UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

# Real-Time Estimation of Remaining UAV Flight Time Based on the Total Energy Balance

**(versão final após defesa)**

**Pedro Alexandre Lourenço Moutinho**

Dissertação para obtenção de Grau de Mestre em
**Engenharia Aeronáutica**
(Ciclo de estudos integrado)

Orientador: Prof. Doutor Pedro Vieira Gamboa

**Covilhã, Novembro de 2017**

# Acknowledgements

This thesis would not have been possible without the invaluable contribution of several people to whom I wish to express my sincere gratitude.

First of all, I would like to express my sincere thanks to my supervisor, Professor Pedro Gamboa, for his support and patience. He will remain in my memory as one of my main academic and human references that I have had the opportunity to meet. I also would like to thank for his knowledge shared with me throughout this work.

I would like to thank to all my friends and colleagues for those moments that will be kept forever. To Pedro Santos that had an essential role in the success of this thesis. To Pedro Alves who likewise encouraged me and gave me a vote of confidence to assist him in the growth of his start-up.

To all my family for their unconditional support despite the distance that separates us. In particular to my parents, Ana and Idálio, who regardless of all the difficulties have always believed in me and let me demonstrate they were right when decided to support me on this long journey. With great satisfaction I dedicate this work to them.

To Tiago Nunes and António Relvas for their vote of confidence when I entered into the world of work for the first time. A big thank you to all Tekever team for their tremendous support from day one.

Finally, but always in my heart since the freshman days I would like to thank Alexandrina Oliveira for her enormous support. Words are powerless to express my gratitude. Thank you for your smile.

# Resumo

Os veículos aéreos não tripulados (UAV's), utilizados inicialmente para aplicações militares, têm-se tornado cada vez mais atrativos para fins civis. O recurso a este tipo de aeronaves tem crescido exponencialmente nos últimos anos, quer para fins profissionais como recreativos, devido às inúmeras vantagens que estes apresentam. No entanto, de um modo geral os UAV's são desprovidos de mecanismos de segurança essenciais à sua operação, pelo que se revela de extrema importância definir/implementar soluções para o aumento da segurança deste tipo de equipamentos. Deste modo e integrado no projeto "Drones' Safe Flight - Gestão de energia, planeamento de missão e deteção de obstáculos em voo" o presente trabalho descreve o desenvolvimento e validação de um algoritmo implementado numa plataforma de código aberto para veículos aéreos não tripulados (Ardupilot) capaz de estimar em tempo real o balanço energético de uma aeronave remotamente pilotada (RPA), tendo em vista o cálculo do tempo remanescente (RRT) de voo. Com esse propósito, numa fase inicial procedeu-se à formulação energética associada ao tipo de aeronaves em estudo. Após descrita a componente teórica presente no algoritmo, abordou-se a metodologia e estrutura onde reproduziu-se a sequência de eventos necessários à estimativa do estado energético. Para além das capacidades referidas anteriormente, foi levada a cabo a implementação da transmissão de dados relevantes à operação da aeronave via telemetria e consequente adição na *ground station* sob a forma numérica e gráfica. Complementando os pacotes de dados enviados via telemetria (MAVLINK), procedeu-se à implementação da transmissão do vídeo *on-board* ao qual foram adicionadas mensagens com indicação do tempo remanescente de voo. Relativamente à validação do algoritmo integrou-se todos os sistemas necessários em duas aeronaves distintas, tendo cada uma efetuado vários voos de modo avaliar o comportamento do conjunto aeronave/algoritmo. Por fim, apresentou-se os resultados que mostraram uma boa capacidade de resposta em situações transitórias, bem como na atenuação do ruído proveniente dos sensores.

# Palavras-chave

Veículos Aéreos não Tripulados, Balanço Energético, Tempo Remanescente de Voo, Plataforma de Código Aberto, Integração de Sistemas

# Abstract

Unmanned aerial vehicles (UAV's), initially used for military applications, have become increasingly attractive for civilian purposes. The use of this type of aircraft has grown exponentially in recent years, both for professional and recreational purposes, due to the numerous advantages they present. However, in general UAV's are devoid of safety mechanisms essential to their operation, so it is extremely important to define/implement solutions to increase the safety of this type of equipment. This way and integrated in the "Drones' Safe Flight - Gestão de energia, planeamento de missão e deteção de obstáculos em voo" project the present work describes the development and validation of an algorithm implemented in an open source platform for unmanned aerial vehicles (Ardupilot) capable of estimating in real time the energy balance of a remotely piloted aircraft (RPA), in order to calculate the remaining run time (RRT). For this, at an initial phase the energy formulation associated to the type of aircraft under study was carried out. After describing the theoretical component present in the algorithm, the methodology and structure were approached where the sequence of events necessary to the estimation of the energy state was reproduced. In addition to the aforementioned capabilities, the transmission of relevant data for the operation of the aircraft via telemetry and consequent addition to the ground station into a numerical and graphical way was carried out. Complementing the data packets sent via telemetry (MAVLINK), the on-board video streaming was implemented and messages with indication of remaining flight time were added. Regarding the validation of the algorithm, all the necessary systems were integrated into two distinct aircraft, each having performed several flights in order to evaluate the behaviour of the group aircraft/algorithm. Finally, the results were presented which have shown a good response capacity in transient situations, as well as the attenuation of the noise coming from the sensors.

# Keywords

Unmanned Aerial Vehicles, Energy Balance, Remaining Flight Time, Open Source Platform, Systems Integration

# Contents

# List of Figures

# List of Tables

# Nomenclature

## Greek Terms

| | |
|---|---|
| $\Delta E_{mechanical}$ | change of mechanical energy |
| $\Delta E_{surroundings}$ | surroundings change of energy |
| $\Delta E_{system}$ | system change of energy |
| $\Delta E_{total}$ | total change of energy |
| $\Delta E^{total}_{mechanical}$ | total change of mechanical energy |
| $\Delta E^{total}_{mechanical(slf)}$ | total change of mechanical energy, steady level flight |
| $\Delta h$ | h component variation |
| $\Delta K$ | change of kinetic energy |
| $\Delta K_{system}$ | system change of kinetic energy |
| $\Delta U$ | change of potential energy |
| $\Delta U_{system}$ | system change of potential energy |
| $\Delta U^{total}$ | change of total potential energy |
| $\Delta z$ | z component variation |
| $\eta$ | effect of the battery overpotential |
| $\eta_m$ | error source in the overpotential model |

## Latin Terms

| | |
|---|---|
| $C_{B1max}$ | maximum energy capacity of a battery element |
| $C_{Bmax}$ | maximum energy capacity of a battery pack |
| $C_N$ | battery rated capacity |
| $E_B$ | battery remaining energy |
| $E_e$ | irradiance received by a surface per unit area |
| $E_{mechanical}$ | mechanical energy |
| $E_{mechanical,0}$ | mechanical energy, initial |
| $E_{mechanical,f}$ | mechanical energy, final |
| $EMF_{dt}$ | EMF detection method |
| $EMF_f$ | EMF fitting method |
| $EMF_m$ | ration between SoC/battery EMF |
| $E_{motor(slf)}$ | electrical energy, steady level flight |
| $E_{Smax}$ | maximum energy delivered by the solar panel |
| $\vec{F}_{cons,i}$ | conservative forces acting on a system |
| $\vec{F}^{g}$ | gravitational force acting on a object |
| $\vec{F}_{non-cons,j}$ | non-conservative forces acting on a system |
| $\vec{F}^{total}$ | total force acting on a system |
| $\vec{F}^{total}_{non-cons}$ | total non conservative force acting on a system |

| | |
|---|---|
| $F_x$ | component of the force along the x-direction |
| $h$ | heigh |
| $h_f$ | final heigh |
| $h_i$ | initial heigh |
| $I_B$ | battery current |
| $I_b$ | backlight-on state current |
| $I_{batt}$ | battery current |
| $I_{Bmax}$ | maximum discharge current of a battery pack |
| $I_{Bmax\_chg}$ | maximum charge current of a battery pack |
| $I_{ch}$ | battery charge current |
| $I_d$ | battery discharge current |
| $I_{lim}$ | battery current limit |
| $I_{loss}$ | battery current losses |
| $I_M$ | current (battery tester) |
| $I_S$ | maximum current provided by the solar panel |
| $I_{sc}$ | short circuit current |
| $I_{Smax}$ | maximum power point current |
| $I_s$ | drawn/received current in standby state |
| $\hat{j}$ | $\hat{j}$ component |
| $K$ | kinetic energy |
| $K_0$ | kinetic energy, initial |
| $K_B$ | discharge rate |
| $K_f$ | kinetic energy, final |
| $m$ | body mass |
| $n_B$ | total number of battery elements |
| $n_{BP}$ | number of battery elements connected in parallel |
| $n_{BS}$ | number of battery elements connected in series |
| $n_S$ | total number of solar cells |
| $n_{SP}$ | number of solar cells connected in parallel |
| $n_{SS}$ | number of solar cells connected in series |
| $P_B$ | power delivered by the battery |
| $P_{Smax}$ | maximum power delivered by a solar panel |
| $Q_{max}$ | maximum capacity of a rechargeable battery |
| $\vec{r}$ | distance vector |
| $SoC_0$ | initial SoC point |
| $SoC_b$ | state of charge: backlight-on state |
| $SoC_ch$ | state of charge: charge state |
| $SoC_d$ | state of charge: discharge state |
| $SoC_i$ | state of charge: initial state |
| $SoC_l$ | battery overpotential in $V$ translated into SoC |
| $SoC_s$ | state of charge: standby state |
| $SoC_t$ | state of charge: transitional state |

| | |
|---|---|
| $T_{batt}$ | battery temperature |
| $t$ | time |
| $t_0$ | initial time instant |
| $t_i$ | time instant |
| $t_{i+1}$ | next time instant |
| $t_M$ | time instant (battery tester) |
| $T_M$ | temperature (battery tester) |
| $t_r$ | time remaining |
| $U$ | potential energy |
| $U_0$ | potential energy, initial |
| $U_B$ | battery voltage |
| $U_{B1max}$ | maximum voltage of a battery element |
| $U_{B1min}$ | minimum voltage of a battery element |
| $U_{Bmax}$ | maximum voltage of a battery pack |
| $U_{Bmin}$ | minimum voltage of a battery pack |
| $U_f$ | potential energy, final |
| $U_{S1max}$ | maximum power point voltage of a single solar cell |
| $U_{SCend}$ | solar charger end voltage |
| $U_{Smax}$ | maximum power point voltage |
| $V$ | voltage |
| $\overrightarrow{v}_0$ | initial position velocity |
| $V_M$ | voltage (battery tester) |
| $\overrightarrow{v}_f$ | final position velocity |
| $V_{OC}$ | open circuit voltage |
| $v_x$ | velocity in the x-axis |
| $v_{x,0}$ | initial velocity in the x-axis |
| $v_{x,0}\hat{i}$ | initial velocity in the x-axis, $\hat{i}$ component |
| $v_{x,f}$ | final velocity in the x-axis |
| $v_{x,f}\hat{i}$ | final velocity in the x-axis, $\hat{i}$ component |
| $v_{y,0}\hat{j}$ | initial velocity in the y-axis, $\hat{j}$ component |
| $v_{y,f}\hat{j}$ | final velocity in the y-axis, $\hat{j}$ component |
| $v_{z,0}\hat{k}$ | initial velocity in the z-axis, $\hat{k}$ component |
| $v_{z,f}\hat{k}$ | final velocity in the z-axis, $\hat{k}$ component |
| $W$ | work |
| $W_{cons}$ | work of conservative forces |
| $W_{drag(slf)}$ | work of drag forces, steady level flight |
| $W^g$ | work done by the gravitational force acting on a object |
| $W_{non-cons}$ | work of non conservative forces |
| $W_{total}$ | total work |
| $x$ | x-axis |
| $x_0$ | initial position along x-axis |

| | |
|---|---|
| $x_f$ | final position along x-axis |
| $z$ | z-axis |
| $z_i$ | initial heigh |
| $z_f$ | final heigh |

## Latin Terms: Computer Code

| | |
|---|---|
| `aircraft_mass` | aircraft gross mas |
| `battery_cap` | real battery capacity |
| `battery.current_amps()` | output battery current (main) |
| `battery.current_amps(1)` | output battery current (secondary) |
| `battery_int_ressist` | internal battery resistance |
| `battery_number` | battery number |
| `battery.voltage()` | battery voltage (main) |
| `battery.voltage(1)` | battery voltage (secondary) |
| `batt_charge_v` | real battery voltage when charging |
| `barometer_alt_last` | last value of relative altitude |
| `barometer.get_altitude()` | relative altitude from airborne to the earth surface (main) |
| `benergy_max` | maximum battery energy (full charge) |
| `benergy_recalc` | real battery energy correction |
| `benergy_update` | battery energy update |
| `charg_trem` | charging time remaining |
| `chg_trem` | charging time remaining (after filter) |
| `chgtrem_filter` | simple moving arithmetic mean of `charg_trem` |
| `consumed_capacity` | consumed capacity |
| `curr_dscg` | discharge current for battery number 3 (discharge test) |
| `delta_v/delta_t` | ground speed derivation with respect to time |
| `deltk_filter` | simple moving arithmetic mean of `k_power` |
| `deltp_filter` | simple moving arithmetic mean of `p_power` |
| `dt_hz` | algorithm update rate |
| `energy_consumed` | electric energy consumed |
| `energy_received` | electric energy received |
| `gps.ground_speed()` | ground speed (main) |
| `ground_spd_last` | last value of ground speed |
| `GRAVITY_MSS` | gravitational acceleration |
| `k_energy` | kinetic energy |
| `k_power` | kinetic power |
| `k_power_avg` | kinetic power (damped) |
| `mtr` | mission time remaining |
| `mtr_max` | maximum mission time remaining |

| | |
|---|---|
| `param_cap` | battery capacity and solar charger end voltage correlation |
| `p_energy` | potential energy |
| `plane.is_flying()` | condition (if plane is flying) |
| `power_load` | battery power output |
| `p_power` | potential power |
| `p_power_avg` | potential power (damped) |
| `R_Climb` | rate of climb |
| `real_capacity` | real battery capacity after correction |
| `solar_charge_vend` | end of charge voltage from solar charger |
| `solar_panels` | solar panels variable |
| `solar_power` | solar power |
| `time_rem` | remaining flight time (after filter) |
| `tpavg_filter` | simple moving arithmetic mean of `total_power` |
| `total_energy` | total energy of the system |
| `total_power` | total power of the system |
| `trem_filter` | simple moving arithmetic mean of `mtr` |
| `var_charg` | charge state |
| `var_mtr` | discharge state |

## Acronyms

| | |
|---|---|
| AeroG | Aeronautics and Astronautics Research Center |
| AHRS | Attitude Heading Reference System |
| AoA | Angle of Attack |
| CC | Coulomb Counting |
| CTR | Charging Time Remaining |
| DC | Direct Current |
| DoF | Degrees of Freedom |
| ECEF | Earth Centered Earth Fixed |
| EKF | Extended Kalmam Filter |
| EMF | Electromotive Force |
| ESC | Electronic Speed Controller |
| FBWA | Fly By Wire_A |
| FEM | Flight Energy Management |
| FPV | First Person View |
| FUDS | Federal Urban Driving Schedule |
| GCS | Ground Control Station |
| GCSS | Ground Control Station Software |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |

| | |
|---|---|
| HAL | Hardware Abstraction Layer |
| IC | Internal Combustion |
| IDE | Integrated Development Environment |
| IDMEC | Institute of Mechanical Engineering |
| ILS | Instrument Landing System |
| IMU | Inertial Measurement Unit |
| INEGI | Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial |
| I/O | Input/Output |
| IST | Instituto Superior Técnico |
| KF | Kalmam Filter |
| LAETA | Associated Laboratory for Energy, Transports and Aeronautics |
| LALE | Low-Altitude Long-Endurance |
| LEEUAV | Long Endurance Electric Unmanned Aerial Vehicle |
| LGPL | Lesser General Public License |
| LiPo | Lithium-ion Polymer |
| MAVLINK | Micro Air Vehicle Link |
| MPP | Maximum Power Point |
| MPPT | Maximum Power Point Tracking |
| MTR | Mission Time Remaining |
| OCV | Open Circuit Voltage |
| OMP | Online Mission Planning |
| OSD | On-Screen Display |
| POSIX OS | Portable Operating System Interface |
| R&D | Research and Development |
| RAM | Random Access Memory |
| RAMP | Range-Amplified MultiPoint |
| RF | Radio Frequency |
| ROV | Remotely Operated Vehicle |
| RPA | Remotely Piloted Aircraft |
| RPAS | Remotely Piloted Aircraft System |
| RRT | Remaining Run Time |
| RSSI | Received Signal Strength Indication |
| RTL | Return to Launch |
| S&A | Sense and Avoid |
| SCC | Solar Charge Controller |
| SLF | Steady Level Flight |
| SoC | State of Charge |
| FBWA | Fly By Wire_A |
| UAV | Unmanned Aerial Vehicle |
| UBEC | Universal Battery Elimination Circuit |
| UBI | Universidade da Beira Interior |
| VDC | Volts of Direct Current |
| XML | eXtensible Markup Language |

# Chapter 1

# Introduction

## 1.1 Contextualization

The present work is integrated in the "Drones' Safe Flight - Gestão de energia, planeamento de missão e deteção de obstáculos em voo" project. A project of research and development (R&D) of LAETA (Associated Laboratory for Energy, Transports and Aeronautics) where three institutions cooperate together, Universidade da Beira Interior (UBI) with the Aeronautics and Astronautics Research Center (AeroG), Instituto Superior Técnico (IST) with the Institute of Mechanical Engineering (IDMEC) and Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial (INEGI), whose main challenge is the development of safety system solutions for implementation in the emerging market of small drones. The proposed research is divided into three major topics, Fig.1.1:

- **Flight Energy Management (FEM)**
    1) Experimental aircraft characterization, through laboratory, wind tunnel and flight tests;
    2) Environmental conditions estimation, which depending on the chosen aircraft type may result in the development of wind, solar exposure and thermal estimators;
    3) Estimation of the available energy/power to perform a desired task, considering environmental conditions and the remaining mission to be executed.
- **Online Mission Planning (OMP)**
    1) Take preventive and corrective actions over the course of the mission due to changes in operating conditions;
    2) Development of a Ground Control Station (GCS) interface to create flights plans, monitor the mission progress, receive information from on-board sensors and change the mission profile in real time.
- **Sense and Avoid (S&A)**
    1) Study of different types of sensors (ultrasound, infra-red, Laser), their response time and range;
    2) Conceptual and detailed definition of different detection systems, taking into account different sensors and configurations;
    3) Implementation of a system that is capable to alert the pilot about obstacles in the flight path and take evasive actions to avoid collisions (or minimize the possibility).

This thesis focuses on the third task of the Flight Energy Management (FEM) topic. However, due to the complexity of the proposed problem this task has been simplified to real-time estimation of the available energy/power to perform a desired task.

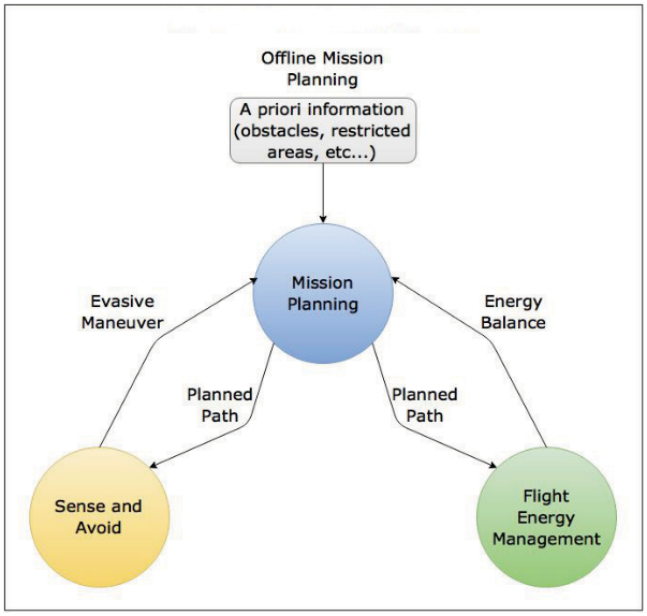



Figure 1.1: Low-cost UAV safety enhancement systems [1].

## 1.2 Motivation

The first applications of unmanned aerial vehicles (UAV's) were military-type. Over time, by reducing their cost and increasing their capacity, they became attractive for civilian applications. In recent years there has been a large increase in technologies that are critical to the development of robust UAV's, technologies such as low-cost and powerful flight controllers, higher energy-density batteries, robust and lightweight materials and more efficient solar cells [2].

New assigned tasks to UAV's are demanding better management of energy to reach specific objectives without compromising safety. To increase the time that the system is active before it exhausts its energy source, it is necessary to manage energy in a more efficient manner. This can be achieved if renewable energy sources are considered and a total energy management algorithm is carried on-board in real time to evaluate the total energy/power consumption (and consequently the system's remaining energy/power).

## 1.3 Objectives

The main objective of this thesis is to develop and validate an algorithm that allows a total energy balance estimation in real time of an aircraft. To achieve this objective, a detailed description of the various tasks is shown below:

- Identification of all energy sources;
- Development of the flight energy management algorithm required to estimate in real time the total/partial energy, as well the rates of change;
- Identification of the parameters that feed the mathematical models;
- Expansion of the ArduPilot source code, using an IDE (Integrated Development Environment);

- Perform the information forwarding via telemetry, from the aircraft to the GCS;
- Implement the reception and data visualization at the GCS;
- Selection and integration of the required sensors in the flight controller;
- Experimental validation of the system, through flight tests.

## 1.4 Thesis Outline

After the introductory chapter, in which the contextualization, motivation and objectives of this master thesis are described, the present work is divided in seven chapters. A brief description of each chapter is presented bellow.

**Chapter 2** In this chapter, a state of the art review is presented. Some key concepts for the course of the present work are summarized and described. Therefore, from a starting point of the document, the reader has a global idea of what will be discussed/analysed. It also provides a detailed insight into the development of the most used SoC (State of Charge) and RRT (Remaining Run Time) methods as well as an overview of the dynamics of a real time SoC evaluation algorithm, where five different phases can be detected: initial state, standby, transitional state, charge and discharge states.

**Chapter 3** The main focus of this chapter is to present the theoretical background behind the SoC and RRT algorithm in order to understand how the energy flows into/from the system, respectively. To accomplish this a detailed review about the aircraft energy formulation is performed.

**Chapter 4** In this chapter the main objective is to describe the methodology and structure of the computer code behind the mathematical algorithm. To have a properly evaluation system, the computational algorithm must have the ability to read input data through sensors or pre-entered parameters, make a state decision, compute the acquired data and output the results either by on-board recording or sending it in real time to the user.

**Chapter 5** This chapter provides a detailed overview of the on-board data transmission to the ground control station and reception/data visualization at the ground control station. The capabilities added at the Ground Control Station Software (GCSS) and On-Screen Display (OSD) are also shown.

**Chapter 6** In this chapter all the required hardware to be used in this thesis as well as its calibration and integration into the UAV is described.

**Chapter 7** The experimental validation of the proposed algorithm is shown. To accomplish this, several flight tests were carried out on two distinct UAV's. For each aircraft, a set of missions with different requirements were also performed to evaluate and validate the behaviour of the system.

**Chapter 8** Conclusions are presented where the fulfilment of the initial objectives is discussed. Additionally, it also describes some improvements to be performed as future work.

# Chapter 2

# State of the Art

According to Merkt (2013) [3], poor aircraft energy management can lead to unsafe and inefficient operations causing significant financial losses to the aviation industry and even worse, fatal aircraft accidents. Further adds that in commercial aviation fuel consumption is the second highest cost to airline operators and a proper energy management can be the answer to reduce the cost of flight operations.
Although on a smaller scale, safety and performance of UAV's may also be in risk due poor energy management. To mitigate the problem, an energy state prediction algorithm should be carried onboard ensuring a safe and reliable operation. Thus, ideally, in case of a low energy state and depending on the mission profile the autopilot can perform a return to launch (RTL) action, landing the UAV safely. On the other hand, depending on the airframe (quadrotor, fixed wing or hybrid), mission profile, meteorological conditions, consumed power on cruise/loiter and terrain orography the autopilot can extend the mission either in terms of endurance or range.

## 2.1 Key Concepts

Before examining the main objective stated in Section 1.3, some important aspects which will facilitate the reading of the document at an early stage are described.

### 2.1.1 Total Energy

Henceforth, total energy will be defined as the sum of total mechanical energy and total chemical energy present in the UAV.
According to the law of energy conservation, the total mechanical energy of an airplane is the sum of its gravitational potential energy derived from height above the ground plus its kinetic energy acquired from speed through the air[1] [4],[3]. The other implication of the law of conservation of energy is that an aircraft can only lose total energy through drag and the only way in which an aircraft can gain total energy is through the energy added by the engine/motor (except in cases of high turbulence, wind and thermal lift). On the other hand, chemical energy can be decomposed into at least two types: electric energy[2] and fossil fuels. For the present work, fossil fuels will not be taken into account.

### 2.1.2 Mission Time Remaining

From common sense, mission time remaining can be defined as the remaining time to execute a predefined mission. However, since the purpose of this thesis is to provide a real time prediction of the energy balance, from now on, it will be defined as the maximum estimated time that

[1] Yet kinetic energy depends on inertial speed, not airspeed.
[2] Depending on the application, the energy received from solar panels can be included.

the system can supply enough energy to maintain a minimum flight altitude and land the UAV safely. Thus, throughout the evolution of the mission, the value of Mission Time Remaining (MTR) is constantly updated.

### 2.1.3 Battery Monitoring

From occasional manual readings of voltage, visual inspection, periodical tests of capacity, manual measurement of battery resistance to fully automated on-line supervision, the term battery monitoring can be used in a large range of meanings or areas [5]. In this context, battery monitoring will be used as on-line reading /supervision.

### 2.1.4 State of Charge

In electric autonomous systems the term state of charge is a very important parameter because batteries work in a very narrow range of voltages. To protect the battery elements against a deep discharge and overcharge, it is necessary to evaluate the energetic state of the system otherwise the risk of damage may occur [6]. According to Sauer et al. (1999) and Pop (2007) ([7],[8]) the SoC of a battery is defined as the available capacity expressed as a percentage of its rated capacity. When the SoC is equal to 100% a full state of charge is reached and when it is equal to 0% reflects an empty battery. Summing up, it is analogous to the fuel gauge for a conventional internal combustion (IC) engine [9].

As referenced with more detail in Section 2.2, there are several ways to determine on the SoC depending on the battery type and application that is being used.

### 2.1.5 Telemetry

Telemetry is an automated communication process by which measurements and/or other relevant data is collected at remote or inaccessible points and transmitted to receiving equipment for monitoring [10]. Although the term commonly refers to wireless data transfer mechanisms, for example radio, ultrasonic or infrared systems, it also encompasses data transferred over other media such as a telephone or computer network, optical link or other wired communications.

### 2.1.6 Ground Control Station

From the definition, a Ground Control Station (GCS) is a ground based control unit that provides the facilities (if in case of that) and resources/equipment for human control of Remotely Piloted Aircraft Systems (RPAS) [11]. Used to plan and execute missions, control the aircraft, modify the waypoints in real-time, keep a detailed log of all the telemetry and receive video in real-time has typically two consoles, one for the aircraft operator and other for the payload operator depending on the purpose. Regarding the ground control station software, it typically runs on a ground-based computer that performs all the data processing, essential for the aircraft operation.

## 2.2 Methods for State of Charge Estimation

Piller et al. (2001) [6] refers that, depending on the battery type and on the application in which the battery is being used, the determination of the SoC may be more or less complex. The following subsections describe in detail the main/most used methods for determining the SoC in LiPo/Li-ion batteries (generic batteries may also be considered).

### 2.2.1 Discharge Test

A discharge test under controlled conditions is one of the most reliable methods for the SoC estimation of a battery. However, it is too time consuming to be considered in most applications since it usually includes a consecutive full charge.

### 2.2.2 Ampere hour counting

Widely used, the ampere hour counting relates directly the supplied or withdrawn current with the charge or discharge state, respectively. Starting from an initial SoC point ($SoC_0$), it integrates the current flow over the time. In other words, the SoC of a battery is updated by adding or subtracting the last period's net cumulative current,

$$SoC = SoC_0 + \frac{1}{C_N}\int_{t_0}^{t} I_{batt}dt \tag{2.1}$$

where $SoC_0$ corresponds to an initial SoC point, $C_N$ is the rated capacity, $I_{batt}$ (takes a negative value when the battery is discharging) is the battery current, $t_0$ is the initial time instant and $t$ is a given time instant.
As demonstrated through practical tests in Chapter7, an incorrect current measurement due to inaccurate equipment can add up a large error over the integration since the method is an open loop SoC estimator. The solution to this passes through the acquisition of accurate equipment which is usually expensive.
Another drawback is that not all the withdrawn/supplied current is consumed by discharging/charging due to losses in the system (e.g. in the form of heat). In addition to this, the error is larger when the battery works at high/low temperatures or when the current fluctuates dramatically. In Eq.2.2 the final equation where a current loss ($I_{loss}$) has been added to the integration is shown.

$$SoC = SoC_0 + \frac{1}{C_N}\int_{t_0}^{t} (I_{batt} - I_{loss})\, dt \tag{2.2}$$

To overcome the aforementioned point, an open circuit voltage (OCV) measurement can be used to correct from time to time the SoC value or when a full charge is detected the SoC is set to one (100%). This way, and although there is a reading error by the sensors, the error coming from them is replaced by the error of the SoC-OCV model/measurement.
Piller et al. (2001) [6] also indicates that the most simple loss estimation method is to apply a constant charge factor to the battery at each recharge. In other words a constant loss is assumed and this loss is additionally returned to the battery. Such method is only suitable for systems that are not too sensitive to overcharge like NiCd or NiMH, in comparison with lead-acid batteries or especially LiPo/Li-ion batteries. For NiCd/NiMH batteries a value of 1.3 is often used to assure full charge of the battery while for lead-acid batteries a charge factor of 1.05-1.2 is usually used. In addition, this method implies that the charging operation is controlled

and a full charge state is reached which may not happen in a UAV (even if a solar powered UAV is being considered).

### 2.2.3 Open Circuit Voltage

Such method is promising where relatively long rest periods are common. However a point to take into account is the concept of rest period, which according to Piller et al. (2001) [6] may not occur since a minimal current flow is required for any devices/system. In this case, the open circuit voltage is never achieved and is incorrect to apply this method. Summarizing, the OCV method should be considered in combination with other methods and not as a main method to be used alone.

### 2.2.4 Kalman Filter

A Kalman filter (KF) is an algorithm to estimate the inner states of a highly dynamic system [6]. Wang et al. (2007) [12] indicates the KF as an established technology which provides a theoretically elegant and time-proven method to filter measurements of system input and output to produce an intelligent estimation of a dynamic system's state. Also adds that it is in common use in many fields including target tracking, navigation and communication, but is not widely used in the battery field. Due to this, there has been some interest to estimate the SoC of a system and consequently the remaining run-time (RRT) with this method.
Taking into account that the ampere hour counting is not a satisfactory method, as the initial SoC and Coulomb efficiency are difficult to measure, new methods are being considered such as the *KalmanAh* [12]. The author indicates a SoC estimation error of 2.5% against 11.4% when using the ampere counting.[3] In Subsection 2.3.2 a more detailed analysis of this method is made. Oettershagen et al. (2016) [13] also refers the use of an Extended Kalman Filter (EKF) in the Low-Altitude Long-Endurance (LALE) solar powered UAV AtlantikSolar (for more information about the project and UAV please see reference [14]). The way the system works is based on the acquisition of data from a 10 Degrees of Freedom (DoF) Inertial Measurement Unit (IMU), a GPS and an airspeed sensor to estimate position, velocity, orientation (attitude and heading), QFF (pressure at sea level), gyroscope biases, accelerometer biases and the wind vector. Moreover sideslip angle and Angle of Attack (AoA) can subsequently be derived from these estimates. The estimator can cope with prolonged GPS outage scenarios through an implemented airspeed sensor fall-back mode and is optimized to run efficiently on an on-board Pixhawk flight controller [15].

## 2.3 Accuracy Analysis of the State-of-Charge

Based on the fact that most electric UAV's are equipped with LiPo and Li-ion batteries, an accurate SoC and RRT calculation is crucial either for the above described reasons or to prolong the lifetime of batteries. Pop et al. (2009) [16] presents an algorithm to predict the SoC and RRT of a Li-ion battery with an error of 1 minute or less under all realistic user conditions, including a wide variety of load currents and temperatures. Thus, an algorithm that combines adaptive systems with an open circuit voltage (OCV or $V_{OC}$) measurement during the equilibrium state and ampere hour counting or coulomb counting (CC) during the charge and discharge states was

[3] Both values are compared with the real SoC obtained from a discharge test.

developed and implemented in a real time system. An electromotive force (EMF) model for the relation between the SoC and the battery $V_{OC}$ was also considered.

### 2.3.1 Online Battery Monitoring System States

An online SoC evaluation system can be divided in five different phases: initial, standby, transitional, charge and discharge states, Fig.2.1. Each time the evaluation system is switched-on, it starts from the initial state where the SoC is determined by means of voltage measurement, temperature measurement and the stored SoC–EMF model. Then, depending on whether the battery is charged, discharged or in equilibrium, the system switches to the real state of the battery.

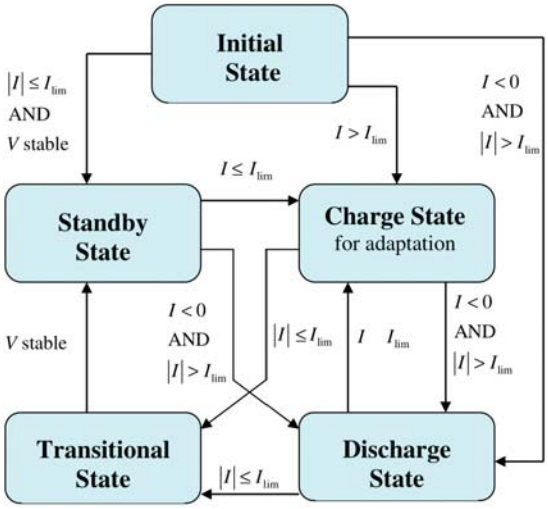



Figure 2.1: State diagram of a real time SoC evaluation system [16].

**Standby State** In standby state the battery is supposed to be in equilibrium and the $SoC_s$ (State of Charge: Standby State) is also determined by means of voltage measurement, temperature measurement and the stored SoC–EMF models. According to Fig.2.1 this state is achieved whenever the current drawn/received ($I_s$) is less than a current limit ($I_{lim}$), the measured battery voltage ($V$) is stable and close to the $V_{OC}$. Depending on the system that is being evaluated this $I_{lim}$ value can be changed to match the requirements.

**Transitional State** When the system changes from either charge or discharge mode to standby state the transitional state is used. Since the battery voltage is not stable, the $SoC_t$ (State of Charge: Transitional State) is determined by means of coulomb counting and when the battery voltage has reached a stable value the system is allowed to enter into the standby state.

**Discharge State** If the battery is being discharged, discharge state, a large negative (signal adopted by the author) current flows out of the battery. In other words, $I_d$<$I_{lim}$ and $|I_d|$>$I_{lim}$ should be verified. As in the previous states, the $SoC_d$ (State of Charge: Discharge State) and RRT is calculated using coulomb couting and the effect of the battery overpotential is considered.

**Charge State** The charge state is established when a charger is connected to the battery and a large positive current flows into the battery, that is equal to $I_{ch}$>$I_{lim}$ and $|I_{ch}|$>$I_{lim}$. In both states, discharge and charge, the $SoC_ch$ (State of Charge: Charge State) estimation is done using coulomb couting.

Finally, Pop et al. (2009) [16] concludes that "... in which state the system is operating depends on the value and sign of the current, which is flowing into or out of the battery and whether the battery voltage is stable or not".

### 2.3.2 SoC and RRT Error Analysis

Several authors, [6] and [16], refer that independently of the measurement method there will be always a difference between the measurement result and the true value. Piller et al. (2001) [6] indicates that the worst scenario happens with the current counting method. As expected the RRT estimation value is imprecise due to errors from current measurement integration over time and, because of that, the available systems are still not acceptable for practical use. To enable a more accurate SoC and RRT calculation, the error sources in each state must be identified and minimised.

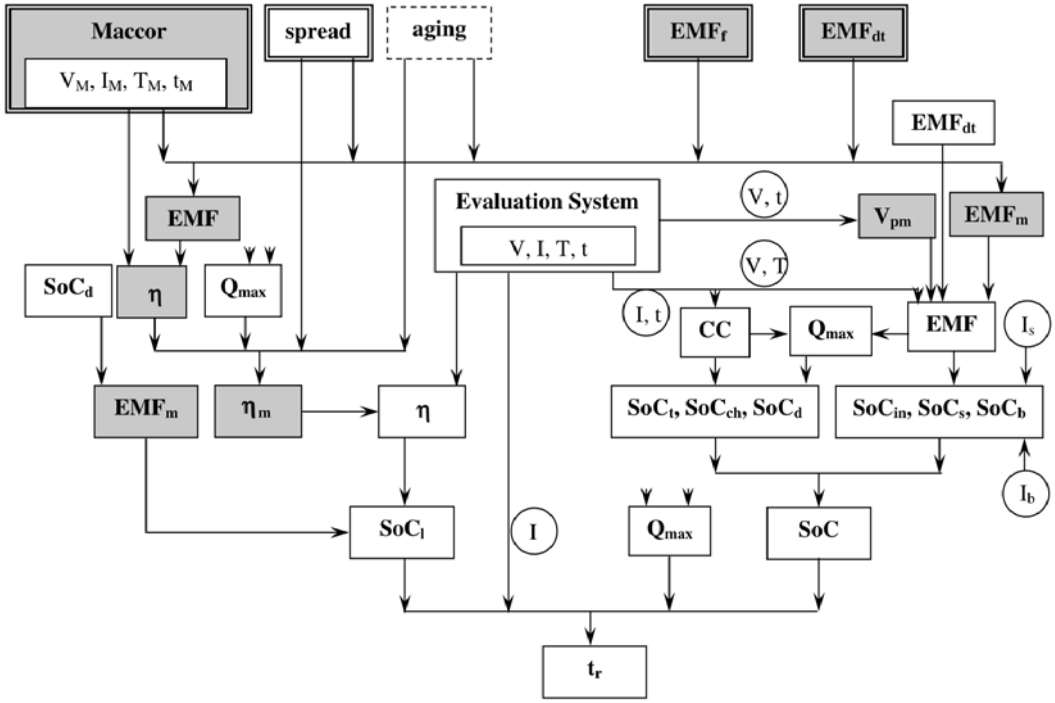



Figure 2.2: Schematic of the error sources in an online SoC & RRT simulation performed by Pop et al. (2009) [16]

In Fig.2.2 a schematic of the various error sources and how they propagate to the final result in a SoC and RRT simulation is shown. At this study both Electromotive Force (EMF) direct measurement and Coulomb Counting (CC) methods have been implemented in a real-time evaluation system. The voltage ($V_M$), current ($I_M$), temperature ($T_M$) and time ($t_M$) obtained from measurements with the battery tester (Maccor[4]) are considered as reference measurements and the errors introduced in the EMF measurements by the EMF fitting ($EMF_f$) and detection ($EMF_{dt}$)

[4]The battery tester model from Maccor Inc. is not provided by the author.

methods are small (less than 0.1% in the SoC). The model for the relation between the SoC and the battery EMF is defined as $EMF_m$, $\eta$ is the effect of the battery overpotential during a discharge state, $\eta_m$ is the error source from spread in the overpotential model, $Q_{max}$ is defined as the maximum capacity of a rechargeable battery, $I_b$ is the current in a backlight-on state, $SoC_l$ is the predicted battery overpotential in $V$ translated into a SoC percentage, $SoC_i$ is the SoC in an initial state, $SoC_b$ is the SoC in a backlight-on state and $t_r$ is the renaiming run time or time remaining.
Since the simulation will not be discussed in this work, please see [17], [18] & [19] for a detailed description. Four different error sources can be identified as:

- Round blocks denote the error sources from the on-line measured variables (e.g. $V_{batt}$, $I_{batt}$, $T_{batt}$, $t$) used as input for the calculations;
- Single line blocks denote the error sources from the real time measurements obtained with the SoC prediction system;
- Grey blocks indicate the error sources from offline measurements and models (e.g SoC–EMF model);
- Dashed and the double-line blocks indicate the error sources that have not been considered.

#### 2.3.2.1 Error Analysis: KF *KalmanAh* Method

As aforementioned in Section 2.2, Wang et al. (2007) [12] proposes a new method to estimate the SoC of a battery. Below, the validation of the model as well some important conclusions is shown.

**Battery experiment** To certify and evaluate the *KalmanAh* method, a NiMH battery (other battery types can be considered) with a nominal voltage of 384V and nominal capacity of 80Ah was submitted to 8.08 continuous federal urban driving schedule (FUDS) cycles which corresponds to a sequence of constant-current discharge pulses and followed by a sequence of constant current charge pulses. The test equipment was an AeroVironment™ 900 [20], AV-900, capable to implement virtually any electrical driving cycle and offer up to 250 kW, with a voltage range of 8–900 VDC and a current range of ±1000 ADC.
The average temperature was 25.911°C when the test started and 27.521°C when the test ended. The maximum discharge current was 129.2A while the maximum charge current was 63.8A. The batteries were charged up to 53.47Ah at a C/3 rate before test and discharged up to 12.79Ah at the C/3 rate after test. The net discharged capacity recorded by the AV-900 was 36.79Ah. The voltage and current profiles of the test are shown in Fig.2.3. For convenience the current is positive while discharging and negative while charging.

**Analysis and Comparison** For terms of analysis and comparison four different SoC estimation methods were considered. The Accurate-Ah, an ampere-hour counting method where accurate initial SoC is given and the coulombic efficiency (ratio between the discharged capacity with the capacity needed to be charged to the initial state before discharge) is considered. *KalmanAh*, the method proposed by the author. AhEquip, an ampere hour counting method estimated by the AV-900 equipment which does not consider the coulombic efficiency. Lastly, the OcvAh

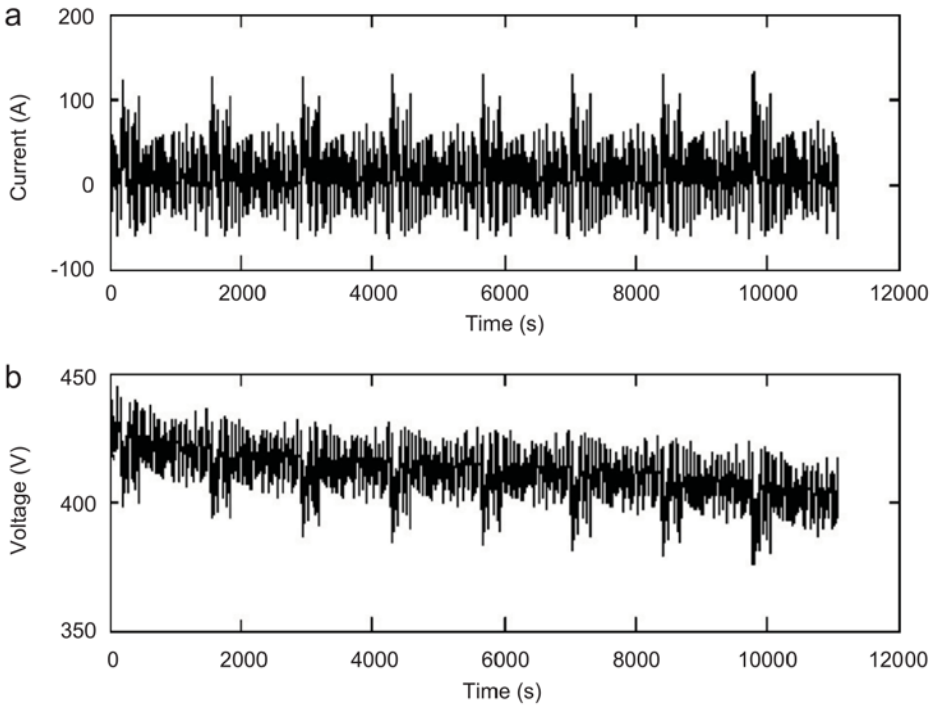



Figure 2.3: Current (a) and voltage (b) profiles of the battery test. [12]

method an ampere hour counting method where the initial SoC is estimated by the open-circuit voltage approach and the coulombic efficiency is not considered.
In the *KalmanAh* approach, given an initial $SoC_{x0}$=0.69, the EKF approach is used within the time interval of the 1st to the 100th second. From Fig.2.4, the SoC value estimated by the *KalmanAh* approach is very close to the real SoC at t=30s. Then after this period the ampere hour counting approach is used.

From Table 2.1 a comparison of the different SoC estimation methods is shown. It can be noted that the SoC obtained from a discharge test at the C/3 rate is the real value. Beyond that, the SoC estimation value from the Accurate-Ah approach is closest from the discharge test approach. Compared with the Accurate-Ah approach, the estimation error from the AhEquip approach is apparently large since the coulombic efficiency is not considered. The OcvAh approach is not able to satisfy the requirements due to larger estimation error, caused by the error contained in the starting estimate that will be carried forward. The Accurate-Ah approach is slightly better than the *KalmanAh* approach, but only if implemented in laboratory since it is not easy to get an accurate initial SoC.

Table 2.1: SoC estimation error [12]

| Method | Discharge test | AhEquip | Accurate-Ah | OcvAh | KalmanAh |
|---|---|---|---|---|---|
| Initial SoC | 0.643 | 0.651 | 0.643 | 0.690 | 0.690 |
| Terminal SoC | 0.160 | 0.201 | 0.183 | 0.274 | 0.185 |
| Estimation error | 0 | 0.041 | 0.023 | 0.114 | 0.025 |

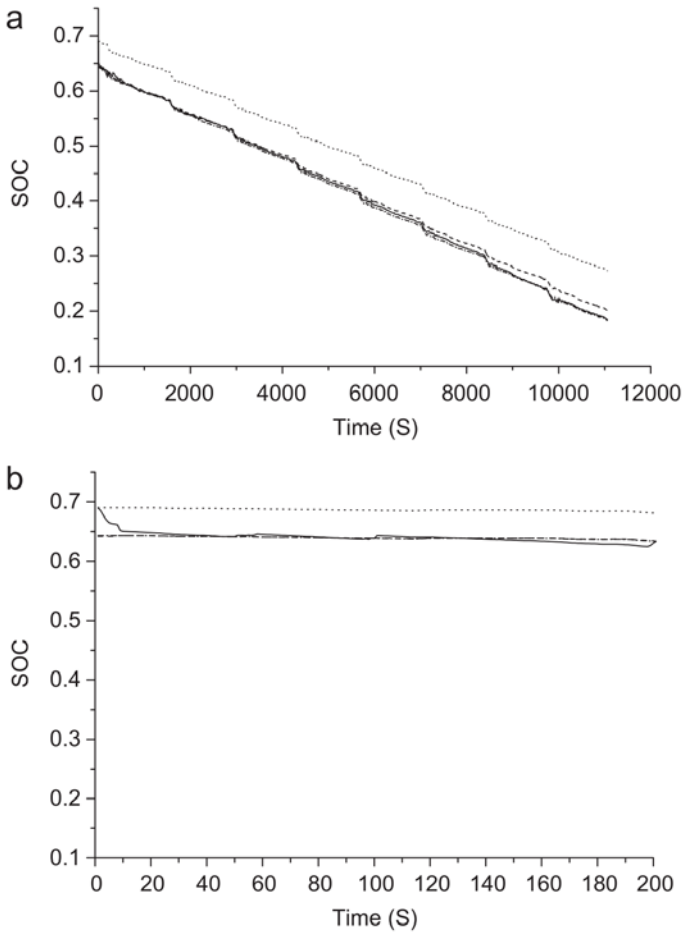



Figure 2.4: SoC estimation results using four methods, Panel (b) is the enlarged part of Panel (a); ....., OcvAh method; —-, AhEquip method;—, KalmanAh method; -·-·-, accurate-Ah method. [12]

## 2.4 Energy Monitoring System

Also integrated in the "Drones' Safe Flight - Gestão de energia, planeamento de missão e deteção de obstáculos em voo" project, Baião (2017) [1] approaches the development of a system capable of generating an updated estimate of the state of total energy remaining onboard the aircraft at the end of the mission, capable of being run on the airborne avionics hardware. Starting with an offline estimation (pre-mission), this estimation is accomplished by constantly monitoring the available energy onboard, the aircraft's energy consumption, the expected required energy to complete the mission and the solar energy available. In addition the system takes into account external conditions (wind, solar radiation, handicapped airframe or trajectory change, either due to a pilot or ground control command or automatic obstacle avoidance manoeuvres), as well as those predicted for the rest of the mission.

### 2.4.1 Energy Flow in UAVs

Baião (2017) [1] also indicates that to have a proper energy management system, it can not be exclusively looking at the past energy flow since that only allows the computation of the present state, which is insufficient to decide whether or not the system has enough energy to complete

the planned mission, and therefore the future energy flow has to be predicted. In Fig.2.5, the flow of energy into and out of the system is shown. In addition, the author indicates that naturally there will be a discrepancy between the expected future consumption, predicted at a previous instant, and the amount of energy that will actually be spent in the future. Due to this, the balance of energy is constantly being recalculated for the full duration of the mission.

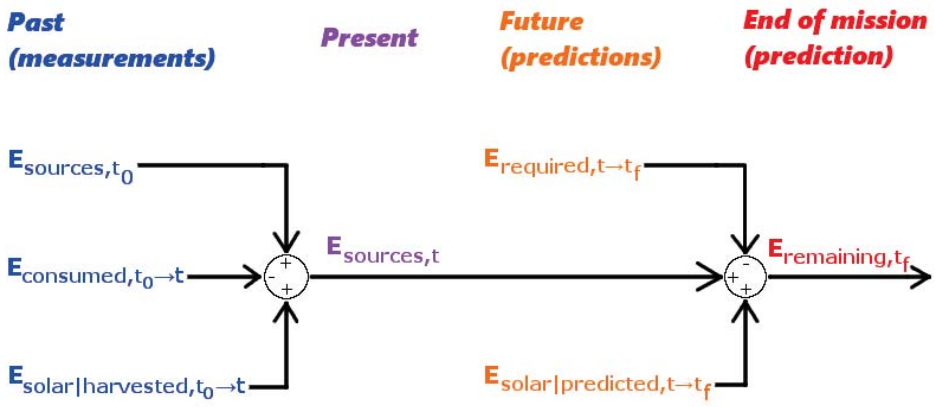



Figure 2.5: Energy balance at end of mission. [1]

### 2.4.2 Matlab® Simulation based on LEEUAV Data

Experimental data of the LEEUAV[5] (Long Electric Endurance Unmanned Aerial Vehicle) was used to perform the Matlab® simulation of two distinct missions. In both missions the aircraft always travels in approximately the same direction to simplify the study on the effect of wind velocity on the required energy to complete the mission, by assuming it has a constant direction. Since for a simulation of a real flight it would be necessary to obtain a map with an estimate for the wind velocity in the area to fly, a temporal distribution of the wind was used.
The MATLAB® algorithm used for this simulations is represented in Fig.2.6. From a file, it reads the set of waypoints for the aircraft to follow and sets the initial position at the first waypoint. The climb angle, ground speed components and acceleration in the inertial reference frame are obtained for the first mission segment. After this, the initial energy available in the energy sources is calculated and the sequence begins by accumulating the predicted solar energy harvested and the energy expected to be required during the time step $t$ chosen for the simulation. At this stage, the algorithm will detect if the aircraft is within range of the next waypoint or not. If the aircraft is not yet within range of the next waypoint, the position, velocity and acceleration are updated using the equations of linearly accelerated uniform motion. The instant of prediction is then incremented by the step value defined for the simulation and the system checks if at the instant $t+1$ the aircraft is within range of the next waypoint. Afterwards, if this completed mission segment was not the last one, the climb angle and the components of velocity and acceleration in the ECEF (Earth Centered Earth Fixed) reference frame are obtained for the next segment and the position is also updated. When the last mission segment is reached, the cycle ends (the code also ends) and the predicted remaining energy at the end of the flight is computed.

[5] (See Appendix D)

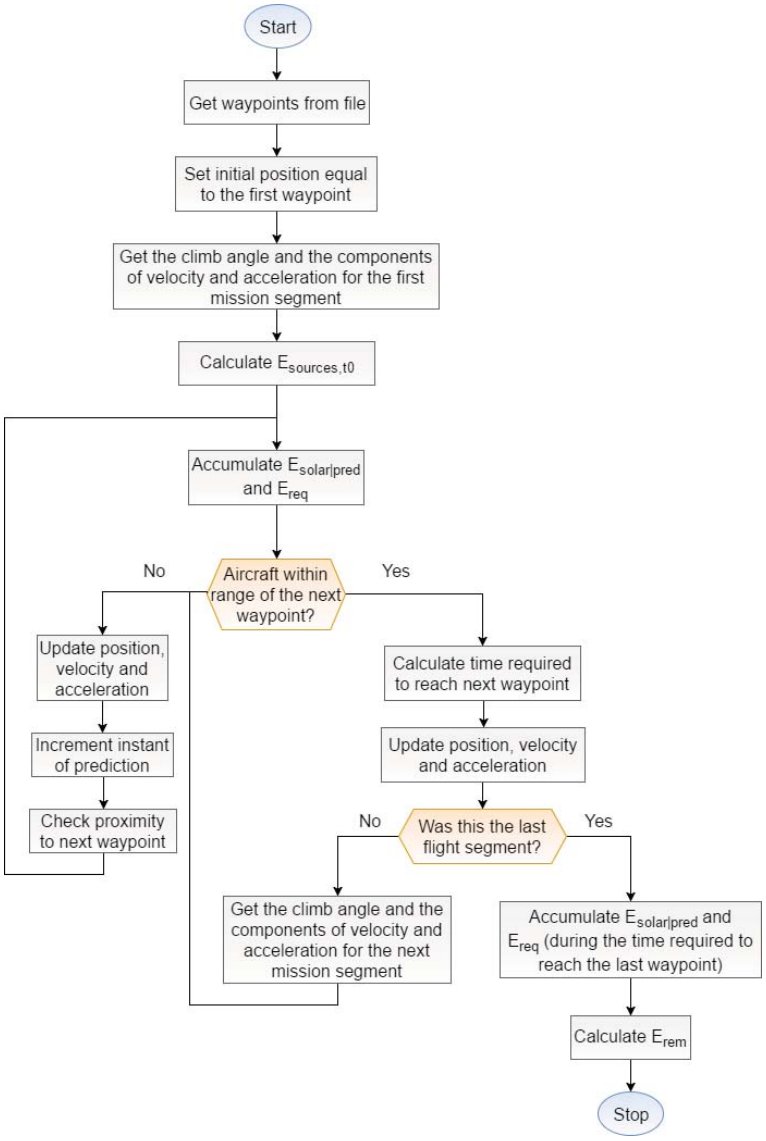



Figure 2.6: MATLAB® algorithm used to perform the simulations.[1]

# Chapter 3

# Aircraft Energy Formulation

The main focus of the current chapter is to present the theoretical background behind the SoC and RRT algorithm proposed in Chapter 4. From Fig.3.1 it can be noticed that different types of energy feed the algorithm. It should be mentioned that although fossil fuels appears in the flowchart, it was not considered in this work since at the moment AeroG does not own internal combustion UAV's.

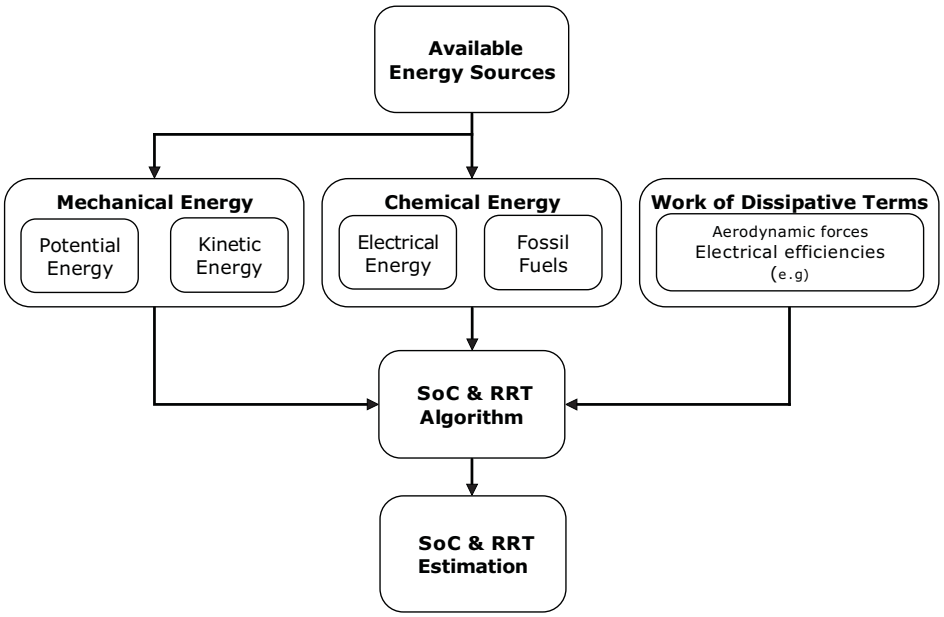



Figure 3.1: Energy sources that feed a real time SoC & RRT evaluation system.

## 3.1 Mechanical Energy

### 3.1.1 Conservation of Mechanical Energy

A physical system consists of a well defined set of bodies that interact by means of forces. Any body that lies outside the boundary of the system resides in the surroundings. A state of the system is a set of measurable physical quantities that completely characterize the system.
From the definition of change of energy, "*the total change in energy of a system and its surroundings between the final state and the initial state is zero*", it is noted that

$$\Delta E_{total} = \Delta E_{system} + \Delta E_{surroundings} = 0 \tag{3.1}$$

where $\Delta E_{total}$ is the total change of energy, $\Delta E_{system}$ is the system's change of energy and $\Delta E_{surroundings}$ corresponds to the surroundings change of energy. Focusing only on the mechanical energy of a system, there are two types of energy: kinetic energy and potential energy. So,

the total change in mechanical energy is defined to be the sum of the changes of kinetic and potential energies,

$$\Delta E_{mechanical} = \Delta K_{system} + \Delta U_{system} \tag{3.2}$$

From both the definition of change of potential energy and the work-energy theorem, the total change in the mechanical energy is zero,

$$\Delta E_{mechanical} = W_{cons} + (-W_{cons}) = 0 \tag{3.3}$$

known as conservation of mechanical energy.

With this, a conservative mechanical system can be defined as

$$E_{mechanical} = K + U \tag{3.4}$$

where $K$ is the kinetic energy and $U$ the potential energy.

The change of mechanical energy when a system goes from an initial state to a final state comes as

$$\Delta E_{mechanical} = E_{mechanical,f} - E_{mechanical,0} = (K_f + U_f) - (K_0 + U_0) \tag{3.5}$$

Thus, when there are no non conservative forces acting on the system, the total mechanical energy of the system is conserved, and

$$E_{mechanical,f} = E_{mechanical,0} \tag{3.6}$$

### 3.1.2 Non conservation of Mechanical Energy

The total force acting on a system may, in general, consist of a vector sum of conservative forces and non conservative forces,

$$\vec{F}^{total} = \sum_i \vec{F}_{cons,i} + \sum_j \vec{F}_{non-cons,j} \tag{3.7}$$

The work done by the total non conservative force, which is the sum of all non conservative forces,

$$\vec{F}^{total}_{non-cons} = \sum_j \vec{F}_{non-cons,j} \tag{3.8}$$

between two points A and B is a path dependent quantity given as

$$W_{non-cons}\,[A,B] = \int_A^B \vec{F}^{total}_{non-cons} \cdot d\,\vec{r} \tag{3.9}$$

Then, the work done by the total force is

$$
\begin{aligned}
W^{total}\left[A,B\right] &= \int_A^B \overrightarrow{F}^{total} \cdot d\,\overrightarrow{r} = \int_A^B \left( \sum_i \overrightarrow{F}_{cons,i} + \sum_j \overrightarrow{F}^{total}_{non-cons,j} \right) \cdot d\,\overrightarrow{r} \\
&= \sum_i \int_A^B \overrightarrow{F}_{cons,i} \cdot d\,\overrightarrow{r} + W_{non-cons}\left[A,B\right] \\
&= -\Delta U^{total} + W_{non-cons}\left[A,B\right]
\end{aligned}
\tag{3.10}
$$

With this, the work-energy theorem (see Subsection 3.1.3) states that the work done by the total force is equal to the change in kinetic energy,

$$
W_{total} = \Delta K = \frac{1}{2} m v_f^2 - \frac{1}{2} m v_0^2 \tag{3.11}
$$

Hence

$$
\Delta K = -\Delta U^{total} + W_{non-cons}\left[A,B\right] \tag{3.12}
$$

The change in mechanical energy of the system between the states represented by A and B,

$$
\Delta E^{total}_{mechanical}\left[A,B\right] = \Delta K + \Delta U^{total} \tag{3.13}
$$

is defined as the sum of the change of the kinetic energy and change of the total potential energy which is independent of the path between the states A and B.
Therefore the total change in the mechanical energy of the system between the states A and B is equal to the work done by the non conservative forces along the chosen path connecting points A and B

$$
\Delta E^{total}_{mechanical}\left[A,B\right] = W_{non-cons}\left[A,B\right] \tag{3.14}
$$

Since the system in this work is non conservative due to the drag force (the aerodynamic force that opposes an aircraft's motion through the air), for steady level flight (SLF) the only way to maintain height and speed is to supply thrust using a motor so that the thrust force is equal to the drag force, and lift equal to weight, Fig.3.2. Thus, rewriting Eq.3.14 we have

$$
\Delta E^{total}_{mechanical(SLF)}\left[A,B\right] = W_{drag(SLF)}\left[A,B\right] \tag{3.15}
$$

which is equal to

$$
\Delta E^{total}_{mechanical(SLF)}\left[A,B\right] = \left|E_{motor(SLF)}\right|\left[A,B\right] = W_{drag(SLF)}\left[A,B\right] \tag{3.16}
$$

Any thrust available in excess of that required to overcome the drag can be applied to accelerate the vehicle (increasing kinetic energy) or to cause the vehicle to climb (increasing potential energy).

### 3.1.3 Kinetic Energy Derivation

Considering a conservative system to proceed the derivation of the kinetic energy equation, there is a relation between the total work done on an object and the change of kinetic energy. The work done by all the applied forces on an object equals the change in kinetic energy of the object, Eq.3.11.

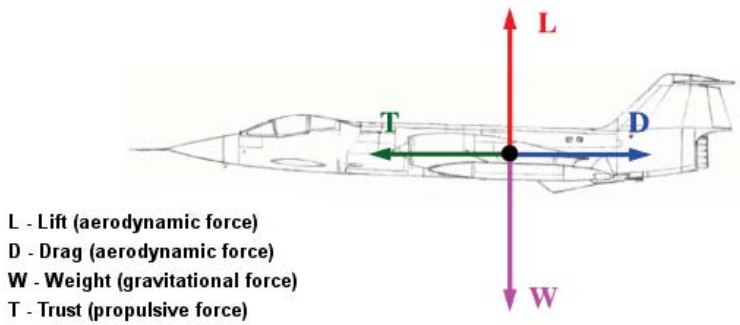



Figure 3.2: Simplified airplane motion/forces on an airplane: steady level flight. (adapted from [21])

In other words, a non zero total work implies that the total force acting on the object is non zero and the object will accelerate. When the total work done on an object is positive the object will increase its speed. When the work done is negative, the object will decrease its speed. When the total work done is zero, the object will maintain a constant speed.

The work-energy theorem holds as well for non constant forces. The work done by a non constant force in a moving object along the x-axis from an initial position $x_0$ to the final position $x_f$ is given by

$$W = \int_{x_0}^{x_f} F_x dx \tag{3.17}$$

where $F_x$ is the component of the force in the x-direction. According to Newton's Second Law it is known that

$$F_x = m\frac{dv_x}{dt} \tag{3.18}$$

Consequently, the work integral can be written as

$$W = \int_{x_0}^{x_f} F_x dx = \int_{x_0}^{x_f} \frac{dv_x}{dt} dx = \int_{x_0}^{x_f} m\frac{dx}{dt} dv_x \tag{3.19}$$

Since the x-component of the velocity is defined as $v_x = dx/dt$, the work integral becomes

$$W = \int_{v_{x,0}}^{v_{x,f}} m\frac{dx}{dt} dv_x = \int_{v_{x,0}}^{v_{x,f}} mv_x dv_x \tag{3.20}$$

It should be noted that the limits of the integral were changed. Instead of integrating from the initial position $x_0$ to the final position $x_f$, the limits of integration are now from the initial x-component of the velocity $v_{x,0}$ to the final x-component of the velocity $v_{x,f}$. Since

$$d\left(\frac{1}{2}mv_x^2\right) = mv_x dv_x \tag{3.21}$$

the integral is

$$W = \int_{v_{x,0}}^{v_{x,f}} mv_x dv_x = \int_{v_{x,0}}^{v_{x,f}} d\left(\frac{1}{2}mv_x^2\right) \tag{3.22}$$

It follows that

$$W = \int_{v_{x,0}}^{v_{x,f}} mv_x dv_x = \int_{v_{x,0}}^{v_{x,f}} d\left(\frac{1}{2}mv_x^2\right) = \frac{1}{2}mv_{x,f}^2 - \frac{1}{2}mv_{x,0}^2 = \Delta K \tag{3.23}$$

The work energy theorem can be then generalized to three-dimensional motion. Supposing an object that is under the action of an applied force, it changes its velocity from an initial velocity,

$$\vec{v}_0 = v_{x,0}\hat{i} + v_{y,0}\hat{j} + v_{z,0}\hat{k} \tag{3.24}$$

to the final velocity,

$$\vec{v}_f = v_{x,f}\hat{i} + v_{y,f}\hat{j} + v_{z,f}\hat{k} \tag{3.25}$$

The kinetic energy is given by

$$K = \frac{1}{2}m\left(v_x^2 + v_y^2 + v_z^2\right) = \frac{1}{2}m\,\vec{v}^2 \tag{3.26}$$

At this point, the kinetic energy equation has been derived and defined. Hence, the time rate of change of kinetic energy (kinetic power) derivation is now simple to perform.
In one dimension, the time rate of change of the kinetic energy is given by

$$\frac{dK}{dt} = \frac{d}{dt}\left(\frac{1}{2}mv_x^2\right) \tag{3.27}$$

so

$$\frac{dK}{dt} = \frac{1}{2}\left(\frac{dm}{dt}{v_x}^2 + 2mv_x\frac{dv_x}{dt}\right) \tag{3.28}$$

If the mass does not change over the time, then

$$\frac{dm}{dt} = 0 \tag{3.29}$$

and finally

$$\frac{dK}{dt} = \frac{1}{2}\left(2mv\frac{dv_x}{dt}\right) = mv_x\frac{dv_x}{dt} \tag{3.30}$$

The generalization to three dimensions becomes

$$\frac{dK}{dt} = \frac{d}{dt}\left(\frac{1}{2}m\left(v_x^2 + v_y^2 + v_z^2\right)\right) = mv_x\frac{dv_x}{dt} + mv_y\frac{dv_y}{dt} + mv_z\frac{dv_z}{dt} \tag{3.31}$$

or

$$\frac{dK}{dt} = \frac{d}{dt}\left(\frac{1}{2}m\,\vec{v}\right) = m\,\vec{v}\,\frac{d\,\vec{v}}{dt} \tag{3.32}$$

### 3.1.4 Potential Energy Derivation

Regarding the example of an object falling near the surface of the earth and considering the earth and object as a system, the gravitational force is now an internal conservative force acting inside the system. The distance separating the object to the centre of mass of the earth, and the velocities of the earth to the object specifies the initial and final states.
Choosing a coordinate system with the origin on the surface of the earth and the positive direction of the z axis pointing away from the centre of the earth, because the displacement of the

earth is negligible, it is only necessary to consider the displacement of the object in order to calculate the change in potential energy of the system.

Supposing that the object starts at an initial height $z_i$, above the surface of the earth and ends at a final height $z_f$, the gravitational force on the object is given by

$$\vec{F}^{g} = -mg\hat{k} \tag{3.33}$$

where the displacement is given by

$$d\,\vec{r} = dz\hat{k} \tag{3.34}$$

and the scalar product,

$$\vec{F}^{g} \cdot d\,\vec{r} = -mg\hat{k} \cdot dz\hat{k} = -mgdz \tag{3.35}$$

The work done by the gravitational force on the object is then

$$W^{g} = \int_{z_i)}^{z_f} \vec{F}^{g} \cdot \vec{r} = \int_{z_i)}^{z_f} -mgdz = -mg\left(z_f - z_i\right) \tag{3.36}$$

The change in potential energy is then given by

$$\Delta U = -W^{g} = mg\Delta z = mgz_f - mgz_i \tag{3.37}$$

A zero reference point for the potential energy can be selected wherever is convenient. This way, it is possible to adapt the zero to best fit a particular problem. Because the change in potential energy only depends on the displacement, $\Delta z$, in the above expression for the change of potential energy (Eq.3.38), let $z_f = z$ (or $h_f = h$) be an arbitrary point and $y_i = 0$ (or $h_i = 0$) denote the surface of the earth.

The potential energy is then given by

$$U = mg\Delta h = mgh_f - mgh_i = mgh \tag{3.38}$$

Similarly to the kinetic energy, the potential energy has been derived and defined. The time rate of change of potential energy (potential power) is shown bellow.

$$\frac{dU}{dt} = \frac{d}{dt}\left(mgh\right) = mg\frac{dh}{dt} \tag{3.39}$$

## 3.2 Electrical Energy

The electrical system's main purpose is to provide electrical energy to all equipment requiring electrical power to run. To understand the relation between the battery energy state with voltage and current, a simple battery model is presented below. A solar panel model that uses a Solar Charge Controller (SCC), Genasun GV-10-Li-12.5V, is also described.

Since LiPo batteries offer a wide array of benefits, such as light weight, high capacities, high discharge rates and high specific energy comparing with other battery types, the algorithm presented in Chapter 4 was developed for them.

### 3.2.1 Battery Model

Batteries are used to provide electrical power to the systems, motor and to store energy if other power sources are present such as solar panels[1].
LiPo batteries can be manufactured in very different configurations with respect to their elements. Hence, the total number of battery elements is given by

$$n_B = n_{BS} \cdot n_{BP} \tag{3.40}$$

where $n_{BS}$ is the number of battery elements connected in series and $n_{BP}$ is the number of battery elements connected in parallel. The voltage of a battery element must never exceed a maximum value of 4.2v ($U_{B1max}$) and fall below a minimum cut-off value of 3.2v ($U_{B1min}$). The maximum and minimum voltages of a battery pack are then

$$U_{Bmax} = n_{BS} \cdot n_{B1max} \tag{3.41}$$

$$U_{Bmin} = n_{BS} \cdot n_{B1min} \tag{3.42}$$

If each battery element has a maximum energy capacity of $C_{B1max}$, the total maximum capacity of the battery pack comes as

$$C_{Bmax} = n_{BP} \cdot C_{B1max} \tag{3.43}$$

Another important value of a single element is the maximum current drain that the element can support without damage. Usually, this is expressed as a multiple of the element capacity known as discharge rate or "C" rate, $K_B$. Thus, the maximum discharge current for a battery pack is

$$I_{Bmax} = K_B \cdot C_{Bmax} \tag{3.44}$$

The electrical power delivered by the battery is then defined as

$$P_B = U_B I_B \tag{3.45}$$

where $U_B$ is the battery voltage and $I_B$ the battery discharge current. The battery voltage and discharge current are functions of the remaining energy as shown in Fig.3.3,

$$E_B = f\left(U_B, I_B\right) \tag{3.46}$$

Hence,

$$\frac{d\left(E_B\right)}{dt} = P_B = U_B I_B \tag{3.47}$$

that corresponds to

$$E_B = \int_{t_i}^{t_i+1} \left(U_B I_U\right) dt \tag{3.48}$$

Regarding that the maximum charging current of a LiPo is (generally) limited to a value equal to their nominal capacity,

[1] Depending on the application a dedicated battery can be used to power the systems.

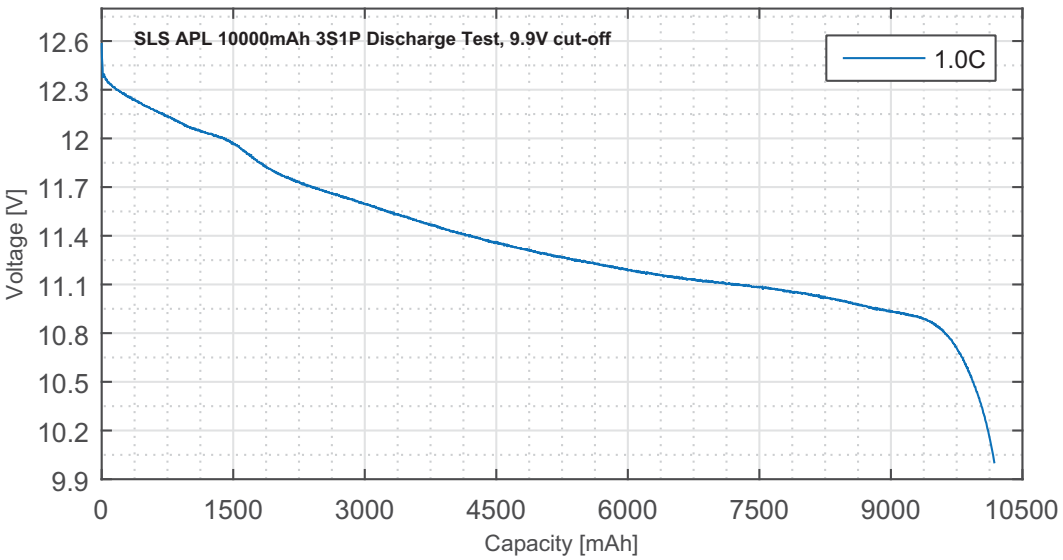



Figure 3.3: SLS APL 10000mAh 3S1P LiPo battery discharge curves (performed with an iCharger 4010 Duo in the laboratory).

$$I_{Bmax\_chg} = C_{Bmax} \tag{3.49}$$

if there is a solar charger connected in parallel, its selection must be carried out with caution under the risk of damaging the battery. In other words, the maximum current provided by the solar charger ($I_S$) must be at least equal to or less than the maximum charging current admitted by the battery, Eq.3.50.

$$I_S \leq I_{Bmax\_chg} \tag{3.50}$$

### 3.2.2 Solar Panel Model

Similarly to a LiPo battery, a solar panel can be connected in many different configurations with respect on its elements, the solar cells. Thus, depending on the application, a solar panel can operate at different voltages by connecting the solar cells to each other in series. This is especially important since the Solar Charge Controller (SCC), Genasun GV-10-Li-12.5V, has a maximum input voltage ($V_{oc}$) of 27V (recommended).
The total number of cells in a solar panel is given by

$$n_S = n_{SS} n_{SP} \tag{3.51}$$

where $n_{SS}$ is the total of solar cells in series and $n_{SP}$ the total number of solar cells in parallel. When the solar pads of a solar cell are not connected, no current is produced and the voltage equals to $V_{oc}$, open circuit voltage. When short circuited, the voltage is zero and the current equals to $I_{sc}$, short circuit current. Between these two points there is a working point, called as Maximum Power Point (MPP)[2], where power takes the highest value possible as shown in Fig.3.4. Hence, the maximum power point voltage ($U_{Smax}$) of a solar panel is defined as

$$U_{Smax} = n_{SS} U_{S1max} \tag{3.52}$$

[2]See Joana (2015) [22] for further info.

where $U_{S1max}$ corresponds to the maximum power point voltage of a single solar cell. Regarding the maximum power output current, it is defined as

$$I_{Smax} = n_{SP} I_{S1max} \tag{3.53}$$

where $I_{S1max}$ corresponds to the maximum power point current of a single solar cell. The maximum power point current ($I_{Smax}$) for a given voltage is a function of temperature[3] and the solar radiation received by the solar panel,

$$I_{Smax} = f\left(U, T, E_e\right) \tag{3.54}$$

where $E_e$ is the irradiance received by a surface per unit area. From Fig.3.5[4] the relation between voltage, current and irradiance for a given temperature is presented.
Lastly, the maximum power delivered by a solar panel is

$$P_{Smax} = U_{Smax} I_{Smax} \tag{3.55}$$

From Eq.3.55, the maximum energy delivered by the solar panel comes as

$$E_{Smax} = \int_{t_i}^{t_i+1} \left(U_{Smax} I_{Smax}\right) dt \tag{3.56}$$

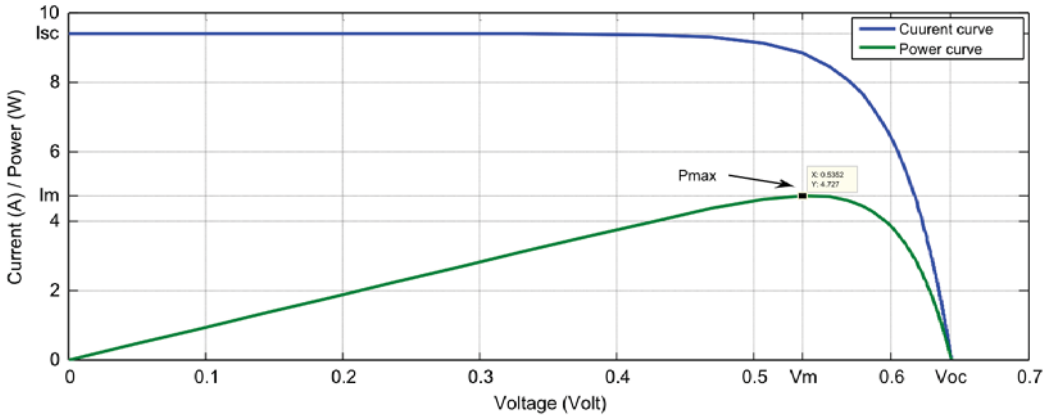



Figure 3.4: Example of a solar cell I-V and P-V characteristic [23].

[3]The higher the temperature, the lower the efficiency of photovoltaic cells.
[4]In the figure $Im$ corresponds to $I_{S1max}$ and $Vm$ corresponds to $U_{S1max}$.

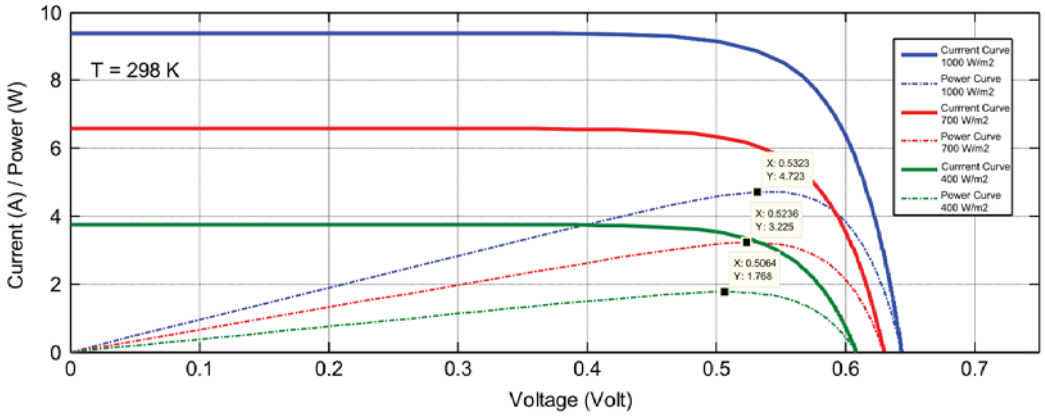



Figure 3.5: Example of a solar cell I-V and P-V characteristic for different irradiations at constant temperature [23].

# Chapter 4

## Energy Algorithm Implementation

In this chapter, the methodology and structure of the computer code behind the mathematical algorithm is explained. In order to have a properly evaluation system, the computational algorithm must have the ability to read input data through sensors or pre-entered parameters, make a state decision, compute the derived data and output the results either by on-board recording or sending it in real time to the user, Fig.4.1.

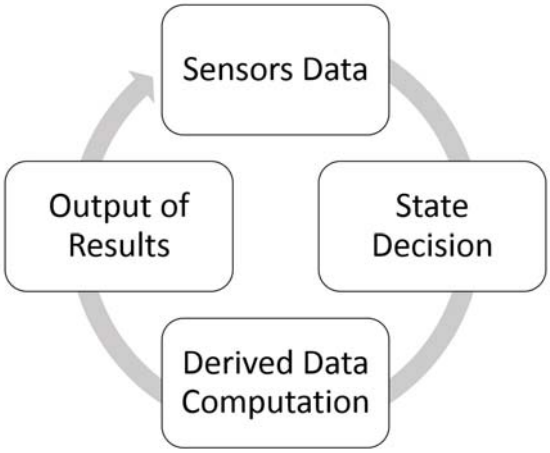



Figure 4.1: Circular flowchart showing the basic sequence of events in a dynamic system.

Considering that the mathematical algorithm is based on the integral of the energy balance, the sequence of events that better express the algorithm is as follows for a given time instant ($t_i$):

1) Check if the system is ready;
2) Get all the pre-defined parameters and sensors data;
3) Obtain potential and kinetic energy;
4) Determine kinetic and potential power;
5) Obtain battery power output;
6) Obtain solar panel power output;
7) Determine the electric energy consumed and received;
8) Determine the real energy left in the batteries;
9) Calculate the total power;
10) Calculate the total energy;
11) Obtain the energy status;
12) Determine the Mission Time Remaining (MTR) or Charging Time Remaining (CTR);

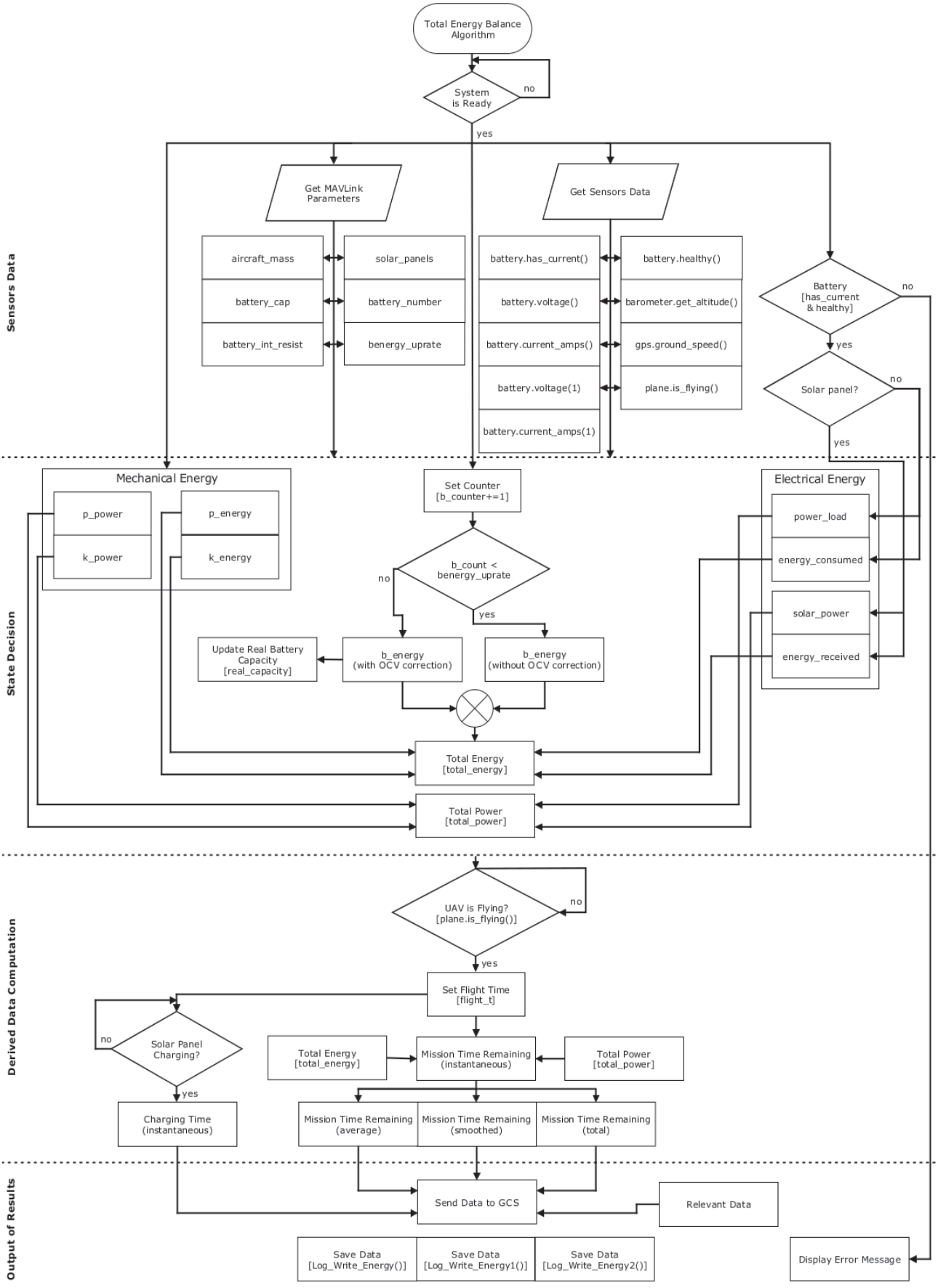



Figure 4.2: Flowchart showing the algorithm structure required to estimate in real time the remaining flight time.

13) Send data to the Ground Control Station (GCS);

14) Save data on-board;

15) Repeat the sequence for the next time instant ($t_{i+1}$).

For a better understanding of the aforementioned sequence of events, below the explanation of each point is presented. In Fig.4.2 a flowchart showing the algorithm structure required to estimate in real time the remaining flight time is also illustrated.

**1) Check if the system is ready**
Regarding the verification of the system's state, this can be divided into two types:

- When the system boots for the first time;
- When the system returns from the last point of the sequence.

If the system boots for the first time some aspects should be considered, otherwise the risk of failure is quite high leading to some unexpected behaviour by the UAV or in the worst case scenario a crash. Thus, it must be verified if the autopilot has booted properly. Since the autopilot can communicate its state via audio signals and LED's, it is fairly easy and quick to make this verification (see [24] and [25] for the meanings of each buzzer sequence and color patterns, respectively). In case of a failed boot, some quick checks may indicate the point of failure which are usually related with corrupted firmware or when the memory card is not inserted in the card socket. If the problem persists the boot logs should be carefully analysed to ensure that the source of the problem is discovered.
After this and before allowing arming (when the UAV is operational) the autopilot checks a set of conditions, that should be verified otherwise the arming is not allowed. From the ArduPilot documentation [26] the arming checks procedure is explained in detail.
With the UAV armed and ready to takeoff one final check needs to be done. With the flight mode switched to *Stabilize*[1] or *FBWA*, the pilot must pitch and tilt the aircraft to confirm that the control surfaces move the correct way to return the aircraft to level flight. In the same way, the pilot should move the control surfaces to make sure that nothing is reversed.
If the system returns from the last point of the sequence, the algorithm will check if the required sensors namely the battery power module, solar power module (if in use), GPS, barometer and airspeed sensor are working properly. If a problem is detected, a warning message is shown at the ground control station and the computation will stop.

**2) Get all the pre-defined parameters and sensors data**
After initialized and armed, the autopilot starts the algorithm by getting all the MAVLink parameters defined by the pilot before each mission (see Subsection 5.1.1 & Appendix A.1). This way and without re-compiling the algorithm, the pilot can change some parameters that feed the system. From Fig.4.2 it can be seen that those parameters are:

- Aircraft gross mass (`aircraft_mass`) - before each flight the UAV should be weighed in order to have a precise value;

[1] Offering basic stabilization, the flight controller has a tendency to maintain the wings levelled. In addition, with the *FBWA* mode enabled, it is not possible to roll/pitch the aircraft past the defined roll/pitch limit.

- Real battery capacity (`battery_cap`) - a complete discharge test followed by a full charge, from 4.2V to 3.2V per cell and vice versa, is performed to determine the real capacity of the flight battery;
- Measured internal battery resistance (`battery_int_ressist`) - in the same way as the discharge/charge test, the internal resistance has to be obtained and defined before each mission;
- Battery number (`battery_number`) - each battery must be numbered in order to avoid failure with respect to `battery_cap` and `battery_int_ressist`;
- Are solar panels installed? (`solar_panels`) - depending on the aircraft setup, solar panels may be installed or not. If solar panels are installed, this parameter should be assigned with 1, if not installed with 0;
- Battery energy update (`benergy_update`) - In 7.3 it is shown that the power modules used to determine the withdrawn/received current are not as accurate as desirable. To overtake this problem, the open circuit voltage (OCV) method (Subsection 2.2.3) is applied in an interval of time defined by the pilot.

In addition to the autopilot and all the required sensors to perform an autonomous flight there are sensors that feed the mathematical model. The battery power module, solar panel power module, GPS and barometer are the ones that contribute directly for the algorithm. For this reason and for a given time instant ($t_i$) each sensor mentioned above is called.

**3) Obtain potential and kinetic energy**

Taking into account equation Eq.3.26 and Eq.3.38, kinetic and potential energy are rewritten respectively as

```
k_energy = 0.5f * g.aircraft_mass * gps.ground_speed() * gps.ground_speed()
```
(4.1)

```
p_energy = g.aircraft_mass * GRAVITY_MSS * barometer.get_altitude()
```
(4.2)

where in Eq.4.1, `g.aircraft_mass` is defined as the aircraft mass and `gps.ground_speed()` the UAV ground velocity. In Eq.4.2, `GRAVITY_MSS` is the gravitational acceleration of the body and `barometer.get_altitude()` the relative altitude from airborne to the earth surface where the UAV was armed. From `gps.ground_speed()` and `barometer.get_altitude()` it can be noted that in both cases the parentheses are empty (i.e. without any number inside) once there is no redundancy of these sensors. If a second GPS is installed the notation should be `gps.ground_speed(1)`.

With both kinetic and potential energy, their time rate of change (kinetic and potential power) can be determined.

**4) Determine kinetic and potential power**

At this point, the time rate of change for both kinetic and potential energy is performed. Eq.4.3 shows the kinetic power equation

```
k_power = -(g.aircraft_mass * gps.ground_speed() * delta_v/delta_t)
```
(4.3)

The ground speed derivation with respect to time is equal to

```
delta_v/delta_t = (gps.ground_speed() - ground_spd_last) * dt_hz
ground_spd_last = gps.ground_speed()
```
(4.4)

where `dt_hz` is the algorithm update rate.
Since `k_power` is highly affected by sensors noise, a filter was used to solve this problem. Named as `_deltk_filter`, this filter is a simple moving arithmetic mean of 5 samples.

```
k_power_avg = _deltk_filter.apply(k_power)
```
(4.5)

At last, Eq.4.6 shows the potential power equation

```
p_power = -(g.aircraft_mass * GRAVITY_MSS * R_Climb)
```
(4.6)

where the relative altitude derivation with respect to time (rate of climb) is equal to

```
R_Climb = (barometer.get_altitude() - barometer_alt_last) * dt_hz
barometer_alt_last = barometer.get_altitude()
```
(4.7)

In the same way to `k_power`, the average of `p_power` is equal to

```
p_power_avg = _deltp_filter.apply(p_power)
```
(4.8)

where `_deltp_filter` corresponds to a simple moving arithmetic mean of 5 samples.

**5) Obtain battery power output**
From Eq.3.45, the battery power output is equal to the product between the battery voltage and the output current which corresponds to

```
power_load = battery.voltage() * battery.current_amps()
```
(4.9)

From Eq.4.9 it can be noticed that `power_load` is a positive value. Henceforth the energy supplied by the UAV to its surroundings takes positive values while the energy received from surroundings takes negative values.
Since the power module that reads battery voltage and current is the main power module there is no need to assign a number inside the parentheses. If a secondary power module is used the notation should be `battery.voltage(1)` and `battery.current_amps(1)`.

**6) Obtain solar panel power output**
Depending on the aircraft setup the user has the ability to indicate whether solar panels are installed or not by assigning the `solar_panels` parameter with 1 or 0, respectively. If solar panels are not installed a MAVLink message (`var_charg`)[2] is sent to the ground station and solar power is considered zero (`solar_power = 0.0f`) as shown bellow.

```
if (g.solar_panels == 0){
    solar_power = 0.0f;
} else {
    solar_power = -(battery.voltage(1) * battery.current_amps(1));
```

[2](See Appendices A.2 and A.3)

```
}
```

If solar panels are installed, then we have

```
solar_power = -(battery.voltage(1) * battery.current_amps(1))
```
(4.10)

which corresponds to the output power of the solar charge controller.

**7) Determine the electric energy consumed and energy received**

From Eq.3.48, `energy_consumed` corresponds to the `power_load` integration over the time.Thus, knowing that Eq.4.11 translates the integral according to the computational language, we have

```
energy_consumed += power_load / dt_hz
```
(4.11)

In the same way, energy received is equal to

```
energy_received += abs(solar_power) / dt_hz
```
(4.12)

where += represents the increment of the previous variable plus the current one.
As explained in Section 2.2 the aforementioned method (based on the ampere hour counting) can add up a large error over the integration since the method is an open loop SoC estimator. Due to this, however small is the associated error with voltage and current readings (named as power modules) the system will have an energy balance significantly different from the calculated by the algorithm.

**8) Determine the real energy left in the batteries**

To overcome the problem mentioned in the previous point, a real battery energy correction (`benergy_recalc`) is constantly fulfilled in an interval of time defined by the pilot, (`benergy_update`). This correction is accomplished through the correspondence between the battery open-circuit battery voltage reading and the battery discharge curve.
In order to better represent the discharge curve shown in Fig.3.3, this was decomposed into a function of three branches as shown below.

```
if (g.battery_number ==3){
    if (battery.voltage() >= 11.65f){
        consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
            12.33966f) / (-0.00027f);
    } else if (battery.voltage() >= 10.85f && battery.voltage() < 11.65f){
        consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
            11.89127f) / (-0.00011f);
    } else {
        consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
            18.84327f) / (-0.00084f);
    }
}

real_capacity = g.battery_cap - consumed_capacity;

//If is case of that set real_capacity = g.battery_cap.
if (real_capacity > g.battery_cap){
```

```
    real_capacity = g.battery_cap;
}
```

Therefore, depending on the flight battery, `battery_number` changes as well as the coefficients to be used in each branch. Since after 12.4V the voltage rises sharply without a near increase of battery capacity, it can be considered that at 12.4V the system reaches a charged/almost charged state.
For a detailed review about this part of the code please see Appendix B.

**9) Calculate the total power**

Intuitively from Eq.4.13 the total power of the system corresponds to the sum of potential power, kinetic power, required power (`power_load`) and received power (`solar_power`).

$$\texttt{total\_power} = \texttt{\_tpavg\_filter.apply(p\_power\_avg + k\_power\_avg + power\_load + solar\_power)} \tag{4.13}$$

Once again a simple moving arithmetic mean of 5 samples has to be used otherwise the flight time estimation becomes difficult to read due to its high fluctuation. In addition, the high fluctuation of time may also induce erratic behaviour by the pilot due to the false sense of remaining flight time.

**10) Calculate the total energy**

The total energy of the system corresponds to the sum of battery energy, solar energy, kinetic energy and potential energy. From Eq.4.14, `benergy_recalc` corresponds to battery energy and solar energy sum, Appendix B.

$$\texttt{total\_energy = benergy\_recalc + kenergy + penergy} \tag{4.14}$$

Although there is no direct application of a buffer in the previous case, the most relevant term of `total_energy` is already damped. This term, `benergy_recalc` was damped with a simple moving arithmetic mean of 150 samples which at an acquisition rate of 5hz, gives a time span of 30 seconds.

**11) Obtain the energy status**

For a given time instant and depending on the battery status, a variable named `var_mtr` that ranges from 0 to 3 is assigned. Hence, depending on the number of `var_mtr` a different energy state is obtained. If `var_mtr = 0` the main battery is being discharged, if `var_mtr = 1` the main flight battery is being charged, if `var_mtr = 2` the main flight battery is fully charged and if `var_mtr = 3` the system is in standby until the aircraft starts the takeoff run.

```
if (plane.is_flying()){
...
  //Calculation of mission_time_remaining for different flight phases. Charging,
      discharging, charged
  if (g.solar_panels == 1 && (abs(solar_power) > power_load) && batt_charge_v <
      solar_charge_vend){
       var_mtr = 1;                    //charging flight battery
  } else if (g.solar_panels == 1 && (abs(solar_power) > power_load) && batt_charge_v >=
      solar_charge_vend){
```

```
        var_mtr = 2;                    //flight battery charged
   } else {
        var_mtr = 0;                    //discharging flight battery
   }
...
} else {
    var_mtr = 3;
}
```

**12) Determine the Mission Time Remaining (MTR) or Charging Time Remaining (CTR)**
At this stage `total_energy` and `total_power` are known. From Eq.4.15 the mission time remaining (`mtr`) is equal to the ratio of `total_energy` and `total_power`. For convenience `mtr` is converted from seconds to minutes. It is also established that any values of mission time remaining (with damping) presented at the ground station are based on `mtr`[3].

```
mtr = total_energy / total_power / 60.0f
```
(4.15)

As explained above when `var_mtr = 1` the main flight battery is being charged and it is desirable to know the charging time remaining (`charg_trem`). This value expresses the remaining time until the battery has a full charge. Consequently, when `charg_trem = 0` a full charge was achieved and `var_mtr = 2`. Eq.4.16 shows how this value is obtained

```
charg_trem = (benergy_max * param_cap - (benergy_recalc * param_cap)) /
             (abs(solar_power) - (power_load)) / 60.0f
```
(4.16)

where `benergy_max` corresponds to the maximum battery energy (full charge) and `param_cap` is the correlation between the battery capacity and the solar charger end voltage, $U_{SCend} = 12.5$ V.

**13) Send data to the Ground Control Station (GCS)**
With all data already processed on-board it is advantageous to send this data in real time to the pilot. For such, this information is sent through two types of telemetry: Telemetry 1-Micro Air Vehicle Communication Protocol and Telemetry 2-Video Link, respectively. Both cases are covered in detail in Section 5.1.

**14) Save data on-board**
Dataflash logs are stored on the Pixhawk on-board dataflash memory and can be downloaded after a flight. On a Plane or a Rover, dataflash logs are created soon after the start-up. Thus, with the use of a Ground Control Software (e.g. MavProxy), it is possible to debug any errors that can occur during the testing phase of the system. Each time the system is executed, three data packets with respect to the algorithm are saved as `Energy`, `Energy1` and `Energy2`. Appendix C shows the messages written for each data packet.

**15) Repeat the sequence for the next time instant ($t_{i+1}$)**
Supposing that the sequence of events shown above are for a given time instant ($t_i$), every 200 miliseconds (5 Hz) the system will repeat this action. It should be noted that although the

[3] (See Section 5.2)

entire calculation is computed at 5Hz (data packets `Energy` and `Energy1`), a part of the code was extended to run at 0.5Hz (data packet `Energy2`). In specific cases where autopilot overloading may occur (due to abnormally high processing), the autopilot will first perform the primary tasks and then the secondary tasks as the case of the proposed algorithm[4].

[4]With the actual autopilot version (Pixhawk) this problem has not yet been reported by the ArduPilot development team.

# Chapter 5

# Data Transfer and Presentation

Founded by Chris Anderson in May 2007, the ArduPilot project is an open source unmanned aerial vehicle platform supporting multi-copters, helicopters, fixed wing aircraft and rovers. In the same way more recently was implemented the ArduSub for remotely operated underwater vehicles (ROV's). Hosted on GitHub and developed by a large community of diverse professional engineers and computer scientists denominated as ArduPilot Development Team, the source code is still being changed and improved nowadays [27] & [28]. Developers who want to contribute just have to "fork" the project, create a "branch" on their fork with new features, and then raise a pull request to get the changes merged into the "master" project.
Although the main flight code is written in C++, some support tools are written in a variety of languages, most commonly in Python. Apart from that, it was also written on top of the AP-HAL (Hardware Abstraction Layer) making it possible to port the code to a wide range of autopilot boards (see Section 6.1). Fig.5.1 shows a high-level view of the ArduPilot architecture for the ArduCopter main code which shares the same libraries as the ArduPlane and ArduRover.
Receiving inputs from a 3-axis gyroscope, accelerometer, magnetometer, barometer, GPS and air speed sensors, it provides outputs to all control surface servos allowing a fully autonomous navigation, control of altitude, speed and direction. It is possible to set a fail-safe for loss of radio contact, GPS or breaching a virtual boundary around the area to fly that goes from a return to launch action to a loiter command around a predefined waypoint in space. Besides the possibility to fly up to 13 different flight modes in a fixed wing UAV, from fully manual to fully autonomous, it also allows automatic take-off, landing, stall prevention and terrain following. For more detailed information about the aforementioned flight features it is recommended to see references [29] and [30].

## 5.1 On-Board Data Transmission to the Ground Control Station

Before going into detail with regard to the types of data transmission from air to ground and their implementation, it is extremely important to note the key concept behind this.
Succinctly, the concept of transmission of on-board data to a ground-station or similar device is named as telemetry. Telemetry is the automatic measurement and transmission of data from remote sources[10]. In general, sensors at the source measure either electrical data (such as voltage or current) or physical data. Commonly transferred via wireless, for example radio, ultrasonic or infrared systems, it also covers data transferred over computer networks, optical links or other wired communications.
Depending on the application, there are several options to implement such a feature. For the current work, telemetry is only associated with wireless data transmission.

### 5.1.1 Telemetry 1: Micro Air Vehicle Communication Protocol

Having the need to receive data from the UAV to the GCS (and vice-versa) a telemetry communication protocol should be used or implemented. For this function the Micro Air Vehicle

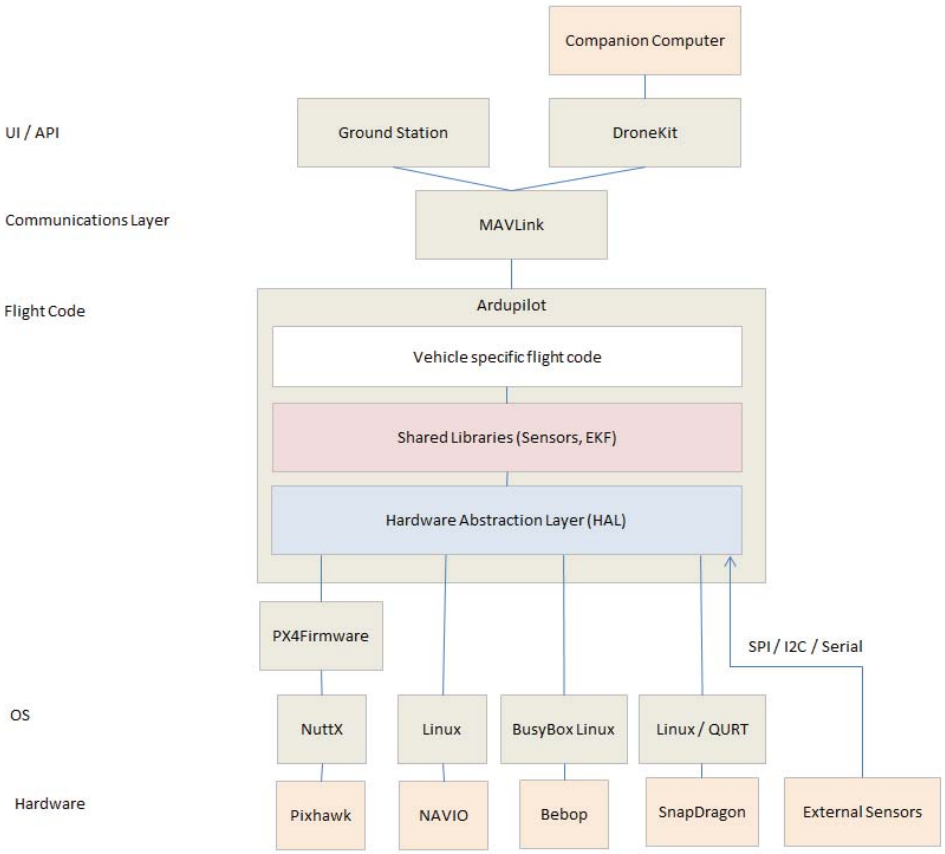



Figure 5.1: High-level view of the Ardupilot architecture (for ArduCopter firmware).[31].

Comunication Link (MAVLink) was considered since is a very lightweight, header-only message marshalling library for micro air vehicles. It is currently being used as the communication protocol for many autopilots[1] like ArduPilotMega (from APM 1 to Pixhawk 2), pxIMU, PX5FMU, SLUGS, SmartAP, AutoQuad 6 (etc.) as well as for a wide range of software packages (mainly for ground control stations) like Mission Planner, APM Planner, MAVProxy, QGroundControl, UgCS (etc.). MAVLink messages are defined in XML (eXtensible Markup Language) and then converted to C/C++, C# or Python code depending on the autopilot and GCS language. Since new data needs to be sent from air to ground, Appendices A.2 & A.3 show a set of three new MAVLink data packets added to the *ardupilotmega.xml* file.
In order to create a valid message it should be noted that the ground control station is only able to read messages defined by the XML files with the following structure:

- Each message is encapsulated by <*message*></*message*>;
- *id="0"* means the message has an id/index number zero. Valid numbers range from 0 to 255, with id's from 150 to 240 being reserved for new additional messages;
- *name="HEARTBEAT"* encodes the human-readable name. The name is only used in the code and not transmitted. The system itself only refers to the ID;
- <*description*></*description*> is a very important, but optional field. This description is shown in user interfaces and in code comments. It should contain all information to fully understand the message;

[1]Only referenced those which use MAVLink as main protocol.

- <*field*></*field*> Encodes one field of the message. It is similar to one variable in a C-struct. Fields can be integers of 8, 16, 32 and 64 bit length (signed and unsigned) and single/double precision IEEE754 floating point numbers;

- *type="uint8_t"* Defines this field as unsigned integer with 8 bits size. Arrays can be defined like *type="uint8_t[5]"* for an array of size 5. This field is read-only and is automatically filled by MAVLink during the transmission. It allows the receiver to decode the protocol version.

### 5.1.2 Telemetry 2: Video Link

In addition to the data sent by MavLink, it proves to be useful the broadcast of on-board video. This way the pilot is able to see what the "UAV sees" and can have an active role in the course of the mission if necessary. Although the working principle is the same as Telemetry 1 (see Subsection 6.2.6 & 6.2.7) this one does not have such a complex communication protocol. The on-board transmitter broadcasts a specific frequency that is received at the ground and converted into the form of video. Furthermore, using an On Screen-Display (OSD) board some relevant flight information can be overlaid with the video. Depending on what is desired this information can go just from a simple battery voltage to speed, altitude, position, orientation, current drawn, flight mode and so on.
For the present work, messages like remaining flight time (Time inst & Time avg), required power (Power load) and received power (Solar power) were also added into the video link due to its importance (see Fig.5.5b).

## 5.2 Reception and Data Visualization at the Ground Control Station

The capabilities added by the Ground Control Station Software (GCSS) and On-Screen Display (OSD) are described below. Its improvements regarding the user experience, are also addressed.

### 5.2.1 Ground Control Station Software

A ground control station is typically a software application, running on a ground-based computer, that communicates with the UAV via wireless telemetry. It displays real-time data of the UAV's performance and position as well as can also be used to control a UAV in flight, uploading new mission commands, setting flight modes and parameters.
Created and maintained by Tridgell [32] the chosen GCS was MAVProxy, Fig.5.2, since it is a light-weight fully-functioning open source GCSS for UAV's written in python and extensible via python modules. It is portable making it possible to be used in any POSIX OS (Portable Operating System Interface) such as Linux, OS X or Windows.
The way the pilot interacts with the Ground Control Station Software (GCSS) is also a very important aspect to be considered in the current work. Therefore, the original MAVProxy's GUI (Graphical User Interface) was modified to improve the user experience when using the system. This modification was based on the addition of information in the main console and the creation of a module called Energy Module in which the user has the same data available in a graphical

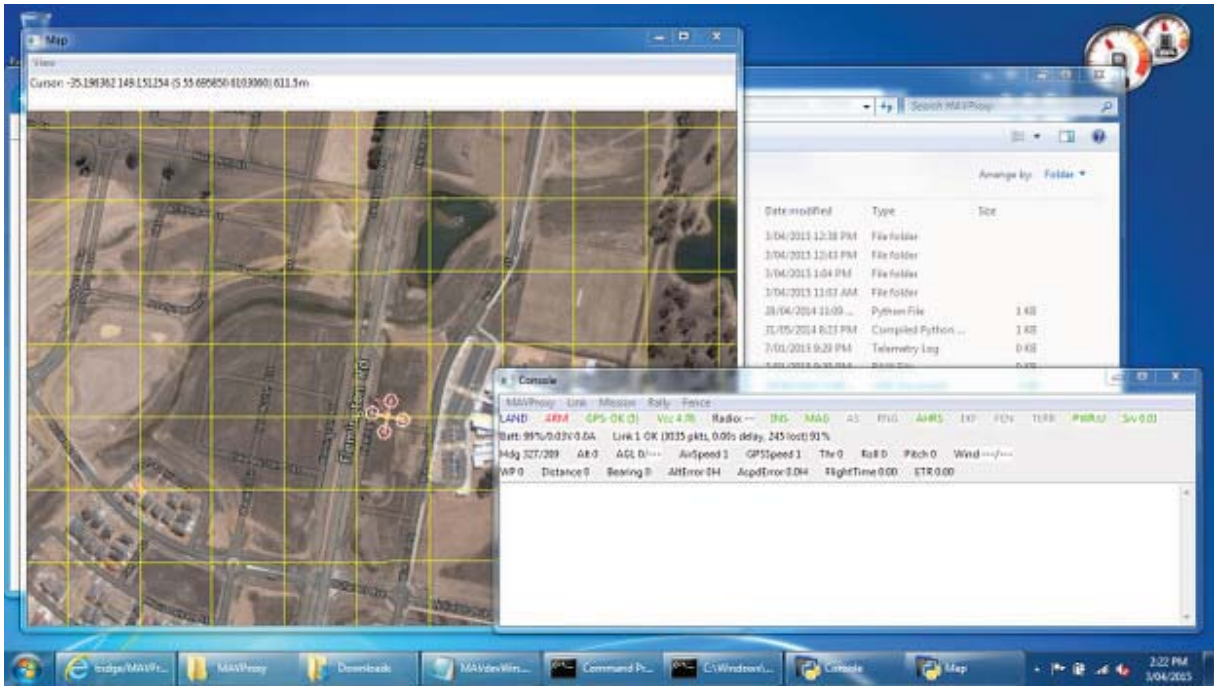


Figure 5.2: MAVProxy running under Windows®. Map Module loaded in the left and main Console Module in the right [33].

form, Fig.5.3 & Fig.5.4. To achieve this, it was necessary to expand/modify the source code using an Integrated Development Environment (IDE) software, Microsoft Visual Studio 2013.

**1) Console Module**

Since it is in this module that the most relevant information is shown to the user, the following messages were added:

- Time Counter - When `is_flying` action is detected (due to a change of airspeed, altitude and position) a timer is activated. If this time is less then one hour, the result will be given as minutes:seconds. If greater then one hour, the time will be given as hours:minutes.
- Power Load - Depending on the UAV, propulsive system, flight phase and flight speed, the required power is different. Due to this, it proves to be very useful to show this information. The result is shown with two decimal places and “W” in the final representing watt.
- Solar Power - If the aircraft carries solar panels, the solar power converted by the solar charge controller (see Subsection 6.2.2) will be shown. In the same way as the previous point, depending on the solar panel configuration, irradiation and aircraft orientation with respect to the sun this power will be different and it is important to observe its fluctuation with the course of the mission. The result is shown with two decimal places and “W” in the final representing watt.
- Energy Flow - Depending on the energetic state of the aircraft four different messages can de displayed: DISCHARGING, TOO LOW, CHARGING or CHARGED.
  - DISCHARGING: If `abs(solar_power) < power_load` the energy balance is negative, more power is being consumed than received. When this state occurs, the flight battery is being discharged and a decrease of remaining flight time should be expected.

Since the result of Eq.4.15 is quite unstable a simple moving arithmetic mean of 50 samples has to be used, `_trem_filter.apply(mtr)`. Another problem verified regarding the reading of `time_rem` focuses on the instants following an abrupt change of `power_load/solar_power` since without any adjustment, the estimation changes in the form of steps (very abrupt). One way to solve the problem is to make a simple arithmetic mean of `_trem_filter.apply(mtr)` with the previous value of `time_rem` as shown bellow.

```
//Mission time remaining (discharging)
time_rem = (_trem_filter.apply(mtr) + time_rem) / 2
```

- TOO LOW: When `time_rem` is less then five minutes the "(time)/DISCHARGING" message changes to "TOO LOW". At this stage, the pilot should land the UAV as quick as possible since it has more five minutes of estimated flight time (maintaining constant the previous flight conditions);
- CHARGING: If `abs(solar_power) > power_load` the energy balance is positive, more power is being received than consumed. When this state occurs, the flight battery is being charged and an increase of remaining flight time should be expected. Again, since the result of Eq.4.16 is quite unstable a simple moving arithmetic mean of 50 samples is used.

```
//Battery charging time
chg_trem = _chgtrem_filter.apply(charg_trem)
```

- CHARGED: When `abs(solar_power) > power_load` and the battery voltage is greater or equal to a know voltage[2] (`batt_charge_v >= solar_charge_vend`) the battery is considered fully charged and the message changes from "(time)/CHARGING" to "CHARGED".

- Global RTime - This message represents the global average of `time_rem`. In other words has equal meaning as the average fuel consumption in a car (full tank).
- RTime - With a different rate of acquisition (0.5Hz) this message is based on the `time_rem` value. In this case a simple moving arithmetic mean of 60 samples is used. Hence, the result is a time span of influence with 120 seconds.
- AvgRTime - Through several tests it was concluded that the ideal time span of influence would be 5 minutes. However, depending on the mission and aircraft, this value may be changed. The result is based on a simple moving arithmetic mean of 150 samples (0.5Hz) of `time_rem`.

[2] (See Subsection 6.2.2)

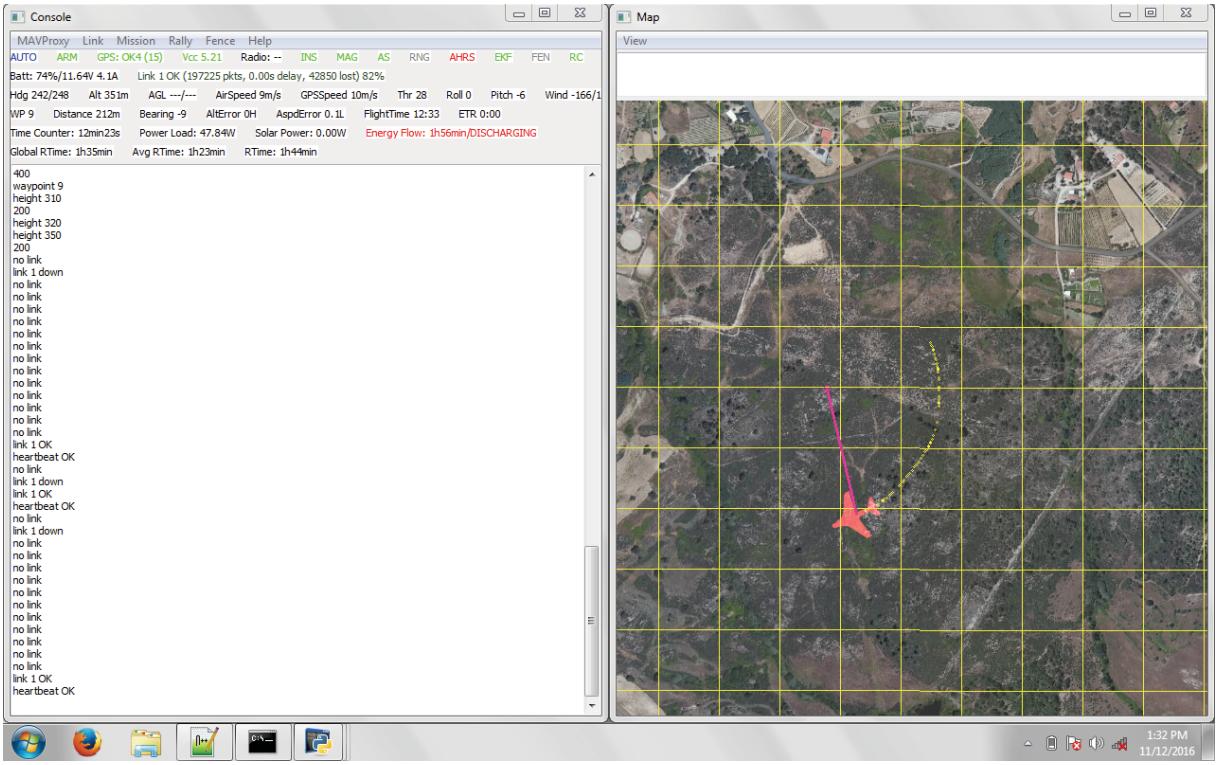



Figure 5.3: MAVProxy running under Windows®. Main Console module customized with new relevant data.

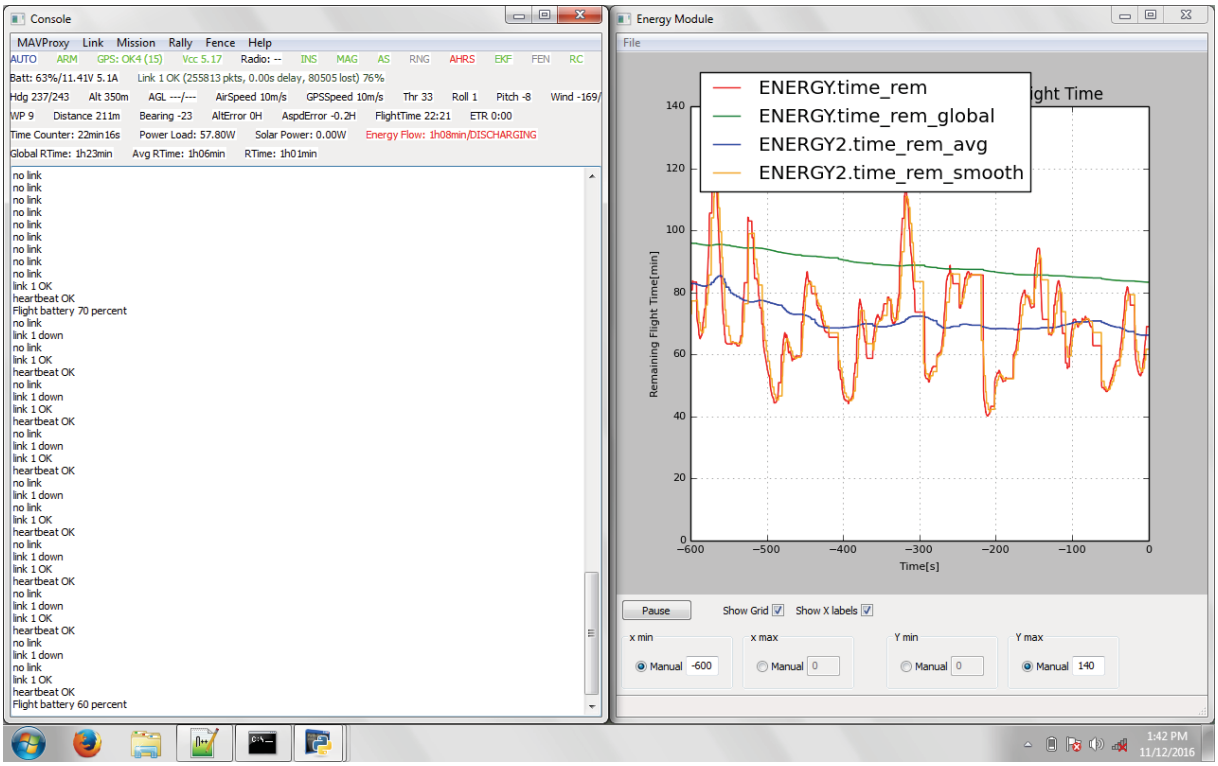



Figure 5.4: MAVProxy running under Windows®. Energy Module loaded on the right window.

**2) Energy Module**

Regarding the Energy Module, Fig.5.4 (right window), its advantage was verified several times in the field since it is faster and more practical to check the various remaining flight time predictions in a graphic mode.

This module has the ability to save the graphic image in JPEG extension, adjust both x and y-axis, pause the data reception and show/hide a graphic grid. To avoid overloading the ground control station, the xmin-axis is limited to 10 minutes (600 seconds) which at a reception rate of 1Hz (see Subsection 6.2.6) corresponds to 600 samples in buffer. Depending on the purpose, there is the ability to remove or add messages to the module.

### 5.2.2 On-Screen Display

An on-board video camera was used to record aerial video and/or directly transmit the live video back to the pilot (Subsection 5.1.2), which is called FPV (First Person View). Since the connection was lost several times in a short period of time, the weak capacity to maintain communications when using Telemetry 1: MAVLink (2.4Ghz) can be observed in Fig.5.3 and Fig.5.4. When this situation occurs a "no link" and "link 1 down" message is displayed to inform the pilot. Since Telemetry 2: Video Link uses a lower frequency (1.3Ghz) a better communication capacity over long distances can be expected (see Chapter 6). Thus, it is essential to ensure that all the relevant information for the mission is overlaid into the video link since it will be theoretically the last to fail.

Created by Alves, (2017) [34], the AlceOSD is a graphical OSD that uses the MAVLink protocol. Being an open source code, it is still in expansion and new functions are expected to be added.


(a) Long endurance flight without solar panels using an AlceOSD (standard configuration).

(b) Long endurance flight with solar panels using an AlceOSD (custom configuration)

Figure 5.5: First Person View using the AlceOSD to overlay important data into a graphical mode.

Listed from (1) to (20) in Fig.5.5 the main capabilities of this OSD are shown:

(1) Battery Percent: Indicates the charge level of the main battery. This value is shown in percent (Power Module is required, Subsection 6.2.3);

(2) Voltage/Current: Shows the main battery voltage and current draw in the present instant (Power Module is required, Subsection 6.2.3);

(3) Throttle Percent: Indicates the amount of throttle in percent;

(4) Barometric Altitude: Indicates the aircraft height above the home position in meters;

(5) Home Distance: Indicates the aircraft distance from home in meters (GPS is required, Subsection 6.2.1);

(6) Heading Home: Indicates the direction to home in degrees;

(7) Airspeed: Indicates the airspeed, which is speed of an aircraft with respect to the air in m/s (Airspeed Sensor is required);

(8) Radar: Shows the home and take-off direction. If both take-off and landing are executed in the same direction, it will give to the pilot a notion of an imaginary landing spot (ILS) in the radar;

(9) Flight Mode: Indicates the current selected flight mode from Pixhawk;

(10) GPS Coordinates: Shows the exact GPS coordinates (GPS is required, Subsection 6.2.1);

(11) Artificial Horizon: Shows the artificial horizon, pitch and roll in degrees;

(12) Heading Compass: A simple compass that shows the real north;

(13) RSSI: Indicates the radio signal strength (Received Signal Strength Indication). The value is displayed in percent.

(14) Wind Speed: Shows horizontal wind direction and speed in m/s. Since the difference between airspeed and ground speed is the wind speed, both Airspeed Sensor and GPS are required[3];

(15) Absolute Altitude: Indicates the height above sea level in meters (GPS is required, Subsection 6.2.1

(16) Climb Rate: Indicates how fast the aircraft gains or looses altitude in m/min;

(17) RRT Instantaneous: Displays the `time_rem` estimation;

(18) RRT Average: Displays a simple moving arithmetic mean of 150 samples of `time_rem`;

(19) Power Load: Indicates the required power by the aircraft in the present instant (Power Module is required, Subsection 6.2.3);

(20) Solar Power: Indicates the solar power converted by the solar charger in the present instant (Power Module and Solar Charge Controller are required, Subsection 6.2.2 & 6.2.4).

[3] (See Subsection 6.2.1)

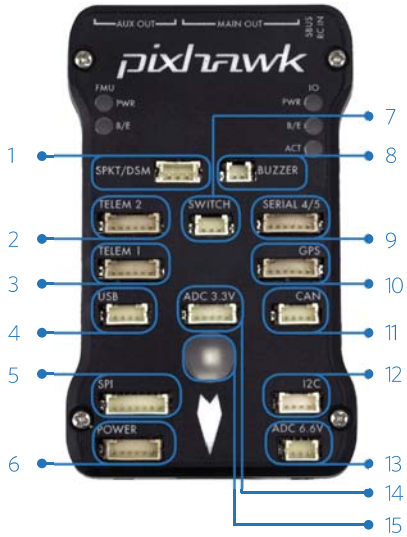
# Chapter 6

# Systems Integration

This chapter provides detailed information about the required hardware to be used in this work as well as its calibration and integration. All hardware that was not used directly in the development of the proposed algorithm is not described.

## 6.1 Flight Controller Selection

The flight controller board used to perform the experimental tests was the Pixhawk designed by PX4 open-hardware project and manufactured by 3D Robotics, Fig.6.1. It has a STM32F427 32-bit primary microcontroller, two megabytes of flash program memory, 256 kilobytes of RAM (random access memory) and a STM32F103 backup failsafe 32-bit co-processor. It is also equipped with a 3-axis 16-bit ST Micro L3GD20H gyro for determining orientation, a 3-axis 14-bit accelerometer and compass for determining outside influences, a MEAS MS5611 barometric pressure sensor for determining altitude, provision for external compass with automatic switch-over, built in voltage and current sensing for battery condition determination, connections for externally-mountable GPS (Global Positioning System) units for determining absolute position and an extensive I/O (input/output) interface with dedicated connectors,[35]. For more detailed information about Pixhawk see references [36] & [37].




1 Spektrum DSM receiver
2 Telemetry (radio telemetry)
3 Telemetry (on-screen display)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator

Figure 6.1: Pixhawk connector assignments,[35].

## 6.2 Sensor Selection

In order to accomplish a real-time estimation of the remaining flight time based on the total energy balance, it is necessary to integrate several sensors in the AutoPilot controller board. A

telemetry modem integration at the ground station is also essential to perform the information forwarding from air to ground and vice-versa.
Together, they allow to have a full autonomous aircraft, capable of doing unmanned missions by following pre-programmed waypoints from the ground control station and return to launch if some fail-safe situation is detected, minimizing any inherent risk related with the operation of this type of vehicles.
The following subsections describe in detail the sensors used and their preparation for integration in the UAV.

### 6.2.1 GPS

An Ublox NEO-M8N (XL) GPS module with an Honeywell HMC5983 compass to provide highly accurate heading data assembled by DROTEK as shown in Fig.6.2 was choosen. This GPS unit is ideal for UAV applications as it is small, lightweight (23g) and delivers an update rate up to 5Hz. It uses the same interface as the telemetry system which is RX, TX, GND and 5V. The GPS plugs directly into the GPS port of the autopilot and transmits the location information to the GCS using the same protocol as the telemetry module.

### 6.2.2 MPPT Solar Charge Controller

The MPPT (Maximum Power Point Tracking) solar charge controller used was the Genasun GV-10-Li-12.5V, Fig.6.3, already presented in previous experimental works, [22]. A complete list of specifications is presented in Appendix E.1.
The use of a solar charge controller with MPPT allows the solar panels to run at their most efficient voltage, applying appropriate resistance to obtain the highest power output possible. When solar input is insufficient to meet propulsion needs, all power is delivered to the motor via the ESC (Electronic Speed Controller). When the output power from Genasun solar charge controller exceeds the required power, this excess power is stored in a 3s LiPo battery operating at a nominal voltage of 11.1V. As explained in Subsection 3.2.1, in this type of batteries, the most common found on small UAV's, the maximum charging current is usually limited to a value equal to their nominal capacity. Once the maximum rated output current from the charger is 10.5A, a battery with a nominal capacity of 10.5Ah should be considered otherwise serious damages can occur. Genasun also specifies the use of a battery management system to maintain cell balance, however in a primarily phase this was not considered, so no additional components were required.


Figure 6.2: Ublox NEO-M8N GPS + HMC5983 magnetometer module (XL).


Figure 6.3: Genasun GV10-Li-12.5V MPPT controller with heat-shrink tubing.

### 6.2.3 Power Module: Motor

To accurately measure battery voltage and current consumption a power module from 3D Robotics was used, Fig.6.6a. The power module plugs directly into the power port of the autopilot and besides to allow power consumption measurement, it was also used to provide a stable voltage (5.37V) and current (2.25A) to Pixhawk, reducing the changes of a brownout. Although it may be used to provide power to other small electronic modules, in addition to the autopilot, servos or other hight current devices must be powered by a separate ESC or UBEC. It is also limited to a maximum input voltage of 18V (up to a 4s LiPo battery) and a maximum current of 90A. In Rodrigues, 2017 [38], this was already taken into account and a full scheme of all connections and components used at the flight test UAV are shown.

#### 6.2.3.1 Power Module Calibration

Although the power module was configured from factory, a manual calibration was performed so that the readings of battery voltage and current were the most accurate possible. Since these current sensors are not very accurate at low currents, less than 3A, the manufacturer suggests to perform a manual calibration at around 10A. This calibration is usually done using an external power analyser which is connected between the power module and the LiPo battery. Then, the user raises the throttle until the desired current is achieved according to the power analyser. Once a very small variation of throttle position leads to a very abrupt current variation, a different approach was used. Thus, instead of using throttle position to achieve the desired amount of drawn current, a LiPo charger was programmed to discharge at a constant rate of 10A. This way, any small current variation is mitigated leading to a more effective calibration. In Fig.6.4 all the components needed to calibrate the power module are shown. A brief description of those is presented below:

1) Laptop: The Mission Planner open source software [39], a full-featured ground station, was used to update the new values of *Measured battery voltage* and *Amperes per volt* in the autopilot, Fig.6.5a. Besides that, to save and analyse the data from the charger/discharger, a data acquisition software from LogView® was used [40], Fig.6.5b;

2) Discharger: An iCharger 4010 Duo charger/discharger, from the New Zealand manufacturer iCharger® was also used [41]. The main feature of this charger is its ability to charge/discharge at least two different batteries at the same time since it has two independent channels. Furthermore, it is capable to discharge a battery at 130W per channel (without the use of regenerative systems) witch is enough for this setup[1];

3) Power Supply: A switching mode DC (direct current) power supply from Diamond Antenna®, GZV6000 [42], rated with 60 A continuous and a selectable voltage up to 15 VDC (volts of direct current) was used to power the charger/discharger;

4) Power Module: Being the component to calibrate it was connected to the Pixhawk and LiPo battery;

5) Multimeter: In order to read the battery open circuit voltage and set the new value of *Measured battery voltage* at the Mission Planner a Vici VC97 multimeter [43] was used.

[1] At least, a discharger capable of 126W is required if a fully charged 3s1p 10000mah LiPo battery is used.

6) Pixhawk: With the Pixhawk connected to the laptop it was possible to update the new values of *Measured battery voltage* and *Amperes per volt* in the Mission Planner;

7) LiPo Battery: A fully charged 3s1p (12.6V) 10000mAh LiPo battery from SLS® [44] with a discharging rate of 25C (continuous discharge) and 40C (short-term discharge) was used as discharging element.

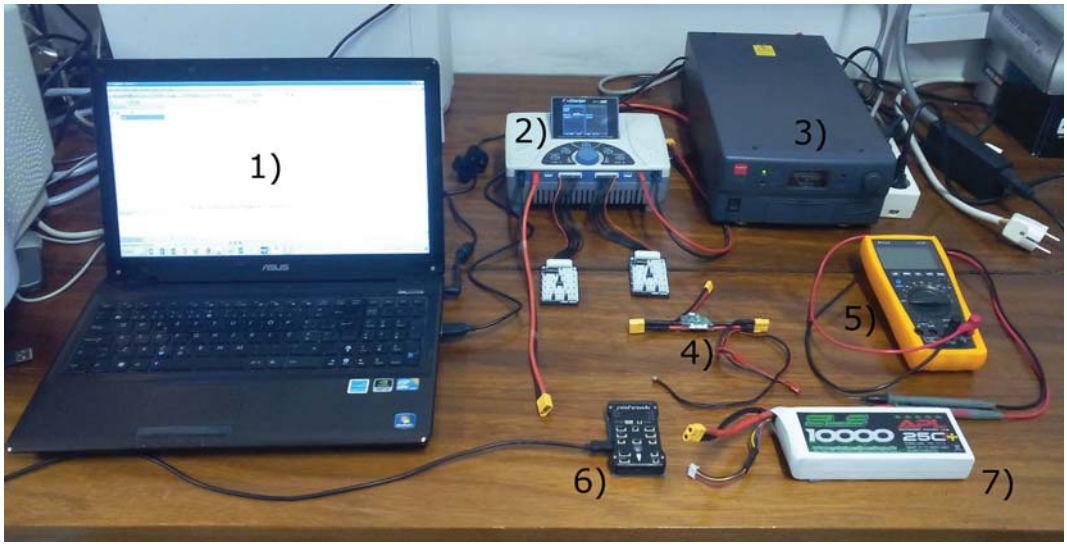
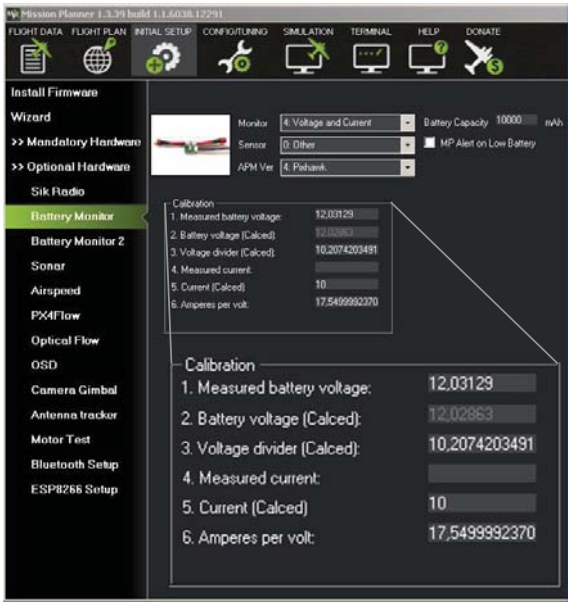
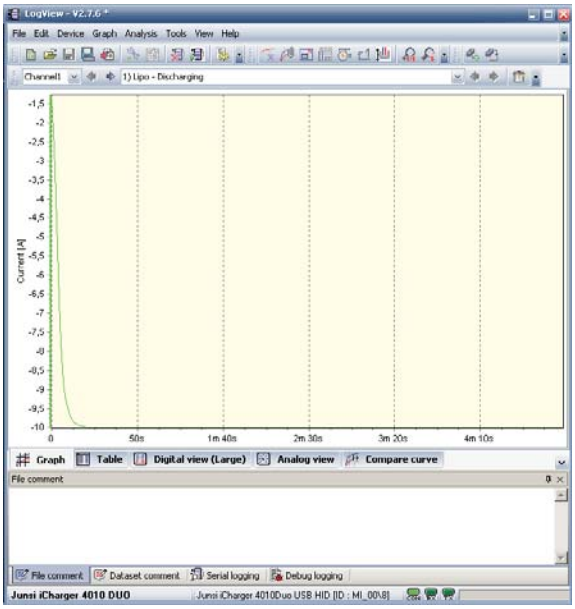



Figure 6.4: Arrangement of the components required to perform the power module calibration.




(a) Mission Planner software showing the new values of *Measured battery voltage* and *Amperes per volt*.



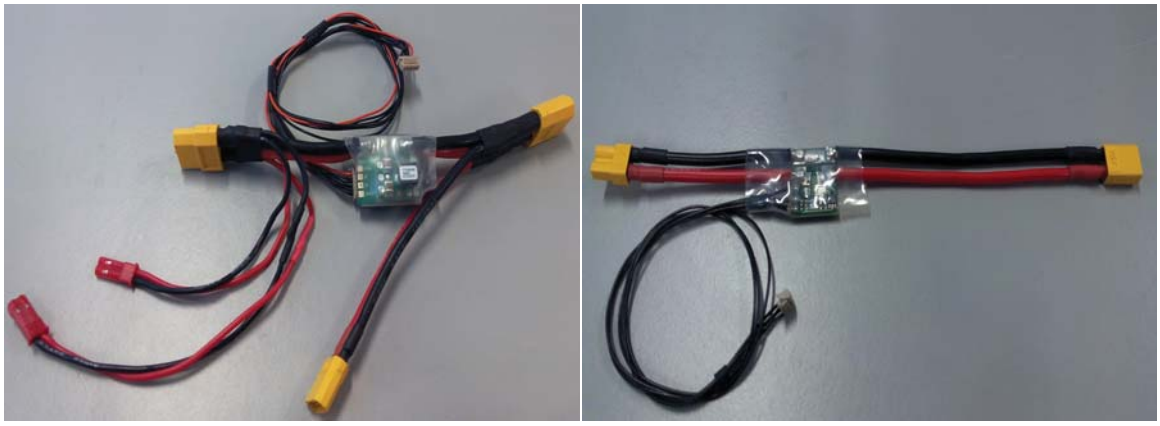

(b) LogView software showing a stable 10A current being drawn from the LiPo battery.

Figure 6.5: Screenshot of both software used to carry out the power module calibration.

### 6.2.4 Power Module: Solar Panels

If the proposed algorithm is used in an aircraft equipped with solar panels, it is necessary to measure the power delivered by the solar charge controller. Therefore, in addition to the

main power module a second power module from 3D Robotics was used, Fig.6.6b. Regarding its calibration, this was executed in the same way as the main power module.
This power module plugs into the ADC 3.3V port of the autopilot and only allows voltage and current measurement[2].



(a) Main power module from 3D Robotics used to power the Pixhawk, measure battery voltage and current consumption.

(b) Secondary power module from 3D Robotics which is used only to measure voltage and current from the MPPT charge controller.

Figure 6.6: Picture of both power modules that feed the algorithm.

### 6.2.5 Barometer

As already stated in Section 6.1, Pixhawk comes with several embedded sensors. One example of that is the MS5611 barometer module [45] from MEAS® Switzerland. The MS5611 is capable to communicate via SPI or I²C bus interface, comes with a hight precision barometric pressure sensor module, which gives it an altitude resolution of 10cm. It is capable of converting the uncompensated analogue voltage to a 24bit digital value, as well as providing a 24bit value for the temperature. Apart from this, it can be interfaced to virtually any micro-controller making it a viable choice in small autopilots.

### 6.2.6 Telemetry 1: Micro Air Vehicle Communication Protocol

The telemetry system is the communication tool between UAV and GCS. It is a bi-directional communication system as the autopilot is able to transmit all the flight information as well as the ground operater is able to send commands and actions to the autopilot. A Laird wireless RM024 series 2.4GHz range-amplified multipoint (RAMP) transceiver module coupled to an adapter board from SparkFun was chosen as telemetry system, Fig.6.7. Based on Laird LT2510 core technology, it is available in two versions, one with 125mW maximum conducted output power and other with 10mW maximum conducted output power. To ensure a system capable of long range communications, the version with 125mw was selected. These modules are identical except for output power, transmit power consumption, and the number of RF (Radio Frequency) channels available. Deciding which RF data rate to choose depends on the purpose. Faster RF data rate delivers much faster throughput, but less range will be noticed. Thus, a compromise of throughput and range should be achieved.

[2]The flight controller is always powered by the main power module regardless of whether more power modules are being used.

#### 6.2.6.1 Ground System

According to Coleman et al.[46], every 6dBi of extra antenna gain will double the usable distance of an RF signal. Therefore, high-gain antennas allow more of the transmitted power to be sent in the direction of the receiver, increasing the received signal strength. Taking this into account, a 2.4GHz 8dBi patch antenna was connected to the transceiver. Due to the high directionality of the antenna, the use of an antenna tracker or a ground station operator to redirect the antenna to the UAV whenever necessary should be considered, for example.

#### 6.2.6.2 Air System

Regarding the system on board the UAV, a transceiver was also installed but this time connected to a 2.4GHz 2.5dBi low gain antenna (linear polarization). This setup is used because the aircraft is a moving point in space, thus to make use of a high gain antenna, a ground station tracker should be used which is neither practical or appropriate.

### 6.2.7 Telemetry 2: Video Link

Since the 2.4Ghz band was already being used for Telemetry 1, a 1.3Ghz video system from RMRC was selected to broadcast the on-board video. Thus, while the antenna selection is more limited than the 2.4GHz band it offers more capacity to penetrate through solid objects and the RF environment is often more friendly[3]. Hence, a greater range can be expected when compared with the 2.4Ghz band.

However, special attention should be taken into account when using this band with 2.4GHz or 433MHz for control. When using 2.4GHz for control, a low pass filter or a notch filter must be placed on the video transmitter in order to cut the harmonic interference that decreases the control range. When using 433MHz for control, a low pass filter or notch filter should be used on the 433MHz radio controller to keep the 3$^{rd}$ order harmonic from disturbing the video signal.

Regarding the antenna selection, circular polarized antennas were considered instead of linear polarized antennas. Circular polarization has two distinct advantages over linear polarization: it rejects multipath interference (which is the most common reason for video issues) and does not lose polarization when the plane banks to make a turn.

#### 6.2.7.1 Ground System

At the ground control station an ORACLE video diversity controller is installed , Fig.6.8. Mainly used in situations where the transmitter is always moving around, this device can bridge two wireless video receivers together. This diversity controller uses a sophisticated real-time video analysis technique in which the video display is switched to the best receiver based on its video quality (strongest video signal). Therefore, two video receivers from RMRC with the same band/ frequency as the transmitter (1.3Ghz) were used and connected into the diversity controller. Connected to the receivers are a low gain skew planar wheel antenna (1.99dBi) and an hight gain 3 turn helical antenna (7.5bBi). Both antennas use circular polarization.

[3]There are several uses of the 2.4 GHz band such as: cellphones, bluetooth devices, microwaves and Wi-Fi networks (among others) which may decrease signal quality.

#### 6.2.7.2 Air System

With respect to the air side (on-board), it has a 1.3Ghz 800mW video transmitter. Powered in-line with the flight battery this video transmitter requires a 12V step-up/step-down voltage regulator since the input voltage vary greatly during the course of the mission. A circular polarized low gain skew planar wheel antenna (1.99dBi) is connected to the transmitter for the same reason as a linear polarized low gain antenna was connected into the Telemetry 1 transceiver (on-board).


Figure 6.7: Laird LT2510 RM024-P125-C-01 (125mw power version).


Figure 6.8: Oracle video diversity controller used to bridge two video receivers.

## 6.3 Aircraft Hardware Integration

Having all the required sensors and components selected, Fig.6.9 shows them connected to be installed in the UAV[4]. This way and to facilitate the understanding of the figure, the description of each component follows below: 1)FPV camera; 2)Telemetry 2 video transmitter/antenna; 3)Pitot tube; 4)Microphone; 5)12V step-up/step-down; 6)Alce OSD; 7)Buzzer; 8)LED switch; 9)GPS; 10)MPPT charge controller; 11)RC receiver; 12)Telemetry 1 antenna; 13)Pixhawk; 14)Solar panel power module; 15)Telemetry 1 transceiver; 16)Motor power module; 17)ESC; 18)Solar panels; 19)UBEC

[4]The figure is a example of connection between components. It must be taken into account that for a specific UAV model the length of cables as well the components itself may change. Rodrigues, 2017 [38] shows a detailed final assembly wiring diagram for LEEUAV.

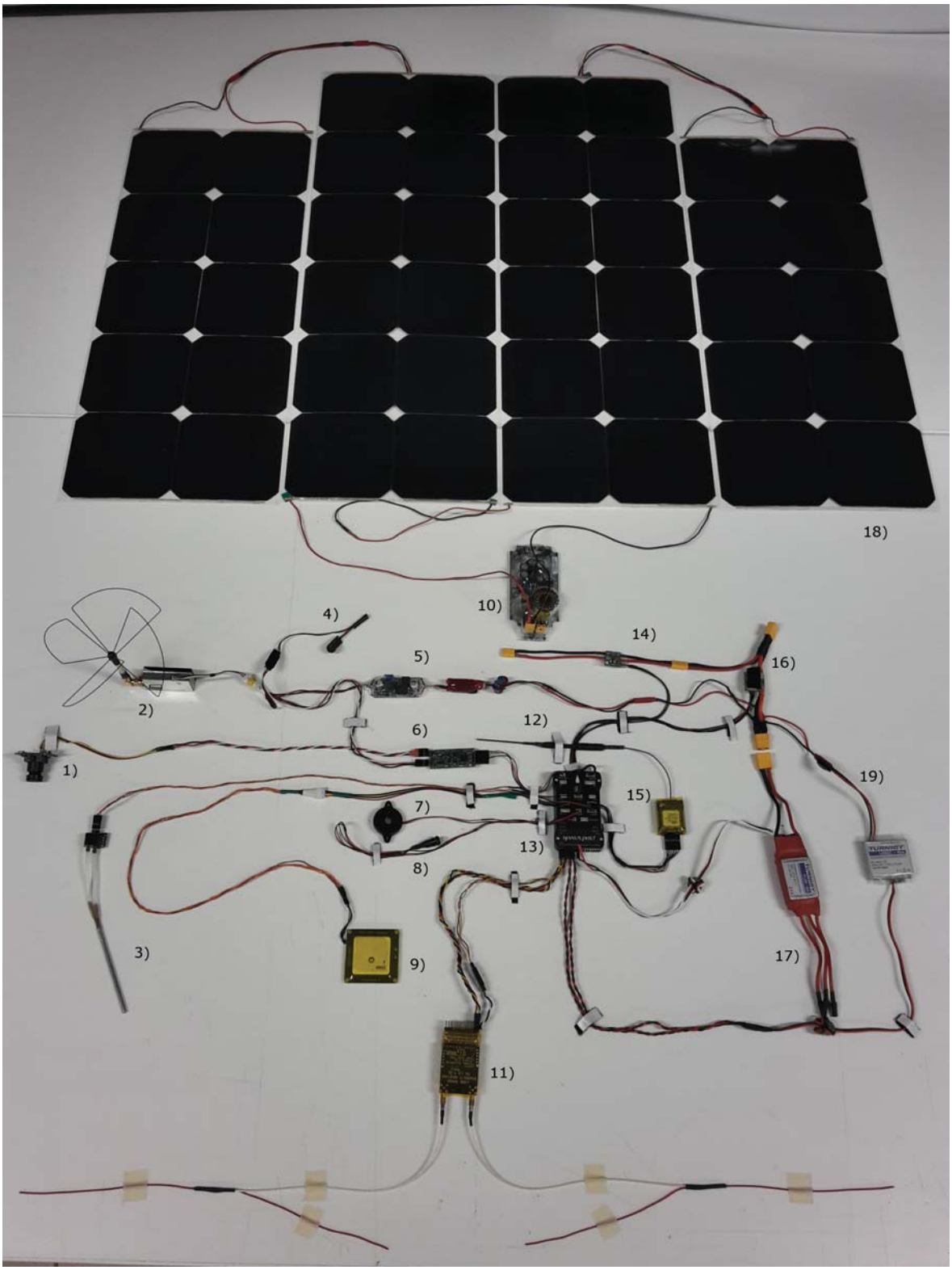



Figure 6.9: Schematic of the required hardware to perform a full autonomous flight (airborne side). Electric motor, battery and servos are not shown.

# Chapter 7

# System Tests and Results

In this chapter, the experimental validation of the proposed algorithm is shown. To accomplish this, several flight tests were carried out on two distinct UAV's[1]. For each aircraft, a set of flights with different requirements were also performed to evaluate and validate the behaviour of the system.
Throughout the validation process three phases can be distinguished:

- **Phase 0**: In an initial phase, several tests were carried out on a structure placed on the roof of a car. In this structure, all the necessary systems for an autonomous flight were considered. Thus, several bugs were detected and solved without the risk of an incident/ accident.
- **Phase 1**: At this stage, the systems were integrated into a fixed-wing aircraft called Skywalker. This aircraft was used to validate the algorithm with the parameter `solar_panels = 0`.
- **Phase 2**: Finally, a solar powered UAV named LEEUAV was used to validate the algorithm when `solar_panels = 1`.

## 7.1 Phase 0

As referred above, at an initial phase several tests were carried out on a structure placed on the roof of a car minimizing considerably the number of flights to resolve some problems detected during the system tests, Fig.7.1. To have a platform capable to simulate an autonomous aircraft system, all the necessary systems were considered, Fig.7.2.
Bellow the aspects (problems) that have been modified (resolved) in the course of the aforementioned tests are listed:

- Inability to acquire Telemetry 1 data packets in real time due to incompatible XML files between GCS and UAV;
- Excessive fluctuation of both kinetic and potential energy due erratic measurements from the barometer and GPS sensors. This situation was resolved through the use of a simple moving arithmetic mean of 5 samples;
- Excessive fluctuation of the remaining flight time estimation value. To overcome this, four different remaining flight time values were considered: Energy Flow, Global RTime, RTime and AvgRTime (see Subsection 5.2.1);
- Errors associated with the transition between discharging/charging states and null power consumption. This situation was solved by making a compass off time when the system changes from a discharging state to a charging state and by defining a maximum mission

[1] (See Appendix D)

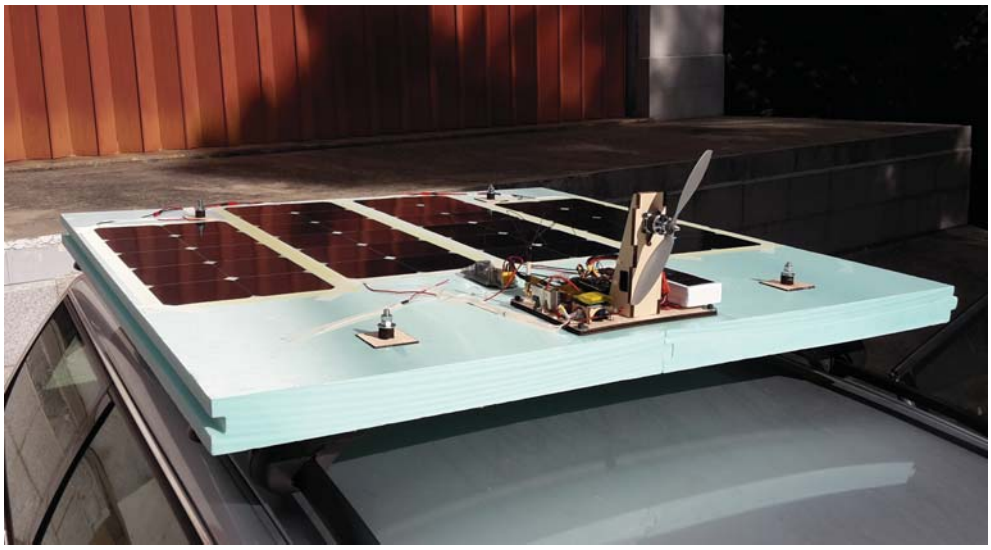
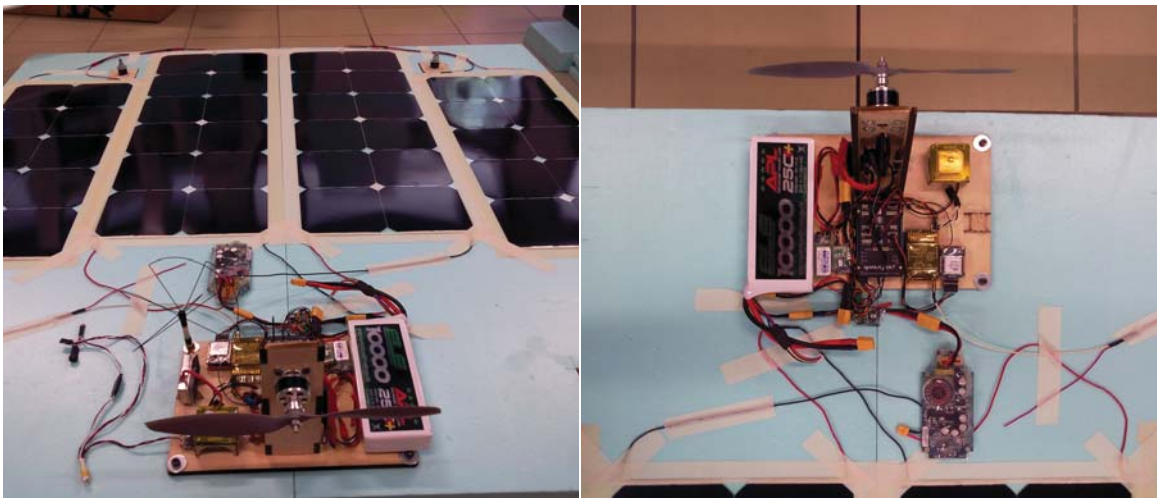


Figure 7.1: Structure placed on the roof of a car to test several mission scenarios at an early stage of the current work.



(a) Overview of the structure placed on the roof of the car.

(b) Detailed view of the hardware needed to simulate an autonomous flight.

Figure 7.2: Platform used to simulate an autonomous aircraft system.

time remaining (`mtr_max`) when the system is in a null (or almost null) power consumption state, respectively;

- Unrealistic remaining flight time estimation moments before takeoff. Since, the power consumption is very low (when the aircraft is stopped on the ground) the estimate tends to infinity. This situation was resolved by initializing the time prediction when the parameter is_flying = true is detected.

## 7.2 Phase 1

At this stage, the systems were integrated into a fixed-wing aircraft called Skywalker, Appendix D. This aircraft was used to validate the algorithm when it runs with the parameter `solar_panels = 0`.

A typical mission was predefined consisting of the following flight phases: takeoff, climb, cruise, successive loiter stages with constant airspeed, final descend and landing. The mission is illustrated in the following mission profile, Fig.7.3, where each letter corresponds to a specific mission command from Table 7.1.

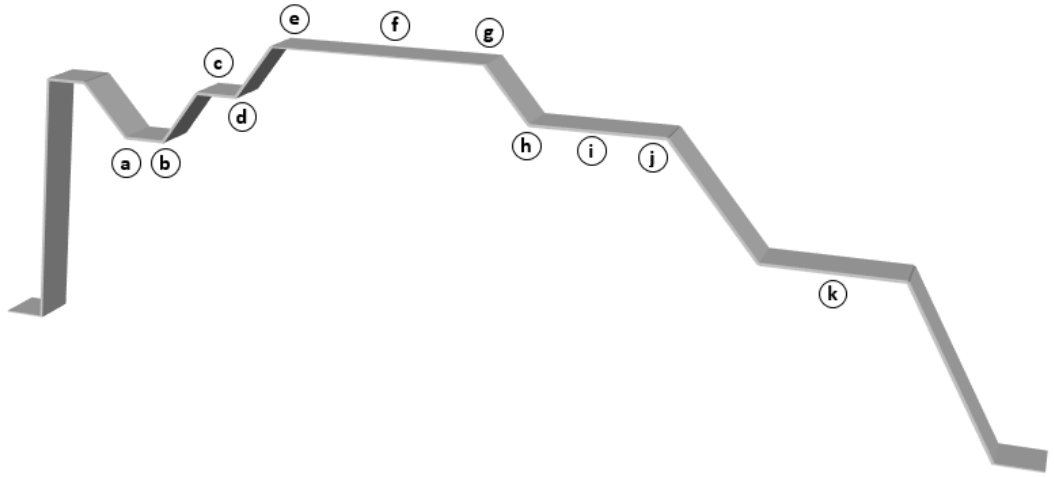



Figure 7.3: Mission profile used to proceed the system validation, Phase 1.

Table 7.1: Pixhawk commands to be executed in the course of the mission, Phase 1.

| Mission | Command | Time [s] | Direction | Heading | Latitude | Longitude | Altitude [m] |
|---|---|---|---|---|---|---|---|
| a | Waypoint | - | - | - | 40,294503 | -7,44283 | 200 |
| b | Waypoint | - | - | - | 40,293766 | -7,44283 | 200 |
| c | Waypoint | - | - | - | 40,293815 | -7,431092 | 250 |
| d | Waypoint | - | - | - | 40,294175 | -7,431092 | 250 |
| e | Loiter_Time | 1200 | 1 | 0 | 40,294143 | -7,437015 | 300 |
| f | Waypoint | - | - | - | 40,294044 | -7,443173 | 300 |
| g | Waypoint | - | - | - | 40,294683 | -7,44313 | 300 |
| h | Loiter_Time | 900 | 1 | 0 | 40,294748 | -7,437015 | 250 |
| i | Waypoint | - | - | - | 40,294617 | -7,431135 | 250 |
| j | Waypoint | - | - | - | 40,294323 | -7,431114 | 250 |
| k | Loiter_Unlimited | - | 1 | 0 | 40,295223 | -7,437165 | 150 |

Regarding Fig.7.4, some important aspects that were subject of analysis are listed bellow:

- By convention the energy transfer from the system (UAV) to its surroundings has a positive signal while the energy received has a negative signal;
- From the flight time instant t=25min and t=41min it can be verified that the responsiveness of the algorithm is satisfactory since there is no divergence of results. On the other hand, the user can notice a slightly increase of remaining flight time (`time_rem` and `time_rem_smooth`) for a short period of time due to the reduction of `total_power`;
- The `time_rem` damping is performed through `time_rem_avg`. With this damping there is a gradual increase/decrease of the estimated time so that the experience while flying an fixed-wing UAV will be more user friendly;
- Fluctuation in `time_rem` is mostly due to the periodic variation of wind heading;
- The mission was considered aborted when the “Energy Flow” message changed from “DISCHARGING” to “TOO LOW”, see Section 5.2.

- In the approaching phase for landing `time_rem` increases its value since although in a very low energy state (empty battery) the power consumed was also low (to power the flight systems only). This aspect is taken into account in `time_rem_avg`.

## 7.3 Phase 2

At this stage, the systems were integrated into a fixed-wing solar power aircraft called LEEUAV, Appendix D. This aircraft was used to validate the algorithm when it runs with the parameter `solar_panels = 1` (solar panels installed).
A basic mission was predefined consisting of the following flight phases: takeoff, climb, successive changes of airspeed between loiters at a constant altitude, final descend and landing. Each change of airspeed command (among other commands) is presented in Table 7.2.

Table 7.2: Pixhawk commands to be executed in the course of the mission, Phase 2.

| Mission | Command | Turns | Speed | Heading | Latitude | Longitude | Altitude |
|---|---|---|---|---|---|---|---|
| a | Do_Change_Speed | - | 7 | - | - | - | - |
| a | Loiter_Turns | 2 | - | 0 | 40,290992 | -7,440062 | 350 |
| b | Do_Change_Speed | - | 8 | - | - | - | - |
| b | Loiter_Turns | 2 | - | 0 | 40,291147 | -7,439869 | 350 |
| c | Do_Change_Speed | - | 9 | - | - | - | - |
| c | Loiter_Turns | 2 | - | 0 | 40,290959 | -7,439793 | 350 |
| d | Do_Change_Speed | - | 10 | - | - | - | - |
| d | Loiter_Turns | 2 | - | 0 | 40,291107 | -7,439632 | 350 |
| e | Do_Change_Speed | - | 11 | - | - | - | - |
| e | Loiter_Turns | 2 | - | 0 | 40,290959 | -7,439568 | 350 |
| f | Do_Change_Speed | - | 12 | - | - | - | - |
| f | Loiter_Turns | 2 | - | 0 | 40,29109 | -7,439396 | 350 |
| g | Do_Change_Speed | - | 13 | - | - | - | - |
| g | Loiter_Turns | 2 | - | 0 | 40,290935 | -7,439343 | 350 |
| h | Do_Change_Speed | - | 14 | - | - | - | - |
| h | Loiter_Turns | 2 | - | 0 | 40,29107 | -7,439171 | 350 |
| i | Do_Change_Speed | - | 15 | - | - | - | - |
| i | Loiter_Turns | 2 | - | 0 | 40,29091 | -7,439144 | 350 |
| j | Do_Change_Speed | - | 7.5 | - | - | - | - |
| j | Loiter_Turns | 2 | - | 0 | 40,29107 | -7,438946 | 350 |
| k | Do_Change_Speed | - | 8.5 | - | - | - | - |
| k | Loiter_Turns | 2 | - | 0 | 40,290931 | -7,438892 | 350 |
| l | Do_Change_Speed | - | 9.5 | - | - | - | - |
| l | Loiter_Turns | 2 | - | 0 | 40,291086 | -7,438731 | 350 |
| m | Do_Change_Speed | - | 10.5 | - | - | - | - |
| m | Loiter_Turns | 2 | - | 0 | 40,290943 | -7,43864 | 350 |
| n | Do_Change_Speed | - | 11.5 | - | - | - | - |
| n | Loiter_Turns | 2 | - | 0 | 40,291102 | -7,438533 | 350 |
| o | Do_Change_Speed | - | 12.5 | - | - | - | - |
| o | Loiter_Turns | 2 | - | 0 | 40,290959 | -7,438425 | 350 |
| p | Do_Change_Speed | - | 13.5 | - | - | - | - |
| p | Loiter_Turns | 2 | - | 0 | 40,291127 | -7,438324 | 350 |
| q | Do_Change_Speed | - | 14.5 | - | - | - | - |
| q | Loiter_Turns | 2 | - | 0 | 40,290992 | -7,438211 | 350 |

Regarding Fig.7.5, bellow are listed some important aspects that were subject of analysis:

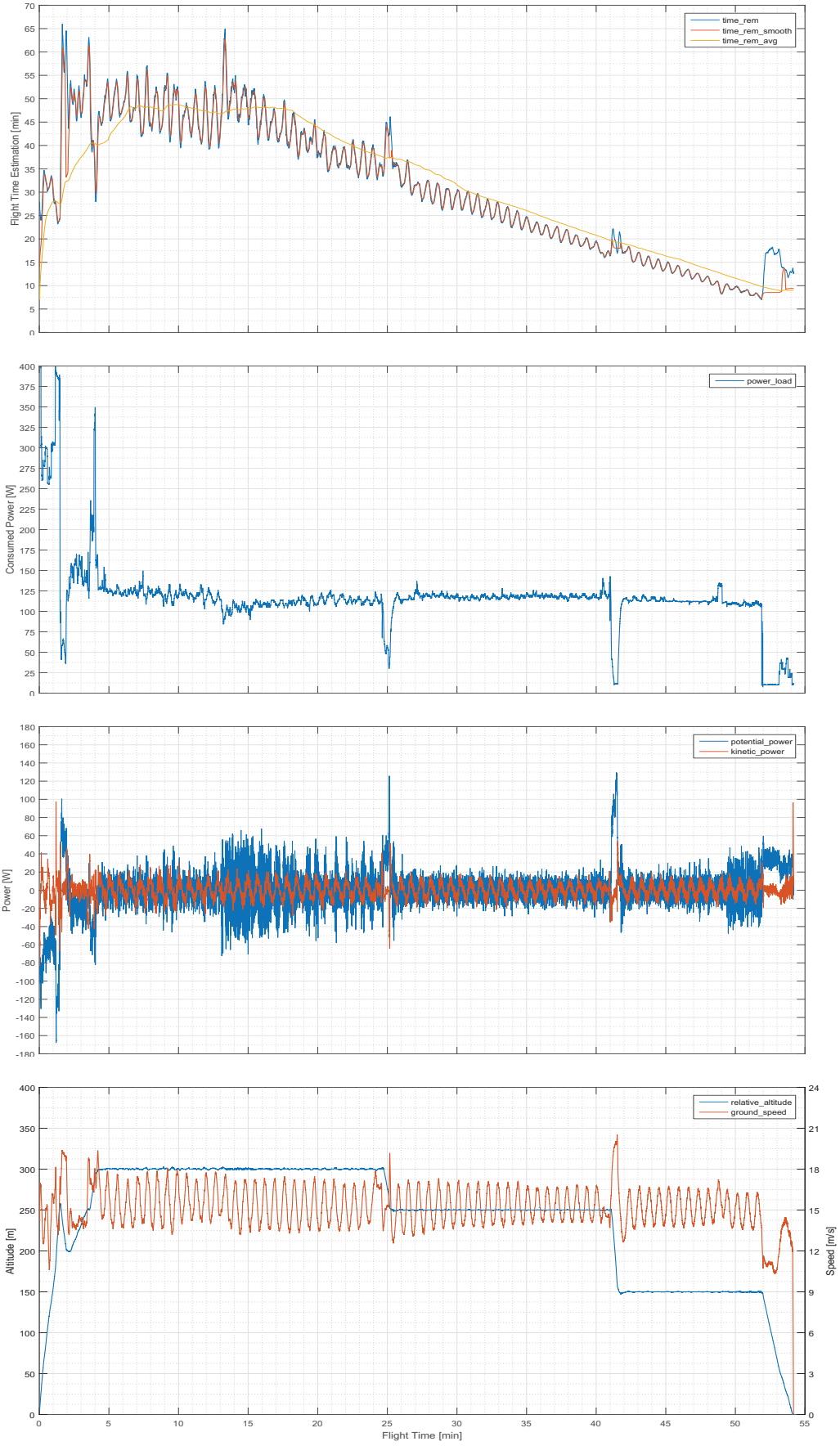



Figure 7.4: Pixhawk dataflash log results, Phase 1

- Due to the low sun light during the test, the maximum solar power achieved is approximately 40% lower compared to a day with good sun light[2];
- The power consumption peaks are due to throttle increase to change to a higher airspeed. This situation can be solved through throttle damping;
- Again, the wind component induces `ground_speed` oscillations that are directly related with the fluctuation of `time_rem`;
- Compared with Phase 1 there is one more variable (`solar_power`) in the calculation of `time_rem` increasing the system noise;
- Through several tests it was verified that the power modules reading error is too high. In very specific cases an error of 25% was registered. Through the battery discharge curves from Fig.3.3 and based on the OCV method (see Section 2.2) a recalibration every 5 minutes of remaining flight time was performed. This way, the error associated with the power modules that was being integrated over the full mission was replaced by a reading error every 5 minutes reducing his propagation dramatically. This change takes a very important role for aircraft that have low power requirements and fly for long periods of time (more than a day) since an error of 25% represents a very conservative estimation (the system has more energy than the announced at the ground control station) or in the worst case an optimistic estimation (the system has less energy than the announced at the ground control station) and an emergency landing has to be performed;
- The convention of "energy transferred from the system (UAV) to its surroundings has a positive signal while the energy received has a negative signal" remains. To facilitate data analysis the signal of `solar_power` was changed;
- Contrary to the initially expected `time_rem_global` is not useful from the user's point of view since in certain circumstances it can induce an erratic sensation of excessive remaining energy;
- During the final phase of descent for landing there was a lot of turbulence that is reflected in the potential an kinetic power;
- Contrary to Phase 1, the variable `time_rem_avg` is only shown after 5 minutes of flight time. In other words the time required until the damping of `time_rem` is reliable/stable;
- The mission was also considered aborted when the "Energy Flow" message changed from "DISCHARGING" to "TOO LOW", see Section 5.2;
- Between the flight time instants t=32min and t=48min it can be verified the influence of `solar_power` in `time_rem` since a small increase of `solar_power` leads to a considerable increase of the expected flight time;
- Under normal light conditions and taking into account the results obtained, it can be verified that the aircraft in question has capacity to fly much longer. In Rodrigues, 2017 [38] a theoretical estimation of flight endurance for a typical mission profile[3], considering a 111Wh battery was accomplished. Starting the mission at 6h with a full charged battery, the plane can fly until 19h45, corresponding to a final endurance of 13h45.

[2]The absolute maximum solar power achieved with this setup was $100W$ while during the present test was only $62.5W$.

[3]Mission takes place at the equinox for Castelo Branco, Portugal.

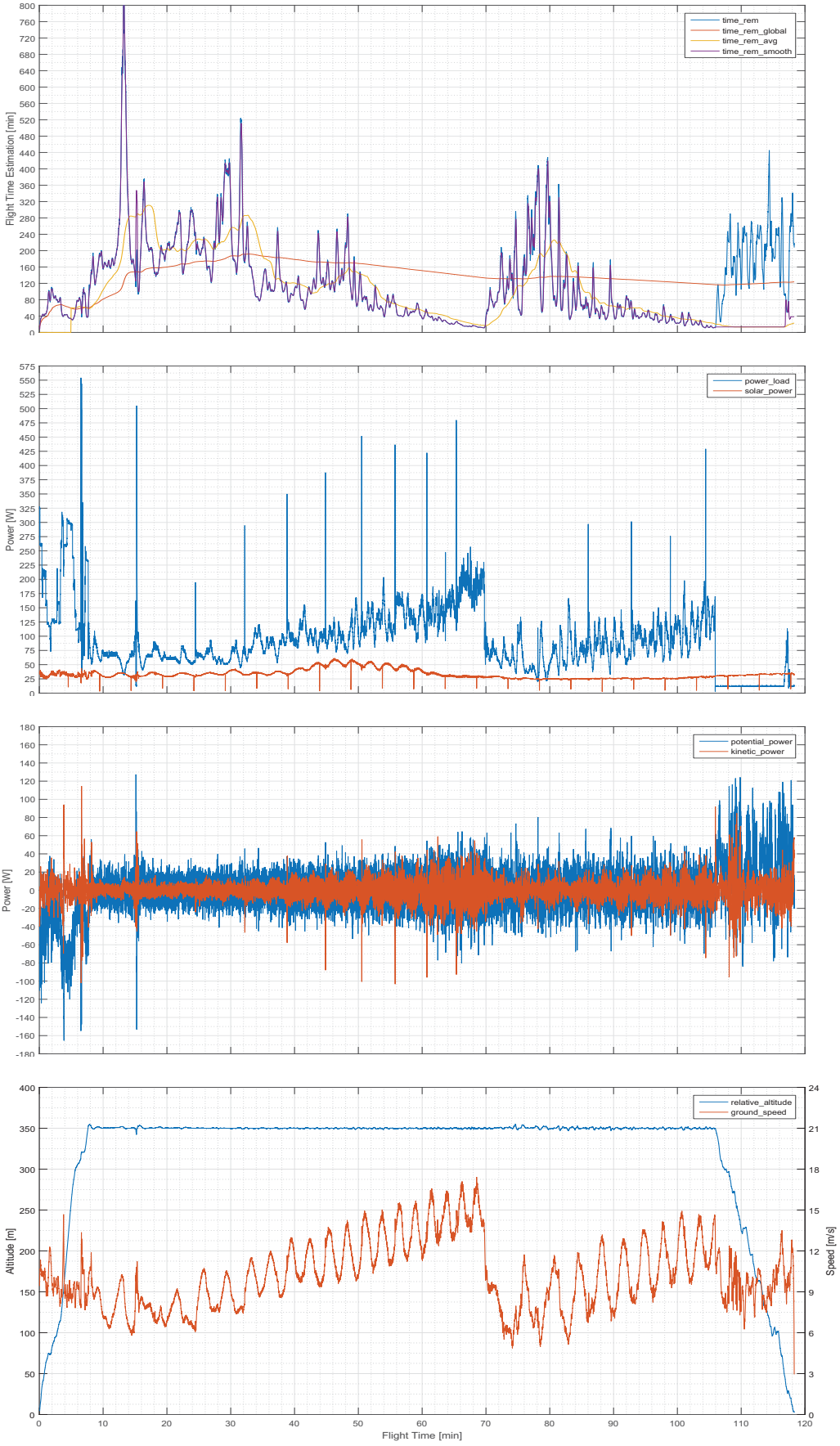



Figure 7.5: Pixhawk dataflash log results, Phase 2

# Chapter 8

# Conclusions

## 8.1 Achievements

An algorithm that allows a total energy balance estimation in real time for an UAV has been developed and experimental validated through flight tests. A set of missions with different requirements were also performed for two distinct aircraft to evaluate and validate the behaviour of the system (aircraft/autopilot). To achieve the main objective outlined initially some other tasks were also performed such as the identification of all energy sources associated with the aircraft, the development of the flight energy computation algorithm required to estimate in real time the total/partial energy (as well the rates of change), the identification of the parameters that feed the mathematical models, the expansion of the ArduPilot source code, the information forwarding via telemetry, the implementation of both reception and data visualization at the GCS, the selection and integration of the required sensors and the experimental validation of the system, through flight tests.

In general, the algorithm presents good results in terms of precision, response in transient situations and damping capacity due to sensors noise. This way, the user/pilot is capable to make a better energy management depending on the type of mission to be performed. Hence, for each combination of power consumption state[1], altitude and ground speed it is possible to understand their relation with the estimated remaining flight time.

All the relevant information associated with the algorithm is recorded in the internal memory of the autopilot and can be analysed at the end of the flight. In addition, a graphical module was created at the ground station so the user can review the last 10 minutes of the mission. This data was also overlaid into the video link (Telemetry 2) since it will theoretically fail after the Telemetry 1 link.

Although the main objective has been achieved some aspects need to be carefully analysed. The poor accuracy of the power modules is one example of this. In fact, it is expected that most of the estimated remaining flight time error is due to the low resolution of the power modules adding some uncertainty to the results.

## 8.2 Future Work

As discussed throughout the current thesis, some improvements should be performed. These improvements focus mainly on the ground control station interface, algorithm optimization and sensors selection. Below, these aspects are analysed in detail:

- Perform a careful adjustment of the autopilot gains so the transition between flight stages (speed increment) is smoother and induces less noise into the algorithm;

[1] If a solar panel is installed the power consumption is the subtraction between `power_load` and `solar_power`.

- Implement sound warnings when `time_rem` is less then five minutes. At this moment the message “TOO LOW” is also shown at the ground control station. This way, if for some reason the pilot can not look at the screen, he knows that must start the landing manoeuvres when the buzzer sounds;
- Expansion of the algorithm for automatic mission planning and optimization depending on the mission profile. Based on the energy state the mission could be optimized for range, endurance or other objective;
- Acquisition of voltage and current sensors (power module) with better resolution such as MAUCH power modules [47], in order to reduce the uncertainty of results;
- Update of the algorithm to the latest stable autopilot version and perform a pull request to get the changes merged into the GitHub “master” project;
- Implement Baião (2017) ([1]) algorithm in the autopilot and perform several flight tests to evaluate and validate the behaviour of the system as done in the present work. After this, the algorithm should also be merged into the GitHub “master” project;
- Combine both Flight Energy Management (FEM), Online Mission Planning (OMP) and Sense and Avoid (S&A) results into a single system capable to interact with these three topics at the same time.

# Bibliography


[1] T. Baião, "Energy Monitoring System for Low-Cost UAVs," MSc thesis, Instituto Superior Técnico, Lisboa, May 2017. 2, 13, 14, 15, 62

[2] H. T. Ortega, G. G. Torales, R. E. Marmolejo, and J. L. Flores, "Design and Implementation of a Power Management Module for a MUAV," in *Reliability of Photovoltaic Cells, Modules, Components, and Systems IV*, vol. 8112. SPIE Proceedings, 2011. 2

[3] J. R. Merkt, "Flight Energy Management Training: Promoting Safety and Efficiency," *Journal of Aviation Technology and Engineering*, vol. 3, no. 1, pp. 24–36, 2013. 5

[4] M. Amelink, M. Mulder, M. Rene, and J. Flach, "Theoretical Foundations for a Total Energy-Based Perspective Flight-Path Display," *The International Journal of Aviation Psychology*, vol. 15, no. 3, pp. 205–231, 2005. 5

[5] E. Meissner and G. Richter, "Battery Monitoring and Electrical Energy Management Precondition for Future Vehicle Electric Power Systems," *Journal of Power Sources*, vol. 116, pp. 79–98, 2003. 6

[6] S. Piller, M. Perrin, and A. Jossen, "Methods for state-of-charge determination and their applications," *Journal of Power Sources*, vol. 96, pp. 113––120, 2001. 6, 7, 8, 10

[7] D. Sauer, G. Bopp, A. Jossen, J. Garche, M. Rothert, and M. Wollny, "State of charge - What do we really speak about ?" in *INTELEC´99*, 1999. 6

[8] V. Pop, "Universal State-of-Charge Indication for Portable Applications," PhD thesis, Universiteit Twente, Eindhoven, Feb. 2007. 6

[9] J. Chiasson and B. Vairamohan, "Estimating the State of Charge of a Battery," in *Proceedings of the American Control Conference*, 2003. 6

[10] "Telemetry: Summary of concept and rationale," NASA Scientific and Technical Information (STI), Tech. Rep., Dec. 1987. 6, 37

[11] Ground Control Stations for Unmanned Aerial Vehicles (UAV's) are Becoming Networking-Hub Cockpits on the Ground for U.S. Unmanned Forces. Military&Aerospace. Accessed July,2017. [Online]. Available: http://www.militaryaerospace.com/articles/2010/06/ground-control-stations.html 6

[12] J. Wang, B. Cao, Q. Chen, and F. Wang, "Combined state of charge estimator for electric vehicle battery pack," *Control Engineering Practice*, vol. 15, pp. 1569–1576, 2007. 8, 11, 12, 13

[13] P. Oettershagen, A. Melzer, T. Mantel, K. Rudin, T. Stastny, B. Wawrzacz, T. Hinzmann, K. Alexis, and R. Siegwart, "Perpetual flight with a small solar-powered UAV: Flight results, performance analysis and model validation," in *IEEE Aerospace Conference*. ETH Eidgenössische Technishe Hochschule-Zürich, 2016. 8

[14] AtlantinkSolar: The first-ever crossing of the Atlantic Ocean using a solar-powered Unmanned Aerial Vehicle (UAV). ETH Eidgenössische Technishe Hochschule-Zürich. Accessed January,2017. [Online]. Available: http://atlantiksolar.ethz.ch/wp-content/downloads/AtlantikSolar_ProjectBrochure_websiteversion.pdf 8

[15] L. Meier, D. Honegger, and M. Pollefeys, “PX4: A Node-Based Multithreaded Open Source Robotics Framework for Deeply Embedded Platforms,” in *IEEE International Conference on Robotics and Automation*. ETH Eidgenössische Technishe Hochschule-Zürich, 2015. 8

[16] V. Pop, H. Bergveld, P. Notten, J. Op het Veld, and P. Regtien, “Accuracy analysis of the State-of-Charge and remaining run-time determination for lithium-ion batteries,” *Measurement*, vol. 42, pp. 1131–1138, 2009. 8, 9, 10

[17] H. Bergveld, V. Pop, and P. Notten, “Method of estimating the State-of-Charge and of the use time left of a rechargeable battery, and apparatus for executing such a method,” U.S. Patent 20 080 150 491 A1, Feb. 23, 2005. 11

[18] V. Pop, H. Bergveld, J. Op het Veld, P. Regtien, and D. Danilov, “Modelling battery behaviour for accurate State-of-Charge indication,” *Journal of the Electrochemical Society*, vol. 153, no. 11, pp. A2013–A2022, 2006. 11

[19] *Battery & Cell Test Equipment Worldwide*, Maccor Inc., 2016. [Online]. Available: http://www.maccor.com/ 11

[20] AV-900 Heavy Duty Dual Channel Cycling Station. AeroVironment™. Accessed January,2017. [Online]. Available: http://www.avtestsystems.com/Source/PDF/AV-900(CE)/AV900_FINAL_112509_AeroVironment.pdf 11

[21] P. Gamboa and M. Silvestre, “Introdução ao Avião e Disciplinas de Projecto,” presented in Design Aeronáutico Computacional, Univ. da Beira Interior, Covilhã, 2012. 20

[22] J. Sousa, “Solar System for a Long Endurance Electric UAV,” MSc. thesis, Universidade da Beira Interior, Covilhã, Oct. 2015. 24, 46

[23] D. Verma, S. Nema, A. Shandilya, and S. Dash, “Maximum power point tracking (MPPT) techniques: Recapitulation in solar photovoltaic systems,” *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1018–1034, 2016. 25, 26

[24] Buzzer Sounds Boot-(Pixhawk/PX4). ArduPilot. Accessed February,2017. [Online]. Available: http://ardupilot.org/copter/docs/common-sounds-pixhawkpx4.html 29

[25] Color Patterns Boot-(Pixhawk/PX4). ArduPilot. Accessed February,2017. [Online]. Available: https://pixhawk.org/users/status_leds 29

[26] Arming Plane-(Pixhawk/PX4). ArduPilot. Accessed March,2017. [Online]. Available: http://ardupilot.org/plane/docs/arming-your-plane.html 29

[27] Welcome to the ArduPilot Development Site. ArduPilot. Accessed October,2016. [Online]. Available: http://ardupilot.org/dev/ 37

[28] History of Ardupilot. ArduPilot. Accessed November,2016. [Online]. Available: http://ardupilot.org/dev/docs/common-history-of-ardupilot.html 37

[29] Flight Modes. ArduPilot. Accessed November,2016. [Online]. Available: http://ardupilot.org/plane/docs/flight-modes.html 37

[30] Flight Features. ArduPilot. Accessed November,2016. [Online]. Available: http://ardupilot.org/plane/docs/flight-features.html 37

[31] Code Overview (Copter). ArduPilot. Accessed January,2017. [Online]. Available: http://ardupilot.org/dev/docs/apmcopter-code-overview.html 38

[32] A. Tridgell. MAVLink proxy and command line ground station. ArduPilot. Accessed January,2017. [Online]. Available: https://github.com/tridge/MAVProxy 39

[33] A UAV ground station software package for MAVLink based systems. ArduPilot. Accessed January,2017. [Online]. Available: http://ardupilot.github.io/MAVProxy/html/index.html 40

[34] L. Alves. AlceOSD. AlceOSD. Accessed April,2017. [Online]. Available: https://github.com/ardupilot/alceosd 43

[35] *Pixhawk Kit User Guide V8*, 3D Robotics, 2014. 45

[36] P. Barker, H. Willee, and R. Mackay. Pixhawk Overview. ArduPilot. Accessed October,2016. [Online]. Available: https://github.com/ArduPilot/ardupilot_wiki/blob/master/common/source/docs/common-pixhawk-overview.rst 45

[37] Pixhawk Autopilot. Px4 autopilot. Accessed May,2017. [Online]. Available: https://pixhawk.org/modules/pixhawk 45

[38] A. Rodrigues, "Airframe Assembly, Systems Integration and Flight Testing of a Long Endurance Electric UAV," MSc thesis, Universidade da Beira Interior, Covilhã, Feb. 2017. 47, 51, 58

[39] M. Oborne. Mission Planner. ArduPilot. Accessed November,2016. [Online]. Available: https://github.com/ArduPilot/ardupilot_wiki/tree/master/planner 47

[40] LogView Software. LogView Studio. Accessed November,2016. [Online]. Available: http://logview.info 47

[41] iCharger 4010 Duo. iCharger NZ. Accessed November,2016. [Online]. Available: http://www.icharger.co.nz/Products/4010-Duo.aspx 47

[42] GZV6000 Switching Mode DC Power Supply. Diamond Antenna. Accessed October,2016. [Online]. Available: http://www.diamondantenna.net/gzv6000.html 47

[43] Vici VC97 Multimeter. Shenzhen Vicimeter Technology. Accessed October,2016. [Online]. Available: http://www.vicimeter.com/en/productshow.asp?sendid=14 47

[44] SLS APL LiPo Battery. SLS - StefansLipoShop GmbH. Accessed October,2016. [Online]. Available: https://www.stefansliposhop.de/Akkus/SLS-Multirotor/hohe-Kapazitaeten/SLS-APL-10000mAh-3S1P-11-1V-25C-40C::1095.html 48

[45] *MS5611 Barometer Module*, MEAS® Switzerland, 2016. [Online]. Available: https://infusionsystems.com/support/MS5611-01BA03.pdf 49

[46] D. Coleman and D. Westcott, *CWNA: Certified Wireless Network Administrator Official Study Guide: Exam CWNA-106*, ser. ISBN: 978-1-118-89370-8. SYBEX, 2014. 50

[47] Mauch Power Module. Mauch Electronic. Accessed June,2017. [Online]. Available: http://ardupilot.org/copter/docs/common-mauch-power-modules.html 62

# Appendices

# Appendix A

# MAVLink Protocol

## A.1 MAVLink Parameters

```
// @Param: ENER_UAV_MASS
// @DisplayName: Energy aircraft mass
// @Description: Set the aircraft mass
// @Units: kilogram
// @User: Advanced
GSCALAR(aircraft_mass,   "ENER_UAV_MASS",     ENER_UAV_MASS_DEFAULT),

// @Param: ENER_BATT_CAPAC
// @DisplayName: Energy battery capacity
// @Description: Set the real flight battery capacity (12.6v -> 10.5V, 4.2 ->
    3.5V/cell)
// @Units: mah
// @User: Advanced
GSCALAR(battery_cap,     "ENER_BATT_CAPAC",     ENER_BATT_CAPAC_DEFAULT),

// @Param: ENER_BATT_RESSIS
// @DisplayName: Energy battery int ressist
// @Description: Set the Battery internal ressitance
// @Units: ohm
// @User: Advanced
GSCALAR(battery_int_ressist,   "ENER_BATT_RESSIS",   ENER_BATT_RESSIS_DEFAULT),

// @Param: ENER_SOLAR_PANEL
// @DisplayName: Energy solar panels
// @Description: Set if solar panels are being used
// @Values: 0:Disabled,1:Enabled
// @User: Advanced
GSCALAR(solar_panels,    "ENER_SOLAR_PANEL",    ENER_SOLAR_PANEL_DEFAULT),

// @Param: ENER_BATT_NUMBER
// @DisplayName: Battery number
// @Description: Set the battery number
// @Range: 1 360
// @User: Advanced
GSCALAR(battery_number,    "ENER_BATT_NUMBER",    ENER_BATT_NUMBER_DEFAULT),

// @Param: ENER_BATT_UPDT
// @DisplayName: Battery energy update (interval)
// @Description: Set the interval in which the battery energy update should be done
 // @Units: second
```

```
    // @Range: 1 500
    // @User: Advanced
    GSCALAR(benergy_update,     "ENER_BATT_UPDT",    ENER_BATT_UPDT_DEFAULT),
```

## A.2 MAVLink Messages

```
void Plane::send_energy(mavlink_channel_t chan)
{
    mavlink_msg_energy_send(
    chan,
    millis(),
    power_load,
    solar_power * -1.0f,
    var_mtr,
    mtr,
    time_rem,
    time_rem_global,
      var_charg,
    charg_trem,
    chg_trem);
}

void Plane::send_energy1(mavlink_channel_t chan)
{
    mavlink_msg_energy1_send(
    chan,
    millis(),
    penergy,
    delta_pot,
    kenergy,
    delta_kin,
    benergy,
    energy_consumed,
    energy_received,
    total_energy,
    total_power,
    flight_t,
    flight_timer);
}

void Plane::send_energy2(mavlink_channel_t chan)
{
    mavlink_msg_energy2_send(
    chan,
    millis(),
    time_rem_smooth,
    time_rem_avg,
    var_avg,
```

```
    var_smooth,
    telem_pl,
    telem_sp);
}
```

## A.3 MAVLink Messages (XML File)

```
<message id="227" name="ENERGY">
    <description>Energy Output</description>
    <field name="time_boot_ms" type="uint32_t">Timestamp (milliseconds since system
        boot)</field>
    <field name="power_load" type="float">Flight power</field>
    <field name="solar_power" type="float">Solar power</field>
    <field name="var_mtr" type="uint8_t">Discharge state</field>
    <field name="mtr" type="float">Remaining flight time</field>
    <field name="time_rem" type="float">Remainig flight time (average)</field>
    <field name="time_rem_global" type="float">Remainig flight time (average of global
        average)</field>
    <field name="var_charg" type="uint8_t">Charge state</field>
    <field name="charg_trem" type="float">Remaining charging time</field>
    <field name="chg_trem" type="float">Remainig charging time (average)</field>
</message>

<message id="228" name="ENERGY1">
     <description>Energy Output1</description>
     <field name="time_boot_ms" type="uint32_t">Timestamp (milliseconds since system
         boot)</field>
     <field name="penergy" type="float">Potential energy</field>
     <field name="delta_pot" type="float">Delta potential energy</field>
     <field name="kenergy" type="float">Kinetic energy</field>
     <field name="delta_kin" type="float">Delta kinetic energy</field>
     <field name="benergy" type="float">Battery energy</field>
     <field name="energy_consumed" type="float">Energy Consumed since start up</field>
     <field name="energy_received" type="float">Energy received since start up</field>
     <field name="total_energy" type="float">Total remaining energy since start up</field>
     <field name="total_power" type="float">Total power</field>
     <field name="flight_t" type="float">Flight time since is_flying is true</field>
     <field name="flight_timer" type="float">Counter</field>
</message>

<message id="229" name="ENERGY2">
    <description>Energy Output2</description>
    <field name="time_boot_ms" type="uint32_t">Timestamp (milliseconds since system
        boot)</field>
    <field name="time_rem_smooth" type="float">Remainig flight time (smooth)</field>
    <field name="time_rem_avg" type="float">Remainig flight time (5min average)</field>
    <field name="var_avg" type="uint8_t">Discharge state (avg)</field>
    <field name="var_smooth" type="uint8_t">Discharge state (smooth)</field>
```

```
    <field name="telem_pl" type="float">Power load (OSD)</field>
    <field name="telem_sp" type="float">Solar power (OSD)</field>
</message>
```

# Appendix B

## Battery Energy Correction

```
//Battery energy calculation. To avoid that the error of the power module readings is
    propagated in time, is done a correction every 5 minutes
benergy_counter += 1;
if (benergy_counter < g.benergy_update){
   if (var_benergy == 0){
      //Algorithm 1
      //First term of mission_time_remaining estimation
      if (battery.has_current() && battery.healthy() && g.solar_panels == 1){
         benergy = (g.battery_cap * 0.001f * 12.6f * 3600.0f) - energy_consumed +
             energy_received;
      } else if (battery.has_current() && battery.healthy() && g.solar_panels == 0){
         benergy = (g.battery_cap * 0.001f * 12.6f * 3600.0f) - energy_consumed;
      } else {
         benergy = 0.0f;
      }
   } else {
      //Algorithm 1
      //First term of mission_time_remaining estimation
      if (battery.has_current() && battery.healthy() && g.solar_panels == 1){
         benergy = benergy_init - energy_consumed + energy_received;
      } else if (battery.has_current() && battery.healthy() && g.solar_panels == 0){
         benergy = benergy_init - energy_consumed;
      } else {
         benergy = 0.0f;
      }
   }
} else {
   energy_consumed = 0.0f; //Reset of energy consumed to mitigate the error
   energy_received = 0.0f; //Reset of energy received to mitigate the error
   benergy_counter = 0;
   var_benergy = 1;
    if (g.battery_number ==3){
      if (battery.voltage() >= 11.65f){
         consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
             12.33966f) / (-0.00027f);
      } else if (battery.voltage() >= 10.85f && battery.voltage() < 11.65f){
         consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
             11.89127f) / (-0.00011f);
      } else {
         consumed_capacity = ((g.battery_int_ressist * curr_dscg + battery.voltage()) -
             18.84327f) / (-0.00084f);
      }
   }
```

```
    real_capacity = g.battery_cap - consumed_capacity;

    //If is case of that set real_capacity = g.battery_cap.
    if (real_capacity > g.battery_cap){
        real_capacity = g.battery_cap;
    }

    if (battery.has_current() && battery.healthy() && g.solar_panels == 1){
        benergy_init = (real_capacity * 0.001f * (g.battery_int_ressist *
            (battery.current_amps() - battery.current_amps(1)) + battery.voltage()) *
            3600.0f);
    } else {
        benergy_init = (real_capacity * 0.001f * (g.battery_int_ressist *
            battery.current_amps() + battery.voltage()) * 3600.0f);
    }
}

//System converged after 10 seconds
benergy_recalc = (_brecalc_filter.apply(benergy_recalc) + benergy) / 2.0f;
```

# Appendix C

# DataFlash Package

## C.1 DataFlash Messages

```
// Write an Energy data packet (same for Mavlink)
void DataFlash_Class::Log_Write_Energy(float power_load, float solar_power, uint8_t
    var_mtr, float mtr, float time_rem, float time_rem_global, uint8_t var_charg, float
    charg_trem, float chg_trem)
{
   struct log_Energy pkt = {
      LOG_PACKET_HEADER_INIT(LOG_ENERGY_MSG),
      time_us          : AP_HAL::micros64(),
      power_load       : power_load,
      solar_power      : solar_power * -1.0f,
      var_mtr          : var_mtr,
      mtr              : mtr,
      time_rem         : time_rem,
      time_rem_global  : time_rem_global,
      var_charg        : var_charg,
      charg_trem       : charg_trem,
      chg_trem         : chg_trem
   };
   WriteBlock(&pkt, sizeof(pkt));
}

// Write an additional Energy (Energy1) data packet
void DataFlash_Class::Log_Write_Energy1(float penergy, float delta_pot, float kenergy,
    float delta_kin, float benergy, float energy_consumed, float energy_received, float
    total_energy, float total_power, float flight_t, float flight_timer)
{
   struct log_Energy1 pkt = {
      LOG_PACKET_HEADER_INIT(LOG_ENERGY1_MSG),
      time_us          : AP_HAL::micros64(),
      penergy          : penergy,
      delta_pot        : delta_pot,
      kenergy          : kenergy,
      delta_kin        : delta_kin,
      benergy          : benergy,
      energy_consumed  : energy_consumed,
      energy_received  : energy_received,
      total_energy     : total_energy,
      total_power      : total_power,
      flight_t         : flight_t,
      flight_timer     : flight_timer
```

```
    };
    WriteBlock(&pkt, sizeof(pkt));
}

// Write an additional Energy (Energy2) data packet
void DataFlash_Class::Log_Write_Energy2(float time_rem_smooth, float time_rem_avg,
     uint8_t var_avg, uint8_t var_smooth)
{
    struct log_Energy2 pkt = {
        LOG_PACKET_HEADER_INIT(LOG_ENERGY2_MSG),
        time_us           : AP_HAL::micros64(),
        time_rem_smooth   : time_rem_smooth,
        time_rem_avg      : time_rem_avg,
        var_avg           : var_avg,
        var_smooth        : var_smooth
    };
    WriteBlock(&pkt, sizeof(pkt));
}
```

## C.2 Log Structure

```
   #define LOG_BASE_STRUCTURES \
    { LOG_ENERGY_MSG, sizeof(log_Energy),\
      "ENE","QffBfffBff","Time,p_l,s_p,v_mtr,mtr,t_rem,t_rem_global,v_chg,chg_trm,chg_avg"},\
    { LOG_ENERGY1_MSG, sizeof(log_Energy1),\
      "ENE1","Qfffffffffff","T,pene,d_p,kene,d_k,bene,ene_con,ene_rec,t_en,t_pw,t,i"},\
    { LOG_ENERGY2_MSG, sizeof(log_Energy2),\
      "ENE2","QffBB","Time,t_rem_smooth,t_rem_avg,v_avg,v_smooth"},\
```

# Appendix D

## Flight Test Aircraft

(a) Skywalker 2013 carbon fiber tail version. (b) LEEUAV-Long Endurance Electric UAV.

Figure D.1: Aircraft used to validate the proposed algorithm.

Table D.1: Aircraft specifications/setup.

| | **Skywalker 2013 CFT** | **LEEUAV** |
|---|---|---|
| **Wingspan** (b) | 1.88m | 4.50m |
| **Length** (l) | 1.18m | 2.52m |
| **Airframe Weight** (W_struct) | 10.93N | 14.80N |
| **Maximum Takeoff Weight** (MTOW) | up to 17.65N | up to 48.07N |
| **Motor** | T-Motor AT 2820 | Hyperion ZS 3025-10 |
| **Propeller** | APC 11x7 | Aero-naut 16x8 |
| **Battery** | LiPo 4s Hyperion 4000mah | LiPo 3s SLS 10000mah |
| **Servos** | 4 x Emax ES08AII | 4 x Corona 939MG (Ailerons+Elevator)<br>1 x Hitec HS-5125MG (Rudder) |

# Appendix E

## Hardware Specifications

### E.1 Genasun GV-10-Li-12.5V

| | GV-10-Pb-12V | | GV-10-Li-**.*V | |
|---|---|---|---|---|
| Maximum Recommended Panel Power: | 140W | | GV-10-Li-12.5V | 120W |
| | | | GV-10-Li-14.2V | 140W |
| | | | GV-10-Li-16.7V | 160W |
| Rated Battery (Output) Current: | 10.5A | | 10.5A | |
| Nominal Battery Voltage: | 12V | | N/A | |
| Max Panel Voltage (Voc): | 34V | | 34V | |
| Recommended Max Voc at STC: | 27V | | 27V | |
| Minimum Battery Voltage for Operation: | 8.5V | | 8.5V | |
| Charge Profile: | Multi-Stage with Temperature Compensation | | CC-CV | |
| Charging Voltages: | FLOODED Setting | SEALED Setting | | |
| Equalization Voltage: | 15.0V | - | - | |
| Equalization Time: | 2 Hours | - | - | |
| Equalization Interval: | 30 Days | - | - | |
| Bulk Voltage: | 14.6V | 14.3V | - | |
| Absorption Voltage: | 14.4V | 14.1V | - | |
| Absorption Time: | 2.5 Hours | | - | |
| Float Voltage (Pb models) or CV Voltage (Li models): | 13.5V | 13.7V | GV-10-Li-12.5V | 12.5V |
| | | | GV-10-Li-14.2V | 14.2V |
| | | | GV-10-Li-16.7V | 16.7V |
| Battery Temperature Compensation: | -28mV/°C | | - | |
| Operating Temperature: | -40°C - 85°C | | -40°C - 85°C | |
| Electrical Efficiency: | 96% - 98% typical | | 96% - 98% typical | |
| Tracking Efficiency: | 99+% typical | | 99+% typical | |
| MPPT Tracking Speed: | 15Hz | | 15Hz | |
| Night Consumption: | 0.9mA (900uA) | | 0.9mA (900uA) | |
| Marine Grade: | Yes | | Yes | |
| Connection: | 4-position terminal block for 10-30AWG wire | | 4-position terminal block for 10-30AWG wire | |
| Weight: | 6.5oz., 185g | | 6.5oz., 185g | |
| Dimensions: | 5.5x2.5x1.2", 14x6.5x3.1cm | | 5.5x2.5x1.2", 14x6.5x3.1cm | |
| Warranty: | 5 years | | 5 years | |

Figure E.1: Genasun GV-10-Li-12.5V specifications.