Ramsay Rafaël Mac Donald

Instrumentation platform and Maximum Power Point Tracking control for a Hydrokinetic turbine

Brasilia-DF

2017

Ramsay Rafaël Mac Donald

Instrumentation platform and Maximum Power Point Tracking control for a Hydrokinetic turbine

Universidade de Brasília – UnB

Faculdade de Tecnologia

Programa de Pós-Graduação

Supervisor: Dr. Rudi Henri van Els

Brasilia-DF
2017

Ramsay Rafaël Mac Donald

Instrumentation platform and Maximum Power Point Tracking control for a Hydrokinetic turbine/ Ramsay Rafaël Mac Donald. – Brasilia-DF, 2017-

134 p. : il. (algumas color.) ; 30 cm.

Supervisor: Dr. Rudi Henri van Els

Dissertation (Masters) – Universidade de Brasília – UnB
Faculdade de Tecnologia
Programa de Pós-Graduação, 2017.

1. Hydrokinetic turbine. 2. Instrumentation. 2. Maximum Power Point Tracking. I. R. Els. II. Universidade de Brasília. III. Faculdade de Tecnologia. IV. Instrumentation platform and Maximum Power Point Tracking control for Hydrokinetic turbine.

Ramsay Rafaël Mac Donald

Instrumentation platform and Maximum Power Point Tracking control for a Hydrokinetic turbine

Trabalho aprovado. Brasilia-DF, 4 de Abril, 2017:

Dr. Rudi Henri van Els
Orientador

Dr. Daniel Mauricio Muñoz Arboleda
Convidado 1

Dr. Antônio Cesár Pinho Brasil Junior
Convidado 2

Brasilia-DF
2017

Acknowledgements

To my parents, my brother, and my family for their love and support.

To my colleagues and friends at the laboratory: Aramis, Eugenia, Francis, Francisco, Giselle, Miguel, Nela, Paulo, Rafael, and Vinicius for their moral and academic support, and for the pleasant working environment they have created.

To my friends in Brazil, for sharing their families and culture with me, be it Brazilian, Colombian, or Cuban.

To my supervisor Dr. Rudi van Els, who took the initiative to create cooperation between the Surinamese University and the University of Brasilia, and dragged me to Brazil during my undergrad and has kept guiding me through my master's. May you keep guiding more Surinamese students into higher education.

To Dr. Antônio Brasil Junior for the guidance, patience, and the opportunity to participate in the hydrokinetic turbine development project Hydro-K.

To the SuriBraz Academic Network, for the guidance and support.

To AES who has supported this work.

Abstract

A project named Hydro-K is aimed at developing a hydrokinetic turbine. The prototype or preliminary model of this turbine will inherently present uncertainties. Eliminating them requires testing and data gathering. The data is acquired using an instrumentation platform designed in this dissertation. Another challenge will be controlling the turbine. Controlling traditional horizontal-axis-variable-speed-fixed-pitch hydrokinetic turbines requires mapping its efficiency at various water and rotor speeds, i.e., the C_P curve of the turbine, in advance. Whereafter, its rotational velocity is measured and manipulated in order to maintain maximum efficiency of the rotor. Recently, Maximum Power Point Tracking (MPPT) control has been applied to wind turbines. This control method eliminates the necessity of C_P curve mapping by measuring the generator's voltage and current output, and periodically adjusting the electrical load on the generator to maintain the maximum electrical power output. Considering the similarities between hydrokinetic and wind turbines, it is likely that MPPT can be effectively applied for the hydrokinetic variant. Therefore, a second objective is assessing the effectiveness of MPPT for hydrokinetic turbines by comparing a conventional control strategy with MPPT using simulations in "SIMULINK". The electrical load on the generator is varied by using Pulse Width Modulation (PWM). The Hydro-K turbine developed at the University of Brasilia (UnB) is equipped with a three-phase-permanent-magnet-synchronous- generator (PMSG). The results of the simulations are presented by comparing the two aforementioned control methods under different water velocity conditions, and show that MPPT control delivers a superior electrical output.

Keywords: hydrokinetic turbine. instrumentation. maximum power point tracking.

Resumo

O projeto Hydro-K tem como objetivo o desenvolvimento de uma turbina hidrocinética. O protótipo ou o modelo preliminar desta certamente apresenta incertezas e para eliminá-las são necessários testes e coleta de dados. Essa coleta foi feita através de uma plataforma instrumental desenvolvida neste trabalho. Um outro desafio foi o controle da turbina. O controle tradicional das turbinas hidrocinéticas com um eixo horizontal, velocidade variável e um angulo da pá fixo, requer o mapeamento da curva C_P da turbina com antecedência. Em seguida, a sua velocidade de rotação é medida e manipulada de modo a manter a máxima eficiência do rotor. Recentemente, o controle do *Maximum Power Point Tracking* (MPPT) foi aplicado às turbinas eólicas. Esse método de controle elimina a necessidade de mapeamento de curva C_P medindo a tensão e a saída de corrente do gerador e ajustando periodicamente a carga elétrica no gerador para manter a potência máxima de saída elétrica. Considerando as semelhanças entre as turbinas hidrocinéticas e as eólicas, é provável que o MPPT possa ser aplicado para os hidrogeradores. Um segundo objetivo desta dissertação é avaliar a eficácia do MPPT para turbinas hidrocinéticas comparando uma estratégia de controle convencional com o MPPT usando simulações no "SIMULINK". A carga elétrica no gerador é variada usando o *Pulse Width Modulation* (PWM). A turbina hidrocinética do Hydro-K desenvolvida na Universidade de Brasília (UnB) é equipada com um gerador síncrono-trifásico com ímã permanente. Os resultados das simulações e experiências são apresentados comparando os dois métodos de controle acima mencionados sob diferentes condições de velocidade da água, e mostram que o controle MPPT fornece uma saída elétrica superior.

Palavras-chave: turbina hidrocinética. instrumentação. maximum power point tracking.

List of Figures

List of Tables

List of abbreviations and acronyms

FSM	Finite State Machine
G1	Generation 1 Turbine
G2	Generation 2 Turbine
G3	Generation 3 Turbine
HK-10	10 kW Hydro-K turbine
HK^3-10	Arrangement of three 10 kW Hydro-K turbines
HKT	Hydrokinetic Turbine
HPP	Hydropower Plant
IDE	Integrated Development Environment
IoT	Internet of things
MPP	Maximum Power Point
MPPT	Maximum Power Point Tracking
ORC	Optimal Regimes Characteristic
PCB	Printed Circuit Board
PMSG	Permanent Magnet Synchronous Generator
PV	Photovoltaic
QFD	Quality Function Deployment
RTC	Real-Time Clock
SCADA	Supervisory Control and Data Acquisition
SMC	Sliding Mode Control
TTL	Transistor Transistor Logic

List of symbols

α	Angular Acceleration	rad/s^2
λ	Tip-speed Ratio	dimensionless unit
ρ	Density	kg/m^3
τ	Torque	N·m
ω	Rotor Angular Velocity	rad/s
A	Rotor swept Area	m^2
C_P	Power Coefficient	dimensionless unit
i	Current	A
I	Inertia	kg·m^2
P	Power	W
R	Resistance	Ω
r	Rotor radius	m
t	Time	s
U	Water velocity	m/s
V	Voltage	V

Contents

1 Introduction

Renewable energy is defined by Twidell and Weir (2015, p. 3) as: "energy obtained from naturally repetitive and persistent flows of energy occurring in the local environment". The usage of renewable energy as opposed to fossil fuels, such as coal, natural gases, and oil, has become more significant between 2010, when 7.83% of the primary energy consumption was renewable energy, and 2015 when it was 9.57% (World Energy Council, 2016) (See figure 1 which was adapted from (World Energy Council, 2016)). This is due to the growing world population, the development of human society and modern technologies, the depletion of fossil fuel sources, and also the greenhouse gas emissions emitted by these fossil fuels. Each square meter of habitable surface on Earth has an estimated average renewable energy potential of 500 W. This includes solar, wind, and other renewable energy forms in an overall estimate (TWIDELL; WEIR, 2015). The technological challenge with this type of energy is harnessing the available power.

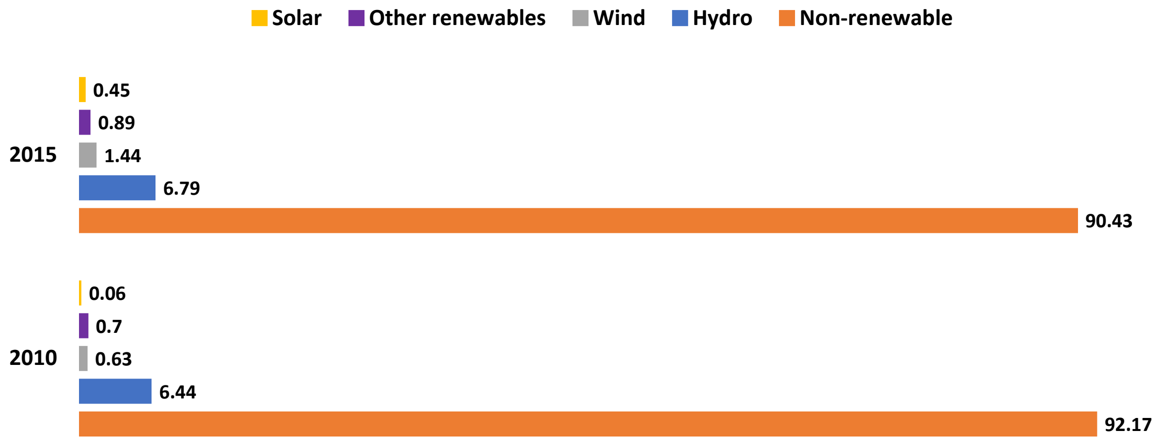


Figure 1 – Comparative primary energy consumption with renewables accounting for 7.83% in 2010, and 9.57% of the total energy consumption in 2015.

Hydro and wind energy, are the leading renewable energy sources. According to the World Energy Council (2016) hydropower supplies 71% of all renewable electricity (1064 GW capacity by the end of 2015), which amounts to 16.4% of the world's electricity from all sources. In Brazil, the installed hydropower capacity reached 91.7 GW with a production of 382 TWh in 2015, which accounted for 64.0% of the total domestic electricity supply (Brazilian Energy Research Company (EPE), 2016). The estimated global growth of hydropower capacity is around 29% by the year 2020. And the global wind energy reached around 435 GW capacity by the end of 2015. Wind and hydro power show great potential in competing with fossil energy sources (World Energy Council, 2016).

After decades of research and development, solar and wind energy have already been deployed all over the world. Marine and hydro-kinetic energy, such as wave energy, tidal current, open-ocean current, river current, and thermal gradients in the ocean, are less common (THRESHER, 2011) and have a combined energy capacity of around 0.5 GW of which 99% is produced by tidal current energy (Ocean Energy Systems, 2015). This energy is harnessed using hydrokinetic turbines (HKT) which convert the kinetic energy of free flowing water into electric energy. Information regarding estimations of riverflow energy potential for hydrokinetic turbines is scarce. Nevertheless, flowing rivers are an energy source for hydrokinetic energy converters.

Hydrokinetic turbines are hydrokinetic energy converters that harnesses energy from a free flowing current, i.e., no dams are needed. The efficiency of HKT's are greatly impacted by two parts in particular: the rotor and the generator. The rotor converts the kinetic energy of the flowing water into a rotational torque that drives the generator, which in turn, produces electricity. The problem with this system is that the point of optimal efficiency is dependent on the fluctuating river velocity and the varying electrical load applied to the generator. Therefore, controlling a HKT in a wide range of velocity and load variation is a challenge (YUEN; APELFRÖJD; LEIJON, 2013).

A Brazilian project named Hydro-K started designing a fixed-pitch variant of such a river flow HKT in cooperation with the University of Brasilia (UnB) in 2015. The research presented in this dissertation is part of this project. The author has contributed to this project in the form of instrumentation and a control strategy for the HKT.

There are several control strategies for HKT's. Most strategies involve rudimentary control principles such as: load management, which regulates the load on a generator instead of the output of a generator; regulating the electromagnetic torque of a generator to manipulate the generator output; adjusting the input torque of the generator mechanically, by for example changing the rotor-pitch angles or using a disc brake; and less common, controlling the kinetic-energy-carrying-fluid flow in order to adjust the energy available for harnessing.

A moderately researched control strategy for HKT's is the "maximum power point tracking" (MPPT) control. As the name suggests, it strives to find the maximum power, and maintain the operating point at maximum power. This type of control has been more amply researched and implemented in wind turbines. An interesting property of this strategy is that the generator voltage and current measurements are sufficient for estimating the generator's operating point, while other strategies require some parameters to be mapped. The fact that the generator's power output is used as a reference also means that the electrical output should be at it's peak, because the peak electrical power does not occur at the peak mechanical power (AUBRÉE et al., 2016)

The first objective of this study will be to develop a MPPT algorithm to control the Hydro-K HKT and thereby ensuring optimal energy harnessing from the riverflow. The inputs for this algorithm should exclusively be the current and voltage generated by the generator, with the goal of maintaining the maximum generator output at various water velocities.

Keeping in mind that the turbine is experimental, there are a number of parameters that will be monitored for further research and development of the hydrokinetic turbine. Monitoring these parameters remotely poses many challenges. An even greater challenge is creating a platform capable of monitoring similar or future projects. Therefore, the second objective of this dissertation is the design of an instrumentation platform with flexibility at heart. And thereby enabling the use of this platform for various types of turbines.

The methodology for assessing the functionality of the MPPT algorithm will be to firstly do a literature review to identify MPPT algorithms and to identify the problems with these existing algorithms. the HKT will at first be modeled and then the control algorithm can be simulated to study the system's behavior at varying river velocities. Next, an algorithm will be created with a solution of the previously identified problems. To test the created algorithm, a model of the hydrokinetic turbine with all of its components will be generated in "SIMULINK". Finally, the algorithm will be implemented in this model and compared with the different algorithms found in literature.

The methodology on which the design of the instrumentation platform is based is the "product development process" developed by ROZENFELD et al., which includes four main development phases. The first phase is the informational design phase, here the designer attempts to acquire the customer's needs. And through a process and several tools, the designer can formulate product specifications. The next step is the conceptual design phase, which is focused on the use of the product specifications to generate alternative solutions and define the architecture of the product. The following step, the detailed design, is a more technical phase, where the necessary calculations and drawings are made for the last phase, which is the production and testing of prototypes. In the latter phase, prototypes are made to improve the product by testing its operation for the identification of unforeseen problems.

The contributions of this work to the academic community will be the comparison of various MPPT algorithms and the availability of an instrumentation platform for various applications.

This dissertation is structured as follows: First off, the various hydrokinetic energy converter technologies will be presented in chapter 2 to clarify which type of HKT will be studied. Chapter 3 will discuss the history UnB has with hydrokinetic turbines and introduce the current Hydro-K project. Chapter 4 covers the design and fabrication of the instrumentation platform which will monitor the turbine's functionality. Next, chapter 5 will review the control of hydrokinetic turbines and MPPT in more detail. Chapter 6 will explain the simulation models. And finally, chapters 7 and 8 will discuss the results and conclusions.

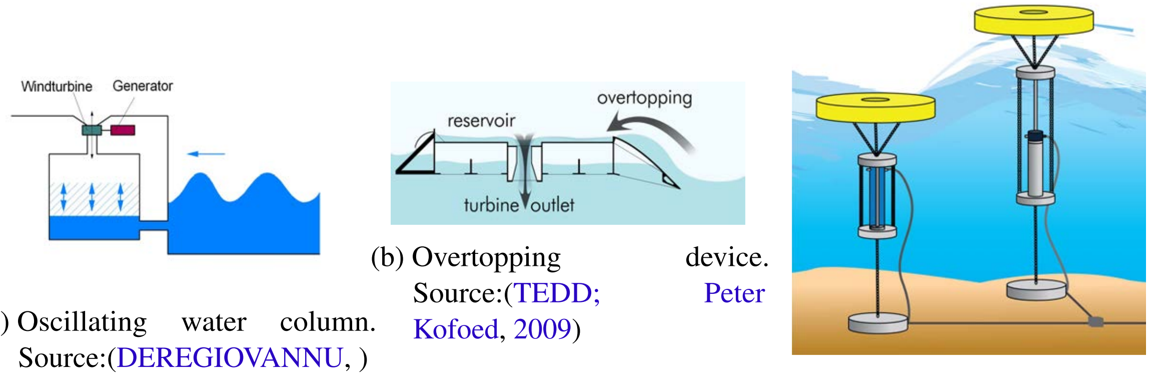
2 Hydrokinetic energy converters

2.1 Classification of hydrokinetic energy converters

Hydrokinetic energy converters harvest kinetic energy from flowing water and convert this energy directly into electricity without the need of elevated water reservoirs commonly known as dams. This is why they are also called free flow turbines, ultra-low- or zero-head-hydro turbines. Sources of hydrokinetic energy range from natural flowing rivers, tidal estuaries, ocean currents, and waves, to various man-made waterways.

Hydrokinetic energy technology has been developing in the last 2 decades and is one of the fastest growing renewable energy technology (Ocean Renewable Energy Coalition (OREC); Verdant Power, 2012). The majority of the hydrokinetic energy converter systems are in the research and development phase, and a small group is in the pre-commercial phase (KHAN; BHUYAN, 2009). There are more than 100 conceptual designs of hydrokinetic energy converter systems (COPPING et al., 2013)(EDENHOFER et al., 2011).

Hydrokinetic energy converters are classified in two main groups (YUCE; MURATOGLU, 2015). The first group is the Wave Energy Conversion (WEC) systems. As the name suggests, these systems utilize the movements and kinetic energy originating from waves. These systems vary in size, orientation, and distance from the shore, and can be divided into three main sub categories, namely, Oscillating Water Columns (OWC), Over Topping Devices (OTD), and Wave Activated Bodies (WAB). As WEC's are not within the scope of this work, they will not be discussed in detail, but figure 2 illustrates the main working principles. Figure 2a shows an OWC which rotates an air turbine using the suction and overpressure caused by waves. Figure 2b shows an OTD which drives a turbine with water entering a reservoir due to waves. And lastly, figure 2c shows a WAB, which hydraulically pumps water through a turbine, as a buoy moves with the waves.



(a) Oscillating water column. Source:(DEREGIOVANNU,)

(b) Overtopping device. Source:(TEDD; Peter Kofoed, 2009)

(c) Wave activated bodies. Source:(SIGMAHELLAS,)

Figure 2 – Various wave energy conversion systems.

The second group is the Current (in-stream) energy conversion system (CEC), or rotating energy conversion systems. Hydrokinetic turbines such as Tidal in- stream energy converters (TISEC), marine current turbines (MCT) and river energy conversion systems (RECS) belong to this category. The majority of these converters have a propeller with multiple blades rotating about an axis. The orientation of the axis can be either fully or principally aligned with the water flow (axial flow), or parallel to the water flow (cross flow). Axial flow turbines are horizontal axis turbines and inclined axis turbines. Cross flow turbines are either vertical axis turbines, or in-plane axis turbines. The first vertical axis turbine was designed by Darrieus in the 1920's. The Darrieus turbine has multiple vertically placed hydrofoil shaped blades with a top and a bottom support.

In-plane axis turbines are one of the earliest applied hydrokinetic energy converter systems. A well-known example of this of CEC is the water wheel. Water wheels consist of a large wheel, with various vanes, rotating around a horizontal axis. The vanes are placed on the wheel such that they are perpendicular to the water flow. Variations of axial flow and cross flow turbines can have helical rotors instead of propellers, they can be ducted, and they can have a rigid mooring or a buoyant mooring. Helical turbine variations, such as the Gorlov turbine, do not use hydrofoil blades, but instead use helical shaped blades. This enables the turbine to capture water flow in any direction and at low speeds. Another variation, such as a ducted turbine, has an efficiency that is not constrained to Betz limit, which will be explained shortly hereafter. Figure 3 shows the categorization for hydrokinetic turbines.

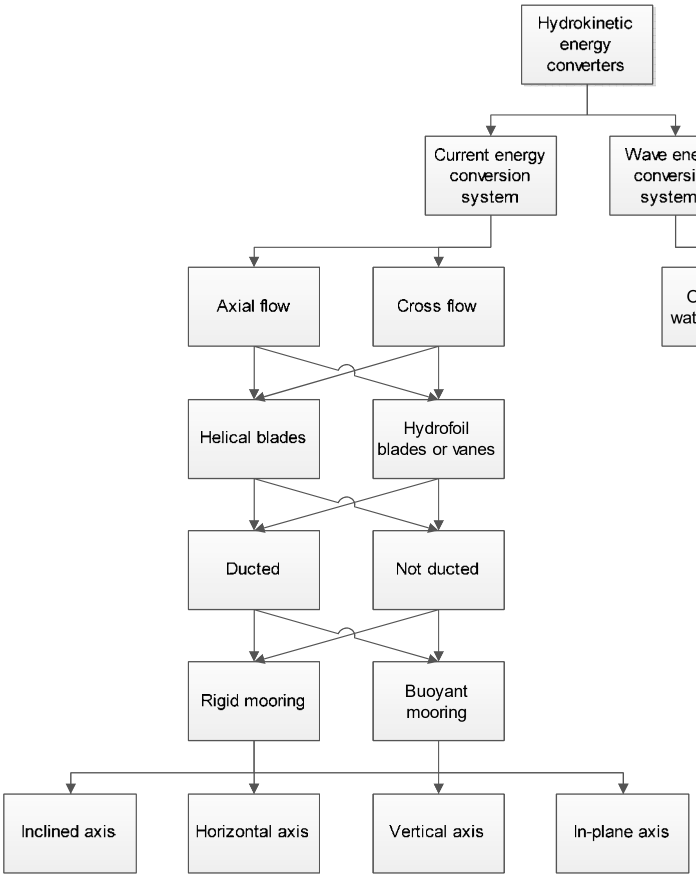


Figure 3 – Classification of hydrokinetic energy converters

Hydrokinetic turbines are mostly designed for a fixed rotor speed, but can also have a variable speed mechanism, or a variable rotor pitch mechanism. The power output of rotating current energy conversion systems is determined as follows:

$$P = \frac{1}{2}\rho A U_{\infty}^{3} C_P \tag{2.1}$$

With P as the total power output from the turbine in Watts, ρ as the density of the fluid in $\frac{kg}{m^3}$, A is the swept area of the rotor blades in m^2, U_∞ is the water velocity (m/s) and C_P is the power coefficient of the turbine.

The power coefficient, is a dimensionless ratio between the energy extracted from the fluid and the total energy of the fluid flowing into the turbine. This means that it is a measure of the overall efficiency of the turbine. For hydrokinetic energy converters without a duct or diffuser, there is a theoretical limit for the C_P, because the fluid has to keep moving after the turbine and hence keeps some of its initial kinetic energy. This theoretical limit is called Betz's coefficient of performance or Betz limit and is described as (BETZ, 1920):

$$C_{pB} = \frac{16}{27} = 0.593 \tag{2.2}$$

This means that unducted turbines have a maximum theoretical efficiency of 59.3%. The efficiency of hydrokinetic turbines can also be given in $\frac{C_P}{C_{pB}}$, which is the efficiency of the turbine relative to its maximum theoretical efficiency.

The power coefficient can be calculated as follows:

$$C_P = \frac{P}{0.5\rho A U_\infty^3} \tag{2.3}$$

Where P is the total power output from the turbine in Watts, ρ is the density of the fluid in $\frac{kg}{m^3}$, A is the swept area of the rotor blades in m^2, and U_∞ is the water velocity in $\frac{m}{s}$. Hydrokinetic turbines are designed to operate most efficiently at a specific blade tip speed and a specific water flow speed. The ratio between the blade tip speed and the water flow speed is represented by a dimensionless variable λ:

$$\lambda = \frac{\omega r}{U_\infty} \tag{2.4}$$

With ω as the rotor rotational speed $\left(\frac{rad}{s}\right)$, r as the radius of the rotor in m, and U_∞ is the water velocity in $\frac{m}{s}$. Mapping the C_P and at different tip speed ratios results in a C_P-λ curve (see figure 4, source:(CP-LAMBDA,)).

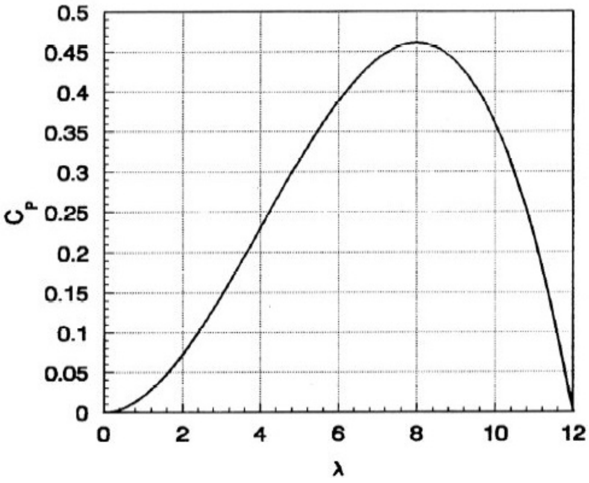


Figure 4 – Example of a C_P-λ curve.

With this curve it is possible to derive the power output at any water velocity and any rotor speed, which is in essence the mapping of a hydrokinetic turbine's operating conditions. This mapping can be used to control HKT's, and will be discussed in more detail in chapter 6.

2.2 Comparing hydrokinetic energy converters with hydro power

It is the author's experience that hydrokinetic energy converters are often overlooked as sources of electricity, and that one tends to opt for hydro power instead of HKT's. In the case of large scale power generation, hydro power has the clear advantage, but in small scale, hydrokinetic energy converters can be more convenient. And by combining both technologies, one can increase the efficiency of hydro power plants by harnessing the residual kinetic energy at the plant's outlet, which is exactly the case for the Hydro-K project. To advocate the consideration of using hydrokinetic energy converters, this section is dedicated to comparing the advantages and disadvantages of hydrokinetic energy converters.

First of all, hydrokinetic energy converters require very little civil work compared to dams, but only generate power on a small scale. Eventhough hydrokinetic energy converters can be installed in–wind farm like–multi-unit arrays, they have a lower efficiency of barely 35%, compared to the 80-90% efficiency of hydro power plants. The energy predictability of hydrokinetic turbines is similar to that of hydro power plants, and both hinder navigation and fishery in their installed areas, but hydrokinetic turbines have the advantage of having a larger range of applicability, for example, they can be installed in remote- off-grid areas, in areas with seismic hazards, and in highly populated areas. Above all, hydrokinetic turbines have much less environmental effects, and eventhough the electric energy cost of this technology is high, the development of the technology will increase its efficiency, and will most likely render it a major source of cost effective electricity by 2050 (MAGANA et al., 2014).

The lower environmental effects of hydrokinetic turbines include evading: the relocation of people, the inundation of agriculture, historical sites and animal habitats, the sedimentation of fertile lands, the production of methane due to submerged vegetation, and the alteration of river regimes. Although it's obvious environmental advantages, this technology can still affect the environment due to its parts, the chemical agents, and the noise and vibrations of the turbine can also affect the water habitat.

2.3 Hydrokinetic turbine control strategies

Tidal turbines are subject to forecasted tide conditions. River turbines on the other hand may not be dependent on tide conditions. River turbines downstream of Hydro Power Plants (HPP) are strongly dependent on the plant's output flow, be it through spillgates and/or turbine outlets. The prediction of tidal conditions is somewhat straightforward, as readily available tide tables are available and have shown a 98% accuracy for decades. These tables can be used in coordinating the tidal plant's operation (ZHOU et al., 2014). Depending on the configuration of the HPP, the resource for river turbines downstream of HPP's, can be less predictable due to fluctuations in power demand. However, the advantages for this HPP-river turbine configuration are that the majority of HPP's monitor the plant's output flow, and this flow tends to change more gradually. Using the plant's output flow data and the site's bathymetry, a correlation between the plant's output flow and the river velocity can be made. This correlation could then be used to coordinate the operation of the river turbine. The cases described above have more predictable resources, contrary to wind turbines.

Many of the hydrokinetic turbine control strategies have been priorly implemented in wind turbines. The main objectives for hydrokinetic turbine control are: maintaining the turbine within it's rated velocity limits, regulating the generator output to meet certain power quality standards, and maximizing the power harvested from the water flow (as long as the turbine is within it's rated velocity limits). As mentioned in the introduction, there are various ways to achieve turbine control. Figure 5 (Adapted from (MUNTEANU et al., 2008)) shows three main control subsystems that a HKT can have.

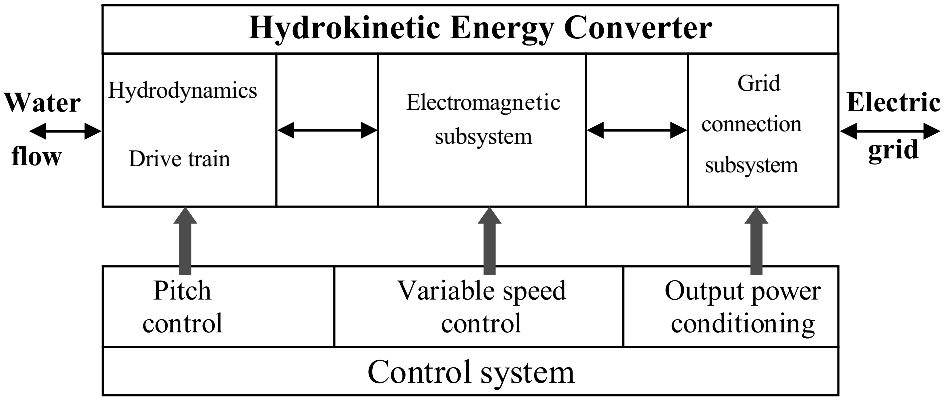

Hydrokinetic Energy Converter
Water flow
Hydrodynamics
Drive train
Electromagnetic subsystem
Grid connection subsystem
Electric grid
Pitch control
Variable speed control
Output power conditioning
Control system

Figure 5 – Main control systems of a hydrokinetic energy converter.

The first control subsystem controls the pitch angle of the rotor. Thereby changing the power extracted from the water flow and consequently the input torque and input velocity of the generator. In case of an over-flow situation, in which the water flow exceeds the machines rated flow, the pitch can be changed in four ways: active-pitch control, active-stall control, passive-pitch control, and passive-stall control. Actively varying the speed or the pitch of the turbine blades is a form of active control. Passive control on the other hand, can be achieved by designing a rotor that deforms appropriately to specific load conditions. The difference between pitch control and stall control is the direction of the angle change. Pitch control attempts to angle the leading edge of the blade towards the incoming flow. Stall control is also known as negative pitch control, and directs the leading edge of the blade away from the incoming flow. (MOTLEY; BARBER, 2014) examined the possibilities of passive pitch adaption for marine hydrokinetic turbines.

The second control subsystem manages the generator in order to obtain a variable speed regime. Traditional horizontal axis turbines fall into one of four categories: fixed speed - fixed pitch, variable speed - fixed pitch, fixed speed - variable pitch, or variable speed - variable pitch. For each water velocity, there is a certain rotational speed at which the power curve of a hydrokinetic turbine reaches a maximum. The curve described by connecting all the maxima of different water velocities is known as the Optimal Regimes Characteristic (ORC). The Variable speed control system aims to keep the generator velocity at the velocity corresponding to this maximum.

The last control subsystem, which is "output power conditioning", smoothens the transfer of the generator output electric power to the electric load. The smoothing can be achieved by adding capacitor banks or other power electronics devices. Adding these devices implies a supplementary power loss, but they favor generator motion control, which allows the positioning of the operation point on the ORC.

Considering the theme of this dissertation, chapter 5 is dedicated to discuss the optimal control approaches of variable speed - fixed pitch turbines.

3 History and technological development of hydrokinetic turbines at UnB

3.1 First generation hydro-kinetic turbine (G1)

The first recorded Brazilian experience with hydrokinetic turbines (HKT) was in 1981, when Harwood and Almeida tested a HKT in the amazonian "Solimões River" (HARWOOD, 1985). Research on HKT's started at UnB when a research group of the Mechanical Engineering Department tested various prototypes, and installed the first operational HKT in 1995 at a river located near the city Correntina in the state of Bahia. This HKT was named "Generation 1" and with it's 1 kW capacity, provided electricity to a nearby medical post. Generation 1 was tested with various axial-rotors with 2 blades, 6 blades, and 18 blades. The axial-6-bladed rotor with a diameter of 0.8 m proved to be the most efficient of the tested rotors (ELS et al., 2003). The rotor was protected from large and potentially harmful debris by a conical steel grid as shown in figure 6 (Brasil Jr. et al., 2007). Furthermore, the turbine had a stator which guided the entering water flow into the rotor. The HKT had a brushed AC generator with a manually variable stator excitation and an overvoltage protection system based on a 1 kW thyristor driven load (see figure 7).

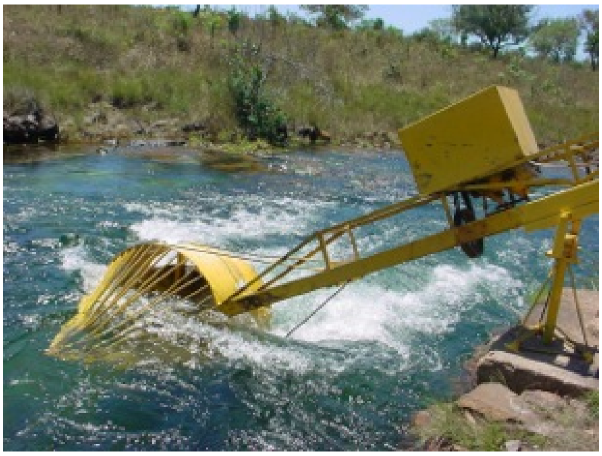


Figure 6 – Generation 1 turbine source: Els et al. (2003).

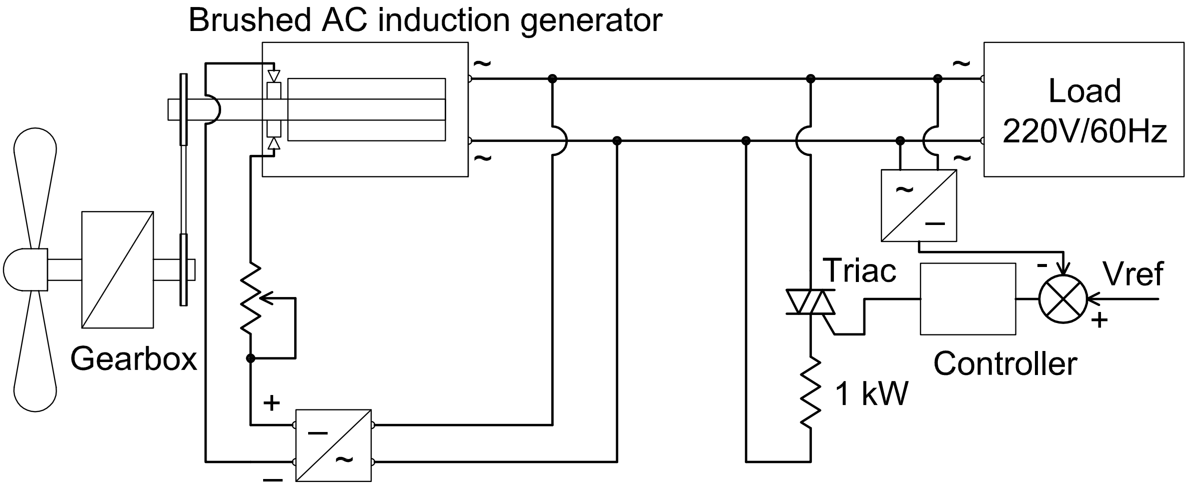


Figure 7 – Control schematic for the Generation 1 HKT.

An attempt was made to implement a controller that automatically varies the stator excitation using a second thyristor, thereby controlling the voltage generation. But unfortunately, this method was never implemented. The schematic of the of the proposed system is shown in figure8.

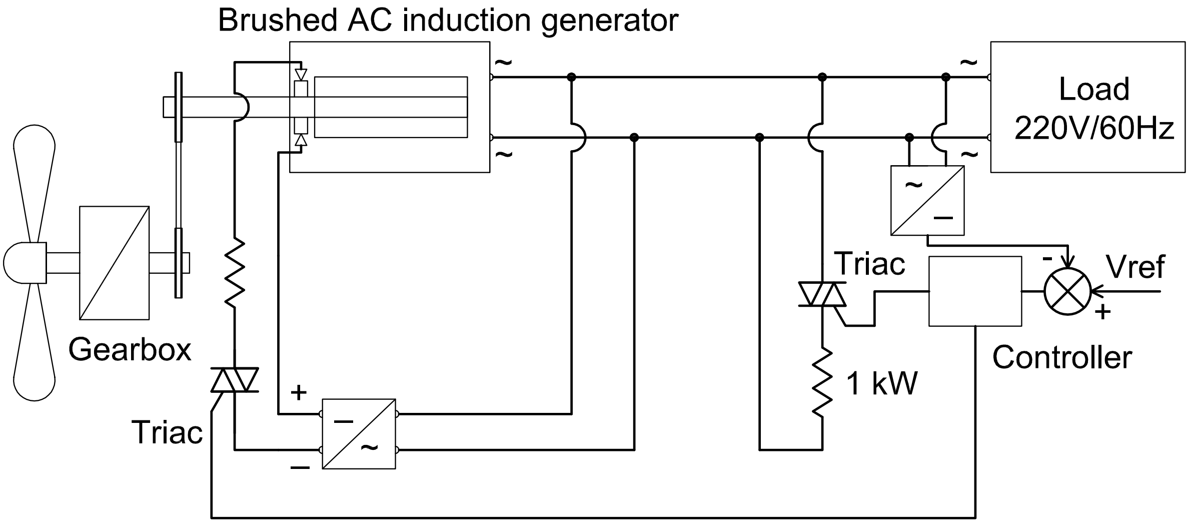


Figure 8 – Control schematic for the Generation 1 HKT. With an automatically controlled stator excitation.

3.2 Second generation hydro-kinetic turbine (G2)

In 2005, some improvements were made on Generation 1. One of the improvements was the addition of a diffuser, which is technology originating from wind turbines. The diffuser slows down the outlet flow by increasing the outlet diameter, this creates a suction at the turbine outlet. The lower outlet pressure induces a greater mass flow through the turbine and enables a turbine to surpass Betz limit. Another change was the enlargement of the stator vanes at the turbine inlet. This directed the flow directly towards the 6-bladed 1.2 m diameter rotor. It is not consensus whether the enlarged stator improved the efficiency, but combined with the diffuser, it reached a C_P around 0.5. This improved HKT was dubbed "Generation 2" and is presented in figure 9. The G2 turbine had a self-excited synchronous AC generator with a 2 kW capacity. Due to the generator's self-excited characteristic, it was not possible to control the stator current, and thus the control system used for this generation was similar to that of the Generation 1 turbine, instead of the improved strategy shown in figure 8. See figure 10 for the G2 control diagram.

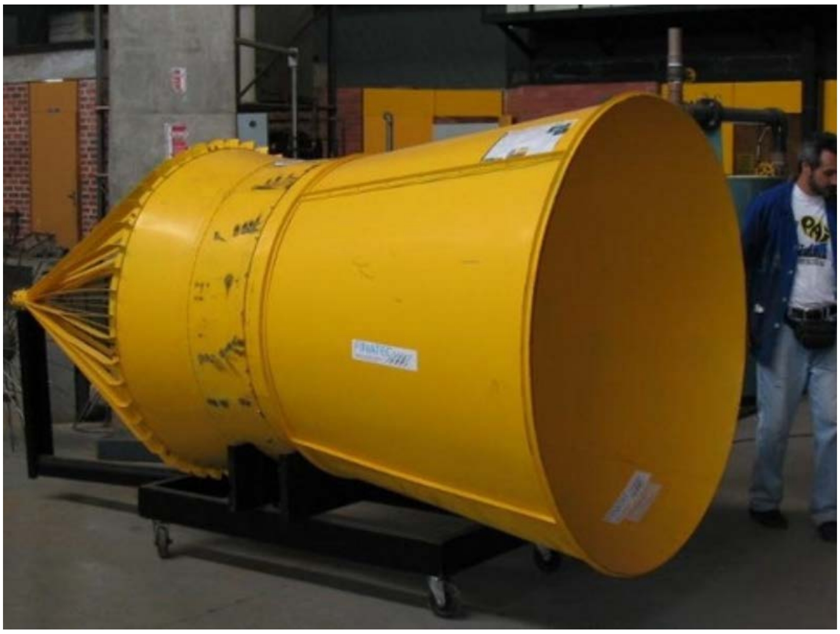


Figure 9 – Generation 2 turbine source: Brasil Jr. et al. (2007).

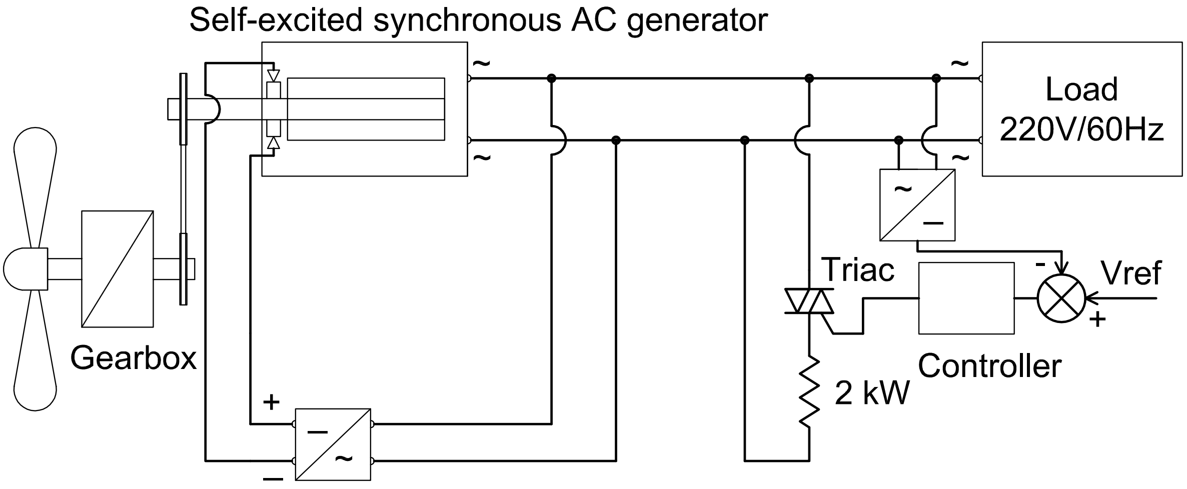


Figure 10 – Control schematic for the Generation 2 HKT.

3.3 Third generation hydro-kinetic turbine (G3)

The diffuser developed for Generation 2 increased the efficiency of the turbine, but had the drawback of a larger geometry. Which meant it could not be deployed in shallow rivers. For the next generation, dubbed "Generation 3", a more compact diffuser was developed and a gap was added between the main body and the diffuser in order to minimize the boundary layer detachment (refer to figure 11). The 4-bladed rotor had a diameter of 0.70 m and a C_P of about 1.0. Additionally, the DC generator was placed in the extension of the rotor, thereby creating a single rotor-generator unit. The generator had no control system installed (see figure 12).

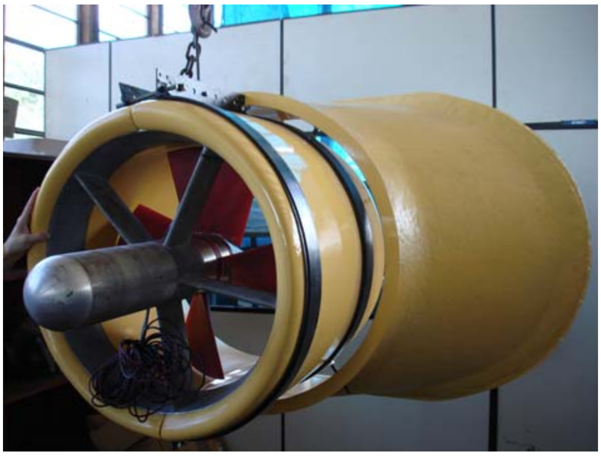


Figure 11 – Generation 3 turbine source: Brasil Jr. et al. (2007).

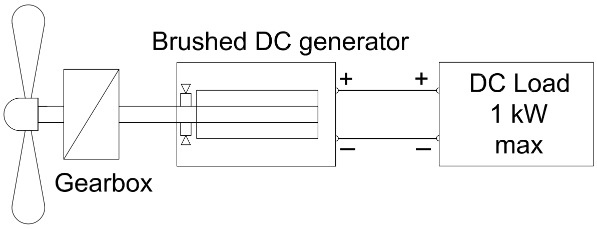


Figure 12 – Control schematic for the Generation 3 HKT.

3.4 Tucunaré hydro-kinetic turbine

A different hydro-kinetic project was that in cooperation with Electronorte, a Brazilian energy company. The project was named Tucunaré and had the goal of utilizing residual water flow from the Tucuruí HPP, in the Tocantins river, to generate a maximum of 500 kW. The turbine designed to reach this goal consisted of a three-bladed 11 m diameter rotor with a diffuser (FONSECA; ARAUJO, 2013). As of March 2017, the turbine has not been constructed, but various simulations have been realized, and experiments have been conducted on a scaled down prototype in a windtunnel. Furthermore, it is unknown which type of generator would be used, and the control system for this turbine has yet to be designed. Figure 13 (Source:(FONSECA; ARAUJO, 2013)) illustrates the Tucunaré turbine.

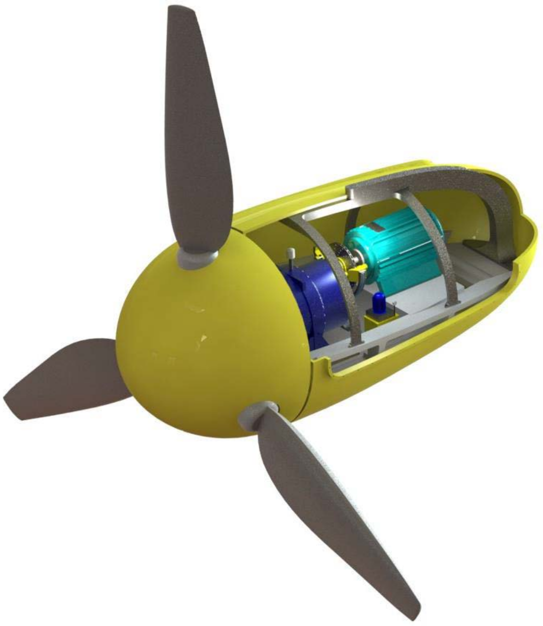
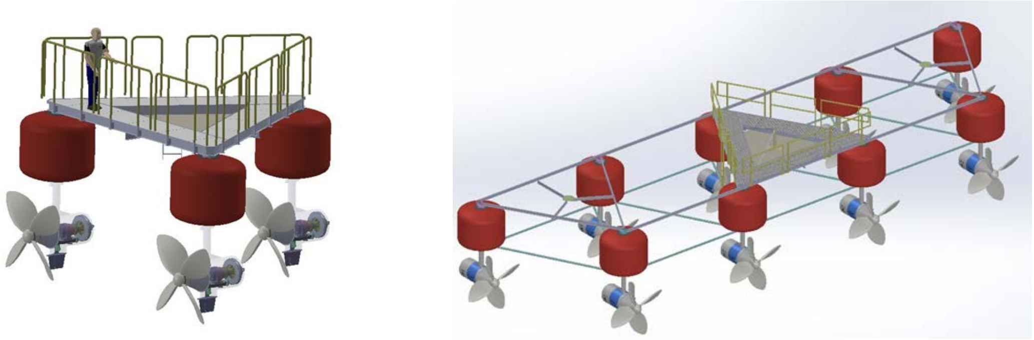


Figure 13 – Tucunaré turbine concept.

3.5 Hydro-K project

As aforementioned, this dissertation contains part of the contributions made to the Hydro-K project by the author. The Hydro-K project started in 2015 and was envisioned as a modular-floating hydrokinetic energy system that would generate electricity from the residual kinetic energy exiting a hydropower plant (HPP). The system's modularity emanates from the triangular arrangement of 3 hydrokinetic turbines, with the possibility to add standardized modules to increase the energy harnessing capacity as illustrated in figure 14b. Each turbine was designed with a capacity of 10 kW and was dubbed "HK-10". The HK-10 turbine is a horizontal-axis-axial-flow turbine and has a fixed-pitch-four-bladed 1.1 m radius rotor, a gearbox with a 1:12 ratio, and a permanent magnet synchronous generator (PMSG). A simulation of the turbine rotor indicated that theoretically a maximum C_P of 0.38 should be attainable (Brasil Jr. et al., 2016). The triangular arrangement of three such turbines was named "HK^3-10" (See figure 14a). As of February 2017, the construction of one HK-10 turbine and the triangular platform have been completed. On the 13th of March 2017, this arrangement was tested in Niterói substituting the unfinished HK-10 turbines with two counterweights.



(a) Triangular arrangement of three HK-10 turbines (HK^3-10). source: Brasil Jr. et al. (2017).

(b) Expanded arrangement of HK-10 turbines (HK-10 hydrokinetic farm).

Figure 14 – Different arrangements of HK-10 turbines.

It is expected that a HK^3-10 arrangement will be tested in the outlet channel of the Bariri HPP situated in the Tietê river in the Brazilian state of São Paulo. Bittencourt and Nunes describe a methodology for estimating the river velocity downriver of a HPP using a computational tool, a river velocity and depth measurement at various points in the area of interest, and the plant's outlet flow history. Although the nominal capacity of 10 kW is expected to be reached at a river velocity of 2.5 m/s, the aforementioned methodology resulted in a simulated river velocity, at the Bariri HPP, in the range of 0.61 m/s to 2.05 m/s (BITTENCOURT; NUNES, 2016).

Two of the parts that greatly impact the efficiency of these turbines are the rotor and the generator. The rotor converts the kinetic energy of the flowing water into a rotational torque that drives the generator, which in turn, produces electricity. The problem with this system is that the point of optimal efficiency is dependent on the fluctuating river velocity, and the varying electrical load applied to the generator. Therefore, controlling a HKT in a wide range of velocity and load variation is a challenge.

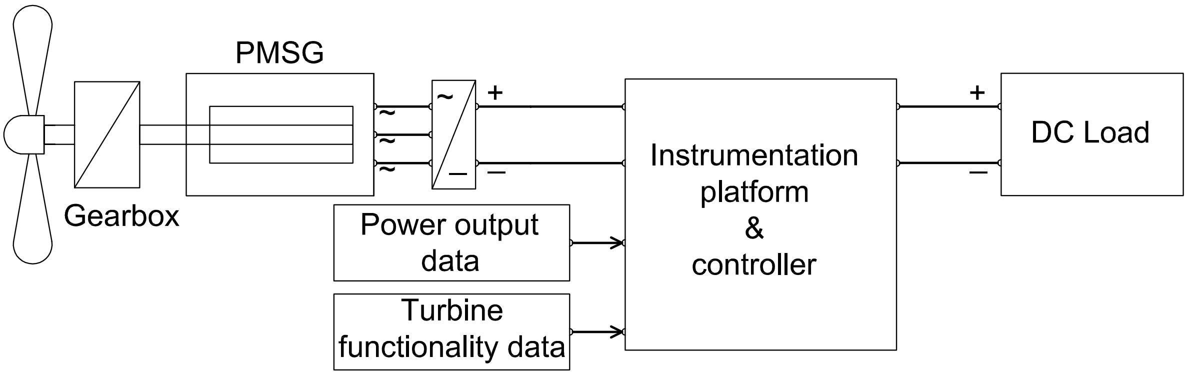

PMSG
~
~
~
~
+
−
+
−
Gearbox
Instrumentation platform & controller
DC Load
Power output data
Turbine functionality data

Figure 15 – General overview of the instrumentation and control schematic for the HK-10 turbine.

Seeing as the HK-10 is a new and experimental turbine, it is important to monitor its functionality (see figure 15). This will be achieved by designing and implementing an instrumentation platform. It will monitor both the generator power output and other turbine functionality parameters such as the temperature, vibrations, rotor velocity, river velocity, and also the presence of water inside the turbine. The conceptual instrumentation and control system for this turbine is shown in figure 16 and the designed MPPT algorithm is discussed in chapter 5. But first, the conception of the instrumentation platform used to monitor the functionality of the turbine arrangement will be discussed in the next chapter.

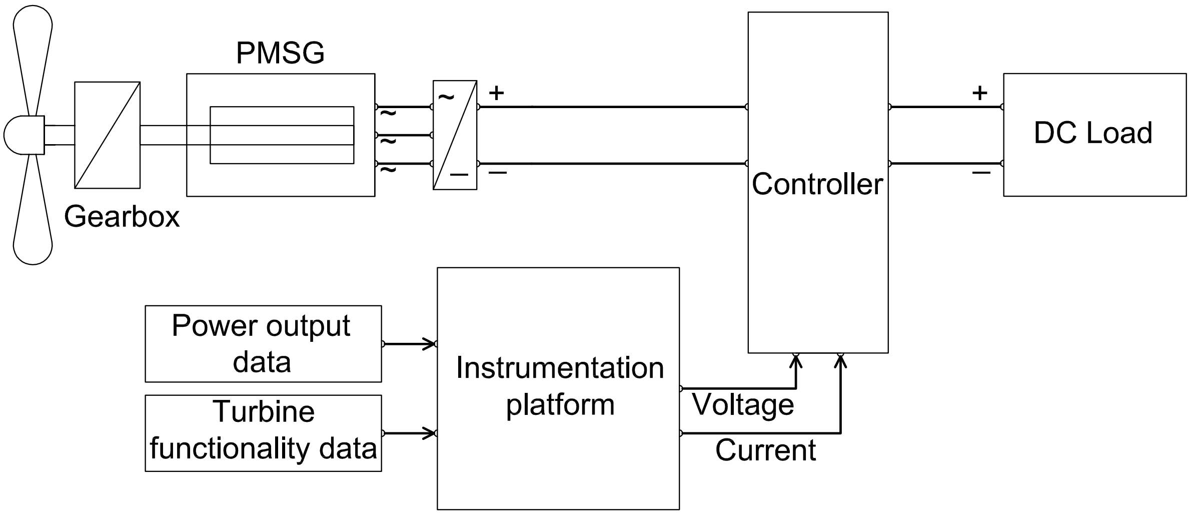


Figure 16 – Conceptual instrumentation and control schematic for the HK-10 turbine.

4 Instrumentation platform

An instrumentation platform was designed for hydrokinetic turbine monitoring applications with specifically the HK-10 turbine in mind. This platform is a base on which other instrumentation applications can be realized, because reconfigurability was one of the main objectives in its design. Instrumentation for wind turbines, and hydrokinetic turbine- or wind turbine windtunnel testing applications have many similar instrumentation requirements. One of those similarities is that the phenomenon under study is, when compared to electrical phenomenon, slow. Also, many of the individual measurement requirements are similar. Therefore, this instrumentation platform is focused on these three applications, but it is not limited to these applications.

4.1 Informational project

The primary goal of this platform is to facilitate research on wind- and hydrokinetic-turbines with the HK-10 turbines as a specific application, and secondarily incorporating flexibility for diverse instrumentation applications. One of the main reasons why the development of an instrumentation platform was chosen over the purchase of one, is the high cost of such products. With that being said, another secondary goal is to minimize the cost of this product. The following sections will discuss its development.

4.1.1 Stakeholder analysis

A survey of stakeholders is required to get more details on the project requirements as the product goes through its life cycle. By means of this survey, exhaustive product specification list can be generated.

The globally identified stakeholders for this project are listed below:

- Investors: the investors of this project are AES (Applied Energy Services), which is the Brazilian company funding the Hydro-K project, and UnB, which has produced various hydrokinetic turbines and is stimulating its research.
- Consumers: the consumers vary from a hobbyist to a research facility such as UnB.

- Production: production is dependent on parts suppliers, assembly lines, the author, packaging, manufacturer, etc.
- Standards and regulations: these are the laws which govern the restrictions and safety of the product.
- The environment: the effects of the product on the environment and it's habitants.
- Sales: the distribution and advertisement of the product.
- Concurrents: the similar products.
- Maintenance: maintenance of the product including spare parts production and sales.
- Disposal: the end of life for the product or reuse by recycling.

The stakeholders and their classification are shown in table 1.

Stakeholders' classification					
External	**Intermediate**	**Internal**	**Direct**	**Indirect**	**Stakeholders**
x		x		x	AES/UnB
x			x		Laboratories
x			x		Researchers
		x	x		Hydro-K project
x			x		Students
		x	x		Production team
		x	x		Parts suppliers
		x	x		Manufacturers
		x		x	ABNT/ISO
	x		x		Distribution
	x		x		Sales
	x			x	Shipping
x				x	Concurrents
x			x		Maintenance
x				x	Disposal services
x				x	Recyclers
x				x	Environment

Table 1 – List of Stakeholders with their classification.

4.1.2 Life cycle analysis

The life cycle of the product must be analyzed to identify all clients and their area of interest. A typical mechatronic product life cycle is shown in figure 17 (Adapted from (ROZENFELD et al., 2005)). After identifying the clients, their requirements could be obtained. The only client that clarified it's requirements was the Hydro-K development team, which is arguably the most important client. Regarding students and researchers as clients, a survey was made to obtain their individual requirements. Unfortunately, only four people chose to participate in this survey, but their opinions and requirements were considered. The remaining client requirements were obtained by brainstorming and the complete list of requirements is shown in table 2.

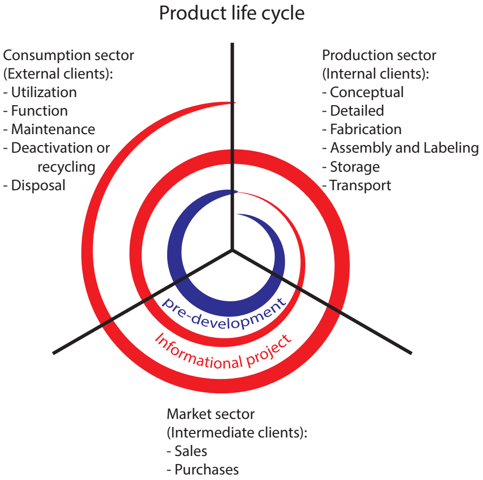


Figure 17 – Product life cycle.

4.1.3 Client requirements

In table 2 the requirements of the client were analyzed and, where possible, grouped into sub requirements minimize the number of requirements without losing the essence of the individual requirements. These were then, evaluated for importance using the "Mudge diagram" method. This method compares the requirements in pairs with an index from 0 to 5. For example, If requirement D dominates requirement "B" by 3, the symbol "D3" will be placed at the BD-intersection. If two requirements have the same importance, a "0" will be placed in the intersection. When all requirements have been compared, the last column indicates the sum of the symbols' indexes. These values indicate the importance of each requirement, and thus the impact on customer satisfaction.

Client requirements	**Client sub requirements**
Be easy to manufacture	Use standardized parts
	Easy to assemble
Be easy to ship	Lightweight
	Small volume
	Stackable
Be attractive to buy	Have an attractive label
	Have an attractive Packaging
Be easy to use	Good instruction manual
	Be simple
	Easy to calibrate
	Plug and play
Have long battery life	Low energy components
	Must have battery
	Low energy consumption
Be low cost	Low cost production
Be easy to dispose	Eco friendly materials
Be safe	Safe packaging
	Have built in safety
	Comply with safety regulations
Have high precision	High quality sensors
	High quality programming
Be Robust	High quality production
	Quality check
	Be reliable
	Comply with industry standards
Be eco friendly	Clean production
	Minimial residue
	Eco friendly materials
Allow serial connectivity	Serial port
Register captured data	Data storage
	RTC clock
Allow for data acquisition using IoT servers	Allow wireless connectivity
Allow efficiency measurements of wind- and hydro turbines	Be able to measure air-/water- speed
	Be able to measure atmospheric pressure
	Be able to measure humidity
	Be able to measure temperature
	Be able to measure forces
	Be able to measure electric voltage
	Be able to measure electric current
	Be able to measure rotational speed
	Be able to measure flowrate
	Be able to measure differential pressure
	Be able to detect water presence
	Be able to measure vibrations
Have large measurement range	High quality sensors
	Large range ADC
Color Legend:	**Internal client**
Intermediate client	**External client**

Table 2 – Full list of client sub requirements, grouped into main client requirements.

		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Sum	(%)
Be easy to manufacture	**A**		A4	A4	D3	E3	F1	A2	H3	I3	J3	K1	L3	M3	N3	O5	P3	10	3.76
Be easy to ship	**B**			C1	D5	E4	F2	B2	H4	I4	J4	K2	L3	M3	N4	O5	P3	2	0.75
Be attractive to buy	**C**				D5	E4	F1	G1	H4	I4	J4	C1	L3	M3	N4	O5	P3	2	0.75
Be easy to use	**D**					D2	D2	D4	0	D1	D2	D3	D1	D1	D2	O2	0	31	11.65
Have long battery life	**E**						E2	E2	H2	0	0	E1	L1	M1	N1	O3	E1	17	6.39
Be low cost	**F**							F4	H3	F2	F2	F3	L1	M1	F2	O3	F2	19	7.14
Be easy to dispose	**G**								H3	I3	J4	K2	L3	M3	N4	O5	P3	1	0.38
Be safe	**H**									H1	H1	H2	H1	H1	H1	0	H1	27	10.15
Have high precision	**I**										J1	I2	I1	I1	N1	O2	0	18	6.77
Be Robust	**J**											J2	L1	M1	0	O2	J1	19	7.14
Be eco friendly	**K**												L1	M1	N2	O3	P2	5	1.88
Allow serial connectivity	**L**													0	N2	O3	P1	16	6.02
Register captured data	**M**														N1	O3	P1	16	6.02
Allow for data acquisition using IoT servers	**N**															O1	N1	23	8.65
Allow eff. measurements of wind- and hydro turbines	**O**																O2	44	16.54
Have large measurement range	**P**																	16	6.02
Total																		266	100.0

Table 3 – Mudge diagram of the platform’s functions.

Index of importance: 0 = equally important, 1 = slightly more, 3 = moderately more, and 5 = extremely more important.

Client Requirements	Importance(%)
Allow eff. measurements of wind- and hydro turbines	**16.5**
Be easy to use	**11.7**
Be safe	**10.2**
Allow for data acquisition using IoT servers	**8.6**
Be low cost	**7.1**
Be Robust	**7.1**
Have high precision	**6.8**
Have long battery life	**6.4**
Allow serial connectivity	**6.0**
Register captured data	**6.0**
Have large measurement range	**6.0**
Be easy to manufacture	**3.8**
Be eco friendly	**1.9**
Be easy to ship	**0.8**
Be attractive to buy	**0.8**
Be easy to dispose	**0.4**

Table 4 – Results mudge diagram of customer function requirements.

The importance of the customer function requirements, resulting from the mudge diagram are shown in table 4.

The Kano Model of Customer Satisfaction classifies product attributes based on how they are perceived by customers and their effect on customer satisfaction. This classifications is useful for guiding project decisions, because they indicate when good is good enough, and when more is better. The Kano diagram for this project is shown in figure 18.

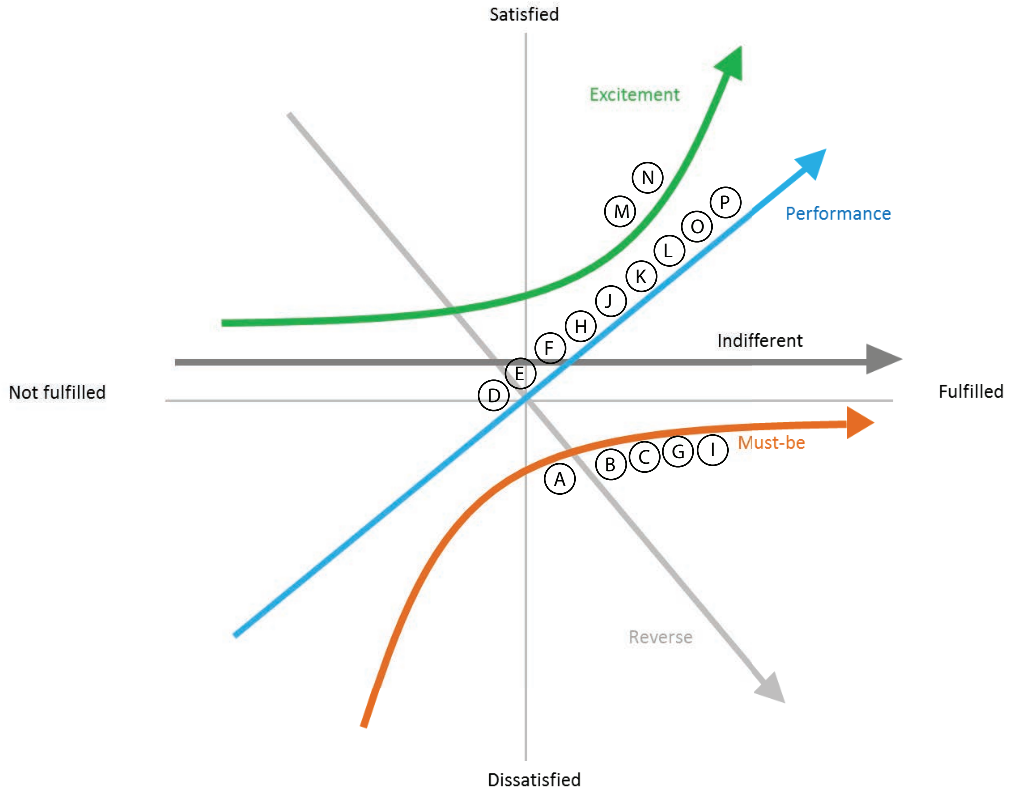


Figure 18 – Kano diagram of client satisfaction with requirements.

4.1.4 Product requirements

Client requirements are not always measurable or understood in the field of fabrication. This problem can be solved by converting the client requirements into measurable "product requirements" that collectively attend to the client requirements. Accordingly, the conversion of client requirements to product requirements is done for each client requirement by seeking one or more measurable parameters that can address that client requirement. The result of this conversion is shown in table 5.

Client Requirements	**Product Requirements**	**Direction of Improvement & Function**
Be easy to manufacture	Number of standardized components	Higher number of components that are homogeneous
	Number of components	Lower number of components for assembly
Be easy to ship	Weight	Lower shipping weight
	Volume	Lower shipping volume
Be attractive to buy	Body geometry design	Generally accepted allure
Be easy to use	Number of user's actions per measurement	Lower number of user input for measurements
	Number of user's actions per calibration	Lower number of user input for calibrations
Have long battery life	Battery capacity	Higher battery capacity for longer uninterrupted product usage
	Energy consumption	Lower energy consumption of the product for longer uninterrupted product usage
Be low cost	Price	Lower cost
Be easy to dispose	Number of recycleable components	Higher number of components adequate for recycling
Be safe	Number of electricaly isolated components	Higher isolation for electrical components for insured safety
Have high precision	ADC resolution	Higher analog to digital resolution for higher accuracy
Be Robust	Number of electricaly isolated components	Higher isolation for electrical components for reliability
	Frequency of system failure	Lower number of system failures
	Body material tensile strength	Higher material tensile strength for increased drop resistance
Be eco friendly	Number of recycleable components	Higher number of components adequate for reusability of materials
	Usable lifetime	Higher lifetime for decreased material waste
Allow serial connectivity	Number of serial ports	Higher number of serial ports for variable serial connectivity
Register captured data	Data storage capacity	Higher data storage capacity for longer uninterrupted product usage
Allow for data acquisition using IoT servers	Internet speed	Higher internet speeds for stable data transfer
Allow eff. measurements of wind- and hydro turbines	Number of data channels	Higher number of input data channels for simultaneous measurement of various parameters
Have large measurement range	Level of interference	Lower level of interference for an increased sensitivity of sensors

Table 5 – Product requirements in accordance with the client requirements .

Quality function deployment (House of quality, QFD) is a method of linking the client requirements with the product requirements and finding their importance. QFD is designed to help planners focus on characteristics of a new or existing product from the viewpoint of the market needs, company needs, or technology development needs. The QFD for this project is shown in table 6. For more literature regarding QFD see Hauser and Clausing (1988).

Number of standardized components																					
Number of components	++																				
Weight		++																			
Volume		++	++																		
Number of user's actions per measurement																					
Energy consumption					+																
Body geometry design	-	+		+																	
Number of user's actions per calibration																					
Number of electricaly isolated components																					
Number of data channels																					
Body material tensile strength																					
Level of interference									++												
ADC resolution												+									
Price	+						+		-	-	-		-								
Number of serial ports										+				-							
Data storage capacity													-	-							
Battery capacity			-	-	+	++				-				-		-					
Frequency of system failure									+			+									
Usable lifetime						+			+		+			-			+	+			
Number of recycleable components	++																				
Internet Speed		-				-				+		+		-			-				
Direction of improvement >>>>	▲	▼	▼	▼	▼	▼	x	▼	▲	▲	▲	▼	▲	▼	▲	▲	▲	▼	▲	▲	▲

Correlation between PR	
++	Positive strong
+	Positive weak
	Non existent
-	Negative weak
- -	Negative strong

Correlation between PR & CR	
Strong	5
Medium	3
Weak	1

Direction of improvement for PR	
X	Has to be the precise value
▲	The bigger the better
▼	The lesser the better

																										Prioritization			**Benchmarking**		
		Level of importance for CR	**Product requirements (PR)**	Number of standardized components	Number of components	Weight	Volume	Number of user's actions per measurement	Energy consumption	Body geometry design	Number of user's actions per calibration	Number of electricaly isolated components	Number of data channels	Body material tensile strength	Level of interference	ADC resolution	Price	Number of serial ports	Data storage capacity	Battery capacity	Frequency of system failure	Usable lifetime	Number of recycleable components	Internet Speed		**Client**	**Kano (internal)**	**General degree of importance**	**Our product**	**National instruments**	**Bare Arduino**
Customer requirements (CR)	Be easy to manufacture	4		5	3							1					3	1	1	1			3	1		**1**	**B**	**3**	**4**	**2**	**5**
	Be easy to ship	1			1	3	3			1							1			1						**1**	**B**	**3**	**4**	**3**	**5**
	Be attractive to buy	1				3	1	1	1	5	1		1	1		1	1	1	1	1	3	1				**4**	**B**	**2**	**4**	**3**	**5**
	Be easy to use	12				3	3	5	1		3		1		1			1	1							**5**	**P**	**2**	**3**	**2**	**3**
	Have long battery life	7				1		1	3			3	1			1	3	3		5	1			3		**5**	**P**	**5**	**5**	**X**	**X**
	Be low cost	2		3		3						1		1		1	5	1	1	3			3	1		**5**	**P**	**5**	**4**	**2**	**5**
	Be easy to dispose	1		3	3	1	1			1				1			1			1			3	1		**2**	**B**	**5**	**3**	**2**	**4**
	Be safe	10		1		1						3		1									1			**5**	**P**	**5**	**4**	**3**	**5**
	Have high precision	8							1		5	3			5	5	1			1	1					**5**	**B**	**3**	**3**	**5**	**3**
	Be Robust	8		3	3	1				1		5		5		1			1	3	5	5	3	1		**5**	**P**	**5**	**3**	**5**	**4**
	Be eco friendly	3		1	3	1	1		1											1	1	1	1			**3**	**P**	**4**	**4**	**3**	**5**
	Allow serial connectivity	6							3						1		1	5		1						**4**	**P**	**5**	**5**	**5**	**5**
	Register captured data	6							3		1		1		1		1		5	1						**4**	**EX**	**4**	**4**	**5**	**1**
	Allow for data acquisition using IoT servers	9							1				1			1	1			1				5		**4**	**EX**	**3**	**3**	**2**	**X**
	Allow eff. measurements of wind- and hydro turbines	17						3			1	1	1		1	1	1			1	1					**3**	**P**	**3**	**4**	**5**	**x**
	Have large measurement range	7									3				3	5	1									**5**	**P**	**4**	**4**	**x**	**x**
	Importance level for PR			63.2	45.6	75.1	43.3	119.2	88.9	13.0	118.4	134.5	51.7	52.1	99.2	113.8	95.4	69.0	57.1	116.1	74.3	41.8	55.6	79.7	1527.2						
	Percentages			4.1	3.0	4.9	2.8	7.8	5.8	0.9	7.8	8.8	3.4	3.4	6.5	7.5	6.2	4.5	3.7	7.6	4.9	2.7	3.6	5.2	100						

Table 6 – Quality Function Deployment for product features.

The results of the QFD in terms of the product requirements, their importance, their goals, and sensors are listed in table 7.

QFD Score	Product requirements	Objective	Units	Goal	Sensor	Unwanted Results
8.5	Number of electrically isolated components	High number of component isolation	Number	All components shielded and isolated	Counter	No shielded components
8.1	Battery capacity	High battery capacity for enhanced time of use	mAh	Minimum 3500 mAh	Wattmeter	Less than 3500 mAh
7.6	Price	Low price	$	Maximum $100	Price	More than $100
7.4	Number of user's actions per measurement	Minimal manual actions per measurement	Number	Least number of actions	Counter	More than 7
7.1	ADC resolution	High ADC resolution for enhanced precision	bits	Minimum 8 bits	Tests	Less than 8 bits
7.1	Number of user's actions per calibration	Minimal manual actions for calibration	Number	Least number of actions	Counter	Difficult calibration process
5.9	Level of interference	Low noise and interference for measurements	dB	Maximum 5 dB	Oscilloscope	More than 15 dB
5.6	Weight	Light weight for shipping	Kg	Maximum 1 kg	Balance	More than 2 kg
5.5	Energy consumption	Low energy consumption	Watt	Maximum 100 mAh	Wattmeter	More than 400 mAh
5.2	Internet Speed	Higher internet speeds for stable data transfer	kbps	Minimum 512 kbps	Speedtest	Less than 64kbps
4.8	Number of standardized components	High number of standardized-readily-available components	Number	Maximum number of standardized components	Counter	No standardized components
4.7	Number of serial ports	Number of Serial communication capabilities	Number	At least 1 port	Counter	No serial ports
4.5	Frequency of system failure	Low occurrences of system failure	Hz	0 Hz	Tests	More than 1 Hz
4.3	Number of recyclable components	High number of recyclable components	Number	All components recyclable	Counter	No recyclable components
3.9	Data storage capacity	Large space for data saving	bytes	Minimum 2 Gb	Datasheet	No data storage
3.5	Body material tensile strength	High tensile strength for durability of body	Pa	100 Mpa minimum	Tests	less than 20 Mpa
3.2	Number of data channels	High number of input for measurement of various parameters	Number	40 minimum	Counter	Less than 30
2.6	Number of components	Low number of components for assembling	Number	Least number of components	Counter	High number of components
2.6	Volume	Small dimensions for shipping	m^3	Maximum 0.0015 m^3	Ruler	More than 0.0030 m^3
2.5	Usable lifetime	Long life of usage	Years	Minimum 5 year life	Clock	Less than one year
0.8	Body geometry design	Attractive and robust design		Attractive	Opinions	Unattractive design

Table 7 – Product requirements.

4.2 Conceptual design

The conceptual design phase of a project is one of the most decisive parts for the success of a product, because this phase encapsulates the inception of solutions, which requires creativity to meet or surpass the product requirements in a unique and most efficient manner, compared to the concurrents. As such, the product requirements obtained in the informational phase form the basis of the search for solutions and alternative solutions.

4.2.1 Functional structure of the product

The search for solutions and alternative solutions starts with the modeling of the product's functional structure, which is a description of the product on an abstract level. Resulting with the structure of the product without any restrictions in terms of solutions (ROZENFELD et al., 2005).

The global function of the product is shown in figure 19 where continuous lines represent an energy, dotted lines represent a signal, and dashed lines represent a material.

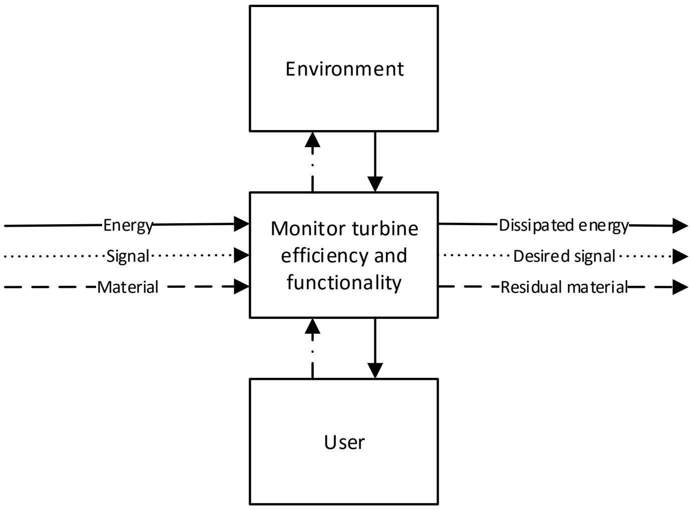


Figure 19 – Global function of the product.

As this is a mechatronic product, it will most likely consist of mechanical, electronic, and computational components. As such, they will be perceptible in the partial functions of the system (figure 20).

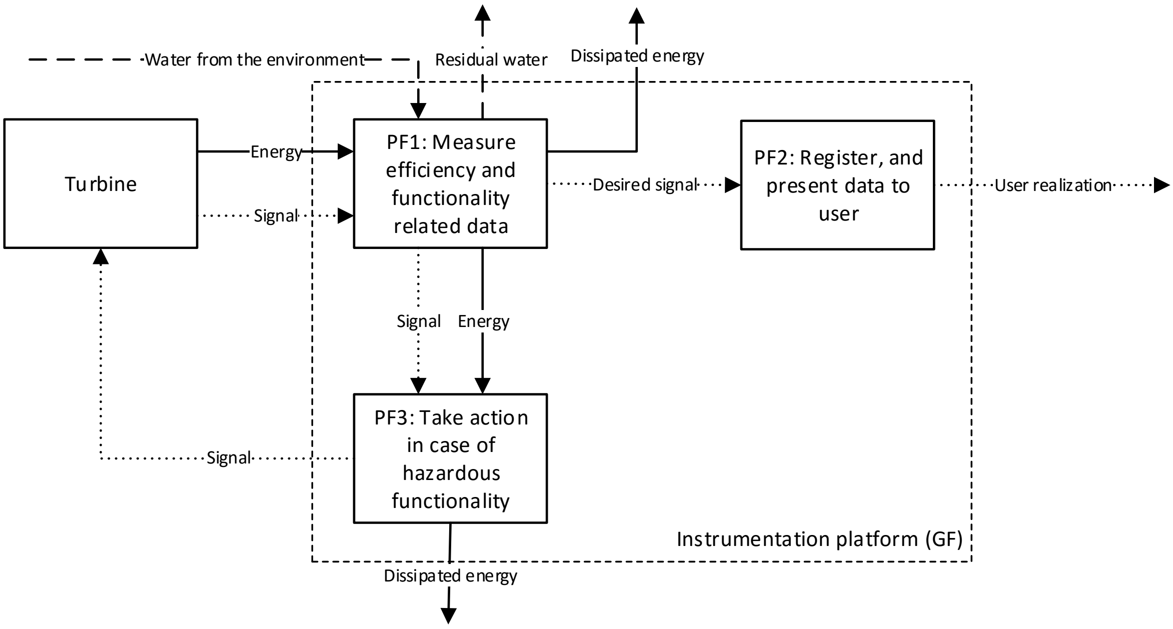

Water from the environment
Residual water
Dissipated energy
Turbine
Energy
Signal
PF1: Measure efficiency and functionality related data
Desired signal
PF2: Register, and present data to user
User realization
Signal
Energy
PF3: Take action in case of hazardous functionality
Signal
Instrumentation platform (GF)
Dissipated energy

Figure 20 – Partial functions of the product.

The more detailed elementary functions of the partial functions can be structured in various forms. The three best alternatives are shown in the following figure 21, 22, and 23. Alternative one, shown in figure 21, is structured to have all the elementary functions on one single platform without the utilization of any modules. This alternative satisfies all client requirements, but may have a larger electronic noise sensitivity.

Therefore, alternative two, shown in figure 22, was formed. Separating the high noise prone functions from the low noise prone functions. Thereby reducing the probability of electronic interference. Note that the registering and presenting of data is also included on the platform. This could also be affected by electronic interference. Another observation is that the high noise functions are those that describe the power generation of the turbine, while the functionality parameters constitute the low noise functions.

Consequently, a third alternative, shown in figure 23, was formed. This alternative separates the registering and presenting of data from the acquisition of the data. Resulting in a greater reduction in electronic interference, but may not be necessary, due to the separation of the high noise prone functions from the low noise prone functions, and depending on the final solution principle, may show an adverse effect.

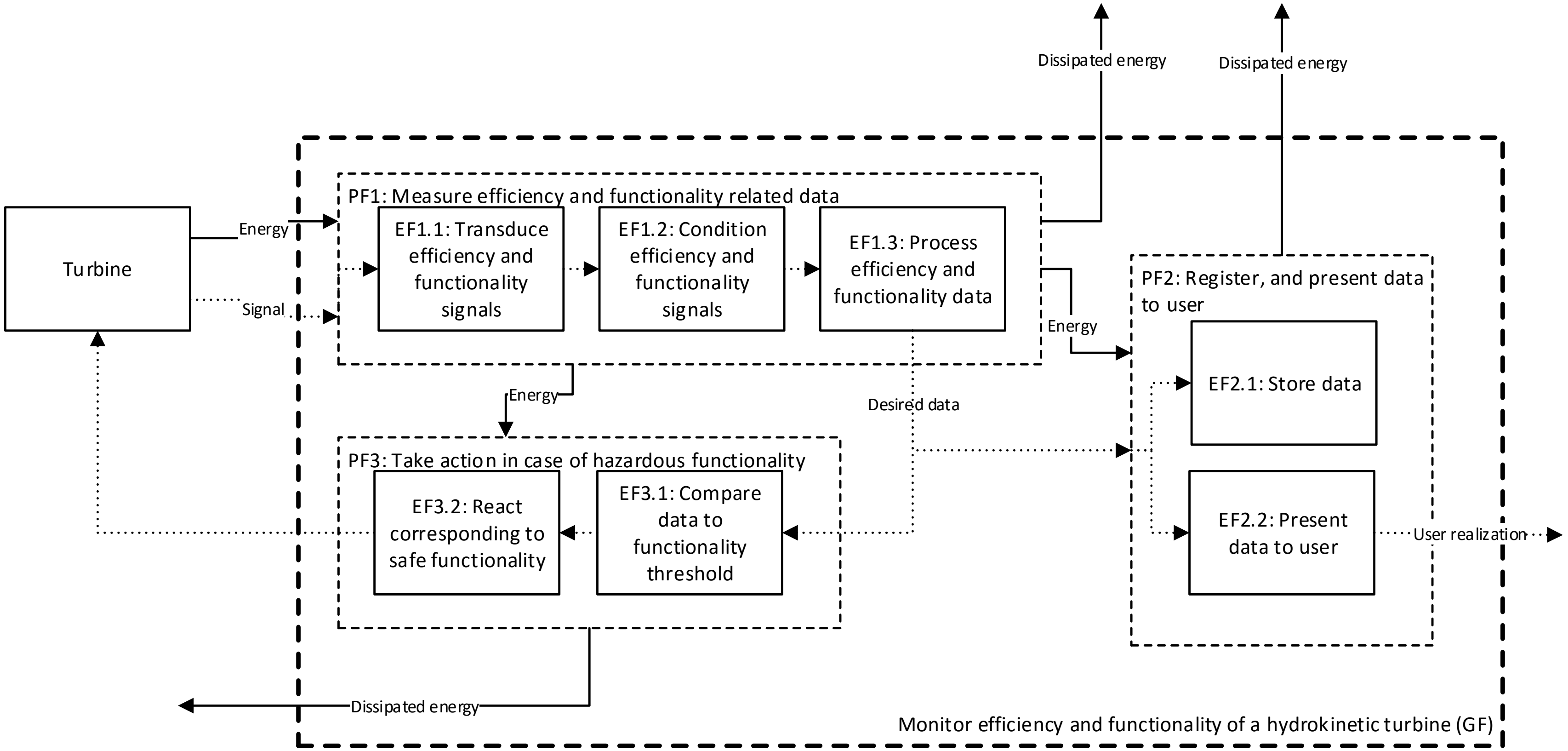

Turbine
Energy
Signal
PF1: Measure efficiency and functionality related data
EF1.1: Transduce efficiency and functionality signals
EF1.2: Condition efficiency and functionality signals
EF1.3: Process efficiency and functionality data
Dissipated energy
Dissipated energy
Energy
PF2: Register, and present data to user
EF2.1: Store data
EF2.2: Present data to user
User realization
Energy
Desired data
PF3: Take action in case of hazardous functionality
EF3.2: React corresponding to safe functionality
EF3.1: Compare data to functionality threshold
Dissipated energy
Monitor efficiency and functionality of a hydrokinetic turbine (GF)

Figure 21 – Elementary functions of the 1st alternative functional model.

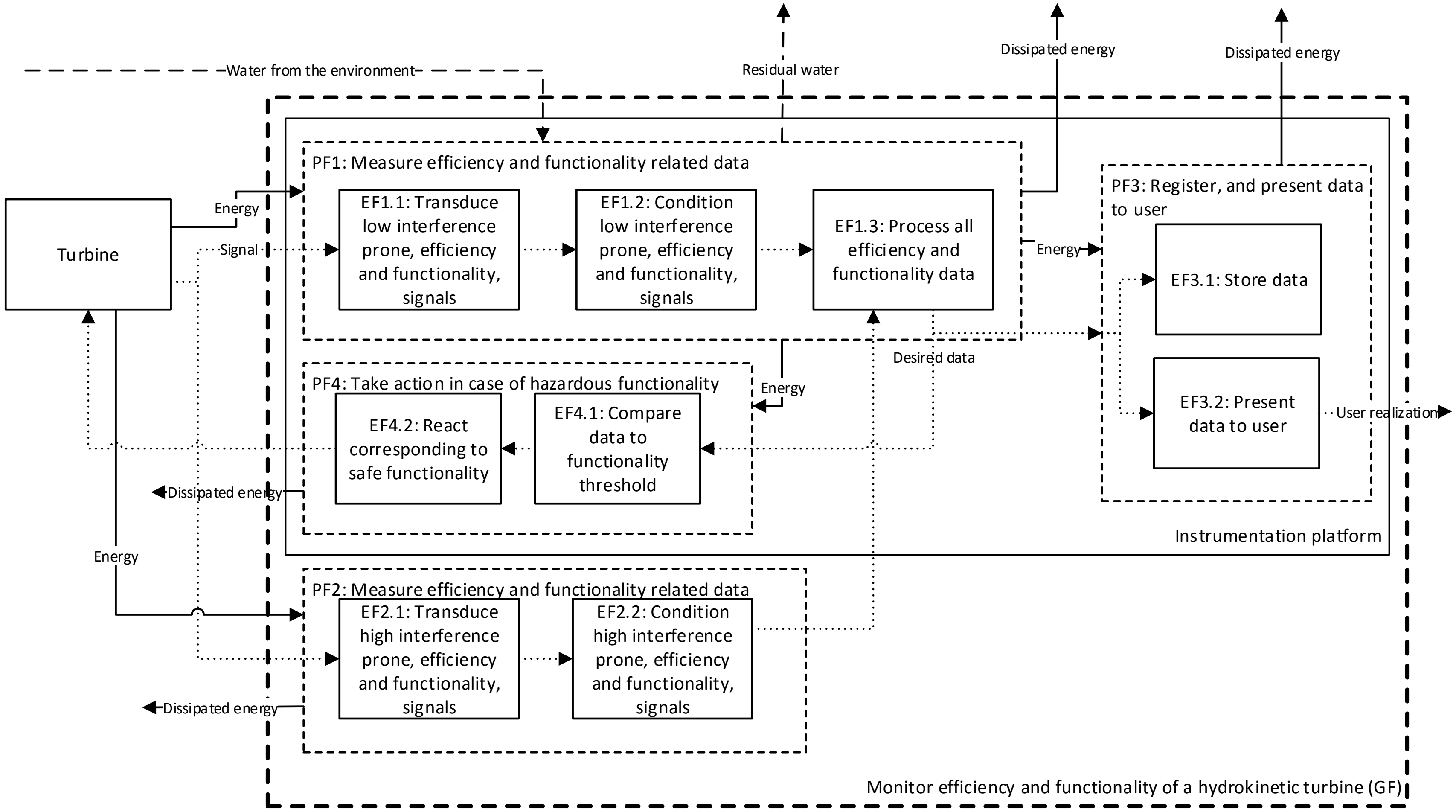

Water from the environment
Residual water
Dissipated energy
Dissipated energy
Turbine
Energy
Signal
PF1: Measure efficiency and functionality related data
EF1.1: Transduce low interference prone, efficiency and functionality, signals
EF1.2: Condition low interference prone, efficiency and functionality, signals
EF1.3: Process all efficiency and functionality data
Energy
PF3: Register, and present data to user
EF3.1: Store data
EF3.2: Present data to user
User realization
Desired data
Energy
PF4: Take action in case of hazardous functionality
EF4.2: React corresponding to safe functionality
EF4.1: Compare data to functionality threshold
Dissipated energy
Instrumentation platform
Energy
PF2: Measure efficiency and functionality related data
EF2.1: Transduce high interference prone, efficiency and functionality, signals
EF2.2: Condition high interference prone, efficiency and functionality, signals
Dissipated energy
Monitor efficiency and functionality of a hydrokinetic turbine (GF)

Figure 22 – Elementary functions of the 2nd alternative functional model.

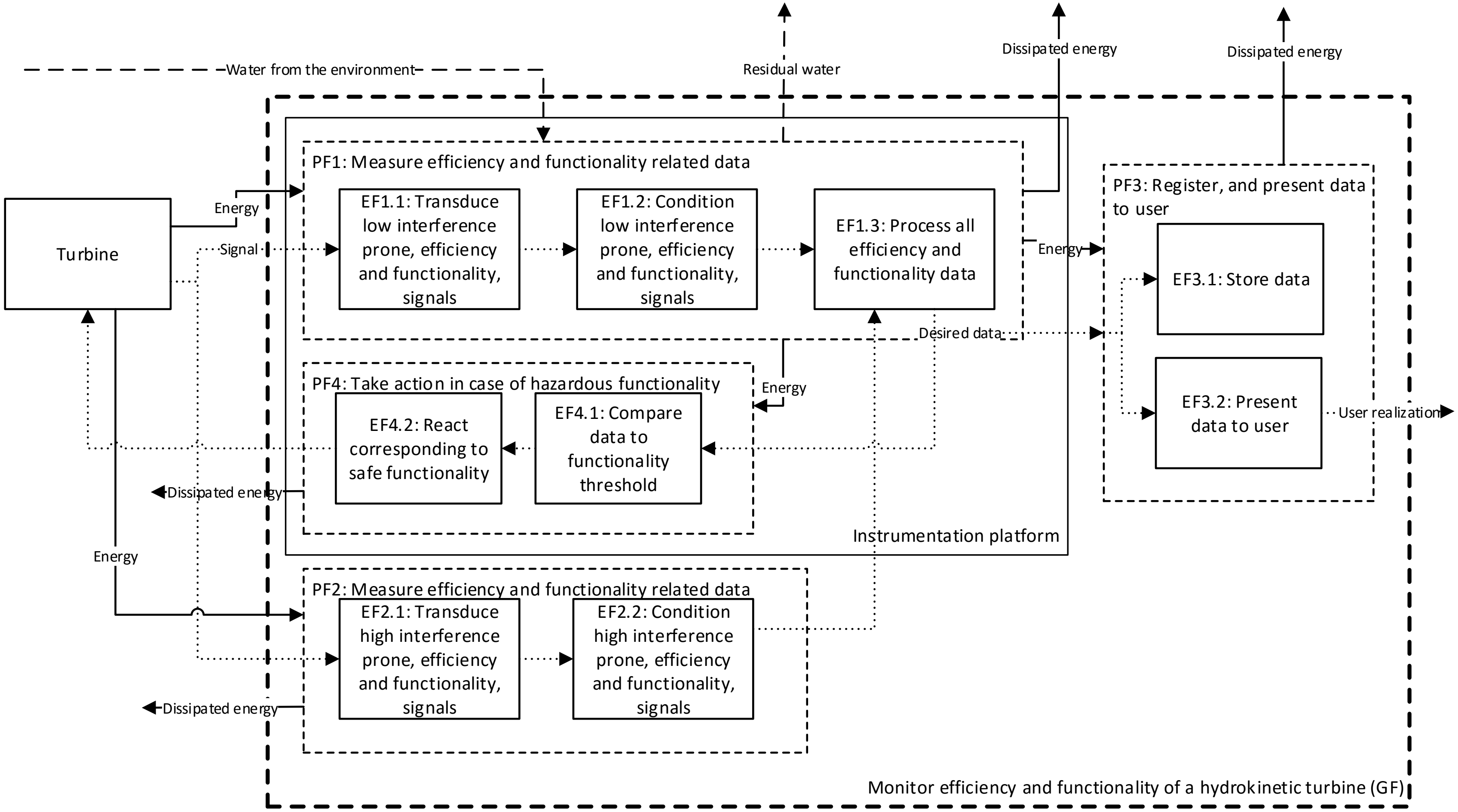

Water from the environment
Residual water
Dissipated energy
Dissipated energy
Turbine
Energy
Signal
PF1: Measure efficiency and functionality related data
EF1.1: Transduce low interference prone, efficiency and functionality, signals
EF1.2: Condition low interference prone, efficiency and functionality, signals
EF1.3: Process all efficiency and functionality data
Energy
Desired data
PF3: Register, and present data to user
EF3.1: Store data
EF3.2: Present data to user
User realization
PF4: Take action in case of hazardous functionality
EF4.2: React corresponding to safe functionality
EF4.1: Compare data to functionality threshold
Energy
Dissipated energy
Instrumentation platform
Energy
PF2: Measure efficiency and functionality related data
EF2.1: Transduce high interference prone, efficiency and functionality, signals
EF2.2: Condition high interference prone, efficiency and functionality, signals
Dissipated energy
Monitor efficiency and functionality of a hydrokinetic turbine (GF)

Figure 23 – Elementary functions of the 3rd alternative functional model.

Client requirements	**Importance**	**Alternatives** **for the functional structure of the product**		
		Alternative 1	**Alternative 2**	**Alternative 3**
Allow eff. measurements of wind- and hydro turbines	5	3	5	5
Be easy to use	4	1	5	3
Be safe	4	1	3	3
Allow for data acquisition using IoT servers	3	3	5	3
Be low cost	3	3	3	3
Be Robust	3	3	5	5
Have high precision	3	3	3	3
Have long battery life	2	3	3	3
Allow serial connectivity	2	3	3	3
Register captured data	2	5	5	5
Have large measurement range	2	3	5	5
Be easy to manufacture	2	5	3	3
Be eco friendly	1	3	3	3
Be easy to ship	1	3	3	3
Be attractive to buy	1	3	3	1
Be easy to dispose	1	3	3	1
Sum of "alternative score" × "importance"		**109**	**155**	**137**

Table 8 – Selection matrix for the functional structure

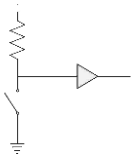
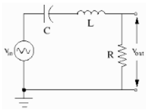
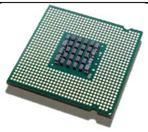
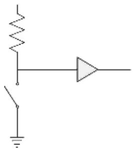
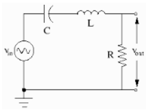
Table 8 shows the selection matrix of the functional structure, comparing the aforementioned alternatives to find the best of the three by assigning values of the degree in which the structure satisfies the client requirements, where 5 represents an excellent satisfaction, 3 represents a good satisfaction, and 1 represents a weak satisfaction.

Alternative two presented the highest satisfaction of the three alternatives. And was therefore selected as the product's functional structure.

4.2.2 The search for alternative solutions and concepts

Each elementary function of the selected function structure (see figure 22) has various possibilities of solutions. Identifying them, requires one's imagination, and increases the chance of the product's success, by considering all possible solutions, and selecting the most appropriate solution satisfying the client requirements.

Function	**Solution principle**			
	Option A	Option B	Option C	Option D
EF1.1 Transduce rotor rpm	rotary encoder	infrared tachometer	mechanical tachometer	proximity sensor
EF1.1 Transduce temperatures	analog thermometer	digital thermometer	infrared thermometer	thermocouple
EF1.1 Transduce vibration	analog accelerometer	digital accelerometer	piezo vibration sensor	vibration switch
EF1.1 Detect water presence	barometer	chemical water level sensor	digital water level sensor	float switch

EF1.2 Condition signals	 amplifier	 decoupler	 pull up or down resistor	 RLC filter
EF1.3 Process signals	 FPGA	 microcontroller	 microprocessor	 PLC
EF2.1 Transduce river velocity	 acoustic dopler	 laser dopler	 pitot tube	 propeller
EF2.1 Transduce current	 digital ammeter	 galvanometer	 current shunt	 current transformer
EF2.1 Transduce voltage	 digital voltmeter	 voltage divider	 galvanometer	 voltage transformer
EF2.2 Condition signals	 amplifier	 decoupler	 pull up or down resistor	 RLC filter

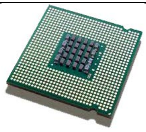
EF3.1 Store data	 EEPROM	 HDD	 SD	 SSD
EF3.2 Present data	 display	 ethernet transmission	 fiber optics ttl	 wireless transmission
EF4.1 Compare data	 FPGA	 microcon-troller	 micropro-cessor	 PLC
EF4.2 Actuate safety measures	 IGBT	 mechanical relay	 SSR	 TRIAC

Table 9 – Selection matrix for the functional structure

4.2.3 Product concept

Using the alternative solutions and concepts found in the last subsection one can generate various product concepts. A product concept is one way of achieving the product's main goal and satisfying the client's requirements. A product can have numerous concepts. Each has its advantages and disadvantages. The purpose of this subsection is to acquire the most uncompromising product concept by first formulating various concepts and then selecting the best concept.

Before any concepts are formulated, the modularity of the product must be analyzed. This can lead to faster development, reduced costs, and improved manufacturing conditions. There are various methods for defining the modules of a product. The method that will be used is the "Heuristic" method described by (STONE; WOOD; CRAWFORD, 2000). The heuristic method was chosen for its simplified empirical rules, which have been scientifically validated. The empirical rules are as follows:

- Derive the functional model of the product as described by (STONE; WOOD; CRAWFORD, 2000) (see figure 22);
- Identify the modules using the "Dominant flow heuristic" (see figure 24, source: (STONE; WOOD; CRAWFORD, 2000));
- Identify the modules using the "Branching flow heuristic" (see figure 25, source: (STONE; WOOD; CRAWFORD, 2000));
- Identify the modules using the "Conversion—transmission heuristic" (see figure 26, source: (STONE; WOOD; CRAWFORD, 2000));
- The modules identified by the heuristic that resulted in the least modules define the product modularity (STONE; MCADAMS; KAYYALETHEKKEL, 2004).

The dominant heuristic is described by (STONE; WOOD; CRAWFORD, 2000) as:

"The set of sub-functions which a flow passes through, from entry or initiation of the flow in the system to exit from the system or conversion of the flow within the system, define a module."

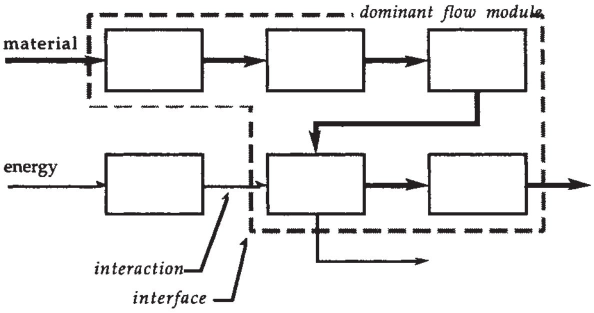


Figure 24 – Dominant flow heuristic applied to a functional structure.

"The branching heuristic is described by (STONE; WOOD; CRAWFORD, 2000) as: "The limbs of a parallel function chain constitute modules. Each of the modules interface with the remainder of the product through the flow at the branch point."

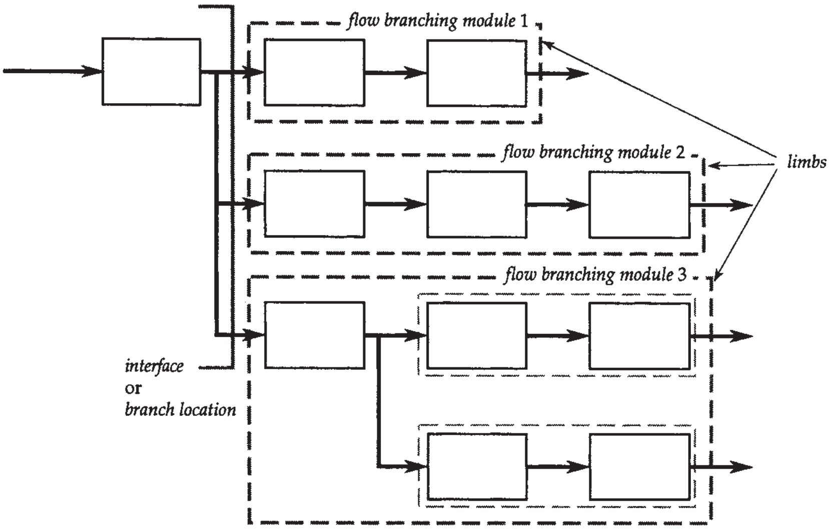


Figure 25 – Branching flow heuristic applied to a functional structure.

And the Conversion—transmission heuristic is stated by (STONE; WOOD; CRAWFORD, 2000) as:

"A conversion sub-function or a conversion—transmission pair or proper chain of sub-functions constitutes a module." 3.3.1"

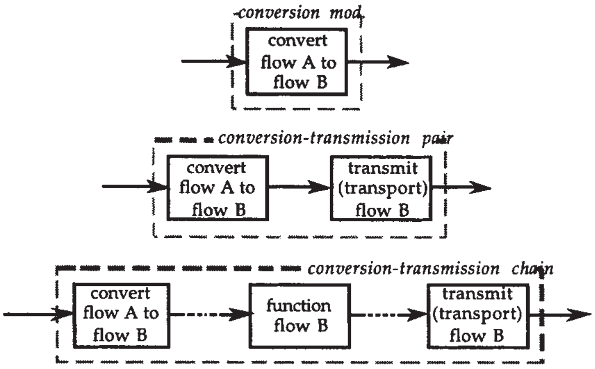


Figure 26 – Conversion—transmission heuristic applied to a functional structure.

From figure 22, the dominant heuristic resulted in the least amount of modules, and therefore describes the modularity of the product. The identified modules are:

Module 1: Data processing, registering, and safety measures (actuation),
Module 2: Power data gathering,
Module 3: Functionality data gathering.

Various product concepts have been formulated with the abovementioned modularity. Three concepts are presented in figures 27, 28, and 29.

Concept 1 (figure27) uses a PLC as data processor, saves data to a hard drive, sends data to a user via ethernet, and enables or disables the safety measures using an Insulated Gate Bipolar Transistor (IGBT).

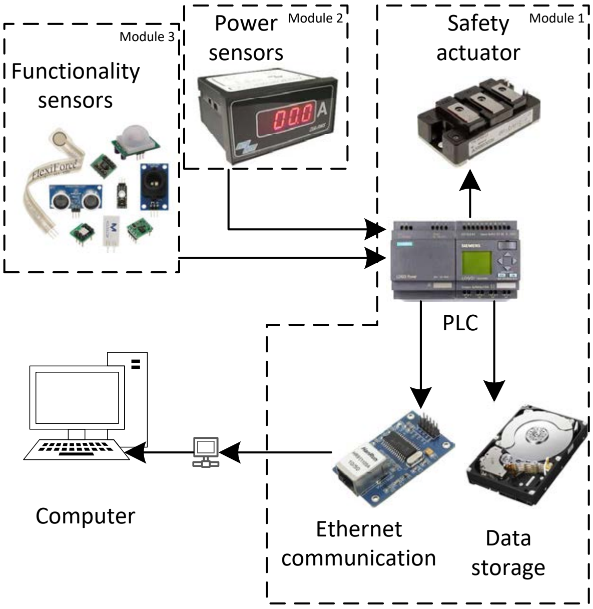


Figure 27 – Concept 1. PLC for data processing

Concept 2 (figure28) presents a microprocessor for processing data, a hard drive as data storage, a solid state actuator, and fiber optics TTL communication.

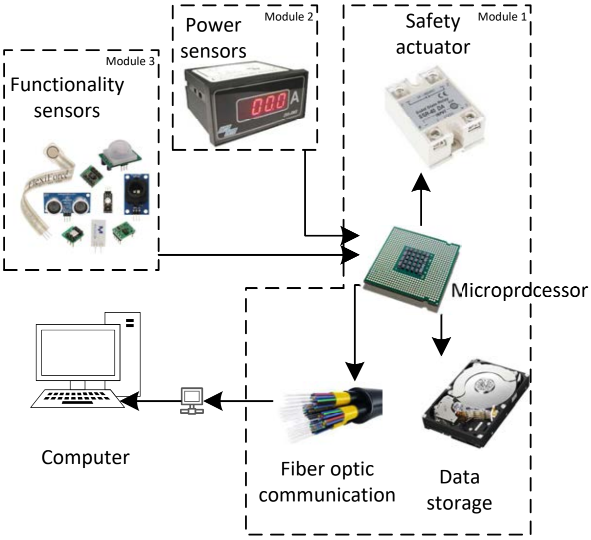


Figure 28 – Concept 2. Microprocessor as data processor

Concept 3 (figure29) introduces wireless connectivity, a microcontroller as data processor, a solenoid relay as safety actuator, and an SD card as data storage.

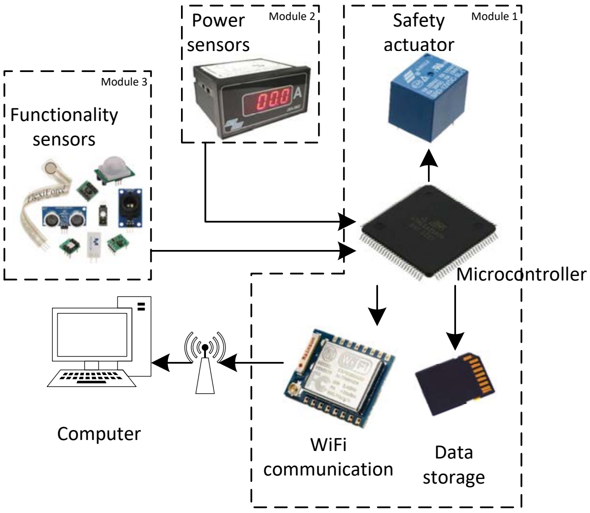


Figure 29 – Concept 3. Microcontroller as data processor

The concept selection was once more determined using a selection matrix shown in table 10. Note that the selection of the functionality- and power- sensors was left out of this matrix as this would clutter the table. Nevertheless, a selection matrix was employed to determine the most beneficial sensors (of the four options as displayed in table 9) and is presented in table 13 of appendix A. Furthermore, non of the signal conditioning components has been identified as superior, because non of them is suitable for all situations. In many cases, a combination of the signal conditioning components is needed. The complete list of the selected components, with the concept identified as superior, is presented in table 11.

Client requirements	**Importance**	**Alternatives product concepts**		
		Concept 1	**Concept 2**	**Concept 3**
Allow eff. measurements of wind- and hydro turbines	5	3	3	5
Be easy to use	4	1	3	5
Be safe	4	5	3	3
Allow for data acquisition using IoT servers	3	1	5	3
Be low cost	3	1	3	5
Be Robust	3	5	3	3
Have high precision	3	3	3	3
Have long battery life	2	3	3	5
Allow serial connectivity	2	5	5	3
Register captured data	2	1	3	5
Have large measurement range	2	3	3	1
Be easy to manufacture	2	1	1	3
Be eco friendly	1	3	3	3
Be easy to ship	1	3	3	3
Be attractive to buy	1	1	3	3
Be easy to dispose	1	3	3	3
Sum of "concept score" × "importance"		**105**	**123**	**145**

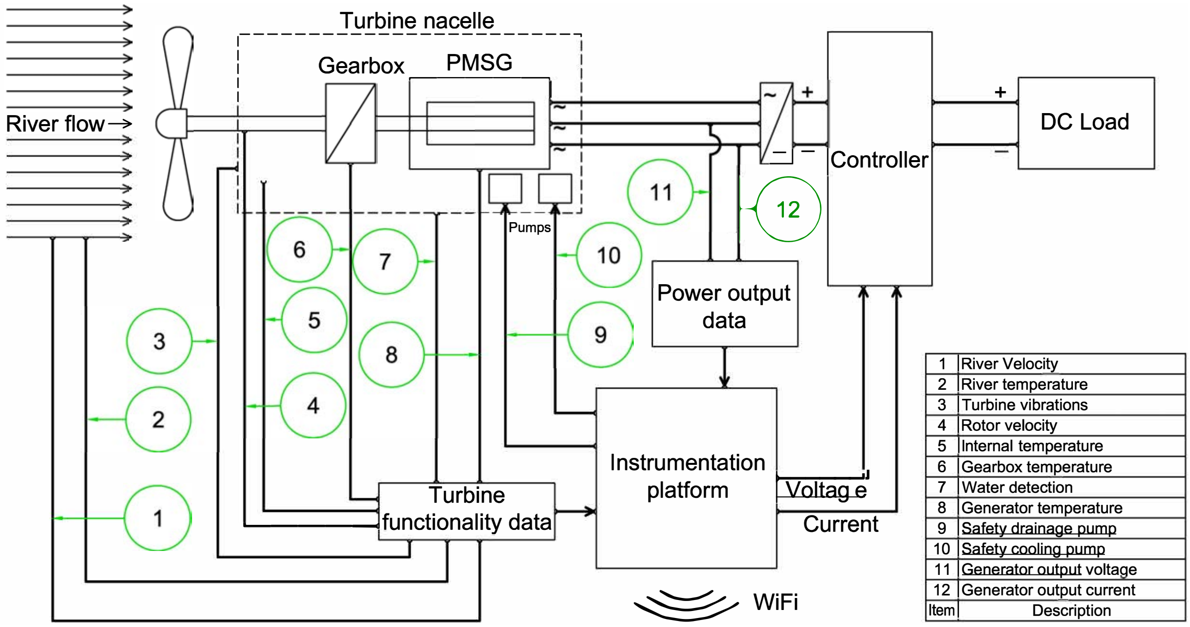
Table 10 – Product concept selection matrix

Function		**Solution**
EF1.1	Transduce rotor velocity	Proximity sensor
EF1.1	Transduce temperatures	Digital thermometer
EF1.1	Transduce vibration	Analog accelerometer
EF1.1	Detect water presence	Float switch
EF1.2	Condition signals	Undefined
EF1.3	Process signals	Microcontroller
EF2.1	Transduce river velocity	Propeller
EF2.1	Transduce current	Current transformer
EF2.1	Transduce voltage	Voltage divider
EF2.2	Condition signals	Undefined
EF3.1	Store data	SD Card
EF3.2	Present data	WiFi
EF4.1	Compare data	Microcontroller
EF4.2	Actuate signals	Mechanical Relay

Table 11 – Selected components of the product concept

4.3 Detailed design

As the instrumentation platform is primarily designed for the Hydro-K project, the following will describe the detailed design as required for this project. Nevertheless, the platform has been projected to be flexible in application, up to this point, and will remain so primarily due to the selected data processor. This hardware will be discussed in the next subsection. But first, refer to figure 30 for the instrumentation necessities.



Item	Description
1	River Velocity
2	River temperature
3	Turbine vibrations
4	Rotor velocity
5	Internal temperature
6	Gearbox temperature
7	Water detection
8	Generator temperature
9	Safety drainage pump
10	Safety cooling pump
11	Generator output voltage
12	Generator output current

Figure 30 – Instrumentation requirements for the Hydro-K project

4.3.1 Hardware

Having selected a microcontroller as data processor, a decision was made to use the Arduino Mega 2560. This open-source electronics platform is based on an ATMEL ATmega2560 microcontroller that has a clock speed of 16 MHz, with 256 KB of flash memory, of which 8 KB is used by a pre-programmed bootloader, it also has 8 KB of SRAM, and 4 KB of EEPROM. The Arduino Mega 2560 operates at 5 VDC, is powered with a voltage of between 6 - 20 VDC, and also offers a 3.3 V power output. Furthermore, the 54 digital I/O pins and 16 analog input pins that the MEGA provides, support and array of other features such as: I^{2}C, PWM, external interrupts, SPI, TTL serial communication, and a built in LED. Each turbine uses 5 analog ports and 8 digital ports, summing up to 16 analog and 38 digital ports when including the platform's functions (see table 14 in appendix B for the platform's pin reference).

The maximum supported current on each of these pins is 20 mA, and 50 mA on the 3.3 V pin. A 10-bit ADC converts an input on its analog pins from 0 - 5 V into a digital signal from 0 - 1024. The features that this electronics platform offers will safeguard the instrumentation platform's application flexibility.

The remaining hardware components of module 1 (defined in sub section 4.2.3) will be described next. First, the WiFi component added to this module is an "ESP8266-07". This component is low cost and offers: SPI, I^{2}C, and TTL serial communication, a 10-bit ADC, a ceramic antenna plus a U-FL connector, and has its own microcontroller running at 80 MHz. The TTL serial communication was used to communicate with the Arduino Mega. Note that a CC3000 WiFi shield has been tested and is discussed in sub section 4.4, but the final WiFi component is the ESP8266-07. Next, the safety actuation component is a JQX-3F mechanical relay which has a 5 V activation coil, a rated load of 10 A 250 VAC/30 VDC, and an insulation resistance capable of handling voltages greater than 1000 VAC/500 VDC. The last component of module 1 is the SD Card. A standard SD Card slot was added to save data on an SD Card. The SD Card slot operates at 3.3 V or 5 V and uses the SPI protocol to communicate with the Arduino Mega. Moreover, a "Tiny RTC" was included in module 1 to store the date and time along with the data to the SD card. This RTC communicates with the Mega using the I^{2}C protocol.

Module 2 (power data gathering) consists of a current transformer and a voltage divider. The selected current transformer is a SCT-013-000. This transformer was chosen for its low price and allows non-invasive AC-current measurements of up to 100 A and consists of a 2000 turn coil. Interfacing this sensor with module 1 required signal conditioning. The calculations for this signal conditioning are described next.

First of all, measuring currents with the selected microcontroller is impossible. The current must somehow be transformed into a voltage between 0 - 5 V. The transformation can be achieved with a burden resistor, which needs to be calculated to provide a good resolution. Due to the 2000 turns of its coil, the current transformer will decrease the measured current 2000 times. And according to the generator specialist of the Hydro-K project, the maximum current delivered by the generator is 35 A RMS. This means that the primary peak current will be:

$$Primary\ peak\ current = RMS\ current \times \sqrt{2} = 35A \times \sqrt{2} = 49.50A$$

The secondary peak current will therefore be:

$$Secondary\ peak\ current = Primary\ peak\ current/no.\ of\ turns\ in\ coil = 49.50A/2000 = 24.75mA$$

Converting this secondary current into a DC current requires a rectifier. Using a simple diode bridge, the AC wave is cut in half and thereby increasing the measurement resolution. The ideal burden resistor was calculated as follows:

$$Ideal\ burden\ resistor = AREF/secondary\ peak\ current = 5V/0.02475A = 202.02\Omega$$

Where AREF is the maximum reference voltage of the microcontroller, which is 5 V. Seeing as resistors are not ideal, it is recommended to choose a somewhat higher resistance than the ideal resistance to avoid surpassing 5 V. In this case though, the schottky diode bridge already contributes to the resistance, and therefore a lower burden resistance of 200Ωwas chosen. A 10 µF capacitor was also added to smoothen the rectified current. Figure 31 presents the schematic of the current transformer's signal conditioning and figure 72 in appendix E presents the PCB built for interfacing this current transformer with module 1.

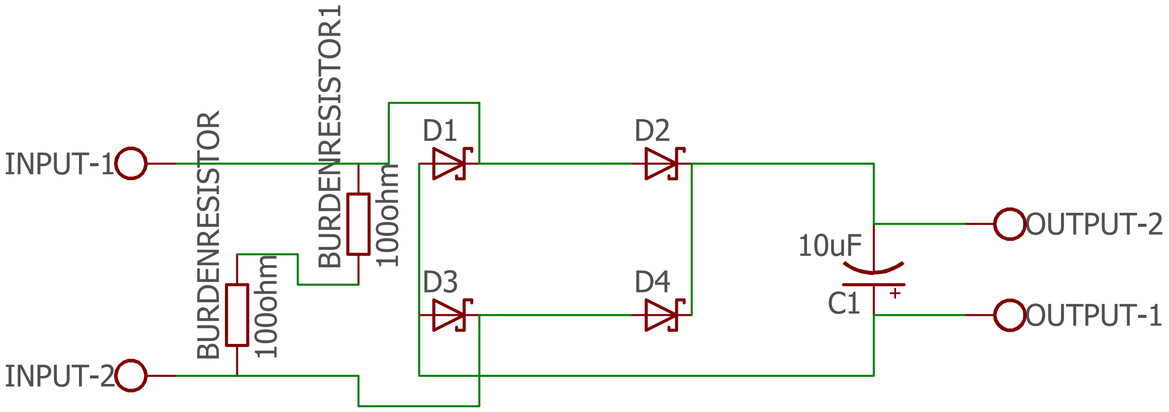


Figure 31 – Signal conditioning for current transformer

The voltage divider required for generator power output measuring, was unavailable on the market. Therefore, a custom voltage divider was designed. Originally, the voltage measurement would be on the DC side of the generator output (figure 30) due to the anticipation that in the HK^3-10 formation, multiple generators could be connected in parallel on the DC side of the rectifier. And thereby reducing the amount of generator output cables from 9 to 2, but this would prevent the voltage measurement of a single turbine, and would instead measure the voltage of the array.

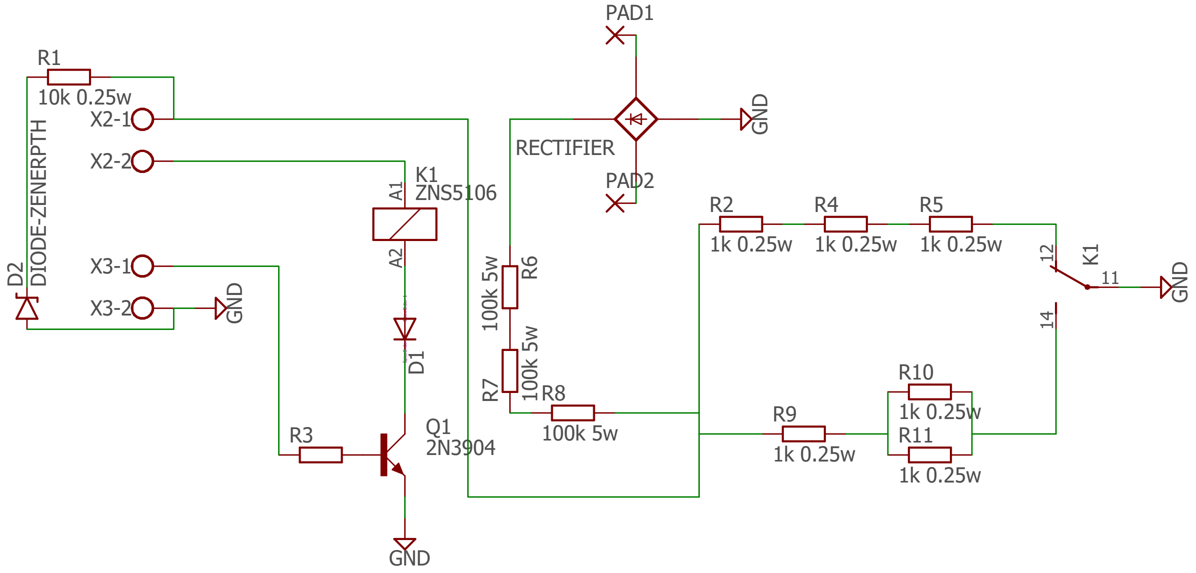


Figure 32 – Schematic of the designed voltage divider

For the operating conditions of the voltage divider, the generator specialist was consulted once more. The nominal voltage of the turbine should be around 400 V, but during an open-circuit or no-load test of the generator, a voltage higher than 1000 V was measured. In reality, the generator should never be in a no-load situation, but for safety reasons the voltage divider was designed to measure up to 1000 V. Figure 32 shows the schematic of the designed voltage divider. Notice that the voltage divider includes a 10 kW capacity rectifier, a mechanical relay, and a zener diode. The rectifier converts the AC voltage into DC voltage, whereafter the voltage is divided in a 1000 V to 5 V ratio. If the voltage is lower than 500 V, the relay is activated by the microcontroller which then produces a 500 V to 5 V ratio. In case the 5 V is surpassed, a zener diode connects the surplus voltage to ground. The main reason for designing a custom voltage divider, was the commercial unavailability of a low cost voltage sensor capable of measuring voltages within the desired range. Beware, when high voltages are measured, the ground of the rectifier may have a potential much higher than 0 V. In this case TTL communication with a computer can be destructive for the computer.

The last module, module 3 (functional data gathering), includes a plethora of sensors. Some of which do not need any special signal conditioning to be interfaced with module 1, whereas others do. The simple signal conditioning hardware were added directly on module 1. The DS18B20 digital temperature sensors for example, only need a 4.7 kΩ pull-up resistor. These sensors were used to measure all the required temperatures illustrated in figure 30, because they are waterproof, have a 9- to 12-bit ADC with a range of -55°C to 125°C, a ± 0.5°C accuracy, are pre calibrated by the manufacturer, and multiple sensors can be read using only one data bus. As temperature sensors are common for instrumentation purposes, their pull-up resistors were added directly onto module 1.

Another sensor that was easily interfaced with module 1 is the LJ12A3 proximity sensor that was used to measure the rotor's rotational velocity. The sensor requires a power source of at least 12 V, and the output is binary (on = 12 V, off > 1.5 V). A simple 2-resistor voltage divider changed the output voltage to a 5 V maximum. These resistors were also directly added to module 1. The working principle of this sensor is that it detects metal cogs of a cogwheel added to the rotor axle. Knowing the number of cogs, and the time elapsed between two cogs, one can calculate the rotational velocity.

The two most easily interfaced sensors were the water detection sensor, which was a simple On/Off floating switch, and the analog ADXL335 accelerometer. The accelerometer measures vibrations in 2 axes, is supplied with 3.3 V, and also has an output of 3.3 V. The output being 3.3 V, implies that no signal conditioning was needed.

The most challenging sensor interface was that of the FLOWATCH water velocity measuring impeller. Its operating principle is based on a magnet and a coil. The magnet rotates with the impeller and every full rotation it induces a small alternating current in the waterproof coil. The impeller comes in a kit complete with a handheld meter that can display the water velocity in various units of speed. Interfacing this impeller with the Arduino Mega meant designing an amplifier, that could convert and amplify the small AC current into an AC voltage with 5 V peak. Figure 33 shows the schematic of the designed amplifier. The impeller's signal was first filtered with an RC filter, afterwards the signal was amplified with an instrumentational amplifier, which was made using three of the four op-amps of a LM324. The fourth op-amp was used as a comparator, which switched to 5 V if a certain input voltage was passed and switched back to 0 V if the input was under the voltage threshold. A correlation between the river velocity and the time elapsed between two peaks could then be made. The calibration of all the above sensors will be discussed in subsection 4.4 after the software design.

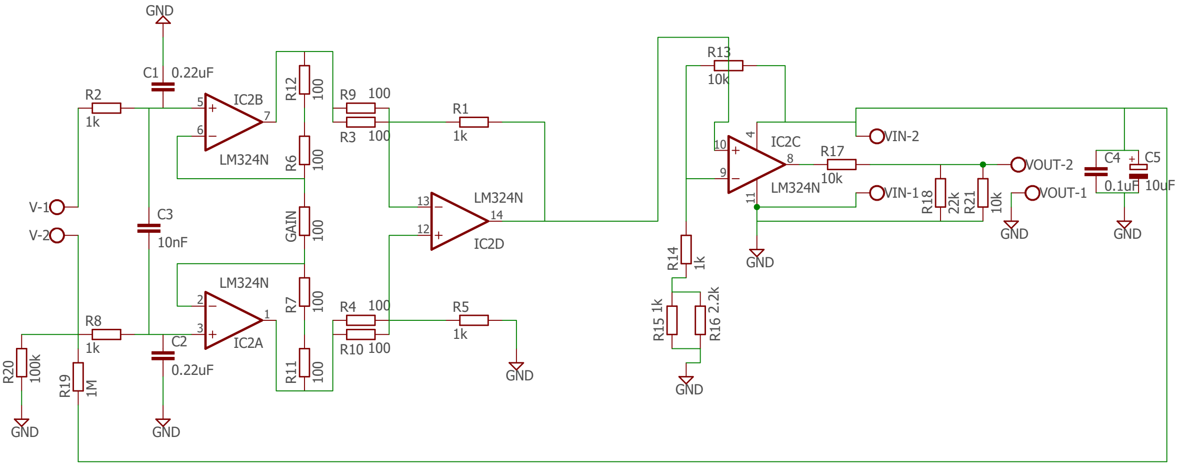


Figure 33 – Schematic for designed water flow sensor

The power for all these sensors and components is supplied by two 7000 mAH batteries (see figure 35a), which are projected to be charged by the turbine to ensure an uninterrupted power supply.

4.3.2 Software development

To send instructions to the selected microcontroller, the Arduino Integrated Development Environment (IDE) was used along with the "Arduino programming language" This language is similar to C++ and a program in this language is called a sketch. In other words, using the Arduino programming language, one creates a sketch inside the IDE and uploads the instructions to the microcontroller on the Arduino board. A sketch was made for each interfaced component in order to test them. In order to create the final sketch which will integrated all components, a Finite State Machine was first devised.

A Finite State Machine (FSM) is a mathematical model used for programming. The machine can only be in one state at a time and can change from one state to another when a certain condition is met or an event occurs. Transition conditions are unique per state, which means that there can never be two or more identical transition conditions leaving one state. The program can also remain a state with a transition condition that returns to the same state. All states and conditions should be analyzed in order to prevent any undefined conditions, i.e., all possible situations must be anticipated. For these reasons, this type of model aids with the anticipation of lockups, and undesired functions. This is imperative for any programs intended for remote monitoring use. A simplified finite state machine diagram is shown in figure 34. The legend for the diagram is shown in table 12. The simplifications that were made in this diagram are: the merging of the three types of anticipated dangers, and the simplification of the data acquisition. If these simplifications were not made, the FSM would most likely consist of more than 64 states. This would diminish clarity and contradict the purpose of the diagram.

The logic of the resulting FSM diagram is as follows. The setup state is the initial and final state, meaning the program never stops running. After the setup has completed, transition condition "A" is met, and the program transitions to state "1". In state "1" the program constantly checks for safety parameters, acquires sensor data, checks for lockups using a watchdog timer, checks the time that has passed since the last data registration, and checks the date. The safety parameters that are checked for are the temperatures inside the turbine, the presence of water inside the turbine (as a result of unwanted water infiltration into the turbine), and a high voltage output of the generator. A cooling pump is used to respond to high temperatures, a drainage pump is activated in case of water presence, and a relay is switched off, increasing the resistance in case of high voltage. Responding to detected dangers is done in state "2" and "4". The responding action is also the only difference between state "1" and "2". Reading sensor data includes voltages, currents, temperatures, vibrations, rotational velocities, river velocity and temperature, and water presence. And if the elapsed time, between the last data registration and the current time, is larger than the time threshold of data registration, the machine enters state "3" or state "4" depending on whether there is a danger or not. In state "3" and "4" the acquired data is stored on an SD card with the time and sent to a server via Modbus IP/TCP. Note that in sub section 4.4 the "HTTP protocol" was first tested, but the Modbus IP protocol proved to be more superior. Another detail is that the date is checked in order to save the data from each day in a separate file. Hereby facilitating the data analysis at a later stage.

Implementing the FSM, resulted in a robust program. During the conception of the final sketch, the logic of the FSM was maintained for the instrumentation of 1, 2, or 3 HK-10 turbines. Changing from one configuration to another only requires the program's intial parameters to be changed, as all of the program's tasks were grouped into "functions". This made the sketch less cluttered and easier to reprogram. The final sketch is presented in appendix H, and by changing parameters and uploading the adapted sketch to the microcontroller, it can serve for the instrumentation of 1, 2 or 3 HK-10 turbines. This completes the backend application for informing an observer of the turbine functionality and the generator output. The backend application sends the data to a database, which is also part of the backend, via WiFi.

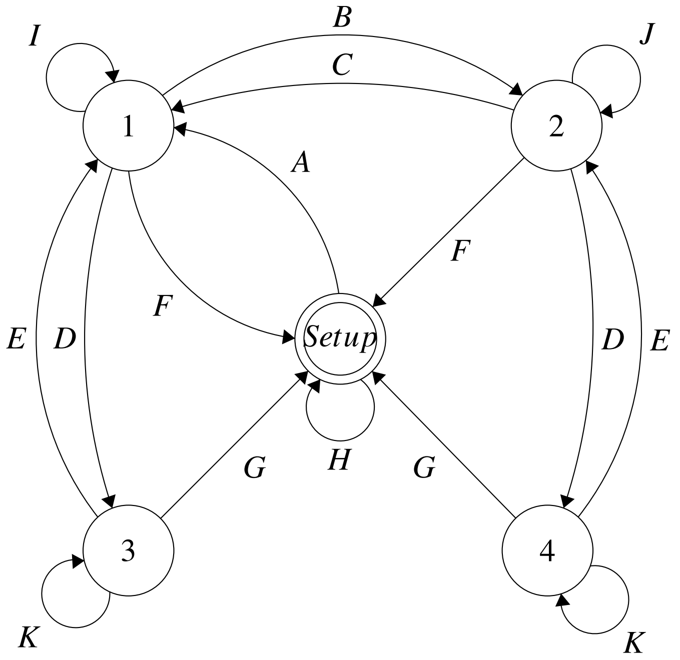


Figure 34 – Finite State Machine diagram for program

State #	Description	Transition #	Transition condition
Setup	Setting up system	**A**	Setup finished
1	Checking safety parameters	**B**	Danger detected
	Reading sensor data	**C**	Danger resolved
	Checking watchdog timer	**D**	Δt >Data registering threshold
	Checking Δt	**E**	Data succesfully registered
	Checking date	**F**	Lockup detected
2	Responding to detected danger		Date change detected
	Checking safety parameters	**G**	Lockup detected
	Reading sensor data		Data not registered
	Checking watchdog timer	**H**	Lockup detected
	Checking Δt	**I**	Δt <Data registering threshold
	Checking date		No danger detected
3	Saving data to SD card with time and date		Date not changed
	Sending data via WiFi modbus tcp	**J**	Δt <Data registering threshold
4	Responding to detected danger		Danger detected
	Saving data to SD card with time and date		Date not changed
	Sending data via WiFi modbus tcp	**K**	Data registering in progress

Table 12 – Legend for Finite State Machine diagram displayed above

A MySQL database was created, and ScadaBR was used to serve as a frontend. ScadaBR is a tomcat based open-source Supervisory Control and Data Acquisition (SCADA) system, which as default offers a derby database, but can be configured to support the created MySQL database. ScadaBR offers various communication protocols including the Modbus IP protocol.

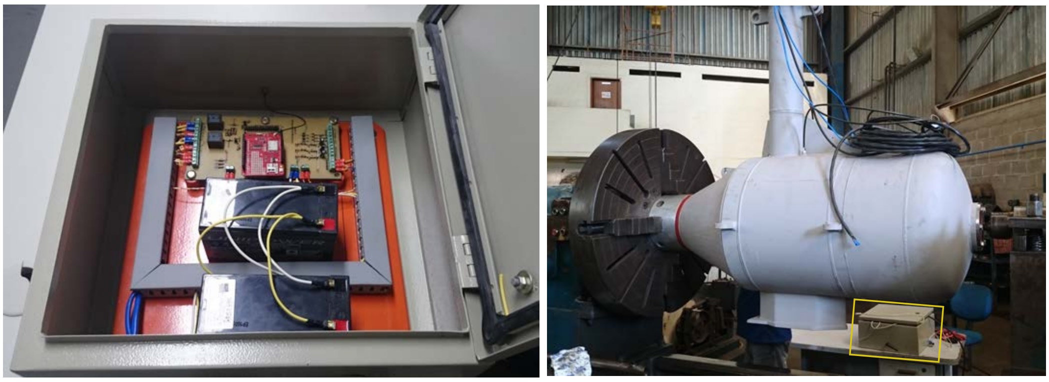
Thus, if SacadBR is configured correctly, the data encapsulated in the Modbus IP protocol and sent by the instrumentation platform via WiFi, is then received by ScadaBR, saved in the MySQL database, and presented to an observer on the tomcat web server. Another handy feature of ScadaBR is that one can publish the received data to IoT websites. Thingspeak.com was tested as IoT server, and if desired, the data could be publicly published on the internet, allowing anyone to observe the turbine's functionality.

4.4 Prototypes and tests

After finishing the detailed design phase, a first prototype was made. This prototype was a good first experience in making printed circuit boards (PCB). As outlined in the detailed design, it was equipped with an Arduino Mega 2560, a CC3000 WiFi shield, 3 inputs for temperature sensors, 1 input for a water detection sensor, 2 inputs for rotational speed sensors, 3 inputs for the x, y, and z axes of an accelerometer, three inputs for current sensors, 3 inputs for voltage sensors, and a redundant analog input. Figure 69 shows the PCB design of prototype version 0.5. The sensors were tested and all functioning correctly.

Note that the safety actuating mechanical relays were not present on the first prototype. In the second prototype however, the relays were embedded in the PCB. The microcontroller, WiFi shield, and inputs remained unchanged. Figure 70 in appendix C shows the PCB design of prototype version 1.0. This prototype was installed in a metal instrumentation box (see figure 35a)

The instrumentation box was mounted in such a way that a larger PCB could be installed to monitor all three turbines. Prototype V1.0 was tested on a turbine mounted on- and driven by a lathe as illustrated in figure 35b.



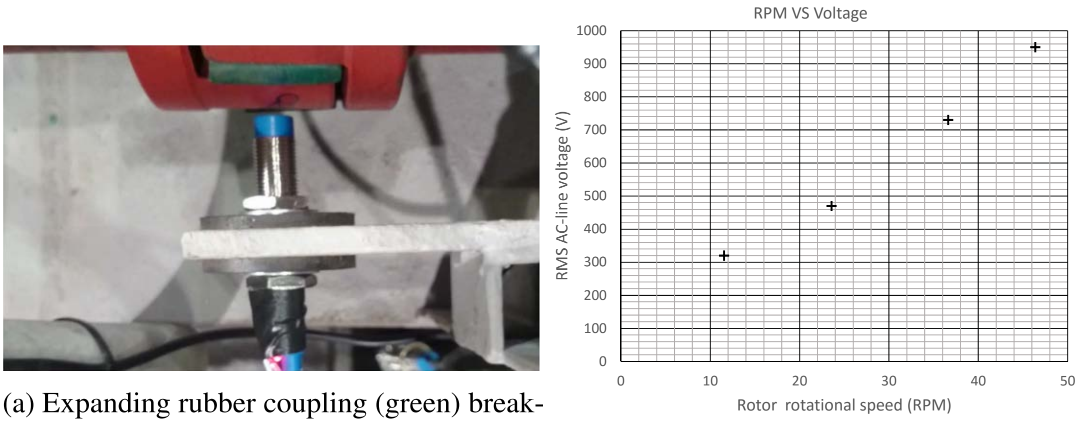
(a) The second instrumentation prototype installed in the instrumentation box

(b) Turbine testing mounted on and driven by a lathe

Figure 35 – Instrumentation platform testing on lathe

During the tests, various problems were discovered. One of the problems was that in no-load situations, the voltage surpassed 1000 V in some cases. Figure 36b shows the no-load RMS AC-line voltage output at various rotational velocities. This problem was solved by designing a high-voltage, selective resolution, voltage divider with a zener diode, that limits the voltage on the analog input of the Arduino to 5 v (shown in figure 73 of appendix F). As mentioned in the previous subsection, it has a maximum rated voltage measurement of 1000VDC with a relay to double the resolution at voltages lower than 500VDC.

Another issue, was the flexible coupling, between the rotor and the generator, which destroyed the proximity/RPM sensor during the first tests, due to an unintentional-excessive expansion of the rubber coupling (see figure 36a). This was either due to an under dimensioned flexible coupling, or an abnormally high torque, generated by the lathe. The proximity sensor was installed near this flexible coupling because it served as the cogwheel necessary to make the velocity calculation possible, as described in the last subsection. The issue of the breaking proximity sensor was resolved by relocating the RPM sensor to a different area of the axle, where there is no expanding material and a purposely designed metal cogwheel.



(a) Expanding rubber coupling (green) breaking tip of proximity sensor (blue)

(b) No-load RMS AC-line voltage output VS turbine rotational velocity

Figure 36 – Results of tests on the lathe

Another problem that emerged from the tests, was that the WiFi stability was very poor, partially due to the CC3000 shield's small WiFi signal radius, and in a certain quantity also as a result of the upper level "HTTP protocol" which was used to send the measured data to the server. Therefore, the more stable "ModBus IP protocol" and an ESP8266-07 WiFi module were tested. The change to the Modbus IP protocol showed a remarkable improvement in stability and, combined with the ESP8266, had a maximum tested data transmission range of about 120m in open air. Logically the "HTTP protocol" was replaced with a "ModBus IP protocol", and the CC3000 was swapped for an ESP8266-07. Thus, an ESP8266 shield was made, complete with a Real Time Clock (RTC) and an SD-card slot, both of which were not included in the previous prototypes. See figure 68 for the test ESP8266-07 shield.

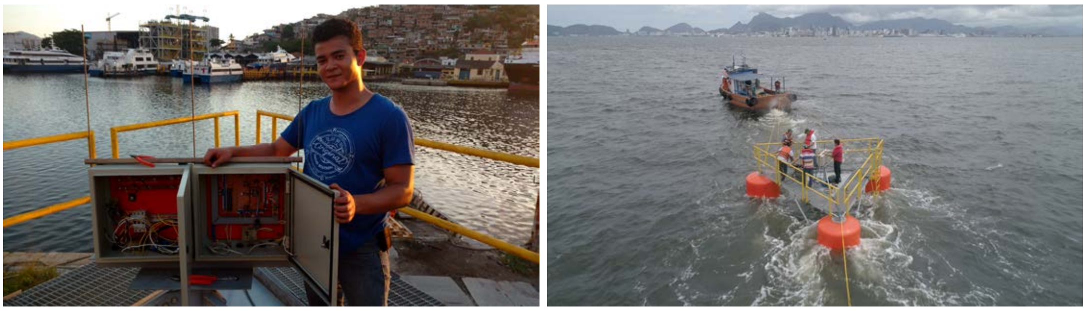
The new ESP8266 WiFi shield showed improved stability, and incorporated the SD-card and RTC which serve as a backup in case of a failed WiFi connection. The latest prototype (V 2.0), embedded all the ESP8266 shield functions on-board, so that no shield is required. Another development for V 2.0, was the incorporation of all three turbines on the PCB. This included an input for the river water temperature (to calculate the water density), a battery voltage indicator, a river water velocity sensor, and a secondary serial port. The extra serial port was added to facilitate an eventual test using fiber optics TTL connection. A fiber optics TTL connection directly with the server should be much faster and much more stable than the use of WiFi, if favorable for the turbine installation site. The pin reference for V 2.0 is presented in appendix B.

Finally, because the Arduino Mega only allows a maximum current draw of 40 mA and considering the number of sensors and relays monitored and controlled by the Arduino Mega on the PCB, this improved prototype contained voltage regulators. These supply a stable voltage and sufficient current for all the sensors, relays, and the ESP8266 which has an operating current of 80 mA.

As for the calibration of the sensors, all sensors were tested and calibrated. The calibration curves can be found in appendix I. First off, the amplified river water velocity sensor was calibrated using the output of its handheld meter as a reference. The calibration curve is shown in figure 75, and for the amplifier PCB see figure 74 in appendix G). Secondly three of the four temperature sensors were calibrated using a temperature calibration bath. The resulting calibration curves can be seen in figure 76. It can be concluded that the digital temperature sensors are very accurate, as all three sensors showed almost identical curves.

The current transformer was also tested and calibrated using a multimeter with a current clamp. The results of the calibration are presented in figure 77. And lastly, the voltage divider was tested at various voltages, and with the relay on and off. The calibration of these sensors is shown in figure 78.

Figure 37a shows prototype V 2.0 in the box on the right. It was installed for tests, on the 13th of March 2017, of a HK-10 turbine in Niterói (see figure 37b) and the results of the tests related to the instrumentation platform will be presented in section 7.1. But first, the proposed MPPT control will be discussed in the next chapter.



(a) Instrumentation platform installed on HK^3-10 array

(b) Test run of a HK-10 turbine with the instrumentation platform

Figure 37 – Instrumentation testing on water

5 Control of hydrokinetic turbines

As introduced in section 2.3, this chapter will be dedicated to discussing various approaches for the optimal control of variable speed - fixed pitch turbines starting with the more conventional variable speed - fixed pitch turbine control strategies.

5.1 Non-MPPT variable speed - fixed pitch turbine control

One approach for optimal control of variable speed - fixed pitch turbines uses a setpoint from the water velocity information to control the rotational speed of the rotor shaft. This approach is only suitable if the optimal value of the tip speed ratio (λ_{opt}), and the current river velocity (U_∞) are known. The optimal rotational velocity (ω_{opt}) is that where:

$$\lambda(t) = \lambda_{opt} \Rightarrow \omega_{opt}(t) = \frac{\lambda_{opt}}{r} U_\infty(t) \tag{5.1}$$

A drawback to this approach is that λ_{opt} should be known, which implies that a mapping of the C_P - λ curve must be done.

Another approach for controlling variable speed - fixed pitch turbines is using a setpoint from the shaft rotational speed information to actively control the turbine. This method is only suitable if λ_{opt} and the maximum power coefficient ($C_{p\ max} = C_P(\lambda_{opt})$) are known, which again implies that a mapping of the C_P - λ curve must be done. By combining equations 2.1, 2.3, and 2.4 follows:

$$P = \frac{1}{2} C_P \rho \pi r^2 U_\infty^3 = \frac{1}{2} \frac{C_P}{\lambda^3} \rho \pi r \omega^3 \tag{5.2}$$

Inserting $\lambda = \lambda_{opt}$ and $C_P = C_{Pmax}$ in equation 5.2 one obtains the power reference:

$$P_{opt} = P_{ref} = K \cdot \omega_{opt}^3 \tag{5.3}$$

Using equations 2.1, 2.4, and 5.1 we can deduce that:

$$K = \frac{1}{2} \frac{C_{Pmax}}{\lambda_{opt}^3} \rho \pi r^5 \tag{5.4}$$

This approach supposes an active power control loop is being used whose reference power is determined using equation 5.3 and is widely employed for medium and high-power turbines. Both of the above methods use an Optimal Regimes Characteristic (ORC) tracking control loop. And various control laws can be employed. In most cases, PI or PID control are favored.

5.2 Maximum Power Point Tracking (MPPT)

A relatively new approach for controlling variable speed - fixed pitch turbines is Maximum Power Point Tracking (MPPT). This approach does not require λ and $C_{p\ max}$ to be known. MPPT is a concept which originated from, and has been extensively studied for photovoltaic (PV) systems. PV systems have current-voltage and power-voltage curves as shown in figure 38 (Adapted from (MCNULTY; HORTA; PLAISIME, 2006)). As evident, every irradiance and temperature has an operating point where the power is at the maximum, or MPP. At any given time, the system's operation point is on a similar I-V and P-V curve. The operation point depends on the electrical load applied to the system and the ambient conditions. In order to extract the maximum power from the PV system, it is important to enforce the system to operate at the operation point corresponding to the MPP. This point corresponds to the knee of the I-V curve or the peak of the P-V curve at its current ambient conditions. There are various ways to achieve a system operating at MPP. The simplest is to regulate the system's voltage to the voltage of the MPP. Another way is to regulate the current to the current of the MPP.

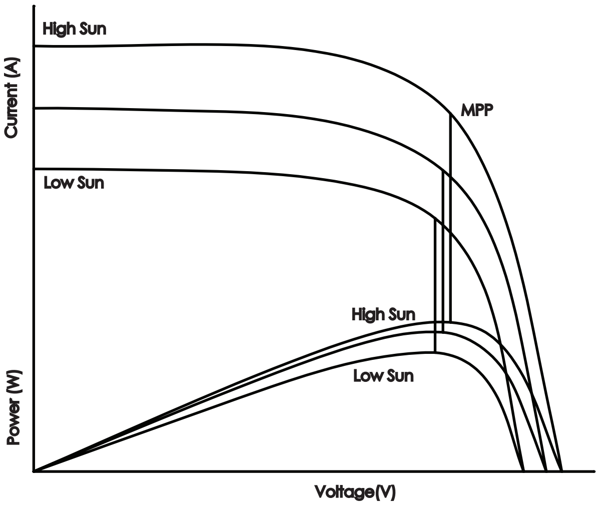


Figure 38 – I-V and P-V curves for PV systems.

Given that the I-V and P-V curves depend on the ambient conditions, such as irradiance and temperature, the MPP will fluctuate with changing ambient conditions. In order to regulate the system's voltage or current to the voltage or current of the MPP, the maximum power point should be tracked. This process is called Maximum Power Point Tracking (MPPT). MPPT devices track the MPP using one or more MPPT algorithms.

The two main groups of MPPT algorithms are the indirect MPPT and the direct, or hill climb MPPT method. The first one uses simple measurements to make assumptions and periodic estimations of the MPPT. The direct MPPT methods, measure current, voltage, or power and have a faster response than the indirect methods.

5.2.1 MPPT for PV systems

As stated above, the concept of MPPT originated from PV systems. This section will discuss a few algorithms for PV systems, starting with the indirect MPPT algorithms.

One of the indirect MPPT algorithms is the Fixed voltage method. This method is based on adjusting the operational voltage of the system only according to the season. This method is less efficient, as there are irradiance fluctuations independent of the season on a daily basis. Another indirect MPPT method is the Fractional open circuit voltage method. This method exploits the fact that the MPP is located at a certain distance from the open circuit voltage. Therefore the operation voltage is regulated to $V_{MPP} = k \times V_{OC}$. Where k is a constant smaller than 1 (in general 0.7-0.8) and approximates the MPP, as it is a ratio between MPP voltage (V_{MPP}) and the open circuit voltage (V_{OC}). This ratio diverges very little with changing irradiance. A disadvantage of this method is that the load on the PV system has to be temporarily and repeatedly disconnected in order to measure the V_{OC}, resulting in a loss of power extraction.

The more involved of the two main MPPT algorithms, the direct MPPT algorithms, measure current, voltage, or power and have a faster response than the indirect methods.

One of the direct MPPT methods is the Perturb and observe method. This method exploits the fact that the P-V curve is always increasing on the left side of the MPP, and decreasing on the right side of the MPP. Changing the load on the system will result in an increase or decrease in the power. Periodically increasing or decreasing the load in the direction of an increasing power, results in a high power extraction, but never the maximum, because the algorithm can not remain on the MPP, but hovers around it. Another disadvantage of this method is that it can fail to converge in rapidly fluctuating ambient conditions.

The incremental conductance method is another hill climb MPPT algorithm. This method exploits the fact that the MPP corresponds to the knee of the I-V curve and that the slope of the P-V curve is zero at the MPP. Therefore, $\frac{dP}{dV}$ can be rewritten as:

$$\frac{dP}{dV} = \frac{d(I \times V)}{dV} = I + \frac{V \times dI}{dV}$$

And since the slope is $\frac{dP}{dV} = 0$, $\frac{\Delta I}{\Delta V} = -\frac{I}{V}$. As the conductance of the system is $\frac{I}{V}$, the system's operation point relative to the MPP is:

$$\frac{\Delta I}{\Delta V} = -\frac{I}{V} \; at \; MPP$$

$$\frac{\Delta I}{\Delta V} > -\frac{I}{V} \; to \; the \; left \; of \; MPP$$

$$\frac{\Delta I}{\Delta V} < -\frac{I}{V} \; to \; the \; right \; of \; MPP$$

At every iteration the algorithm imposes a voltage to the system, measures the resulting incremental change in conductance, compares it with the instantaneous conductance, and thereby discovering the operation point relative to the MPP. This method can be more efficient than the previous algorithms because it can remain on the MPP, and does not need to hover around it in steady state conditions. High sampling frequencies make this method less susceptible to fluctuating ambient conditions. The drawbacks of this method are that in highly fluctuating, and partial shading conditions the algorithm is unlikely to converge towards the MPP. Ngan and Tan (2011) discusses an algorithm suitable for the aforementioned conditions.

5.2.2 MPPT for wind energy systems

In the last decade the literature concerning MPPT for wind energy systems has advanced. Various MPPT control methods have been described, simulated, and applied to wind energy conversion systems. MPPT has been implemented in various wind turbines (HUA; GENG, 2006; HAQUE; NEGNEVITSKY; MUTTAQI, 2010; AUBRÉE et al., 2016), and has recently also been applied to HKT's (ZHOU, 2012) (MOFEI, 2014). HKT's and wind turbines basically have the same working principle. What differentiates the hydro from the wind turbine is that the hydro kinetic turbine; extracts power from a different fluid, has lower tip speed ratio ranges, cavitation limits and is exposed to a more harsh environment (GINTER; PIEPER, 2009). There is a large amount of literature describing the control of wind turbines and relatively few papers on the control of hydro-kinetic turbines. Munteanu et al. (2008) describes various conventional ways of controlling wind turbines including MPPT.

5.2.3 MPPT for hydrokinetic turbines

For HKT's different MPPT methods have been encountered in literature. Two algorithms will be modeled, simulated and compared with the designed MPPT algorithm, which will be featured after discussing the benchmark algorithms in the following sections.

5.3 Algorithm by ZHOU

Zhou (2012) utilized a MPPT method with a relatively simple algorithm (see figure 39). It is based on the hill climb method, which simply evaluates whether the current working point is to the left or to the right of the maximum power point (MPP), and modifies the duty cycle of the PWM controller, with a fixed step, towards the MPP.

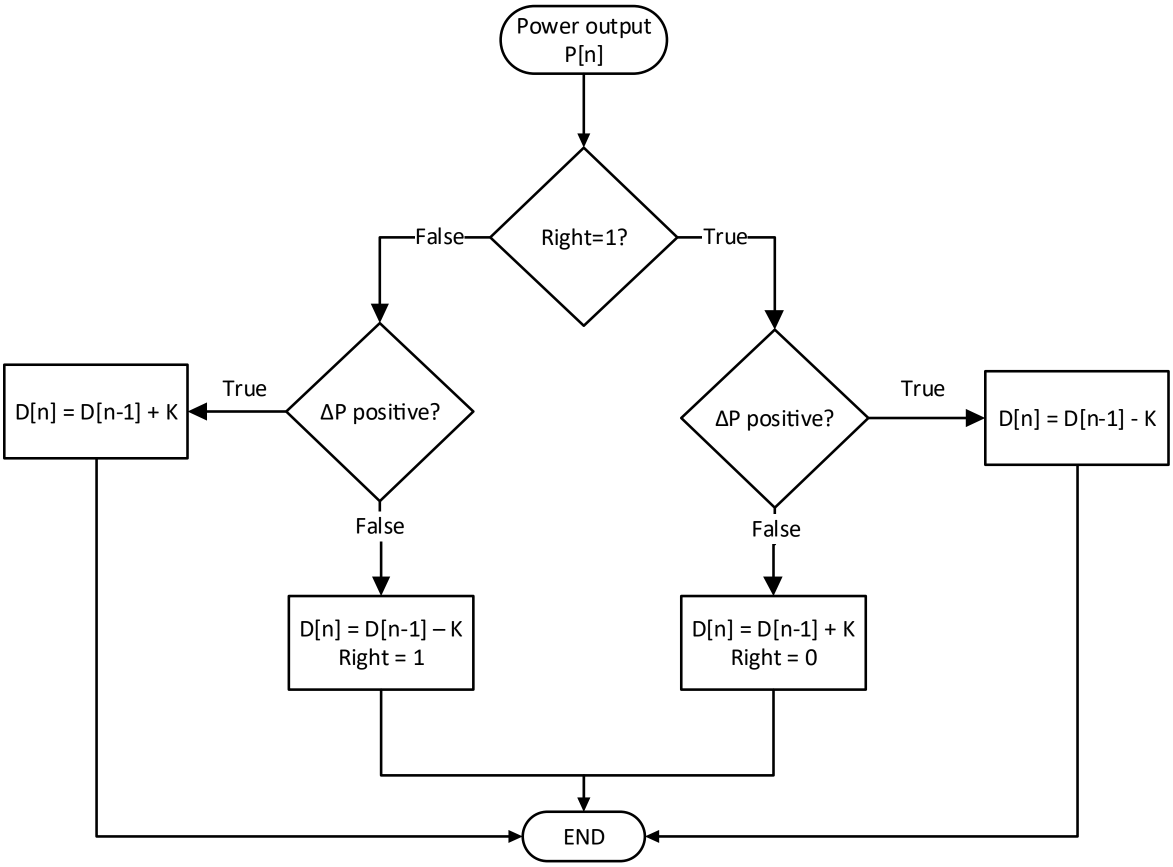


Figure 39 – Flowchart of MPPT algorithm by Zhou (2012).

5.4 Algorithm by MOFEI

Mofei (2014) presented a more elaborate algorithm (see figure 40), which is also based on the hill climb method, and evaluates whether the current working point is to the left, to the right, or on the MPP. Therefore, this algorithm allows the turbine to stay on or near the MPP instead of oscillating around it in constant velocity conditions.

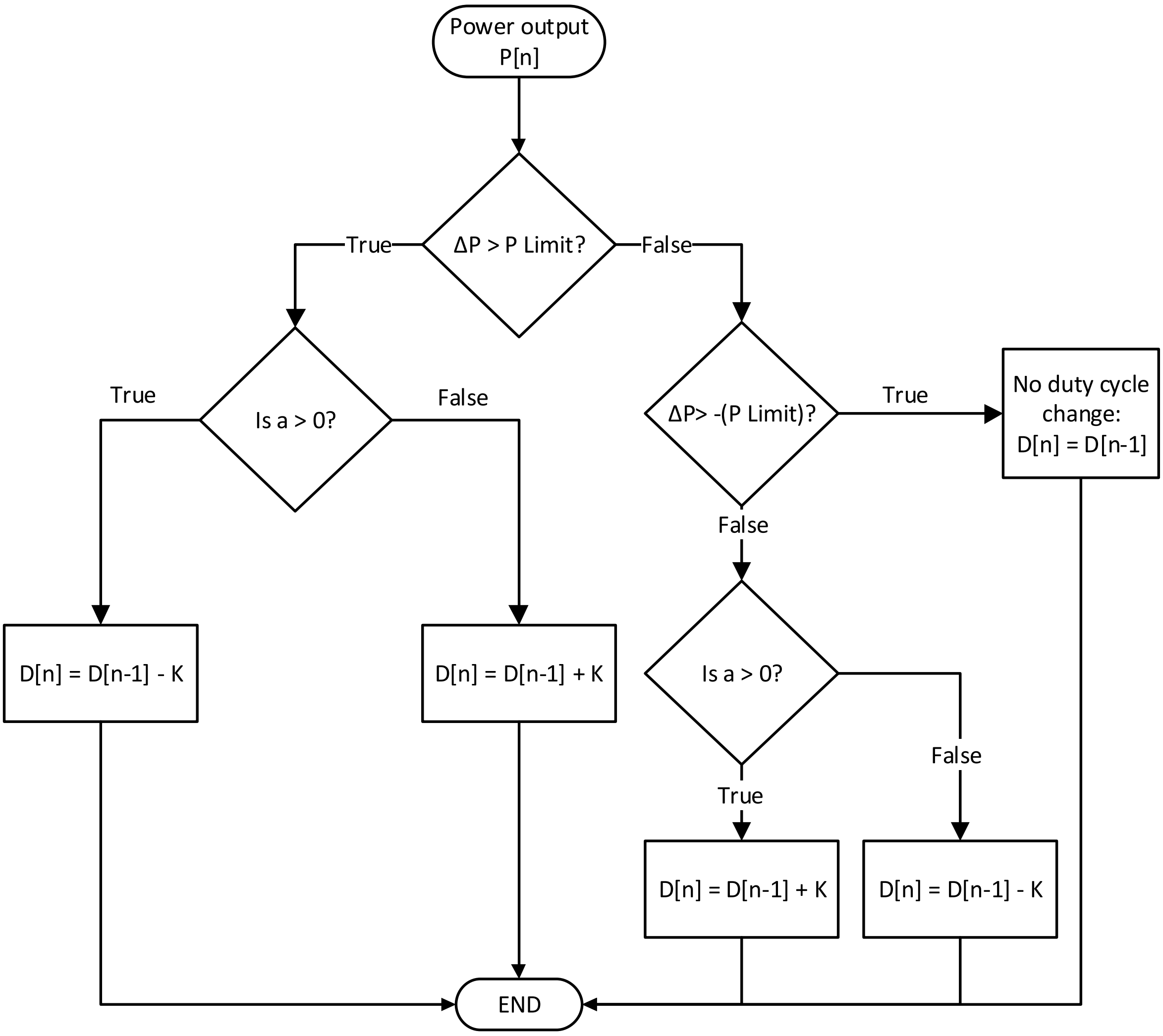


Figure 40 – Flowchart of MPPT algorithm by Mofei (2014).

Noticeably, both of the previously mentioned algorithms use one or two fixed step sizes for the duty cycle. Specifically, the algorithm by MOFEI distinguishes between a "large step" and a "small step". This dissertation will investigate the use of an "adaptive-step-size" MPPT algorithm for HKT’s.

5.5 Proposed MPPT algorithm

The algorithm is presented in figure 42 and is based on the hill climb method and tracks the effects of the river velocity on the voltage, current, and power. As any other hill limb method, the slope of the power - voltage curve is used to determine the direction of the next duty cycle change if no river velocity alterations are detected. A positive slope implies that the operating point is to the left of the MPP, and a negative slope vice versa (see figure 41). With a constant electrical load, altering river velocities affect the power, voltage, and current change in a various combinations. Some of the combinations can also occur at constant river velocities, however, two situations that guarantee a changing river velocity are a simultaneous increase or decrease of both the power, voltage, and current. If all three deltas are positive, the river velocity is increasing, as opposed to a decreasing river velocity when all three deltas are negative.

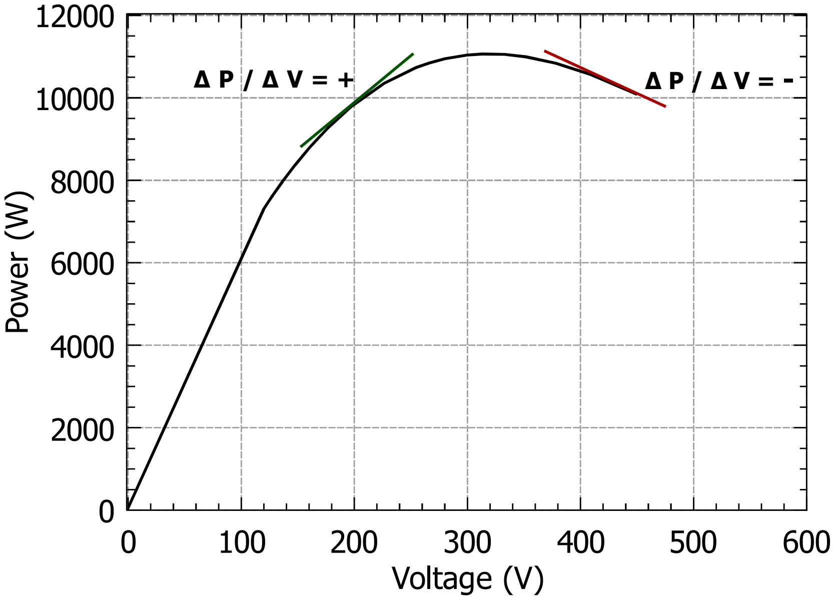


Figure 41 – Slope on the P-V curve

Furthermore, an adaptive step size is implemented by multiplying a fixed maximum step size variable, presented as "a" in figure 42, by a limited ΔP. This should allow the algorithm to reach the MPP faster, and approximate the MPP more closely, due to the nature of the P-V curve. Namely, the slope, and hence the ΔP, on the P-V curve is smaller as it approaches the MPP, and larger as the distance increases. Finally, when the turbine reaches its operation point, and the ΔP reaches a limit, a deadzone is applied, and all values of ΔP smaller than the limit become zero. Hence, no duty cycle change is made until the ΔP surpasses the limit.

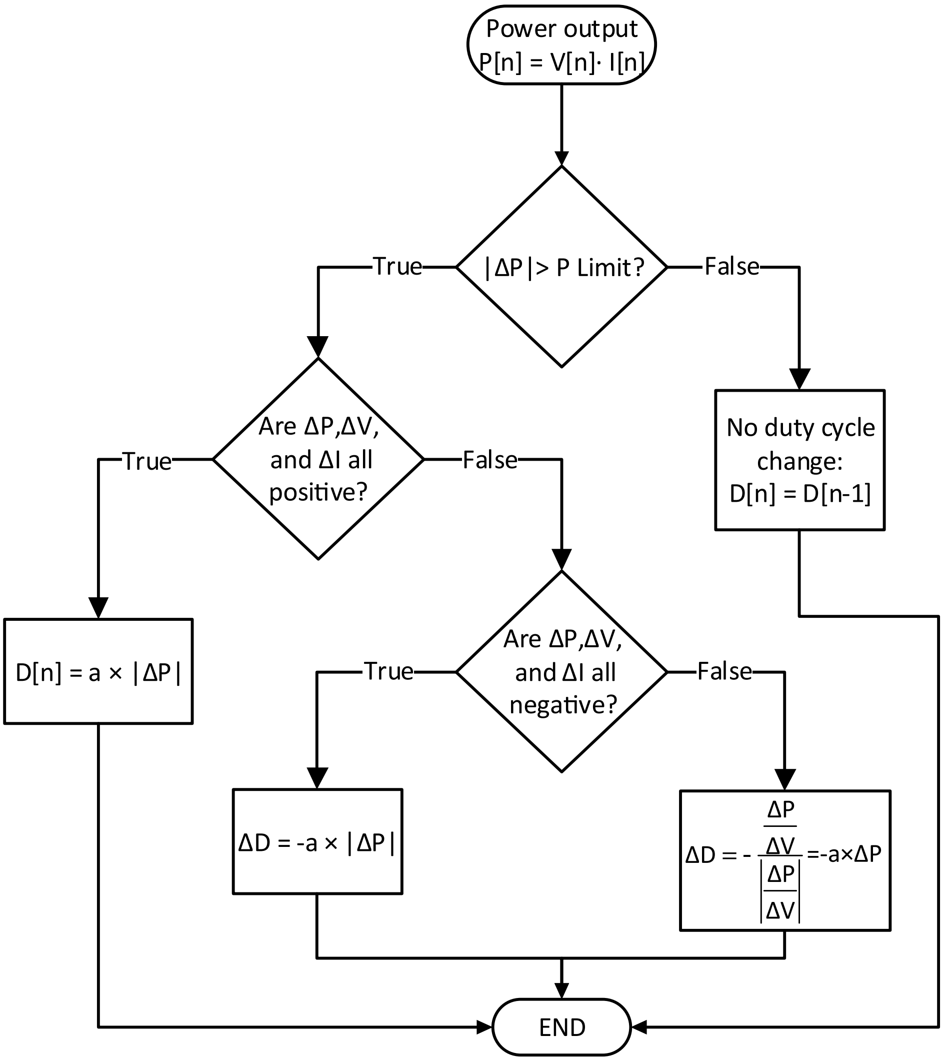


Figure 42 – Flowchart of MPPT algorithm by the author

6 Simulation modeling

A simulation of the proposed MPPT algorithm was done in SIMULINK. This chapter will describe the process of modeling the hydrokinetic turbine and the control algorithm. The starting conditions of this simulation phase were; a rotor with a known C_P curve was available, a three-dimensional computer model of the turbine was accessible, and the river velocity at the installation site was measured. These readily available data were quite convenient for the conception of the simulation, as will be evident in the following sections. Figure 43 shows a simplified block diagram of the intended turbine model.

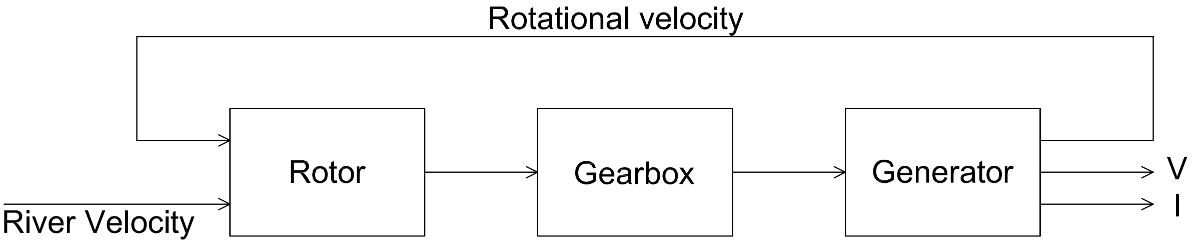


Figure 43 – Simplified block diagram of simulation model

6.1 Rotor modeling

As mentioned before, the power (P), harnessed by the rotor with a swept area, A, resulting from the river velocity (v) with a water density, ρ, and a power coefficient, C_P, is calculated with equation 2.1.

$$P = \frac{1}{2}\rho A v^3 C_P \tag{2.1}$$

The power coefficient C_P was simulated using ANSYS CFX by Brasil Jr. et al. (2016). The C_P VS λ curve was fitted using a fourth order polynomial.

The resulting fourth order polynomial was:

$$C_P = -0.0295\lambda^4 + 0.2169\lambda^3 - 0.6754\lambda^2 + 1.0122\lambda - 0.1958 \tag{6.1}$$

Where λ can be calculated using equation 2.4. One can subsequently calculate the power coefficient at any river- and rotor velocity, with a radius, r, using equation 6.1, which is a curve fit of the $C_P - \lambda$ curve.

$$\lambda = \frac{r\omega}{v} \tag{2.4}$$

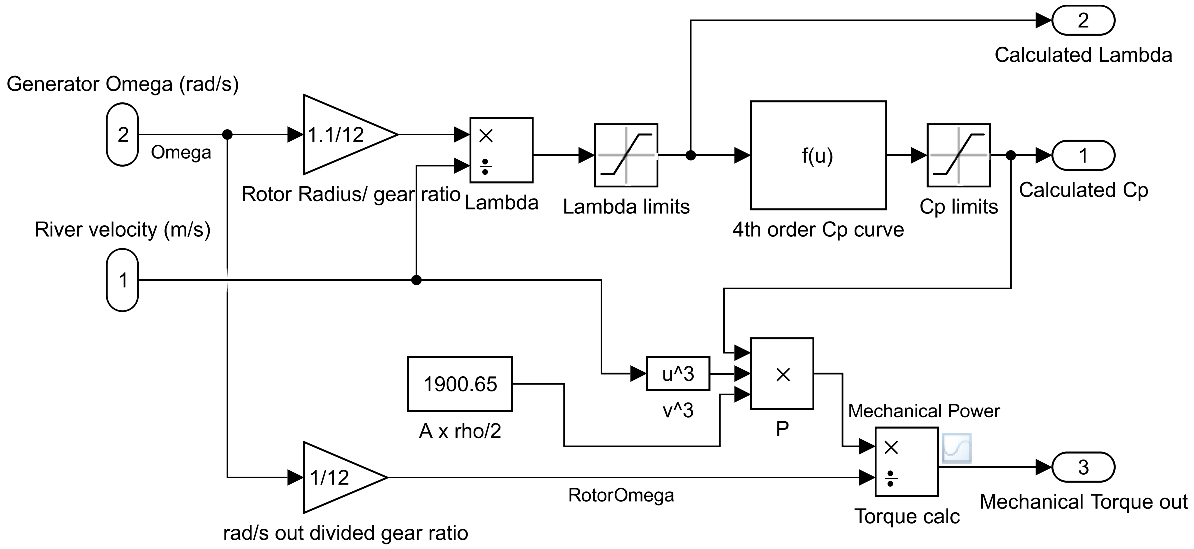


Figure 44 – Simulink model of the C_P curve and mechanical torque output.

The Simulink model of the C_P calculation and mechanical torque output is shown in figure 44 and was a subsystem of the complete model. To insure a realistic C_P, the λ and C_P were limited to their respective ranges in the $C_P - \lambda$ curve using a saturation block.

6.2 Gearbox modeling

The following subsystem that was modeled is the gearbox. The torque entering the gearbox was calculated with equation 6.2 as shown in figure 44.

$$\tau = \frac{P}{\omega} \tag{6.2}$$

The inertia of the rotor, I_R ($65.48kgm^2$), and the generator, I_G ($2.91kgm^2$), and a gearbox efficiency of 98%, were accounted for in the gearbox model. The available geometry and material properties of the rotor and the generator were imported into "Solidworks" where the "Solidworks motion" add-in was used to perform a motion analysis. The software simulated a rotation of the geometry with a constant α and, consequently, a constant torque. The resulting data allowed the inertia to be calculated using equation 6.3.

$$I = \frac{\tau}{\alpha} \tag{6.3}$$

The subsystem is shown in figure 45 and was mostly built from the "physical signals" library of Simulink. This was advantageous as it avoided algebraic loops. The ratio between the input- and output rotational velocity was 1 : 12.

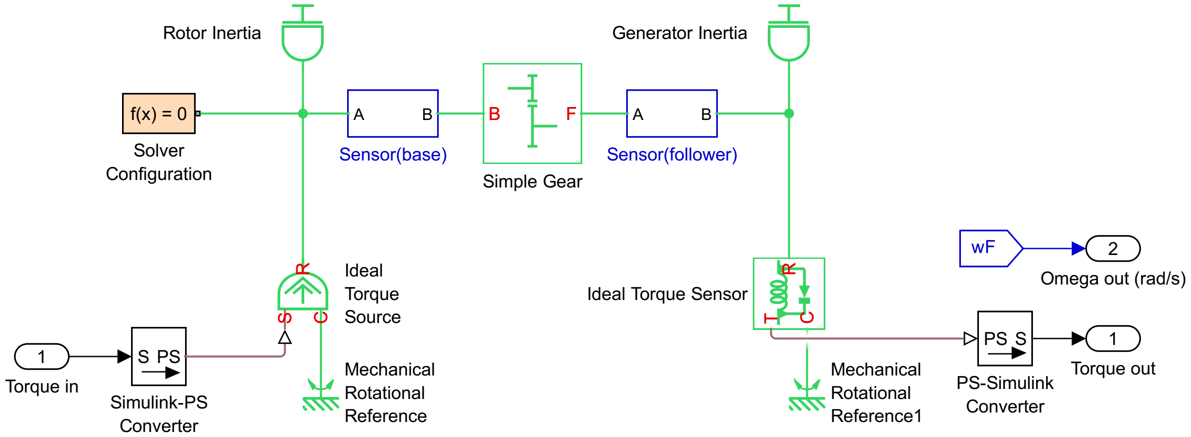


Figure 45 – Simulink physical signal gear box model.

6.3 Generator modeling

The Permanent Magnet Synchronous Generator (PMSG) was modeled using the PMSG machine block in the Simulink "powerlib". The rectifier, load, RLC filter, and IGBT were also modeled using this library. Figure 46 shows the generator subsystem. This step of the simulation presented some difficulties due to the unknown generator constants. Therefore, a best estimate was made to represent the generator as realistically as possible. The Insulated Gate Bipolar Transistor also presented some difficulties as the change in PWM duty cycle did not result in a linear change of its resistance. This problem was solved by deriving the "Duty cycle - Resistance" curve, and linearizing it using a the fitted power function: $1.023 \times x^{-1.031} - 0.03178$. The maximum resistance variance of the IGBT was limited from 1 to 30 Ω. Figure 49 illustrates the linearization subsystem.

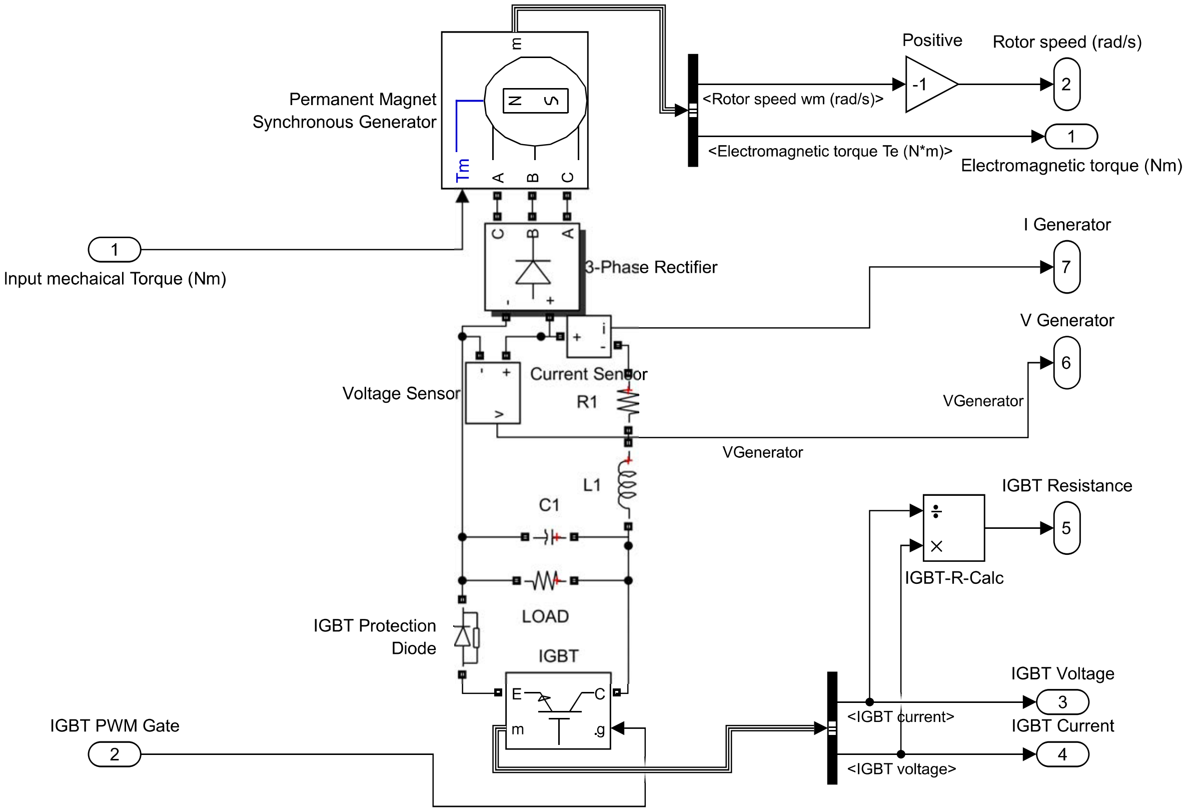


Figure 46 – Simulink PMSG subsystem.

The generator model concludes the hydrokinetic turbine modeling. Worth noting, is that it does not account for the inertia of the water surrounding the rotor. The effect of this variable on the model is expected to be minimum.

6.4 Signal processing

Signal filtering of the generator output was required for improved accuracy and was achieved in the "process power data" subsystem (See figure 47). Included in this subsystem, was an moving average filter which is shown in figure 48. Combined with a lowpass filter, this subsystem outputs the mean value of data captured during one iteration of the MPPT algorithm. And proved to be more stable than when solely using either a lowpass filter or a mean filter.

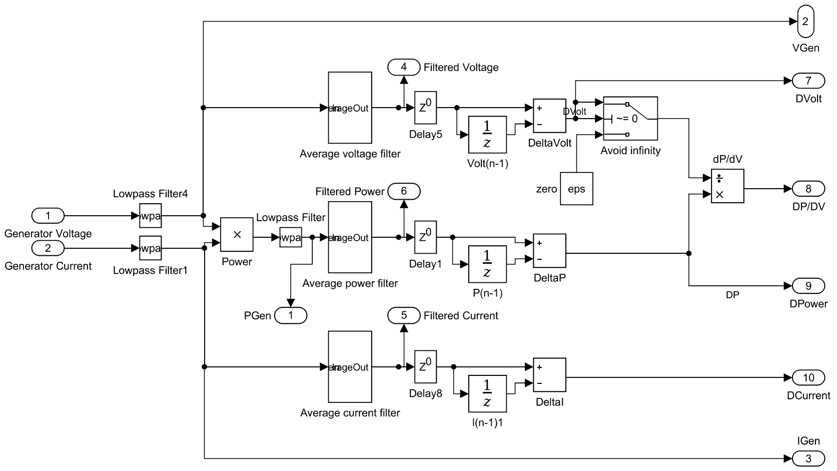


Figure 47 – Simulink filtering and processing power data subsystem.

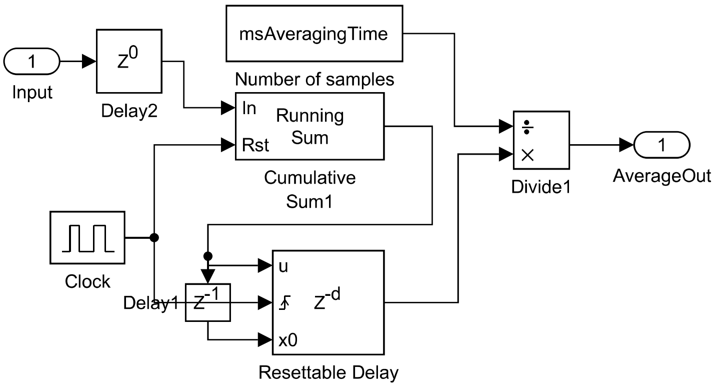


Figure 48 – Simulink fixed average filter subsystem.

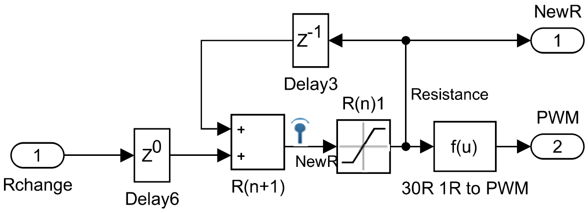


Figure 49 – Linearization of the resistance determined by the algorithm, and the necessary duty cycle

6.5 Modeling of the MPPT algorithms

As for the simulation of the control algorithms, see figures 51, 52, and 54. The "algorithm" subsystem, shown in figure 56, was the only subsystem that changed for the simulation of each algorithm. Excluding the parameters and variables of the algorithms, everything was maintained identical as illustrated in the block diagram of figure 50.

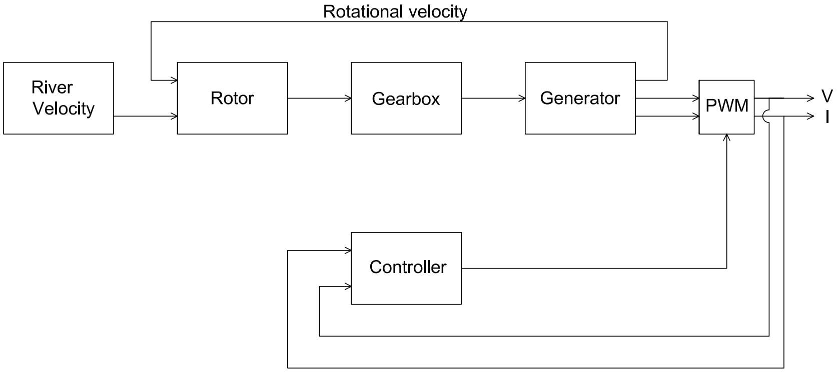


Figure 50 – Block diagram of the MPPT control model

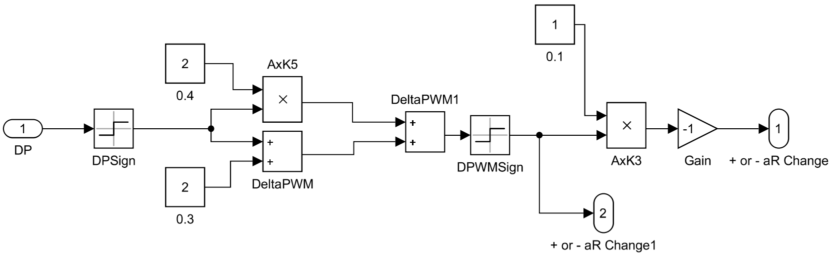


Figure 51 – Model of MPPT control algorithm by Zhou (2012)

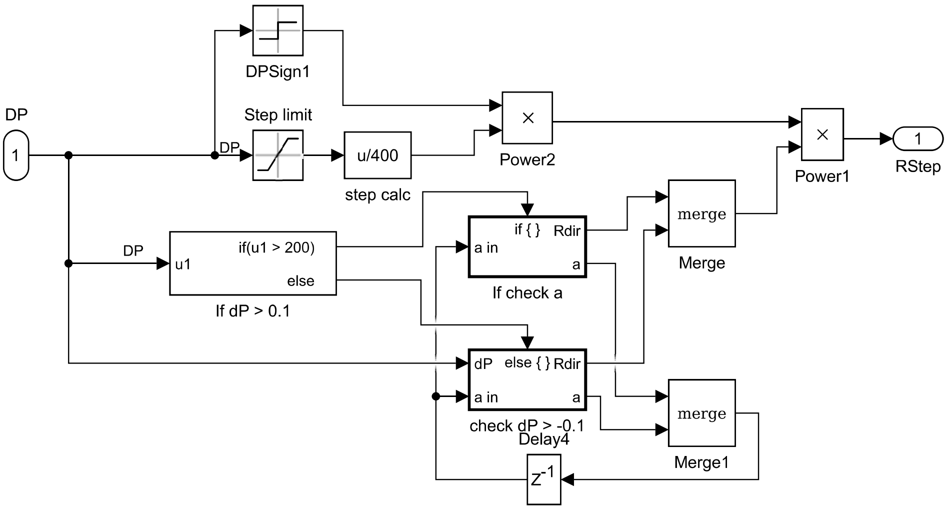


Figure 52 – Model of MPPT control algorithm by Mofei (2014)

6.6 Modeling of the optimal ω control

In order to evaluate the effectiveness of MPPT, a more conventional control system was modeled, simulated and compared with MPPT control. The optimal ω control discussed in section 5.1, and calculated with equation 5.1, was chosen as the benchmark. A simplified block diagram of the control system can be observed in figure 53.

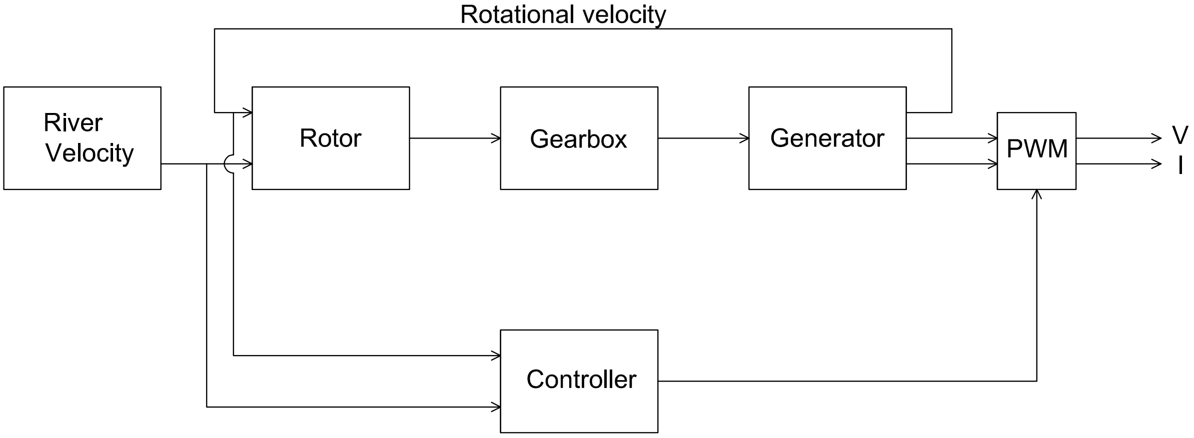


Figure 53 – Block diagram of the optimal ω control model

Simulating the optimal ω control with PID did not require the subsystem for duty cycle conversion nor the power data processing subsystem. But the latter subsystem was maintained in order to visualize the power data. Also, a subsystem was included to calculate the reference or optimal ω. See figure 57 for the complete model of the hydrokinetic turbine with optimal ω control and figure 55 for the optimal ω calculating subsystem. Furthermore, the PID constants were determined using Matlab's PIDtuner application and resulted in a K_d of 0.2558, a K_i of 4.1654, and a K_p of 11.0381. In the process of determining these constants, the plant's settling time was also noted, in consideration of the time between MPPT steps. A settling time of 1.11 seconds was identified for a 10% duty cycle change, and a settling time of 1.62 seconds for a 90% duty cycle change. Therefore, a fixed two-second delay was set between MPPT control iterations.

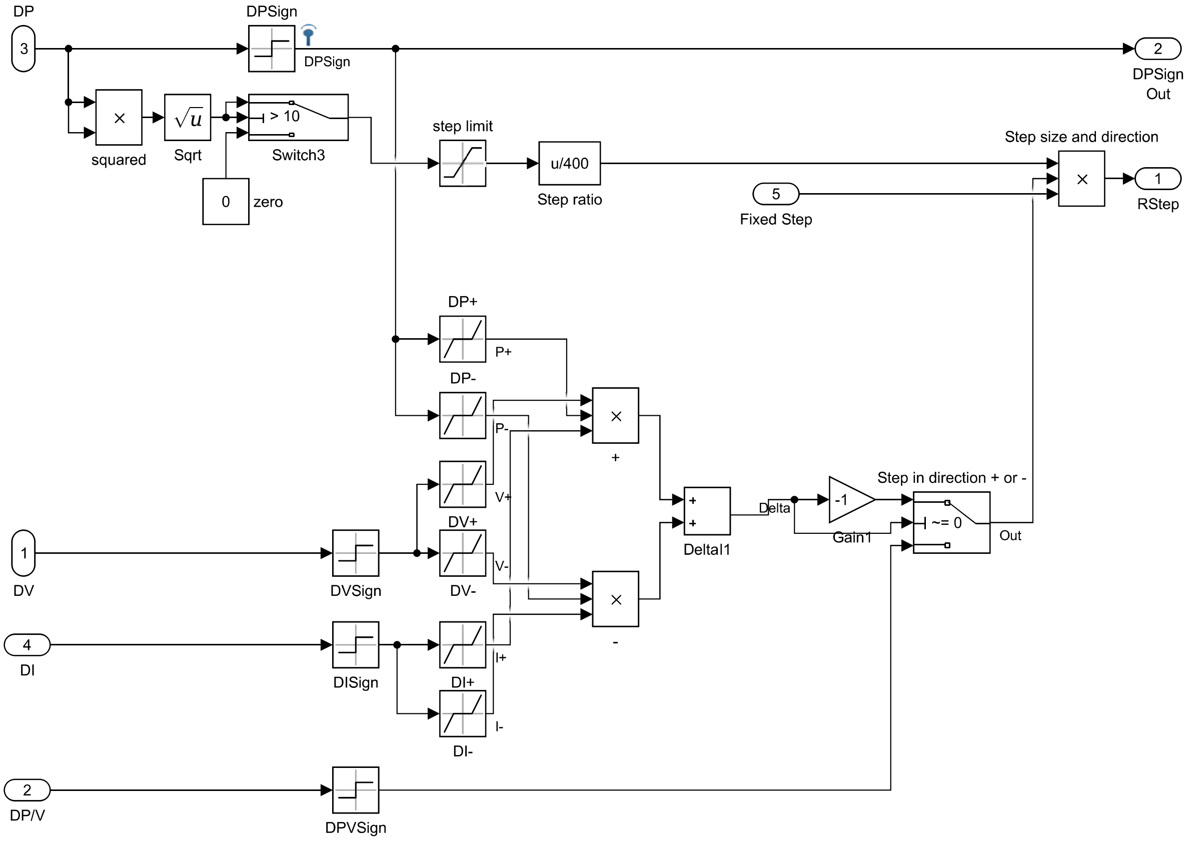


Figure 54 – Model of MPPT control algorithm by Mac Donald (2017)

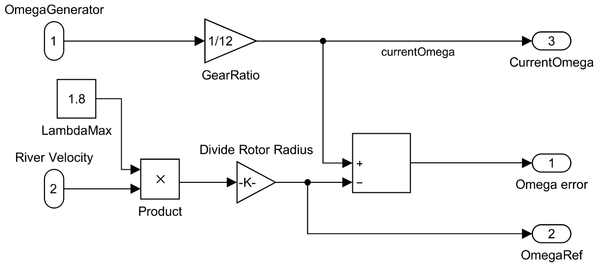


Figure 55 – Calculation of the reference ω

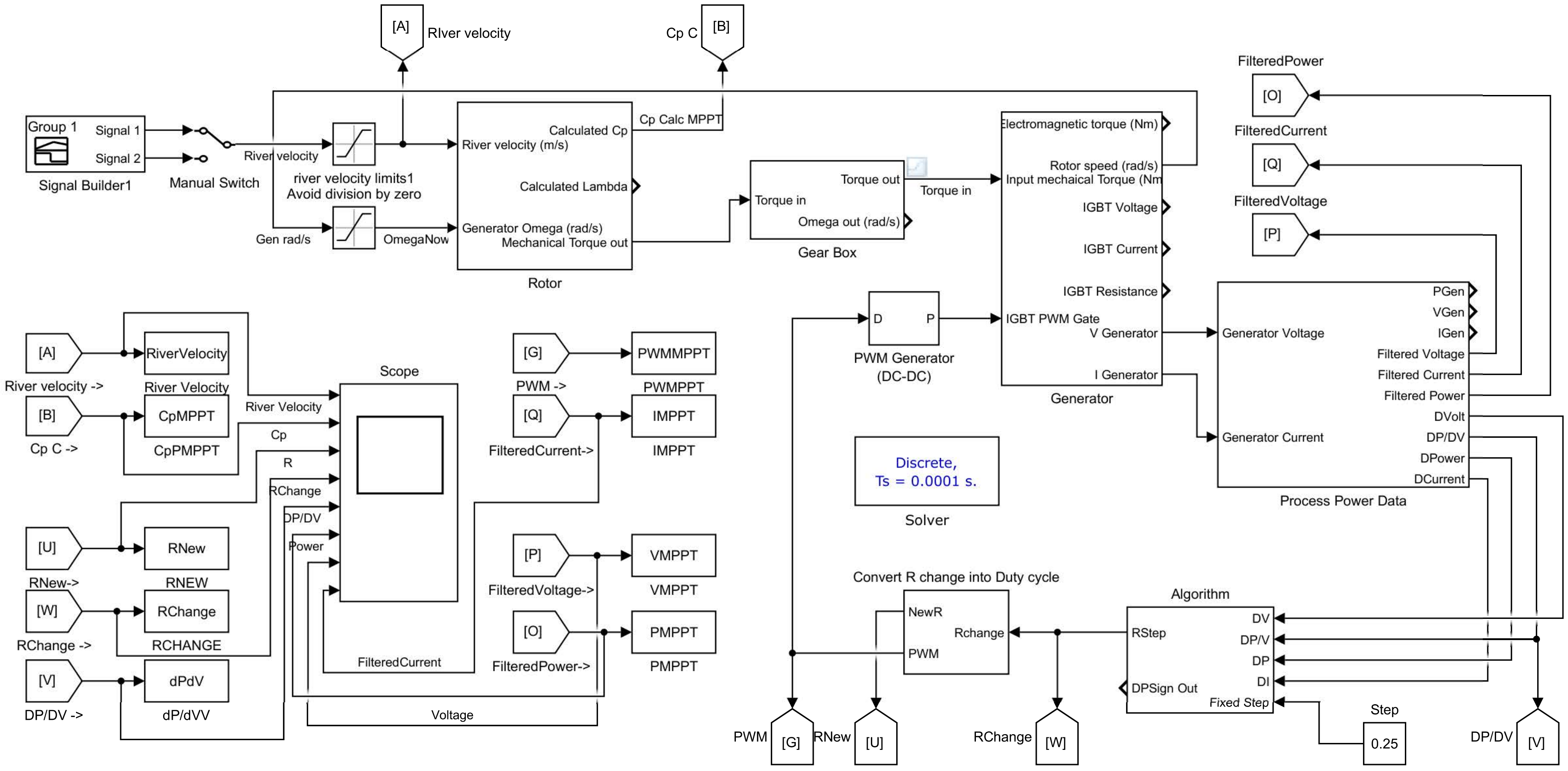


Figure 56 – Complete model of the hydrokinetic turbine with MPPT control

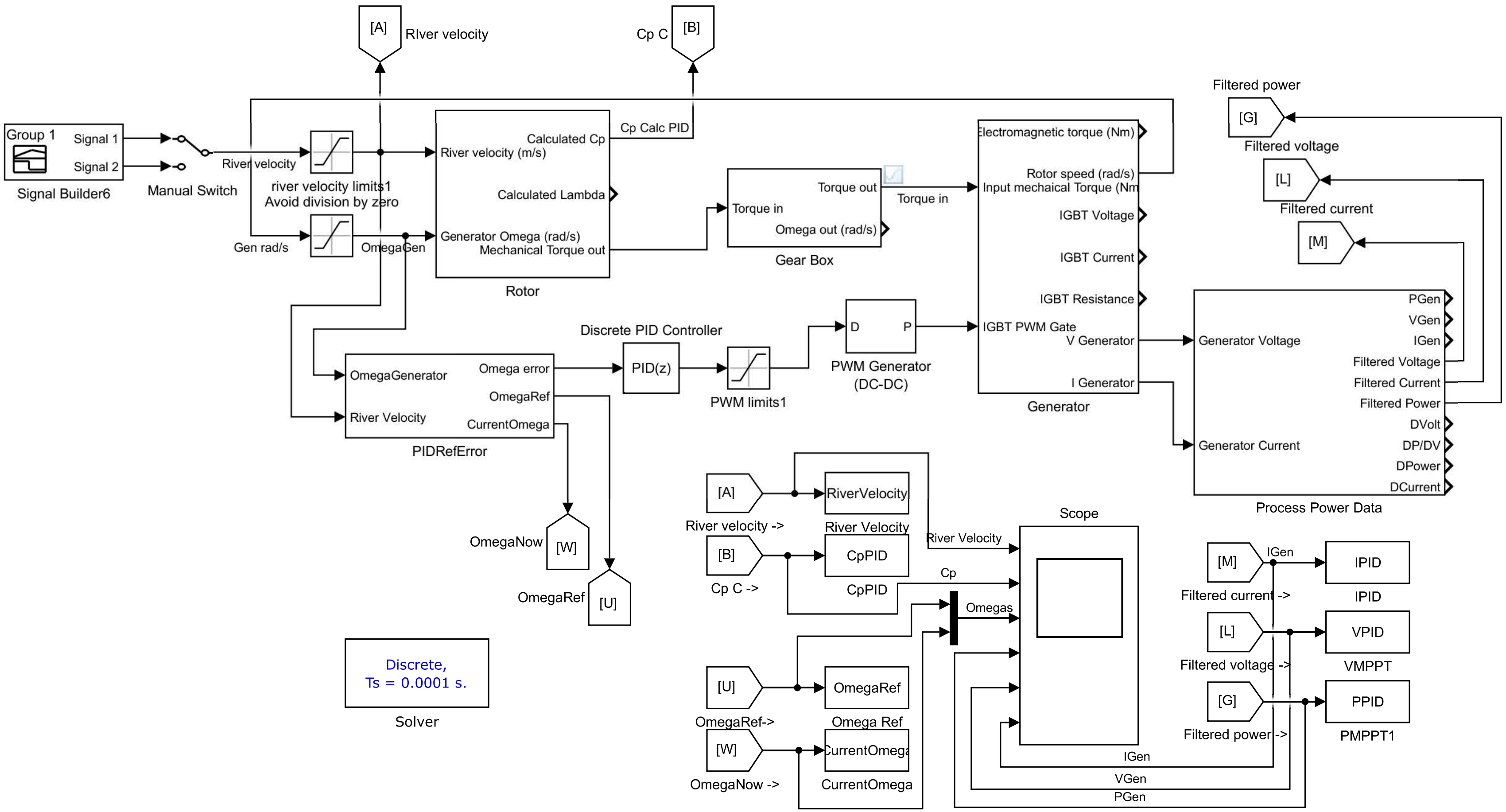


Figure 57 – Complete model of the hydrokinetic turbine optimal using ω control with PID

7 Results

7.1 Instrumentation results

The results of the instrumentation platform are presented here, and will discuss a field test of a HK-10 turbine. Two test runs conducted on the 13th of March 2017 in Niterói revealed various shortcomings and advantages of the instrumentation platform. The first was the robustness of the platform. During the trip to the test site, the "adjust" pin of an adjustable voltage regulator was disconnected from the PCB, resulting in a 12V output where 3.3V was desired, causing the destruction of SD-cards and the WiFi module. Fortunately, the replacement of the defective parts was possible for they were locally available and inexpensive, which was an objective for the design of the platform. Hence, it was repaired before the tests.

Another reliability issue was the malfunctioning amplifier. Unfortunately the amplifier could not be repaired before the tests, thereby failing to read the incoming water velocity with the instrumentation platform.

Regarding the sensors, a flaw was identified in the interfacing of the current and voltage sensors, as they were not suited nor calibrated for values lower than 10 A and lower than 110 V respectively. At lower values one could doubt the functionality of the sensors, their linearity, and the accuracy of the measurements (See Appendix I for the sensor calibrations).

Finally, the first test run revealed that the presentation of data was missing a filter, as the values of the voltage fluctuated heavily, and the registration of data was missing the raw input values and only registered treated data. Thanks to the flexibility of the system, the latter flaw was resolved quickly and in time for the second test run.

Positive results of the instrumentation platform were that the temperature, vibration, and rotational velocity sensors functioned without any faults or interference from electrical noise, as will be show in the next paragraph. Furthermore, the Modbus TCP data transfer operated normally, the collected data was successfully saved to the SD-Card with time and date, and the system proved to be adaptable.

The test data was collected in two test runs with a total time span of about 40 minutes and are shown in the subsequent figures. During the tests, the only form of control was a resistor bank that allowed the electrical resistance to be changed manually.

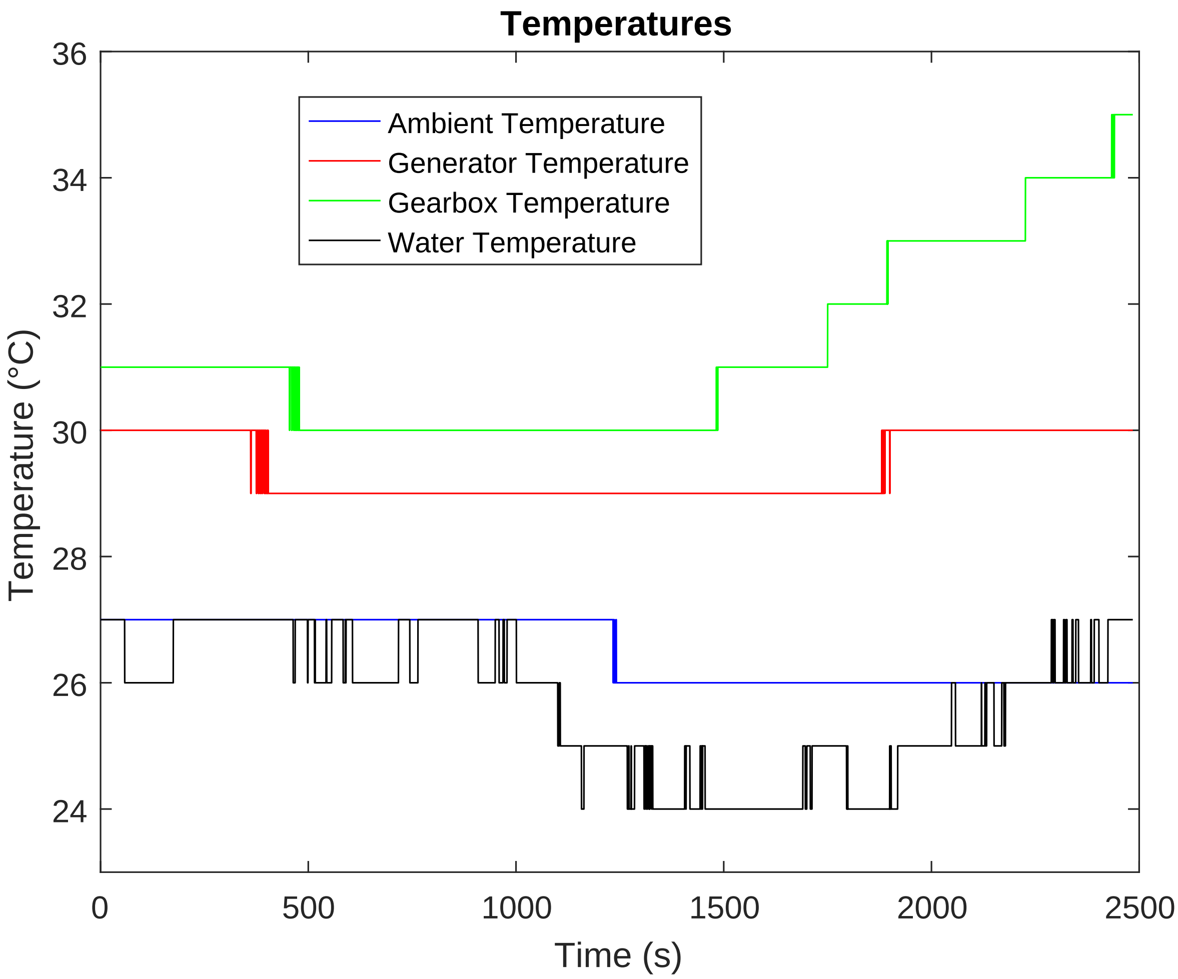


Figure 58 – Temperature data

In figure 58 the four measured temperatures are displayed and have shown no abnormalities. Noticeably, the temperature change is only registered in a full degree due to the "int" declaration of the variable in the programming. This could be ameliorated by multiplying the "int" saved to the SD card by 100 and after, divide by 100 to get a decimal value of the temperature.

Figure 59 shows the vibrations and the orientation of the turbine. This sensor also demonstrated normal operation.

The battery voltage data is shown in figure 60. Due to a low battery voltage after the first test run, the battery was changed. This change can be seen in the figure.

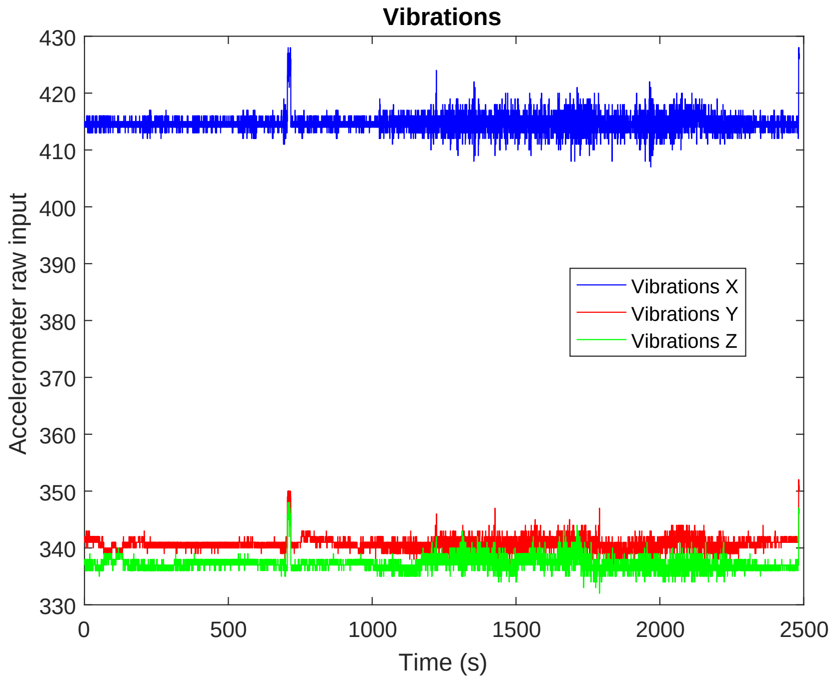


Figure 59 – Vibration data

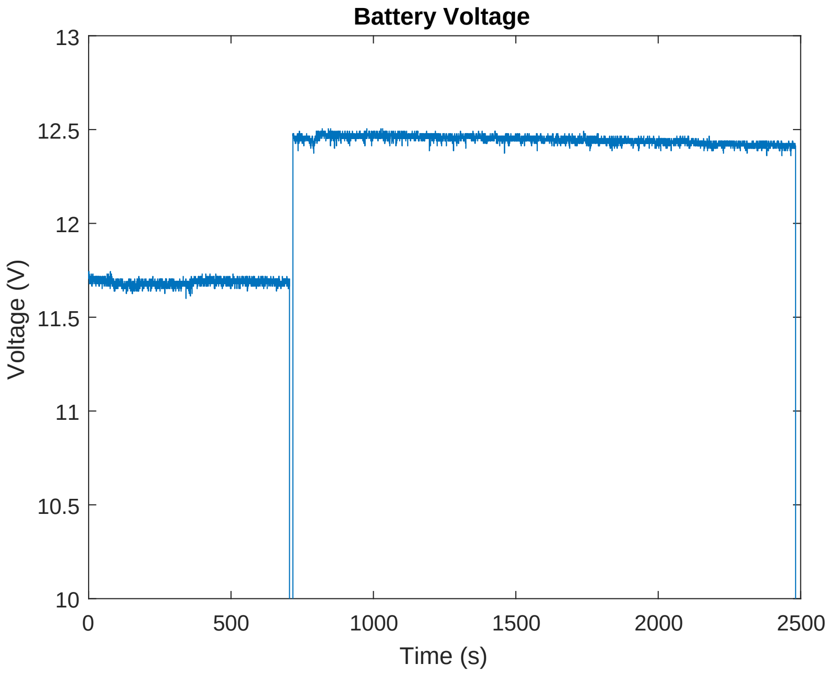


Figure 60 – Battery voltage

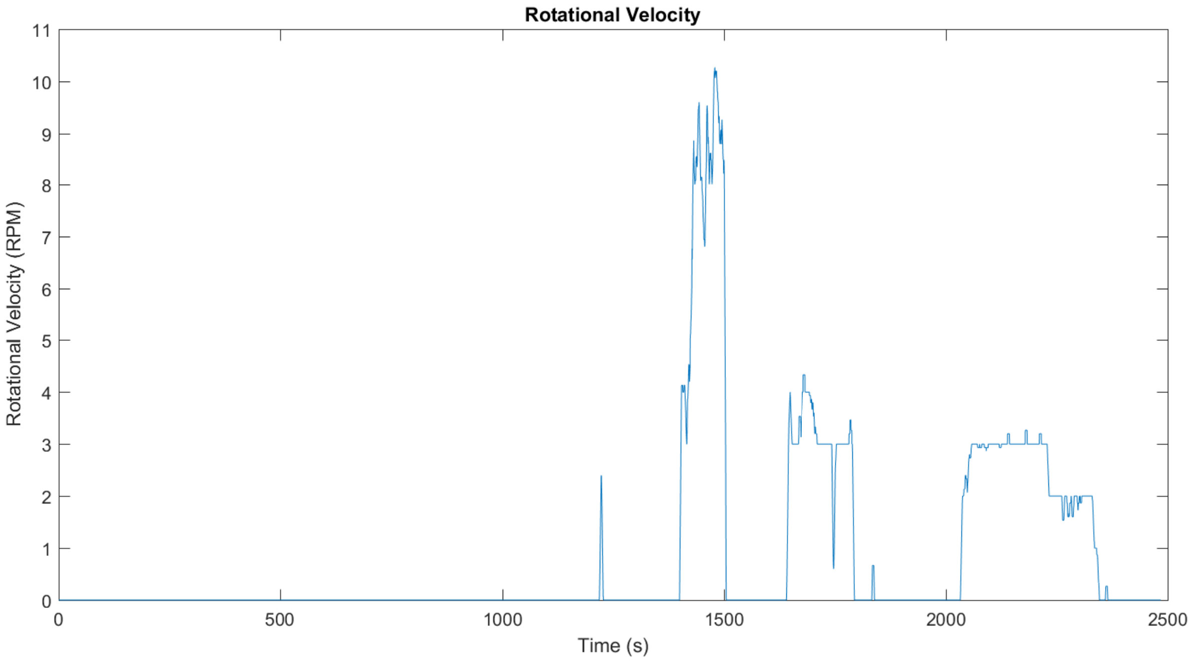


Figure 61 – Rotor rotational velocity data

Figure 61 presents the rotational velocity of the turbine. At first, the functionality of the sensor was doubted as the rotational velocity could hardly be zero at the water velocity achieved during the first test run, but afterwards, it was discovered that even the highest settable electrical resistance was too low for the generator to rotate, i.e., the electrical torque was too high for the rotor to rotate at the achieved water velocity.

The achieved three-phase power was calculated using the measured phase current and line voltage as follows:

$$phase\ current = line\ current \tag{7.1}$$

$$phase\ voltage = \frac{line\ voltage}{\sqrt{3}} \tag{7.2}$$

$$3 - phase\ power = \frac{3}{\sqrt{3}} \times phase\ voltage \times phase\ current \Leftrightarrow \tag{7.3}$$

$$3 - phase\ power = \sqrt{3} \times phase\ voltage \times phase\ current \tag{7.4}$$

In figure 62 the voltage and current were filtered using a "Savitzky Golay" filter to calculate the power using equation 7.4.

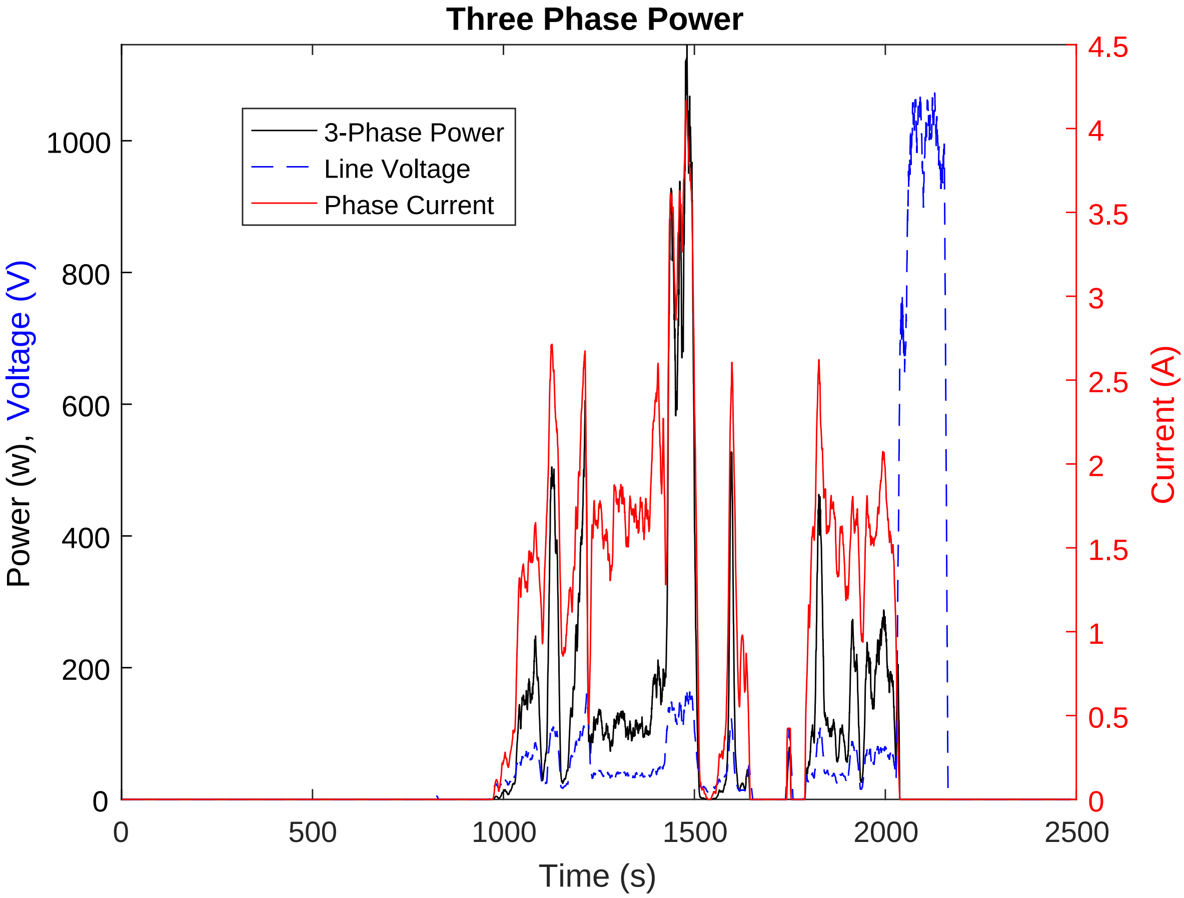


Figure 62 – Power data

7.2 Simulation results

This section will discuss the results of the simulations. Each simulation was executed using the same river velocity input as shown in figure 63. This profile was chosen to: first simulate a turbine startup, followed by a constant river velocity, after which a sudden change from decreasing to increasing river velocity is simulated, and ultimately a constant decreasing river velocity and a constant increasing river velocity. In the following "results" figures, the time axes correspond to the river velocity.

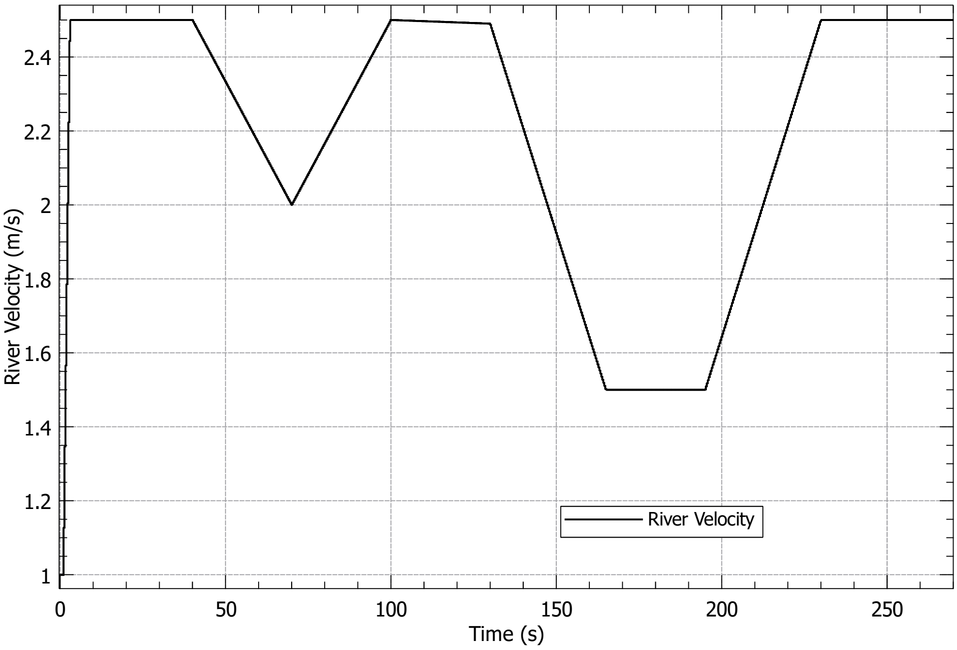


Figure 63 – River velocity input

The comparison of the C_P, total power output, and electrical resistance at various river velocities are shown in figures 64, 65, and 66 respectively. In figure 64 it is apparent that the optimal ω control performs exceptionally at maintaining the C_P at the maximum in varying river velocities. However, as can be concluded from figure 65, the maximum C_P of the rotor does not correspond with the maximum power output of the generator. In fact, at steady river velocities all three of the tested MPPT control systems show a higher generator power output.

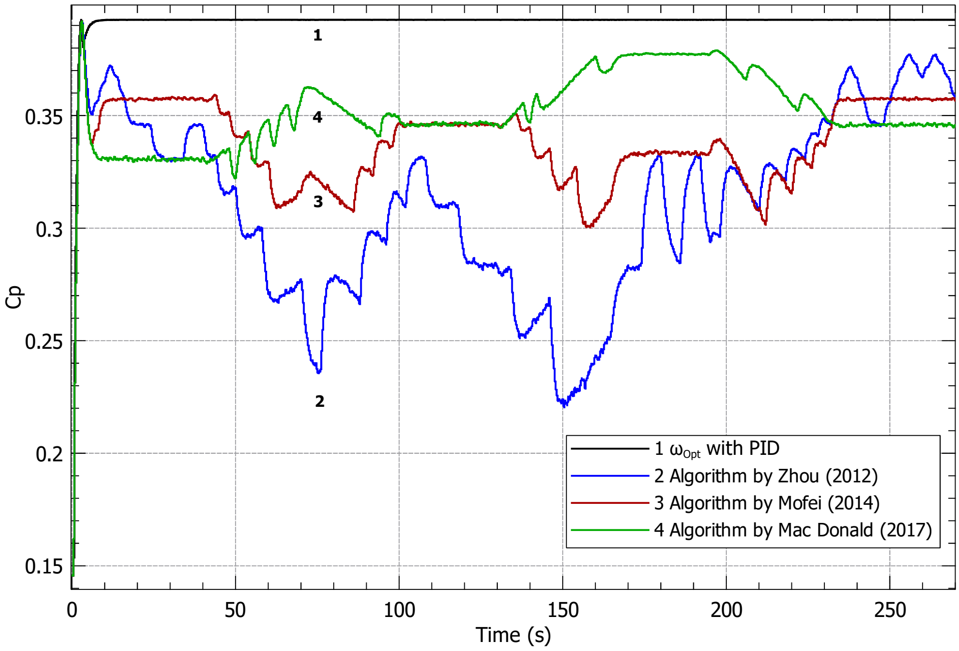


Figure 64 – Comparison of resulting C_P at various river velocities

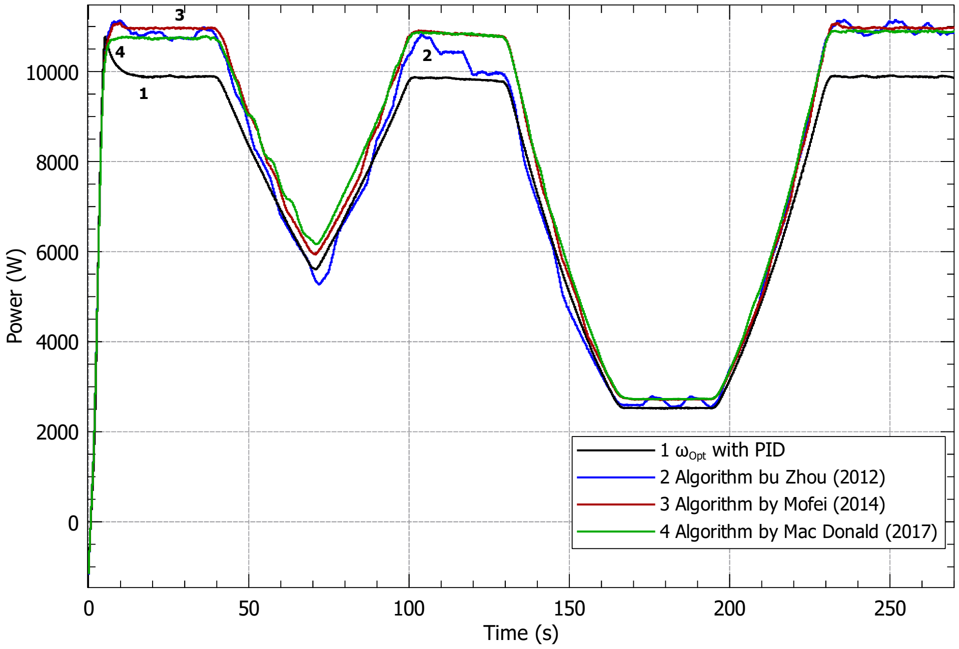


Figure 65 – Comparison of resulting generator power output at various river velocities

Comparing the three MPPT algorithms, one could conclude that in terms of the C_P, neither of the algorithms attains a constant C_P. Even at constant river velocities. Furthermore, the algorithm by MOFEI exhibited a slightly more stable C_P. The algorithm by the author showed a lower C_P at some constant river velocities, but a higher C_P at varying river velocities. And as expected, the algorithm by ZHOU exhibited the least stable C_P.

In terms of the power output, firstly, notice that the generator is most efficient at higher RPM's than that corresponding with the rotor's optimal ω, because the maximum power output occurs on the right side of the C_P-λ curve. Next in order, the algorithm by MOFEI showed slightly higher power outputs at constant river velocities, but results in a lower power output at varying river velocities when compared with the algorithm designed in this dissertation. The algorithm by ZHOU shows unstable power outputs, mainly due to the absence of a "zero" in the algorithm. The lower power output of the proposed algorithm at constant river velocities may just be the result of an inadequately chosen maximum step size and ΔP multiplier. Because the proposed algorithm settles at a higher speed at the startup, while at the second constant velocity section, the power output of the algorithm by MOFEI equals that of the proposed algorithm.

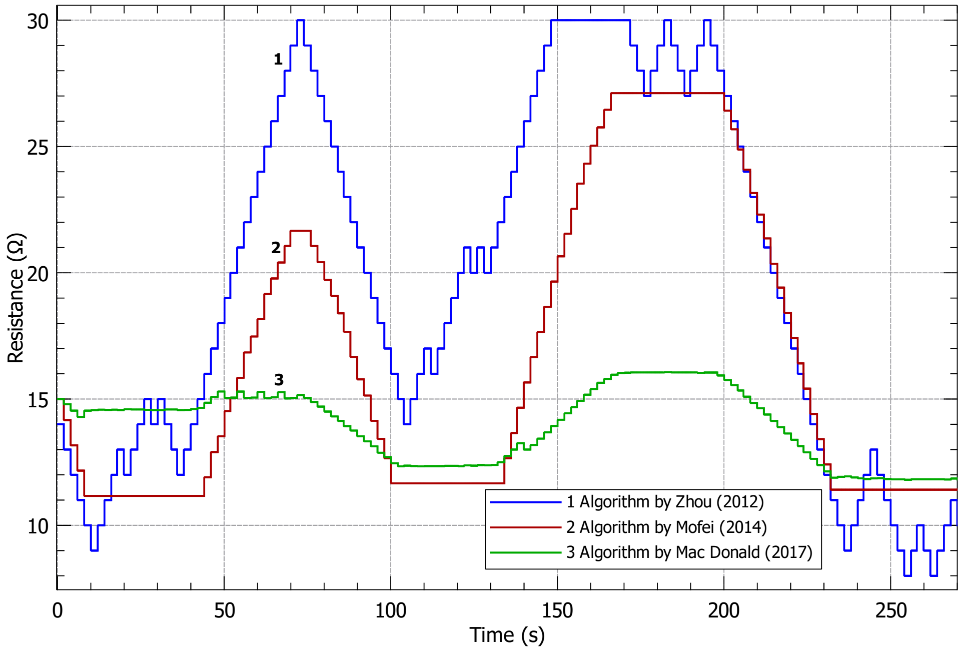


Figure 66 – Comparison of resulting electrical resistance change at various river velocity conditions

The increased power at varying river velocities, using the algorithm designed by the author, can be explained by looking at figure 66. One can see that the algorithm does not remain in an upward or downward trend in varying river velocity conditions, but instead assesses the situation with every iteration (located where the "3" is placed in figure 66). This is advantageous because as the river velocity varies, the C_P of the rotor also varies. An increase in C_P can result in a higher power output than an attempt to undo the effect of the varying river velocity.

For example, in a decreasing river velocity situation, it can be more favorable to decrease the turbine's rotational velocity even more than it is already being decreased as result of the river velocity, because the additional decrease increases the rotor's C_P, and subsequently the generator's input torque. The other algorithms remain in an upward or downward trend until a certain condition is met, resulting in decreased C_P values and notice that the algorithm by ZHOU even reached the IGBT's 30 Ohm limit.

8 Conclusions

8.1 Instrumentation conclusions

The conception goal of the instrumentation was creating a platform that facilitates research on wind- and hydrokinetic turbines, with the HK-10 turbine as a specific application. Low cost and flexibility, were the main reasons for developing such a platform instead of purchasing a similar product.

Using the methodology detailed in chapter 4, the product's architecture was defined, and various prototypes were produced, tested, and analyzed for improvements. The latest prototype proved to lack robustness, more accurate voltage and current sensors, a filtered data presentation, and a more detailed data registering system. Nevertheless, prototypes by definition are used to identify faults, and fortunately it did manage to measure the turbine's power output and functionality to a certain extent in field tests.

With low cost as a main objective, the robustness of the platform and the accuracy of some sensors showed to be less than desired, but the instrumentation platform has proven its flexibility and low cost advantages during the field tests.

8.1.1 Future work

From the conclusions drawn in this chapter, emerges the necessity to:

- Increase robustness of the platform using low cost components such as:
  diodes to avoid reversed polarity,
  LED's to quickly confirm or disprove the functionality of components,
  implementation of fuses to insure safe operation conditions,
  the enlarging of PCB soldering pads for critical components such as voltage regulators,
  and varistors to avoid undesired high voltages;
- Assess the viability of using an instrumentation platform for each HK-10 turbine instead of for each HK^3-10 arrangement to further improve robustness;
- Evaluate the behavior of the current and voltage sensors for lower values to insure the functionality of the sensors at lower values and their accuracy in this range. If they

perform poorly, it may be necessary to utilize the industrial grade sensors, which are more robust and accurate, but may come at a price of multiple instrumentation platforms;
- And lastly, the necessity of an improved software that filters the data presented directly to an observer, and registers raw and unfiltered inputs to the SD-Card.

8.2 Simulation conclusions

The objective of the MPPT development was to increase the power extracted by HK-10, with the focus on an adaptive-step-size MPPT algorithm. The methodology used to asses the proposed algorithm was discussed in chapter 6, and consists of modeling and simulating the hydrokinetic turbine and control systems.

The results derived from these simulations, firstly, coincide with the fact that the optimal electrical power output does not occur at the maximum power delivered by the rotor. This resulted in higher efficiencies with the assessed MPPT algorithms, when compared to the control strategy which aims for the maximum C_P.

Secondly, comparing the algorithm proposed by the author to the remaining algorithms, it showed an increase of efficiency in changing river velocities. However, at constant river velocities, the algorithm presented lower power output at some stages, but may eventually be increased with fine-tuning. Compared to the maximum C_P control, the algorithm provided a superior power output in all the simulated river velocity conditions.

And lastly, the expected result of an adaptive-step-size MPPT was only partially observed, as the algorithm did not present any significant decrease in the time required to reach the MPP, but did however show the desired delicacy in step change near the MPP at constant and gradually changing river velocities.

8.2.1 Future research

The points identified as interesting for future research on this subject are:

- Increase the algorithm's robustness by, e.g., an adaptive delay between iterations and thereby eliminating the need of knowing the system's response time;
- For an increased flexibility, an IGBT with a manually settable limit would be beneficial for future field applications;
- And most importantly, it is essential to experimentally validate the proposed algorithm.

Bibliography

AUBRÉE, R. et al. Design of an efficient small wind-energy conversion system with an adaptive sensorless MPPT strategy. *Renewable Energy*, vol. 86, p. 280–291, 2016. ISSN 18790682. Cited 2 times on pages 18 and 74.

BETZ, A. The Maximum of the Theoretically Possible Exploitation of Wind by Means of a Wind Motor. *Wind Engineering*, vol. 37, no. 4, p. 441–446, 1920. ISSN 0309-524X. Available at: <http://journals.sagepub.com/doi/abs/10.1260/0309-524X.37.4.441>. Cited on page 23.

BITTENCOURT, M. d. P.; NUNES, M. A. Avaliação de potencial hidrocinético remanescente a jusante de uhes na bacia hidrográfica do rio tietê. *CILAMCE*, 2016. Available at: <http://www.periodicos.unb.br/index.php/ripe/article/viewFile/23294/16719>. Cited 2 times on pages 32 and 33.

Brasil Jr., A. C. P. et al. Turbina Hidrocinética Geração 3. *CITENEL*, p. 1–10, 2007. Available at: <http://www.mfap.com.br/pesquisa/arquivos/20081205101337-it46.pdf>. Cited 4 times on pages 7, 27, 29, and 30.

Brasil Jr., A. C. P. et al. On the hydrodynamics of a row arrangement of hydrokinetic propeller turbines. *American Journal of Hydropower, Water and Environment Systems*, 2016. Cited 2 times on pages 32 and 79.

Brasil Jr., A. C. P. et al. Conversão de Energia Hidrocinética com Sistema Flutuante Modular. *CITENEL*, 2017. Cited on page 32.

Brazilian Energy Research Company (EPE). Brazilian Energy Balance 2016. p. 14 – 18, 2016. Available at: <www.epe.gov.br>. Cited on page 16.

COPPING, A. et al. Tethys: Developing a commons for understanding environmental effects of ocean renewable energy. *International Journal of Marine Energy*, Elsevier Ltd, vol. 3-4, p. 41–51, 2013. ISSN 22141669. Available at: <http://dx.doi.org/10.1016/j.ijome.2013.11.004>. Cited on page 20.

CP-LAMBDA. *The Cp-lambda curve*. Available at: <https://goo.gl/YsNtne>. Cited on page 23.

DEREGIOVANNU. *De RegioVannU*. Available at: <http://deregiovannu.nl/html/waterenergie3.html>. Cited on page 21.

EDENHOFER, O. et al. *IPCC, 2011: Summary for Policymakers. In: IPCC Special Report on Renewable Energy Sources and Climate Change Mitigation*. [s.n.], 2011. 246 p. ISSN 0009-4978. ISBN 9789291691319. Available at: <https://goo.gl/Ww7caK>. Cited on page 20.

ELS, R. H. van et al. Hydrokinetic turbine for isolated villages. *PCH Notícias & SHP News*, vol. 19, p. 24–25, 2003. Cited 2 times on pages 7 and 27.

FONSECA, E. N.; ARAUJO, I. G. de. *Projeto Do Sistema De Transmissão E Estrutura De Turbina Hidrocinética*. Thesis (PHD) — UnB - Universidade de Brasília, 2013. Available at: <https://goo.gl/PBiUa4>. Cited on page 31.

GINTER, V. J.; PIEPER, J. K. Robust Gain Scheduled Control of a Hydrokinetic Turbine Part 2: Testing. *2009 IEEE Electrical Power & Energy Conference (EPEC)*, no. November 2009, p. 1–5, 2009. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5420971>. Cited on page 74.

HAQUE, M.; NEGNEVITSKY, M.; MUTTAQI, K. A Novel Control Strategy for a Variable-Speed Wind Turbine With a Permanent-Magnet Synchronous Generator. *IEEE Transactions on Industry Applications*, vol. 46, no. 1, p. 331–339, 2010. ISSN 0093-9994. Cited on page 74.

HARWOOD, J. H. Protótipo de um cata-água que gera 1 kW de eletricidade. *ACTA Amazônica*, vol. 15, 1985. Available at: <http://www.scielo.br/pdf/aa/v15n3-4/1809-4392-aa-15-3-4-0403.pdf>. Cited on page 27.

HAUSER, J. R.; CLAUSING, D. The house of quality.pdf. *Harvard Business Review*, p. 63–73, 1988. Available at: <https://hbr.org/1988/05/the-house-of-quality>. Cited on page 42.

HUA, G.; GENG, Y. A novel control strategy of MPPT taking dynamics of wind turbine into account. *PESC Record - IEEE Annual Power Electronics Specialists Conference*, 2006. ISSN 02759306. Cited on page 74.

KHAN, J.; BHUYAN, G. S. Global Technology Development Status, Report prepared by Powertech Labs for the IEA-OES. 2009. Available at: <https://goo.gl/W8jYD9>. Cited on page 20.

MAGANA, D. et al. Wave and Tidal Energy Strategic Technology Agenda. *Si Ocean*, no. February, p. 1–44, 2014. Available at: <https://goo.gl/qjX2pE>. Cited on page 24.

MCNULTY, T.; HORTA, J.; PLAISIME, J. *Maximum power point motor control*. 2006. Available at: <https://www.google.com/patents/US20060290317>. Cited on page 72.

MOFEI, L. *Hydrokinetic turbine power converter and controller system design and implementation*. Thesis (PHD) — THE UNIVERSITY OF BRITISH COLUMBIA, 2014. Available at: <https://goo.gl/gyDGDq>. Cited 6 times on pages 8, 14, 74, 76, 95, and 96.

MOTLEY, M. R.; BARBER, R. B. Passive control of marine hydrokinetic turbine blades. *Composite Structures*, Elsevier Ltd, vol. 110, no. 1, p. 133–139, 2014. ISSN 02638223. Available at: <http://dx.doi.org/10.1016/j.compstruct.2013.11.026>. Cited on page 26.

MUNTEANU, I. et al. *Optimal control of wind energy systems: towards a global approach*. 1. ed. Springer, 2008. xxii, 283 p. p. ISBN 1848000790. Available at: <https://www.springer.com/br/book/9781848000797>. Cited 2 times on pages 25 and 74.

NGAN, M. S.; TAN, C. W. Multiple Peaks Tracking Algorithm using Particle Swarm Optimization Incorporated with Artificial Neural Network. *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 5, no. 10, p. 379–385, 2011. Cited on page 74.

Ocean Energy Systems. Annual Report. p. 138, 2015. ISSN 1477-9234. Available at: <https://www.ocean-energy-systems.org/library/annual-reports/document/oes-annual-report-2015/>. Cited on page 17.

Ocean Renewable Energy Coalition (OREC); Verdant Power. *Marine and Hydrokinetic Environmental Policy Workshop Marine and Hyrdrokinetic Technology Background and Perspective for New York State NYSERDA ' s Promise to New Yorkers : New Yorkers can count on NYSERDA for*. [S.l.], 2012. Available at: <https://www.nyserda.ny.gov/-/media/Files/Publications/Research/Environmental/marine-hydrokinetic-technologies-background-perspective.pdf>. Cited on page 20.

ROZENFELD, H. et al. *Gestão De Desenvolvimento De Produtos*. 1. ed. São Paulo: Saraiva, 2005. ISBN 978-85-02-05446-2. Cited 3 times on pages 18, 37, and 45.

SIGMAHELLAS. *Sigma Hellas wave powered desalination*. Available at: <http://www.sigmahellas.gr/index.php?lang=2{&}thecatid=3{&}thesubcat>. Cited on page 21.

STONE, R. B.; MCADAMS, D. A.; KAYYALETHEKKEL, V. J. A product architecture-based conceptual DFA technique. *Design Studies*, vol. 25, no. 3, p. 301–325, 2004. ISSN 0142694X. Cited on page 54.

STONE, R. B.; WOOD, K. L.; CRAWFORD, R. H. a Heuristic Method for Identifying Modules for Product Architectures a Heuristic Method for Identifying Modules for. *Design Studies*, vol. 21, p. 1–47, 2000. ISSN 0142694X. Cited 2 times on pages 54 and 55.

TEDD, J.; Peter Kofoed, J. Measurements of overtopping flow time series on the Wave Dragon, wave energy converter. *Renewable Energy*, Elsevier Ltd, vol. 34, no. 3, p. 711–717, 2009. ISSN 09601481. Available at: <http://dx.doi.org/10.1016/j.renene.2008.04.036>. Cited on page 21.

THRESHER, R. A commercialization path and challenges for marine hydrokinetic renewable energy. *IEEE Power and Energy Society General Meeting*, p. 1–8, 2011. ISSN 19449925. Cited on page 17.

TWIDELL, J.; WEIR, T. *Renewable Energy Resources*. 3. ed. [S.l.]: Routledge, 2015. ISBN 978-0-415-58438-8. Cited on page 16.

World Energy Council. World Energy Resources. *World Energy Council Report*, vol. 24, p. 1028, 2016. Available at: <https://www.worldenergy.org/wp-content/uploads/2016/10/World-Energy-Resources-Full-report-2016.10.03.pdf>. Cited 2 times on pages 16 and 17.

YUCE, M. I.; MURATOGLU, A. Hydrokinetic energy conversion systems: A technology status review. *Renewable and Sustainable Energy Reviews*, Elsevier, vol. 43, p. 72–82, 2015. ISSN 13640321. Available at: <http://dx.doi.org/10.1016/j.rser.2014.10.037>. Cited on page 20.

YUEN, K.; APELFRÖJD, S.; LEIJON, M. Implementation of control system for hydrokinetic energy converter. *Journal of Control Science and Engineering*, vol. 2013, 2013. ISSN 16875249. Cited on page 17.

ZHOU, H. *Maximum power point tracking control of hydrokinetic turbine and low-speed high-thrust permanent magnet generator design*. Thesis (PHD) — MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY, 2012. Cited 6 times on pages 8, 14, 74, 75, 96, and 97.

ZHOU, Z. et al. An up-to-date review of large marine tidal current turbine technologies. *Proceedings - 2014 International Power Electronics and Application Conference and Exposition, IEEE PEAC 2014*, p. 480–484, 2014. Cited on page 25.

Appendix

APPENDIX A – Sensor selection matrix

		EF1.1	EF1.1	EF1.1	EF1.1	EF2.1	EF2.1	EF2.1	
	Importance	**Transduce rotor velocity**	**Transduce temperatures**	**Transduce vibration**	**Detect water presence**	**Transduce river velocity**	**Transduce current**	**Transduce voltage**	**Option**
Allow eff. Measurements of wind- and hydro turbines	**5**	5	5	5	5	5	5	5	**A**
		5	5	5	5	5	5	5	**B**
		5	5	5	5	5	5	5	**C**
		5	5	5	5	5	5	5	**D**
Be easy to use	**4**	5	1	5	1	3	3	5	**A**
		3	5	3	5	1	1	5	**B**
		1	3	3	3	5	3	3	**C**
		5	3	3	5	5	5	1	**D**
Be safe	**4**	5	1	3	1	3	3	5	**A**
		1	5	3	3	3	3	5	**B**
		1	3	3	3	3	3	3	**C**
		5	5	3	5	3	5	1	**D**
Allow for data acquisition using IoT servers	**3**	3	1	3	3	3	3	3	**A**
		1	5	3	3	3	1	3	**B**
		1	3	3	3	3	5	1	**C**
		5	5	3	3	3	5	3	**D**
Be low cost	**3**	3	5	5	1	1	3	3	**A**
		1	5	3	1	1	1	5	**B**
		1	1	1	3	5	5	1	**C**
		5	3	5	3	3	3	3	**D**
Be robust	**3**	3	5	3	1	5	5	3	**A**
		3	5	3	3	3	3	3	**B**
		3	3	3	3	5	3	5	**C**

		5	5	3	5	5	5	1	**D**
Have high precision	**3**	5	1	5	3	5	3	3	**A**
		3	5	5	5	5	5	3	**B**
		3	3	3	3	3	3	5	**C**
		3	3	1	1	5	3	1	**D**
Have long battery life	**2**	3	5	5	1	3	3	3	**A**
		1	5	3	3	1	5	5	**B**
		5	3	5	3	3	5	5	**C**
		5	5	5	5	5	5	5	**D**
Allow serial connectivity	**2**	5	1	3	3	3	3	3	**A**
		1	5	3	3	3	1	3	**B**
		1	3	3	3	3	3	1	**C**
		5	5	3	3	3	3	3	**D**
Register captured data	**2**	5	1	3	3	3	3	3	**A**
		1	5	3	3	3	1	3	**B**
		1	3	3	3	3	3	1	**C**
		5	5	3	3	3	3	3	**D**
Have large measurement range	**2**	3	1	5	3	5	3	5	**A**
		3	3	3	5	5	3	3	**B**
		3	3	3	3	3	1	1	**C**
		3	5	1	1	5	5	5	**D**
Be easy to manufacture	**2**	3	1	3	3	3	3	3	**A**
		1	5	3	3	1	1	3	**B**
		1	3	3	3	5	3	1	**C**
		5	5	3	3	3	3	3	**D**
Be eco friendly	**1**	3	1	3	3	3	3	3	**A**
		3	5	3	3	3	3	3	**B**
		3	3	3	3	3	3	3	**C**
		3	5	3	3	3	3	3	**D**
Be easy to ship	**1**	3	3	3	3	3	3	3	**A**
		1	5	3	3	3	3	3	**B**
		1	1	3	3	3	3	3	**C**
		5	5	3	3	3	3	1	**D**
Be attractive to buy	**1**	5	1	3	3	3	3	3	**A**
		1	5	3	3	3	1	3	**B**

		1	3	3	3	3	3	1	**C**
		5	5	3	3	3	3	1	**D**
Be easy to dispose	**1**	3	1	3	3	3	3	3	**A**
		3	5	3	3	3	3	3	**B**
		3	5	3	3	3	3	3	**C**
		3	5	3	3	3	3	3	**D**
Sum of "option score" times "importance"		159	93	155	95	137	133	147	**A**
		87	191	133	139	115	103	153	**B**
		87	121	125	127	151	139	113	**C**
		181	175	127	143	155	163	103	**D**

Table 13 – Selection matrix for sensors

APPENDIX B – Pin reference

PIN	Special function	Hydro-K function
A0		Accelerometer X turbine 1
A1		Accelerometer Y turbine 1
A2		Accelerometer Z turbine 1
A3		Current turbine 1
A4		Current turbine 2
A5		Current turbine 3
A6		Accelerometer X turbine 2
A7		Accelerometer Y turbine 2
A8		Accelerometer Z turbine 2
A9		Voltage turbine 1
A10		Voltage turbine 2
A11		Voltage turbine 3
A12		Accelerometer X turbine 3
A13		Accelerometer Y turbine 3
A14		Accelerometer Z turbine 3
A15		Battery voltage indicator
D0	RX0	TTL / TX TTL FibreOptics
D1	TX0	TTL / RX TTL FibreOptics
D2	Interrupt / PWM	RPM Turbine 1
D3	Interrupt / PWM	RPM Turbine 2
D4	PWM	Drain Relay Turbine 3
D5	PWM	Cooling Relay Turbine 3
D6	PWM	Drain Relay Turbine 2
D7	PWM	Cooling Relay Turbine 2
D8	PWM	Drain Relay Turbine 1
D9	PWM	Cooling Relay Turbine 1
D10	PWM	Voltage range selector 1
D11	PWM	Voltage range selector 2
D12	PWM	Voltage range selector 3
D13	PWM / LED	

D14	TX3	RX WiFi
D15	RX3	TX WiFi
D16	TX2	RX TTL FibreOptics
D17	RX2	TX TTL FibreOptics
D18	TX1 / Interrupt	RPM Turbine 3
D19	RX1 / Interrupt	RPM Flow sensor
D20	SDA / Interrupt	SDA RTC
D21	SCL / Interrupt	SCL RTC
D22		
D23		Ambient Temperature 1
D24		
D25		Generator Temperature 1
D26		
D27		Gearbox Temperature 1
D28		
D29		Ambient Temperature 2
D30		
D31		Generator Temperature 2
D32		
D33		Gearbox Temperature 2
D34		
D35		Ambient Temperature 3
D36		
D37		Generator Temperature 3
D38		
D39		Gearbox Temperature 3
D40		
D41		River Temperature
D42		
D43		Water sensor 1
D44	PWM	
D45	PWM	Water sensor 2
D46	PWM	
D47		Water sensor 3

D48		
D49		
D50	MISO	MISO SD Card
D51	MOSI	MOSI SD Card
D52	SCK	SCK SD Card
D53	CS	CS SD Card

Table 14 – Pin reference

B.1 Instrumentation platform schematic

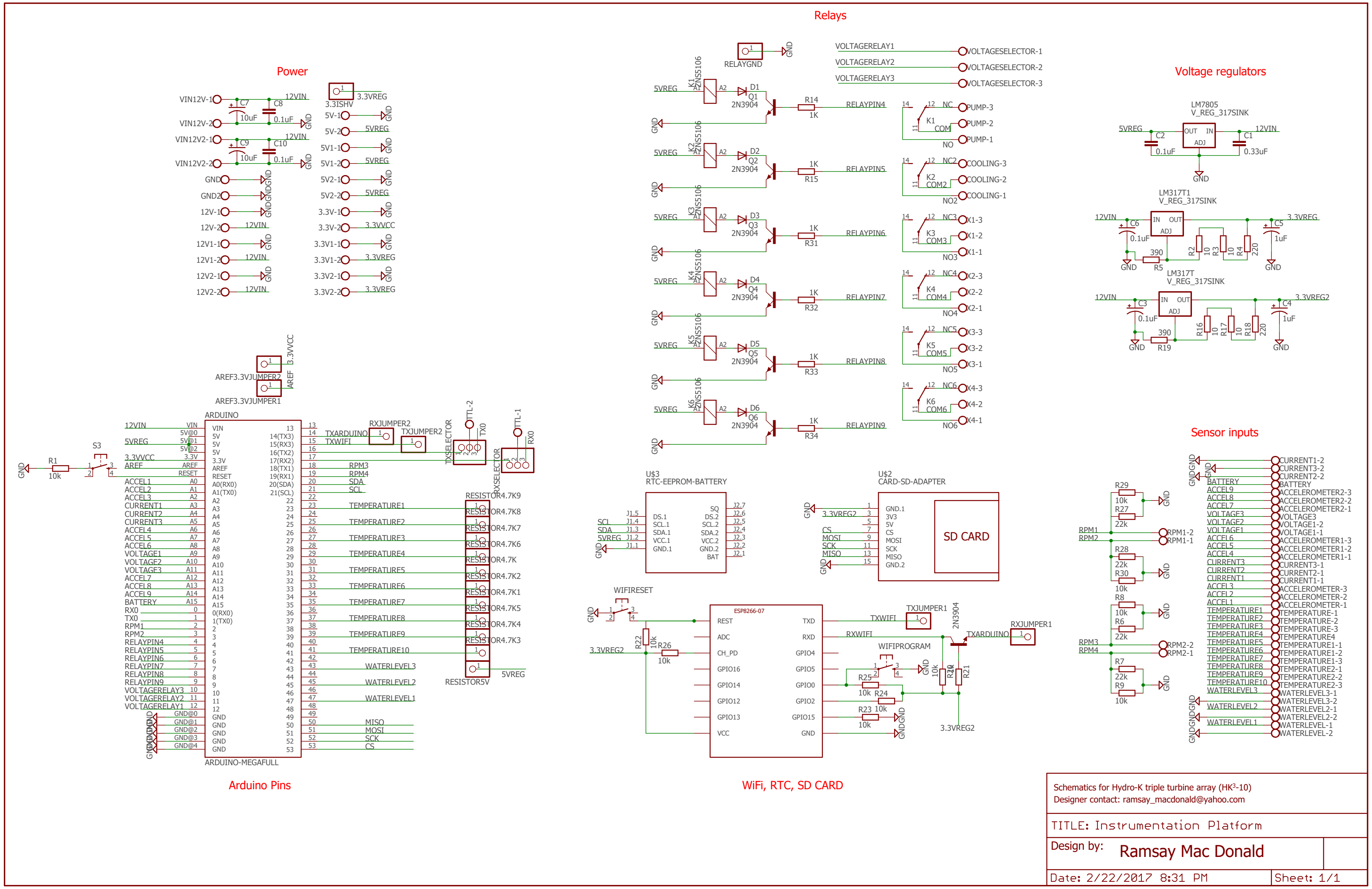


Figure 67 – Schematic for the instrumentation platform

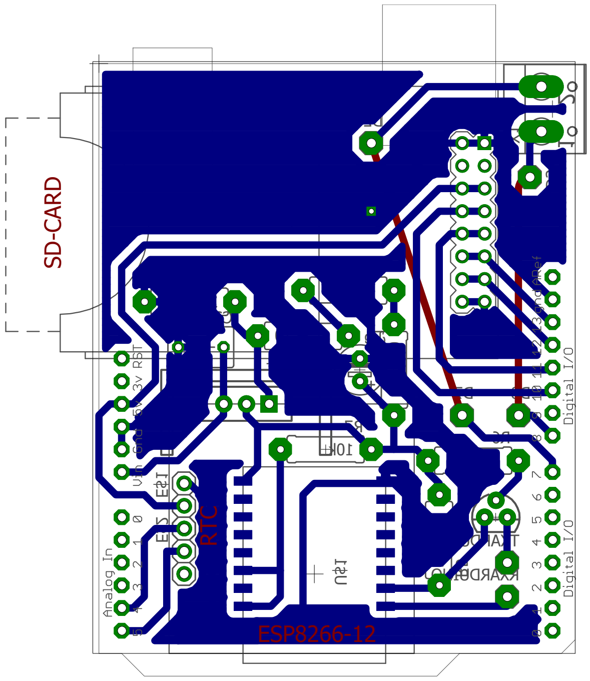


Figure 68 – ESP8266 WiFi shield complete with RTC and SD-Card as replacement for CC3000 WiFi shield

APPENDIX C – Previous prototypes PCB

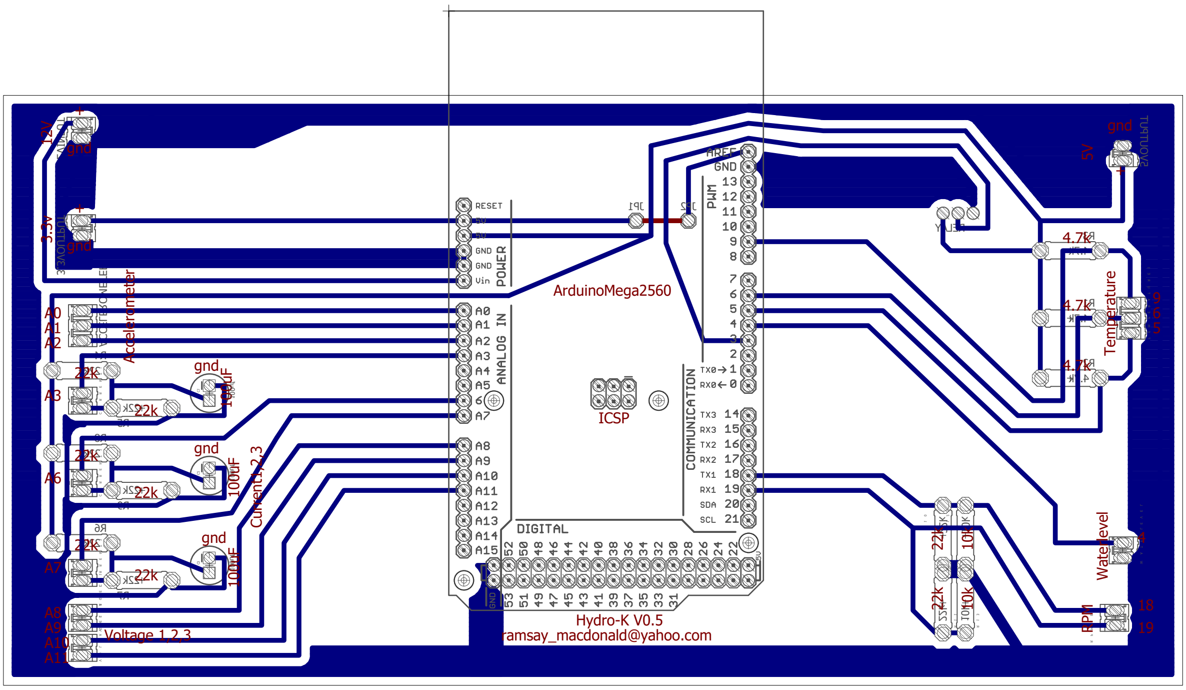


Figure 69 – The first instrumentation prototype for a single Hydro-K turbine

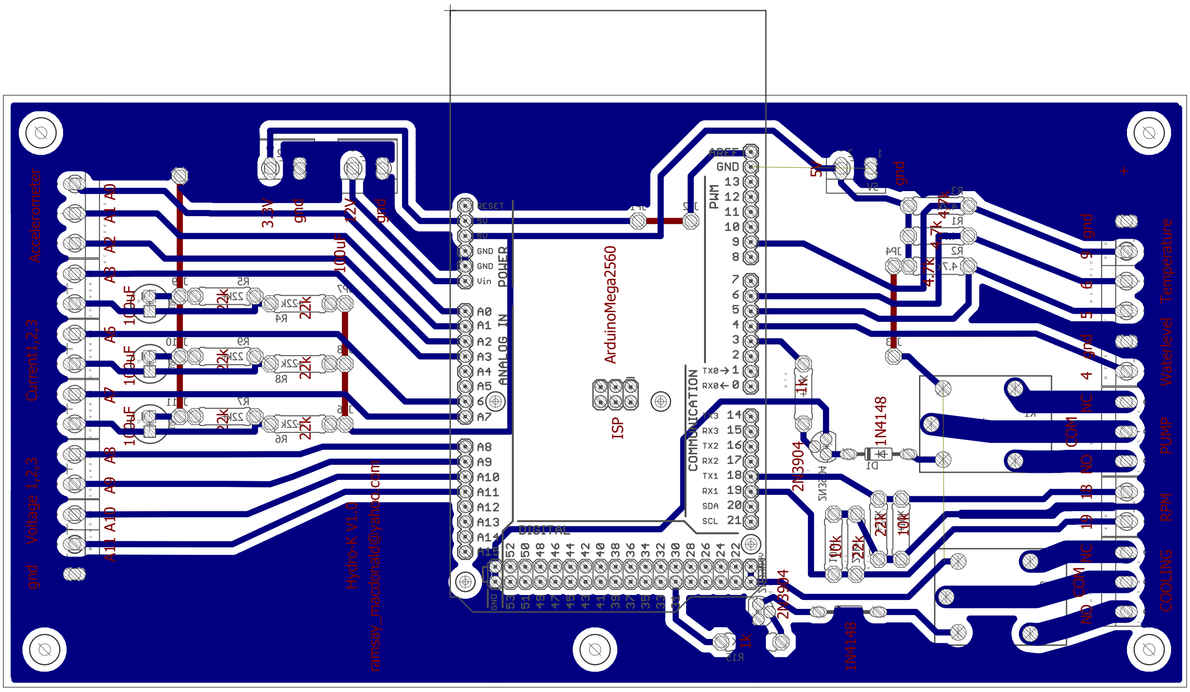


Figure 70 – The second instrumentation prototype for a single Hydro-K turbine

APPENDIX D – Latest prototype PCB

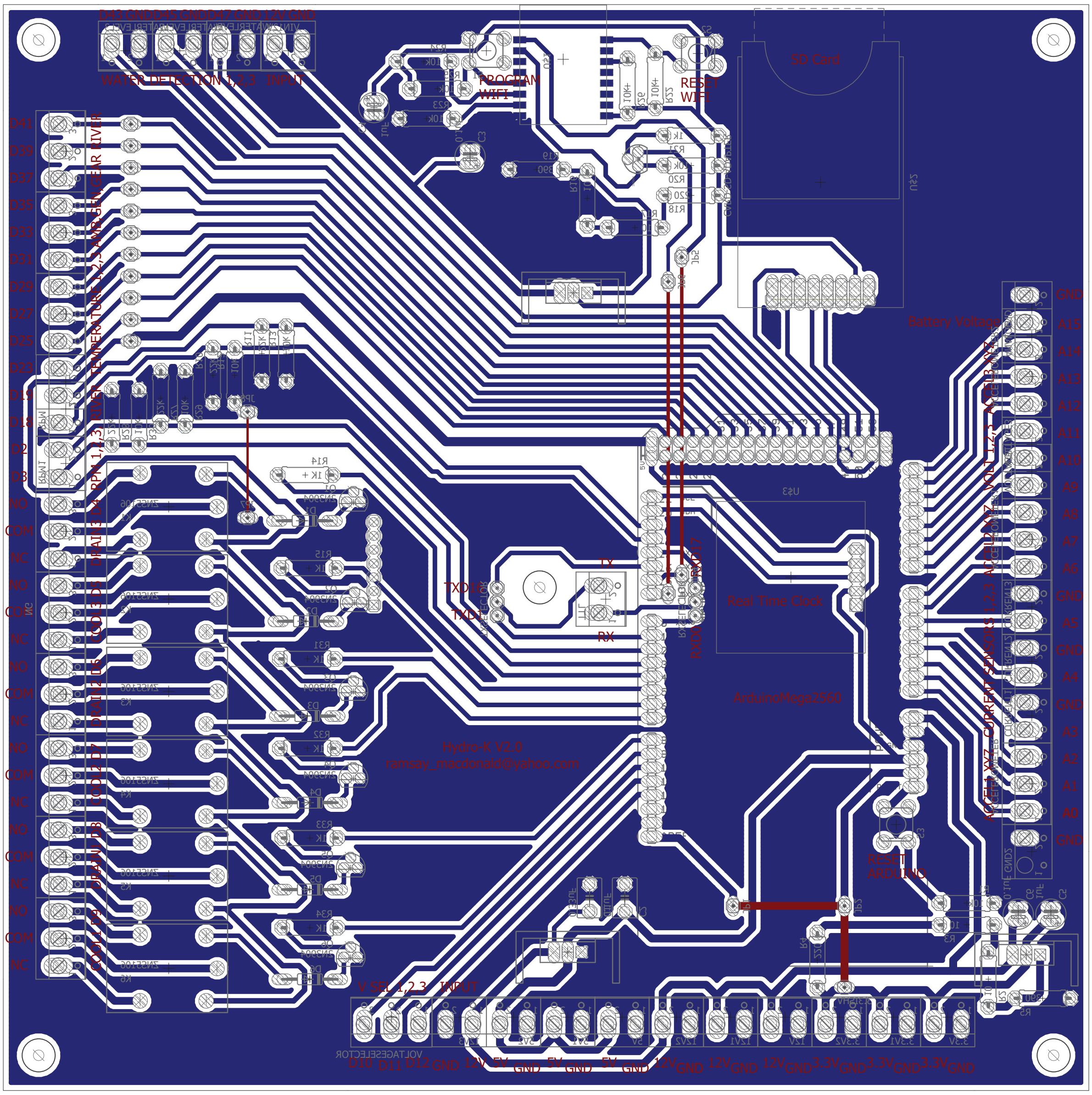


Figure 71 – The first instrumentation prototype for all three Hydro-K turbines

APPENDIX E – Current transformer's signal conditioning PCB

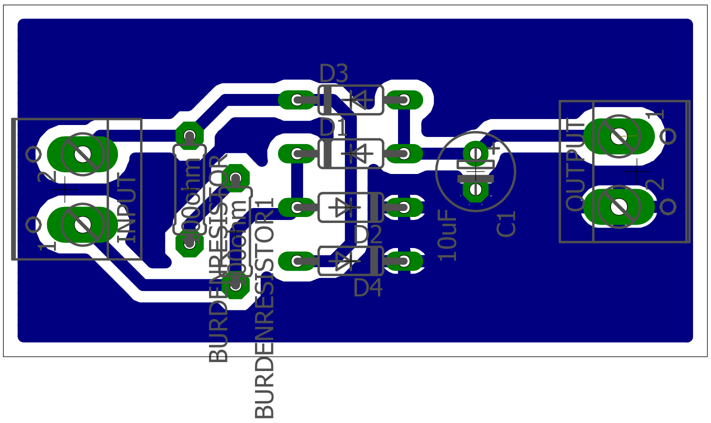


Figure 72 – Current transformer's signal conditioning PCB

APPENDIX F – Voltage divider PCB

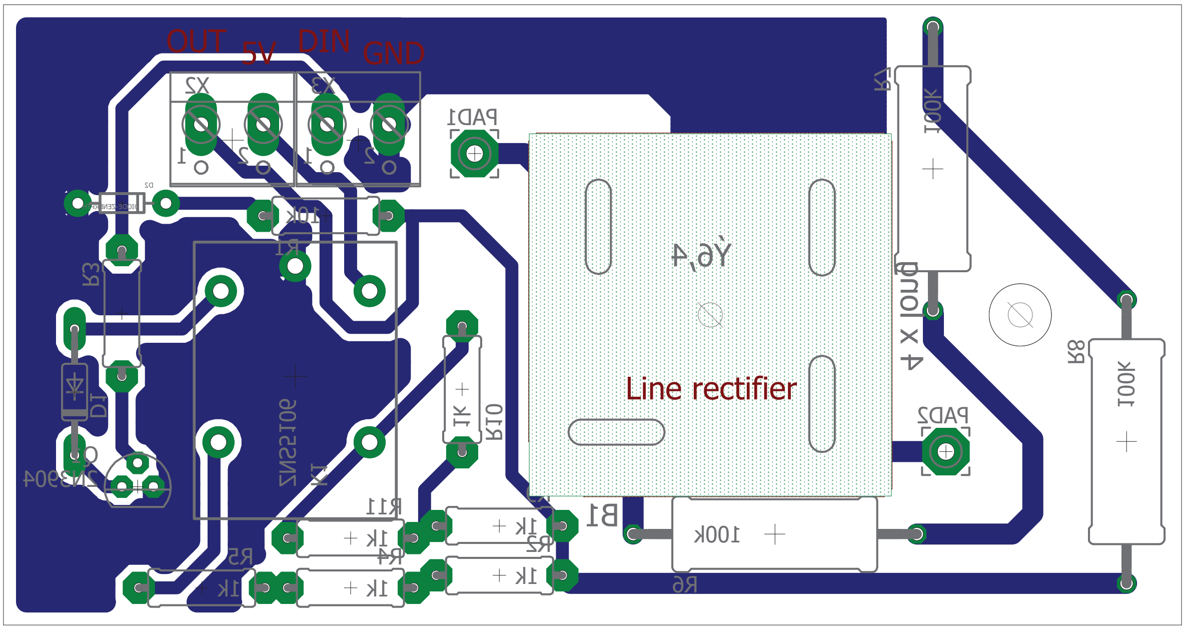


Figure 73 – Voltage sensor with selectable resolution

APPENDIX G – Amplifier PCB

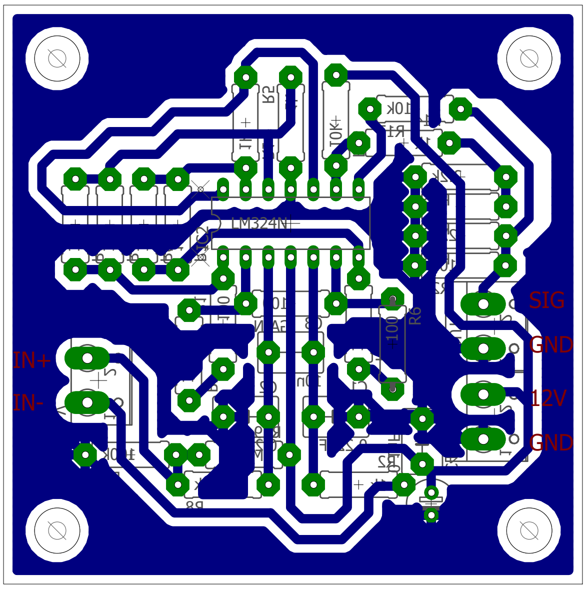


Figure 74 – PCB of designed water flow sensor

APPENDIX H – Arduino Code

```
//Instrumentation platform sketch
//Written by Ramsay Mac Donald
//ramsay_macdonald@yahoo.com
//WARNING!!! DO NOT CONNECT THE ARDUINO TO ANY COMPUTER WHEN HIGH VOLTAGES ARE BEING MEASURED
#include <ESP8266.h>
#include <Modbus.h>
#include <ModbusIP_ESP8266AT.h>
//-----------------------------------------------------------------------------Wifi Pre Program
ESP8266 wifi(Serial3, 115200);
const String wifiNetwork = "WhyPhy";                        //Enter the WiFi network name here
const String password = "P@ssw0rd";                     //Enter the WiFi network password here
ModbusIP mb;
#define NUMBER_OF_REGISTERS 31                  //How many parameters will you send using WiFi?
int IREGS[NUMBER_OF_REGISTERS];
int data2send[NUMBER_OF_REGISTERS];
//-------------------------------------------------------------------------------SD Pre Program
#include <SD.h>
#include <SPI.h>
const int chipSelect = 53;   //Independent of which pin you select, always keep 53 as an OUTPUT
#define NUMBER_OF_SDVARIABLES 31            //How many parameters will you save on the SD Card?
char fileName[16];
//------------------------------------------------------------------------------RTC Pre Program
#include <Wire.h>
#include "RTClib.h"
RTC_DS1307 RTC;
//----------------------------------------------------------------------Temperature Pre Program
#include <OneWire.h>
#define NUMBER_OF_BUSSES 10                        //How many temperatures busses will you use?
#define NUMBER_OF_COOLING_RELAYS 3                   //How many cooling pumps will you control?
#define COOLING_START_TEMP 55                                       //Cooling pump on threshold
#define COOLING_STOP_TEMP 50                                       //Cooling pump off threshold
// {AT1, GT1, MT1, AT2, GT2, MT2, AT3, GT3, MT3, riverTemp}
const int busList[NUMBER_OF_BUSSES] = {23, 25, 27, 29, 31, 33, 35, 37, 39, 41};      //bus pins
OneWire *ds[NUMBER_OF_BUSSES];
float celsius[NUMBER_OF_BUSSES];
byte busMatrix[NUMBER_OF_BUSSES][12];
byte addrMatrix[NUMBER_OF_BUSSES][8];
const int coolingRelayPins[NUMBER_OF_COOLING_RELAYS] = {9, 7, 5};      //Set cooling relay pins
boolean tempTrigger [NUMBER_OF_COOLING_RELAYS] = {false, false, false};
unsigned long tempCompareTime[NUMBER_OF_COOLING_RELAYS] = {0, 0, 0};
boolean coolingRelayState[NUMBER_OF_COOLING_RELAYS] = {0, 0, 0};
//-------------------------TEMPERATURE CLOCK---------------------------------------------------
unsigned long zero = 0;
unsigned long PTR = 0;                                                //PreviousTemperatureRead
const unsigned long ONE = 1000;                                             //OneSecondInterval
boolean ReadOK = true;                                     //Check if temperature has been read
//------------------------------------------------------------------------------RPM Pre Program
#define NUMBER_OF_INTERRUPTS 4             //Declare number of pins for interrupt (RPM sensors)
float RPM[NUMBER_OF_INTERRUPTS];
// {turbine1, turbine2, turbine3, riverVelocity}
const int RPMpin[NUMBER_OF_INTERRUPTS] = {3, 2, 18, 19};//Declare which pins to attach interrupt
unsigned long pulse[NUMBER_OF_INTERRUPTS];
const int pulsesPerRot[NUMBER_OF_INTERRUPTS] = {10, 10, 60, 60};//Declare how many pulses per rotation
boolean inter = false;
//--------------------------------------------------------------------Accelerometer Pre Program
#define NUMBER_OF_ACCEL 3                                    //Declare number of accelerometers
#define NUMBER_OF_AXES 3                                               //Declare number of axes
#define SAMPLE_SIZE 64                      //Declare how many samples to take the average from
```

```
// accelerometerAmplitude[0][0] = X-turbine 1, accelerometerAmplitude[0][1] = Y-turbine 1, accelerometerAmplitude[0][2] = Z-turbine 1
// accelerometerAmplitude[1][0] = X-turbine 2, accelerometerAmplitude[1][1] = Y-turbine 2, accelerometerAmplitude[1][2] = Z-turbine 2
// accelerometerAmplitude[2][0] = X-turbine 3, accelerometerAmplitude[2][1] = Y-turbine 3, accelerometerAmplitude[2][2] = Z-turbine 3
const int accelerometerPins[NUMBER_OF_ACCEL][NUMBER_OF_AXES] = {
{A0, A1, A2},
{A6, A7, A8},
{A12, A13, A14}
};
int accelerometerAmplitude[NUMBER_OF_ACCEL][NUMBER_OF_AXES];
//--------------------------------------------------------------------------Voltage Pre Program
#define NUMBER_OF_VOLTAGE_SENSORS 3                         //Declare number of voltage sensors
#define AX_axbVOLTAGEACTIVE 1.131                         //Voltage sensor calibration relay on
#define B_axbVOLTAGEACTIVE 0                              //Voltage sensor calibration relay on
#define AX_axbVOLTAGE 0.5711                             //Voltage sensor calibration relay off
#define B_axbVOLTAGE 0                                   //Voltage sensor calibration relay off
// voltageVal[0] = turbine 1, voltageVal[1] = turbine 2, voltageVal[2] = turbine 3
const int voltageSensors[NUMBER_OF_VOLTAGE_SENSORS] = {A9, A10, A11};     //Voltage sensor pins
// resolutionRelayPins[0] = turbine 1, resolutionRelayPins[1] = turbine 2, resolutionRelayPins[2] = turbine 3
const int resolutionRelayPins[NUMBER_OF_VOLTAGE_SENSORS] = {10, 11, 12};   //Voltage relay pins
float voltageVal[NUMBER_OF_VOLTAGE_SENSORS];
unsigned long compareTime[NUMBER_OF_VOLTAGE_SENSORS] = {0, 0, 0};
boolean relayInTransition[NUMBER_OF_VOLTAGE_SENSORS] = {false, false, false};
//--------------------------------------------------------------------------Current Pre Program
#define NUMBER_OF_CURRENT_SENSORS 3                         //Declare number of current sensors
#define AX_axbCURRENT 0.0316                                       //Current sensor calibration
#define B_axbCURRENT 3.2206                                        //Current sensor calibration
// currentVal[0] = turbine 1, currentVal[1] = turbine 2, currentVal[2] = turbine 3
const int currentSensors[NUMBER_OF_CURRENT_SENSORS] = {A3, A4, A5};      //Declare current pins
float currentVal[NUMBER_OF_CURRENT_SENSORS];
//---------------------------------------------------------------------Water Sensor Pre Program
#define NUMBER_OF_DIGITAL_SWITCHES 3                          //Declare number of water sensors
// {turbine1, turbine2, turbine3}
const int switchPins[NUMBER_OF_DIGITAL_SWITCHES] = {43, 45, 47};              //Water sensor pins
// {turbine1, turbine2, turbine3}
const int drainPins[NUMBER_OF_DIGITAL_SWITCHES] = {8, 6, 4};                //Drainage pump pins
boolean waterPresent[NUMBER_OF_DIGITAL_SWITCHES] = {false, false, false};
boolean lastPresenceState[NUMBER_OF_DIGITAL_SWITCHES] = {false, false, false};
unsigned long presenceCompareTime[NUMBER_OF_DIGITAL_SWITCHES] = {0, 0, 0};
//----------------------------------------------------------------------------Relay Pre Program
#define NUMBER_OF_RELAYS 9                                           //Declare number of relays
//{cool1, drain1, cool2, drain2, cool3, drain3, Vsel1, Vsel2, Vsel3}
const int relayPins[NUMBER_OF_RELAYS] = {9, 8, 7, 6, 5, 4, 10, 11, 12};            //Relay pins
//--------------------------------------------------------------------------Program Pre Program
#define NUMBER_OF_TURBINES 3                                       //Declare number of turbines
long ts;
unsigned long CTime = 0;                                            //Reference of current time
//---------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//-------------------------------------------------------------------------------WiFi Functions
//------------------------CREATE REGISTERS FUNCTION--------------------------------------------
void addRegisters(int IREG[])
{
for (int r = 0; r < NUMBER_OF_REGISTERS; r++) {
IREG[r] = r + 1;
mb.addIreg(IREG[r]);
}
}
//---------------------------------------------------------------------------------SD Functions
//------------------------END SD LINE FUNCTION---------------------------------------------
void endLineSD(int chipselect, char filen[16])
```

```
{
char file[16];
for (int f = 0; f < 16; f++)
{
file[f] = filen[f];
}
File dataFile = SD.open(file, FILE_WRITE);
if (dataFile)
{
dataFile.println("");
dataFile.close();
}
else
{
SD.begin(chipselect);
}
}
//------------------------WRITE TO SD CARD FUNCTION--------------------------------------------
void writingDataToSD(char filen[16], int data2Send[])
{
char file[16];
for (int f = 0; f < 16; f++)
{
file[f] = filen[f];
}
File dataFile = SD.open(file, FILE_WRITE);
if (dataFile)
{
if (! RTC.isrunning())
{
for (int da = 0; da < NUMBER_OF_SDVARIABLES; da++)
{
dataFile.print(data2Send[da]);
dataFile.print("\t");
}
}
else {
DateTime now = RTC.now();
dataFile.print(now.unixtime());
dataFile.print("\t");
for (int da = 0; da < NUMBER_OF_SDVARIABLES; da++)
{
dataFile.print(data2Send[da]);
dataFile.print("\t");
}
}
dataFile.close();
}
else
{
Serial.println("error opening file");
}
}
//------------------------INITIALIZE SD LOG FILE FUNCTION--------------------------------------
void initializeLogFile(int chipselect, char charBuff[])
{
pinMode(chipselect, OUTPUT);
Serial.println("Initializing SD card...");
if (!SD.begin(chipselect))
{
Serial.println("Card failed, or not present");
return;
}
else
```

```
{
DateTime now = RTC.now();
if (! RTC.isrunning())
{
Serial.println("RTC not running");
for (int a = 0; a < 99; a++)
{
if (a < 9)
{
int b = a + 1;
String number = String(b);
String file = "log0" + number + ".txt";
file.toCharArray(charBuff, 16);
if (SD.exists(charBuff))
{
Serial.print(charBuff); Serial.println(" present");
}
else
{
Serial.print("Writing to "); Serial.println(charBuff);
File dataFile = SD.open(charBuff, FILE_WRITE);
if (dataFile)
{
dataFile.println("No RTC");
dataFile.close();
}
else {
Serial.println("error opening file");
}
break;
}
}
else
{
int b = a + 1;
String number = String(b);
String file = "log" + number + ".txt";
file.toCharArray(charBuff, 16);
if (SD.exists(charBuff))
{
Serial.print(charBuff); Serial.println(" present");
}
else
{
Serial.print("Writing to "); Serial.println(charBuff);
File dataFile = SD.open(charBuff, FILE_WRITE);
if (dataFile)
{
dataFile.println("No RTC");
dataFile.close();
}
else
{
Serial.println("error opening file");
}
break;
}
}
}
}
else {
int ye = now.year();
int mo = now.month();
int da = now.day();
String months;
String days;
if (mo < 10)
{
months = '0' + String(mo);
}
else {
months = String(mo);
}
if (da < 10)
{
days = '0' + String(da);
}
else {
days = String(da);
}
String file = String(ye) + months + days + ".txt";
file.toCharArray(charBuff, 16);
File dataFile = SD.open(charBuff, FILE_WRITE);
if (dataFile)
{
Serial.print("Writing to "); Serial.println(charBuff);
dataFile.print("Initialized at: "); dataFile.println(now.unixtime());
dataFile.close();
}
else
{
Serial.println("error opening file");
}
}
}
}
//------------------------------------------------------------------------Temperature Functions
//------------------------CREATE BUSSES FUNCTION-----------------------------------------------
void createBusses(void)
{
//busses = (OneWire *) malloc(sizeof(OneWire) * NUMBER_OF_BUSSES);
for (int i = 0; i < NUMBER_OF_BUSSES; i++) {
ds[i] = new OneWire(busList[i]);
}
}
//------------------------DISCOVER ADRESSES FUNCTION-------------------------------------------
void discoverBusAdresses(byte addressMatrix[NUMBER_OF_BUSSES][8] )
{
//  byte addressMatrix[NUMBER_OF_BUSSES][8];
byte j;
byte present = 0;
byte data[12];
for (int i = 0; i < NUMBER_OF_BUSSES; i++) {
ds[i]->search(addressMatrix[i]);
for ( j = 0; j < 8; j++) {
Serial.print("0x");
if (addressMatrix[i][j] < 16) {
Serial.print('0');
}
Serial.print(addressMatrix[i][j], HEX);
if (j < 7) {
Serial.print(", ");
}
}
if ( OneWire::crc8( addressMatrix[i], 7) != addressMatrix[i][7]) {
Serial.print("CRC is not valid!\n\r");
return;
}
```

```
Serial.println();
Serial.print("pin: ");
Serial.println(busList[i]);
ds[i]->reset_search();
}
}
//------------------------REQUEST TEMPERATURE FUNCTION------------------------------------------
void requestTemperature(void)
{
for (int r = 0; r < NUMBER_OF_BUSSES; r++) {
ds[r]->reset();
ds[r]->select(addrMatrix[r]);
ds[r]->write(0x44, 1);
}
}
//------------------------READ TEMPERATURE FUNCTION---------------------------------------------
void readTemp(float temp[NUMBER_OF_BUSSES]) {
for (int s = 0; s < NUMBER_OF_BUSSES; s++)
{
ds[s]->reset();
ds[s]->select(addrMatrix[s]);
ds[s]->write(0xBE);
for ( byte i = 0; i < 9; i++) {              // we need 9 bytes
busMatrix[s][i] = ds[s]->read();
}
int16_t raw = (busMatrix[s][1] << 8) | busMatrix[s][0];    // Convert to actual temperature
temp[s] = (float)raw / 16.0;    // because the result is a 16 bit signed integer, it should
}
}
//------------------------AUTOMATICALLY CHANGE COOLING RELAY STATE FUNCTION---------------------
void controlCoolingRelayState(const int coolingrelaypins[], const float temperatures[], unsigned long
     tempcomparetime[], boolean trigger[], boolean coolingrelaystate[])
{
unsigned long referenceTime[NUMBER_OF_COOLING_RELAYS];
float maxTemp[NUMBER_OF_COOLING_RELAYS] = {0, 0, 0};
float tempmatrix[NUMBER_OF_COOLING_RELAYS][NUMBER_OF_COOLING_RELAYS];
for (int X = 0; X < NUMBER_OF_COOLING_RELAYS; X++)
{
if (X == 0)
{
for (int Z = 0; Z < 3; Z++)
{
tempmatrix[X][Z] = temperatures[Z];
}
}
if (X == 1)
{
for (int Z = 3; Z < 6; Z++)
{
tempmatrix[X][Z - 3] = temperatures[Z];
}
}
if (X == 2)
{
for (int Z = 6; Z < 9; Z++)
{
tempmatrix[X][Z - 6] = temperatures[Z];
}
}
}
for (int Y = 0; Y < 3; Y++)
{
maxTemp[Y] = max(tempmatrix[Y][0], tempmatrix[Y][1]);
maxTemp[Y] = max(maxTemp[Y], tempmatrix[Y][2]);
```

```
}
boolean button[NUMBER_OF_COOLING_RELAYS] {false, false, false};
for (int sst = 0; sst < NUMBER_OF_COOLING_RELAYS; sst++)
{
referenceTime[sst] = millis();
if (maxTemp[sst] > COOLING_START_TEMP) button[sst] = true;
else if (maxTemp[sst] < COOLING_STOP_TEMP) button[sst] = false;
else button[sst] = trigger[sst];
if (button[sst] != trigger[sst]) tempcomparetime[sst] = referenceTime[sst];
if (tempcomparetime[sst] + 500UL < referenceTime[sst])
{
if (button[sst] != coolingrelaystate[sst]) {
coolingrelaystate[sst] = ! coolingrelaystate[sst];
digitalWrite(coolingrelaypins[sst], coolingrelaystate[sst]);
}
}
trigger[sst] = button[sst];
}
}
//--------------------------------------------------------------------------------RPM Functions
//------------------------attachInterrupts FUNCTION--------------------------------------------
void attachInterrupts(int interruptPins[])
{
attachInterrupt(digitalPinToInterrupt(interruptPins[0]), rotCounter, FALLING);
if (NUMBER_OF_INTERRUPTS < 2) return;
attachInterrupt(digitalPinToInterrupt(interruptPins[1]), rotCounter1, FALLING);
if (NUMBER_OF_INTERRUPTS < 3) return;
attachInterrupt(digitalPinToInterrupt(interruptPins[2]), rotCounter2, FALLING);
if (NUMBER_OF_INTERRUPTS < 4) return;
attachInterrupt(digitalPinToInterrupt(interruptPins[3]), rotCounter3, FALLING);
if (NUMBER_OF_INTERRUPTS < 5) return;
attachInterrupt(digitalPinToInterrupt(interruptPins[4]), rotCounter4, FALLING);
if (NUMBER_OF_INTERRUPTS < 6) return;
attachInterrupt(digitalPinToInterrupt(interruptPins[5]), rotCounter5, FALLING);
}
//------------------------detachInterrupts FUNCTION--------------------------------------------
void detachInterrupts(int interruptPins[])
{
for (int p; p < NUMBER_OF_INTERRUPTS; p++)
{
detachInterrupt(digitalPinToInterrupt(interruptPins[p]));
}
}
//------------------------checkZeroRPM FUNCTION------------------------------------------------
void checkZeroRPM(float RPMval[], unsigned long lastPulse[])
{
for (int c; c < NUMBER_OF_INTERRUPTS; c++)
{
if (micros() - lastPulse[c] >= 5000000 || RPMval[c] >= 2000) {
RPMval[c] = 0;
}
}
}
//------------------------RPM FUNCTIONS--------------------------------------------------------
// Here there are 6 rotCounter functions because there's a maximum
// of 6 interrupt pins on Arduino Mega.
// There will be no rollover problem as long as "unsigned long" is used
// and a pulse was measured within 71.5 minutes. If not, only the first
// measurement will be incorrect.
//------------------------rotCounter1 FUNCTION-------------------------------------------------
void rotCounter()
{
volatile unsigned long cPulse = micros();
volatile const int n = 0;
```

```
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//------------------------rotCounter2 FUNCTION-------------------------------------------------
void rotCounter1()
{
volatile unsigned long cPulse = micros();
volatile const int n = 1;
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//------------------------rotCounter3 FUNCTION-------------------------------------------------
void rotCounter2()
{
volatile unsigned long cPulse = micros();
volatile const int n = 2;
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//------------------------rotCounter4 FUNCTION-------------------------------------------------
void rotCounter3()
{
volatile unsigned long cPulse = micros();
volatile const int n = 3;
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//------------------------rotCounter5 FUNCTION-------------------------------------------------
void rotCounter4()
{
volatile unsigned long cPulse = micros();
volatile const int n = 4;
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//------------------------rotCounter6 FUNCTION-------------------------------------------------
void rotCounter5()
{
volatile unsigned long cPulse = micros();
volatile const int n = 5;
RPM[n] = ((float) 60.0 / (float) pulsesPerRot[n]) * (float) ((float) 1000000.00 / (cPulse - pulse[n]));
pulse[n] = cPulse;
}
//----------------------------------------------------------------------Accelerometer Functions
//------------------------INITIALIZE ACCELEROMETERS FUNCTION-----------------------------------
void initializeAccelerometers(const int accelerometers[NUMBER_OF_ACCEL][NUMBER_OF_AXES])
{
for (int rea = 0; rea < NUMBER_OF_ACCEL; rea++)
{
for (int rea2 = 0; rea2 < NUMBER_OF_ACCEL; rea2++)
{
pinMode(accelerometers[rea][rea2], INPUT);
}
}
}
//------------------------MEASURE AMPLITUDES FUNCTION------------------------------------------
void readAmplitudes(const int accelerometers[NUMBER_OF_ACCEL][NUMBER_OF_AXES], int accelerometeramplitude[
     NUMBER_OF_ACCEL][NUMBER_OF_AXES])
{
int tempMatrix[NUMBER_OF_ACCEL][NUMBER_OF_AXES];
int minMatrix[NUMBER_OF_ACCEL][NUMBER_OF_AXES];
int maxMatrix[NUMBER_OF_ACCEL][NUMBER_OF_AXES];
for (int q = 0; q < SAMPLE_SIZE; q++)
{
```

```
for (int x = 0; x < NUMBER_OF_ACCEL; x++)
{
for (int y = 0; y < NUMBER_OF_AXES; y++)
{
tempMatrix[x][y] = analogRead(accelerometers[x][y]);
if (q == 0)
{
minMatrix[x][y] = tempMatrix[x][y];
maxMatrix[x][y] = tempMatrix[x][y];
}
if (tempMatrix[x][y] < minMatrix[x][y]) minMatrix[x][y] = tempMatrix[x][y];
if (tempMatrix[x][y] > maxMatrix[x][y]) maxMatrix[x][y] = tempMatrix[x][y];
}
}
}
for (int s = 0; s < NUMBER_OF_ACCEL; s++)
{
for (int t = 0; t < NUMBER_OF_AXES; t++)
{
accelerometeramplitude[s][t] = maxMatrix[s][t] - minMatrix[s][t];
}
}
}
//----------------------------------------------------------------------------Voltage Functions
//------------------------INITIALIZE VOLTAGE SENSORS FUNCTION----------------------------------
void initializeVoltageSensors(const int voltagesensors[])
{
for (int rev = 0; rev < NUMBER_OF_VOLTAGE_SENSORS; rev++)
{
pinMode(voltagesensors[rev], INPUT);
}
}
//------------------------INITIALIZE SWITCH RELAYS FUNCTION------------------------------------
void initializeResolutionRelays(const int relaypins[])
{
for (int rer = 0; rer < NUMBER_OF_VOLTAGE_SENSORS; rer++)
{
pinMode(relaypins[rer], OUTPUT);
digitalWrite(relaypins[rer], LOW);
}
}
//------------------------CHECK IF RELAY IS ACTIVE FUNCTION------------------------------------
bool resolutionRelayActive(int relaypin)
{
bool relayState = digitalRead(relaypin);
if (relayState == true) return true;
else return false;
}
//------------------------AUTOMATICALLY CHANGE VOLT RELAY STATE FUNCTION-----------------------
void selectVoltRelayState(const int relaypins[], const int vSensorPins[], unsigned long comparetime[],
     boolean trigger[])
{
unsigned long referenceTime[NUMBER_OF_VOLTAGE_SENSORS];
int inputV[NUMBER_OF_VOLTAGE_SENSORS];
boolean button[NUMBER_OF_VOLTAGE_SENSORS] {false, false, false};
for (int ssr = 0; ssr < NUMBER_OF_VOLTAGE_SENSORS; ssr++)
{
referenceTime[ssr] = millis();
inputV[ssr] = analogRead(vSensorPins[ssr]);
if (inputV[ssr] > 900) button[ssr] = true;
else if (inputV[ssr] < 450) button[ssr] = false;
else button[ssr] = trigger[ssr];
if (button[ssr] != trigger[ssr]) comparetime[ssr] = referenceTime[ssr];
if (comparetime[ssr] + 500UL < referenceTime[ssr])
```

```
{
if (button[ssr] != resolutionRelayActive(relaypins[ssr])) digitalWrite(relaypins[ssr], button[ssr]);
}
trigger[ssr] = button[ssr];
}
}
//------------------------READ VOLTAGE FUNCTION------------------------------------------------
void readVolt(float voltageval[], int voltagesensors[])
{
for (int coun = 0; coun < NUMBER_OF_VOLTAGE_SENSORS; coun++)
{
if (resolutionRelayActive)
{
voltageval[coun] = (analogRead(voltagesensors[coun]) * AX_axbVOLTAGEACTIVE) + B_axbVOLTAGEACTIVE;
}
else
{
voltageval[coun] = (analogRead(voltagesensors[coun]) * AX_axbVOLTAGE) + B_axbVOLTAGE;
}
}
}
//----------------------------------------------------------------------------Current Functions
//------------------------INITIALIZE CURRENT SENSORS FUNCTION----------------------------------
void initializeCurrentSensors(const int currentsensors[])
{
for (int rec = 0; rec < NUMBER_OF_CURRENT_SENSORS; rec++)
{
pinMode(currentsensors[rec], INPUT);
}
}
//------------------------READ CURRENT FUNCTION------------------------------------------------
void readAmp(float currentval[], int currentsensors[])
{
for (int cou = 0; cou < NUMBER_OF_CURRENT_SENSORS; cou++)
{
currentval[cou] = (analogRead(currentsensors[cou]) * AX_axbCURRENT) + B_axbCURRENT;
}
}
//-----------------------------------------------------------------------Water Sensor Functions
//------------------------INITIALIZE SWITCHES FUNCTION-----------------------------------------
void initializeSwitches(int switchpins[])
{
for (int swi = 0; swi < NUMBER_OF_DIGITAL_SWITCHES; swi++)
{
pinMode(switchpins[swi], INPUT);
digitalWrite(switchpins[swi], HIGH);
}
}
//------------------------READ WATER PRESENCE FUNCTION-----------------------------------------
void readWaterPresence(const int switchpins[], unsigned long presencecomparetime[], bool trigger[],
     boolean switchstate[], const int drainpins[])
{
unsigned long referenceTime[NUMBER_OF_DIGITAL_SWITCHES];
int IO[NUMBER_OF_DIGITAL_SWITCHES];
for (int rwr = 0; rwr < NUMBER_OF_DIGITAL_SWITCHES; rwr++)
{
referenceTime[rwr] = millis();
IO[rwr] = digitalRead(switchpins[rwr]);
if (IO[rwr] != trigger[rwr]) presencecomparetime[rwr] = referenceTime[rwr];
if (presencecomparetime[rwr] + 500UL < referenceTime[rwr])
{
if (IO[rwr] == switchstate[rwr])
{
switchstate[rwr] = !switchstate[rwr];
digitalWrite(drainpins[rwr], switchstate[rwr]);
}
}
trigger[rwr] = IO[rwr];
}
}
//-------------------------INITIALIZE DRAIN RELAYS FUNCTION--------------------------------------
void initializeDrainRelays(const int drainpins[])
{
for (int rer = 0; rer < NUMBER_OF_DIGITAL_SWITCHES; rer++)
{
pinMode(drainpins[rer], OUTPUT);
digitalWrite(drainpins[rer], LOW);
}
}
//-------------------------------------------------------------------------------Relays Functions
//-------------------------INITIALIZE RELAYS FUNCTION--------------------------------------------
void initializeRelays(int relaypins[])
{
for (int re = 0; re < NUMBER_OF_RELAYS; re++)
{
pinMode(relaypins[re], OUTPUT);
digitalWrite(relaypins[re], LOW);
}
}
//-----------------------------------------------------------------------End of Program Functions
//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------------------
//--------------------------PROGRAM SETUP--------------------------------------------------------
void setup()
{ //----------------------------------------------------------------------------------------------
//-----------------------------------------------------------------------------------Serial Setup
Serial.begin(115200);                          //Initialize serial communication and set baudrate
//--------------------------------------------------------------------------------------RTC Setup
//initialize RTC
Wire.begin();
RTC.begin();
if (! RTC.isrunning()) {
Serial.println("RTC is NOT running!");
//The following line sets the RTC to the date & time this sketch was compiled
RTC.adjust(DateTime(__DATE__, __TIME__));
}
RTC.adjust(DateTime(__DATE__, __TIME__));
//-------------------------------------------------------------------------------------WiFi Setup
//Initialize WiFi
mb.config(wifi, wifiNetwork, password);
//Add SENSOR_IREG register - Use addIreg() for analog Inputs
addRegisters(IREGS);
//---------------------------------------------------------------------------------------SD Setup
//Make sure that the default chip select pin is set to
//output, even if you don't use it. For Arduino Mega:
//pinMode(53, OUTPUT);
//Initialize SD card and create or select the log file
initializeLogFile(chipSelect, fileName);
//------------------------------------------------------------------------------Temperature Setup
//Initialize busses
createBusses();
//Discover and allocate addresses connected to the busses
discoverBusAdresses(addrMatrix);
initializeRelays(coolingRelayPins);
//--------------------------------------------------------------------------------------RPM Setup
//Initialize interrupt pins
if (inter == false) {
```

```
attachInterrupts(RPMpin);
inter = true;
}
//--------------------------------------------------------------------------Accelerometer Setup
initializeAccelerometers(accelerometerPins);
//--------------------------------------------------------------------------------Voltage Setup
initializeResolutionRelays(resolutionRelayPins);
initializeVoltageSensors(voltageSensors);
//--------------------------------------------------------------------------------Current Setup
initializeCurrentSensors(currentSensors);
//---------------------------------------------------------------------------Water Sensor Setup
initializeSwitches(switchPins);
initializeDrainRelays(drainPins);
//----------------------------------------------------------------------------------Relay Setup
//initializeRelays(relayPins);
//--------------------------------------------------------------------------------Program Setup
zero = millis() + 3000UL;   //Reference (only start the program when current time is zero time)
PTR = zero;                                                               //Previous time read
ts = millis();                                      //Refernce time for data saving and sending
}//------------------------MAIN CODE-----------------------------------------------------------
void loop()
{ //-------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//---------------------------------------------------------------------------------------------
//----------------------------------------------------------------------------Synchronize Clock
while (millis() < zero) {} //set clock "zero" (to avoid functions running too soon after setup)
CTime = millis();//use CTime to sync several functions (if no sync is needed just use millis())
//------------------------------------------------------------------RPM Sensor Main Code PART 1
//Re-attachInterrupts after modbus transmission
attachInterrupts(RPMpin);
//------------------------------------------------------------------------Temperature Main Code
if (ReadOK) {                                    //If previous temperature reading has finished
requestTemperature();                                 //Send new tempereature reading request
ReadOK = false;
}
if (PTR + ONE <= CTime) {                         //If one second has passed since data request
//celsius[] = {AT1, GT1, MT1, AT2, GT2, MT2, AT3, GT3, MT3, riverTemp}
readTemp(celsius);                                          //Read requested temperature data
for (int nt = 0; nt < NUMBER_OF_BUSSES; nt++)
{ //data2send[0] - data2send[9] for Temperature
data2send[nt] = celsius[nt];
}
ReadOK = true;
PTR += ONE;
}
//Check if cooling needs to be applied. If so, apply cooling
//IMPORTANT!! The following automatically switches relay to keep turbine at a safe temperature
controlCoolingRelayState(coolingRelayPins, celsius, tempCompareTime, tempTrigger, coolingRelayState);
//---------------------------------------------------------------------Voltage Sensor Main Code
//Check if the voltage sensor relay has to be activated, and act accordingly
//IMPORTANT!! The following automatically switches relay to keep pin voltage lower than 5V
selectVoltRelayState(resolutionRelayPins, voltageSensors, compareTime, relayInTransition);
// voltageVal[0] = turbine 1, voltageVal[1] = turbine 2, voltageVal[2] = turbine 3
readVolt(voltageVal, voltageSensors);                                       //Read voltage data
for (int nv = 0; nv < NUMBER_OF_VOLTAGE_SENSORS; nv++)
{ // data2send[10] - data2send[12] for Voltage
data2send[nv + NUMBER_OF_BUSSES] = voltageVal[nv];
}
//-----------------------------------------------------------------------Water Sensor Main Code
//Check if water needs to be pumped out, and act accordingly
//IMPORTANT!! The following automatically switches relay to pump water out of the turbine
```

```
readWaterPresence(switchPins, presenceCompareTime, lastPresenceState, waterPresent, drainPins);
// waterPresent[0] = turbine 1, waterPresent[1] = turbine 2, waterPresent[2] = turbine 3
for (int nw = 0; nw < NUMBER_OF_DIGITAL_SWITCHES; nw++)
{ //data2send[13] - data2send[15] for Water Presence
data2send[nw + NUMBER_OF_BUSSES + NUMBER_OF_VOLTAGE_SENSORS] = waterPresent[nw];
}
//----------------------------------------------------------------------Accelerometer Main Code
// accelerometerAmplitude[0][0] = X-turbine 1, accelerometerAmplitude[0][1] = Y-turbine 1, accelerometerAmplitude[0][2] = Z-turbine 1
// accelerometerAmplitude[1][0] = X-turbine 2, accelerometerAmplitude[1][1] = Y-turbine 2, accelerometerAmplitude[1][2] = Z-turbine 2
// accelerometerAmplitude[2][0] = X-turbine 3, accelerometerAmplitude[2][1] = Y-turbine 3, accelerometerAmplitude[2][2] = Z-turbine 3
readAmplitudes(accelerometerPins, accelerometerAmplitude);     //Read vibration amplitudes data
for (int na = 0; na < NUMBER_OF_ACCEL; na++)
{ //data2send[16] - data2send[24] for Accelerometers
for (int nax = 0; nax < NUMBER_OF_AXES; nax++)
{
data2send[(nax + NUMBER_OF_BUSSES + NUMBER_OF_VOLTAGE_SENSORS + NUMBER_OF_DIGITAL_SWITCHES) + (na * 3)] = accelerometerAmplitude[na][nax];
}
}
//---------------------------------------------------------------------Current Sensor Main Code
// currentVal[0] = turbine 1, currentVal[1] = turbine 2, currentVal[2] = turbine 3
readAmp(currentVal, currentSensors);                                        //Read current data
for (int nc = 0; nc < NUMBER_OF_CURRENT_SENSORS; nc++)
{ //data2send[25] - data2send[27] for Current
data2send[nc + NUMBER_OF_BUSSES + NUMBER_OF_VOLTAGE_SENSORS + NUMBER_OF_DIGITAL_SWITCHES + (NUMBER_OF_ACCEL * NUMBER_OF_AXES)] = currentVal[nc];
}
//------------------------------------------------------------------RPM Sensor Main Code PART 2
// RPM[0] = turbine 1, RPM[1] = turbine 2, RPM[2] = turbine 3
checkZeroRPM(RPM, pulse);                      //Check if RPM is zero, if so change RPM to zero
for (int nr = 0; nr < NUMBER_OF_CURRENT_SENSORS; nr++)
{ //data2send[28] - data2send[30] for RPM
data2send[nr + NUMBER_OF_BUSSES + NUMBER_OF_VOLTAGE_SENSORS + NUMBER_OF_DIGITAL_SWITCHES + (NUMBER_OF_ACCEL * NUMBER_OF_AXES) + NUMBER_OF_CURRENT_SENSORS] = RPM[nr];
}
detachInterrupts(RPMpin);                     //DetachInterrupts for better modbus transmission
inter = false;
//----------------------------------------------------------------------------Debugging Purpose
//                                                                                          //
//WARNING!! DO NOT CONNECT THE ARDUINO TO ANY COMPUTER WHEN HIGH VOLTAGES ARE BEING MEASURED//
/*
for (int n = 0; n < NUMBER_OF_REGISTERS; n++)
{
data2send[n] = n + 1;
Serial.print(data2send[n]);                   //Serial print data that will be saved and sent
Serial.print("\t");
}
//Serial.println();
for (int j = 0; j < NUMBER_OF_VOLTAGE_SENSORS; j++)
{
Serial.print(voltageVal[j]);                                      //Serial print voltage data
Serial.print("\t");
Serial.print(resolutionRelayActive(resolutionRelayPins[j]));//Serial print relay is on or off
Serial.print("\t");
}
//Serial.println();
for (int tint = 0; tint < NUMBER_OF_BUSSES; tint++)
{
Serial.print(celsius[tint]);                                  //Serial print temperature data
Serial.print(" C ");
}
```

```
//Serial.println();
for (int i = 0; i < NUMBER_OF_DIGITAL_SWITCHES; i++)
{
Serial.print(waterPresent[i]);                                  //Serial print water presence
Serial.print("\t");
}
//Serial.println();
for (int l = 0; l < NUMBER_OF_ACCEL; l++)
{
for (int m = 0; m < NUMBER_OF_ACCEL; m++)
{
Serial.print(accelerometerAmplitude[l][m]);                   //Serial print vibration data
Serial.print("\t");
}
Serial.println();
}
//Serial.println();
for (int jee = 0; jee < NUMBER_OF_CURRENT_SENSORS; jee++)
{
Serial.print(currentVal[jee]);                                    //Serial print current data
Serial.print("\t");
}
//Serial.println();
for (int c; c < NUMBER_OF_INTERRUPTS; c++)
{
Serial.print(RPM[c]);                                                 //Serial print RPM data
Serial.print("\t");
}
Serial.println();
*/
//---------------------------------------------------Modbus, SD, Serial (Data Output) Main Code
//Initialize modbus
mb.task();
//Print each three hundred miliseconds
if (millis() > ts + 300)
{
writingDataToSD(fileName, data2send);
endLineSD(chipSelect, fileName);
// send data2send via modbus
for (int dn = 0; dn < NUMBER_OF_REGISTERS; dn++)
{
Serial.print(data2send[dn]);
Serial.print("\t");
mb.Ireg(IREGS[dn], data2send[dn]);
}
ts = millis();
}
//----------------------------------------------------------------------------Debugging Purpose
/*Serial.println(millis() - CTime);*/                         //Serial print time spent on loop
}//-------------------------END-----------------------------------------------------------------
//--------------------------------------------------------------------------------End of Sketch
```

The sketch file displayed above is available at:<https://mega.nz/#!aJhGEC5D!hmLKN-enZim1f8_C2-g4qqHUhVSgjJ-X3omA0N3A5C8>

APPENDIX I – Sensor calibration

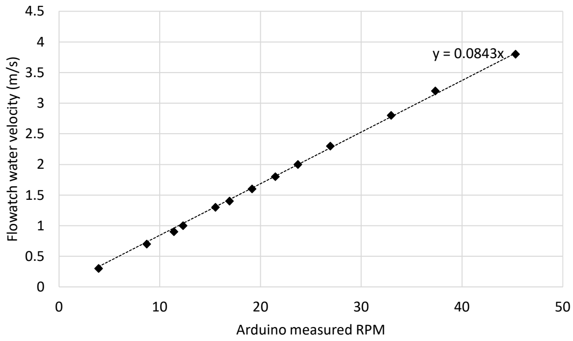


Figure 75 – Calibration curve of the river velocity sensor (impeller)

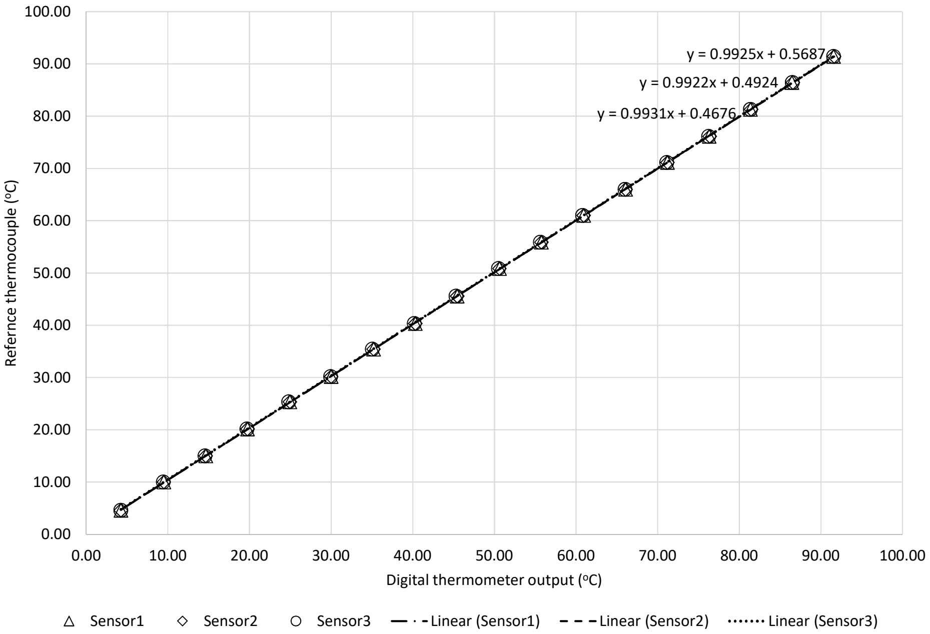


Figure 76 – Calibration curve of the digital temperature sensors

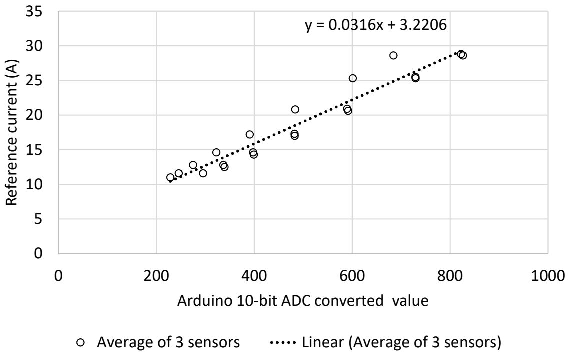


Figure 77 – Calibration curve of current transformers

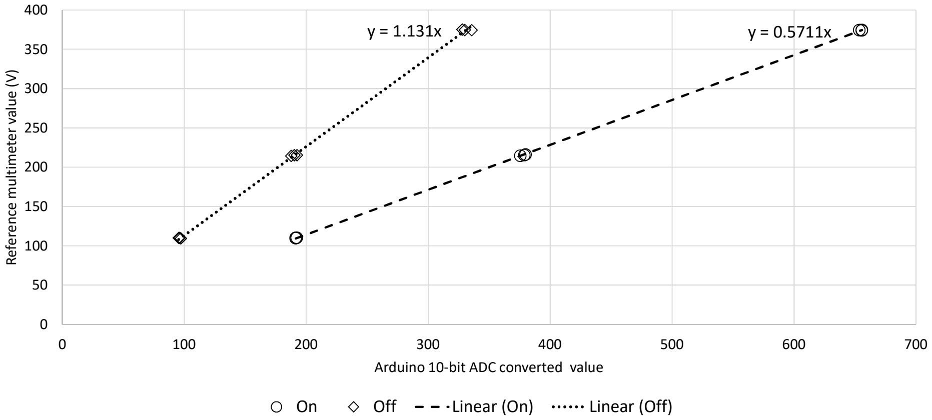


Figure 78 – Calibration curve of the voltage dividers