MASTER THESIS

Energy Consumption Model of Underwater Robots

Presented by
Ammar ELMELIGY

Supervisors
Dr. Didier CRESTANI - Professor at University of Montpellier
Dr. Lionel LAPIERRE - Associate Professor at University of Montpellier

A thesis submitted in fulfillment of the requirements
for the degree of Master 2
in
Mechanical Automatic Control Engineering

Jury
Frederic HEIM - Responsible of master at ENSISA
Bernard DURAND - Vice President at UFE
Magdy Maurice - Lecturer at Cairo University
Osama BADR - Head of Mechanical department at UFE
Rania AlANWAR - Lecturer at Mechanical department at UFE

July 2020

"There are no incurable diseases — only the lack of will.
There are no worthless herbs — only the lack of knowledge."

Avicenna

UHA

Abstract

Engineering School
ENSISA

Master 2

Energy Consumption Model of Underwater Robots

by Ammar ELMELIGY

Autonomous robotic systems have to do complex missions in terms of duration and task required. So, The energy-awareness becomes more crucial for mission completion because of the development of new algorithms which require new resources. Having a generic energy model for the robotic systems is the key element for management of resources allocation which gives the ability to predict the required resources for a specific task tacking into account the energy limit. Especially, for the robots exploring unknown and hazardous areas for long term missions. This thesis represents a global energy consumption model, with methodologies, experimental tests and analysis, for Autonomous Underwater Vehicle (AUV) in terms of Software and hardware consumption.

Key Words : Energy Consumption, CPU consumption, Consumption models, Energy estimation, predictive models.

Acknowledgements

I would like to express my deepest appreciation to my supervisors Prof.Didier CRESTANI and Prof.Lionel LAPIERRE for giving my the opportunity join EXPLORE team and for continuous support of my research, for their patience, motivation especially in hard days, and for their immense knowledge. their guidance helped me in all the time of research and writing of this thesis. Without they precious support it would not be possible to conduct this research. I could not have imagined having a better advisors and mentors.

I would also like to extend my deepest gratitude to EXPOLRE team members who gave me a lot of support, motivation and companionship especially, for Rodolfo Villalobos-Martinez who shared with me this research topic, and for Adrien Hereau and Jose Luis Vilchis Medina who gave a lot of technical knowledge and helped me throughout the journey.

I am also grateful to Dr. Rania AlANWAR and Prof. Osama BADR for their support and motivation throughout last years.

Many thanks to my class mates for there relentless support.

Last but not the least, I would like to thank my family: my parents and sisters for supporting me spiritually throughout my life.

Contents

List of Figures

List of Tables

List of Abbreviations

AUV	**A**utonomous **U**nderwater **V**ehicle
DoA	**D**egree **o**f **A**utonomy
ROV	**R**emotely **O**perated **V**ehicle
UGV	**U**nmanned **G**round **V**ehicle
UAV	**U**nmanned **A**erial **V**ehicle
UMV	**U**nmanned **M**arine **V**ehicle
PANORAMA	**P**erformance and **A**uto**NO**my using **R**esources **A**llocation **MA**nagement
BUBOT	**B**etter **U**nderstanding **B**iodiversity changes thanks to new **O**bservation **T**ools
SLAM	**S**imultaneous **L**ocalization**A**nd **M**apping
AMCL	**A**daptive **M**onte**C**arlo **L**ocalization
EKF	**E**xtended **K**alman **F**ilter
CMOS	**C**omplementary **M**etal**O**xide **S**emiconductor
PID	**P**rocess **ID**
CPU	**C**entral **P**rocessing**U**nit

List of Symbols

a	Distance	m
P	Power	W ($\mathrm{J\,s^{-1}}$)
E	Energy	J
t	Time	s
f	Frequency	Hz
I	Current	A
V	Voltage	V
ω	angular frequency	rad

Chapter 1

Introduction

Long missions for autonomous robots are very challenging in robotics field. The robot must have an environment awareness in addition to a self-awareness to ensure a successful mission especially if the environment was previously unknown for the objective of exploration for instance. Therefor, a successful mission should be defined with performance measurements. These performance measurements are considered as a previously defined constraints to guarantee that mission is finished according to the robot awareness. Then, developing robot's self-aware system that help to make the right decision at the right time based on defined constraints.

For example, if mission's time is considered as performance measurement, the robot will define a time constraint that marks the mission is succeeded or not. Therefor, developing robot's time-awareness is crucial for this kinds of decisional level.

In this context, energy is also considered as performance measurement axis. The objective of this research is to develop an energy estimation model which helps the robot to advance its energy-awareness level. The proposed energy models is for *HotelPower* which consists of two parts: Software and Sensors. Next, we mention some related works concerning Autonomy and Energy.

1.1 Autonomy

Robots are becoming increasingly autonomous. Yet, there are no commonly accepted terms and measures of how "autonomous" a robot is,[Wad+04].

We consider in the following that an autonomous robot is able to perform tasks without any human help and is capable to pursue its goals having an active use of its capabilities. To be autonomous the robot must have some questions to ask:

- **Decisional autonomy** : *"What to do ?"*
- **Behavioral autonomy**: to check mission success like *"How to perform what has been decided?"*

Whatever the decision, behavioral autonomy raises 3 key factors: an objective to be satisfied, a finite set of alternatives to execute this objective, and a criterion to decide which alternative must be chosen, [LLC19].

The field of autonomous vehicles shows a promising future. Since, last few years witness the great capabilities that autonomous vehicles can perform by merging multiple fields like Computer Vision, Artificial Intelligence and Control Theory. By performing the three principles components in robotic systems ,which are Perception, Control and Action, vehicles can do multiple tasks and perform complex missions with higher accuracy within lesser amount of time, giving it the ability to open new ground in many fields.

Searching for *Performance Guarantee* for autonomous robotic missions is challenging in the field of autonomy. Since a lot of field use different types of performance measurements. Looking into business domain [Nee04] performance is multi-dimensional, related to an objective, and is the result of an action, according to the involved resources. Measure (with regard to a reference), estimation (confidence range) and evaluation (interpretation) can be distinguished. Performance indicators influence the performance, performance indicators measure/estimate the performance.

In robotics, industrially, the most common ones are load capacity, workspace, speed, acceleration, repeatability, and accuracy. More specific ones can be used for manipulation or dexterity performance estimation. Some performance criteria have been standardized like (ISO/DIS 9283, 2015). However the industrial context is quite different from the autonomous vehicles robotics. Most of the time the industrial environment can be considered as static, and the available energy can be considered as infinite.

In Patrolling and Exploration robots, *Security* is a fundamental performance measurements. For instance, Karst environments are very dangerous, fragile and dynamic. Therefore the robots should have a high level of awareness about the environment and the decisions could be made in different situations.

In this context, EXPLORE team has developed a new methodology on using different performance axes and to find a Resources allocation Solution called **PANORAMA** (Performance and AutoNOmy using Resources Allocation MAnagement).

This methodology,in [Jai+16], for using performance points of view to guide hardware and software resources management according to mission execution and fault occurrence. the main performance axes are:

- **Safety**: the safety must be ensured all along the mission. It is implemented considering obstacle avoidance capacity. To avoid an obstacle the robot must be able to detect it by selecting relevant sensors, and to avoid the obstacle. But also the robot must be harmless when an obstacle cannot be avoided (situation of a moving obstacle, potentially human). ISO-10128 imposes a maximal velocity of 0.25 m/s. French law imposes that the impact energy must be less than 4J.

- **Duration**:

  The robot's speed is supposed to be constant during an activity (i.e task which takes a defined amount of time with constant velocity). This

choice depends on the security constraints and on the motion energy consumption.

- **Energy**:

  To estimate the energy consumption of detailed Alternative Implementations (AIs) and complete identifications of motion (depending on the chosen velocity), used sensors, and software energy consumption models have been realized. It demonstrates that the amount of consumed motion energy to travel a given distance presents a unique minimum.

- **Localization**:

  the robot must be able to locate themselves in their environment. In exploration missions, the robot is unable to locate itself at the start of its mission but it must do so later to finish his mission and return to the base (e.g. SLAM by creating a map during the mission).

- **Stability**:

  This is the criterion for evaluating the mobility of locomotion systems. They are able to cross and avoid obstacles. The robot must ensure the stability of the control loop whatever the context of the mission. This is a key condition for ensuring the expected mobility.

The energy axes is one of the most challenging in the field. Especially, for outdoor robots rather than indoor which has an infinite energy source available, if they are directly connected to the source which is not the case with autonomous outdoor robots. With a finite energy source, robots are required to perform their mission under the constraints of Time, Safety and of course the available energy resources, batteries.

The robot should be able to calculate the estimated range (i.e Time and Distance bounded by the performance limitation) for its mission given the tasks, the available energy, equipped resources (e.g. Algorithms, Sensors). The problem becomes complex for the robot to predict and estimate the energy consumed during the mission using series of various resources for each task which the robot should decide to use during the mission. this rises the flag for the need of a generic energy consumption model for a autonomous robotic systems.

In this research, A model for global energy consumption for robotic systems, Especially for *Hotel Load* (i.e. Software and Sensors). We present a methodologies to develop these terms of energy providing the preferences between these methodologies. The research has been on Autonomous Underwater Vehicles (AUVs).

1.2 State Of Art

1.2.1 Energy Related Work

The main objective is to find an accurate estimation of the energy consumption before the mission which allows the vehicle to predict the potential range for the given tasks. In addition to the identification of resources which could be used for each task. this idea has been already explored in the past.

in [Bra+01], They used a simple energy metric and a relation between Power, Speed and Range to provide a good estimation while the power requirements of the vehicle is assumed to be constant giving a particular navigation trajectory most of the mission time. But with the increasing complexity of AUVs missions and tasks such as inspection and different set of tasks like maintenance [Lan+15], that simple assumption of constant power consumption and trajectory is not validated. As in [Lan+15], they equipped the AUV with numerous navigation and planning capabilities that give the AUV an adaption advantage to autonomously adapt its resources and algorithms with the available finite energy. With a self-awareness capability of the AUV when the energy is rapidly consumed, its responsibility to manage this situation.

Under a task constraint deployment, Dressler and Fuchs in [DF05] managed to have an experimental energy estimation technique to find the remaining energy for a mobile robot system based on models of the energy consuming parts mounted on the system. Then, the result is used for task allocation and behavior adaptation of each autonomously acting system. Under this estimation, the system is able to calculate ts remaining energy, the required remaining energy for the current allocated task, and therefore, the local state (i.e. the required duration) of the desired operation. The task management is arranged to increase the probability of current task completion at least, an ongoing task can be interrupted and moved to another node if the remaining energy falls below a given limit.

Under Energy and Timing constraint deployment [Mei+06], An investigation is conducted on Pioneer 3XD deployment for some tasks like covering a particular area by number of robots.They present a deployment strategy that can reduce the deployment overhead, and use a smaller number of robots to cover the same area. with both timing and energy constraints are considered; the robots carry limited energy and need to finish the tasks before deadlines.

The power model part is very interesting to our research. They are using a power estimation mode for a generic robot system. Any robotic system should consists of three main groups of components see Figure 1.1. The main components are Motion (e.g. Motors), Sensors and CPU (Embedded

computer and Micro-controller). Their power model includes all these components separately.

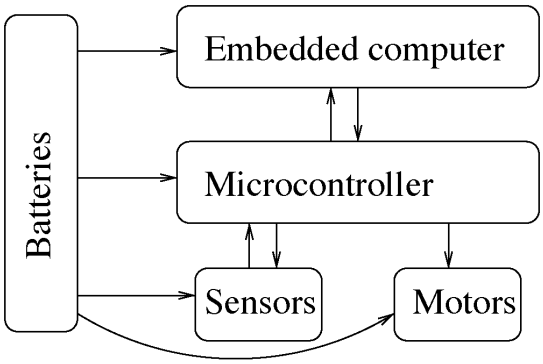


FIGURE 1.1: Robots main components

For motion part, motors transform electrical energy into mechanical energy. The power consumption of the motors is the sum of the mechanical output power and the transforming loss. Let m be the robot's mass, and the ground friction constant be μ. When the robot travels with a speed of v and an acceleration of a , it needs a traction force of $m(a + g\mu)$. Therefore, the output mechanical power is $m(a + g\mu)v$, where g is the gravity constant. The motion power can be modeled as a function of the speed, the acceleration, and the mass:

$$p_m(m, v, a) = p_l + m(a + g\mu)v \tag{1.1}$$

where p_m is the motion power, and p_l is the transforming loss. For DC motors, the power loss consists of armature loss, internal mechanical loss, and eddy-current loss. The power loss increases as the speed increases.

For sensors part, Sensing power varies from different sensors and sensing frequencies. by denoting the sensing frequency by f_s . For video cameras, it is the number of frames per second; for laser rangers, it is the firing frequency. they conjectured that a linear function can model the power consumption of sensors :

$$p_s(f_s) = c_{s_0} + c_{s_1} f_s \tag{1.2}$$

where p_s is the sensing power, and c_{s_0} and c_{s_1} are two positive constant coefficients. Their values depend on the sensors used.

For CPU part, the micro-controller periodically sends commands to motors and sensors, polls sensors' readings, and communicates with the embedded computer. The micro-controller's tasks are usually fixed, and the power consumption of the micro-controller can be modeled by a constant. The embedded computer is more complex than the micro-controller.

In [Par+15], Almost the approach of [Mei+06] is considered to present an on-line method for predicting energy requirements based on the pre-determined power models for a mobile robot. A small mobile robot, Khepera III is used for the experimental study and the results are promising with high prediction accuracy.They considered a general architecture of a tele-operated robot and the components used in the proposed energy management module (in dotted-line boxes) is shown in Figure 1.2:

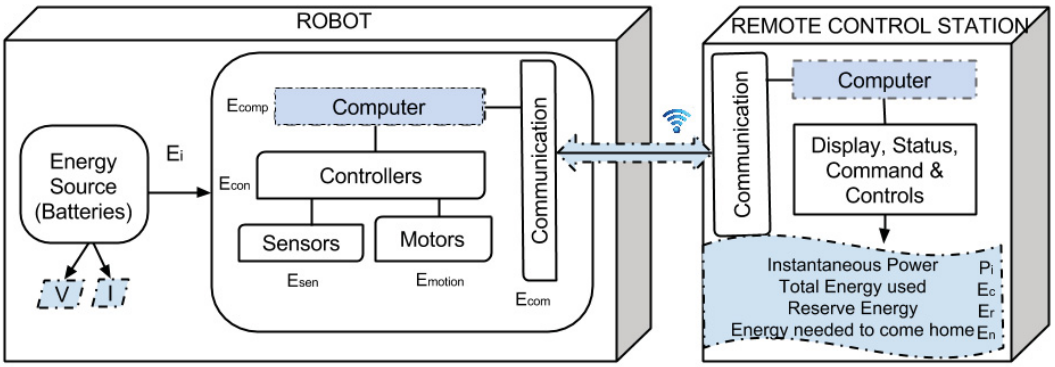


FIGURE 1.2: A general architecture of a tele-operated robot

As shown in Figure 1.2, most of robotic systems are composed of these general components such as motors, sensors, and micro-controllers, embedded computers and communication devices. They build their power model based on the *Active* components at a specific amount of time and on a $on-time$ process for multiple operations can be concurrently programmed or considered as series of sequential operations as shown in Figure 1.3:

For the instantaneous power consumption P_i, They considered to different types of power. Firstly, Dynamic power $P_{dynamic}$ which composed of the sum of the power consumed by dynamic components such as sensors and motors. Secondly, Static power P_{static} from static components such as computer, controllers, and communication devices.

$$P_i = \sum_{i=1}^{n_1} P_{dynamic} + \sum_{i=1}^{n_1} P_{static} \quad , where\ n = n_1 + n_2 \tag{1.3}$$

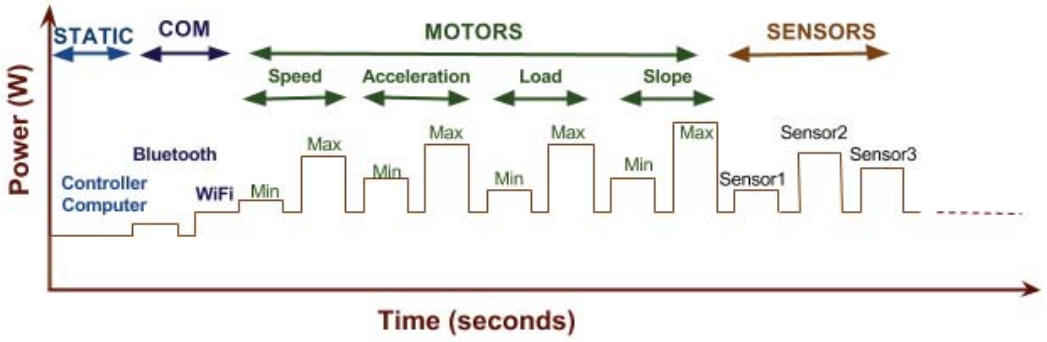


FIGURE 1.3: Sequence of operations to determine power models of various components

Here, $n_1$is the number of static components, n_2 is the number of dynamic components and n is the total number of components in the mobile robot. To obtain the total energy consumed by the robot E_c in driving along a path of length p for t_p seconds, integrate all the n components' energy consumption:

$$E_c = E_{dynamic} + E_{static} \tag{1.4}$$

$$= \int_0^{t_p} \sum_{j=1}^{n} G_{c_j}^i . P_{c_j}^i \tag{1.5}$$

Where $P_{c_j}^i$ is the maximum power consumed by a component j at time instant i and $g_{c_j}^i$ is the weight function applied to each component based on its type:

$$G_{c_j}^i = \begin{cases} 1 & when component j is active and static \\ \alpha & when component j is active and dynamic \\ 0 & when component j is passive \end{cases}$$

considering the robot architecture Figure 1.2, Then the total energy consumed will be cal as following :

$$E_c = E_{comp} + E_{con} + \sum_{j=1}^{n_s} E_{sen_j} + \sum_{j=1}^{n_m} E_{motion_j} + E_{com} \tag{1.6}$$

Where E_{comp} is computer energy, E_{con} is controller energy , E_{sen_j} is activated sensor j energy , E_{motion_j} is activated motor j energy and E_{com} is communication energy. Activated sensors and motors could be identified either by the operator using *teleoperation mode* or autonomously by the robot giving a specific task.

in [Jai+16], The same approach is considered on a Pioneer 3XD mobile robot to create an identification protocol to have the energy consumption models which is used later to have a successful "nominal" mission. Considering different performance axis such as *Stability*, *Safety*, *Localization*, *Energy* and *Duration*.

In [Tiw+19], they presented a methodology to unify the energy consumption models for various robotic platforms Figure 1.4 thereby allowing us to estimate operational range in both offline and online fashions by classifying the consumed energy into two categories: locomotion and ancillary energy for Unmanned Ground Vehicle (UGV) Unmanned, Unmanned Aerial Vehicle (UAV) and Unmanned Marine Vehicle (UMV).

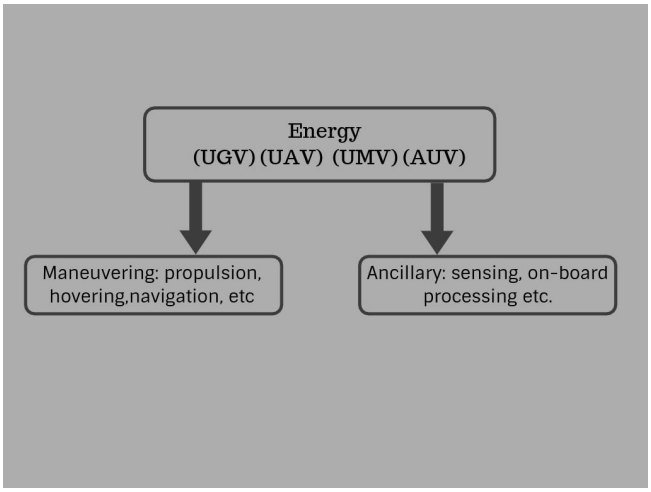


FIGURE 1.4: Kshitij Tiwari energy distribution model

Maneuvering refers to Propulsive Energy (PE) where the dynamic part of motion. Ancillary energy (AE) where the software and sensors consumption. Energy from battery (E) becomes as following:

$$E = PE + AE \tag{1.7}$$

From the algorithmic aspect several works has been conducted.

in [SJU12], a Two principle algorithms are presented to accurately predict the energy usage for the mission. The prediction occurs in real time during the mission of Unmanned Ground Vehicles (UGVs). This approach gives the vehicle the ability to avoid mission failure. the first algorithm is to use a linear regression which is built upon the longitudinal dynamics model. The second approach is to use a Bayesian regression model which requires a prior knowledge (e.g. operator driving style). Their simulation results the effectiveness of Bayesian model during the prediction. These prediction model

could be used as an adaptive behavior of the vehicle during the mission considering the energy constraint.

Also,in [Set11] , generic algorithm solves the issue of unexpected energy shortages. by using cognitive AUV capabilities to re-plan missions trajectory to adapt to unpredictable changing conditions in both the environment and robot. This is achieved with an on-board knowledge-based autonomous agent using on-line learning through a genetic algorithm to optimize re-planned missions. They use Energy Consumption Model of the AUV as :

$$e_t = \frac{(p_s + p_p + p_v).v}{3600.r} \tag{1.8}$$

Where: e_t is total energy on-board (kWh) [fixed for mission]
r is survey range (km) [changes with mission]
v is AUV water speed (m / s)
p_v is on-board vehicle equipment power (W) [changes with control plane use, sensors on different power setting, ballast system use, computations, etc.]
p_s is sonar power (W) [hotel load 2, with sonar range]
p_p is propulsion power (W) [AUV resistance [4] × AUV water speed]

in [ŠH11], They compared navigation and guidance algorithms using an energy scheme as metric benchmark. they presents two ways of how to tackle a problem of vehicle navigation and guidance, and focuses on evaluation of the performance and energy consumption of both methods. A simple energy-aware computing scheme, intended for control, navigation and guidance of autonomous unmanned aircraft, is proposed to try to make the most of both methods - providing good tracking accuracy whenever needed, minimizing power consumption otherwise. This scheme is based on the idea of using simple, non-demanding algorithms whenever possible and switching to sophisticated, more accurate control methods only when required by the mission profile. Simple algorithms may be executed in modules with less computing power, allowing high performance on-board computers (needed for real-time execution of complex tasks) to be suspended, thus conserving energy.

Our research has been conducted on AUVs as an example of robotic system. next, we are talking about Marine Robots with some examples, including AUVs, in general even if they are not autonomous robots.

1.2.2 Marine Robotics

Marine Robotics research has been an important field in robotics since 1970s. Because of its difficulty, The field is very collaborative between several branches such as computer science, engineering, environmental science and oceanology.

In [Lan+15], they showed some recent development area in the Marine robotics. Due to the difficulties that vehicles face in their mission, It should be supported with reliable, robust and highly autonomous hardware platforms. In addition, an Energy-Efficient Platforms which are capable to perform a long range missions with high tasks accuracy. For example, Sea gliders, first prototype in 1990s,represent the debut of the long-endurance robotic platforms in maritime applications. sea gliders are designed based on a buoyancy-driven propulsion mechanism that consumes minimal power and a feedback control mechanism to adjust the direction of motion. the sea gliders are able to perform missions that last several months on a single battery charge. Also, Wave gliders, which are capable to transfer the wave motion energy to provide forward propulsion, have successfully accomplished a mission to cross the Pacific Ocean and are performing day-to-day missions across the world.

Remotely Operated Vehicles (ROVs) is also a growing industrial aspect in robotics. The ROVs are capable to do long missions like some inspection processes in offshore oil industry. Also for . ROVs innovations aims to increase its reliability for deep water surveys and the human interface to give the operator the full control on the vehicle. ROVs are consuming a lot of resources for example an operator boat and a human operator should be present. In addition some wires, in some designs, are connected to the vehicle that carry the information and they can easily reach over kilometers. Wires prevent the flexibility of the ROV while maneuvering. Also, It have disadvantages like operation range because of the long distance between them and the operator. This give the AUVs to be developed to overcome of ROVs in-capabilities.

AUVs became feasible and cheaper alternatives to other existing technologies for numerous applications. AUVs are very efficient at survey/mapping type applications that are too monotonous and tedious for human operators as shown in [DF05]. AUVs have certain clear advantages which can be used with other technologies making them an important tool for scientists. AUVs are important tools for several tasks like hydrothermal-vent discovery, exploration, and sampling as mentioned in [Yoe+07]. AUVs are able to work in places that might be too risky for manned submersibles and inconvenient ROVs.

In research for example, BUBOT research project (Better Understanding Biodiversity changes thanks to new Observation Tools), which focus on the

development and usage of novel tools for exploration, observation and monitoring of the marine environment.

AUVs are designed with limited power systems, batteries, which affect their perform ace especially when they are not under the human super vision as in [Bra+01]. The clear definition of assessment criteria for autonomous AUV attributes can assist developers in selecting the most suitable architecture for AUVs. The challenge is to know what should be measured from the AUVs, and what metrics should be applied (based on the above measures) to determine their Degree of Autonomy (DoA) as in [IL12].

1.3 Contribution

As a part of seeking guarantees for autonomous mission, This thesis will address the Energy performance axis by developing a global energy consumption model which will help the robot to estimate the energy margin required to finish mission task in addition to management of resources.

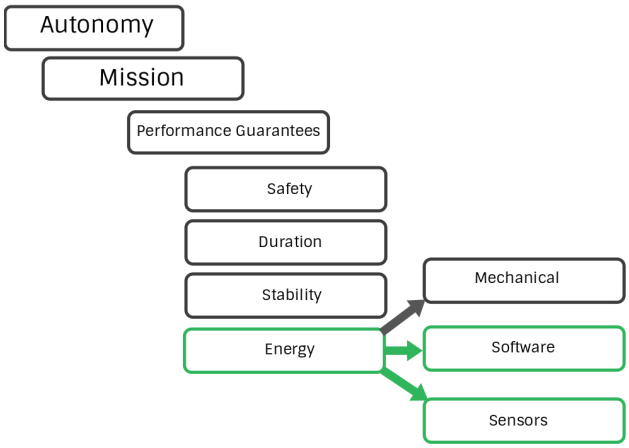


FIGURE 1.5: Contribution Context

As mentioned in Equation 1.7 and as used in [Jai+16], we can use it as the general model for energy and though separating the Ancillary energy (AE) into both software and sensor part. So the Energy model becomes:

$$E = E_{Mechanical} + E_{Software} + E_{Sensors} \tag{1.9}$$

The research is mainly for both $E_{Software}$ software (i.e Algorithms and onboard processing) and $E_{Sensors}$ sensors energy models tacking an AUV as a robotic system.

In the First chapter, We are going through the main architecture of the robot. Explaining the main energy through our robotic system (i.e AUVs). Also, it shows the important components of the robot. Finally, it presents the communication and the software development frame which helps to understand the interaction between the robot and the user (i.e Developer and operator).

The second chapter discusses the main methodologies used to obtain the energy model of the software part going through the different methodologies and tools. It mentions some of state of art about how the energy of the CPU is consumed which helps to create a prediction model for the algorithms.

The Third one talks in details about the different sensors used in the robot. then the tests on which the sensor model is created. Concluding with a proposed Sensor Energy Model.

The Fourth chapter is to analysis the obtained models, It talks mainly how much important these models are and how much we trust them. It also differentiate between the methodologies and tests used in the previous chapters. Also, It addresses the models' validation step.

Finally, The conclusion and Future works where we mention the next steps using these models to create an energy-aware robotic system in the context of performance guarantees and management of resources allocation.

Chapter 2

Robot Energy Architecture

Recalling that the objective is to have an approach to construct the generic energy consumption predictive model that suits all robotic systems. Thus, we should generalize the energy architecture of the robotic system, especially for outdoor robots. As mentioned in [Par+15], [Mei+06] and [Jai+16] , a global architecture shown in Figure 2.6 could be used to identify the general energy consuming components.

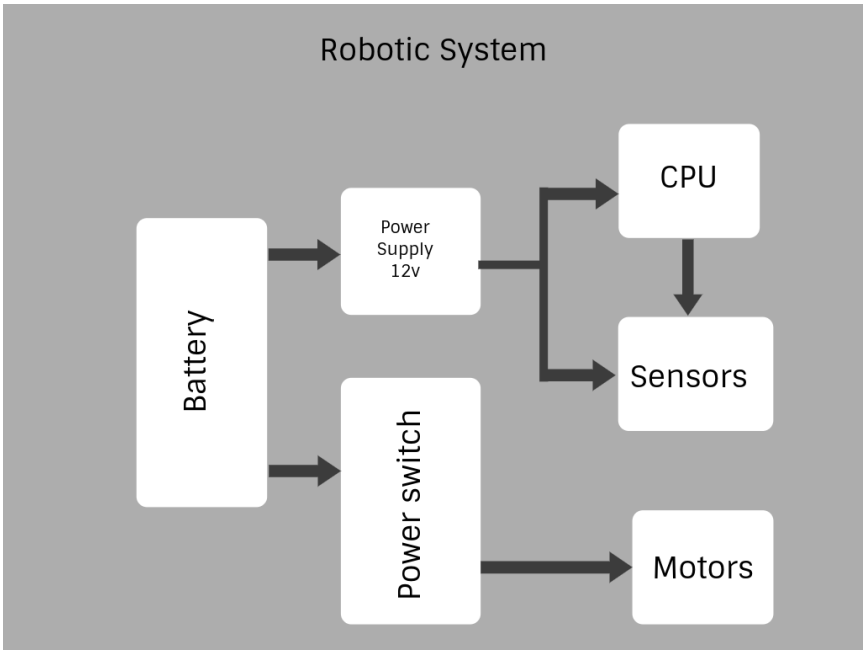


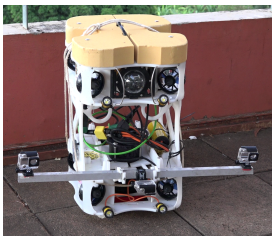
FIGURE 2.1: Power flow for the main components of a robotic system

As shown in Figure 2.1, The power flow aspect not for a control aspect. The main power supply is battery. Sensors could be supplied either by the CPU, Controllers, for the sensor which requires voltages between 3 to 5 voltages or by the power supply for the sensors which requires more than 5v , 12v and 24v maximum. Power switches, Drivers, are detected for the motors since the motors require different voltage during operations to get different speed levels.

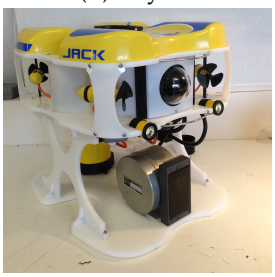
2.1 Robots

In this research, we have conducted our model on some AUVs[1] which have almost the same components, controllers and software but with different dimensions which affects only the dynamic term in the energy model. However, The main AUV is Remi shown in Figure 2.2a and in the next sections will go through the main architecture and component.

(A) Remi



(B) Ulysse



(C) Jack

FIGURE 2.2: Underwater Autonomous vehicles

2.2 Hardware Architecture

As shown in Figure 2.5, The detailed architecture for Remi. All the components exist and are connected to the batteries as shown in Figure 2.1.

[1]For more Platforms check Explore Platformsand for more Info about BlueROV2 check link.

2.2.1 Board: CPU/Controller

The main computer is an embedded card called the UP-Board[2]. The installed operating system is Ubuntu.

with the following ports:

28 [GPIO]	1 high-speed [UART] ports
2400kHz [I2S] ports	119.2MHz [I2S] port
125MHz [SPI] port	2 [PWD] out-ports (293Hz-97.6kHz)

TABLE 2.1: Digital I/O Up-Board quick description.

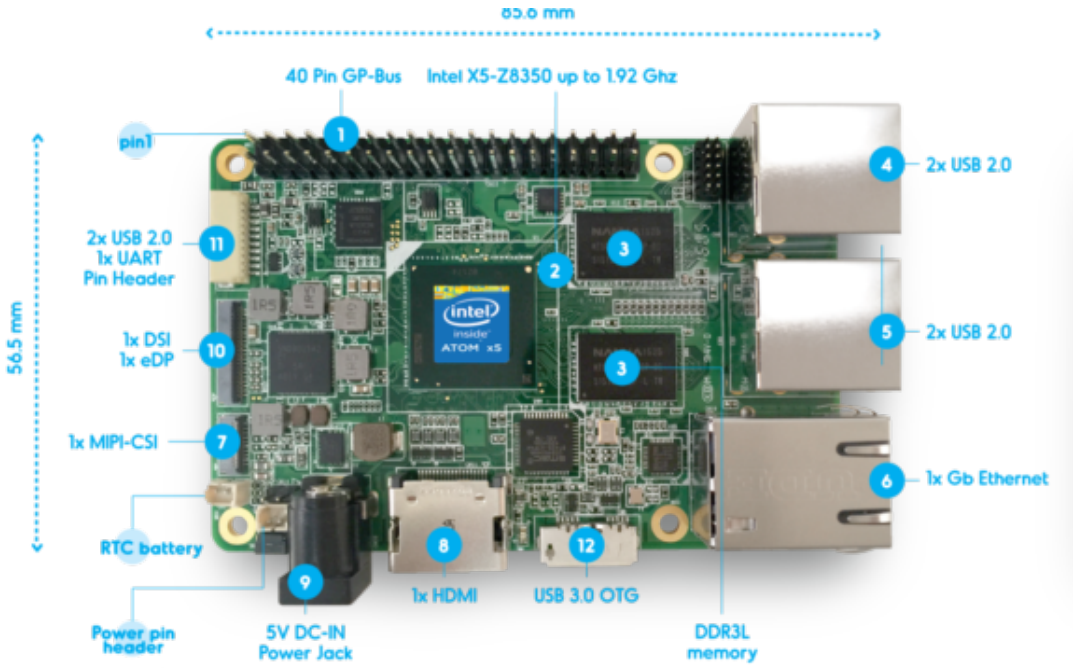


FIGURE 2.3: UP-Board Embedded Card

[2]For more info about the Board check specification link

2.2.2 Motors: Thrusters

The AUVs are equipped with 8 DC motors as actuators. The motors are equipped with propellers which gives it the Thrust force. They as positioned as shown in Figure 2.6.

The T200 Thruster[3], shown in Figure 2.4,is used on a number of underwater ROVs, AUVs, and underwater drones including our flagship underwater ROV, the BlueROV2. The thrusters provide compact but powerful underwater propulsion on the ROV.

FIGURE 2.4: T200 Thruster

An Electronic Speed Controller (ESC) is necessary to run any three-phase brush-less motor like thrusters and motors.

NOTE: ESCs can generate a significant amount of heat when operated. It's important to consider this when mounting and operating the ESC to ensure that it is not damaged by overheating. Most of the heat is generated in the MOSFETs, which are underneath the blue aluminum heat spreader. so, It's important to follow manufacturer's Installation instruction.

FIGURE 2.5: Blue Robotics ESC

[3]For more Info about T200 check this link

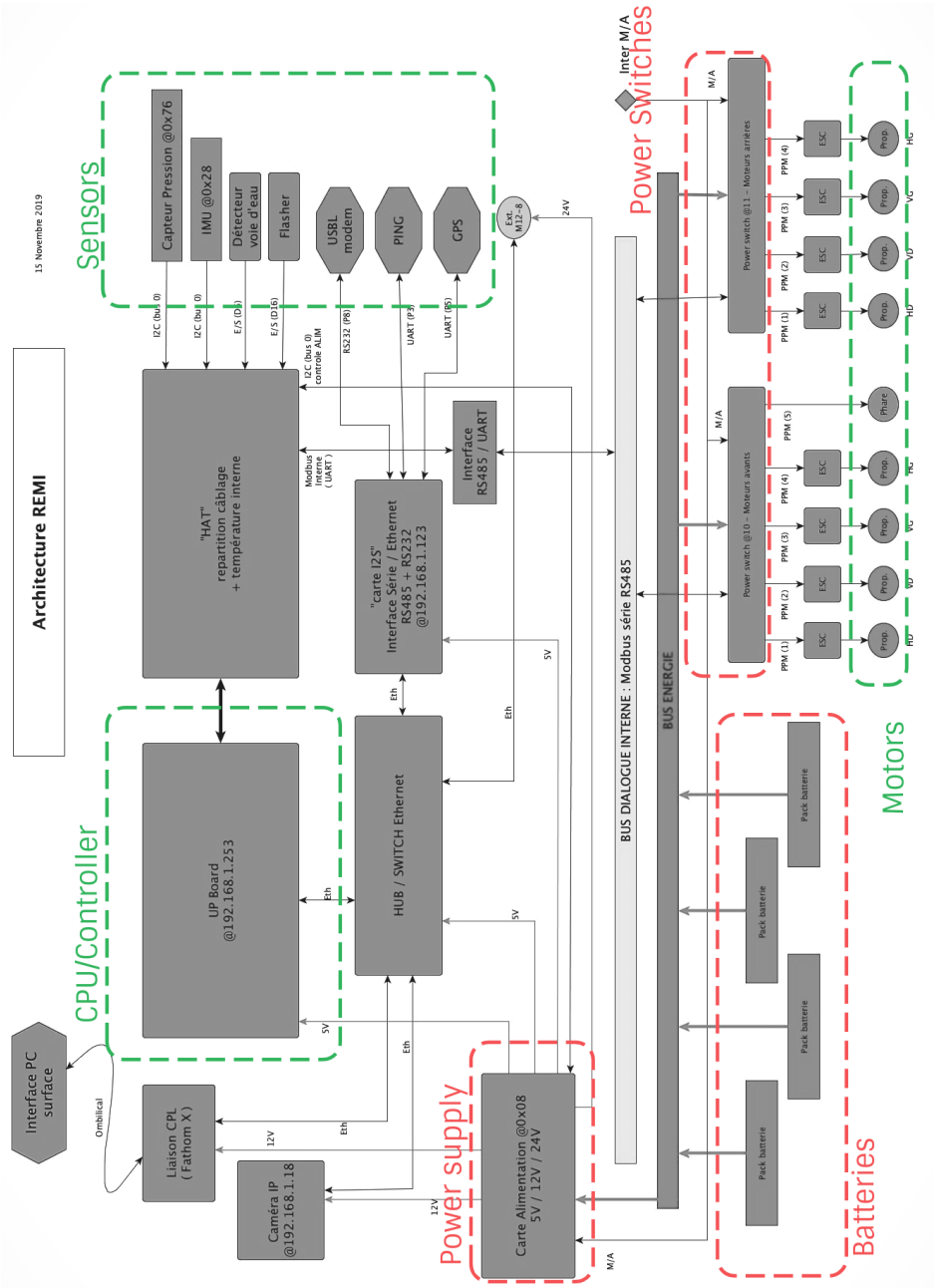


FIGURE 2.6: Remi's Detailed Architecture

Figure 2.6 shows the actual detailed architecture for Remi. As shown, The main components are connected together so the power flow is reaching to each component as in Figure 2.1.

2.2.3 Sensors

There are several sensors used for the robot for different measurements. sensors used:
Acoustic sensors:

- `Ping Echosounder`
- `X150 USBL: BP00795`
- `DVL`
- `GPS`

Non-Acoustic sensors:

- `Pressure/Temperature sensor:`
- `IMU: BNO055`
- `SOS-Leak-Sensor`
- `Camera`

Flasher is a light but not a sensor. However, we will consider it as a related to the camera since its functionality is merged with camera algorithm and control scheme.
we will be exposed to the functionality of each sensor later.

2.3 Development Framework: PID

PID is a global development methodology supported by many tools including a CMake[4] API and dedicated C++ projects.

Here is the list of frameworks officially developed with PID:

- **PID**[5] : PID is a CMake API, but also a framework. It provides many third party or native general purpose projects for developers. Furthermore it provides specific projects like **pid-rpath** and **pid-log** that implements runtime mechanisms of PID.

[4]For More about CMake check this Link
[5]For More about PID check this Link

- **RPC**[6] : Robot Packages Collection (RPC) is a a place where you can find lot of useful packages for develop ping robotics applications.
- **ethercatcpp** [7]: Ethercatcpp provides a way to define and reuse drivers for robot and device based on ethercat technology.
- **hardio** [8] : hardio provides a way to define and reuse drivers for embedded system like raspberry Pi, Beagleboard, etc. This is particularly useful for mobile robotics applications.

As mentioned before, Remi has UP-Board from Intel. so, a **PID-workspace** could be easily installed on the Ubuntu Linux OS.

The **PID-workspace** should have at least these items,:

- hardio: which is hardware configuration for the used embedded system(i.e. UP-Board in our case).
- Algorithms: The developed algorithms which will be run on the system (e.g. SLAM ,perception, etc.). The- algorithms is developed as a **Package**[9] in the **PID-workspacse**.

More items here.

Currently, Remi is working on a development package called **aurov-pid** developed by LIRMM team. It contains the Algorithms and threads which will be run later on the AUV. mainly, our work will be related and inside to this package.

To keep writing into the main context and keep the flow away from technical information, Appendix A contains deep technical information about AUV's communicating and Human interfacing and control.

[6]For More about RPC check this Link
[7]For More about ethercatcpp check this Link
[8]For More about hardio check this Link
[9]Check how a native PID package is made from a raw source code, Link

Chapter 3

Software Energy Model

In this chapter, we are considering the software part of the robot. According to the great variety in algorithms and software could be used by the robot, we present an energy model to predict the consumption of CPU, in particular the algorithmic part.

3.1 State of Art

Because of the increasing usage of computers and other electronic devices (e.g., smartphones, sensors) is continuously impacting our overall energy consumption.

The energy from the CPU is presented in [Mar12], the high-level power consumption characteristics of CPUs and other system components are derived from the circuits used to build those components. Today, virtually all digital systems are built with CMOS[1] circuitry. The detailed circuit characteristics are best left to a study of VLSI[2] design, but the basic sources of CMOS power consumption are easily identified:

- **Power supply voltage**: The power consumption of a CMOS circuit is proportional to the square of the power supply voltage (V2). Therefore, by reducing the power supply voltage to the lowest level that provides the required performance, we can significantly reduce power consumption. We also may be able to add parallel hardware and even further reduce the power supply voltage while maintaining required performance as in [CSB98].

- **Capacitive toggling**: A CMOS circuit uses most of its power when it is changing its output value. This provides two ways to reduce power consumption. By reducing the speed at which the circuit operates, we can reduce its power consumption (although not the total energy required for the operation, because the result is available later). We can actually reduce energy consumption by eliminating unnecessary changes to the inputs of a CMOS circuit—eliminating unnecessary glitches at the circuit outputs eliminates unnecessary power consumption.

[1] CMOS
[2] VLSI

- **Leakage**: Even when a CMOS circuit is not active, some charge leaks out of the circuit's nodes through the substrate. The only way to eliminate leakage current is to remove the power supply. Completely disconnecting the power supply eliminates power consumption, but it usually takes a significant amount of time to reconnect the system to the power supply and reinitialize its internal state so that it once again performs properly.

The proposed software model will be based as in [SMM07], the energy cost of software is computed based on the following formula:

$$E_{software} = E_{comp} + E_{com} + E_{infera} \tag{3.1}$$

where E_{comp} is the computational cost (i.e., CPU processing, memory access, I/O operations), E_{com} is the cost of exchanging data over the network, and E_{infera} is the additional cost incurred by the OS and runtime platform.

considering that Infrastructure power is included in the computational cost of our power models. we can abstract our global power formula as follows:

$$P_{software} = P_{comp} + P_{com} \tag{3.2}$$

At this stage, we developed two models, one for CPU computational costs and one for network communication costs. P_{comp} is therefore equal to the CPU power consumed by software, and P_{comp} is equal to the power consumed by the network card for transmitting software's data. Next, we will detail the CPU and network energy models

- **Global CPU power**: The overall power consumption for the majority of modern processors (CMOS) follows the standard equation:

$$P_{CPU}^{f,V} = C * f * V^2 \tag{3.3}$$

where f is the frequency, V the voltage and C a constant value depending on the hardware materials (such as the capacitance).

Power consumption is not always linearly dependent to the percentage of CPU utilization. This is due to DVFS[3] (Dynamic Voltage and Frequency Scaling) and also to the fact that power depends on the voltage (and subsequently the frequency) of the processor. For example, A process at 100% CPU utilization will not necessarily consume more power than a process running at 50% CPU utilization but with a higher voltage. Therefore, a simple CPU utilization profiler is not enough in order to monitor power consumption.

As in [Nou+12] and according to Equation 3.3 formula 2, calculating the overall CPU power for a given time is equal to calculating a static part (the constant c) and a dynamic part (the frequency f and its associated

[3]For more about DVS check this Wiki

voltage V). For the static part, the c constant is a set of data describing the physical CPU (e.g., capacitance, activity factor). Manufacturers may provide this constant, but in most of the cases this value is not available. To alleviate this problem, we use the existing relation between the overall power of a processor and its Thermal Design Power (TDP)[4] value. TDP represents the power the cooling system of a computer is required to dissipate the heat produced by the processor. Therefore, the overall CPU power can be associated with the TDP as described by:

$$P_{CPU}^{f_{TDP},V_{TDP}} \simeq 0.7 * TDP \tag{3.4}$$

where f_{TDP} and V_{TDP} represent respectively the frequency and the voltage of the processor within the TDP state. The benefit of using this formula is that TDP is a value provided by most manufacturers.

For the dynamic part, the frequency f is associated to a specific voltage V. One or more frequencies can be associated to a specific voltage. Lowering the voltage results in changing frequency. The other way around is also valid. Frequencies used by a processor are provided by the operating system APIs, while voltages are given by manufacturers.

Considering that we are looking for a model for a specific algorithm runing on the CPU (is called a Process for any OS and each Process has a specific ID called PID[5]), we should consider **Process CPU usage**: In order to calculate the CPU usage for a given process (identified by its PID), [Nou+12] proposed to calculate the ratio between the CPU time for this PID and the global CPU time (the time the processor is active for all processes) during a duration d:

$$P_{CPU}^{PID}(d) = P_{CPU}(d) * \frac{t_{CPU}^{PID}}{t_{CPU}}(d) \tag{3.5}$$

Therefore:

$$P_{comp} = 0.7 * TDP * f * V^2 * \frac{t_{CPU}^{PID}}{t_{CPU}} \tag{3.6}$$

- **Network Model**: The network power of a process is calculated using a formula similar to the CPU power formula. We base our model on available information whether they are collected at run time or provided by manufacturers' documentations. as in [Nou+12]:

$$P_{comp} = \frac{\sum_{i \in states} t_i * P_i * d}{t_{total}} \tag{3.7}$$

where P_i is the power consumed by the network card in the state i (provided by manufacturers), d is the duration of the monitoring cycle, and

[4]For more about TDP
[5]For More About PIDs

t_{total} is the total time spent in transmitting data with the network card. Based on these models, The required model for software is:

$$P_{Software} = 0.7 * TDP * f * V^2 * \frac{t_{CPU}^{PID}}{t_{CPU}}(d) + \frac{\sum_{i \in states} t_i * P_i * d}{t_{total}} \quad (3.8)$$

3.2 Implementation

In this section, we present 2 different methodologies could be used for estimating the consumption of a specific software algorithm.

To implement the software power model, We will use a tool called **PowerAPI** Which has the advantage to estimate the power according to a given formula. Between the previous mention equation, we chose to implement Equation 3.8.

3.2.1 PowerAPI

PowerAPI [6]is a middleware toolkit for building software-defined power meters. Software-defined power meters are configurable software libraries that can estimate the power consumption of software in real-time. PowerAPI supports the acquisition of raw metrics from a wide diversity of sensors (eg., physical meters, processor interfaces, hardware counters, OS counters) and the delivery of power consumptions via different channels (including file system, network, web, graphical).

A power meter is an application build with the PowerAPI components that can measure the power consumption of software running on a single machine or on a cluster of machine. This section present each PowerAPI component type and how connect them to build a power meter.

Power meter Architecture

A power meter is a set of two components, a sensor and a formula, used to produce an estimation of the power consumption of a monitored software.

The sensor collects raw data correlated with the power consumption of the software. The formula is a model that use the collected data to compute the power consumption. Both of them are connected by a database that is used to transfer data from the sensor to the formula.

It consists mainly of two items

- **Sensor**:

  A sensor is an independent software that collects raw data correlated with the power consumption of monitored software.

[6]For more info about PowerAPI Check Their Website

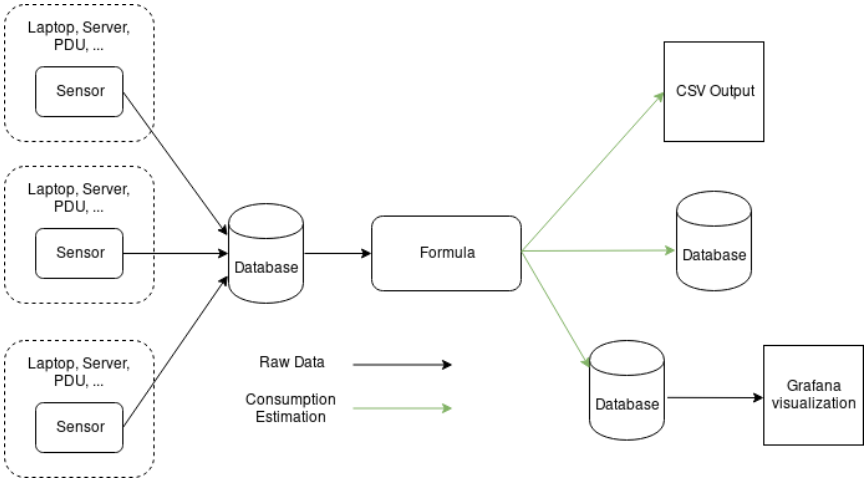


FIGURE 3.1: PowerAPI Arch (source)

Data are collected by querying the hardware's machine that hosts the monitored software. The sensor must be executed on the same machine as the monitored software. The data are collected throughout the duration of the software. For this reason, the sensor must operate in parallel.

Collected data are stored in an external database to make the data available to the formula. This database may be hosted on an other machine.

Usage: Because they collect from different hardware, each sensor are very different from one another. Refer to each sensor documentation to know how to use them.

- **Formula**:

  A formula is an independent software that use model to compute the power consumption of monitored software from the data collected by the sensor.

  Sensor Connection : A formula is connected to a sensor via a database (e.g MongoDB[7]). The sensor writes the collected data to the database and the formula reads this data from the database.

In our case will just monitor the power consumption during running the Algorithm, We have tried it on a Tower of Hanoi[8] Algorithm which uses a recursive[9] function.

We managed to implement the formula Equation 3.8 and to monitor the Power estimation over the time of running the algorithm. PowerAPI is managed to give a power estimate using our implemented model with respect to time and monitor these data as shown in Figure 3.2.

For more about this particular PowerAPI implementation, Pleas check the documentation repository.

[7]Find More About MongoDB here
[8]More About **Tower of Hanoi** problem Check this Wiki
[9]More About **Recursion** Check this Wiki

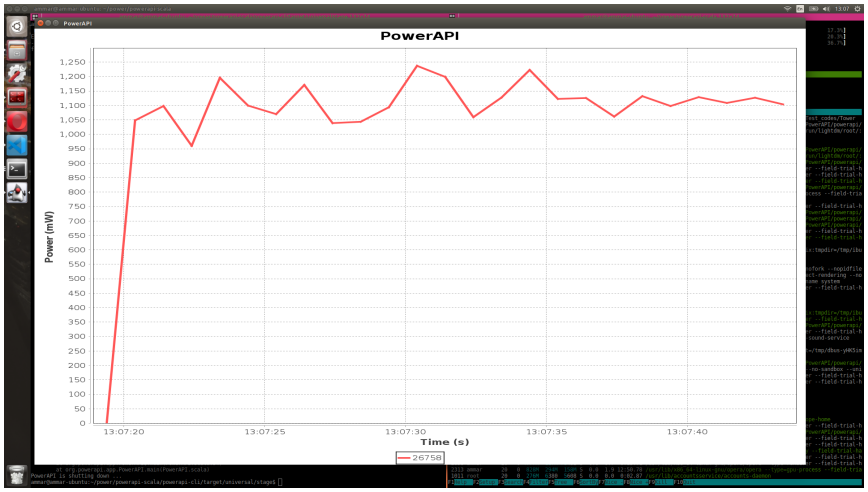


FIGURE 3.2: PowerAPI: Tower Of Hanoi Power Estimation

Of course, Tower of Hanoi algorithm is just a software example to implement the functionality of our model and PowerAPI. But the AUVs have different algorithms. For instance they can use

- SLAM or AMCL for localization and mapping.
- EKF for sensor fusion
- A* for path planning
- Perception algorithms (e.g. Camera image processing and object detection)

3.2.2 CPU Energy Meter Tools

Another approach could be used but with less accuracy, Using Energy tools provided for Linux and Windows systems.

These tools are capable to give an estimation of the total energy consumed by the CPU within a given time. The required algorithms should be run at the same time of launching these tools.

Some of these tools are:

- **Intel® Power Gadget**[10] (Windows and MacOS):

  Intel® Power Gadget is a software-based power usage monitoring tool enabled for Intel® Core™ processors (from 2nd Generation up to 10th Generation Intel® Core™ processors).

- **Power Statistics** (Linux):

  There is a new power statistics history window. This can be accessed by clicking the battery item in the application indicator menu then selecting Laptop Battery tab.

[10]For More About IPG check this

- **Powertop**[11] (Linux):

  this program provides information on per process/device power usage, Check thisGuide.

- **Powerstat**[12] (Linux):

  Another alternative that measures process/device power usage is powerstat that was written for Ubuntu by Colin King. There is a detailed review of its features on hecticgeek.

- **CPU Energy Meter**[13] (Linux):

  Very Interesting tool to find the Energy per Kj consumed by the CPU

- **likwid**[14] (Linux):

  is a simple to install and use toolsuite of command line applications for performance oriented programmers. It works for Intel, AMD, ARMv8 and POWER9 processors on the Linux operating system. There is support for ARMv7 and POWER8 but there is currently no test machine in our hands to test them properly.

The main concept using these tools is very simple:

- Launch a tool
- Record the estimated power consumption P_{Idle} (Which will be the reference of measuring).
- Launch the Algorithm on the CPU.
- Record the estimated power consumption $P_{Running}$.
- then $P_{Software}$ will be calculated manually as follows:

$$P_{Software} = P_{Running} - P_{Idle} \tag{3.9}$$

3.3 Conclusion

In this chapter, we proposed a Software Energy Model with some implementation methods. Some tools are presented to help to implement this model. Also, we should mention that this implementation is providing the robot with prior knowledge of the power consumption of the software used. This is different from a real time tracking during the mission in which the robot should use the software first during the mission to estimate the consumption. Next, we considering Sensor energy model and stating state of art and proposing some methodologies.

[11] For More About Powertop check this repo
[12] For More About Powerstat check this
[13] For More About CPU Energy Meter check this repo
[14] For More About likwid check this repo

Chapter 4

Sensors Energy Model

In this chapter, we present the sensors' energy model and some related work. Before the model,The functionality of each sensor is introduced. Then, we present some of tests methodologies which gives the model more accurate values. Including with Tests Results and the model.

4.1 Related Work

Some works related to the field of Internet Of Things (IOT) have presented energy models for the sensor consumption.

In [Ter+09], They based their node[1] energy consumption on $E_{Sensors}$ which is an important part for the node. y. A node is composed of sensors, analog to digital converters (ADC), processing unit, memory, RF transceiver and battery. The total energy dissipated by the sensor node could be expressed as the sum of the energy dissipated by each element:

$$E_{Node} = E_{Sensor} + E_{ADC} + E_{\mu C} + E_{Rec} + E_{Trans} \tag{4.1}$$

The energy dissipated by an element depends on its state: active or idle. First, the energy dissipated by the sensor E_{Sensor} is expressed as follow:

$$E_{Sensor} = P_{on}(t_{stable} + t_{measure}) + P_{off}(t_{cycle} - (t_{stable} + t_{measure})) \tag{4.2}$$

where t_{stable} corresponds to the stabilization time of the sensor and $t_{measure}$ to the duration of the sensing phase which depends on the number of measure to realize and on the ADC conversion time. t_{cycle} defines the periodicity of the micro-sensor node activity.

[1] IoT is a combination of hardware and software - where in sensors and actuators play the role of capturing data and taking action respectively. Embedded modules, packaged devices, smart thermostat, wifi and iot enabled tubelight and the iot gateways or controllers can all be classified as iot **Nodes**.

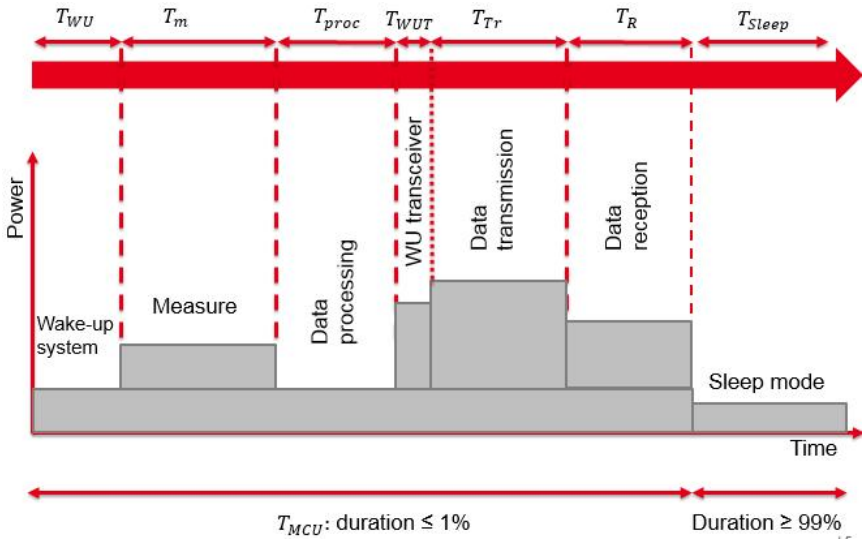


FIGURE 4.1: Sensor working scenario

In [Bou+18], They based their sensor working scenario as shown in Figure 4.1 inside the active node. Composing it as an operation cycle which runs at frequency of the embedded card (i.e Micro-Controller).

The total consumed energy E_{Total} used by the communicating sensor for one cycle is given by:

$$E_{Total} = E_{Active} + E_{Sleep} \tag{4.3}$$

where E_{Sleep} and E_{Active} are the dissipated energy by the node in the sleep mode and the total energy consumption during the active mode of the micro-controller, respectively. E_{Sleep} is expressed as:

$$E_{Sleep} = P_{Sleep} + T_{Sleep} \tag{4.4}$$

where P_{Sleep} and T_{Sleep} are the power consumption and the time duration in the sleep mode, respectively.

And E_{Active} is expressed as:

$$E_{Active} = E_{WE} + E_m + E_{proc} + E_{WUT} + E_{Tr} + E_r \tag{4.5}$$

where E_{WE} , E_m , E_{proc} , E_{WUT} , E_{Tr} and E_r are, respectively, the consumed energies in the system wake-up, the data measurement, the micro-controller processing, the wake-up of the LoRa[2] transceiver, the transmission mode and the reception mode.

Mainly almost every term depends on frequency of the micro-controller f_{MCU}. for example , the consumed energy E_{Tr} by the transmit mode is expressed as:

$$E_{Tr} = ((P_{ON} * f_{MCU}) + P_{Tr}) * T_{Tr}$$

[2]about LoRa Check [Bou+18]

where P_{Tr} is the dissipated power by the transmit mode and T_{Tr} is its time duration.

In this research, We are considering the f_{MCU} is constant and the frequency of recruitment f_{rec} for all the threads, algorithms and sensor data processing is equals to 10 Hz. So, the Active and Sleep terms are almost sensors (as will be seen with the tests, the power fluctuation dose not exceed 5%). Since there are a lot of terms (like the time for each stage) are hard to detect, we will base our model on the tests results considering 2 important information P_{Active} is constant and whether the sensor is active or not.

4.2 Sensors

Sensors are mainly considered as fundamental parts for any robotic system. Their mission is describing the environment around the robot with high level of accuracy. This give the robot a better chance to decide the best decision within a particular situation given a set of collected data from the sensors. After data processing, treatment and fusion, in some cases, the result is almost the best estimation describing the reality. For example, If the robot needs to ensure that it's heading north, the compass should give an accurate read around zero degree.

One of the most important part about data is calibration. So, before starting a mission, it is strongly recommended to configure (i.e calibrate) the sensors. This is due to the changes in the geographical areas and the fact that the sensors will change the measurements so the result of the control stage calculations will be poorly predicted. This section will briefly but concisely describe the sensors used as well as their calibration.

The following Table 4.1 contains an overview for the sensors used in Remi. Also, It contains some resources could be useful for more information about each one (e.g. Data sheets, User's Guide ...).

Sensor name	ID	Protocol	PID	Tested	Calibrated	Information
Leak indicator	SOS-Leak-Sensor	Digital	done	yes	no	link
Flasher	Led	Digital	done	yes	no	Link
Bar30	MS5837	I^2C	done	yes	no	link
IMU	BNO055	I^2C	done	yes	yes	link
Camera	-	IP	-	yes	no	Digital and Analog
GPS–Robot	DP0104	UART	-	yes	no	NEO-M8N GNSS + LIS3MDL compass XL link
Echo-sounder	Ping Sonar	UART	done	-	no	Altimeter and echo-sounder (30m range) link
SeaTrac	X150 USBL Transponder	UART	done	-	no	Depth,attitude and orientation link

TABLE 4.1: Overview of Sensors connected to Remi

4.2.1 Pressure sensor : MS5837

This sensor provides depth which is estimated indirectly thanks to pressure. This pressure sensor can measure up to 30 Bar (300m depth) with a depth resolution of 2mm.

This sensor includes a temperature sensor accurate to ±1°C, with data also accessible through I2C.

Before its utilisation a previous calibration[3] is strongly recommended. The gel-filled sensor must be completely dried once per day or the pressure and temperature readings will drift.

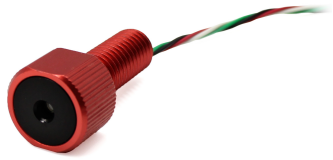


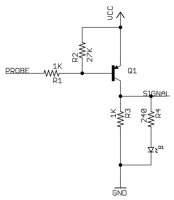
FIGURE 4.2: Pressure sensor (MS5837)

4.2.2 SOS Leak Sensor

The SOS Leak Sensor can detect water leaking into an improperly sealed Watertight Enclosure quickly and reliably, before any major damage can occur.

Actually, It's only a transistors which has only two states : ON and OFF. Also, Provided with small diod as shown in **??**

(A) Component



(B) Technical Schematic Design

FIGURE 4.3: SOS Leak Sensor

[3]Calibration guide can be found here

4.2.3 Echo Sounder

Ping Echo-sounder Sonar measures the distance to objects underwater. The Ping has a 30 meter range, 30 degree beam width, and 300 meter depth rating. It can be used as an altimeter on an ROV or AUV, for bathymetric surveys aboard a USV or boat, or as an obstacle avoidance sonar for any marine robot.

FIGURE 4.4: Echo-sounder

The Ping[4] device emits a brief 115 kHz acoustic pulse from the transducer at the face of the device. The device then listens back and measures the strength of returned acoustic energy. As the acoustic sound waves travel through water they reflect or echo off of solid objects and then travel back to the device. The Ping then calculates the distance to the object with the equation:

$$distance = known speed of sound in water * (measured time for echo to return/2). \tag{4.6}$$

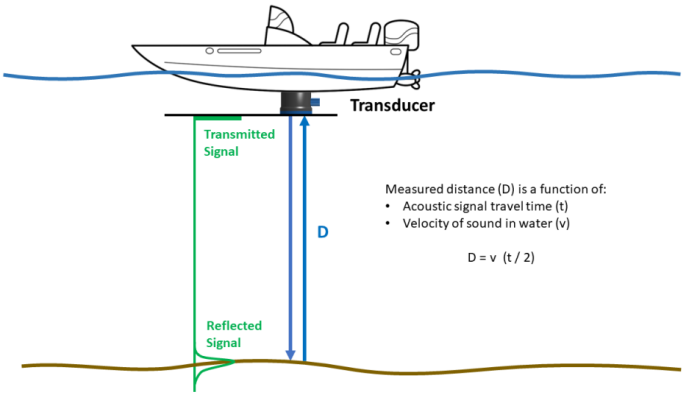


FIGURE 4.5: Echo-sounder operating principle

[4]User Guide For more Info about echosounder sonars work check this Wiki

4.2.4 Sea Trac: USBL

One of the challenges in underwater robotics is the location, since GPS is not suitable underwater, for this the USBL sensor will be used. This is an acoustic sensor that has two parts: one placed in the underwater robot and another in the upper base station.

It's very useful in

- Remote control and interrogation of sub-sea equipment and
- Remote depth, attitude and orientation measurement.

USBL[5] is used as **Request/Response** device and there are test communications between beacons using the **"PING"** and **"ECHO"** protocols.

- **PING**: The PING protocol forms the simplest and shortest acoustic message possible addressed to the required beacon, and requests a USBL response is made.
- **ECHO**: The ECHO protocol forms a data message addressed to the required beacon, and with a user specified payload of up to 30 characters. The message is sent and upon successful reception, re-transmitted as a response back to the beacon.

Due to their data payload, Echo messages are significantly longer (in time) than PING messages, and so are useful for testing the integrity of an acoustic communication channel.

FIGURE 4.6: X150 USBL Transponder Beacon

[5]For more Info about USBL check this Guide and User Manual

4.2.5 IMU

BNO055[6] is a multiple embedded sensors which allows us to measure accelerations, angular velocities, magnetic fields, Quaternary's and Euler's angles.

The BNO055 is a System in Package (SiP), integrating a tri-axial 14-bit accelerometer, a tri-axial 16-bit gyroscope with a range of ±2000 degrees per second, a tri-axial geomagnetic sensor and a 32-bit cortex M0+ micro-controller running Bosch Sensortec sensor fusion software, in a single package.

Avery important thing is that this sensor should be calibrated before a real test because geographical zones are mainly magnetic variations and sensor needs to be configured in the zone where it will be used. By the way, even the effect of the motors can affect the magnetic sensor.

For calibration there are different methods (C++, Python, Qt,. . .) such as FreeIMU software one of the most popular in this field.

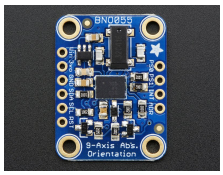
4.2.6 Camera

We are using two types of generic camera types: Digital and Analog. it's used for perception, 3D construction and some path planing algorithms. in addition a Flashing LED is used synchronously with cameras.

(A) Analog Camera

(B) Digital Camera

(C) Flasher



(D) IMU : BNO055

FIGURE 4.7: Cameras, IMU and LEDs

[6]For more about IMU check Data Sheet and this Link

4.3 Methodology

The goal is to define how much energy which every sensor use during the mission.

That requires defining the correct current and voltage requires to operate the sensor in every condition that the sensor can operate with.

$$E_s = I * V * operationtime \tag{4.7}$$

For some sensors, they just turn OFF and ON while operating. we will call turning off as **Idle** state and **Operating** state for ON condition.

Some sensors, like USBl, can have multiple operating states (e.g. Receiving or Sending Data) which requires more tests to accurately define energy parameters.

In this section, some tests methodologies will be presented stating the conditions and parameters to get correctly the operating current and voltage. Then, presenting tests' results and concluding with the final sensor model stating the compromised values between the different tests.

4.3.1 Manufacturers Electrical Specifications

First, a benchmark should be used to identify whether our tests values are correct or not. Every sensor has a data sheet where operating conditions and states is presented. Including Maximum and Minimum values will help to correctly getting the right values since tests' results must be located between Max. and Min. values.

The Following Table is stating the electrical specifications for each sensor.

Sensor	Voltage (V)	Max. Current (mA)	Power (mW) (Idle/standby)	Power (mW) (operating)
Pressure/Temperature	3	1.25	0.03	3.75
SOS-Leak-Sensor	3.35	20	67	0
IMU: BNO055	3.6	12.3	1.4	44.3
X150 USBL: BP00795	9:28	-	500	10,000
Ping Echosounder	5.5	100	-	550
GPS	3.6	67	-	241
USBL	9-28	-	0.5	10,000
DVL	12 : 48	1500	-	1300 (Avg.)
Light	6	1400	0	8400
digital HD USB camera	5	110	0	1100
Analog Camera	12	120	0	1440

TABLE 4.2: Optimal Sensors' Electrical Specifications

4.3.2 Test 1

The first test applied on the sensor is using the traditional measurement tools like multi-meters and oscilloscopes

The next step is to configure the sensors to have to both modes: Operation mode and Standby mode.

The operation mode, the sensor is measuring for 1000 times and print the reading to the terminal, is to have a steady state current during. So, we have the chance to measure the voltage, current and the power.

Multi-meter is measuring the values on the head of sensors, Not considering the power from the embedded board.

The following table shows some the results from this test

Sensor	Voltage (V)	Current (mA) (standby)	Current (mA) (operating)
Pressure/Temperature	3.35	0.03	0.945
SOS-Leak-Sensor	3.35	11.47	0
IMU: BNO055	3.35	11	14.3

TABLE 4.3: Test 1: Results for some Sensors

Comparing the initial results with the expected from data sheets shown in Table 4.2, There is some drift to the accurate reading.

Reasons for getting drifted readings:

- This test is dependent on the human skill for reading. Since the reading is wrote down with manually, It is almost an estimate from the operator not a accurate estimation.
- Adjust the wiring correctly with sensors headers requires a lot of wires which might affect the functions of sensors also the measurements tools
- Sensors' Functionality might be dependent on the embedded board. Since the board is not considered, Reading might be different for different conditions (e.g. different algorithms)
- Readings are not saved. so, we don't have a chance to analyse it.

Also, Some sensors are not suitable for that test which requires a lot wiring with sensors pins. So next, we are proposing another test procedure.

4.3.3 Test 2

In this test, The embedded board is considered. Tacking the advantage that Power supply 12v, "Carte d'Alimentation" in Figure 2.6, which supply the energy to both Up-Board and Sensors. By making a PID-Package (i.e hardio-a2e3s)for this particular component, we can find both Voltage and the Current due to an embedded voltmeter and Ammeter. These readings can be easily saved in a *.CSV* file where we can analyse and monitor the data with respect to time of operation.

Considering that Voltage and Current Readings belongs to the supplied power to sensors and board.

Important Note : Since The board is always exist, the test operation will go for 3 steps:

- Connect only the one sensor to the board.
- Operate the board and sensor is in Idle state.
- operate the sensor for few seconds

By these steps, we are considering the board energy as the reference of measurements (e.g. Zero of idle sensor) which will be removed during data analysis later.

The following figures show some of sensors' tests.

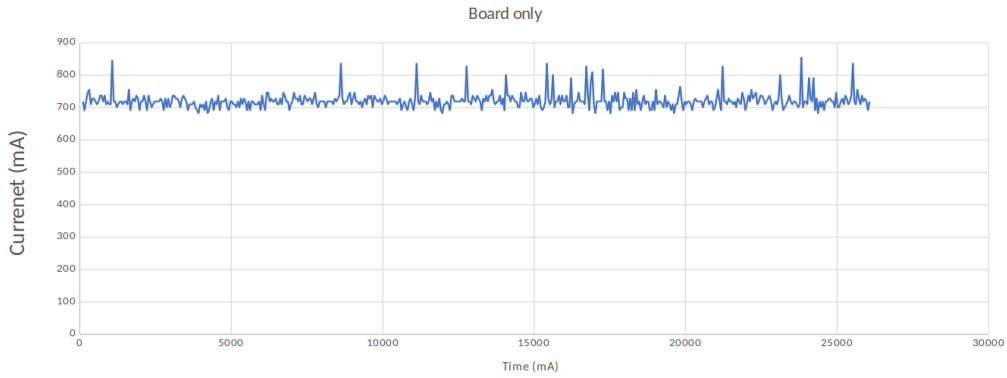


FIGURE 4.8: Board

As shown in Figure 4.8,Figure 4.9,Figure 4.10 and Figure 4.11, Sampling rate is very important. Some times the data are fuzzy to analyse. so to avoid Aliasing[7], half of the Battle is with choosing the right sampling rate which give the analysis an accurate data.

[7]Chick this Wiki for more about Aliasing

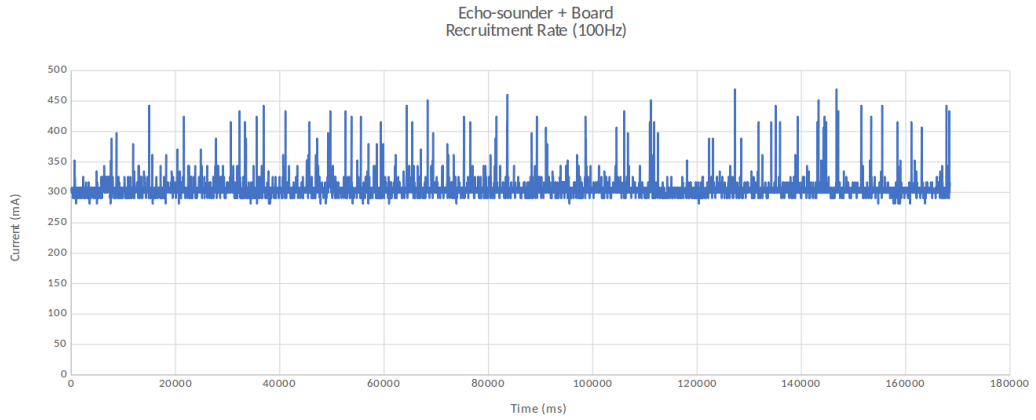


FIGURE 4.9: Echo-Sounder (Rate 100 Hz)

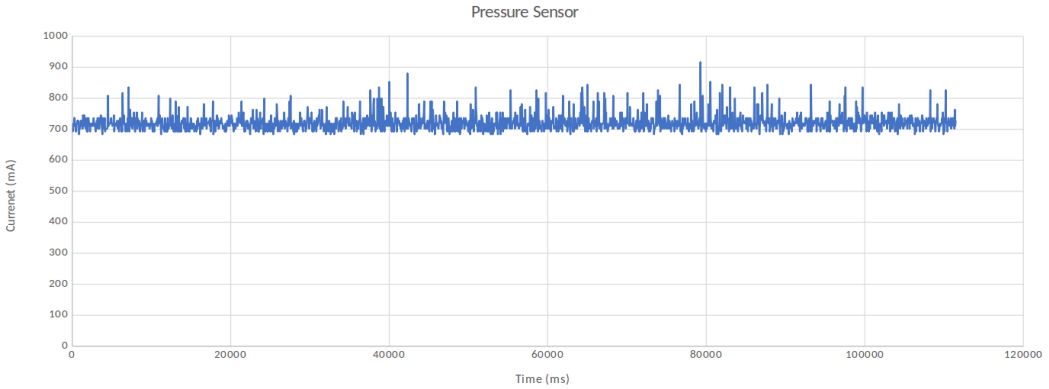


FIGURE 4.10: Pressure Sensor

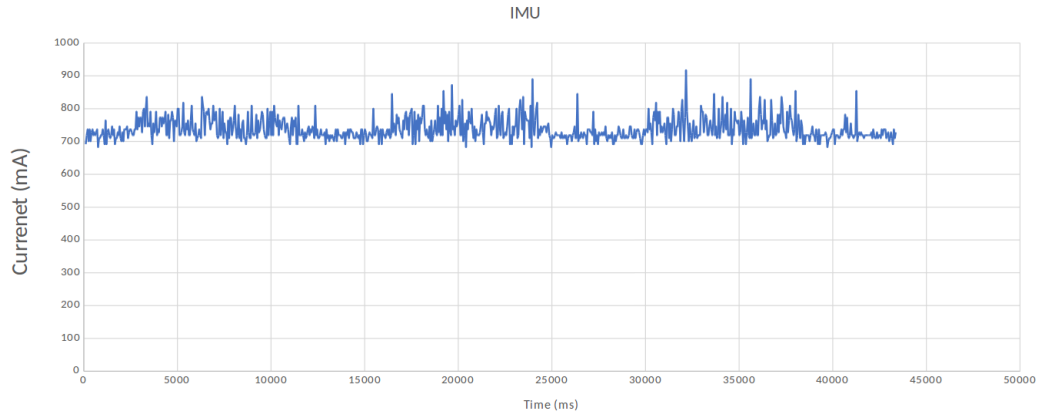


FIGURE 4.11: IMU: recruited 3 times

Some sensors have not been tested like : USBL and DVL.

USBL is especially needed to be tested online to take the average of power consumed from each protocol at different recruitment frequencies. Also, the time required to finish each protocol. Meanwhile the online test is not available, it's recommended that The Response Time (10 ms, 100Hz) can be used

to be the average of transmitting operation for either PING and ECHO protocols at frequency 50Hz (20 ms to send another Request). With this configuration the power from the USBL will be 5W.

Also, this applying for DVL.

- Frequency of operation 1 MHz
- DC input 12-48 V
- Maximum peak current 1.5 A
- Average power 1.3 W

Concluding, this test is more accurate than the previous one and it gives more data to analyse. Also, It shows the sensors behavior during it's operating. Next, we propose the final senor model for our robotic system.

4.4 Sensors Model

The power consumed from all the "Carte Alimentation" components are constant except for the Main processing board and USBL Device. The processing board could use different algorithms. Also, not all the components would be connected every mission.

The following Table 4.4 shows the results from the latter test after analysing and averaging.

Component	Power (mW)	Notes
Pressure/Temperature	2.76	Constant
SOS-Leak-Sensor	38.4	Constant
IMU: BNO055	342	Constant
X150 USBL: BP00795 (50Hz)	5,000	Frequency dependent
Ping Echosounder (100Hz)	1,750	Frequency dependent
GPS	241	Constant
Light(only single Flash)	8,400	Constant
DVL(Avg.)	1,300	Constant
digital HD USB camera	1,100	Constant
Analog Camera	1,440	Constant

TABLE 4.4: Sensors Power

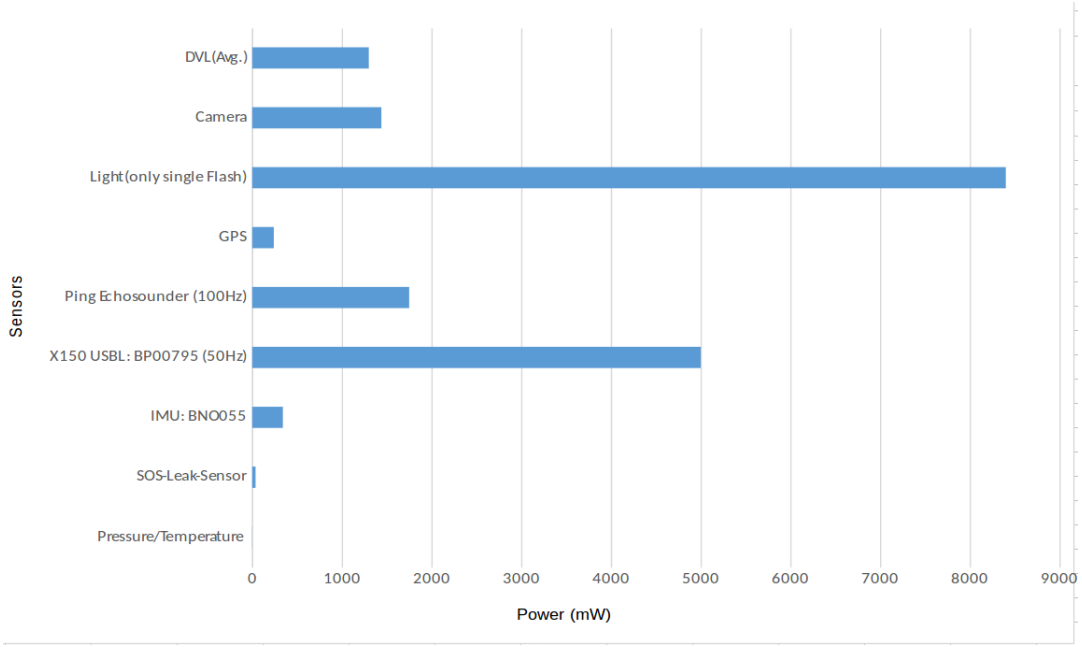


FIGURE 4.12: Sensors Power Graph

As mentioned before, All the consumption values are dependent on the f_{MCU} frequency of the micro-controller. but with a constant f_{MCU} these values are constant. However, Retarding to USBL, Echo-Sounder and DVL depend on another frequency $f_{Operation}$ because they are *AcousticSensors*. $f_{Operation}$ is defined according to the type of mission and the tasks included. This gives especially these sensors to be experimental test (i.e. under the water) to find out the exact relation between $f_{Operation}$ and consumption *Power*

Concluding, The sensor model is related with the power from "Carte Alimentation" (CA). It composes of sensors and the embedded card (Board).

The Power Model for can be defined as:

$$P_{CA} = P_{board} + \sum_{i}^{N_{max}} \beta_i * P_i + \alpha * P_{Flasher} \tag{4.8}$$

Where :

- β_i is a Boolean coefficient denoting if the device is used or not.
- α is number of flash lights used.
- N_{max} is number of sensors used at the moment and is dependent on the software used.
- P_{board} is 8 watts in average (Based on the current setup $f_{MCU} = 10Hz$)

To get the instantaneous Energy the robot should identify the time within which an update occurs to the running threads. and following the next equation:

$$E_{CA} = P_{CA} * t_{update} \tag{4.9}$$

Where t_{update} time for requesting new measurement and based on the f_{rec} recruitment frequency of the control scheme and the threads that operating sensor measurements.

4.5 Conclusion

In conclusion of this chapter, we have proposed a Sensor Energy Model to give a prior estimate for the consumption based on experimental tests. Next, we are going through the analysis of the proposed models and their methodologies and experiments. Then, mentioning how to use these model and validate their accuracy.

Chapter 5

Models Analysis

Recalling that our objective that to obtain a general energy consumption predictive model for a robotic system which is accurate, reliable and suitable for long term missions.

For instance, the robot will face a situation where it should allocate and manage resources in the constraint of energy. The robot's mission will be a sequence of several activities (time defined activities), see Figure 5.1. Each activity could be done with several combination (i.e. Alternatives that can make the same task even if they have different accuracy or other trades off aspects) of algorithms and sensors. Here the robot should decide which combination will satisfy it's performance axes ,including energy. in Figure 5.1, robot has to choose the second combination. since the $P_2 < P_1$ which is required for complete its mission under time and energy constraint. so, models reliability and accuracy are very important of a successful mission .

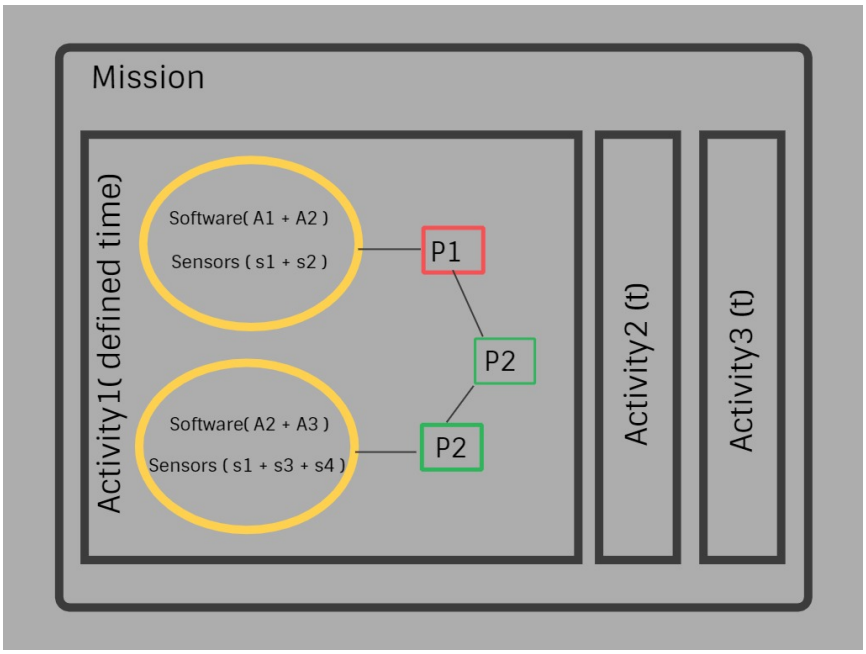


FIGURE 5.1: Robot's Mission

In this chapter, we answer some questions to analyse the models obtained in previous chapter for both software and sensors and their tests such as:

- *"How much reliable and accurate these models?"*
- *"What are the differences between the tests?"*
- *"How to validate these models?"*
- *"How to use these models?"*
- *"Will Software and Sensor models make a difference in the global energy model?"*

5.1 Software Energy Model

In chapter 3, we presented two ways to find out how much energy consumed by the a specific algorithm. First one is using Equation 3.8 as an input model to PowerAPI package to give a estimation of the power. The second is using other tools which give an overall power consumption of the CPU and then we calculate the power manually by Equation 3.9.

PowerAPI is powerful for some reasons:

- The predictive model:

  With PowerAPI, we managed to use a particular formula for estimating the power. This gives the freedom to modify the models or use several other state-of-art formulas. this becomes a great advantage even at later stage of the project if we decided to add or neglect some terms in the model.

- Process Level:

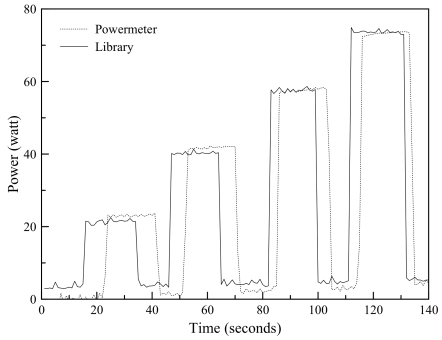
  PowerAPI is more preferable because it woks on the process level of the CPU kernel. This gives an accurate estimation only for the required algorithm without the risk of interfering of other algorithms.

- Validated:

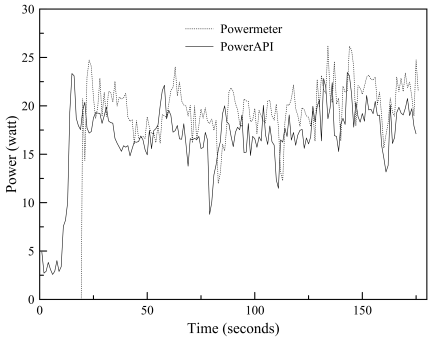
  There are some tests presented in [Nou+12] which shows that PowerAPI is only 0.5% of estimation error, see Figure 5.2.

- The Architecture:

  As mentioned before that PowerAPI is using a sensor modules which are the responsible to measure some CPU variable like Frequency f and Voltage V and uses it as an input to the formula. So, the user does not have to input these variables or to calculte the power manually.



(A) Stressing the processor cores using the STRESS command.



(B) Running a video under MPlayer.

FIGURE 5.2: PowerAPI Validation

- Results Monitoring and Recording:

  PowerAPI has the ability to record the estimated power so, it becomes available later to analyse. This option is useful to ensure that along the time the code consumes the same amount of energy which is really recommended for the long term missions.

The other test is not really recommended because of some reasons:

- Overall consumption:

  The other tools mentioned for this test is not at the process level of the kernel but they give an overall power consumption. This estimation contains all the power from all the algorithms running on the CPU which will give a lot of calculation error while both defining a reference value (i.e. Idle State) also to the running value since we are not sure that only the required software is the added value because of software interfering on the background. This makes the user to do the test several times and then calculte an average value for the power consumption which drops it into a probability problem since we should answer the question *"How many tests' values required to have almost an accurate value (say 90% accurate)?"* and that is time consuming.

- No result records:

  These tools does not have a recording option for the results obtain and this give it to user hand to read and save the results by him self. As shown previously in Figure 3.2 for PowerAPi implementation on Tower Of Hanoi algorithms, There is a fluctuation in the resultant power. so, using the other tool does not provide this kind of observation which make the user in lucky way to read the average value.

- No CPU model:

Some of these tools uses unknown CPU estimation models which could not be reliable. Even the one we could find out these models, we can't change it to perfectly suit our general model while validation.

So, PowerAPI is clearly the right choice. Since it gives the value of the power consumption for a specific algorithm, we could use it follows on the robot:

- Find $P_{Software}$ runs on the system
- Make a Look Up table on which each algorithm is cleared with their Consumption value.
- The robot should then calculate at specific time the total $P_{Software}$ given the task and the algorithms required for this task.

By this way the system has an estimated prediction for the tasks which it's about to do. this will help in the process of choosing the best (i.e. for best performance with least energy consumption) algorithms and resources along the mission.

5.2 Sensor Energy Model

In chapter 4, We presented some state-of-art for the energy model for sensors (or Nodes - more related to IOT field) the final derived Equation 4.3. The model composed into Active and Sleep components.

Neglecting sleep component is on the way to simplify the model since the data extracted from the data sheets ,see Table 4.2, shows that almost the sleeping mode (Idle state) is close to zero. However this neglection is simplifying our model, it has some drawbacks over the long term mission which could last for days and month.

For example, the use of IMU only costs 1.4 mW in Idle state. This amount of power is not significant for short task. But if the robotic system is required to perform a mission for some days then the amount used (even in the idle state) is recognizable and affect power supplier with a significant amount of energy (i.e 121 Joules for the IMU Idle state). So,The sleeping term should be considered if the mission is very long. However, This amounnt of energy is still very small compared to the other components (Mechanical and Software) and our goal is to have a generic model which could estimate the energy for the system. so, in our model we will negelct the P_{Sleep}.

For P_{Active} in Equation 4.5, almost every term depends on frequency of the micro-controller f_{MCU}. for example , the consumed energy E_{Tr} by the transmit mode is expressed as:

$$E_{Tr} = ((P_{ON} * f_{MCU}) + P_{Tr}) * T_{Tr}$$

where P_{Tr} is the dissipated power by the transmit mode and T_{Tr} is its time duration.

Here, we considered that the f_{MCU} is constant for all the threads, algorithms and sensor data processing and equals 10 Hz. So, the Active terms are almost constant which appears in the results of some sensors , see Figure 4.11 IMU test, the power fluctuation dose not exceed 5% . Since there are a lot of terms (like the time for each stage) are hard to detect for every sensor and every system. so, we only considered that the

$$P_{Sensor} = P_{Active}$$

Where P_{Active} is for most of the sensor (except USBL and DVL) and The system will use Figure 4.12 to have the Power values for each sensor and based on the mission and the task, it can define whether this sensor is used at this period or not, using the next formula Equation 4.8.

The Constant power values is obtained using some tests, mentioned in chapter 4, the main reliable test is the one using the PID-Package to accurately measure the values from the Power Supply "Carte d'Alimentation" which supplies the energy to the board and the sensor. The reasons for this reliability are:

- Result Record:

  Using this package gives the user the ability to save results data for analysis, review sensors' energy stability and monitoring these data later.

- Measurements:

  Since the power supply is giving the values from an embedded wattmeters, The values is more accurate that the traditional measurements tools which also depends on the human skill to find the average value.

- Resolution:

  The package has the ability to tune the measurement's *SamplingRate* which makes the Aliasing avoidable.

Concerning the USBL and DVL, They are dependent on 2 frequencies, Frequency of the embedded card as the case with the other sensors in addition to the Frequency of operation , Sending or receiving information. In case of the USBL, It as multiple operation modes and each mode have different amount of energy consumption. So, each mode should be related with the operation frequency with an equation. So, these sensors are especially needed to be tested online to take the average of power consumed from each protocol at different recruitment frequencies. Also, the time required to finish each protocol, Meanwhile the online test is not available.

The USBL mentioned value is based on the current configuration on the system, it's recommended that The Response Time (10 ms, 100Hz) can be

used to be the average of transmitting operation for either PING and ECHO protocols at frequency 50Hz (20 ms to send another Request). With this configuration the power from the USBL will be 5W.

And for the DVL the mentioned value is with Frequency of operation 1 MHz , as the default value of the sensor.

5.3 Model Validation

The obtained model are not valid to be used right away yet. Theses model should go for validation and experimentation step to ensure these model is actually describing the consumption of the system.

A step toward the validation, we have implemented a thread running a long besides the system's algorithms **Online** (i.e. During the mission). the main functionality of the thread is described as following:

- Measure and record time stamped power related components (i.e. Current and Voltage) from the suppliers (Power Switches and Power supply).
- Calculate the accumulative energy from the beginning of the mission.
- Then save these records and calculation after the mission.

Using this thread enable us to :

- Keep tracking the main energy consuming sources (Mechanical, Software and Sensors).
- Track the energy along the mission with the different tasks
- Validate the energy model: Using the predictive model estimation and the resultant Energy recorded from online mission, we could modify and tune the global model.

In our case, as shown in Figure 2.1, the Mechanical part is supplied by power switches. The Sotware and Sensors are supplied by the power supply "Carte d'Alimentation".

A sample of the recorded data:

Note: The values of embedded card "Carte d'Alimentation" is not recorded as the implementation of its hardware package has not been ready yet.

elapsed time	pws1_I	pws1_V	pws2_I	pws2_V	card_I	card_I	totalEnergy
0.01	0	0	0	0	0	0	0
0.11	0	14136	59	14045	0	0	166267.90625
0.21	0	14167	59	14045	0	0	332718.703125
0.31	0	14136	0	14045	0	0	332718.703125
0.41	0	14167	59	14045	0	0	499169.5
0.511	0	14136	59	14045	0	0	665437.40625
0.611	0	14136	59	14045	0	0	831705.3125
0.711	0	14136	59	14045	0	0	997973.21875
0.811	0	14167	59	14045	0	0	1164424.015625
0.911	0	14136	0	14045	0	0	1164424.015625
1.011	0	14167	59	14045	0	0	1330874.8125

TABLE 5.1: Energy Thread Recorded Data

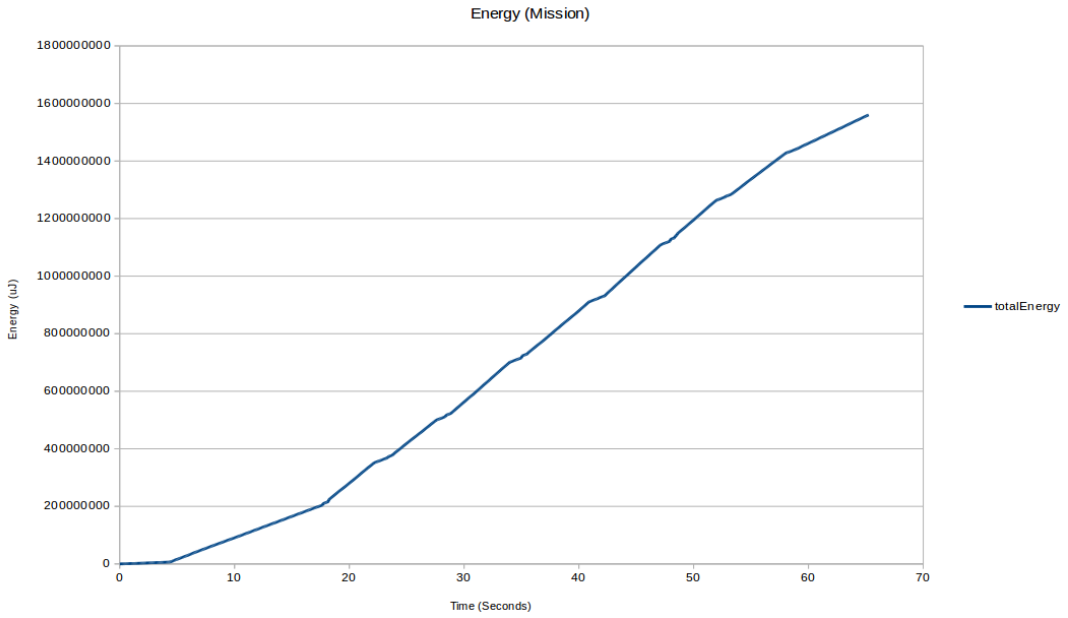


FIGURE 5.3: Energy calculated and recorded during a mission

Figure 5.3 shows the accumulative energy data calculated and recorded during a mission. This data is missing the embedded card energy Since these data is not available in the current version of the thread. but It will be very easy to integrate it once its hardware package is available.

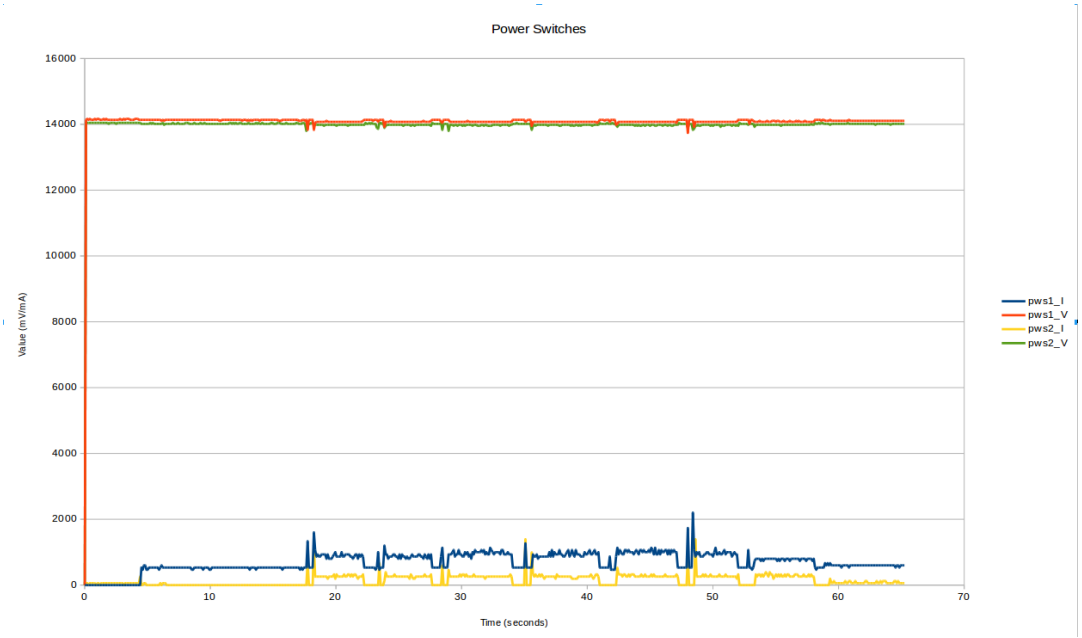


FIGURE 5.4: Power Switches recorded data during a mission

In Figure 5.4, The data measured from the power switches (Responsible for Mechanical part of the global model). During the mission, The robot is performing some basic moves (Forward, Backward , Up, etc.). As shown the resultant the data is monitored and at every operation we could recognize the variation of the current and voltages.

Finally ,This Thread is a main component for validation step and also for further experiments.

5.4 Models Importance

Looking to the numbers of how much energy could consumes are. from the first sight these numbers are very small especially compared to motion part consumption. For instance, Mayotte experiment where the propulsive power (i.e. power from motion part (Motors)) measurements is done. Figure 5.5 shows the difference between propulsive power and Hotel power (i.e. Software and sensors power which are considered constant for simplification).

As shown, Hotel Power is very small compared to the propulsive power. But considering the time, this can make a lot of difference. Figure 5.6 shows the percentage of Hotel Power to the total power along the mission which is in average is 43%. which is a great percentage which robot should not ignore or neglect during its resource allocation management.

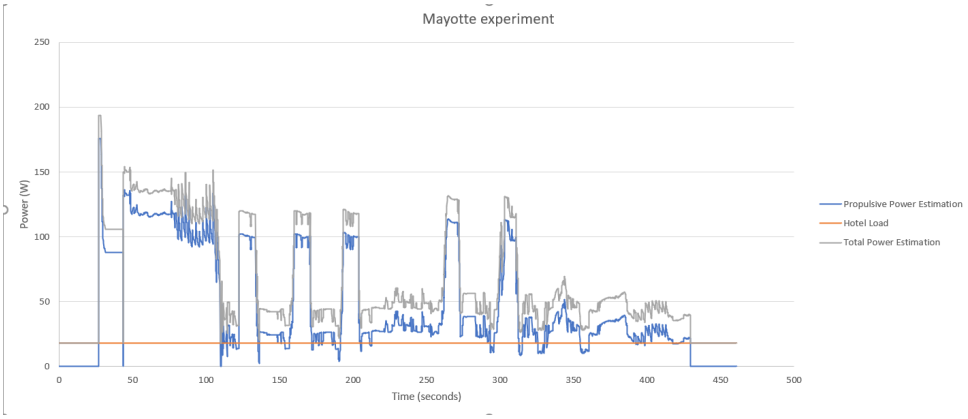


FIGURE 5.5: Mayotte Experiment : showing the difference between propulsive power and Hotel power

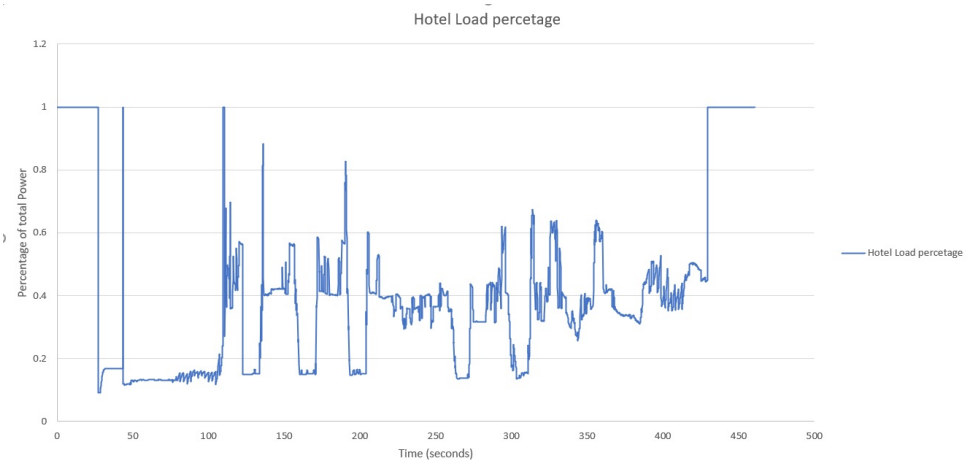


FIGURE 5.6: Percentage of Hotel power to the total power along the mission

5.5 Conclusion

In this chapter, we presented the analysis of the presented models. Also, we answered some critical questions like accuracy about the methodologies and tests used along this research. In addition, we give a hint about the importance of these models and how the robot will use it. Also, we gave a step towards models validation by creating a running thread which help the robot to track the energy consumption during the mission.

Chapter 6

Conclusion

This research addressed the problem of the global energy consumption predictive model in the context of energy aware robotic systems. The model is used as part of PANORAMA, a global control and resources management scheme.

Based on experimental tests and analytic process on AUVs, we proposed a predictive generic models suited for both Software and Sensors. In addition to the mechanical and motion consumption model, the global energy model is able to predict the range (i.e. time to finish the mission, task or a process) for its patrolling and exploring missions which expected to be long mission. Also, An Energy Thread is presented for future model's tuning and validation.

Based on the performance axes which determine the limitation for a success mission, the system will be able to successfully manage its resources (Algorithms, Sensor and Motion) to be bounded by these limitation to guarantee Security, Duration, Energy, Localization and Stability.

The future work will be focused on the experimental and analytical identification the gap between the predictive model and the real system energy consumption. Then, An energy management methodology for autonomous robotic mission should be developed to give the ability the system the ability to accurately allocate the both dynamic hardware and software resources along the mission with regard to the other performance constraints.

Applying PANORAMA methodology on AUVs will require some steps. Firstly, Integrating the Software and Sensor energy model with Motion model to construct the global model for the AUVs. Then, validation o the global model experimentally during a mission given both the prior estimated energy consumption from the global model and real time tracked consumption from the energy thread. Finally, Using PANORAMA scheme to have a robust control and resources allocation management to give the robot the ability to finish a autonomous mission successfully considering the all the performances axes and constraints.

Appendix A

AUV

A.1 Control and Communication

We define PC as a base station, from this point the robot can be controlled by humans operators. On the other hand the robot has an embedded computer (UPBoard) which we will connect. First, Robot needs to be connected to PC via Ethernet. Second step, PC should have a bridge in order to establish communication to the robot. The parameters that we consider are :

- ip = `192.168.1.44`,
- mask = `255.255.255.0`
- gateway = `192.168.1.1`

after this configuration we could connect it from a terminal :

```
ssh remi@192.168.1.253
```

if everything goes well we just enter the password : `remi` and we will be in the embedded computer : `/home/remi`.

A.1.1 Internet configuration

Normally the embedded computer of the robot that we use has not available internet. To have access a configuration it should be done in both sides, PC and embedded computer. We just need type a couple of commands to activated some flags that allows to share our PC internet connection to the robot computer.

A.1.2 PC–Base station

This command allows to know if our PC is in share mode:

```
$#  cat /proc/sys/net/ipv4/ip\_forward
```

it will print 0 or 1, it should be to 1, if don't it should try:

```
$#   sudo sysctl -w net.ipv4.ip\_forward=1
```

with `ifconfig` command we can know network information as `wlo1` and `enp1s0` which are the ids that we will use for share mode configuration.

Then we should execute these linux commands :

```
$#   sudo iptables -t nat -A POSTROUTING -o enp1s0 -j MASQUERADE
$#   sudo iptables -A FORWARD -i wlo1 -j ACCEPT
$#   sudo iptables -t nat -A POSTROUTING -o wlo1 -j MASQUERADE
$#   sudo iptables -A FORWARD -i enp1s0 -j ACCEPT
```

A.1.3 Embedded Computer

On the other hand, the robot should have also a configuration. One time PC–Robot are connected via Ethernet configuration is going to be use `ssh` protocol. For instance our PC has an

ip : `192.168.1.44`.

Then we should verify the presence of `auto enp1s0` (interface name) in `/etc/network/interfaces`.

This is a fragment of configuration :

```
auto enp1s0
iface enp1s0 inet static
address 192.168.1.33
netmask 255.255.255.0
gateway 192.168.1.44
nameservers 8.8.8.8 8.8.4.4
```

We should also modify or verify the name of DNS in : `/etc/systemd/resolved.conf` and change the line : `DNS=IPNAMESERVER` (this is Google ip : 8.8.8.8). Finally for testing our configuration interface (IP/GATEWAY) : `ifup -a` so this is going to check the configuration file : `/etc/network/interfaces`.

After configuration, we need to be sure that physical connection are well done as Figure **??**. Fathom is an interface with long distance over two points.

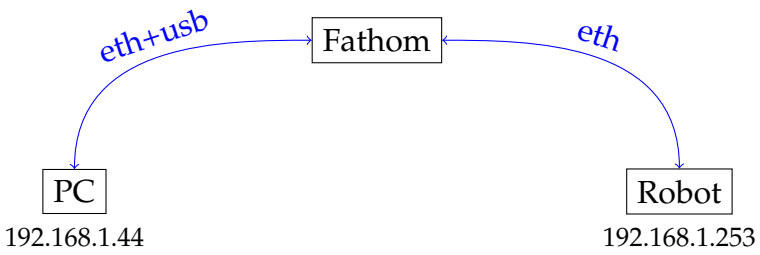


FIGURE A.1: Elements of PC–Robot connection.

PC and Fathom are connected via Ethernet and USB. On the other hand, Fathom is connected to Robot via Ethernet only.

A.2 Uploading Program to Underwater Robot

There are different ways to transfer file to the robot but because of PIDzation one way it is possible for the moment. The project was thought to follow this manner: all modification about code and compilation it should be done in the base station computer after that, if not issues appears, it should be sent it to the robot via `ssh`.

The right steps are explained in the following sequences :

- Modification PC base station and compilation,
- Transfer files using

```
$#  rsync -razv file destination
```

- In embedded computer at

  `pid-workspace/pid` folder, type

```
$#  make resolve package=aurov\_pid version=0.1.0
```

  this command will link all symbols in the uploaded file.

NOTE: When using `scp` problems with symbols (they are symbols where the code were compiled, from PC) will be send to the robot. Using `scp` and running from the robot it will be problems because the code in the robot will try to search symbols from the PC compiled so base station PC. A solution it is to use

`rsync` with different option as parameters.

Bibliography

[Bou+18] Taoufik Bouguera et al. "Energy consumption model for sensor nodes based on LoRa and LoRaWAN". In: *Sensors (Switzerland)* 18.7 (2018), pp. 1–23. ISSN: 14248220. DOI: 10.3390/s18072104.

[Bra+01] Albert M. Bradley et al. "Power systems for autonomous underwater vehicles". In: *IEEE Journal of Oceanic Engineering* 26.4 (2001), pp. 526–538. ISSN: 03649059. DOI: 10.1109/48.972089.

[CSB98] Anantha P. Chandrakasan, Samuel Sheng, and Robert W. Brodersen. "Low-Power CMOS Digital Design". In: *Low-Power CMOS Design* 27.4 (1998), pp. 36–46. DOI: 10.1109/9780470545058.part1.

[DF05] Falko Dressler and Gerhard Fuchs. "Energy-aware Operation and Task Allocation of Autonomous Robots". In: (2005), pp. 163–168.

[IL12] Carlos C. Insaurralde and David M. Lane. "Autonomy-assessment criteria for underwater vehicles". In: *2012 IEEE/OES Autonomous Underwater Vehicles, AUV 2012* (2012). DOI: 10.1109/AUV.2012.6380746.

[Jai+16] Lotfi Jaiem et al. "A step toward mobile robots autonomy: Energy estimation models". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9716 (2016), pp. 177–188. ISSN: 16113349. DOI: 10.1007/978-3-319-40379-3_18.

[Lan+15] David M. Lane et al. "PANDORA - Persistent autonomy through learning, adaptation, observation and replanning". In: *IFAC-PapersOnLine* 28.2 (2015), pp. 238–243. ISSN: 24058963. DOI: 10.1016/j.ifacol.2015.06.039.

[LLC19] P. Lambert, L. Lapierre, and D. Crestani. "An approach for fault tolerant and performance guarantee autonomous robotic mission". In: *Proceedings - 2019 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2019* (2019), pp. 87–94. DOI: 10.1109/AHS.2019.00009.

[Mar12] Marilyn Wolf. *Computer s as Components*. 2012, p. 528. ISBN: 978-0123884367.

[Mei+06] Yongguo Mei et al. "Deployment of mobile robots with energy and timing constraints". In: *IEEE Transactions on Robotics* 22.3 (2006), pp. 507–522. ISSN: 15523098. DOI: 10.1109/TRO.2006.875494.

[Nee04] A. Neely. "Business Performance Measurement: Theory and Practice". In: *Cambridge University Press* (2004).

[Nou+12] Adel Noureddine et al. "A preliminary study of the impact of software engineering on GreenIT". In: *2012 1st International Workshop on Green and Sustainable Software, GREENS 2012 - Proceedings* (2012), pp. 21–27. DOI: 10.1109/GREENS.2012.6224251.

[Par+15] Ramviyas Parasuraman et al. "Model based on-line energy prediction system for semi-autonomous mobile robots". In: *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS* 2015-September (2015), pp. 411–416. ISSN: 21660670. DOI: 10.1109/ISMS.2014.76.

[Set11] Mae L. Seto. "On-line learning with evolutionary algorithms towards adaptation of underwater vehicle missions to dynamic ocean environments". In: *Proceedings - 10th International Conference on Machine Learning and Applications, ICMLA 2011* 1 (2011), pp. 235–240. DOI: 10.1109/ICMLA.2011.110.

[ŠH11] Ondřej Špinka and Zdeněk Hanzálek. "Energy-aware navigation and guidance algorithms for unmanned aerial vehicles". In: *Proceedings - 1st International Workshop on Cyber-Physical Systems, Networks, and Applications, CPSNA 2011, Workshop Held During RTCSA 2011* 2 (2011), pp. 83–88. DOI: 10.1109/RTCSA.2011.22.

[SJU12] Amir Sadrpour, J. Jin, and A. G. Ulsoy. "Mission energy prediction for unmanned ground vehicles". In: *Proceedings - IEEE International Conference on Robotics and Automation* (2012), pp. 2229–2234. ISSN: 10504729. DOI: 10.1109/ICRA.2012.6224860.

[SMM07] Chiyoung Seo, Sam Malek, and Nenad Medvidovic. "An Energy Consumption Framework for Distributed Java-Based Systems". In: *Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering*. ASE '07. Atlanta, Georgia, USA: Association for Computing Machinery, 2007, pp. 421–424. ISBN: 9781595938824. DOI: 10.1145/1321631.1321699. URL: https://doi.org/10.1145/1321631.1321699.

[Ter+09] Guillaume Terrasson et al. "Energy Model for the Design of Ultra-Low Power Nodes for Wireless Sensor Networks". In: *Procedia Chemistry* 1.1 (2009), pp. 1195–1198. ISSN: 18766196. DOI: 10.1016/j.proche.2009.07.298. URL: http://dx.doi.org/10.1016/j.proche.2009.07.298.

[Tiw+19] Kshitij Tiwari et al. "A unified framework for operational range estimation of mobile robots operating on a single discharge to avoid complete immobilization". In: *Mechatronics* 57.March (2019), pp. 173–187. ISSN: 09574158. DOI: 10.1016/j.mechatronics.2018.12.006. URL: https://doi.org/10.1016/j.mechatronics.2018.12.006.

[Wad+04] Robert Wade et al. "AUTONOMY MEASURES FOR ROBOTS". In: (2004), pp. 1–7.

[Yoe+07] Dana R. Yoerger et al. "Remotely operated vehicle technology exploration, and sampling". In: *Oceanography* 20.1 (2007).