

Álvaro Brándež Górriz

# **Design, Implementation, and Testing of Hardware for Sensing and Controlling the Dynamics of Rotor-Bladed Systems**

Master's Thesis, February 2019

Álvaro Brández Górriz

# **Design, Implementation, and Testing of Hardware for Sensing and Controlling the Dynamics of Rotor-Bladed Systems**

Master's Thesis, February 2019



# Design, Implementation, and Testing of Hardware for Sensing and Controlling the Dynamics of Rotor-Bladed Systems

**Author(s):**

Álvaro Brándež Górriz

**Supervisor(s):**

Ilmar Ferreira Santos, Professor Dr.-Ing. Dr. Tech., MEK, DTU

**Department of Electrical Engineering**

Centre for Electric Power and Energy (CEE)

Technical University of Denmark

Elektrovej, Building 325

DK-2800 Kgs. Lyngby

Denmark

[www.elektro.dtu.dk/cee](http://www.elektro.dtu.dk/cee)

Tel: (+45) 45 25 35 00

Fax: (+45) 45 88 61 11

E-mail: [cee@elektro.dtu.dk](mailto:cee@elektro.dtu.dk)

---

Release date: February 24nd, 2019.

Class: 1 (Public)

Edition: First

Comments: This report is part of the requirements to achieve the Master of Science in Engineering (M.Sc.Eng.) at the Technical University of Denmark. This report represents 30 ECTS points.



# PREFACE

---

This thesis is the final written submission of the Electrical Engineering M.Sc. degree at the Technical University of Denmark. This master thesis is part of a bigger project including other students: Maria Beneyto Gomez-Polo [1] and Ignacio Escudero Sarabia [2]. The other two thesis depended on the results of this one; this lead to some decisions that affected the overall scope of this master thesis project.

The whole project was supervised by Professor Ilmar F. Santos who encouraged the project to be developed. The research in this thesis is built on several other projects developed in this specific field. The main source of the information needed was extracted from the work of R.H Christensen [3], however the reports of Christian S. Jakobsen [4] and Jesper B. Hansen [5] were really helpful.

# ABSTRACT

---

In this thesis, the development of the hardware and software of active vibration control in a rotor-blade system is researched. The implementation of feedback loop control in real systems has several issues that will be discussed in this report. The development of this kind of projects has a proper methodology that is also detailed in the thesis.

The project is based on the specific rotor-blade system simulated by a test-rig located in the DTU Mechanical Engineering department. This structure had already a complete active vibration control system, however the results were not satisfactory when some control strategies were implemented. The purpose of this thesis is also to analyze the characteristics of the previous control system and describe its weak points.

Once the analysis is performed, a new system is designed; the new design is based on embedded technology. The three main components of the feedback control loop are updated: sensors, actuators and control unit. Moreover, an improvement of the interconnections of the system is implemented adding wireless technology to the communications in the active vibration control system. The development of the software is also present in this report.

The active vibrations control loop software and hardware are implemented in the actual test-rig and the performance of the embedded system and its components is tested. The results are analyzed and some possible future improvements were included.

# TABLE OF CONTENTS

---

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 State of the art . . . . .	1
1.2 Scope . . . . .	2
<b>2 Test-Rig</b>	<b>3</b>
2.1 Physical description . . . . .	3
2.2 Mathematical model . . . . .	4
2.2.1 Model dynamics . . . . .	6
2.3 Initial electronics . . . . .	6
2.3.1 Sensors . . . . .	7
2.3.2 Actuators . . . . .	8
2.3.3 Control unit . . . . .	8
2.3.4 Initial state of the electronics . . . . .	8
<b>3 Sensing System</b>	<b>9</b>
3.1 General considerations . . . . .	9
3.2 Hub sensors . . . . .	10
3.3 Blade sensors . . . . .	12
3.4 Amplifying electronics . . . . .	16
3.4.1 Powering of electronics . . . . .	17
3.5 Conclusions . . . . .	18
<b>4 Actuating system</b>	<b>20</b>
4.1 General considerations . . . . .	20
4.2 Actuators . . . . .	21
4.2.1 DC motor . . . . .	25
4.3 Power electronics . . . . .	25
4.3.1 Dynamics of the power electronics . . . . .	28
4.4 Conclusions . . . . .	28
<b>5 Control unit</b>	<b>31</b>
5.1 Initial control system . . . . .	31
5.2 Control unit design . . . . .	32
5.2.1 Requirements . . . . .	33

5.2.2	Design criteria . . . . .	34
5.2.3	Development environment . . . . .	37
5.3	Embedded Software . . . . .	38
5.3.1	Hardware and software configuration . . . . .	39
5.3.2	Code description . . . . .	40
5.4	Software testing . . . . .	47
5.5	Conclusions . . . . .	47
<b>6</b>	<b>System interconnections</b>	<b>49</b>
6.1	Previous interconnection structure . . . . .	49
6.2	New wiring configuration . . . . .	50
6.3	Wireless connection . . . . .	53
6.3.1	First implementation and results . . . . .	53
6.3.2	Wi-fi protocol improvement . . . . .	55
6.4	Final system configuration . . . . .	57
<b>7</b>	<b>Conclusion</b>	<b>58</b>
7.1	Results . . . . .	58
7.1.1	Perspectives . . . . .	59
7.2	Future work . . . . .	59
	<b>References</b>	<b>60</b>
<b>A</b>	<b>Planes and data sheets</b>	<b>63</b>
<b>B</b>	<b>Software code</b>	<b>86</b>

# LIST OF FIGURES

---

2.1	Picture taken of the used test rig . . . . .	3
2.2	Mathematical model of the rotor-blade system . . . . .	4
2.3	Rotor and blade physical and geometrical properties. By Rene . . . . .	5
2.4	Basic feedback control loop by ISA . . . . .	6
2.5	Schematic of the active vibration control setup by Rene . . . . .	7
3.1	Continuous blade scheme [3] . . . . .	9
3.2	First 3 mode shapes of blade 1 [5] . . . . .	10
3.3	Hub eddy current sensors (7&8) [3] . . . . .	10
3.4	Hub eddy current sensor (9) [3] . . . . .	11
3.5	Form of a resistance strain gauge [6] . . . . .	12
3.6	Form of a Wheatstone bridge [7] . . . . .	12
3.7	Previous electronics inside the rigid disk [3] . . . . .	13
3.8	Table with different sensors [8] . . . . .	14
3.9	Designed location for blade sensor [3] . . . . .	15
3.10	Axonometric view of the sensor supporting structure . . . . .	15
3.11	Sensor mounted into the blades . . . . .	16
3.12	Scheme of non-inverting amplifier [2] . . . . .	17
3.13	Non-inverting amplifiers in the PCB . . . . .	17
3.14	Signals from the sensors at 300 rpm . . . . .	18
3.15	Signals from the blade sensor with a zero-input response . . . . .	19
4.1	Schematic of the actuating setup [3] . . . . .	20
4.2	Schematic of the electromagnet physics [9] . . . . .	21
4.3	Electromagnet RS PRO 65 mm . . . . .	22
4.4	Theoretical waterfall diagram showing normalized frequency responses of the rotor (a) and the blade movement (b) as function of the rotor angular velocity [3] . . . . .	22
4.5	Electric circuit of electromagnet [1] . . . . .	23
4.6	Desired voltage (-) vs output voltage(-) [3] . . . . .	24
4.7	DC motor implementation . . . . .	25
4.8	H-bridge schematics . . . . .	26
4.9	H-bridge driver by ST electronics . . . . .	27
4.10	Power electronics inside the rigid disk . . . . .	27
4.11	Effects of PWM frequency by [10] . . . . .	28
4.12	Force of the magnet at 5 hertz . . . . .	29
4.13	Frequency spectrum with 5 Hz sinusoidal input . . . . .	29
4.14	Experimental bode of the electromagnets . . . . .	30
5.1	dspace DS1103 PPC . . . . .	32
5.2	Schematic of the microcontroller internal setup [11] . . . . .	33
5.3	Schematic of MCU requirements . . . . .	34

5.4	Arduino MEGA 2560 Rev3 . . . . .	35
5.5	ST electronics STM32F411RET6 . . . . .	36
5.6	Raspberry pi 3 model b+ . . . . .	36
5.7	Configuration of the MCUs and the computer . . . . .	39
5.8	AD converter ADS1105 by Texas Instruments . . . . .	40
5.9	Software PWM generation . . . . .	42
5.10	Control loop scheme in both Raspberry Pi´s . . . . .	43
5.11	Control loop scheme in the Linux CPU . . . . .	44
5.12	Configuration of the final software system . . . . .	48
6.1	Slip-ring internal schematics . . . . .	49
6.2	Electronics that handle the actuating signals . . . . .	50
6.3	Slip-ring installed in the test-rig . . . . .	51
6.4	Connections inside the rotating disk: Power drivers . . . . .	52
6.5	Connections inside the rotating disk: PCB, ADC and Raspberry Pi . . . . .	52
6.6	Wireless router installed: TP-Link Trådløs Gigabit Wi-Fi . . . . .	53
6.7	TCP protocol header format . . . . .	54
6.8	Definite system configuration . . . . .	57
7.1	Final state of the Test-rig . . . . .	59



# 1 INTRODUCTION

---

Rotor-bladed systems are present in many different industries, such as turbines, compressors or electrical motors. Industry is demanding more efficiency and durability to these systems. Performance of rotor-bladed systems with rotating dynamics is strictly connected to vibration reduction. In order to reduce vibrations, two main paths arise: Active and passive control.

Active control of vibrations with electromechanical components is one of many approaches to the problem. This technique consists of continuously detecting oscillations in the system and then externally actuate in the system to reduce vibrations. Choosing active control comes with several issues regarding sensing, actuating and controlling algorithms.

In order to be able to control a rotating system, compactness and robustness are key to perform a satisfactory of active vibration control. The position and size of each element could affect the dynamics of the system. The approaches used so far are based on general purpose computer systems [12]. In these systems, a computer gets the information from the sensors and sends the control signal to the actuators. The control feedback loop is closed with these three elements.

## 1.1 State of the art

Implementing feedback control in many different systems has become more and more feasible within the past years. Both hardware and software are continuously improving in performance and cost. Since the 1960's compact electronic systems defined as embedded systems have been used in many different applications.

An embedded system is based on a microcontroller instead of a computer and it is focused on one or few tasks. The main advantage of embedded systems is compactness and reduced cost. However, the development of applications in microcontrollers takes more time than the same application in a general purpose computer.

Currently, embedded systems are being used in many different applications. From solutions in high tech industry to small homemade projects. There is in the market a huge variety of microcontrollers from different distributors such as: Atmel, STM or Microchip.

The last issue regarding this applications is the implementation of Internet of Things (IoT). IoT is the concept of interconnecting all the subsystems within a wireless network. This new paradigm offers substantial possible advantages regarding the possibilities of sharing information between all the different systems in an industry. This innovation permits the unification of smaller subsystems into a unique system. In consequence of this trend, embedded systems are now able to be wireless connected within each other, offering new possibilities to the performance of this type of compact systems.

## 1.2 Scope

The objective of this project is to develop an electronic system that is able to perform active vibration control in a rotor-bladed system. Specifically, the control is going to be implemented in a test-rig of the mechanical department of DTU.

Previous approaches have been implemented into the test-rig and part of this project is to analyze the previous state of the system. This analysis will conclude which parts of the system have to be change and which ones can still be used.

The requirements of the system will be chosen after the analysis of the test-rig. The characteristic of the new system regarding sensing, actuation and control will need to be sufficiently satisfactory to implement successfully any kind of control algorithm.

This report will be organized in several chapters, the first one will analyze in detail the test-rig and the initial state when the project started. The next two chapters will describe the sensing and actuating subsystems. The report will continue with the control unit and communications and finally some conclusion with a discussion of the results of the thesis. The thesis is said to be concluded with the implementation of an embedded system in the test-rig that is capable to perform active control of vibrations of the rotor-blade system.

## 2 TEST-RIG

---

This chapter is a description of the test-rig, first the physical characteristics and second the electrical components that were implemented at the beginning of the project.

### 2.1 Physical description

The test-rig is a rotor-bladed system, this mechanical structure was designed to emulate other mechanic systems such as: turbines, compressors or pumps.

The mechanic structure starts with a fixed external frame made of steel. This supporting structure is going to be the inertial frame of reference. Attached to the external frame is the hub. The hub is the mechanical structure where the rotor is located. The hub joins with the external frame through several flexible beams made of a metal whose characteristics are known. Coupled to the hub through two bearings is the shaft. The rotor is mounted to the shaft . This component transmits the forces from the rotor to the hub and vice versa.

The rotor is a steel circular structure. This metallic disk has within it, four blades. Each blade is a metallic beam with known mechanical characteristics. The blade is radially clamped to the center of the rotor. In the tip of each beam there is a mass in order to accentuate the vibrations. The layout of the test rig is shown below on Figure 2.1.



*Figure 2.1.* Picture taken of the used test rig



The final mathematical model is described on Figure 2.2. The reference, as explained before is the external frame, as it is considered as completely rigid and fixed. Therefore, horizontal ( $x_h$ ) and vertical ( $y_h$ ) movements of the hub are parallel to the respective edges of the external frame. Located on the frame are the strings and dampers that represent the hub suspension.

The angular position of the blades is determined by the angle between the horizontal axis and the first blade with the rest of the blades referenced to this first blade. As it is a two dimensional scheme, the angular position and velocity are explained with  $\theta$  and  $\Omega$ .

The forces of the actuators are here represented by:  $F_{hx}$  and  $F_{hy}$  for the shaft forces and  $F_b$  for the force applied to the blade. The position of the forces depend on the allocation of the actuators. The actuators that generate each forces will be described in the next sections.

The measurements considered in this model are the displacements of the hub and the tip of the blades. The measurement of the displacement of the hub coincides with the vertical and horizontal axes. On the other hand, the displacement of the blades is located to its own local reference frame. This displacement  $d_i$  is orthogonal to the surface of the blade.

The different physical parameters that are needed to complete the description of the physical model are in Figure 2.3

<b>Rotor/Hub</b>			
Mass	$m_{hx}$	10.5	[kg]
	$m_{hy}$	8.6	[kg]
Stiffness <sup>†</sup>	$K_{hx}$	$66 \cdot 10^3$	[N/m]
	$K_{hy}$	$77 \cdot 10^3$	[N/m]
Damping <sup>†</sup>	$D_{hx}$	$1.2 \cdot 10^{-6}$	[N·s/m]
	$D_{hy}$	$1.5 \cdot 10^{-6}$	[N·s/m]
Eccentricity	$\epsilon$	$1 \cdot 10^{-3}$	[m]
	$\kappa$	0	[rad]
Diameter	$r$	0.04	[m]
Rotating disk mass	$m_r$	3.0	[kg]
<b>Blades and Tip masses</b>			
Length	$L_{bi}$	$80 \cdot 10^{-3}$	[m]
Distributed mass	$\rho_{bi}$	0.195	[kg/m]
Elasticity	$E_{bi}$	$2.0 \cdot 10^{11}$	[N/m <sup>2</sup> ]
Moment of inertia (area)	$I_{bi}$	$2.08 \cdot 10^{-12}$	[m <sup>4</sup> ]
Damping <sup>†</sup>	$D_{bi}$	0.8	[N·s/m]
Tip mass	$m_{ti}$	0.109	[kg]
Tip mass moment of inertia	$J_{ti}$	$3.35 \cdot 10^{-5}$	[kg·m <sup>2</sup> ]
Tip mass length	$L_{ti}$	$30 \cdot 10^{-3}$	[m]
Blade angular location ( $i = 1, 2, 3, 4$ )	$\alpha_i$	$(i - 1) \cdot \frac{\pi}{2}$	[rad]

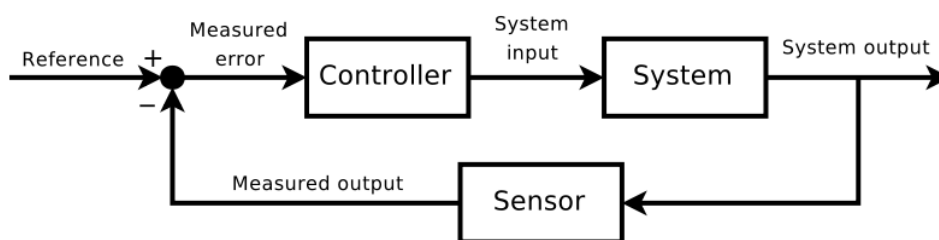
**Figure 2.3.** Rotor and blade physical and geometrical properties. By Rene

### 2.2.1 Model dynamics

Once the mathematical model is defined, an analysis of the dynamics of the system is realized. The dynamics can be described based on previous thesis or articles [13] [12] [3]. The hub displacements are modelled as a damped-mass string system whereas the blade's dynamics are described with the previous explained mode shapes. The natural frequency of the vibrations of the hub is around 11-13 hertz while the blade's natural frequency is 18 hertz. These frequencies will be key in diverse design decisions regarding sensing, actuating and the controller. On the other hand the couplings are also non negligible and have to be taken into account in the controlling algorithm. Further detail of the model dynamics in [12] [1].

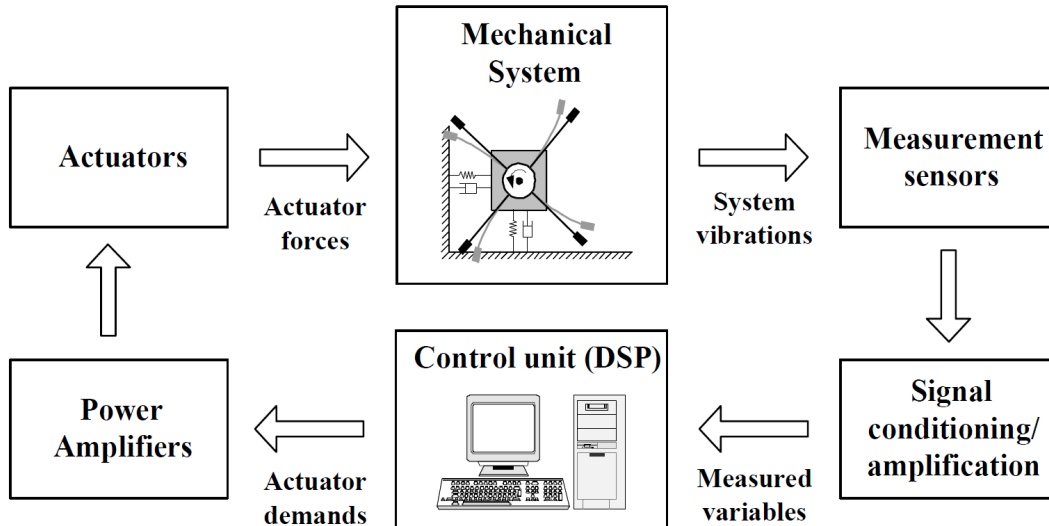
## 2.3 Initial electronics

The initial set up was designed entirely to test active control of vibrations in the rotor-blade system. In order to perform such control several units are needed. The electrical system has to sustain the basic feedback control. This feedback loop includes sensing, compute and process the control signal based on this sensing and finally actuating the control signal into the system. An example of basic control feedback diagram is explained in Figure 2.4, this diagram is taken from the international society of automation (ISA).



*Figure 2.4.* Basic feedback control loop by ISA

The electronics located in the system before the start of this project had all the different elements shown in Figure 2.4. In [3], the complete loop diagram of active control is already defined. In Figure 2.5 it is shown the initial set up for this test-rig. The different units of the diagram are going to be presented in this chapter. As it is seen on the diagram, the active vibration control system is based on general purpose computer systems.



*Figure 2.5.* Schematic of the active vibration control setup by Rene

### 2.3.1 Sensors

The sensing system of the active control setup is responsible to feed the controller with the output of the system. As the setup is designed to control every movement of the rotor-blade system (Rotor and blades), the information that the sensors send into the controller has to be at least sufficient to calculate all these displacements (Rotor and blades).

In order to fulfill the requirements of the active control system, the sensing system is divided into two subsystems. Firstly, the actual sensors and secondly the electrical circuit that adapts the sensors output in order to fit into the control unit.

The sensors election is key in order to achieve a proper control. For sensing the vibrations of the hub, a huge variety of solutions are available in the sensors market . The solution chosen for he test-rig was to use two eddy-current sensors in each direction of the hub. The explanation in detail of this sensors is located in Chapter 3.

However, the blades sensing is specifically problematic because the different bending modes of the blade. Depending on which motion it is required to detect, different solutions arise. In [3] it is already discussed how to choose the position and type for the sensor. The approach that was implemented into the test-rig is a strain-gauge. The strain-gauges were located at 40 mm from the start of the blade. This sensors do not measure directly the displacement of the blade, they sense the blade deflection.

All these measured signals are of different natures and voltages. Therefore an other subsystem is required in order to adapt these signals for the control unit. This signal treatment stage has to be designed taking into consideration the amplitude of the sensor signal, the capacity of the control unit acquisition system, the signal noise and the frequencies that need to be processed.

### 2.3.2 Actuators

The actuating system is in charge of converting the control signal into an actual input to the system. The controllability of the rotor-bladed system depends on these actuators. The actuators present in this test-rig are electromagnets. These are located by pairs in each direction of the possible motions. For the hub there is 4 magnets with one pair of magnets for each direction (vertical and horizontal). In the case of the blades, one pair of magnets is located on the rigid disk, these magnets are able to input a force orthogonally to the surface of the blade.

The control signal generated by the control unit is not sufficiently powerful to actuate the electromagnets. Power electronic circuits are needed in order to feed the actuators with the proper signal. In this specific test-rig, dc amplifiers were used.

### 2.3.3 Control unit

The control unit is the responsible of analyzing the signal of the sensing system and compute a control signal that is sent to the actuating system.

As it was defined at the beginning of the description of the initial state of the electronic system, the control unit is based on a general purpose computer. In the case of this test-rig, a commercial digital control unit is in charge of all the computations. The computer works in digital, and the signals from the sensors and actuators are analogical. Therefore an interface between signals is needed. Analogical to digital converters and vice versa are used in a huge variety of applications. There is an infinity of variations of this devices. In this case, the digital signal processor is dSPACE. The characteristics of this device will be detailed explained in chapter 5.

### 2.3.4 Initial state of the electronics

At the beginning of this project, the systems above explained, were tested. First, information of the previous experimental results was gathered. Several previous projects [4] [14] [5] used this set up and had several problems. Added to this, different tests were applied to the different parts of the active control loop. In the next three chapters, different problems and solutions will be discussed.



# 3 SENSING SYSTEM

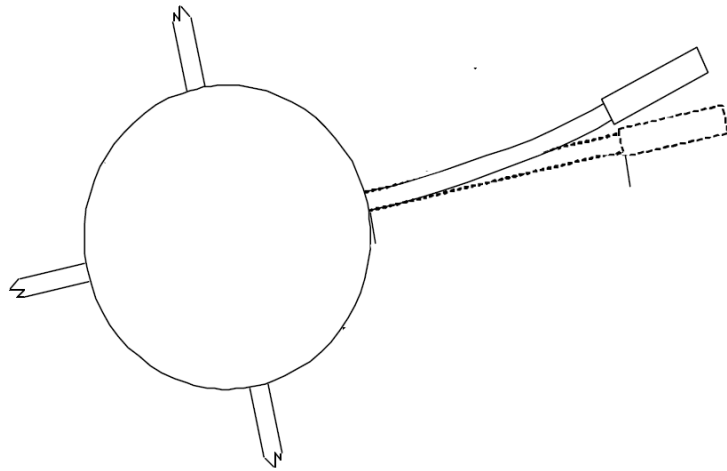
---

The scope of this chapter is to explain in detail the requirements of rotor-blade systems in general and this test-rig in particular regarding the sensing system.

The information available of any system is really diverse. It has to be defined first, what information it is needed. In chapter 2, the mathematical model was defined. The vibrations of the hub are defined by its two displacements (horizontal and vertical). However, as it is explained in [3] and [12] the blade's motion is more complex. Therefore, previous to the definition of the information needed, a discussion among the blade's sensing is needed.

## 3.1 General considerations

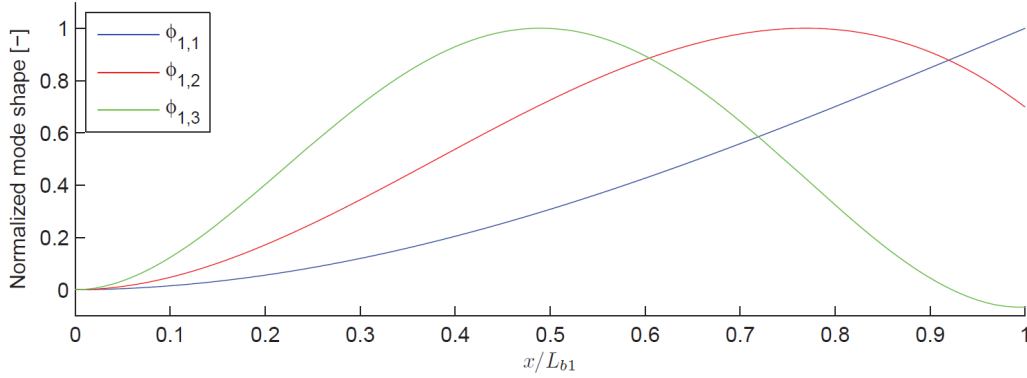
The blades are a continuous system with infinite degrees of freedom. In order to be capable of explain the motion of this continuous system, the degrees of freedom will be discretized into several mode shapes. Mode shapes are patterns of motion of a system. Figure 3.1 represents the scheme of blade's movement.



*Figure 3.1.* Continuous blade scheme [3]

The blade motion can be separated into infinite number of mode shapes. Nevertheless, the importance of each mode shape is not equal in the overall movement of the blade. It is chosen by the designer of the model the quantity of mode shapes used to define the system. The more mode shapes, the more accuracy of the system. However, including many mode shapes would complicate the equations exponentially, in this trade-off, it was chosen 2 mode shapes [3].

This first and second mode have each one natural frequency, this has to be taken into account when the signal is processed. However, the issue that will be discussed in this section is how the mode shapes behave along the blade longitude, and how the sensing system will get the maximum information about them.

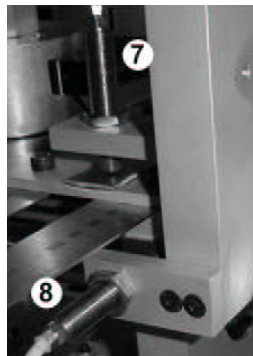


**Figure 3.2.** First 3 mode shapes of blade 1 [5]

In both [5] and [3] the mode shapes form has been defined. In Figure 3.5, it is displayed the form of three first mode shapes of the blade along its longitude. As the mode shapes are superposed in the blade in every moment, only knowing the displacement of every infinitesimal section of the blade would conclude in having the complete description of the blade's motion. This is no feasible in reality, for simplicity of the sensing system only one sensor was located in each blade. From the points above and Figure 3.5 it is derived that the location of the sensor is key in terms of which mode shapes are acknowledged.

### 3.2 Hub sensors

Firstly, as it explained above, the motion of the hub is defined by measuring the horizontal and vertical movements. The sensors used in this test-rig at the beginning of this thesis were a pair of eddy-current sensors measuring the displacements of certain little metallic plates located for this purpose.



**Figure 3.3.** Hub eddy current sensors (7&8) [3]

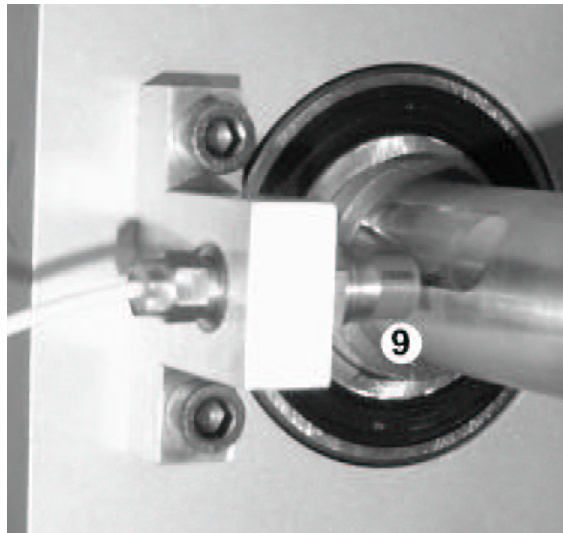
Eddy-Current sensors are based on magnetic fields technology. The driver creates a current in the sensing coil in the end of the probe. This generates a magnetic field which induces small currents in the target material; these currents are called eddy currents.

The eddy currents create an opposing magnetic field which reacts to the field generated by the probe. The amplitude of the generated magnetic field depends on the distance between the probe and the target. The electronics inside the sensor produce a voltage output which is proportional to the change in distance between the probe and target.

The model of the sensors present in the hub is Pulsotronic kj4-m12mn50-anu, this sensor has been updated since the hub probes were installed, however, the site of the producer assures that the main characteristics remain the same. The sensor datasheet is given in the Appendix A . The initial location of the sensors is unknown because the pieces that sustained them were missing and had to be built again.

The sensitivity was calculated by experimental tests because the site of company does not specify. Both sensors had a 2.67 volts per millimeter. The sensor is completely linear, that means that the initial distance will not affect the results of the sensing. It was decided that the maximum displacement in the hub was 3 millimeters in total. The location of the sensors was designed in order to make the zero of the sensor coincide with the zero distance to the left and top magnet. In conclusion the final signal varies between approximately 0 and 8 volts.

Apart from these two sensors, an additional sensor is located in the hub to measure the angular velocity of the shaft. A mark is found in the beginning of the shaft, an eddy-current sensor detects the change in depth of the radial dimension of the shaft. the angular velocity can be calculated with the time between two detections of the mark.



*Figure 3.4.* Hub eddy current sensor (9) [3]

### 3.3 Blade sensors

The blade measures 80 mm in length, the solution taken previous to this project was to locate a strain gauge at 40 mm from the clamp of the blade. The objective was to be able to acknowledge both the first and second bending mode.

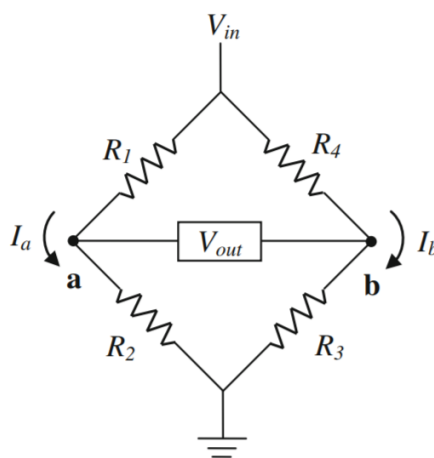
The strain gauge is a common sensor used to measured deflections [15] [6]. It has been used for more than half a century and still it is utilized in many present applications. The strain gauge technology is based on the change of resistance of a wire (within certain limits) when it is strained. This change of resistance can be easily measured.



*Figure 3.5.* Form of a resistance strain gauge [6]

In order to measure this change in the resistance, an additional electrical circuit is needed. There are several options for measuring this, however, the most common solution, and the one that was adopted in this test-rig, is called the Wheatstone bridge [6].

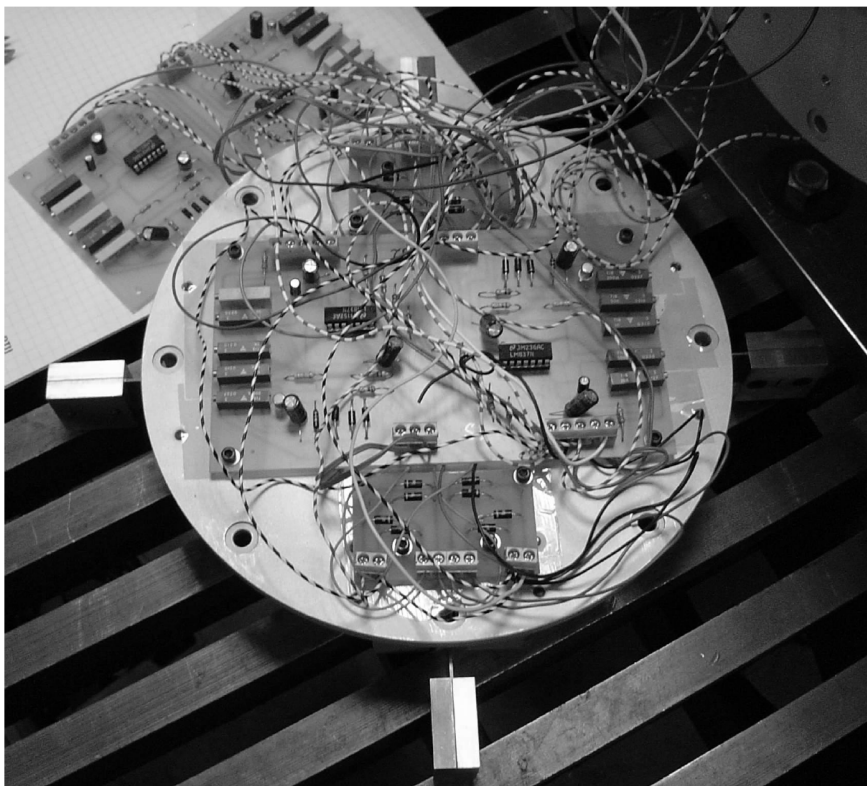
A Wheatstone bridge is an electrical circuit used to measure a very small change in resistance [7], of the order of one per thousand compared to the original resistance of the gauge. The Wheatstone bridge consists of four resistors arranged in a diamond configuration Figure 3.6. A DC voltage is used to excite the circuit, this voltage difference between the top and the bottom of the diamond generates a voltage difference in the middle of the diamond. This output voltage results from the change in resistance from the gauge and its proportional to the strain in the blade.



*Figure 3.6.* Form of a Wheatstone bridge [7]

In the case of the test-rig, two strain gauges were located in the blade, one in each side. These 2 variable resistances can be placed in several configurations depending on your sensing objectives. In [3] it is not specified which one was used, but after analyzing the wiring of the actual circuit, the configuration was determined. The two gauges were connected one opposite to the other, this configuration doubles the amplitude of the output voltage, adding sensibility to the system.

The output voltage of the Wheatstone bridge is very small, therefore, an amplification of the signal is needed. After opening the rigid disk (Figure 3.8) and look into the electronics inside, operational amplifiers are found. This amplifiers are the basis of any amplifying electronic circuit. The wires were damaged, therefore, the original configuration was not possible to be determined. However, it is clear that the signal of the Wheatstone bridge was fed to the operational amplifiers and converted into a higher amplitude.



*Figure 3.7.* Previous electronics inside the rigid disk [3]

Right after the operational amplifiers, an analog filter is located. This filter is based on a resistor-capacitor (RC) circuit. This is a first order filter and can be used as a high or low pass filter. In this case, it is documented [4] that it is used as a low pass filter, however, its cutoff frequency it is unknown.

After the complete analysis of the sensing circuit, the final sensitivity of the system is tried to be obtained. As the original circuit was damaged, a new Wheatstone bridge and amplification must be created. Once the new signal is generated, the noise will be filtered with a new circuit.

The first issue that appeared was the soldering of the gauges to the wires of the Wheatstone bridge. The soldering was damaged, the vibrations of the blades led to the degradation of the joints of the gauge with the circuit. The gauges were re-soldered, their null strain resistance was measured, 120 ohms. The Wheatstone bridge was build and amplified with an instrumental amplifier.

Once the circuit was implemented, it was tested. A strain was applied into the blade so that the deflection of the blade was just the maximum permitted physically. The signal was measured with an oscilloscope and the results were not optimal. The amplification needed to sense a minimum voltage was adding noise to the signal to a point of making impossible to obtain decent data from it.

In [5] the same issue is discussed, in order to obtain real information of the deflection of the blade, the filtering would cause a delay into the information fed into the control unit.

Other solutions arouse in order to improve the sensing system. The first point is to research sensors that suit the test-rig application, the main points being compactness, sensitivity and noise. Various solutions are available for non-contact proximity sensors.

Item	Non-contact type			
	Optical type	Eddy current type	Ultrasonic type	Laser focus method
Detection targets	Most targets	Metal	Most targets	Most targets
Measurement distance	Normal	Short	Long	Short
Accuracy	High	High	Low	High
Response speed	Fast	Fast	Slow	Normal
Dust, water, oil, etc.	Normal	Strong	Normal	Normal
Measurement surface	Small	Normal	Large	Small

**Figure 3.8.** Table with different sensors [8]

As the distance is significantly short, the order of millimeters, only laser and eddy-currents are suitable. As eddy-current sensors were already being used and the results were satisfactory, it was decided to use this type of sensor also in the blades.

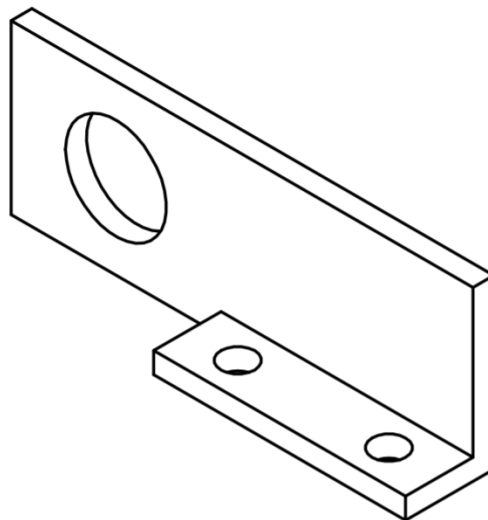
The location of the sensor is the most critical part because the size of the sensor is not negligible, Figure 3.3. In the description of the sensing of the blade, the different effects of the position of the sensor are detailed. In Figure 3.5, the magnitude of each mode shape depending on the position is shown. Taking into consideration these two points, the chosen point is to measure right below the mass of the tip of the blade.



*Figure 3.9.* Designed location for blade sensor [3]

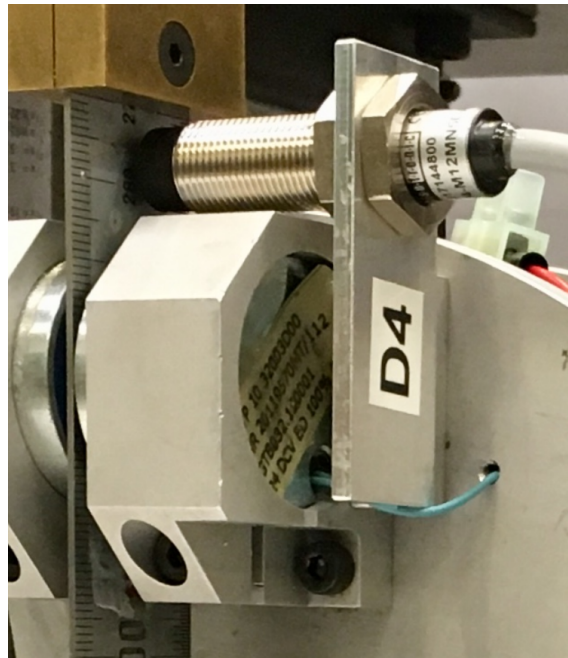
In order to locate an inductive sensor in that position, a new constructive element is needed. Taking profit of the already built structure on the rigid disk, the supporting piece will be clamped to the same holes as the magnets seen in Figure 3.9 (position 4).

Once decided where to fix the piece, the next step is to measure the distances from the wholes to the place where the sensor is designed to be allocated. For this purpose, 3d designing software was used. The scheme of the piece was developed in Solidworks. The resulting 3d model is shown in Figure 3.10.



*Figure 3.10.* Axonometric view of the sensor supporting structure

A prototype was 3D printed before the final piece was sent to production. The measurements taken were not accurate and therefore, the dimensions were readjusted. The final measurements and the complete planes can be found in Appendix B. The final mounted piece is shown in Figure 3.11. The final sensor is located at 62.5 mm from the beginning of the blade. This derives in obtaining a predominant signal from the first mode shape



*Figure 3.11.* Sensor mounted into the blades

With the objective of having the same sensitivity in all the sensors, the sensor chosen is the same as in the hub. From this, the sensitivity of the blade sensors is 2.67 volts per millimeter. The blade can vibrate in a range of 3 millimeters, therefore, the range of the signal is 8 volts. Using the same criteria of the hub sensors, the sensor is located in a position resulting in having 0 volts when the blade is touching the right magnet.

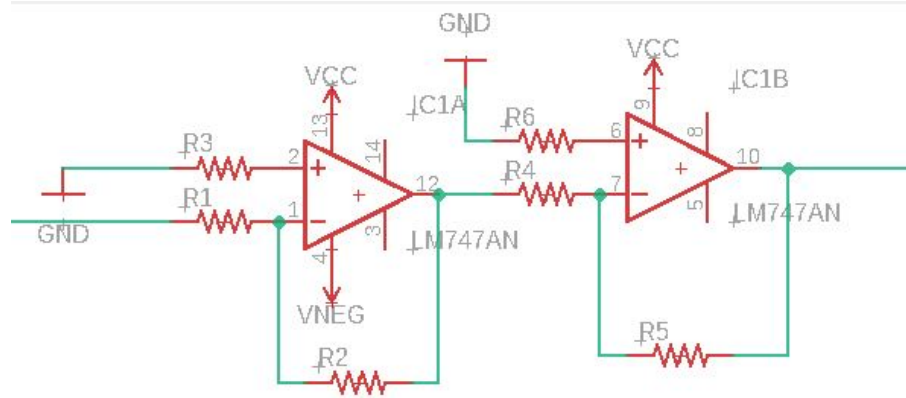
### 3.4 Amplifying electronics

The signal of the sensors is analogical and with a minimum of 0 volts and a maximum of 8, the eddy-current sensors a completely linear within its range. The next step in the design of the sensing system is to adequate this signal into the control unit.

The control unit is described in Chapter 5. In order of designing the amplifying electronics, the input voltage capacity of the controller is needed. The controller has a maximum capacity of 5 volts and can not manage negative voltages. For security reasons, the maximum output of the circuit is designed to be less than 4.5 volts and the minimum 0.15 volts. Therefore, the designed circuit will reduce the signal by half before the control unit.

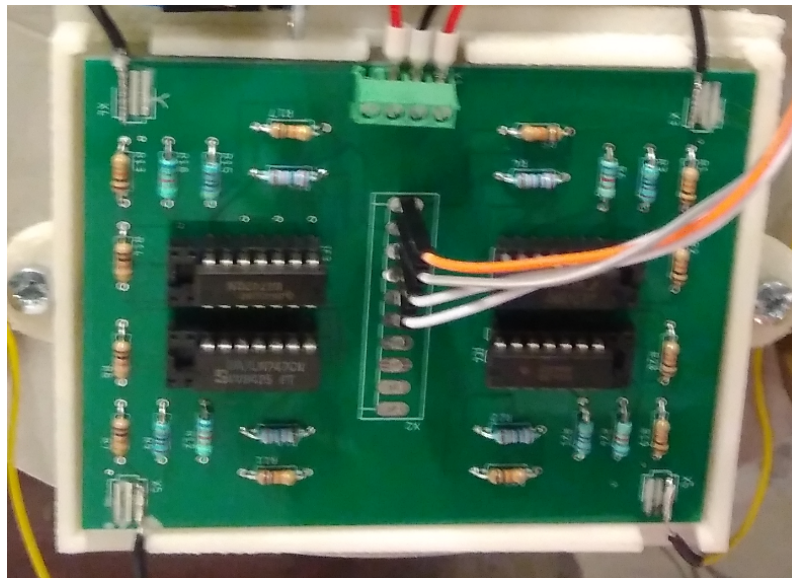
The circuit design is based on operational amplifiers. the operational amplifier is a really common component in the electronics industry. Its simplicity and reliability are its strong attributes. Using the operational amplifiers equations and properties [16] a non-inverting amplifier is designed with 0.5 volt/volt gain. The circuit is shown in figure 3.12 and the numerical values of the resistors and voltages are in Appendix A. The operational amplifier used is LM 741 by Texas instruments (data-sheet in Appendix A).





*Figure 3.12.* Scheme of non-inverting amplifier [2]

This circuit is implemented in a printed circuit base (PCB). The design of this PCB is done in a standard PCB design software and sent to an external company. The dimensions of the PCB were decided with the dimensions of the microcontroller to assure that all fit inside the rotating disk. The 4 circuits needed for the 4 blades are installed in the same PCB. One identical extra PCB is installed in the external circuitry for the signals from the hub.



*Figure 3.13.* Non-inverting amplifiers in the PCB

### 3.4.1 Powering of electronics

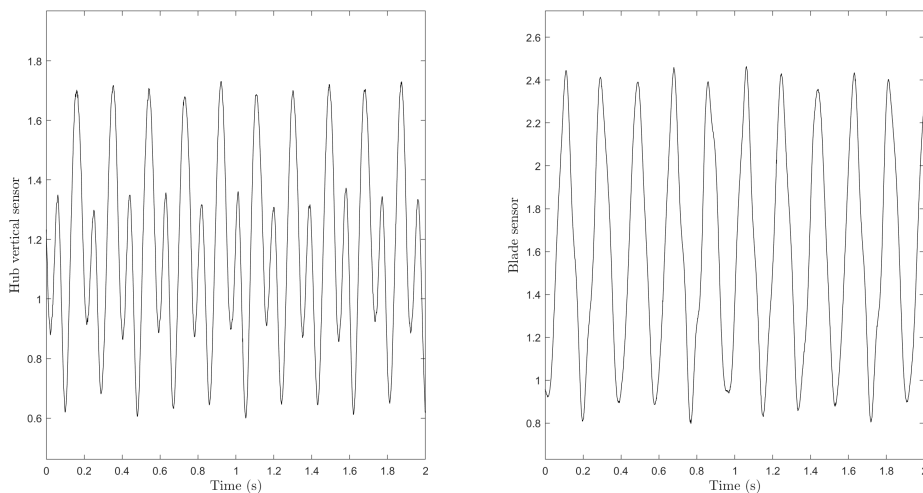
All active components described above need powering voltage. The means of sending this power signal to the rotating disk is explained in Chapter 6. With the purpose of reducing the power signals, the sensor and the operational amplifier are powered with the same 20 volts signal, the operational amplifier needs also a -20 volts signal.

### 3.5 Conclusions

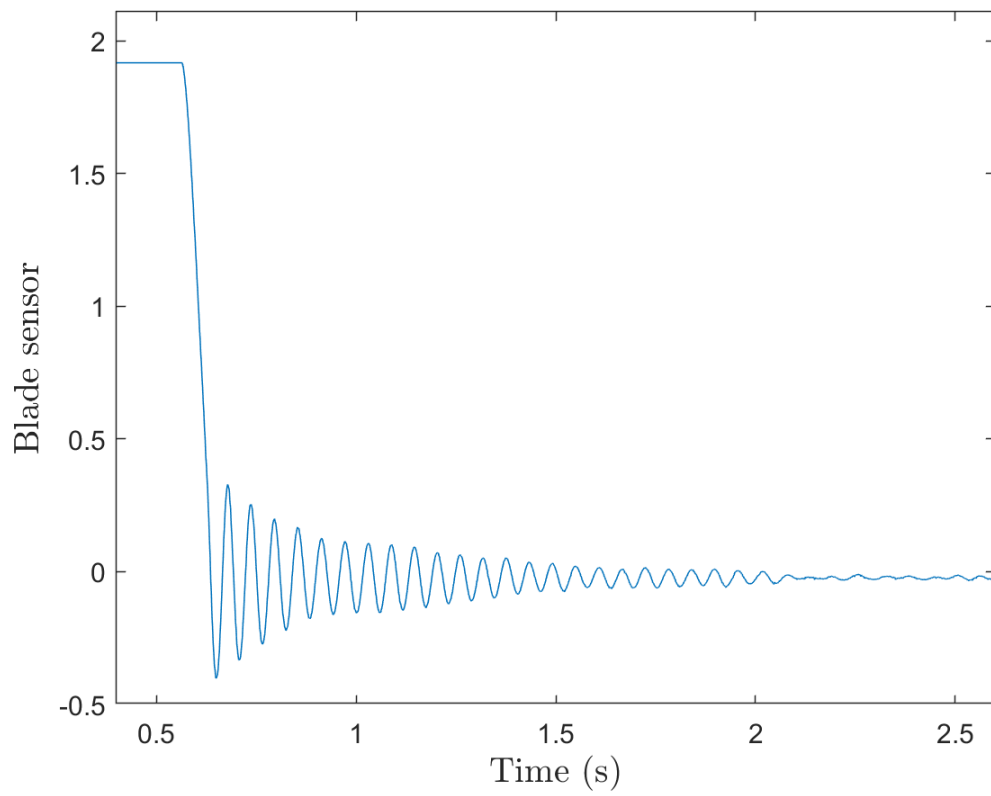
As a summary of all the sections above, the new sensing system has 7 signals and all of them come from analogical eddy-current sensors. The new system detects vibrations in the hub and the blades as well as the angular velocity of the shaft. Regarding the blade vibrations, the signal is focused in the first bending mode but the second and third are still observable.

These signals are in a range between 0.15 and 4.5 volts. Regarding the sensitivity of the measurements, the final sensitivity taking into account both sensor and amplifying circuit is 1.33 volts per millimeter. The noise of the signal is not detectable with human sight in the oscilloscope, however, the final noise input to the controller is described with detail in Chapter 5. The signal is completely linear and robust to any kind of dust or other disturbance in the system.

Here are shown the final signals from the hub sensor in the horizontal direction and the blade sensor. The first test was implemented with a 300 rpm constant angular velocity Figure 3.14. The second test is a zero-input response in the blade to show its natural frequency (18 Hz) Figure 3.15



**Figure 3.14.** Signals from the sensors at 300 rpm



*Figure 3.15.* Signals from the blade sensor with a zero-input response

# 4 ACTUATING SYSTEM

---

The scope of this chapter is to explain in detail the requirements of rotor-blade systems in general and this test-rig in particular regarding the actuating system.

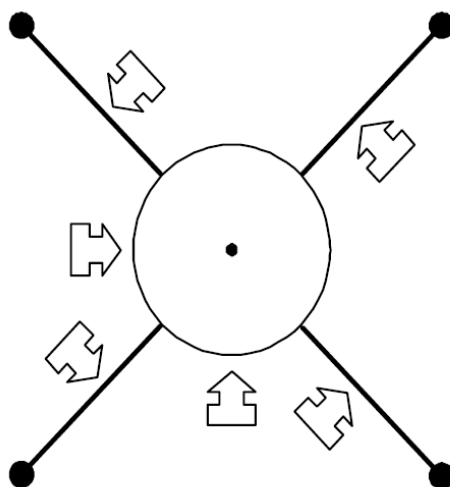
Performing active control in any system is to deliberately input a external signal or force in order to force the output of the controlled system. The actuator is the component that converts the control output into the signal or force induced to the system.

## 4.1 General considerations

In the case of Rotor-blade system, two main paths appear. The first one being to actuate in all the mechanic components with degree of freedom ( rotor and blades), and the second one being actuating only in the rotor or shaft. Choosing between one of the two not only depends on the designer will, but in several other issues. Several papers discuss this dichotomy [17] [13] [18].

In a parallel working thesis [1] it is demonstrated that only actuating in the shaft is sufficient to perform active vibration control both in rotor and blades. However, it is also stated that the performance decreases if there is not control input in the blades.

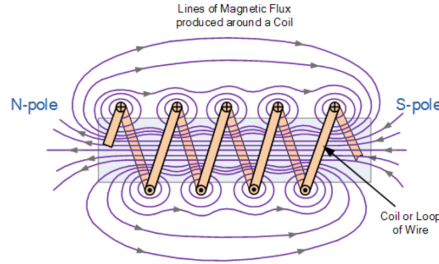
The test-rig where the active vibration control is implemented is prepared to actuate both in shaft and blades [12] [3]. The scheme of how the actuation forces are input to the rotor-blade system is shown in Figure 5.2



*Figure 4.1.* Schematic of the actuating setup [3]

The devices that generate the forces shown in Figure 5.2 are electromagnets. In this test-rig, one pair of electromagnets is installed for every degree of freedom of the rotor-blade system. There is twelve electromagnets , four in the hub and eight in the blades.

Electromagnets are common actuators in industry [9] [19]. The physics behind these devices is the interaction between currents and ferromagnetic materials. The standard electromagnet is formed by one core of ferromagnetic metal and a coil surrounding it (Figure 4.2).



**Figure 4.2.** Schematic of the electromagnet physics [9]

The electromagnetic force created can be described with Ampere’s law. The electrical current passing through the coil generates a magnetic flux in the core. This magnetic flux intends to close the magnetic circuit, in this case between the electromagnet surface and the hub.

The static force generated by the electromagnets can be stated by means of its physical properties, the air gap between surfaces and the current passing through the coil. The force is proportional to the number of turns of the magnet coil ( $n_m$ ), the magnetic permeability of the air  $\mu_a$ , the surface of the magnet  $A_a$  and the current passing through the coil  $i_m$ . On the other hand the force of the magnet is reduce with the quadratic of the distance between surfaces  $z$ .

$$F_m = \frac{n_m \mu_a A_a}{4} \left( \frac{i_m}{z_0 + z} \right)^2 \quad (4.1)$$

## 4.2 Actuators

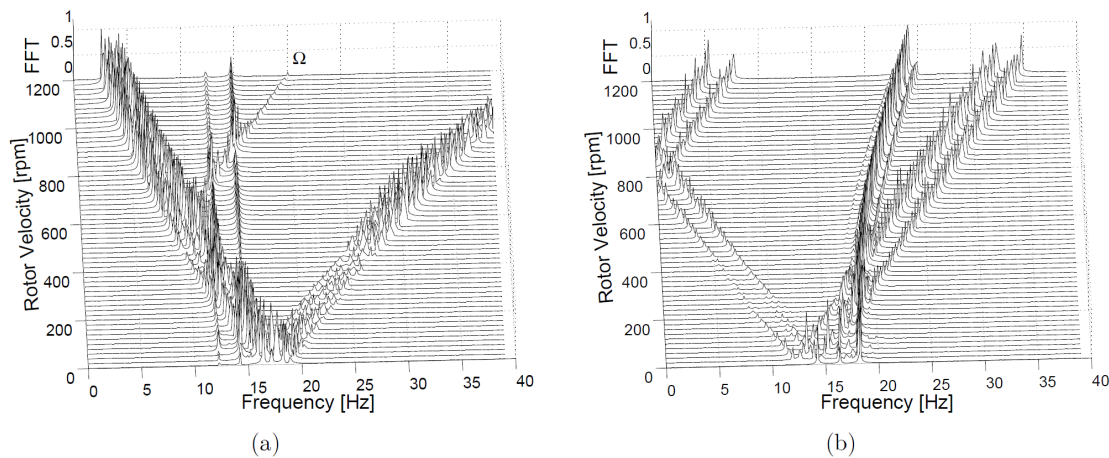
The actuators implemented to induce electromagnetic forces in the hub are RS PRO Magnetic Lock model of 65 millimeters in diameter Figure 4.3. The devices installed in the test-rig are an outdated model. The static forces that can provide the new models are much higher than the experimental forces documented by [3]. Nevertheless, the magnets are to supposed to be able to perform static forces up 100 N. The forces needed according to several control strategies implemented by [5] and [4] are around that values. The magnets are located in the external frame of the hub. One pair of electromagnets for each hub displacement direction (X and Y). The air gap between the magnets and the hub is 2 millimeters in each direction.



**Figure 4.3.** Electromagnet RS PRO 65 mm

The electromagnets located in the blades are of the same nature of the hub actuators but in 32 millimeters in diameter model. The dynamics of these, are similar to the hub magnets but with less peak voltages due to the smaller magnetic flux generated. The static force was also measured in [3] to be 25 N. They are located at 40 mm from the clamping of the blades, the air gap between blade and magnet is of the order of 2 millimeters, similar as the hub magnet gap.

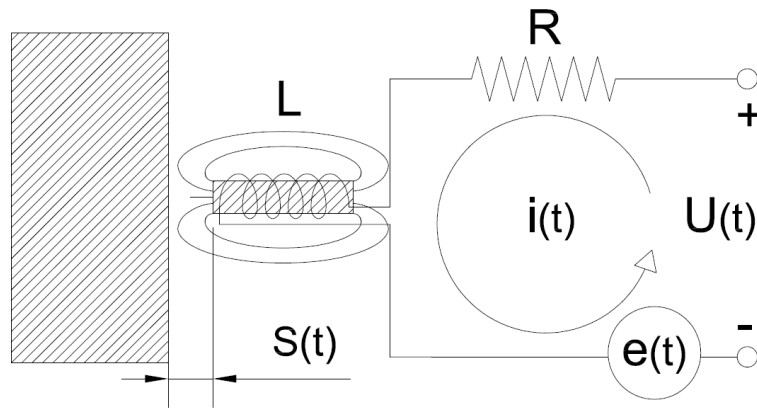
All the discussion above is only for static forces, however, active vibration control needs to actuate in a certain frequency bandwidth. The frequencies at which the actuating system has to have its best performance are related to the excitation frequencies of the system. In [3] is detailed the causes of the vibrations and their frequency depending on the angular velocity of the system. The next figure shows the range of the frequencies that appear in the system depending on the angular velocity of the rotor Figure 4.4.



**Figure 4.4.** Theoretical waterfall diagram showing normalized frequency responses of the rotor (a) and the blade movement (b) as function of the rotor angular velocity [3]

This analysis leads to the conclusion that the maximum frequencies that the controller has to minimize are around 40 Hz. Moreover, this is only when the angular velocity is around 1000 rpm, the velocity at which this test-rig rotate is usually around 600 rpm. In conclusion, the magnets need to be able to perform at least at 30 Hz.

The electromagnets dynamics are not a simple issue. The equations behind them present a substantial quantity of non-linear behaviours [20] [19]. From engineering perspective, the electromagnet can be modelled as a electric circuit in order the apply simpler equations. This component can be explained with a resistor in series with an inductance that has an electromotive force induced into the circuit ( $e(t)$ ) Figure 4.5. The equation defining this electric model is 4.2

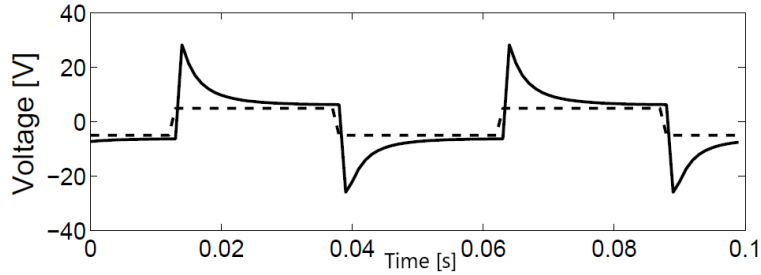


*Figure 4.5.* Electric circuit of electromagnet [1]

$$U(t) + Ri_R(t) + L \frac{di_L}{dt} + e(t) = 0 \quad (4.2)$$

The electromotive force induced to the circuit is non-linear and depends on the change in the magnetic flux, this is usually linearized in order to simplify the system [1] [21]. Despite the non-linearities being modelled for the small vibrations, this electromotive force has to be take into account in some situations that are going to be discussed in the next paragraphs.

In the initial state of the test-rig, no amplifiers were found, however, in [3] it is stated that the signal of the control unit was amplified with  $\pm 28$  volts. Without adding any other device the results of the actuating output were the following (Figure 4.6).



*Figure 4.6.* Desired voltage (-) vs output voltage(-) [3]

This peak voltages are due to the non-linear electromotive force and the inductance of the coil. When there is a sudden change in the input voltage of the circuit and the current tries to also change rapidly, the magnetic flux of the electromagnet counteracts with a voltage peak. This voltage peak rises to maintain the magnetic flux constant. This voltage peak is theoretically infinite, however, when implemented in practice the voltage rises to the maximum is possible. In this specific case, the voltage reaches the maximum that the amplifier can afford (28 volts).

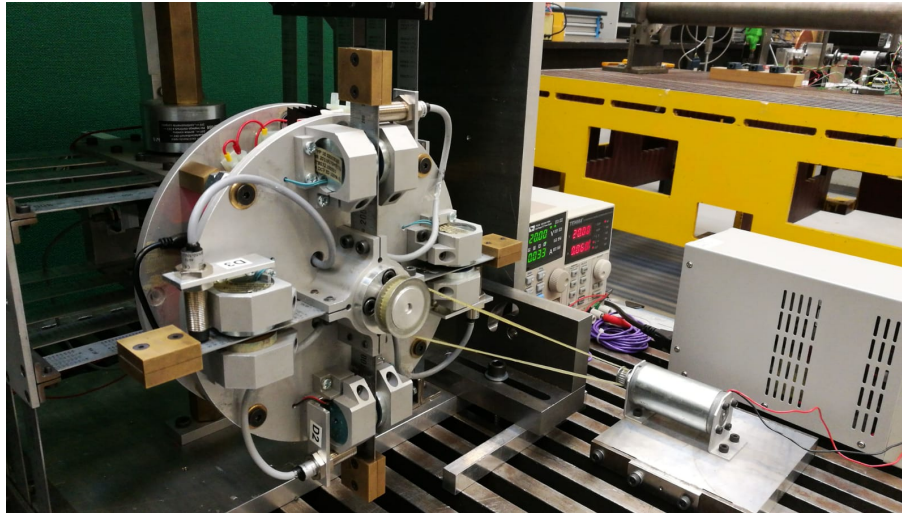
This phenomenon is the main cause of failure when trying to use electromagnets at high frequencies. The control signal is usually the voltage, not the current, and therefore these effects deteriorate both actuators performance and durability. In other projects such as [5] or [4], this made impossible to use the electromagnets as actual actuators of the control loop in the rotor blade system.

In order to attenuate this voltage peaks a different system of power electronics will be applied. The voltage range is between 0 and 24 volts and the electrical characteristics are in the Appendix A.



### 4.2.1 DC motor

In order to rotate the shaft, a DC motor is added to the system. This DC motor is not an actuator of the active vibration control. However it is important to mention that the angular velocity is proportional to the voltage in the terminals of the DC motor. The setup is shown in Figure 4.7



*Figure 4.7.* DC motor implementation

## 4.3 Power electronics

The power electronics of the actuating system is responsible of converting the output of the controller in a proper actuating system. In the case of the test-rig, the power electronics need to convert the 5 volts signal of the control unit Chapter 5 into the actuating range of the electromagnets.

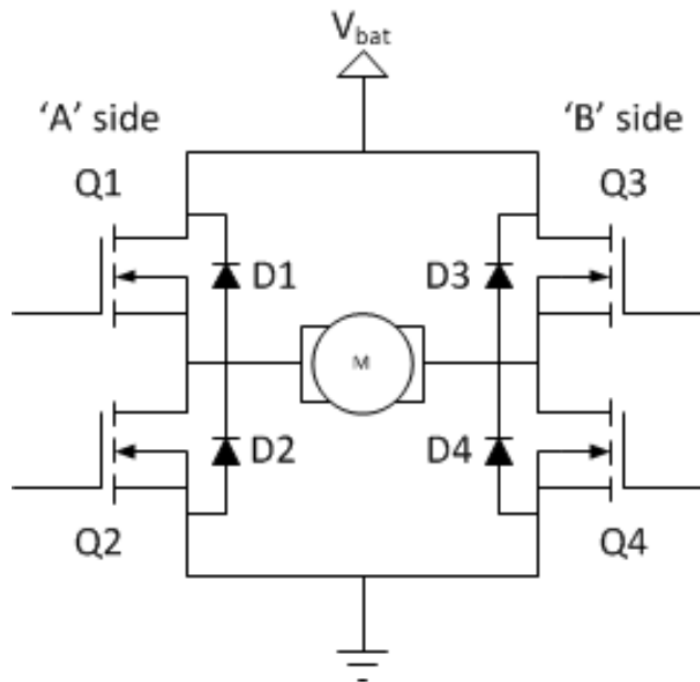
The designing of a new continuous analogical amplifying system was rejected due to several reasons. The first one is to try to avoid the non-linearities of the amplifiers. The second one is the price of this kind of devices, buying amplifiers for the 12 magnets would be really expensive. The last one is the compactness of these devices, this thesis is based on the designing of an embedded system and continuous amplifiers would consume a majority of the volume of the whole active vibration control system.

The power electronics chosen for powering the magnets are based in a non-linear signal generation method called pulse-width modulation (PWM). This method converts a low voltage digital signal into a power analog signal [22].

The regulation of the amplitude of the signal is achieved by changing the duty cycle of another, usually a square signal. The amplitude of the output PWM signal is defined not by the amplitude of the square signal, which is constant, but with time window this signal is activated. The classic circuit that generates this signal consists of a device that compares two inputs and one output.

The working principle is to regulate the duty cycle with the two inputs in order to generate

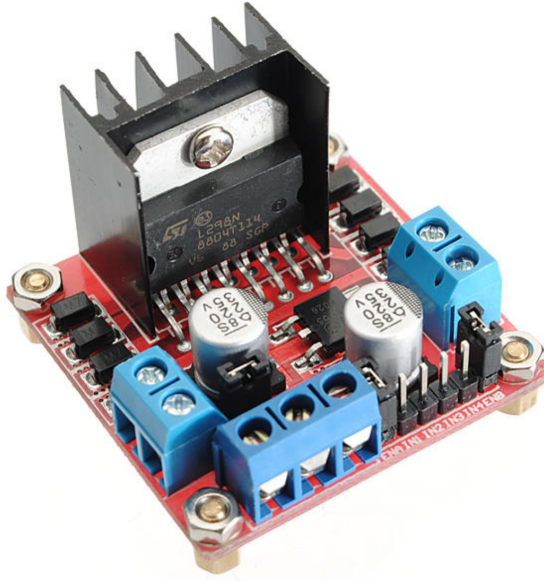
the PWM signal as the output. In this case the two inputs will be compared by software in the control unit and further detail about this is explained in Chapter 5. On the other hand the power signal will be handled with a H-bridge circuit. This circuit is composed by several switches that handle the time window when the constant power is let into the actuator or load (L), this case the electromagnet Figure 4.8.



*Figure 4.8.* H-bridge schematics

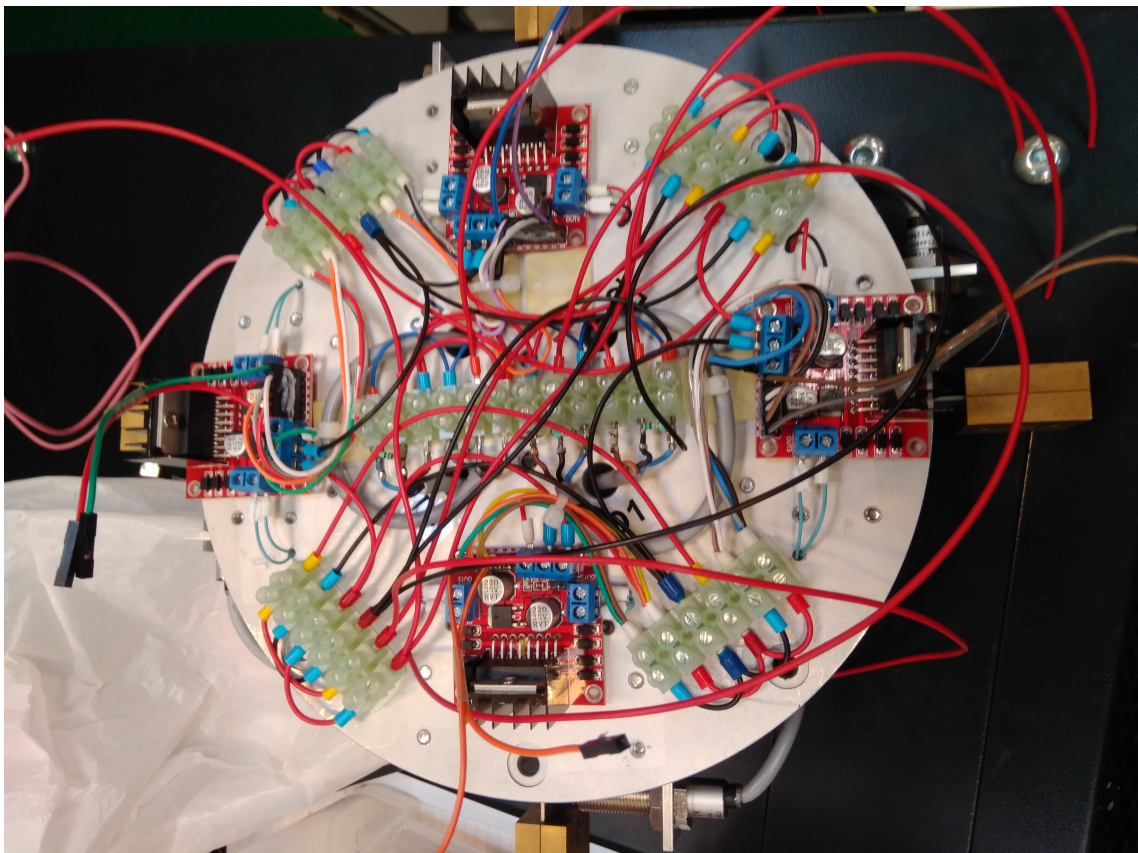
There H-bridge can be halved if it is not necessary to choose the polarity of the signal. The electromagnet output sees no difference whether the input voltage is positive or negative, the force remains the same magnitude and direction. The magnets power consumption is in the range of commercial H-bridges, the chosen H-bridge is the L298n from ST electronics. The datasheet is in Appendix A, each driver can handle two magnets and the maximum current per magnet is 0,5 Amperes.

The input to the driver is the digital signal of the controller and the output signal is the PWM that goes into the magnet. The device is formed by several components. The main components are the switches, in this case MOSFET transistors. The MOSFET transistor has 3 ports, when the control port is excited the other two ports open and let the energy flow through the circuit. In order to absorb peak voltages protection diodes are located in every switch. This diodes prevent short circuit currents and will absorb the peak voltages explained in Figure 4.6. Finally the are some fins to dissipate the energy driven by the switch. The wiring of the system and the power transmissions will be discussed in Chapter 6.



*Figure 4.9.* H-bridge driver by ST electronics

The circuitry includes 6 drivers, these components are installed both in the external frame and inside the rotor disk. The wiring of the electronics, drivers and power signals is detailed in Chapter 6. Figure 4.10 shows how the final circuit is mounted inside the rotor.

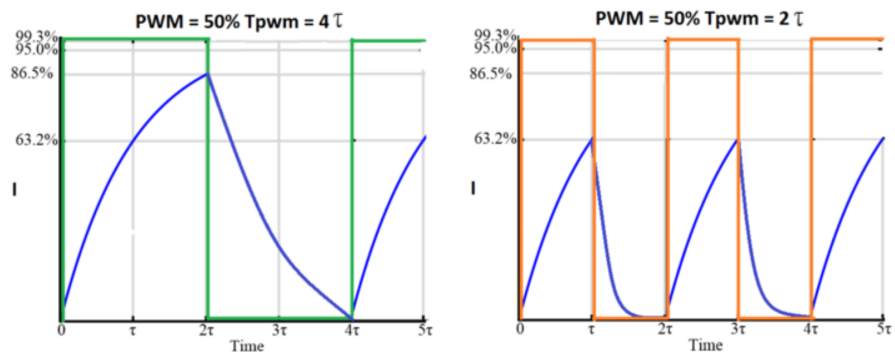


*Figure 4.10.* Power electronics inside the rigid disk

### 4.3.1 Dynamics of the power electronics

The PWM signals have several advantages respect the classic analogical amplifiers. The power losses are mostly produced in the switches and the MOSFET transistors are really efficient. In the case of inductive loads like the electromagnets, the dynamics of the square signal are neglected because the inductance smooths the voltage drops.

The only parameter that can affect the dynamics of the system is frequency of the whole duty cycle. If this and the desired output signal frequencies are too close, the dynamics of the PWM modulation are non negligible. The objective is to maintain the current in the most linear regime. The higher the frequency of the PWM the more linear the whole system will be Figure 4.11. The final frequency of the PWM control signal depends on the control unit, therefore, further details can be found in Chapter 5.

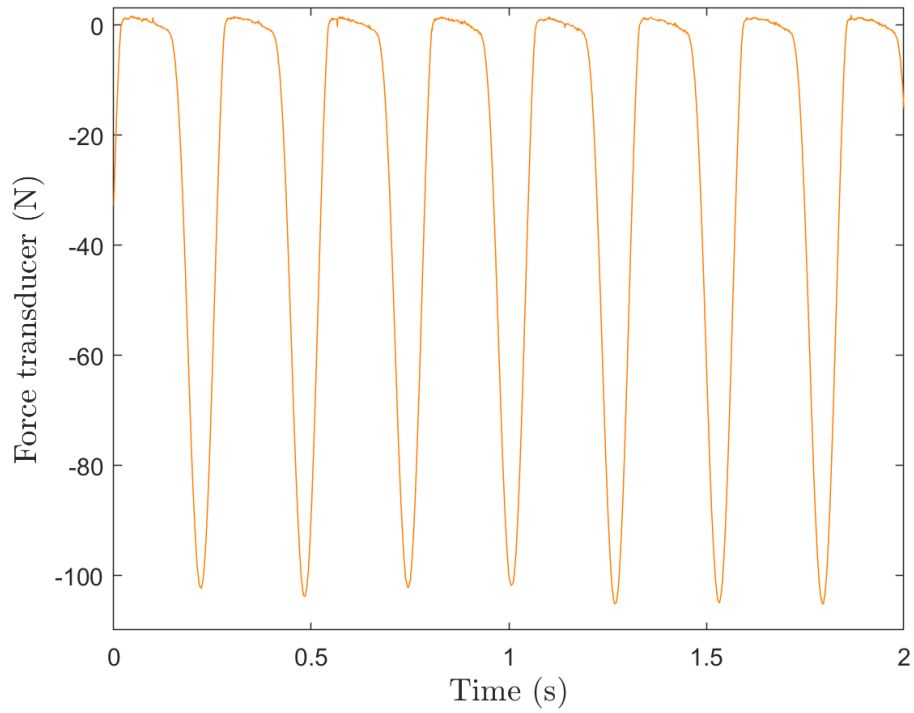


*Figure 4.11.* Effects of PWM frequency by [10]

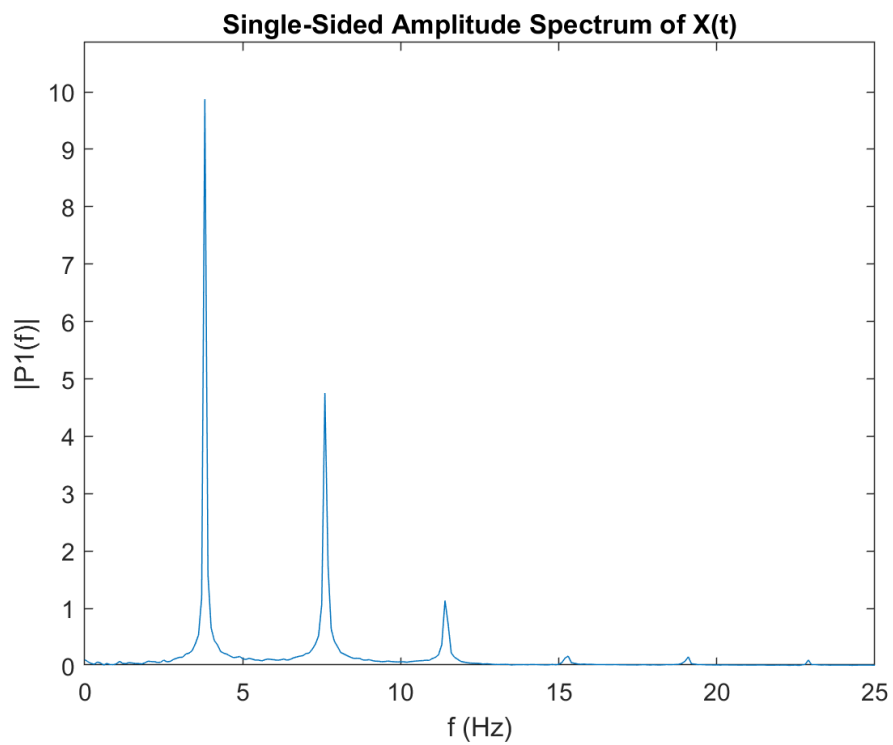
## 4.4 Conclusions

The final output of the electromagnets should be tested to check if it is possible to actuate in the active vibrations control loop. In order to measure the actuation signal, a force transducer has been used. The force transducer is from DTU mechanical department and it has a sensitivity of 0.22 Volts per Newton. This sensor uses a filter because it can only measure dynamic forces, this made not possible to measure the final static forces of the electromagnets.

The tests are performed with sinusoidal input signals of different frequencies and always in a range of 0 to 20 volts. The results are shown in the next Figures. The first ones show the force as an output of exciting the magnets with a sinusoidal of 5 Hz with the PWM signal.



*Figure 4.12.* Force of the magnet at 5 hertz

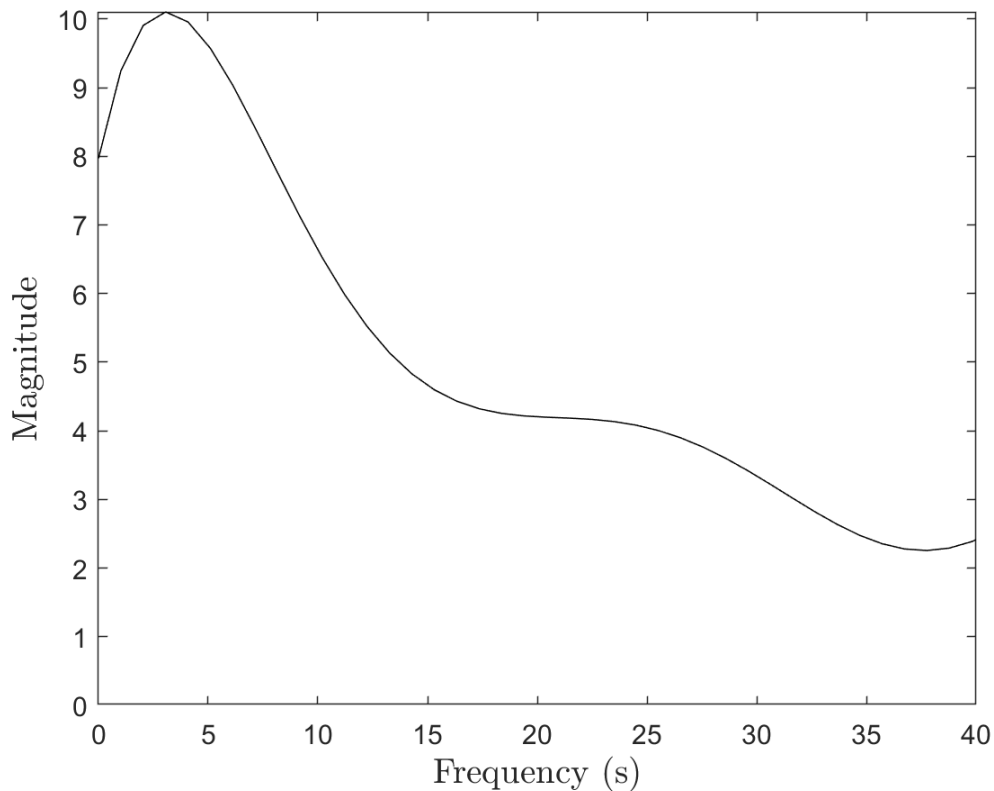


*Figure 4.13.* Frequency spectrum with 5 Hz sinusoidal input

In the first Figure 4.12 it is shown the non-linear effects of the whole actuating dynamics. The peaks of attractive force reach 100 N which are similar to the static forces related in [3]. On the other hand, the null peaks are wider due to the remnant magnetization. This remnant due to the hysteresis loop delays the dynamics of the magnet.

In the second picture 4.13 it is clear that the square wave produces harmonics in the frequencies multiple of the frequency of actuation. The circuit as a whole reduces the frequency of the signal, in this case from 5 to 4 Hz.

This test is repeated for each frequency until the magnet force is not almost null. The frequency analysis is captured in an experimental bode Figure 4.14.



*Figure 4.14.* Experimental bode of the electromagnets

The experiments derive into the conclusion that these magnets are not appropriate to active vibration control. The inductance of both, hub and blade, electromagnets is too high. The effects of this inductance do not allow to fulfill the requirements in the frequency domain. Because of this, it is encourage to install different electromagnets with better frequency response.

# 5 CONTROL UNIT

---

The control unit is the core of an active control system. This subsystem is in charge of using the sensing and actuating system in order to accomplish the control objectives. A control unit handles all processor control signals. It directs all input and output flow, uses and analyses code and directs other units and peripherals by providing control and timing signals. A metaphor usually used to explain the importance of this subsystem is to compare it to the human body brain. The control unit component directs orders to just about every aspect of the system and ensures correct instruction execution.

In the case of the rotor-blade system, it will decide which force should the electromagnets induce to the system and analyze the sensing system signals. In the previous chapters the measurements that the sensing system provide and the signal needed by the actuators have been defined. However, in this point of the design of the active vibration control system the measurements are just 7 voltages and the actuators are some electromagnets. It is only when the control unit is implemented that the measurements are understood as displacements or vibrations, and the electromagnets converted into real forces into the system.

The hardware where the control unit is implemented has not a unique possible configuration. There are several options to implement the control unit of a feedback control loop in a physical system. In the introduction the two most common solutions were presented. The first one is to base the controller in a commercial computer and the second one is to build the hardware based on microcontrollers technology.

In [12] several solutions to previous active vibrations control systems are described, in all of them the implementation of the controller is based on general purpose computer. This is due to the fact that computers are able to perform really heavy computation tasks and the development is more accessible to engineers that do not have profound knowledge in electrical hardware solutions. The test-rig was indeed equipped with this kind of controlling configuration.

## 5.1 Initial control system

The control unit installed at the beginning of this thesis was composed of a general purpose computer with a Dspace interface. Dspace is a digital signal processor (DSP) in conjunction with a software implemented in the host computer. The hardware then is divided into three devices. The computer, the DSP and the I/O board. The board dSPACE DS1103 PPC is the responsible to transform the analog signals into digital ones. This board has up to 16 analog input ports and 8 output ports.





*Figure 5.1.* dspace DS1103 PPC

The DSP is in charge of performing the calculations of the control algorithm. The control algorithm is programmed by the Host computer by using Matlab code. The DSP can be configured to a sampling time and diverse control architectures. The Matlab code is written in the graphical interface Simulink and the sent to the DSP.

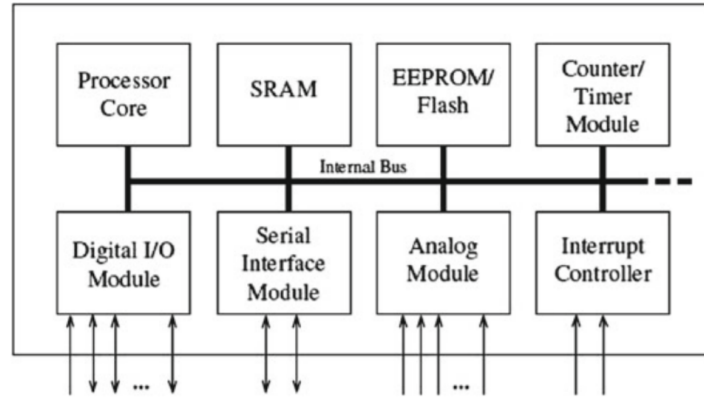
The installed control unit is from 2013 and is outdated in some specifications. The sampling is up to 1000 Hz that is sufficient for the test-rig. On the other hand, the board can not handle really high frequency digital signals such as the PWM explained in the previous chapter. Moreover, the interface only handles Matlab code which limits the features that the system can have apart from the controller.

The Dspace also has presented some problems in the close-loop performance, in the model installed in the test-rig there is no way to ensure that the input and the output of the controller are synchronized when the computing load is high. For low frequencies this may not come to a problem but leaves no room for improvement. The fact that it is an all-in-one hardware makes impossible to improve an specific part of the hardware. These reasons added to the fact that new technologies in computational hardware led this thesis to choose to build a new control unit.

## 5.2 Control unit design

In the past years, the computational capacity per hardware size has improved greatly. The Moore's law [23] is still valid and every year with less size there is better performance.+, this led to the use of embedded systems in every kind of automated solution. An embedded system is a compact electric system whose core is a microcontroller unit. A microcontroller (MCU) is a programmable logic device that combines electronics components integrated into a single chip: CPU, memory, peripheral devices and I/O.





*Figure 5.2.* Schematic of the microcontroller internal setup [11]

The description of every component of the microcontroller structure is out of the boundaries of this thesis. For further detail in this subject, a complete explanation of every component is available in [11]. The design of the microcontroller starts with understanding the requirements of the system to be controlled.

The microcontroller that is intended to be designed is only going to be implemented in one system, therefore the price is not going to be the main parameter to be taken into account. Nevertheless, all MCU cost between 10 and 500 danish crowns, this fact is the main reason embedded systems are being so popular. The requirements of the system will be the main decision criteria. The test-rig system needs the control unit to have some features that will be described in the following paragraphs.

### 5.2.1 Requirements

The first point to start to design your embedded system is to build a use case diagram. The use case diagram defines the interaction of the user and the system and the inner relationship within the subsystems. In the case of a feedback control loop is already standardized. The user only interacts with the control with the user interface, this interface can have several features, however due to time constraints, in this report it will only be possible to start and stop the control loop.

The point of the use case diagram that is important for the design is the interaction of the control unit with the rest of the system. This has been discussed through all the report, the control unit needs to analyze the signals from the sensing system and generate control inputs to the actuating system.

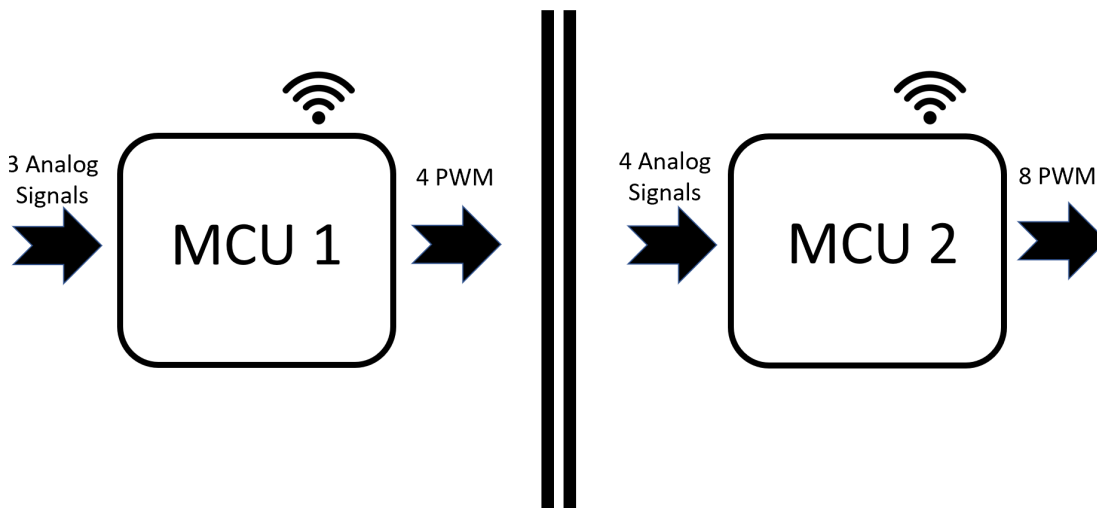
As it is stated in Chapter 3, the measurements of the the vibrations and the rotational speed are a continuous analog signal with a range that goes from 0 to 5 volts. The seven signals have to be analyzed by the computational core, this means it is needed to pass them from analog to digital nature. On the other hand in the Chapter 4 is explained that, although the actuators need analog power signals, the actuating system needs 6 digital PWM signals of 5 volts.

Another key point of the system is the fact that the blade actuators and sensors need to be able to rotate with the rigid disk. The performance of the rotating subsystem can not depend on the rotational speed of the rotor. Although all the detail about how the power is transmitted in the system is explained in Chapter 6, it is necessary to explain that the best solution to this problem was to install 2 microcontrollers instead of only one. This way the signals from the blades are static in reference to one microcontroller, simplifying the connections of both power and sensing.

### 5.2.2 Design criteria

The advantages of using two microcontrollers apart from the stated above is that the performance needed for each MCU is much less and the signals that these must handled are the half of the original system. However, this new design comes with a new issue, communication between the microcontrollers. The solution to this problem is described in Chapter 6, the main microcontroller will be located in the external frame and the MCU in charge of handling the rotor subsystem will be located inside the rotor disk. This configuration leads to decide the specifications of each microcontroller with new criteria.

After the description of the signals needed, the peripherals that the embedded system needs can be stated. For the external frame the MCU needs to be capable of generating 4 PWM for the 4 electromagnets and analyze 3 analog signals, two for the displacements and one for the rotational speed. On the other hand the MCU located in the rotor must handle 8 PWM signals for 8 electromagnets and analyze the 4 analog signals of the blade measurements. Moreover, both MCUs need 1 peripheral in charge of the telemetry communications 5.3.



*Figure 5.3.* Schematic of MCU requirements

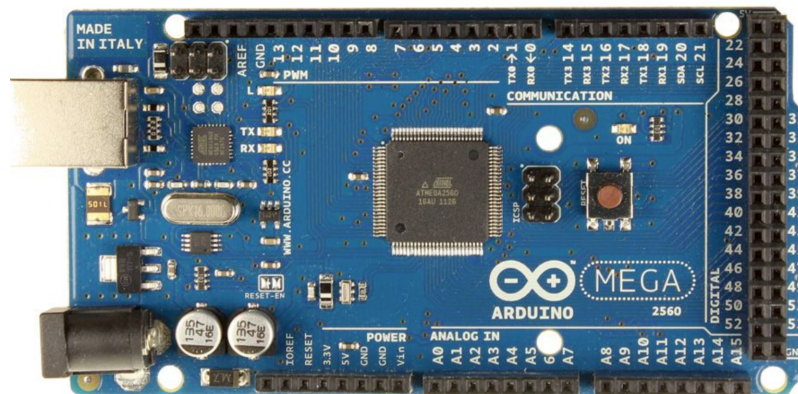
Once the needed peripherals of the microcontroller are defined, the next step is to decide the inner architecture. The architecture of a microcontroller is the method it has to compute orders, two options exist. RISC and CISC [24]. The main difference between both is that CISC microcontrollers execute fewer but more complex instructions and vice versa. The RISC architecture resulted in more energy efficient and compactness, this is

why the vast majority of MCUs are based on this technology. Now the next step is to choose, the instructions bit size, from 8 to 64 bit architectures. The memory of the CPU is fixed so the more bits the instruction needs the less instruction. However, The new MCU come with sufficient CPU memory to handle 32 bit without being overloaded [25]. After all, the the architecture chosen is RISC 32 bit. The most common microprocessors that works with the chosen architecture are the ARM processors. ARM is only the architecture, the processor is built by other companies such as Texas Instruments, Microchip or ST electronics.

The next step once is chosen the architecture and the peripherals needed is to chose either to build a microcontroller from scratch, printing the circuit in a PCB with the custom characteristics or to purchase an already built microcontroller board. The development of a complete new board gives the designer more liberty in means of performance and software and hardware solutions. However, this option requires a great amount of time and this thesis is time constrained. Therefore, the option taken is to search for a board compatible with the project requirements.

After researching the market, three boards were suitable for our requirements. The three first elected MCU were Arduino MEGA 2560 Rev3, Raspberry Pi 3 Model B and the STM32F411.

First the Arduino based board MEGA 2560 Rev3 was evaluated for this purpose, Arduino is well known because the high quantity of home made applications that are based on this kind of microcontrollers the two main key values that this board presents are the overwhelming software documentation that can be found on the web and the amount of peripherals this board an handle.



*Figure 5.4.* Arduino MEGA 2560 Rev3

The reasons for rejecting this board are several, first the architecture does not suit our requirements, this board runs in 16 bit architecture. Secondly the board has not wireless connections, it would need another module installed. This board may reach the requirements for the software that is intended to be implemented in this thesis, but in the case of adding features to application, the board may collapse.

## 5. Control unit

---

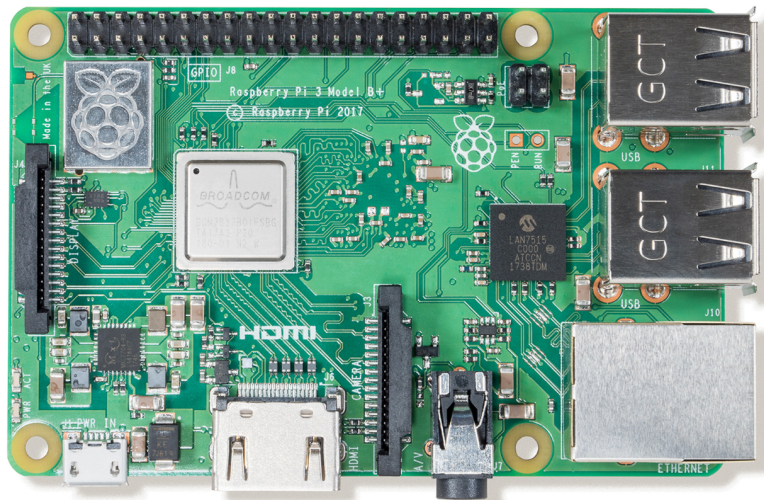
The second board is the STM32F411, this board has really powerful processor and is based on ARM cortex-M4, this processors are used in important industries such as mobile phones or portable video games consoles. The peripherals are covered and with high specifications. The reliability of this board is also a really positive point.



*Figure 5.5.* ST electronics STM32F411RET6

This board was also rejected for the lack of wireless module in the board. The development of this boards needs some previous knowledge because the documentation available on the internet is not as wide as the Arduino one.

The last board and therefore the chosen one is the raspberry pi 3 model B. It is as powerful as the STM board, able to handle the peripherals required by the system and is based on ARM microprocessor A-53 that handles 32 bit instruction. The single based board has 4 cores that assure that will not collapse when the application grows.



*Figure 5.6.* Raspberry pi 3 model b+

Despite all the explained above, the two main points that were decisive to choose raspberry pi were the wireless included module and the possibility to run an operative system (OS) in the board. The first one is clear, it simplifies the connections of the board and reduces in one element the embedded system that has to fit in the test rig. On the other hand, basing your application in a OS has diverse consequences both negative and positive. The datasheet of the raspberry pi is located in the Appendix A.

It has to be remembered that this thesis is not isolated and is running at the same as two other ones that depend on the development of this embedded system. Therefore time is the main issue to take into account to select the software of the system. Even though the application is the same, the development environment can make the difference between the success or failure of the project. Nowadays with the growth of IT industries and embedded applications, a whole technology is raising that focuses only on how these systems are developed [26].

### 5.2.3 Development environment

The development implemented in an OS in the raspberry pi is the most common usage of the board and therefore there is great documentation in the internet. However, before explaining why is easier and quicker to develop in the OS environment, the drawbacks will be detailed.

The operative system is a complete software that has all the features be managed by a user with no software deep knowledge. This features include an scheduler that decides the order of executions of the instructions. This scheduler can not be programmed without corrupting the operative system software so this has to be taken into account.

The fact of having an instructions scheduler that decides when each instruction is going to be processed eliminates the possibility of real time programming. Real time computing describes the systems that operate in a time that can be reliable to external time events. This means that the computational time has to be consistent, in the case of a close loop system, it can be simplified to stating that the time consumed in each loop is constant over time. The scheduler of the OS can not assure this feature because the criteria for processing the instructions is submitted to other criteria rather than consistent time consumption.

This inconsistency of the closed loop latency has to be compared to the time constraints of the dynamics of the system, if this time variance is of the order of magnitude of the final frequency of the system, the performance would drastically decrease. In the case of this test-rig the design assumed that the frequencies that handle the system, order of 50 Hz are nothing compared to this time variance. This variance depends on the application, but in the initial test, the time inconsistency was negligible compared to the physical system.

The operative system installed is Raspbian Stretch, this is the OS that is more developed for raspberry. Raspbian is a lightweight operative system that has a simple interface with the MCU peripherals. Inside the OS the integrated development environment (IDE) is installed, this software is the tool for developing all the code that will be implemented in the microcontrollers. This is the main advantage of using an OS inside the control unit. The code can be developed in a user friendly environment, reducing the coding time in a

great percentage.

The IDE selection depends entirely on the developer, in the case of this project, a text editor (Sublime Text) will be the chosen IDE. The main reason to select a text editor is because it is possible to develop in different coding languages in the same environment. This derives into the next issue, code language for the test-rig control unit software. There exist several coding languages in the software industry, they all have different advantages or disadvantages respect each other.

In the case of the raspberry the most common code language to develop with is Python. Python is an interpreted language, this means that the processor translate each line of code individually into the machine code that the processor understands. This feature means that the interpreted languages do not need a compiler that previously converts the python code into machine code. The lack of compiler increases the development velocity reducing the time needed to build the whole application. Added to all the previous advantages, Python is a very high abstracted coding language, this derives into more user friendly instructions, reducing even more the development time [27].

On the other hand, the main drawback of Python is code performance, because it needs to translate every instruction individually, the code takes more time to be executed. This fact in a closed loop feedback control could imply the failure of the application, this is why other languages were also taken into consideration. C is the most famous compiled language, because its simplicity and performance.

The final solution is to develop in Python taking advantage of the development velocity, try the application to test its performance and if it is not satisfactory, translate the code into C language.

The final application will use two MCUs, raspberry pi 3 model b, and the development code is python with the possibility to change to C if the software is not efficient enough for the requirements of the system.

### 5.3 Embedded Software

The software that defines the embedded system functionality is now described. The application can be separated into several subsystems: Acquisition system, control signal generation, user interface, synchronization, control algorithm and communications. However the development of each part is not isolated from each other, the software grows as a whole. Despite this fact, the report is organized as if the software parts were developed isolated and in order with the purpose of simplifying the explanation.

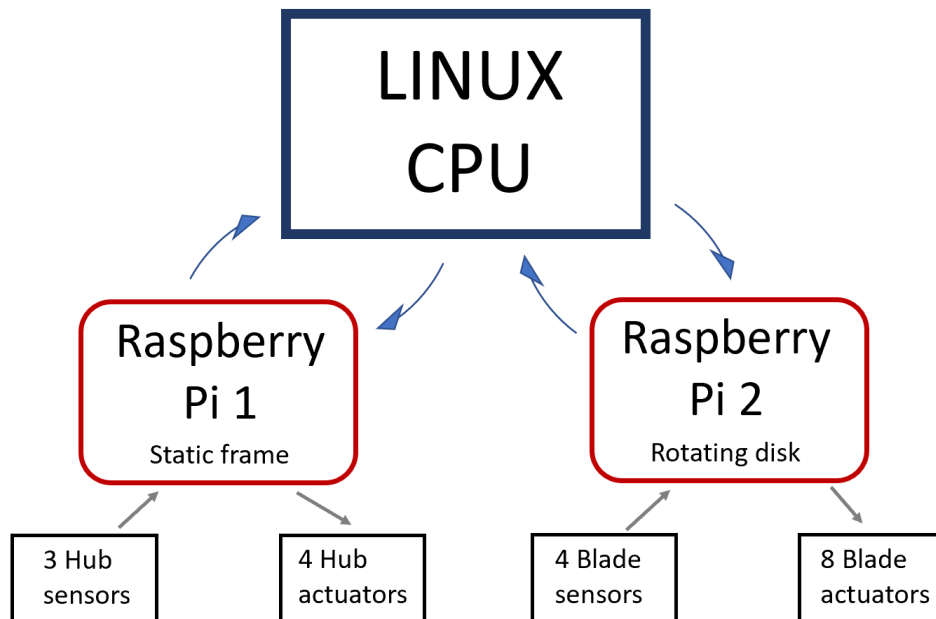
The software was completely developed in Python, tested and then translated to C because the performance did not match the system requirements in terms of time consumed per loop. Therefore the code explained in the report will only be the one developed in C language. This will be detailed in the performance section.



### 5.3.1 Hardware and software configuration

Before the description of the code, the final configuration of the Micro-controllers and its functions will be described. As it was stated before, the development is not a unidirectional procedure and several configurations were tested during this process. The computational time of the simplest control algorithms as PID or Full-state feedback only require arithmetic operations and therefore the raspberry can process them without any performance drop. However in previous works, [4] [5] more complex controls were designed. This was reinforced by the parallel thesis [2], this project needs to implement a full real time identification of the system, this computational load can not be taken by the implemented microcontrollers.

All things considered, the final application is a distributed control with a Linux computer as the core of the computational operations. This configuration can be also explained as if the raspberry pi only were the interface between the sensing and actuating system and the controller is held by the Linux computer Figure 5.8.



*Figure 5.7.* Configuration of the MCUs and the computer

From this point until the end of the chapter, the software will be divided into two programs, the program installed in the Linux CPU and the program implemented into the Raspberry pis. The coding problem will be solved as a server-client problem. The clients being the MCUs submit the information to the server being the CPU. Therefore the main program is the one installed in the computer and the secondary ones are the installed in the microcontrollers. Moreover, the code installed in the MCUs is practically the same but managing a different quantity of signals, in order to simplify the explanation, this programs will be treated as a unique one.

### 5.3.2 Code description

The code described in this section is already the C code, the code in Python was substituted but the functionality remains the same. The annex collects both codes for consulting but in this report the key points are going to be detailed with screen shots of the code. The description starts with the main issues solved in the MCUs and ends with the features of the program installed in the Linux CPU.

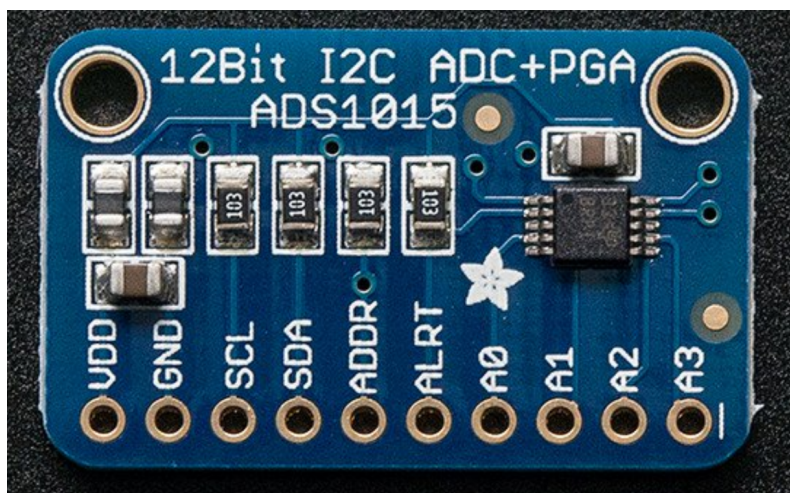
#### Microcontroller software

The microcontrollers have 3 main features: Acquisition, PWM signal generation and wireless connection with the server. The wireless connection is described in Chapter 6.

The acquisition system has to convert the signal described in Chapter 3 to the digital domain. All the computations processed in software applications are in the discrete domain. In order to acquire the signal into the MCU, an analog to digital converter is required. This device will be connected to the output of the sensing system and to the microcontrollers, each MCU needs one AD converter. Raspberry has several options in the market when it comes to AD converters. The AD converter 3 main parameters are resolution, number of channels and acquisition rate.

The resolution is a parameter that describes the precision of each conversion. The AD converter works by comparison between the input analog voltage and a number of voltage levels that depend on the AD converter model. On the other hand the conversion rate is the amount of samples that the device can perform per second.

In the market there is a trade off between resolution, conversion rate, noise and price. In the case of the test rig only 4 channel converters were taken into account. Two models were tested from the same company, ADS 1115 and ADS 1105 from Texas instruments. The first converter has a 16 bit resolution and 860 maximum data rate while the second has 12 bit resolution and 3300 maximum conversion rate. The performance of both devices was tested and the 12 bit converter (ADS 1105) was chosen due to the faster conversions. The datasheet is located in Appendix A.



*Figure 5.8.* AD converter ADS1105 by Texas Instruments



The ADS 1115 has already libraries prepared by the company developing them. However the usage that this application needs does not match them. The software implemented was slow due to how the code handled the selection of the channel. In order to improve the performance a new code was developed in the AD converter.

The AD communication with the Raspberry Pi is via I2c bus [28]. This protocol is reliable and needs only two wires, however, it transmits the data in series making the transmission slow. The default code was design to request a conversion, request confirmation until confirmed and then receive the converted value. This makes that the program waits to at least 3 I2C transmissions slowing the code. The solution implemented was to sleep the program instead of requesting confirmation.

The sleeping time was tested with several values and was decided to have the fastest with 0 mistaken conversion out of one million times. In conclusion the code instruction is now to request a conversion, sleep the program for a specified time and then get the converted value.

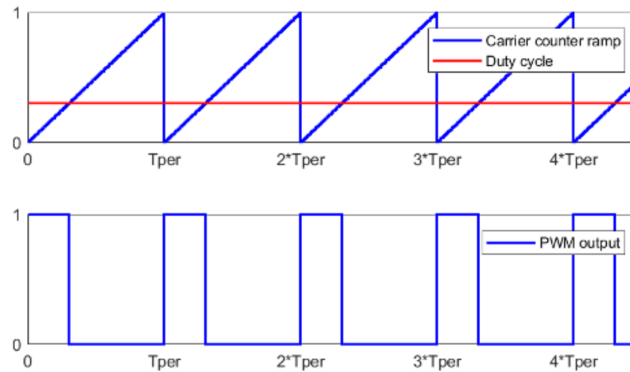
---

```
1   wiringPiI2CWriteReg16(fd,0x01,CONFIG_V[j]);
2   nanosleep((const struct timespec []){{0, t_delay}}, NULL);
3   data= wiringPiI2CReadReg16(fd,0x00) ;
4   data=Byte_swapper(data);
5   DATA_SEND[j]=data>>4;
```

---

The final conversion data rate chosen is 3300 values per second that is the maximum data rate possible. The time consumed within the 4 conversions is 1.5 milliseconds, slightly slower than the ideal 4 conversions in series. The resolution is 12 bits, that derives in a resolution of 0.0026 millimeters per bit.

The other feature of the MCU's software is the PWM signal generation. The PWM power signal was detailed in Chapter 4, in this chapter the description of how the digital signal is generated is presented. The PWM digital signal needs to be a square signal with the highest frequency possible. The software produces the signal with a counter and a threshold. The counter is a real time clock that counts until a maximum value defined in the PWM software, when this counter arrives to the threshold the PWM switches from off to on. The parameters that define the PWM signal are the counter resolution and maximum value and the threshold Figure 5.9.



**Figure 5.9.** Software PWM generation

The raspberry Pi has only two PWM timers and 8 channels are needed in the case of the rotor actuators. The PWM board handles the lack of hardware PWM timers with direct access memory (DMA) peripheral[29]. This DMA feature gives all the I/O pins of the Raspberry to work as PWM signals with less performance. The maximum frequency that is handled by the DMA is 8000 Hz, this frequency was tested and no dynamics appeared because of the PWM in the actuators. The high inductance of the installed electromagnets allows to use this feature instead of hardware real time PWM signals. The counter range is set to 2000 for the maximum 20 volts, this gives a resolution of 10 millivolts in the output signal.

---

```

1  {
2  ////////////////////////////////////////////////// SEND THE SENSOR DATA
3      send(sock , DATA_SEND , sizeof(DATA_SEND) , 0 );
4  ////////////////////////////////////////////////// WAIT FOR THE SERVER CONTROL SIGNAL
5      valread = read(sock , data_rec, 40);
6  ////////////////////////////////////////////////// EXIT THE PROGRAM IF THE SERVER ASKS FOR IT
7      if (data_rec[0]==close_vector[0])
8          {flag=0;
9            printf("%s\n","EXIT");
10         }
11     else {
12         ////////////////////////////////////////////////// IMPLEMENT THE CONTROL SIGNAL INTO THE PWM
13         gpioPWM(magnet1r, data_rec[0]);
14         gpioPWM(magnet1l, data_rec[1]);
15         gpioPWM(magnet2r, data_rec[2]);
16         gpioPWM(magnet2l, data_rec[3]);
17         gpioPWM(magnet3r, data_rec[4]);
18         gpioPWM(magnet3l, data_rec[5]);
19         gpioPWM(magnet4r, data_rec[6]);
20         gpioPWM(magnet4l, data_rec[7]);}
21     ////////////////////////////////////////////////// READ THE AD CONVERTER DATA
22     for (int j = 0; j < 4; j++)
23     {
24         wiringPiI2CWriteReg16(fd,0x01,CONFIG_V[j]);
25         nanosleep((const struct timespec []){{0, t_delay}}, NULL);
26         data= wiringPiI2CReadReg16(fd,0x00) ;
27         data=Byte_swapper(data);
28         DATA_SEND[j]=data>>4;

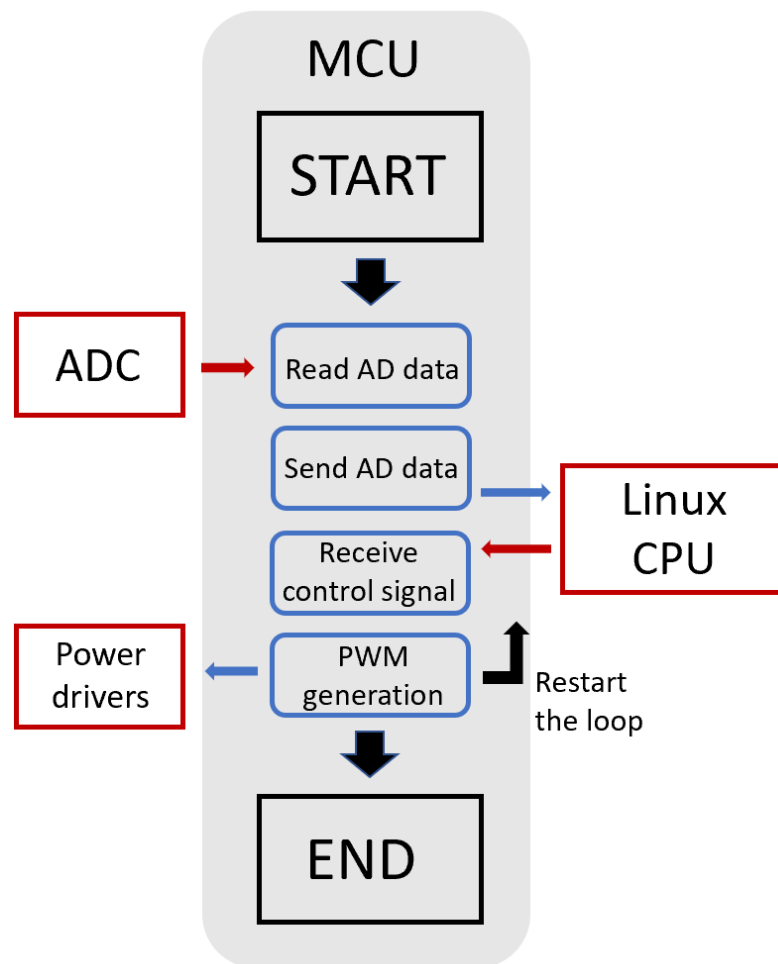
```

---

The MCU software in conclusion has the following structure:

- Establish connection with the server
- Start the loop
- Read the AD converter data
- Send the data to the server
- Receive the control signal
- Implement the control signal into the PWM software
- Stop the loop

Firstly the connection with the server is established, then the control loop starts and, finally, when the server decides, the program is ended. Inside the loop, the MCU reads the value from the AD converter and sends it to the server, then, it receives and implements the control signal from the server into the PWM software.

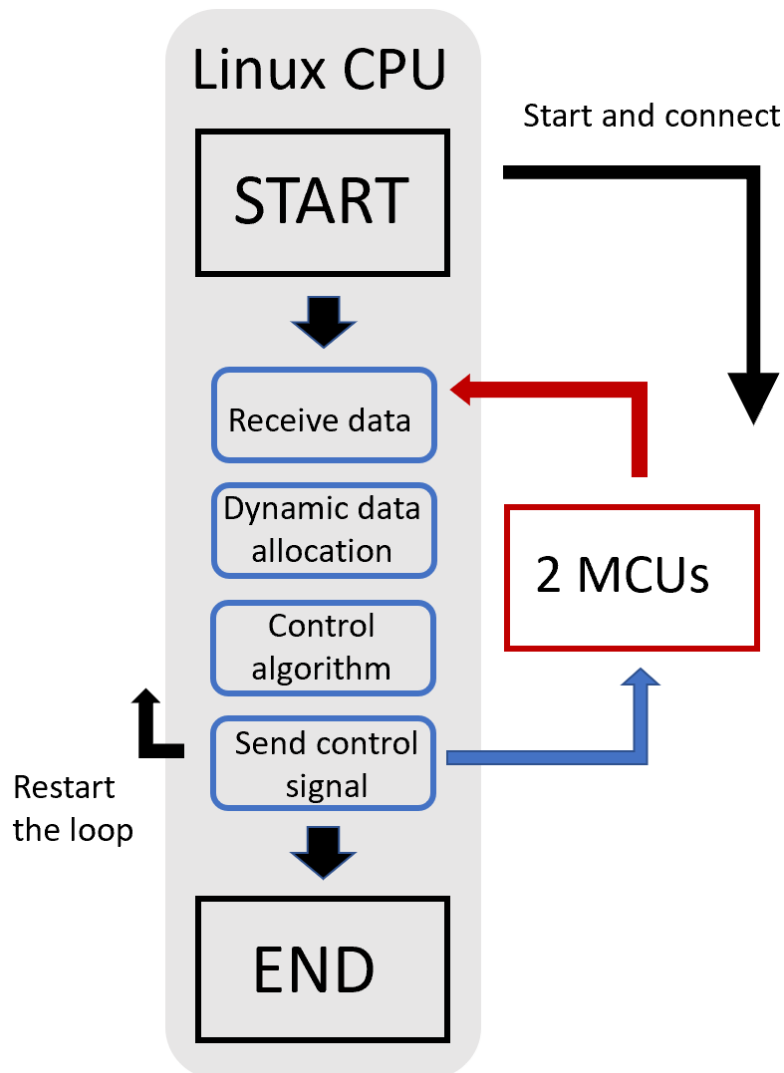


*Figure 5.10.* Control loop scheme in both Raspberry Pi's

### Linux CPU software

The code implemented in Linux has to handle both data of the two raspberry pi's and generate a control signal that can be implemented in the PWM software. All the arithmetical conversions of the different data types will be calculated in this processor. Therefore the program features are the following:

- Start both MCU's programs
- Establish connection with the microcontrollers
- Start the loop
- Receive and process the data from the raspberry pi into measurements
- Save the data in the CPU memory
- Implement the control algorithm
- Convert and send the control signal into the raspberry
- Stop the loop and process all the data



*Figure 5.11.* Control loop scheme in the Linux CPU

The full code is scripted in the Appendix B, however, some key parts of the coding structure are briefly explained in the next paragraphs. The connection with the microcontrollers is detailed in Chapter 6. The first step once enter the feedback loop is to wait for the microcontrollers to send the data. Once these seven 12 bit data are in the CPU, the next step is to translate it to measurement in millimeters. After this the data is saved into a vector in order to be used by the controller. Because C does not handle the memory automatically, it is necessary to create a dynamic memory to save the endless feedback loop data.

Once the data is prepared, the control algorithm starts. This algorithm is out of this thesis, however, the code structure is prepare to handle every control algorithm. All the previous measurements and input voltages are available for the control algorithm. The control signal is generated and converted to a number within 0 and 2000 for the PWM software, after this the 12 voltages are sent to the corresponding MCUs.

---

```

1  i++;
2  ///TIME APPENDING
3  gettimeofday(&t1, NULL);
4  d_array[i] = t_s;
5  ///RECEIVING DATA FROM CLIENTS
6  valread = read(socket_s , data_rec_s, 40);
7  valread = read(socket_r , data_rec_r, 40);
8  ///DATA APPENDING
9  for (int b = 0; b < n_data_t-1; b++)
10 {
11     i++;
12     if (b<n_data_r){d_array[i] = data_rec_r[b];}
13     if (b>=n_data_r){d_array[i]=data_rec_s[b-4];}
14 }
15 k++;
16 ///ENLARGING THE DATA VECTOR FOR SAVING MORE DATA
17
18 if (k+1==SIZE_DATA/n_data_t){
19     j++;
20     d_array= realloc(d_array, (SIZE_DATA*j) * sizeof(double));
21     k=0;}
22
23 /// CALCULATE ACTUATION
24 /// IMPLEMENT HERE ANY CONTROL ALGORITHM
25 for (int mag = 0; mag < 8; mag++)
26 {
27
28     data_send_r[mag]=data_send_r[mag]+1;
29
30     if (data_send_r[mag]>2000)
31     {
32         data_send_r[mag]=0;
33     }}
34 for (int mag = 0; mag < 4; mag++)
35 {
36
37     data_send_s[mag]=data_send_s[mag]+1;
38
39     if (data_send_s[mag]>2000)

```

## 5. Control unit

---

```
40 {
41     data_send_s[mag]=0;
42 }}
43
44
45 ///////////////////////////////////////////////////SENDING ACTUATION
46 send(socket_s , data_send_s , sizeof(data_send_s) , 0 );
47 send(socket_r , data_send_r , sizeof(data_send_r) , 0 );
48 ///////////////////////////////////////////////////TIME CALCULATION
49 gettimeofday(&t2, NULL);
50 t_s=(t2.tv_sec - t1.tv_sec) + (t2.tv_usec - t1.tv_usec) / 1000000.0f;
51 tnext=tnext+t_s;
52 }
53
54 //// END OF LOOP
```

---

Completely in parallel the user interface is implemented. This code section could not be completed due to time constraints. Despite being prepared to hold any added options only the start and stop could be coded.

---

```
1  char command[1];
2  flag=1;
3  while ( flag==1){
4      scanf("%s",command);
5      if (strcmp(command,"s")==0){
6          flag=0;}}
7
8  printf("Thread interface closing everything \n");
9  return NULL;
10 }
11
12 //////////////////////////////////////// MAIN
```

---

Finally when the user presses the stop button the control loop stops and the collected data is processed. A text file is generated with 7 columns, the first for the time values and the rest for the measurements is millimeters.

---

```
1  mean_f=n_samples/cpu_time_used;
2  printf("done in %f \n",cpu_time_used);
3  printf("mean frequency = %f\n",mean_f);
4  printf("mean time_step = %f\n",1/mean_f);
5
6  /////// DATA PROCESSING
7
8
9  printf("Data clection ended with %d samples\n",n_samples );
10 ml=0;
11 for(int m = 0; m < n_samples; m++) {
12     for (int l = 0; l < n_data_t; l++)
13     { ml++;
14         fprintf (f, "%f",d_array[ml]);
15         if (l!=n_data_t-1){fprintf(f, " ");}
16     }
17     fprintf (f, "\n");}
```

---

```

18 //////////////////////////////////////////////////// CLOSING EVERYTHING
19 pthread_join(thread_id, NULL);

```

---

The final program implemented has the following structure Figure 5.11. Once the complete program is installed, the performance of the code will be tested. Firstly, assure that the code executes the instructions as it was designed. Secondly to test the performance of every isolated part of the code and clean any redundant instructions generated during the development process. Finally, to test the time consumption of the whole program and search for the bottle necks that the software has.

## 5.4 Software testing

As it was stated, every software has to pass some process of testing before the project is completed. The first test is to assure the function of the program as a whole. The code developed with this objective were several open loop codes. To input some signals into the actuators and read the measurements. The previous control unit was used to double check the measurements processed by the new sensing system. The first tests were successful, the output of the system coincide in both sensing systems and matched the expected response. The synchronization of the system is assure by the different threads into which the code is divided. Both CPU and the microcontrollers block their code until the receiving the data is completed.

The second part was to clean the software redundancies and rewrite each part in order to reduce instruction and therefore computational time. This procedure was completed and the code is clean and minimized.

Finally, the performance test were implemented in the embedded system. The objective is to check the final feedback control loop sample time and to analyze if it is possible to be reduced. After several isolated tests, the two main bottle necks the system has are the AD conversion and the wireless connection.

The AD conversion main issue is the communication protocol, as it was explained in the previous section, the connection between AD and the Raspberry pi is based on i2c. After several improvements and building a personalized library for this device the final conversion time is 0.375 milliseconds, this multiplied times 4 in the case of the rotating Raspberry Pi gives a AD conversion time of 1.5 milliseconds.

The second bottle neck are the communications, the wireless connection has a computation time that goes from 1 to 6 milliseconds. The reasons of this inconsistency as well as with the different improvements implemented are described in Chapter 6.

## 5.5 Conclusions

After the implementation of the program, it can be stated that the active vibration control loop is operative. Respect the measurements of the system, the 6 measurements are sampled within 1.5 milliseconds, the resolution is 12 bit and the noise is of the order of 5 AD levels, that gives a signal-to-noise ratio of 410, this means that the maximum noise

per sample is 410 times lower than the signal. The velocity of the system is known using the seventh sensor.

Regarding the electromagnets, the PWM has a frequency high enough (8000 Hz) to state that there is not added dynamics from the powering signal. All the 12 actuators are operational and the bandwidth is limited only by the own electromagnet physical characteristics (maximum 50 Hz bandwidth).

The overall sampling time of the embedded system is its weak point, the inconsistency of the wireless connection velocity is not improvable within this code. In Chapter 6 all the reasons are detailed. Nevertheless the test-rig is now operational and this its final working scheme Figure 5.12.

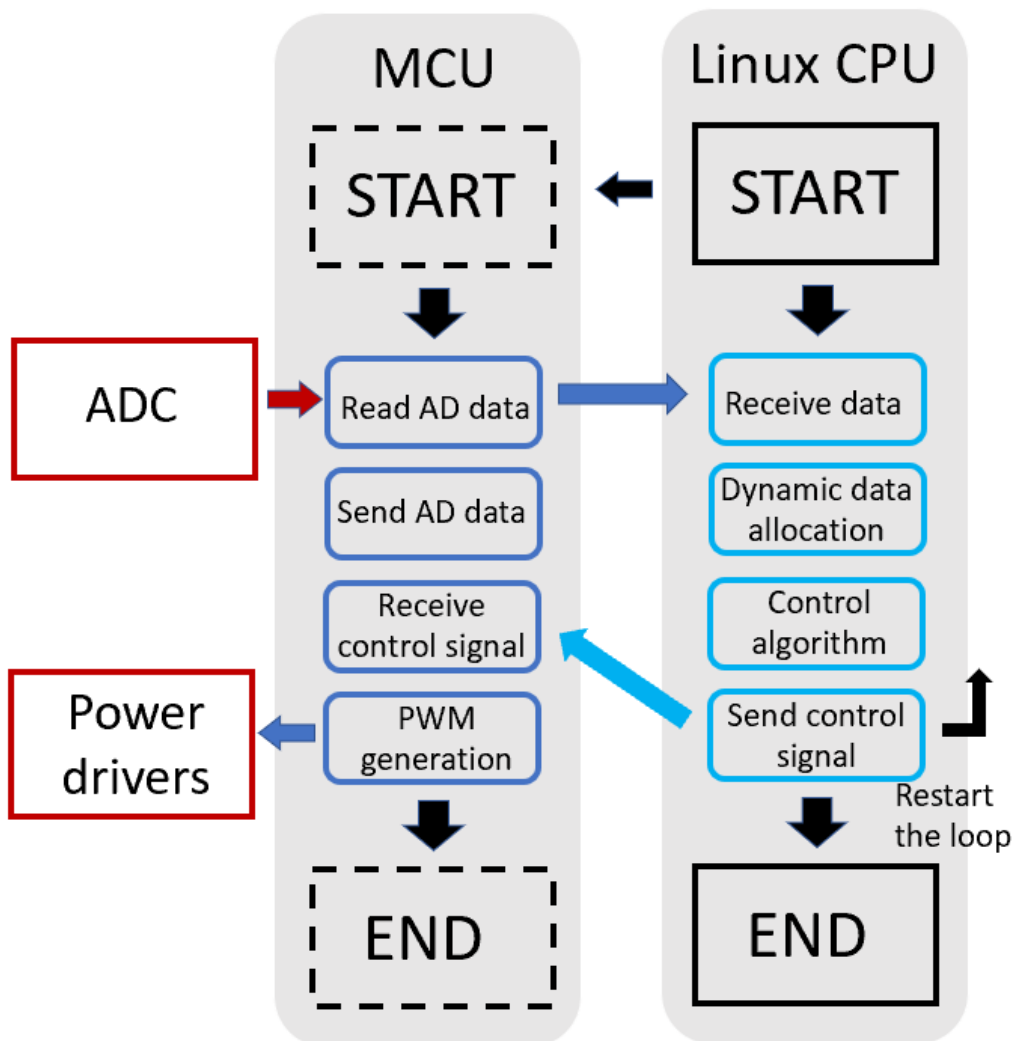


Figure 5.12. Configuration of the final software system



# 6 SYSTEM INTERCONNECTIONS

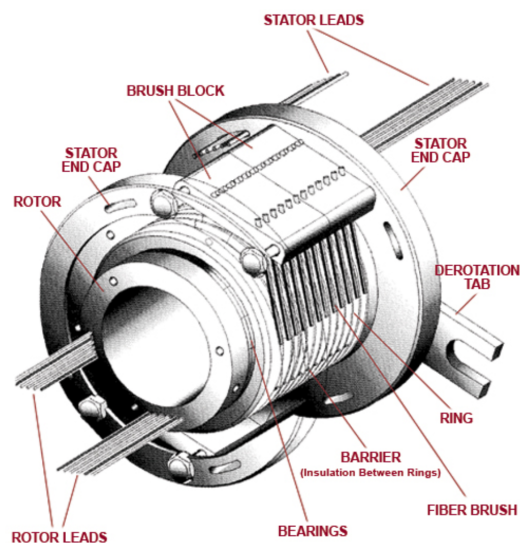
---

The system description is completed within the previous chapters, all the subsystems of the active vibrations control loop are described in the report. However, the interconnections of this test-rig in particular played an important role both in the design and the implementation of the feedback loop. The connections can be divided into wired and wireless connections, in order to understand the reasons of the new design, a brief description of the previous test-rig connections is offered.

## 6.1 Previous interconnection structure

As it has been described during this report, the previous control system was based in a computer that took all the computations and sent and receive all the signals from the test-rig system to the control unit located on the outside of the physical frame. This configuration needed a device that could transmit signals into the rotating frame, the 8 magnets and the 4 sensors related to the blades. The installed device is a slip-ring of 12 channels.

The slip ring is an active component whose structure permits to transmit electrical signals from a static to a rotating frame of reference [30]. The slip ring working principle is similar to a dc motor. Inside the slip ring, a number of brushes equal to the number of channels, maintains a physical contact with a rotating metallic disk. This contact is in charge of transmitting the electrical signal into the shaft Figure 6.1.



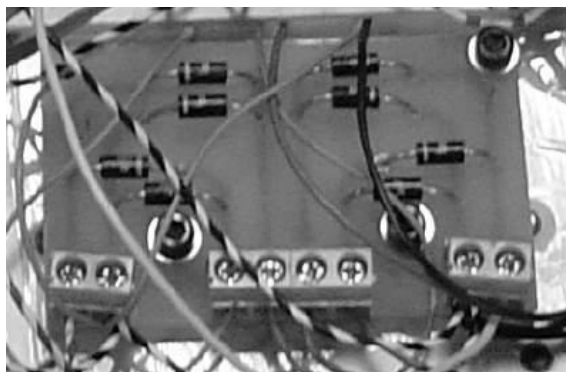
*Figure 6.1.* Slip-ring internal schematics

The slip-ring installed in this test-rig has 12 channels, this is the maximum signals including power signals that can be transmitted to the rotating frame. In the initial electronics the total devices in need of electrical connection with the external frame are the following:

- 4 sensor signals
- 1 power supply for the sensing electronics
- 8 analog signals to the electromagnets
- Neutral wire for sensing signals
- Neutral wire for power signals

These add up to 15 needed channels making not feasible the system, however a solution to this issue was documented in [3]. At least 3 signals needed to be reduced, the subsystem that could handle this cut in the number of signals was the actuating system. In Chapter 4 was introduced the solution, as each blade has a pair of electromagnets, it was decided to only actuate one magnet simultaneously. This means that either you activate the left or the right magnet. In order to rearrange the 2 signals into a unique one, it was decided that negative signal would mean to actuate the left magnet and positive the right one.

This solution requires additional circuitry, the electrical circuit is not documented, however, the working principle was described. A diode based circuit Figure 6.3 would redirect the negative voltage into a positive value and transmit this power to the left magnet while blocking the supply to the opposite magnet. With positive voltages the circuit blocks the left magnet and transmits the power to right one.



*Figure 6.2.* Electronics that handle the actuating signals

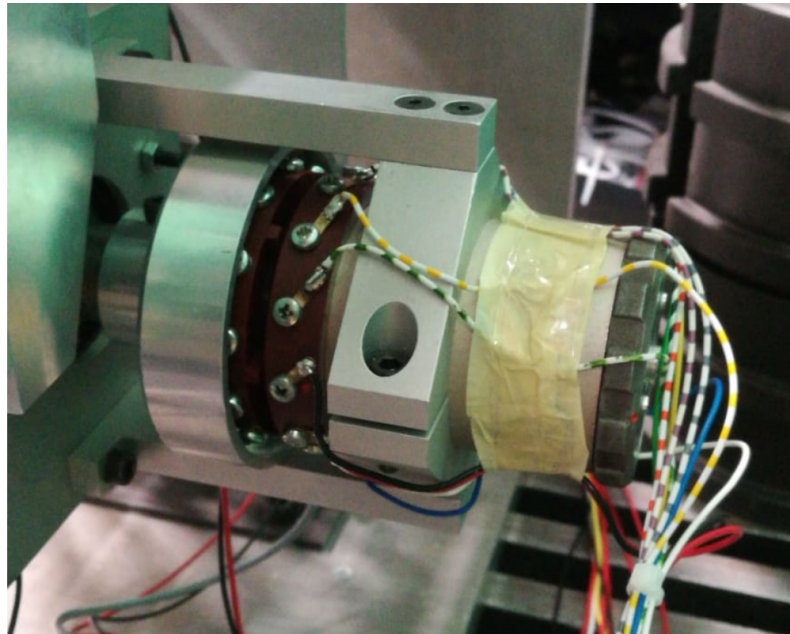
The final solution used only 11 signals through the slip-ring, however, the main drawback was the decrease in the quality of the actuating system increasing the non-linear behaviour of the electromagnets. Nevertheless, at the beginning of this project only 8 wires were operational. This added to the fact that the slip-ring brushes added noise to the sensor signal encourage the design of a new interconnection structure.

### 6.2 New wiring configuration

The configuration for the new active vibration control system has change principally because the addition of the wireless communication between the control unit CPU and

the two microcontrollers. This wireless connection reduces the use of slip-ring. Firstly, the sensor information now travels as a digital signal through the this wireless channel. Secondly the actuation signal is now generated in the drivers, not in the control unit. The digital control signal is sent wirelessly to the microcontrollers who will send it to the drivers. These two points reduce significantly the channels needed in the slip-ring. Only the power wires need to be connected through this device.

- 1 power supply to the Raspberry pi
- 1 power supply for the PCB and the eddy-current sensors
- 1 power supply to the drivers
- Neutral wire for power signals



*Figure 6.3.* Slip-ring installed in the test-rig

These are the connections from the external frame, in the inside of the rotating disk there are some connections worth to mention. The first one is the connections of the Analog to digital converter (ADC). The ADC needs power supply of 5 volts, two pin connection with the raspberry (SDA and SCL) and connection to the 4 outputs of the sensing system. The Raspberry Pi located in the rotor has a specific pin out that is documented in the Appendix A, the connections needed are the power supply, the connections with the ADC and the pins that generate the PWM signal are connected to the drivers. Finally, the drivers have as an input the power supply and the digital PWM signal. The final rotating disk connections are shown in Figure 6.4 and Figure 6.6.

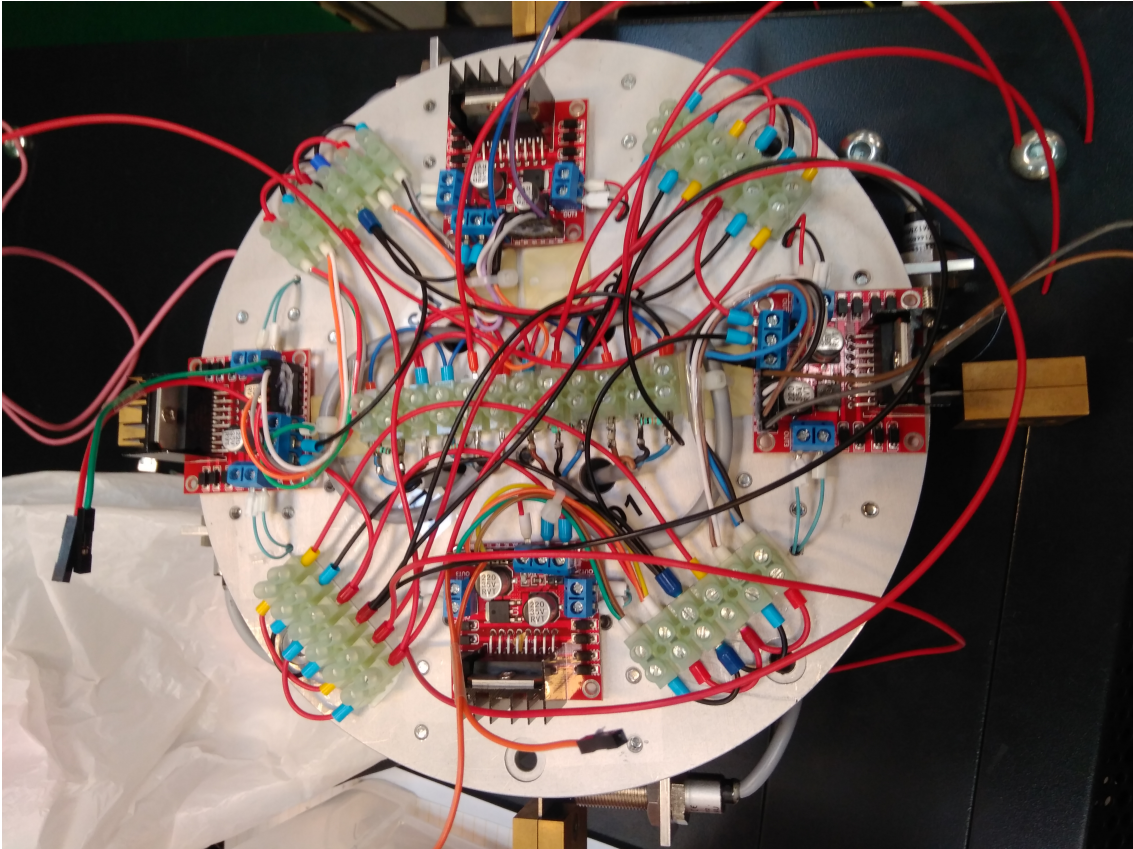


Figure 6.4. Connections inside the rotating disk: Power drivers

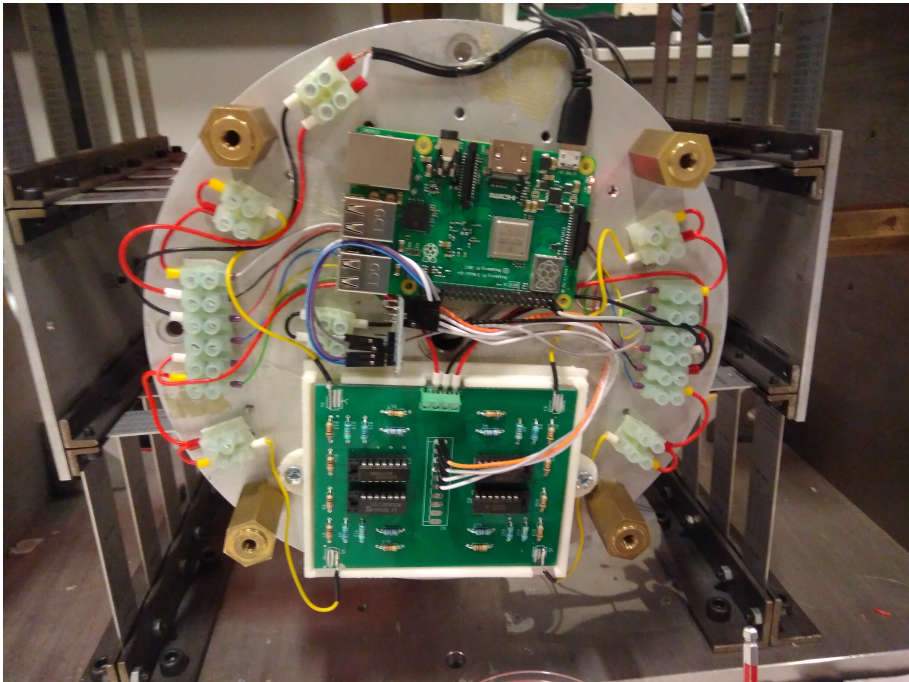


Figure 6.5. Connections inside the rotating disk: PCB, ADC and Raspberry Pi



## 6.3 Wireless connection

Wireless connection in embedded systems is a relatively new issue. The new technologies in telecommunication industries has lead into smaller and cheaper devices that transmit information without any wire. These new devices are now to be implemented into every kind of embedded system, the fact that a commercial board as Raspberry Pi has this feature incorporated reinforces this statement.

In the case of this test-rig, the microcontroller offers 2 main types of wireless connection: Bluetooth or Wi-fi. Bluetooth [31] technology is suitable for short ranges so could be used in the test-rig. However, the technology is not being develop in the last years in detriment of the Wi-fi technology.

Wi-fi uses high radio-frequency in conjunction with IEEE 802.11 standards [32]. This technology is being developed in every industry due to its high compatibility with every kind of software and hardware. In the case of embedded systems, every company has its own module in charge of bringing Wi-fi into the system. Also when it comes to software development, there are significantly more tools for developing Wi-fi applications than Bluetooth ones. Therefore, the wireless technology used in the test-rig connections is Wi-fi.

### 6.3.1 First implementation and results

The first action before implement the connection between the devices is to choose a network. In the location of the test-rig several networks were available, however, due to the congestion of these, a commercial router was used as a host. This commercial router was not connected to any worldwide network, instead it was configured as a wireless local area network (WLAN).



*Figure 6.6.* Wireless router installed: TP-Link Trådløs Gigabit Wi-Fi

The implementation of the Wi-fi technology is located in the sending and receiving sensor

and actuators data between the two Raspberry Pis and the Linux computer. The code that builds this connection requires the IP address of each device and the protocol to use.

The IP address was configured via the commercial router. Inside the wireless router an IP was defined to each device. This IP is now static and every time the system reconnects to this WLAN, the IP will be the same.

On the other hand the protocol to use depends on the application, two options are available UDP and TCP-IP [33]. The protocol defines how,when and what to send inside the information package. The information needs to be serialized into bits, ordered and sent, the receiver, in order to differentiate the new package from others needs extra information. This information is organized in form of a header that contains all the information before the actual message, for example TCP header is shown in Figure 6.7.

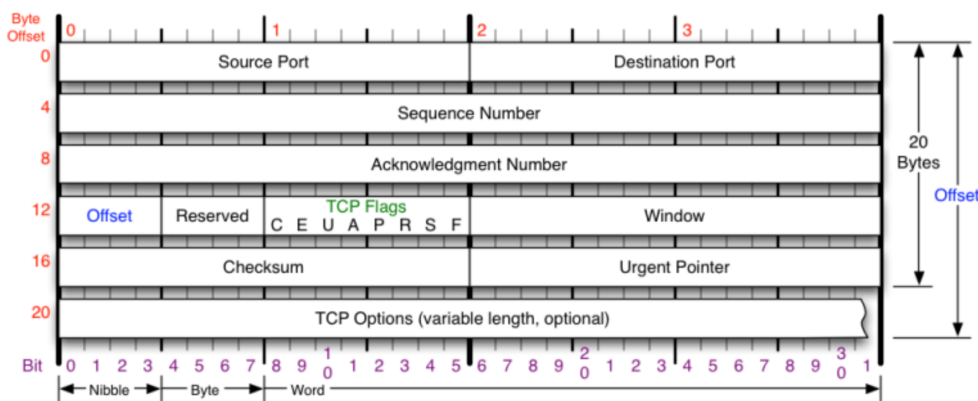


Figure 6.7. TCP protocol header format

This header occupies 40 bytes of information slowing down the transmissions, however, this protocol extra information assures the messages are sent to the right device and that the message is exactly the same as it was transmitted. On the other hand, UDP protocol has less header (8 bytes) however there is no way to assure from the emitter that the message was received correctly.

In the case of the test-rig both protocols offer their advantages to the close loop. The first one is more reliable but the second one is faster. Despite this trade-off there is one issue that made the UDP impossible to implement during the tests. The UDP protocol does not block the code when a transmission is started, this offers less waiting time but neglects the synchronization between the devices. This is not suitable feedback loop control theory where each data is following a continuous loop. Therefore the final election was to use TCP protocol.

```

1 setsockopt(server_fd, IPPROTO_TCP, TCP_NODELAY, (char *) &flag,sizeof(int));
2 address.sin_family = AF_INET;
3 address.sin_addr.s_addr =INADDR_ANY; //ADDRESS of the host here
4 address.sin_port = htons( PORT );    // Port defined up in the beggining 8080
5
6
7
8 //////////////////////////////////////////// PROGRAM

```

```
9
10 ///////////////////////////////////////////////////BIND THE SERVER
11 bind(server_fd, (struct sockaddr *)&address, sizeof(address));
12 printf("%s\n", "binded" );
13 listen (server_fd, 3);
14 /////////////////////////////////////////////////// CONNECTION WITH CLIENT
15 socket_s = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen);
16 socket_r = accept(server_fd, (struct sockaddr *)&address, (socklen_t *)&addrlen);
17 if (socket_r >= 0) { printf("%s\n", "connected to the stator" );}
18 if (socket_s >= 0) { printf("%s\n", "connected to the rotor" );}
19
20 data_send_r[0]=0;data_send_r[1]=1000;data_send_r[2]=500;data_send_r[3]=1500;data_send_r
   [4]=250;data_send_r[5]=1250;data_send_r[6]=750;data_send_r[7]=1750;
```

---

The results of the feedback loop were already explained in the previous Chapter 5. The time consumed in a whole feedback loop from the sensor measurement till the control action is implemented is not consistent. It can take from 2 to 7 milliseconds. The reason was discovered to be only the wireless connection code. The waiting time in the receiver is not constant, a further study of the reasons is needed.

### 6.3.2 Wi-fi protocol improvement

The time inconsistency issue was tried to improve in several ways. The first one being reordering the code in both the server and the clients to try to minimize the waiting times. This was effective in reducing the mean sample time but the variance on the sample time remain the same.

The problem, therefore, the issue was not in the software structure. This led to a deeper research in how the code decides when to send and receive the packages. The C libraries are standardized when it comes to establish TCP server client connections. However, the developer can establish more advanced options in order to adapt better the wireless communications to the developed application.

#### Nagel's algorithm and Delayed acknowledgment

The TCP protocol is designed to perform in world wide networks with an overwhelming quantity of data packages being transmitted each second. The TCP protocol needs to be able to handle this saturation of the network. In order to reduce the amount of packages with the TCP header (a message of 1 byte occupies 41 bytes) two main algorithms were design to reduce the congestion in the network. These are the Nagel's algorithm and the delayed acknowledgement.

The Nagel's algorithm [34] decides which packages are urgent to send and which ones can be put in queue in order to reduce the total messages transmitted through the network. The total information transmitted remains the same but the header bytes are drastically reduced. The delayed acknowledgment [35] works in the same direction of the Nagel's algorithm, they complement each other. The acknowledge time is defined as the time that takes the receiver to state that it has received a package since the first byte is received. This

delay derives in the accumulation of package in the receiver, converting several different messages into a unique one.

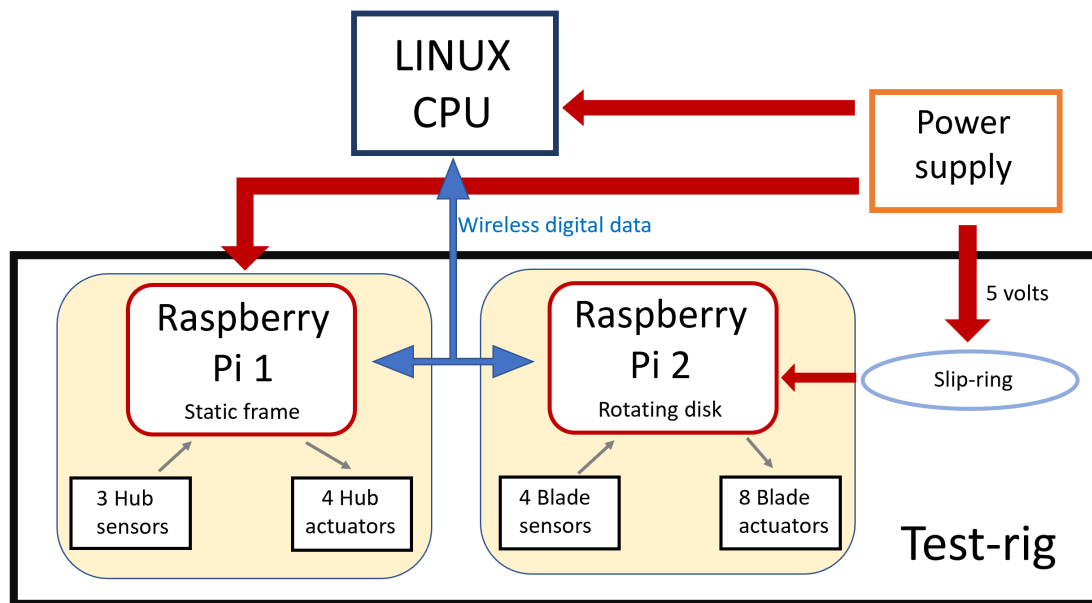
The implementation of these algorithms in the test-rig software has several consequences. In real time application these algorithms make impossible to assure not only the sampling time but also it adds the possibility that several sensing or actuating data gets accumulated in the system before applying them. Because of this other protocols are being develop in order to avoid these effects. The most real time protocol is TDMA. TDMA is more a code program than a protocol, this method establishes a loop in the client until the server acknowledges that the client has receive the information. This method synchronizes better the loop reducing the variance of each loop time step. The drawback that neglects the possibility to use this protocol is the total time consumed per transmitted packages. The method of enter into a loop waiting for a response adds computation stress to the system and gives a minimum time step of 10 milliseconds.

The last resource implemented is the deactivation of the algorithms explained in the Linux CPU and the microcontrollers. The C libraries offer this option for the code developer. Once this change was implemented the loop was again tested and there was no significant improvement in the results. The reason is that even the developer disables this algorithms in the implemented program, both server and clients run on the Linux OS. The OS has higher priority when scheduling the transmission of packages over the network.



## 6.4 Final system configuration

The embedded system connections are now totally described. The frame actuators and sensors are directly connected to the external power supply and electronics. The actuating and sensing system inside the rotating disk is connected to the power electronics through the slip-ring and to the control unit via Wi-fi with TCP protocol. The issue regarding the sampling time of the whole loop was determined to be caused because the inner scheduler of the Linux OS and the TCP protocol



*Figure 6.8.* Definite system configuration

# 7 CONCLUSION

---

The scope of this project was to design the hardware and software of an active vibration control system and implemented in a real test-rig. Several problems were formulated in each chapter with their respective solution. The project was developed with a time constraint external to the thesis given time, two other projects were based on implementing control strategies into this active vibration control system [1] [2]. This time constraint was decisive in some decisions in the design of the embedded system.

The objective of this report is also to offer a design method to implement feedback loops in real systems. The decisions and requirements are specified for each subsystem (sensors, actuators and control unit). The development process is described and explained, not only is important to achieve the desired characteristics in the embedded application, the development time is also a key point to consider.

## 7.1 Results

The embedded system was finally implemented in the actual test-rig with the following results. The complete feedback loop is accomplish, all the subsystems are connected to each other with success. For every subsystem the characteristics were changed and improved.

The sensing system has now a different set of sensors with less noise and more sensitivity. The measurements are now all processed by the same sensor model, unifying the results among the different measurements. The electronics following the sensors adequate the signal to a standard 5 volt range that can be acquire by the standard analog to digital converters.

The actuating system was only change in the actuation signal generation method. The new system prevents peaks of voltage that could damage the power supplies or electronics and reduces the non-linearities of the electromagnets. However, the electromagnets were not able to perform up to the rest of the system characteristics. Both hub and blade actuators do not perform correctly when the actuating signals have higher frequencies.

The control unit is completely different now. The system is now decentralized, the embedded system consists of a computer and two microcontrollers that connect wirelessly. The computational unit has now the computational capacity to implement any kind of advanced control. All the software is programmed in standard C, making the software accessible and open source.

The final configuration is really modular, meaning that each subsystem is easy to change in order to improve the system. The power drivers, AD converters and MCUs could change and the system could be easily reconfigured. The final system only uses wires for powering the devices, decreasing the rotating wires and the noise associated with the slip-ring.

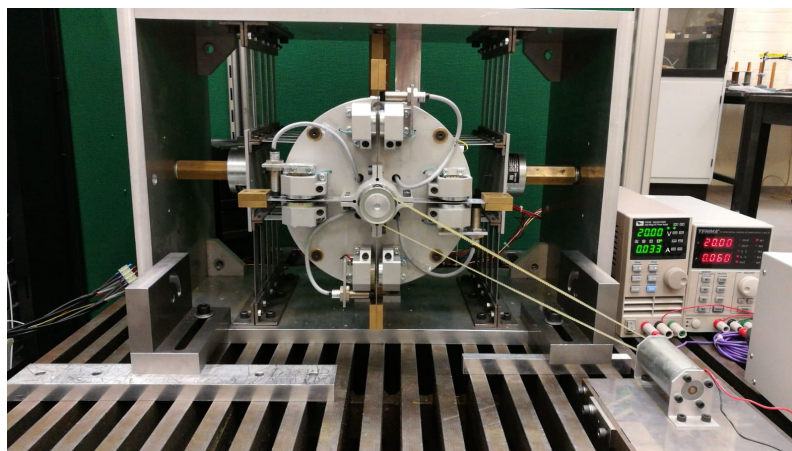
### 7.1.1 Perspectives

This thesis project is only effective if other researchers can use the test-rig to implement active vibration control into a real system. This thesis objective is to facilitate future users the active vibration control research. Secondly, this thesis can also be used as an example for the developing of different embedded systems in completely different scenarios, the methodology is general to any system. Finally the network connections permit to research the effects of each wireless technology and configuration in a real feedback loop.

## 7.2 Future work

Despite all the achievements described above, the system created of this project is far from perfect or finished. Technologies improve quicker year to year and it is assumed that in the next years, the improvements in software and hardware make this configuration outdated. Nevertheless there are some key points to change in order to improve the overall characteristics of the active vibration control system.

- Change the electromagnets to other con better performance, the static forces are correct, however, the inductance in the actuators is too high. The electromagnets are usually designed to generate the maximum magnetic flux leading to a high inductance in the coil . This is why designing personalized electromagnets would be a path worth to research.
- Research other wireless protocol whose sampling time is more constant. The different protocols tried on this thesis are the most common ones. Other protocols could be developed in C language and installed into the software.
- Change the microcontroller to a non operative system based one in order to improve the loop time consistency.
- Change the AD converter of 4 channels to 4 ADC of 1 channel. The addition of AD devices would enable the acquisition in parallel, reducing by 4 the sampling time of the sensors.
- Develop a graphical user interface. Although all the programs are coded in C language, the Linux CPU could handle the user interaction with standard python interfaces.



*Figure 7.1.* Final state of the Test-rig

# REFERENCES

---

- [1] Maria Beneyto Gomez-Polo. Classical control design theory applied to mitigate rotor-blade vibrations - a numerical investigation, 2019.
- [2] Ignacio Escudero Sarabia. Implementation of different types of controllers to reduce rotor-blade vibrations – an experimental approach, 2019.
- [3] René H. Christensen and Ilmar Santos. A study of active rotor-blade vibration control using electro-magnetic actuation: Part 1 — theory. In *A Study of Active Rotor-Blade Vibration Control Using Electro-Magnetic Actuation: Part 1 — Theory*, volume 6, 01 2004. doi: 10.1115/GT2004-53509.
- [4] Christian Sidelmann Jakobsen. Shaft based attenuation of coupled rotor-blade vibrations via gain-scheduled controlled – theory, akselbaserede reduktion af koblede rotorblad vibrationer via tids varierende regulator – teori, 2012.
- [5] Jesper Berg Hansen. Theoretical and experimental control of rotor-blade systems, teoretisk og eksperimentel regulering af rotorbladesystemer, 2013.
- [6] A.F. Twizell. Resistance strain gauges. *Students Quarterly Journal*, 21(82):93, 1950. ISSN 20537875, 00392871. doi: 10.1049/sqj.1950.0082.
- [7] Jeong-Yeol Yoon. Wheatstone bridge. *Introduction To Biosensors*, pages 79–90, 2016. doi: 10.1007/978-3-319-27413-3\_5,10.1007/978-3-319-27413-3.
- [8] Keyence. What are Displacement Sensors (Displacement Gauges) and Dimension Measurement Systems? | Measurement Library | KEYENCE America, 2017. URL [https://www.keyence.com/ss/products/measure/measurement\\_library/basic/products\\_info/](https://www.keyence.com/ss/products/measure/measurement_library/basic/products_info/).
- [9] S. Fizek, M. Reisinger, S. Silbers, and W. Amrhein. An electromagnet model comprehending eddy current and end effects. In *2015 IEEE 11th International Conference on Power Electronics and Drive Systems*, pages 668–672, June 2015. doi: 10.1109/PEDS.2015.7203454.
- [10] AB-022 : PWM Frequency for Linear Motion Control - Precision Microdrives. URL <https://www.precisionmicrodrives.com/content/ab-022-pwm-frequency-for-linear-motion-control/>.
- [11] Maurizio Di Paolo Emilio and Maurizio Paolo Emilio. Microcontroller design. *Embedded Systems Design for High-speed Data Acquisition and Control*, pages 33–48, 2014. doi: 10.1007/978-3-319-06865-7\_3,10.1007/978-3-319-06865-7.
- [12] Ilmar Santos. Mechatronics applied to machine elements with focus on active control of bearing, shaft and blade dynamics, 2010.

- 
- [13] Rene H. Christensen and Ilmar F. Santos. Active rotor-blade vibration control using shaft-based electromagnetic actuation. *Journal of Engineering for Gas Turbines and Power-transactions of the Asme*, 128(3):644–652, 2006. ISSN 15288919, 07424795. doi: 10.1115/1.2056533.
- [14] Juan Camino and Ilmar Santos. A periodic h2 state feedback controller for a rotor-blade system. In *A periodic H2 state feedback controller for a rotor-blade system*, 09 2018.
- [15] E. Mirambell and E. Real. On the calculation of deflections in structural stainless steel beams: An experimental and numerical investigation. *Journal of Constructional Steel Research*, 2000. ISSN 0143974X. doi: 10.1016/S0143-974X(99)00051-6.
- [16] Maurizio Di Paolo Emilio and Maurizio Paolo Emilio. Operational amplifier. *Microelectronics*, pages 45–54, 2015. doi: 10.1007/978-3-319-22545-6\_4,10.1007/978-3-319-22545-6.
- [17] René Hardam Christensen and Ilmar Ferreira Santos. Control of rotor-blade coupled vibrations using shaft-based actuation. *Shock and Vibration*, 13(4-5):255–271, 2006. ISSN 18759203, 10709622. doi: 10.1155/2006/398658.
- [18] René Hardam Christensen and Ilmar Ferreira Santos. Control of rotor-blade coupled vibrations using shaft-based actuation. 2006.
- [19] TL James. Electro magnets. *Nature*, 64(1650):168–170, 1901. ISSN 14764687, 00280836. doi: 10.1038/064168a0.
- [20] Improvements in electro magnets. *Scientific American*, 54(7):102–102, 1886. ISSN 19467087, 00368733. doi: 10.1038/scientificamerican02131886-102.
- [21] C Walter and John Chrassont. Linear and nonlinear state-space controllers for magnetic levitation. Technical Report 11, 1996.
- [22] D. G. Holmes and T. A. Lipo. *Pulse Width Modulation for Power Converters: Principles and Practice*. IEEE, 2003. ISBN 9780470546284. doi: 10.1109/9780470546284.app1.
- [23] Gordon E Moore, Carver Mead, Alan Turing, and Douglas Engelbart. Moore ’ s law. *Online*, 2007.
- [24] Allan G. Bromley. Hardware experiments with cisc and risc computer architectures. *Acm International Conference Proceeding Series*, 129322:207–215, 1997. doi: 10.1145/299359.299389.
- [25] (1) What is different between 16 bit and 32 bit microcontroller? - Quora. URL <https://www.quora.com/What-is-different-between-16-bit-and-32-bit-microcontroller>.
- [26] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):7458761, 94–100, 2016. ISSN 19374194, 07407459. doi: 10.1109/MS.2016.68.

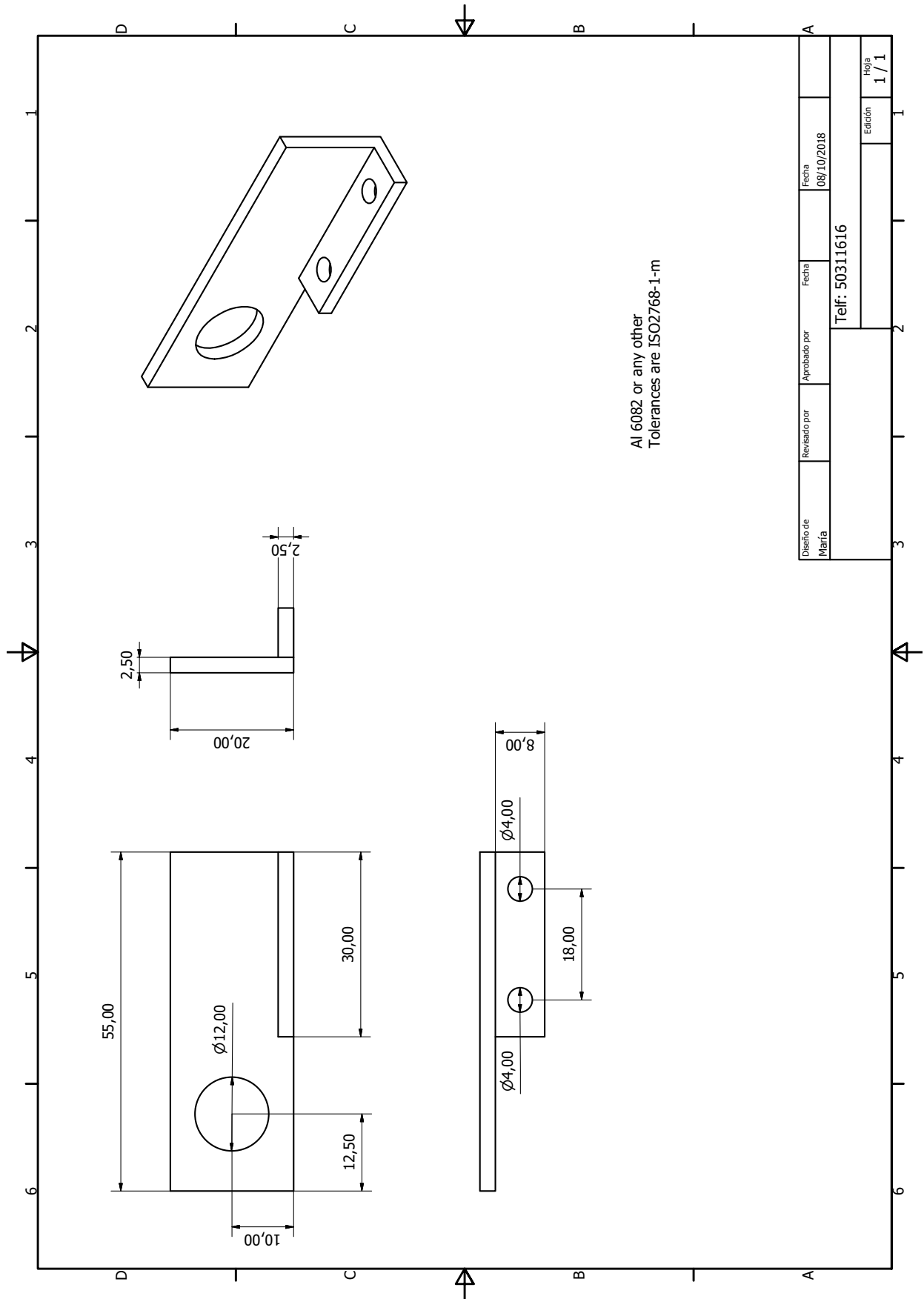
- [27] José Miguel, David Báez-López, and David Alfredo Báez Villegas. Object-Oriented Programming. In *MATLAB® Handbook with Applications to Mathematics, Science, Engineering, and Finance*. 2019. doi: 10.1201/9781315228457-7.
- [28] Warren Gay. I2c. *Beginning Stm32*, pages 195–221, 2018. doi: 10.1007/978-1-4842-3624-6\_11,10.1007/978-1-4842-3624-6.
- [29] PWM signal generation using DMA. Technical report, 2010. URL <http://www.renesas.com>.
- [30] What is a slip ring? <http://www.trolexengineering.co.uk/what-is-a-slip-ring.html>, 2019. Accessed: 2019-01-01.
- [31] Myra Dideles. Bluetooth. *Crossroads*, 9(4):11–18, 2003. ISSN 15284980, 15284972. doi: 10.1145/904080.904083.
- [32] Robin Singh. Wi-fi. *Computer Bulletin (london, 1986)*, 45(6):28, 2003. ISSN 1464357x, 00104531. doi: 10.1093/combul/45.6.28.
- [33] What is the difference between TCP and UDP? URL <https://support.holmsecurity.com/hc/en-us/articles/212963869-What-is-the-difference-between-TCP-and-UDP->.
- [34] Nagel’s Algorithm – TCP\_NODELAY vs. TCP\_LOW\_LATENCY – Bearded\_Admin, . URL <http://beardedadmin.net/?p=140>.
- [35] Tweaking TCP for Real-time Applications: Nagle’s Algorithm and Delayed Acknowledgment · CodeAhoy, . URL <https://codeahoy.com/2017/03/19/tweaking-tcp-for-real-time-applications-nagle-algorithm-and-delayed-acknowledgment/>.

# A PLANES AND DATA SHEETS

---

This appendix includes the plane of the sensor support piece described in Chapter 3 as well as all the datasheet of the different electronic components present in this project. The components are the following.

1. Sensor supporter piece planes
2. Eddy-current sensor Pulsotronic kj4-m12mn50-anu
3. Operational amplifier LM 741
4. Electromagnets RS Pro Magnetic Lock, 1670N
5. H-bridge driver L298n
6. Microcontroller board Raspberry Pi 3 model b+
7. Analog to digital converter ADS 1105






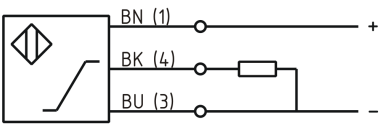




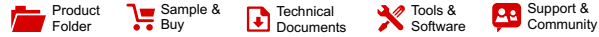
**KJ4-M12MN50-ANU**

Pulsotronic GmbH & Co. KG Neue Schichtstraße 14b 09366 Niederdorf Deutschland +49 (0)37296 930-200

[Article Page](#)

Product Description		inductive sensor analogue
Name	KJ4-M12MN50-ANU	
Order number	08317144800	
Switching distance	4 mm	
Mounting	 non shielded	
Switch type	 analog	
Signal Type	Analog Voltage	
Connection	 Cable	
Connection diagramm		
Dimension (in mm)	fine-pitch thread M12 x 50	
Technical Specification		
Operating voltage	11 - 35 VDC	
Max. ripple	≤ 10 %	
No load current	≤ 5 mA	
Switching frequency	400	
Operating temperature	-25 ° C to 70 ° C	
Temperature drift	+/-5 %	
Repeat accuracy	≤ 1 %	
Linearity	≤ 5 %	
Analog output	Pulso_has1	
Digital output		
Protection category	IP67	
EMC-level	DIN EN 60947-5-7:2004-06	
Material active face	PCB	
Termination	2m PCV 3x0,34mm <sup>2</sup>	

**Errors and omissions exceptet**



LM741

SNOSC25D –MAY 1998–REVISED OCTOBER 2015

## LM741 Operational Amplifier

### 1 Features

- Overload Protection on the Input and Output
- No Latch-Up When the Common-Mode Range is Exceeded

### 2 Applications

- Comparators
- Multivibrators
- DC Amplifiers
- Summing Amplifiers
- Integrator or Differentiators
- Active Filters

### 3 Description

The LM741 series are general-purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439, and 748 in most applications.

The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common-mode range is exceeded, as well as freedom from oscillations.

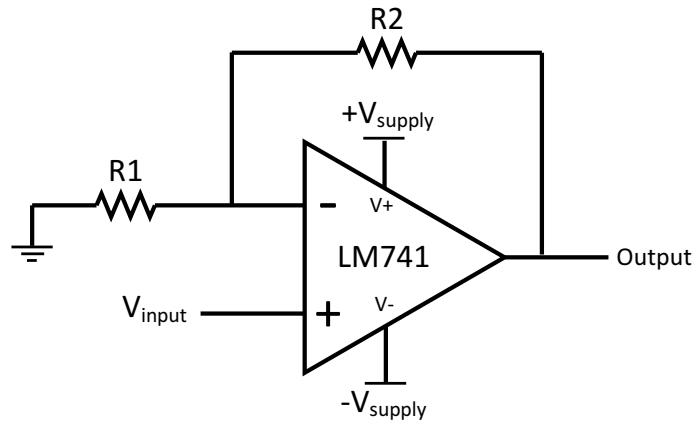
The LM741C is identical to the LM741 and LM741A except that the LM741C has their performance ensured over a 0°C to +70°C temperature range, instead of –55°C to +125°C.

#### Device Information<sup>(1)</sup>

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM741	TO-99 (8)	9.08 mm x 9.08 mm
	CDIP (8)	10.16 mm x 6.502 mm
	PDIP (8)	9.81 mm x 6.35 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

#### Typical Application



An IMPORTANT NOTICE at the end of this data sheet addresses availability, warranty, changes, use in safety-critical applications, intellectual property matters and other important disclaimers. PRODUCTION DATA.

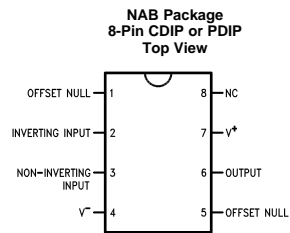
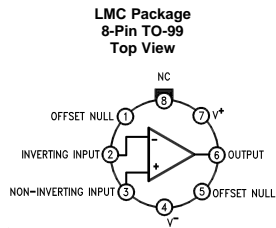


**LM741**

www.ti.com

SNOSC25D – MAY 1998 – REVISED OCTOBER 2015

**5 Pin Configuration and Functions**



LM741H is available per JM38510/10101

**Pin Functions**

PIN		I/O	DESCRIPTION
NAME	NO.		
INVERTING INPUT	2	I	Inverting signal input
NC	8	N/A	No Connect, should be left floating
NONINVERTING INPUT	3	I	Noninverting signal input
OFFSET NULL	1, 5	I	Offset null pin used to eliminate the offset voltage and balance the input voltages.
OFFSET NULL			
OUTPUT	6	O	Amplified signal output
V+	7	I	Positive supply voltage
V-	4	I	Negative supply voltage



**LM741**

SNOSC25D – MAY 1998 – REVISED OCTOBER 2015

www.ti.com

**6 Specifications**

**6.1 Absolute Maximum Ratings**

over operating free-air temperature range (unless otherwise noted)<sup>(1)(2)(3)</sup>

		MIN	MAX	UNIT
Supply voltage	LM741, LM741A		±22	V
	LM741C		±18	
Power dissipation <sup>(4)</sup>			500	mW
Differential input voltage			±30	V
Input voltage <sup>(5)</sup>			±15	V
Output short circuit duration		Continuous		
Operating temperature	LM741, LM741A	-50	125	°C
	LM741C	0	70	
Junction temperature	LM741, LM741A		150	°C
	LM741C		100	
Soldering information	PDIP package (10 seconds)		260	°C
	CDIP or TO-99 package (10 seconds)		300	
Storage temperature, T <sub>stg</sub>		-65	150	°C

- (1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
- (2) For military specifications see RETS741X for LM741 and RETS741AX for LM741A.
- (3) If Military/Aerospace specified devices are required, please contact the TI Sales Office/Distributors for availability and specifications.
- (4) For operation at elevated temperatures, these devices must be derated based on thermal resistance, and T<sub>j</sub> max. (listed under "Absolute Maximum Ratings"). T<sub>j</sub> = T<sub>A</sub> + (θ<sub>JA</sub> P<sub>D</sub>).
- (5) For supply voltages less than ±15 V, the absolute maximum input voltage is equal to the supply voltage.

**6.2 ESD Ratings**

			VALUE	UNIT
V <sub>(ESD)</sub>	Electrostatic discharge	Human body model (HBM), per ANSI/ESDA/JEDEC JS-001 <sup>(1)</sup>	±400	V

- (1) Level listed above is the passing level per ANSI, ESDA, and JEDEC JS-001. JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

**6.3 Recommended Operating Conditions**

over operating free-air temperature range (unless otherwise noted)

		MIN	NOM	MAX	UNIT
Supply voltage (VDD-GND)	LM741, LM741A	±10	±15	±22	V
	LM741C	±10	±15	±18	
Temperature	LM741, LM741A	-55		125	°C
	LM741C	0		70	

**6.4 Thermal Information**

THERMAL METRIC <sup>(1)</sup>		LM741			UNIT
		LMC (TO-99)	NAB (CDIP)	P (PDIP)	
		8 PINS	8 PINS	8 PINS	
R <sub>θJA</sub>	Junction-to-ambient thermal resistance	170	100	100	°C/W
R <sub>θJC(top)</sub>	Junction-to-case (top) thermal resistance	25	—	—	°C/W

- (1) For more information about traditional and new thermal metrics, see the *Semiconductor and IC Package Thermal Metrics* application report, [SPRA953](#).



**LM741**

www.ti.com

SNOSC25D – MAY 1998 – REVISED OCTOBER 2015

**6.5 Electrical Characteristics, LM741<sup>(1)</sup>**

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Input offset voltage	$R_S \leq 10 \text{ k}\Omega$	$T_A = 25^\circ\text{C}$	1	5	mV
		$T_{AMIN} \leq T_A \leq T_{AMAX}$		6	mV
Input offset voltage adjustment range	$T_A = 25^\circ\text{C}, V_S = \pm 20 \text{ V}$		$\pm 15$		mV
Input offset current	$T_A = 25^\circ\text{C}$ $T_{AMIN} \leq T_A \leq T_{AMAX}$		20	200	nA
			85	500	
Input bias current	$T_A = 25^\circ\text{C}$ $T_{AMIN} \leq T_A \leq T_{AMAX}$		80	500	nA
				1.5	$\mu\text{A}$
Input resistance	$T_A = 25^\circ\text{C}, V_S = \pm 20 \text{ V}$	0.3	2		M $\Omega$
Input voltage range	$T_{AMIN} \leq T_A \leq T_{AMAX}$		$\pm 12$	$\pm 13$	V
Large signal voltage gain	$V_S = \pm 15 \text{ V}, V_O = \pm 10 \text{ V}, R_L \geq 2 \text{ k}\Omega$	$T_A = 25^\circ\text{C}$	50	200	V/mV
		$T_{AMIN} \leq T_A \leq T_{AMAX}$	25		
Output voltage swing	$V_S = \pm 15 \text{ V}$	$R_L \geq 10 \text{ k}\Omega$	$\pm 12$	$\pm 14$	V
		$R_L \geq 2 \text{ k}\Omega$	$\pm 10$	$\pm 13$	
Output short circuit current	$T_A = 25^\circ\text{C}$		25		mA
Common-mode rejection ratio	$R_S \leq 10 \Omega, V_{CM} = \pm 12 \text{ V}, T_{AMIN} \leq T_A \leq T_{AMAX}$	80	95		dB
Supply voltage rejection ratio	$V_S = \pm 20 \text{ V}$ to $V_S = \pm 5 \text{ V}, R_S \leq 10 \Omega, T_{AMIN} \leq T_A \leq T_{AMAX}$	86	96		dB
Transient response	Rise time	$T_A = 25^\circ\text{C}, \text{unity gain}$		0.3	$\mu\text{s}$
	Overshoot			5%	
Slew rate	$T_A = 25^\circ\text{C}, \text{unity gain}$		0.5		V/ $\mu\text{s}$
Supply current	$T_A = 25^\circ\text{C}$		1.7	2.8	mA
Power consumption	$V_S = \pm 15 \text{ V}$	$T_A = 25^\circ\text{C}$	50	85	mW
		$T_A = T_{AMIN}$	60	100	
		$T_A = T_{AMAX}$	45	75	

(1) Unless otherwise specified, these specifications apply for  $V_S = \pm 15 \text{ V}, -55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$  (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to  $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ .

**6.6 Electrical Characteristics, LM741A<sup>(1)</sup>**

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Input offset voltage	$R_S \leq 50 \Omega$	$T_A = 25^\circ\text{C}$	0.8	3	mV
		$T_{AMIN} \leq T_A \leq T_{AMAX}$		4	mV
Average input offset voltage drift				15	$\mu\text{V}/^\circ\text{C}$
Input offset voltage adjustment range	$T_A = 25^\circ\text{C}, V_S = \pm 20 \text{ V}$		$\pm 10$		mV
Input offset current	$T_A = 25^\circ\text{C}$ $T_{AMIN} \leq T_A \leq T_{AMAX}$		3	30	nA
				70	
Average input offset current drift				0.5	nA/ $^\circ\text{C}$
Input bias current	$T_A = 25^\circ\text{C}$		30	80	nA
	$T_{AMIN} \leq T_A \leq T_{AMAX}$			0.21	$\mu\text{A}$
Input resistance	$T_A = 25^\circ\text{C}, V_S = \pm 20 \text{ V}$	1	6		M $\Omega$
	$T_{AMIN} \leq T_A \leq T_{AMAX}, V_S = \pm 20 \text{ V}$		0.5		
Large signal voltage gain	$V_S = \pm 20 \text{ V}, V_O = \pm 15 \text{ V}, R_L \geq 2 \text{ k}\Omega$	$T_A = 25^\circ\text{C}$	50		V/mV
		$T_{AMIN} \leq T_A \leq T_{AMAX}$	32		
		$V_S = \pm 5 \text{ V}, V_O = \pm 2 \text{ V}, R_L \geq 2 \text{ k}\Omega, T_{AMIN} \leq T_A \leq T_{AMAX}$	10		

(1) Unless otherwise specified, these specifications apply for  $V_S = \pm 15 \text{ V}, -55^\circ\text{C} \leq T_A \leq +125^\circ\text{C}$  (LM741/LM741A). For the LM741C/LM741E, these specifications are limited to  $0^\circ\text{C} \leq T_A \leq +70^\circ\text{C}$ .



ENGLISH

Datasheet

## RS Pro Magnetic Lock, 1670N

RS Stock No: **739-3233**



### Product Details

RS Pro magnetic lock provides a holding force when de-energised and release when pulsed. This electromagnetic lock offers 1670 N dynamic holding force capacity and is suitable for typical applications such as machine mechanisms, door/guard locking and remote hold/release requirements.

### Features and Benefits

- Holding force of 1670 N
- Suitable for typical applications such as machine mechanisms, door/guard locking and remote hold/release requirements
- Provides a holding force when de-energised and release when pulsed

RS, Professionally Approved Products, gives you professional quality parts across all products categories. Our range has been testified by engineers as giving comparable quality to that of the leading brands without paying a premium price.



ENGLISH

**Specifications:**

AC or DC Operation	DC
Diameter	65 mm
Holding Force	1670 N
Length	111 mm
Supply Voltage	24 V dc
Depth	35 mm
Application	Machine Mechanisms, Door/Guard Locking & Remote Hold/Release Requirements
Current Rating	340 mA
Finish	Bright Nickel Plated with Machined Face
IP Rating	IP20
Power Consumption	8.2 W

RS, Professionally Approved Products, gives you professional quality parts across all products categories. Our range has been testified by engineers as giving comparable quality to that of the leading brands without paying a premium price.



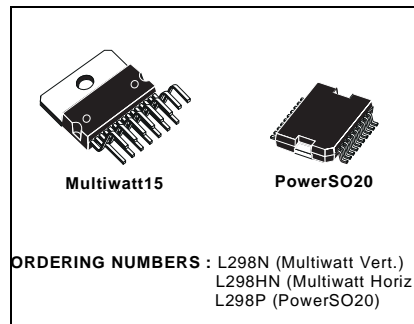
# L298

## DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)

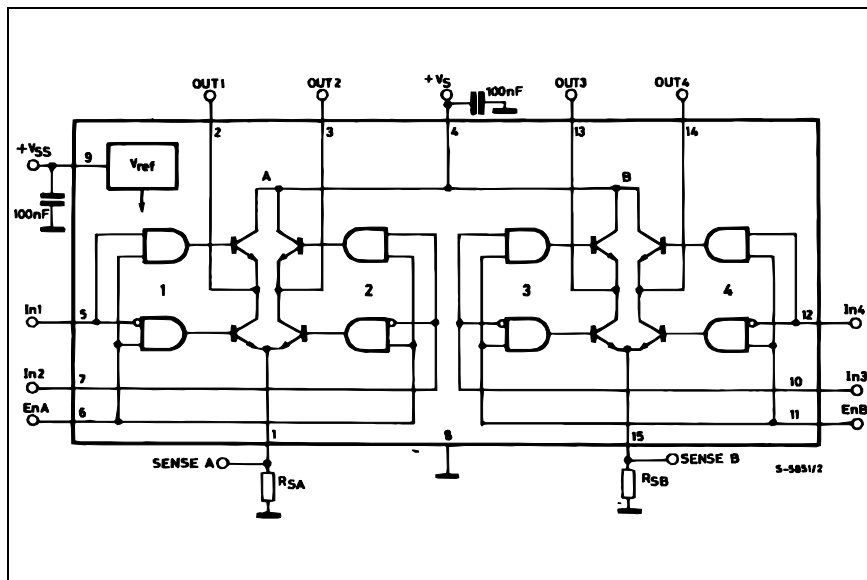
### DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the con-



nection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

### BLOCK DIAGRAM



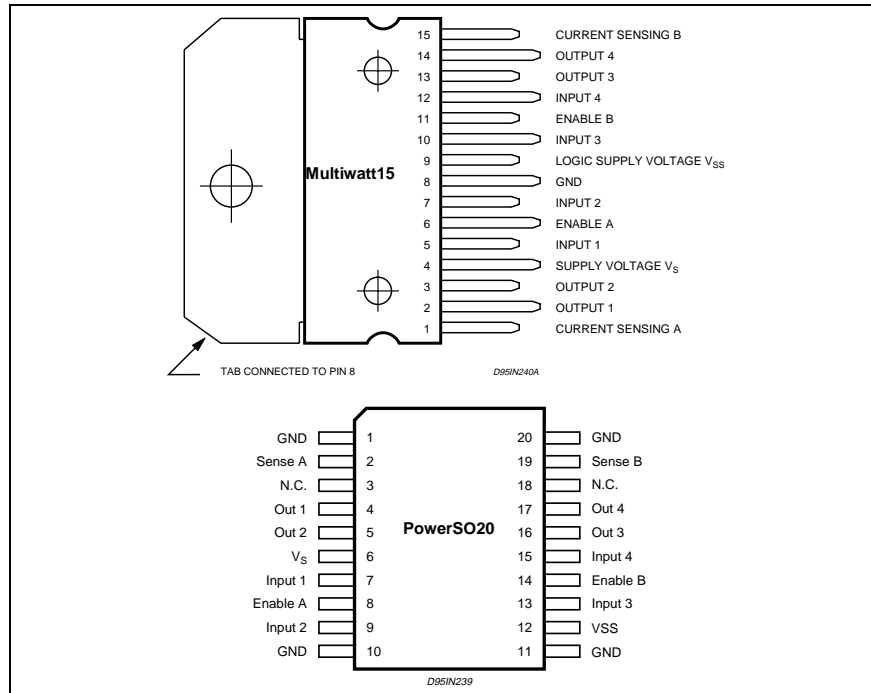


**L298**

**ABSOLUTE MAXIMUM RATINGS**

Symbol	Parameter	Value	Unit
$V_S$	Power Supply	50	V
$V_{SS}$	Logic Supply Voltage	7	V
$V_i, V_{en}$	Input and Enable Voltage	-0.3 to 7	V
$I_O$	Peak Output Current (each Channel)		
	- Non Repetitive ( $t = 100\mu s$ )	3	A
	- Repetitive (80% on -20% off; $t_{on} = 10ms$ )	2.5	A
	- DC Operation	2	A
$V_{sens}$	Sensing Voltage	-1 to 2.3	V
$P_{tot}$	Total Power Dissipation ( $T_{case} = 75^\circ C$ )	25	W
$T_{op}$	Junction Operating Temperature	-25 to 130	$^\circ C$
$T_{stg}, T_J$	Storage and Junction Temperature	-40 to 150	$^\circ C$

**PIN CONNECTIONS (top view)**



**THERMAL DATA**

Symbol	Parameter	PowerSO20	Multiwatt15	Unit
$R_{th\ j-case}$	Thermal Resistance Junction-case	Max. -	3	$^\circ C/W$
$R_{th\ j-amb}$	Thermal Resistance Junction-ambient	Max. 13 (*)	35	$^\circ C/W$

(\*) Mounted on aluminum substrate



**PIN FUNCTIONS** (refer to the block diagram)

MW.15	PowerSO	Name	Function
1;15	2;19	Sense A; Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2;3	4;5	Out 1; Out 2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V <sub>S</sub>	Supply Voltage for the Power Output Stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5;7	7;9	Input 1; Input 2	TTL Compatible Inputs of the Bridge A.
6;11	8;14	Enable A; Enable B	TTL Compatible Enable Input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	V <sub>SS</sub>	Supply Voltage for the Logic Blocks. A100nF capacitor must be connected between this pin and ground.
10; 12	13;15	Input 3; Input 4	TTL Compatible Inputs of the Bridge B.
13; 14	16;17	Out 3; Out 4	Outputs of the Bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
-	3;18	N.C.	Not Connected

**ELECTRICAL CHARACTERISTICS** (V<sub>S</sub> = 42V; V<sub>SS</sub> = 5V, T<sub>j</sub> = 25°C; unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V <sub>S</sub>	Supply Voltage (pin 4)	Operative Condition	V <sub>IH</sub> +2.5		46	V
V <sub>SS</sub>	Logic Supply Voltage (pin 9)		4.5	5	7	V
I <sub>S</sub>	Quiescent Supply Current (pin 4)	V <sub>en</sub> = H; I <sub>L</sub> = 0	V <sub>i</sub> = L	13	22	mA
			V <sub>i</sub> = H	50	70	mA
		V <sub>en</sub> = L	V <sub>i</sub> = X		4	mA
I <sub>SS</sub>	Quiescent Current from V <sub>SS</sub> (pin 9)	V <sub>en</sub> = H; I <sub>L</sub> = 0	V <sub>i</sub> = L	24	36	mA
			V <sub>i</sub> = H	7	12	mA
		V <sub>en</sub> = L	V <sub>i</sub> = X		6	mA
V <sub>IL</sub>	Input Low Voltage (pins 5, 7, 10, 12)		-0.3		1.5	V
V <sub>IH</sub>	Input High Voltage (pins 5, 7, 10, 12)		2.3		V <sub>SS</sub>	V
I <sub>IL</sub>	Low Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = L			-10	μA
I <sub>IH</sub>	High Voltage Input Current (pins 5, 7, 10, 12)	V <sub>i</sub> = H ≤ V <sub>SS</sub> -0.6V		30	100	μA
V <sub>en</sub> = L	Enable Low Voltage (pins 6, 11)		-0.3		1.5	V
V <sub>en</sub> = H	Enable High Voltage (pins 6, 11)		2.3		V <sub>SS</sub>	V
I <sub>en</sub> = L	Low Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = L			-10	μA
I <sub>en</sub> = H	High Voltage Enable Current (pins 6, 11)	V <sub>en</sub> = H ≤ V <sub>SS</sub> -0.6V		30	100	μA
V <sub>CEsat</sub> (H)	Source Saturation Voltage	I <sub>L</sub> = 1A I <sub>L</sub> = 2A	0.95	1.35 2	1.7 2.7	V
V <sub>CEsat</sub> (L)	Sink Saturation Voltage	I <sub>L</sub> = 1A (5) I <sub>L</sub> = 2A (5)	0.85	1.2 1.7	1.6 2.3	V
V <sub>CEsat</sub>	Total Drop	I <sub>L</sub> = 1A (5) I <sub>L</sub> = 2A (5)	1.80		3.2 4.9	V
V <sub>sens</sub>	Sensing Voltage (pins 1, 15)		-1 (1)		2	V



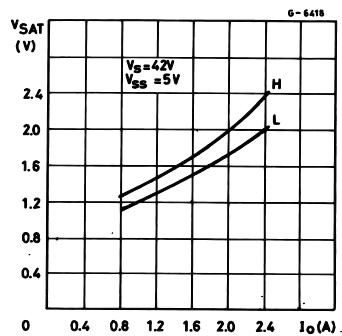
**L298**

**ELECTRICAL CHARACTERISTICS** (continued)

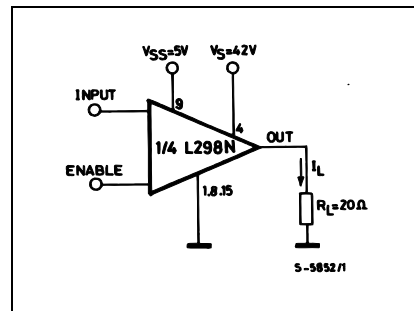
Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
T <sub>1</sub> (V <sub>i</sub> )	Source Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (2); (4)		1.5		μs
T <sub>2</sub> (V <sub>i</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		0.2		μs
T <sub>3</sub> (V <sub>i</sub> )	Source Current Turn-on Delay	0.5 V <sub>i</sub> to 0.1 I <sub>L</sub> (2); (4)		2		μs
T <sub>4</sub> (V <sub>i</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.7		μs
T <sub>5</sub> (V <sub>i</sub> )	Sink Current Turn-off Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		0.7		μs
T <sub>6</sub> (V <sub>i</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>7</sub> (V <sub>i</sub> )	Sink Current Turn-on Delay	0.5 V <sub>i</sub> to 0.9 I <sub>L</sub> (3); (4)		1.6		μs
T <sub>8</sub> (V <sub>i</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.2		μs
f <sub>c</sub> (V <sub>i</sub> )	Commutation Frequency	I <sub>L</sub> = 2A		25	40	KHz
T <sub>1</sub> (V <sub>en</sub> )	Source Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (2); (4)		3		μs
T <sub>2</sub> (V <sub>en</sub> )	Source Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (2); (4)		1		μs
T <sub>3</sub> (V <sub>en</sub> )	Source Current Turn-on Delay	0.5 V <sub>en</sub> to 0.1 I <sub>L</sub> (2); (4)		0.3		μs
T <sub>4</sub> (V <sub>en</sub> )	Source Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (2); (4)		0.4		μs
T <sub>5</sub> (V <sub>en</sub> )	Sink Current Turn-off Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		2.2		μs
T <sub>6</sub> (V <sub>en</sub> )	Sink Current Fall Time	0.9 I <sub>L</sub> to 0.1 I <sub>L</sub> (3); (4)		0.35		μs
T <sub>7</sub> (V <sub>en</sub> )	Sink Current Turn-on Delay	0.5 V <sub>en</sub> to 0.9 I <sub>L</sub> (3); (4)		0.25		μs
T <sub>8</sub> (V <sub>en</sub> )	Sink Current Rise Time	0.1 I <sub>L</sub> to 0.9 I <sub>L</sub> (3); (4)		0.1		μs

- 1) Sensing voltage can be -1 V for t ≤ 50 μsec; in steady state V<sub>sens</sub> min ≥ -0.5 V.
- 2) See fig. 2.
- 3) See fig. 4.
- 4) The load must be a pure resistor.

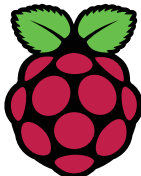
**Figure 1 :** Typical Saturation Voltage vs. Output Current.



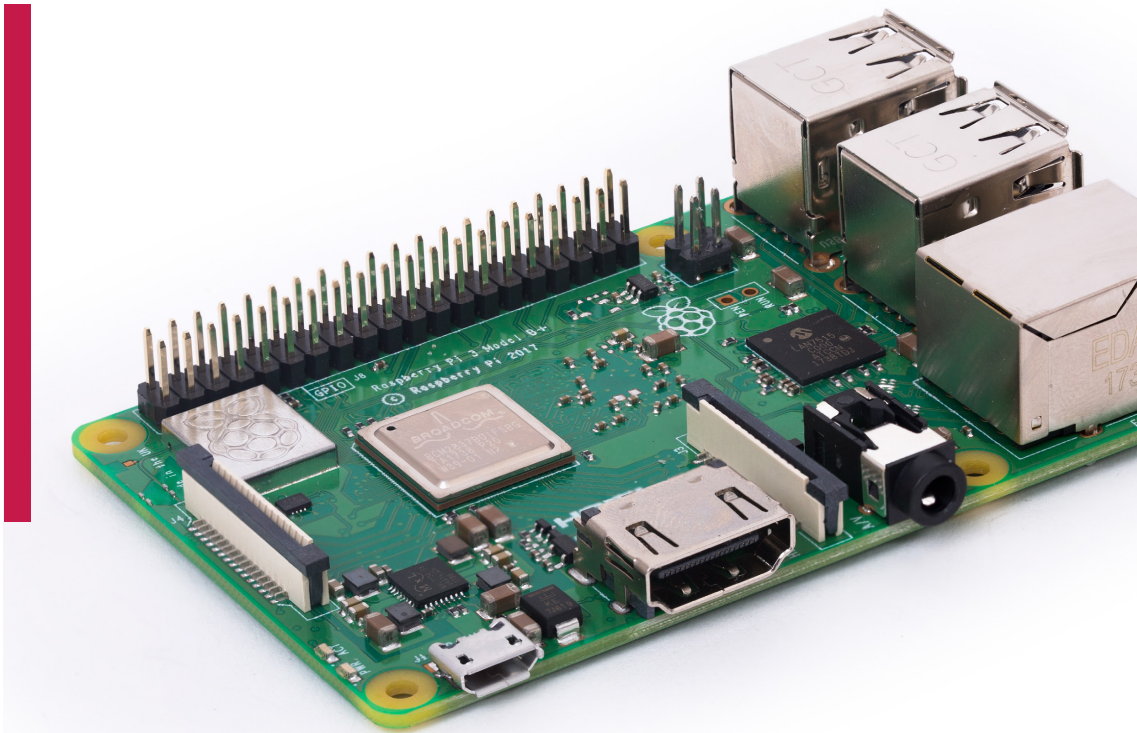
**Figure 2 :** Switching Times Test Circuits.



Note : For INPUT Switching, set EN = H  
For ENABLE Switching, set IN = H



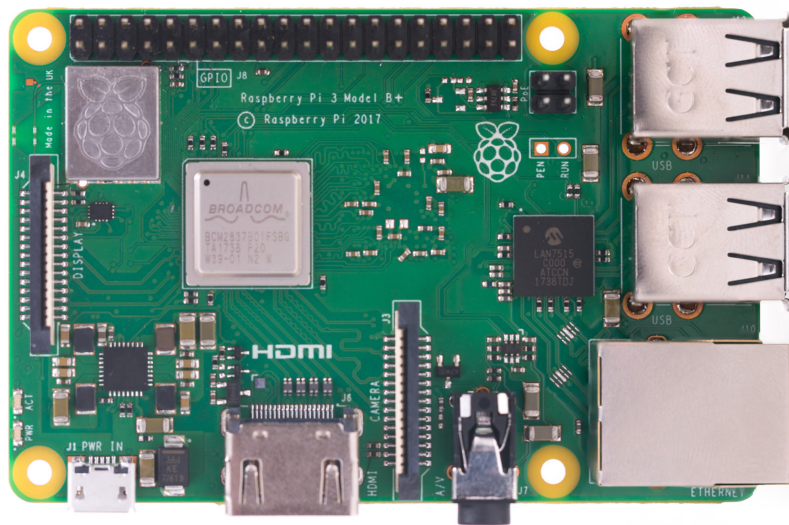
# Raspberry Pi 3 Model B+



Raspberry Pi 3 Model B+

1

## Overview



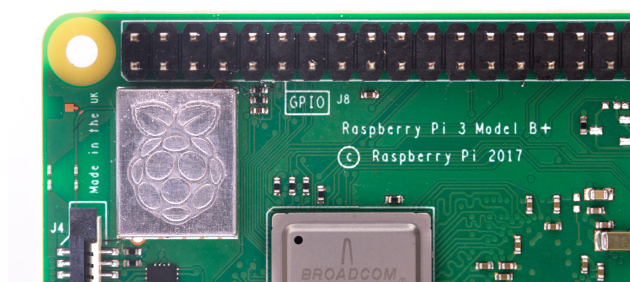
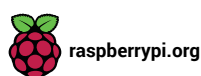
The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting a 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT

The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market.

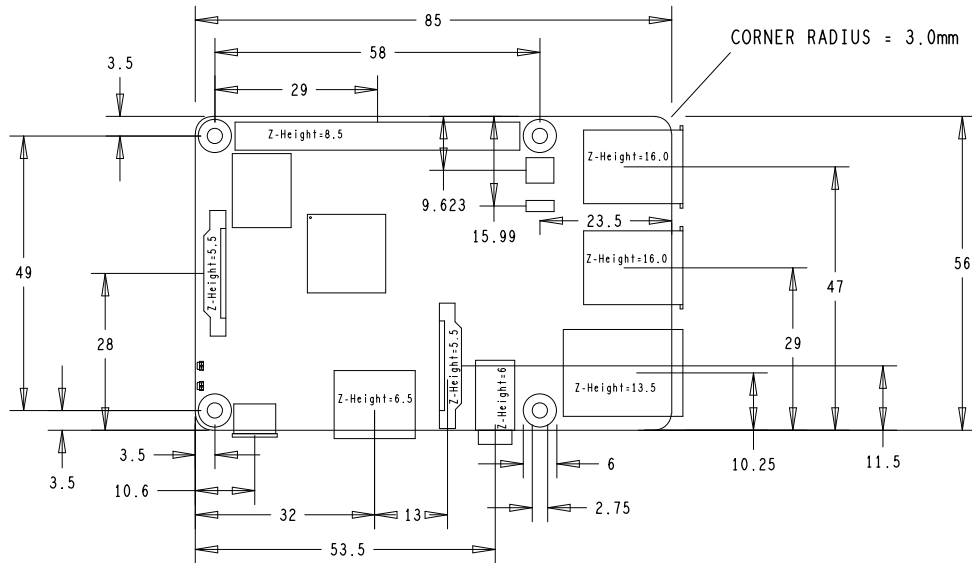
The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

## Specifications

<b>Processor:</b>	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz
<b>Memory:</b>	1GB LPDDR2 SDRAM
<b>Connectivity:</b>	<ul style="list-style-type: none"><li>■ 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE</li><li>■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)</li><li>■ 4 × USB 2.0 ports</li></ul>
<b>Access:</b>	Extended 40-pin GPIO header
<b>Video &amp; sound:</b>	<ul style="list-style-type: none"><li>■ 1 × full size HDMI</li><li>■ MIPI DSI display port</li><li>■ MIPI CSI camera port</li><li>■ 4 pole stereo output and composite video port</li></ul>
<b>Multimedia:</b>	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
<b>SD card support:</b>	Micro SD format for loading operating system and data storage
<b>Input power:</b>	<ul style="list-style-type: none"><li>■ 5V/2.5A DC via micro USB connector</li><li>■ 5V DC via GPIO header</li><li>■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)</li></ul>
<b>Environment:</b>	Operating temperature, 0–50°C
<b>Compliance:</b>	For a full list of local and regional product approvals, please visit <a href="http://www.raspberrypi.org/products/raspberry-pi-3-model-b+">www.raspberrypi.org/products/raspberry-pi-3-model-b+</a>
<b>Production lifetime:</b>	The Raspberry Pi 3 Model B+ will remain in production until at least January 2023.



## Physical specifications



### Warnings

- This product should only be connected to an external power supply rated at 5V/2.5 A DC. Any external power supply used with the Raspberry Pi 3 Model B+ shall comply with relevant regulations and standards applicable in the country of intended use.
- This product should be operated in a well-ventilated environment and, if used inside a case, the case should not be covered.
- Whilst in use, this product should be placed on a stable, flat, non-conductive surface and should not be contacted by conductive items.
- The connection of incompatible devices to the GPIO connection may affect compliance, result in damage to the unit, and invalidate the warranty.
- All peripherals used with this product should comply with relevant standards for the country of use and be marked accordingly to ensure that safety and performance requirements are met. These articles include but are not limited to keyboards, monitors, and mice when used in conjunction with the Raspberry Pi.
- The cables and connectors of all peripherals used with this product must have adequate insulation so that relevant safety requirements are met.

### Safety instructions

To avoid malfunction of or damage to this product, please observe the following:

- Do not expose to water or moisture, or place on a conductive surface whilst in operation.
- Do not expose to heat from any source; the Raspberry Pi 3 Model B+ is designed for reliable operation at normal ambient temperatures.
- Take care whilst handling to avoid mechanical or electrical damage to the printed circuit board and connectors.
- Whilst it is powered, avoid handling the printed circuit board, or only handle it by the edges to minimise the risk of electrostatic discharge damage.





**ADS1013  
ADS1014  
ADS1015**

www.ti.com

SBAS473C—MAY 2009—REVISED OCTOBER 2009

**Ultra-Small, Low-Power, 12-Bit  
Analog-to-Digital Converter with Internal Reference**

Check for Samples: [ADS1013](#) [ADS1014](#) [ADS1015](#)

**FEATURES**

- **ULTRA-SMALL QFN PACKAGE:**  
2mm × 1,5mm × 0,4mm
- **WIDE SUPPLY RANGE:** 2.0V to 5.5V
- **LOW CURRENT CONSUMPTION:**  
Continuous Mode: Only 150µA  
Single-Shot Mode: Auto Shut-Down
- **PROGRAMMABLE DATA RATE:**  
128SPS to 3.3kSPS
- **INTERNAL LOW-DRIFT VOLTAGE REFERENCE**
- **INTERNAL OSCILLATOR**
- **INTERNAL PGA**
- **I<sup>2</sup>C™ INTERFACE: Pin-Selectable Addresses**
- **FOUR SINGLE-ENDED OR TWO DIFFERENTIAL INPUTS (ADS1015)**
- **PROGRAMMABLE COMPARATOR (ADS1014 and ADS1015)**

**APPLICATIONS**

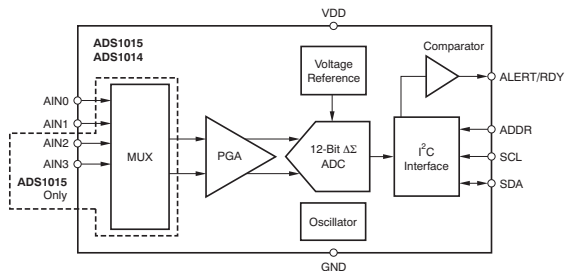
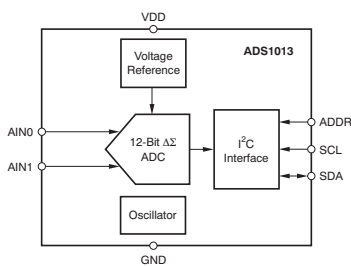
- **PORTABLE INSTRUMENTATION**
- **CONSUMER GOODS**
- **BATTERY MONITORING**
- **TEMPERATURE MEASUREMENT**
- **FACTORY AUTOMATION AND PROCESS CONTROLS**

**DESCRIPTION**

The ADS1013, ADS1014, and ADS1015 are precision analog-to-digital converters (ADCs) with 12 bits of resolution offered in an ultra-small, leadless QFN-10 package or an MSOP-10 package. The ADS1013/4/5 are designed with precision, power, and ease of implementation in mind. The ADS1013/4/5 feature an onboard reference and oscillator. Data are transferred via an I<sup>2</sup>C-compatible serial interface; four I<sup>2</sup>C slave addresses can be selected. The ADS1013/4/5 operate from a single power supply ranging from 2.0V to 5.5V.

The ADS1013/4/5 can perform conversions at rates up to 3300 samples per second (SPS). An onboard PGA is available on the ADS1014 and ADS1015 that offers input ranges from the supply to as low as ±256mV, allowing both large and small signals to be measured with high resolution. The ADS1015 also features an input multiplexer (MUX) that provides two differential or four single-ended inputs.

The ADS1013/4/5 operate either in continuous conversion mode or a single-shot mode that automatically powers down after a conversion and greatly reduces current consumption during idle periods. The ADS1013/4/5 are specified from -40°C to +125°C.



Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.  
I<sup>2</sup>C is a trademark of NXP Semiconductors.  
All other trademarks are the property of their respective owners.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of the Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2009, Texas Instruments Incorporated



**ADS1013**  
**ADS1014**  
**ADS1015**


SBAS473C –MAY 2009–REVISED OCTOBER 2009

[www.ti.com](http://www.ti.com)

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

### ORDERING INFORMATION

For the most current package and ordering information, see the Package Option Addendum at the end of this document, or see the TI web site at [www.ti.com](http://www.ti.com).

### ABSOLUTE MAXIMUM RATINGS<sup>(1)</sup>

	ADS1013, ADS1014, ADS1015	UNIT
VDD to GND	-0.3 to +5.5	V
Analog input current	100, momentary	mA
Analog input current	10, continuous	mA
Analog input voltage to GND	-0.3 to VDD + 0.3	V
SDA, SCL, ADDR, ALERT/RDY voltage to GND	-0.5 to +5.5	V
Maximum junction temperature	+150	°C
Storage temperature range	-60 to +150	°C

(1) Stresses above those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. Exposure to absolute maximum conditions for extended periods may affect device reliability.

### PRODUCT FAMILY

DEVICE	PACKAGE DESIGNATOR MSOP/QFN	RESOLUTION (Bits)	MAXIMUM SAMPLE RATE (SPS)	COMPARATOR	PGA	INPUT CHANNELS (Differential/Single-Ended)
ADS1113	BR0I/N6J	16	860	No	No	1/1
ADS1114	BRNI/N5J	16	860	Yes	Yes	1/1
ADS1115	BOGI/N4J	16	860	Yes	Yes	2/4
ADS1013	BRMI/N9J	12	3300	No	No	1/1
ADS1014	BRQI/N8J	12	3300	Yes	Yes	1/1
ADS1015	BRPI/N7J	12	3300	Yes	Yes	2/4



**ADS1013  
ADS1014  
ADS1015**

www.ti.com

SBAS473C–MAY 2009–REVISED OCTOBER 2009

**ELECTRICAL CHARACTERISTICS**

All specifications at –40°C to +125°C, VDD = 3.3V, and Full-Scale (FS) = ±2.048V, unless otherwise noted. Typical values are at +25°C.

PARAMETER	TEST CONDITIONS	ADS1013, ADS1014, ADS1015			UNIT
		MIN	TYP	MAX	
<b>ANALOG INPUT</b>					
Full-scale input voltage <sup>(1)</sup>	$V_{IN} = (AIN_P) - (AIN_N)$		±4.096/PGA		V
Analog input voltage	$AIN_P$ or $AIN_N$ to GND	GND		VDD	V
Differential input impedance			See Table 2		
Common-mode input impedance	$FS = \pm 6.144V^{(1)}$		10		MΩ
	$FS = \pm 4.096V^{(1)}, \pm 2.048V$		6		MΩ
	$FS = \pm 1.024V$		3		MΩ
	$FS = \pm 0.512V, \pm 0.256V$		100		MΩ
<b>SYSTEM PERFORMANCE</b>					
Resolution	No missing codes	12			Bits
Data rate (DR)			128, 250, 490, 920, 1600, 2400, 3300		SPS
Data rate variation	All data rates	–10		10	%
Output noise		See Typical Characteristics			
Integral nonlinearity	$DR = 128SPS, FS = \pm 2.048V$ , best fit <sup>(2)</sup>			0.5	LSB
Offset error	$FS = \pm 2.048V$ , differential inputs		0	±0.5	LSB
	$FS = \pm 2.048V$ , single-ended inputs		±0.25		LSB
Offset drift	$FS = \pm 2.048V$		0.005		LSB/°C
Gain error <sup>(3)</sup>	$FS = \pm 2.048V$ at 25°C		0.05	0.25	%
Gain drift <sup>(3)</sup>	$FS = \pm 0.256V$		7		ppm/°C
	$FS = \pm 2.048V$		5	40	ppm/°C
	$FS = \pm 6.144V^{(1)}$		5		ppm/°C
PGA gain match <sup>(3)</sup>	Match between any two PGA gains		0.02	0.1	%
Gain match	Match between any two inputs		0.05	0.1	%
Offset match	Match between any two inputs		0.25		LSB
<b>DIGITAL INPUT/OUTPUT</b>					
Logic level					
$V_{IH}$		0.7VDD		5.5	V
$V_{IL}$		GND – 0.5		0.3VDD	V
$V_{OL}$	$I_{OL} = 3mA$	GND	0.15	0.4	V
Input leakage					
$I_H$	$V_{IH} = 5.5V$			10	μA
$I_L$	$V_{IL} = GND$	10			μA

(1) This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.  
 (2) 99% of full-scale.  
 (3) Includes all errors from onboard PGA and reference.

**ADS1013  
ADS1014  
ADS1015**



SBAS473C –MAY 2009–REVISED OCTOBER 2009

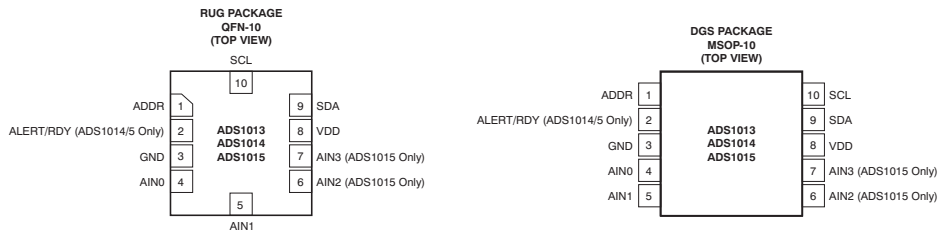
www.ti.com

**ELECTRICAL CHARACTERISTICS (continued)**

All specifications at –40°C to +125°C, VDD = 3.3V, and Full-Scale (FS) = ±2.048V, unless otherwise noted. Typical values are at +25°C.

PARAMETER	TEST CONDITIONS	ADS1013, ADS1014, ADS1015			UNIT
		MIN	TYP	MAX	
<b>POWER-SUPPLY REQUIREMENTS</b>					
Power-supply voltage		2		5.5	V
Supply current	Power-down current at 25°C		0.5	2	µA
	Power-down current up to 125°C			5	µA
	Operating current at 25°C		150	200	µA
	Operating current up to 125°C			300	µA
Power dissipation	VDD = 5.0V		0.9		mW
	VDD = 3.3V		0.5		mW
	VDD = 2.0V		0.3		mW
<b>TEMPERATURE</b>					
Storage temperature		–60		+150	°C
Specified temperature		–40		+125	°C

**PIN CONFIGURATIONS**



**PIN DESCRIPTIONS**

PIN #	DEVICE			FUNCTION	DESCRIPTION
	ADS1013	ADS1014	ADS1015		
1	ADDR	ADDR	ADDR	Digital input	I <sup>2</sup> C slave address select
2	NC <sup>(1)</sup>	ALERT/RDY	ALERT/RDY	Digital output	Digital comparator output or conversion ready (NC for ADS1013)
3	GND	GND	GND	Supply	Ground
4	AIN0	AIN0	AIN0	Analog input	Differential channel 1: Positive input or single-ended channel 1 input
5	AIN1	AIN1	AIN1	Analog input	Differential channel 1: Negative input or single-ended channel 2 input
6	NC	NC	AIN2	Analog input	Differential channel 2: Positive input or single-ended channel 3 input (NC for ADS1013/4)
7	NC	NC	AIN3	Analog input	Differential channel 2: Negative input or single-ended channel 4 input (NC for ADS1013/4)
8	VDD	VDD	VDD	Supply	Power supply: 2.0V to 5.5V
9	SDA	SDA	SDA	Digital I/O	Serial data: Transmits and receives data
10	SCL	SCL	SCL	Digital input	Serial clock input: Clocks data on SDA

(1) NC pins may be left floating or tied to ground.



www.ti.com

SBAS473C –MAY 2009–REVISED OCTOBER 2009

ADS1013  
ADS1014  
ADS1015

When reading from the ADS1013/4/5, the previous value written to the Pointer register determines the register that is read from. To change which register is read, a new value must be written to the Pointer register. To write a new value to the Pointer register, the master issues a slave address byte with the R/W bit low, followed by the Pointer register byte. No additional data need to be transmitted, and a STOP condition can be issued by the master. The master may now issue a START condition and send the slave address byte with the R/W bit high to begin the read. Figure 16 details this sequence. If repeated reads from the same register are desired, there is no need to continually send Pointer register bytes, because the ADS1013/4/5 store the value of the Pointer register until it is modified by a write operation. However, every write operation requires the Pointer register to be written.

**REGISTERS**

The ADS1013/4/5 have four registers that are accessible via the I<sup>2</sup>C port. The Conversion register contains the result of the last conversion. The Config register allows the user to change the ADS1013/4/5 operating modes and query the status of the devices. Two registers, Lo\_thresh and Hi\_thresh, set the threshold values used for the comparator function.

**POINTER REGISTER**

The four registers are accessed by writing to the Pointer register byte; see Figure 16. Table 6 and Table 7 indicate the Pointer register byte map.

**Table 6. Register Address**

BIT 1	BIT 0	REGISTER
0	0	Conversion register
0	1	Config register
1	0	Lo_thresh register
1	1	Hi_thresh register

**CONVERSION REGISTER**

The 16-bit register contains the result of the last conversion in binary two's complement format. Following reset or power-up, the Conversion register is cleared to '0', and remains '0' until the first conversion is completed.

The register format is shown in Table 8.

**CONFIG REGISTER**

The 16-bit register can be used to control the ADS1013/4/5 operating mode, input selection, data rate, PGA settings, and comparator modes. The register format is shown in Table 9.

**Table 7. Pointer Register Byte (Write-Only)**

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	0	0	0	0	0	Register address	

**Table 8. Conversion Register (Read-Only)**

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

**Table 9. Config Register (Read/Write)**

BIT	15	14	13	12	11	10	9	8
NAME	OS	MUX2	MUX1	MUX0	PGA2	PGA1	PGA0	MODE

BIT	7	6	5	4	3	2	1	0
NAME	DR2	DR1	DR0	COMP_MODE	COMP_POL	COMP_LAT	COMP_QUE1	COMP_QUE0

Default = 8583h.

Bit [15]

**OS: Operational status/single-shot conversion start**

This bit determines the operational status of the device. This bit can only be written when in power-down mode.

For a write status:

- 0 : No effect
- 1 : Begin a single conversion (when in power-down mode)

For a read status:

- 0 : Device is currently performing a conversion
- 1 : Device is not currently performing a conversion

**ADS1013**  
**ADS1014**  
**ADS1015**


SBAS473C –MAY 2009–REVISED OCTOBER 2009

www.ti.com

<b>Bits [14:12]</b>	<b>MUX[2:0]: Input multiplexer configuration (ADS1015 only)</b> These bits configure the input multiplexer. They serve no function on the ADS1013/4. 000 : AIN <sub>P</sub> = AIN <sub>0</sub> and AIN <sub>N</sub> = AIN <sub>1</sub> (default)      100 : AIN <sub>P</sub> = AIN <sub>0</sub> and AIN <sub>N</sub> = GND 001 : AIN <sub>P</sub> = AIN <sub>0</sub> and AIN <sub>N</sub> = AIN <sub>3</sub> 101 : AIN <sub>P</sub> = AIN <sub>1</sub> and AIN <sub>N</sub> = GND 010 : AIN <sub>P</sub> = AIN <sub>1</sub> and AIN <sub>N</sub> = AIN <sub>3</sub> 110 : AIN <sub>P</sub> = AIN <sub>2</sub> and AIN <sub>N</sub> = GND 011 : AIN <sub>P</sub> = AIN <sub>2</sub> and AIN <sub>N</sub> = AIN <sub>3</sub> 111 : AIN <sub>P</sub> = AIN <sub>3</sub> and AIN <sub>N</sub> = GND
<b>Bits [11:9]</b>	<b>PGA[2:0]: Programmable gain amplifier configuration (ADS1014 and ADS1015 only)</b> These bits configure the programmable gain amplifier. They serve no function on the ADS1013. 000 : FS = ±6.144V <sup>(1)</sup> 100 : FS = ±0.512V 001 : FS = ±4.096V <sup>(1)</sup> 101 : FS = ±0.256V 010 : FS = ±2.048V (default)                            110 : FS = ±0.256V 011 : FS = ±1.024V                                         111 : FS = ±0.256V
<b>Bit [8]</b>	<b>MODE: Device operating mode</b> This bit controls the current operational mode of the ADS1013/4/5. 0 : Continuous conversion mode 1 : Power-down single-shot mode (default)
<b>Bits [7:5]</b>	<b>DR[2:0]: Data rate</b> These bits control the data rate setting. 000 : 128SPS    100 : 1600SPS (default) 001 : 250SPS     101 : 2400SPS 010 : 490SPS     110 : 3300SPS 011 : 920SPS     111 : 3300SPS
<b>Bit [4]</b>	<b>COMP_MODE: Comparator mode (ADS1014 and ADS1015 only)</b> This bit controls the comparator mode of operation. It changes whether the comparator is implemented as a traditional comparator (COMP_MODE = '0') or as a window comparator (COMP_MODE = '1'). It serves no function on the ADS1013. 0 : Traditional comparator with hysteresis (default) 1 : Window comparator
<b>Bit [3]</b>	<b>COMP_POL: Comparator polarity (ADS1014 and ADS1015 only)</b> This bit controls the polarity of the ALERT/RDY pin. When COMP_POL = '0' the comparator output is active low. When COMP_POL = '1' the ALERT/RDY pin is active high. It serves no function on the ADS1013. 0 : Active low (default) 1 : Active high
<b>Bit [2]</b>	<b>COMP_LAT: Latching comparator (ADS1014 and ADS1015 only)</b> This bit controls whether the ALERT/RDY pin latches once asserted or clears once conversions are within the margin of the upper and lower threshold values. When COMP_LAT = '0', the ALERT/RDY pin does not latch when asserted. When COMP_LAT = '1', the asserted ALERT/RDY pin remains latched until conversion data are read by the master or an appropriate SMBus alert response is sent by the master, the device responds with its address, and it is the lowest address currently asserting the ALERT/RDY bus line. This bit serves no function on the ADS1013. 0 : Non-latching comparator (default) 1 : Latching comparator
<b>Bits [1:0]</b>	<b>COMP_QUE: Comparator queue and disable (ADS1014 and ADS1015 only)</b> These bits perform two functions. When set to '11', they disable the comparator function and put the ALERT/RDY pin into a high state. When set to any other value, they control the number of successive conversions exceeding the upper or lower thresholds required before asserting the ALERT/RDY pin. They serve no function on the ADS1013. 00 : Assert after one conversion 01 : Assert after two conversions 10 : Assert after four conversions 11 : Disable comparator (default)

(1) This parameter expresses the full-scale range of the ADC scaling. In no event should more than VDD + 0.3V be applied to this device.

# B SOFTWARE CODE

---

The final code is appended in the next pages. The programs are the following.

1. Server in C
2. Client in the rotor in C
3. Client in the hub in C
4. Server in Python (not implemented)
5. Client in the rotor in Python (not implemented)
6. Client in the hub in Python (not implemented)

---

```

1 #include <stdio.h>
2 #include <time.h>
3 #include <stdlib.h>
4 #include <pthread.h>
5 #include <unistd.h>
6 #include <string.h>
7 #include <sys/socket.h>
8 #include <netinet/in.h>
9 #include <sys/time.h>
10 #include <arpa/inet.h>
11 #define PORT 8080
12
13 ////////////////////////////////////////////////////GLOBAL VARIABLES
14 int flag=1;
15 ////////////////////////////////////////////////////FUNCTIONS
16 void *UserGUI(void *vargp)
17 {
18     char command[1];
19     flag=1;
20     while (flag==1){
21         scanf("%s",command);
22         if (strcmp(command,"s")==0){
23             flag=0;}}
24
25     printf("Thread interface closing everything \n");
26     return NULL;
27 }
28
29 //////////////////////////////////////////////////// MAIN
30 int main() {
31
32 //////////////////////////////////////////////////// VARIABLES
33 struct sockaddr_in address;
34 long long i,t_delay=9e7;
35 int k,j,n_samples,n_data_t,n_data_r,n_data_s,SIZE_DATA,ml,server_fd, socket_r,socket_s,
    valread,opt,addrlen;
36 i=k=0; j=1;n_data_t=8;n_data_r=4;n_data_s=3; SIZE_DATA=n_data_t*1000;opt=1;addrlen=
    sizeof(address);
37 int data_rec_r[4]={0,0,0,0};int data_send_r[8]={0,0,0,0,0,0,0,0};int close_vector_r
    [8]={-1,-1,-1,-1,-1,-1,-1,-1};
38 int data_rec_s[3]={0,0,0};int data_send_s[4]={0,0,0,0}; int close_vector_s[4]={-1,-1,-1,-1};
39 double mean_f,cpu_time_used,t_s,tnext,id;
40 mean_f=t_s=cpu_time_used=id=tnext=0.0;
41 unsigned long long t_s_micro=0;
42 unsigned m1r,m1l,m2r,m2l,m3r,m3l,m4r,m4l; //pwm
43 struct timeval t0,t1,t2,tf;
44
45
46 // thread creation
47 pthread_t thread_id;
48 pthread_create(&thread_id, NULL, UserGUI, NULL);
49
50 // dynamic array creatiom
51 double* d_array;
52 d_array =calloc(SIZE_DATA*j,sizeof(double));

```

## B. Software code

---

```
53 if(d_array == NULL) {
54     printf("malloc of d_array failed!\n");
55     exit(1);}
56 // data file creation
57 FILE *f = fopen("data.txt", "w");
58 if (f == NULL){
59     printf("Error opening file!\n");
60     exit(1);}
61
62 //server creation
63 server_fd = socket(AF_INET, SOCK_STREAM,0);
64 setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt, sizeof(opt));
65 setsockopt(server_fd, IPPROTO_TCP, TCP_NODELAY, (char *) &flag,sizeof(int));
66 address.sin_family = AF_INET;
67 address.sin_addr.s_addr =INADDR_ANY; //ADDRESS of the host here
68 address.sin_port = htons( PORT );    // Port defined up in the beggining 8080
69
70
71
72 //////////////////////////////////////////////////// PROGRAM
73
74 ////////////////////////////////////////////////////BIND THE SERVER
75 bind(server_fd, (struct sockaddr *)&address, sizeof(address));
76 printf("%s\n","binded" );
77 listen (server_fd, 3);
78 //////////////////////////////////////////////////// CONNECTION WITH CLIENT
79 socket_s = accept(server_fd, (struct sockaddr *)&address,(socklen_t*)&addrlen);
80 socket_r = accept(server_fd, (struct sockaddr *)&address,(socklen_t*)&addrlen);
81 if (socket_r>=0){printf("%s\n","connected to the stator" );}
82 if (socket_s>=0){printf("%s\n","connected to the rotor" );}
83
84 data_send_r[0]=0;data_send_r[1]=1000;data_send_r[2]=500;data_send_r[3]=1500;data_send_r
85     [4]=250;data_send_r[5]=1250;data_send_r[6]=750;data_send_r[7]=1750;
86 data_send_s[0]=0;data_send_s[1]=1000;data_send_s[2]=500;data_send_s[3]=1500;
87
88 ////////////////////////////////////////////////////LOOP
89 gettimeofday(&t0, NULL);
90 while( flag==1) {
91
92     i++;
93     ////TIME APPENDING
94     gettimeofday(&t1, NULL);
95     d_array[i] = t_s;
96     //////////////////////////////////////////////////// RECEIVING DATA FROM CLIENTS
97     valread = read(socket_s , data_rec_s, 40);
98     valread = read(socket_r , data_rec_r, 40);
99     ////////////////////////////////////////////////////DATA APPENDING
100    for (int b = 0; b < n_data_t-1; b++)
101    {
102        i++;
103        if (b<n_data_r){d_array[i] = data_rec_r[b];}
104        if (b>=n_data_r){d_array[i]=data_rec_s[b-4];}
105    }
106    k++;
107    ////////////////////////////////////////////////////ENLARGING THE DATA VECTOR FOR SAVING MORE DATA
108
```



```

109  if (k+1==SIZE_DATA/n_data_t){
110      j++;
111      d_array= realloc(d_array, (SIZE_DATA*j) * sizeof(double));
112      k=0;}
113
114  //////////////// CALCULATE ACTUATION
115  //////////////// IMPLEMENT HERE ANY CONTROL ALGORITHM
116  for (int mag = 0; mag < 8; mag++)
117  {
118
119      data_send_r[mag]=data_send_r[mag]+1;
120
121      if (data_send_r[mag]>2000)
122      {
123          data_send_r[mag]=0;
124      }}
125  for (int mag = 0; mag < 4; mag++)
126  {
127
128      data_send_s[mag]=data_send_s[mag]+1;
129
130      if (data_send_s[mag]>2000)
131      {
132          data_send_s[mag]=0;
133      }}
134
135
136  ////////////////SENDING ACTUATION
137  send(socket_s , data_send_s , sizeof(data_send_s) , 0 );
138  send(socket_r , data_send_r , sizeof(data_send_r) , 0 );
139  ////////////////TIME CALCULATION
140  gettimeofday(&t2, NULL);
141  t_s=(t2.tv_sec - t1.tv_sec) + (t2.tv_usec - t1.tv_usec) / 1000000.0f;
142  tnext=tnext+t_s;
143  }
144
145  ///// END OF LOOP
146  gettimeofday(&tf, NULL);
147
148  //////////////// END CLIENTS
149  valread = read( socket_s , data_rec_s, 1024);
150  valread = read( socket_r , data_rec_s, 1024);
151  send(socket_s , close_vector_s , sizeof(close_vector_s) , 0 );
152  send(socket_r , close_vector_r , sizeof(close_vector_r) , 0 );
153  /////// TIME CALCULATION
154  cpu_time_used = (tf.tv_sec - t0.tv_sec) + (tf.tv_usec - t0.tv_usec) / 1000000.0f;
155  n_samples=i/n_data_t;
156  mean_f=n_samples/cpu_time_used;
157  printf("done in %f \n",cpu_time_used);
158  printf("mean frequency = %f\n",mean_f);
159  printf("mean time_step = %f\n",1/mean_f);
160
161  ////////// DATA PROCESSING
162
163
164  printf("Data clection ended with %d samples\n",n_samples );
165  ml=0;

```

## B. Software code

---

```
166 for(int m = 0; m < n_samples; m++) {
167   for (int l = 0; l < n_data_t; l++)
168     { ml++;
169       fprintf (f, "%f",d_array[ml]);
170       if (l!=n_data_t-1){fprintf(f, ", ");}
171     }
172   fprintf (f, "\n");}
173 //////////////////////////////////////////////////// CLOSING EVERYTHING
174 pthread_join(thread_id, NULL);
175 free (d_array);
176 fclose (f);
177 printf ("Program ended");
178 return(0);
179 }
```

---

```
1 /// THIS CODE IS IMPLEMENTED IN THE ROTOR: IT INCLUDES
2 /// SENSING THE 4 DISPLACEMENTS OF THE BLADES
3 /// ACTUATING IN 8 MAGNETS OF THE BLADES
4
5 #include <stdio.h>
6 #include <time.h>
7 #include <stdlib.h>
8 #include <pthread.h>
9 #include <unistd.h>
10 #include <string.h>
11 #include <sys/socket.h>
12 #include <netinet/in.h>
13 #include <sys/time.h>
14 #include <arpa/inet.h>
15 #include <wiringPiI2C.h>
16 #include <wiringPi.h>
17 #include <pigpio.h>
18 #define PORT 8080
19 #define DevAddr 0x48
20 #define B_convert 0x8000
21 #define B_null 0x0000
22 #define Mux_com_1_0 0x0000
23 #define Mux_com_3_0 0x1000
24 #define Mux_com_3_1 0x2000
25 #define Mux_com_3_2 0x3000
26 #define Mux_com_0 0x4000
27 #define Mux_com_1 0x5000
28 #define Mux_com_2 0x6000
29 #define Mux_com_3 0x7000
30 #define Amp_3_2 0x0000
31 #define Amp_1 0x0200
32 #define Amp_1_2 0x0400
33 #define Amp_1_4 0x0600
34 #define Amp_1_8 0x0800
35 #define Amp_1_16 0x0a00
36 #define B_cont 0x0000
37 #define B_sing 0x0100
38 #define DR_128 0x0000
39 #define DR_250 0x0020
40 #define DR_490 0x0040
41 #define DR_920 0x0060
```

```

42 #define DR_1600 0x0080
43 #define DR_2400 0x00a0
44 #define DR_3300 0x00c0
45 #define COMP_MODE_T 0x0000
46 #define COMP_MODE_W 0x0001
47 #define COM_POL_L 0x0000
48 #define COM_POL_H 0x0008
49 #define COMP_LN 0x0000
50 #define COMP_LA 0x0004
51 #define COMP_QUE_1 0x0000
52 #define COMP_QUE_2 0x0001
53 #define COMP_QUE_3 0x0002
54 #define COMP_QUE_N 0x0003
55 //////////////////////////////////////////////////GLOBAL VARIABLES
56 int DATA_SEND[4]={0,0,0,0};
57 int flag=1;
58
59
60 //////////////////////////////////////////////////FUNCTIONS
61
62 int Byte_swapper(int data16)
63 {
64     int data,data1,data2;
65     data1=(data16<<8)&0xff00;
66     data2=data16>>8;
67     data=data1 | data2;
68     return data;
69 }
70 int Config_Channel(int Config_16num,int Channel)
71 {
72     switch(Channel)
73     {
74     case 1:
75         Config_16num=Config_16num|B_convert|Mux_com_0;
76         break;
77     case 2:
78         Config_16num=Config_16num|B_convert|Mux_com_1;
79         break;
80     case 3:
81         Config_16num=Config_16num|B_convert|Mux_com_2;
82         break;
83     case 4:
84         Config_16num=Config_16num|B_convert|Mux_com_3;
85         break;
86     default :
87         printf("%s\n%s\n","Error: Not a valid Channel. ","Choose between channel 1 and 4");
88     }
89
90     return Config_16num;
91 }
92 int Config_DR(int Config_16num,int DR)
93 {
94     switch(DR)
95     {
96     case 128:
97         Config_16num=Config_16num|DR_128;
98         break;

```

## B. Software code

---

```
99     case 250:
100         Config_16num=Config_16num|DR_250;
101         break;
102     case 490:
103         Config_16num=Config_16num|DR_490;
104         break;
105     case 920:
106         Config_16num=Config_16num|DR_920;
107         break;
108     case 1600:
109         Config_16num=Config_16num|DR_1600;
110         break;
111     case 2400:
112         Config_16num=Config_16num|DR_2400;
113         break;
114     case 3300:
115         Config_16num=Config_16num|DR_3300;
116         break;
117     default :
118         printf ("%s\n%s\n", "Error: Not a valid Data Rate. ", "Choose between possible Data Rates: 128,
119             250, 490, 920, 1600, 2400, 3300");
120     }
121     return Config_16num;
122 }
123 int Config_Gain(int Config_16num,int gain)
124 {
125     switch(gain)
126     {
127     case 1:
128         Config_16num=Config_16num|Amp_1;
129         break;
130     case 2:
131         Config_16num=Config_16num|Amp_1_2;
132         break;
133     case 4:
134         Config_16num=Config_16num|Amp_1_4;
135         break;
136     case 8:
137         Config_16num=Config_16num|Amp_1_8;
138         break;
139     case 16:
140         Config_16num=Config_16num|Amp_1_16;
141         break;
142     default :
143         printf ("%s\n%s\n", "Error: Not a valid Gain. ", "Choose between possible gains: 1, 2, 4, 8, 16");
144     }
145     return Config_16num;
146 }
147 int Config_Mode(int Config_16num, int mode)
148 {
149     switch (mode)
150     {
151     case(0):
152         Config_16num=Config_16num|B_sing;
153         break;
154     case(1):
155         Config_16num=Config_16num|B_cont;
```

```

155     break;
156 default:
157     printf("%s\n%s\n", "Error: Not a valid Mode. ", "Choose between Continuous (1) or Single-Shot
(0)");
158     }
159 return Config_16num;
160 }
161 int Config_Default(int Config_16num)
162 {
163     Config_16num=Config_16num | COMP_MODE_T;
164     Config_16num=Config_16num | COM_POL_L;
165     Config_16num=Config_16num | COMP_LN;
166     Config_16num=Config_16num | COMP_QUE_N;
167     return Config_16num;
168 }
169 //////////////////////////////////////////////////// MAIN
170 int main() {
171     int data, Init , fd, reg, Config1, Config2, Config3, Config4; int CONFIG_V[4]={0,0,0,0};
172     struct timeval t1, t2, t0, tf;
173     unsigned magnet1r, magnet1l, magnet2r, magnet2l, magnet3r, magnet3l, magnet4r, magnet4l;
174     magnet1r=14; magnet1l=15; magnet2r=17; magnet2l=27; magnet3r=23; magnet3l=24; magnet4r=10;
magnet4l=9;
175     struct sockaddr_in address, serv_addr;
176     int i, k, j, n_samples, n_data, SIZE_DATA, ml, server_fd, new_socket, valread, sock, addrlen,
nano_to_micro;
177     i=k=0; j=1; SIZE_DATA=3000; n_data=3; sock=0; nano_to_micro=1e3; addrlen=sizeof(address);
178     int data_send[4]={0,0,0,0}; int data_rec [8]={0,0,0,0,0,0,0,0}; int close_vector
[8]={-1,-1,-1,-1,-1,-1,-1,-1};
179     double mean_f, cpu_time_used, t_s, id;
180     mean_f=t_s=cpu_time_used=id=0.0;
181     long long t_delay=1e2*nano_to_micro;
182     ////////////////////////////////////////////////////SOCKET CONFIGURATION FOR WIRELESS CONNECTION
183     //////////////////////////////////////////////////// CLIENT
184     sock = socket(AF_INET, SOCK_STREAM, 0);
185     memset(&serv_addr, '0', sizeof(serv_addr));
186     serv_addr.sin_family = AF_INET;
187     serv_addr.sin_port = htons(PORT);
188     inet_pton(AF_INET, "10.16.184.180", &serv_addr.sin_addr);
189     // PROGRAM STARTS HERE
190     //////////////////////////////////////////////////// TRY TO CONNECT TO THE LINUX CPU OR SERVER
191     connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
192
193     //////////////////////////////////////////////////// INITIALISE THE I2C PROTOCOL LIBRARY
194     fd= wiringPiI2CSetup (DevAddr);
195     //////////////////////////////////////////////////// INITIALISE THE PWM BY DMA SOFTWARE
196     if ( gpioInitialise () < 0)
197     {
198         printf("pigpio initialisation failed.\n");
199     }
200     //////////////////////////////////////////////////// SET THE PWM PARAMETERS
201     gpioSetPWMrange(magnet1r,2000);
202     gpioSetPWMrange(magnet1l,2000);
203     gpioSetPWMrange(magnet2r,2000);
204     gpioSetPWMrange(magnet2l,2000);
205     gpioSetPWMrange(magnet3r,2000);
206     gpioSetPWMrange(magnet3l,2000);
207     gpioSetPWMrange(magnet4r,2000);

```

## B. Software code

---

```
208  gpioSetPWMrange(magnet4l,2000);
209  gpioSetPWMfrequency(magnet1r,8000);
210  gpioSetPWMfrequency(magnet1l,8000);
211  gpioSetPWMfrequency(magnet2r,8000);
212  gpioSetPWMfrequency(magnet2l,8000);
213  gpioSetPWMfrequency(magnet3r,8000);
214  gpioSetPWMfrequency(magnet3l,8000);
215  gpioSetPWMfrequency(magnet4r,8000);
216  gpioSetPWMfrequency(magnet4l,8000);
217
218  gpioPWM(magnet1r, 0);
219  gpioPWM(magnet1l, 0);
220  gpioPWM(magnet2r, 0);
221  gpioPWM(magnet2l, 0);
222  gpioPWM(magnet3r, 0);
223  gpioPWM(magnet3l, 0);
224  gpioPWM(magnet4r, 0);
225  gpioPWM(magnet4l, 0);
226  //////////////////////////////////// PERSONALIZED CODE FOR THE AD CONVERTER
227  ////////////////////////////////////WRITNG REGISTER CONFIGURATION
228  Config1=0;
229  Config1=Config_Default(Config1);
230  Config1=Config_Channel(Config1,1);//Channel 1
231  Config1=Config_DR(Config1,3300);
232  Config1=Config_Gain(Config1,1);
233  Config1=Config_Mode(Config1,0);
234  Config1=Byte_swapper(Config1);
235  CONFIG_V[0]=Config1;
236
237  Config2=0;
238  Config2=Config_Default(Config2);
239  Config2=Config_Channel(Config2,2);//Channel 2
240  Config2=Config_DR(Config2,3300);
241  Config2=Config_Gain(Config2,1);
242  Config2=Config_Mode(Config2,0);
243  Config2=Byte_swapper(Config2);
244  CONFIG_V[1]=Config2;
245
246  Config3=0;
247  Config3=Config_Default(Config3);
248  Config3=Config_Channel(Config3,3);//Channel 3
249  Config3=Config_DR(Config3,3300);
250  Config3=Config_Gain(Config3,1);
251  Config3=Config_Mode(Config3,0);
252  Config3=Byte_swapper(Config3);
253  CONFIG_V[2]=Config3;
254
255  Config4=0;
256  Config4=Config_Default(Config4);
257  Config4=Config_Channel(Config4,4);//Channel 4
258  Config4=Config_DR(Config4,3300);
259  Config4=Config_Gain(Config4,1);
260  Config4=Config_Mode(Config4,0);
261  Config4=Byte_swapper(Config4);
262  CONFIG_V[3]=Config4;
263
264
```

```

265
266     //////////////////////////////////////////////////// START THE LOOP
267     while(flag==1)
268     {
269     //////////////////////////////////////////////////// SEND THE SENSOR DATA
270         send(sock , DATA_SEND , sizeof(DATA_SEND) , 0 );
271     //////////////////////////////////////////////////// WAIT FOR THE SERVER CONTROL SIGNAL
272         valread = read(sock , data_rec, 40);
273     //////////////////////////////////////////////////// EXIT THE PROGRAM IF THE SERVER ASKS FOR IT
274         if (data_rec[0]==close_vector[0])
275             { flag=0;
276               printf( "%s\n","EXIT");
277             }
278         else {
279             //////////////////////////////////////////////////// IMPLEMENT THE CONTROL SIGNAL INTO THE PWM
280             gpioPWM(magnet1r, data_rec[0]);
281             gpioPWM(magnet1l, data_rec[1]);
282             gpioPWM(magnet2r, data_rec[2]);
283             gpioPWM(magnet2l, data_rec[3]);
284             gpioPWM(magnet3r, data_rec[4]);
285             gpioPWM(magnet3l, data_rec[5]);
286             gpioPWM(magnet4r, data_rec[6]);
287             gpioPWM(magnet4l, data_rec[7]);}
288         //////////////////////////////////////////////////// READ THE AD CONVERTER DATA
289         for (int j = 0; j < 4; j++)
290             {
291             wiringPiI2CWriteReg16(fd,0x01,CONFIG_V[j]);
292             nanosleep((const struct timespec []){{0, t_delay}}, NULL);
293             data= wiringPiI2CReadReg16(fd,0x00) ;
294             data=Byte_swapper(data);
295             DATA_SEND[j]=data>>4;
296             }
297         }
298
299     //////////////////////////////////////////////////// CLOSE ALL PROGRAM
300     printf( "%s\n","exit while" );
301     gpioPWM(magnet1r, 0);
302     gpioPWM(magnet1l, 0);
303     gpioPWM(magnet2r, 0);
304     gpioPWM(magnet2l, 0);
305     gpioPWM(magnet3r, 0);
306     gpioPWM(magnet3l, 0);
307     gpioPWM(magnet4r, 0);
308     gpioPWM(magnet4l, 0);
309
310     return(0);
311 }

```

---

```

1 ///   THIS CODE IS IMPLEMENTED IN THE STATOR: IT INCLUDES
2 ///   SENSING THE 2 DISPLACEMENTS AND THE ROTATIONAL SPEED
3 ///   ACTUATING IN 4 MAGNETS OF THE HUB
4
5 #include <stdio.h>
6 #include <time.h>
7 #include <stdlib.h>
8 #include <pthread.h>

```

## B. Software code

---

```
9 #include <unistd.h>
10 #include <string.h>
11 #include <sys/socket.h>
12 #include <netinet/in.h>
13 #include <sys/time.h>
14 #include <arpa/inet.h>
15 #include <wiringPiI2C.h>
16 #include <wiringPi.h>
17 #include <pigpio.h>
18 #define PORT 8080
19 #define DevAddr 0x48
20 #define B_convert 0x8000
21 #define B_null 0x0000
22 #define Mux_com_1_0 0x0000
23 #define Mux_com_3_0 0x1000
24 #define Mux_com_3_1 0x2000
25 #define Mux_com_3_2 0x3000
26 #define Mux_com_0 0x4000
27 #define Mux_com_1 0x5000
28 #define Mux_com_2 0x6000
29 #define Mux_com_3 0x7000
30 #define Amp_3_2 0x0000
31 #define Amp_1 0x0200
32 #define Amp_1_2 0x0400
33 #define Amp_1_4 0x0600
34 #define Amp_1_8 0x0800
35 #define Amp_1_16 0x0a00
36 #define B_cont 0x0000
37 #define B_sing 0x0100
38 #define DR_128 0x0000
39 #define DR_250 0x0020
40 #define DR_490 0x0040
41 #define DR_920 0x0060
42 #define DR_1600 0x0080
43 #define DR_2400 0x00a0
44 #define DR_3300 0x00c0
45 #define COMP_MODE_T 0x0000
46 #define COMP_MODE_W 0x0001
47 #define COM_POL_L 0x0000
48 #define COM_POL_H 0x0008
49 #define COMP_LN 0x0000
50 #define COMP_LA 0x0004
51 #define COMP_QUE_1 0x0000
52 #define COMP_QUE_2 0x0001
53 #define COMP_QUE_3 0x0002
54 #define COMP_QUE_N 0x0003
55 //////////////////////////////////////////////////GLOBAL VARIABLES
56 int DATA_SEND[4]={0,0,0,0};
57 int flag=1;
58
59
60
61
62
63
64 //////////////////////////////////////////////////FUNCTIONS
65
```



```

66 int Byte_swapper(int data16)
67 {
68     int data,data1,data2;
69     data1=(data16<<8)&0xff00;
70     data2=data16>>8;
71     data=data1 | data2;
72     return data;
73 }
74 int Config_Channel(int Config_16num,int Channel)
75 {
76     switch(Channel)
77     {
78     case 1:
79         Config_16num=Config_16num|B_convert|Mux_com_0;
80         break;
81     case 2:
82         Config_16num=Config_16num|B_convert|Mux_com_1;
83         break;
84     case 3:
85         Config_16num=Config_16num|B_convert|Mux_com_2;
86         break;
87     case 4:
88         Config_16num=Config_16num|B_convert|Mux_com_3;
89         break;
90     default :
91         printf("%s\n%s\n","Error: Not a valid Channel. ","Choose between channel 1 and 4");
92     }
93
94     return Config_16num;
95 }
96 int Config_DR(int Config_16num,int DR)
97 {
98     switch(DR)
99     {
100    case 128:
101        Config_16num=Config_16num|DR_128;
102        break;
103    case 250:
104        Config_16num=Config_16num|DR_250;
105        break;
106    case 490:
107        Config_16num=Config_16num|DR_490;
108        break;
109    case 920:
110        Config_16num=Config_16num|DR_920;
111        break;
112    case 1600:
113        Config_16num=Config_16num|DR_1600;
114        break;
115    case 2400:
116        Config_16num=Config_16num|DR_2400;
117        break;
118    case 3300:
119        Config_16num=Config_16num|DR_3300;
120        break;
121    default :
122        printf("%s\n%s\n","Error: Not a valid Data Rate. ","Choose between possible Data Rates: 128,

```

## B. Software code

---

```
    250, 490, 920, 1600, 2400, 3300");
123 }
124 return Config_16num;
125 }
126 int Config_Gain(int Config_16num,int gain)
127 {
128     switch(gain)
129     {
130     case 1:
131         Config_16num=Config_16num|Amp_1;
132         break;
133     case 2:
134         Config_16num=Config_16num|Amp_1_2;
135         break;
136     case 4:
137         Config_16num=Config_16num|Amp_1_4;
138         break;
139     case 8:
140         Config_16num=Config_16num|Amp_1_8;
141         break;
142     case 16:
143         Config_16num=Config_16num|Amp_1_16;
144         break;
145     default :
146         printf("%s\n%s\n","Error: Not a valid Gain. ","Choose between possible gains: 1, 2, 4, 8, 16");
147     }
148     return Config_16num;
149 }
150 int Config_Mode(int Config_16num, int mode)
151 {
152     switch (mode)
153     {
154     case(0):
155         Config_16num=Config_16num|B_sing;
156         break;
157     case(1):
158         Config_16num=Config_16num|B_cont;
159         break;
160     default :
161         printf("%s\n%s\n","Error: Not a valid Mode. ","Choose between Continuous (1) or Single-Shot
162         (0)");
163     }
164     return Config_16num;
165 }
166 int Config_Default(int Config_16num)
167 {
168     Config_16num=Config_16num | COMP_MODE_T;
169     Config_16num=Config_16num | COM_POL_L;
170     Config_16num=Config_16num | COMP_LN;
171     Config_16num=Config_16num | COMP_QUEUE_N;
172     return Config_16num;
173 }
174 //////////////////////////////////////////////////// MAIN
175 int main() {
176     int data,Init ,fd,reg,Config1,Config2,Config3,Config4;int CONFIG_V[4]={0,0,0,0};
177     struct timeval t1,t2,t0,tf;
178     unsigned magnet1r,magnet1l,magnet2r,magnet2l;
```

```

178 magnet1r=14;magnet1l=15;magnet2r=17;magnet2l=27;
179 struct sockaddr_in address, serv_addr;
180 int i,k,j,n_samples,n_data,SIZE_DATA,ml,server_fd, new_socket, valread,sock,addrlen,
    nano_to_micro;
181 i=k=0; j=1; SIZE_DATA=3000;n_data=3;sock=0;nano_to_micro=1e3;addrlen=sizeof(address);
182 int data_send[3]={0,0,0};int data_rec[4]={0,0,0,0};int close_vector[4]={-1,-1,-1,-1};
183 double mean_f,cpu_time_used,t_s,id;
184 mean_f=t_s=cpu_time_used=id=0.0;
185 long long t_delay=1e2*nano_to_micro;
186 ////////////////SOCKET CONFIGURATION FOR WIRELESS CONNECTION
187 //////////////// CLIENT
188 sock = socket(AF_INET, SOCK_STREAM, 0);
189 memset(&serv_addr, '0', sizeof(serv_addr));
190 serv_addr.sin_family = AF_INET;
191 serv_addr.sin_port = htons(PORT);
192 inet_pton(AF_INET, "10.16.184.180", &serv_addr.sin_addr);
193 // PROGRAM STARTS HERE
194 //////////////// TRY TO CONNECT TO THE LINUX CPU OR SERVER
195 connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));
196
197 //////////////// INITIALISE THE I2C PROTOCOL LIBRARY
198 fd= wiringPiI2CSetup (DevAddr);
199 //////////////// INITIALISE THE PWM BY DMA SOFTWARE
200 if ( gpioInitialise () < 0)
201 {
202     printf("pigpio initialisation failed.\n");
203 }
204 //////////////// SET THE PWM PARAMETERS
205 gpioSetPWMrange(magnet1r,2000);
206 gpioSetPWMrange(magnet1l,2000);
207 gpioSetPWMrange(magnet2r,2000);
208 gpioSetPWMrange(magnet2l,2000);
209
210 gpioSetPWMrange(magnet1r,8000);
211 gpioSetPWMrange(magnet1l,8000);
212 gpioSetPWMrange(magnet2r,8000);
213 gpioSetPWMrange(magnet2l,8000);
214
215 //////////////// INITIALISE THE PWM TO 0 VOLTAGE
216 gpioPWM(magnet1r, 0);
217 gpioPWM(magnet1l, 0);
218 gpioPWM(magnet2r, 0);
219 gpioPWM(magnet2l, 0);
220
221 //////////////// PERSONALIZED CODE FOR THE AD CONVERTER
222 ////////////////WRITNG REGISTER CONFIGURATION
223 Config1=0;
224 Config1=Config_Default(Config1);
225 Config1=Config_Channel(Config1,1);//Channel 1
226 Config1=Config_DR(Config1,3300);
227 Config1=Config_Gain(Config1,1);
228 Config1=Config_Mode(Config1,0);
229 Config1=Byte_swapper(Config1);
230 CONFIG_V[0]=Config1;
231
232 Config2=0;
233 Config2=Config_Default(Config2);

```

## B. Software code

---

```
234 Config2=Config_Channel(Config2,2);//Channel 2
235 Config2=Config_DR(Config2,3300);
236 Config2=Config_Gain(Config2,1);
237 Config2=Config_Mode(Config2,0);
238 Config2=Byte_swapper(Config2);
239 CONFIG_V[1]=Config2;
240
241 Config3=0;
242 Config3=Config_Default(Config3);
243 Config3=Config_Channel(Config3,3);//Channel 3
244 Config3=Config_DR(Config3,3300);
245 Config3=Config_Gain(Config3,1);
246 Config3=Config_Mode(Config3,0);
247 Config3=Byte_swapper(Config3);
248 CONFIG_V[2]=Config3;
249
250
251 //////////////// START THE LOOP
252
253 while( flag==1)
254 {
255 //////////////// SEND THE SENSOR DATA
256 send(sock , DATA_SEND , sizeof(DATA_SEND) , 0 );
257 //////////////// WAIT FOR THE SERVER CONTROL SIGNAL
258 valread = read(sock , data_rec, 40);
259 //////////////// EXIT THE PROGRAM IF THE SERVER ASKS FOR IT
260 if (data_rec[0]==close_vector[0])
261 { flag=0;
262 printf ("%s\n", "EXIT");
263 }
264 else {
265 //////////////// IMPLEMENT THE CONTROL SIGNAL INTO THE PWM
266 gpioPWM(magnet1r, data_rec[0]);
267 gpioPWM(magnet1l, data_rec[1]);
268 gpioPWM(magnet2r, data_rec[2]);
269 gpioPWM(magnet2l, data_rec[3]);}
270 //////////////// READ THE AD CONVERTER DATA
271 for (int j = 0; j < 3; j++)
272 {
273 wiringPiI2CWriteReg16(fd,0x01,CONFIG_V[j]);
274 nanosleep((const struct timespec []){{0, t_delay}}, NULL);
275 data= wiringPiI2CReadReg16(fd,0x00) ;
276 data=Byte_swapper(data);
277 DATA_SEND[j]=data>>4;
278 }
279 }
280 //////////////// CLOSE ALL PROGRAM
281 printf ("%s\n", "exit while" );
282 gpioPWM(magnet1r, 0);
283 gpioPWM(magnet1l, 0);
284 gpioPWM(magnet2r, 0);
285 gpioPWM(magnet2l, 0);
286
287 return(0);
288 }
```

---

---

```

1 import socket
2 import threading
3 import time
4 import paramiko
5 import sys
6 import json
7 global c_flag
8 import time
9 import numpy as np
10 import csv
11
12 c_flag=1
13 s_flag=1
14 cl_vector_s=[-1]*4
15 cl_vector_r=[-1]*8
16 #server bind socket
17 serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18 serversocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
19 host = socket.gethostbyname(socket.gethostname())
20 port = 9999
21 serversocket.bind((host, port))
22
23 ##### SSH pi in the rotor
24 # PARAMETERS
25 SSH_ADDRESS = "thesis"
26 SSH_USERNAME = "pi"
27 SSH_PASSWORD = "thesis"
28 SSH_COMMAND = "python ~/raspberrypi_rotor/rotor4.py "+socket.gethostbyname(socket.gethostname())
29 #SSH creation
30 ssh = paramiko.SSHClient()
31 ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
32 ssh_stdin = ssh_stdout = ssh_stderr = None
33 # Order
34 try:
35     ssh.connect(SSH_ADDRESS, username=SSH_USERNAME, password=SSH_PASSWORD)
36     ssh_stdin, ssh_stdout, ssh_stderr = ssh.exec_command(SSH_COMMAND)
37 except Exception as e:
38     sys.stderr.write("SSH connection error: {0}".format(e))
39     print('ssh error')
40 ##### SSH pi in the estator
41 # PARAMETERS
42 SSH_ADDRESS2 = "thesis2"
43 SSH_USERNAME2 = "pi"
44 SSH_PASSWORD2 = "thesis2"
45 SSH_COMMAND2 = "python ~/raspberrypi_stator/stator4.py "+socket.gethostbyname(socket.gethostname())
46 # SSH creation
47 ssh2 = paramiko.SSHClient()
48 ssh2.set_missing_host_key_policy(paramiko.AutoAddPolicy())
49 ssh_stdin = ssh_stdout = ssh_stderr = None
50 #Order
51 try:
52     ssh2.connect(SSH_ADDRESS2, username=SSH_USERNAME2, password=SSH_PASSWORD2)
53     ssh_stdin, ssh_stdout, ssh_stderr = ssh2.exec_command(SSH_COMMAND2)
54 except Exception as e:

```

## B. Software code

---

```
55     sys.stderr.write("SSH connection error: {0}".format(e))
56     print('ssh error')
57
58
59 # queue up to the 2 requests
60 serversocket.listen(2)
61
62
63
64 def threaded_server():
65     i=0
66     global data_arr
67     while c_flag:
68         data_actuation_s=[0]*4
69         data_actuation_r=[0]*8
70         states=[0]*7
71         states_nc=[0]*7
72         state_vector=[]
73         time_series=[]
74         tnext=0
75         # establish a connection
76         r_socket,addr_r = serversocket.accept()
77         print("Got a connection from %s the rotor" % str(addr_r))
78         s_socket,addr_s = serversocket.accept()
79         print("Got a connection from %s the stator" % str(addr_s))
80         # receive calibrated data
81         data_sensor_rj = r_socket.recv(4096)
82         data_sensor_sj = s_socket.recv(4096)
83         load_sensor_r = json.loads(data_sensor_rj.decode())
84         load_sensor_s = json.loads(data_sensor_sj.decode())
85         cal_r=load_sensor_r.get("adc_values_r_cal")
86         cal_s=load_sensor_s.get("adc_values_s_cal")
87         data_send_r = json.dumps({"act_values_r": data_actuation_r})
88         data_send_s = json.dumps({"act_values_s": data_actuation_s})
89         r_socket.send(data_send_r.encode())
90         s_socket.send(data_send_s.encode())
91         j=0
92         i=0
93         while c_flag:
94             i=i+1
95             t1=time.time()
96             ### SENSING
97             #tcom=time.time()
98             data_sensor_sj = s_socket.recv(1024)
99             #tcom1=time.time()-tcom
100            data_sensor_rj = r_socket.recv(1024)
101            #tcom2=time.time()-tcom1-tcom
102            load_sensor_r = json.loads(data_sensor_rj.decode())
103            load_sensor_s = json.loads(data_sensor_sj.decode())
104            data_sensor_r=load_sensor_r.get("adc_values_r")
105            data_sensor_s=load_sensor_s.get("adc_values_s")
106            for k in range(7):
107                if k < 4:
108                    states_nc[k]=data_sensor_r[k]
109                    states[k]=states_nc[k]-cal_r[k]
110                else:
111                    states_nc[k]=data_sensor_s[k-4]
```

```

112         states[k]=states_nc[k]-cal_s[k-4]
113         states[k] = states[k]*4.096/2048
114         states[k] = states[k]*1.5/2
115     state_vector.extend(states)
116     #print(states)
117     ##### ACTUATING
118     ### Control computation
119     for p in range(4):
120         data_actuation_s[p] = 50*float(np.random.rand(1,1,1)) #4 external magnets
121         ##             |-pair 1-| |-pair 2-| |--pair 3-|
122     for p in range(8):
123         data_actuation_r[p] = 50*float(np.random.rand(1,1,1))
124     ##### end of control computation
125     ### data send has to be a value from 0 to 100
126     data_send_r = json.dumps({"act_values_r": data_actuation_r})
127     data_send_s = json.dumps({"act_values_s": data_actuation_s})
128     r_socket.send(data_send_r.encode())
129     s_socket.send(data_send_s.encode())
130     t2=time.time()-t1
131     tnext=tnext+t2
132     time_series.append(tnext)
133
134
135     close_msg_r = json.dumps({"act_values_r": cl_vector_r})
136     r_socket.send(close_msg_r.encode())
137     close_msg_s = json.dumps({"act_values_s": cl_vector_s})
138     s_socket.send(close_msg_s.encode())
139     time.sleep(0.2)
140     print("cerrando socket")
141     r_socket.close()
142     s_socket.close()
143     print("socket cerrado")
144     state_data=[0]*7
145     length=len(state_vector)
146     n_samples=int(length/7)
147     state_data= [[0 for x in range(8)] for y in range(n_samples)]
148     c=0
149     for v in range(n_samples):
150         for w in range(8):
151             if w==0:
152                 state_data[v][w]=time_series[v]
153             else:
154                 state_data[v][w]=state_vector[c]
155             c=c+1
156
157     with open('data_vector.csv', 'w') as f:
158         writer = csv.writer(f)
159         writer.writerows(state_data)
160
161     print(state_data)
162     print(tnext/n_samples)
163     # Put the server in a thread
164     threads=[]
165     for i in range(1):
166         t = threading.Thread(name=str(i),target=threaded_server)
167         t.daemon = True
168         threads.append(t)

```

## B. Software code

---

```
169     t.start()
170 time.sleep(5)
171 while s_flag:
172     command=input("Introduce your command")
173     if command=="c":
174         c_flag=0
175     if command=="s":
176         c_flag=0
177         time.sleep(1)
178         s_flag=0
179
180
181
182
183 print('closing threads')
184 for c in threads:
185     print('ENDING threads')
186     c.join()
187 print('END')
```

---

```
1 import socket
2 import time
3 import sys
4 import threading
5 import json
6 import numpy as np
7 import Adafruit_ADS1x15
8 import RPi.GPIO as GPIO
9 i=0
10 global c_flag
11 c_flag=True
12 GPIO.setmode(GPIO.BCM)
13 GPIO.setwarnings(False)
14
15 #GPIO ports to control the magnets, set them as output information
16 GPIO.setup(14,GPIO.OUT)
17 GPIO.setup(15,GPIO.OUT)
18 GPIO.setup(17,GPIO.OUT)
19 GPIO.setup(27,GPIO.OUT)
20 GPIO.setup(23,GPIO.OUT)
21 GPIO.setup(24,GPIO.OUT)
22 GPIO.setup(10,GPIO.OUT)
23 GPIO.setup(9,GPIO.OUT)
24
25 #Establish the magnets right and left for both directions
26 magnet1r =GPIO.PWM(14,10000)
27 magnet1r.start(0)
28 magnet1l =GPIO.PWM(15,10000)
29 magnet1l.start(0)
30 magnet2r =GPIO.PWM(17,10000)
31 magnet2r.start(0)
32 magnet2l =GPIO.PWM(27,10000)
33 magnet2l.start(0)
34 magnet3r =GPIO.PWM(23,10000)
35 magnet3r.start(0)
36 magnet3l =GPIO.PWM(24,10000)
```



```

37 magnet3l.start(0)
38 magnet4r =GPIO.PWM(10,10000)
39 magnet4r.start(0)
40 magnet4l =GPIO.PWM(9,10000)
41 magnet4l.start(0)
42
43
44
45 # create a socket object
46 client_rotor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
47 # create an adc object
48 adc = Adafruit_ADS1x15.ADS1015()
49 GAIN = 1
50 d_t=3300
51
52 # get local machine name
53 host = sys.argv[1]
54 port = 9999
55 time.sleep(1)
56 # connection to hostname on the port.
57 client_rotor.connect((host, port))
58
59 # Receive no more than 1024 bytes..
60
61
62 cl_vector_r=[-1]*8
63 values = [0]*4
64 data_actuators_r=[0]*8
65 for i in range(4):
66     values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
67 data = json.dumps({"adc_values_r_cal": values})
68 client_rotor.send(data.encode())
69 data_act_r = client_rotor.recv(1024)
70 actuation_r = json.loads(data_act_r.decode())
71 data_actuators_r=actuation_r.get("act_values_r")
72
73 while c_flag:
74     ### measure and send
75     if data_actuators_r != cl_vector_r:
76         for i in range(4):
77             values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
78             data = json.dumps({"adc_values_r": values})
79             client_rotor.send(data.encode())
80     else :
81         c_flag=False
82         for i in range(4):
83             values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
84             data = json.dumps({"adc_values_r": values})
85             client_rotor.send(data.encode())
86         break
87
88     ### Receive and actuate
89     data_act_r = client_rotor.recv(1024)
90     actuation_r = json.loads(data_act_r.decode())
91     data_actuators_r=actuation_r.get("act_values_r")
92     ##### actuate
93

```

## B. Software code

---

```
94 magnet1r.ChangeDutyCycle(data_actuators_r[0])
95 magnet1l.ChangeDutyCycle(data_actuators_r[1])
96 magnet2r.ChangeDutyCycle(data_actuators_r[2])
97 magnet2l.ChangeDutyCycle(data_actuators_r[3])
98 magnet3r.ChangeDutyCycle(data_actuators_r[4])
99 magnet3l.ChangeDutyCycle(data_actuators_r[5])
100 magnet4r.ChangeDutyCycle(data_actuators_r[6])
101 magnet4l.ChangeDutyCycle(data_actuators_r[7])
102
103
104 print('closing threads')
105 for c in threads:
106     c.join()
```

---

```
1 import socket
2 import time
3 import sys
4 import threading
5 import json
6 import numpy as np
7 import Adafruit_ADS1x15
8 import RPi.GPIO as GPIO
9 i=0
10 global c_flag
11 c_flag=True
12 GPIO.setmode(GPIO.BCM)
13 GPIO.setwarnings(False)
14
15 #GPIO ports to control the magnets, set them as output information
16 GPIO.setup(14,GPIO.OUT)
17 GPIO.setup(15,GPIO.OUT)
18 GPIO.setup(17,GPIO.OUT)
19 GPIO.setup(27,GPIO.OUT)
20
21 #Establish the magnets right and left for both directions
22 magnet1r =GPIO.PWM(14,10000)
23 magnet1r.start(0)
24 magnet1l =GPIO.PWM(15,10000)
25 magnet1l.start(0)
26 magnet2r =GPIO.PWM(17,10000)
27 magnet2r.start(0)
28 magnet2l =GPIO.PWM(27,10000)
29 magnet2l.start(0)
30
31
32 # create a socket object
33 client_stator = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
34 # create an adc object
35 adc = Adafruit_ADS1x15.ADS1015()
36 GAIN = 1
37 d_t=3300
38 # get local machine name
39 host = sys.argv[1]
40 port = 9999
41 time.sleep(1.5)
42 # connection to hostname on the port.
```

---

```
43 client_stator.connect((host, port))
44
45
46 cl_vector_s=[-1]*4
47 values = [0]*3
48 data_actuators_s=[0]*4
49 for i in range(3):
50     values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
51 data = json.dumps({"adc_values_s_cal": values})
52 client_stator.send(data.encode())
53 data_act_s = client_stator.recv(1024)
54 actuation_s = json.loads(data_act_s.decode())
55 data_actuators_s=actuation_s.get("act_values_s")
56 while c_flag:
57     ### measure and send
58     if data_actuators_s != cl_vector_s:
59         for i in range(3):
60             values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
61             data = json.dumps({"adc_values_s": values})
62             client_stator.send(data.encode())
63     else :
64         c_flag=False
65         for i in range(3):
66             values[i] =adc.read_adc(i, gain=GAIN,data_rate=d_t)
67             data = json.dumps({"adc_values_s": values})
68             client_stator.send(data.encode())
69         break
70
71     ### Receive and actuate
72 data_act_s = client_stator.recv(1024)
73 actuation_s= json.loads(data_act_s.decode())
74 data_actuators_s=actuation_s.get("act_values_s")
75
76 magnet1r.ChangeDutyCycle(data_actuators_s[0])
77 magnet1l.ChangeDutyCycle(data_actuators_s[1])
78 magnet2r.ChangeDutyCycle(data_actuators_s[2])
79 magnet2l.ChangeDutyCycle(data_actuators_s[3])
80
81 client_stator.close()
```

---

**Department of Electrical Engineering**  
Centre for Electric Power and Energy (CEE)  
Technical University of Denmark  
Elektrovej, Building 325  
DK-2800 Kgs. Lyngby  
Denmark

[www.elektro.dtu.dk/cee](http://www.elektro.dtu.dk/cee)

Tel: (+45) 45 25 35 00

Fax: (+45) 45 88 61 11

E-mail: [cee@elektro.dtu.dk](mailto:cee@elektro.dtu.dk)