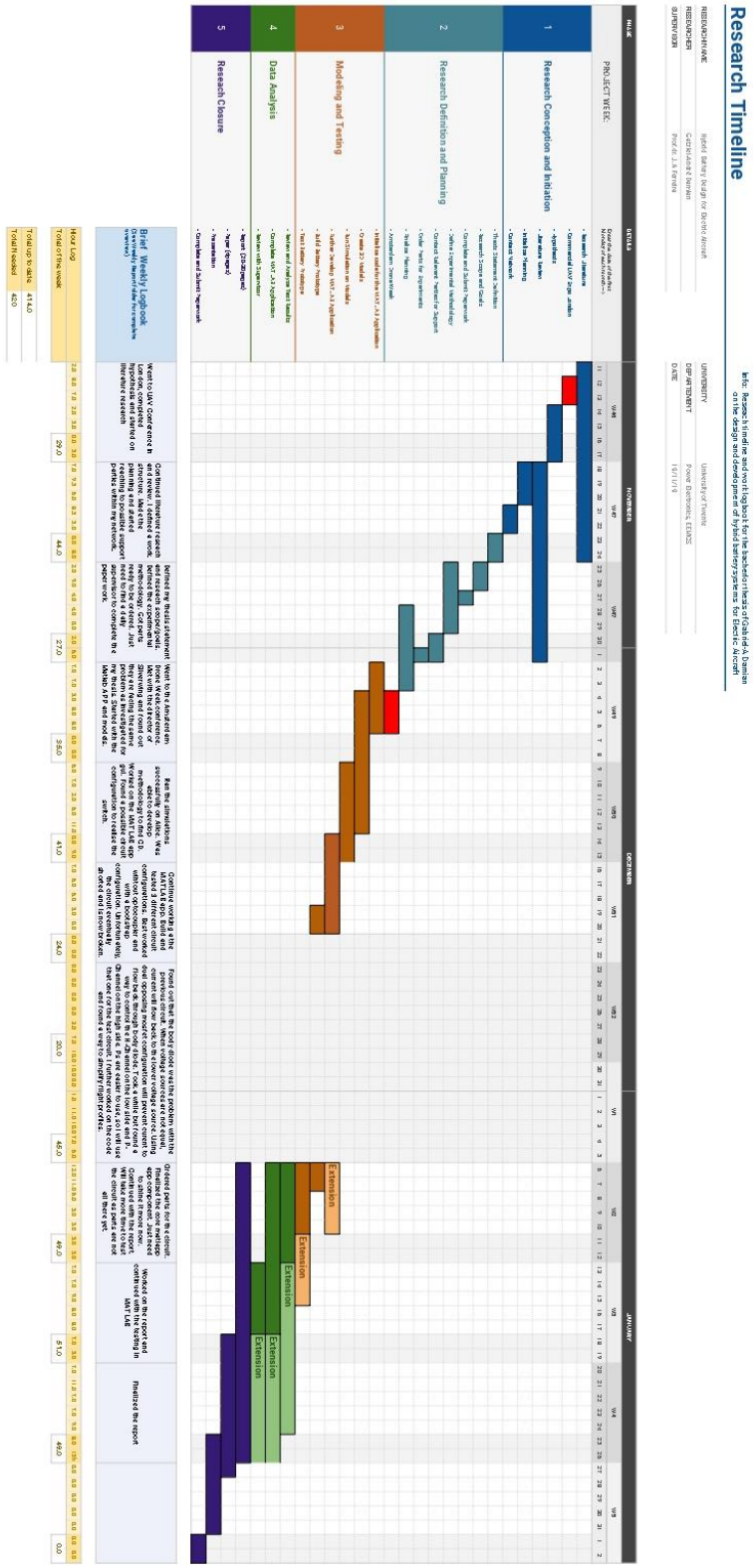


Appendix B 3: Schematic of P-Channel Load Switch Circuit designed on EasyEDA



Appendix B 4: Built PCB

Appendix C


Research Timeline
Research Conception and Initiation
Research Definition and Planning
Modeling and Testing
Data Analysis
Reseach Closure
Extension

Appendix D

Reference Variable	Fixed-wing	Multicopter	VTOL
Cd at Take-off	0.04	0.3	0.04
Cd at Climb	0.035	0.3	0.035
Cd at Cruise	0.025	0.3	0.03
Cd at Descend	0.025	0.3	0.025
Cd Transition	N/A	N/A	0.3
Front Area (m2)	30	1.47	1
Top Area (m2)	N/A	2.11	1.5
Climb Rate (m/s)	4	3	6
Descent Rate (m/s)	3	3	6
TOW (KG)	6000	325	6.9
Acceleration (g)	0.5	0.5	0.5
Speed (km/h)	480	100	70
Stall Speed (km/h)	100	0	50
Cruise Altitude (m)	3000	500	20
Propeller Diameter (m)	1.65	1.57	0.33
Range (km)	1000	30	40
50% Power Efficiency (W/g)	12	7.5	7.3
100% Power Efficiency (W/g)	4	4.5	4.69
50% Thrust (kg)	400	36.5	1.669
100% Thrust (kg)	1200	102.4	4.235
Motor Efficiency	0.94	0.94	0.94
Number of motor	3	8	2
Number of motor VTOL	N/A	N/A	4

Appendix D 1: List of reference values used for the simulations

For visualize all results, please refer to the
Test_Results_Parametric_Analysis.xlsx spreadsheet.

Appendix E

MATLAB Application

```
classdef HybridConfig < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TabGroup                        matlab.ui.container.TabGroup
        EnergyPowerProfileTab           matlab.ui.container.Tab
        DragDataPanel                   matlab.ui.container.Panel
        CdatTakeoffLabel                matlab.ui.control.Label
        CdTakeOff                       matlab.ui.control.NumericEditField
        CdatCruiseLabel                 matlab.ui.control.Label
        CdCruise                        matlab.ui.control.NumericEditField
        SurfaceAeraCruise               matlab.ui.control.NumericEditField
        FrontAeraLabel                  matlab.ui.control.Label
        CdDescendLabel                  matlab.ui.control.Label
        CdDescend                       matlab.ui.control.NumericEditField
        CdatClimbLabel                  matlab.ui.control.Label
        CdClimb                         matlab.ui.control.NumericEditField
        TopAeraLabel                    matlab.ui.control.Label
        SurfaceAeraTakeOff              matlab.ui.control.NumericEditField
        TakeOffandLandingDataPanel      matlab.ui.container.Panel
        AccelerationEditFieldLabel      matlab.ui.control.Label
        AccelerationTakeOff             matlab.ui.control.NumericEditField
        TOW                             matlab.ui.control.NumericEditField
        TOWEditFieldLabel               matlab.ui.control.Label
        ClimbRateEditFieldLabel         matlab.ui.control.Label
        ClimbRate                       matlab.ui.control.NumericEditField
        DescentRateEditFieldLabel       matlab.ui.control.Label
        DescentRate                     matlab.ui.control.NumericEditField
        CdTransitionLabel               matlab.ui.control.Label
        CdTrans                         matlab.ui.control.NumericEditField
        CruiseDataPanel                 matlab.ui.container.Panel
        SpeedEditFieldLabel             matlab.ui.control.Label
        Speed                           matlab.ui.control.NumericEditField
        AltitudeEditFieldLabel          matlab.ui.control.Label
        Altitude                        matlab.ui.control.NumericEditField
        Range                           matlab.ui.control.NumericEditField
        RangeEditFieldLabel             matlab.ui.control.Label
        StallSpeedLabel                 matlab.ui.control.Label
        StallSpeed                      matlab.ui.control.NumericEditField
        PropellerDiameter               matlab.ui.control.NumericEditField
        PropellerDLabel                 matlab.ui.control.Label
        CalculateButton                 matlab.ui.control.Button
        ThrustEfficiencyMainMotorPanel  matlab.ui.container.Panel
        At50EditFieldLabel              matlab.ui.control.Label
        MidThrottlePowerMain            matlab.ui.control.NumericEditField
        At100EditFieldLabel             matlab.ui.control.Label
        FullThrottlePowerMain           matlab.ui.control.NumericEditField
        At50EditFieldLabel_2            matlab.ui.control.Label
        MidThrottleThrustMain           matlab.ui.control.NumericEditField
        At100EditFieldLabel_2           matlab.ui.control.Label
        FullThrottleThrustMain          matlab.ui.control.NumericEditField
        MotorsLabel                     matlab.ui.control.Label
        NumMain                         matlab.ui.control.Spinner
        Label                           matlab.ui.control.Label
        MainEfficiency                  matlab.ui.control.NumericEditField
        ConfigurationDropDownLabel      matlab.ui.control.Label
        ConfigurationDropDown           matlab.ui.control.DropDown
        ThrustEfficiencyAuxMotorPanel   matlab.ui.container.Panel
        At50EditFieldLabel_3            matlab.ui.control.Label
        MidThrottlePowerAux             matlab.ui.control.NumericEditField
        At100EditFieldLabel_3           matlab.ui.control.Label
        FullThrottlePowerAux            matlab.ui.control.NumericEditField
        At50EditFieldLabel_4            matlab.ui.control.Label
        MidThrottleThrustAux            matlab.ui.control.NumericEditField
        At100EditFieldLabel_4           matlab.ui.control.Label
        FullThrottleThrustAux           matlab.ui.control.NumericEditField
        MotorsLabel_2                   matlab.ui.control.Label
        NumAux                          matlab.ui.control.Spinner
        Label_2                         matlab.ui.control.Label
        AuxEfficiency                   matlab.ui.control.NumericEditField
        EnergyPowerInfo                 matlab.ui.container.TabGroup
        PowerProfileTab                 matlab.ui.container.Tab
        UIAxesPowerProfile              matlab.ui.control.UIAxes
        EnergyforTakeoffEditFieldLabel  matlab.ui.control.Label
        EnergyTakeOff                   matlab.ui.control.NumericEditField
        MaxPowerTakeoffLabel            matlab.ui.control.Label
        PowerTakeOff                    matlab.ui.control.NumericEditField
        MaxPowerClimbLabel              matlab.ui.control.Label
        PowerClimb                      matlab.ui.control.NumericEditField
        EnergyforCruiseEditFieldLabel   matlab.ui.control.Label
        EnergyCruise                    matlab.ui.control.NumericEditField
        TotalEnergyEditFieldLabel       matlab.ui.control.Label
        TotalEnergy                     matlab.ui.control.NumericEditField
        EnergyforClimbLabel             matlab.ui.control.Label
        EnergyClimb                     matlab.ui.control.NumericEditField
        EnergyforDescendLabel           matlab.ui.control.Label
        EnergyDescend                   matlab.ui.control.NumericEditField
        PowerCruiseLabel                matlab.ui.control.Label
        PowerCruise                     matlab.ui.control.NumericEditField
        MaxPowerDescendLabel            matlab.ui.control.Label
        PowerDescend                    matlab.ui.control.NumericEditField
        DragProfileTab                  matlab.ui.container.Tab
        UIAxesDragProfile               matlab.ui.control.UIAxes
        MaxDragTakeOffLabel             matlab.ui.control.Label
        DragTakeOff                     matlab.ui.control.NumericEditField
        MaxDragDescendLabel             matlab.ui.control.Label
        DragDescend                     matlab.ui.control.NumericEditField
        ForceAccelerationLabel          matlab.ui.control.Label
        ForceAcceleration               matlab.ui.control.NumericEditField
        MaxDragClimbLabel               matlab.ui.control.Label
        DragClimb                       matlab.ui.control.NumericEditField
        DragCruiseLabel                 matlab.ui.control.Label
        DragCruise                      matlab.ui.control.NumericEditField
        SpeedProfileTab                 matlab.ui.container.Tab
        UIAxesSpeedProfile              matlab.ui.control.UIAxes
        TimeTakeOffLabel                matlab.ui.control.Label
        DisplayTimeTakeOff              matlab.ui.control.NumericEditField
        TimeDescendLabel                matlab.ui.control.Label
        DisplayTimeDescend              matlab.ui.control.NumericEditField
        TotalTimeLabel                  matlab.ui.control.Label
        DisplayTotalTime                matlab.ui.control.NumericEditField
        TimeClimbLabel                  matlab.ui.control.Label
        DisplayTimeClimb                matlab.ui.control.NumericEditField
        TimeCruiseLabel                 matlab.ui.control.Label
        DisplayTimeCruise               matlab.ui.control.NumericEditField
        RangeClimbLabel                 matlab.ui.control.Label
        DisplayRangeClimb               matlab.ui.control.NumericEditField
        RangeCruiseLabel                matlab.ui.control.Label
        DisplayRangeCruise              matlab.ui.control.NumericEditField
        RangeDescendLabel               matlab.ui.control.Label
        DisplayRangeDescend             matlab.ui.control.NumericEditField
        MotorEfficiencyTab              matlab.ui.container.Tab
        UIAxesEfficiency                matlab.ui.control.UIAxes
        TakeOffLabel                    matlab.ui.control.Label
        TakeOffEfficiency               matlab.ui.control.NumericEditField
        DescendLabel                    matlab.ui.control.Label
        DescendEfficiency               matlab.ui.control.NumericEditField
        ClimbLabel                      matlab.ui.control.Label
        ClimbEfficiency                 matlab.ui.control.NumericEditField
        CruiseLabel                     matlab.ui.control.Label
        CruiseEfficiency                matlab.ui.control.NumericEditField
        OveralEfficiencyLabel           matlab.ui.control.Label
        OveralEfficiency                matlab.ui.control.NumericEditField
        SweepButton                     matlab.ui.control.Button
        TestDropDownLabel               matlab.ui.control.Label
        TestDropDown                    matlab.ui.control.DropDown
        BatteryProfileTab               matlab.ui.container.Tab
        BatteryRequiremensPanel         matlab.ui.container.Panel
        EnergyforTakeoffandClimbEditFieldLabel  matlab.ui.control.Label
        EnergyforTakeoffandClimb        matlab.ui.control.NumericEditField
        MaxPowerforTakeoffClimbEditFieldLabel  matlab.ui.control.Label
        MaxPowerTakeoffOrClimb          matlab.ui.control.NumericEditField
        MaxPowerforCruiseEditField_2Label  matlab.ui.control.Label
        MaxPowerCruise                  matlab.ui.control.NumericEditField
        EnergyforCruiseandDescendEditFieldLabel  matlab.ui.control.Label
        EnergyforCruiseandDescend       matlab.ui.control.NumericEditField
        SafetyExtraEditFieldLabel       matlab.ui.control.Label
        SafetyEnToCl                    matlab.ui.control.NumericEditField
        SafetyExtraEditField_2Label     matlab.ui.control.Label
        SafetyEnCrDc                    matlab.ui.control.NumericEditField
        SafetyExtraEditField_3Label     matlab.ui.control.Label
        SafetyPoToCl                    matlab.ui.control.NumericEditField
        MaxPossiblePowerEditFieldLabel  matlab.ui.control.Label
        MaxPossiblePower                matlab.ui.control.NumericEditField
        SafetyFactorinLabel             matlab.ui.control.Label
        SafetyExtraEditField_4Label     matlab.ui.control.Label
        SafetyPoCr                      matlab.ui.control.NumericEditField
        DoDLabel                        matlab.ui.control.Label
        DischargeDepth                  matlab.ui.control.NumericEditField
        CalculateBatteryButton          matlab.ui.control.Button
        PowerDensePackAuxPanel          matlab.ui.container.Panel
        BatteryNameEditFieldLabel       matlab.ui.control.Label
        AuxBatteryName                  matlab.ui.control.EditField
        CapacityEditFieldLabel          matlab.ui.control.Label
        AuxCapacity                     matlab.ui.control.NumericEditField
        ConstantEditFieldLabel          matlab.ui.control.Label
        AuxConstantPower                matlab.ui.control.NumericEditField
        BurstEditFieldLabel             matlab.ui.control.Label
        AuxBurstPower                   matlab.ui.control.NumericEditField
        WeightLabel_2                   matlab.ui.control.Label
        AuxWeight                       matlab.ui.control.NumericEditField
        AddListButton                   matlab.ui.control.Button
        EnergyDensePackMainPanel_2      matlab.ui.container.Panel
        BatteryNameEditField_2Label     matlab.ui.control.Label
        MainBatteryName                 matlab.ui.control.EditField
        CapacityEditField_2Label        matlab.ui.control.Label
        MainCapacity                    matlab.ui.control.NumericEditField
        ConstantEditField_2Label        matlab.ui.control.Label
        MainConstantPower               matlab.ui.control.NumericEditField
        BurstEditField_2Label           matlab.ui.control.Label
        MainBurstPower                  matlab.ui.control.NumericEditField
        WeightLabel                     matlab.ui.control.Label
        MainWeight                      matlab.ui.control.NumericEditField
        ResultsPanel                    matlab.ui.container.Panel
        BatterytoAicraftWeightRatioGaugeLabel  matlab.ui.control.Label
        BatteryWeightRatio              matlab.ui.control.NinetyDegreeGauge
        GoodLampLabel                   matlab.ui.control.Label
        GoodLamp                        matlab.ui.control.Lamp
        OkayLampLabel                   matlab.ui.control.Label
        OkayLamp                        matlab.ui.control.Lamp
        BadLampLabel                    matlab.ui.control.Label
        BadLamp                         matlab.ui.control.Lamp
        HybridSelected                  matlab.ui.control.StateButton
        BatteryListLabel                matlab.ui.control.Label
        RefPackPanel                    matlab.ui.container.Panel
        BatteryNameEditField_2Label_2   matlab.ui.control.Label
        RefBatteryName                  matlab.ui.control.EditField
        CapacityEditField_2Label_2      matlab.ui.control.Label
        RefCapacity                     matlab.ui.control.NumericEditField
        ConstantEditField_2Label_2      matlab.ui.control.Label
        RefConstantPower                matlab.ui.control.NumericEditField
        BurstEditField_2Label_2         matlab.ui.control.Label
        RefBurstPower                   matlab.ui.control.NumericEditField
        WeightLabel_3                   matlab.ui.control.Label
        RefWeight                       matlab.ui.control.NumericEditField
        BatteryWeightComparaisonPanel   matlab.ui.container.Panel
        HybridLabel                     matlab.ui.control.Label
        HybridWeight                    matlab.ui.control.NumericEditField
        SingleLabel                     matlab.ui.control.Label
        RefWeight2                      matlab.ui.control.NumericEditField
        DifferenceLabel                 matlab.ui.control.Label
        DifferenceWeight                matlab.ui.control.NumericEditField
        HybridLabel_2                   matlab.ui.control.Label
        HybridCapacity                  matlab.ui.control.NumericEditField
        SingleLabel_2                   matlab.ui.control.Label
        RefCapacity2                    matlab.ui.control.NumericEditField
        DifferenceLabel_2               matlab.ui.control.Label
        DifferenceCapacity              matlab.ui.control.NumericEditField
        HybridLabel_3                   matlab.ui.control.Label
        HybridEDensity                  matlab.ui.control.NumericEditField
        SingleLabel_3                   matlab.ui.control.Label
        RefEDensity                     matlab.ui.control.NumericEditField
        DifferenceLabel_3               matlab.ui.control.Label
        DifferenceEDensity              matlab.ui.control.NumericEditField
    end


    methods (Access = private)

        % Function to calculate Air Density relative to the Altitude
        function AirDensity = BaroCalc(~, Altitude)

            % Based on Nasa Earth Atmosphere Model in Metric Units
            if Altitude < 11000 % Within Troposphere
                Temp = 15.04 - 0.00649*Altitude;
                Pressure = 101.29*((Temp+273.1)/288.08)^5.256;

            elseif Altitude < 25000 % Lower Stratosphere
                Temp = -56.46;
                Pressure = 22.65*exp(1.73-0.000157*Altitude);

            else % Upper Stratosphere
                Temp = -121.21 + 0.00299*Altitude;
                Pressure = 2.488 *((Temp+273.1)/216.6)^-11.388;
            end

            AirDensity = Pressure/(0.2869 * (Temp+273.1));
        end

        % Function to calculate the Drag Force relative to input factors
        function ForceDrag = FDragCalc(app, Altitude, Velocity, CD, Aera)
            AirDensity = BaroCalc(app, Altitude);
            ForceDrag = 0.5*AirDensity*Velocity^2*CD*Aera;
        end

        function Power =
PowerMainCalc(app,ThrustRequired,Altitude,Velocity,phase)

            % Propeller Aera
            PropAera = pi*(app.PropellerDiameter.Value/2)^2;

            % Efficiency adjustment at altitude
            AirDensityGround = 1.2266;
            AirDensity = BaroCalc(app,Altitude);
            CorrectionFactorAltitude = AirDensity/AirDensityGround;

            % Efficiency altitude adjustment of motor
            CorrectionFactorMotor = 1/app.MainEfficiency.Value;
```

```
            % Efficiency air speed intake adjustment of motor
            CorrectionForce =
0.5*AirDensity*PropAera*Velocity^2*0.10197*0.25;

            % Power Estimation
            if app.ConfigurationDropDown.Value == "VTOL" && phase == 1
                y1 = app.MidThrottlePowerAux.Value;
                y2 = app.FullThrottlePowerAux.Value;
                x1 = app.MidThrottleThrustAux.Value*CorrectionFactorAltitude;
                x2 =
app.FullThrottleThrustAux.Value*CorrectionFactorAltitude;
                x=[x1;x2];
                y=[y1;y2];
                PowerFit = fit(x,y,'poly1'); %Return g/w
                MotorNum = app.NumAux.Value;
            else
                y1 = app.MidThrottlePowerMain.Value;
                y2 = app.FullThrottlePowerMain.Value;
                x1 =
app.MidThrottleThrustMain.Value*CorrectionFactorAltitude;
                x2 =
app.FullThrottleThrustMain.Value*CorrectionFactorAltitude;
                x=[x1;x2];
                y=[y1;y2];
                PowerFit = fit(x,y,'poly1'); %Return g/w
                MotorNum = app.NumMain.Value;
            end

            % Conversion N to kg
            ThrustRequired = ThrustRequired*0.10197+CorrectionForce*MotorNum;

            % Power Calculation
            Efficiency = PowerFit(ThrustRequired/MotorNum); % Find efficiency
based on motors
            Power = CorrectionFactorMotor*ThrustRequired*1000/Efficiency; %
Conversion kg to g
        end
    end


    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: ConfigurationDropDown
        function ConfigurationDropDownValueChanged(app, event)
            value = app.ConfigurationDropDown.Value;
            if isequal(value, 'Fixed Wing')
                app.CdTakeOff.Value = 0.04;
                app.CdCruise.Value = 0.025;
                app.CdDescend.Value = 0.025;
                app.CdClimb.Value = 0.035;
                app.SurfaceAeraCruise.Value = 30;
                app.AccelerationTakeOff.Value = 0.5;
                app.TOW.Value = 6000;
                app.ClimbRate.Value = 4;
                app.DescentRate.Value = 3;
                app.Speed.Value = 480;
                app.Altitude.Value = 3000;
                app.Range.Value = 1000;
                app.StallSpeed.Value = 100;
                app.PropellerDiameter.Value = 1.65;
                app.MidThrottlePowerMain.Value = 12;
                app.FullThrottlePowerMain.Value = 4;
                app.MidThrottleThrustMain.Value = 400;
                app.FullThrottleThrustMain.Value = 1200;
                app.NumMain.Value = 3;
                app.MainEfficiency.Value = 0.94;
                app.MainEfficiency.Value = 0.94;
                app.MidThrottlePowerAux.Value = 0;
                app.FullThrottlePowerAux.Value = 0;
                app.MidThrottleThrustAux.Value = 0;
                app.FullThrottleThrustAux.Value = 0;
                app.NumAux.Value = 0;
                app.AuxEfficiency.Value = 0;
            elseif isequal(value, 'Multicopter')
                app.CdTakeOff.Value = 0.3;
                app.CdCruise.Value = 0.3;
                app.CdDescend.Value = 0.3;
                app.CdClimb.Value = 0.3;
                app.SurfaceAeraCruise.Value = 1.47;
                app.SurfaceAeraTakeOff.Value =2.11;
                app.AccelerationTakeOff.Value = 0.5;
                app.TOW.Value = 325;
                app.ClimbRate.Value = 3;
                app.DescentRate.Value = 3;
                app.CdTrans.Value = 0.3;
                app.Speed.Value = 100;
                app.Altitude.Value = 500;
                app.Range.Value = 30;
                app.StallSpeed.Value = 0;
                app.PropellerDiameter.Value = 1.57;
                app.MidThrottlePowerMain.Value = 7.5;
                app.FullThrottlePowerMain.Value = 4.5;
                app.MidThrottleThrustMain.Value = 36.5;
                app.FullThrottleThrustMain.Value = 102.4;
                app.NumMain.Value = 8;
                app.MainEfficiency.Value = 0.94;
                app.MidThrottlePowerAux.Value = 0;
                app.FullThrottlePowerAux.Value = 0;
                app.MidThrottleThrustAux.Value = 0;
                app.FullThrottleThrustAux.Value = 0;
                app.NumAux.Value = 0;
                app.AuxEfficiency.Value = 0;
            elseif isequal(value, 'VTOL')
                app.CdTakeOff.Value = 0.04;
                app.CdCruise.Value = 0.03;
                app.CdDescend.Value = 0.025;
                app.CdClimb.Value = 0.035;
                app.SurfaceAeraCruise.Value = 1;
                app.SurfaceAeraTakeOff.Value = 1.5;
                app.AccelerationTakeOff.Value = 0.5;
                app.TOW.Value = 6.9;
                app.ClimbRate.Value = 6;
                app.DescentRate.Value = 6;
                app.CdTrans.Value = 0.3;
                app.Speed.Value = 70;
                app.Altitude.Value = 20;
                app.Range.Value = 40;
                app.StallSpeed.Value = 50;
                app.PropellerDiameter.Value = 0.33;
                app.MidThrottlePowerMain.Value = 7.3;
                app.FullThrottlePowerMain.Value = 4.69;
                app.MidThrottleThrustMain.Value = 1.669;
                app.FullThrottleThrustMain.Value =4.235;
                app.NumMain.Value = 2;
                app.MainEfficiency.Value = 0.94;
                app.MidThrottlePowerAux.Value = 7.3;
                app.FullThrottlePowerAux.Value = 4.69;
                app.MidThrottleThrustAux.Value = 1.669;
                app.FullThrottleThrustAux.Value =4.235;
                app.NumAux.Value = 4;
                app.AuxEfficiency.Value = 0.94;
            else
                app.CdTakeOff.Value = 0;
                app.CdCruise.Value = 0;
                app.CdDescend.Value = 0;
                app.CdClimb.Value = 0;
                app.SurfaceAeraCruise.Value = 0;
                app.AccelerationTakeOff.Value = 0;
                app.TOW.Value = 0;
                app.ClimbRate.Value = 0;
                app.DescentRate.Value = 0;
                app.CdTrans.Value = 0;
                app.Speed.Value = 0;
                app.Altitude.Value = 0;
                app.Range.Value = 0;
                app.StallSpeed.Value = 0;
                app.PropellerDiameter.Value = 0;
                app.MidThrottlePowerMain.Value = 0;
                app.FullThrottlePowerMain.Value = 0;
                app.MidThrottleThrustMain.Value = 0;
                app.FullThrottleThrustMain.Value = 0;
                app.NumMain.Value = 0;
                app.MainEfficiency.Value = 0;
            end
        end

        % Button pushed function: CalculateButton
        function CalculateButtonPushed(app, event)
            % Clear previous Data
            app.EnergyTakeOff.Value = 0;
            app.EnergyClimb.Value = 0;
            app.EnergyCruise.Value = 0;
            app.EnergyDescend.Value = 0;
            app.PowerTakeOff.Value = 0;
            app.PowerClimb.Value = 0;
            app.PowerCruise.Value = 0;
            app.PowerDescend.Value = 0;
            app.TotalEnergy.Value = 0;

            % Data Struct Construct
            ArraySize = 500000;
            Data.Time = linspace(0,ArraySize,ArraySize+1);  % in s
            Data.Altitude = zeros(1,ArraySize+1);           % in m
            Data.Range = zeros(1,ArraySize+1);
            Data.Speed = zeros(1,ArraySize+1);              % in m/s
            Data.ForceDrag = zeros(1,ArraySize+1);          % in Newtons
            Data.ForceTotal = zeros(1,ArraySize+1);         % in Newtons
            Data.Power = zeros(1,ArraySize+1);              % in W
            Data.Efficiency = zeros(1,ArraySize+1);         % in g/W
            Data.Phase = zeros(1,ArraySize+1);              % Defined unit
see below

            % Phase = 1, Take Off
            % Phase = 2, Climb
            % Phase = 3, Cruise
            % Phase = 4, Descend
            % Phase = 5, Landing
            % Phase = 0, Not Assigned

            % Initialize variables
            CurrentSpeed = 0;
            CurrentAltitude = 0;
            TransitionAltitude =0;
            CurrentRange = 0;
            Step = 1;
% Take Off -
            if app.ConfigurationDropDown.Value =="Fixed Wing"
                % Calculate TakeOff
                CD = app.CdTakeOff.Value;
                Aera = app.SurfaceAeraCruise.Value;
                while CurrentSpeed < (app.StallSpeed.Value*1.1/3.6) %
Conversion from km/h to m/s times factor 1.1
                    Data.Phase(1,Step) = 1;
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
CurrentSpeed, CD, Aera);
                    Data.ForceTotal(1,Step) = Data.ForceDrag(1,Step) +
app.TOW.Value*app.AccelerationTakeOff.Value*9.81;
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step), CurrentAltitude,CurrentSpeed,Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed = app.AccelerationTakeOff.Value * 9.81 *
Data.Time(1,Step); % Conversion g to m/s2
                    Data.Speed(1,Step) = CurrentSpeed;
                    Step = Step+1;
                    if Step>ArraySize+1
                        break
                    end
                end
                app.PowerTakeOff.Value = Data.Power(1,Step-1)/1000;
                TimeTakeOff = Data.Time(1,Step);
                MarkerTakeOff = Step;
            end

            if app.ConfigurationDropDown.Value =="Multicopter" ||
app.ConfigurationDropDown.Value =="VTOL"
                % Calculate TakeOff
                CD = app.CdTakeOff.Value;
                Aera = app.SurfaceAeraTakeOff.Value;
                CurrentAltitude = 0;
                CurrentSpeed_V = 0;
                while CurrentSpeed_V < (app.ClimbRate.Value)
                    Data.Phase(1,Step) = 1;
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
CurrentSpeed_V, CD, Aera);
                    Data.ForceTotal(1,Step) = Data.ForceDrag(1,Step) +
(app.TOW.Value*(app.AccelerationTakeOff.Value+1))*9.81;
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step), CurrentAltitude, CurrentSpeed_V,Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed_V = app.AccelerationTakeOff.Value * 9.81 *
Data.Time(1,Step); % Conversion g to m/s2
                    CurrentAltitude = CurrentAltitude + CurrentSpeed_V;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    Step = Step+1;
                    if Step>ArraySize+1
                        break
                    end
                end
                app.PowerTakeOff.Value = Data.Power(1,Step-1)/1000;
                TimeTakeOff = Data.Time(1,Step);
                MarkerTakeOff = Step;
                TransitionAltitude = CurrentAltitude;
            end
% Transition
            if app.ConfigurationDropDown.Value =="VTOL"
                % Calculate TakeOff
                CD = app.CdTrans.Value;
                Aera = app.SurfaceAeraCruise.Value;
                while CurrentSpeed < (app.StallSpeed.Value*1.1/3.6) %
Conversion from km/h to m/s times factor 1.1
                    Data.Phase(1,Step) = -1;
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
CurrentSpeed, CD, Aera);
                    Data.ForceTotal(1,Step) = Data.ForceDrag(1,Step) +
app.TOW.Value*app.AccelerationTakeOff.Value*9.81;
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step), CurrentAltitude,CurrentSpeed,Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.91397/Data.Power(1,Step);
                    CurrentSpeed = app.AccelerationTakeOff.Value * 9.81 *
Data.Time(1,Step); % Conversion g to m/s2
                    Data.Speed(1,Step) = CurrentSpeed;
                    Step = Step+1;
                    if Step>ArraySize+1
                        break
                    end
                end
                TimeTakeOff = Data.Time(1,Step);
                MarkerTakeOff = Step;
            end
% Climb ----
            if app.ConfigurationDropDown.Value =="Fixed Wing" ||
app.ConfigurationDropDown.Value =="VTOL"
                % Calculate Climb
                CD = app.CdClimb.Value;
                TimeClimb = (app.Altitude.Value-
TransitionAltitude)/app.ClimbRate.Value; % in s
                ClimbAcceleration = (app.Speed.Value/3.6 -
CurrentSpeed)/TimeClimb; % in m/s2
                while CurrentAltitude < app.Altitude.Value
                    Data.Phase(1,Step) = 2;
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
CurrentSpeed, CD, Aera);
                    Data.ForceTotal(1,Step) = Data.ForceDrag(1,Step) +
app.TOW.Value*ClimbAcceleration;
```

```
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step),CurrentAltitude,CurrentSpeed,Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed = CurrentSpeed+ClimbAcceleration*1; % add
1sec of acceleration each loop
                    Data.Speed(1,Step) = CurrentSpeed;
                    CurrentAltitude = CurrentAltitude + app.ClimbRate.Value;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    CurrentRange = CurrentRange + sqrt(CurrentSpeed^2-
app.ClimbRate.Value^2);
                    Data.Range(1,Step) = CurrentRange;
                    Step = Step+1;
                    if Step>ArraySize
                        break
                    end
                end
                app.PowerClimb.Value = Data.Power(1,Step-1)/1000;
                RangeClimb = Data.Range(1,Step-1);
                MarkerClimb = Step; % Marker to place data array later
            end

            if app.ConfigurationDropDown.Value =="Multicopter"
                % Calculate Climb
                CD = app.CdClimb.Value;
                Aera = app.SurfaceAeraTakeOff.Value;
                TimeClimb = app.Altitude.Value/app.ClimbRate.Value; % in s
                ClimbAcceleration = (app.Speed.Value/3.6)/TimeClimb; % in
m/s2, horizontal
                while CurrentAltitude < app.Altitude.Value
                    Data.Phase(1,Step) = 2;
                    RelativeSpeed = sqrt(app.ClimbRate.Value^2 +
CurrentSpeed^2);
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
RelativeSpeed, CD, Aera);
                    DragAngle = atand(app.ClimbRate.Value/CurrentSpeed);
                    ForceDrag_x = cosd(DragAngle)*Data.ForceDrag(1,Step);
                    ForceDrag_y = sind(DragAngle)*Data.ForceDrag(1,Step);
                    ForceTotal_x = ForceDrag_x +
ClimbAcceleration*9.81*app.TOW.Value;
                    ForceTotal_y = app.TOW.Value*9.81 + ForceDrag_y;
                    TiltAngle = atand(ForceDrag_y/ForceDrag_x);
                    Data.ForceTotal(1,Step) =
sqrt(ForceTotal_x^2+ForceTotal_y^2);
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step),CurrentAltitude,...
                        (sind(TiltAngle)*CurrentSpeed),Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed = CurrentSpeed+ClimbAcceleration*1; % add
1sec of acceleration each loop
                    Data.Speed(1,Step) = CurrentSpeed;
                    CurrentAltitude = CurrentAltitude + app.ClimbRate.Value;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    CurrentRange = CurrentRange + CurrentSpeed;
                    Data.Range(1,Step) = CurrentRange;
                    Step = Step+1;
                    if Step>ArraySize
                        break
                    end
                end
                app.PowerClimb.Value = Data.Power(1,Step-1)/1000;
                RangeClimb = Data.Range(1,Step-1);
                MarkerClimb = Step; % Marker to place data array later
            end
% Range DC -
            if app.ConfigurationDropDown.Value =="Fixed Wing" ||
app.ConfigurationDropDown.Value =="VTOL"
                %Calculate Range Descend to scale cruise range approriately
                TimeDescend = floor((app.Altitude.Value-
TransitionAltitude)/app.DescentRate.Value);
                CurrentSpeed = app.Speed.Value/3.6;
                DescendAcceleration = (app.StallSpeed.Value*1.3/3.6 -
app.Speed.Value/3.6)/TimeDescend;
                RangeDescend = 0;
                for n = 0:TimeDescend
                    if CurrentSpeed>=app.DescentRate.Value
                        RangeDescend = RangeDescend + sqrt(CurrentSpeed^2-
app.DescentRate.Value^2);
                        CurrentSpeed = app.Speed.Value/3.6 +
DescendAcceleration * n;
                    end
                end
            end

            if app.ConfigurationDropDown.Value =="Multicopter"
                %Calculate Range Descend to scale cruise range approriately
                TimeDescend =
floor(app.Altitude.Value/app.DescentRate.Value);
                CurrentSpeed = app.Speed.Value/3.6;
                DescendAcceleration = (-app.Speed.Value/3.6)/TimeDescend;
                RangeDescend = 0;
                for n = 0:TimeDescend
                    if CurrentSpeed>=app.DescentRate.Value
                        RangeDescend = RangeDescend + sqrt(CurrentSpeed^2-
app.DescentRate.Value^2);
                        CurrentSpeed = app.Speed.Value/3.6 +
DescendAcceleration * n;
                    end
                end
            end
% Cruise ---
            if app.ConfigurationDropDown.Value =="Fixed Wing" ||
app.ConfigurationDropDown.Value =="VTOL"
                % Calculate for Cruise
                CD = app.CdCruise.Value;
                RangeCruise = app.Range.Value*1000 - RangeClimb -
RangeDescend;
                if RangeCruise < 0
                    TimeCruise = 0;
                else
                    TimeCruise = RangeCruise/(app.Speed.Value/3.6);
                end
                CurrentAltitude = app.Altitude.Value;
                CurrentSpeed = app.Speed.Value/3.6;
                CurrentForceDrag =
FDragCalc(app,CurrentAltitude,CurrentSpeed,CD,Aera);
                CurrentPower =
PowerMainCalc(app,CurrentForceDrag,CurrentAltitude,CurrentSpeed,Data.Phase(1,
Step));
                app.PowerCruise.Value = CurrentPower/1000;
                CurrentEfficiency = CurrentForceDrag*0.10197/CurrentPower;
                for Step = MarkerClimb:(TimeCruise+MarkerClimb)
                    Data.Phase(1,Step) = 3;
                    Data.ForceDrag(1,Step) = CurrentForceDrag;
                    Data.ForceTotal(1,Step) = CurrentForceDrag;
                    Data.Power(1,Step) = CurrentPower;
                    Data.Efficiency(1,Step) = CurrentEfficiency;
                    Data.Speed(1,Step) = CurrentSpeed;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    Data.Range(1,Step) =
Data.Range(1,MarkerClimb)+CurrentSpeed*(n-MarkerClimb);
                end
                MarkerCruise = Step;
            end

            if app.ConfigurationDropDown.Value =="Multicopter"
                % Calculate for Cruise
                CD = app.CdCruise.Value;
                Aera = app.SurfaceAeraCruise.Value;
                RangeCruise = app.Range.Value*1000 - RangeClimb -
RangeDescend;
                if RangeCruise < 0
                    TimeCruise = 0;
                else
                    TimeCruise = RangeCruise/(app.Speed.Value/3.6);
                end
                CurrentAltitude = app.Altitude.Value;
                CurrentSpeed = app.Speed.Value/3.6;
                RelativeSpeed = sqrt(app.DescentRate.Value^2 +
CurrentSpeed^2);
                CurrentForceDrag = FDragCalc(app, CurrentAltitude,
RelativeSpeed, CD, Aera);
                DragAngle = atand(app.ClimbRate.Value/CurrentSpeed);
                ForceDrag_x = cosd(DragAngle)*CurrentForceDrag;
                ForceDrag_y = sind(DragAngle)*CurrentForceDrag;
                ForceTotal_x = ForceDrag_x;
                ForceTotal_y = app.TOW.Value*9.81 + ForceDrag_y;
                TiltAngle = atand(ForceDrag_y/ForceDrag_x);
                CurrentForceTotal = sqrt(ForceTotal_x^2+ForceTotal_y^2);
                CurrentPower =
PowerMainCalc(app,CurrentForceTotal,CurrentAltitude,...
                    (sind(TiltAngle)*CurrentSpeed),Data.Phase(1,Step));
                app.PowerCruise.Value = CurrentPower/1000;
                CurrentEfficiency = CurrentForceTotal*CurrentPower;
                for Step = MarkerClimb:(TimeCruise+MarkerClimb)
                    Data.Phase(1,Step) = 3;
                    Data.ForceDrag(1,Step) = CurrentForceDrag;
                    Data.ForceTotal(1,Step) = CurrentForceTotal;
                    Data.Power(1,Step) = CurrentPower;
                    Data.Efficiency(1,Step) = CurrentEfficiency;
                    Data.Speed(1,Step) = CurrentSpeed;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    Data.Range(1,Step) =
Data.Range(1,MarkerClimb)+CurrentSpeed*(n-MarkerClimb);
                end
                MarkerCruise = Step;
            end
% Descend---
            % Calculate Descend
            if app.ConfigurationDropDown.Value =="Fixed Wing"
                CD = app.CdDescend.Value;
                CurrentSpeed = app.Speed.Value/3.6;
                CurrentRange = RangeClimb + RangeCruise;
                CurrentAltitude = app.Altitude.Value;
                while CurrentAltitude > 0
                    Data.Phase(1,Step) = 4;
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
CurrentSpeed, CD, Aera);
                    Data.ForceTotal(1,Step) = Data.ForceDrag(1,Step) +
app.TOW.Value*DescendAcceleration;
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step),...
                        CurrentAltitude,CurrentSpeed,Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed = CurrentSpeed + DescendAcceleration*1; %
add 1sec of Desceleration each loop
                    Data.Speed(1,Step) = CurrentSpeed;
                    CurrentAltitude = CurrentAltitude -
app.DescentRate.Value;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    CurrentRange = CurrentRange + sqrt(CurrentSpeed^2-
app.DescentRate.Value^2);
                    Data.Range(1,Step) = CurrentRange;
                    if Step>ArraySize
                        break
                    end
                    Step = Step+1;
                end
                app.PowerDescend.Value = Data.Power(1,MarkerCruise+1)/1000;
            end

            % Calculate Descend
            if app.ConfigurationDropDown.Value =="Multicopter"
                CD = app.CdDescend.Value;
                Aera = app.SurfaceAeraTakeOff.Value;
                CurrentSpeed = app.Speed.Value/3.6;
                CurrentRange = RangeClimb + RangeCruise;
                CurrentAltitude = app.Altitude.Value;
                while CurrentAltitude > 0
                    Data.Phase(1,Step) = 4;
                    RelativeSpeed = sqrt(app.DescentRate.Value^2 +
CurrentSpeed^2);
                    Data.ForceDrag(1,Step) = FDragCalc(app, CurrentAltitude,
RelativeSpeed, CD, Aera);
                    DragAngle = atand(app.ClimbRate.Value/CurrentSpeed);
                    ForceDrag_x = cosd(DragAngle)*Data.ForceDrag(1,Step);
                    ForceDrag_y = sind(DragAngle)*Data.ForceDrag(1,Step);
                    ForceTotal_x = ForceDrag_x -
ClimbAcceleration*9.81*app.TOW.Value;
                    ForceTotal_y = app.TOW.Value*9.81 - ForceDrag_y;
                    TiltAngle = atand(ForceDrag_y/ForceDrag_x);
                    Data.ForceTotal(1,Step) =
sqrt(ForceTotal_x^2+ForceTotal_y^2);
                    Data.Power(1,Step) = PowerMainCalc(app,
Data.ForceTotal(1,Step),CurrentAltitude,...
                        (sind(TiltAngle)*CurrentSpeed),Data.Phase(1,Step));
                    Data.Efficiency(1,Step) =
Data.ForceTotal(1,Step)*0.10197/Data.Power(1,Step);
                    CurrentSpeed = CurrentSpeed + DescendAcceleration*1; %
add 1sec of Desceleration each loop
                    Data.Speed(1,Step) = CurrentSpeed;
                    CurrentAltitude = CurrentAltitude -
app.DescentRate.Value;
                    Data.Altitude(1,Step) = CurrentAltitude;
                    CurrentRange = CurrentRange + CurrentSpeed;
                    Data.Range(1,Step) = CurrentRange;
                    if Step>ArraySize
                        break
                    end
                    Step = Step+1;
                end
                app.PowerDescend.Value = Data.Power(1,MarkerCruise+1)/1000;
            end
%----------
            % Resize Array
            Data.Time = Data.Time(1:Step);
            Data.Altitude = Data.Altitude(1:Step);
            Data.Range = Data.Range(1:Step);
            Data.Speed = Data.Speed(1:Step);
            Data.ForceDrag = Data.ForceDrag(1:Step);
            Data.ForceTotal = Data.ForceTotal(1:Step);
            Data.Power = Data.Power(1:Step);
            Data.Efficiency = Data.Efficiency(1:Step);
            Data.Phase = Data.Phase(1:Step);

            % Graph Output and data display ----------------------------------
------
            plot(app.UIAxesPowerProfile,Data.Power(1,:));
            plot(app.UIAxesDragProfile,Data.ForceDrag(1,:));
            plot(app.UIAxesSpeedProfile,Data.Speed(1,:));
            plot(app.UIAxesEfficiency,Data.Efficiency(1,:));

            % Power Profile Tab
            app.EnergyTakeOff.Value = sum(Data.Power(1,1:MarkerTakeOff-
1))/(1000*60*60);
            app.EnergyClimb.Value =
sum(Data.Power(1,MarkerTakeOff:MarkerClimb-1))/(1000*60*60);
            app.EnergyCruise.Value =
sum(Data.Power(1,MarkerClimb:MarkerCruise-1))/(1000*60*60);
            app.EnergyDescend.Value =
sum(Data.Power(1,MarkerCruise:end))/(1000*60*60);
            app.TotalEnergy.Value = sum(Data.Power(1,:))/(1000*60*60);

            % Drag Profile Tab
            app.DragTakeOff.Value = Data.ForceDrag(1,MarkerTakeOff-1)/1000;
            app.DragClimb.Value = Data.ForceDrag(1,MarkerClimb-1)/1000;
            app.DragCruise.Value = Data.ForceDrag(1,MarkerCruise-1)/1000;
            app.DragDescend.Value = Data.ForceDrag(1,MarkerCruise)/1000;
            app.ForceAcceleration.Value =
app.TOW.Value*app.AccelerationTakeOff.Value*9.81/1000;

            % Speed Profile Tab
            app.DisplayTimeTakeOff.Value = TimeTakeOff;
            app.DisplayTimeClimb.Value = TimeClimb;
            app.DisplayTimeCruise.Value = TimeCruise;
            app.DisplayTimeDescend.Value =TimeDescend;
```

```
                app.DisplayRangeClimb.Value = RangeClimb/1000;
                app.DisplayRangeCruise.Value = RangeCruise/1000;
                app.DisplayRangeDescend.Value = RangeDescend/1000;
                app.DisplayTotalTime.Value = (Step-1)/60;

                % Motor Efficiency Tab
                %app.TakeOffEfficiency.Value = Data.Efficiency(1,MarkerTakeOff-
1);
                %app.ClimbEfficiency.Value = Data.Efficiency(1,MarkerClimb-1);
                %app.CruiseEfficiency.Value = Data.Efficiency(1,MarkerCruise-1);
                %app.DescendEfficiency.Value = Data.Efficiency(1,MarkerCruise);

                % Data Transfer to Battery Profile
                app.EnergyforTakeoffandClimb.Value = app.EnergyTakeOff.Value +
app.EnergyClimb.Value;
                app.EnergyforCruiseandDescend.Value = app.EnergyCruise.Value +
app.EnergyDescend.Value;
                app.MaxPowerTakeoffOrClimb.Value = max(app.PowerClimb.Value,
app.PowerTakeOff.Value);
                app.MaxPowerCruise.Value = app.PowerCruise.Value;
                app.MaxPossiblePower.Value  =
(app.FullThrottleThrustMain.Value/app.FullThrottlePowerMain.Value)*...
                    app.NumMain.Value*(1/app.MainEfficiency.Value);

            end

            % Button pushed function: AddListButton
            function AddListButtonPushed(app, event)
                global BatteryList
                [file, path] = uigetfile('*.xlsx');
                if isequal(file,0)
                    msgbox('Please input an Excel file')
                else
                BatteryList = readtable(fullfile(path, file));
                end
            end

            % Button pushed function: CalculateBatteryButton
            function CalculateBatteryButtonPushed(app, event)
                global BatteryList

                % Battery Requirements correction for DOD and safety
                app.EnergyforTakeoffandClimb.Value =
app.EnergyforTakeoffandClimb.Value * 100/(app.DischargeDepth.Value);
                app.EnergyforTakeoffandClimb.Value =
app.EnergyforTakeoffandClimb.Value * 100/(100-app.SafetyEnToCl.Value);
                app.EnergyforCruiseandDescend.Value =
app.EnergyforCruiseandDescend.Value * 100/app.DischargeDepth.Value;
                app.EnergyforCruiseandDescend.Value =
app.EnergyforCruiseandDescend.Value * 100/(100-app.SafetyEnCrDc.Value);
                app.MaxPowerTakeoffOrClimb.Value =
app.MaxPowerTakeoffOrClimb.Value * 100/(100-app.SafetyPoToCl.Value);
                app.MaxPowerCruise.Value = app.MaxPowerCruise.Value * 100/(100-
app.SafetyPoCr.Value);

                % Data Struct Construct
                ArraySize = size(BatteryList,1);
                BatteryOutputMain = zeros(1,ArraySize); % Weight in kg
                BatteryOutputAux  = zeros(1,ArraySize); % Weight in kg
                BatteryOutputRef  = zeros(1,ArraySize); % Weight in kg

                % Calculate weight of Main Battery
                for n = 1:ArraySize
                    BatteryOutputMain(1,n) =
max(app.EnergyforCruiseandDescend.Value*1000/BatteryList{n,7},...
                        app.MaxPowerCruise.Value*1000/BatteryList{n,8});
                end

                % Calculate weight of Aux Battery
                for n = 1:ArraySize
                    BatteryOutputAux(1,n) =
max(app.EnergyforTakeoffandClimb.Value*1000/BatteryList{n,7},...
                        app.MaxPowerTakeoffOrClimb.Value*1000/BatteryList{n,8});
                end

                % Calculate weight of Aux Battery
                for n = 1:ArraySize
                    EnergyRef =
app.EnergyforTakeoffandClimb.Value+app.EnergyforCruiseandDescend.Value;
                    PowerRef =
max(app.MaxPowerCruise.Value,app.MaxPowerTakeoffOrClimb.Value);
                    BatteryOutputRef(1,n) =
max(EnergyRef*1000/BatteryList{n,7},PowerRef*1000/BatteryList{n,8});
                end

                % Display for Main
                MWeight = min(BatteryOutputMain);
                BestMain = BatteryOutputMain == MWeight;
                app.MainBatteryName.Value = char(BatteryList{BestMain,1});
                app.MainCapacity.Value = MWeight*BatteryList{BestMain,7}/1000;
                app.MainWeight.Value = min(BatteryOutputMain);
                app.MainConstantPower.Value =
MWeight*BatteryList{BestMain,8}/1000;
                app.MainBurstPower.Value = MWeight*BatteryList{BestMain,9}/1000;

                % Display for Aux
                XWeight = min(BatteryOutputAux);
                BestAux = BatteryOutputAux == XWeight;
                app.AuxBatteryName.Value = char(BatteryList{BestAux,1});
                app.AuxCapacity.Value = XWeight*BatteryList{BestAux,7}/1000;
                app.AuxWeight.Value = min(BatteryOutputAux);
                app.AuxConstantPower.Value = XWeight*BatteryList{BestAux,8}/1000;
                app.AuxBurstPower.Value = XWeight*BatteryList{BestAux,9}/1000;

                % Display for Ref
                RWeight = min(BatteryOutputRef);
                BestAux = BatteryOutputRef == RWeight;
                app.RefBatteryName.Value = char(BatteryList{BestAux,1});
                app.RefCapacity.Value = RWeight*BatteryList{BestAux,7}/1000;
                app.RefWeight.Value = min(BatteryOutputRef);
                app.RefConstantPower.Value = RWeight*BatteryList{BestAux,8}/1000;
                app.RefBurstPower.Value = RWeight*BatteryList{BestAux,9}/1000;

                % Battery Weight Comparaison
                app.HybridWeight.Value = app.MainWeight.Value +
app.AuxWeight.Value;
                app.RefWeight2.Value = app.RefWeight.Value;
                app.DifferenceWeight.Value = app.RefWeight.Value -
app.HybridWeight.Value;
                app.HybridCapacity.Value = app.MainCapacity.Value +
app.AuxCapacity.Value;
                app.RefCapacity2.Value = app.RefCapacity.Value;
                app.DifferenceCapacity.Value = app.HybridCapacity.Value -
app.RefCapacity.Value;
                app.HybridEDensity.Value = app.HybridCapacity.Value *1000 /
app.HybridWeight.Value;
                app.RefEDensity.Value = app.RefCapacity.Value * 1000 /
app.RefWeight.Value;
                app.DifferenceEDensity.Value = app.HybridEDensity.Value -
app.RefEDensity.Value;

                if app.DifferenceWeight.Value < 0
                    app.DifferenceWeight.BackgroundColor = [0.9, 0.5, 0.5];
                else
                    app.DifferenceWeight.BackgroundColor = [1, 1, 1];
                end

                if app.DifferenceCapacity.Value < 0
                    app.DifferenceCapacity.BackgroundColor = [0.9, 0.5, 0.5];
                else
                    app.DifferenceCapacity.BackgroundColor = [1, 1, 1];
                end

                if app.DifferenceEDensity.Value < 0
                    app.DifferenceEDensity.BackgroundColor = [0.9, 0.5, 0.5];
                else
                    app.DifferenceEDensity.BackgroundColor = [1, 1, 1];
                end
            end


            % Button pushed function: SweepButton
            function SweepButtonPushed(app, event)

                %Prompt for file name
                prompt = {'Enter excel file name:'};
                dlgtitle = 'Input';
                dims = [1 35];
                definput = {'hsv'};
                filename = string(inputdlg(prompt,dlgtitle,dims,definput));

                variance = 0.75:0.05:1.25;
                CollectArray = size(variance,2);
                Collect.EnergyTO = zeros(1,CollectArray);
                Collect.EnergyCL = zeros(1,CollectArray);
                Collect.EnergyCR = zeros(1,CollectArray);
                Collect.EnergyDC = zeros(1,CollectArray);
                Collect.EnergyTotal = zeros(1,CollectArray);
                Collect.PowerTO = zeros(1,CollectArray);
                Collect.PowerCL = zeros(1,CollectArray);
                Collect.PowerCR = zeros(1,CollectArray);
                Collect.WeightHyb = zeros(1,CollectArray);
                Collect.WeightRef = zeros(1,CollectArray);
                Collect.Weightdif = zeros(1,CollectArray);
                Collect.CapaHyb = zeros(1,CollectArray);
                Collect.CapaRef = zeros(1,CollectArray);
                Collect.Capadif = zeros(1,CollectArray);
                Collect.EDensHyb = zeros(1,CollectArray);
                Collect.EDEnsRef = zeros(1,CollectArray);
                Collect.EDensdif = zeros(1,CollectArray);
                position = 1; % initialize position

                % Collect mid point
                if app.TestDropDown.Value == "Altitude"
                        test = app.Altitude.Value;
                    elseif app.TestDropDown.Value == "Drag Coefficients"
                        testTO = app.CdTakeOff.Value;
                        testCR = app.CdCruise.Value;
                        testCD = app.CdDescend.Value;
                        testCL = app.CdClimb.Value;
                        testTS = app.CdTrans.Value;
                    elseif app.TestDropDown.Value == "Surface"
                         testSCR = app.SurfaceAeraCruise.Value;
                         testSTO = app.SurfaceAeraTakeOff.Value;
                    elseif app.TestDropDown.Value == "Motor Efficiency"
                         testMPM = app.MidThrottlePowerMain.Value;
                         testMPX = app.MidThrottlePowerAux.Value;
                         testFTM = app.FullThrottlePowerMain.Value;
                         testFTX = app.FullThrottlePowerAux.Value;
                    elseif app.TestDropDown.Value == "Speed"
                         test = app.Speed.Value;
                    elseif app.TestDropDown.Value == "Range"
                         test = app.Range.Value;
                    elseif app.TestDropDown.Value == "Acceleration"
                         test = app.AccelerationTakeOff.Value;
                    elseif app.TestDropDown.Value == "TOW"
                         test = app.TOW.Value;
                end

                for factor = variance
                    if app.TestDropDown.Value == "Altitude"
                        app.Altitude.Value = test*factor;
                    elseif app.TestDropDown.Value == "Drag Coefficients"
                        app.CdTakeOff.Value = testTO*factor;
                        app.CdCruise.Value = testCR*factor;
                        app.CdDescend.Value = testCD*factor;
                        app.CdClimb.Value = testCL*factor;
                        app.CdTrans.Value = testTS*factor;
                    elseif app.TestDropDown.Value == "Surface"
                        app.SurfaceAeraCruise.Value = testSCR*factor;
                        app.SurfaceAeraTakeOff.Value = testSTO*factor;
                    elseif app.TestDropDown.Value == "Motor Efficiency"
                        app.MidThrottlePowerMain.Value = testMPM*factor;
                        app.MidThrottlePowerAux.Value = testMPX*factor;
                        app.FullThrottlePowerMain.Value = testFTM*factor;
                        app.FullThrottlePowerAux.Value = testFTX*factor;
                    elseif app.TestDropDown.Value == "Speed"
                        app.Speed.Value =  test*factor;
                    elseif app.TestDropDown.Value == "Range"
                        app.Range.Value = test*factor;
                    elseif app.TestDropDown.Value == "Acceleration"
                        app.AccelerationTakeOff.Value = test*factor;
                    elseif app.TestDropDown.Value == "TOW"
                        app.TOW.Value = test*factor;
                   end

                    CalculateButtonPushed(app, event);
                    CalculateBatteryButtonPushed(app,event);
                    Collect.EnergyTO(1,position) = app.EnergyTakeOff.Value;
                    Collect.EnergyCL(1,position) = app.EnergyClimb.Value;
                    Collect.EnergyCR(1,position) = app.EnergyCruise.Value;
                    Collect.EnergyDC(1,position) = app.EnergyDescend.Value;
                    Collect.EnergyTotal(1,position) = app.TotalEnergy.Value;
                    Collect.PowerTO(1,position) = max(app.PowerTakeOff.Value);
                    Collect.PowerCL(1,position) = max(app.PowerClimb.Value);
                    Collect.PowerCR(1,position) = app.PowerCruise.Value;
                    Collect.WeightHyb(1,position) = app.HybridWeight.Value;
                    Collect.WeightRef(1,position) = app.RefWeight.Value;
                    Collect.Weightdif(1,position) = app.DifferenceWeight.Value;
                    Collect.CapaHyb(1,position) = app.HybridCapacity.Value;
                    Collect.CapaRef(1,position) = app.RefCapacity.Value;
                    Collect.Capadif(1,position) = app.DifferenceCapacity.Value;
                    Collect.EDensHyb(1,position) = app.HybridEDensity.Value;
                    Collect.EDEnsRef(1,position) = app.RefEDensity.Value;
                    Collect.EDensdif(1,position) = app.DifferenceEDensity.Value;
                    position = position +1;
                end

                % Write File
                Xaxis = [ "Variable", 1,2,3,4,5,6,7,8,9,10,11];
                Yaxis = [ "Energy TO";"Energy CL";"Energy CR";"Energy DC";"Energy
Total";"Power TO";"Power CL";"Power CR";...
                                "Hybrid Weight"; "Reference Weight"; "Difference
Weight"; "Hybrid Capacity"; "Reference Capacity";...
                                    "Difference Capacity"; "Hybrid EDensity"; "Reference
EDensity"; "Difference EDensity"];
                    Content =
[Collect.EnergyTO;Collect.EnergyCL;Collect.EnergyCR;Collect.EnergyDC;Collect.
EnergyTotal;...
                    Collect.PowerTO;Collect.PowerCL;Collect.PowerCR;...
                    Collect.WeightHyb; Collect.WeightRef; Collect.Weightdif;
Collect.CapaHyb; Collect.CapaRef; Collect.Capadif;...
                    Collect.EDensHyb;Collect.EDEnsRef; Collect.EDensdif];
                writematrix(Xaxis,filename,'Sheet',1,'Range','A1');
                writematrix(Yaxis,filename,'Sheet',1,'Range','A2');
                writematrix(Content,filename,'Sheet',1,'Range','B2');
            end

            % Value changed function: HybridSelected
            function HybridSelectedValueChanged(app, event)
                value = app.HybridSelected.Value;
                if value == 1
                    app.BatteryWeightRatio.Value =
100*app.HybridWeight.Value/app.TOW.Value;
                else
                    app.BatteryWeightRatio.Value =
100*app.RefWeight.Value/app.TOW.Value;
                end

                if app.BatteryWeightRatio.Value <25
                    set(app.GoodLamp, "Enable", "on");
                    set(app.OkayLamp, "Enable", "off");
                    set(app.BadLamp, "Enable", "off");
                elseif app.BatteryWeightRatio.Value <50
                    set(app.GoodLamp, "Enable", "off");
                    set(app.OkayLamp, "Enable", "on");
                    set(app.BadLamp, "Enable", "off");
                elseif app.BatteryWeightRatio.Value <100
                    set(app.GoodLamp, "Enable", "off");
```

```
                    set(app.OkayLamp, "Enable", "off");
                    set(app.BadLamp, "Enable", "on");
                else
                    set(app.GoodLamp, "Enable", "off");
                    set(app.OkayLamp, "Enable", "off");
                    set(app.BadLamp, "Enable", "off");
                end
            end
        end

        % Component initialization
        methods (Access = private)

            % Create UIFigure and components
            function createComponents(app)

                % Create UIFigure and hide until all components are created
                app.UIFigure = uifigure('Visible', 'off');
                app.UIFigure.Color = [0.302 0.7451 0.9333];
                app.UIFigure.Position = [100 100 518 740];
                app.UIFigure.Name = 'UI Figure';

                % Create TabGroup
                app.TabGroup = uitabgroup(app.UIFigure);
                app.TabGroup.Position = [1 0 494 741];

                % Create EnergyPowerProfileTab
                app.EnergyPowerProfileTab = uitab(app.TabGroup);
                app.EnergyPowerProfileTab.Title = 'Energy & Power Profile';
                app.EnergyPowerProfileTab.BackgroundColor = [1 1 1];
                app.EnergyPowerProfileTab.ForegroundColor = [0 0.4471 0.7412];

                % Create DragDataPanel
                app.DragDataPanel = uipanel(app.EnergyPowerProfileTab);
                app.DragDataPanel.Title = 'Drag Data';
                app.DragDataPanel.BackgroundColor = [0.8353 0.8863 0.9216];
                app.DragDataPanel.Position = [10 587 475 82];

                % Create CdatTakeoffLabel
                app.CdatTakeoffLabel = uilabel(app.DragDataPanel);
                app.CdatTakeoffLabel.HorizontalAlignment = 'right';
                app.CdatTakeoffLabel.Position = [7 35 76 22];
                app.CdatTakeoffLabel.Text = 'Cd at Takeoff';

                % Create CdTakeOff
                app.CdTakeOff = uieditfield(app.DragDataPanel, 'numeric');
                app.CdTakeOff.Position = [91 35 63 22];
                app.CdTakeOff.Value = 0.3;

                % Create CdatCruiseLabel
                app.CdatCruiseLabel = uilabel(app.DragDataPanel);
                app.CdatCruiseLabel.HorizontalAlignment = 'right';
                app.CdatCruiseLabel.Position = [160 35 72 22];
                app.CdatCruiseLabel.Text = 'Cd at Cruise';

                % Create CdCruise
                app.CdCruise = uieditfield(app.DragDataPanel, 'numeric');
                app.CdCruise.Position = [240 35 61 22];
                app.CdCruise.Value = 0.3;

                % Create SurfaceAeraCruise
                app.SurfaceAeraCruise = uieditfield(app.DragDataPanel,
'numeric');
                app.SurfaceAeraCruise.ValueDisplayFormat = '%11.4gm2';
                app.SurfaceAeraCruise.Position = [399 35 63 22];
                app.SurfaceAeraCruise.Value = 1.47;

                % Create FrontAeraLabel
                app.FrontAeraLabel = uilabel(app.DragDataPanel);
                app.FrontAeraLabel.HorizontalAlignment = 'right';
                app.FrontAeraLabel.Position = [327 35 62 22];
                app.FrontAeraLabel.Text = 'Front Aera';

                % Create CdDescendLabel
                app.CdDescendLabel = uilabel(app.DragDataPanel);
                app.CdDescendLabel.HorizontalAlignment = 'right';
                app.CdDescendLabel.Position = [161 5 71 22];
                app.CdDescendLabel.Text = 'Cd Descend';

                % Create CdDescend
                app.CdDescend = uieditfield(app.DragDataPanel, 'numeric');
                app.CdDescend.Position = [240 5 61 22];
                app.CdDescend.Value = 0.3;

                % Create CdatClimbLabel
                app.CdatClimbLabel = uilabel(app.DragDataPanel);
                app.CdatClimbLabel.HorizontalAlignment = 'right';
                app.CdatClimbLabel.Position = [15 5 68 22];
                app.CdatClimbLabel.Text = 'Cd at Climb';

                % Create CdClimb
                app.CdClimb = uieditfield(app.DragDataPanel, 'numeric');
                app.CdClimb.Position = [91 5 63 22];
                app.CdClimb.Value = 0.3;

                % Create TopAeraLabel
                app.TopAeraLabel = uilabel(app.DragDataPanel);
                app.TopAeraLabel.HorizontalAlignment = 'right';
                app.TopAeraLabel.Position = [339 5 54 22];
                app.TopAeraLabel.Text = 'Top Aera';

                % Create SurfaceAeraTakeOff
                app.SurfaceAeraTakeOff = uieditfield(app.DragDataPanel,
'numeric');
                app.SurfaceAeraTakeOff.ValueDisplayFormat = '%11.4gm2';
                app.SurfaceAeraTakeOff.Position = [399 5 63 22];
                app.SurfaceAeraTakeOff.Value = 2.11;

                % Create TakeOffandLandingDataPanel
                app.TakeOffandLandingDataPanel =
uipanel(app.EnergyPowerProfileTab);
                app.TakeOffandLandingDataPanel.Title = 'Take Off and Landing
Data';
                app.TakeOffandLandingDataPanel.BackgroundColor = [0.8353 0.8863
0.9216];
                app.TakeOffandLandingDataPanel.Position = [10 495 475 82];

                % Create AccelerationEditFieldLabel
                app.AccelerationEditFieldLabel =
uilabel(app.TakeOffandLandingDataPanel);
                app.AccelerationEditFieldLabel.HorizontalAlignment = 'right';
                app.AccelerationEditFieldLabel.Position = [322 35 71 22];
                app.AccelerationEditFieldLabel.Text = 'Acceleration';

                % Create AccelerationTakeOff
                app.AccelerationTakeOff =
uieditfield(app.TakeOffandLandingDataPanel, 'numeric');
                app.AccelerationTakeOff.ValueDisplayFormat = '%11.4gg';
                app.AccelerationTakeOff.Position = [401 35 61 22];
                app.AccelerationTakeOff.Value = 0.5;

                % Create TOW
                app.TOW = uieditfield(app.TakeOffandLandingDataPanel, 'numeric');
                app.TOW.ValueDisplayFormat = '%11.4gkg';
                app.TOW.Position = [240 35 61 22];
                app.TOW.Value = 350;

                % Create TOWEditFieldLabel
                app.TOWEditFieldLabel = uilabel(app.TakeOffandLandingDataPanel);
                app.TOWEditFieldLabel.HorizontalAlignment = 'right';
                app.TOWEditFieldLabel.Position = [199 35 33 22];
                app.TOWEditFieldLabel.Text = 'TOW';

                % Create ClimbRateEditFieldLabel
                app.ClimbRateEditFieldLabel =
uilabel(app.TakeOffandLandingDataPanel);
                app.ClimbRateEditFieldLabel.HorizontalAlignment = 'right';
                app.ClimbRateEditFieldLabel.Position = [18 35 65 22];
                app.ClimbRateEditFieldLabel.Text = 'Climb Rate';

                % Create ClimbRate
                app.ClimbRate = uieditfield(app.TakeOffandLandingDataPanel,
'numeric');
                app.ClimbRate.ValueDisplayFormat = '%11.4gm/s';
                app.ClimbRate.Position = [91 35 63 22];
                app.ClimbRate.Value = 3;

                % Create DescentRateEditFieldLabel
                app.DescentRateEditFieldLabel =
uilabel(app.TakeOffandLandingDataPanel);
                app.DescentRateEditFieldLabel.HorizontalAlignment = 'right';
                app.DescentRateEditFieldLabel.Position = [3 4 78 22];
                app.DescentRateEditFieldLabel.Text = 'Descent Rate';

                % Create DescentRate
                app.DescentRate = uieditfield(app.TakeOffandLandingDataPanel,
'numeric');
                app.DescentRate.ValueDisplayFormat = '%11.4gm/s';
                app.DescentRate.Position = [91 4 63 22];
                app.DescentRate.Value = 3;

                % Create CdTransitionLabel
                app.CdTransitionLabel = uilabel(app.TakeOffandLandingDataPanel);
                app.CdTransitionLabel.HorizontalAlignment = 'right';
                app.CdTransitionLabel.Position = [156 4 76 22];
                app.CdTransitionLabel.Text = 'Cd Transition';

                % Create CdTrans
                app.CdTrans = uieditfield(app.TakeOffandLandingDataPanel,
'numeric');
                app.CdTrans.Position = [238 4 63 22];
                app.CdTrans.Value = 2.11;

                % Create CruiseDataPanel
                app.CruiseDataPanel = uipanel(app.EnergyPowerProfileTab);
                app.CruiseDataPanel.Title = 'Cruise Data';
                app.CruiseDataPanel.BackgroundColor = [0.8353 0.8863 0.9216];
                app.CruiseDataPanel.Position = [10 403 475 82];

                % Create SpeedEditFieldLabel
                app.SpeedEditFieldLabel = uilabel(app.CruiseDataPanel);
                app.SpeedEditFieldLabel.HorizontalAlignment = 'right';
                app.SpeedEditFieldLabel.Position = [42 35 40 22];
                app.SpeedEditFieldLabel.Text = 'Speed';

                % Create Speed
                app.Speed = uieditfield(app.CruiseDataPanel, 'numeric');
                app.Speed.ValueDisplayFormat = '%11.4gkm/h';
                app.Speed.Position = [90 35 61 22];
                app.Speed.Value = 100;

                % Create AltitudeEditFieldLabel
                app.AltitudeEditFieldLabel = uilabel(app.CruiseDataPanel);
                app.AltitudeEditFieldLabel.HorizontalAlignment = 'right';
                app.AltitudeEditFieldLabel.Position = [185 35 46 22];
                app.AltitudeEditFieldLabel.Text = 'Altitude';

                % Create Altitude
                app.Altitude = uieditfield(app.CruiseDataPanel, 'numeric');
                app.Altitude.ValueDisplayFormat = '%11.4gm';
                app.Altitude.Position = [239 35 62 22];
                app.Altitude.Value = 800;

                % Create Range
                app.Range = uieditfield(app.CruiseDataPanel, 'numeric');
                app.Range.ValueDisplayFormat = '%11.4gkm';
                app.Range.Position = [401 35 64 22];
                app.Range.Value = 40;

                % Create RangeEditFieldLabel
                app.RangeEditFieldLabel = uilabel(app.CruiseDataPanel);
                app.RangeEditFieldLabel.HorizontalAlignment = 'right';
                app.RangeEditFieldLabel.Position = [346 35 41 22];
                app.RangeEditFieldLabel.Text = 'Range';

                % Create StallSpeedLabel
                app.StallSpeedLabel = uilabel(app.CruiseDataPanel);
                app.StallSpeedLabel.HorizontalAlignment = 'right';
                app.StallSpeedLabel.Position = [15 4 67 22];
                app.StallSpeedLabel.Text = 'Stall Speed';

                % Create StallSpeed
                app.StallSpeed = uieditfield(app.CruiseDataPanel, 'numeric');
                app.StallSpeed.ValueDisplayFormat = '%11.4gkm/h';
                app.StallSpeed.Position = [90 4 61 22];

                % Create PropellerDiameter
                app.PropellerDiameter = uieditfield(app.CruiseDataPanel,
'numeric');
                app.PropellerDiameter.ValueDisplayFormat = '%11.4gm';
                app.PropellerDiameter.Position = [240 4 61 22];
                app.PropellerDiameter.Value = 1.57;

                % Create PropellerDLabel
                app.PropellerDLabel = uilabel(app.CruiseDataPanel);
                app.PropellerDLabel.HorizontalAlignment = 'right';
                app.PropellerDLabel.Position = [163 4 69 22];
                app.PropellerDLabel.Text = 'Propeller D.';

                % Create CalculateButton
                app.CalculateButton = uibutton(app.EnergyPowerProfileTab,
'push');
                app.CalculateButton.ButtonPushedFcn = createCallbackFcn(app,
@CalculateButtonPushed, true);
                app.CalculateButton.BackgroundColor = [0.0745 0.6235 1];
                app.CalculateButton.FontColor = [1 1 1];
                app.CalculateButton.Position = [420 683 64 22];
                app.CalculateButton.Text = 'Calculate';

                % Create ThrustEfficiencyMainMotorPanel
                app.ThrustEfficiencyMainMotorPanel =
uipanel(app.EnergyPowerProfileTab);
                app.ThrustEfficiencyMainMotorPanel.Title = 'Thrust Efficiency -
Main Motor';
                app.ThrustEfficiencyMainMotorPanel.BackgroundColor = [0.8353
0.8863 0.9216];
                app.ThrustEfficiencyMainMotorPanel.Position = [10 273 230 120];

                % Create At50EditFieldLabel
                app.At50EditFieldLabel =
uilabel(app.ThrustEfficiencyMainMotorPanel);
                app.At50EditFieldLabel.HorizontalAlignment = 'right';
                app.At50EditFieldLabel.Position = [1 73 44 22];
                app.At50EditFieldLabel.Text = 'At 50%';

                % Create MidThrottlePowerMain
                app.MidThrottlePowerMain =
uieditfield(app.ThrustEfficiencyMainMotorPanel, 'numeric');
                app.MidThrottlePowerMain.ValueDisplayFormat = '%11.4gg/W';
                app.MidThrottlePowerMain.Position = [50 73 52 22];
                app.MidThrottlePowerMain.Value = 7.5;

                % Create At100EditFieldLabel
                app.At100EditFieldLabel =
uilabel(app.ThrustEfficiencyMainMotorPanel);
                app.At100EditFieldLabel.HorizontalAlignment = 'right';
                app.At100EditFieldLabel.Position = [110 73 51 22];
                app.At100EditFieldLabel.Text = 'At 100%';

                % Create FullThrottlePowerMain
                app.FullThrottlePowerMain =
uieditfield(app.ThrustEfficiencyMainMotorPanel, 'numeric');
                app.FullThrottlePowerMain.ValueDisplayFormat = '%11.4gg/W';
                app.FullThrottlePowerMain.Position = [166 73 55 22];
                app.FullThrottlePowerMain.Value = 4.5;

                % Create At50EditFieldLabel_2
```

```
            app.At50EditFieldLabel_2 =
uilabel(app.ThrustEfficiencyMainMotorPanel);
            app.At50EditFieldLabel_2.HorizontalAlignment = 'right';
            app.At50EditFieldLabel_2.Position = [1 42 44 22];
            app.At50EditFieldLabel_2.Text = 'At 50%';

            % Create MidThrottleThrustMain
            app.MidThrottleThrustMain =
uieditfield(app.ThrustEfficiencyMainMotorPanel, 'numeric');
            app.MidThrottleThrustMain.ValueDisplayFormat = '%11.4gkg';
            app.MidThrottleThrustMain.Position = [50 42 52 22];
            app.MidThrottleThrustMain.Value = 36.5;

            % Create At100EditFieldLabel_2
            app.At100EditFieldLabel_2 =
uilabel(app.ThrustEfficiencyMainMotorPanel);
            app.At100EditFieldLabel_2.HorizontalAlignment = 'right';
            app.At100EditFieldLabel_2.Position = [111 42 51 22];
            app.At100EditFieldLabel_2.Text = 'At 100%';

            % Create FullThrottleThrustMain
            app.FullThrottleThrustMain =
uieditfield(app.ThrustEfficiencyMainMotorPanel, 'numeric');
            app.FullThrottleThrustMain.ValueDisplayFormat = '%11.4gkg';
            app.FullThrottleThrustMain.Position = [167 42 55 22];
            app.FullThrottleThrustMain.Value = 102.4;

            % Create MotorsLabel
            app.MotorsLabel = uilabel(app.ThrustEfficiencyMainMotorPanel);
            app.MotorsLabel.HorizontalAlignment = 'right';
            app.MotorsLabel.Position = [50 10 107 22];
            app.MotorsLabel.Text = 'Motors';

            % Create NumMain
            app.NumMain = uispinner(app.ThrustEfficiencyMainMotorPanel);
            app.NumMain.Limits = [1 Inf];
            app.NumMain.Position = [167 10 56 22];
            app.NumMain.Value = 8;

            % Create Label
            app.Label = uilabel(app.ThrustEfficiencyMainMotorPanel);
            app.Label.HorizontalAlignment = 'right';
            app.Label.Position = [20 10 25 22];
            app.Label.Text = 'η';

            % Create MainEfficiency
            app.MainEfficiency =
uieditfield(app.ThrustEfficiencyMainMotorPanel, 'numeric');
            app.MainEfficiency.Limits = [0 1];
            app.MainEfficiency.Position = [50 10 52 22];
            app.MainEfficiency.Value = 0.94;

            % Create ConfigurationDropDownLabel
            app.ConfigurationDropDownLabel =
uilabel(app.EnergyPowerProfileTab);
            app.ConfigurationDropDownLabel.HorizontalAlignment = 'right';
            app.ConfigurationDropDownLabel.Position = [10 683 77 22];
            app.ConfigurationDropDownLabel.Text = 'Configuration';

            % Create ConfigurationDropDown
            app.ConfigurationDropDown =
uidropdown(app.EnergyPowerProfileTab);
            app.ConfigurationDropDown.Items = {'Fixed Wing', 'Multicopter',
'VTOL'};
            app.ConfigurationDropDown.ValueChangedFcn =
createCallbackFcn(app, @ConfigurationDropDownValueChanged, true);
            app.ConfigurationDropDown.Position = [94 683 100 22];
            app.ConfigurationDropDown.Value = 'Fixed Wing';

            % Create ThrustEfficiencyAuxMotorPanel
            app.ThrustEfficiencyAuxMotorPanel =
uipanel(app.EnergyPowerProfileTab);
            app.ThrustEfficiencyAuxMotorPanel.Title = 'Thrust Efficiency -
Aux Motor';
            app.ThrustEfficiencyAuxMotorPanel.BackgroundColor = [0.8392
0.8902 0.9216];
            app.ThrustEfficiencyAuxMotorPanel.Position = [247 273 236 120];

            % Create At50EditFieldLabel_3
            app.At50EditFieldLabel_3 =
uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.At50EditFieldLabel_3.HorizontalAlignment = 'right';
            app.At50EditFieldLabel_3.Position = [1 73 44 22];
            app.At50EditFieldLabel_3.Text = 'At 50%';

            % Create MidThrottlePowerAux
            app.MidThrottlePowerAux =
uieditfield(app.ThrustEfficiencyAuxMotorPanel, 'numeric');
            app.MidThrottlePowerAux.ValueDisplayFormat = '%11.4gg/W';
            app.MidThrottlePowerAux.Position = [50 73 52 22];

            % Create At100EditFieldLabel_3
            app.At100EditFieldLabel_3 =
uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.At100EditFieldLabel_3.HorizontalAlignment = 'right';
            app.At100EditFieldLabel_3.Position = [114 73 51 22];
            app.At100EditFieldLabel_3.Text = 'At 100%';

            % Create FullThrottlePowerAux
            app.FullThrottlePowerAux =
uieditfield(app.ThrustEfficiencyAuxMotorPanel, 'numeric');
            app.FullThrottlePowerAux.ValueDisplayFormat = '%11.4gg/W';
            app.FullThrottlePowerAux.Position = [170 73 55 22];

            % Create At50EditFieldLabel_4
            app.At50EditFieldLabel_4 =
uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.At50EditFieldLabel_4.HorizontalAlignment = 'right';
            app.At50EditFieldLabel_4.Position = [1 42 44 22];
            app.At50EditFieldLabel_4.Text = 'At 50%';

            % Create MidThrottleThrustAux
            app.MidThrottleThrustAux =
uieditfield(app.ThrustEfficiencyAuxMotorPanel, 'numeric');
            app.MidThrottleThrustAux.ValueDisplayFormat = '%11.4gkg';
            app.MidThrottleThrustAux.Position = [50 42 52 22];

            % Create At100EditFieldLabel_4
            app.At100EditFieldLabel_4 =
uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.At100EditFieldLabel_4.HorizontalAlignment = 'right';
            app.At100EditFieldLabel_4.Position = [114 42 51 22];
            app.At100EditFieldLabel_4.Text = 'At 100%';

            % Create FullThrottleThrustAux
            app.FullThrottleThrustAux =
uieditfield(app.ThrustEfficiencyAuxMotorPanel, 'numeric');
            app.FullThrottleThrustAux.ValueDisplayFormat = '%11.4gkg';
            app.FullThrottleThrustAux.Position = [170 42 55 22];

            % Create MotorsLabel_2
            app.MotorsLabel_2 = uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.MotorsLabel_2.HorizontalAlignment = 'right';
            app.MotorsLabel_2.Position = [53 10 107 22];
            app.MotorsLabel_2.Text = 'Motors';

            % Create NumAux
            app.NumAux = uispinner(app.ThrustEfficiencyAuxMotorPanel);
            app.NumAux.Limits = [0 Inf];
            app.NumAux.Position = [170 10 55 22];

            % Create Label_2
            app.Label_2 = uilabel(app.ThrustEfficiencyAuxMotorPanel);
            app.Label_2.HorizontalAlignment = 'right';
            app.Label_2.Position = [21 10 25 22];
            app.Label_2.Text = 'η';

            % Create AuxEfficiency
            app.AuxEfficiency =
uieditfield(app.ThrustEfficiencyAuxMotorPanel, 'numeric');
            app.AuxEfficiency.Limits = [0 1];
            app.AuxEfficiency.Position = [51 10 52 22];
            app.AuxEfficiency.Value = 0.94;

            % Create EnergyPowerInfo
            app.EnergyPowerInfo = uitabgroup(app.EnergyPowerProfileTab);
            app.EnergyPowerInfo.Position = [10 11 473 252];

            % Create PowerProfileTab
            app.PowerProfileTab = uitab(app.EnergyPowerInfo);
            app.PowerProfileTab.Title = 'Power Profile';
            app.PowerProfileTab.BackgroundColor = [0.9412 0.9412 0.9412];

            % Create UIAxesPowerProfile
            app.UIAxesPowerProfile = uiaxes(app.PowerProfileTab);
            title(app.UIAxesPowerProfile, 'Power Profile')
            xlabel(app.UIAxesPowerProfile, 'Time (s)')
            ylabel(app.UIAxesPowerProfile, 'Power Requirements (kWh)')
            app.UIAxesPowerProfile.PlotBoxAspectRatio = [1.75925925925926 1
1];
            app.UIAxesPowerProfile.Position = [6 5 285 212];

            % Create EnergyforTakeoffEditFieldLabel
            app.EnergyforTakeoffEditFieldLabel =
uilabel(app.PowerProfileTab);
            app.EnergyforTakeoffEditFieldLabel.HorizontalAlignment = 'right';
            app.EnergyforTakeoffEditFieldLabel.Position = [296 200 103 22];
            app.EnergyforTakeoffEditFieldLabel.Text = 'Energy for Takeoff';

            % Create EnergyTakeOff
            app.EnergyTakeOff = uieditfield(app.PowerProfileTab, 'numeric');
            app.EnergyTakeOff.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyTakeOff.Editable = 'off';
            app.EnergyTakeOff.Position = [410 200 58 22];

            % Create MaxPowerTakeoffLabel
            app.MaxPowerTakeoffLabel = uilabel(app.PowerProfileTab);
            app.MaxPowerTakeoffLabel.HorizontalAlignment = 'right';
            app.MaxPowerTakeoffLabel.Position = [292 103 107 22];
            app.MaxPowerTakeoffLabel.Text = 'Max Power Takeoff';

            % Create PowerTakeOff
            app.PowerTakeOff = uieditfield(app.PowerProfileTab, 'numeric');
            app.PowerTakeOff.ValueDisplayFormat = '%11.4gkW';
            app.PowerTakeOff.Editable = 'off';
            app.PowerTakeOff.Position = [410 103 58 22];

            % Create MaxPowerClimbLabel
            app.MaxPowerClimbLabel = uilabel(app.PowerProfileTab);
            app.MaxPowerClimbLabel.HorizontalAlignment = 'right';
            app.MaxPowerClimbLabel.Position = [300 79 99 22];
            app.MaxPowerClimbLabel.Text = 'Max Power Climb';

            % Create PowerClimb
            app.PowerClimb = uieditfield(app.PowerProfileTab, 'numeric');
            app.PowerClimb.ValueDisplayFormat = '%11.4gkW';
            app.PowerClimb.Editable = 'off';
            app.PowerClimb.Position = [410 79 58 22];

            % Create EnergyforCruiseEditFieldLabel
            app.EnergyforCruiseEditFieldLabel = uilabel(app.PowerProfileTab);
            app.EnergyforCruiseEditFieldLabel.HorizontalAlignment = 'right';
            app.EnergyforCruiseEditFieldLabel.Position = [300 152 99 22];
            app.EnergyforCruiseEditFieldLabel.Text = 'Energy for Cruise';

            % Create EnergyCruise
            app.EnergyCruise = uieditfield(app.PowerProfileTab, 'numeric');
            app.EnergyCruise.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyCruise.Editable = 'off';
            app.EnergyCruise.Position = [410 152 58 22];

            % Create TotalEnergyEditFieldLabel
            app.TotalEnergyEditFieldLabel = uilabel(app.PowerProfileTab);
            app.TotalEnergyEditFieldLabel.HorizontalAlignment = 'right';
            app.TotalEnergyEditFieldLabel.Position = [327 5 72 22];
            app.TotalEnergyEditFieldLabel.Text = 'Total Energy';

            % Create TotalEnergy
            app.TotalEnergy = uieditfield(app.PowerProfileTab, 'numeric');
            app.TotalEnergy.ValueDisplayFormat = '%11.4gkWh';
            app.TotalEnergy.Editable = 'off';
            app.TotalEnergy.Position = [410 5 58 22];

            % Create EnergyforClimbLabel
            app.EnergyforClimbLabel = uilabel(app.PowerProfileTab);
            app.EnergyforClimbLabel.HorizontalAlignment = 'right';
            app.EnergyforClimbLabel.Position = [304 176 95 22];
            app.EnergyforClimbLabel.Text = 'Energy for Climb';

            % Create EnergyClimb
            app.EnergyClimb = uieditfield(app.PowerProfileTab, 'numeric');
            app.EnergyClimb.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyClimb.Editable = 'off';
            app.EnergyClimb.Position = [410 176 58 22];

            % Create EnergyforDescendLabel
            app.EnergyforDescendLabel = uilabel(app.PowerProfileTab);
            app.EnergyforDescendLabel.HorizontalAlignment = 'right';
            app.EnergyforDescendLabel.Position = [288 128 111 22];
            app.EnergyforDescendLabel.Text = 'Energy for Descend';

            % Create EnergyDescend
            app.EnergyDescend = uieditfield(app.PowerProfileTab, 'numeric');
            app.EnergyDescend.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyDescend.Editable = 'off';
            app.EnergyDescend.Position = [410 128 58 22];

            % Create PowerCruiseLabel
            app.PowerCruiseLabel = uilabel(app.PowerProfileTab);
            app.PowerCruiseLabel.HorizontalAlignment = 'right';
            app.PowerCruiseLabel.Position = [318 55 81 22];
            app.PowerCruiseLabel.Text = ' Power Cruise';

            % Create PowerCruise
            app.PowerCruise = uieditfield(app.PowerProfileTab, 'numeric');
            app.PowerCruise.ValueDisplayFormat = '%11.4gkW';
            app.PowerCruise.Editable = 'off';
            app.PowerCruise.Position = [410 55 58 22];

            % Create MaxPowerDescendLabel
            app.MaxPowerDescendLabel = uilabel(app.PowerProfileTab);
            app.MaxPowerDescendLabel.HorizontalAlignment = 'right';
            app.MaxPowerDescendLabel.Position = [283 31 116 22];
            app.MaxPowerDescendLabel.Text = 'Max Power Descend';

            % Create PowerDescend
            app.PowerDescend = uieditfield(app.PowerProfileTab, 'numeric');
            app.PowerDescend.ValueDisplayFormat = '%11.4gkW';
            app.PowerDescend.Editable = 'off';
            app.PowerDescend.Position = [410 31 58 22];

            % Create DragProfileTab
            app.DragProfileTab = uitab(app.EnergyPowerInfo);
            app.DragProfileTab.Title = 'Drag Profile';
            app.DragProfileTab.BackgroundColor = [0.9412 0.9412 0.9412];

            % Create UIAxesDragProfile
            app.UIAxesDragProfile = uiaxes(app.DragProfileTab);
            title(app.UIAxesDragProfile, 'Drag Profile')
            xlabel(app.UIAxesDragProfile, 'Time (s)')
            ylabel(app.UIAxesDragProfile, 'Force (N)')
            app.UIAxesDragProfile.PlotBoxAspectRatio = [1.75925925925926 1
1];
            app.UIAxesDragProfile.Position = [6 5 285 212];

            % Create MaxDragTakeOffLabel
            app.MaxDragTakeOffLabel = uilabel(app.DragProfileTab);
```

```
            app.MaxDragTakeOffLabel.HorizontalAlignment = 'right';
            app.MaxDragTakeOffLabel.Position = [294 158 102 22];
            app.MaxDragTakeOffLabel.Text = 'Max Drag TakeOff';

            % Create DragTakeOff
            app.DragTakeOff = uieditfield(app.DragProfileTab, 'numeric');
            app.DragTakeOff.ValueDisplayFormat = '%11.4gkN';
            app.DragTakeOff.Editable = 'off';
            app.DragTakeOff.Position = [407 158 58 22];

            % Create MaxDragDescendLabel
            app.MaxDragDescendLabel = uilabel(app.DragProfileTab);
            app.MaxDragDescendLabel.HorizontalAlignment = 'right';
            app.MaxDragDescendLabel.Position = [288 86 108 22];
            app.MaxDragDescendLabel.Text = 'Max Drag Descend';

            % Create DragDescend
            app.DragDescend = uieditfield(app.DragProfileTab, 'numeric');
            app.DragDescend.ValueDisplayFormat = '%11.4gkN';
            app.DragDescend.Editable = 'off';
            app.DragDescend.Position = [407 86 58 22];

            % Create ForceAccelerationLabel
            app.ForceAccelerationLabel = uilabel(app.DragProfileTab);
            app.ForceAccelerationLabel.HorizontalAlignment = 'right';
            app.ForceAccelerationLabel.Position = [291 46 106 22];
            app.ForceAccelerationLabel.Text = 'Force Acceleration';

            % Create ForceAcceleration
            app.ForceAcceleration = uieditfield(app.DragProfileTab,
'numeric');
            app.ForceAcceleration.ValueDisplayFormat = '%11.4gkN';
            app.ForceAcceleration.Editable = 'off';
            app.ForceAcceleration.Position = [408 46 58 22];

            % Create MaxDragClimbLabel
            app.MaxDragClimbLabel = uilabel(app.DragProfileTab);
            app.MaxDragClimbLabel.HorizontalAlignment = 'right';
            app.MaxDragClimbLabel.Position = [305 134 91 22];
            app.MaxDragClimbLabel.Text = 'Max Drag Climb';

            % Create DragClimb
            app.DragClimb = uieditfield(app.DragProfileTab, 'numeric');
            app.DragClimb.ValueDisplayFormat = '%11.4gkN';
            app.DragClimb.Editable = 'off';
            app.DragClimb.Position = [407 134 58 22];

            % Create DragCruiseLabel
            app.DragCruiseLabel = uilabel(app.DragProfileTab);
            app.DragCruiseLabel.HorizontalAlignment = 'right';
            app.DragCruiseLabel.Position = [326 110 70 22];
            app.DragCruiseLabel.Text = 'Drag Cruise';

            % Create DragCruise
            app.DragCruise = uieditfield(app.DragProfileTab, 'numeric');
            app.DragCruise.ValueDisplayFormat = '%11.4gkN';
            app.DragCruise.Editable = 'off';
            app.DragCruise.Position = [407 110 58 22];

            % Create SpeedProfileTab
            app.SpeedProfileTab = uitab(app.EnergyPowerInfo);
            app.SpeedProfileTab.Title = 'Speed Profile';

            % Create UIAxesSpeedProfile
            app.UIAxesSpeedProfile = uiaxes(app.SpeedProfileTab);
            title(app.UIAxesSpeedProfile, 'Speed Profile')
            xlabel(app.UIAxesSpeedProfile, 'Time (s)')
            ylabel(app.UIAxesSpeedProfile, 'Speed (km/h)')
            app.UIAxesSpeedProfile.PlotBoxAspectRatio = [1.75925925925926 1
1];
            app.UIAxesSpeedProfile.Position = [6 5 285 212];

            % Create TimeTakeOffLabel
            app.TimeTakeOffLabel = uilabel(app.SpeedProfileTab);
            app.TimeTakeOffLabel.HorizontalAlignment = 'right';
            app.TimeTakeOffLabel.Position = [317 186 79 22];
            app.TimeTakeOffLabel.Text = 'Time Take Off';

            % Create DisplayTimeTakeOff
            app.DisplayTimeTakeOff = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayTimeTakeOff.ValueDisplayFormat = '%11.4gs';
            app.DisplayTimeTakeOff.Editable = 'off';
            app.DisplayTimeTakeOff.Position = [407 186 58 22];

            % Create TimeDescendLabel
            app.TimeDescendLabel = uilabel(app.SpeedProfileTab);
            app.TimeDescendLabel.HorizontalAlignment = 'right';
            app.TimeDescendLabel.Position = [313 114 83 22];
            app.TimeDescendLabel.Text = 'Time Descend';

            % Create DisplayTimeDescend
            app.DisplayTimeDescend = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayTimeDescend.ValueDisplayFormat = '%11.4gs';
            app.DisplayTimeDescend.Editable = 'off';
            app.DisplayTimeDescend.Position = [407 114 58 22];

            % Create TotalTimeLabel
            app.TotalTimeLabel = uilabel(app.SpeedProfileTab);
            app.TotalTimeLabel.HorizontalAlignment = 'right';
            app.TotalTimeLabel.Position = [336 6 60 22];
            app.TotalTimeLabel.Text = 'Total Time';

            % Create DisplayTotalTime
            app.DisplayTotalTime = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayTotalTime.ValueDisplayFormat = '%11.4gmins';
            app.DisplayTotalTime.Editable = 'off';
            app.DisplayTotalTime.Position = [407 6 58 22];

            % Create TimeClimbLabel
            app.TimeClimbLabel = uilabel(app.SpeedProfileTab);
            app.TimeClimbLabel.HorizontalAlignment = 'right';
            app.TimeClimbLabel.Position = [330 162 66 22];
            app.TimeClimbLabel.Text = 'Time Climb';

            % Create DisplayTimeClimb
            app.DisplayTimeClimb = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayTimeClimb.ValueDisplayFormat = '%11.4gs';
            app.DisplayTimeClimb.Editable = 'off';
            app.DisplayTimeClimb.Position = [407 162 58 22];

            % Create TimeCruiseLabel
            app.TimeCruiseLabel = uilabel(app.SpeedProfileTab);
            app.TimeCruiseLabel.HorizontalAlignment = 'right';
            app.TimeCruiseLabel.Position = [326 138 70 22];
            app.TimeCruiseLabel.Text = 'Time Cruise';

            % Create DisplayTimeCruise
            app.DisplayTimeCruise = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayTimeCruise.ValueDisplayFormat = '%11.4gs';
            app.DisplayTimeCruise.Editable = 'off';
            app.DisplayTimeCruise.Position = [407 138 58 22];

            % Create RangeClimbLabel
            app.RangeClimbLabel = uilabel(app.SpeedProfileTab);
            app.RangeClimbLabel.HorizontalAlignment = 'right';
            app.RangeClimbLabel.Position = [321 90 75 22];
            app.RangeClimbLabel.Text = 'Range Climb';

            % Create DisplayRangeClimb
            app.DisplayRangeClimb = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayRangeClimb.ValueDisplayFormat = '%11.4gkm';
            app.DisplayRangeClimb.Editable = 'off';
            app.DisplayRangeClimb.Position = [407 90 58 22];

            % Create RangeCruiseLabel
            app.RangeCruiseLabel = uilabel(app.SpeedProfileTab);
            app.RangeCruiseLabel.HorizontalAlignment = 'right';
            app.RangeCruiseLabel.Position = [317 66 79 22];
            app.RangeCruiseLabel.Text = 'Range Cruise';

            % Create DisplayRangeCruise
            app.DisplayRangeCruise = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayRangeCruise.ValueDisplayFormat = '%11.4gkm';
            app.DisplayRangeCruise.Editable = 'off';
            app.DisplayRangeCruise.Position = [407 66 58 22];

            % Create RangeDescendLabel
            app.RangeDescendLabel = uilabel(app.SpeedProfileTab);
            app.RangeDescendLabel.HorizontalAlignment = 'right';
            app.RangeDescendLabel.Position = [305 42 91 22];
            app.RangeDescendLabel.Text = 'Range Descend';

            % Create DisplayRangeDescend
            app.DisplayRangeDescend = uieditfield(app.SpeedProfileTab,
'numeric');
            app.DisplayRangeDescend.ValueDisplayFormat = '%11.4gkm';
            app.DisplayRangeDescend.Editable = 'off';
            app.DisplayRangeDescend.Position = [407 42 58 22];

            % Create MotorEfficiencyTab
            app.MotorEfficiencyTab = uitab(app.EnergyPowerInfo);
            app.MotorEfficiencyTab.Title = 'Motor Efficiency';

            % Create UIAxesEfficiency
            app.UIAxesEfficiency = uiaxes(app.MotorEfficiencyTab);
            title(app.UIAxesEfficiency, 'Motor Efficiency')
            xlabel(app.UIAxesEfficiency, 'Time (mins)')
            ylabel(app.UIAxesEfficiency, 'Power Requirements (kWh)')
            app.UIAxesEfficiency.PlotBoxAspectRatio = [1.75925925925926 1 1];
            app.UIAxesEfficiency.Position = [6 5 285 212];

            % Create TakeOffLabel
            app.TakeOffLabel = uilabel(app.MotorEfficiencyTab);
            app.TakeOffLabel.HorizontalAlignment = 'right';
            app.TakeOffLabel.Position = [346 158 50 22];
            app.TakeOffLabel.Text = 'Take Off';

            % Create TakeOffEfficiency
            app.TakeOffEfficiency = uieditfield(app.MotorEfficiencyTab,
'numeric');
            app.TakeOffEfficiency.ValueDisplayFormat = '%11.4gg/W';
            app.TakeOffEfficiency.Editable = 'off';
            app.TakeOffEfficiency.Position = [407 158 58 22];

            % Create DescendLabel
            app.DescendLabel = uilabel(app.MotorEfficiencyTab);
            app.DescendLabel.HorizontalAlignment = 'right';
            app.DescendLabel.Position = [343 86 53 22];
            app.DescendLabel.Text = 'Descend';

            % Create DescendEfficiency
            app.DescendEfficiency = uieditfield(app.MotorEfficiencyTab,
'numeric');
            app.DescendEfficiency.ValueDisplayFormat = '%11.4gg/W';
            app.DescendEfficiency.Editable = 'off';
            app.DescendEfficiency.Position = [407 86 58 22];

            % Create ClimbLabel
            app.ClimbLabel = uilabel(app.MotorEfficiencyTab);
            app.ClimbLabel.HorizontalAlignment = 'right';
            app.ClimbLabel.Position = [360 134 36 22];
            app.ClimbLabel.Text = 'Climb';

            % Create ClimbEfficiency
            app.ClimbEfficiency = uieditfield(app.MotorEfficiencyTab,
'numeric');
            app.ClimbEfficiency.ValueDisplayFormat = '%11.4gg/W';
            app.ClimbEfficiency.Editable = 'off';
            app.ClimbEfficiency.Position = [407 134 58 22];

            % Create CruiseLabel
            app.CruiseLabel = uilabel(app.MotorEfficiencyTab);
            app.CruiseLabel.HorizontalAlignment = 'right';
            app.CruiseLabel.Position = [356 110 40 22];
            app.CruiseLabel.Text = 'Cruise';

            % Create CruiseEfficiency
            app.CruiseEfficiency = uieditfield(app.MotorEfficiencyTab,
'numeric');
            app.CruiseEfficiency.ValueDisplayFormat = '%11.4gg/W';
            app.CruiseEfficiency.Editable = 'off';
            app.CruiseEfficiency.Position = [407 110 58 22];

            % Create OveralEfficiencyLabel
            app.OveralEfficiencyLabel = uilabel(app.MotorEfficiencyTab);
            app.OveralEfficiencyLabel.HorizontalAlignment = 'right';
            app.OveralEfficiencyLabel.Position = [302 46 95 22];
            app.OveralEfficiencyLabel.Text = 'Overal Efficiency';

            % Create OveralEfficiency
            app.OveralEfficiency = uieditfield(app.MotorEfficiencyTab,
'numeric');
            app.OveralEfficiency.ValueDisplayFormat = '%11.4gg/W';
            app.OveralEfficiency.Editable = 'off';
            app.OveralEfficiency.Position = [408 46 58 22];

            % Create SweepButton
            app.SweepButton = uibutton(app.EnergyPowerProfileTab, 'push');
            app.SweepButton.ButtonPushedFcn = createCallbackFcn(app,
@SweepButtonPushed, true);
            app.SweepButton.BackgroundColor = [0.9294 0.6941 0.1255];
            app.SweepButton.FontColor = [1 1 1];
            app.SweepButton.Position = [332 683 64 22];
            app.SweepButton.Text = 'Sweep';

            % Create TestDropDownLabel
            app.TestDropDownLabel = uilabel(app.EnergyPowerProfileTab);
            app.TestDropDownLabel.HorizontalAlignment = 'right';
            app.TestDropDownLabel.Position = [209 683 27 22];
            app.TestDropDownLabel.Text = 'Test';

            % Create TestDropDown
            app.TestDropDown = uidropdown(app.EnergyPowerProfileTab);
            app.TestDropDown.Items = {'Drag Coefficients', 'Surface', 'Motor
Efficiency', 'Speed', 'Range', 'Altitude', 'Acceleration', 'TOW'};
            app.TestDropDown.Position = [238 683 90 22];
            app.TestDropDown.Value = 'Altitude';

            % Create BatteryProfileTab
            app.BatteryProfileTab = uitab(app.TabGroup);
            app.BatteryProfileTab.Title = 'Battery Profile';
            app.BatteryProfileTab.BackgroundColor = [1 1 1];
            app.BatteryProfileTab.ForegroundColor = [0 0.4471 0.7412];

            % Create BatteryRequirementsPanel
            app.BatteryRequirementsPanel = uipanel(app.BatteryProfileTab);
            app.BatteryRequirementsPanel.Title = 'Battery Requirements';
            app.BatteryRequirementsPanel.BackgroundColor = [0.8353 0.8863
0.9216];
            app.BatteryRequirementsPanel.Position = [19 471 458 196];

            % Create EnergyforTakeoffandClimbEditFieldLabel
            app.EnergyforTakeoffandClimbEditFieldLabel =
uilabel(app.BatteryRequirementsPanel);
            app.EnergyforTakeoffandClimbEditFieldLabel.HorizontalAlignment =
'right';
            app.EnergyforTakeoffandClimbEditFieldLabel.Position = [14 148 160
22];
            app.EnergyforTakeoffandClimbEditFieldLabel.Text = 'Energy for
Takeoff and Climb';
```

```
            % Create EnergyforTakeoffandClimb
            app.EnergyforTakeoffandClimb =
uieditfield(app.BatteryRequirementsPanel, 'numeric');
            app.EnergyforTakeoffandClimb.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyforTakeoffandClimb.Position = [185 148 74 22];
            app.EnergyforTakeoffandClimb.Value = 33.5;

            % Create MaxPowerforTakeoffClimbEditFieldLabel
            app.MaxPowerforTakeoffClimbEditFieldLabel =
uilabel(app.BatteryRequirementsPanel);
            app.MaxPowerforTakeoffClimbEditFieldLabel.HorizontalAlignment =
'right';
            app.MaxPowerforTakeoffClimbEditFieldLabel.Position = [12 80 162
22];
            app.MaxPowerforTakeoffClimbEditFieldLabel.Text = 'Max Power for
Takeoff /Climb';

            % Create MaxPowerTakeoffOrClimb
            app.MaxPowerTakeoffOrClimb =
uieditfield(app.BatteryRequirementsPanel, 'numeric');
            app.MaxPowerTakeoffOrClimb.ValueDisplayFormat = '%11.4gkW';
            app.MaxPowerTakeoffOrClimb.Position = [185 80 74 22];
            app.MaxPowerTakeoffOrClimb.Value = 789.7;

            % Create MaxPowerforCruiseEditField_2Label
            app.MaxPowerforCruiseEditField_2Label =
uilabel(app.BatteryRequirementsPanel);
            app.MaxPowerforCruiseEditField_2Label.HorizontalAlignment =
'right';
            app.MaxPowerforCruiseEditField_2Label.Position = [53 48 121 22];
            app.MaxPowerforCruiseEditField_2Label.Text = 'Max Power for
Cruise';

            % Create MaxPowerCruise
            app.MaxPowerCruise = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.MaxPowerCruise.ValueDisplayFormat = '%11.4gkW';
            app.MaxPowerCruise.Position = [185 48 73 22];
            app.MaxPowerCruise.Value = 252.4;

            % Create EnergyforCruiseandDescendEditFieldLabel
            app.EnergyforCruiseandDescendEditFieldLabel =
uilabel(app.BatteryRequirementsPanel);
            app.EnergyforCruiseandDescendEditFieldLabel.HorizontalAlignment =
'right';
            app.EnergyforCruiseandDescendEditFieldLabel.Position = [1 116 173
22];
            app.EnergyforCruiseandDescendEditFieldLabel.Text = 'Energy for
Cruise and Descend';

            % Create EnergyforCruiseandDescend
            app.EnergyforCruiseandDescend =
uieditfield(app.BatteryRequirementsPanel, 'numeric');
            app.EnergyforCruiseandDescend.ValueDisplayFormat = '%11.4gkWh';
            app.EnergyforCruiseandDescend.Position = [184 116 75 22];
            app.EnergyforCruiseandDescend.Value = 464.8;

            % Create SafetyExtraEditFieldLabel
            app.SafetyExtraEditFieldLabel =
uilabel(app.BatteryRequirementsPanel);
            app.SafetyExtraEditFieldLabel.HorizontalAlignment = 'right';
            app.SafetyExtraEditFieldLabel.Position = [322 149 71 22];
            app.SafetyExtraEditFieldLabel.Text = 'Safety Extra';

            % Create SafetyEnToCl
            app.SafetyEnToCl = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.SafetyEnToCl.Position = [404 149 31 22];
            app.SafetyEnToCl.Value = 20;

            % Create SafetyExtraEditField_2Label
            app.SafetyExtraEditField_2Label =
uilabel(app.BatteryRequirementsPanel);
            app.SafetyExtraEditField_2Label.HorizontalAlignment = 'right';
            app.SafetyExtraEditField_2Label.Position = [323 117 71 22];
            app.SafetyExtraEditField_2Label.Text = 'Safety Extra';

            % Create SafetyEnCrDc
            app.SafetyEnCrDc = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.SafetyEnCrDc.Position = [405 117 30 22];
            app.SafetyEnCrDc.Value = 20;

            % Create SafetyExtraEditField_3Label
            app.SafetyExtraEditField_3Label =
uilabel(app.BatteryRequirementsPanel);
            app.SafetyExtraEditField_3Label.HorizontalAlignment = 'right';
            app.SafetyExtraEditField_3Label.Position = [323 81 71 22];
            app.SafetyExtraEditField_3Label.Text = 'Safety Extra';

            % Create SafetyPoToCl
            app.SafetyPoToCl = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.SafetyPoToCl.Position = [405 81 30 22];
            app.SafetyPoToCl.Value = 20;

            % Create MaxPossiblePowerEditFieldLabel
            app.MaxPossiblePowerEditFieldLabel =
uilabel(app.BatteryRequirementsPanel);
            app.MaxPossiblePowerEditFieldLabel.HorizontalAlignment = 'right';
            app.MaxPossiblePowerEditFieldLabel.Position = [59 12 114 22];
            app.MaxPossiblePowerEditFieldLabel.Text = 'Max Possible Power';

            % Create MaxPossiblePower
            app.MaxPossiblePower = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.MaxPossiblePower.ValueDisplayFormat = '%11.4gkW';
            app.MaxPossiblePower.Position = [184 12 74 22];

            % Create SafetyFactorinLabel
            app.SafetyFactorinLabel = uilabel(app.BatteryRequirementsPanel);
            app.SafetyFactorinLabel.Position = [316 173 103 22];
            app.SafetyFactorinLabel.Text = 'Safety Factor in %';

            % Create SafetyExtraEditField_4Label
            app.SafetyExtraEditField_4Label =
uilabel(app.BatteryRequirementsPanel);
            app.SafetyExtraEditField_4Label.HorizontalAlignment = 'right';
            app.SafetyExtraEditField_4Label.Position = [323 49 71 22];
            app.SafetyExtraEditField_4Label.Text = 'Safety Extra';

            % Create SafetyPoCr
            app.SafetyPoCr = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.SafetyPoCr.Position = [405 49 30 22];
            app.SafetyPoCr.Value = 20;

            % Create DoDLabel
            app.DoDLabel = uilabel(app.BatteryRequirementsPanel);
            app.DoDLabel.HorizontalAlignment = 'right';
            app.DoDLabel.Position = [307 13 43 22];
            app.DoDLabel.Text = 'DoD %';

            % Create DischargeDepth
            app.DischargeDepth = uieditfield(app.BatteryRequirementsPanel,
'numeric');
            app.DischargeDepth.Position = [361 13 74 22];
            app.DischargeDepth.Value = 80;

            % Create CalculateBatteryButton
            app.CalculateBatteryButton = uibutton(app.BatteryProfileTab,
'push');
            app.CalculateBatteryButton.ButtonPushedFcn =
createCallbackFcn(app, @CalculateBatteryButtonPushed, true);
            app.CalculateBatteryButton.BackgroundColor = [0.0745 0.6235 1];
            app.CalculateBatteryButton.FontColor = [1 1 1];
            app.CalculateBatteryButton.Position = [412 676 64 22];
            app.CalculateBatteryButton.Text = 'Calculate';

            % Create PowerDensePackAuxPanel
            app.PowerDensePackAuxPanel = uipanel(app.BatteryProfileTab);
            app.PowerDensePackAuxPanel.Title = 'Power Dense Pack - Aux';
            app.PowerDensePackAuxPanel.BackgroundColor = [0.9412 0.9412
0.9412];
            app.PowerDensePackAuxPanel.Position = [247 309 230 154];

            % Create BatteryNameEditFieldLabel
            app.BatteryNameEditFieldLabel =
uilabel(app.PowerDensePackAuxPanel);
            app.BatteryNameEditFieldLabel.HorizontalAlignment = 'right';
            app.BatteryNameEditFieldLabel.Position = [1 99 79 22];
            app.BatteryNameEditFieldLabel.Text = 'Battery Name';

            % Create AuxBatteryName
            app.AuxBatteryName = uieditfield(app.PowerDensePackAuxPanel,
'text');
            app.AuxBatteryName.Editable = 'off';
            app.AuxBatteryName.Position = [88 99 131 22];

            % Create CapacityEditFieldLabel
            app.CapacityEditFieldLabel = uilabel(app.PowerDensePackAuxPanel);
            app.CapacityEditFieldLabel.HorizontalAlignment = 'right';
            app.CapacityEditFieldLabel.Position = [26 71 52 22];
            app.CapacityEditFieldLabel.Text = 'Capacity';

            % Create AuxCapacity
            app.AuxCapacity = uieditfield(app.PowerDensePackAuxPanel,
'numeric');
            app.AuxCapacity.ValueDisplayFormat = '%11.4gkWh';
            app.AuxCapacity.Editable = 'off';
            app.AuxCapacity.Position = [88 71 67 22];

            % Create ConstantEditFieldLabel
            app.ConstantEditFieldLabel = uilabel(app.PowerDensePackAuxPanel);
            app.ConstantEditFieldLabel.HorizontalAlignment = 'right';
            app.ConstantEditFieldLabel.Position = [1 10 54 22];
            app.ConstantEditFieldLabel.Text = 'Constant';

            % Create AuxConstantPower
            app.AuxConstantPower = uieditfield(app.PowerDensePackAuxPanel,
'numeric');
            app.AuxConstantPower.ValueDisplayFormat = '%11.4gkW';
            app.AuxConstantPower.Editable = 'off';
            app.AuxConstantPower.Position = [60 11 56 22];

            % Create BurstEditFieldLabel
            app.BurstEditFieldLabel = uilabel(app.PowerDensePackAuxPanel);
            app.BurstEditFieldLabel.HorizontalAlignment = 'right';
            app.BurstEditFieldLabel.Position = [126 10 34 22];
            app.BurstEditFieldLabel.Text = 'Burst';

            % Create AuxBurstPower
            app.AuxBurstPower = uieditfield(app.PowerDensePackAuxPanel,
'numeric');
            app.AuxBurstPower.ValueDisplayFormat = '%11.4gkW';
            app.AuxBurstPower.Editable = 'off';
            app.AuxBurstPower.Position = [168 10 50 22];

            % Create WeightLabel_2
            app.WeightLabel_2 = uilabel(app.PowerDensePackAuxPanel);
            app.WeightLabel_2.HorizontalAlignment = 'right';
            app.WeightLabel_2.Position = [34 41 43 22];
            app.WeightLabel_2.Text = 'Weight';

            % Create AuxWeight
            app.AuxWeight = uieditfield(app.PowerDensePackAuxPanel,
'numeric');
            app.AuxWeight.ValueDisplayFormat = '%11.4gkg';
            app.AuxWeight.Editable = 'off';
            app.AuxWeight.Position = [87 41 67 22];

            % Create AddListButton
            app.AddListButton = uibutton(app.BatteryProfileTab, 'push');
            app.AddListButton.ButtonPushedFcn = createCallbackFcn(app,
@AddListButtonPushed, true);
            app.AddListButton.BackgroundColor = [0.0745 0.6235 1];
            app.AddListButton.FontColor = [1 1 1];
            app.AddListButton.Position = [97 676 64 22];
            app.AddListButton.Text = 'Add';

            % Create EnergyDensePackMainPanel_2
            app.EnergyDensePackMainPanel_2 = uipanel(app.BatteryProfileTab);
            app.EnergyDensePackMainPanel_2.Title = 'Energy Dense Pack -
Main';
            app.EnergyDensePackMainPanel_2.BackgroundColor = [0.9412 0.9412
0.9412];
            app.EnergyDensePackMainPanel_2.Position = [19 309 226 154];

            % Create BatteryNameEditField_2Label
            app.BatteryNameEditField_2Label =
uilabel(app.EnergyDensePackMainPanel_2);
            app.BatteryNameEditField_2Label.HorizontalAlignment = 'right';
            app.BatteryNameEditField_2Label.Position = [1 99 79 22];
            app.BatteryNameEditField_2Label.Text = 'Battery Name';

            % Create MainBatteryName
            app.MainBatteryName = uieditfield(app.EnergyDensePackMainPanel_2,
'text');
            app.MainBatteryName.Editable = 'off';
            app.MainBatteryName.Position = [88 99 131 22];

            % Create CapacityEditField_2Label
            app.CapacityEditField_2Label =
uilabel(app.EnergyDensePackMainPanel_2);
            app.CapacityEditField_2Label.HorizontalAlignment = 'right';
            app.CapacityEditField_2Label.Position = [26 71 52 22];
            app.CapacityEditField_2Label.Text = 'Capacity';

            % Create MainCapacity
            app.MainCapacity = uieditfield(app.EnergyDensePackMainPanel_2,
'numeric');
            app.MainCapacity.ValueDisplayFormat = '%11.4gkWh';
            app.MainCapacity.Editable = 'off';
            app.MainCapacity.Position = [88 71 67 22];

            % Create ConstantEditField_2Label
            app.ConstantEditField_2Label =
uilabel(app.EnergyDensePackMainPanel_2);
            app.ConstantEditField_2Label.HorizontalAlignment = 'right';
            app.ConstantEditField_2Label.Position = [1 10 54 22];
            app.ConstantEditField_2Label.Text = 'Constant';

            % Create MainConstantPower
            app.MainConstantPower =
uieditfield(app.EnergyDensePackMainPanel_2, 'numeric');
            app.MainConstantPower.ValueDisplayFormat = '%11.4gkW';
            app.MainConstantPower.Editable = 'off';
            app.MainConstantPower.Position = [62 10 56 22];

            % Create BurstEditField_2Label
            app.BurstEditField_2Label =
uilabel(app.EnergyDensePackMainPanel_2);
            app.BurstEditField_2Label.HorizontalAlignment = 'right';
            app.BurstEditField_2Label.Position = [127 10 34 22];
            app.BurstEditField_2Label.Text = 'Burst';

            % Create MainBurstPower
            app.MainBurstPower = uieditfield(app.EnergyDensePackMainPanel_2,
'numeric');
            app.MainBurstPower.ValueDisplayFormat = '%11.4gkW';
            app.MainBurstPower.Editable = 'off';
            app.MainBurstPower.Position = [169 10 50 22];

            % Create WeightLabel
            app.WeightLabel = uilabel(app.EnergyDensePackMainPanel_2);
            app.WeightLabel.HorizontalAlignment = 'right';
            app.WeightLabel.Position = [35 41 43 22];
            app.WeightLabel.Text = 'Weight';
```

```
            % Create MainWeight
            app.MainWeight = uieditfield(app.EnergyDensePackMainPanel_2,
'numeric');
            app.MainWeight.ValueDisplayFormat = '%11.4gkg';
            app.MainWeight.Editable = 'off';
            app.MainWeight.Position = [88 41 67 22];

            % Create ResultsPanel
            app.ResultsPanel = uipanel(app.BatteryProfileTab);
            app.ResultsPanel.Title = 'Results';
            app.ResultsPanel.Position = [248 140 229 160];

            % Create BatterytoAicraftWeightRatioGaugeLabel
            app.BatterytoAicraftWeightRatioGaugeLabel =
uilabel(app.ResultsPanel);
            app.BatterytoAicraftWeightRatioGaugeLabel.HorizontalAlignment =
'center';
            app.BatterytoAicraftWeightRatioGaugeLabel.Position = [33 6 166
22];
            app.BatterytoAicraftWeightRatioGaugeLabel.Text = 'Battery to
Aicraft Weight Ratio';

            % Create BatteryWeightRatio
            app.BatteryWeightRatio = uigauge(app.ResultsPanel,
'ninetydegree');
            app.BatteryWeightRatio.Position = [33 30 97 97];

            % Create GoodLampLabel
            app.GoodLampLabel = uilabel(app.ResultsPanel);
            app.GoodLampLabel.HorizontalAlignment = 'right';
            app.GoodLampLabel.Enable = 'off';
            app.GoodLampLabel.Position = [180 102 35 22];
            app.GoodLampLabel.Text = 'Good';

            % Create GoodLamp
            app.GoodLamp = uilamp(app.ResultsPanel);
            app.GoodLamp.Enable = 'off';
            app.GoodLamp.Position = [157 103 20 20];

            % Create OkayLampLabel
            app.OkayLampLabel = uilabel(app.ResultsPanel);
            app.OkayLampLabel.HorizontalAlignment = 'right';
            app.OkayLampLabel.Enable = 'off';
            app.OkayLampLabel.Position = [181 64 34 22];
            app.OkayLampLabel.Text = 'Okay';

            % Create OkayLamp
            app.OkayLamp = uilamp(app.ResultsPanel);
            app.OkayLamp.Enable = 'off';
            app.OkayLamp.Position = [157 65 20 20];
            app.OkayLamp.Color = [1 1 0];

            % Create BadLampLabel
            app.BadLampLabel = uilabel(app.ResultsPanel);
            app.BadLampLabel.HorizontalAlignment = 'right';
            app.BadLampLabel.Enable = 'off';
            app.BadLampLabel.Position = [188 30 27 22];
            app.BadLampLabel.Text = 'Bad';

            % Create BadLamp
            app.BadLamp = uilamp(app.ResultsPanel);
            app.BadLamp.Enable = 'off';
            app.BadLamp.Position = [157 31 20 20];
            app.BadLamp.Color = [1 0 0];

            % Create HybridSelected
            app.HybridSelected = uibutton(app.ResultsPanel, 'state');
            app.HybridSelected.ValueChangedFcn = createCallbackFcn(app,
@HybridSelectedValueChanged, true);
            app.HybridSelected.Text = 'Hybrid';
            app.HybridSelected.BackgroundColor = [0.302 0.7451 0.9333];
            app.HybridSelected.FontColor = [1 1 1];
            app.HybridSelected.Position = [8 105 46 22];

            % Create BatteryListLabel
            app.BatteryListLabel = uilabel(app.BatteryProfileTab);
            app.BatteryListLabel.Position = [20 676 66 22];
            app.BatteryListLabel.Text = 'Battery List';

            % Create RefPackPanel
            app.RefPackPanel = uipanel(app.BatteryProfileTab);
            app.RefPackPanel.Title = 'Single Pack';
            app.RefPackPanel.BackgroundColor = [0.9412 0.9412 0.9412];
            app.RefPackPanel.Position = [20 140 226 160];

            % Create BatteryNameEditField_2Label_2
            app.BatteryNameEditField_2Label_2 = uilabel(app.RefPackPanel);
            app.BatteryNameEditField_2Label_2.HorizontalAlignment = 'right';
            app.BatteryNameEditField_2Label_2.Position = [1 105 79 22];
            app.BatteryNameEditField_2Label_2.Text = 'Battery Name';

            % Create RefBatteryName
            app.RefBatteryName = uieditfield(app.RefPackPanel, 'text');
            app.RefBatteryName.Editable = 'off';
            app.RefBatteryName.Position = [88 105 131 22];

            % Create CapacityEditField_2Label_2
            app.CapacityEditField_2Label_2 = uilabel(app.RefPackPanel);
            app.CapacityEditField_2Label_2.HorizontalAlignment = 'right';
            app.CapacityEditField_2Label_2.Position = [26 77 52 22];
            app.CapacityEditField_2Label_2.Text = 'Capacity';

            % Create RefCapacity
            app.RefCapacity = uieditfield(app.RefPackPanel, 'numeric');
            app.RefCapacity.ValueDisplayFormat = '%11.4gkWh';
            app.RefCapacity.Editable = 'off';
            app.RefCapacity.Position = [88 77 67 22];

            % Create ConstantEditField_2Label_2
            app.ConstantEditField_2Label_2 = uilabel(app.RefPackPanel);
            app.ConstantEditField_2Label_2.HorizontalAlignment = 'right';
            app.ConstantEditField_2Label_2.Position = [1 16 54 22];
            app.ConstantEditField_2Label_2.Text = 'Constant';

            % Create RefConstantPower
            app.RefConstantPower = uieditfield(app.RefPackPanel, 'numeric');
            app.RefConstantPower.ValueDisplayFormat = '%11.4gkW';
            app.RefConstantPower.Editable = 'off';
            app.RefConstantPower.Position = [62 16 56 22];

            % Create BurstEditField_2Label_2
            app.BurstEditField_2Label_2 = uilabel(app.RefPackPanel);
            app.BurstEditField_2Label_2.HorizontalAlignment = 'right';
            app.BurstEditField_2Label_2.Position = [127 16 34 22];
            app.BurstEditField_2Label_2.Text = 'Burst';

            % Create RefBurstPower
            app.RefBurstPower = uieditfield(app.RefPackPanel, 'numeric');
            app.RefBurstPower.ValueDisplayFormat = '%11.4gkW';
            app.RefBurstPower.Editable = 'off';
            app.RefBurstPower.Position = [169 16 50 22];

            % Create WeightLabel_3
            app.WeightLabel_3 = uilabel(app.RefPackPanel);
            app.WeightLabel_3.HorizontalAlignment = 'right';
            app.WeightLabel_3.Position = [35 47 43 22];
            app.WeightLabel_3.Text = 'Weight';

            % Create RefWeight
            app.RefWeight = uieditfield(app.RefPackPanel, 'numeric');
            app.RefWeight.ValueDisplayFormat = '%11.4gkg';
            app.RefWeight.Editable = 'off';
            app.RefWeight.Position = [88 47 67 22];

            % Create BatteryWeightComparaisonPanel
            app.BatteryWeightComparaisonPanel =
uipanel(app.BatteryProfileTab);
            app.BatteryWeightComparaisonPanel.Title = 'Battery Weight
Comparaison';
            app.BatteryWeightComparaisonPanel.BackgroundColor = [0.9412
0.9412 0.9412];
            app.BatteryWeightComparaisonPanel.Position = [19 12 458 119];

            % Create HybridLabel
            app.HybridLabel = uilabel(app.BatteryWeightComparaisonPanel);
            app.HybridLabel.HorizontalAlignment = 'right';
            app.HybridLabel.Position = [9 71 40 22];
            app.HybridLabel.Text = 'Hybrid';

            % Create HybridWeight
            app.HybridWeight = uieditfield(app.BatteryWeightComparaisonPanel,
'numeric');
            app.HybridWeight.ValueDisplayFormat = '%11.4gkg';
            app.HybridWeight.Position = [60 71 74 22];

            % Create SingleLabel
            app.SingleLabel = uilabel(app.BatteryWeightComparaisonPanel);
            app.SingleLabel.HorizontalAlignment = 'right';
            app.SingleLabel.Position = [140 71 39 22];
            app.SingleLabel.Text = 'Single';

            % Create RefWeight2
            app.RefWeight2 = uieditfield(app.BatteryWeightComparaisonPanel,
'numeric');
            app.RefWeight2.ValueDisplayFormat = '%11.4gkg';
            app.RefWeight2.Position = [190 71 74 22];

            % Create DifferenceLabel
            app.DifferenceLabel = uilabel(app.BatteryWeightComparaisonPanel);
            app.DifferenceLabel.HorizontalAlignment = 'right';
            app.DifferenceLabel.Position = [296 71 60 22];
            app.DifferenceLabel.Text = 'Difference';

            % Create DifferenceWeight
            app.DifferenceWeight =
uieditfield(app.BatteryWeightComparaisonPanel, 'numeric');
            app.DifferenceWeight.ValueDisplayFormat = '%11.4gkg';
            app.DifferenceWeight.Position = [367 71 74 22];

            % Create HybridLabel_2
            app.HybridLabel_2 = uilabel(app.BatteryWeightComparaisonPanel);
            app.HybridLabel_2.HorizontalAlignment = 'right';
            app.HybridLabel_2.Position = [9 41 40 22];
            app.HybridLabel_2.Text = 'Hybrid';

            % Create HybridCapacity
            app.HybridCapacity =
uieditfield(app.BatteryWeightComparaisonPanel, 'numeric');
            app.HybridCapacity.ValueDisplayFormat = '%11.4gkWh';
            app.HybridCapacity.Position = [60 41 74 22];

            % Create SingleLabel_2
            app.SingleLabel_2 = uilabel(app.BatteryWeightComparaisonPanel);
            app.SingleLabel_2.HorizontalAlignment = 'right';
            app.SingleLabel_2.Position = [140 41 39 22];
            app.SingleLabel_2.Text = 'Single';

            % Create RefCapacity2
            app.RefCapacity2 = uieditfield(app.BatteryWeightComparaisonPanel,
'numeric');
            app.RefCapacity2.ValueDisplayFormat = '%11.4gkWh';
            app.RefCapacity2.Position = [190 41 74 22];

            % Create DifferenceLabel_2
            app.DifferenceLabel_2 =
uilabel(app.BatteryWeightComparaisonPanel);
            app.DifferenceLabel_2.HorizontalAlignment = 'right';
            app.DifferenceLabel_2.Position = [296 41 60 22];
            app.DifferenceLabel_2.Text = 'Difference';

            % Create DifferenceCapacity
            app.DifferenceCapacity =
uieditfield(app.BatteryWeightComparaisonPanel, 'numeric');
            app.DifferenceCapacity.ValueDisplayFormat = '%11.4gkWh';
            app.DifferenceCapacity.Position = [367 41 74 22];

            % Create HybridLabel_3
            app.HybridLabel_3 = uilabel(app.BatteryWeightComparaisonPanel);
            app.HybridLabel_3.HorizontalAlignment = 'right';
            app.HybridLabel_3.Position = [9 9 40 22];
            app.HybridLabel_3.Text = 'Hybrid';

            % Create HybridEDensity
            app.HybridEDensity =
uieditfield(app.BatteryWeightComparaisonPanel, 'numeric');
            app.HybridEDensity.ValueDisplayFormat = '%11.4gWh/kg';
            app.HybridEDensity.Position = [60 9 74 22];

            % Create SingleLabel_3
            app.SingleLabel_3 = uilabel(app.BatteryWeightComparaisonPanel);
            app.SingleLabel_3.HorizontalAlignment = 'right';
            app.SingleLabel_3.Position = [140 9 39 22];
            app.SingleLabel_3.Text = 'Single';

            % Create RefEDensity
            app.RefEDensity = uieditfield(app.BatteryWeightComparaisonPanel,
'numeric');
            app.RefEDensity.ValueDisplayFormat = '%11.4gWh/kg';
            app.RefEDensity.Position = [190 9 74 22];

            % Create DifferenceLabel_3
            app.DifferenceLabel_3 =
uilabel(app.BatteryWeightComparaisonPanel);
            app.DifferenceLabel_3.HorizontalAlignment = 'right';
            app.DifferenceLabel_3.Position = [296 9 60 22];
            app.DifferenceLabel_3.Text = 'Difference';

            % Create DifferenceEDensity
            app.DifferenceEDensity =
uieditfield(app.BatteryWeightComparaisonPanel, 'numeric');
            app.DifferenceEDensity.ValueDisplayFormat = '%11.4gWh/kg';
            app.DifferenceEDensity.Position = [367 9 74 22];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = HybridConfig

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
en
```